

Deep Learning and Quantum Generative Models for High Energy Physics Calorimeter Simulations

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Florian Rehm

M.Eng.

aus Forchheim, Deutschland

Berichter: Universitätsprofessor Dr. Kerstin Borras
Universitätsprofessor Dr. Oliver Pooth

Tag der mündlichen Prüfung: 25 August 2023

Diese Dissertation ist auf den Internetseiten der
Universitätsbibliothek online verfügbar.



I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.



Geneva, December 2022
Florian Rehm

Abstract

The aim of this thesis is to explore Deep Learning and Quantum Computing approaches to reduce the challenging computing resource requirements of future High Energy Physics detector simulations while maintaining the necessary accuracy. In the forthcoming High Luminosity phase at the Large Hadron Collider at CERN, the number of collisions is increasing drastically. The demand for detector simulations to analyze the collision products is growing by the same scale. This causes an acute shortage in computing resources in both hardware and efficient software. Therefore, resource-optimizing simulation alternatives are needed, and novel technologies to be explored.

In this thesis, Deep Learning Generative Adversarial Networks (GANs) are investigated as a first option to be deployed for a future representative high-granularity calorimeter. Novel neural network architectures are developed and reduced precision computing techniques applied to optimize the GAN models. The results are realistic and accurate models that are about five orders of magnitude faster than the current Monte Carlo simulation.

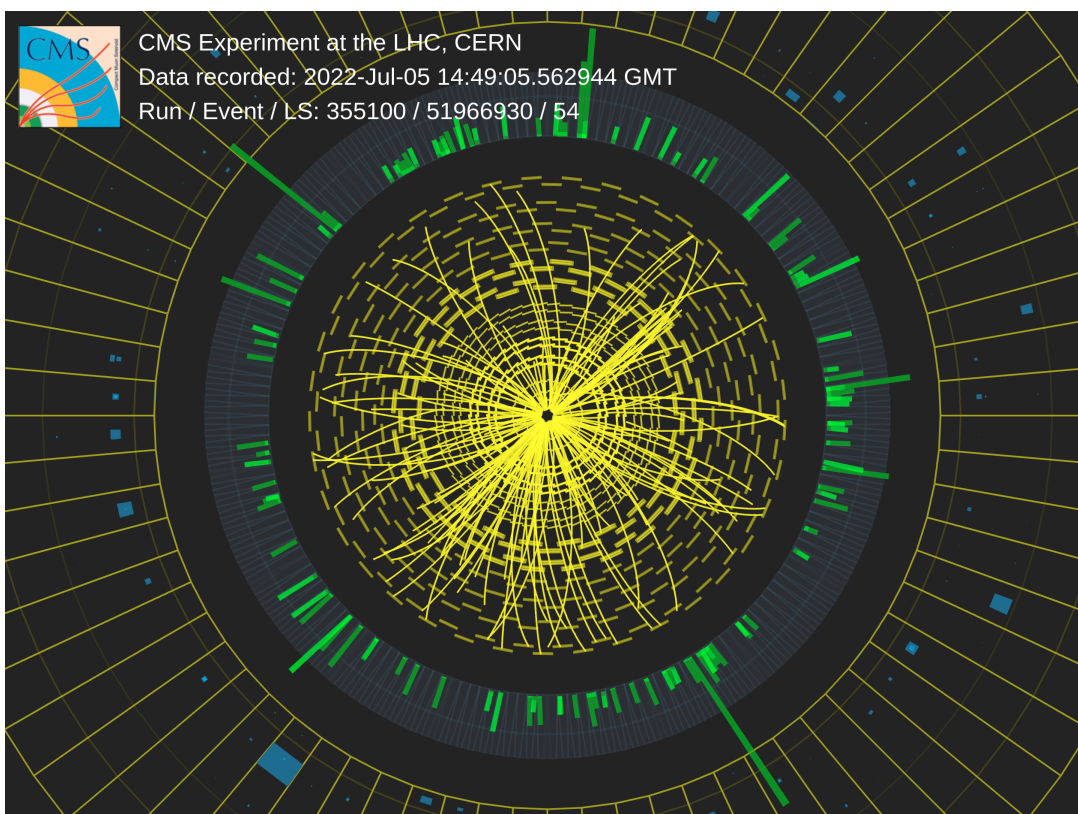
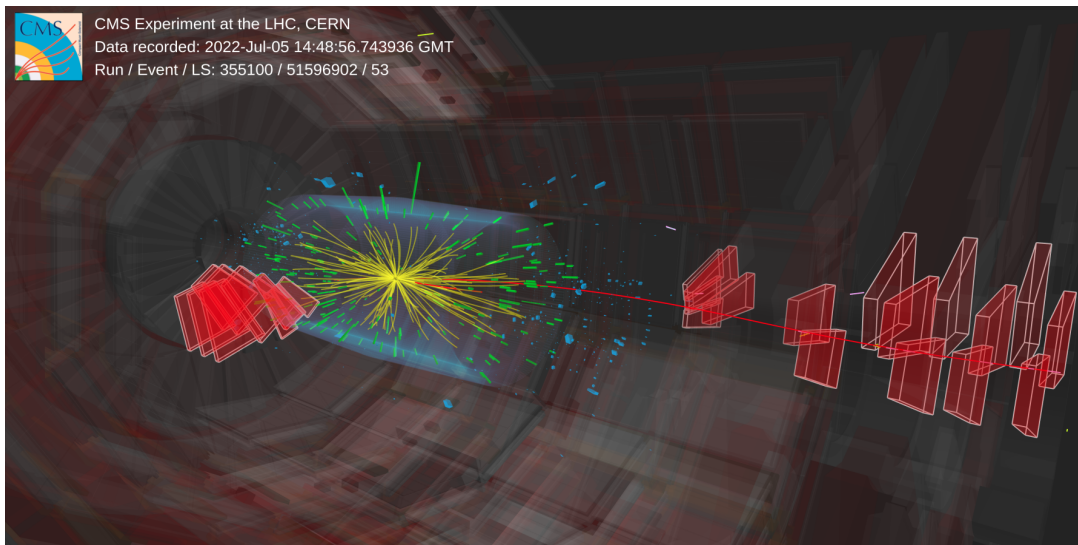
The second study of this thesis addresses the novel and rapidly evolving field of Quantum Computing for a simplified calorimeter. The initial attempt employs a model available in the IBM software suite that has been successfully adapted and modified. It runs in a hybrid GAN mode employing a quantum generator and a conventional discriminator. However, this model learned only average distributions and suffers from fluctuations. Therefore, a new fully quantum model was designed to produce individual images. The in this thesis developed Quantum Angle Generator achieves extremely precise results. Due to its noise robustness and lightweight training, it is successfully executed on current noisy quantum devices with a limited number of qubits. The superior expressive power of quantum circuits compared to classical neural networks, combined with a potential exponential scaling, makes Quantum Computing a promising future computing approach. This thesis has pioneered and conducted studies which confirm that this potential has a realistic chance to materialize.

Zusammenfassung

Das Ziel dieser Dissertation ist die Erforschung von Deep Learning und Quantum Computing Ansätzen, um die Hardwareanforderungen von zukünftigen Hochenergiephysik-Detektorsimulationen zu reduzieren. In der bevorstehenden High Luminosity Phase des Large Hadron Colliders am CERN wird die Anzahl an Partikelkollisionen drastisch ansteigen. Im gleichen Maße wächst die Nachfrage nach Detektorsimulationen zur Analyse der entstehenden Kollisionsprodukte. Dies führt sowohl zu einem akuten Mangel an Rechenressourcen in Hardware als auch zu einem Mangel an effizienter Software. Daher müssen ressourcenschonende Simulationsalternativen entwickelt und neue Technologien erforscht werden.

Der erste in dieser Arbeit untersuchte Ansatz sind Deep Learning Generative Adversarial Networks (GANs). Diese werden für ein zukünftiges repräsentatives Kalorimeter mit hoher Granularität entwickelt. Zur Optimierung der GAN-Modelle werden neuartige neuronale Netzwerksarchitekturen entwickelt und Zahlenformate mit niedrigeren numerischen Präzessionen angewandt. Das Ergebnis, realistische Modelle mit hoher Genauigkeit, welche mehr als fünf Größenordnungen schneller sind als die aktuell verwendete Monte-Carlo-Simulation.

Der zweite Ansatz dieser Dissertation befasst sich mit dem neuartigen und sich schnell entwickelnden Gebiet des Quantum Computings für ein vereinfachtes Kalorimeter. Der erste Versuch verwendet ein von IBM zur Verfügung gestelltes Modell, welches erfolgreich angepasst und modifiziert wurde. Es handelt sich um ein hybrides Modell mit einem Quantengenerator und einem konventionellen Diskriminator. Dieses Modell erzeugt allerdings nur gemittelte Verteilungen und leidet unter zufälligen Schwankungen. Daher wurde ein neues Vollquantenmodell entwickelt, das Einzelbilder erzeugt. Das in dieser Arbeit entwickelte Modell des Quantum Angle Generators erlaubt das Erzielen von äußerst präzisen Ergebnissen. Aufgrund seiner Robustheit gegen Fehler und seines leichten Trainings kann es erfolgreich auf aktuellen fehleranfälligen Quantencomputern mit nur einer begrenzten Anzahl an Qubits ausgeführt werden. Die überlegene Repräsentationsfähigkeit von Quantenschaltkreisen im Vergleich zu klassischen neuronalen Netzen, kombiniert mit einer möglichen exponentiellen Skalierung, macht Quantum Computing zu einem vielversprechenden zukünftigen Computing-Ansatz. Diese Arbeit hat wegweisende Studien durchgeführt, die bestätigen, dass die Ansätze in dieser Dissertation eine realistische Chance haben, sich zu verwirklichen.



One of the first reconstructed proton-proton collisions inside the Compact Muon Solenoid (CMS) detector at the Large Hadron Collider (LHC) restart for Run 3 on the 5th of July 2022 at CERN. On that day, a new world record for stable particle collisions at the highest energy of 13.6 TeV was set. The author of this thesis was contributing to the CMS experiment as a Detector Control Shifter and a Data Quality Monitoring Shifter, and was awarded the CMS Shifter of the Year 2022.

Acknowledgements

I would like to share my deepest gratitude to my CERN supervisor, Sofia Vallecorsa, from whom I learned a lot during my doctoral time and for the opportunity to research on such an interesting topic. Through her enormous expertise, she guided me through this project and enabled me to participate in valuable external collaborations.

This endeavor would not have been possible without my university supervisor, Kerstin Borrás. She professionally and personally promoted me and took care for a smooth PhD procedure at my university and established me to be part of a great research collaboration at DESY.

I am delighted by the involvement of Dirk Krücker and Michele Grossi into the supervision. Both are proven experts in their fields and supported my work with fruitful discussions, answering technical questions and providing constructive feedback.

I would like to extend my sincere thanks to my CERN section head, Alberto Di Meglio, who set the foundation for a constructive and promotional work environment. Despite the Covid situation, he enabled a remote atmosphere, allowing work progress like usual and social and amusing interactions.

Thanks to the collaborators Simon Schnake, Alexis-Harilaos Verney-Provatas and Valle Varo from DESY, it was a pleasure to work and exchange ideas with you.

Furthermore, I would like to thank Sofia Vallecorsa, Dirk Krücker and Valle Varo for correction reading my thesis.

In addition, I would like to thank all colleagues, who have made my time at CERN as one of the best experiences in my life, and everyone who made this PhD thesis possible.

It was a great experience working together with Intel from the US. I gained precious work insights, and we accomplished remarkable results and papers.

Der größte Dank geht allerdings an meine Familie, welche immer hinter mir stand, mich unterstützt hat und mir den Mut zugesprochen hat dieses Abenteuer am CERN in der Schweiz zu starten.

I would like to thank all my friends who supported and encouraged me.

This work has been sponsored by the Wolfgang Gentner Programme of the German Federal Ministry of Education and Research and by the CERN Quantum Technology Initiative. Access to the IBM Quantum Services was obtained through the IBM Quantum Hub at CERN. The views expressed are those of the author and do not reflect the official policy or position of IBM or the IBM Q team.

Contents

Abstract	i
Zusammenfassung	ii
Acknowledgements	v
1 Introduction	1
1.1 Thesis Objectives	4
1.2 Structure of the Content	5
2 High Energy Physics Experiments	7
2.1 Accelerator Physics at the Large Hadron Collider	11
2.2 The Principles of Particle Physics Detectors	15
2.3 The Importance of Computing in High Energy Physics	24
2.3.1 Worldwide Large Hadron Collider Computing Grid	26
2.3.2 Future High Energy Physics Computing Constraints	27
2.3.3 Current Monte Carlo Simulation Methods	31
2.3.4 Alternative Fast Simulation Approaches	33
2.4 3D Calorimeter Shower Images	34
3 Deep Learning for Future High Energy Physics Simulations	39
3.1 Introduction to Deep Learning	39
3.2 Generative Models: An Overview	43
3.3 Convolutional Layers: 2D and 3D Operations	48
3.3.1 Convolutional 2D Layers	48
3.3.2 Convolutional 3D Layers	52
3.4 Supplementary Neural Network Layers	53
4 3D Generative Adversarial Network Models	55
4.1 Generative Adversarial Networks in Detail	56
4.2 Generative Adversarial Network Training	58
4.2.1 Common Occurring Training Failure Modes	58
4.2.2 Techniques for Improved Training	60
4.3 Variations of Generative Adversarial Networks	61

4.4	3D Convolutional Architectures	62
4.4.1	Baseline 3DGAN Model	62
4.4.2	The Conv3D Generator Architecture	64
4.4.3	The Conv3D Discriminator Architecture	65
4.5	Wasserstein Generative Adversarial Networks	68
5	Transition from Convolutional 3D to faster 2D Networks	69
5.1	Motivation for Alternative 2D Architectures	70
5.2	Related Work: 2D Networks for 3D Images	71
5.3	Novel 2D Convolutional Neural Network Architectures	71
5.3.1	Naive Approach: Channel Dimension as 3rd Image Dimension	71
5.3.2	Generator 1: The Three Branches Architecture	72
5.3.3	Generator 2: The 25 Branches Architecture	74
5.3.4	Generator 3: The Single Branch Architecture	75
5.4	Training Evaluation	77
5.4.1	Dying ReLU Problem in Training	79
5.4.2	The Training Loss Evolution	80
5.5	Inference Evaluation	83
5.5.1	Computational Evaluation	84
5.5.2	Physics Accuracy Evaluation	87
5.6	Discussion and Future Work of the New Convolutional 2D Model	98
6	Reduced Precision Strategies	101
6.1	The Principle of Quantization to a Reduced Precision	102
6.2	Related Work in Quantization	103
6.3	Quantization Tools	104
6.3.1	Intel Neural Compressor	105
6.3.2	TensorFlow Lite	107
6.4	Evaluation of the Quantized Models	109
6.4.1	Computational Evaluation	110
6.4.2	Physics Accuracy Evaluation	114
6.5	Discussion and Future Work of the Reduced Precision Research	122
7	Quantum Computing	125
7.1	Qubits: The Computing Units of Quantum Computers	126
7.2	One Qubit Quantum Gates	129
7.2.1	X-Gates	130
7.2.2	Hadamard Gates	130
7.2.3	Y-Rotational Gates	131
7.3	Multiple Qubit Circuits	132
7.3.1	Two Qubit Gates	134
7.3.2	Quantum Circuits	135
7.4	Noise Intermediate Scale Quantum Devices	136
7.5	Quantum Machine Learning	142

7.6	Quantum Generative Models	143
8	Hybrid Quantum - Classical Generative Adversarial Network Model	145
8.1	Principles of the Hybrid Model	146
8.2	Quantum Circuit Parameter Update	147
8.3	The Initial 1-Dimensional Model	148
8.3.1	Quantum Generator Circuit Architectures	148
8.3.2	1D Training Data Preparation	151
8.3.3	Training and Inference without Noise	152
8.3.4	Inference with Noise	154
8.3.5	Training with Noise	158
8.3.6	Training on Quantum Hardware	162
8.3.7	Hybrid Quantum-Classical Training Approach	165
8.4	Enhanced 2-Dimensional Model	166
8.4.1	2D Model Circuit Architecture	166
8.4.2	Training and Inference without Noise	169
8.4.3	Inference with Noise	170
8.5	Discussion of the Hybrid Quantum - Classical Model	171
9	Quantum Angle Generator	173
9.1	Related Work to Quantum Generative Models	174
9.2	The Quantum Angle Generator in Detail	175
9.2.1	Model Description	176
9.2.2	Objective Training Function	180
9.2.3	Quantum Circuit Study	183
9.3	In-depth Accuracy Analysis	191
9.3.1	Training Process Evaluation	192
9.3.2	Inference Evaluation	194
9.4	The Effect of Quantum Noise	197
9.4.1	Noise Effect on the Inference Process	198
9.4.2	Noise Effect on the Training Process	201
9.5	Number of Shots Dependency	205
9.6	Discussion and Future Work of the Quantum Angle Generator	207
10	Full Quantum Generative Adversarial Networks	211
10.1	Motivation for Adversarial Training	211
10.2	The Generator and Discriminator Quantum Circuits	212
10.3	The Adversarial Training Optimization	216
10.4	Training Evaluation	218
10.5	Inference Evaluation	219
10.6	Hybrid Model Comparison	221
10.7	Discussion of the Full Quantum Generative Adversarial Training	225
11	Summary, Future Work and Conclusion	227

11.1	Summary	227
11.2	Future Work	229
11.3	Conclusion	230
11.4	List of Publications Related to this Thesis	231
11.5	List of Workshop Presentations	233
A	Appendix Deep Learning	235
A.1	Hyperparameters for training the GAN Models	235
A.2	The GAN Training Losses	235
A.3	SSIM Index: Determination of the Adjustable Parameter	237
A.4	Pixel-wise Correlation Plots	238
B	Appendix Quantum Computing	241
B.1	The CZ-Entangling Gate	241
B.2	QAG Inference Algorithm	243
B.3	QAG Circuit Architecture Search	244
B.4	QAG Circuit Study MMD Loss	245
B.5	Mathematical Definitions of the Accuracy Metrics	246
B.5.1	MSE of the Average Shower Shape	246
B.5.2	Energy Sum Histogram	246
B.6	Training Data Covariance Compared to Correlation	248
B.7	QAG Inference Noise Study Shower Shapes	248
B.8	QAG Quantum Hardware Training	251
B.9	Full qGAN Model Accuracy Metrics	253
B.10	Hybrid S and Hybrid L qGAN Model Accuracy Metrics	254
	Bibliography	255
	Abbreviations	283

List of Figures

2.1	The fundamental particles of the Standard Model.	8
2.2	The fundamental forces of the Standard Model.	9
2.3	The CERN accelerator complex, including the LHC, its pre-acceleration chain and the four large LHC experiments ATLAS, CMS, ALICE and LHCb.	12
2.4	The LHC Run plan including the High Luminosity LHC	14
2.5	The CMS detector concept and its sub-detectors.	16
2.6	Feynman diagram of pair production and Bremsstrahlung.	19
2.7	Electromagnetic calorimeter shower examples.	20
2.8	Feynman diagram of the decay of a Higgs boson into two photons and into four leptons.	21
2.9	All the sub-detector components of CMS in a cross-sectional view, including exemplary particle trajectories.	24
2.10	The Worldwide LHC Computing Grid layers.	26
2.11	The generated amount of data from the LHC compared to other big data fields.	27
2.12	Estimated CPU time and disk space for CMS in the HL-LHC phase.	29
2.13	Projected CPU requirements of CMS between 2021 and 2038.	30
2.14	Determination of π with Monte Carlo methods.	32
2.15	The uniform distribution of the primary particle energies.	34
2.16	A 3D example shower image.	35
2.17	A simplified representation of the three-dimensional image and an example shower development at $y = 13$	36
2.18	Example shower shape plots.	37
2.19	Example shower shape plots in a logarithmic scale.	37
2.20	The polynomial fit between the primary particle energy E_p and the measured energy sum within the detector E_{sum}	38
3.1	Number of Google searches of AI and ML during the last years.	40
3.2	The hierarchy of the computer science fields Artificial Intelligence, Machine Learning and Deep Learning.	41
3.3	Comparison of the achieved accuracy between Deep Learning and Machine Learning algorithms.	42
3.4	An example of a bijection in Normalizing Flows.	44

3.5	Transformation of a Gaussian distribution into a multi modal distribution by multiple invertible transformations f_i .	45
3.6	The encoding-decoding process of an Autoencoder.	46
3.7	A regular versus an irregular latent space.	46
3.8	The principle of a Variational Autoencoder.	47
3.9	The working principle of a Conv2D layer.	49
3.10	A Conv2D layer example with multiple filters.	50
3.11	2D convolutional process of an input image with three dimensions to a 2D output image.	50
3.12	2D convolution of a 3D input image into a 2D output image with multiple channels.	51
3.13	Same padding in convolutional layers to maintain the same output dimension as the input dimension.	51
3.14	A Conv3D layer applied to a 3D matrix	52
3.15	The sigmoid activation function, the standard ReLU activation function and the LeakyReLU activation function.	53
4.1	The Generative Adversarial Network principle.	56
4.2	Random samples of human faces generated by a GAN model.	57
4.3	The 3DGAN generator neural network.	63
4.4	The Conv3D generator architecture.	65
4.5	The Conv3D discriminator architecture.	66
4.6	The losses of the Wasserstein GAN training.	68
5.1	The G1 generator: three branches architecture.	73
5.2	The discriminator following the G1 architecture.	74
5.3	The G2 generator: 25 branches architecture.	75
5.4	The G3 generator: single branch architecture.	76
5.5	The total weighted training loss for the different models: Conv3D, G1, G2 and G3.	80
5.6	Unweighted True/Fake training losses.	81
5.7	Unweighted AUX training losses.	82
5.8	Unweighted Lambda training losses.	82
5.9	Optimization of GPU utilization of the Conv2D G1 model.	85
5.10	Optimization of GPU utilization of the Conv3D model.	86
5.11	Generated example 3D shower images.	90
5.12	Particle shower shape distributions.	91
5.13	Mean energy deposition validation metric.	92
5.14	Single cell energy deposition.	93
5.15	Calibrated SSIM index.	94
5.16	The number of hits validation metric.	96
5.17	The energy sum metric.	97

6.1	Representation of the float32 format, signed int8 format and unsigned int8 format.	102
6.2	Overview of the Intel Neural Compressor quantization tool.	106
6.3	The workflow of the Intel Neural Compressor quantization tool.	107
6.4	The TensorFlow Lite quantization of a network segment taken from our generator network.	108
6.5	Inference throughput in shower images per seconds of the quantized and not quantized models.	111
6.6	The percentage of the total inference run-times for individual operations.	113
6.7	Shower shapes validation metric.	115
6.8	Example 3D shower images.	118
6.9	Mean energy validation metric.	119
6.10	Single cell energy deposition.	119
6.11	The number of hits metric.	120
6.12	The energy sum metric.	121
7.1	A classical bit compared to a quantum bit.	127
7.2	A quantum state on the Bloch sphere.	128
7.3	Three pairs of possible quantum basis states.	129
7.4	The operation principle of a Hadamard gate.	131
7.5	The operation principle of a Y-rotational gate.	132
7.6	Visualization of the CX -, CY -gate and CZ -gate.	135
7.7	A simple example quantum circuit.	136
7.8	Development roadmap for the IBM quantum devices.	138
7.9	An exemplary quantum computer from IBM.	139
7.10	A seven qubit processor of a superconducting quantum computer from IBM.	140
8.1	The schematics of the hybrid quantum-classical GAN model.	146
8.2	A fully expressive quantum circuit for three qubits.	149
8.3	The more shallow quantum circuit employed in the hybrid qGAN model.	150
8.4	1D classical discriminator neural network architecture.	150
8.5	The training data downsampling to 8 pixels.	151
8.6	The output of the 1D generator without noise.	152
8.7	The statistics training evaluation of the hybrid QGAN model.	153
8.8	The influence of simulated hardware noise on inference.	155
8.9	Example shower images generated with various noise models and noise levels.	157
8.10	The average probability distribution generated on the <code>ibmq_athens</code> device.	158
8.11	The influence of simulated hardware noise to training.	159
8.12	Relative entropy of qGAN training in the presence of readout noise.	160
8.13	The statistics evaluation of multiple training trials with hardware noise.	161

8.14	Relative entropy as a function of training epochs for different bit-flip probabilities.	163
8.15	The eight pixel shower image, the relative entropy and the loss functions of the training on the real quantum device in fairshare mode. . . .	164
8.16	The eight pixel shower image, the relative entropy and the loss function of the hybrid training in fairshare mode.	166
8.17	The downsampling of the training data to 2D.	167
8.18	The initial 2D quantum generator circuit with a linear entangling.	168
8.19	The schematic Tree Tensor Network architecture.	168
8.20	The 2D qGAN Tree Tensor Network generator circuit used in this study. . . .	169
8.21	The eight pixel shower image and relative entropy of the best 2D qGAN model.	170
8.22	The influence of hardware noise to inference of the 2D model.	170
9.1	The general structure of the QAG model.	176
9.2	Angle decoding of an example state.	177
9.3	The pixel energies of the training and test data.	179
9.4	The operated θ region on the Bloch sphere.	179
9.5	The pixel-wise correlation plot of the Geant4 training data.	181
9.6	The MMD and Corr loss weight evolution during training of the QAG model.	182
9.7	The quantum circuit architectures under study.	184
9.8	The characteristic circuit numbers for various architectures.	185
9.9	The accuracy of the various trained circuit architectures depending on the circuit characteristic numbers.	187
9.10	The correlation loss of the various trained circuit architectures as a function of the circuit characteristic numbers is provided.	188
9.11	The results of manipulating individual gates for multiple circuit architectures as an entanglement measure.	190
9.12	The unweighted loss values of the QAG training.	192
9.13	The weighted loss values of the QAG training.	193
9.14	The generated shower shape by the QAG model.	194
9.15	The pixel-wise correlation plot of QAG.	195
9.16	The energy sum histogram including the statistical values.	196
9.17	The four k-means cluster validation metric.	196
9.18	The pixel-wise energy deposition histograms for Geant4 and QAG.	198
9.19	Example shower images generated by the QAG model.	198
9.20	QAG Inference noise study.	199
9.21	The noise levels of the <code>ibmq_montreal</code> device at the time the inference test was executed.	200
9.22	Qubit mapping from the quantum circuit to the real quantum device. . . .	201
9.23	The QAG training noise study.	202
9.24	The losses during the <code>ibmq_montreal</code> and <code>ibmq_cairo</code> training.	204

9.25	Average shower images of the QAG model trained on <code>ibmq_montreal</code> and <code>ibmq_cairo</code>	205
9.26	The correlation plot for the <code>ibmq_cairo</code> training and the energy sum histogram.	205
9.27	QAG training accuracy depending on the number of shots.	206
9.28	QAG inference accuracy depending on the number of shots.	207
10.1	The MERA-downsampling quantum circuit architecture employed as discriminator.	213
10.2	The quantum circuit employed for training the discriminator with the true training images.	215
10.3	Quantum circuit for training the discriminator with fake images and generator training.	216
10.4	The quantum circuit for generating events in inference.	217
10.5	The exponential decay of the generator and discriminator learning rate during training.	218
10.6	The training statistics of the full quantum GAN training.	219
10.7	The characteristic training losses and the generated shower shape.	219
10.8	The pixel-wise correlation plots.	220
10.9	The three hybrid qGAN model discriminator architectures.	222
10.10	The training convergence statistics plot of the three hybrid qGAN models.	223
10.11	The shower shape and the pixel-wise correlation of the hybrid M qGAN model.	224
A.1	The SSIM index depending on the adjusting parameter L	237
A.2	The pixel-wise correlation plots for the Geant4 data and the generated images by the Conv2D model.	239
B.1	The different visualization methods of the CZ -gates.	241
B.2	The MMD loss of the various models from the circuit study of the QAG model.	245
B.3	The eight pixel pixel-wise covariance plot of the Geant4 training data compared to the correlation plot.	248
B.4	QAG example shower images generated with various noise models and noise levels.	249
B.5	The generated shower images of the QAG model with hardware noise and on real hardware.	250
B.6	The noise levels of the <code>ibmq_montreal</code> device.	251
B.7	The noise levels of the training on the <code>ibmq_cairo</code> device.	252
B.8	The pixel-wise correlation and the k-means clustering metric of the full qGAN model.	253
B.9	The pixel-wise energy deposition histograms of the full qGAN model.	253
B.10	The average shower shape plot of the hybrid S qGAN model and the hybrid L qGAN model.	254

B.11 The pixel-wise correlation plot of the hybrid S qGAN model and the hybrid L qGAN model.	254
---	-----

List of Tables

5.1	The number of parameters and the number of convolutional layers for each in this thesis used generator model.	76
5.2	Training computing time evaluation on an Nvidia Tesla T4 GPU.	83
5.3	Inference computing resources comparison.	84
5.4	The Mean Squared Error performance measure as physics validation metric.	88
5.5	Statistic values of the number of hits validation metric.	96
5.6	The statistics values of the energy sum metric.	97
6.1	The speed-ups of the best configurations of the quantized models.	112
6.2	The memory occupation of the quantized and not quantized models.	113
6.3	The Mean Squared Error performance measure.	114
6.4	Statistic values for the shower shape metric.	116
6.5	Pixel-wise image comparison.	117
6.6	The statistic values of the energy sum metric.	122
7.1	The required memory in bytes to represent one quantum state on a classical computer.	141
8.1	Conversion of a quantum state into a classical bit-string.	147
8.2	The readout and CNOT noise levels for running inference and training.	156
8.3	The readout and CNOT noise at each calibration of the hybrid hardware training.	165
9.1	The characteristic circuit values of the quantum circuit study.	186
10.1	The various quantum models tested in this thesis including their accuracy in training and inference.	224
A.1	The hyperparameters for training the different GAN models.	235
A.2	The final training losses of the GAN models.	236
B.1	A verbal description of all circuits used in the circuit study.	244

Chapter 1

Introduction

High Energy Physics (HEP) also called particle physics, studies the building blocks of matter, the elementary particles and their interactions via fundamental forces, which are summarized in the Standard Model (SM) according to current knowledge [1]. Despite the success of the SM, it is incomplete and missing explanations for some physics phenomena, for example dark matter [2]. Today, HEP research explores particle properties of the SM to a better and better precision, but also searches for physics beyond the SM by employing complex experiments [3]. In these experiments, particles are accelerated almost to the speed of light in enormous accelerators and then controlled collided. The largest existing accelerator is the Large Hadron Collider (LHC) at CERN [4]. The particles produced by the collisions can be detected by experiments such as the Compact Muon Solenoid (CMS) [5] detector at the LHC [6].

The evaluation of detector measurements relies on computer-aided simulations to determine, for example, efficiencies, acceptances and purities [7]. Detector simulations calculate the effects of particles traversing through the detector and interacting with the material or decaying [8]. These simulations allow theoretically modeling the detector response as a function of the incident particles and are a key component of particle physics research today. Currently, Monte Carlo based methods are employed to simulate the stochastic behavior of particles [9]. However, they are very computing resource demanding. To reduce the hardware requirements for future experiments, fast simulation approaches are being developed to accelerate the simulation process while maintaining a satisfying level of accuracy [7].

The LHC beams setup will be upgraded to reach a higher luminosity in 2029 [10]. A higher luminosity will lead to much higher collision rates, which will allow to observe extremely rare processes and to reduce the statistical uncertainties of rare collisions, and thus compare the predictions of the SM with a greatly improved experimental

precision. However, higher collision rates are accompanied by a greater need for simulated events. Recent estimations expect that the computational demand increases in the order of a hundred times during the high luminosity phase [11]. With this, the expected improvement in hardware as well as the financial means to purchase additional hardware cannot be sustained [11]. Therefore, an intense research is ongoing in searching for alternative, faster simulation approaches to replace the conventional Monte Carlo simulations currently in use.

Artificial intelligence represents one of the most popular new computing techniques in many research areas due to the increasing amount of available data, better computing resources, and an improving efficiency of the underlying software algorithms [12]. The Deep Learning (DL) sub-field is changing the way complex data is processed on a large-scale with computers. Through the use of deep neural networks, it is possible to achieve accuracies similar to the level of humans in complex tasks [13]. Currently, Generative Adversarial Networks (GANs) [14] represent a successful approach to solve unsupervised generative tasks. A significant portion of this thesis addresses the optimization of a DL GAN approach for calorimeter simulations to generate precise particle shower images. The GAN model is evaluated and compared to the current Monte Carlo simulations by measuring the speed-up and providing a detailed assessment of the physics accuracy. Multiple physics metrics will allow a deeper physics understanding of both the training process and the generated shower images.

One state-of-the-art research topic explores the quantization of neural network parameters into lower precision formats [15]. Quantization can offer the benefits of shorter computing times and lower hardware memory occupation, albeit at the cost of potentially compromising the accuracy. In this thesis, complex generative models are quantized to a lower precision, in contrast to simpler classification models in other previous studies. The quantization is performed by employing the brand new accuracy-driven Intel Neural Compressor framework [16]. The astonishing 3.0x speed-up of the Neural Compressor model on the latest Intel CPU compared to the initial not quantized model is assessed. Further, a nearly similar level of accuracy is maintained due to the accuracy driven quantization. For comparison, a model quantized using the not accuracy driven TensorFlow Light framework [17] suffers from decreasing accuracy.

A new realm of improvements is opened up by the novel and rapidly evolving technology of quantum computing. Quantum algorithms possess potential advantage over classical algorithms in terms of reducing the computational complexity, reaching a higher accuracy, calculating faster and more energy-efficient, or even solve tasks that cannot be solved classically at all [18]. Developments in quantum computing are currently showing tremendous progress and popularity. The HEP community expects quantum computing to have a significant impact on physics research [19]. This PhD work is a part of the CERN Quantum Technology Initiative [20], which was established to navigate and coordinate quantum technology research in HEP. In addition to

the expected increase in speed, there is another potential advantage of quantum computing specifically for HEP applications because the employed HEP simulation data sets originate from quantum mechanical effects [21]. This might induce advantages since quantum computing employs quantum mechanics processes, likewise.

Quantum Machine Learning (QML) represents a rapidly evolving field that follows the track of classical Machine Learning (ML) on quantum devices. QML turns out to be one of the most promising applications on Noisy Intermediate-Scale Quantum (NISQ) devices [22]. The reason behind this lies in the fact that QML is anticipated, and to a certain extent already demonstrated, to exhibit noise resistance within a specific range of noise levels, a characteristic highly relevant to present NISQ devices. To meaningfully apply quantum algorithms today, primarily hybrid approaches are tested where a part of the computations are performed on a quantum computer and another part on a classical computer. In this research, initially hybrid quantum Generative Adversarial Networks (qGANs) are studied, which exhibit a potential exponential speed-up compared to classical GAN models [23]. The robustness of the hybrid qGAN model is thoroughly assessed through an extensive noise study. First, a 1D and then an extended 2D model is employed.

The current quantum generative models are associated with limitations. For example, the previous introduced hybrid qGAN merely generates probability distributions and cannot produce absolute energy values. Further, qGAN training is complex and computing resource inefficient. Other quantum models often do not scale well in terms of number of qubits or gates required to run on NISQ devices. To overcome these obstacles, the development of an enhanced quantum generative model was undertaken. The Quantum Angle Generator (QAG) employs angle encoding to correctly generate shower images. Due to the training by objective functions, the training converges fast and stable. The generated individual images are extremely precise, as proven by several in-detail physics accuracy metrics. On top of that, the QAG model is suitable to run on current NISQ devices due to its noise resistance and fast training. Therefore, training and inference are successfully executed on real quantum devices. Multiple quantum circuit architectures are tested and evaluated in a circuit study.

As a proof of concept, the QAG model is transformed into a full qGAN, where the quantum generator is trained versus a quantum discriminator. This represents one of the first successfully trained full quantum GAN models. The training by a discriminator could be advantageous for larger problem sizes on future improved quantum devices compared to training with objective functions. Furthermore, the full qGAN model is used to perform a comparison of the representational power of quantum circuits to classical neural networks.

1.1 Thesis Objectives

The aim of this thesis is to alleviate future physics computing by developing and testing alternative high-performance computing techniques and validate them with deploying physics metrics. This thesis consists of two parts: classical computing to overcome near-term limitation, and quantum computing for possible future improvements in computing.

The key objectives of the classical computing part of this thesis are:

- Developing an alternative DL approach to the current Monte Carlo calorimeter simulations as a proof of concept. Compare multiple state-of-the-art techniques and find the best suited model for this PhD work.
- Improving the first model to reach the best possible accuracy, while maintaining a considerably shorter simulation time.
- Develop and employ multiple validation metrics for a detailed physics accuracy evaluation and measure the speed-up of the models on CPU and GPU.
- Optimize the model with state-of-the-art techniques for minimal hardware resource consumption, like memory occupancy, and run time.

The second topic of this thesis presents first approaches of employing quantum computing for calorimeter simulations. The objectives are:

- Start with a proof of concept. Modify and adapt a first quantum model able to generate averaged simplified calorimeter shower images.
- Improve this model regarding training time and accuracy and investigate its limitations.
- Perform studies investigating the robustness of the algorithm against noise with simulations as well as for real quantum computing hardware systems.
- Develop a novel full quantum model which overcomes the limitations of the initial model. The novel model should deliver useful results on current NISQ devices because it should demonstrate robustness against noise while maintaining a high accuracy.
- Compare quantum versus classical approaches and explore potential advantages and disadvantages.

1.2 Structure of the Content

The content of this thesis is systematically divided into the following chapters:

- Chapter 1 (this chapter): *Introduction*. The topic of this thesis is introduced. The research motivations and objectives are explained.
- Chapter 2: *High Energy Physics Experiments*. The current physics knowledge and the techniques of searching for new physics are provided. Furthermore, the use case of calorimeter detector simulations is introduced and the computing hardware limitations of future detector simulations, which motivate the search and development of alternative more computing resource efficient simulation approaches. Moreover, the employed training data set is presented and explained.
- Chapter 3: *Deep Learning for Future High Energy Physics Simulations*. A brief introduction into modern DL and existing generative models. Furthermore, neural networks and the utilized layers are introduced.
- Chapter 4: *3D Generative Adversarial Network Models*. The GAN approach including advanced GAN variations are introduced in detail and how the GAN model is adopted for the calorimeter use case. Initial neural network architectures are explained, as well as difficulties and solutions to overcome some frequently confronted GAN training problems.
- Chapter 5: *Transition from Convolutional 3D to faster 2D Networks*. Completely novel neural network architectures are developed, which solve 3D image tasks with only 2D operations. The gain in computational speed-up, reduction of memory occupancy and physics accuracy are evaluated and discussed.
- Chapter 6: *Reduced Precision Strategies*. The process of quantizing the number format of the trained neural network parameters to a reduced precision is explained. The possible benefits of reduced precision computing are explained, and the accomplished 3x speed-up of the quantized model is evaluated.
- Chapter 7: *Quantum Computing*. Introduction to quantum computing, the second key topic of this thesis. Additionally, information on possible advantages of quantum computing over classical computing and the limitations of current NISQ devices are provided.
- Chapter 8: *Hybrid Quantum - Classical Generative Adversarial Network Model*. Presentation of a first hybrid quantum-classical 1D qGAN model for generating calorimeter shower images as a proof of concept. The model is assessed for quantum hardware noise, and the initial 1D eight pixels model is enlarged to a 2D model with 64 pixels.

- Chapter 9: *Quantum Angle Generator*. The QAG, a novel quantum model, is developed and introduced. It overcomes the limitations of the previously optimized 1D qGAN model. Furthermore, it achieves a very good physics accuracy and is capable to run on current NISQ devices. Therefore, a quantum hardware training is executed and evaluated, demonstrating robustness against noise while maintaining good physics accuracy.
- Chapter 10: *Full Quantum Generative Adversarial Networks*. The QAG model is converted into a full qGAN model as a proof of concept of full quantum adversarial training. Furthermore, the representational power of quantum circuits is compared to classical neural networks.
- Chapter 11: *Summary, Future Work and Conclusion*. The content and achievements of this thesis is summarized. Ideas and improvements for future research are presented, and a conclusion is drawn.

Chapter 2

High Energy Physics Experiments

High Energy Physics (HEP), also called particle physics, studies the building blocks of matter, the elementary particles and the forces of nature. The current theory of HEP is summarized in the Standard Model (SM) of particle physics [1], which successfully describes the structure of matter and the fundamental forces. It classifies the known elementary particles and describes their properties. The theoretical model is tested in various experiments and verified today to a high precision [24]. The SM contains fermions, which are elementary point-like particles, and gauge bosons, which represent the fundamental interactions between fermions, but also between bosons themselves. Fermions and bosons differ in their spin, fermions remain half-integer spin ($\frac{1}{2}$, $\frac{3}{2}$, $\frac{5}{2}$, ...), and follow the Fermi-Dirac statistics, whereas bosons have integer spins (0, 1, 2, ...) and follow the Bose-Einstein statistics [25].

The group of fermions can be further divided into leptons and quarks. Leptons interact with the electroweak force, which represents a unification between the electromagnetic and the weak force. The quarks interact additionally with the strong force mediated by the gluons with eight different color charges. The quarks and leptons are grouped into three generations. They are ordered from lower masses in the first generation to higher masses in the third generation [26]. More massive particles require higher energies to produce them. The higher the necessary energy, the longer it took to develop and build the accelerator technology to discover them. The particle generations are further motivated by a similar weak isospin T_3 component within one generation. The weak isospin represents a quantum number related to the weak interaction [27]. The leptons in each generation contain a negative charged particle (charge $-1e$) and a particle with zero charge called the neutrino. The particle charge is consistently related to the elementary charge e . The charged leptons are named electron, muon and tau. Their respective neutrinos are electron-neutrino, muon-neutrino and tau-neutrino. The quarks are divided into up-type quarks with charge $+\frac{2}{3}e$ and

		Fundamental Particles		
		1. Generation	2. Generation	3. Generation
Quarks		up	charm	top
		down	strange	bottom
Leptons		electron	muon	tau
		electron neutrino	muon neutrino	tau neutrino

Figure 2.1: The fundamental particles of the Standard Model.

down-type quarks with charge $-\frac{1}{3}e$. The up-type quarks are named up, charm and top. The down-type quarks are down, strange and bottom. In figure 2.1 an overview of the fermions of the SM is presented. Each fermion particle in the SM includes its corresponding antiparticle. The antiparticle exhibits identical properties to its corresponding particle, except it has the opposite charge. Therefore, all additive conserved quantum numbers are changed in sign [27]. Because of the natural decay of more massive particles to lighter ones, the most often occurring particles in our universe represent first-generation particles. All first-generation particles are stable [27, 28].

The interactions between particles of the SM are described by forces based on quantum field theory [29]. Four fundamental forces of the SM are known, as indicated in figure 2.2. Each force is described by an underlying theory mediated by a boson. The electroweak and the strong force are successfully described by the SM, however, the SM lacks a description for gravity. Gravity is properly understood in large scale physics, but because of its relatively weak strength compared to the other forces, it is not yet measured on particle physics level and is usually neglected in the SM and HEP analysis [30]. The electromagnetic interaction is mediated by photons as force carriers, the weak interaction by two charged $W^\pm$ bosons and neutral Z bosons and the strong force is mediated by eight gluons. The fundamental forces are described in gauge invariant quantum field theory with the symmetry groups U(1) (hypercharge), SU(2) (weak isospin) and SU(3) (color) [29]. After many years of research, the SM was further supported by a missing puzzle piece with the discovery of the Higgs boson responsible for the masses of the particles [31]. The Higgs boson was already suggested back in 1960, but merely found in 2012 with an accelerator with a sufficient energy and luminosity. It was detected at the LHC at Conseil Européen pour la Recherche Nucléaire (CERN) in Geneva [31]. The Higgs represents an electric neutral particle with integer spin zero and counts, therefore, to the group of bosons.

Fundamental Forces			
Fundamental Force		Theory	Mediator
Electroweak unification	Electromagnetic	Quantum Electrodynamics (QED)	Photon
	Weak	Electroweak Theory (EWT)	W and Z boson
	Strong	Quantum Chromodynamics (QCD)	Gluons
	Gravity	General Relativity (GR)	Gravitons (hypothetical)

Figure 2.2: The fundamental forces of the Standard Model and the corresponding mediating bosons.

Today, particle physics is driven by achieving precision measurements and trying to discover new physics phenomena to answer fundamental questions left open by the SM [2]. Not all particle physics parameters are predicted by the SM. Some fundamental parameters are observed by measurement, for example the Higgs boson mass. To provide unambiguous predictions, HEP research is focusing on measuring the parameters of the SM to a better and better precision [3]. High precision measurements require enormous amounts of recorded experimental data of particle collisions, which are then analyzed with high performant statistical and artificial intelligence methods [2].

Another goal of highest priority is to observe a breakdown of the SM to gain insights into physics not yet included in the SM [32]. Complex particle accelerators reaching the highest particle energies and intensities are built to test the SM and to search for physics beyond the SM to explain the multiple open questions which are not resolved by the SM. The major open physics questions are:

- **Dark matter and dark energy:** Dark matter represents a kind of matter which is not visible to us but measurable and, therefore, one of the most puzzling mysteries of our time. In cosmology, the movement and the speed of galaxies can solely be explained with the help of matter which we cannot see. Because of this, it is entitled dark matter. The existence of dark matter is not yet proven, but it is assumed, that only $\frac{1}{6}$ -th of the universe matter is visible [33]. Thus, dark matter searches represent a key area of interest.
- **Matter-antimatter asymmetry:** It is assumed that at the Big Bang the equivalent

amount of matter and antimatter was created due to conservation laws. Over the following evolution, parts of matter and antimatter annihilated back into energy. But today, only matter can be discovered in the universe. This provides a strong hint of a present matter-antimatter asymmetry in our universe. It is attempted to be measured by CP (charge-parity) violation in fundamental particle processes, for example in neutrino oscillations [34, 35].

- **A unified theory:** In particle physics, a model is searched which can combine the electromagnetic, the weak and the strong forces into a single unified force. There are at present several unification approaches suggested which need to be tested and optimized, for example Supersymmetry [36]. However, until now, there has been no evidence that a unified theory exists. On the other side, in the past, it was already possible to confirm with experiments the combination of the electromagnetic interaction and the weak interaction into the electroweak interaction [27].
- **Gravity:** As already mentioned, the graviton represents the hypothetical force mediator of the gravitational force. Although gravity is defined by general relativity and has a huge impact in our every-days life, it has not been possible to measure it in particle physics as a fundamental force. Mathematical theory expects the graviton to be a massless spin two boson [37].

In particle physics, nuclear physics and astrophysics, typically the “natural units” are employed as a system of units [38, 39]. To prevent large or small numbers, physics constants are set to one ($\hbar = c = \epsilon_0 = 1$). The energy is typically expressed in the unit electron volt (eV), which corresponds to the energy a particle gains by travelling through a potential of one volt. The typical order of energies in HEP experiments is $10^9 \text{ eV} = \text{GeV}$ (giga-electron volt). Due to conventions, the mass, momentum and energy of particles are expressed in GeV, and time and distance are expressed in $1/\text{GeV}$ [40]. Furthermore, the electric charge of particles is represented by a multiple or a fraction of the elementary charge e [38]. The natural units are equally employed in this thesis.

The structure of the following content of this chapter is: Initially, the relevance of particle physics accelerators is introduced. Subsequently, the detector principle and especially calorimeter simulations are explained. Next, the importance of computing in HEP is demonstrated. Particularly, the current operated simulation approach is introduced and motivations for replacing it with faster methods. This is followed by the visualization and discussion of the calorimeter shower images, which correspond to the training data of the latter introduced models in this thesis.

2.1 Accelerator Physics at the Large Hadron Collider

In this section, the relevance and the operating principle of particle physics accelerators are described based on the LHC as a representative. Some inspiration of the presented content is derived from Ref. [24]. The LHC is the world's largest particle accelerator built across the border of Switzerland and France, close to the city of Geneva. The LHC belongs to the group of circular circumference accelerators because it is constructed in a ring architecture. Its circumference is 26.7 km. Because of this tremendous circumference and the advanced operated technology, the LHC can currently accelerate particles to energies up to $E_{beam} = 6.8 \text{ TeV}$, higher than every other particle collider at present. The accelerated particles are controlled collided in particle physics detectors, often denoted as experiments. The particles produced in the interactions are then accurately measured for further precise analysis. The LHC represents a proton-proton collider where bunches of protons are accelerated close to the speed of light in two counter-rotating beams in separate beam pipes. Particles of the two beams are controlled collided in head-on collisions in one of the four larger experiments of the LHC: ATLAS, ALICE, CMS and LHCb, respectively. This head-on collision technique enables the highest total collision energy, because it adds the energies of the two counter-rotating particle beams, which results in $\sqrt{s} = 2 \cdot E_{beam} = 13.6 \text{ TeV}$ collisions. This record energy was only recently set in July 2022 and represents the current world record for the highest man-made energy collisions. The center-of-mass energy $\sqrt{s}$ of the collision is a characteristic number of the power of an accelerator.

The CERN accelerator complex is schematically provided in figure 2.3. It can be noted that the protons go through smaller accelerators for pre-acceleration before they are injected into the LHC. Pre-acceleration is required because radio-frequency cavities (RF cavities), which accelerate the beam with electromagnetic fields, effectively operate and accelerate particles merely in certain frequency ranges. Therefore, the beam-lines (including bending magnets and RF cavities) of each accelerator are optimized for specific energy ranges.

In the following, the pre-acceleration process of the CERN accelerator complex from figure 2.3 is explained. The pre-acceleration chain of the LHC starts with hydrogen gas stored in a gas bottle. First, one electron is added to the hydrogen atoms, resulting in negatively charged hydrogen ions (H^-) [42]. The hydrogen ions are accelerated to higher energies by the Linear Accelerator 4 (LINAC4). LINAC4 replaced LINAC2 just recently in 2020 and is a key element for increasing the luminosity of the LHC in the following decade. The luminosity represents a characteristic of a particle accelerator indicating the number of interactions that can be produced and is in more detail described later in this thesis. After the pre-acceleration in the LINAC4 up to around 150 MeV, the negative hydrogen ions are sent into the Proton Synchrotron Booster (BOOSTER). In the BOOSTER, the hydrogen ions are stripped off their two electrons

The CERN accelerator complex Complexe des accélérateurs du CERN

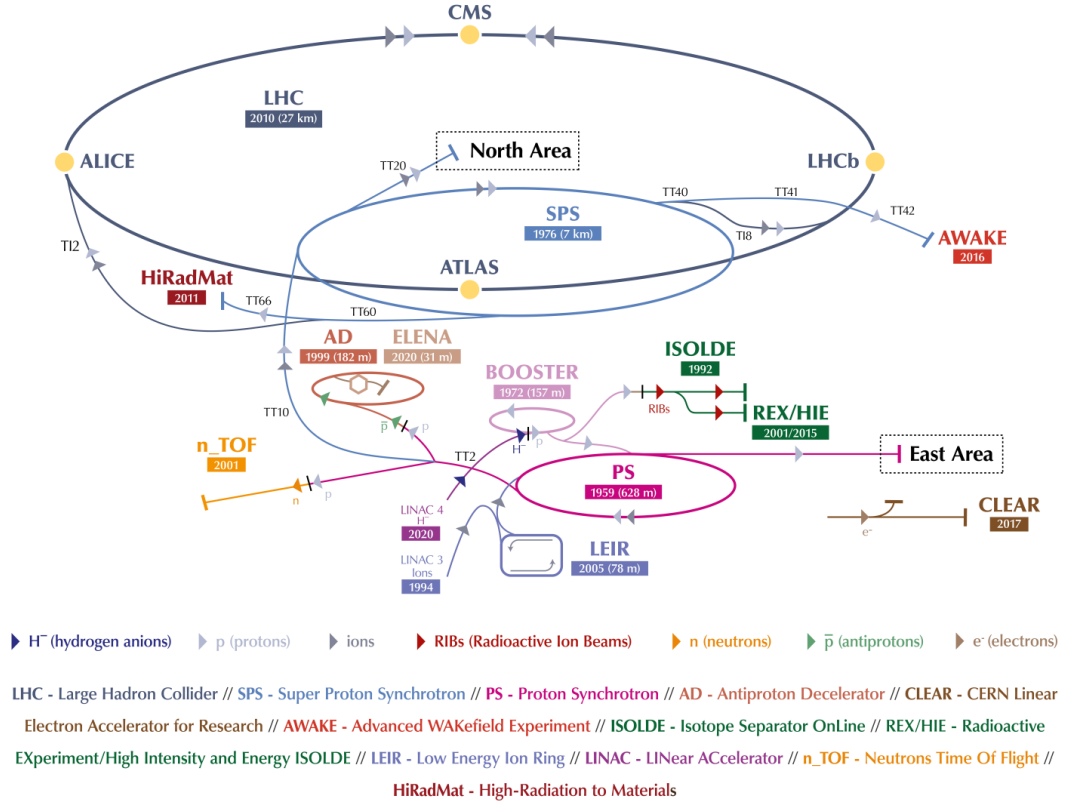


Figure 2.3: The CERN accelerator complex, including the LHC and its pre-acceleration chain. Among others, the four large LHC experiments ATLAS, ALICE, CMS and LHCb are illustrated. The figure is taken from Ref. [41].

and only protons leave. The BOOSTER was added in 1972 to the pre-acceleration chain and allows the following accelerator to accept 100 times more protons. The protons in the BOOSTER are accelerated to 2 GeV and then injected into the Proton Synchrotron (PS). The PS was built in 1959 with a 628 meters circumference and accelerates the protons to approximately 26 GeV. Subsequently, the extracted protons are further injected into the next larger accelerator, the Super Proton Synchrotron (SPS). The SPS was built in 1972 and has a 7 km circumference. In the SPS, the protons are accelerated to 450 GeV and injected into the LHC. The injection to the LHC is performed with the two beam transfer pipes TI2 and TI8 in different clock directions, as shown in figure 2.3. In the LHC, the protons are accelerated to energies up to 6.8 TeV in two counter rotating beam pipes and then controlled collided [43].

In the LHC, not only protons can be accelerated and collided. Also, composite particles, such as heavy ions, are successfully accelerated and collided. Some examples are Pb+Pb, Xe-Xe and p+Pb collisions at multiple energy levels to study specific particle properties and quark-gluon plasma configurations [44]. For accelerating the ions, a different pre-acceleration chain is chosen. The ions are initiated in LINAC3 and then accumulated and further accelerated in the Low Energy Ion Ring (LEIR) and only then sent to PS, SPS followed by the LHC, as visualized in figure 2.3. Furthermore, the CERN accelerator complex consists of multiple other experiments and smaller accelerators for specific types of research. In the following, the major four experiments at the LHC are briefly described.

The largest and most famous two LHC experiments are ATLAS and CMS. In HEP language, experiments refer to large detectors in which the particles are collided, and the results measured similar as in laboratory experiments. ATLAS and CMS are multipurpose detectors, which have a broad range of physics goals, from studying the SM up to physics beyond the SM [45]. One of the primary goals was to detect the Higgs boson. This was successfully completed in both, the ATLAS and CMS detector, in 2012. For CERN and the physics community, it was decided that two independently designed detectors are vital for cross-checking novel discoveries. The other two major experiments at the LHC represent ALICE and LHCb, designed to focus on specific physics aspects. ALICE examines the collisions of heavy ions at ultra-relativistic energies at the highest energy densities. LHCb focuses on investigating differences between matter and antimatter with the help of the bottom quark (in this context, the bottom quark is often denoted as beauty quark). All four experiments are positioned at the LHC ring in large underground caverns.

Instead of single particles, in the LHC bunches with more than 10^{11} particles are accelerated to increase the probability of collisions when two bunches "hit" each other. Only a minor fraction of the particles of the bunches collide when the two counter rotating bunches hit each other, because of tiny collision probabilities. However, the not colliding leftover particles are not lost. They are recuperated in the LHC and are prepared for new collisions.

The pipes which guide the beam around the accelerators are operated at an ultrahigh vacuum to prevent unwanted collisions with air particles. The bending of the beam is performed with the help of superconducting magnets. To maintain the superconductivity of the magnets, ultra-low temperatures below $-271\text{ }^{\circ}\text{C}$ close to the absolute zero point are required. The cooling is provided by a complex liquid helium system. The full LHC beam-line consists of 1232 dipole magnets to bend the beam around in a circle, each with a 15-meters length. Additionally, 392 quadrupole magnets with a five to seven meters length keep the beam focused. The whole LHC is built in a tunnel in 50 to 150 meters depth below the ground surface [46].

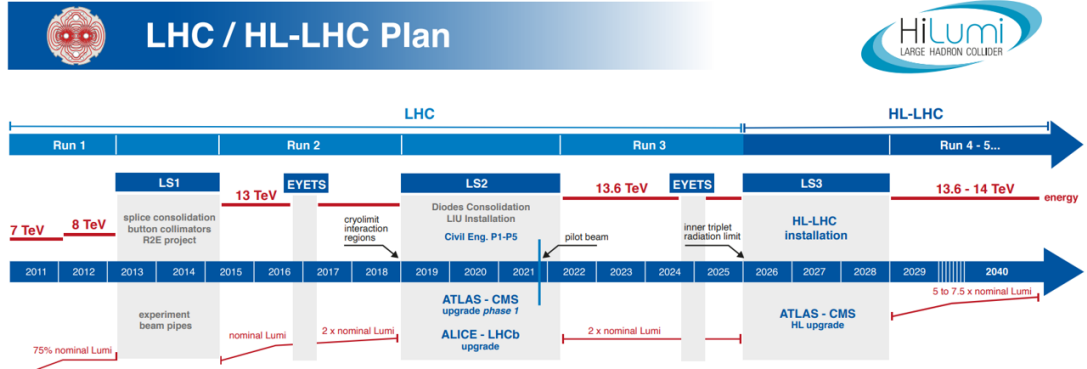


Figure 2.4: The LHC Run plan (July 2022). After Run 3, the LHC will be upgraded to the High Luminosity LHC, starting its operation from 2029 on-wards [47].

The LHC data taking time is split into several phases, denoted as Runs. The individual Runs are separated by long shutdowns (LS) in which the accelerator and the detectors are upgraded to superior technology. The LHC Run plan from CERN is provided in figure 2.4. Currently, Run 3 is ongoing, which is supposed to end in 2025. Then, the installations of the LS3 start. The High-Luminosity LHC (HL-LHC) in Run 4 is planned to start collecting data from 2029 onwards. The luminosity is a significant indicator of the performance of an accelerator on the technical side and is a measure for the frequency of interactions. The luminosity L depends exclusively on the properties of the accelerator and can be determined by the equation from Ref. [48]:

$$L = \frac{N_1 N_2 N_B}{4\pi\sigma_1\sigma_2} f. \quad (2.1)$$

N_1 and N_2 represent the numbers of particles in each of the two colliding bunches, N_B the numbers of bunches within one LHC fill, σ_1 and σ_2 the cross-sections of each bunch and f the frequency of colliding bunches per second. Equation 2.1 is idealized and lacks correction factors. For example, in reality the two beams are colliding under a narrow angle in order to separate the two beams afterwards to re-use them. A higher luminosity aims for more collisions per time-slice and, therefore, for a superb performance leading to potential physics discoveries. With the HL-LHC, it is expected to produce at least five times as many Higgs bosons compared to the LHC Run 2 in 2017 [49]. Among other advancements, some key innovations in accelerator technology included in the HL-LHC are new cutting-edge twelve Tesla superconducting magnets and more compact superconducting cavities for beam steering along the beam pipe [49].

At the LHC, primarily protons are accelerated, which are complex composite particles consisting of quarks and gluons. More specific, protons consists of three valence quarks — two up-quarks and one down-quark — and a so-called “sea” of gluons and quarks. Because of the compound particles, in reality, the sub-particles of the protons

are collided by deep inelastic scattering processes, in which the initial proton is destroyed [50]. Proton accelerators have the advantage that – due to the high particle mass compared to elementary particles, for example electrons – they emit a considerably lower synchrotron radiation. Synchrotron radiation is the physics phenomenon in which accelerated (and de-accelerated) charged particles emit electromagnetic radiation [51]. At bending, synchrotron radiation is emitted tangential to the direction of motion and the radiation amount depends on the magnitude of the acceleration. In linear accelerators, the particles experience only an acceleration in the direction of the beam axis, leading to negligible synchrotron radiation. In circular accelerators, however, the particles are deflected additionally transverse to the direction of motion by the bending magnets in order to guide them around the ring. This transverse acceleration is significantly large, and consequently the emitted synchrotron radiation is large compared to linear accelerators. The mathematical equation from Ref. [51] which describes the synchrotron radiation ΔE perpendicular to the direction of particle motion is:

$$\Delta E = \frac{e^2}{3\epsilon_0(m_0c^2)^4} \frac{E^4}{\rho}. \quad (2.2)$$

In the equation, e describes the elementary charge of the accelerated particle, ϵ_0 the dielectric constant, m_0 the mass of the accelerated particle, c the velocity of light, E the electric field and ρ the radius of the circular trajectory. The mass m_0 of the accelerated particle is present with the fourth power inside the denominator. For lighter particles, such as electrons (mass m_e), the synchrotron radiation becomes extremely large, whereas for heavier particles, as protons (mass m_p), the synchrotron radiation decreases strongly by an estimated factor of $(\frac{m_p}{m_e})^4 \approx 10^{13}$. This allows accelerating protons to high energies in ring accelerators with a smaller circumference because they lose less energy by synchrotron radiation. More details about synchrotron radiation from the theoretical side is provided in Ref. [51] and from the practical side applied to particle accelerators in Ref. [52].

2.2 The Principles of Particle Physics Detectors

The goal of HEP experiments is to accurately measure the final state of the particles produced in the collisions, what requires special designed detectors. There is not one single detector principle which can measure all the physics properties of interest [53]. Therefore, detectors are built by multiple sub-detectors in the onion principle cylindrical around the beam pipe. Each sub-detector is designed to measure particular aspects or properties of the collision particles. Thus, HEP detectors are dominantly multipurpose detectors. The most important measured properties are positions, momenta, energies, angles, timing and charges of the individual particles.

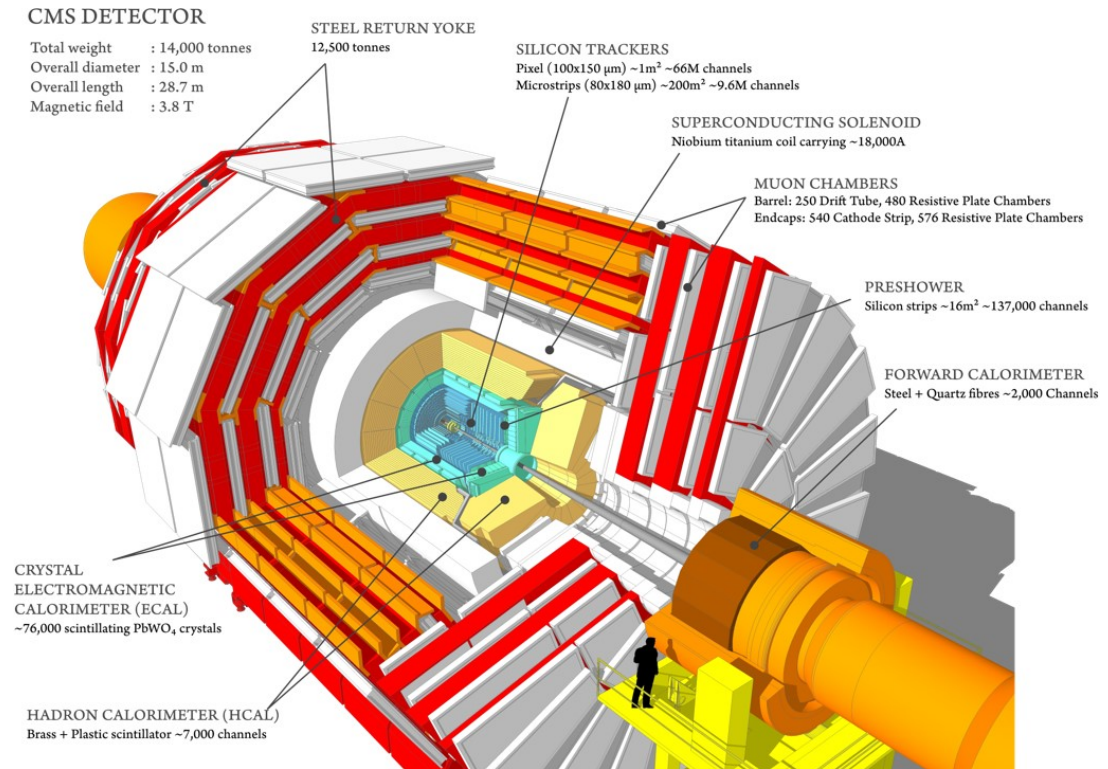


Figure 2.5: The CMS detector concept from Ref. [5]. The CMS detector consists of multiple sub-detector systems, illustrated in the figure.

The Compact Muon Solenoid Detector

The principle of detectors is described based on the Compact Muon Solenoid (CMS) detector [45]. As already introduced before, the CMS detector constitutes one of the four main LHC experiments built as a multipurpose device. As demonstrated in figure 2.5, the CMS detector represents a gigantic construction with a diameter of 14.6 meters, a length of 21.6 meters and an absolute weight of more than 12.5k tonnes. The detectors are so large because they have to stop tremendously energetic particles created by the beam particles in the extremely powerful accelerators. Principally, the size of the detectors scales with the collision energy in the accelerator. The CMS detector aims at similar measurement goals as the ATLAS detector, however, it operates other technology and a different magnetic system for its measurements. The CMS apparatus consists of several layers of sub-detectors. Each designed to measure certain types of particles and properties [5]. In the following, the CMS sub-detectors, including their working principles and purposes, are explained from the inside close to the beam pipe to the outside.

Tracking System

The CMS tracking system comprises vertex and strip tracking detectors. The innermost detectors are the vertex detectors (named PIXEL). They consist of silicon layers with more than 124 million pixels and maintain an extremely precise resolution. Each pixel has approximately the size of two hairs' width ($\approx 100 \mu\text{m}$) [5]. Charged particles which pass through a sensor produce free electron-hole pairs from the silicon atoms. These electrons generate small electric signals which are amplified and measured to derive the particle trajectories in a 3D picture [5].

The principal purpose of the vertex detectors is to separate different particle collisions by reconstructing the primary vertex of the single interaction events. Because in the accelerator, when bunches of particles are colliding, regularly multiple particle interactions occur at the same time. At each collision at the LHC, around 1 000 particles traverse the detector from more than 20 overlapping proton-proton collisions [54]. For finding the original collision point, the tracks in the vertex detector are extrapolated backward. Additionally, vertex detectors are responsible for reliably distinguishing between primary vertices and possible secondary vertices from short-lived particles, as for example employed for b -jet tagging [24]. This requires that vertex detectors are built as close as possible around the beam pipe with the maximum realizable spatial resolution. In addition, the CMS tracking system technology is optimized for a very fast response, as well as a high identification reliability [45].

The vertex system measures around 600 million particles per square centimeter per second [5]. If the entire tracker would be built by vertex pixel modules, the number of readout channels would become enormous. To keep this number reasonable, the vertex detector is followed by a detector with a more limited granularity, the strip tracking detector (STRIPS). It operates similar to the vertex detector and is employed to measure the tracks and the momenta of particles, to reconstruct the trajectories of charged particles in the outer region. The charged particles ionize the silicon material by creating an electron-hole pair, which gives a small detector signal enabling to locate the particles. By involving a strong homogeneous magnetic field with a known strength B , it is possible to convert the position readouts of the particles into their momenta. In the magnetic field, the particle paths are bent by the radius R . This curved trajectory can be precisely measured by the multiple layers of the tracking detectors. The direction of the bending of a particle path is given by positive or negative charge. The bending radius R is determined by the magnitude of the momenta of the particles, because particles with lower momentum bend more than particles with larger momentum. The transverse momenta p_T – the momentum component transverse to the magnetic field – can be calculated by the equation:

$$p_T = 0.3 B R, \quad (2.3)$$

as explained for example in Ref. [55]. With tracking detectors, the particle type can

be deduced by analyzing its charge, bending radius and direction of motion in the magnetic field. With the track measurements, additionally the origin of the particle can be determined.

Calorimeter Detectors

The next sub-detectors are the calorimeters, which measure the energy of particles at collider experiments. With the help of particle energy measurements inside calorimeters, the spatial location of the energy deposition can be measured, and in some cases, the direction of the incoming particle. In addition, specialized calorimeters can provide the timing of the traversing particles. These quantities are helpful to reconstruct the mass of the particles. Calorimeters are destructive devices. Primary particles enter the calorimeter and deposit their energies by generating secondary particles by interacting with the detector material [56]. The secondary particles, in turn, create other particles by the same mechanism, resulting in a particle shower. In the following, this procedure is analyzed in more detail.

Calorimeters are typically categorized into electromagnetic calorimeters (ECAL) and hadronic calorimeters (HCAL) [57]. ECALs are designed to measure the energies of electrons, positrons and photons. The underlying electromagnetic interactions in the detector volume are primarily Bremsstrahlung for electrons and pair production for photons. Pair production occurs when a photon with more than 1.022 MeV (twice the electron mass, $2 \cdot 511$ keV) passes near a nucleus of an atom in matter. The strong field from the nucleus induces the photon γ decay into an electron-positron pair:

$$\gamma \rightarrow e^- + e^+. \quad (2.4)$$

The electron e^- is negatively charged and the positron e^+ , its anti-particle, is positively charged. The produced electron and positron conserve the energy and momentum of the photon. However, the interacting nucleus may experience some recoil [58]. The probability of the pair production processes increases with increasing photon energies and with higher atomic numbers of the interacting nucleus present in the calorimeter. The Feynman diagram of the pair production process is provided in figure 2.6 on the left. Feynman diagrams are in particle physics utilized to visualize the interaction of particles.

Bremsstrahlung occurs when an electric charged particle becomes accelerated or decelerated in the longitudinal direction (similar to synchrotron radiation created by acceleration or deceleration in the perpendicular direction of motion). In calorimeters, Bremsstrahlung naturally appears when an electron is slowed-down in matter, and the electron reduces its energy by emitting a photon, as illustrated in the Feynman diagram of figure 2.6 on the right. It can be noted, both Feynman diagrams, pair

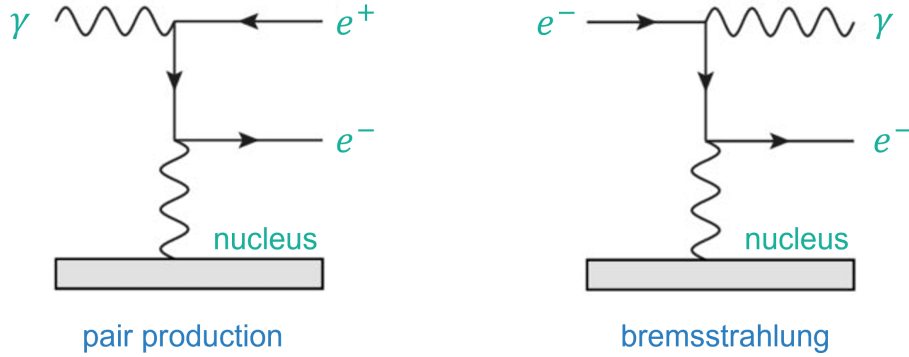


Figure 2.6: Feynman diagram of pair production (left) and Bremsstrahlung processes (right) with a nucleus.

production and Bremsstrahlung, emit two particles while merely one enters the process. This causes the multiplication of particles in the shower process. The lost energy E of an incoming particle due to Bremsstrahlung depends on the path x it traverses through the material and can be determined by:

$$E(x) = E_0 \exp\left(\frac{-x}{L}\right), \quad (2.5)$$

where L is a mass-weighted characteristic radiation length. The characteristic length L depends on the mass M of the incoming particle, as well as on the absorber material properties in which the particle is stopped. It can be calculated by the equation of Ref. [27]:

$$L \propto \frac{M^2 A}{\rho Z^2}. \quad (2.6)$$

The absorber material properties are the atomic mass A , the material density ρ and the atomic number Z [27]. Equation 2.6 explains why tracking detectors are built preferably with as less material as possible and with materials with low Z to reduce the number of reactions before the ECAL detector to not distort the energy measurement inside the calorimeter. Eventually, the secondary particles energies decrease below the critical energy, and they participate in ionization and excitation processes instead of generating more particles in the shower process [57]. Hence, the particle shower comes to an end. The critical energy E_c is defined as the energy at which the energy loss dE per length dx of Bremsstrahlung is equal to the energy loss of ionization

$$\frac{dE(\text{Bremsstrahlung})}{dx} = \frac{dE(\text{Ionization})}{dx}. \quad (2.7)$$

For example, in copper the critical energy of electrons is around $E_c \approx 20$ MeV, whereas for heavier muons, the critical energy is at a few hundred GeV [27].

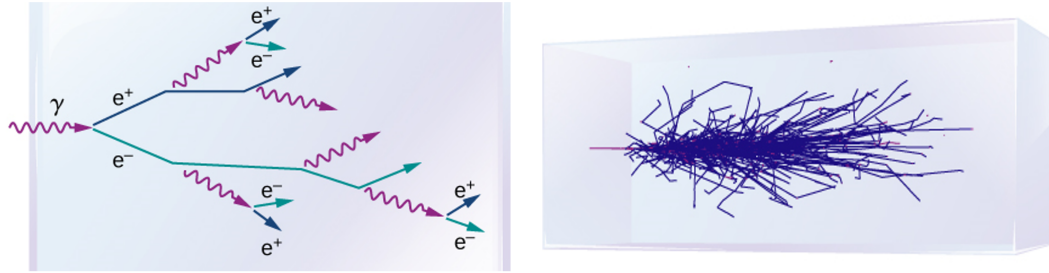


Figure 2.7: On the left: schematic shower process inside an electromagnetic calorimeter. On the right: a more complex simulated shower example as it would be investigated inside a real calorimeter volume. Both images are taken from Ref. [26].

The shower creation mechanism inside an electromagnetic calorimeter detector is demonstrated in figure 2.7. It includes both processes, Bremsstrahlung and pair production, and generates many particles from one initial particle, named primary particle. While the shower evolves, the energy of the particles gradually decreases and is absorbed and measured by the calorimeter sensors.

There exist two principal building types of calorimeters. The first type are homogeneous calorimeters, which consist merely of one material and the second type are sampling calorimeters, which consist of two alternating material layers. The CMS electromagnetic calorimeter represents a homogeneous calorimeter which uses almost 80 000 lead-tungsten (PbWO_4) crystals to detect the scintillation light [59, 60]. The lead-tungsten characteristics of a high density and short radiation length make it the ideal material for a fine granularity and compact calorimeter.

The energy of the particle showers can be measured in the form of charge or light proportional to the energy of the particle. The photon energy is transferred into light by scintillators [5]. Scintillation represents the process when passing particles excite the electrons of the detector material to higher orbits. These excited electrons fall then back and release photons of a specific energy. The emitted photons are subsequently converted into an electrical signal by employing special designed photodetectors for the high magnetic field. As photodetectors, avalanche photodiodes (APDs) are operated in the barrel and vacuum phototriodes (VPTs) in the endcaps [59]. The barrel represents the detector component around the beam pipe, whereas the endcaps are placed perpendicular to the beam pipe at both ends of the barrel. The two calorimeter endcaps measure particles with a trajectory close to the beam pipe [45]. The goal is to catch all particles to obtain precise energy measurements of the collisions.

Among others, the precise energy resolution required of the CMS ECAL is based on the driving criteria to detect the decay channel of the Higgs boson to two photons [45], as visualized in figure 2.8 on the left. The energy resolution of the ECAL is determined

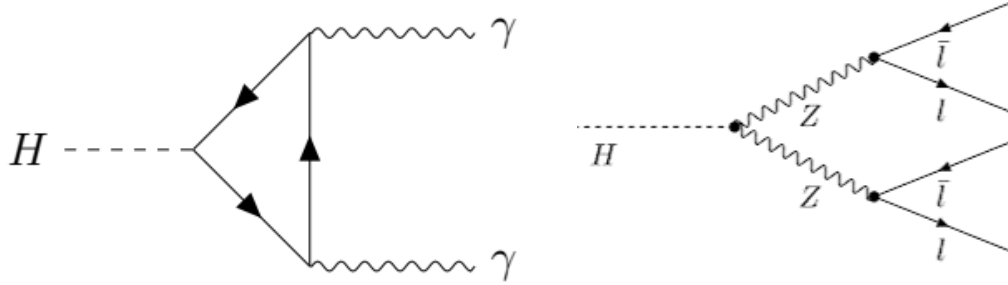


Figure 2.8: Feynman diagram of the decay of a Higgs boson into two photons (left) and into four leptons (right).

by the function from Ref. [45]:

$$\left(\frac{\sigma}{E}\right)^2 = \left(\frac{a}{\sqrt{E}}\right)^2 + \left(\frac{b}{E}\right)^2 + c^2. \quad (2.8)$$

The term (a) represents the stochastic term, (b) the noise term and (c) the constant term. The stochastic term includes sampling fluctuations, representing event by event fluctuations in the lateral shower containment [45]. The stochastic term contributing to the energy resolution increases for lower energies E of the primary particles. On the contrary, the higher the energy of the particles, the better the measured energy precision. The noise term comprises the electric noise in the readout chain, as well as pile-up noise. Pile-up noise is referred to measured energy components from other primary particles. In practice, it is challenging to distinguish the collision of interest from other simultaneous happening collisions. The noise term decreases for higher energetic particles. Lastly, the constant term does not depend on the energy and covers among others: detector aging, radiation damage, temperature effects and calibration errors [57]. Typical values for the CMS detector in the first Run were $a = 2.8\%$, $b = 12\%$ and $c = 0.3\%$ [24].

HCAL detectors are utilized for measuring the energies of hadrons with producing hadronic particle showers. In HCAL detectors, the particles are stopped by strong interactions, regardless of their electric charge. Particles which interact with the strong force consist of quarks and gluons, which mainly transverse the ECAL detector. Representatives are neutrons, protons, pions, kaons and many more. One major role of HCAL represents the measurement of hadron jets and its contribution to the determination of missing transverse energy of non-interacting particles. A hadron jet represents a cone consisting of many hadrons produced by the hadronization of quarks or gluons in the collision [61]. For obtaining the missing energy, it is crucial to measure the full energy of all particles produced in the collision. A more comprehensive description of determining the missing energy is provided later in this thesis. In fact, the total

energy reconstruction of hadrons is a combination of HCAL and ECAL, because some particles already experience their first interaction with the material of the ECAL.

The CMS HCAL calorimeter represents a sampling calorimeter, which enables a more compact building type compared to homogeneous calorimeters [45]. This compactness is of essential meaning for the CMS detector, therefore, it is part of the CMS experiment name (**Compact** Muon Solenoid). The two alternating layers are: (1) absorber layers with highly dense material for fast interaction of the particles with cascade evolution and (2) sensor layers, in which the passing particles are detected [57]. The material utilized for the absorber layers in the HCAL is composed of 70% copper and 30% zinc and the active sensor layers are made from 3.7 mm thick plastic scintillators [45].

The ECAL and the HCAL at CMS are located within the solenoid magnet and must, therefore, consist of non-magnetic materials to prevent deforming the magnetic field. To deliver sufficiently good data quality, the calorimeters have to be built in a high granularity to measure with a high resolution. These aspects are crucial to separate pile-up energy clusters or particle showers of various primary particles. Due to the measurement, if the particles are stopped in the ECAL or HCAL, an additional hint about the particle type is provided.

Solenoid Magnet

The electronic and the hadronic calorimeters are followed by the cylindrical superconducting solenoid magnet, which is one of the key features of the CMS detector. A solenoid represents a specific type of magnet. Due to a tightly wound coil around a helix, it establishes an almost uniform magnetic field inside the magnet. The CMS solenoid creates an enormous maximal 4 T (Tesla) magnetic field, which is around 100 000 times as strong the magnetic field of the earth. Because of this immense magnetic field, the term solenoid is contained in the CMS name (Compact Muon **Solenoid**) [45]. The magnet has a diameter of 6 m and a 12.5 m length. The magnetic field is confined by two steel endcaps at both ends of the barrel magnet. The purpose of the intense magnetic field is to achieve a significant particle bending for receiving the desired momentum resolution within the tracking detector [45].

Muon Detectors

The muon system represents the outermost part of the detector around the solenoid magnet. Muons are able to traverse through meters of dense material without any interaction and, therefore, their detectors are positioned at the outer position with thick

material layers to provide a clear muon identification. The reason why muons are not stopped in the ECAL is that their mass is ≈ 200 times higher than the mass of electrons. With equation 2.5 and equation 2.6, the energy loss depends $1/M^2$ on the mass M of the particle. This results in a roughly 40 000 times lower energy loss of a muon than an electron due to Bremsstrahlung in matter. Therefore, for muons, the energy loss due to ionization dominates completely. The CMS muon system contains four different detection systems, the drift tubes (DT), the cathode strip chambers (CSC), the resistive plate chambers (RPC) and the gas electron multipliers (GEM) [45], which are not further discussed here. The muon momentum can be calculated by fitting a curve into the hits of the measurements of the single layers created by the muon track inside the muon system.

The muon detection is of fundamental importance to CMS and, therefore, implied by the middle name of the experiment (Compact **Muon** Solenoid) [45]. One key reason for the significant importance of the muon system for CMS represents the measurement of the Higgs boson decay channel into two Z bosons, which in turn decay into four leptons, as indicated with the Feynman diagram in figure 2.8 on the right. In Ref. [45] it is stated, in addition to the relative ease in detecting muons, the most appropriate mass resolution for the Higgs search can be achieved if all four leptons are muons. Therefore, the precise muon system supports the Higgs research.

Neutrinos

Other known particles which go through all detector layers are neutrinos. Because of their low interaction rate with matter, they are not detectable at all. Nevertheless, if a physics model predicts the existence of a neutrino in the process, the approximate direction and energy can be indirectly determined by the measurement of the missing energy with the balance of the energies due to energy conservation, if except for neutrino energies all particle energies of a collision are measured. Sufficient precision can only be achieved by employing accurate detector simulations to determine important correction quantities like efficiencies or acceptances. These detector simulations are introduced in the subsequent section.

In figure 2.9, the CMS detector is visualized from a cross-sectional view, including all sub-detectors. Additionally, the trajectories of some characteristic particles through the detector are illustrated, and in which detector component they can be typically measured.

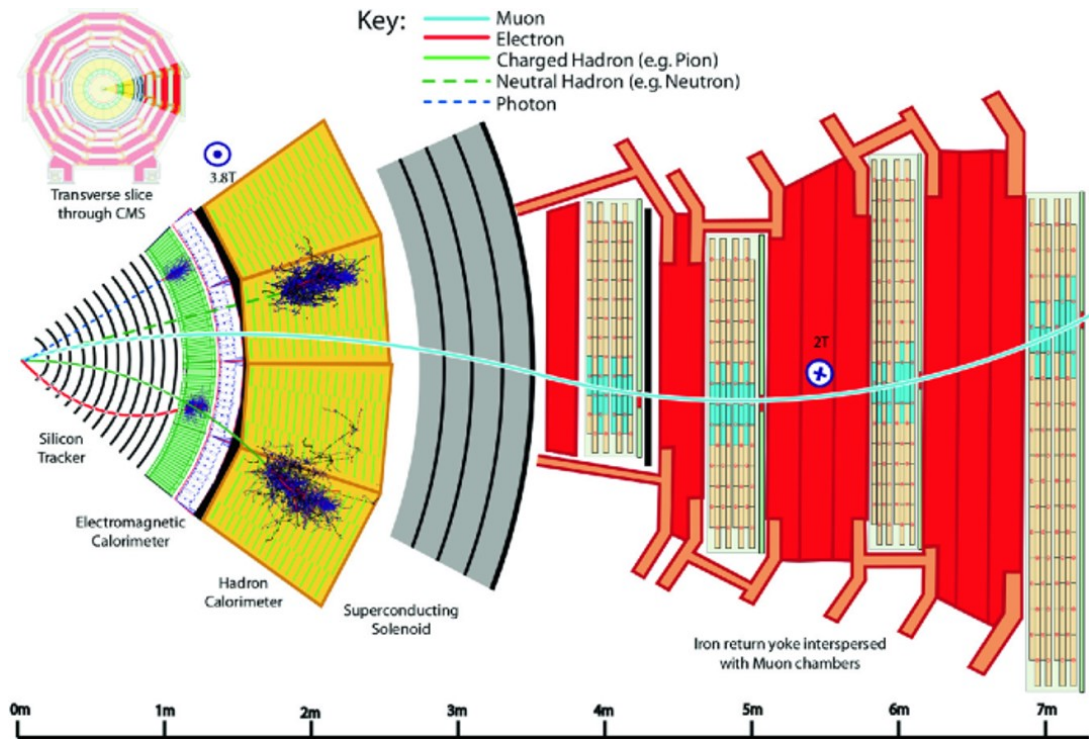


Figure 2.9: Summary of all the physics sub-detector components of CMS in a cross-sectional view, including exemplary particle trajectories. The figure is taken from Ref. [62].

2.3 The Importance of Computing in High Energy Physics

The challenge of HEP research is not only to develop and build the technology for accelerators and experiments. It is additionally about capturing and analyzing the tremendous amount of data measured by the detectors. Furthermore, accurate simulations of interactions of elementary particles in HEP detectors are fundamental for reproducing and interpreting the experimental results and reconstructing particle flows by associating which deposition belongs to which particle [45]. However, identifying the characteristic signatures of the particles remains an enormously complex process, which exclusively can be performed with the help of computers and smart algorithms. The importance of computing for evaluating detector responses has grown over time, because of the increased detector sizes, granularities and precision requirements [7].

Multiple types of simulations exist for the various stages [8]. In general, the detectors require the greatest portion of simulations and are, therefore, in more detail assessed. Simulations are essential to design, develop and build extremely sophisticated detectors and then to evaluate the experimental results by comparing them to theoretical

models [63]. In addition, there exist event generators, Monte Carlo detector simulations, simulations for particle energy depositions, the corresponding signal generation and simulations for the assessment of radiation effects [8].

With the collected data of all the particle sub-detectors, the entire collision can be evaluated. The ultimate goal of LHC detector measurements is to reconstruct the four-vector, or more specifically the four-momentum

$$p = \left(\frac{E}{c}, p_x, p_y, p_z \right) \quad (2.9)$$

of each particle. The speed of light c is by convention set to $c = 1$, E represents the relativistic energy of the particle and $\mathbf{p}$ its momentum with the three spacious components $\mathbf{p} = (p_x, p_y, p_z)$. With knowing the four-momentum, the mass of the corresponding particle can be determined. The mass represents a characteristic value of each particle, which allows identifying the particle type. With the four-momentum and the mass, the energy of the particle can be calculated. With all this information, it is possible to link different particles by the physics laws of the SM together and to develop hypotheses about the interactions from which the measured sub-particles might originate. Ideally, the total detected energy is equal to the center-of-mass energy of the colliding beams. However, in practice, frequently not all particles participating in a collision are identified. Certain particles remain difficult to detect (for example neutrinos which pass through all layers of the detectors because they interact barely with matter) and a few particles "slip through." In theory, this information is demanded to analyze the particle interactions and to understand the physics behavior. In practice, detector simulations help to overcome this obstacle and allow the reconstruction of the whole decay chains, even containing a few "slipped through" particles.

This mathematical reconstruction work of the aforementioned processes is extremely complicated. Furthermore, particle collisions happen at high frequencies, and usually multiple collisions take place simultaneously (pile-up). Obtaining a considerable number of collisions is fundamental for the statistical evaluation of the physics probabilities, which frequently require more than millions of collisions [7]. Today, it is impossible to perform these calculations by hand. Therefore, this work is taken over by detector simulations, carried out by large computing clusters. This makes computing a fundamental ingredient of HEP research and highlights the acute need for detector simulations to gain new physics results. Without computing and detector simulations, it would not be possible to analyze the immense amount of measured data and achieve the necessary precision to extract new insights into physics laws or even make discoveries at sufficient low uncertainties.

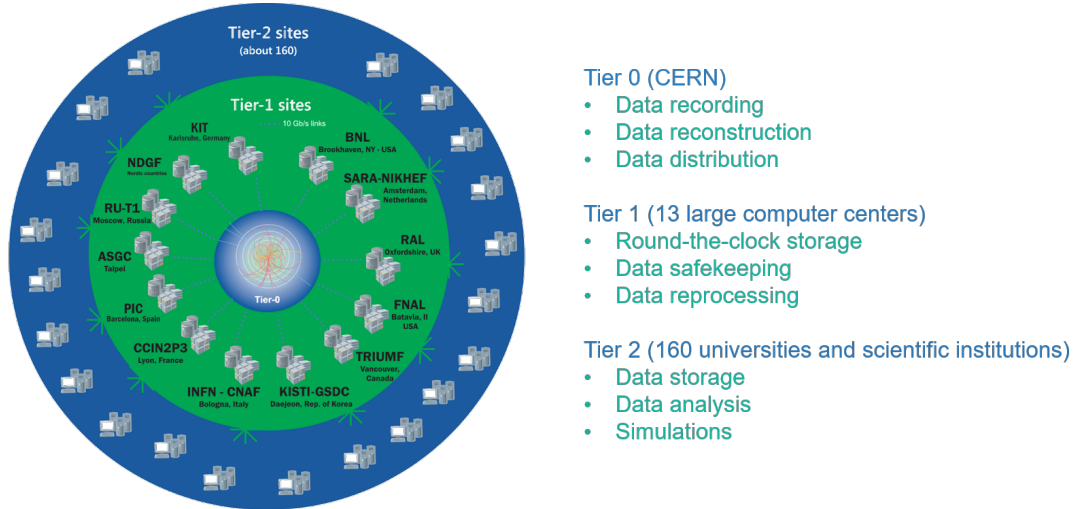


Figure 2.10: The Worldwide LHC Computing Grid layers, denoted as “tiers.” Each tier provides a specific set of services [65].

2.3.1 Worldwide Large Hadron Collider Computing Grid

The Worldwide LHC Computing Grid (WLCG) represents the world’s largest computing infrastructure available for HEP [64]. It provides computing resources to store, distribute and analyze the data generated by the LHC experiments among CERN member states all around the world. THE WLCG is coordinated by CERN and managed between the experiments (ALICE, ATLAS, CMS and LHCb). However, only around 20% of the computing capacity is located directly at CERN. The rest is distributed to 170 computing centers, distributed globally between 42 countries. In total, the WLCG comprises more than one million computer cores with more than two exabyte of storage capacity [65].

The WLCG is constructed by four layers denoted as “tiers”; tier 0 to 3. Each tier is responsible for a specific set of services [65]. Tier 0 represents the data center located at CERN, through which all the recorded data passes. Tier 1 consists of thirteen large and constantly available computer centers for data safekeeping. Tier 2 consists of scientific research institutes that store and analyze the data. Tier 3 represents independent institutes or scientists with no formal engagement to WLCG. The WLCG is summarized in figure 2.10. More comprehensive information about the WLCG is provided in Ref. [66]. The creation of the WLCG at the time of its invention, the largest computing infrastructure worldwide, underlines the importance of computing in HEP and the highly innovative developments to satisfy the needs with novel technology and new approaches.

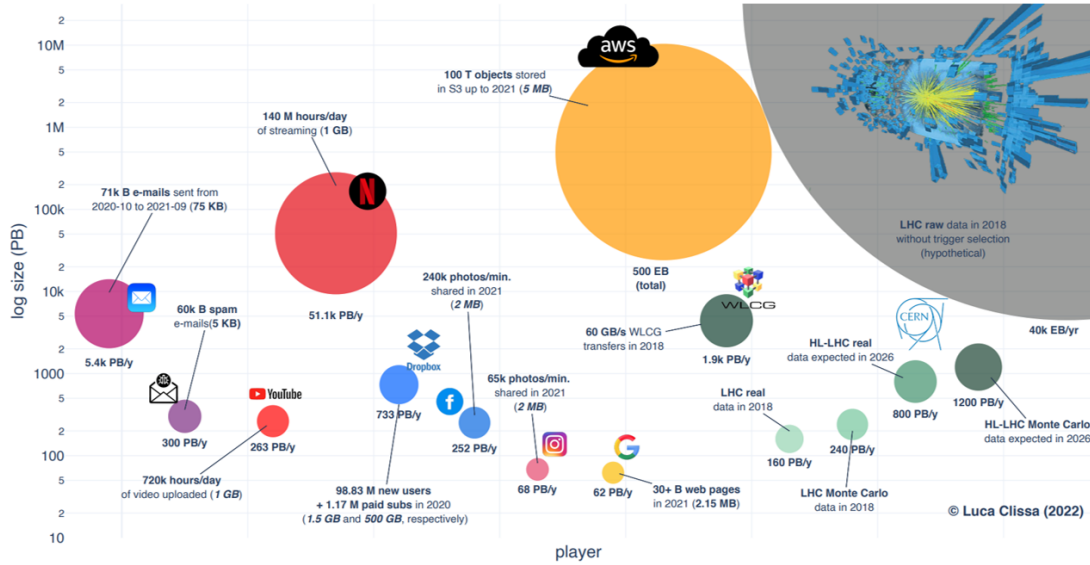


Figure 2.11: The generated amount of data from the LHC compared to other big data fields in 2018 as a bubble plot. The figure is published in Ref. [67].

2.3.2 Future High Energy Physics Computing Constraints

Taking as many unique data by measuring the final states of the particles produced in the LHC collisions as precise as possible by the experiments by employing a vast amount of accurate simulations for data correction, is the indispensable ingredient to accomplish the primary goal of novel physics results and insights. In Ref. [67], the generated amount of data from the LHC is studied and compared to other big data producers. The results are summarized as a bubble plot in figure 2.11. The LHC in Run 2 is by far the largest data producer indicated by the largest bubble in the top right corner with about 40k exabytes (EB) of raw data generated in 2018. Over one billion collisions per second were produced in Run 2, what corresponds to approximately one petabyte of generated data per second [46]. Nevertheless, with current computing systems it is unfeasible to record these enormous amounts of data and, furthermore, not all the produced information is required. Therefore, the raw data is filtered to collect only the most significant events, according to predefined metrics. The hardware and software triggers take over the filtering of the recorded data and shrink the amount of stored data down to about one petabyte (PB) per day [67, 68], what is indicated in figure 2.11 as “LHC real.” As a future outlook, Ref. [67] predicts that the amount of generated data by the HL-LHC increases approximately by a factor of four, as indicated in figure 2.11.

For comparison, the largest big data producers, according to the study in Ref. [67], are Amazon, Netflix and e-mail exchange. Further, in figure 2.11, the data transfer of

the WLCG is listed, which is merely three times below the global email exchange. The WLCG transfer is larger than the global YouTube video stream, the Dropbox uploads and the photos shared on Facebook and Instagram together. It should be noted that the bubbles in figure 2.11 for the services of the private companies are only approximations, because the official values are not published. A similar study about big data sources was already carried out in 2013 in Ref. [69]. The results from the study in 2013 agree with the one in 2018. Both the generated data by the LHC and the generated data by other big data producers were constantly growing between 2013 and 2018.

The message of this comparison is that computing represents a crucial key component of HEP experiments. The enormous amount of data, together with the high data transfer rates of the WLCG, clearly demonstrate that HEP research operates at the frontier in the field of data processing and high performance computing. In addition, the fact that HEP is taking over a leading role in pushing the boundaries of computing is proven by the invention of the World Wide Web at CERN to enable extremely efficient sharing of experimental data [70].

Data collection represents not the only exclusive computational demanding task at HEP experiments. The comparison of experimental results to a vast number of accurate simulations is a complex process, which produces an equally intense demand for computational hardware resources. Simulations require at present more than half of the WLCG hardware resources [47]. In figure 2.11, the amount of simulated data is illustrated. The simulations are denoted as "Monte Carlo" because detector simulations rely today mainly on Monte Carlo-based methods. In Run 2, the amount of simulated data was comparable to the YouTube video stream. In the future HL-LHC phase, the amount of simulated data is expected to rise by a factor of five, indicated by the "HL-LHC Monte Carlo" bubble. The reason for this will be presented in the following.

The estimated luminosity and integrated luminosity of the LHC during its operational lifetime are displayed in figure 2.4 at the bottom. Integrated luminosity is the integral, or the sum, of the delivered luminosity over time and corresponds to the total collected data size. It can be noted that the luminosity (denoted as nominal Lumi) increases strongly in the HL-LHC phase, from Run 4 on-wards. This is directly correlated with an increasing amount of required detector simulations.

In the roadmap for HEP computing in Ref. [47] in 2020, predictions about the estimated CPU time and the required disk space are provided. The predictions utilize the current computing models for the HL-LHC phase and are summarized in figure 2.12. The Monte Carlo simulations (labeled as HL-LHC MC) are predicted to take a significant fraction of the CPU time (more than one-third). Furthermore, the plots in figure 2.12 clearly indicate that the expectations for CPU time and disc space increase drastically in 2026 compared to the previous years. The year 2026 corresponds in the

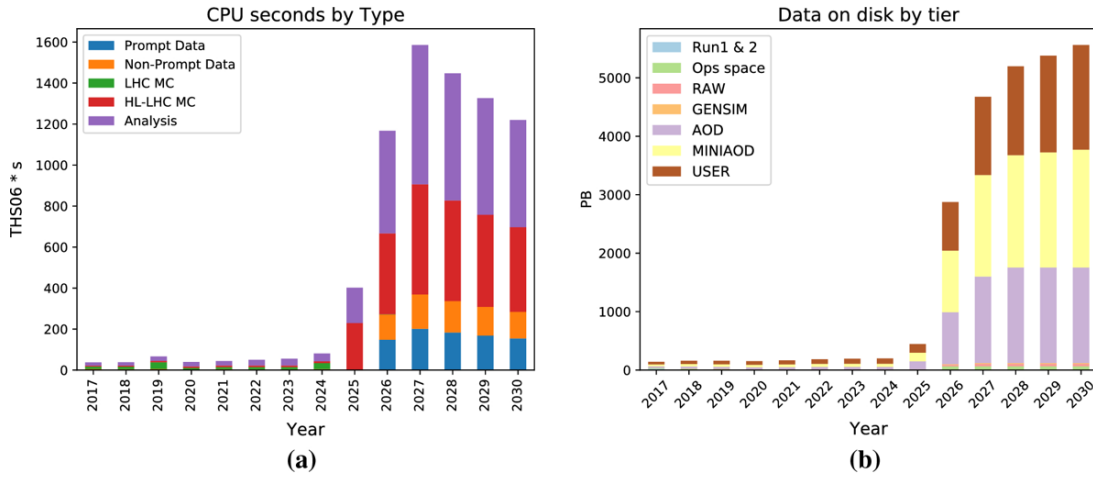


Figure 2.12: Estimated CPU time (left) and disk space (right) for CMS in the HL-LHC phase with employing the current computing models. The figure is taken from Ref. [47] and comprises the status of 2020.

plots when the HL-LHC was supposed to launch at the time the study was carried out. As indicated before, the HL-LHC start is shifted forward to 2029, but the message of the plot remains the same.

The authors of Ref. [71] assessed in 2021 the predicted CPU requirements for CMS as a representative for LHC HEP experiments. Figure 2.13 provides the in 2022 updated version from Ref. [72] for the years 2021 to 2038. Two different software development scenarios and the sustained budget model are provided. The sustained budget model is visualized for a 10% to 20% increase of computational hardware per year to represent the predicted available hardware resources. The first baseline scenario, with no research and development improvements in simulation software, shoots up after 2029, when the HL-LHC is expected to start. The second scenario incorporates probable research advances. Only with an annual hardware increase of 20% the second scenario lies within the available hardware band. This underlines the significance of improved computational models for the HL-LHC phase. For simulations, no research means that the simulations are carried out with the current Monte Carlo methods. For the second scenario, faster simulation approaches probably replace the Monte Carlo methods. Similar results were assessed in a survey for the ATLAS experiment in Ref. [11].

Furthermore, in the HL-LHC phase, an increased simulation precision will be required. The precision of Higgs measurements will be dominated by theory errors within the simulation models [73]. To overcome this, theoretical events generated at the next-to-leading-order (NLO) must be applied more and more extensively. However, the NLO events considerably increase the simulation time and require an ad-

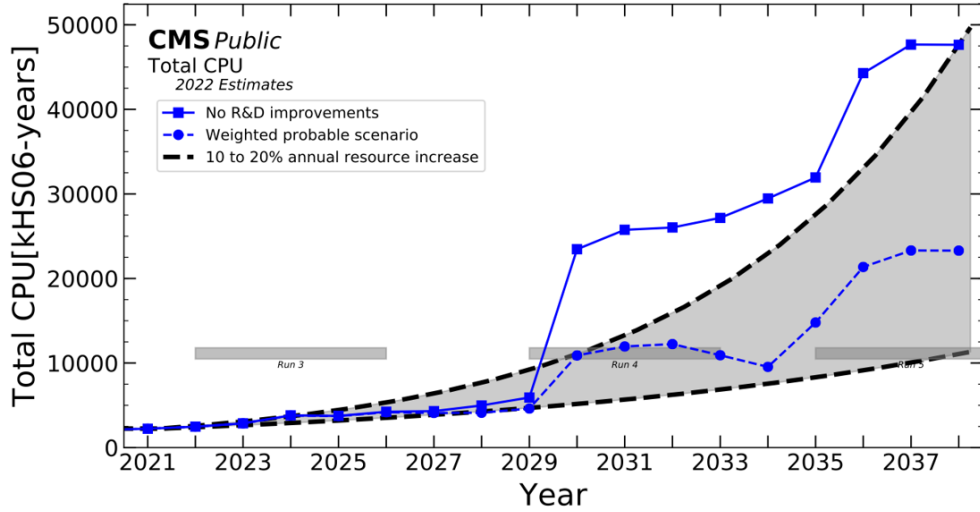


Figure 2.13: Projected CPU requirements of CMS between 2021 and 2038 based on an assessment in 2022. The gray band represents the projected hardware resource capacity. The blue lines correspond to the two considered scenarios. The figure is taken from Ref. [72].

ditional amount of computing resources. In the far future, the simulation precision is predicted to become increasingly important and even next-to-next-to-leading-order (NNLO) corrections have to be taken into account [74]. On top of a higher collision rate, due to the higher luminosity, future experiments will require more accurate detectors, which even further increase the geometry and simulation complexity. For example, calorimeter detectors will be built with a higher granularity with more sensors, what leads to a higher data collection rate and a more excessive demand of simulation time for collision reconstructions [7]. Therefore, researching for better and faster computing algorithms remains a crucial task to minimize the required hardware resources as much as possible.

To conclude, the LHC experiments produce an enormous amount of data comparable to renowned private companies. For future LHC runs, the increasing number of collisions leads to more produced data, which has to be processed. Because of this, the number of to be simulated detector events will increase. Additionally, the simulation complexity rises, resulting in longer simulation times and higher hardware resource requirements. Taking all these factors into account, future simulations of the LHC, especially in the high luminosity phase, are expected to require much more hardware resources, assuming current software and analysis techniques. This exceeds the ex-

pected computing resources, even with taking the computing hardware technological developments into account [49].

2.3.3 Current Monte Carlo Simulation Methods

Detector simulations rely today on Monte Carlo-based methods and are employing the Geant4 (GEometry ANd Tracking version 4) toolkit [9]. Geant4 simulates the passage of particles through matter and is applied in a number of research projects around the world, including electromagnetic calorimeter simulations, which are studied in this thesis. At the LHC, Geant4-based simulations are currently determined as one key factor in delivering physics results with an outstanding quality and are among others applied in ATLAS, CMS and LHCb [7]. Simulations in HEP are designed to generate identical events to those from the actual experiment. Geant4 does not only propagate particles through a geometric representation of a detector. Additionally, it contains a comprehensive library of physics models to simulate interactions of particles with matter and modelling the passage of particles by applying physical laws and interaction cross-sections [7].

In the past and still today, many simulation approaches do employ Monte Carlo simulation methods to estimate the signals left from the final state particles produced in the beam collisions [12]. Equally, Geant4 utilizes in a considerable fraction Monte Carlo simulation methods [9]. In Monte Carlo simulations, samples are drawn from a probability distribution to provide approximations to sums and integrals difficult to solve classically. For example, some integrals require an exponential number of terms to be exactly determined, but they maintain no exact simplifications [12]. These kinds of problems can be solved more easily by employing Monte Carlo simulation methods.

Monte Carlo algorithms are based on the law of large numbers and do not deliver exact results. Therefore, the responses contain some calculation errors, which can be reduced by calculating more samples. For an increasing number of independent and identical distributed samples n , the result $\hat{s}_n$ converges to the expected value s :

$$\lim_{n \rightarrow \infty} \hat{s}_n = s. \quad (2.10)$$

One remarkable example of Monte Carlo simulation methods is given by the approximation of the circle number π . It is represented by the area A of a circle to the square of the radius r :

$$\pi = \frac{A}{r^2}. \quad (2.11)$$

For determining π with Monte Carlo simulations, random numbers are sampled within the interval of $x, y \in [-1, 1]$. For visualization, a red square is drawn in figure 2.14 containing the random samples. Next, a blue circle with radius one is plotted.

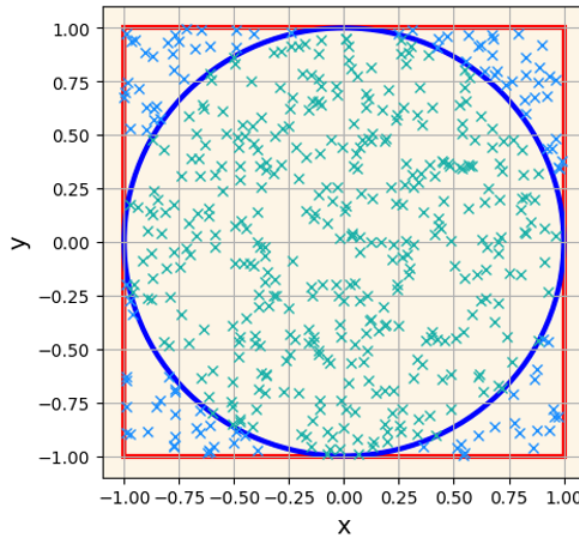


Figure 2.14: Determination of π with Monte Carlo methods.

It is measured how many random samples are placed within the circle (green data points) and how many are outside the circle (blue data points). By computing the ratio between the green points inside the circle and the blue points outside the circle, $\pi/4$ can be determined. Based on the rule for large numbers, the variance of the result for estimating π decreases for an increasing number of samples. It applies: the larger the number of samples, the more accurate π can be determined.

For more complex calculations, the Monte Carlo simulations require a tremendous amount of computing resources for accomplishing precise results, as they are required for HEP experiments. Because of the aforementioned computing constraints for future HEP experiments, intense research is ongoing for seeking alternative software solutions than the standard Monte Carlo approach to combat the increasing demand of computing resources [75–78]. In parallel, work is done on parallelizing calculations of Geant4, multi-threading, improved instruction pipe-lining, data locality and vectorization of single instructions with multiple data for accelerating the current Monte Carlo simulations [7]. However, Monte Carlo simulations do not scale, and parallelization remains difficult and, to some extent, impossible. Thus, maximal a speed-up of a factor of five can be expected [7]. This represents a noticeable improvement, but it is insufficient for facing future simulation demands. Therefore, Monte Carlo simulations do not meet the requirements for the higher amount of simulated data in the future HL-LHC phase [12]. To keep-up with the increasing simulation complexity and number of simulated events, in the future, the experiments need to rely more and more on fast and novel simulation methods.

2.3.4 Alternative Fast Simulation Approaches

The detailed simulation of particle reactions with Geant4 to a high accuracy is called the “full simulation” approach. In the full Geant4 simulations, the passage of particles through the detector proceeds in multiple stages. At first, the initial interaction is simulated by event generators. Next, the secondary particles are tracked through the detector. Lastly, the simulation includes the moment when the particle hits a detector sensor, and the information is read out by a signal [7]. However, modeling all physical interactions of each particle remains computing resource expensive and can take for LHC detector simulations on modern computers up to a few minutes [7].

Because of their dense material and the highly granular geometry, calorimeters are among the detectors that are computationally most demanding and require the most substantial fraction of simulation time [47]. In the calorimeter shower process, an enormous number of secondary particles is involved. For each of them, all the underlying physics interactions with the detector material are precisely computed. According to their probability, this leads to lengthy simulations. Because of the sequential creation of the secondary particles, the response of the previous interaction is required to determine the following one. Therefore, parallelization remains difficult. In order to master the future computing challenge of calorimeter simulations, as a first step, this research will focus on alternative and faster approaches for replacing the current calorimeter Monte Carlo simulations. This offers the potential to accomplish a significant reduction in required hardware resources.

Fast simulation methods are implemented to reduce the simulation time, while aiming at maintaining the accuracy for defined key components. In Ref. [79], different current fast calorimeter shower simulation approaches are introduced, depending on the energy range of the incident particle and the required level of accuracy. One elementary example represents the parametrization method for frozen shower images. The idea is that for the longitudinal and radial distributions, it is sufficient to describe the spatial energy distribution of electromagnetic showers inside the calorimeter. However, this oversimplified approach reduces the accuracy for most physics applications. A general difficulty in searching for alternative simulation approaches is the difficult compromise of sacrificing some level of accuracy in order to gain computational speed.

Today, physicists are working on Machine Learning and Deep Learning (DL) techniques for replacing distinct parts of the current Monte Carlo Geant4 simulations. Several prototypes based on DL models have demonstrated immense potential. They achieve a significant speed-up, while maintaining almost a similar level of accuracy compared to traditional Monte Carlo simulations [63, 80–83]. The first alternative simulation approach studied and further developed in this thesis is building on DL methods. Therefore, the following chapter is dedicated to a detailed introduction to DL

models and how they can be employed for HEP simulations.

2.4 3D Calorimeter Shower Images

This section describes the data set employed throughout the entire thesis. The data set contains electromagnetic calorimeter shower images with electrons as primary particles. It is generated by Geant4 simulations for a future representative high granularity calorimeter for the Compact Linear Collider (CLIC) [84]. The original full size data set contains three-dimensional electron shower images with $51 \times 51 \times 25$ pixels (x -, y - and z -axes) and is published in Ref. [85]. For generative tasks, a reduced image size of $25 \times 25 \times 25$ pixels is employed, which is equally published in Ref. [85]. The reduced data set used in this thesis consists of 200 000 images and is publicly available at Ref. [86]. The primary particle energies E_p are uniform distributed within the range of 10-510 GeV. The distribution of the primary particle energies is demonstrated in figure 2.15.

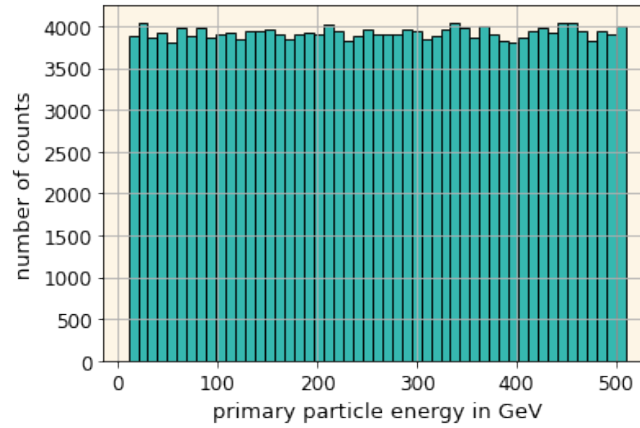


Figure 2.15: The uniform distribution of the primary particle energies E_p within the 10-510 GeV range plotted as a histogram with 50 bins for 200 000 images.

The calorimeter consists of 25 layers in depth, which makes it convenient to choose the image dimension 25 on the image z -axis. The z -axis is parallel to the incident primary particle. The other two image dimensions are chosen in order to contain almost the full shower development around the shower barycenter and are set to 25, likewise. Figure 2.16 visualizes a three-dimensional example shower image, where the particle enters the calorimeter with a momentum perpendicular to the x - y -plane at position $x = 13$, $y = 13$ and $z = 0$. The shower is evolving in the z -direction, the longitudinal direction, or the depth of the calorimeter. A cutout of the central layer ($y = 13$) of the

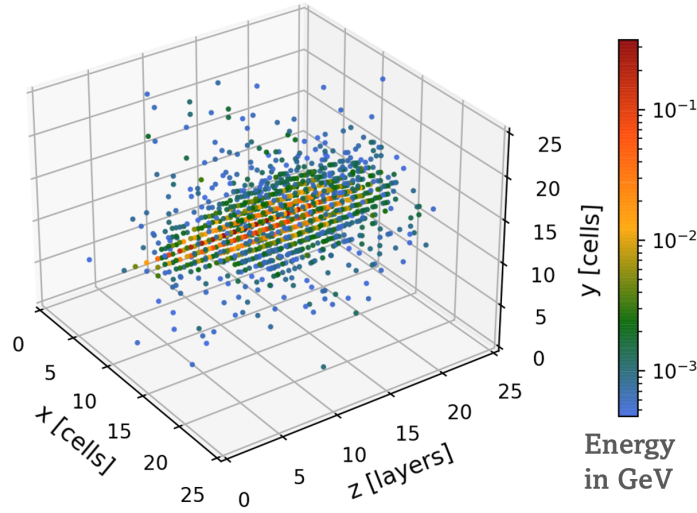


Figure 2.16: A 3D example shower image created by a primary particle with the energy $E_p = 500$ GeV. For visual reasons, an arbitrary chosen $4 \cdot 10^{-4}$ GeV energy threshold is applied to suppress low energy pixels.

same shower is illustrated in figure 2.17. Multiple scattering of electrons and positrons apart from the shower axes equally create a lateral shower development. Another contribution to the shower lateral spread represent Bremsstrahlung-photons emitted by electrons and positrons. The transverse shower spread can be quantified by the Moliere radius [57]. The two shower images demonstrate how typically only a fraction of below 20% of the calorimeter cells show some energy deposition. Moreover, the energy range of the single pixels can span over more than six orders of magnitude, from 10^{-6} GeV to 1 GeV.

It is worth noticing, there is no timing information present in the shower images: only static images of the energy deposits are employed. Additionally, there are no pile-up effects from other particle showers included. Each image contains the shower exclusively created by one primary particle. Further, all primary particles are entering the calorimeter perpendicular without any incident angle. Due to the limited size of calorimeter detectors, the full particle shower may not be contained in the calorimeter, what leads to a shower leakage [87]. This can undoubtedly be perceived in figure 2.18 in the z -axis shower shape. At the end of the shower, in the last calorimeter cell (pixel 24), the energy deposition does not drop to zero. Therefore, some energy component of the particle shower leaves the calorimeter undetected, what influences the detector energy resolution. These effects occupy a more critical role in the longitudinal shower

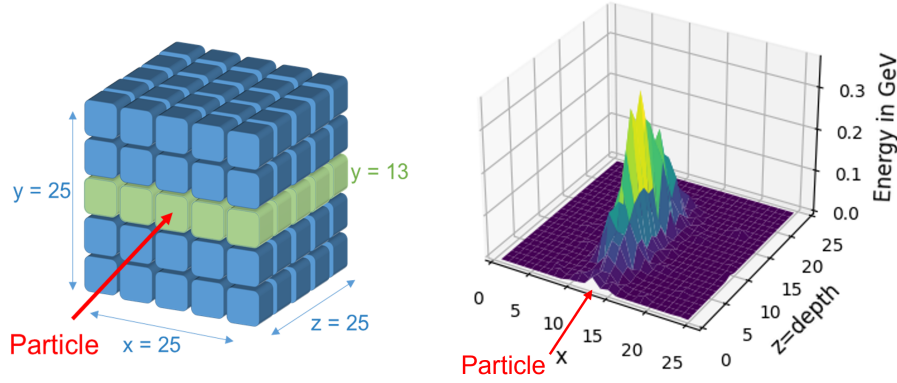


Figure 2.17: (left): A simplified representation of the three-dimensional image. (right): An example shower development at $y = 13$ for an $E_p = 500$ GeV event.

direction (along z -axis) compared to the lateral shower direction (along x - and y -axis).

It remains challenging to visualize and compare three-dimensional shower images to each other. It is more straightforward to exploit two-dimensional projections along the different canonical axes – x , y and z respectively – to visualize the characteristics of the showers. These plots are called shower shape plots and are visualized in figure 2.18. Figure 2.18 demonstrates that the shower shapes behave similar in x - and y -dimension. The physics of the shower is rotationally invariant with respect to the shower axis, and the CLIC calorimeter is isotropic in x and y . In the x - and y -axis histograms, most of the energy is deposited within the middle pixels 10-14. The z -axis plot along the longitudinal axis maintains a different shape. The primary particle enters from the left and the released energy is increasing until it reaches the shower peak and then decreases steadily. At the shower peak, the highest number of secondary particles exists simultaneously. The plots in figure 2.18 represent the shower shapes for different primary particle energies E_p within an energy range of ± 3 GeV. In the z -axis shower shape plot, the standard deviations of the corresponding energy curves are provided as colored bands around the mean. Even for specific primary particle energies, the shower profile slightly varies and does not consistently create exactly the same shower profile.

In figure 2.19, the same shower shapes are plotted on a logarithmic energy scale on the y -axis. The plots with the logarithmic scale have the advantage that the low energy pixels at the tails of the shower distributions can be investigated in more detail. The shower shape plots remain one of the key metrics to accurately evaluate the results of the different simulation methods. In this thesis, they will be employed multiple times to evaluate and compare the generated images to the current Geant4 simulation. Both the linear and logarithmic scale are utilized. Because the x - and y -axis shower shapes

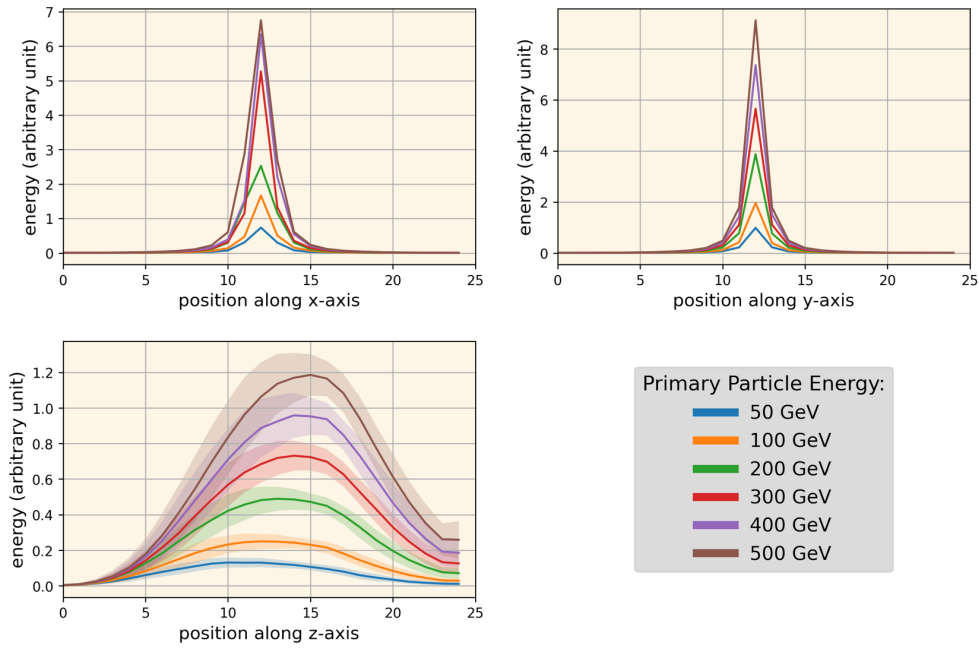


Figure 2.18: Example shower shape plots along the x -, y - and z -axis for different primary particle energies. The colored band in the z -axis plot indicates one standard deviation.

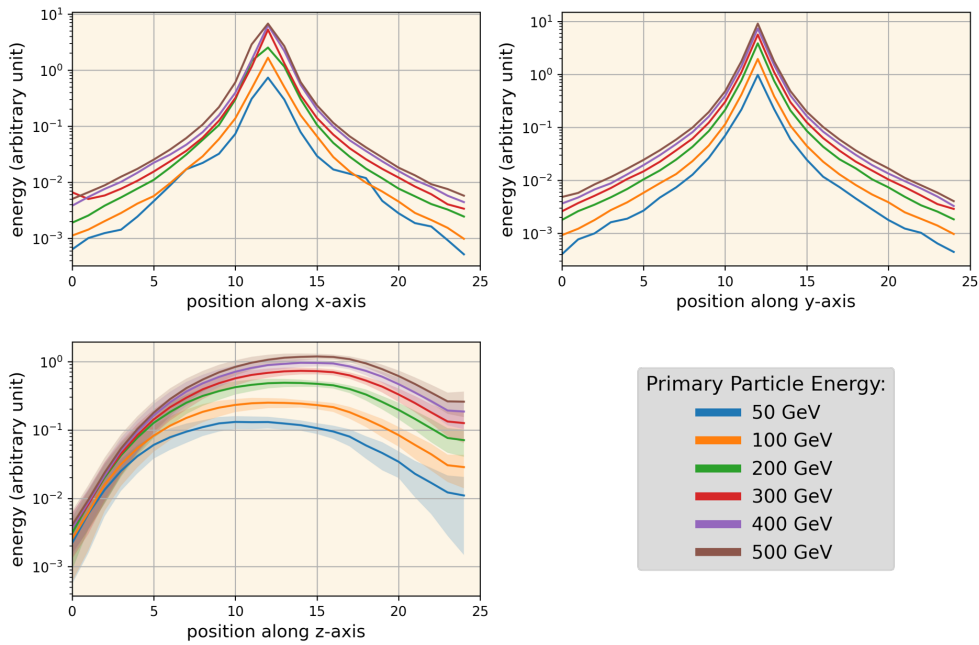


Figure 2.19: Example shower shape plots in a logarithmic scale.

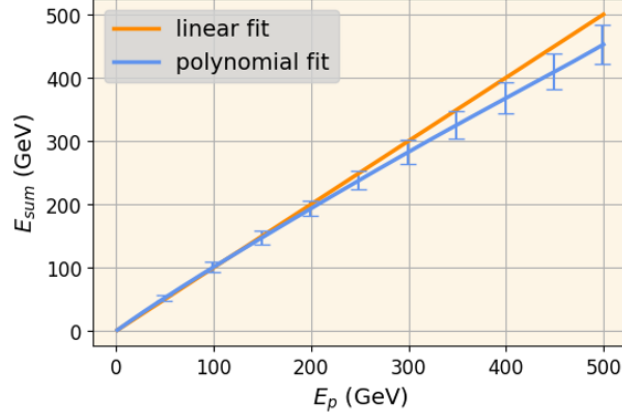


Figure 2.20: The polynomial and linear fit between the primary particle energy E_p and the measured energy sum E_{sum} within the detector of the operated calorimeter training data. The error bars at the polynomial fit visualize the E_{sum} deviation of the training data within ± 5 GeV bins on the primary particle energy E_p . The linear fit is plotted as reference.

behave similarly, the x -axis plot is in the following neglected and only the y - and z -axis shower shapes are provided.

The calorimeter shower images are a function of the energy E_p of the incoming electron. In addition, the energy sum over all calorimeter cells, E_{sum} , is used. The model that will be described later is conditioned on these two variables. For a sampling calorimeter – such as the CLIC ECAL – and before any calibration, E_{sum} is typically smaller than E_p . The relation between E_{sum} and E_p depends on the type, geometry and material of the calorimeter. For the data operated in this thesis, it is empirically approximated by the polynomial function:

$$E_{sum} = 0.0018 \cdot \left(\frac{E_p}{500}\right)^5 - 0.023 \cdot \left(\frac{E_p}{500}\right)^4 + 0.011 \cdot \left(\frac{E_p}{500}\right)^3 - 0.28 \cdot \left(\frac{E_p}{500}\right)^2 + 2.21 \cdot \left(\frac{E_p}{500}\right). \quad (2.12)$$

The parameters of the polynomial function were found by fitting the training data. The polynomial function is plotted in figure 2.20. At higher primary particle energies E_p , the values E_p and E_{sum} visually diverge. The measured energy E_{sum} lies below the linear fit. The primary contribution to this is most probably the result of the shower leakage effect. Because, the larger the primary particle energy, the longer the shower development and, therefore, a larger fraction of energy leaves the calorimeter. As the calorimeter size is limited, the energy absorption is at some point cut off and not all energy is measured.

Chapter 3

Deep Learning for Future High Energy Physics Simulations

Deep Learning (DL) is a well-established computing technique that is already successfully employed in simulations in almost every industry and research branch [88]. Due to their success, DL techniques are today often among the first choices to improve and accelerate simulations. In this thesis, DL is applied and tested to accelerate HEP detector simulations. Calorimeter simulations are the most computing resource intensive part of the detector simulations. Therefore, more and more computing resource efficient generative DL models are studied and developed to generate calorimeter shower images and to replace the current Monte Carlo calorimeter simulations.

This chapter comprises the principles behind DL. Additionally, the concept of generative models is explained, and existing models are listed. Next, the mathematical principle of convolutional layers and their successful operation in neural networks is provided. Lastly, some supplementary required neural network layers used in this thesis are introduced.

3.1 Introduction to Deep Learning

DL constitutes a specific area of the general topic of Artificial Intelligence (AI). AI summarizes all simulation approaches which employ algorithms to mimic the intelligence of humans and can make decisions on their own. AI represents a growing field, which already has an extensive impact on our ordinary life. Some striking examples are: digital voice assistants, chat bots, music and Google search recommendations, image

object detections and the first autonomous driving cars.

In general, the term intelligence and AI are not strictly defined. Classical AI comprises primarily conventional hard-coded algorithms [89]. Naive classical AI algorithms can already start with the employment of if-conditions: if the input value is larger than zero, classify to class α , if it is lower or equal to zero, classify to class β . After programming this condition, no human is involved to assist the computer in making the decision. Consequently, this can be taken as an AI example. Furthermore, beyond classical AI, the area of AI contains more sophisticated algorithms introduced in the following.

Machine Learning (ML) stands for a subgroup of AI, which focuses on self-learning algorithms. ML algorithms have the ability to acquire knowledge by extracting meaningful patterns or information from raw data [88]. It enables computers to address many real-world problems, which remain difficult or even unsolvable by hard-coding algorithms [89]. In ML, the programmer does not provide exact codes of what the algorithm should do in which case. Instead, it focuses on creating an environment and conditions in which an algorithm can train itself to perform a task. ML training is based on training data and statistical models. Some ML examples represent K-means, K-Nearest Neighbor, Naive Bayes classifier, Decision Tree, Random Forest, Expectation–Maximization and Support Vector Machines [90].

Figure 3.1 illustrates the trend of the Google search index for the entire world over the past years. It can be observed, approximately in 2016, the public interest in ML overtook the interest in standard AI algorithms. Today, ML is twice as often searched than AI. This indicates that ML became more and more important during the past decade.

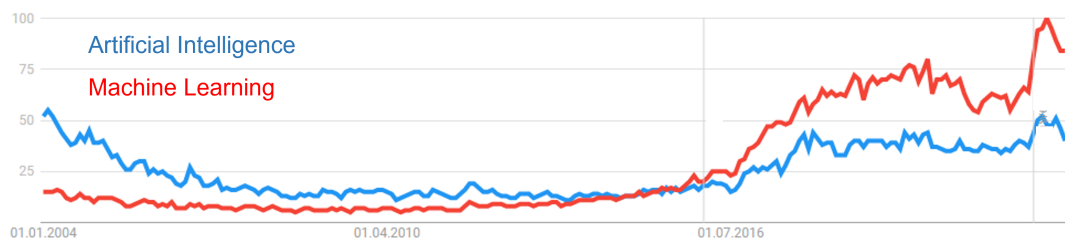


Figure 3.1: Number of Google searches of AI and ML during the last years [91]. The y-axis is scaled, 100 corresponds to the maximum number of searches.

When it comes to complex and computationally demanding problems, DL represents the first choice [92]. DL itself is a subgroup of ML, as indicated in figure 3.2. DL algorithms are similar to ML methods trained on data. The difference is that DL models employ deep neural network architectures to hierarchically learn correlations present

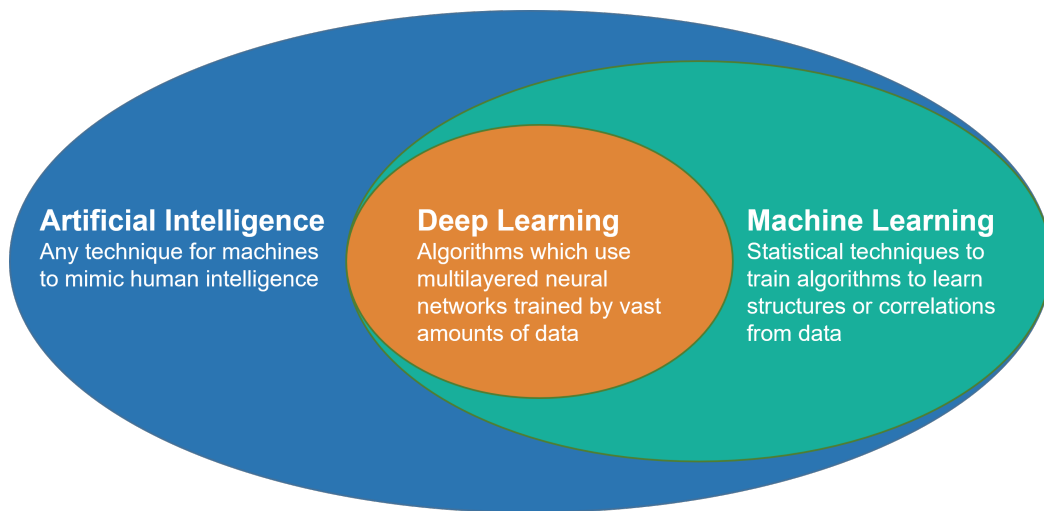


Figure 3.2: Illustration of the hierarchy of the computer science fields Artificial Intelligence, Machine Learning and Deep Learning.

in the training data. Neural networks attempt to mimic the functionality of the human brain [93]. In fact, some principal constituents of neural networks have even been copied from brain research. Neural networks are denoted to be “deep”, when multiple hidden layers are operated. Additionally, the larger the amount of available data and, the more complex the data, the larger the benefit for DL techniques compared to ML [88].

DL is an old and long known technique with a rich history [94]. However, it became only in the last years more and more popular and applied in practice. The primary reasons for the rise of interest in DL during the past years are the technology improvements (e.g. parallel processing on Graphical Processing Units (GPUs)) and the increasing sizes of available data sets for training the DL models. The hardware to carry out the computations became faster, which allows the execution of the learning processes within short time periods. Short calculation times are fundamental for effectively optimizing the algorithms. This leads to a chain reaction: better technology supported by the change of digitalization allows recording and storing more data with a higher efficiency on larger hard drives. To achieve higher accuracies, the neural network models have grown in size and complexity in the past years [95]. This is only possible by improvements of the computer infrastructure – both software and hardware – for DL approaches [12].

The performance of DL algorithms depends strongly on the amount and quality of the available training data [12]. Therefore, DL is part of the computer-science field of “big data.” The fundamental constituents behind DL simulations are deep neural networks applied to learn correlations between features within the training data. Features rep-

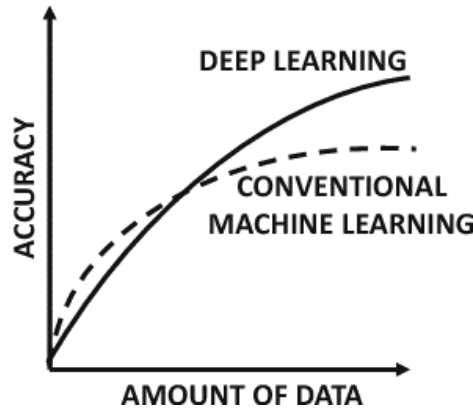


Figure 3.3: Comparison of the achieved accuracy between Deep Learning and Machine Learning algorithms [96]. For a sufficient amount of data, Deep Learning outperforms traditional Machine Learning.

represent the sole pieces of information present in the data. The ability of a network to forecast unseen data, i.e. data not included in the training set, is denoted as generalization. An associated phenomenon is overfitting: on a second, independent test data set, the performance typically is lower than on the training data, and this performance gap often increases during training. In contrast, traditional ML models can successfully operate with small data sets. However, for more available and more complex data, DL models outperform such ML models. Summarized, for few available data with limited complexity, conventional ML is the better choice. For high dimensional data sets with many features, DL performs better. This is indicated by figure 3.3. As the utilized calorimeter simulations in this thesis represent highly complex use cases with a large amount of available training data, DL models are employed.

Applications related to images and visual data, belong to the category of computer vision [90]. Computer vision tasks are effortless for humans but challenging for computers and even impossible to solve with explicit hard-coded algorithms. By employing trainable models, DL resolves this issue.

In the DL training, the parameters of the model are optimized by minimizing a cost function to a local or global minimum. Within the forward path, the parameters are stored to be available for the parameter optimization in backpropagation. During backpropagation, all the parameter gradients are calculated while going through the model in the backward direction. With gradient descent, the parameters of the neural networks are updated in the direction of the steepest descent of the loss function [97]. The learning rate adjusts the size of the optimization steps. The learning rate has to be chosen carefully, a too large learning rate overshoots the minimum, while a too small learning rate results in a slow learning process. Furthermore, too low learning rates can cause the algorithm to get stuck in a local minimum in the gradient landscape

instead of the desired global minimum. This parameter updating process is repeated multiple times until training convergence is reached. The model is said to have converged, if further training does not cause a loss or accuracy improvement.

ML tasks are divided into supervised and unsupervised learning. Supervised learning is aimed at learning the relations between data samples and their labels, e.g. created by humans. On the other side, unsupervised learning focuses on bringing patterns or structures of unlabeled data samples to the surface. Therefore, it supports humans in understanding and interpreting data. Unsupervised learning techniques sample from high-dimensional probability distributions present in the training data, which are often not obvious to humans [12]. Generative models represent one class of unsupervised learning techniques [98]. They are employed in this thesis and are introduced in the following.

3.2 Generative Models: An Overview

Generative models learn the probabilistic distribution of a given data set to generate new data points close to the existing data distribution [99]. They belong to the class of unsupervised learning, because the models discover and learn patterns present in the training data. Generative models demonstrate remarkable results in computer vision tasks, natural language processing and robotics [98]. For image generation, many smart and innovative approaches have been suggested. The most established and successful ones represent Variational Autoencoders (VAEs), Normalizing flows and Generative Adversarial Networks (GANs) [100]. Each of them is in the following briefly introduced, including their primary advantages and disadvantages. Recently, diffusion models [101, 102] appear to be very competitive and are added to the list of generative models. However, they are not covered in this thesis. There exists more types of generative models, for example Gaussian Mixture Models, Hidden Markov Models, Restricted Boltzmann Machines, Deep Belief Networks and Deep Boltzmann Machines [99]. They all maintain advantages and disadvantages, which make them suitable for specific use cases. A good summary including theoretic details is for example provided in the book in Ref. [12] or in Ref. [103].

Normalizing Flows

Normalizing Flows are generative models employing the so-called change of variables formula [103]. With the change of variables formula, one distribution $p(x)$ is replaced by another distribution π :

$$p(x) = \pi(z) \left| \det \left(\frac{\partial f(z)}{\partial z} \right) \right|^{-1}, \quad (3.1)$$

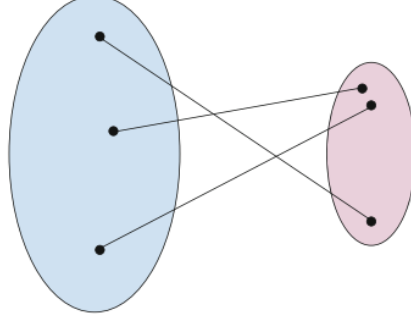


Figure 3.4: An example of a bijection. Points of the blue set are transferred to points of the purple set and vice versa. The figure is taken from Ref. [103].

where $z = f^{-1}(x)$, and f is an invertible function (bijection) [103]. The bijection f maps points x of the function p to other points z of the function π , as for example visualized in figure 3.4. It must be obeyed that the function f can be inverted to obtain the original points. The second term $|\det(\frac{\partial f(z)}{\partial z})|^{-1}$ is responsible for normalizing the distribution $\pi(z)$ as a form of regularization [103]. And $\det$ represents the matrix determinant.

In principle, most complex distributions can be converted into a simpler one. For this, in Normalizing Flows, multiple invertible transformations K can be sequentially applied [103]:

$$p(x) = \pi(z_0) \sum_{i=1}^K \left| \det \left(\frac{\partial f_i(z_{i-1})}{\partial z_{i-1}} \right) \right|^{-1}. \quad (3.2)$$

In figure 3.5, an example transformation of a Gaussian distribution into a multi-modal distribution is visualized. The logarithmic probability can be obtained by:

$$\ln p(x) = \ln \pi(z_0) \sum_{i=1}^K \ln \left| \det \left(\frac{\partial f_i(z_{i-1})}{\partial z_{i-1}} \right) \right|^{-1}, \quad (3.3)$$

where $\ln$ denotes the natural logarithm. In practice, neural networks are used to model the invertible transformations. For multiple sequences of transformations, multiple neural network layers K are employed.

Generating images with the trained Normalizing Flow model, frequently starts by sampling z from a Gaussian distribution (the "latent" space), which is manipulated sequentially by the neural network layers into the desired distribution $\pi(z)$ (pixel space), from left to right in figure 3.5.

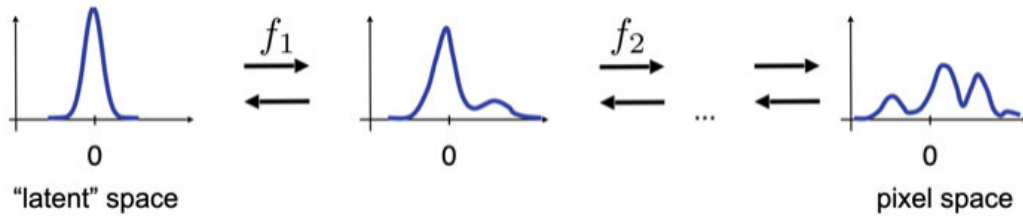


Figure 3.5: Transformation of a Gaussian distribution into a multi modal distribution by multiple invertible transformations f_i . The figure is taken from Ref. [103].

There are some constraints related to Normalizing Flows. All layers must be invertible, which is not the case for all neural network architectures [103]. Further, to calculate the determinants of large matrices, e.g. in the second term $\left| \det \left(\frac{\partial f_i(z_{i-1})}{\partial z_{i-1}} \right) \right|^{-1}$ of equation 3.3, is non-trivial and computationally demanding [104]. Therefore, additionally to invertible neural network architectures, the networks must be chosen to simply calculate the second term. This constraints the choice of layers [103, 104]. Advantages of the Normalizing Flows approach are that the model can generate precise images and that the training is stable [104]. More detailed information about the theory of Normalizing Flows is provided in Ref. [103] and about current enhanced Normalizing Flow models in Ref [105]. Some exemplary enhancements of the Normalizing Flow model are Spline Flows [106] or GLOW models [107].

Variational Autoencoders

First, the principle of Autoencoders (AEs) is introduced, which is then extended to Variational AEs (VAEs). AEs consist of an encoder E and a decoder D , typically constructed by neural networks. The encoder compresses the dimensionality of the input data x down to a latent space z , while still accumulating the significant information of the input data distribution. This is achieved by extracting features from x and creating new features z , which are in general more compressed. The decoder reconstructs the data close to the initial distribution [108]. The idea behind AEs is to learn the lower-dimensional representation of the training data, denoted as the latent variables, assuming that these have generated the actual training data. The true latent variables cannot be directly observed, but are inferred through a mathematical model, represented by neural networks. The latent variables preserve all the relevant information (features) about the output (image), which needs to be generated [109].

To train the AE, x is encoded into z . The latent space z represents a function of $e(x)$, where e denotes the operation of the encoder network. Then z is decoded back into $\hat{x}$, which is a function of the decoder network d : $\hat{x} = d(z)$. This process is visualized in figure 3.6. Ideally, when the model is trained: $\hat{x} = x$. Therefore, to train the

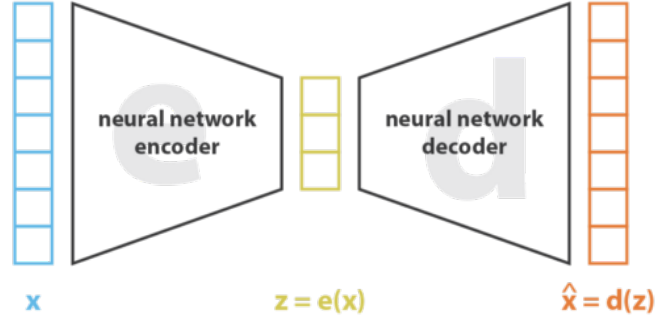


Figure 3.6: The principle of the encoding-decoding process of an Autoencoder. The figure is taken from Ref. [108].

autoencoder, the distance between $\hat{x}$ and x must be minimized by the loss l :

$$l = \|x - \hat{x}\|^2 = \|x - d(z)\|^2 = \|x - d(e(x))\|^2. \quad (3.4)$$

At this stage, the autoencoder can encode and decode the content. However, it is strongly overfitting and cannot yet generate new content with sampling from the latent space, because the latent space of single points is extremely irregular [110]. The representation of a regular and an irregular latent space is visualized in figure 3.7. Regularization leads to overlapping latent space distributions, enhancing the quality of decoding outputs. This overlap helps to prevent undefined regions in the latent space, ensuring the model consistently produces meaningful results [110]. Therefore, the training and sampling of VAEs require some regularization methods.



Figure 3.7: The representation of a regular (left) versus an irregular latent space (right). The figure is taken from Ref. [110].

For achieving regularization, instead of sampling from single points z in the latent space, the VAE samples from a distribution. With applying variational inference, the VAE searches for the best approximation to reconstruct the parametric distribution [103]. Frequent applied representations of the latent space are Gaussian distributions

[103], similar as in Normalizing Flows. The regularization should force the encoded normal distribution $N(\mu_x, \sigma_x)$ with the mean μ_x and the standard deviation σ_x to be as close as possible to the Gaussian normal distribution $N(0, 1)$. This is achieved by adding the Kullback-Leibler divergence (KL) [111] to the equation 3.4, calculated between the Gaussian normal distribution and the returned latent distribution. The modified training loss equation is:

$$l = ||x - d(z)||^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, 1)] . \quad (3.5)$$

The closer the two distributions in terms of μ and σ , the smaller the regularization term. The benefit of the encoding of a distribution is that the latent space is better organized allowing the generation of more meaningful results, where close points in the latent space are decoded into similar results. The principle of VAEs is visualized in figure 3.8. The more mathematical and probabilistic details are for example provided in Ref. [103] or Ref. [112].

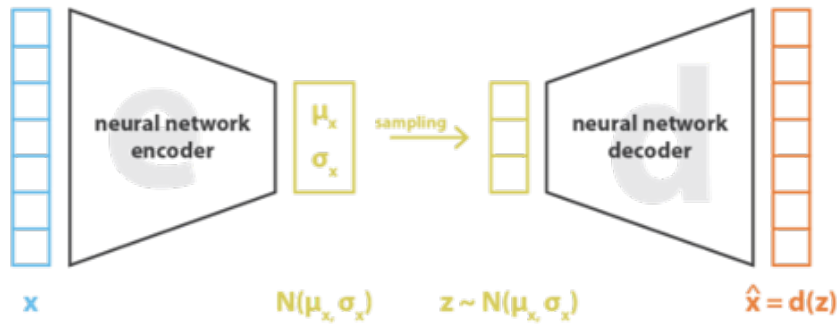


Figure 3.8: The principle of a Variational Autoencoder. The figure is taken from Ref. [108].

VAEs have the advantage that they learn both, the generative and the inference model. However, generated images from VAEs suffer from blurriness and look for humans generally less realistic compared to for example GAN images. A comprehensive survey between VAEs and GANs, including their advantages and disadvantages, is provided in Ref. [82].

Generative Adversarial Networks

GANs show an impressive performance in many applications. For example, GANs allow the realistic generation of images, music and art [99]. For this reason they represent, in general, the first choice for generative tasks [113]. Underlined by the statement of Yann LeCun in 2014, the research director of AI at Facebook [114]: "Generative Adversarial Networks is the most interesting idea in the last 10 years in Machine Learn-

ing.” In physics, GANs are investigated for replacing some parts of the traditional Monte Carlo detector simulations. It is already demonstrated that GANs achieve similar levels of accuracy as Monte Carlo simulations for specific applications, while considerably decreasing the simulation time by orders of magnitude, as demonstrated for example in the references [77, 78, 115–117] and many more. Because of the success of GANs in previous physics and non-physics researches, they are in this thesis further studied and evaluated and in detail introduced in chapter 4.

3.3 Convolutional Layers: 2D and 3D Operations

For image tasks, convolutional neural networks (CNNs) [118] often yield better results. These networks primarily use convolutional layers as trainable layers, which operate with much fewer parameters compared to fully connected layers [12]. To underline this, Ref [119] provides an example to calculate the hidden representation $\mathbf{H}$ of two-dimensional images $\mathbf{X}$, where $[\mathbf{X}]_{i,j}$ and $[\mathbf{H}]_{i,j}$ denote the pixel location (i, j) . The fully connected layer is represented by the weight tensor $\mathbf{W}$ and the biases $\mathbf{U}$. The operation of the fully connected layer can be expressed as:

$$\begin{aligned} [\mathbf{H}]_{i,j} &= [\mathbf{U}]_{i,j} + \sum_k \sum_l [\mathbf{W}]_{i,j,k,l} [\mathbf{X}]_{k,l} \\ &= [\mathbf{U}]_{i,j} + \sum_a \sum_b [\mathbf{V}]_{i,j,a,b} [\mathbf{X}]_{i+a,j+b}. \end{aligned} \quad (3.6)$$

In the equation, the subscripts k and l are set to $k = i + a$ and $l = j + b$, where a and b cover the entire image with positive and negative offsets. To remove k and l from the equations, $[\mathbf{W}]_{i,j,i+a,j+b}$ is set to $[\mathbf{V}]_{i,j,a,b}$. With equation 3.6, operating a single fully connected layer for a one-megapixel image (dimension 1 000x1 000) would require 10^{12} trainable parameters [119]. This number is unfeasible, and the utilization of multiple layers to increase the network depth would make the number even larger. To combat this issue, for DL image tasks, the more efficient CNNs are operated. This section explains the mathematical principle of convolutional layers and describes the differences between convolutional 2D (Conv2D) and Convolutional (Conv3D) layers. This introduction is essential to understand and recognize the key features of the neural network architectures developed for the following studies.

3.3.1 Convolutional 2D Layers

Figure 3.9 illustrates the operation principle of a Conv2D layer. The image is represented by a 2D input matrix on which a 2D convolutional kernel or filter is applied. As

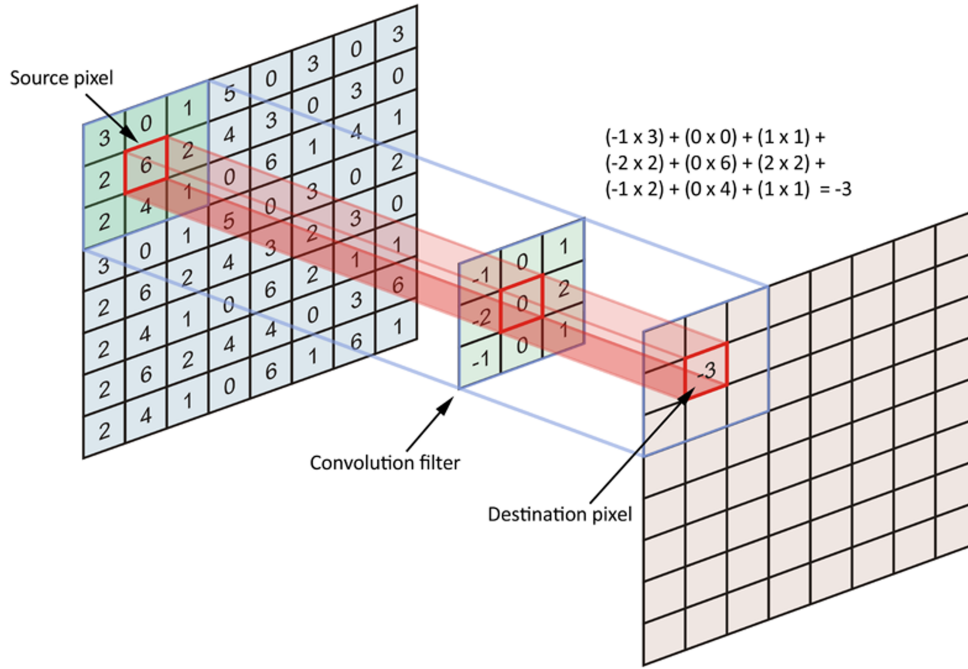


Figure 3.9: Schematically explanation of the working principle of a Conv2D layer [120]. The 2D input matrix is convoluted with a 2D filter to calculate the 2D output matrix. The matrix entries are randomly chosen. The picture is taken from Ref. [120].

displayed, this computation leads to a 2D output volume. The convolution operation can be imagined as laying the filter above a section of the input matrix and calculating the sum of the pixel-wise multiplications of the elements from the input matrix and the filter matrix, which lie on top of each other. For clarification, an example calculation is provided in figure 3.9. The convolution operation results in one destination pixel in the output matrix. To calculate the other elements of the output matrix, the filter is pixel-wise shifted over the whole input image.

In DL, the parameters of the convolutional kernel matrix are learned from data. These kernel parameters are at the beginning of the neural network training randomly initialized and during the training optimized. To increase the power of the convolutional layer, not only one filter is applied to the input matrix. Preferably, multiple filters are applied, which result in multiple 2D output matrices. These 2D output matrices can then be stacked to a 3D matrix. Therefore, the output of Conv2D layers with multiple filters N are 3D matrices, where the third dimension is denoted as the channel dimension, as visually demonstrated in figure 3.10. Each channel contains one feature trained by a separate filter. The output of all channels is denoted as the feature map. The filter size determines the size of the learned features: the larger the filters, the more neighboring pixels are considered, which allow learning larger features in pixel

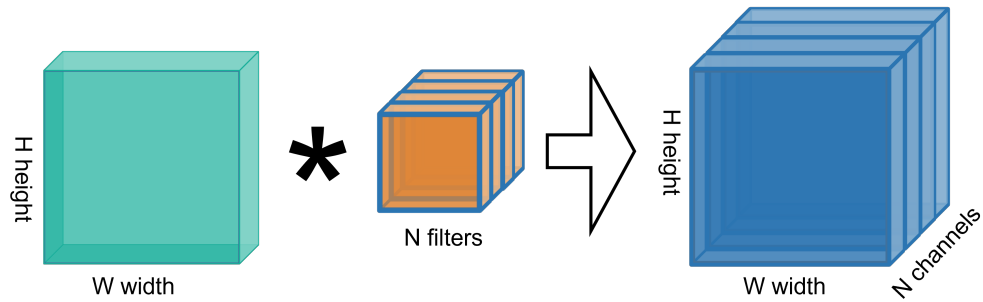


Figure 3.10: A Conv2D layer example with multiple filters: a 2D image is convoluted by N filters, resulting in a 3D output volume with the channel dimension N in the third axis.

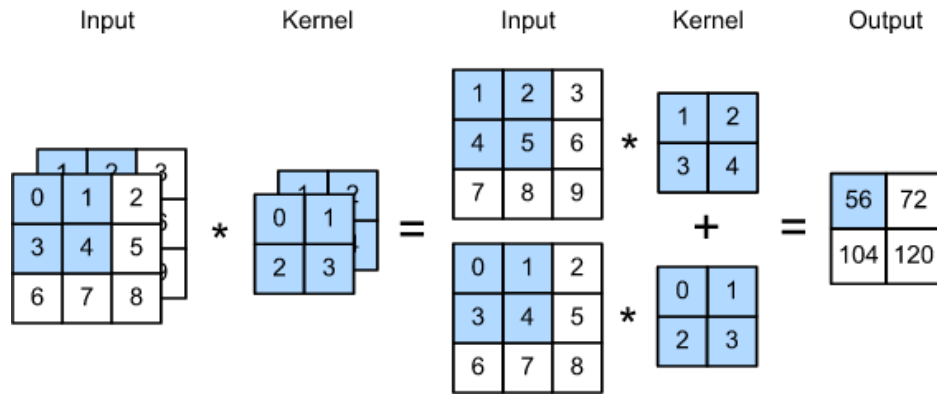


Figure 3.11: Demonstration of the 2D convolutional process of an input image with three dimensions to a 2D output image. The picture is taken from Ref. [119].

size [121].

The input matrix for further Conv2D layers can then be the generated three-dimensional volume with N channels. Multiple sequential convolutional layers are operated to construct a convolutional neural network. Therefore, the input images to a Conv2D layer are in general 3D volumes. However, it is important to note that the third dimension always represents a channel dimension. The distinct channels provide no correlations between each other, compared to an actual 3D image. When the input image to a Conv2D layer contains multiple layers in the third dimension a , the kernel requires equally multiple layers b , as visualized in figure 3.11. Each kernel layer b_i is convoluted with one specific input layer a_i of the input image. At the end, the results of all convolutions are pixel-wise summed-up to obtain a one-dimensional

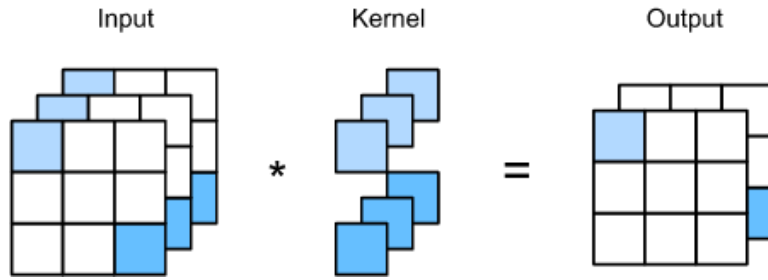


Figure 3.12: A 2D convolution of a 3D input image into a 2D output image with multiple channels. The picture is taken from Ref. [119].

output.

To detect multiple features present in the input image, additional kernels are included. For a 3D input image with a layers in the third dimension, to detect c features, $a \cdot c$ kernels are required. The process is demonstrated in figure 3.12. The input image contains three layers in the third dimension ($a = 3$), and two features are detected ($c = 2$), thus two output channels are generated. Therefore, two sets of three kernels each are required, what results in $a \cdot c = 3 \cdot 2 = 6$ kernels. The first set in light blue causes the first output channel, and the second set in blue the second one.

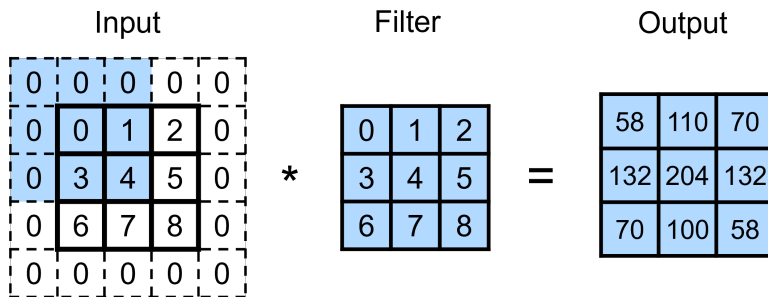


Figure 3.13: An example of Same padding to maintain the same output dimension as the input dimension. For a 3x3 filter dimension, the input dimension can be preserved by padding one layer of zeros around the input matrix.

In general, the output matrix of the convolution has a more narrow width and height than the input matrix, if a filter size larger than 1x1 is chosen. It is common to extend the input size virtually by adding border rows/column around the input image. If the boarder is filled with zeros, this is called "zero padding" (ZeroPadding). In libraries like TensorFlow [122] and PyTorch [123], this is denoted as "Same" padding, since it preserves the dimensionality of the output. An example of Same padding is presented in figure 3.13. Another established convolution setting is the "Valid" approach, where no padding is applied, and it may happen that columns or rows are dropped to ac-

commodate the discrete filter. Therefore, in Valid padding, the output dimensions can be more compact than the input dimensions.

3.3.2 Convolutional 3D Layers

The in the previous subsection introduced Conv2D layers are able to learn image correlations within two image dimensions, while utilizing a third channel dimension. In the following, this concept is extended to learn image correlations within three dimensions by employing Conv3D layers. An example of a Conv3D layer applied to a 3D input image is illustrated in figure 3.14. It can be recognized that the input image (denoted as source) and the filter of the Conv3D operation require three dimensions to calculate a particular value of the output matrix. The principle is the same as for Conv2D layers: the filter is pixel-wise shifted over the whole input matrix to calculate the output by the convolution operation. However, this time, the filter is shifted in three dimensions instead of only two dimensions as for the Conv2D layer. If a Conv3D layer with multiple filters is applied, then the output matrix is four dimensional, where the fourth dimension represents the channel dimension.

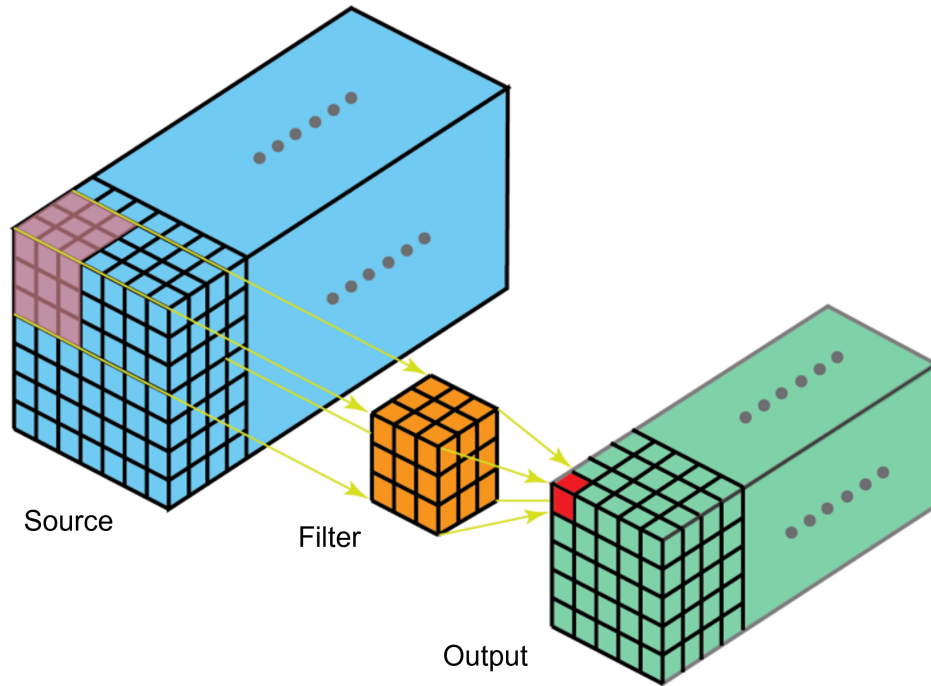


Figure 3.14: An example operation of a Conv3D layer applied to a 3D matrix. The picture is taken from [124].

The 3D filter matrices allow the Conv3D layers to learn correlations between neighboring pixels within all three spatial image dimensions. On the contrary, as explained previously, Conv2D layers are merely capable of learning the correlations within a plane of two image dimensions when applied to a 3D image. Therefore, traditionally, Conv3D layers are applied to 3D images and Conv2D layers to 2D images.

3.4 Supplementary Neural Network Layers

Additionally to convolutional layers, the neural network architectures introduced in the following also contain so-called LeakyReLU [125], BatchNorm [126] and Dropout layers [127]. These special layers are introduced in the following.

Activation functions introduce non-linearity into the network [128]. Non-linearity is required by neural network architectures to allow learning complex patterns from the data. Without non-linearity, summing-up the yields of multiple linear layers results only in another linear function. The used LeakyReLU activation function differs from the standard Rectifier Linear Unit (ReLU) activation function [129] by a small slope α for negative input parameters t . The chosen slope α corresponds to a so-called hyperparameter of the neural network. In DL, hyperparameters are parameters which are defined at the beginning of the training and are not changed during the training. The slope α is by default set to $\alpha = 0.3$, as visualized in figure 3.15.

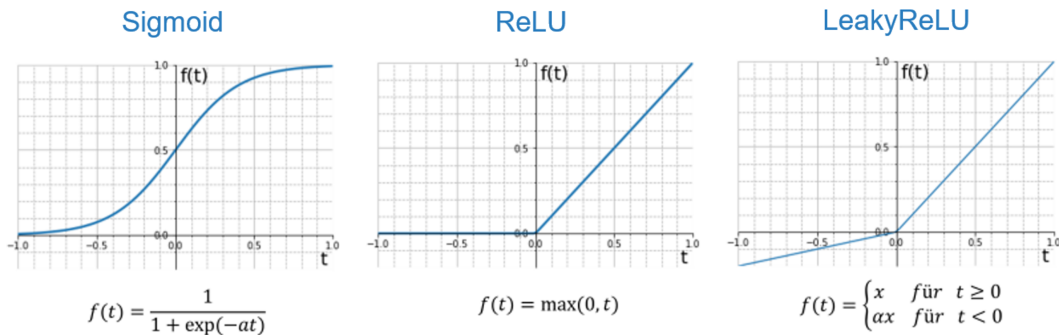


Figure 3.15: Shown are: on the left, the sigmoid activation function; in the middle, the standard ReLU activation function; and on the right, the LeakyReLU activation function with a small slope $\alpha = 0.3$ for negative input values t .

Batch normalization (BatchNorm) layers re-centralize and re-scale the data to execute the computations of the parameters of the previous convolutional layer in a more limited range. BatchNorm layers possess significant importance in DL. In particular, they

stabilize the training process and support deep neural network architectures to converge [126].

The last special layers are Dropout layers, which represent a method of regularization for neural networks [127]. They reduce the possibility of overfitting by turning off – denoted as “dropout” – some nodes from the previous convolutional layer by setting their output to zero. It should be noted that turned off nodes change randomly in each forward iteration of the neural network.

Chapter 4

3D Generative Adversarial Network Models

The types of GAN models applied in this research are the Deep Convolutional Neural Network GANs [109]. As the name implies, these neural networks consist primarily of convolutional layers. Deep indicates that the network consists of many hidden layers and therefore belongs to the category of DL. Since the images for which the models are trained are three-dimensional, an initial approach is chosen which utilizes Conv3D layers – or 3D operations – to represent the images. Conv3D layers are currently the first choice for processing 3D volumes in DL. The accuracy and the computing time for GAN simulations depend primarily on the neural network architectures. This chapter employs the Conv3D neural network architectures as a proof of concept for generating calorimeter shower images.

The content of this chapter is structured as follows. First, the generative adversarial network approach operated in this thesis is explained in detail. Subsequently, a description of the neural network training for the specific calorimeter use case and connected arising difficulties are discussed. Next, possible variations of generative adversarial networks are explored. Following this, initial Conv3D GAN architectures and the training loss functions are introduced. Furthermore, a “baseline” model from another study in Ref. [130] is presented, to which the results of this thesis are compared. Lastly, the so-called Wasserstein GAN (WGAN) [131] variation applied to the calorimeter training data is discussed.

4.1 Generative Adversarial Networks in Detail

GANs were first introduced by Ian Goodfellow in 2014 [14] and represent today a state-of-the-art technique for image generation. The principle is that two models, a generator and a discriminator (figure 4.1), carry out an adversarial training based on game theory. As already mentioned, in general, generative models belong to the category of unsupervised learning. However, the two competing models are re-framing the training and transforming it into a supervised learning task [132], where each model is supervised and evaluated by the other one.

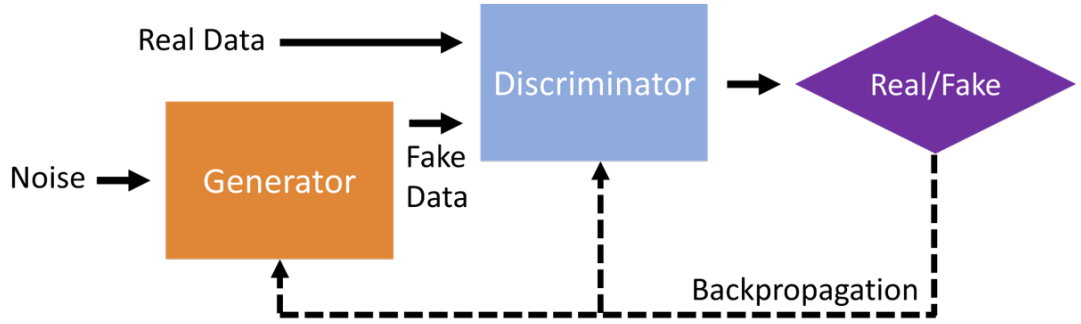


Figure 4.1: The GAN principle with the two models, generator and discriminator.

The generator network G generates fake images x from a random latent variable z . G tries to fool the discriminator network D by providing fake images labeled as true images (images from the training data set). Contrarily, the discriminator aims to differentiate between the true and fake images. The training goes in alternating turns. In the first turn, the generator improves its strategy to make the generated images closer to the ones from the training data. In the next turn, the discriminator improves its model to correctly distinguish between the generated and the training data. Subsequently, the generator is trained again, etc. The training is successful, when the fake images show an identical distribution compared to the true images. Therefore, the discriminator is no longer able to distinguish between true and fake images. This causes a classification prediction y of 50% for each class.

The two models, generator and discriminator, are typically deep neural networks, simultaneously trained by the objective function:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]. \quad (4.1)$$

Put differently, the generator and discriminator play a two player min-max game. The generator attempts to minimize the objective function in order to make the generated images more similar to the training data until they are indistinguishable. On the other hand, the discriminator maximizes the objective function to distinguish between the

generated data and the training data [14]. The GAN principle including its two models is visualized in figure 4.1.

The training of the models aims to achieve the so-called Nash equilibrium [133]. Nash equilibrium means that the models converged to a stable solution where they cannot be further improved and, therefore, the generated images possess a comparable quality as the training images [134]. In fact, in Ref. [135], it is theoretically proven that not all GAN models maintain a Nash equilibrium, depending primarily on the model architectures, the hyperparameters and the training data. Therefore, in reality, the training process is only aimed at converging as close to the Nash equilibrium as possible. In practice, it represents a non-trivial task to obtain satisfactory results in GAN training, as demonstrated later in this thesis. In Ref. [14], the theoretical proof is given that GANs can learn all data distributions if the networks are good enough (a reasonable number of trainable parameters and a compatible neural network architecture), if sufficient training data is provided and the training time is high enough.

To provide an example of the astonishing results of GAN models: a well-known use case of GANs is the image generation of human faces. The GAN model learns to create unknown synthetic faces from random noise. Figure 4.2 shows some example images generated by a GAN model from Ref. [136]. The generated images look incredibly realistic and accomplish an excellent resolution, which makes it unexpectedly challenging for humans to distinguish the generated images from real ones. Today, various modifications of GANs are being tested to generate more realistic images with a larger numbers of pixels.



Figure 4.2: Random samples of human faces generated by a GAN model in Ref. [136].

For training the GAN models, the calorimeter shower images from section 2.4 are employed as training data. The 200 000 available shower images are divided into 90% training data (180 000 images) and 10% validation data (20 000 images). The training data set is used for learning the parameters of the model throughout the training. The validation data represents an unseen data set for the neural network during training to measure the accuracy and avoid overfitting. When the model is overfitted, it will precisely represent the training data but performs worse on the validation data [137].

4.2 Generative Adversarial Network Training

GAN training is under no circumstances a trivial task. If the training converges at all, it is not simple to achieve the desired quality of the generated images. Finding convergence in GAN training for complex and high-dimensional data sets remains not straightforward. The two neural networks, generator and discriminator, have to be chosen in a correct size and ratio so that none is overpowering the other. The two networks are trained in an alternating way, each generator step follows one discriminator step. For the generator training, the discriminator parameters are kept constant. This is often denoted as freezing the network. For the discriminator training, the generator parameters are kept constant. The most applied DL use case are classification networks, which are – depending on their size and use case – difficult to train. The discriminator of the GAN model corresponds to a classification network. The composite GAN consists, of an additional generator network, making the training even more difficult. At the beginning of the training, the parameters are randomly initiated and, therefore, the generator outputs are random images. Setting up an environment that both models converge and the generator outputs realistic images remains a delicate task.

To achieve training convergence, the generator needs to improve during training. This results in a decrease of the discriminator performance, because the better the generator, the more difficult it is for the discriminator to distinguish between real and fake images. The generator achieves its best results when the discriminator outputs a 50% accuracy, where it cannot tell the difference between true and fake images. Therefore, the discriminator feedback gets less meaningful over time when the generator performance improves. When the discriminator cannot distinguish between real and fake data anymore, it outputs a random feedback [138].

4.2.1 Common Occurring Training Failure Modes

This section first examines some of the frequently arising GAN problems, which may cause the training to fail. This helps to better understand the possible solutions provided subsequently. Most of these problems still remain an active area of research. In particular the so-called: mode collapse, vanishing gradients, non-comparable models and non-convergence are investigated [139, 140].

Mode Collapse

Mode collapse appears when the generator falls into a state where it produces realistic images only for one image mode. For example, in the generation of handwritten digits, the generator produces only reasonable digits for the number five, but not for the

others. It is possible that these generated images look good enough to be able to trick the discriminator. Because the discriminator only evaluates if the provided images look reasonable and does not check if the evaluated images are always similar. In mode collapse, the generator becomes stuck in a local minimum while searching for the global minimum. Therefore, it fails to represent the complex data distribution with a high image variety as demanded.

Vanishing Gradients

Vanishing gradients appear when during the training process the discriminator is trained to perfection. In this case, the discriminator correctly distinguishes between true and fake images. Therefore, its True/Fake loss becomes close to zero because there is nothing to optimize. This phenomenon leads to low or even no gradient updates during the learning iterations. Differently said, to vanishing gradients. This aspect introduces a dilemma in GAN training: if the discriminator behaves badly, the generator does not receive accurate feedback because the loss function does not represent the reality. On the other side, if the discriminator performs great, the gradients of the loss function vanish to zero and the learning process becomes extremely slow [141]. It represents a delicate matter to design the power of the two GAN models to overcome the vanishing gradient obstacle.

Unstable Training Process

A major GAN problem is when it fails to converge at all, because the two simultaneously trained models lead to an unstable training process [142]. The GAN training goal is to obtain the Nash equilibrium in which both models perform perfectly. However, in the adversarial training of the two competing models, the improvement of one model leads to a deterioration of the other model. In some cases, this leads to continuing oscillations where one model outperforms the other, and after a few epochs this changes to the opposite [142]. A training epoch is the iteration over all training samples.

No comparable Models

The last discussed failure mode of GANs is that they lack a good objective function which examines how well the trained model performs. The loss values represent a relative measurement of the discriminator compared to the generator. If the generator outperforms the discriminator, then already after a few iterations the discriminator cannot distinguish anymore between the generated and fake images. This leads to a 50% classification probability what is previously defined as optimal, although the generated images do not follow the training distribution. Therefore, the discriminator loss presents no absolute information about the accuracy of the generator. This lack of a general accuracy metric makes it especially difficult to compare the results of multiple trained models to each other. Additionally, it does not provide precise details when to stop the training.

4.2.2 Techniques for Improved Training

In the following, some known techniques for improving the training performance are introduced. They are employed in this work for enhancing the quality of the generated images and to overcome some of the aforementioned obstacles.

Label Smoothing

In the GAN training, the discriminator determines the True/Fake loss by calculating if the input data represents a true or a fake image and comparing the result to its label. Typically, the label is 1 for true images from the training data set and 0 for fake images generated by the generator. In label smoothing, the 0 and 1 labels are replaced by the smoother values 0.1 and 0.9 [143]. Label smoothing can be regarded as a regularization method. It avoids that the model becomes overly confident on specific features in its prediction and, therefore, avoids over-fitting. By empirical tests, it is found that one-sided label smoothing performs best for our GAN model. In one-sided label smoothing is 0 kept as 0 and only the positive labels are smoothed from 1 to 0.9.

Label Flipping

Another technique employed in GAN training is label flipping. Hereby, the labels of some data points are with a low random probability switched from 0 to 1, or the other way around from 1 to 0. This random noise within the labels prevents the discriminator to be trained perfect, especially within the first epochs where the generator performs bad. Hence, it helps to avoid vanishing gradients, since the discriminator loss cannot completely fall to zero and, therefore, the training process does not stop. In our GAN training, 5% of the labels are randomly flipped.

Mini-Batches

One general known DL approach to reduce mode collapse of a GAN model is to train the models with mini-batches. The number of samples for one training and optimization step is denoted as the mini-batch size (often in short only batch size). The concept is that the gradients are calculated on multiple generated images during training, instead of calculating the gradients with one image in isolation. Updating the gradients based on the outputs of multiple images increases the variance of the gradients. This can help to reduce mode collapse during training. A mathematical proof is provided in Ref. [144]. Additionally, training with mini-batches creates a more robust training convergence and makes the training computationally more efficient due to possible parallelization.

4.3 Variations of Generative Adversarial Networks

As previously explained, bringing GAN training to convergence remains a complex task with many possible failure modes. In general, training the two GAN models is computationally demanding and time-consuming. Furthermore, the sensitive hyperparameter selection process requires training the models multiple times to test various configurations. All these reasons motivate the development of variations of the GAN training process to overcome some of the before mentioned issues.

One frequently applied variation represents the Progressive GAN approach [145]. The idea behind this is to grow the generator and discriminator networks progressively. In the first training iterations, low resolution images are created with only a few pixels, which are then made more and more detailed with an increasing number of pixels. This results in a more stable training and allows to efficiently train the model for even larger image sizes [145]. However, also the Progressive GAN implementation is non-trivial. In a parallel work with collaborators from Intel, the Progressive GAN implementation failed for our use case. Potential problems are the extremely sparse shower images combined with the enormous pixel energy range.

A second GAN variation are the Wasserstein GANs (WGANs). The WGAN approach is intended to resolve the difficulties of instabilities, mode collapse and avoid vanishing gradients in the training and aims at faster convergence [146]. The WGAN approach is in the scope of this thesis tested and, therefore, described in more detail in the following. WGANs introduce an alternative measure of the distance between two probability distributions, which is called Earth Mover distance or Wasserstein distance W . The Earth Mover distance is defined by:

$$W(\mathbb{P}_1, \mathbb{P}_2) = \inf_{\gamma \in \Pi(\mathbb{P}_1, \mathbb{P}_2)} (\mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]), \quad (4.2)$$

where $\mathbb{P}_1$ and $\mathbb{P}_2$ represent two probability distributions for the true and fake data and $\Pi(\mathbb{P}_1, \mathbb{P}_2)$ corresponds to all joint distributions $\gamma(x, y)$ [131]. In the initial WGAN paper from Ref. [131], some exemplary distributions are provided which converge under the Earth Mover distance but do not converge for other distance measures.

Because the initial suggested WGAN approach fails in some cases to converge, Ref. [146] proposes to apply a gradient penalty to the discriminator to ensure smooth training. In the context of WGANs, the discriminator network is renamed in critic that scores the realness of an image. Gradient penalty limits the L2 norm of the gradient of the critic to at most 1 everywhere with respect to its input to stabilize training [146]. The L2 norm is calculated by applying the square root of the sum of the squared vectors. The objective function for the WGAN loss from Ref. [146] is described as follows:

$$L = \underbrace{\mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} [D(\mathbf{x})]}_{\text{original critic loss}} + \lambda \underbrace{\mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}} [(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2]}_{\text{gradient penalty}}. \quad (4.3)$$

There exist further extensions to the standard GAN approach. The first represent conditional GANs (cGANs). In cGANs additional input parameters are provided to the neural network to generate specific images or images from a distinct class. In our use case, the GAN is conditioned by the input energy of the primary particle to generate shower images of the required particle energy levels. The second employed GAN extensions are Auxiliary Classifier GANs (AC-GANs). They implement additional losses to the True/Fake loss to improve the model convergence, which we also make use of, as described later.

4.4 3D Convolutional Architectures

In general, in machine learning two types of neural network architectures exist, generative and discriminative models. Discriminative models predict the labels of a data sample based on its initial distribution and are, therefore, applied in classification or regression tasks. Generative models aim to learn the probability distribution from a data set to generate new samples similar to the training distribution. The GAN approach contains both architecture types. This section provides a baseline 3DGAN model. Furthermore, the generator and discriminator neural network architectures of the in this research developed Conv3D GAN model and their key features are explained. The application and evaluation of these models are provided in the following chapter.

4.4.1 Baseline 3DGAN Model

The models developed in this research are compared to a baseline GAN model defined in Ref. [78] for the same calorimeter simulation task and trained on the same data set. The model is denoted as 3DGAN and introduced below. It is available on GitHub in Ref. [130]. Its generator architecture is displayed in figure 4.3. The generator neural network consists of a simple forward architecture. Forward architecture denotes that the information of each layer flows only to the subsequent one. The architecture employs Conv3D layers for learning image correlations.

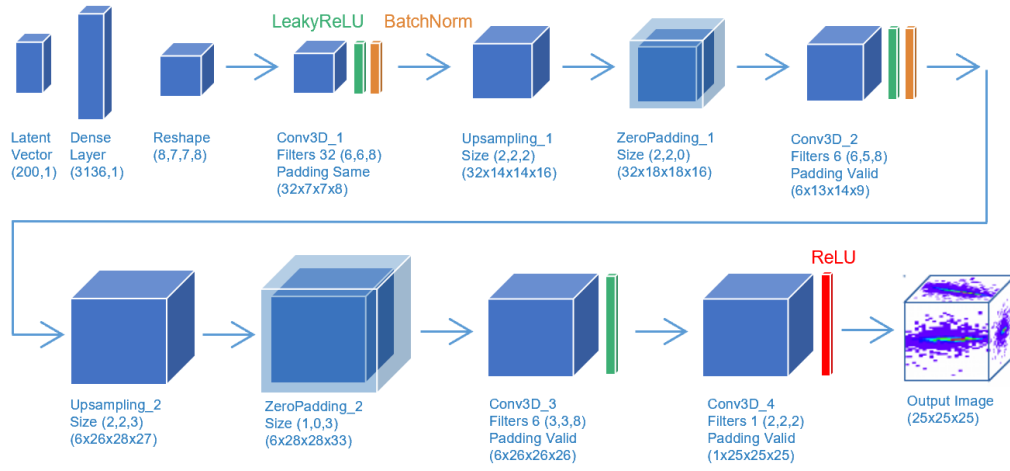


Figure 4.3: The baseline 3DGAN generator neural network architecture from Ref. [78].

In discriminative neural networks, the information flow from the beginning to the end of the network typically becomes reduced by compressing the information into a lower dimension. For generative models, it is the opposite. A more compact representation is upsampled through the network propagation to the final image size. In our specific case, the output of the generator represents a 3D image with the dimension $25 \times 25 \times 25$ pixels, the same dimension as the training images. The input into the generator is random noise. It is implemented as a latent vector consisting of 200 random numbers drawn from a uniform distribution of the interval $[0, 1]$. Random noise input is required to enable the GAN model to generate a broad variety of data samples following the target distribution. If there were no noise input, the generator would produce at each forward step exactly the same output. This is not appropriate for calorimeter data, which is subject to statistical fluctuations. The generator must also be able to generate different 3D calorimeter images according to the energy of the incoming particle. Therefore, the generator is conditioned on the particle energy E_p . Here, a special form of conditioning is chosen. The 200 dimensional latent vector is multiplied by E_p , representing the only dependency of the generated images. After placing a fully connected layer connecting each input node to each output node (denoted as Dense layer), the input is reshaped to a tensor with the dimensions $8 \times 7 \times 7 \times 8$. This volume is the starting point for the upsampling process in the following layers.

Conv3D layers are embedded to learn the correlations and features within the images and correctly reproducing them. For upsampling in image size, Upsampling layers are included which simply multiply the input dimension by copying the input values multiple times and do not involve any trainable parameters. Further, ZeroPadding layers are operated to adopt the volume sizes. LeakyReLU layers are applied after the standard convolutional layers, as indicated by the green bars in figure 4.4. Some

LeakyReLU functions are followed by a BatchNorm layer (brown bar) to stabilize the training. At the end of the neural network architecture, one standard ReLU layer (red bar) is placed to cut-off potential negative output values and set them to zero. For the GAN use case in this thesis, the output values of each pixel correspond to energies. Negative measured energies are in calorimeters non-physical and have to be avoided. After the GAN training is completed, the generated images from the generator should follow the same distribution as the training images. Visually, the generated images should look similar to the shower images from the training data set.

It can be noted that the 3DGAN model consists of merely four convolutional layers containing trainable parameters. Furthermore, most of the parameters of the model are concentrated in the first Dense layer (630k of 753k parameters $\approx 84\%$). However, as already explained before, for image processing, convolutional layers typically tend to maintain a higher representational power while operating smaller numbers of parameters [147]. Therefore, generally convolutional layers are the preferred layers in image processing. A better usage of the trainable parameters would be operating more convolutional layers and spreading the trainable parameters over the entire neural network depth. Based on the aforementioned aspects, an improved neural network is introduced in the following subsection.

The discriminator is only required to train the generator. As the best trained generator from Ref. [78] is used in the following, no discriminator and no further training are required. Therefore, the discriminator of the 3DGAN model is not visualized. However, it is provided in Ref. [78]. The 3DGAN model is in the following chapter utilized as baseline for the physics results comparison to the newly developed models.

4.4.2 The Conv3D Generator Architecture

The new developed more advanced Conv3D generator architecture is presented in figure 4.4. Opposite to the 3DGAN model, Conv3D_Transpose layers are placed for upsampling the volume. Conv3D_Transpose layers have trainable parameters and perform an inverse convolution to increase the image size. The principle of Conv3D_Transpose layers is more complex and for example explained in Ref. [148]. However, it is not significant to understand the neural network architectures in this thesis.

After each standard convolutional layer, a block of LeakyReLU, BatchNorm and Dropout layers are applied, as indicated by the brown bars in figure 4.4. LeakyReLU activation functions are applied to introduce non-linearity into the network. The LeakyReLU function is followed by a BatchNorm layer to stabilize the training. Lastly, a Dropout layer regularizes the training and reduces the possibility of overfitting.

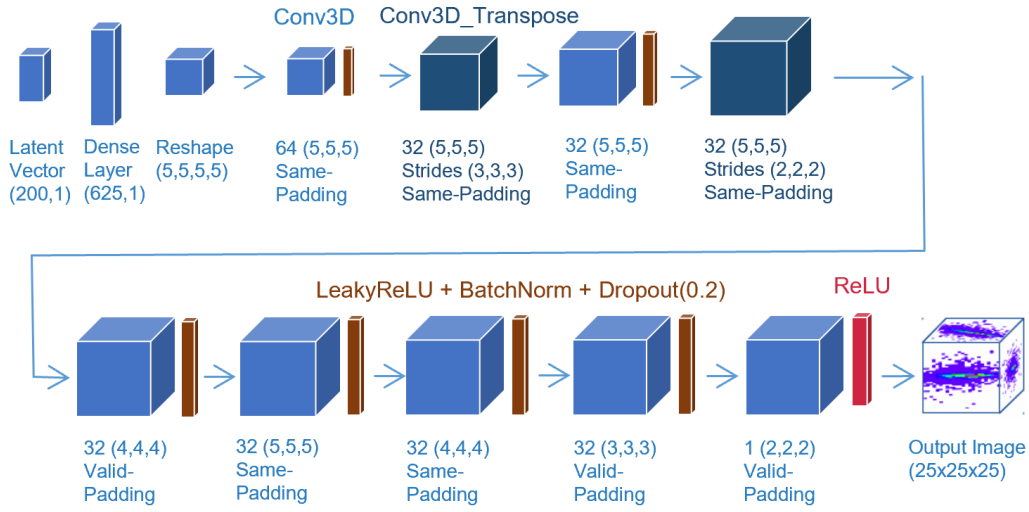


Figure 4.4: The Conv3D generator architecture.

A dropout rate of 20% is applied. This means that randomly 20% of the nodes are turned off. Dropout is not only applied as a regularization technique during training, also during inference it provides an additional source of randomness and helps to avoid mode collapse [149]. Similarly as for the 3DGAN baseline model, at the end of the neural network architecture, one standard ReLU layer is placed to cut-off potential negative output values.

For comparison, the Conv3D model has nine convolutional layers and its parameters are better distributed across the network. And, there are no dropout layers implemented in the baseline 3DGAN model.

4.4.3 The Conv3D Discriminator Architecture

The corresponding Conv3D discriminator architecture is displayed in figure 4.5. The inputs of the discriminator are either generated images (fake images) or images from the training data set (true images). The discriminator outputs three values representing the loss values. During training, the parameters of the neural networks are optimized in order to minimize these three loss values. The losses are explained in the following.

- The first loss term represents the typical GAN True/Fake probability L_{TF} [14]. It predicts how likely it is that the input image represents a true image labeled with 1, or contrary, how likely it is that the input image represents a fake image

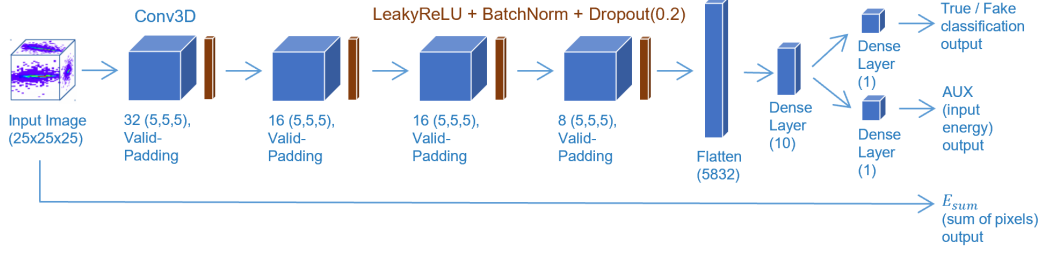


Figure 4.5: The Conv3D discriminator architecture.

labeled with **0**. The True/Fake loss is calculated by the binary cross entropy [150] after applying a sigmoid activation function (visualized in figure 3.15). The equation of the binary cross entropy for the True/Fake loss L_{TF} for N samples is:

$$L_{TF} = \frac{1}{N} \sum_{i=0}^{N-1} -(l_i \cdot \log(p_i) + (1 - l_i) \cdot \log(1 - p_i)). \quad (4.4)$$

The parameter l_i corresponds to the label of the i -th sample, which is either **0** or **1**, and p_i corresponds to the probability output of the discriminator for the same sample. The first term multiplied by l_i corresponds to the probability of class **1** and the second term with $(1 - l_i)$ to the probability of class **0**. During the discriminator training, the True/Fake probability measures if the discriminator can correctly distinguish between the true and fake images. If not, the parameters of the discriminator are adapted accordingly. This is opposite in the generator training, where the generated fake images are labeled as true images with label **1**, to fool the discriminator. This configuration allows the generator network to train its weights and generate images that are closer to the true images.

- The second loss term, the AUX loss (for AUXiliary) L_{AUX} , gives the result of a regression task operated to model the relation between the initial particle energy E_p present in the image labels and the discriminator estimate E_{AUX} from the images employing a separate Dense layer. It is implemented as a Mean Absolute Percentage Error (MAPE) [151]:

$$L_{AUX} = \frac{1}{N} \sum_{i=0}^{N-1} \left| \frac{E_{p_i} - E_{AUX_i}}{E_{p_i}} \right|. \quad (4.5)$$

This knowledge is then employed in training the generator to output images containing the correct amount of energy. Further, during training, the AUX loss value represents an additional measure of how good the training is.

- The third discriminator output comes from an additional layer, called lambda layer, a custom created layer, calculating the sum over the pixels of the input

image. This corresponds to the total energy E_λ contained in the discriminator input image. The loss is denoted Lambda loss L_{E_λ} and evaluates if the energy of the generated image E_λ agrees to the expectation from the label E_{sum} . The loss is calculated by the MAPE likewise:

$$L_{E_\lambda} = \frac{1}{N} \sum_{i=0}^{N-1} \left| \frac{E_{\text{sum}_i} - E_{\lambda_i}}{E_{\text{sum}_i}} \right|. \quad (4.6)$$

As displayed in the discriminator architecture figure 4.5, L_{E_λ} does not depend on the discriminator network, because it is directly computed from the input image, and no parameterized discriminator layers are involved. Therefore, it only influences the training of the generator network and not the discriminator training.

In addition to the True/Fake loss, the two losses L_{AUX} and E_λ are supporting the GAN to converge faster with fewer oscillations during training, and to achieve better results. Auxiliary losses are an established technique in training deep neural networks. Primarily, they are powerful at the beginning of the training, where the network outputs are purely random. Furthermore, auxiliary losses remain a reliable measure to evaluate the GAN performance during training in more detail.

The total loss L_{total} of the discriminator is calculated by multiplying each of the three losses with an individual scalar weight w as hyperparameter and summing them up:

$$L_{\text{total}} = w_{\text{TF}} \cdot L_{\text{TF}} + w_{\text{AUX}} \cdot L_{\text{AUX}} + w_{E_\lambda} \cdot L_{E_\lambda}. \quad (4.7)$$

The best loss weights w_i were determined to be:

- $w_{\text{TF}} = 6.0$
- $w_{\text{AUX}} = 0.2$
- $w_{E_\lambda} = 0.1$

The discriminator network contains no Conv3D_Transpose layers because in the discriminator the volume size is not increased. On the contrary, the size of the volume is shrunk down during the network by employing Valid padding. The 3D volume at the end of the discriminator becomes then stacked to a 1D vector by a flattening operation, followed by one Dense layer. Last, two separated Dense layers are applied for calculating the True/Fake loss and the AUX loss, respectively. As already indicated, the E_λ loss is directly calculated by summing-up the pixel values of the input image. The outputs of the discriminator network are then the three scalar loss values.

4.5 Wasserstein Generative Adversarial Networks

The WGAN approach with the gradient penalty is applied to the Conv3D calorimeter GAN model. The generator and the discriminator losses of the training, as well as the gradient penalty, are displayed in figure 4.6. It can be seen that after the convergence in epoch 41, the losses diverge again during the training in the following epochs. One epoch comprises the training over the entire training data set. However, exactly this divergence should be prevented by the WGAN approach. The loss at epoch 41 in the plot is visually close to zero. However, regarding the total loss values on the y-axis, they remain still orders of magnitude too high. Plus, the generated images perform poorly (not visualized). Therefore, the WGAN model does not achieve satisfactory results.

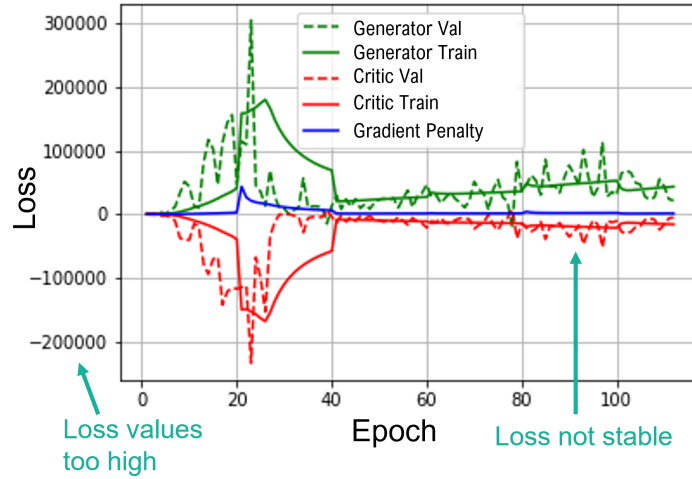


Figure 4.6: The generator and critic losses of the training (Train) and validation (Val) data and the gradient penalty during the WGAN training. The WGAN losses are determined by equation 4.2.

Although many modification and optimization attempts, the WGAN model could not be trained to achieve good results. Maybe the problem lies in the utilized calorimeter images, which are very sparse, or because of the large order of magnitude spanned by the pixel entries. In multiple subsequent internal researches, the WGAN approach was tested for the calorimeter use case, however, never successful. Therefore, the WGAN approach is discarded in this thesis. Nevertheless, the gained knowledge about WGANs might be beneficial at later stages for the quantum GAN researches. Before coming to that, in the following chapters, it is focused on the standard GAN approach.

Chapter 5

Transition from Convolutional 3D to faster 2D Networks

State-of-the-art image applications in computer vision utilize Conv3D layers for representing 3D volumes or images. Likewise, for three-dimensional calorimeter simulations Conv3D layers were operated in earlier researches, as discussed in the previous chapter. However, Conv3D layers have weaknesses. For example, they are computationally demanding and not (yet) implemented in some DL frameworks. Therefore, neural network architectures with Conv2D layers should be employed to overcome these problems. However, Conv2D layers are naturally unable to correctly process 3D images, as described in section 3.3. Within this thesis, smart architectures are developed to successfully handle 3D images by employing only 2D operations. This represents a completely novel and innovative approach.

This chapter describes the research results of Conv2D neural network architectures, with the goal of speeding-up the simulations and lowering the amount of memory while maintaining the best possible physics accuracy. The chapter is structured as follows. It starts with a detailed motivation for transitioning from Conv3D to Conv2D layers. This is followed by a brief literature assessment of associated approaches representing 3D images with only 2D operations. Next, the novel developed Conv2D architectures are introduced. Subsequently, the networks are evaluated for the calorimeter GAN use case. Both the computational improvements and the physics accuracy of the generated images are evaluated in training and inference. The physics validation remains a crucial topic for alternative calorimeter simulations, and therefore, many detailed physics metrics are provided. The chapter concludes with a discussion and possible plans for improving the model in future work. Parts of the content of this chapter, including the Conv2D architectures, as developed for the purpose of this thesis, are published as peer-reviewed papers in Ref. [152], Ref. [153] and Ref. [154].

5.1 Motivation for Alternative 2D Architectures

While achieving an enhanced level of accuracy, Conv3D layers require significant amounts of computational resources because of their 3D matrix operations. The significant hardware resources demand is even more limiting for GAN models, where two networks are simultaneously trained. Additionally, hardware limitations can constrain the selection of the hyperparameters of the models. For example, Conv3D layers in large CNNs require substantial amounts of memory. If the model memory is too large for the operated hardware, the batch size has to be reduced. This can negatively influence the training result. Additionally, to accommodate considerable training times, higher learning rates must be chosen which can, likewise, decrease the accuracy. Even more important, ultimately, the size and resolution of the generated images is constrained [152]. In contrast, due to the missing third dimension, Conv2D layers require significantly less computational effort.

As reported previously, Conv3D layers are in the DL frameworks often not as good optimized as Conv2D layers in terms of internal calculations and hardware allocation. The reason for this could be that Conv2D layers represent one of the most fundamental layers applied in neural networks and are, therefore, among the first implemented and best optimized ones. On the contrary, Conv3D layers are not yet as frequently employed in applications and only implemented at later stages of DL frameworks and possibly less optimized. This can cause longer simulation times and considerably larger hardware resource requirements.

An initial internal analysis with a primitive Conv2D network architecture was carried out in Ref. [155]. It demonstrated that Conv2D networks possess an enormous potential for speeding-up the simulation time by orders of magnitude compared to Conv3D layers for the calorimeter GAN simulations. However, this speed-up came with the negative effect of a strongly decreased physics accuracy. Nevertheless, the possible immense speed-up motivated further research on smarter and more complex neural network architectures which can maintain the accuracy while reducing the simulation time.

Another motivation for Conv2D primitives comes from the practical side. As explained before, Conv2D layers are among the first components to be implemented in any novel DL tool developed by the community. For example, in most reduced precision quantization tools, Conv2D layers are present, oppositely, Conv3D layers are not. In a later chapter of this thesis, studies about quantizing the network parameters to a lower precision format are performed to further speed-up simulation time and reduce memory occupancy (chapter 6). At the time when the reduced precision study was carried out, no quantization tool supported Conv3D nor Conv3D.Transpose layers. However, the respective Conv2D layers are supported. Therefore, to be able to

perform the reduced precision study, it is required to create a compatible network architecture. Not only reduced precision computing frameworks lack Conv3D layers. At earlier stages of this research, it was often tried to test new DL tools or frameworks and to assess them. But the Conv3D layers were repeatedly not supported and made these tests impossible.

5.2 Related Work: 2D Networks for 3D Images

The employment of Conv2D layers to combat three-dimensional image problems is limited. In Ref. [156], 3D image classification is achieved by building three two-dimensional slices along the three canonical axes. These three independent 2D image slices are classified by a neural network. Ref. [157] expanded this method by stacking a set of Conv2D layers along the x -, y - and z - axes, thus rebuilding a 3D image from the initial set of 2D image slices. This approach resulted in memory efficient execution and almost state-of-the-art accuracy with respect to Conv3D-based classification.

Within the field of HEP simulations, Ref. [158] proposed a network consisting of Conv2D layers to generate the 3D volume of a calorimeter: a 3D 12x5x7 image is generated by stacking twelve 2D 15x7 images together. This model yields accurate physics results and an impressive 6 660x speed-up with respect to standard Monte Carlo techniques, running on an NVIDIA GTX 1080 card. The results of the in this thesis newly developed models are compared to the architecture presented in Ref. [158].

5.3 Novel 2D Convolutional Neural Network Architectures

In this section, new Conv2D architectures for reducing the simulation times and memory occupation are introduced and explained. The transition to Conv2D layers, where only two-dimensional operations are involved to generate 3D images, remains a challenging task. Creating adequate Conv2D architectures is extremely complex and difficult. The following description starts with the first naive approach and finishes with three successful network architectures.

5.3.1 Naive Approach: Channel Dimension as 3rd Image Dimension

The first introduced neural network architecture is derived from the simplest possibility for operating Conv2D layers to obtain 3D images. It divides the three-dimensional

image volume into 2D slices, where each slice is put in one channel of the Con2D layer. More detailed, the calorimeter shower images have the three dimensions $25 \times 25 \times 25$ and are cut into 25 two-dimensional images along the z -axis with the dimensions 25×25 . As explained in section 3.3, Conv2D layers can only learn interactions between the image pixels within one channel. That means, for the naive network, only correlations within the two image dimensions x and y can be learned and the correlations between the image z dimension cannot. As expected, the physics accuracy of this model is low, confirming the indication of the previous research in Ref. [155]. Increasing the number of parameters, which means including many more layers and increasing the kernel size, does not improve the accuracy significantly.

Therefore, more innovative and smarter concepts for the network architecture must be invented to allow the network to learn the correlations within all three image dimensions. In the following subsections, three more or less extreme configurations for neuronal networks with Conv2D layers are introduced, to determine the best option. Essentially, all network architectures maintain some transposing operations in which the 3D volume is rotated between the convolutional operations. The models differ for example in the number of branches for the image generation. The first model employs three branches, the second model 25 branches and the third only one single branch and is considerably deeper than the other networks.

5.3.2 Generator 1: The Three Branches Architecture

The first successful Conv2D architecture for 3D image generation is named G1. It is visualized in figure 5.1. The 200 dimensional latent input is mapped by a Dense layer to a 125 component vector, which is reshaped to a 3D $5 \times 5 \times 5$ volume. This volume is copied three times and transposed such that each copy is oriented along a different coordinate axis. This is illustrated in figure 5.1 with different colors (orange, blue and green) for the planes of the 3D volume. Each of the branches is focused to learn the image correlations between two dimensions, while operating the third dimension as the channels.

Subsequently, in each branch, the volume is upsampled inside the *convolutional block* by Conv2D_Transpose layers and processed by some Conv2D layers. Upsampling means that the volume is increased in size to attain at the end of the branches the size of $25 \times 25 \times 25$ pixels. Within each of the convolutional blocks of the three branches, the correlations of the local x - and y -axis are learned, but not within the local z -axis. At the end of the branches, the three volumes are transposed back into the initial orientation and concatenated to a shape of $75 \times 25 \times 25$ pixels. An additional Conv2D layer is placed in order to obtain the desired image dimensions of $25 \times 25 \times 25$ pixels. As last layer, one ReLU layer cuts-off potential non-physically negative values.

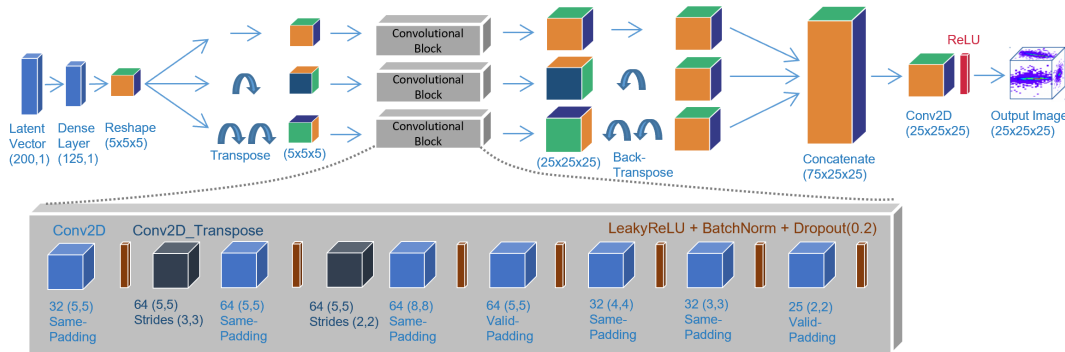


Figure 5.1: The G1 generator: three branches architecture. Each of the brown blocks in the convolutional block comprises a LeakyReLU, a BatchNorm and a Dropout layer.

Later, in the model evaluation section 5.5, it will be proven that with the three-branches and the concatenation, the G1 neural network architecture is able to successfully generate 3D images with only 2D layers. Within each of the branches, the correlations of two different spatial image dimensions are learned. Merging them together allows the network to learn the correlations of all three spatial image dimensions. The *convolutional block* of each branch consists of nine convolutional layers (seven Conv2D layers and two Conv2D_Transpose layers for upsampling). The configurations of the Conv2D layers are provided in figure 5.1. Each Conv2D layer is followed by a LeakyReLU, BatchNorm and Dropout layer, except the last Conv2D layer after the concatenation.

The Three Branches Discriminator Architecture

The corresponding discriminator architecture utilizes the same three branches approach and is illustrated in figure 5.2. The core of the discriminator neural network architecture looks similar to the G1 architecture, it maintains three branches with transposing operations and *convolutional blocks*. The transposing within the three network branches allows evaluating the image correlations along all three canonical axes. The discriminator outputs are the same as in the Conv3D architecture from section 4.4.3. The *convolutional block* of the discriminator consists of six Conv2D layers and no transpose layers. The power of the discriminator, regarding the number of convolutional layers and the number of parameters, is chosen to match the power of the generator, and therefore, it does not out- or under-perform the generator architecture during training.

This three branches discriminator architecture is operated for the training of all three Conv2D generator architectures developed in this section. The reason for this is that in GAN training the generator and the discriminator network need to maintain the appropriate ratio of representational power (defined by number of layers, number of

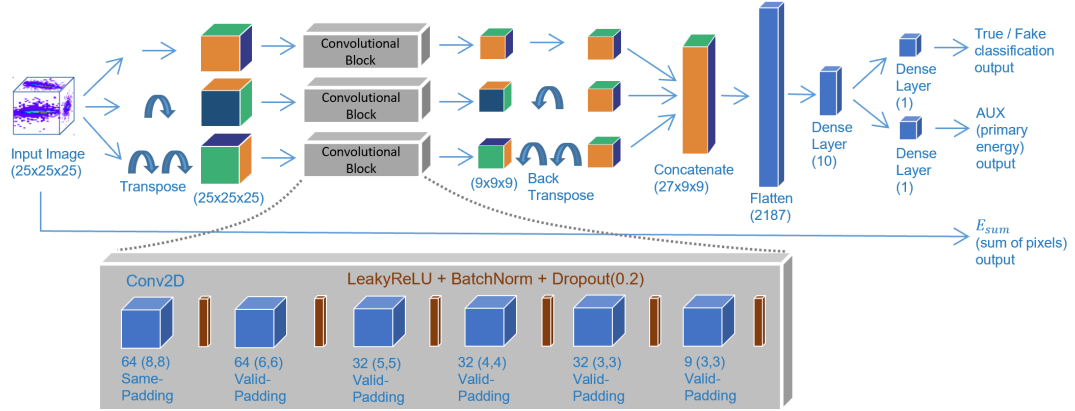


Figure 5.2: The discriminator following the G1 architecture.

parameters, ...) in order to reach convergence during training. All newly created generator architectures are required to demonstrate the same or similar representational power to make the architectures comparable. This can be achieved by optimizing the generator architectures in training with the same discriminator architecture. Because the generator and the discriminator are two separate networks, the network topology of one does not influence the other one. That implies that although the G1 generator shares the same three branches network approach as the discriminator, it has no advantage over the other – in the following introduced – generator networks. Later in this thesis, it will be demonstrated that the G1 architecture achieves the best physics accuracy and additionally the best speed-up and memory occupation of the newly developed Conv2D network architectures. Therefore, the general discriminator architecture is chosen to follow the G1 architecture.

5.3.3 Generator 2: The 25 Branches Architecture

The second generator architecture (G2) follows the structure outlined in another research project from Ref. [77], as described in section 5.2. The network architecture is modified to generate images with the desired dimension for our use case, but the principle remains the same. The G2 network consists of 25 branches, one for each of the 25 layers along the z -axis. Each separate branch creates a two-dimensional image with a 25x25 shape. Afterwards, the 25 2D images are stacked together to rebuild the 3D 25x25x25 volume. Finally, two other Conv2D layers are placed. The G2 generator is presented in figure 5.3.

The G2 generator has the disadvantage that, like the naive approach, it enables little interaction between the layers of the third image dimension. Furthermore, it consists

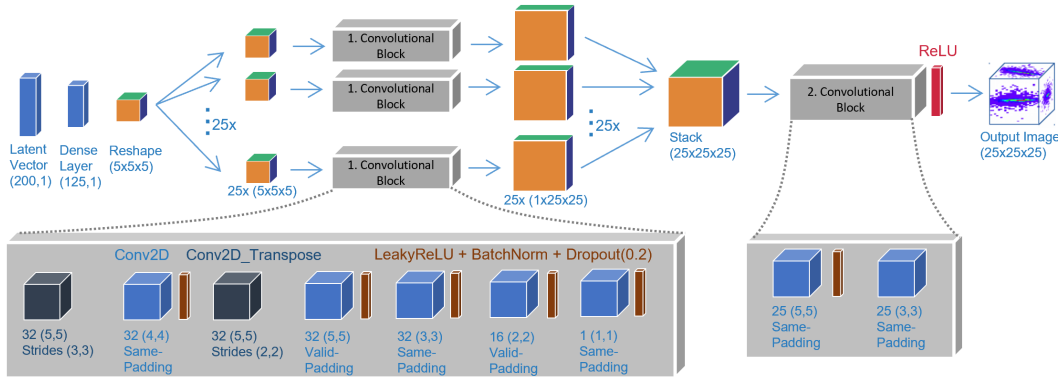


Figure 5.3: The G2 generator: 25 branches architecture.

of 25 branches which contain a convolutional block with seven convolutional layers each. That means, it consists of many convolutional layers ($7 \cdot 25 + 2$, the +2 for the second convolutional block). It should be highlighted that the convolutional layers within each of the 25 branches contain their own trainable parameters, which are not "shared" between the branches. The advantage of the G2 generator is that the architecture is more straight forward and easier to implement, compared to the G1 architecture with the transposing operations. Moreover, in theory, the 25 branches can theoretically be computed in parallel instead of sequential, which could promote some computational advantage.

5.3.4 Generator 3: The Single Branch Architecture

The third G3 generator architecture represents a considerably deeper network without branches. It alternates Conv2D layers to transpositions along the three axes in such a way that each axis is represented once in the channel dimension of the convolutional layer. In order to improve convergence in the training, it employs skip connections across the layers, as illustrated in figure 5.4. Skip connections are proven to help model convergence for extremely deep neural networks [159]. The advantage of the G3 architecture is that there are no concatenations of various branches, which means that the correlations to the neighbor pixels of all three spatial image dimensions are learned within one branch. The disadvantage is given by its depth, for which the training process requires a significant larger number of epochs before reaching convergence. Furthermore, the deeper the network, the more difficult it is to accomplish good accuracies in the training.

In the following, the three generator architectures are compared to each other and to

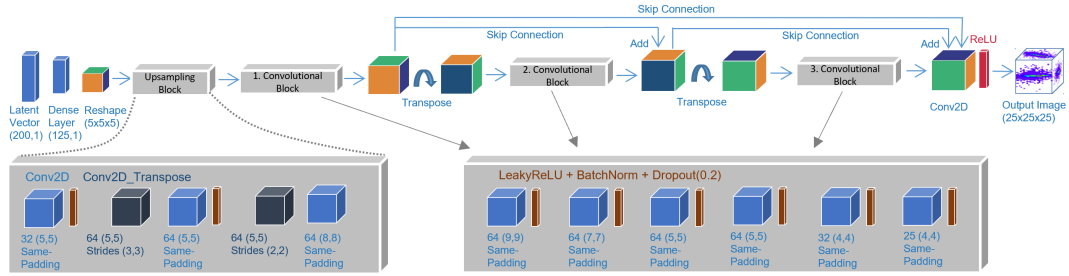


Figure 5.4: The G3 generator: single branch architecture with three skip connections.

the baseline 3DGAN network. Table 5.1 summarizes the total number of parameters for the generator architectures: the initial 3DGAN network has barely half the number of parameters and an extremely limited number of convolutional layers, compared to the newly developed Conv2D architectures. The new Conv3D model developed in this thesis contains slightly more parameters than the baseline 3DGAN model, and also more than double as many convolutional layers which perform superior to Dense layers in image tasks, as described in section 3.3.

Table 5.1: Summary of the number of parameters and the number of convolutional layers for each in this thesis used generator model.

Model	Parameters	Conv Layers
3DGAN (baseline)	752 981	4
Conv3D	965 169	9
Conv2D G1 (3-branches)	2 055 324	$9 \cdot 3 + 1 = 28$
Conv2D G2 (25-branches)	2 131 000	$7 \cdot 25 + 2 = 177$
Conv2D G3 (1-branch)	2 397 022	25

The Conv2D generator architectures contain all more than 2M parameters. The G3 generator consists of the lowest number of convolutional layers, directly followed by the G1 architecture. The G2 generator uses many more convolutional layers than the others. Although the Conv2D models include more parameters than the 3DGAN architecture, they are expected to decrease the computation time because the Conv2D layers require less computational resources compared to the Conv3D layers. In the following sections, a comprehensive evaluation of the training and inference of the models is provided.

5.4 Training Evaluation

The evaluation of DL models can be split into two parts. The first part represents the training from a random parameter initialization until a satisfying accuracy is reached. The second part represents the application of the trained model for which it was developed, denoted as inference. Of those two, the inference process is more important for the later application. Inference is repeated several times and a reasonable accuracy, low inference computing times and low hardware resource consumption are important. On the other side, the training is run once on large clusters without hardware constraints. However, for the complicated GAN use case in this study, the training is executed multiple times with different hyperparameters to test which configuration performs best. Furthermore, in the development process of the Conv2D networks, the training is repeated multiple times to find proper network architectures, where none of the GAN models is outperforming the other. For the calorimeter simulation HEP use case, the training performance is even more important than for other use cases. Because in the future, it is planned to employ the GAN models to validate newly developed high granularity calorimeter designs for the LHC upgrades or future colliders. For testing new calorimeter designs, it is crucial to enable reasonable training times such that tests can be performed quickly, leading to a more rapid development process of calorimeters. And in most cases, the more training trials can be carried out in a short time, the larger the chance to find the best network architectures as well as most suitable best calorimeter designs. Therefore, both the training and inference remain crucial and must be studied precisely. Here it is started with evaluating the training process.

DL training is a complex and tedious iterative process to achieve a satisfactory accuracy level. The DL training contains numerous parameters which have to be chosen at the beginning of the training and influence the training process significantly. These parameters are denoted as hyperparameters. Depending on the use case, there can be numerous hyperparameters. Examples are the learning rate in the training and the neural network architecture (design, type and number of layers, number of parameters). Due to the considerable complexity of the models, it is impossible to theoretically determine the perfect parameters before training. Therefore, the models and hyperparameters are generally optimized with multiple training runs (called trials) with the trial and error method. For example, test a more significant number of parameters in the neural network architecture and measure if the accuracy increases. Moreover, most of the hyperparameters are related to each other. That means, if for example the neural network size increases, also the learning rate must be adopted, resulting in a change of other parameters. Therefore, an iterative parameter optimization, where only one parameter is optimized at a time, is not appropriate. This makes DL training to a complex task, where expertise and imagination of the correlations in the network are necessary. Particularly, it is a delicate process when it comes to GAN training with

two competing neural networks. For GANs, the first complex task is to find a configuration in which the training converges at all and the generator produces reasonable results before starting with the fine-tuning.

It is common to use specialized tools to perform an automated hyperparameter search. The simplest approach is a grid search where all parameters are scanned systematically with a predefined step size within given limits. When the number of parameters becomes large, the simple grid search rapidly becomes too hardware costly, as it requires many trials for finding good hyperparameters. Often a random search is more effective in this case, but there also a variety of refined algorithms that try to adapt to the problem [160]. A typical approach is to formulate the parameter search as a Bayesian optimization [161] problem, but also evolutionary algorithms [162] are used.

Here, first a coarse grid search is used, which is then refined with a Bayesian algorithm. For this, the open-source hyperparameter search tool Optuna [163] is used which allows parallel multi-GPU training. The primary optimized hyperparameters are the neural network architecture, learning rate, learning rate decay and the three discriminator loss weights discussed in section 5.3. The learning rate lr for the individual training epochs t is the initial learning rate lr_0 multiplied by an exponential decay with the learning rate decay constant lr_{dec} :

$$lr(t) = lr_0 \cdot e^{-lr_{dec} \cdot t}. \quad (5.1)$$

For the highly statistical GAN training, which includes a lot of randomness, it is not ensured that hyperparameters reach equally good results in each trial. Therefore, multiple repetitions are carried out with the same hyperparameter configuration.

The GAN model in this research is more stable and converges faster when different learning rates for discriminator and generator network are utilized. As a rule of thumb, it is observed that the ratio of the discriminator learning rate compared to the generator learning rate should be roughly the same ratio as the number of trainable parameters in the discriminator network $\#_d^p$ compared to the number of parameters in the generator network $\#_g^p$, what is for our GAN models approximately 1/3:

$$\frac{\#_d^p}{\#_g^p} \approx \frac{lr_d}{lr_g} \approx \frac{1}{3}. \quad (5.2)$$

The learning rate employed in the Optuna optimization represents the generator learning rate $lr = lr_g$. The corresponding discriminator learning rate lr_d can be calculated by $lr_d = 1/3 \cdot lr$. This reduces the number of hyperparameters and with this the number of search trials, leading to a decrease of the hardware requirements in training.

The best hyperparameters for each model are listed in the appendix A.1. In the following, the training behavior of the three Conv2D networks is evaluated. All subsequent

tests are executed in the Python version 3.6.8 and TensorFlow version 2.2.0.

5.4.1 Dying ReLU Problem in Training

All the introduced generator architectures consist primarily of LeakyReLU layers used as activation functions within the networks. Only for the last layer at the end of the generator network, one standard ReLU layer is operated. It cuts-off all negative values and allows the generator to output only positive pixel values. Shower images can only contain positive energy depositions, negative energy depositions are nonphysical and must be avoided. In the following, it will be explained why the important ReLU layer causes difficulties in the GAN training.

ReLU layers output the value zero for negative input values. This can cause problems when many network outputs are negative, because then, most of the outputs of the following ReLU layer are zero. This leads to inactive or "dead" parts in the network where a broad range of values remain zero. When many neurons output zero, the gradient flow during backpropagation loses its meaning and the weights do not become updated. This part of the network becomes then inactive and does not learn further. Therefore, it is stuck unrecoverable, or "dead." In particular in deep neural networks with many trainable parameters, this frequently leads to completely dying networks [164].

This problem equally appears in the training of our GAN models. With tracking some parameters of the generator neural network, it can be observed that during training more and more parameters change to zero, until the comprehensive network "dies" after a few iterations. Often, the so-called dying ReLU problem can be solved by decreasing the learning rates, eliminating negative bias, applying different weight initialization methods or testing different optimizers. All these techniques were extensively examined for our model, without success. Therefore, the last chance to overcome the dying ReLU problem is to replace the ReLU activation function with a more gentle LeakyReLU activation function, which occupies a small negative slope for negative input values instead of zero outputs, as demonstrated in figure 3.15. Because of this small slope, the outputs of the LeakyReLU function cannot become zero and the dying ReLU problem can be avoided. However, as explained before, the ReLU layer at the end of the generator network is important for physics accuracy reasons.

A very elegant solution to overcome the dying ReLU problem is explained in the subsequent. The network is trained for the first few epochs with only LeakyReLU activation functions. Later, when the model already learned some correlations, the last LeakyReLU layer is replaced by a standard ReLU layer and the training continued until convergence is reached. With this, it is possible to train the network successfully

without encountering the dying ReLU problem. Due to trial and error, it revealed that the ReLU function can already be implemented after two epochs of training. Therefore, as from the third epoch onwards, the model contains the standard ReLU function as a last layer. This approach is employed in the training of all architectures. A possible reason why our GAN use case struggles with the dying ReLU problem might lie in the employed sparse image data.

5.4.2 The Training Loss Evolution

For evaluating the training, the new Conv2D models are compared with the Conv3D model developed in this thesis. The 3DGAN model is only employed in the inference evaluation in the next subsection because the best trained weights from Ref. [78] are used. Therefore, no training is carried out with the 3DGAN model. In figure 5.5, the total weighted training loss from equation 4.7 for the various models is visualized. Weighted means that the individual losses are multiplied by its loss weights. The losses are computed during the training with the validation data set.

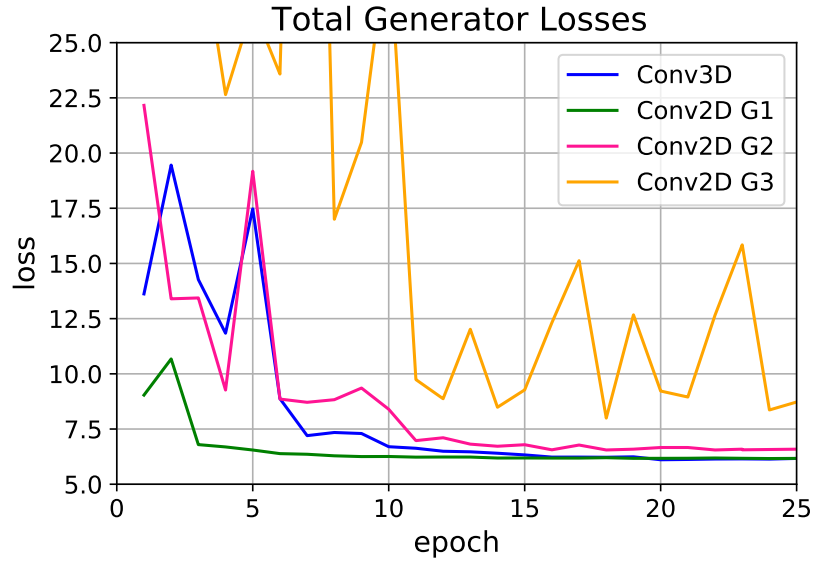


Figure 5.5: Total weighted training loss calculated with validation data as a function of the number of epochs for the different models: Conv3D, G1, G2 and G3.

In figure 5.5, it can be noted that the G1 architecture converges fastest and reaches a low and stable loss value already after around six epochs. The G2 takes a few more epochs, Only around epoch 14 the training remains stable. The Conv3D model takes approximately a similar number of epochs as the G2 architecture to converge in training. However, the loss of the G2 model remains worse than the G1 and Conv3D model.

The training for all networks is run for at least 25 epochs. The G3 training is more unstable due to its deeper neural network. After very high loss values up to a loss value of 35, it takes around 35 epochs until it converges to a stable loss (not shown). The loss values of all models for the last epoch are provided in the appendix in table A.2. The plot shows the results of the best trial and, therefore, no statistical errors are provided. The errors between multiple training trials would be potentially as large as the fluctuations.

The losses of the various models can be analyzed in more detail by observing the contributions of the single loss terms. The True/Fake losses for the various models are provided in figure 5.6. They display a similar behavior as the total losses. The G1 architecture converges fastest, followed by the Conv3D architecture and the G2 architecture, which perform well. Once more, the G3 architecture takes more epochs to converge. The losses are plotted unweighted to be able to compare them.

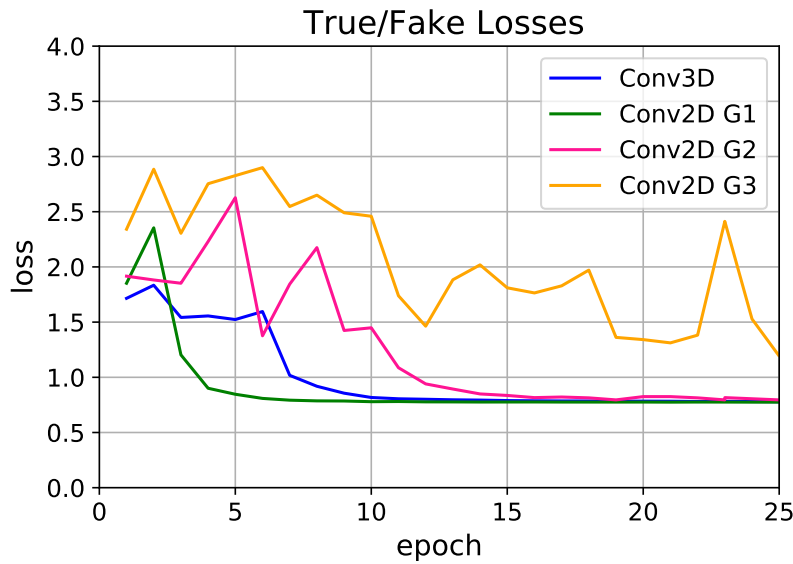


Figure 5.6: Unweighted True/Fake training losses as a function of the number of epochs for the different models: Conv3D, G1, G2 and G3.

The AUX losses are provided in figure 5.7 and the Lambda losses in figure 5.8. With respect to all loss plots, consistently the G1 architecture is converging fastest, followed by the Conv3D and G2 architecture. The G3 architecture takes longest to converge. As in the loss plots only the losses of the best trials are shown, no errors are provided.

Training Time

For calculating which of the models has the lowest training time, not only the number of epochs, is of interest. Furthermore, the computing times per epoch must be taken into account, which are provided in table 5.3. It can be noted that the G1 architecture

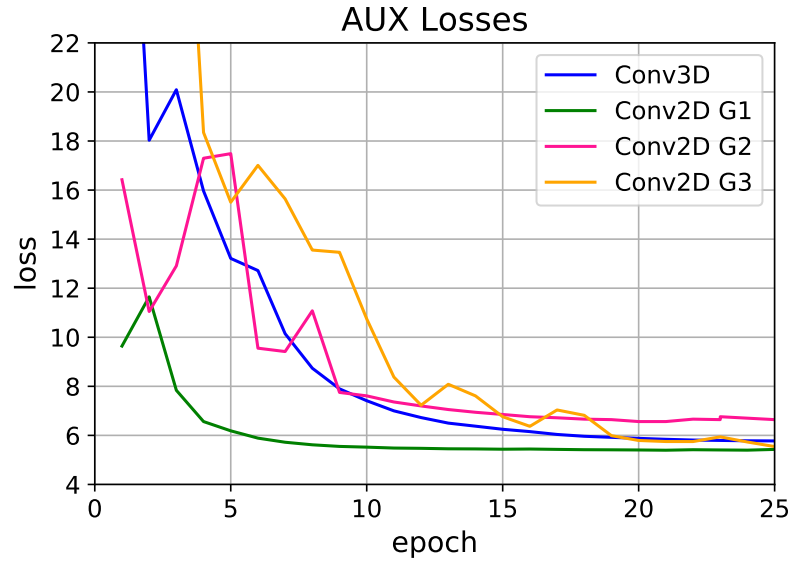


Figure 5.7: Unweighted AUX training losses as a function of the number of epochs for the different models: Conv3D, G1, G2 and G3.

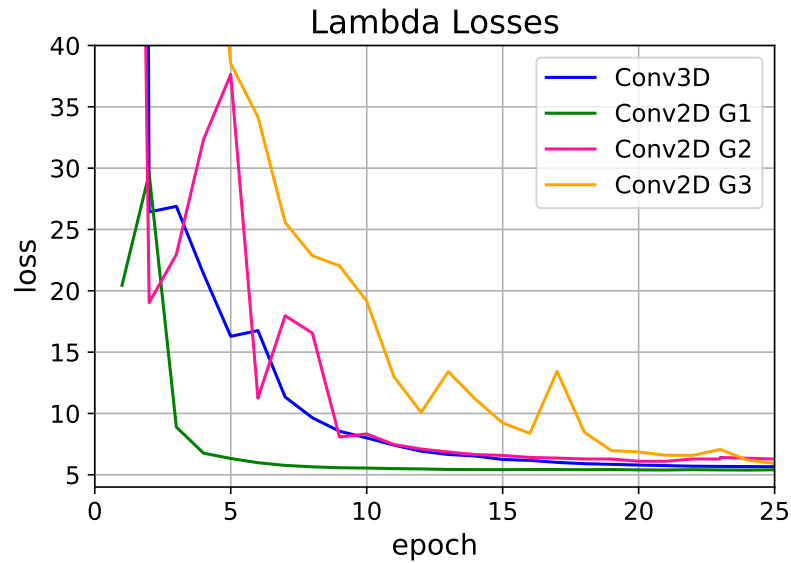


Figure 5.8: Unweighted Lambda training losses as a function of the number of epochs for the different models: Conv3D, G1, G2 and G3.

performs best in terms of computing time, with merely 40 minutes for training one epoch. This is a factor of 6.5x faster than the Conv3D model, which takes 258 minutes for one epoch. The G2 and Conv3D model converge fast in terms of losses, but their

training times per epoch remain the highest. The G2 model requires 71 minutes and the Conv3D model 258 minutes. The G3 model requires the largest number of epochs to converge, but its training time per epoch is with 47 minutes considerably low. With a speed-up of up to a factor **6.5x**, the new Conv2D architectures clearly outperforms the Conv3D model in the training time per epoch. Since the best trained 3DGAN model from Ref. [78] is used for the inference evaluation, no training was performed for this model.

Table 5.2: Average training times per epoch over the full training for the various models on an Nvidia Tesla T4 GPU, and the corresponding speed-up compared to the Conv3D architecture.

Model	Training Time [min]	Speed-Up
Conv3D	258 ± 5	1.0x
G1 3-branches	40 ± 1	6.5x
G2 25-branches	71 ± 2	3.6x
G3 1-branches	47 ± 1	5.5x

All in all, taking into account the training time per epoch and the required number of epochs the models take to converge, the Conv2D G1 architecture clearly performs best. Its training is the most stable, its losses converged fastest and, moreover, it has the lowest training time per epoch. This confirms, from the training side, that a change from Conv3D to Conv2D models comes with considerable computational benefits. Furthermore, a more stable training reduces the number of training trials needed to find the hyperparameters that achieve a model with reasonable accuracy.

5.5 Inference Evaluation

In the following, the performance during inference is evaluated. Inference is the process of using a trained neural network for the specific task. In the case of the GAN model studied here, inference means generating shower images at the given input energies E_p . During inference, only the generator part of the model is used and the discriminator network can be discarded. The goal is to use as few computational resources as possible while maintaining the best accuracy of the generated showers. For this purpose, multiple evaluation metrics are presented and discussed that allow to quantify the performance of the model with respect to computational requirements and the accuracy of the shower physics.

5.5.1 Computational Evaluation

The baseline, to which the GAN models are compared is given by the detailed Geant4 simulation. The average time for generating one shower image with Geant4 was in Ref. [165] (in 2018) determined to 17s on an Intel Xeon processor. This value is referred to when the speed-ups for the GAN models are calculated. Further, the models are compared to the 3DGAN baseline model introduced in section 4.4.1. The further models under study are the three new developed Conv2D models (G1, G2 and G3) and the Conv3D model developed in this thesis (section 4.4.2).

In practice, the computational throughput depends on the model architecture and the hardware device on which it is executed. As hardware, an Nvidia Tesla T4 GPU is employed with TensorFlow version 2.2.0 and a batch size of 128. In order to measure the inference time, an initial set of 20 warm-up batches are executed and then 100 inference steps, including 20 batches for each of the measurements. Warm-up batches are employed to initialize and allocate the GPU memory [166]. Table 5.3 summarizes the inference results for the models. Also, the GPU utilization is provided, which is a measure of GPU usage. Nvidia defines it as the “percent of time over the past sample period during which one or more kernels was executing on the GPU” [167].

Table 5.3: Summary of the number of parameters and the number of convolutional layers (Conv) for each generator model. Additionally, the inference times of the models on GPU and their speed-ups versus the 3DGAN network are provided. The last row presents the average GPU utilization during inference.

Model	Parameters	Conv	Inference Time	Speed-up	Utilization
3DGAN	752k	4	7.00 s	1.00x	78.8 %
Conv3D	965k	9	16.86 s	0.41x	98.1 %
G1 (3-branches)	2 052k	28	4.91 s	1.29x	21.7 %
G2 (25-branches)	2 131k	177	17.48 s	0.40x	21.8 %
G3 (1-branches)	2 397k	25	6.21 s	1.13x	32.9 %

The values for the inference times in table 5.3 clearly demonstrate that the G1 (3 branches) architecture is the fastest with the lowest inference time. Therefore, it provides the highest speed-up (factor of 1.29x), despite it has more than twice as many parameters as the 3DGAN and Conv3D model. Furthermore, the Conv2D G1 architecture has a considerably lower GPU utilization compared to the 3DGAN model (factor of 3.6x). The hardware utilization represents an essential aspect. With fully utilizing the hardware, the inference time could be further decreased, as demonstrated later.

The G2 model exhibits an even higher inference time than the Conv3D model, potentially caused by the larger number of convolutional layers. This is in contrast to

what is expected. While designing the model, it was expected that because of the large number of branches, a lot of operations can be calculated in parallel to accelerate the computations. Apparently, this is not the case since the GPU utilization of 21.8% is considerably low, similar to the value of the G1 model. The G3 model has a marginally higher inference time than the Conv3D model and is 1.13x faster than the 3DGAN model. Additionally, it has a considerably lower GPU utilization than the 3DGAN model, but a larger one than the G1 model. Summarizing the computational results so far, similar as for training, the G1 architecture performs in all aspects best. Therefore, the potential of the Conv2D G1 model is in the following further investigated.

In a subsequent test, it is aimed to make full use of the GPU for the G1 model, instead of only utilizing 21.7% , as provided in table 5.3. In order to increase the GPU utilization, more data must be simulated in parallel. This can be done on GPU by simply increasing the batch size when running the inference. The Conv2D G1 model inference is run with different batch sizes, as illustrated in figure 5.9.

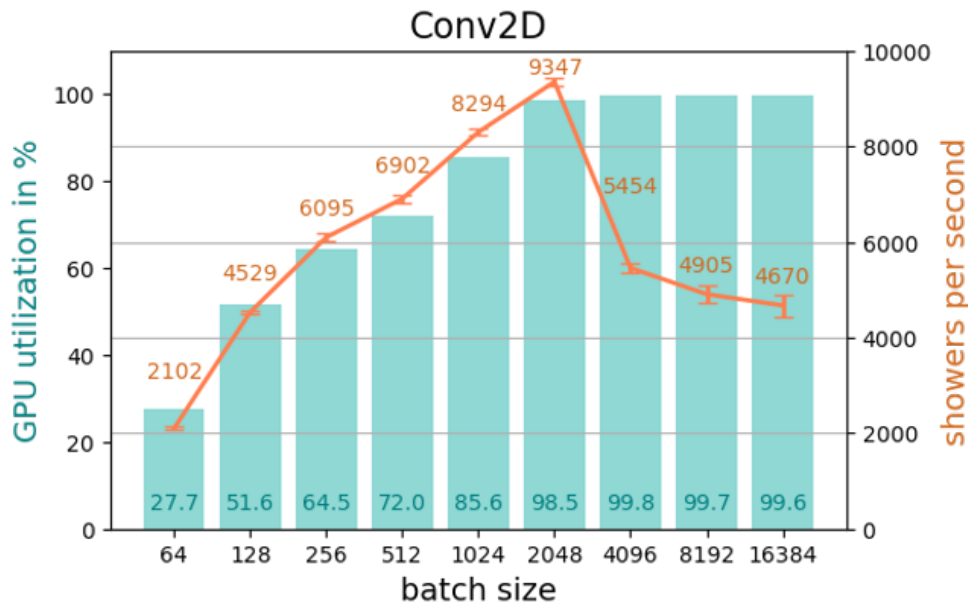


Figure 5.9: Comparison of the number of generated showers per second and the GPU utilization for different batch sizes of the Conv2D G1 model on the Nvidia Tesla T4 GPU.

It can be observed, the GPU utilization and the number of generated showers per second increase approximately linearly from a batch size of 64 upwards to 2048. At the batch size of 2048 the GPU is almost completely utilized, and on average 9347 showers per second are generated. When increasing the batch size further, the GPU

gets oversubscribed and the number of generated showers per second drops down. Therefore, the best inference results are achieved with a batch size of 2048 where 9347 showers are generated by an almost full GPU utilization of 98.5%. When this number is compared to the 17s for generating one shower with Geant4, the G1 model reaches a tremendous **158 000x** speed-up. The error bars at the showers per second measurement points are small and indicate robust inference times.

The same inference test with different batch sizes is performed for the Conv3D model developed in this thesis. The results are presented in figure 5.10. As it can be noted, the Conv3D model utilizes the complete GPU already with a batch size of one. But for rising batch sizes, the number of showers per second increases until it achieves a maximum of 1379 showers per second with a batch size of 1024. With larger batch sizes, the number of showers per second decreases. When comparing the generated showers per second of the best Conv3D configuration to the best configuration of the Conv2D G1 model, the Conv2D G1 model is $9347/1379 = 6.78x$ faster in inference. However, equally, the Conv3D model reaches a significant speed-up of **23 239x** with the fastest configuration compared to Geant4. The reason why the GPU is already fully utilized for a batch size of one is unknown. Presumably, this is connected to the reason that the Conv3D layers are not perfectly optimized for GPU usage.

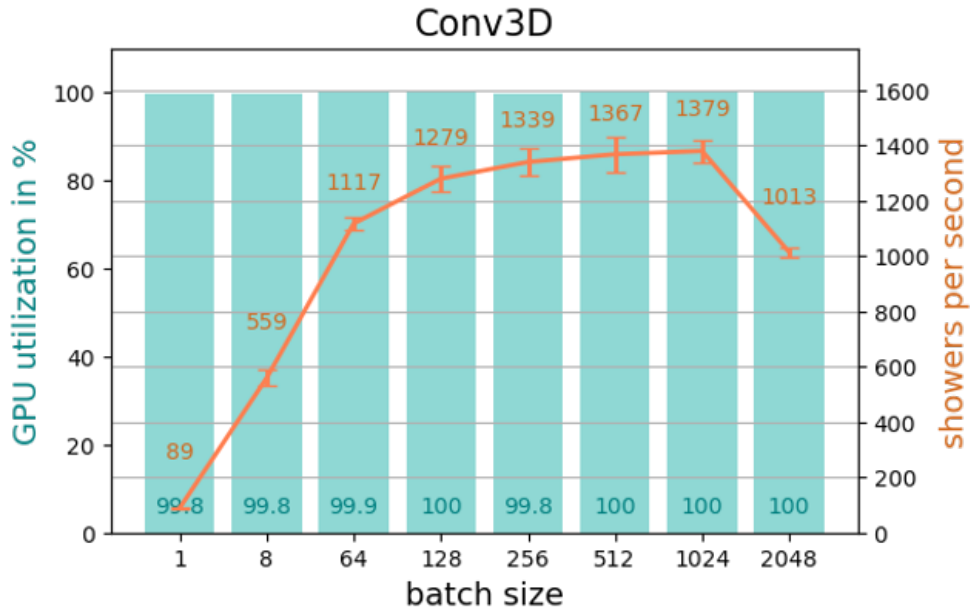


Figure 5.10: Comparison of the number of generated showers per second and GPU utilization for different batch sizes of the Conv3D model.

The 17s simulation time of Geant4 for creating one shower image measured in the previous test in Ref. [165] was measured in 2018 on, at that time, modern hardware. However, the hardware technology evolved and improved and, therefore, this number is outdated and will most probably be lower on current hardware. Additionally, development work is ongoing for increasing the Geant4 hardware performance by employing parallelization and vectorization techniques. It is also investigated, if running on GPUs instead of CPUs comes with advantages. However, the Geant4 developers expect these improvements to obtain a maximal speed-up of a factor five [7]. Even with the potential factor five improvement of Geant4, the GAN simulations remain orders of magnitude faster.

5.5.2 Physics Accuracy Evaluation

In this subsection, the physical accuracy of the new Conv2D and Conv3D models is evaluated in comparison to the previous 3DGAN model and Monte Carlo simulation. In general, accuracy evaluation of generative models for image generation remains a challenging task. For humans, it is straightforward to evaluate if images look reasonable or not. For computers, this is non-trivial. Depending on the specific applications, several methods have been proposed [168]. Because our use case is an image generation task, most of the validation metrics presented in this study remain visual.

For evaluating the physics behind the generated shower images, the energy distributions across the calorimeter volume are studied, focusing on specific quantities typically used to characterize calorimeter properties. Most of the provided validation approaches in this thesis represent already known metrics in the HEP community. However, some metrics are slightly adapted or have been newly developed for this research. The following provides an evaluation, in which the generated images of the GAN models are compared to the ones of Geant4. Since there is no general agreement on which physics metrics are the best for calorimeter simulations, all possible metrics are evaluated. However, all metrics examine different aspects of the shower images, which makes them all important.

1. Mean Squared Error Performance Measure

In order to enable the quantification of the agreement of the GAN models with Geant4, a single number is desirable. The same metric will be used throughout this thesis. It is calculated by building three two-dimensional projections of the particle shower distributions along the canonical axes x , y and z , respectively. For the calorimeter

data samples as described in section 2.4, the average energy in cell ijk is defined by:

$$n_{ijk} = \frac{1}{N_{tot}} \sum_l e_{ijk}^l, \quad (5.3)$$

where l iterates over the number of samples N_{tot} and e_{ijk}^l is the energy of a single hit in a single cell. In the sums, i iterates over the pixels in the shower x -dimension, j in the y -dimension and k in the z -dimension. With this, the three projections can be calculated with:

$$n_{x,i} = \frac{1}{N} \sum_{jk} n_{ijk}, \quad n_{y,j} = \frac{1}{N} \sum_{ik} n_{ijk} \quad \text{and} \quad n_{z,k} = \frac{1}{N} \sum_{ij} n_{ijk}. \quad (5.4)$$

The projections correspond to the mean over the cells of the other two axes. Therefore, N corresponds to the number of cells in the two respective dimensions over which the average is calculated. All axes have the dimension of 25 and, therefore: $N = 25 \cdot 25$. With this, the Mean Squared Error (MSE) performance measure is calculated by:

$$\text{MSE}_p := \frac{1}{3} \left(\sum_i (n_{x,i}^{GAN} - n_{x,i}^{G4})^2 + \sum_j (n_{y,j}^{GAN} - n_{y,j}^{G4})^2 + \sum_k (n_{z,k}^{GAN} - n_{z,k}^{G4})^2 \right). \quad (5.5)$$

The MSE_p value is averaged over 20 000 samples. For the GAN models, images for the same primary particle input energies E_p , as present in the validation data set from Geant4 (abbreviated as G4), are generated. The MSE_p between Geant4 and the corresponding GAN models are summarized in table 5.4. The average MSE_p values and standard deviations are calculated for ten inference repetitions.

Table 5.4: The Mean Squared Error performance measure (MSE_p) between Geant4 images and generated images of the corresponding GAN models and its standard deviations for ten inference repetitions.

Model	MSE_p
3DGAN	0.0651 ± 0.0021
Conv3D	0.0483 ± 0.0018
G1 (3-branches)	0.0270 ± 0.0010
G2 (25-branches)	0.0484 ± 0.0012
G3 (1-branch)	0.0712 ± 0.0015

Since the MSE_p represents an error, the lower the MSE_p , the better the accuracy. The new Conv2D G1 architecture achieves by far the lowest MSE_p ($=0.0270$), which corresponds to the best accuracy of the tested models. The G2 and the Conv3D architecture perform similar, with an MSE_p of around 0.048. The 3DGAN model has an MSE_p of 0.0651 and the G3 model performs worst with an MSE_p of 0.0712. Therefore, the G1 model performs clearly best with an almost three-times better MSE_p than the 3DGAN model. This represents an outstanding result and the first proof of this thesis showing that it is possible to successfully generate 3D images with only Conv2D operations.

Model Reduction:

The following physics validation metrics are all visual. For not overloading the figures and not confusing, it is only focused on the best models:

- In case of the Convolutional 3D networks, it is decided to only provide the 3DGAN model for multiple reasons. First, it is published by another research group and represents a solid baseline to compare to. Second, as demonstrated in the previous subsection in table 5.3, the inference times of the new Conv3D model are considerably larger than the inference time of the 3DGAN model. The goal of this research is to speed up the simulation time and, therefore, no model should be introduced which performs worse in terms of simulation time (although the physics accuracy of the Conv3D model seems to be better). Therefore, in the following physics evaluation metrics only the 3DGAN model is provided.
- Similar thoughts are valid for the new Conv2D architectures. In the following only the values of the best model are provided. The choice clearly falls on the G1 model, which achieves the best simulation time and the best physics accuracy. Before this choice was made, all the available physics accuracy metrics for all Conv2D models were carefully evaluated. In all of them, the G1 model performed best. The G1 model is in the following renamed to Conv2D. It is compared to the 3DGAN model and Geant4.

In order to achieve a good physics performance, the models have to perform good in a variety of validation metrics, which are introduced in the following.

2. Example 3D Showers

An initial visual inspection of example showers is provided in figure 5.11. It demonstrates that the Conv2D and the 3DGAN model can generate realistic shower images. Visually, they are similar to Geant4. As primary particle, a 400 GeV electron is employed. For Geant4, from the test image set, an image with a primary particle energy closest to 400 GeV is taken. However, drawing conclusions from a single image is not feasible. Therefore, the following metrics consider the full image sets.

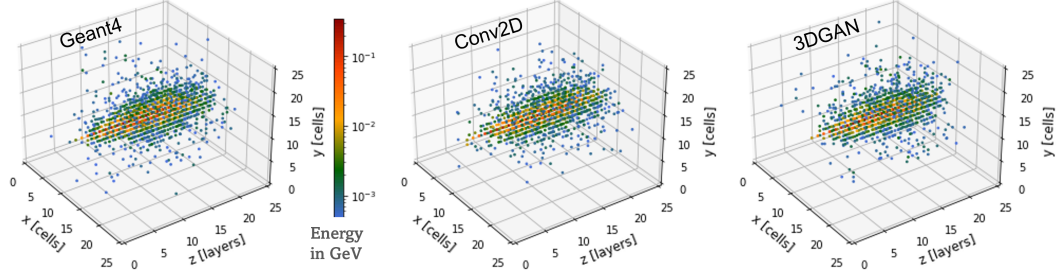


Figure 5.11: Example 3D shower images of Geant4 (left), Conv2D generator (middle) and 3DGAN generator (right) for a 400 GeV electron. The primary particle enters the calorimeter from the left, in the middle of the x - and y -axis ($x = y = 13$) at $z = 0$.

3. Shower Shape Plots

The 2D projections of the energy shapes along the calorimeter y - and z -axis (averaged over 20 000 samples of the validation set) are presented in figure 5.12, in linear (left) and logarithmic scale (right). The x -axis projections are similar to the ones of the y -axis and, therefore, not visualized as discussed in section 2.4. In general, the shower shapes of the GAN models are close to Geant4, resulting in a good physics accuracy. A more comprehensive analysis of the logarithmic y -axis distribution (top right) indicates, the tails of the 3DGAN distribution (in blue) are off for low energies. This was already observed in the publication of the 3DGAN model in Ref. [78]. On the other side, the Conv2D model (green) improves the description of the tails and follows the distributions of Geant4 perfect within statistical uncertainties. This means, in the shower shape metrics, the Conv2D model performs better than the 3DGAN model. In fact, with the Conv2D model, it could be the first time achieved that the distribution tails are represented correctly, even for the extremely low energy tails. Possible reasons might be the more complex neural network architecture of the Conv2D network with the three branches and the transposing operations in combination with the larger number of trainable parameters.

4. Mean Energy Deposition

Figure 5.13 provides details about the deposited energy inside the calorimeter with respect to the incident electron energy E_p . The GAN electrons are generated by the same input energy E_p present in the Geant4 events to minimize statistical effects. The root-mean-square (in the following denoted as mean) is calculated and the standard deviation of the 90% core of the distribution, labeled with μ_{90} and σ_{90} respectively.

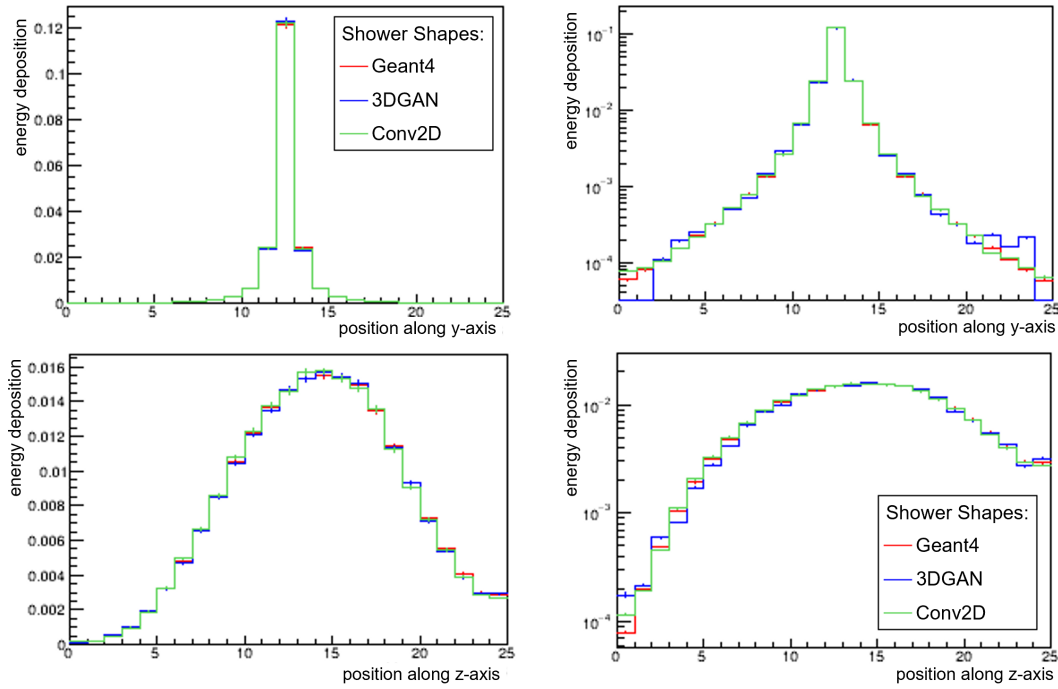


Figure 5.12: Particle shower shape distributions along the calorimeter y - (top) and z -axis (bottom) in linear (left) and logarithmic energy scale (right). The Geant4 prediction, the 3DGAN model and the Conv2D model outputs are provided in red, blue and green respectively. The energy deposition is given in arbitrary units due to the specifics of the shower shape calculation process.

In the upper panel of the left plot of figure 5.13, it can be noted that the generated shower images of both GAN models deposit a similar energy as the Geant4 showers across the full energy range and, therefore, correctly reproduce the mean energy. The lower panel visualizes the percentage deviation of the GAN models with respect to Geant4. Both models have a slight degradation from Geant4, primarily at the start and the end of the energy range, and perform good in between. The Conv2D model has a maximal deviation of 2.5% to Geant4, and the 3DGAN model has a maximal deviation of around 3.5%. This means that the Conv2D model performs slightly better than the 3DGAN model in this metrics. However, for such fast simulation approaches – as the GAN approach – minor deviations from Geant4 within a few percent are tolerated. Therefore, both GAN models lie most probably within a reasonable range. However, specific tolerances levels for fast simulation approaches are still to be defined.

In the right plot in figure 5.13, the relative width of the measured energy is displayed. The relative width is calculated by dividing the standard deviation σ_{90} by the root-mean-square μ_{90} . In the upper panel the absolute relative width is provided and in

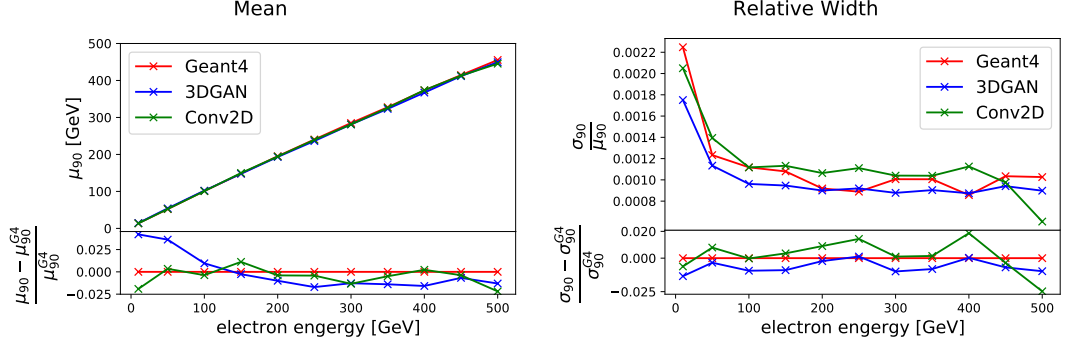


Figure 5.13: Deposited mean energy μ_{90} (left) and relative width σ_{90}/μ_{90} (right) as a function of the primary particle energy. The lower panels provide the relative deviations from Geant4.

the lower one the width relative to Geant4. Both GAN models maintain a relative width similar to the one of Geant4. The relative width means, in physics sense, how large the range of the fluctuation of the measured energy is inside the detector for a specific particle input energy. It is important that the GAN models exhibit fluctuations around the same magnitude as Geant4. This means the GAN models can also generate a variety of different shower images for the same primary particle energy. However, as it is visualized in the upper right plot, the order of magnitude lies in the per-thousand area, therefore, the expressiveness of the relative width metrics may be restricted. This metric was proposed in Ref. [169]. To standardize the evaluation of generative models for calorimeter simulations, this metric follows the format proposed in Ref. [169].

5. Cell Energy Deposition

Figure 5.14 presents the single cell energy deposition for 20 000 shower images with input energies within the full energy range of 10-510 GeV. The plot shows on the x -axis the energy deposition of a single cell and on the y -axis the number of cells which contain the corresponding energy. It is observed that the high energy hits are well described by both GAN models. However, for lower cell energies below 1 MeV both GAN models slightly deviate from the expected distribution of Geant4. In the lower cell energy range, the new Conv2D model performs a bit worse than the 3DGAN model by underestimating Geant4. In a physics sense, this means that the generated GAN model images contain a moderately reduced amount of low energy pixels than Geant4. This problem is most likely due to the fact that the energy range of the single pixels spans across multiple orders of magnitude, which makes it difficult for the GAN models to learn the lower energy depositions. For this reason, an energy threshold at

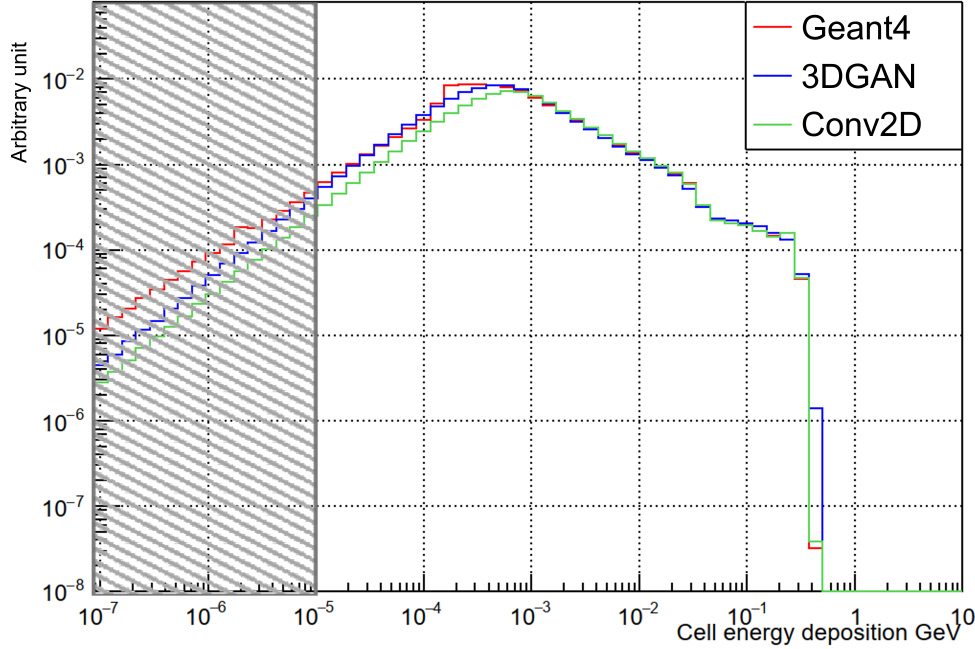


Figure 5.14: Single cell energy deposition. Visualizes how many cells of a corresponding energy level are contained in the shower images.

10^{-5} GeV is applied (gray shaded region in the plot) for all further validation metrics. In general, in realistic calorimeters the cells for extremely small energies are discarded anyway because of a low signal-to-noise ratio. The exact threshold for discarding low energy cells depends on the calorimeter design and material. If the results of the cell energy deposition plot are compared to other researches, for example in Ref. [169], the GAN models in our research clearly perform better for the lower energy depositions.

The small bump at around $2 \cdot 10^{-4}$ GeV in the Geant4 data in figure 5.14 represents a characteristic of calorimeters. Most probably the energy corresponds to the loss of a minimum ionizing particle which passes the calorimeter sensors perpendicular to the incident angle [169]. Minimum ionizing particles are particles whose primary energy loss mechanism is ionization, defined by the Bethe-Bloch formula [170]. The GAN models are not able to perfectly reproduce this behavior.

Although the cell energy depositions are low for low energies, the mean energy deposition provides good results. This is possible because the cells where the single cell energy deposition metrics are off contain tiny energy depositions which do not significantly contribute to the mean energy deposition. It could also be the case that the GAN models equalize the underestimation for small energy cells by overestimating

the higher energy cells slightly. However, the overestimation for high energy cells would be tiny and, due to statistics uncertainties, not significantly measurable.

6. SSIM Index

In figure 5.15, the results of employing the Structural Similarity (SSIM) index [171] are visualized as the sixth evaluation metric. The SSIM measures the perceptual difference between similar images, but does not judge if one image is more appropriate than the other. The SSIM is frequently applied in the photo and video industry to evaluate quality losses [172]. In this thesis, the SSIM index is employed to compare two image sets image-wise to calculate a similarity value. With this, it is measured if the generated GAN images provide a similar image diversity as present in the Geant4 data.

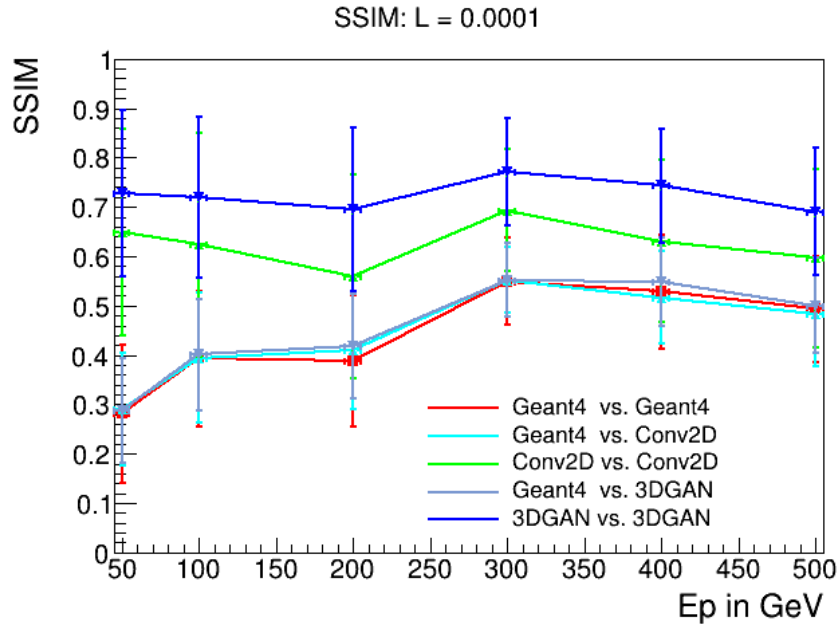


Figure 5.15: The SSIM index ($L = 10^{-4}$) for various model configurations.

The SSIM index is calculated by the scikit-image library [173], which employs the equations from Ref. [171]. The SSIM index is sensitive to the dynamic data range of the image pixels and must be adjusted by the parameter L in order to maintain the sensitivity. Generally, L can be chosen as the "distance between minimum and maximum possible values" of the image entries [173]. For our complex shower images case – with highly sparse images and pixel energy entries spanning over multiple orders of magnitude – several L values ranging from $L = 10^{-6}$ to $L = 0.1$ are scanned

in this test. It is found that the SSIM index stabilizes starting from $L = 10^{-4}$ and below. The SSIM plot for $L = 10^{-4}$ is visualized in figure 5.15. The SSIM plots for the other L parameters are provided in appendix A.3, for providing further insights into how the adjustable parameter L is chosen. The metric requires two images which are compared to each other. The SSIM is calculated between different image sets averaged over 20 000 images, as presented in figure 5.15. The SSIM configurations represent the Monte Carlo Geant4 data set and the corresponding GAN models: Geant4 vs. Geant4, Geant4 vs. Conv2D and Geant4 vs. 3DGAN. Furthermore, the SSIM is calculated between images of the same data set itself: Geant4 vs. Geant4, Conv2D vs. Conv2D and 3DGAN vs. 3DGAN to measure the similarities of the images within the same data set.

In figure 5.15, the configuration Geant4 vs. Geant4 represents the baseline. Ideally, the SSIM index of the generated image configurations should be as close as possible to it. In the Geant4 vs. Geant4 configuration, it can be seen that at low energies E_p the showers have more fluctuations impacting the image stronger than at higher energies. Therefore, the diversities of the images at lower energies are larger, and the values of the normalized SSIM index smaller while they are increasing towards higher energies.

It can be recognized that the SSIM values for Geant4 vs. Conv2D and Geant4 vs. 3DGAN are close to Geant4 vs. Geant4. This proves the similarity between GAN generated images and Geant4 images is of the same level as the similarity measured within Geant4 images. However, the generated images vs. generated images SSIM index is clearly higher, proving that generated images of both GAN models are more similar to each other than the Geant4 images. The physics meaning of this is that the generated shower images maintain less diversity than Geant4 samples. This result highlights a well-known weakness of GAN models, which tend to produce samples exhibiting a more limited diversity than present in the original data set. The ultimate of this effect would be mode collapse (detailed explained in section 4.2.1), where the GAN outputs the same shower image for a given input energy without variation. In our case, the generated images vary for the same input energy, but not as much as it would be desired. Most probable, the similarity of the results are within the precision level and expected for fast simulation models.

7. Number of Hits

Figure 5.16 provides the total number of hits for three different input particle energies spread over the full energy range (50, 200 and 500 GeV). Because the generated energy spectrum is continuous, in order to probe how the number of hits varies with the input particle energy, ± 2 GeV bins are built around the corresponding central energy value. The provided results represent the distribution over the 20 000 samples of the

validation set. The number of hits corresponds to how many cells one shower image contains above the threshold energy of 10^{-5} GeV. For better visualization, the number of hits are put into bins in a histogram in figure 5.16. It can be observed that the GAN models are capable of reproducing the Geant4 distribution for all energy levels. Both the mean and the width of the distributions agree with Geant4, as confirmed by the statistical values of the distributions in table 5.5.

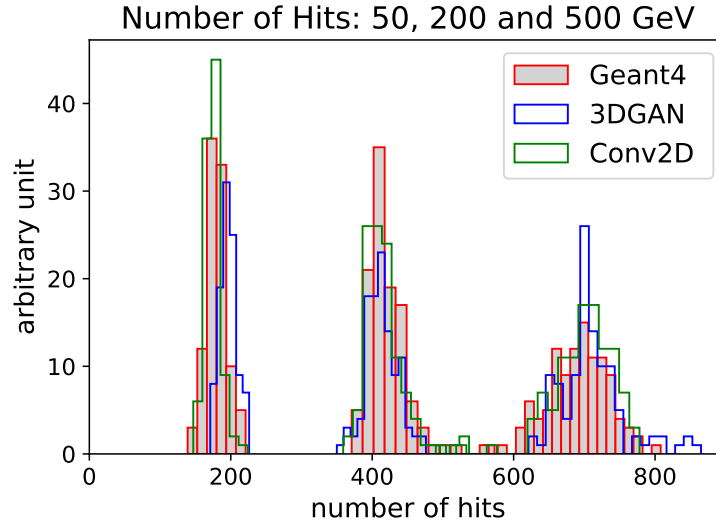


Figure 5.16: The number of hits above the 10^{-5} GeV single cell energy threshold for three different primary particle energies (50, 200 and 500 GeV). Each sample contains the distribution of the number of hits analyzed in 20 000 images.

Table 5.5: The means μ and the standard deviations of the number of hits distributions for each model.

Model	$\mu_{50 \text{ GeV}}$	$\mu_{200 \text{ GeV}}$	$\mu_{500 \text{ GeV}}$
Geant4	179 ± 14	421 ± 30	690 ± 44
3DGAN	196 ± 12	413 ± 22	710 ± 45
Conv2D	175 ± 13	413 ± 30	713 ± 46

8. Energy Sum

The last physics metric analyzed is the total deposited energy, or the energy sum metric, which is visualized in figure 5.17. It is generated for the same energy levels (50, 200 and 500 GeV) as the number of hits plotted before. The energy sum represents the visible energy E_λ measured inside the calorimeter. For this, the pixel energy values of all cells are summed up. However, only cell energies above the threshold energy

of 10^{-5} GeV are taken into account. The visible energy measured from the sampling calorimeter is naturally far lower than the primary input energy due to the sampling design of the calorimeter. Both GAN models maintain energy sum distributions similar to the ones of Geant4 for all energy levels. The shape of the 500 GeV distributions look considerably different and, visually, the distribution mean and width of the GAN models are lower than the one of Geant4. However, this difference is marginal, as proven by the statistical values provided in table 5.6. The reason why the 500 GeV Geant4 distribution drops suddenly after the peak towards larger energy is due to the shower leakage effect. At some primary particle energy E_p , the calorimeter cannot contain the whole shower development anymore.

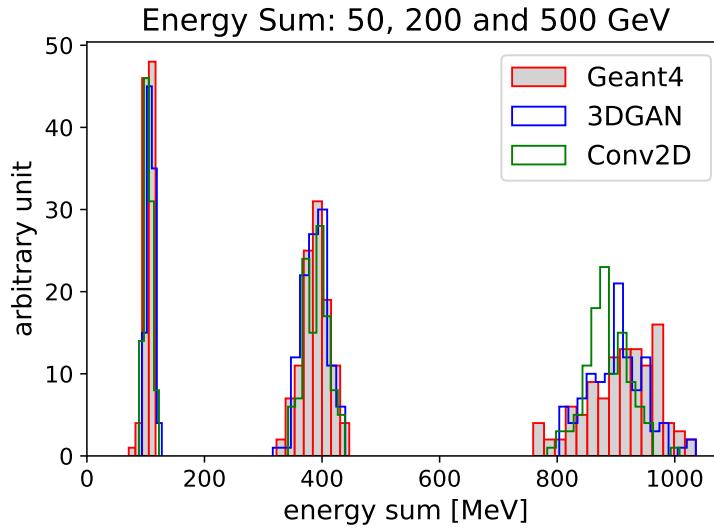


Figure 5.17: The energy sum metric for three different primary particle energies (50, 200 and 500 GeV). A 10^{-5} GeV single cell energy threshold is applied. Each sample contains the distribution of the number of hits analyzed in 20 000 images.

Table 5.6: The means μ and the standard deviations of the energy sum distributions for each model.

Model	$\mu_{50\text{ GeV}}$ in MeV	$\mu_{200\text{ GeV}}$ in MeV	$\mu_{500\text{ GeV}}$ in MeV
Geant4	104 ± 8	389 ± 23	910 ± 61
3DGAN	108 ± 9	387 ± 22	901 ± 48
Conv2D	104 ± 9	389 ± 21	890 ± 40

To complete this, an additional physics accuracy metric which evaluates the pixel correlations is provided and discussed in the appendix A.4. This metric was only developed after the GAN study was finished. However, the Conv2D model performs good.

5.6 Discussion and Future Work of the New Convolutional 2D Model

In this chapter, novel Conv2D neural network architectures were developed to generate 3D images with only 2D operations, which has not been done before. In the evaluation, both the Conv2D models and the 3DGAN model perform well and are capable to generate realistic shower images. When considering all the introduced physics accuracy metrics, the new Conv2D G1 three branches model is performing in most metrics superior to the 3DGAN model. In particular, in the shower shape metrics, the MSE_p and the mean energy deposition metrics. However, also the 3DGAN model is performing overall well. It merely lacks some accuracy when the particular details are examined. The largest recognized difference to Geant4 of the employed GAN approach is that both GAN models exhibit a smaller image diversity than Geant4, measured by the SSIM index. For resolving this, additional research should be performed to investigate, understand and overcome this issue. However, as discussed, the problem of less diverse images is a familiar GAN problem, not only related to our HEP calorimeter use case. Furthermore, some subsequent research should investigate why both GAN models, especially the Conv2D model, are off at lower cell energies in the single cell energy deposition metrics and why the GAN models fail to simulate the leakage effect seen at larger energies. Most probably, the statistics of the highest energies in the training sample is too low to generate the trend in the data.

As discussed, a few percent deviations of physics accuracy for future fast simulation approaches are expected and accepted for replacing the current Geant4 Monte Carlo simulation. However, exact values for the accepted deviations are still to be defined. This minimal accuracy trade-off comes, with an enormous possible 158 000x speed-up of the new Conv2D model over Geant4. This speed-up is sufficient to overcome the expected shortage in hardware resource in the HL-LHC phase in 2029 and on-wards. Because of the tremendous speed-up combined with the high physics accuracy, GANs represent an excellent alternative to Geant4 Monte Carlo simulations.

In the future, the investigations of the GAN approach for calorimeter and other HEP simulations should be further extended, tested and evaluated. The following steps contain among others the extension of the data set dimensionality from 25x25x25 pixels as employed in this study to the full size of 51x51x25 pixels, where the calorimeter consists of 25 layers with 51x51 pixels per layer. For this extension, the proposed Conv2D architecture from this thesis comes handily. Due to the considerably lower hardware resource requirement of the Conv2D model, the training and inference will require less time. This results in a faster development process, enabling to find the best hyperparameters such as the network architecture, number of parameters in the network or the learning rate faster.

Subsequently, it would represent a major study, how the GAN model with Conv2D networks behaves for primary particles with different incident angles. In this research, all particles entered the calorimeter perpendicular with a 90° angle, which is not the case in reality. This new constrain can be achieved by implementing a further condition to the discriminator which exclusively evaluates the incident angle prediction. The following steps can be extending for different primary particle types and multiple incident particles at a time (pile-up). In this research, only electrons are employed as primary particles. However, for electromagnetic calorimeters, photons and positrons also represent possible primary particles.

Ultimately, this research must be expanded from electromagnetic calorimeters to hadronic calorimeters with different hadrons as primary particles. As already mentioned previously, due to the high luminosity in the HL-LHC, multiple particle collisions happen simultaneously. Therefore, pile-up events represent rather the rule than the exception. In current Geant4 simulations it is already delicate to distinguish the events of different primary particles, but in the HL-LHC phase with more than a hundred simultaneous pile-up events this becomes a critical topic, leading to even more complex simulations. With examining all the above-mentioned possibilities, there is a lot of research to be carried out in the following years. With the production and sufficient testing of future GAN approaches, it might become a tight time frame to provide GAN algorithms ready for "application mode" by the beginning of the HL-LHC phase. However, research is already ongoing in some of the aforementioned topics, supported by a "Fast Calorimeter Simulation Challenge" (May 2023) [174] to accelerate the development.

Thesis Objective Fulfilled:

The GAN model was successfully optimized for accuracy and speed-up by exploring several alternative convolutional 2D neural network architectures for 3D image generation, which has never been successfully done before. The Conv2D G1 model optimized for GPU showcased a tremendous **158 000x** speed-up compared to Geant4, while maintaining the required physical accuracy. To evaluate the accuracy, eight physics metrics were presented and discussed, in which the Conv2D G1 three branches model performed well. This outcome suggests that GAN models are an attractive alternative to Geant4 simulation.

Chapter 6

Reduced Precision Strategies

Reduced precision computing could be a possible way for more efficient large-scale deployment of the previously introduced Conv2D model. In this thesis, an approach for post-training quantization was studied to reduce the number of bits required to store the learned network weights. In this context, quantization means mapping the learned neural network parameters to a lower number of bits. These concepts will be explained in the following. Any improvement in simulation time and memory occupation is valuable for such large-scale calorimeter detector simulations and reduces hardware expenditures. This represents the main motivation for the reduced precision inference studies conducted in the course of this chapter.

To achieve higher throughput during inference, the size of the biases and weights of the trained neural network can be shrunk to a more limited precision. This approach is denoted as reduced precision computing, and the process of converting numbers from higher to lower precision formats is named quantization [15]. The standard number format in ML is floating point 32 (float32) also named single precision, specified in IEEE 754 [175]. It uses four bytes or 32 bits for representing a single number. The format float32 utilizes one bit for the sign, eight bits for the exponent and 23 bits for the fraction [175]. For illustration, the largest possible number of float32 is $2^{128} = 3.40 \cdot 10^{38}$. The format which requires only half of the bits of float32 is floating point 16 (float16), called half precision. It uses 16 bits, one bit for the sign, five bits for the exponent and ten bits for the fraction. The largest possible number of float16 is $2^{16} = 6.55 \cdot 10^4$. The smallest format in which the model is quantized in this research is integer-8 (int8) which requires, as the name indicates, eight bits (or one byte) for representing a single number. This is a quarter of the number of bits of the initial float32 format. Integer means only whole numbers without fraction are allowed. The number can be either signed int8 (sint8) with a range of $[-128, 127]$ or unsigned int8 (uint8) with a range of $[0, 255]$. Figure 6.1 provides graphically the float32, sint8 and

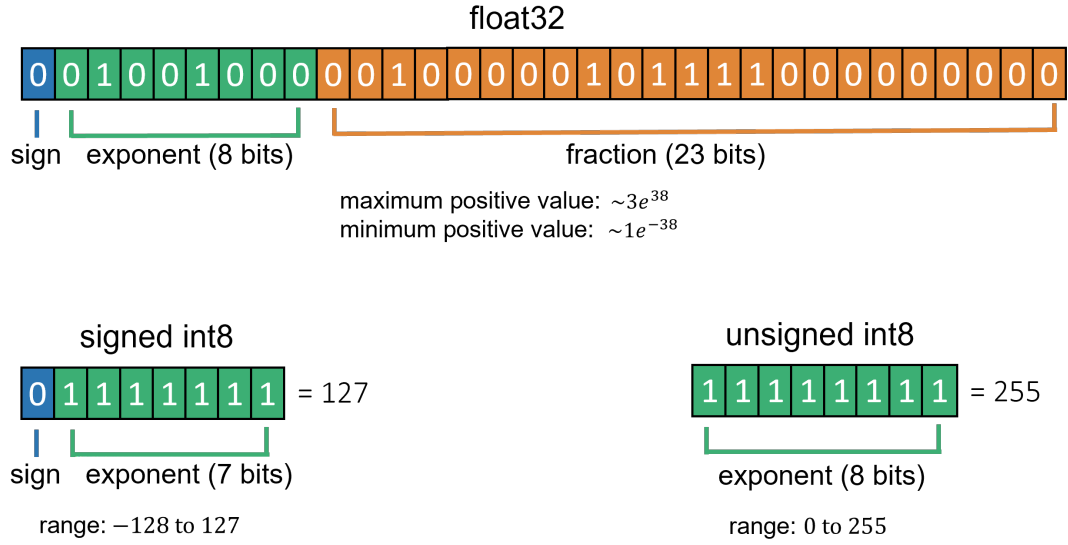


Figure 6.1: (top) Representation of the float32 format, (bottom left) signed int8 format and (bottom right) unsigned int8 format. The int8 formats require only a quarter of the number of bits of the float32 format.

uint8 formats for comparison.

This chapter comprises the following. First, the work about related approaches and results of other reduced precision researches are introduced. Next, the mathematical principles behind quantization to a reduced precision are explained. This is followed by the quantization tools employed in this research. Subsequently, the quantized models are in detail evaluated for physics accuracy, computational speed-up and memory. The chapter concludes with a discussion of the results and strategies for possible future researches. The content and the results of this chapter were in an effort of this thesis first published in Ref. [176]. Because of the very high interest in this topic, the work was selected for an extended publication in Ref. [177].

6.1 The Principle of Quantization to a Reduced Precision

In the following, the quantization of float32 to int8 is explained in detail. The less extreme reduction from float32 to float16 applies accordingly. In the post-training quantization process the numbers are scaled-down from a range of $[-3.4 \cdot 10^{38}, 3.4 \cdot 10^{38}]$ (float32) to a range of $[0, 255]$ for uint8 format or $[-128, 127]$ for sint8. In order to accomplish this, first the maximum absolute numbers of the weights B_w of the neural network are identified. These are required to obtain the boundaries for calculating

scaling factors which allow to use the full range of the int8 numbers after quantization. To measure the maxima B_w , inference must be run for a few batches with the trained model before quantizing it. The scaling factor can then be calculated with

$$S_{w, \text{unsigned}} = \frac{256}{B_w}, \quad (6.1)$$

for making use of the full range of uint8 numbers ($\in [0, 255]$) or

$$S_{w, \text{signed}} = \frac{128}{B_w}, \quad (6.2)$$

for sint8 numbers ($\in [-128, 127]$). The quantized weight Q_{wi} can then be calculated with

$$Q_{wi} = \text{round}(S_w \cdot I_i), \quad (6.3)$$

where the results are rounded to the closest integer number. I_i represents the initial not quantized weight, and the index i denotes the i^{th} weight.

6.2 Related Work in Quantization

Quantizing trained DL models to a reduced precision is being actively researched in recent years to allow, particularly, fast inference on mobile devices where the computational power is strongly constrained. In more general, with increasing model size and complexity, constraints on computing time and memory consumption become relevant for other architectures, CPUs and GPUs alike. Typically, DL models are trained in float32 precision on multi-node clusters, then quantized to lower precision formats and then loaded on light-weight devices such as edge-devices for executing inference. In most cases, quantization techniques are targeted to int8 format. Because int8 is supported by most modern hardware (lower formats frequently are not) and it maintains almost the equivalent level of accuracy. Even lower precision formats than int8 come typically with a drop of accuracy [178]. However, for our calorimeter use case, an accurate description of the generated image data is mandatory. Most of the existing reduced precision benchmarks represent classification problems [179]. As already discussed earlier, the application of generative tasks is usually more difficult than the application of classification tasks. This is also the case for the quantization of our calorimeter GAN model, and sint8 is taken as the limit.

There exist two major quantization techniques. The first represents post-training quantization, where the model is trained in full precision and quantized only after training. For quantization to a lower precision, the use of a calibration data set is mandatory to calculate the extreme (maximum, minimum) of the weights and biases. Post-training quantization represents a straightforward approach which leads in most

cases to a satisfying level of accuracy, therefore, it is commonly the first approach chosen [180]. Hence, for the study in this thesis post-training quantization is applied, and in the following section, it is described in more detail. Ref. [181] describes the post-training quantization process for quantizing convolutional networks down to 8-bits with solely operating small calibration data sets while avoiding overfitting. Another research was carried out in Ref. [182], where they applied different quantization techniques for quantizing pre-trained state-of-the-art neural networks down to 4-bit while only losing a few percent of accuracy.

An alternative method to the post-training quantization is the so-called quantization-aware training. There, the models are directly trained in lower precision formats. However, quantization-aware training is more complicated and, therefore, not considered in this research. The benefit of quantization-aware training is that the quantization errors which decrease the total loss of the model can be reduced by optimizing the neural network parameters accordingly. In Ref. [183], they compare the training of neural networks in various precision formats. In Ref. [184], they demonstrate a 2-4x training time improvement with the use of 8-bit floating point numbers above today's 32-bit systems. In the future, due to the technology enhancement in hardware and software, it will most probably become more straightforward to implement quantization-aware training methods.

One can go even further and train models with mixed precision formats: for example, operating both, float32 and int8. In this case, some weights are represented in int8 format to achieve a speed-up, but others are kept in float32 to maintain the required level of accuracy. Choosing which weights are represented in higher or lower precision is carried out by intelligent algorithms. Numerous studies have evaluated mixed precision for training [185, 186]. In particular, Ref. [187] successfully employs a mixed precision approach for training GANs.

6.3 Quantization Tools

Reduced precision computing is by now an established technique, and there exist many quantization tools. The primary constraint for quantizing the generator network into lower precision in this study is that in most of the quantization tools not all layers are integrated which are present in our neural network. In chapter 5 is already indicated that Conv3D layers were at the time this research was carried out not supported in any quantization tool. In order to explore and study reduced precision computing, it was first necessary to introduce a new neural network architecture which contains only the more common Conv2D layers, as described in chapter 5.

Additionally to Conv3D layers, there are other layers in our network which

cause difficulties in finding an appropriate quantization tool. These comprise Conv2D_Transpose layers and LeakyReLU activation functions (see layout in figure 5.1 and description in section 5.3.2), which are at this time not supported in most quantization tools. This is because quantization in DL is typically applied to classification problems that employ simpler neural network outputs (probability values of classes) compared to the complex 3D shower images from our generator network. However, as explained in section 5.4.1, the LeakyReLU activation functions are required to prevent the dying ReLU problem. Additionally, ordinary classification problems have no need for Conv2D_Transpose layers, because there is no upsampling in image size. Therefore, also Conv2D_Transpose layers are in most quantization tools not implemented at this time.

For employing quantized LeakyReLU layers, an additional problem appears in TensorFlow. TensorFlow supports by default only uint8 (unsigned int8) operations and no sint8 (signed int8) operations. The sint8 operations are not required for classification networks with standard ReLU activations and, therefore, sint8 operations are by default not implemented in TensorFlow. However, sint8 operations are necessary for our model, since LeakyReLU functions can generate negative values. Therefore, sint8 operations must be implemented within a patched TensorFlow version, provided by Intel. In the following, the tools used for quantizing our GAN model are introduced.

6.3.1 Intel Neural Compressor

The Intel Neural Compressor [16] (formerly known as Intel Low Precision Optimization Tool, LPOT) represents an open-source python library for quantizing DL models and running low precision inference across multiple frameworks. It utilizes the Intel oneAPI Deep Neural Network Library (oneDNN) [188], which contains building blocks for DL applications to improve the performance on Intel processors. In the initial research for this thesis in Ref. [176], the brand-new Intel Neural Compressor tool is for the first time employed in a real use-case. These studies were updated and considerably extended in a later publication in Ref. [177]. An overview of the Intel Neural Compressor infrastructure is provided in figure 6.2.

The Intel Neural Compressor represents a post-processing quantization tool which automatically optimizes DL models with low-precision recipes, in order to achieve the optimal objectives. The objectives for which Intel Neural Compressor tunes are inference time, memory usage and model accuracy [16]. For achieving these, it searches in the quantization process for the best order of magnitude for the scaling factors of an entire layer, instead of single pixels. A key feature is that Intel Neural Compressor can drop single "outlier" bits which are far away from other values within the corresponding layer and which would distort the results. With a recurrent refinement

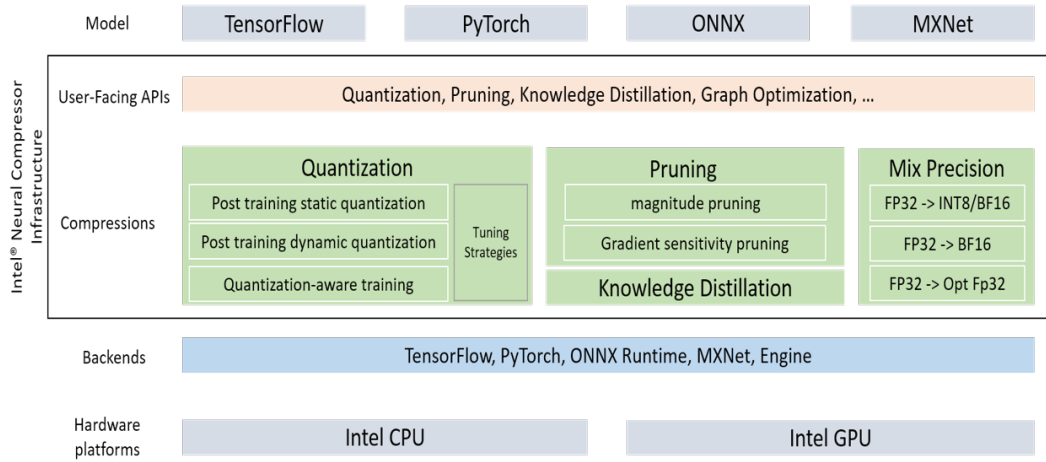


Figure 6.2: Overview of the Intel Neural Compressor quantization tool. The graphic is taken from Ref. [16].

of the topology and dropping outliers, the Intel Neural Compressor quantization remains a challenging process instead of a trivial one-step task compared to most other quantization tools. The quantization workflow of the Intel Neural Compressor that aids in increasing performance is displayed in figure 6.3.

Furthermore, in the automatic quantization process, the Intel Neural Compressor retains a few nodes of the neural network in float32 precision in order to increase or to maintain a superior accuracy. Therefore, the Intel Neural Compressor is a mixed precision post-processing tool which can operate in multiple data formats. This represents another advantage compared to other simpler quantization tools, which enables the Intel Neural Compressor to reach a superior accuracy. A comparison of the Intel Neural Compressor versus TensorFlow Lite is demonstrated by us in Ref. [176]. There, the Intel Neural Compressor int8 model performed in terms of accuracy much better than the TensorFlow Lite int8 model. For measuring the accuracy during the quantization process, the single value MSE_p validation metric is employed, which is introduced in the evaluation section 5.5 of the new Conv2D architectures.

As mentioned earlier, sint8 operations are by default not implemented in TensorFlow, which is employed for this research. This leads to the fact that the current quantization tools have not implemented quantized LeakyReLU functions. Therefore, in this study a customized TensorFlow version supporting sint8 operations is applied, which is provided by the collaborators from Intel. Furthermore, the Intel Neural Compressor team implemented explicitly for this study the quantized LeakyReLU function into the Intel Neural Compressor quantization tool. The Intel Neural Compressor with the patched TensorFlow version is at this time the only set-up which provides all required functionalities to quantize our model.

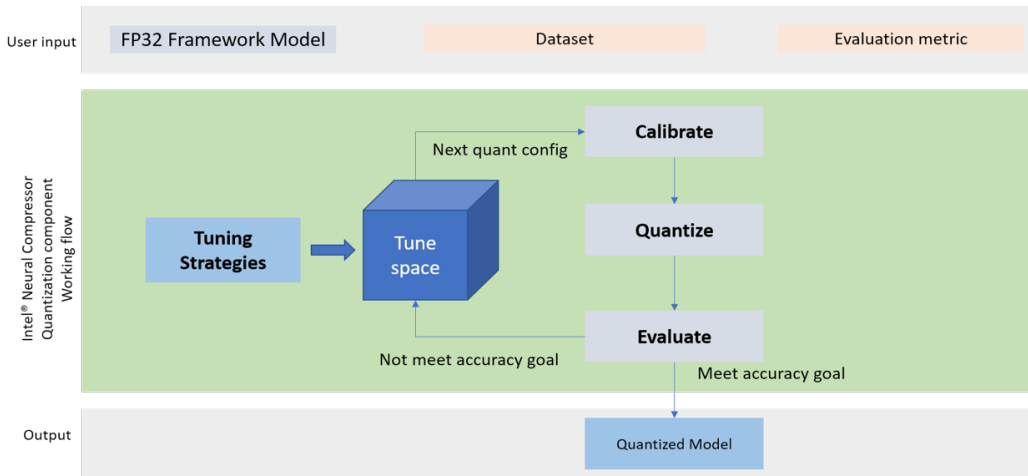


Figure 6.3: The workflow of the Intel Neural Compressor quantization tool. The graphic is taken from Ref. [16].

Previous to this study, the Intel Neural Compressor team applied their quantization tool to popular neural network architectures, such as ResNet [189] and Inception [190]. They achieved on average a two to three times speed-up in inference, while only losing less than a half percent in accuracy, see Ref. [191]. Due to the optimization process of the Intel Neural Compressor, they are in some cases even capable of increasing the accuracy for the quantized model in comparison to the initial not quantized model.

In many neural network architectures, a set of layers is multiple times repeated. This is also the case in our generator network, where multiple times the sequences of a Conv2D layer followed by a LeakyReLU and a BatchNorm layer are present. By combining or fusing these often-repeated layers of a network, the number of operations within the network can be significantly reduced. This further speeds-up the inference process. This layer fusion strategy can be chosen in the Intel Neural Compressor to accelerate the inference of quantized deep neural network models. In the used Intel Neural Compressor release v1.12, it is possible to fuse [Conv2D + LeakyReLU] together. Forthcoming releases should also support a full [Conv2D + LeakyReLU + BatchNorm] fuse, which could significantly improve the inference time performance.

6.3.2 TensorFlow Lite

The second tool employed for quantizing our model to a lower precision represents TensorFlow Lite [17]. It is part of the TensorFlow library and is created for deploying ML models on mobile and Internet of Things (IoT) devices. IoT devices are wireless connected devices to allow permanent interaction and autonomous intelligent deci-

sion making [192]. It contains packages for converting TensorFlow models to TensorFlow Lite models and packages to quantize and deploy models. Furthermore, it can quantize networks layer-wise from float32 into float16 and int8.

Because TensorFlow does not support sint8 operations, TensorFlow Lite does not support quantized LeakyReLU activation functions. Additionally, TensorFlow Lite does not support Conv2D_transpose functions. Therefore, in TensorFlow Lite, the respective not supported layers are simply operated in float32 format. This leads to an additional dequantization layer before, and to an additional quantization layer after, every not supported layer, which increases the inference time. A network equipped with the additional quantization and dequantization layers is for demonstration purposes visualized in figure 6.4. These additional quantization and dequantization layers are expected to introduce a significant impact on the inference time. Therefore, with employing TensorFlow Lite, there is no significant speed-up for the quantized model expected due to these additional layers.

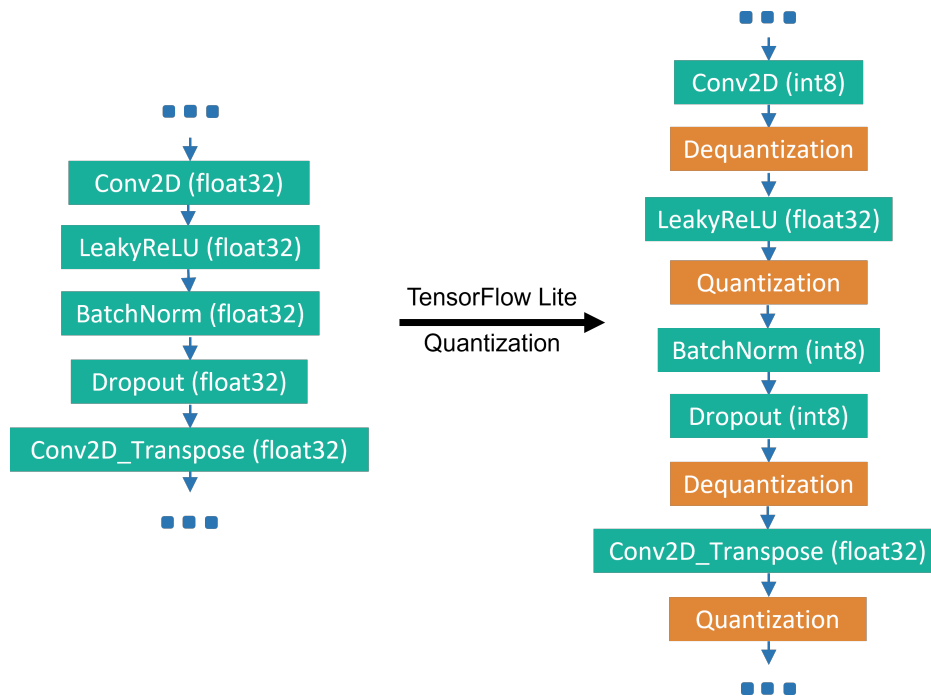


Figure 6.4: The TensorFlow Lite quantization of a network segment taken from our initial Conv2D generator network in float32 format (left) and quantized into int8 format (right). As visualized in orange, the quantization process introduces some additional quantization and dequantization layers at every not supported layer.

In any case, the compatible hardware for running the quantized TensorFlow Lite models is not available for this study. Therefore, TensorFlow Lite is only employed for

measuring the accuracy of the quantized models and not for measuring inference speed-up. Hence, the quantized TensorFlow Lite models are operated in this thesis only as comparison to the Intel Neural Compressor results in terms of physics accuracy to provide a reference quantization tool. In fact, the TensorFlow Lite models have a small advantage over the Intel Neural Compressor model. When in TensorFlow Lite some operations (e.g. Conv2D_Transpose layers) are executed in float32, the model theoretically can achieve better accuracy values by executing the computations at a higher precision.

Theoretically, there is an expected decrease in inference time and model memory size of a factor close to four due to the quantization from float32 to int8, because the number of bits for each single parameter is reduced by a factor of $32 \text{ bit} / 8 \text{ bit} = 4$. Undoubtedly, no decrease in model size of that factor will be achievable, because there are other aspects, like model architecture, which are engraved in the model memory and will not be affected by quantization. Additionally, the Intel Neural Compressor tool maintains some operations in float32 format, which also suppress the reduction in model memory size. Similar, in theory, there would be a decrease in inference time and model size of a factor of two for the quantized float16 model expected compared to the float32 model. Also this is not really achievable due to the various influences.

6.4 Evaluation of the Quantized Models

This section describes the evaluation process of the quantized Intel Neural Compressor model (in the following abbreviated as NC int8 model) and TensorFlow Lite models (abbreviated as TFLite float16 and TFLite int8). The computational performance, in terms of speed-up and occupied model memory reduction, and the physics accuracy results are evaluated.

The model deployed for the reduced precision research represents a preliminary version of the model introduced in chapter 5. The major difference in the neural network architecture is that for the reduced precision research, the last ReLU activation function was not included in the generator. The reason is that the reduced precision research was performed before the Conv2D model study was finalized and, the quantization was carried out on an intermediate model. Therefore, the accuracy of the initial not quantized float32 model in this chapter is different from the accuracy of the model evaluated in the Conv2D chapter 5. However, the important point here is only the difference in performance of the quantized model relative to the model without quantization. For inference time measurements, the missing ReLU layer is expected to deliver only a negligible contribution, and both the quantized and not quantized models do not contain the final ReLU layer.

6.4.1 Computational Evaluation

To evaluate the computational performance of the Intel NC int8 model, the inference time as well as the model memory consumption are measured and compared to the float32 model. It is aimed at a significant improvement in order to make the calorimeter simulations as computing resource efficient as possible. The computational throughput depends in practice on the model and the used hardware. The hardware tests in the reduced precision study are performed on Intel Ice Lake 2S Xeon 8380 processors, which provide 40 cores and 80 threads. Cores are physical and threads virtual CPU hardware components which manage the tasks. The Intel Ice Lake architecture supports reduced precision data formats and, is employed to benchmark the quantized NC int8 model versus the initial not quantized float32 model.

The number of generated shower images per second is measured for both models. For all hardware tests, the Python version 3.6.8, the customized TensorFlow version 2.3.0 and a batch size of 128 are employed. The model in this research represents the Conv2D G1 architecture introduced in the previous chapter and in the following denoted as Conv2D model. All tests include 20 warm-up batches. The inference is executed with multiple configurations of processor cores and streams. Multiple data streams mean that the inference process is run in multiple parallel processes on independent data on the CPU. The results are summarized in figure 6.5. It can be observed that for all configurations, the int8 model (in orange) generates more shower images per second than the float32 model (in blue), resulting in a higher throughput. The number of shower images per second rises for the configurations with more cores and streams, as expected (from left to right in figure 6.5). However, in the last configuration with 80 cores and 20 streams, the number of shower images per second drops slightly. Most probably because the memory bandwidth limit is exceeded, and the CPU becomes oversubscribed. This phenomenon is already experienced and investigated in our previous research in Ref. [176].

The best result in terms of number of shower images per second is accomplished with the configuration of 80 cores and 16 streams, where the NC int8 model is capable to generate 7 309 shower images per second. Compared to the float32 model with 2 845 shower images per second, the int8 model is a factor of 2.6 times faster in inference, as summarized in table 6.1. With this configuration, the int8 model is 124 200 times faster than the Geant4 simulation, which is aimed to be replaced. For the Geant4 simulation time, it is referred to the previous measurement performed in Ref. [193], where it took 17 seconds for generating a single shower image on an Intel Xeon processor (similar than in the previous Conv2D chapter 5.5). The largest speed-up of int8 vs. float32 is achieved with the configuration of 80 cores and 20 streams, where a speed-up of 3.0x is measured. This results in an inference time of 0.35 ms for the float32 model and 0.14 ms for the int8 model.

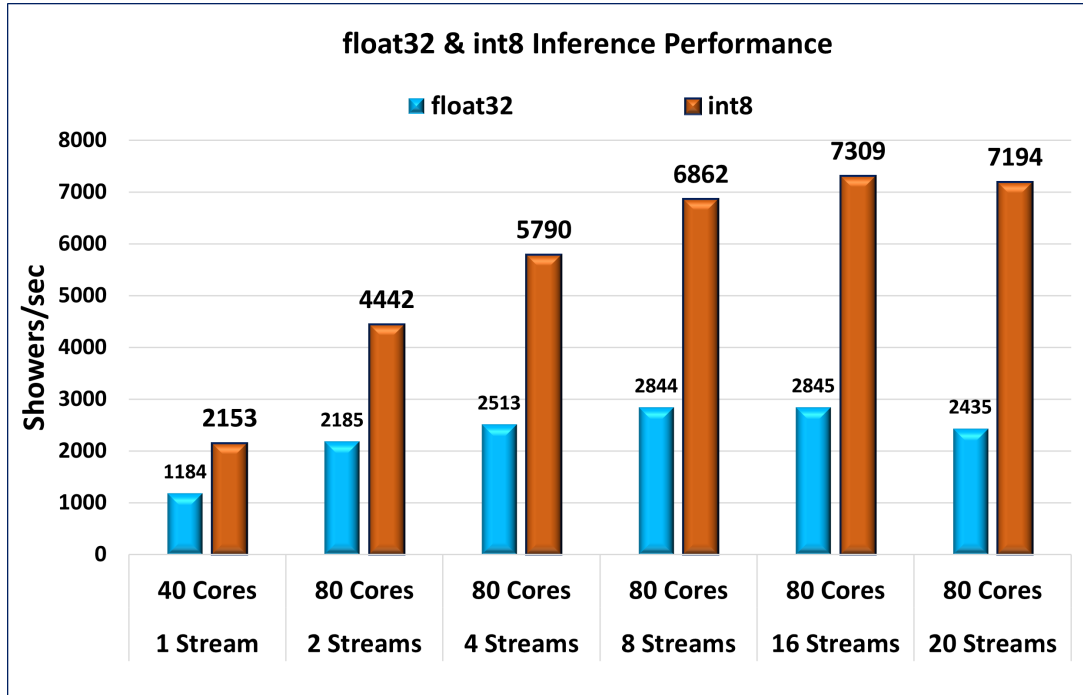


Figure 6.5: Inference throughput in shower images per seconds for the initial float32 (blue) and the quantized NC int8 model (red). Multiple configurations in terms of number of streams and cores are provided.

It is worth to note that in chapter 5, a larger speed-up is achieved by the not quantized float32 model versus Geant4. The reason for this is that in chapter 5 the inference is run on a GPU instead of a CPU, as it is done for the reduced precision research of this chapter. Usually, GPUs are superior for DL models when it comes to inference speed because of considerably more computational units, leading to a larger possible parallelism. GPUs are for certain operations, like simple linear algebra procedures present in DL algorithms, much faster [194]. However, this research is carried out in cooperation with Intel, therefore, it was decided to perform the reduced precision tests on their latest CPU hardware. Furthermore, Intel retains as manufacturer many more options and much more expertise in optimizing run-times on their hardware, which this study makes use of. Also, currently, most of the hardware devices in the WLCG remain CPUs. Therefore, a good CPU optimization is highly valuable.

The speed-up represents a result of savings in memory, bandwidth and faster calculations with the int8 arithmetic. Because the number format of int8 is four times smaller than float32, this leads to the expectation of a 4x speed-up. However, because of the quantization some additional operations are included in the model, which lead to a slight increase in computing time compared to the theoretical expectation. As an example for an additional operation, at the beginning of the model, the input must be

Table 6.1: The speed-ups of the best configurations of the quantized int8 Intel Neural Compressor model compared to float32 (denoted as int8/float32), the generated shower images per second of the int8 model and the speed-up of the int8 model versus Geant4 simulation.

Configuration	int8/float32	int8 [showers/sec]	int8/Geant4
80 Cores, 16 Streams	2.6x	7 309	124 200x
80 Cores, 20 Streams	3.0x	7 197	122 300x

quantized from float32 down to int8. Because the input vector to the model remains in float32 format, where the model requires int8. To quantize the input down, an additional quantization layer is added. The opposite appears at the end of the model, where a dequantization layer dequantizes the output of the model from int8 back to float32 format, such that it can be processed by following algorithms.

On the TensorFlow Lite web-page it is advertised that in practise a speed-up of a factor of three is expected for int8 inference [195]. Thus, their developers do also not anticipate the full 4x speed-up. Similar, the PyTorch team measures speed-ups in the order of 2-4x [196]. These ranges agree with our measurement. In the preliminary research in Ref. [176], the measured acceleration is only 1.7x of int8 vs. float32 and only a 67 000x speed-up vs. Geant4. Due to the advanced hardware and further model optimization, the speed-up could be increased to 3.0x. A speed-up of 3.0x therefore represents a pretty good state-of-the-art improvement.

In the following, further details about the run-times of some operations are provided. The run-times of the distinct operations of the quantized int8 model are summarized in figure 6.6. The Fused Batch Normalization – which represents the fusion of all BatchNorm operations together – alone yields about 30% of the total run-time: replacing this operation with a fused [Conv2D + LeakyReLU + BatchNorm] might result in another significant speed-up. The time required by the added quantization operation at the beginning of the model and the dequantization layer at the end of the model corresponds to 18.7% of the total run-time.

Roughly speaking, larger models might exhibit a greater inference acceleration in run-time because the quantization/dequantization steps have a smaller weight with respect to the other quantized network operations. As an example, ResNet-50 [189] maintains over 23 million parameters, Inception-v3 [143] over 24 million parameters, whereas our model contains merely around two million parameters. Being ten times smaller, our GAN network is more sensitive to the size of the quantization and dequantization steps with respect to the total run time. An additional effect arises due to the fact that a low-precision unit internally demands broader bandwidth, i.e. scratch memory bandwidth or cache bandwidth, and this demand can limit operations de-

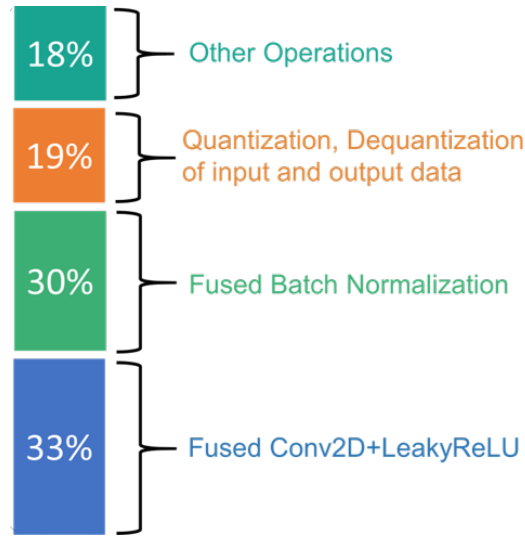


Figure 6.6: The percentage of the total inference run-times for the operations with the strongest impact.

pending on the specific hardware implementation. Therefore, practical implementations of low-precision in hardware and software are challenged to accelerate inference to the full ratio of type-width [comments from Intel developers].

In terms of memory occupation, the quantized NC int8 model decreases the total utilized physical memory by a factor **2.26x** with respect to the initial float32 (from 8.07 MB down to 3.57 MB), as summarized in table 6.2. The memory size of the quantized TFLite int8 model is 3.42 MB. Therefore, the NC model remains only 3.57 MB – 3.41 MB = 0.15 MB larger in model size while keeping a considerably higher accuracy (as it will be shown in the next subsection). The reason why the NC int8 model has a larger memory size is because it retains some nodes of the models in float32 format to maintain a good accuracy. Unfortunately, with the utilized Intel Neural Compressor release, it is not possible to output the exact fraction of nodes kept in float32 format. In the following, the physical accuracy of the shower images generated by the quantized models is evaluated.

Table 6.2: The hardware memory occupation of the initial float32 model and the quantized NC int8 and TFLite int8 model.

Model	Memory Occupation
float32	8.07 MB
NC int8	3.57 MB
TFLite int8	3.42 MB

6.4.2 Physics Accuracy Evaluation

For measuring the physics accuracy of the quantized models, the same metrics as for the Conv2D networks in chapter 5 are evaluated. In this section, only the results are discussed. Further information on how the validation metrics are produced can be acquired in section 5.5. The quantized models are compared in three different formats (NP int8, TFLite int8 and TFLite float16) to the baseline Conv2D GAN model (float32) and the Geant4 validation data set.

By comparing the physics accuracy of the intermediate Conv2D model provided in this chapter with the one of the best Conv2D architectures in chapter 5, the importance of the last ReLU layer in the architecture can clearly be observed. Furthermore, the model was only trained for events with primary particle energies E_p within the interval [100, 510] GeV.

1. Mean Squared Error Performance Measure

The MSE_p defined in section 5.5.2 is again used to quantify the quality of the generated showers. The Intel NC model is able to optimize the quantization with respect to a given metric. The MSE_p performance metric is a natural choice here to allow the tool to reach optimal performance.

Table 6.3: The average MSE_p and the standard deviation of ten inference runs with 20 000 samples each. The lower MSE_p , the better the accuracy.

Model	MSE_p
float32	0.0620 ± 0.0012
NC int8	0.0529 ± 0.0018
TFLite float16	0.2539 ± 0.0038
TFLite int8	0.3405 ± 0.0050

The MSE_p values of the models are provided in table 6.3. The mean and the standard deviation of the MSE_p in the table are created by running ten times inference with 20 000 samples each. It can be noted, the MSE_p for the quantized NC int8 model (0.0529) is even statistically significant lower than the MSE_p of the initial float32 model (0.0620). This results in a better accuracy of the NC int8 model. Usually, the accuracy of models drops during quantization. In our case, it is possible to accomplish a better accuracy because, the Intel Neural Compressor tool optimized during the accuracy driven quantization process for the single validation number, as mentioned before. If the MSE_p of the Intel Neural Compressor model is compared to the ones of the TFLite models, it can be clearly observed that this optimization is key for maintaining

a reasonable accuracy after quantizing the model. Because both TFLite models have a considerably larger MSE_p values (0.2539 for TFLite float16 and 0.3405 for TFLite int8), resulting in a worse accuracy.

2. Shower Shape Results

Figure 6.7 visualizes the shower shape metrics for the various models. All models are compared to the Geant4 prediction in red. The blue curves represent the shower shapes of the initial float32 model without any quantization, the green curves correspond to the shower shapes created by the NC int8 model, the orange curves are from the quantized TFLite float16 model and the pink ones from the quantized TFLite int8 model. In the linear energy deposition shower shape plots in figure 6.7 on the left, the not quantized float32 prediction is close to the Geant4 validation data. Also, the quantized NC int8 model does not deviate recognizable from Geant4. The TFLite float16 model exhibits some minor deviations, and the TFLite int8 model is clearly the worst.

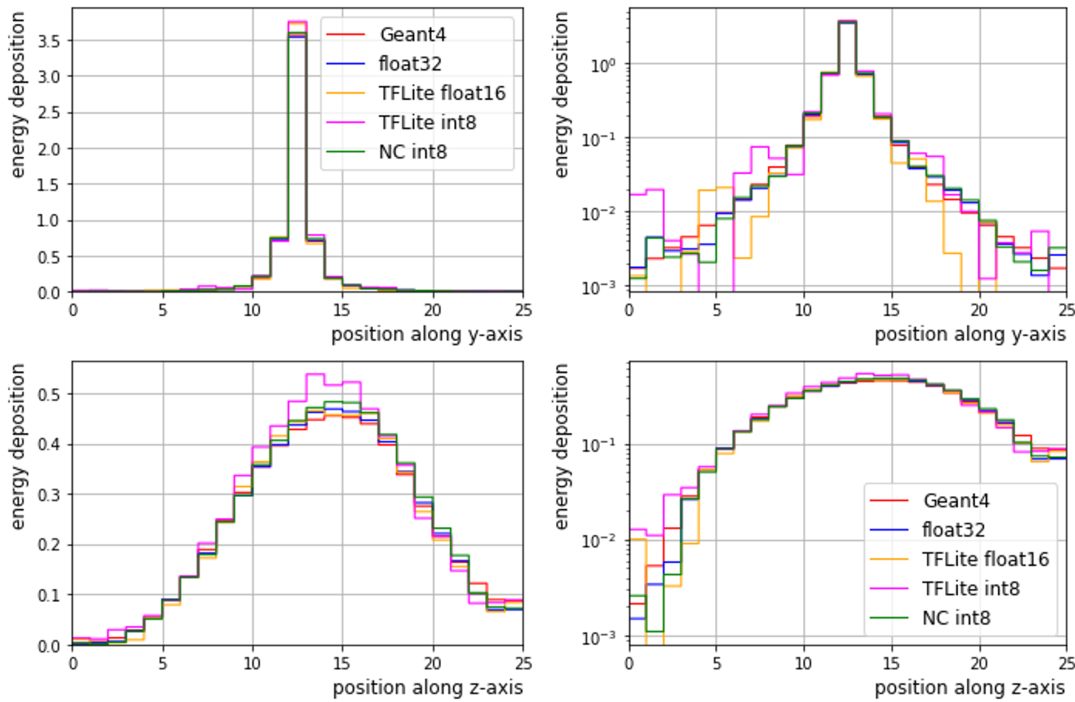


Figure 6.7: Particle shower distributions along the y - (top) and z -axes (bottom) in linear (left) and logarithmic (right) scale. The Geant4 prediction, the float32 model and the quantized models are provided.

In figure 6.7 on the right, the logarithmic representation is examined to gain a deeper insight into the lower energies at the distribution tails. In the y -axis shower shape plot, it can be observed that the TFLite float16, and especially the TFLite int8 model, struggle with the low energies at the distribution tails. On the other side, the NC int8 model performs also well in the logarithmic shower shape plots, even for the extremely low energetic distribution tails.

Table 6.4: Statistic values for mean and standard deviation (STD) for the various models of the shower shape distributions from figure 6.7.

Model	Mean	STD
y -axis Geant4	12.00	1.45
y -axis float32	12.02	1.46
y -axis NC int8	12.03	1.45
y -axis TFLite float16	12.05	1.13
y -axis TFLite int8	12.10	1.19
z -axis Geant4	13.85	4.62
z -axis float32	13.81	4.50
z -axis NC int8	13.91	4.48
z -axis TFLite float16	13.78	4.50
z -axis TFLite int8	14.25	4.62

Table 6.4 summarizes the statistical values mean and standard deviation for the particle shower distributions from figure 6.7. The statistical values of all models agree with what is visually observed in figure 6.7. With both, the visual shower shape evaluation and the statistical evaluation, it can be clearly recognized that the NC int8 model performs better than both TensorFlow Lite models. In fact, the NC int8 model does not reveal a significant deviation and, therefore, no accuracy loss. Furthermore, the results of the shower shape metrics agree thoroughly with the results of the single validation MSE_p numbers above.

3. Pixel-Wise Image Comparison

In order to evaluate the possible impact of quantization on the image quality, in addition to the validation metrics from the Conv2D chapter 5, a pixel-wise image comparison is carried out. This is done by fixing the latent input vector into the generator networks and generating one synthetic image for all models. Then element-wise, the entries of the single pixels of the generated image of the quantized model are subtracted from the initial not quantized float32 model. Lastly, the absolute values per image are summed up. The results are averaged for the $N = 20\,000$ images of the validation data set to receive the pixel-wise image comparison measure. The equation

for the pixel-wise comparison Z is:

$$Z = \frac{1}{N} \sum_{l=0}^{N-1} \sum_{i,j,k=0}^{24} |X_{i,j,k,l}^{\text{float32}} - X_{i,j,k,l}^{\text{quantized}}|, \quad (6.4)$$

where $X^{\text{quantized}}$ can be replaced by the corresponding images of the quantized models (NC int8, TFLite float16 and TFLite int8). The index l iterates over the corresponding image set and i, j, k over the individual pixels from the three spacial axes from pixel 0 to pixel 24. All quantized models are measured with respect to the not quantized float32 model from which they are derived.

Because the quantized models are all originating from the same initial not quantized float32 model, this metric measures the difference between the generated images of the distinct models with respect to the float32 model. With the corresponding distances, it can be directly evaluated how different the created images are compared to the original float32 model. Ideally, the distance would be low, which results in a minor accuracy loss. The smaller the pixel-wise validation value Z , the closer the images are to the initial model and the better the accuracy. The results are presented in table 6.5. It can be noted that the TFLite float16 model performs best, closely followed by the NC int8 model. The NC int8 model is approximately three times better than the TFLite int8 model in the pixel-wise comparison.

Table 6.5: Pixel-wise image comparison validation values Z for the different quantized models compared to the initial float32 model. The lower the value Z , the better the accuracy.

Model	Z
NC int8	1.550 ± 0.191
TFLite float16	1.330 ± 0.291
TFLite int8	4.054 ± 0.721

In the following, the physics accuracy is analyzed in more detail. Because the TFLite models perform worse than the NC int8, they are not further shown in order not to overload the plots.

4. Example 3D Shower Images

Figure 6.8 visualizes example 3D shower images. The shower images are generated with the same input vector for the float32 and the NC int8 model for a 400 GeV electron. From a first impression, the shower images look similar. Visually, it could be that the quantized NC int8 model may have fewer low energy pixels around the shower

center compared to the initial float32 model. But this might purely be a subjective visual impression or the result of statistical effects. For observing this further, the following physics validation metrics are evaluated.

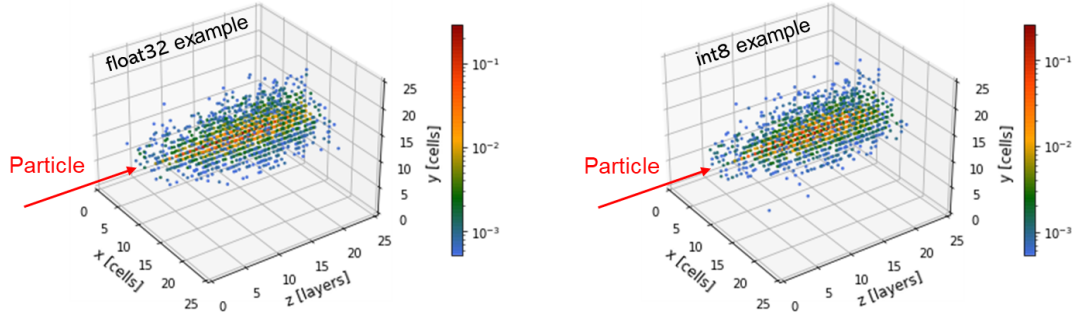


Figure 6.8: Example 3D shower images of the not quantized float32 Conv2D generator (left) and the quantized NC int8 generator (right). The primary particle with 400 GeV enters the calorimeter from the left in the center of the x - and y -axis ($x = y = 13$) at $z = 0$.

5. Mean Energy Deposition

The mean energy deposition inside the calorimeter is provided in figure 6.9 (left) with respect to the incident electron energy E_p . The upper plot shows the root-mean-square energy μ_{90} (as defined in 5.5.2) and the lower plot the relative error. Visually, there is no observable difference between the mean energy deposition of the float32 and the NC int8 model. Both are within a 2% range to Geant4. It should be noted, deviations up to a few percent compared to Geant4 are expected for fast simulation approaches, as already stated in the previous chapter.

In the right plot in figure 6.9, the relative width σ_{90}/μ_{90} is provided, where σ_{90} represents the standard deviation of the 90% core of the distribution. The quantized model exhibits a relative width in a similar range as the float32 model. Therefore, in the relative width metrics, no decrease in accuracy is visible for the quantized NC int8 model.

6. Cell Energy Deposition

Figure 6.10 contains the single cell energy depositions for 20 000 shower images. The plot provides on the x -axis the energy deposition of a single cell and on the y -axis the

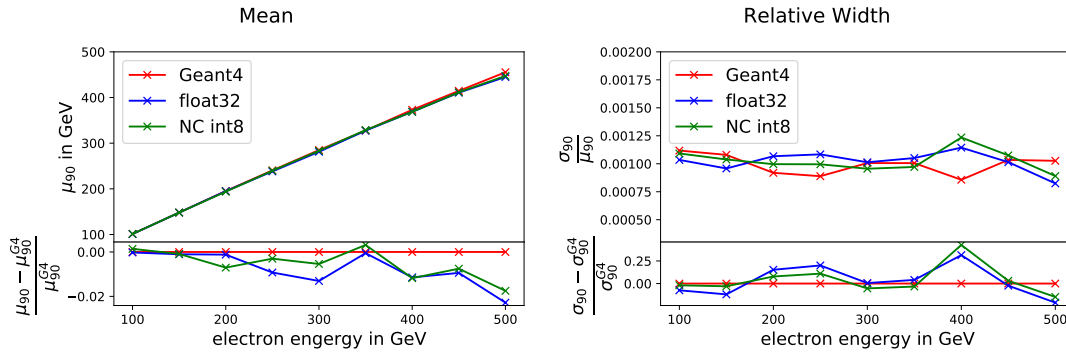


Figure 6.9: Deposited energy mean μ_{90} (left) and relative width σ_{90}/μ_{90} (right). The lower panels provide the relative deviations compared to Geant4.

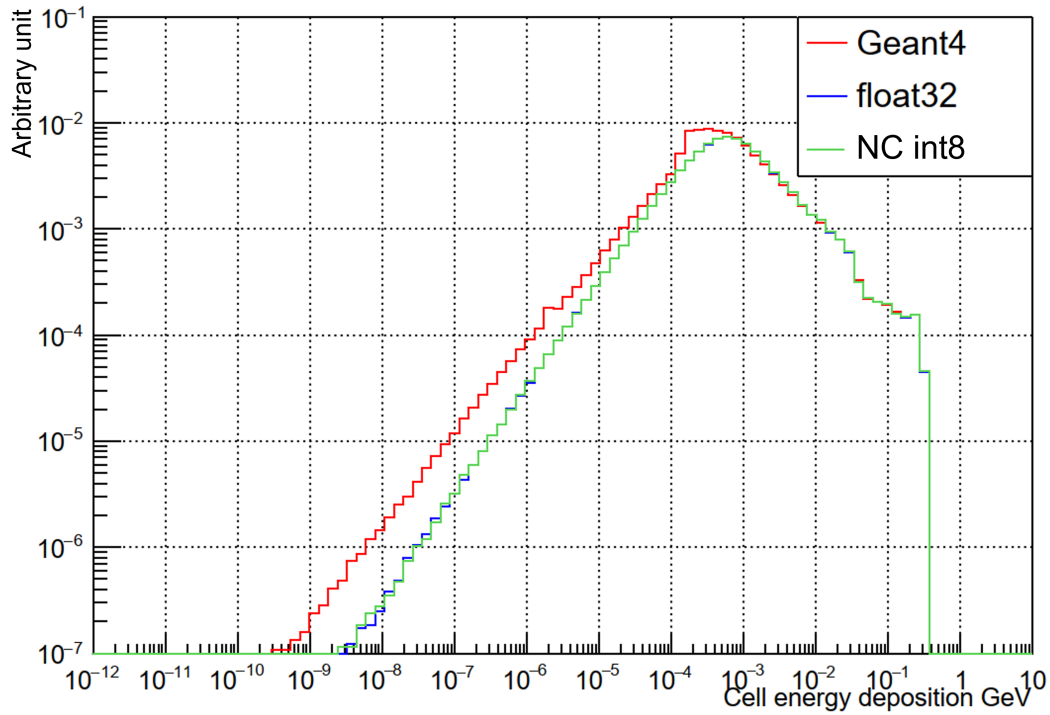


Figure 6.10: The single cell energy deposition metric of Geant4, the float32 and the NC int8 model.

number of cells which contain the corresponding energy in an arbitrary unit. In this plot, it is important that the NC int8 model is close to the float32 model. It is observed that the high energy hits are well described by both models in comparison to Geant4.

However, for lower cell energies below 1 MeV on the x-axis, the float32 model fails to correctly describe the expected distribution. This effect is already observed and further discussed for the new Conv2D GAN models in section 5.5. The NC int8 model behaves similar to the float32 model. This means, there is no decrease in the accuracy of the cell energy deposition metric due to the quantization process.

7. Number of Hits

Figure 6.11 visualizes the total number of hits for three different input particle energies (100, 300 and 500 GeV). As it can be recognized, already the float32 model completely fails to correctly reproduce the distributions. Therefore, it is difficult to draw conclusions for the NC int8 model, which does not provide a visually observable drop in accuracy with respect to the float32 model.

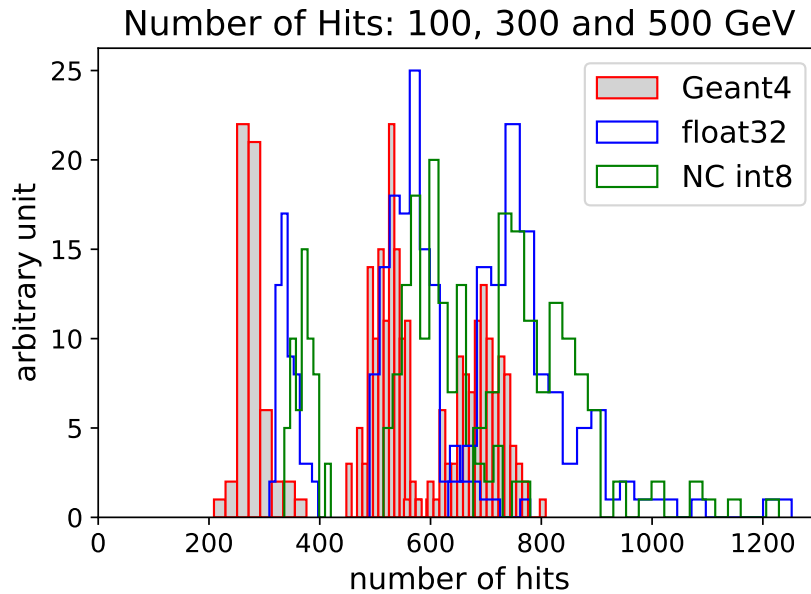


Figure 6.11: The number of hits for three different primary particles energies (100, 300 and 500 GeV). Each sample contains the distribution of the number of hits analyzed in 20 000 images.

This metric clearly highlights the difference in the Conv2D float32 model employed in the reduced precision research compared to the Conv2D model from the previous chapter, which includes a ReLU layer at the end of the generator. The model with the ReLU layer can correctly reproduce the number of hits plot (figure 5.16), providing a superior accuracy.

8. Energy Sum

The energy sum metric represents the sum of the energies of all pixels of the shower image and corresponds to the total measured energy for one event. In figure 6.12, the energy sum is displayed for the events with the primary particle energy E_p of 100, 300 and 500 GeV. On the x-axis, the measured energy is plotted in MeV and on the y-axis the number of counts within an energy bin.

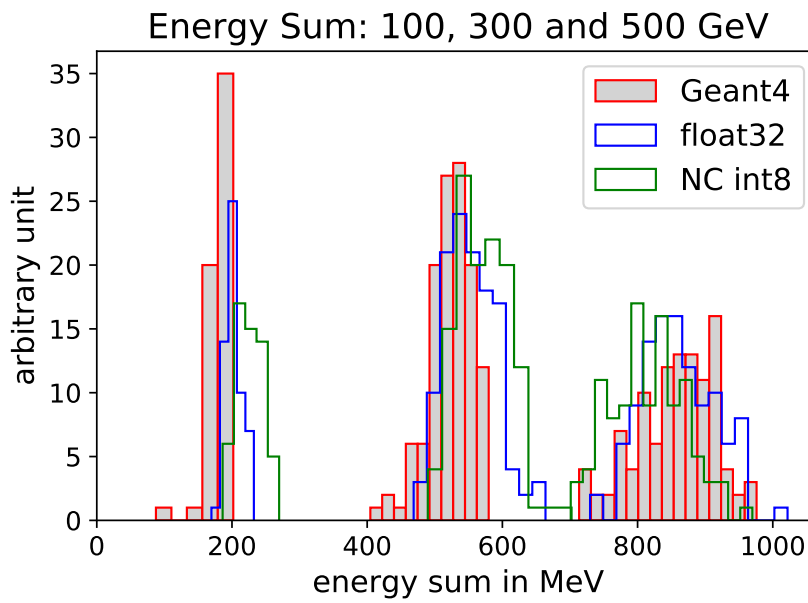


Figure 6.12: The energy sum metric corresponding to the total deposited energy for three different primary particles energies (100, 300 and 500 GeV). Each sample contains the distribution of the number of hits analyzed in 20 000 images.

The initial float32 distribution already indicates deviations from Geant4, again, most probably due to the missing ReLU layer. The NC int8 model remains close to the one of float32. However, visually slight deviations in the distribution shapes can be observed. The distribution parameters mean and standard deviation are provided in table 6.6. Only at the 100 GeV histogram the statistic values of the NC int8 model significantly disagree with the ones of the float32 model.

Overall, the quantization process minorly influences the measured energy sums. However, it is difficult to provide a clear statement, as the initial float32 model does already not perform very good.

Table 6.6: The means and the standard deviations of the energy sum distributions (100, 300 and 500 GeV) for each model.

Model	$\mu_{100 \text{ GeV}}$ in MeV	$\mu_{300 \text{ GeV}}$ in MeV	$\mu_{500 \text{ GeV}}$ in MeV
Geant4	180 ± 16	524 ± 32	857 ± 58
float32	202 ± 13	553 ± 38	861 ± 53
NC int8	225 ± 18	561 ± 38	854 ± 54

6.5 Discussion and Future Work of the Reduced Precision Research

In this chapter, a deep Conv2D neural network GAN model for a complex high energy physics calorimeter use-case is quantized into a reduced precision. As quantization tool, the MSE_p accuracy driven Neural Compressor from Intel is applied to quantize a pre-trained model from float32 down to the int8 format. With the quantized NC int8 model, a state-of-the-art inference time improvement of **3.0x** is achieved, compared to the not quantized float32 model on a Intel Ice Lake CPU. This results in a tremendous **120 000x** speed-up over the presently employed Geant4 simulation, which is aimed to be replaced by modern generative models. Additionally, the model memory size is reduced by a factor of **2.24**.

For the physics calorimeter use-case in this research, the highest possible accuracy is required. Therefore, the generated images are with multiple physics validation metrics investigated. Depending on the metric, the NC int8 model shows small or no deviations in comparison to the initial not quantized float32 model. On the other side, the quantized float16 and int8 TensorFlow Lite models perform both clearly worse than the NC int8 model. This indicates that an accuracy driven quantization tool such as the Intel Neural Compressor can maintain a reasonable physics accuracy.

All in all, the proposed reduced precision strategies for GAN models in this research achieved encouraging results. These findings can help to reduce the required computing resources for future high energy physics simulations and other generative models use-cases. Especially, in the case of calorimeter simulations, where a significant number of events must be simulated at the same time, reduced precision computing decreases the hardware requirements greatly.

A natural continuation of this research would be to consider repeating the reduced precision research with the final Conv2D model presented in the previous chapter to validate the results. For quantization, accuracy driven quantization tools are recommended, such as the Intel Neural Compressor used in this study, in order to maintain a high physics accuracy and to reach a reasonable inference speed-up. At present, quan-

tization remains a complex task with many obstacles. In the future, when the quantization tools are more developed and support all the layers present in our Conv2D model, the quantization process could be repeated with other quantization tools for comparison.

Thesis Objective Fulfilled:

One objective of this thesis was to demonstrate the optimization of the hardware resource consumption and runtime of the GAN model which is successfully fulfilled with the reduced precision research. With an accuracy driven quantization tool, a 3x speed-up and a 2.2x reduction in model memory could be achieved while maintaining the required accuracy.

Chapter 7

Quantum Computing

The novel technology of quantum computing is emerging and evolving rapidly. By controlling information units according to the rules of quantum mechanics, it has the potential to become a new paradigm of computing. In physics, a “quantum” represents the smallest possible unit. In the so-called second Quantum Revolution [197] presently ongoing, science has enabled to identify, control and manipulate these individual quanta, like photons, electrons, atoms and molecules, making entirely new technologies possible, for example also for computers. Quantum computing takes advantage of subatomic physic properties to process information. It is predicted that quantum computing can exponentially speed-up certain tasks compared to classical computing [198, 199]. This makes it an extremely interesting future computing approach. Furthermore, quantum computing may make it possible to solve problems that are even entirely intractable for classical computing methods [18]. Some computations are very inefficient on classical computers but are predicted to be efficient on quantum computers [18, 200]. In general, quantum computing might allow the learning of more complex correlation patterns which classical computers may not be able to reproduce efficiently [132]. Furthermore, the operation of quantum computers for large-scale problems might be more energy efficient due to faster computations and, thus reduce its carbon footprint [201, 202].

Because of the immense potential and market size of quantum computing, prominent IT companies, such as Google, IBM, Alibaba, Baidu, XANADU, Origin Quantum and many more are researching on the development of real quantum computers, as well as software frameworks for implementing quantum computing [23]. In HEP, the CERN Quantum Technology Initiative [20] coordinates the research in this area and provides a network for exchanging ideas and promoting innovation. At DESY in Germany, quantum technologies for quantum computing, quantum materials and quantum sensors are bundled in the new strategic direction of DESY Quantum [203]. Both, CERN

as well as DESY, became IBM hubs for quantum computing and are recognized driving forces to explore and widen the limitations of present quantum computers. Most of the employed simulation data in HEP initially originate from quantum mechanical processes. The simulation of these data on quantum computers, which equally relies on quantum mechanics, might reveal data correlations which classical models cannot reveal nor reproduce and, therefore, lead to more accurate results [21]. Furthermore, because of the enormous data sets operated in HEP, which sizes and complexity will even more increase in far future, the potentially expected speed-up of quantum computing can help to overcome the future computing challenges [19]. Therefore, also for HEP, quantum computing is considered as a promising future simulation approach.

This chapter is structured as follows. In the subsequent section, the elementary computing unit operated on quantum computers is introduced. Then, the quantum gates and circuits required to construct quantum computing algorithms are explained. Next, current physical quantum computers, their limits and challenges are briefly described. Subsequently, an overview of quantum machine learning and quantum generative models is given. Inspiration for the content of this chapter is derived from Ref. [204], because it provides a clear and comprehensible introduction.

7.1 Qubits: The Computing Units of Quantum Computers

Classical computers employ so-called bits as the basis unit to store information, whereas quantum computers manipulate quantum bits, in short "qubits." The information of classical bits is restricted to the discrete values 0 and 1. Qubits rely on quantum theory, which allows them to be in a superposition of multiple classical states. Possible basis states of qubits are $|0\rangle$ and $|1\rangle$, where the brackets around the numbers 0 and 1 correspond to the Dirac- or "bra-ket"-notation for quantum mechanic states. Hereby, a *bra* vector $\langle v|$ represents a row vector and a *ket* vector $|w\rangle$ a column vector, where w is the complex conjugate of v .

Due to superposition, the theoretical computing potential of quantum computers grows exponentially with an increasing number of qubits, whereas the power of classical transistors merely grows linearly with an increasing number of classical bits. The exponential growth is explained in the following: one qubit can exist in two states $|0\rangle$ and $|1\rangle$ simultaneously due to superposition. If two qubits are combined, there exist four ($2 \cdot 2 = 4$) quantum states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. Three qubits maintain $2 \cdot 2 \cdot 2 = 8$ states, etc. Due to superposition of multiple qubits, the size of the state space doubles for each additional qubit, leading to the exponential 2^n growth of the state space for an n -qubit system [204]. As a physics example, a two qubit register could be implemented by two spin 1/2 particles, where each particle can be in the eigenstate

spin-up $|\uparrow\rangle$ or spin-down $|\downarrow\rangle$ [205]:

$$\frac{1}{\sqrt{2}}[(|\downarrow\rangle + |\uparrow\rangle) \otimes (|\downarrow\rangle + |\uparrow\rangle)], \quad (7.1)$$

where $\otimes$ denotes the tensor product. In superposition, the qubits represent a combination of all possible states until they are observed. The operation of observing the state of a qubit is denoted as measurement. The measurement results in either 0 or 1 and destroys the superposition. The measurement is an irreversible process and if the value 0 or 1 is measured depends on the relative probabilities of the state, which will be explained in more detail in the following. More information about quantum measurements from the phase-space into a physical-space can be acquired in Ref. [206].

Theorists employ the Hilbert space $\mathcal{H}$ – a complex vector space – to describe the principles and laws of quantum mechanics [207]. Single qubits can be visualized by projecting their two-dimensional complex Hilbert space onto the surface of a unit sphere, which serves as a convenient visual representation [204]. In quantum language, the unit sphere is frequently designated as the Bloch sphere [208, 209], as shown in figure 7.1. If the state of the qubit is located on the North Pole of the Bloch sphere, it corre-

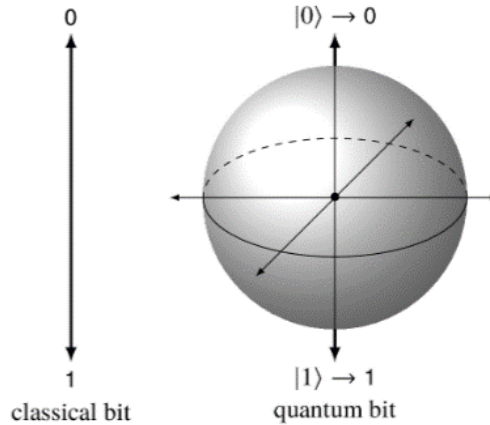


Figure 7.1: On the left, a classical bit which can take on either the value Zero or One. And right a quantum bit with the Bloch sphere representation, which can take on all values on the sphere surface. The figure is taken from Ref. [204].

sponds to state $|0\rangle$ and on the South Pole to state $|1\rangle$. If the state is somewhere on the sphere in between, it is in superposition of state $|0\rangle$ and state $|1\rangle$. The closer the state is to the north, the larger the probability to measure it in state $|0\rangle$. If it is placed exactly on the equator, then it represents a perfect randomness with an equal probability of 50% to measure either of the both ground states. Mathematically, the superposition of a state $|\Psi\rangle$ is represented by a linear combination of the basis vectors $|0\rangle$ and $|1\rangle$:

$$|\Psi\rangle = a|0\rangle + b|1\rangle, \quad (7.2)$$

where

$$|a|^2 + |b|^2 = 1 \quad (7.3)$$

due to unitary conditions. The probability of measuring state $|0\rangle$ is $|a|^2$ and the probability of measuring state $|1\rangle$ is $|b|^2$, where a and b are called the probability amplitudes. After measurement, the quantum state collapses into classical bit values of 0 or 1.

The quantum states of equation 7.2 can be rewritten in the frequently applied polar coordinates with the azimuthal angle φ and polar angle θ :

$$|\Psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + \sin\left(\frac{\theta}{2}\right) e^{i\varphi} |1\rangle. \quad (7.4)$$

The angles φ and θ are within the parameter range $0 \leq \varphi \leq 2\pi$ and $0 \leq \theta \leq \pi$. In figure 7.2, an example quantum state $|\Psi\rangle$ is visualized on the Bloch sphere with its corresponding polar coordinates. The conversion factors between equation 7.2 and 7.4 are $a = \cos(\theta/2)$ and $b = \sin(\theta/2)e^{i\varphi}$.

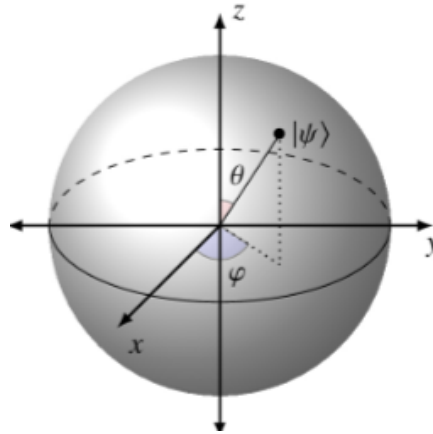


Figure 7.2: The quantum state $|\Psi\rangle$ on the Bloch sphere in polar coordinate representation. The figure is taken from Ref. [204].

There are other basis states defined instead of $|0\rangle$ and $|1\rangle$. Typically, the basis states are placed on the cartesian axes of the Bloch sphere visualized in figure 7.3. Three pairs of basis states are:

1. the Computational basis with state $|0\rangle$ and $|1\rangle$ on the Bloch sphere z-axis,
2. the Hadamard basis with state $|+\rangle$ and $|-\rangle$ on the Bloch sphere x-axis and
3. the Circular basis with state $|i\rangle$ and $|-i\rangle$ on the Bloch sphere y-axis.

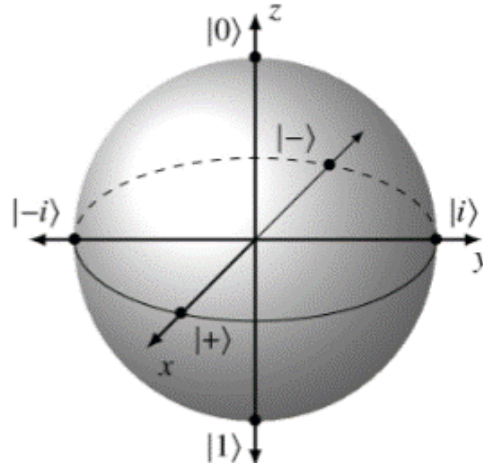


Figure 7.3: Three pairs of possible quantum basis states [204].

By convention, primarily the computational basis is used in the first place. Likewise, the computational basis is employed in this thesis.

To highlight the potential advantage of qubits over classical bits, two examples are provided. One classical so-called byte contains eight bits and can represent $2^8 = 256$ different numbers. However, one byte can merely hold one value at a time. With eight qubits, all 256 values can be represented simultaneously through superposition. In Ref. [204], another example is provided: the small molecule caffeine ($C_8H_{10}N_4O_2$) requires approximately 10^{48} classical bits for representing all its contained information. For comparison, today's storage sizes remain in the terabyte range, which represents only 10^{12} bits. In contrast, only 160 qubits would be required to represent the comprehensive information of a caffeine molecule ($2^{160} \approx 10^{48}$). Therefore, quantum computing has the potential for a better memory efficiency.

7.2 One Qubit Quantum Gates

In this section, quantum operations are introduced to manipulate the states of qubits on quantum devices. These operations are denoted as quantum gates and are analogues to the classical logic gates AND, OR, NOT, etc. All quantum gates are reversible operations [210], which are defined by a unitary matrix U . With the complex conjugate matrix $U^\dagger$, the computation of U can be revised. The measurement gate, from quantum information to classical, represents the only irreversible operation in quantum computing. That means, if the state of a qubit is measured to be either zero or one, the original probability amplitudes cannot be restored. For quantum computing, numerous gates exist. However, as for classical gates, many gates can be created by

a combination of only a few basic gates. In this section, selected quantum basic gates are introduced, which are used extensively in this thesis.

7.2.1 X-Gates

The simplest gate is the one-qubit quantum NOT or X -gate. Single qubits can be represented by two-dimensional complex *ket*-vectors. Therefore, all one-qubit quantum gates can be replaced with two-by-two matrices, including complex entries, which manipulate the state of qubits. The matrices of all gates are unitary. The matrix representation of the X -gate is the Pauli-X matrix [211]:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \quad (7.5)$$

The X -gate switches the amplitudes of the $|0\rangle$ and $|1\rangle$ states:

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle, \quad X|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle. \quad (7.6)$$

7.2.2 Hadamard Gates

The Hadamard gate (H) represents the second important quantum gate. The mathematical matrix representation of the H -gate is:

$$H = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \quad (7.7)$$

Hadamard gates map the basis states of a single qubit into superposition:

$$H(|0\rangle) = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle, \quad H(|1\rangle) = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle. \quad (7.8)$$

Therefore, the Hadamard gate manipulates the state from the computational basis $|0\rangle$ and $|1\rangle$ into the Hadamard basis $|+\rangle$ and $|-\rangle$.

Hadamard gates are among the most frequently applied gates in quantum computing. They are implemented at the beginning of circuits to put the qubits initialized in $|0\rangle$ state into superposition [204]. The Hadamard gate is its own inverse. Hence, applying the Hadamard gate twice results into the identity matrix $HH = H \cdot H^\dagger = I$.

For visualization of the Hadamard gate, the Bloch sphere is employed. In figure 7.4, the evolution of applying a Hadamard gate to various input vectors is visualized. The Hadamard gate can be thought of as a rotation around the diagonal of the $x + z$ -axis (180° rotation around the orange arrow in figure 7.4; the z -axis represents the vertical axis). The blue arrows indicate the initial states and the green arrows, with the matching color saturation, demonstrate the corresponding states after applying a Hadamard gate. Starting from the green arrows and applying a Hadamard gate rotates the quantum states back into the initial position.

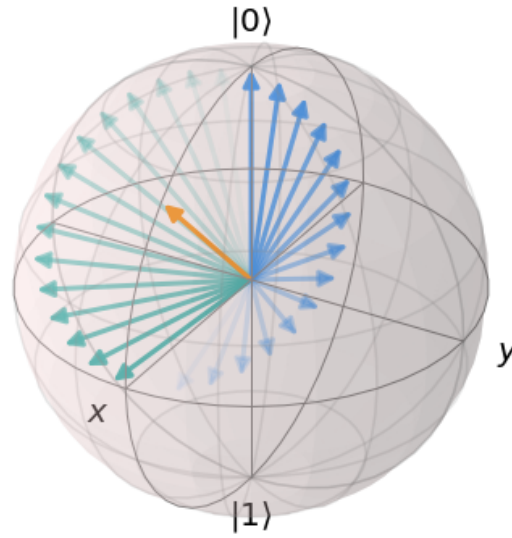


Figure 7.4: The operation principle of a Hadamard gate, visualized by the 180° rotation of vectors around the orange arrow.

7.2.3 Y-Rotational Gates

The parameterized Y-rotational gate (RY) performs a single-qubit rotation with the rotational angle θ (in radians) around the y -axis. The mathematical description of the gate in matrix notation is:

$$RY = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}. \quad (7.9)$$

The RY -gate is visualized in figure 7.5 for different rotational angles $\theta \in [0, 2\pi]$. If a state is initiated in $|0\rangle$, a rotation by $\pi/2$ results into the $|+\rangle$ state on the x -axis

and by π to the $|1\rangle$ state. Frequently, in trainable quantum circuits, the rotational angles θ correspond to the trainable parameters, which are randomly initialized and appropriately optimized during training.

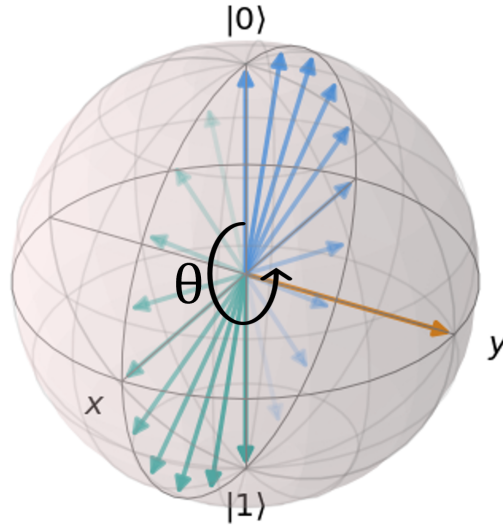


Figure 7.5: The rotation of a state around the black arrow (y-axis), carried out by the Y-rotational gate with the rotational angle θ .

7.3 Multiple Qubit Circuits

The second fundamental feature of quantum computing, alongside superposition, is the entanglement between multiple qubits. Entangled qubits are intrinsically linked to each other. With the measurement of one qubit, conclusions about the other(s) can be deduced [205]. In quantum algorithms, typically several qubits are operated to increase the contained information. This section describes the behavior of multiple qubits, the concept of entanglement and important two-qubit gates.

Mathematically, two qubits are represented by a tensor product, which concatenates two vector spaces. The tensor product of an n and m dimensional vector space results into $n \cdot m$ dimensions. Through this multiplication, the vector space grows fast in dimensions. For example, two 2×2 matrices multiplied with the tensor product result into a 4×4 matrix. The basis states of two qubit systems are $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$.

Two qubits are represented by the linear combination of the basis states:

$$(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) = a_{00} |00\rangle + a_{01} |01\rangle + a_{10} |10\rangle + a_{11} |11\rangle. \quad (7.10)$$

Similar as for the one-qubit case, due to normalization of the basis vectors the squared amplitudes a_{ij} must sum up to 1:

$$|a_{00}|^2 + |a_{01}|^2 + |a_{10}|^2 + |a_{11}|^2 = 1. \quad (7.11)$$

After the measurement, the qubits collapse irreversible into one of the basis states, according to its probabilities $|a_{ij}|^2$.

An example of a maximally entangled quantum state $|\psi\rangle$ of two qubits represents the so-called Bell state [212]:

$$|\psi\rangle = \frac{\sqrt{2}}{2} |00\rangle + \frac{\sqrt{2}}{2} |11\rangle. \quad (7.12)$$

In the Bell state, only the two states $|00\rangle$ and $|11\rangle$ exist, both with an equal probability of 50%. If the relationship of the Bell state is known, and only one qubit is measured, an immediate conclusion about the other state can be drawn. For example on an ideal quantum computer without any noise, when qubit one is measured in state $|0\rangle$, there is a 100% probability that qubit two is equally in state $|0\rangle$ and a zero-percent probability that qubit two is in state $|1\rangle$. Due to the entanglement, the qubits are correlated, and the measurement of one qubit determines the other one. The combination of entanglement and superposition represents a significant advantage of quantum over classical computing.

The general rule is, two-qubit quantum states are entangled if they cannot be expressed as a tensor product of two one-qubit states, meaning if they are linear independent [210]. This statement is explained with in the following. The tensor product of two one-qubit states can be written for example:

$$(a_1 |0\rangle + a_2 |1\rangle) \otimes (b_1 |0\rangle + b_2 |1\rangle) = a_1 b_1 |00\rangle + a_1 b_2 |01\rangle + a_2 b_1 |10\rangle + a_2 b_2 |11\rangle. \quad (7.13)$$

If the Bell state could be constructed, four relationships must be fulfilled:

$$a_1 b_1 = \frac{\sqrt{2}}{2}, \quad a_1 b_2 = 0, \quad a_2 b_1 = 0, \quad a_2 b_2 = \frac{\sqrt{2}}{2}. \quad (7.14)$$

The second condition $a_1 b_2 = 0$ can be fulfilled either if a_1 , b_2 or both are zero. However, if a_1 is zero, the first relationship $a_1 b_1 = \frac{\sqrt{2}}{2}$ cannot be constructed. Similar, if b_2 is zero, the forth relationship $a_2 b_2 = \frac{\sqrt{2}}{2}$ cannot be constructed. Therefore, the Bell state from equation 7.12 represents an entangled state, because it is impossible to divide it into a tensor product of one-qubit states.

7.3.1 Two Qubit Gates

One qubit gates operate 2x2 unitary matrices. For two qubits, the matrix dimension increases to 4x4. One of the most frequently applied two-qubit gates in quantum computing represents the CNOT (or CX) gate. As the name indicates, it remains a NOT gate, while "C" stands for "controlled". It has two qubit inputs and two qubit outputs. The first qubit represents the controller or condition qubit. It applies the NOT operation to the second qubit, the target qubit, only if the first qubit is in state $|1\rangle$. If the first qubit is in state $|0\rangle$ nothing happens. The four possibilities of the CX -gate operation are:

- $CX |00\rangle = |00\rangle,$
- $CX |01\rangle = |01\rangle,$
- $CX |10\rangle = |11\rangle,$
- $CX |11\rangle = |10\rangle.$

The matrix representation of the CX -gate is:

$$CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (7.15)$$

For example, the Bell state can be created by applying a H -gate to the first qubit and then placing a CX -gate between the two qubits, where the first qubit is the "controller" qubit. In literature, and throughout this thesis, the terms CX -gate and CNOT gate are used interchangeably.

In quantum circuits, additionally to the CX -gates, frequently controlled Z-gates (CZ) are applied. CZ -gates are similar to CX -gates, however, CZ -gates only affect the state where both qubits are in state $|1\rangle$, and it changes the sign (highlighted in red):

$$CZ(a |00\rangle + b |01\rangle + c |10\rangle + d |11\rangle) = a |00\rangle + b |01\rangle + c |10\rangle - d |11\rangle. \quad (7.16)$$

The matrix representation of the CZ -gate is:

$$CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \quad (7.17)$$

There is also a controlled Y-gate (CY) represented by the matrix:

$$CY = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{bmatrix}. \quad (7.18)$$

The symbols of the CX -, CY - and CZ -gates in quantum circuits are visualized in figure 7.6. The qubit zero (q_0), where the dot of the gates is placed, represents the "controller" qubit. The black horizontal lines are called (quantum) wires, which connect the gates to one qubit. They show the progress in time from left to right. It can be noted, in the CZ -gate representation (the right most gate in figure 7.6), it does not matter which of the qubits the "controller" qubit is, because both qubits are interchangeable. The mathematical proof for this is provided in the appendix B.1.

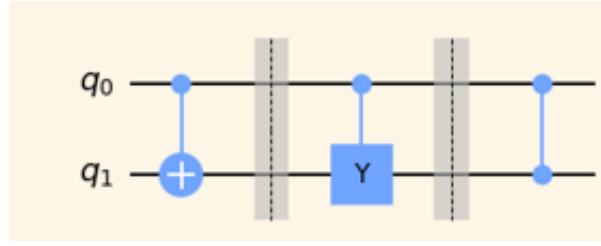


Figure 7.6: On the left, the CX -gate, in the middle the CY -gate and on the right the CZ -gate, as they are typically visualized in quantum circuits.

In today's quantum computing, typically only one- and two-qubit gates are applied, and no gates connecting more than two qubits. The reason for this is given by the hardware noise on current devices: the more qubits are involved, the higher the gate noise and the less reliable the results [213]. However, more than two qubits can be entangled to each other by applying a composition of multiple two-qubit entangling gates.

7.3.2 Quantum Circuits

Each quantum hardware operates a set of primary gates to physically manipulate the qubits. More complex gates can be generated by wiring-up a sequence of primary gates (especially CNOT plus single qubit rotation gates) by utilizing quantum circuits, similar as it is done in classical computing. The qubits employed for one operation are frequently collected into a system denoted as quantum register. For more complex algorithms, sometimes multiple quantum registers are applied. It is a convention to

initialize the qubits in state $|0\rangle$. The qubit can – if required – be transferred to state $|1\rangle$, by applying an X -gate or into equal superposition, by applying a H -gate.

A simple example circuit consisting of one qubit, one X -gate and a measurement operation is provided in figure 7.7. Depending on the use case, multiple visualization techniques are employed. The left comprehensive visualization exhibits the measurement and additionally that the quantum state of qubit q is measured to the classical bit 0 within a classical register c . For more complex circuits with multiple qubits, the qubits and classical bits are indexed with an iterative number. The middle visualization represents a simplification and neglects the classical register, because it is apparent that the quantum state is measured into a classical one. On the right, even the visualization of the measurement is neglected, because when a circuit is provided, usually a measurement operation is applied afterwards. These simplifications are useful when large and complex circuits with many qubits and gates are designed, to simplify the visualization.

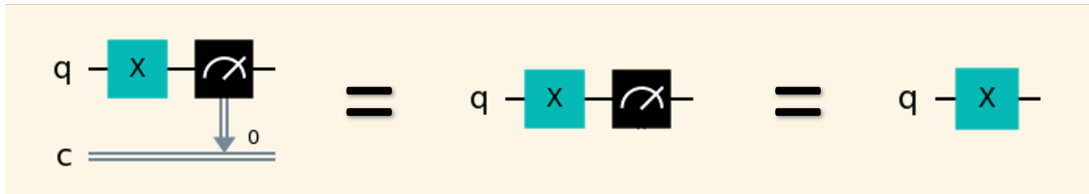


Figure 7.7: A simple example circuit consisting of only one qubit with one X -gate in three different representations. The black gate with the white gauge behind the X -gate symbolizes the quantum measurement gate.

Quantum circuits can be constructed with as many gates and qubits as desired. Usually, the maximum number of sequential gates applied to one qubit is denoted as the circuit depth and the number of qubits as the circuit width. The output of a quantum circuit with multiple qubits, after applying measurement operations, is a classical bit-string of zeros and ones.

7.4 Noise Intermediate Scale Quantum Devices

The development, construction and operation of quantum computers is a highly interdisciplinary field [214], which joins technologies of many physics and technology fields. Examples of different technologies are given at the end of this section. Depending on their physical realization, many of the current quantum computers are

operated at temperatures as close as possible to 0 K. This is chosen to suppress the environmental influences to the operations of the qubits and computations, and because the quantum computer employs superconductors [215]. To achieve these low temperatures, superfluid helium is operated as a cooling fluid [216].

The qubits in today's quantum computers do not represent accurate "logical" qubits, which can operate an indefinite number of gates, while maintaining the correct states [204]. Instead, they are denoted as "physical" qubits, which decay and lose their state after some time denoted as "coherence time" [217]. As the decay of qubits is unavoidable, it is attempted to delay the decoherence of the qubits as long as possible. Or, alternatively, to operate the gates in the quantum circuits as fast as possible. Physical qubits suffer from hardware errors, which cause imperfect operations and reduce the fidelity of the computation results. Furthermore, they are influenced by noise of the environment, for example of the earth magnetic field, cosmic rays and electronic components operated by the quantum computer itself [204]. In general, it is possible to create logical qubits on imperfect quantum computers. For this, one logical qubit requires the operation of many physical qubits (hundreds to thousands), combined with error correcting schemes in sophisticated circuit architectures [204].

IBM proposed a metric to determine the overall size and quality of near-term quantum computers called Quantum Volume [218]. This metric takes into account: the depth and width of executable quantum circuits, calibration errors, coherence times, connectivity between qubits, gate fidelity, and some more influential aspects [218]. In this thesis, the quantum simulations are operated with the IBM Qiskit library [219]. Furthermore, some experiments are executed on real IBM Quantum (IBMQ) devices. Due to the CERN Quantum Hub, premium access to IBMQ devices is available for this thesis.

In figure 7.8, the IBM roadmap of quantum devices is shown, as published in Ref. [220]. It can be noted that current quantum devices can merely operate hundreds of qubits. However, already in 2023, it is predicted to operate a thousand-qubit machine, and from 2026 onwards, tens of thousands of qubits. Because of the operations are limited to a hundred qubits with short coherence times and a limited qubit-to-qubit connectivity of current quantum devices, only the operation of circuits with a limited size is possible. Commonly, current quantum devices are denoted as Noisy Intermediate-Scale Quantum (NISQ) computers. Because of the small number of qubits, NISQ devices do not allow operating error correction techniques, and thus, no logical qubits, making them unsuitable for accurate general-purpose computations [221]. In the future, quantum computers are expected to become fault-tolerant, with the operation of millions of physical qubits. This results then in approximately thousands of logical qubits which could be employed to execute reasonable precise computations.

The characteristic values to present the noise level of current quantum devices are in

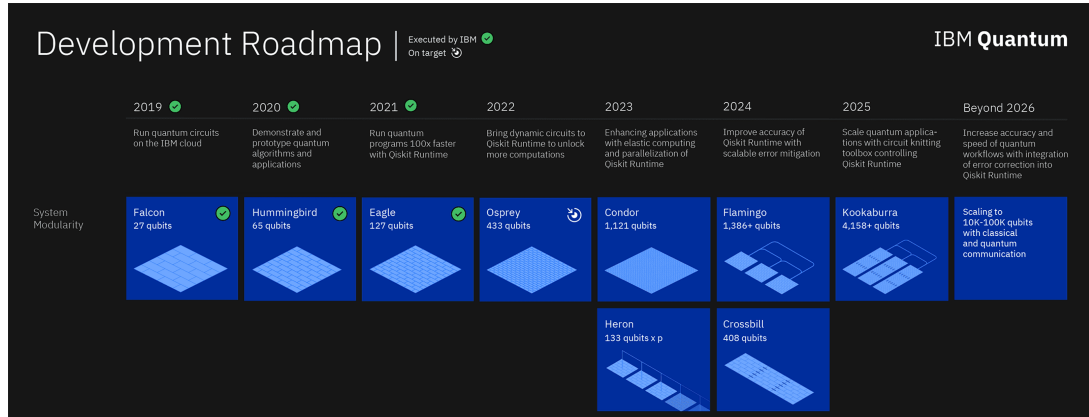


Figure 7.8: Development roadmap for the IBM quantum devices from Ref. [220] published in 2022. Among others, the predicted evolution of the number of qubits over time can be observed.

practice often determined by the CNOT (CX) gate error and the readout error [222]. Predominantly, the two-qubit gate fidelities are considerably worse than the ones of single gates (by about a factor ten). As the CNOT gate is one of the most popular two-qubit gates, it is frequently used as a representative gate to indicate the gate level noise of a NISQ machine. Readout noise represents an error in the measurement operation from quantum states to classical data.

The noise influence on the results can either be reduced by hardware improvements, but also by software solutions. There exist two methods: error correction and error mitigation. As already mentioned, error correction allows compensating for noise of quantum circuits with the implementation of smart circuits, utilizing additional auxiliary qubits [223]. As the number of qubits on current NISQ devices is limited, error mitigation represents an alternative method, which does not require additional qubits and exclusively relies on software solutions and an increasing number of executed circuits. An example technique for mitigating gate errors represents the quasi-probabilistic quantum error mitigation. It allows simulating a noise-free quantum computer, while operating a noisy one. However, error mitigation methods do not come for free. Quasi-probabilistic error mitigation causes an exponentially scaling computational sampling overhead, increasing with the number of corrected gates [224]. Other gate level error mitigation methods are given by Pauli-frame randomization [225] or zero-noise extrapolation [226].

For readout measurement error mitigation, the conventional bit-flip correction technique [227] is applied in this thesis which is in Qiskit implemented as `CompleteMeasFitter`. It employs a calibration model which prepares all 2^n basis states of the Hilbert space and measures the counts. From these results, a calibration

matrix M can be calculated:

$$M = \begin{bmatrix} 1 - p_{01} & p_{10} \\ p_{01} & 1 - p_{10} \end{bmatrix}, \quad (7.19)$$

where p_{01} (p_{10}) is the bit-flip probability of misidentifying 0 as 1 (1 as 0) during read-out. With the calibration matrix M , the average outcome of another experiment can be corrected: $P_{\text{noisy}} = M \cdot P_{\text{ideal}}$. Today's quantum circuit studies frequently search for the shallowest circuit which sufficiently solves the addressed problem. Without error mitigation on NISQ devices, for deep circuits with many gates, the results of the imperfect quantum computers become worse compared to more shallow circuits, because the noise of each gate accumulates and makes the measurement inaccurate.

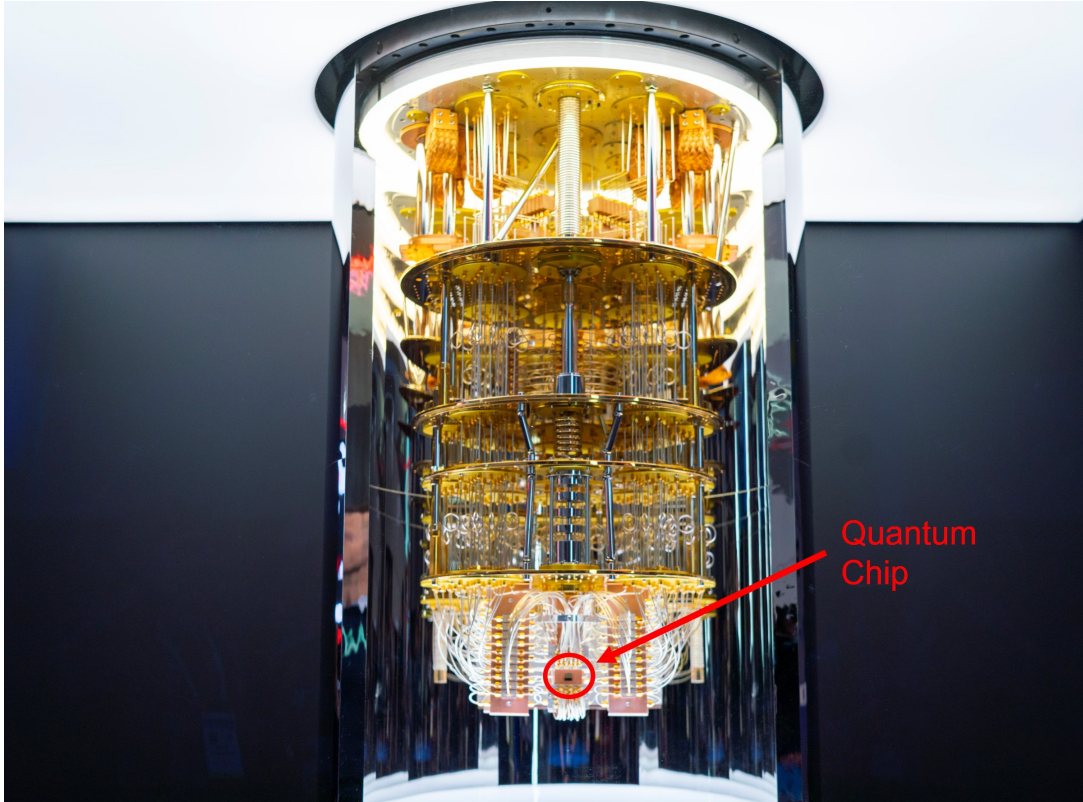


Figure 7.9: An exemplary quantum computer from IBM. Picture taken from the IBM media library in Ref. [228]. The picture shows the dilution refrigerator, which gradually cools from room temperature at the top to ultra-cold conditions at the bottom to maintain stable qubits on the quantum chip. The entire device is approximately one meter in height.

At present, there exist multiple technologies for quantum computers. Qubits can be represented by any two-level system. For example, trapped atom or ion computers

trap charged atoms/ions by magnetic fields and operate orbiting electrons in superposition as qubits [229]. Quantum computers built with photonic qubits utilize single photons as qubits with the help of mirrors and beam splitters [230]. Lastly, superconducting transmon quantum computers operate superconducting qubits at ultra-cold temperatures to perform fast operations [231]. To mention a few examples, the latter mentioned principle is operated by the IBMQ devices, trapped ion quantum computers by IonQ and photonic quantum computers by Xanadu Quantum Technologies. All the aforementioned techniques maintain specific advantages and disadvantages, and no clear favorite is determined yet [230]. To obtain an impression, an exemplary quantum computer from IBM is visualized in figure 7.9. It contains all the electronics cables required to control and readout the qubits, the gates and the cooling system. The actual quantum processor is placed at the bottom of the quantum computer. Figure 7.10 visualizes a seven-qubit chip, which serves as the processor of the IBM quantum computers.

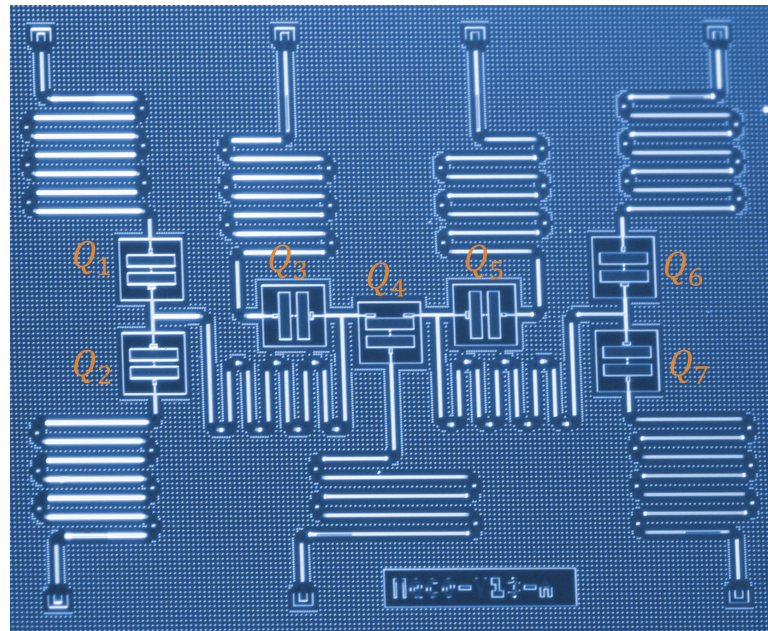


Figure 7.10: A seven-qubit processor of a superconducting quantum computer from IBM including the seven Josephson Junctions representing the superconducting qubits. Picture taken from the IBM media library from Ref. [228]. The size of the chip is about 20-30 mm.

Because of the limits of current NISQ devices, classical simulators are used to develop and study quantum algorithms under pure and noiseless conditions. There exist multiple implementations. Statevector computations return the precise quantum statevector, or the wave-function, for multiple qubits in quantum circuits. On the other hand, on quantum simulators, the quantum algorithm is executed several times, and the

measurement samples estimate the result as counts. This approach is similar to an execution on a real quantum device. Furthermore, on simulators, noise models can be included to mimic and simulate the behavior of a physical NISQ device.

If the quantum simulations can perform similar accurate calculations, one crucial reason why quantum computers are required is given by the amount of memory required to store one single state of a quantum register with n qubits on a classical device. In table 7.1, the evolution of one approach to the calculation of classical hardware memory is provided to represent an n qubit state. The dependence for calculating the required number of bits in a memory is 2^n , which scales exponentially with n . Therefore, each additional qubit doubles the hardware costs for the simulation. In table 7.1, it can be observed that 43 qubits already require 1 TB (Terra byte, $1 \text{ TB} = 10^{12} \text{ B}$) of classical memory only to store the information. This behavior was practically demonstrated with a scaling test on an Intel high performance computing center, where it was only possible to simulate up to 42 qubits with using 2 048 computing nodes, each equipped with two Intel Xeon 8174 CPUs [232]. If the information should be processed by quantum gates, for example a two-qubit gate, then the square of this number is required. Furthermore, similar to the required amount of memory, also the simulation time grows with the number of qubits.

Table 7.1: The number n of qubits and the required memory in bytes to represent one quantum state on a classical computer.

Number of Qubits	Memory
10	128 B
13	1024 B
23	$\approx 1 \text{ MB}$
33	$\approx 1 \text{ GB}$
43	$\approx 1 \text{ TB}$

Optimizations can be applied to decrease the required memory, as explained in Ref. [233]. One such method involves the use of the Feynman path integral, which provides an alternative representation for the previous matrix quantum computations. This approach converts exponential-space computations into polynomial-space computations, denoted as PSPACE. However, while this method benefits in terms of memory requirements, the path integral calculation requires an exponentially growing simulation time with the number of qubits. Therefore, the Feynman path integral effectively shifts the challenge from memory to computation time. In general, simulations of complex quantum circuits with more than 33 qubits become challenging for classical computers. In Qiskit, the maximum number of qubits to be simulated is set to 25.

Due to large matrix calculations, adding noise to the ideal quantum simulation the

simulation cost can increase by a factor up to 100x, as observed in Ref. [234]. The point when quantum computers outperform classical quantum simulators for high fidelity tasks will be when approximately 40 logical qubits can be operated [199]. With current error correction techniques, this corresponds to multiple tenth-of-thousands physical qubits.

7.5 Quantum Machine Learning

Quantum machine learning (QML) represents an emerging field, which promises significant advancements for ML algorithms by porting them to quantum computers [235]. Classical ML algorithms are undeniably successful, however, training and inference of large models can be hardware resource intensive and time-consuming. In order to construct more realistic ML models, in the future, ML models will grow even larger and require more hardware resources [236]. QML on quantum computers can overcome this problem. QML possesses theoretical and experimental evidences to solve problems (exponentially) faster than classical computers [221, 237, 238]. Other possible advantages are that QML algorithms might require fewer data for training, while achieving a similar level of accuracy, or they can deal with more complex data structures, while employing simpler models [236]. Furthermore, ML models are robust to a specific amount of input or weight noise and can in some cases even benefit from it, in terms of better generalization, reduced over-fitting and higher fault tolerance [12, 239–241]. Similar advantages are expected for full QML models, and for specific applications already proven [242–245]. The QML noise tolerance can potentially cope with the error rates of near-term quantum devices, enabling reasonable simulations without the operation of logical qubits.

QML allows a completely novel strategy to solve classical problems with the help of quantum theory and quantum information. For example, the trainable part of a ML model can be replaced by a quantum model, where variational quantum circuits, which exploit the quantum properties of superposition and entanglement, replace classical neural networks [221]. In current strategies, this model is outsourced to a quantum computer but similarly with backpropagation or stochastic gradient descent techniques optimized. In this context, the quantum circuits are commonly denoted as quantum neural networks [246] or variational quantum circuits. Combined with classical optimization algorithms, QML is expected to be among the first practical and most promising applications on NISQ devices for delivering sufficient accurate results [221, 238].

Encoding or embedding the classical data into quantum states and decoding the quantum data back to classical states is a non-trivial task [247]. Currently, many encoding techniques exist, each exhibiting specific advantages and disadvantages, for example,

the computational complexity or the required number of quantum gates for implementation. Identifying “the best” encoding technique in practice is still very much application dependent [247]. In this thesis, multiple encoding methods are tested and evaluated.

7.6 Quantum Generative Models

In the previous chapters 5 and 6, it was demonstrated that classical ML generative models, especially GANs, generate accurate shower images close to the ones of the Monte Carlo training set, while providing tremendous speed-ups. In the following chapters, the performance of quantum generative models is explored for the same calorimeter detector simulation use case. Researches claim a potential advantage of quantum generative models compared to their classical counterparts [248, 249]. Additionally, quantum generative models are expected to maintain a higher expressive power, to be more stable during training and maintain shorter training times [132]. Due to the practical limitations of current quantum devices, in terms of number of qubits, coherence time and connectivity, simplified models are primarily executed on quantum simulators.

Classical generative model training shows multiple difficulties, as documented earlier in section 4.2.1. Therefore, it is investigated if quantum computing provides advantages for generating samples of a given data set. It still remains an unanswered question, if classical techniques, such as label smoothing or batch normalization, can improve the performance of quantum algorithms likewise [132]. Conventionally, quantum models are able to learn both, quantum and classical data [132]. In this thesis, as a first step, the quantum models are operated and trained exclusively with classical data.

In general, most quantum generative models are similarly constructed to their classical analogue. Some model parts, in particular the trainable neural networks, are replaced by quantum methods, specifically variational quantum circuits [250]. The trainable parameters of the variational algorithms are the parameters of the rotational gates in the quantum circuits that can be optimized. Non-linearity is achieved by placing non-linear entanglement gates after individual qubit rotations, allowing to learn complex distributions [251]. In particular, due to the success of classical GANs, qGANs – the quantum analog of GANs – are investigated in this thesis to enhance the accuracy of calorimeter simulations and explore other potential benefits. The qGAN models come in various configurations where either the generator or the discriminator or both are implemented in a quantum mode [13, 23]. The in this thesis applied quantum models are introduced in detail in the following chapters.

Chapter 8

Hybrid Quantum - Classical Generative Adversarial Network Model

This chapter introduces a first quantum generative model as a proof-of-concept for the calorimeter use case and to explore and understand the quantum behavior. Because of today's quantum hardware limitations, a hybrid quantum-classical GAN (qGAN) model is designed as a first step. The model is hybrid because some parts are executed with quantum computing and others with classical computing. It should be noted that the quantum computing part can be performed on a quantum simulator on a classical computer. The structure of the hybrid qGAN is the same as for the classical GAN, where two models are competing with each other. The hybrid quantum-classical approach, however, consists of a quantum generator, defined as a parameterized quantum circuit and a classical neural network as discriminator. The generator quantum circuit consists of available gates in current quantum devices, denoted as hardware efficient Ansatz [252]. These represent simple single-qubit rotation gates around a fixed axis controlled by real tunable parameters and two-qubit entangling gates, as introduced in section 7.2. Similar to neural networks, non-linear parameterized quantum circuits can represent any function and be trained to the desired accuracy, provided the proper depth and architecture [253].

The approach of the hybrid qGAN model operated in this study was initially developed and published by IBM in Ref. [254]. However, the model is modified and optimized for the calorimeter simulation use case. The results of this chapter are as part of this thesis partly published in Ref. [255] and Ref. [244].

8.1 Principles of the Hybrid Model

The working-principle of the hybrid qGAN model is demonstrated in figure 8.1. The key features of the hybrid qGAN model are the quantum generator circuit and the classical discriminator neural network. The input to the quantum generator represents a uniform distribution generated by H -gates applied to qubits initialized in the basis state $|0\rangle$. The generator circuit outputs one quantum state in each forward step.

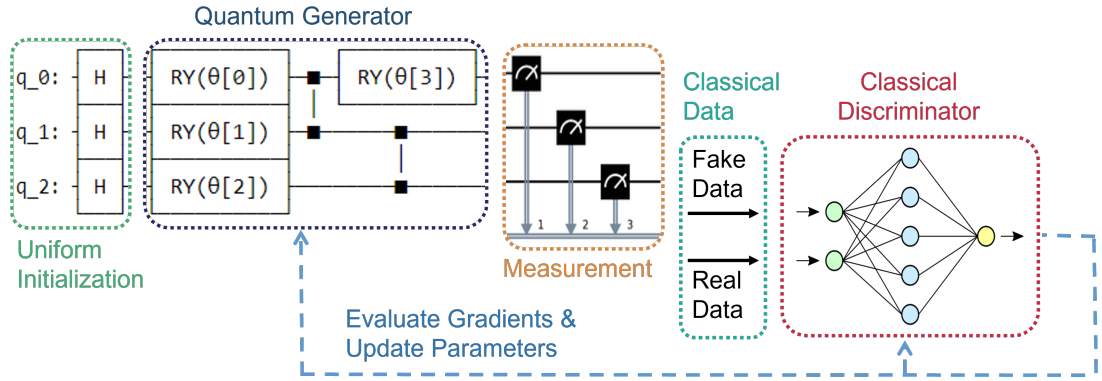


Figure 8.1: The schematics of the hybrid quantum-classical GAN model [255]. The key components are a quantum generator circuit and a classical discriminator neural network.

It is worth highlighting, with a hybrid approach, where the generator is quantum and the discriminator classical, the generator output $|x\rangle$ must be measured and converted from quantum to classical data $|x\rangle \xrightarrow{\text{measurement}} x$ before being proceeded by the discriminator. The measurement of each qubit returns 0 or 1, corresponding to the probability function defined by the quantum circuit. The measurement of multiple qubits leads to a classical bit-string of zeros and ones, where the length of the bit-string is equal to the number of qubits. The binary transformation from the bit-strings into a pixel number is demonstrated in table 8.1.

Amplitude encoding is employed to encode the classical data into the amplitudes of the quantum states and for decoding the quantum states back into classical data. The advantage of amplitude encoding is that it reduces the dimensionality, because N classical features (or amplitudes) can be encoded with only $n \geq \log_2(N)$ qubits [256]. In this study, the amplitudes N represent the number of pixels of the training images. As a first test, merely $N = 8$ pixel images are operated. With $n = 3$ qubits in the generator circuit, it is possible to represent $N = 2^n = 2^3 = 8$ quantum states, which are provided in table 8.1.

For amplitude encoding, each generator output corresponds to one specific pixel of

Table 8.1: Measurement of the quantum state $|x\rangle$ into a classical bit-string x and the binary conversion into a classical pixel number. The provided quantum states correspond to the amplitudes of $n = 3$ qubits.

Quantum State:	$ 000\rangle$	$ 001\rangle$	$ 010\rangle$	$ 011\rangle$	$ 100\rangle$	$ 101\rangle$	$ 110\rangle$	$ 111\rangle$
Bit-String:	000	001	010	011	100	101	110	111
Pixel Number:	0	1	2	3	4	5	6	7

the shower image, imagined as “one-hot” vectors. To get from the single pixel of a generator measurement to a probability distribution, the generator is executed multiple times and the outcomes are averaged. The number of repetitions, how often the generator is executed with the same setting (exactly the same parameters and inputs), is typically denoted as the number of shots. Therefore, the generator output represents a probabilistic distribution, where the pixel entries correspond to probabilities. On the contrary, for real images, the pixel entries correspond to energies that will vary and fluctuate for individual shower images. By increasing the number of shots, the probability function of the output states of the quantum generator circuit can be more accurately reproduced. The necessity of multiple measurements (shots) and that only probability distributions can be generated constitute acknowledged drawbacks of amplitude encoding, but this is accepted for this initial feasibility study, and addressed and improved in the subsequent models.

Similar to the classical GAN, the quantum generator output represents the fake data $|x\rangle$, which is additional to the training data (real data) the input into the classical discriminator neural network. For this simplified use case, the discriminator neural network does not contain auxiliary losses. Therefore, the typical GAN True/Fake probability is the only discriminator output, as described in section 4.4. According to the loss function, the gradients of the generator circuit and the discriminator neural network are evaluated, and the parameters updated. For both the generator and the discriminator, the Adam optimizer [257] is applied. The following section describes how the gradients on quantum computers are determined to optimize the trainable parameters of the generator model.

8.2 Quantum Circuit Parameter Update

There exist multiple approaches for estimating the gradients of a classical numerical computation within finite runs [258]. In classical computing, an automatic differentiation approach is employed. For example, backpropagation for neural networks. There the gradients are computed with intermediate derivatives for the various sub-functions inside the neural network following the chain rule [259]. This approach

makes use of intermediate derivatives. However, these are not available on quantum computers. Accessing intermediate states on quantum devices would affect the computation and destroy the quantum state due to intermediate measurements. Therefore, another approach must be chosen.

The most precise method employed for updating the gradients g in quantum computing follows the parameter-shift rule [260]. For estimating the gradients, each trainable parameter x_i of the quantum circuit is successively shifted a bit Δx in the positive and negative direction. The gradients of the parameters can then be calculated with the formula

$$\nabla g(x_i) \approx \frac{g(x_i + \Delta x/2) - g(x_i - \Delta x/2)}{\Delta x} \quad (8.1)$$

from Ref. [258]. It is worth highlighting that for the parameter-shift rule, the quantum circuit has to be executed two times for each trainable parameter, and the parameter is shifted once in $+\Delta x$ and once in $-\Delta x$ direction. For obtaining the outcome of the current parameters, for example for intermediate accuracy evaluation during training, the circuit must be once additionally executed.

8.3 The Initial 1-Dimensional Model

First, a simple one-dimensional (1D) hybrid qGAN model is developed and tested for generating eight pixel calorimeter shower images. The quantum circuit architecture and the training data preparation are explained before the model is evaluated. The model is first trained without noise to mimic an ideal quantum computer. Then the effect of artificial simulated hardware noise is evaluated in training and inference, including some initial runs on real quantum devices.

8.3.1 Quantum Generator Circuit Architectures

In general, it is a delicate task to determine which circuit architecture is the best for a problem, similarly as for neural networks in classical computing. In terms of restricted hardware, keeping an eye on noisy simulations and also on eventually training on the real quantum hardware with limited available simulation time, the number of gates should be kept as low as possible.

Initially, the generator circuit from figure 8.2 is employed for the qGAN training. The circuit architecture suggested by Ref. [261] is used with the maximum dimensional expressivity for three qubits. Expressivity of a quantum circuit represents the ability to uniformly reach the full Hilbert space [261]. Put differently, this means the circuit is

capable of generating each possible quantum state. Additional gates would be redundant and not increase the expressivity of the circuit.

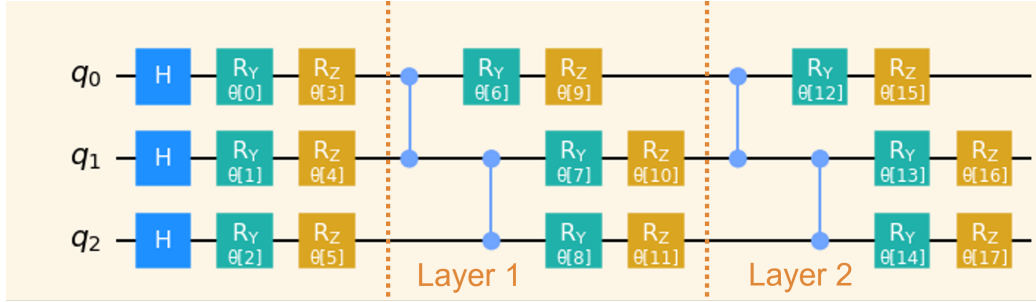


Figure 8.2: A fully expressive quantum circuit for three qubits.

The circuit consists of three qubits to represent the eight pixels of the simplified shower image. The qubits are initialized in the basis state zero $|0\rangle$, followed by H gates to initiate superposition. The generator consists of two layers (circuit depth of two), as indicated in figure 8.2. Parameterized R_Y - and R_Z -gates provide the trainable part: each R_Y - and R_Z -gate has a trainable parameter θ_i corresponding to the rotation angle. Entanglement is created by CZ gates in a linear entanglement topology. Linear entanglement means that the j^{th} qubit is entangled with the $(j + 1)^{th}$ qubit and represents the simplest entanglement scheme. However, for a circuit with three qubits it is sufficient. The only better entanglement scheme for three qubits would be all-to-all qubit entangling, in which each qubit is entangled with each other.

The results are reasonable good, however, the simulation time remains long. Therefore, it was investigated if the circuit can be shrunk for reducing the training time while maintaining a good accuracy level. It is observed that removing the R_Z -gates from the circuit does not result in an accuracy drop. Additionally, the required number of epochs could be reduced due to fewer trainable parameters. The goal is to minimize the number of gates while providing a reasonable level of accuracy. This is similar as for the classical GAN case in the previous chapters of this thesis, where the training time was reduced by optimizing the neural network architecture. The more shallow quantum generator circuit architecture without the R_Z -gates is employed for training the qGAN in this study. The circuit is illustrated in figure 8.3.

Decreasing the number of gates in quantum circuits comes with an additional benefit for simulations carried out with including gate-level noise, as it is present in current physical quantum devices. Each gate represents an additional noise source for the model, which potentially decreases the accuracy. The fewer gates the circuit consists of, the more resilient it is to gate-level noise. As a side note, in the research in Ref. [261], where the fully expressive quantum circuit is introduced, they accomplished to preserve the expressivity while removing single gates from the circuit. However, their

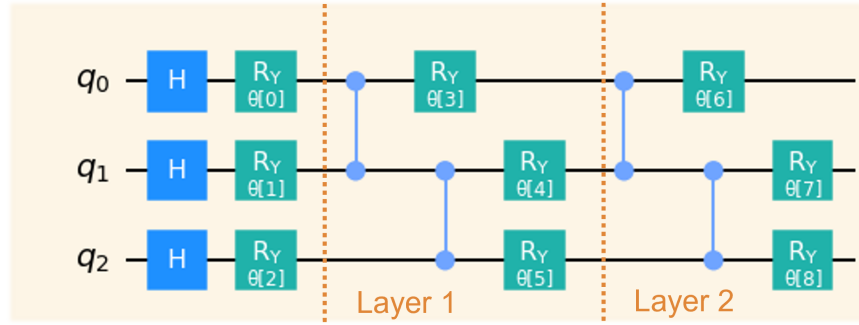


Figure 8.3: The more shallow quantum circuit architecture employed for training the generator of the 1D qGAN model.

approach is for the initial tests in this study not applied.

The corresponding classical discriminator neural network consists of three input features for the three-number bit-string. The hidden layers are two dense layers with 512 nodes and one with 256 nodes. The hidden layers are each followed by LeakyReLU activation functions. The output layer of the discriminator consists of one single output neuron followed by a sigmoid activation function, which provides the True/Fake probability. The discriminator architecture is visualized in figure 8.4. Following the standard GAN approach, a binary cross-entropy loss function is employed. The gradients are classically computed, and the parameters of the quantum generator and the classical discriminator updated. The discriminator network size is found to be appropriate for the chosen generator circuit by empirical tests.

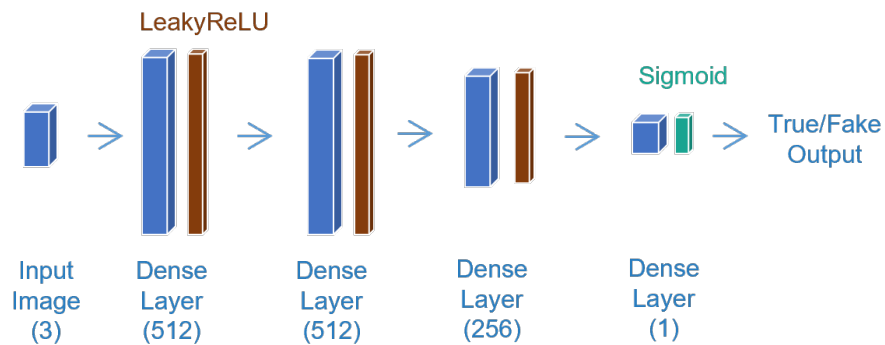


Figure 8.4: 1D classical discriminator neural network architecture.

8.3.2 1D Training Data Preparation

The initial full-size data set contains the 3D 25x25x25 pixel calorimeter shower images from section 2.4, utilized in the classical ML simulation. For the simplified use case in this 1D research, the training data is downsampled to 8 pixels along the longitudinal shower axis. The image size of 8 pixels was chosen to utilize the full amplitude range of three qubits ($2^3 = 8$), as previously outlined and shown in table 8.1.

For downsampling, the training data is first averaged over the full energy range of 10-510 GeV. Then, the average energy image is projected to the z-axis to get a one-dimensional distribution with 25 pixels along the longitudinal shower axis, as it is visualized in figure 8.5 on the left. In a subsequent step, the 25 pixel 1D distribution is downsampled to 8 pixels by calculating the sum of three consecutive pixels, while discarding the last pixel. This represents the distribution aimed to be reproduced by the qGAN model. The 8-pixel image is visualized in figure 8.5 on the right.

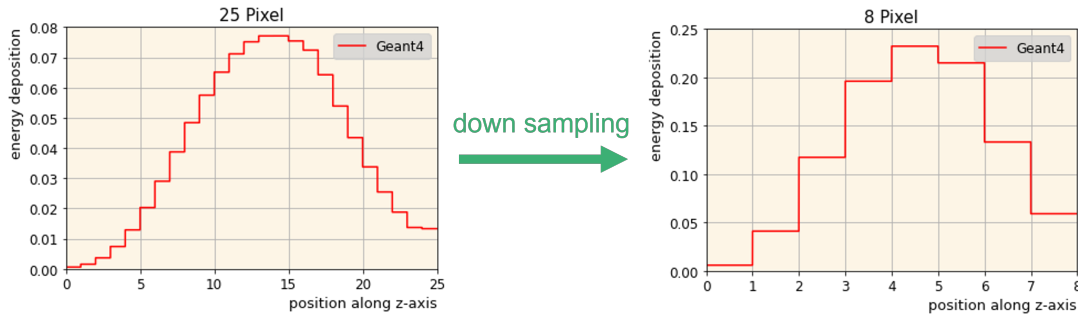


Figure 8.5: On the left, the 1D training data averaged over the calorimeter z-axis with 25 pixels is provided, and on the right, the same distribution downsampled to 8 pixels. The pixel energy deposition on the y-axis is represented in an arbitrary unit.

From the 8-pixel distribution in figure 8.5 (right), the scalar training data set is created, as required for amplitude encoding. The training data set consists of 30 000 scalar integer numbers which follow the energy distribution. The scalar integer numbers range from 0 to 7 to mimic the 8 pixels. As explained previously, they can be imagined as “one-hot” vectors, where each sample corresponds to one pixel of the image. Averaging the scalar numbers over multiple samples results approximately into the training data distribution from figure 8.5 (right). To input these scalar numbers into the discriminator, they are converted into a bit-string, similar to the generator output. The conversion of the scalars into the bit-string follows the binary translation from the amplitude encoding in table 8.1. Employing 30 000 training samples and 3 000 shots, the training is divided into ten equal-sized batches per epoch.

8.3.3 Training and Inference without Noise

The training is performed on the IBM Qiskit [262] quantum simulator, at first without noise, to simulate an ideal quantum computer with logical qubits. In the subsequent section, noise is included and its influence on the accuracy evaluated. All the following tests are carried out with Qiskit as quantum computing framework in version 0.26.2. The classical device on which the quantum simulator is run is an Intel Xeon Gold 6130 CPU.

Figure 8.6 (left) provides as green bars the generator output after random uniform initialization. The x on the x-axis corresponds to the pixel number and $p(x)$ on the y-axis to the measurement probability p for the individual pixels x . All pixel probabilities sum up to one. The blue line provides the pixel probabilities of the Geant4 training data as a reference.

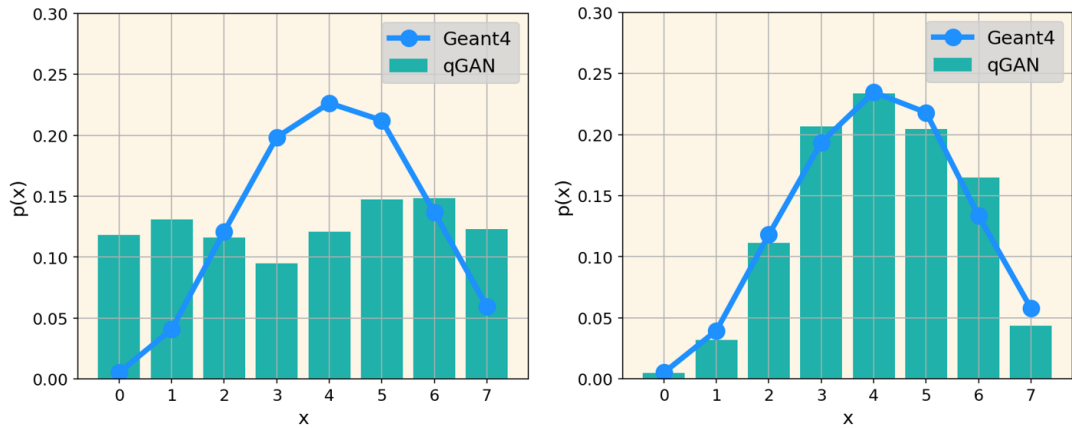


Figure 8.6: (left) The output of the uniform initialized generator and (right) the trained model without noise.

It constitutes a non-trivial and time-consuming task to find appropriate hyperparameters which allow training the hybrid qGAN model to convergence, similar as for classical GAN models. Initial successful converging training trials require 3 000 epochs of training. With this, the execution of the training takes more than one day. After the in the following introduced optimizations, the required number of epochs for training could be reduced to only 300. Separate generator and discriminator learning rates are implemented, as well as an exponential learning rate decay. Furthermore, at each optimization step, the discriminator is trained more frequent than the generator. The best hyperparameters, found by a hyperparameter optimization with Optuna, are: generator learning rate of 0.1, discriminator learning rate of 0.01, a common exponential learning rate decay of 0.01 and the discriminator trained ten times more frequent than

the generator. For training and inference, the images are generated using 2 048 shots. The results of the best trial are provided in figure 8.6 (right). It can be noted that the qGAN model is capable of correctly reproducing the average probability distribution of Geant4.

As (q)GAN training is known to be unstable, it is crucial to test the reproducibility and robustness. Therefore, the training is repeated for 20 trials. For each trial, the training is executed with exactly the same hyperparameters. The relative entropy, also denoted as the Kullback-Leibler-Divergence [111], is used as accuracy metrics. The relative entropy represents a statistical distance between two distributions. In this use case, the two distributions are the true distribution $P(x)$ of the training data for the eight pixel shower images and the fake distribution $Q(x)$ created by the generated images. The equation to calculate the relative entropy is:

$$D_{\text{KL}}(P||Q) = \sum_{x \in X} P(X) \log \left(\frac{P(x)}{Q(x)} \right). \quad (8.2)$$

The evolution and the function of the relative entropy with the increasing number of epochs for the qGAN training is visualized in figure 8.7. The mean of all trials (blue line) is converging smoothly. However, the standard deviation band around the mean is wide, which indicates some fluctuations in the results. The best trial (green line) converges fast to an extremely low relative entropy and maintains then a stable level. Overall, the qGAN training is stable.

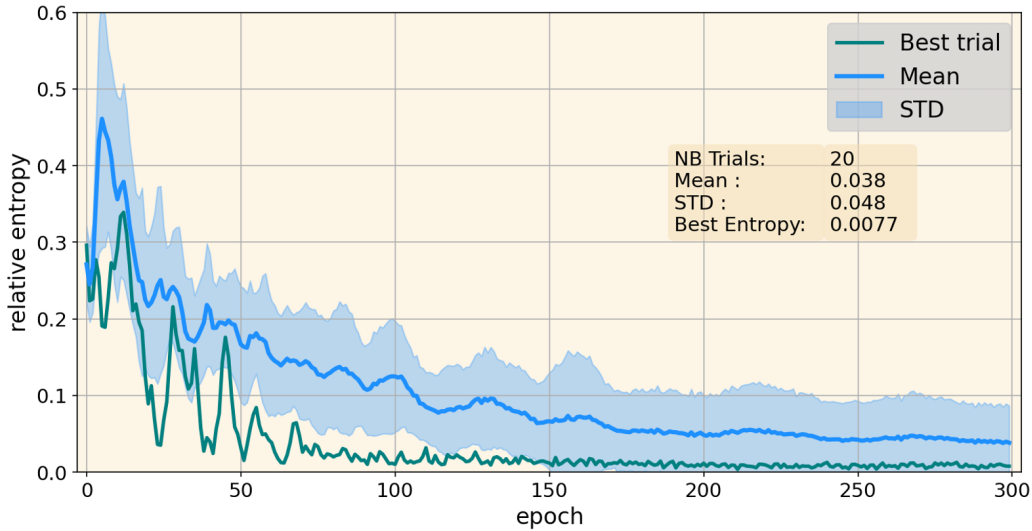


Figure 8.7: The statistics evaluation of 20 training trials with the same hyperparameters. The mean relative entropy, its standard deviation and the best trial are plotted as function of the number of epochs.

8.3.4 Inference with Noise

As explained in section 7.5, QML models are assumed to be able to adapt to the noisy behavior of the quantum hardware. This statement is tested for the qGAN model by introducing noise into the inference process. Furthermore, it is investigated at which noise level the accuracy decreases.

Today, detailed noise studies for quantum models represent a crucial part to overcome the obstacles of NISQ devices [263]. In noise studies, it is evaluated if the models can potentially be executed on real noisy quantum devices without losing too much accuracy. For this, the best model trained without any noise, from the previous study in figure 8.6, is taken and inference is run with different simulated noise values. The influence of readout noise, CNOT noise and both noise configurations combined are measured. In addition, the configurations which contain readout noise are repeated with employing readout error mitigation techniques, as described in section 7.4. Simulated noise configurations follow the procedure:

- Readout noise: The exact same readout error is applied to all qubit measurements with the flip probability p_r .
- CNOT noise: The exact same CNOT noise is applied to each two-qubit gate operation with the flip probability p_g .

For robustness, at each noise level ten inference trials are carried out. The tests are executed on a quantum simulator with discrete noise values in the range 1-5% with a step-size of 0.2% and from 5-15% with a step-size of 1%. In the configurations with both readout and CNOT noise, the noise levels of both types are set to be equal. The results of the five noise configuration are visualized in figure 8.8. On the x-axis, the increasing noise level, and on the y-axis the relative entropy as accuracy measure are provided. First, the configurations without readout error mitigation are evaluated. It can be noted that the configuration with only readout noise (green solid line) and the one with only CNOT noise (blue solid line) perform better than the combined (orange solid line). From 0-3% of noise, the readout noise (green) and CNOT noise model (blue) maintain an almost constant level of accuracy. Applying both noise models together (orange solid line), the accuracy remains merely constant to around 1% of noise. For low noise levels, it almost seems like the accuracy slightly improves. However, due to the standard deviation, indicated by the colored band around the lines of the mean values, this is not statistically significant. For higher noise levels, all configurations lose accuracy, indicated by the increasing relative entropy, and the model with exclusively CNOT noise is performing slightly better than the model with only readout noise. The different behavior of readout noise vs. CNOT noise might depend on the quantum circuit architecture. For larger/deeper circuits with more two-qubit

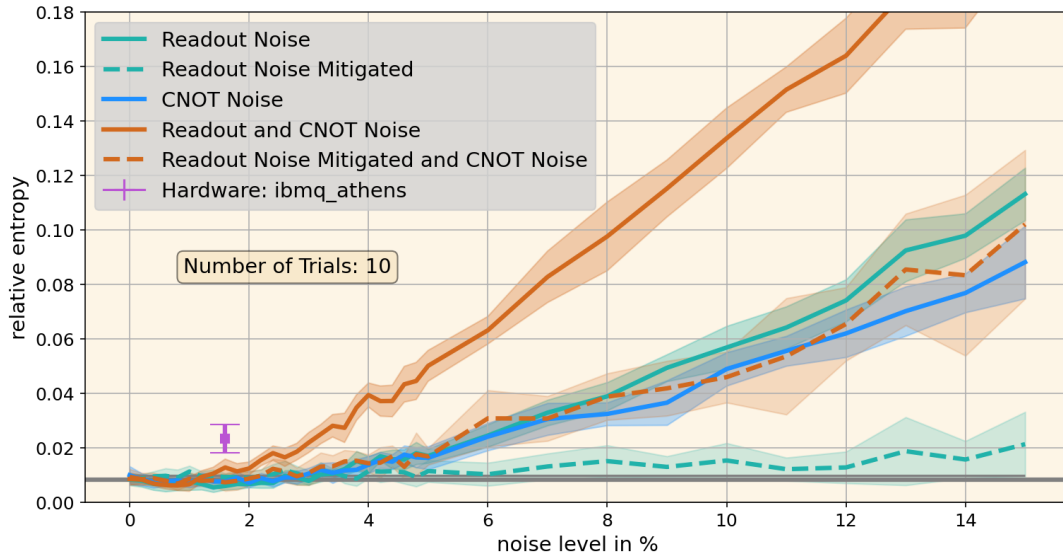


Figure 8.8: The influence of simulated hardware noise on inference for a model trained without noise. Five noise configurations are tested. The dependence of the relative entropy as accuracy measure is shown as a function of the noise level for readout and CNOT noise. The marked point indicates the low noise level of the employed hardware: the `ibmq_athens`, compared to the overall studied noise levels.

gates, the impact of CNOT noise increases and would potentially lead to worse results when applying CNOT noise. The orange model with both, readout and CNOT noise, mimicking real quantum computers, performs worst with a fast increasing relative entropy.

The configurations which contain readout noise are additionally studied with readout error mitigation, as described in section 7.4. The models are visualized by dashed lines. The green dashed line represents the model with exclusively readout noise with readout error mitigation applied. It can be noted that the accuracy remains constant for all noise levels. The dashed orange line represents the model with readout and CNOT noise and additionally applied readout error mitigation. It follows closely the blue model with exclusively CNOT noise. This leads to the conclusion that the employed readout noise can be entirely mitigated with no loss of accuracy independent of the readout noise level.

At last, the training is executed on the real `ibmq_athens` quantum device. The name of the quantum device has nothing to do with its local position and is arbitrary chosen after a city name where IBM has an office. The hardware machine `ibmq_athens` represents a five-qubit machine, which by now is already retired. This machine was chosen, because at the time of the study, it maintained the lowest noise rates. The

noise levels are determined by calibration runs and are summarized in the inference noise column in table 8.2. The result of inference execution on the real hardware is included in figure 8.8. The noise level on the x-axis in figure 8.8 is determined to be the average of the readout and CNOT noise levels over the employed qubits. The relative entropy of the inference run on hardware is worse compared to the one of the combined simulated noise model (orange line). A possible reason for this is that the noise levels between the qubits and gates of the real quantum devices are not equal, as it is assumed in the idealized simulations of the noise models from before. A second possible reason is that the x-axis position of the hardware run might be incorrectly chosen in figure 8.8. Maybe the purple dot must be positioned closer to the highest noise level from table 8.2 and not the average. If the purple point was shifted to the highest noise level of 2.71%, then the relative entropy would agree to the one of the combined noise level (orange line). In any case, the use of real hardware with the lowest noise level indicates that the assumed noise levels in the study are by far too pessimistic.

Table 8.2: The readout and CNOT noise levels for running inference (column 1) and training (column 2) on the `ibmq_athens` device.

Noise Type	Inference	Training
Readout noise of qubit 1:	1.50%	2.38%
Readout noise of qubit 2:	2.23%	1.48%
Readout noise of qubit 3:	2.71%	1.85%
CNOT noise between qubit 1 and 2:	0.78%	0.55%
CNOT noise between qubit 2 and 3:	0.76%	0.55%

In figure 8.9, averaged example shower images for each noise configuration with various noise levels are visualized. For low noise levels (1-2% of noise), the generated shower images remain close to Geant4. However, for larger noise levels deviations can be observed. Especially at the bottom right plot, where 15% of readout and CNOT noise are applied, the image tends to move towards a random uniform distribution. The average generated shower image of running inference on the real `ibmq_athens` device is provided in figure 8.10 (left).

To summarize: with the 1D qGAN model the inference process remains robust to quantum hardware noise up to 3% for only readout or only CNOT noise, and until 1% for both noise models combined. However, no significant accuracy improvement in inference due to hardware noise can be observed. Readout noise can completely be mitigated by applying readout error mitigation. For current quantum devices, only the region up to around 5% of noise is relevant, because usually, the average hardware noise levels are already well below 5%, as it can be seen on the IBMQ platform for compute resources.

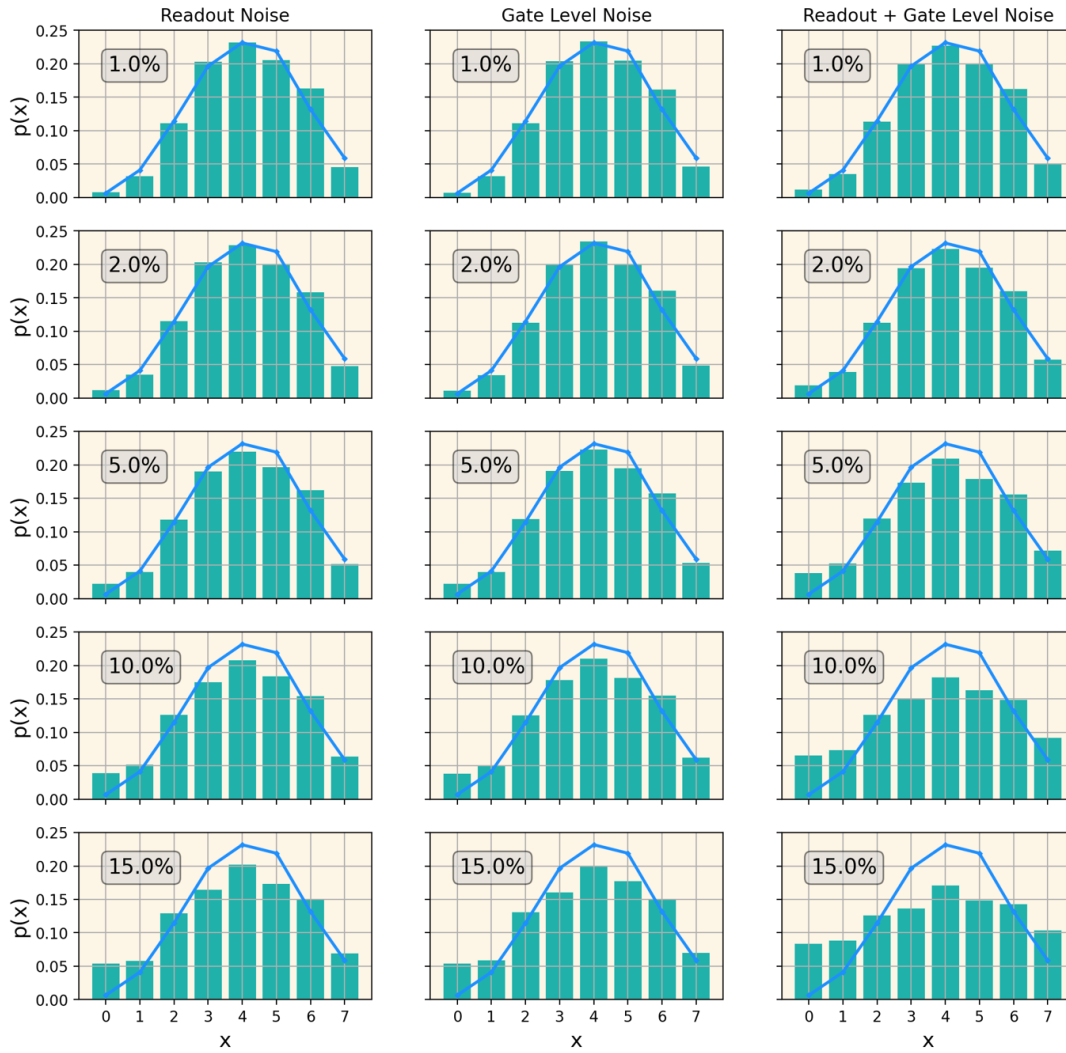


Figure 8.9: Example shower images generated with various noise models and noise levels. The left row contains shower images generated with only readout noise, the middle row with only two-qubit gate level CNOT noise and the right row with both, readout and CNOT noise. From top to bottom, the noise levels in inference are increased. The noise levels are provided in the grey box inside the plots. The blue line indicates the Geant4 data as reference, and the green bars the generator output averaged over 20 images.

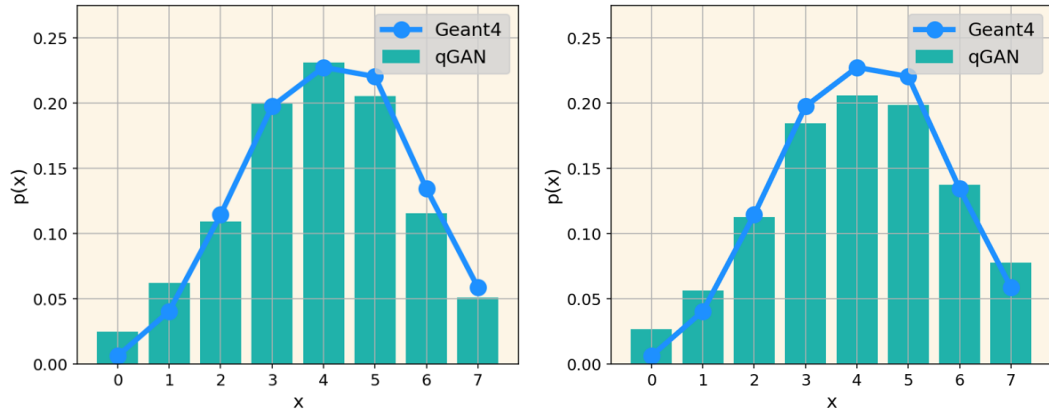


Figure 8.10: (left) The average probability distribution generated on the *ibmq_athens* device. The model is trained without noise. (right) The generated average probability distribution of the model trained on the simulator with loading the real hardware noise of the *ibmq_athens* device.

8.3.5 Training with Noise

In this subsection, the effect of hardware noise on training is evaluated, following the procedure for inference noise from the previous subsection. Initially, a study with various simulated noise levels is carried out. Next, the training is executed by loading the real hardware noise model of a quantum device into the simulator. Subsequently, an attempt is presented to perform the entire training on a real quantum device.

Training with Simulated Noise

The model is trained with different noise configurations: only readout noise, only CNOT noise and both combined. Different noise levels (1%, 2%, 3%, 4%, 5%, 7.5%) are tested for all configurations. For each noise level, five training trials are executed, in order to evaluate the statistics stability. For evaluating the model in inference, the same noise as in training is applied. The results are summarized in figure 8.11.

In figure 8.11, it can be seen that increasing the noise level from zero to one percent improves the average accuracy by reducing the relative entropy of all noise configurations. For the configuration with only readout (green) and only CNOT noise (blue), the accuracy maintains an improved level even until three percent of noise compared to the noiseless reference (gray horizontal bar). This emphasizes the kind of helpful role of noise for convergence and accuracy as observed already in classical ML appli-

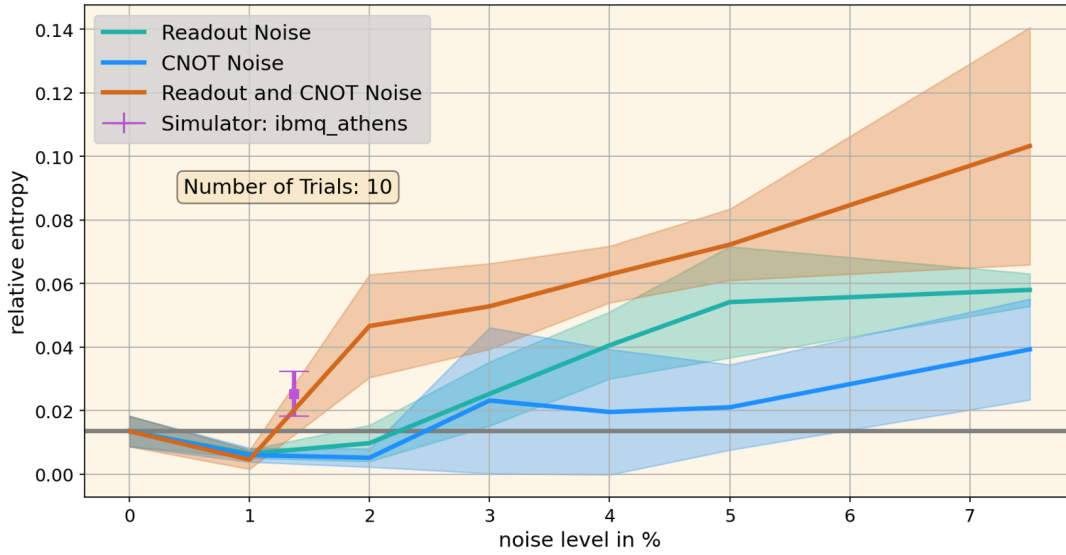


Figure 8.11: The influence of simulated hardware noise to training. The simulated noise levels are increased from zero to 7.5%. Three noise configurations are provided. The purple dot is the result of the training on the simulator with loading the real hardware noise model from the `ibmq_athens` device, as described in section 8.3.5.

cations. This is true for their quantum realization, too. At higher noise levels, the accuracy decreases. Looking at the combined configuration (orange), the accuracy drops already for two percent of noise. The configuration with only readout noise (green) performs slightly worse for larger noise levels than the configuration with only CNOT noise (blue). The training for higher noise levels is more unstable.

Similar results were achieved in our study from Ref. [244], as summarized in figure 8.12. There, for the same model, inference was carried out with only readout noise (orange) and with readout noise and two different error mitigation methods: independent bit-flip mitigation IBF (blue) and bit-flip mitigation BF (green). The error mitigation method used in the previous test was BF mitigation. IBF assumes uncorrelated multi-qubit readout errors which are independent of the other qubits, as described and experimentally verified in Ref. [264]. In figure 8.12, it can be seen that for low bit-flip probabilities (corresponding to the noise levels), the average relative entropy of the models lies frequently below the gray horizontal no noise reference line. However, the error bars are large. For larger bit-flip probabilities, the results of the model without error mitigation become worse, whereas the models including error mitigation remain stable. Both BF and IBF mitigation perform about the same.

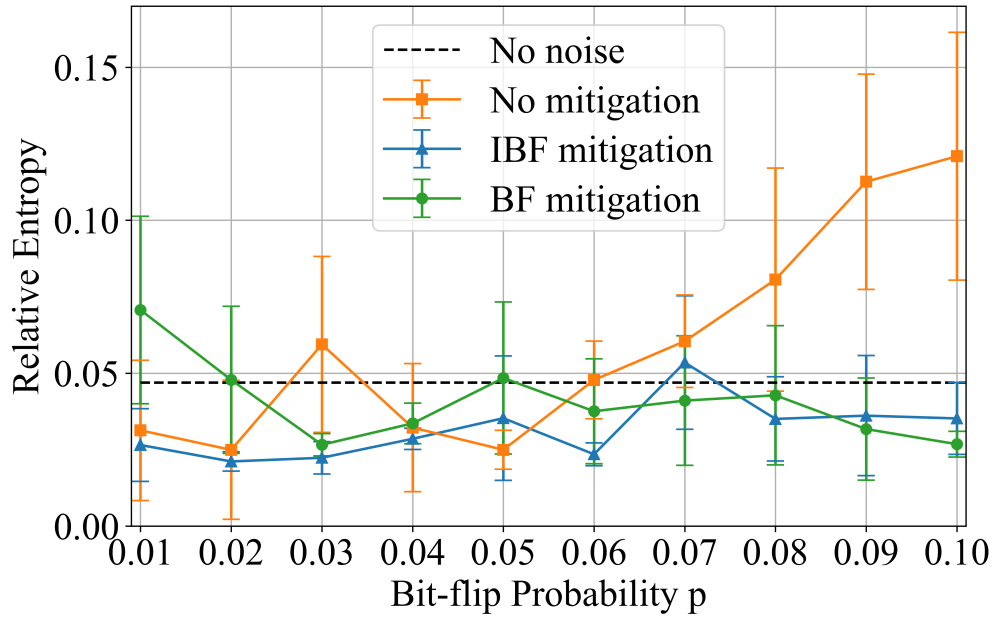


Figure 8.12: Relative entropy after qGAN training in the presence of readout noise, as a function of the bit-flip probability p , averaged over 20 trials. The model with no error mitigation (orange), independent bit-flip error mitigation (blue) and bit-flip error mitigation (green) are shown. The figure was first published in Ref. [244].

Training with Hardware Noise

In order to have a more realistic evaluation of noise effects on physical quantum devices, in the following, the model is trained on the quantum simulator with hardware noise models determined by calibration runs for the `ibmq_athens` device. The noise levels are provided in the training noise column in table 8.2.

The training is repeated for multiple trials with the same hyperparameters. The generated probability distribution of the best trial is provided in figure 8.10 on the right. The distribution overall agrees with the distribution from Geant4. The training statistics of the relative entropy for twenty training trials is summarized and compared to the noiseless training in figure 8.13. The models trained with hardware noise (green line) converge to the same relative entropy as the noiseless model (blue line).

In figure 8.13, there is an increase in the relative entropy in the first few (~ 5) epochs. This behavior is seen also in classical GAN training (e.g. figure 5.6 for the (green) Conv2D G1 model), possibly due to the chosen parameter initialization of the generator and discriminator. For the qGAN model, we have the special situation that the number of parameters in the quantum generator circuit is much smaller (eight

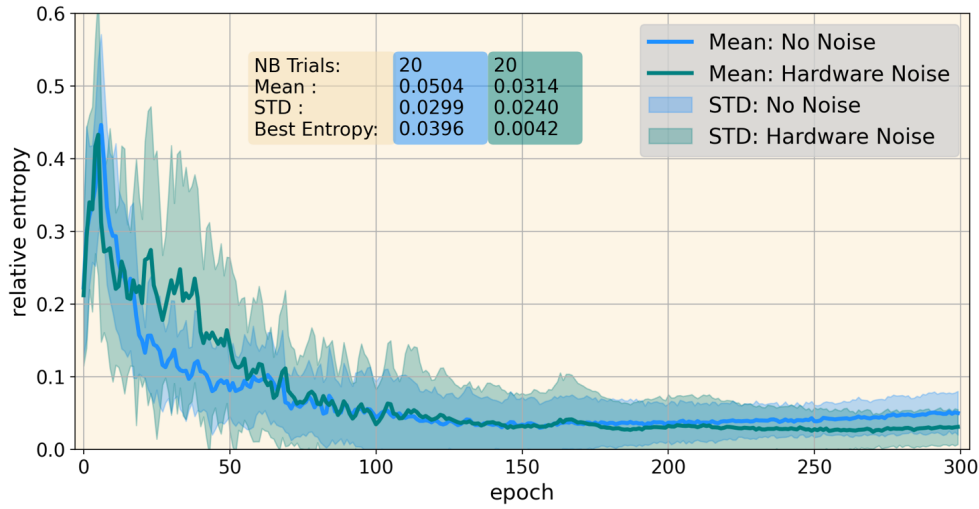


Figure 8.13: The statistics evaluation of multiple training trials with hardware noise (denoted as Hardware Noise in green) compare to the noise less model (denoted as No Noise in blue).

trainable parameters in our case) compared to classical neural networks. This makes it much easier to find a good parameter initialization for quantum circuits than for neural networks. If starting the training with well initialized generator weights, the generated images are already good in the first training iterations. However, in the following adversarial training process, the generator updates are influenced by the output of the discriminator. At the beginning of the training, the discriminator neural network classifies both the true and the fake images with an equal probability of $\sim 50\%$ as fake images. Therefore, in the generator parameter update, this worsens the weights of the generator during the first epochs and causes a decrease in accuracy. Only after a few training iterations, when the discriminator improved and correctly distinguish between true and fake images, the generator weights and its accuracy start to improve, indicated by a decreasing relative entropy.

Until approximately epoch 60 the training of the model with hardware noise is slightly slower than the noiseless training and shows more oscillations, indicating a more unstable training. For a larger number of epochs (>100), the standard deviation band shows a similar width. Furthermore, it can be observed that the model with hardware noise remains at a stable level of accuracy, whereas the accuracy of the model with no noise slightly decreases towards the end of the training.

The inference accuracy of the model trained with hardware noise is added as a purple point to the noise study in figure 8.11. The resulting relative entropy is slightly worse than the model simulating readout and CNOT noise (orange). However, the error bars

are overlapping and the difference is statistically not significant.

To summarize: the expectations that quantum machine learning is robust up to a certain noise level seems to hold for the qGAN model. For training our qGAN model with noise, a slight improvement in the measured accuracy is observed for low noise levels.

The (q)GAN training remains unstable resulting in fluctuations in the relative entropy for different initial training parameters [244]. This instability in the presence of readout noise is explored. Specifically, it is investigated if a larger number of training trials or repetitions n_{rep} leads to more stable results. This test was carried out in a collaborative research in Ref. [244]. The initial training parameters of the generator are drawn by a normal distribution with a mean of 0.1 and a standard deviation of 0.2. The training is executed for various readout noise levels p . At each level of p , a test with 20 trials and a test with 100 trials is executed. The results are summarized in figure 8.14. The standard deviations for larger n_{rep} is visually smoother but the average order of magnitude remains similar. Therefore, the ensemble of simulations converges for all n_{rep} , but individual unstable and non-converging trials appear with a steady probability [244]. Thus, the training instabilities are rooted in the qGAN training and occur independently of the readout-error level and cannot be alleviated by quantum noise or increasing statistics [244].

8.3.6 Training on Quantum Hardware

In the following, an attempt is made to perform the entire training on a physical quantum device. The generator circuit under investigation consists of nine trainable parameters. Therefore, the training with the parameter-shift rule requires $9 \cdot 2 = 18$ jobs per optimization step. One job corresponds to one circuit sent to the quantum device for execution. As introduced before in section 8.3.2, each epoch consists of ten batches, where each batch requires one optimization step. Training for 300 epochs requires $18 \cdot 10 \cdot 300 = 54\,000$ jobs for the full training. At the time this test is carried out, the computing time for one job on the IBM Quantum devices takes around 10-20 seconds, including transpiling, validating and other processes. The reason for the large simulation times on the quantum hardware is given by the network latency and the initialization time of the circuit on the machine. Estimating the average of 15s per job, the total training time would exceed nine days. This enormous time exceeds the available IBM Hub reservation limit by far. Therefore, as described in the following, the training is executed with queuing for each job.

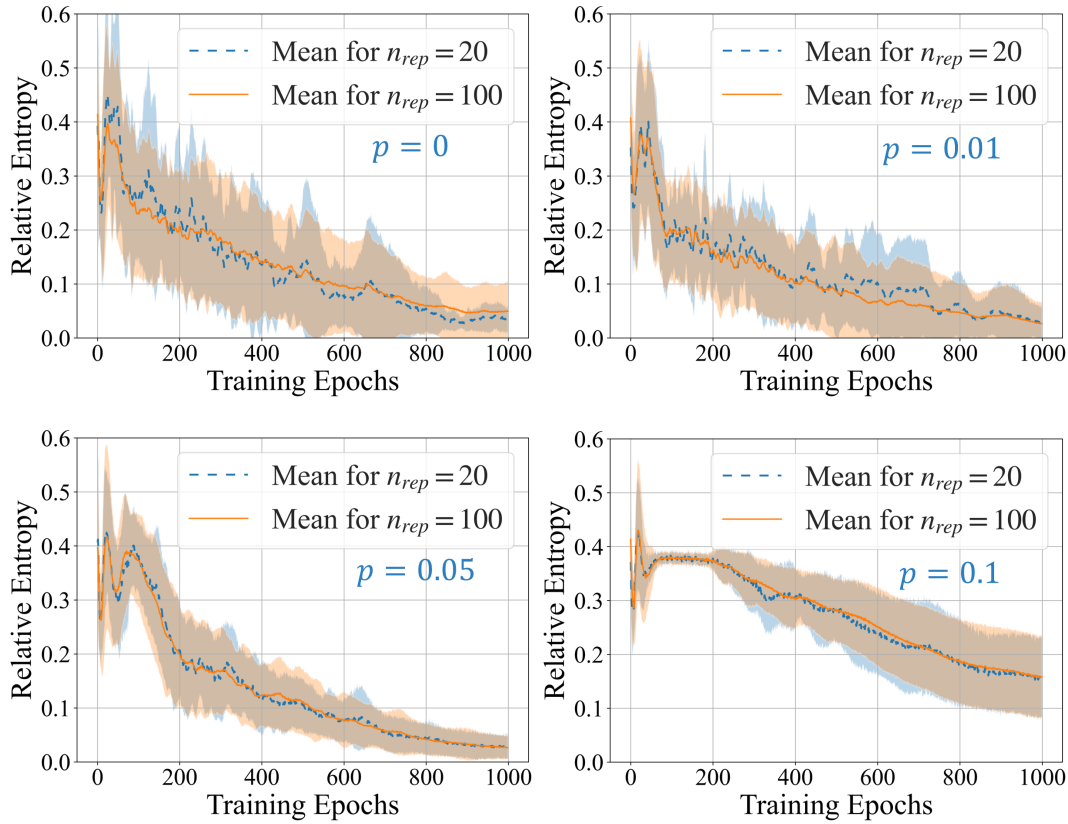


Figure 8.14: Relative entropy as a function of training epochs for different bit-flip probabilities p . Top left: $p = 0$, top right: $p = 0.01$, bottom left: $p = 0.05$ and bottom right $p = 0.1$. The figures are taken from Ref. [244].

Training on Quantum Hardware in Fairshare Mode

The full training is executed on the `ibmq_belem` quantum computer, which has similar characteristics to the `ibmq_athens` device. It is selected, because the average utilization of the machine was modest at the time the training was carried out, resulting in lower queuing times. The training was operated in “fairshare” queuing mode, meaning that each job submitted to the quantum hardware has to queue with jobs from other users before being executed.

Information about hardware execution and queuing times on IBMQ devices are summarized in a study in Ref. [265]. It is measured that the median queuing time is 10x higher than the pure execution time on hardware and for 25% of the submitted jobs the queuing time is even 100x higher. Quantum cloud computing, as the one from IBMQ, is also compared to classical cloud computing and some promising parallels in

the development of computing times over the years are highlighted.

Including queuing, the hardware training summed up to approximately one and a half months. Due to this extremely long training time, several hardware calibrations took place, which caused significant changes in the noise levels. Hardware calibrations at IBMQ are usually performed every 24h to preserve accurate computations. After performing the calibration, IBMQ executes test circuits to determine the current noise model of the machine.

The results of the full hardware training are summarized in figure 8.15. On the left, in the energy distribution plot, the qGAN model is far off from the Geant4 reference. In the middle plot, the relative entropy is plotted as accuracy metrics as a function of the training epochs. At significant hardware calibration changes, the relative entropy is spiking up. Then, the model is able to recover, however, there are too frequent calibration changes which prevent the model to converge. Observing the generator and discriminator losses during training in the right plot, the losses do not converge to equivalent values, implying a failed training. Therefore, as indicated by all plots, the hardware training in fairshare mode is not successful due to too many interruptions by frequent calibrations of the quantum device. Decreasing the job queuing time would improve the situation.

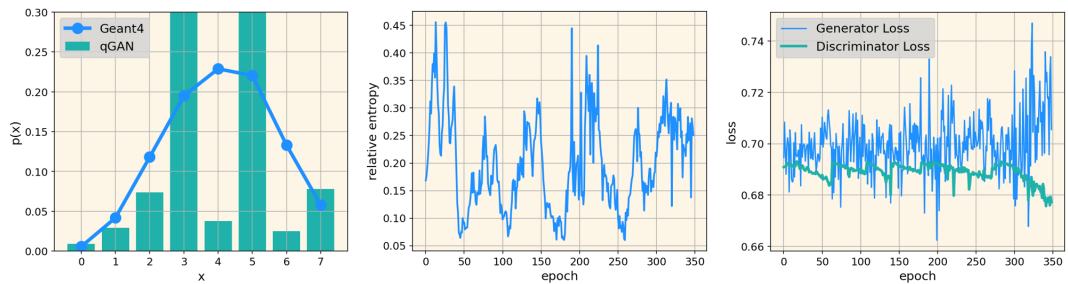


Figure 8.15: (left) The eight pixel shower image, (middle) the relative entropy and (right) the loss functions of the training on the real quantum device in fairshare mode as a function of the training epochs.

From the aforementioned result, it is concluded that it is impossible to execute the training with significant intermediate calibration changes. This issue could potentially be reduced by performing the training in the "dedicated" mode, what means exclusive access to a quantum device without queuing. However, quantum hardware time currently represents a bottleneck and special negotiations would be needed to achieve such an agreement.

8.3.7 Hybrid Quantum-Classical Training Approach

As a full training on quantum hardware cannot be carried out, an intermediate hybrid approach is developed. The training process is denoted hybrid because some parts of the generator training are performed on the real quantum hardware and others on a quantum simulator. To mimic the real hardware, the noise model present on the real hardware is loaded into the quantum simulator performed on classical computer hardware. In the hybrid approach, all generator forward steps are executed on the real quantum device. This ensures that the outcomes (i.e. the generated images, the losses and the relative entropy during training), are calculated by real hardware results. To reduce training time, the majority of the jobs in the training optimization (i.e. the parameter-shift rule for calculating the gradients) are executed on the noisy quantum simulator.

With the hybrid approach in fairshare mode, the training on the `ibmq.belem` quantum computer required two and a half days. With the daily quantum hardware calibrations, the training contains three different noise states, provided in table 8.3. The results of the training are visualized in figure 8.16. In the generated eight pixel shower image on the left, it can be marked that the generated images are close to Geant4. The relative entropy (middle plot) remained stable and the generator and discriminator losses (right plot) retain almost a similar level. Considering all plots, the hybrid training converged to good results.

Table 8.3: The calibration epochs of the hybrid hardware training including the corresponding readout and CNOT noise levels.

Calibration Number	Epoch	Readout Noise	CNOT Noise
0	0	1.04%	0.83%
1	133	3.27%	0.74%
2	272	2.62%	0.86%

To briefly summarize the results of the 1D hybrid qGAN model: the training converges fast and stable, and the model is able to correctly generate the eight-pixel image. Even with low simulated noise levels in training and inference, the accuracy can be maintained, similarly to classical machine learning models. However, for simulated readout and CNOT noise larger than two or three percent, an accuracy decrease can be observed. Due to extensive training times, an entire training cannot be executed on real quantum devices because hardware calibration changes negatively affect the training. Only with the introduced hybrid quantum-classical training, the training could be partly executed on real hardware. In the following, the qGAN model is enhanced to larger image sizes.

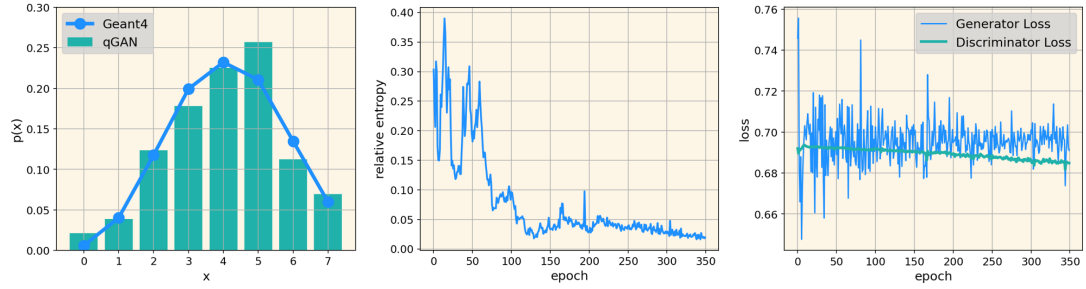


Figure 8.16: ((left) The eight pixel shower image, (middle) the relative entropy and (right) the loss function of the hybrid training in fairshare mode.

8.4 Enhanced 2-Dimensional Model

In this section, the hybrid qGAN model is made more realistic by generating larger two-dimensional (2D) distributions. For the 2D case, the image dimension is increased from eight pixels to $8 \times 8 = 64$ pixels. The data preparation steps to create the 2D training images from the original 3D shower images are visualized in figure 8.17 and explained in the following. First, a 2D slice along the x - and z -axis is cut-out of the middle y -layer of the initial 3D image, averaged over the full energy range (top left plot). This slice is shrunk from 25×25 pixels down to 8×8 pixels, summing-up to a total of 64 pixels (top right plot). To index each pixel, the 2D image is stacked-up (bottom left plot): it is iterated from $x=0$ to $x=7$, and at each value of the x -axis all eight pixels in the z -axis are stacked. As it can be noted in the bottom left plot, the training data primarily consists of two peaks, a large and a more modest one. To ease the training of the 2D qGAN model, the natural logarithm is applied to shrink these peaks and to increase the size of the side peaks. The natural logarithm is chosen since it shows the most consistent dispersion of the peak heights (bottom right plot).

In the following, first the architecture of the 2D quantum generator circuit is discussed and introduced. Next, the model training and inference processes are evaluated before a noise study for inference is carried out.

8.4.1 2D Model Circuit Architecture

The same hybrid qGAN training approach as for the 1D model is employed. However, in order to represent the larger images, the number of qubits in the quantum generator circuit is increased. To counterbalance the power of the generator, additionally, the discriminator network is enlarged by empirically adding a hidden layer with di-

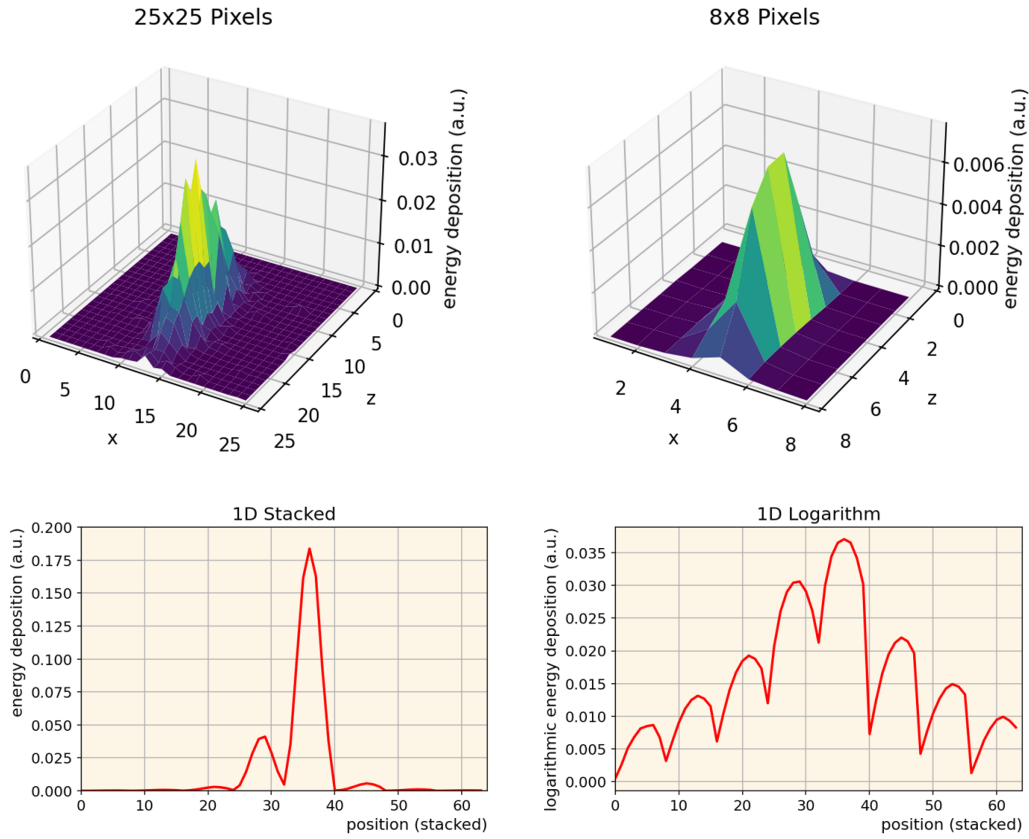


Figure 8.17: (top left) A 2D cutout of the x- and z-axis of the initial 3D calorimeter training data from the middle layer in the y-axis. (top right) The 2D cutout shrunk from 25x25 pixel down to 8x8 pixels. (bottom left) The 8x8 pixel image stacked along the z-axis to a one-dimensional distribution. (bottom right) The natural logarithm applied to the one-dimensional distribution.

mension 512 before the layer of dimension 256. For representing the $N = 64$ pixels of the shower image, the quantum generator must possess 64 quantum states. Therefore, $n = 6$ qubits are required. For the quantum generator circuit, multiple architectures are tested, and two circuit architectures are discussed in the following.

The initial tested quantum generator circuit architecture is visualized in figure 8.18. It is constructed in a linear entangling scheme, similar to the circuit operated for the 1D images before. The circuit consists of two layers and comprises three times six **RZ**-gates, resulting in 18 trainable parameters. However, the training of the model with this architecture did not achieve good results. Tests of including additional **RZ**-gates after each **RY**-gate did not significantly improve the accuracy, although, increased the training time to more than two weeks on quantum simulators. Therefore, an alter-

native architecture is required.

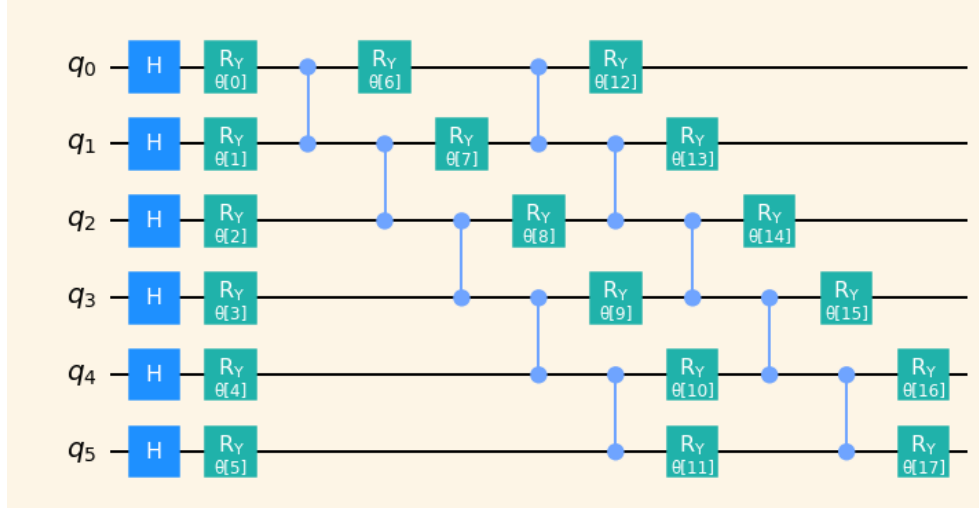


Figure 8.18: The initial 2D quantum generator circuit with a linear entangling scheme.

Taking into account the computational cost for simulating six qubit circuits, more efficient ansätze are considered, in particular hierarchical architectures. Therefore, the Tree Tensor Network (TTN) architecture from Ref. [266] is employed for the 2D qGAN model. The schematic TTN circuit is visualized in figure 8.19.

Since only six qubits are required, a cutout of the initial suggested TTN architecture is used, where from the initial eight qubit architecture, the first and the last qubit are discarded. The TTN architecture can be divided into two halves. The left side of the measurement operation in figure 8.19 is denoted as downsampling part, and the right half is denoted as upsampling part. In the downsampling part, two qubit unitaries

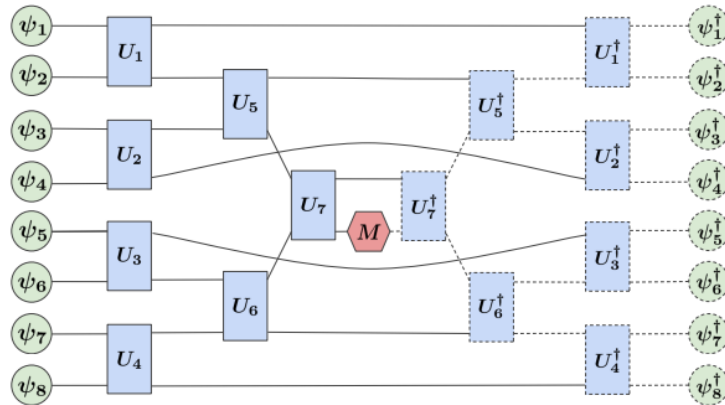


Figure 8.19: The schematic Tree Tensor Network architecture from Ref. [266].

U_i are applied, where only the information of one qubit is further used, and the other one is discarded until only one qubit remains left at the measurement. For our model, the unitarities consist of a CX -gate followed by a RY -gate on the qubit which is not discarded. No measurement operation is executed and in the upsampling part, the conjugate transpose of the unitarities are employed. The six qubits TTN architecture operated in this study is visualized in figure 8.20.

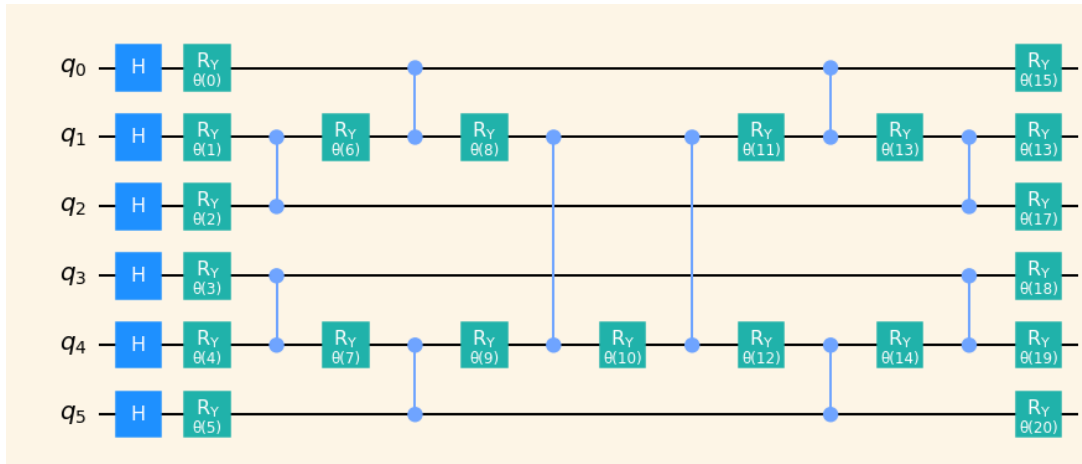


Figure 8.20: The 2D qGAN Tree Tensor Network generator circuit used in this study.

8.4.2 Training and Inference without Noise

The training is carried out for 3 000 epochs on the quantum simulator without noise. Due to the large circuit, the 2D qGAN model training remains time-consuming and, therefore, no statistics evaluation is performed. However, the training is unstable, and convergence to satisfying results is rare. The hyperparameters are the generator and discriminator learning rate of 0.001 and an exponential learning rate decay of 0.0015. The results of the best 2D training are provided in figure 8.21. The generated eight pixel shower image of the 2D qGAN model on the left indicates a general good agreement to Geant4, except for a few pixels that are slightly off. On the right, the training convergence of the relative entropy is visualized for the 3 000 epochs. It can be noted that there are intense oscillations during the first 500 epochs due to the general instability of the GAN training. After the relative entropy converged to reasonable values around epoch 1 500, the training diverges and then only slowly converges again, which is an indication of an unstable training or imperfect hyperparameters.

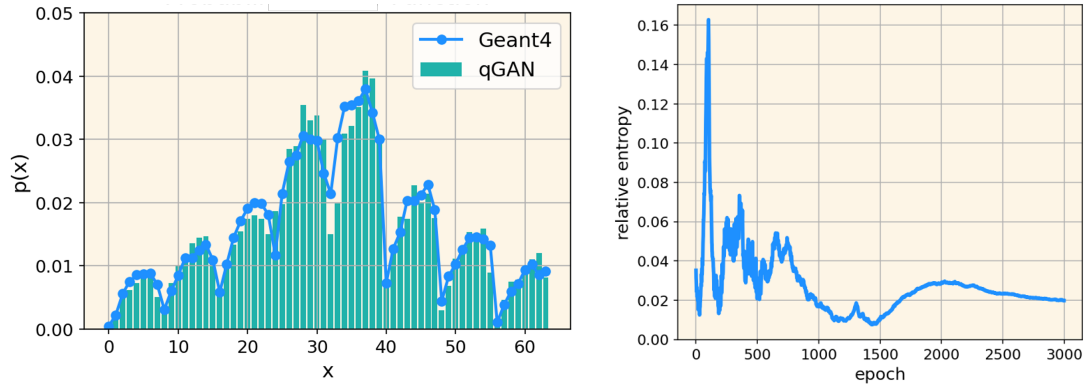


Figure 8.21: (right) The eight pixel shower image of the best trained model. (left) relative entropy during the 2D qGAN training.

8.4.3 Inference with Noise

In the following, the influence of simulated quantum hardware noise to the accuracy of the 2D qGAN model is studied, similar as for the 1D qGAN model in section 8.3.4. Inference is carried out with multiple simulated noise configurations and noise levels.

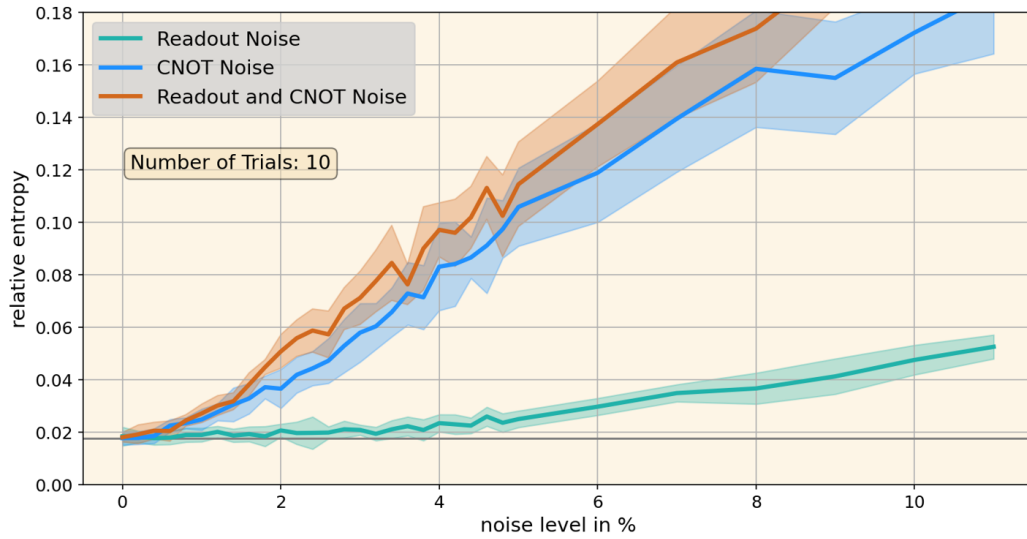


Figure 8.22: The influence of hardware noise to inference of the 2D model trained without noise. Three noise model configurations are tested.

The results of the inference noise study are visualized in figure 8.22. It can be observed, the relative entropy as a measure of accuracy of the model with exclusively readout

noise (green) increases only slightly for increasing noise. This indicates that the 2D qGAN model is almost robust to readout noise. Therefore, for the 2D qGAN model, no error mitigation configurations were tested. In contrast, the configuration with exclusive CNOT noise (blue), and the configuration with readout and CNOT noise (orange) lose accuracy already for low noise levels. Because readout noise has only a minor influence on the accuracy, the orange model with readout and CNOT noise performs slightly worse than the model with exclusive CNOT noise. No noise study for training of the 2D model is carried out, due to the large training time and the unstable training behavior.

8.5 Discussion of the Hybrid Quantum - Classical Model

In this chapter, a first hybrid quantum-classical GAN model was introduced, constructed and evaluated as a proof-of-concept for a quantum image generation of simplified calorimeter images. It has to be noted that the generated images represent only probability distributions of energies and no individual images, as the pixel entries represent probabilities instead of real energy values. After hyperparameter optimization, the 1D qGAN model converges fast and stable in training. In inference, it was able to correctly reproduce the 1D eight pixel images. In a second step, the model is enhanced and made more realistic with enlarging the resolution of the generated images to two dimensions. Due to the exponential qubit to feature scaling, doubling the required number of qubits to six allows the generation of 64 pixel distributions. The training of larger models on quantum simulators is extremely computing-time consuming and, it is challenging to achieve convergence to satisfying results. The best accuracy was achieved with operating a Tree Tensor Network architecture for the quantum generator. With further hyperparameter optimization of the 2D qGAN training process, probably better training convergence could be achieved.

The influence of quantum hardware noise to the accuracy is tested. When the simulated noise is only applied in inference, the 1D model is robust to a modest amount of combined readout and CNOT noise. For the 1D model, the accuracy decrease caused by readout noise could be entirely compensated with employing readout error mitigation. When applying simulated noise in training, the 1D qGAN model even indicated an accuracy improvement at around 1% of combined readout and CNOT noise. Similar, training the model with hardware noise on quantum simulators outperformed the noiseless training. These results verify the expectation that QML models can benefit from quantum hardware noise. Therefore, in theory, QML model can be successfully executed on noisy quantum devices. Similar observations were made in Ref. [267]. There, they reported a noise resilience of measurement and gate level noise and speculate that this phenomenon equally applies to other hybrid variational algorithms. In terms of noise, a possible future step would be examining the influence of gate-level

error mitigation techniques. For the 2D qGAN model, readout noise seems not to have a large impact on the model accuracy. This leads to the conclusion that noise behaviour of a model is probably architecture dependent. It seems for our case that larger quantum circuits are more affected from gate level errors and less from readout errors compared to smaller ones.

In the following, the computing constraints for the qGAN calorimeter simulations are discussed. As provided, already the simple three qubit 1D qGAN model is too computational expensive to be fully trained on a current physical quantum device. This is caused by multiple reasons: first, the model training is inefficient due to the parameter-shift rule, which requires an enormous number of circuit executions. Second, access to quantum devices is currently limited and the execution of jobs time-consuming. On the other hand, the hybrid quantum - classical training approach allowed partial quantum hardware training. With more efficient gradient estimation methods and with more developed quantum devices, this might change in the future. Also, the training on quantum simulators executed on classical computers faces limitations. For the six qubit 2D qGAN model it was demonstrated that the training on quantum simulators takes weeks. Furthermore, the maximum number of qubits to be simulated in Qiskit is set to 25. Therefore, operating complex and large models on quantum simulators remains challenging.

Given that training the hybrid qGAN model is computationally costly and the model can only generate energy distributions instead of real images, alternative models should be developed. Regarding future alternative models: although the parameter-shift rule is currently the most precise parameter optimization method in quantum computing, computationally more efficient optimization methods should be tested. Furthermore, preferably, the model would be a full quantum model and not a hybrid model, where no classical part, i.e. no classical discriminator neural network, is required. Finally, the ideal model should be trainable on current NISQ devices and generate precise images. In the following chapters, alternative and improved models are explored, considering the aforementioned aspects.

Thesis Objective Fulfilled:

A first quantum model was developed as a proof-of-concept to generate averaged simplified calorimeter shower distributions. This model was optimized regarding training time and accuracy and its limitations were explored. Furthermore, the impact of quantum noise on the model outcome was evaluated, and small amounts of noise during training can actually improve the accuracy.

Chapter 9

Quantum Angle Generator

The Quantum Angle Generator (QAG) represents a newly developed generative full quantum model designed to generate precise and individual images on current NISQ devices. To a full quantum model is referred when all trainable parts of a model use quantum computing, e.g. variational quantum circuits instead of classical neural networks. However, the parameter update during training in full quantum models can still be performed classically. As the name implies, the QAG model employs angle encoding to convert the generated quantum data into classical data. The angle encoding is an enhancement over the amplitude encoding of the hybrid qGAN model from the previous chapter, since it allows the QAG model to generate real individual images with energy intensities as pixel entries instead of only probability distributions. This requires only n qubits for the generation of n features, or n pixels. The core of the model consists of variational quantum circuits trained with classical objective functions that aim at generating extremely accurate images. Furthermore, the training setup is optimized for executing the entire training and inference on real quantum devices. Therefore, in addition to an extensive noise study, the QAG model is evaluated on real quantum hardware and multiple circuit architectures are compared to each other.

The content of this chapter is organized as follows. In the next section, motivations for the development of the QAG model are provided, including a literature section. Then, the QAG model and the employed encoding technique are introduced. Several circuit architectures are then compared and the advantages of the most suitable ones are highlighted. An in-depth accuracy analysis of the results follows to highlight the excellent precision of the QAG model. The investigation into the impact of quantum hardware noise is conducted by training and inference on actual quantum hardware and a comprehensive comparison with simulated noise. Lastly, conclusions and possible future plans are presented. Most of the content of this chapter is published in the

reviews in Ref. [268] and Ref. [269].

9.1 Related Work to Quantum Generative Models

Quantum computing represents a rapidly evolving field and multiple quantum generative models have been introduced, as provided in section 7.6. However, all existing models maintain limitations. Some models either do not scale well in terms of required qubits to the number of encoded features, or they do not achieve a satisfying level of fidelity. The limitations of these models are in more detail presented in the following.

The hybrid qGAN model from the previous chapter comes with the disadvantage that merely probability distributions can be generated and no absolute pixel entries, i.e. energy values. In order to overcome this obstacle of average probability distributions, in Ref. [270], a second quantum circuit is added to the hybrid qGAN model, and the new model is called dual-PQC (dual parameterized quantum circuit) GAN. The two circuits share the role of a single generator. The first circuit reproduces the image distributions, while the second generates normalized pixel intensities [270]. However, the two circuits provide a bad qubit to feature scaling and training to convergence is incredibly complicated because in the dual-PQC GAN three models have to be optimized: two quantum generator circuits and one classical discriminator neural network. Therefore, the dual-PQC approach never allowed to successfully generate images with more than four pixels. Consequently, this approach is not suitable for this study. This is because a model with a lightweight training and good scaling to larger images is searched. Furthermore, the operated qGAN model is hybrid with a classical part, but a full quantum model is preferred.

Other potential quantum generative models are for example, the Quantum Circuit Born Machine (QCBM) [249], Quantum Variational Autoencoders [271] or quantum Boltzmann machines [272]. A comprehensive summary about current quantum generative models is provided in Ref. [273]. The QCBMs are frequently applied and also tested in HEP [274]. But equally, QCBMs are not suitable for NISQ devices. Because of the discretization, QCBMs require for each generated feature a set of qubits collected into a register, where the level of discretization is determined by the number of qubits within each register. For example, the generation of one energy value – equal to one pixel – with 512 discrete energy levels requires $n = 9$ qubits, because $2^9 = 512$. For generating an image containing eight pixels, in total $9 \cdot 8 = 72$ qubits are required. On current NISQ quantum devices, with limited number of qubits, qubits decay and hardware noise this number is not reasonable. Furthermore, the vast number of qubits has to be connected by an enormous number of quantum gates to implement entanglement. On current noisy quantum devices, where each gate introduces a source of

noise, this huge number of gates would strongly reduce the output fidelity. Due to a poor scaling, the generation of larger images or more energy levels would be unfeasible. On the other hand, the in this chapter introduced QAG model requires only n qubits for the generation of n pixels.

To achieve potential quantum advantage over classical computing, a good qubit to feature scaling is required [210]. Preferable would be a better than linear scaling, where the number of required qubits is lower than the number of generated features, e.g. logarithmic scaling. On the other hand, models which employ better than linear scaling encoding techniques face disadvantages, making them unsuitable for generating precise images on NISQ devices. For example, amplitude encoding can only generate probability distributions, as demonstrated in the previous chapter. In general, encoding techniques are also constrained by the complexity of the applied state preparation function to avoid losing quantum advantage. The least possible number of quantum gates should be employed to encode the classical data into the quantum states of qubits [275, 276]. The state preparation of angle encoding scales linearly, while for example amplitude encoding demands 2^n gates for n features [276]. However, the state preparation is only for the discriminator quantum circuit of the full qGAN model in the following chapter 10 required, and not for the QAG model, because it does not employ a discriminator.

Most published quantum models do not provide a detailed accuracy analysis of their results. Most probably because they remain at the proof-of-concept stage. The QAG model introduced in this chapter is intended to overcome all of the aforementioned issues. Extremely precise images should be generated for the calorimeter simulation which will be evaluated by multiple physics accuracy metrics. Due to lightweight design and the linear scaling, the full training can and should be carried out on current noisy quantum devices.

9.2 The Quantum Angle Generator in Detail

The QAG represents a QML model that employs angle encoding to generate precise individual images. Angle encoding is a thoroughly established technique [277, 278]. However, to the author's knowledge, it has not yet been applied to quantum generative models. Angle encoding scales linearly with the number of encoded features. Thus, the generation of n features requires n qubits. For the calorimeter simulation use case, the number of features corresponds to the number of pixels present in the training image. As the current NISQ devices cannot yet handle the 25x25x25 pixel images, a down-sampled representation to eight pixels along the longitudinal shower axis is addressed in this study, similarly as for the previous hybrid qGAN model. This study focuses on images corresponding to particles entering the calorimeter within

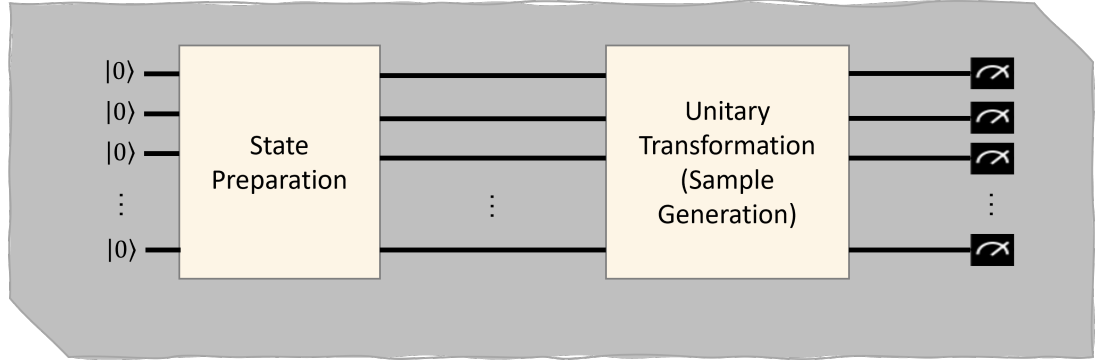


Figure 9.1: The general structure of the QAG model.

the energy range of [225, 275] GeV. The data set is split into a training and a test set. The training set consists of 1 024 and the test set of 980 images. The final preprocessed data sets are available in Ref. [279]. In the following, the QAG model is described and the way how it is trained. Subsequently, multiple quantum circuits are studied to determine the best suiting one for this research.

9.2.1 Model Description

The QAG model consists of variational quantum circuits trained by an objective function. Due to the linear scaling of angle encoding, n qubits are required for generating n features or here the n pixel images. The general model structure is schematically visualized in figure 9.1. All qubits are initialized in the basis state $|0\rangle$. The state preparation function implements the state initialization by employing specific quantum gates. At every qubit, one H -gate is applied to initiate superposition, followed by one RY -gate with the angles Ω to initiate a source of random noise. Random noise is applied to allow the model to generate new samples at each execution.

The unitary transformation constitutes the trainable part of the QAG model. Various architectures are examined in a circuit study in section 9.2.3 to obtain the best suiting circuit. The circuits employ rotational gates parameterized by trainable rotational angles and two qubit entanglement gates. As the angle encoding is primarily making use of the qubits y-axis component, predominantly RY -gates are employed. For some circuit architectures, it is assessed if additional RZ -gates or deeper circuits with depth 2 (denoted as d2) can further improve the results. Entanglement gates are employed to initiate connections between the single qubits and to entangle them. Two-qubit CX -gates are used, which are on IBMQ devices natively implemented. Other entanglement gates on IBMQ devices are composited by multiple basis gates. Keeping an eye on the goal of executing the training on a real quantum device, the

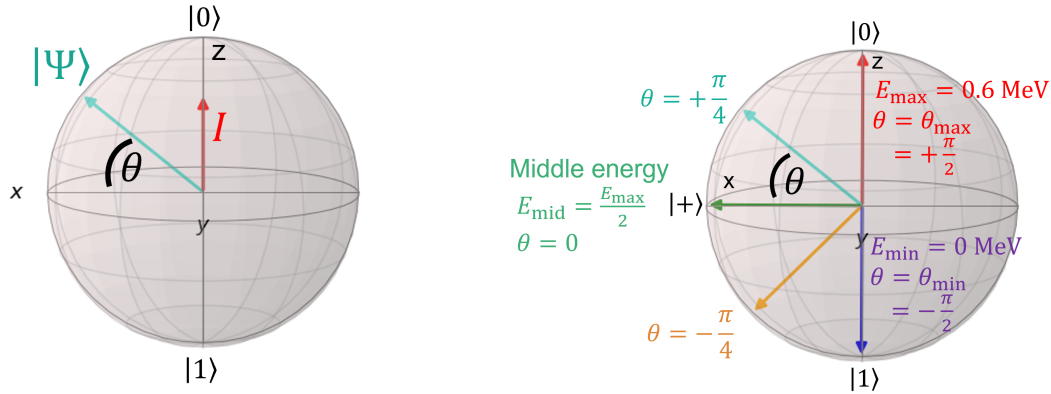


Figure 9.2: (left) Decoding of an example state Ψ into an angle θ . (right) Some representative angles θ and its corresponding energy values.

absolute number of decomposed gates should be maintained as low as possible. On NISQ devices, each gate adds another source of noise, which potentially decreases the accuracy. Therefore, the number of RY -, RZ - and CX -gates is kept as low as possible. The combination of those in smart circuit architectures allows generating accurate images, as it will be detailed analyzed in section 9.3.

Lastly, after the unitary transformation, to convert the quantum states back into classical energy values via angle encoding, the model must be executed multiple times and the quantum states measured. The number of executions is commonly known as the number of shots nb_{shots} , which counts how often state $|0\rangle$ is measured. The decoding process is illustrated in figure 9.2 (left) with a single qubit example state $|\Psi\rangle$, its intersection I and the corresponding angle θ , and described in the following. The scalar intersection I of the vertical axis on the Bloch sphere (z-axis) is calculated with:

$$I = 2 \cdot \frac{\text{counts}(|0\rangle)}{nb_{\text{shots}}} - 1. \quad (9.1)$$

In the subsequent step, the intersection I is transformed into the rotational angle θ by the trigonometric function:

$$\theta = \frac{\arcsin(I)}{1}. \quad (9.2)$$

The denominator of equation 9.2 is one, because the Bloch sphere is a unit sphere. The angle θ operates in the x-z-plane of the Bloch sphere and is defined as zero in the $|+\rangle$ state. Rotating θ clockwise leads to positive angles. The angle θ can be transformed into a pixel energy E by the following linear change of ranges equation:

$$E = \frac{(E_{\text{max}} - E_{\text{min}})(\theta - \theta_{\text{min}})}{\theta_{\text{max}} - \theta_{\text{min}}}. \quad (9.3)$$

$E_{\max}$, $E_{\min}$, $\theta_{\max}$ and $\theta_{\min}$ are defined in figure 9.2 (right): the minimum energy $E_{\min} = 0$ MeV is set to the $|1\rangle$ state at the bottom of the Bloch sphere ($\theta_{\min} = -\pi/2$) and the maximum energy is arbitrarily set to $E_{\max} = 0.6$ MeV at the $|0\rangle$ state at the top of the Bloch sphere ($\theta_{\max} = +\pi/2$). With $E_{\min} = 0$ and $\theta_{\max} = -\theta_{\min}$ equation 9.3 can be simplified into:

$$E = \left(\frac{E_{\max}}{2 \cdot \theta_{\max}} \right) \cdot (\theta + \theta_{\max}). \quad (9.4)$$

For multi-qubit quantum circuits: the final quantum states of all qubits are measured and the outcomes decoded independently. Although the qubit results are individually decoded, the state of each qubit is entangled to others due to the entangling gates applied within the variational quantum circuit, as explained later. The pseudo code for generating a shower image with the QAG model is provided in the appendix B.2.

It is worth to be noted that the angle θ and therefore, the corresponding decoded energy E remain discrete values. Since θ depends on the number of shots nb_{shots} : the larger nb_{shots} , the better the achievable energy precision and resolution. Fortunately, with present quantum devices, the nb_{shots} can be easily chosen to be large. Currently, on IBMQ devices the maximum possible number of shots is $nb_{\text{shots}} = 100\,000$. For the simplified calorimeter use case, this is more than sufficient. As shown in the reduced precision research in chapter 6, 256 discrete energy levels are sufficient for correctly reproducing the original 3D calorimeter shower images with dimension of 25x25x25 pixels.

In fact, the largest pixel energy present in the training data is larger than 0.6 MeV, which is previously set to the maximum energy. However, only a modest fraction of the pixels contain energies larger than 0.6 MeV. For simplification purposes a pixel energy cut at $E_{\max} = 0.6$ MeV is employed, and larger pixel energy values are set to $E_{\max}$, to avoid outliers. In fact, a similar strategy is employed in the Intel Neural Compressor reduced precision tool, where outliers are removed to better utilize the available int8 values and to achieve better accuracies, as described in section 6.3.1. The pixel energy distribution of the entire training and test data set is visualized in figure 9.3. The pixel energies are plotted in an arbitrary unit (a.u.), due to the down-scaling process. It can be noted that the training and test data set maintain almost an equivalent pixel energy distribution. Further, an enormous number of pixel energies falls into the first bin with extremely low or zero energy deposition. For larger pixel energies, the number of hits is progressively decreasing. In future studies, it could be considered to shift $E_{\max}$ to a higher value.

Theoretically, $\theta_{\max}$ can be chosen as $\frac{\pi}{2}$ (≈ 1.57), as indicated in figure 9.2. However, to ensure a smoother decoding and to avoid edge effects, the θ range is restricted to the green region in figure 9.4. The reason is, at large and small angles in the marginal

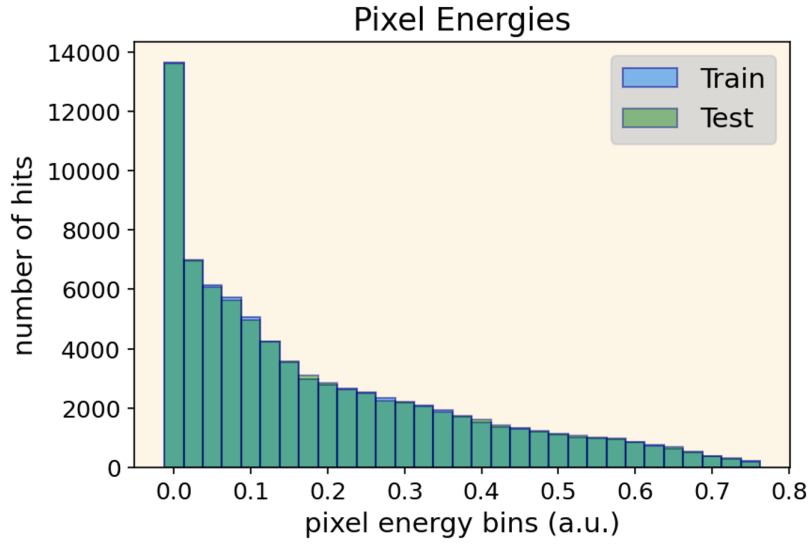


Figure 9.3: The pixel energies of the training and test data set plotted as a histogram. On the x-axis the energy bins and on the y-axis the number of pixel counts within each energy bin are shown. The bars representing the training data align closely with those of the test data, and are often obscured by them.

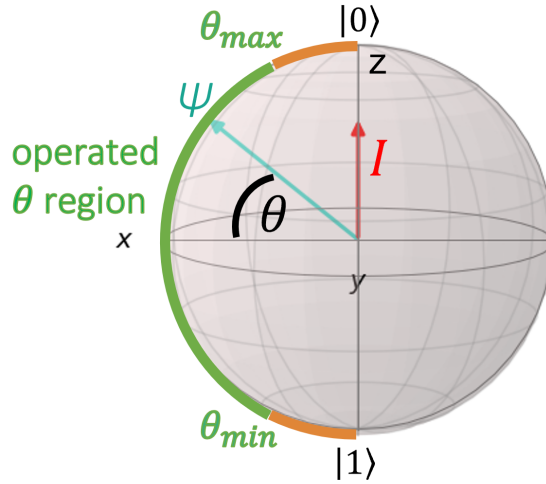


Figure 9.4: Visualizes in green the operated θ region on the Bloch sphere. The orange regions are excluded.

regions close to the $|0\rangle$ and the $|1\rangle$ state (colored orange), the decoding into the intercept I is inaccurate, because a significantly large change in the angle θ results in a tiny change in the intercept I . In other words, the dependence of the decoding function

(equation 9.2) becomes too flat and with this the uncertainty too large. Therefore, the chosen θ range is adopted to ensure most precise decoding. $\theta_{\max}$ is arbitrarily chosen to be $\theta_{\max} = \frac{\pi}{2} - \frac{\pi}{16}$ (≈ 1.37 , the condition $\theta_{\max} = -\theta_{\min}$ remains).

9.2.2 Objective Training Function

The QAG model is trained with a composite objective function built of two terms: the first represents the Mean Maximum Discrepancy (MMD) loss [280], which is already successfully applied by other quantum models, for example the QCBM [249], and describes the distance between two samples from the training distribution P and the generated distribution Q :

$$\text{MMD}^2(P, Q) = \| \mathbb{E}_{X \sim P}[\phi(X)] - \mathbb{E}_{Y \sim Q}[\phi(Y)] \|_{\mathcal{H}}^2. \quad (9.5)$$

$\mathbb{E}$ represents the expectation value and the function ϕ maps X and Y to a high-dimensional reproducing kernel Hilbert space $\mathcal{H}$ (RKHS) [280, 281]. X represents a batch of samples from the training distribution ($X \sim P$) and Y one or more generated sample(s) of the generator distribution ($Y \sim Q$). In practice, it is computationally less demanding to operate in a less-dimensional feature space by applying the kernel trick of Ref. [282]. With the kernel function

$$k(x, y) = \langle \phi(X), \phi(Y) \rangle, \quad (9.6)$$

the feature map ϕ of the inner product can be transformed into a feature space by employing c Gaussian kernels:

$$k(x, y) = \frac{1}{c} \sum_{i=1}^c \exp\left(-\frac{1}{2\sigma_i} |x - y|^2\right). \quad (9.7)$$

This turns the MMD from the RKHS $\mathcal{H}$ into the real space $\mathbb{R}$ [282]:

$$\begin{aligned} \text{MMD}^2 &= \mathbb{E}_{X \sim P, X' \sim P}(X, X') + \mathbb{E}_{Y \sim Q, Y' \sim Q}(Y, Y') \\ &\quad - 2 \mathbb{E}_{X \sim P, Y \sim Q}(X, Y), \end{aligned} \quad (9.8)$$

making the MMD efficiently computable. The moments σ_i determine the width of the Gaussian kernel. Similarly as for the QCBM, the moments $\sigma = [0.01, 0.1, 1, 10, 100]$ are applied, therefore, $c = 5$. Using the kernel trick, the MMD results in zero if and only if the distributions are identical [283]. Otherwise, the lower the MMD, the more similar the distributions.

Training classical generative models with the MMD loss results in simpler and more stable training [284]. One drawback is that the MMD loss is computationally expensive, especially for large sample (image) sizes.

A second term to the objective training function is an auxiliary loss denoted as correlation (Corr) loss and aids the model to learn the complex patterns present in the training data (e.g. image pixel correlations), similar as it is done in classical ML models (for example in section 4.4.3). Including the Corr loss results in excellent accuracy, even analyzing the details of calorimeter shower images, shown in the later section 9.3. The Corr loss is calculated by the pixel-wise mean squared error between the pixel correlation values present in the training data and the ones of the generated data.

For creating the pixel-wise correlation plot the entire data set is used and the dependence of the single pixels to the others calculated. This is done by the Pearson product-moment correlation coefficient – as described in Ref. [285] – with the command `corrcoef` [286] from the NumPy library [287]. For demonstration, the pixel-wise correlation plot of the Geant4 training data is illustrated in figure 9.5: it clearly shows a larger positive correlated group of pixels on the top left (from pixel 0 to 4) and a more compact positive correlated cluster on the bottom right (pixel 5 to 7). The other pixels are negatively correlated. The physics interpretation of the correlations present in the shower images is very difficult, and in this thesis only the agreement between Geant4 images and the generated images by the QAG model is evaluated. As a side note: in the appendix B.6 the eight pixel correlation plot is compared to the eight pixel covariance plot in figure B.3. Furthermore, the correlation plots for the initial 25x25x25 pixel images and the generated images by the classical ML GAN model are shown in the appendix A.4 in figure A.2.

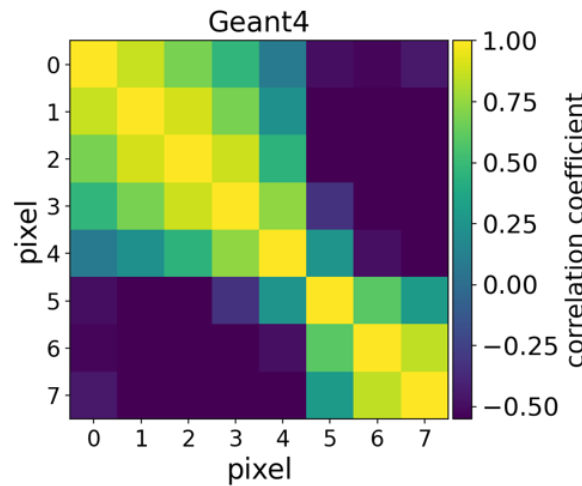


Figure 9.5: The pixel-wise correlation plot of the Geant4 training data.

To train the QAG model, the Simultaneous Perturbation Stochastic Approximation (SPSA) [288] optimizer is employed. The SPSA optimizer makes use of finite-difference methods by employing a stochastic optimization process and requires only

two optimization steps per epoch. IBMQ recommends using the SPSA optimizer because it can achieve good results in the presence of noise with a reasonable number of quantum machine calls [289]. In comparison, gradient based methods (such as the Adam optimizer) require twice the numbers of trainable parameters function calls due to the parameter shift rule, as explained in chapter 8.

An extensive hyperparameter search is done using the Optuna library [163]. All tests in this study are executed in Qiskit [262] version 0.26.2. The models are trained for 500 epochs. Each quantum job contains 512 shots for training and inference. The generator SPSA optimizer learning rate is set to $c_0 = 1$ with an exponential learning rate decay of 0.006 beginning from epoch 50. The dynamic MMD loss weight starts with value one and decays linearly with the decay rate of -0.001 per epoch, beginning from epoch 100. On the contrary, the Corr loss weight increases by the same value starting with zero. The dynamical training batch size is set to one in the first hundred epochs and afterwards to 20 in order to calculate the Corr loss between multiple images. These values have proven to lead to stable training results. The evolution of the MMD and Corr loss weights is visualized in figure 9.6.

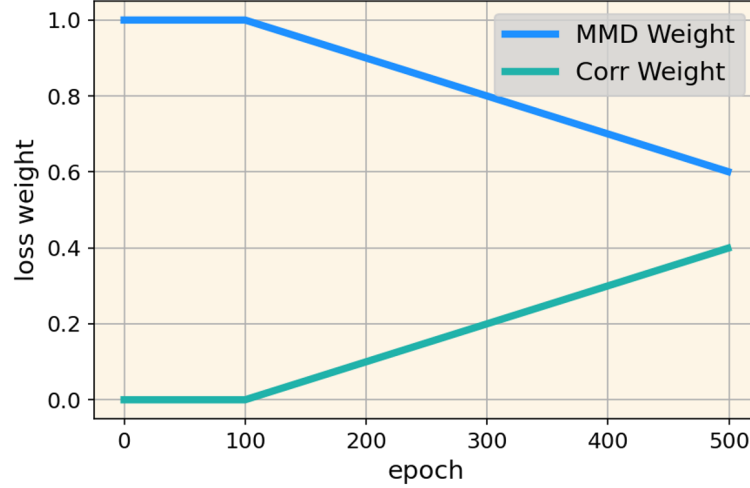


Figure 9.6: The MMD and Corr loss weight evolution during training of the QAG model. After hundred epochs a linear decay is implemented for the MMD loss, and a linear increase for the Corr loss by a factor of 0.001. These functional dependencies have proven to lead to stable results.

9.2.3 Quantum Circuit Study

A quantum circuit study is performed to obtain the best suiting architecture. The ideal circuit should contain a moderate number of parameters to achieve a sufficient level of accuracy. Different circuit architectures are compared to each other based on characteristic values expressing the power of the circuit.

Quantum Circuit Architectures under Study

The investigated quantum circuit architectures are summarized in figure 9.7. In general, hierarchical architectures, such as TTN (Tree Tensor Networks), solve our problem best, while maintaining a reasonable number of quantum gates and parameters. This observation was already obtained for the 2D QGAN model in section 8.4. In detail, the TTN architecture and Multi-scale Entanglement Renormalization Ansatz (MERA) introduced in Ref. [266] are investigated. Multiple variations of these circuits are tested: 1) circuits with a depth two (naming scheme "d2"), which contain two layers to evaluate if deeper circuits perform better. 2) Circuits with RZ -gates are employed after each RY -gate (naming scheme "Rz") to assess if rotations around an additional axis promote an advantage. Both architecture variants double the number of parameters of the initial circuit.

For the MERA architecture, further variations are analyzed where only one half of the original MERA architecture is implemented. Both circuit halves are considered, and the middle RY -gate with parameter number 22 at the border line is contained in both circuits. The left circuit half is denoted as the MERA downsampling (MERA-down) circuit and highlighted in red. The reason for the name is that it can be imagined that the information from all qubits becomes downsampled to only one qubit. In the MERA-down circuit from figure 9.7, the remaining qubit to which the information is compressed represents qubit number two (the third line from the top; indexing starts with zero). On the contrary, the right circuit half is denoted as the MERA upsampling (MERA-up) circuit, highlighted in green. In this case, the information is upsampled from the center qubit and spread to all the other qubits, similarly to what happens in classical ML generative models. On the other hand, the MERA-down circuit compresses the information as typically done in classification tasks. Therefore, this architecture is tested (chapter 10) as a discriminator architecture for the full quantum GAN model. The last tested circuit (top right) contains a simple linear entanglement strategy to explore if simple circuits can compete with more complex ones. This architecture was previously in this thesis employed for the 1D QGAN model in chapter 8, where only three qubits were operated. All basic circuits under investigation are provided in figure 9.7. The extended circuits under study with a higher depth and additional RZ -gates are listed in the appendix in table B.1.

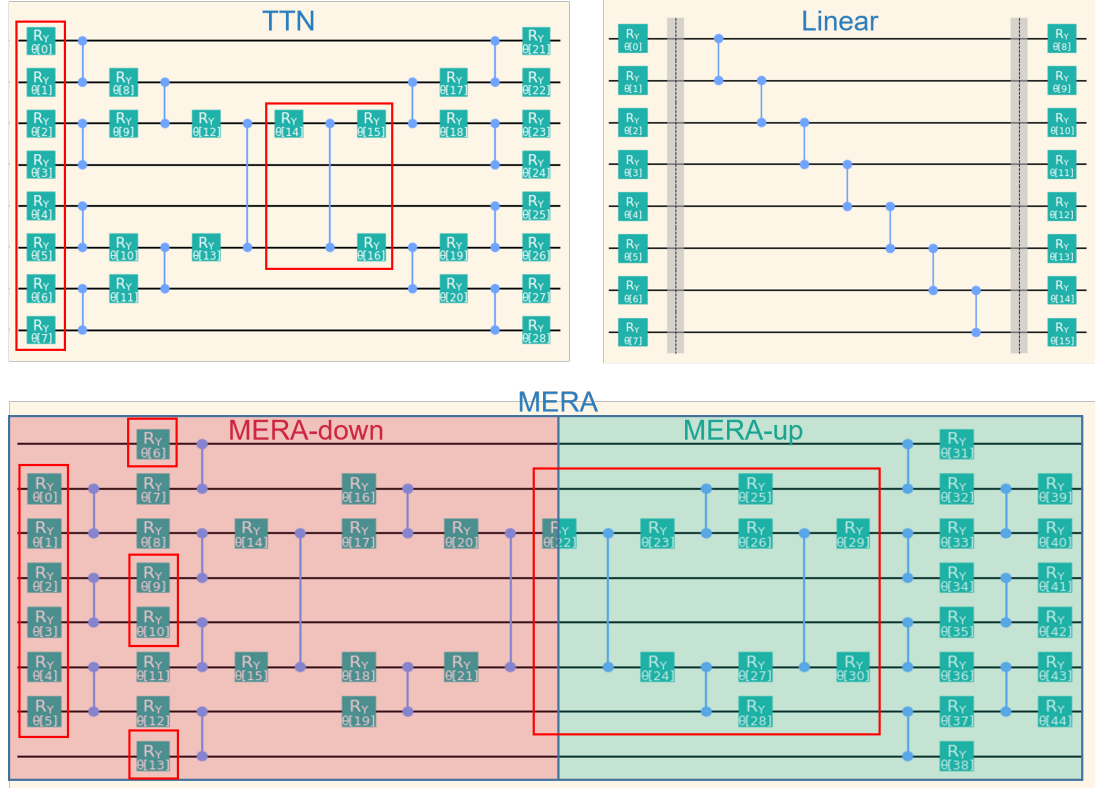


Figure 9.7: The basic quantum circuit architectures under study. Top left: TTN architecture, top right: linear entanglement architecture, bottom: the full circuit corresponds to the MERA architecture, where the red highlighted cut-out on the left corresponds to the MERA-down architecture and the green highlighted cut-out on the right to the MERA-up architecture. The red boxes highlight some of the rotational gates which are used for the manipulated angle metric later in figure 9.11.

Characteristic Circuit Numbers

The characteristic circuit numbers are the number of trainable parameters N_p , the expressibility $\mathbf{X}$ and the entanglement capability $\mathbf{E}$. Larger circuits with more trainable parameters are potentially capable of achieving more accurate results. However, at some point a saturation could be reached where the same task can be solved in similar accuracy by a smaller circuit. $\mathbf{X}$ and $\mathbf{E}$ are used as defined in Ref. [290]: $\mathbf{X}$ describes how well the circuit can represent the pure states of the representative Hilbert space. For a single qubit the expressibility exhibits how many states of the Bloch sphere can be represented. In the tests in this thesis, $1 - \mathbf{X}$ is measured. The closer $1 - \mathbf{X}$ to 1 the better the expressibility of the model, while the closer to 0 the worse. One aim is to generate highly entangled quantum states with low-depth circuits to correctly

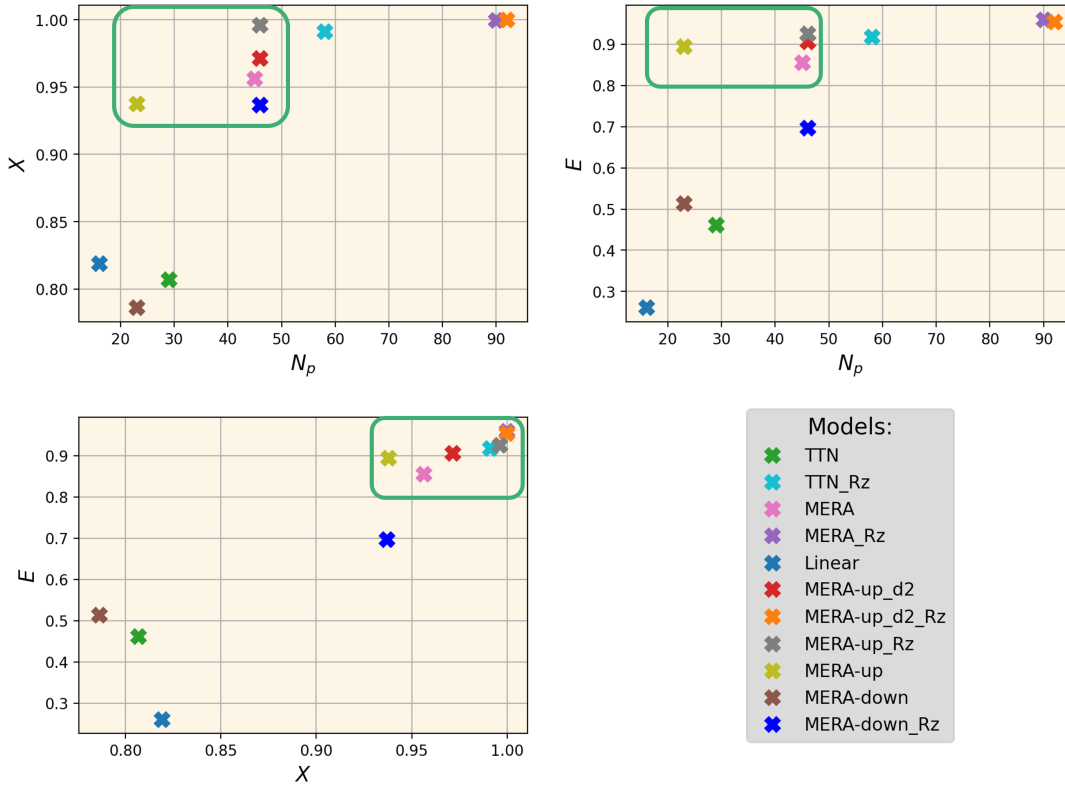


Figure 9.8: The characteristic circuit numbers for various architectures. On the top left X is plotted as a function of N_p , on the top right E versus N_p , and on the bottom left E versus X . For best visibility, some axes have been suppressed and the regions of good values marked with green boxes. The individual circuits with their extensions are described in the appendix B.3.

reproduce the training data. The entanglement capability E is a measure which expresses the circuit's ability to generate entangled states between the qubits. Likewise, E ranges from 0 to 1, where 1 represents the best achievable value.

The results of evaluating the circuit characteristic numbers (N_p , X and E) are presented in figure 9.8. On the bottom left, E is plotted over X : the best performing circuits lie in the top right of the plot. MERA_Rz and MERA-up_d2_Rz perform best and are almost on top of each other, directly followed by MERA-up_Rz and TTN_Rz. The linear architecture, the basic TTN and the MERA-down perform worst. It can be clearly noted that circuits with more gates (Rz, d2) perform better. N_p is studied in the two top plots in figure 9.8. It can be noted that the best two circuits – MERA_Rz and MERA-up_d2_Rz – contain more than 90 parameters, by far the largest number of parameters. However, a more limited number of parameters is desirable for taking NISQ hardware limitations into account. Because more gates/operations lead to more

Table 9.1: The characteristic circuit values N_p (number of trainable parameters), X (expressibility) and E (entanglement capability) are listed for each employed circuit. Additionally, the statistic values mean and standard deviation and the training times t in minutes of the MSE accuracy metric are provided for the individual circuits for repeating the training 25 times on the calorimeter data set.

Circuit Name	N_p	X	E	MSE	t in min
Linear	16	0.8191	0.261	0.00113 ± 0.0008	35.0 ± 0.2
TTN	29	0.8068	0.462	0.00245 ± 0.0012	41.5 ± 0.6
TTN_Rz	58	0.9912	0.918	0.00440 ± 0.0022	85.9 ± 1.5
MERA	45	0.9564	0.855	0.00220 ± 0.0011	51.3 ± 0.5
MERA_Rz	90	0.9997	0.959	0.00634 ± 0.0036	121.5 ± 9.6
MERA-up	23	0.9377	0.894	0.00059 ± 0.0004	39.4 ± 2.9
MERA-up_d2	46	0.9715	0.906	0.00038 ± 0.0002	50.5 ± 0.2
MERA-up_Rz	46	0.9961	0.926	0.00047 ± 0.0005	75.2 ± 7.5
MERA-up_d2_Rz	92	0.9999	0.954	0.00094 ± 0.0008	116.2 ± 4.3

noise. The MERA-up_Rz and TTN_Rz architectures contain approximately half the number of parameters compared to the MERA_Rz and MERA-up_d2_Rz circuits, but perform almost similarly, as demonstrated in figure 9.8. Therefore, in practice, they may be preferred. Further reducing the number of parameters, the MERA-up architecture with 23 parameters still maintains adequate values for E and X . These tests reveal that deeper circuits than for example the MERA-up_d2_Rz would not provide further advantages, because the maximum expressibility and entanglement capability are already reached with less levels and gates. All measured characteristic circuit numbers are summarized in table 9.1.

Circuit Evaluation for Calorimeter Simulations

Similarly as in the previous classical ML and hybrid qGAN studies the MSE which is calculated as the pixel-wise distance between the average generated shower image and the average training data is employed as accuracy metric and shown in figure 9.9. The training is repeated 25 times for each circuit, and the MSE and the standard deviation are plotted. To minimize the influence of outliers, the best and worst two trials are discarded in the analysis. The MSE is plotted as a function of: N_p on the top left, X on the top right and E on the bottom left. By inspecting the plots, it can be recognized that the MSE does not correlate with any of the characteristic circuit values in the plots. The models seem to "arbitrarily" scatter without any observable systematic pattern.

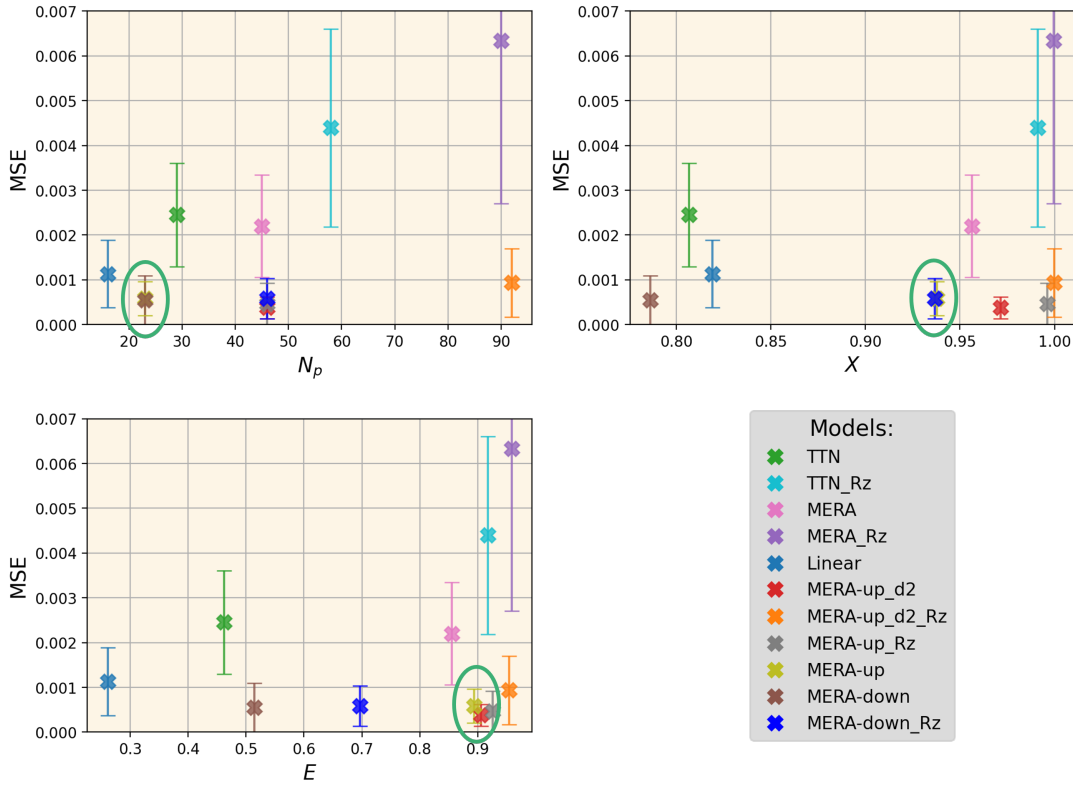


Figure 9.9: The accuracy of the various trained circuit architectures depending on the circuit characteristic numbers. The MSE is plotted as a function of N_p (top left), X (top right) and E (bottom left). Because MERA-up is performing well, but gets hidden by the values of other circuits, its values are marked with green circles.

By scrutinizing the average MSE values of the particular models in figure 9.9, it can be noted that the MERA-up, MERA-down, MERA-up_d2 and MERA-up_Rz architecture perform best with the lowest MSE values. This is in agreement with the observation that they maintain a high X and E , except for the MERA-down architecture. The MERA-up architecture, showing the lowest N_p while keeping good values in X and E , is also very good in the MSE. It has to be noted that the light green cross for MERA-up is hidden in some plots by the values of other circuits and, therefore, highlighted by green circles. In addition, the models with the highest X and E , MERA_Rz and MERA-up_d2_Rz, respectively, perform worse. The linear entanglement circuit performs better than the basic TTN architecture. The error bars of the model MSEs in figure 9.9 provide a hint about the training stability. It can be observed that the better the MSE of the models, the smaller its standard deviation.

In figure 9.10, the Corr loss is visualized for the various models. Although it is denoted as a loss, it can be equally employed as an accuracy metric. It is calculated for

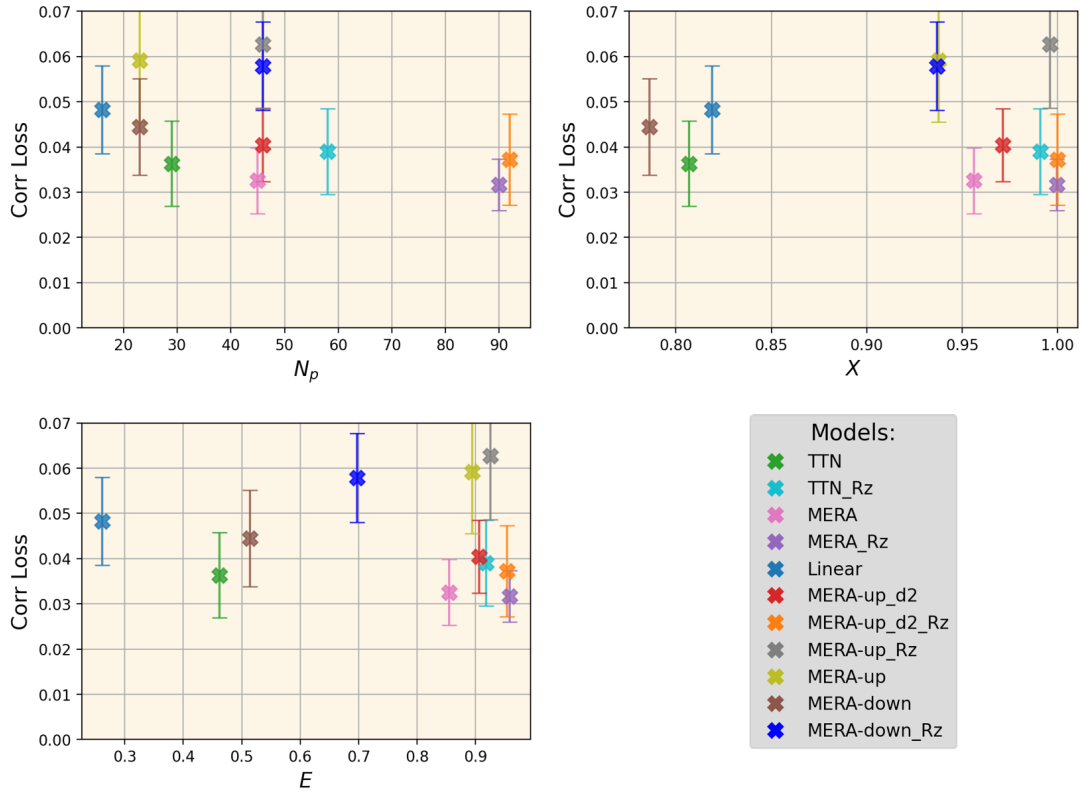


Figure 9.10: The correlation loss of the various trained circuit architectures as a function of the circuit characteristic numbers is provided. The correlation loss is plotted as a function of N_p (top left), X (top right) and E (bottom left).

980 inference shower images and compared to the entire test set. With the Corr loss, it is especially studied if circuits with higher entanglement capability are able to achieve higher pixel correlations in the generated images. However, when evaluating the results in figure 9.10, no correlation between the Corr loss and any of the characteristic circuit numbers can be observed, similar as for the MSE in the previous section. All circuits achieved good results in the Corr loss. As the Corr loss represents only a less important metric, the details of the individual models are not further discussed. The MMD loss as a function of the characteristic circuit values for the various models in the circuit study is provided in the appendix B.4 in figure B.4.

The classical DL method of dropout [127] was studied: both layer-wise and weight-wise implemented dropout resulted in worse accuracies making them unsuitable for the quantum circuits in this study. Probably larger quantum circuits with more trainable gates would be required to benefit from dropout.

In the circuit angle manipulation metric, the influence of the rotational param-

ters of specific gates on the individual pixel outputs is tested. Only one rotational parameter θ_i of the circuit is manipulated at a time, the others are kept constant. The rotational parameter θ_i is manipulated by adding a small value γ to the corresponding trained parameter, with iterating over six γ values: $\gamma \in [+ \pi/4, - \pi/4, + \pi/8, - \pi/8, + \pi/16, - \pi/16]$. With the new parameter $\theta_i^* = \theta_i + \gamma$, $N = 20$ images are generated containing the pixel energies $E_{\theta_i^*}$. Then, the absolute difference Δ between the average manipulated image and the average original image is computed pixel-wise:

$$\Delta = \frac{1}{6 \cdot N \cdot M} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{\gamma} |E_{\theta_i} - E_{\theta_i^*}|. \quad (9.9)$$

The original images with the pixel energies E_{θ_i} are generated without any parameter manipulation. The distance measure Δ to the original image corresponding to an energy change per pixel. This distance can be used to examine the effect to the manipulated gate on all qubit energy outputs. This is used to assesses how well the manipulation propagates within the quantum circuit, ultimately leading to a measure of entanglement. This test is employed for the TTN, the MERA, the MERA-up and the MERA-down architecture. In addition, for each architecture, the results are averaged over all the trained models from the circuit study, excluding the two best and worst models, resulting in $M = 25 - 4 = 21$ models.

The results are provided in figure 9.11. In case of overflows, where the change of the pixel energy values is larger than 0.2 (in an arbitrary energy unit), they are cut and set to the maximum value in yellow. It can be observed that the MERA-up and the MERA architecture provide the largest number of influential gates (more pixels in lighter colors), while the TTN and MERA-down architectures are less impacted by variations of individual gate changes.

Going more into detail, the manipulated gate numbers of all circuits correspond to the parameter indices θ in the architectures in figure 9.7. The manipulated gates were chosen to be primarily the first few gates within a circuit as well as gates on highly entangled qubits centralized in the middle of the circuit. Due to the large number of manipulated gates, not all outputs are individually discussed and, therefore, only the major noticeable findings are explained. In the TTN architecture, it can be noted that pixel six is hardly influenced by any of the manipulated gates. It is not even influenced by the manipulated gate number six which is directly operated on qubit six. In the MERA architecture, a higher influence to the neighboring pixels of the manipulated gates can be observed. In general, it can be noted that the farther the qubits are apart from the qubit on which the manipulated gate is operated, the lower the influence, leading to a fading entanglement.

In the MERA-up architecture, the numbering is chosen according to the MERA circuit numbering in figure 9.7. However, only the right half of the circuit is used. Therefore,

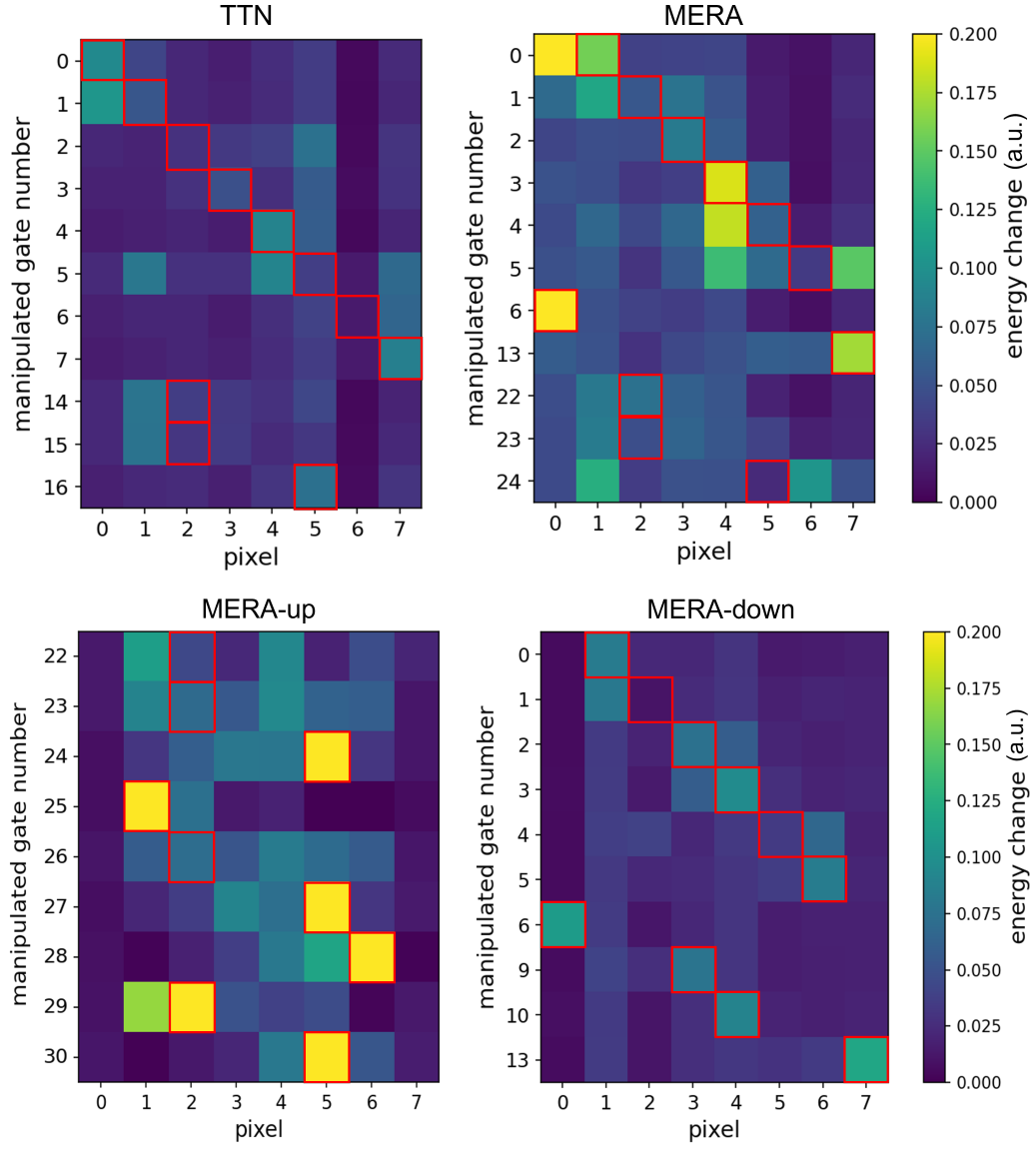


Figure 9.11: The results of manipulating individual gates for multiple circuit architectures. The results for the TTN architecture are provided on the top left, for the MERA on the top right, for the MERA-up on the bottom left and for the MERA-down on the bottom right. The energy change of all plots employ the same color bar to make them comparable. The energy change is plotted in an arbitrary unit. Red boxes highlight on which qubit or pixel the manipulated gate operates. The manipulated gate numbers of the circuits are highlighted in figure 9.7 by the red boxes, corresponding to the parameter indices θ .

the RY -gate with parameter number 22 corresponds to the first gate within the circuit. It can be marked that the influence of the manipulated gates is relatively large in comparison to the other circuits. However, pixels zero and seven are only weakly influenced, but also, there is no manipulated gate directly operated on the corresponding qubits, opposite to the previous circuits. Contrary, the MERA-down architecture overall provides a weak influence by the manipulated gates, similar to the TTN architecture.

The influence of the manipulated gate to all pixels, provides a practical impression about the entanglement in the circuit architectures. It makes the entanglement visible to us and verifies the principle of quantum circuits. The "(v)erification of the functioning of the individual gates and qubits (obeying the principles of quantum computing), especially the creation of entanglement, is a dedicated topic in quantum information theory" [291] to ensure the correctness and reliability of quantum computations. The observed results agree with what is measured by the characteristic number of the entangling capability of the individual circuits. Both the MERA and MERA-up architecture achieve good entangling capabilities, whereas the TTN and the MERA-down architecture perform worse.

All in all, considering the characteristic circuit values and the achieved accuracy with the MSE in training, the MERA-up R_z circuit performs best. However, giving a larger weight to N_p , the plain MERA-up circuit performs almost as well, while only needing half the number of parameters. Additionally, in the metrics for manipulating individual rotational angles, the MERA-up architecture achieved the best results. Therefore, the following studies in this thesis employ the MERA-up circuit for training the QAG model. This also pays tribute to the fact that fewer parameterized gates are better suited for NISQ hardware limitations because less noise is created and the circuit execution is faster, preventing decoherence. The observation that there is no correlation between any of the characteristic circuit numbers and the resulting accuracy values hints to the potential conclusion that these theoretical metrics maintain only a limited meaningfulness in practice and have to be studied application dependent.

9.3 In-depth Accuracy Analysis

In this section, the results of the QAG model with the MERA-up circuit architecture are analyzed. All known accuracy metrics for the calorimeter image use case are evaluated. Some of the metrics have previously been employed in the classical ML GAN simulations. All tests are operated on a noise-free quantum simulator. The tests including quantum hardware noise are provided in the subsequent section.

9.3.1 Training Process Evaluation

Initially, the statistics of the loss functions during training are investigated. The unweighted loss functions and the weighted loss functions are plotted in figure 9.12 and 9.13, respectively. The mean of twenty training repetitions is visualized as a thick solid line and the standard deviation (STD) as a colored band around the mean. The weighted loss values are multiplied by their dynamic loss weights, while the unweighted are not. The evolution of the loss weight is plotted as dashed lines in the weighted loss plot in figure 9.13. The Corr loss does not begin to affect the training until epoch 100 because its weight is set to zero before then.

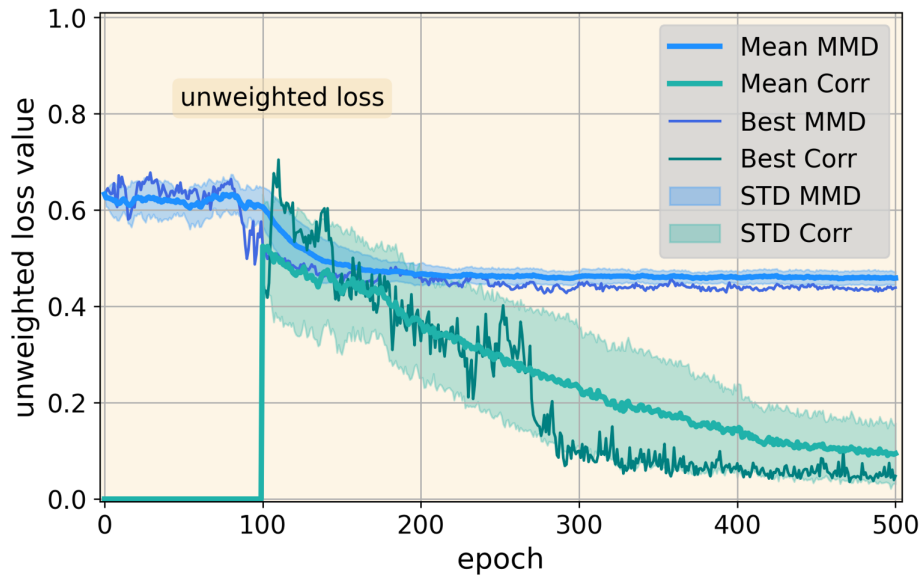


Figure 9.12: The unweighted loss values are displayed for all training epochs.

The unweighted plot in figure 9.12 presents that the MMD loss of all trials converge similarly, as indicated by a moderate STD. In addition, the MMD loss of the best trial does not oscillate and deviate much from the mean. Towards the end of the training (at epoch >200), the MMD loss level remains stable and the STD band narrow. On the other side, the Corr loss maintains a wider STD band that narrows towards the end of the training. Overall, the MMD and Corr loss converge smoothly without intensive oscillations, which is a desirable characteristic for ML training. However, at the end of the training, the Corr loss still decreases. This suggests that longer training potentially further increases the correlation accuracy. However, in order to keep the training time and iterations as small as possible to execute the training on the real hardware later, the number of training epochs is not increased.

Figure 9.13 illustrates the individual loss contributions. The MMD loss contributes far

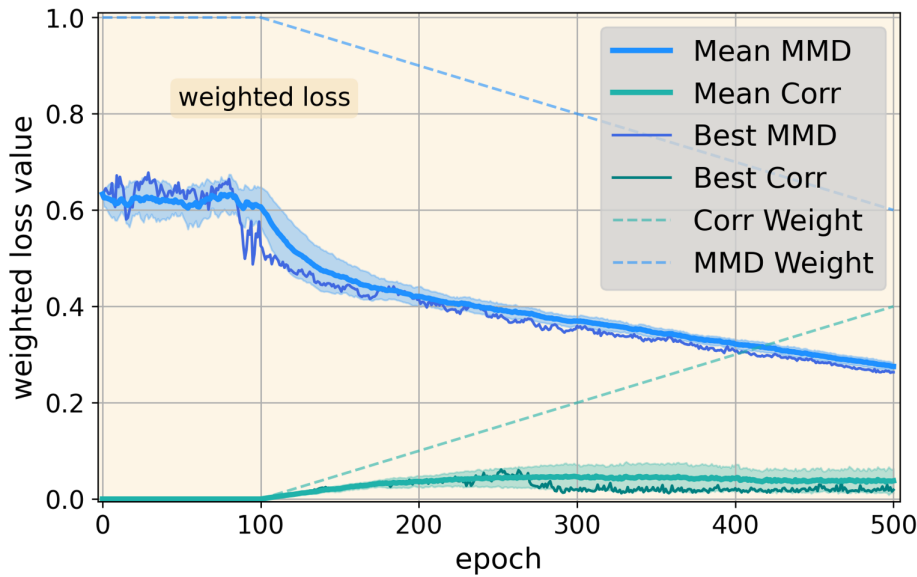


Figure 9.13: The weighted loss values are displayed over the training epochs. Additionally, the dynamic loss weights are plotted as striped lines.

more than the Corr loss throughout the entire training. The weighted MMD loss decreases during training: first, because the loss decreases due to convergence, and second, because the loss weight decreases over the epochs. The Corr loss increases from epoch 100 to about epoch 200 and then remains at a stable level. This stable level can be maintained because the increasing Corr loss weight compensates for the decreasing loss due to convergence. Although the influence of the MMD loss during training is greater than the influence of the Corr loss, the Corr loss plays a significant role in achieving a good physics accuracy in the generated shower images. During model development, it was obtained that implementing the Corr loss already at the beginning of training in epoch zero prevents the model from achieving convergence. This is probably because too many things would be simultaneously trained, what causes difficulties in combination with the SPSA optimizer. Further, implementing the Corr loss weight as a step function, where the Corr loss weight remains constant after implementation, did not result in good accuracies either. Only by smoothly increasing the Corr weight by a linear function resulted in the desired training convergence. To reach this result, an extensive and lengthy hyperparameter search was performed using Optuna.

9.3.2 Inference Evaluation

In the following, the accuracy of the best trained model is evaluated in detail. The generated shower images are compared to the Geant4 test data set consisting of 980 images (referred to as Geant4). Similarly, there are 980 images generated by the QAG model to create the accuracy metrics.

1. Average calorimeter shower shape

The first metric represents the calorimeter shower shape plot displayed in figure 9.14. The x-axis shows the pixels and the y-axis shows the corresponding averaged energy depositions. The energy is given in an arbitrary unit (a.u.) due to the downsampling during the image preprocessing. The shower shape is perfectly reproduced by the QAG model. The Mean Squared Error (MSE) is considered as an additional accuracy metric. It is calculated by the pixel-wise MSE between the average Geant4 and QAG images of the entire generated and test data set. The MSE reaches 0.00017 ± 0.0005 . The detailed equations of the MSE calculation can be found in the appendix B.5.1.

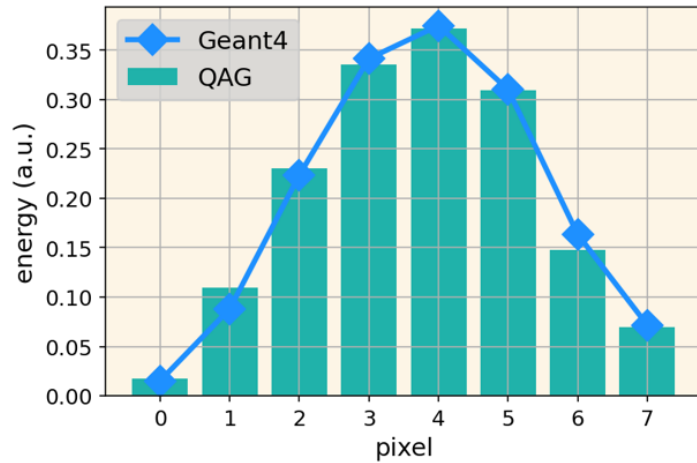


Figure 9.14: The generated shower shapes are compared. The energy on the y-axis is provided in an arbitrary unit (a.u.) due to the initial image downsampling process.

2. Pixel-wise correlation

The second accuracy metric corresponds to the pixel-wise correlation plots. The positive or negative correlations between all pixels are determined. The baseline represents the correlation plot from Geant4, which is the same as displayed separately in figure 9.15 on the left, and reveals the complex data structures present in the training data. Allowing a direct comparison the correlation plot for the generated data from the QAG model is presented in figure 9.15 on the right. It can be concluded that the QAG model accurately reproduces the overall correlation pattern. Inspecting the particular details, different color shades indicate some minor deviations. However, the achieved

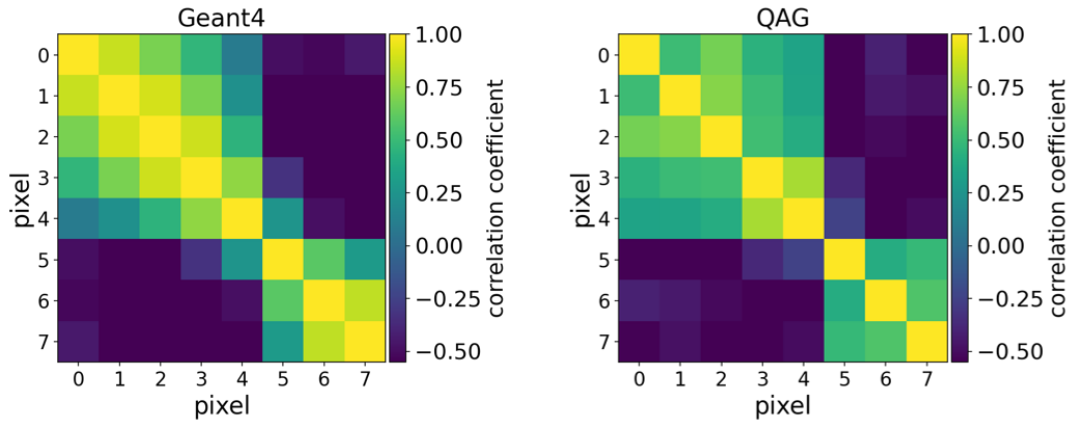


Figure 9.15: The left plot refers to the pixel-wise correlation plot of Geant4 and the right to the one of the generated data by the QAG model.

correlation precision by QAG is astonishing and more than sufficient for a calorimeter fast simulation task. In terms of the quantum perspective, this result proves that quantum circuits are capable of precisely implementing complex correlation patterns between the separate qubits. In order to achieve this, the quantum circuits require a substantial entanglement strategy, such as that present in the MERA-up architecture.

3. Energy sum

The energy sum histogram represents the third accuracy metric. The energy sum E_{sum} corresponds to the total measured energy within a shower image, which is obtained by summing-up the energy depositions of all cells. In figure 9.16, the energy sum of all images is visualized in a histogram. On the x-axis, the energy sum is plotted in an arbitrary unit and on the y-axis the number of entries for each bin are provided, where each entry corresponds to the total energy contained in a shower image. The histogram of Geant4 is visually perfectly reproduced by the QAG model. The visual agreement is confirmed by the histogram values mean μ and standard deviation σ provided in the figure. The QAG model is able to reproduce the events of the full energy range excellently, including the low and high energy events. The equations for calculating μ and σ are provided in the appendix B.5.2.

4. k-means Cluster

The fourth metric allows to evaluate whether the training images contain specific modes which are not correctly generated by the QAG image set. It is a well-known problem that classical ML models often fall into mode collapse, where they can only generate certain image modes, as explained in chapter 4. In this test, it should be observed if this is also true for the QAG model. To find image modes, the shower images are clustered using the k-means algorithm [292], where the clusters found correspond to the image modes. It should be observed whether the images generated by the QAG

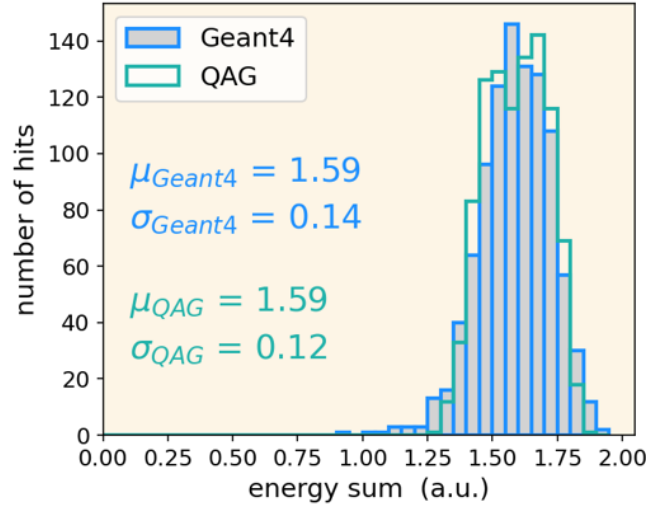


Figure 9.16: The energy sum histogram including the statistical values.

model reveal similar clusters or image modes as present in the Geant4 data.

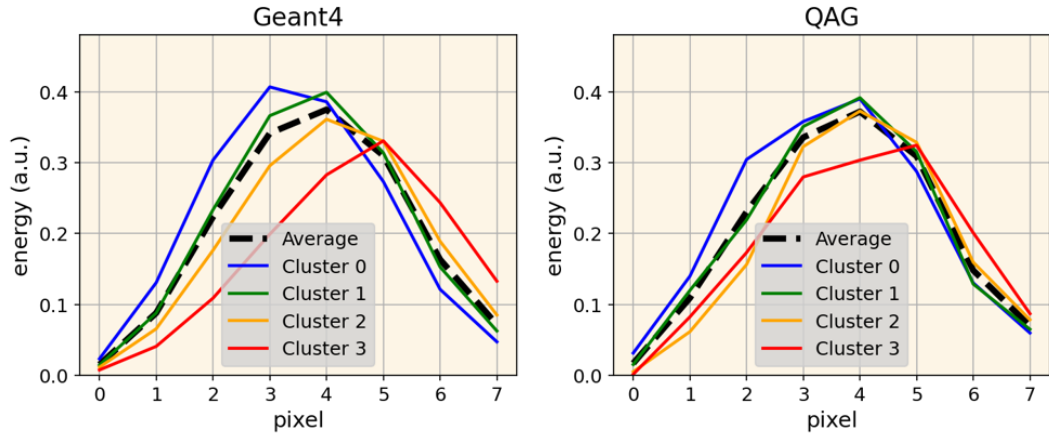


Figure 9.17: The four k-means clusters present in the images of Geant4 (left) and of QAG (right).

As baseline, the Geant4 data is clustered to find four clusters as illustrated in figure 9.17 (left). The number of clusters is chosen in order to obtain a large inter-cluster distance. A larger number of clusters than four would reduce the inter-cluster distance, making the visual cluster distinction increasingly difficult. It can be observed that the shower images of the cluster zero deposit substantially larger fractions of energy in earlier calorimeter cells than shower images of higher cluster numbers. Further, the particles from cluster zero and one occupy a larger energy, estimated by the integral area under the curves. It will be interesting to see if the QAG model can reproduce

this behavior. The black striped lines in figure 9.17 correspond to the average shower images, which are the same as in figure 9.14.

The four clusters identified in the QAG data are provided in figure 9.17 (right). They appear slightly less smooth, but a similar structure can be observed. Therefore, the general structure is correctly learned by the QAG model. Such a cluster evaluation of calorimeter shower images has never been carried out before. Applying this clustering metric to full-scale ML calorimeter simulations to investigate how precisely they can reproduce the clusters along the three spatial shower axes of the data would be an interesting future test.

5. Pixel-wise energy distribution

The last employed metric examines the pixel-wise energy deposition distributions for all eight pixels as illustrated in figure 9.18. The deposited energies of the individual pixels are investigated separately. The histograms for Geant4 and QAG are constructed, to visually compare the histogram shape, histogram mean, and width of the generated images to those of Geant4. As it can be seen in figure 9.18, the agreement is good. For example, the longer tail towards the left side of pixel four is equally present in the QAG histogram. In general, the QAG histograms overlap with the corresponding Geant4 histograms. This indicates that the average energy returned by each pixel is correct.

6. Example shower images

For completeness, some exemplary shower images are provided. Ten random shower images are drawn from the Geant4 data set and from the images generated by the QAG model and visualized in figure 9.19. It can be noted that the images of the QAG model look similar to the images from Geant4.

9.4 The Effect of Quantum Noise

In the current NISQ era, the relatively high hardware error levels are one of the primary limitations to effectively employ algorithms on real quantum devices. QML models appear to be noise resilient to some degree of hardware errors, as demonstrated for the hybrid qGAN model in the previous chapter. In the following, the noise robustness of the QAG model is tested in inference and training. Furthermore, training and inference of the QAG model are executed on real quantum devices.

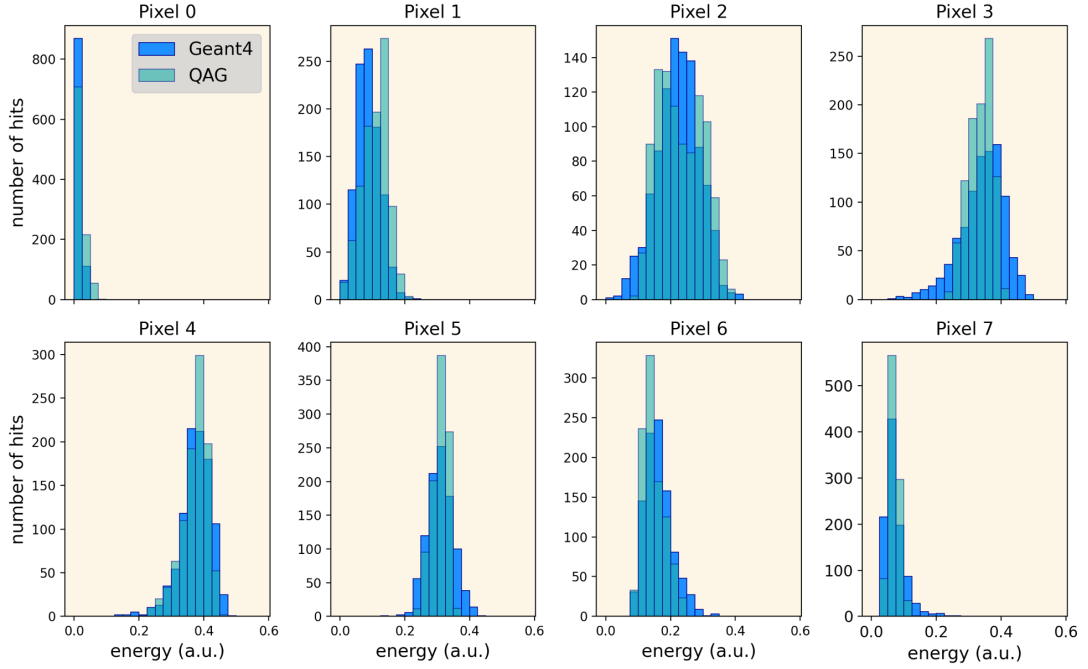


Figure 9.18: The pixel-wise energy deposition histograms for Geant4 and the QAG for all pixels. The x-axis represents the energy depositions in one pixel and the y-axis the number of images containing the corresponding energy.

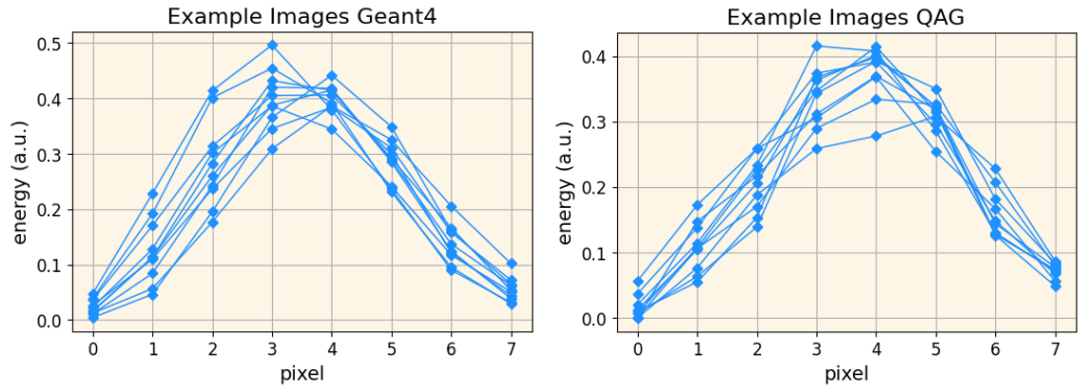


Figure 9.19: Ten random example shower images of Geant4 on the left and generated images by the QAG model on the right.

9.4.1 Noise Effect on the Inference Process

The further investigated baseline model represents the best trained model introduced in section 9.3.2, which was trained without any noise. Inference is performed with

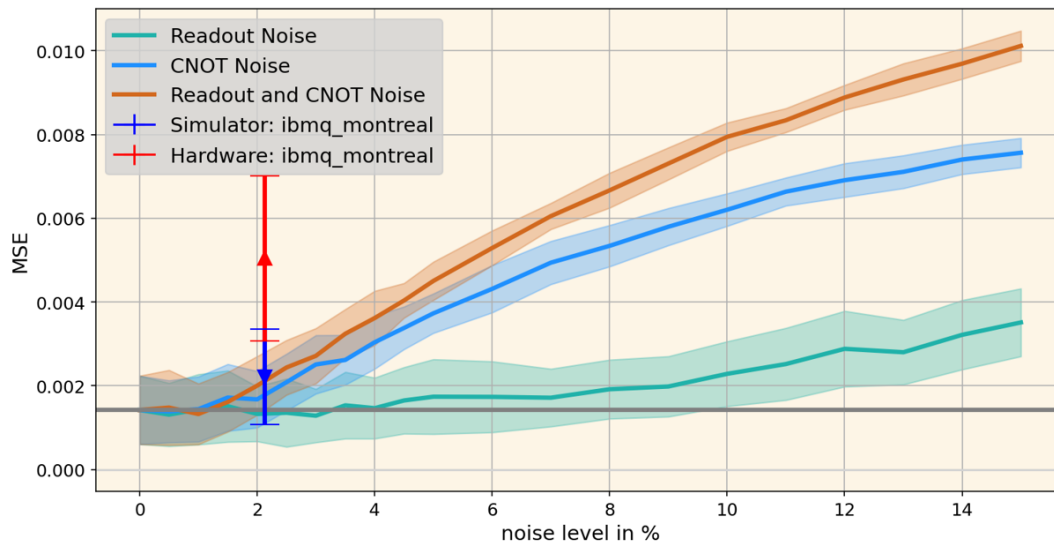


Figure 9.20: Inference noise study. On the x-axis the noise levels are provided and on the y-axis the MSE as accuracy metrics (the lower, the better). In the green configuration of readout noise, all qubits have the same readout noise level, and in the blue configuration of CNOT noise, all two-qubit gates have the same noise levels. The orange configuration combines readout and CNOT noise, where the readout noise and the two-qubit noise levels are the same. Inference is run with various noise configurations on the quantum simulator, on real hardware and on the simulator with the given hardware noise settings.

simulated noise, with simulated hardware noise, and finally on the real quantum hardware. In the simulated noise configurations, each qubit noise is modeled with the same readout measurement error and each inter-qubit connection with the same two-gate level (CNOT) error. In contrast, the noise levels on the real hardware can vary widely, and of course there are different levels of noise between qubits and also two-gate errors. Furthermore, the real hardware noise model takes decoherence and qubit control errors into account.

Several noise configurations and error levels from zero up to 15% are tested. The MSE is utilized as an accuracy measure. The results are illustrated in figure 9.20. For each configuration, the average value of 20 generated images is plotted as a solid line and the standard deviation as a colored band around the mean. The gray horizontal line serves as accuracy reference for the noise-free configuration. All configurations maintain the accuracy up to approximately 1.5% of noise. The configuration with readout noise only (green) suffers the least and maintains a stable accuracy up to 8% of noise. Only CNOT noise (blue) and both readout noise and CNOT noise combined (orange) have a stronger effect. The combined configuration (orange) performs worst,

as expected. As a side note, current quantum devices have average noise levels below 5%, which is expected to gradually decrease further. But as discussed in the following, the noise levels are unstable and can sometimes spike up. Therefore, wider noise ranges are envisioned.

Next, inference is run by loading the real hardware noise model from `ibmq_montreal` into the simulator. The `ibmq_montreal` device consists of a Falcon r4 processor with 27 qubits. The average readout noise over the qubits employed at the time of the test was 2.51% and the average gate level noise 0.97%. The explicit noise model, containing the entries for each qubit, is provided in figure 9.21. The relatively large readout noise level is primarily caused by a single qubit with 8% of readout noise. The result is included in figure 9.20 (blue triangle). The noise level position (x-axis) is determined as the average of readout and CNOT noise. Although the noise levels of the qubits vary strongly, the measured accuracy of the hardware noise simulation agrees well with the simulated noise in mean and standard deviation within the uncertainties.

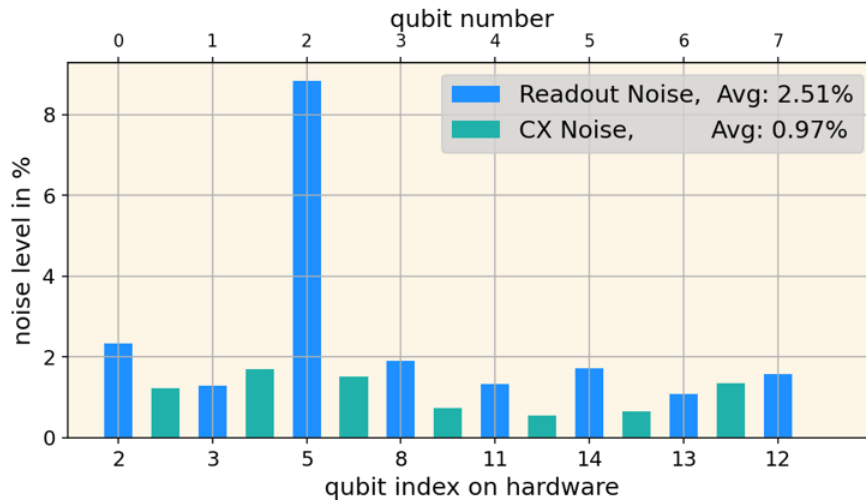


Figure 9.21: The plot visualizes the noise levels of the `ibmq_montreal` device at the time the inference test was executed. Especially, qubit number two in the circuit architecture, which corresponds to qubit five on the quantum device, has a large readout error (blue bar). The two qubit gate noise levels are plotted as green bars between the two connected qubits.

Finally, the inference is executed on the real `ibmq_montreal` device. The qubit mapping from the quantum circuit to the real quantum device is visualized in figure 9.22. To keep the number of swapping operations low when implementing the circuit on the hardware, consecutive qubits have to be chosen. The qubits are selected according to the lowest noise values.

The result is plotted as red triangle in figure 9.20. The accuracy on the real hardware

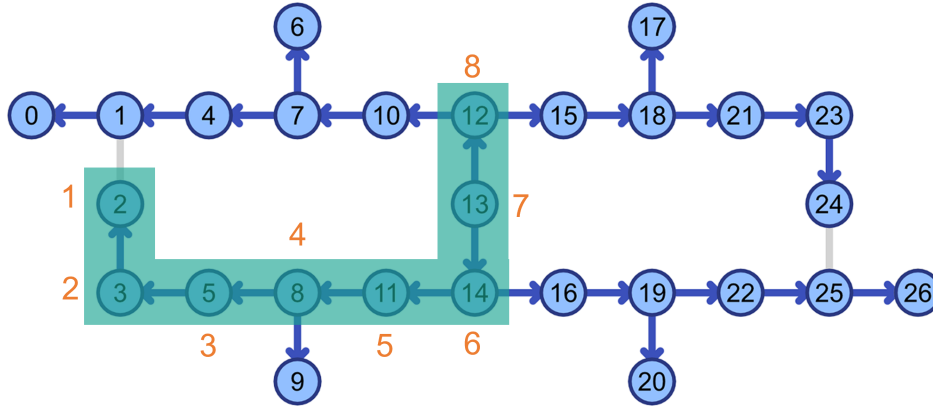


Figure 9.22: Qubit mapping from the quantum circuit to the real quantum device. Only the eight green highlighted qubits are used. The blue numbering corresponds to the qubit numbering on the quantum device and the orange numbers indicate the qubit numbering in the quantum circuit. The original graphics is taken from the IBMQ webpage and was modified according to the used circuit.

is worse than predicted by the hardware noise simulation (blue triangle), indicated by a larger MSE value. The implementation of the circuit on the real hardware involves swapping operations that imply additional two-qubit entanglement gates for the quantum circuit. It is possible that these are not taken into account in the hardware noise simulation and lead to higher noise influence on real hardware and thus to worse results. It is also possible that the noise present on the real hardware is worse than the estimated model of the simulator [219] Or it could be that some error sources - such as thermal fluctuations or qubit cross-talk - are not included in the noise model. For demonstration, the generated average shower images of all utilized noise configurations are provided in the appendix B.7, exhibiting the robustness against read-out noise up to 10% and the strong influence of two-qubit gate noise already at around 2%.

9.4.2 Noise Effect on the Training Process

In the following, the noise study is repeated to include the training, in order to understand whether the QAG model can learn how to compensate for the noise, especially when running on the real quantum device. It is also investigated, up to which noise levels the model can maintain a reasonable accuracy.

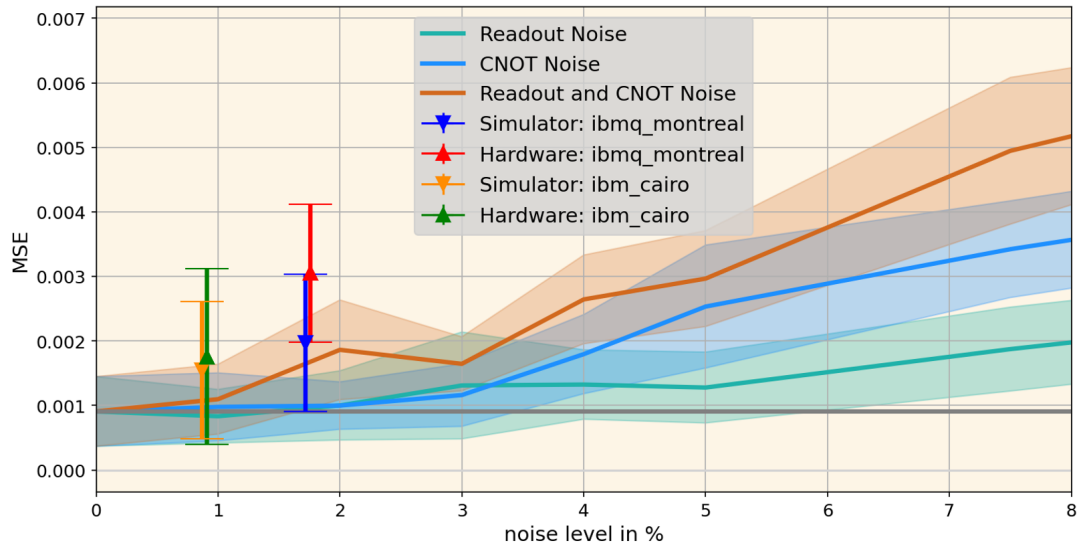


Figure 9.23: Training noise study. Training is run with various noise configurations, inclusive hardware noise simulations and training on real hardware. In the green configuration of readout noise, all qubits have the same readout noise level, and in the blue configuration of CNOT noise, all two-qubit gates have the same noise levels. The orange configuration combines readout and CNOT noise, where the readout noise and the two-qubit noise levels are the same.

The training is carried out with various simulated noise levels and configurations, each repeated ten times. The results are shown in figure 9.23: the configurations with only readout (green) and only CNOT noise (blue) maintain a similar level of accuracy until approximately 3%. The accuracy of the combination of readout and CNOT noise (orange) marginally decreases already from 1% noise. However, up to 3% of noise, the accuracy is still close to the noiseless case. All is still agreeing within the uncertainties. This can indicate that training the model with noise makes the QAG model more robust to noise than applying noise alone to a trained model during inference.

Next, two quantum devices are simulated. `ibmq_montreal` and `ibmq_cairo`, which are both 27 qubit devices, but `ibmq_cairo` has the more advanced Falcon r5.11 processor. Likewise, the training is repeated ten times and the results are added as blue and orange points in figure 9.23. It can be noted that the average accuracy in terms of the training with hardware noise performs slightly worse than the combined noise model (orange). The strong overlap between the error of the simulated noise (orange band) and the hardware noise simulation (orange and blue error bar) indicate that the accuracy difference is not significant. The error bars of the hardware noise simulations are larger than the ones of simulated noise, indicating the introduction of instabilities in the model.

Finally, the entire training is executed on the real quantum devices. First, the training is carried out on `ibmq_montreal`. During training, an unpredicted significant noise change occurred. The readout noise of one qubit increased up to 8% (the same noise levels which were present in the inference test in figure 9.21). This negatively influenced the training outcome. However, after the noise change during training, the model recovered and adapted to the new noisy environment, which is an astonishing observation. Training was also performed on the newer `ibmq_cairo` device without any hardware calibration change during the training. The noise levels of the hardware training are listed in appendix B.8. The losses of the `ibmq_montreal` hardware training are provided in figure 9.24 on the top and for `ibmq_cairo` on the bottom. In the `ibmq_montreal` training, the MSE loss spikes up around epoch 280, due to the previously mentioned calibration change. After that, it decreased, but could not reach the loss level from before the calibration change (probably because of the chosen hyperparameter configuration with a decaying learning rate for higher epochs and the numbering of remaining epochs was not sufficient). On the other hand, the Corr loss was converging to a modest level. In the `ibmq_cairo` training, the MSE and the Corr loss converged to reasonable values.

The results of the two hardware training are included as red and green triangles in the figure 9.23. The error bars correspond to the MSE deviations within 50 generated validation images. It can be observed that the average MSE of the `ibmq_montreal` hardware training (red triangle) is higher than the noise simulation (blue triangle). This is most likely due to the large noise change and the fact that the model did not have enough time to fully recover. This is also indicated by the still decreasing losses towards the end of the training, visualized in figure 9.24. The training on the `ibmq_cairo` machine is completed without calibration changes and with lower hardware noise levels (average readout noise 0.86% and CNOT noise 0.89%). The accuracy of the hardware training on `ibmq_cairo` (green triangle) is only marginally worse than that of the simulator (orange triangle). This indicates that the simulated and real hardware outcomes behave identically at low hardware noise levels, or that the noise model of `ibmq_cairo` is better (or more realistic) than for `ibmq_montreal`.

Comparing the absolute MSE magnitude of the real hardware training (≈ 0.002 and ≈ 0.003) with the one from inference from the previous section (≈ 0.005), the MSE improved slightly. Suggesting that the QAG model is able to adapt its parameters to the noisy hardware to improve its precision. This is also verified by the `ibmq_montreal` training, where the accuracy dropped entirely after the calibration change, but then recovered.

In figure 9.25 (left), the average shower image created by the `ibmq_montreal` device is visualized, and (right) the one of `ibmq_cairo`. Due to limited hardware access, only 100 shower images are generated for validation. The shower image for `ibmq_cairo` matches well with Geant4, while `ibmq_montreal` indicates some discrepancies. The

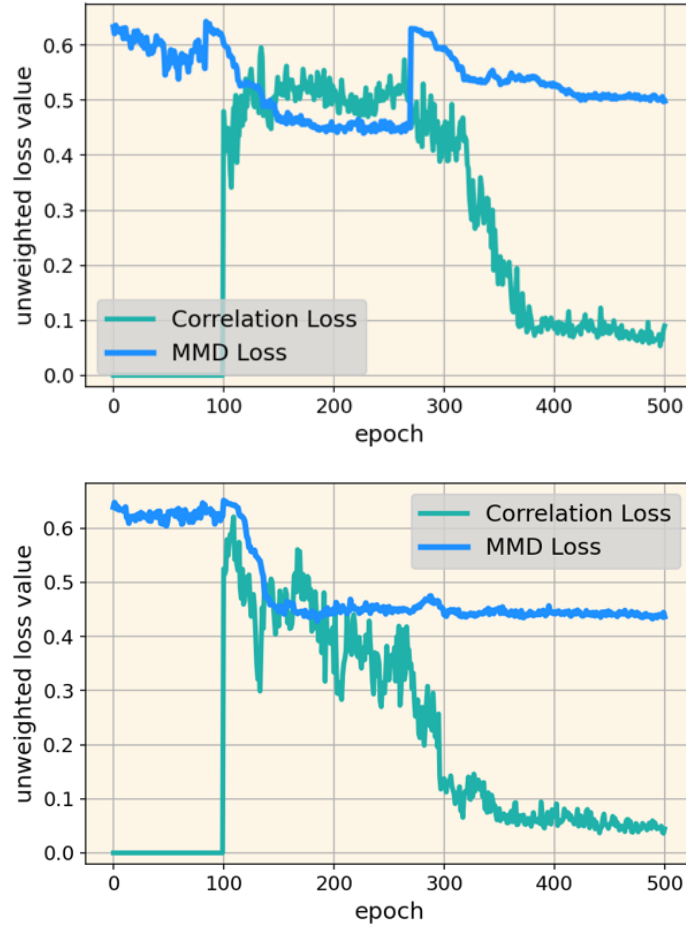


Figure 9.24: The losses during the `ibmq_montreal` training on the top and the losses of the `ibmq_cairo` training on the bottom.

correlation plot for the `ibmq_cairo` training is provided in figure 9.26. The overall correlation pattern is correctly reproduced. In the right plot of figure 9.26, the energy sum histogram is plotted. It can be noted that the model trained on `ibmq_cairo` reproduces the Geant4 histogram almost perfectly. Therefore, all in all, the model trained on `ibmq_cairo` performs accurately. Due to the calibration change on the `ibmq_montreal` device, the training accuracy is not satisfying. However, the training recovered some accuracy, and further training might have improved the results. Therefore, the detailed accuracy plots are not further investigated nor shown.

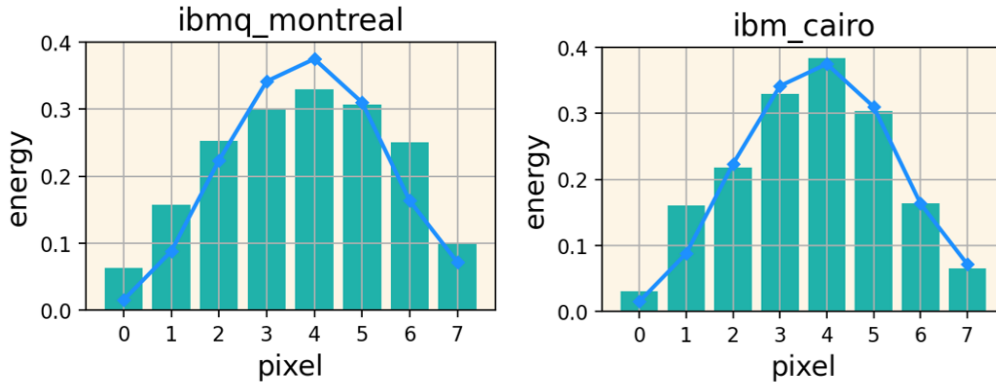


Figure 9.25: On the left are the average shower images from the QAG model trained on `ibmq_montreal` and on the right on `ibmq_cairo`.

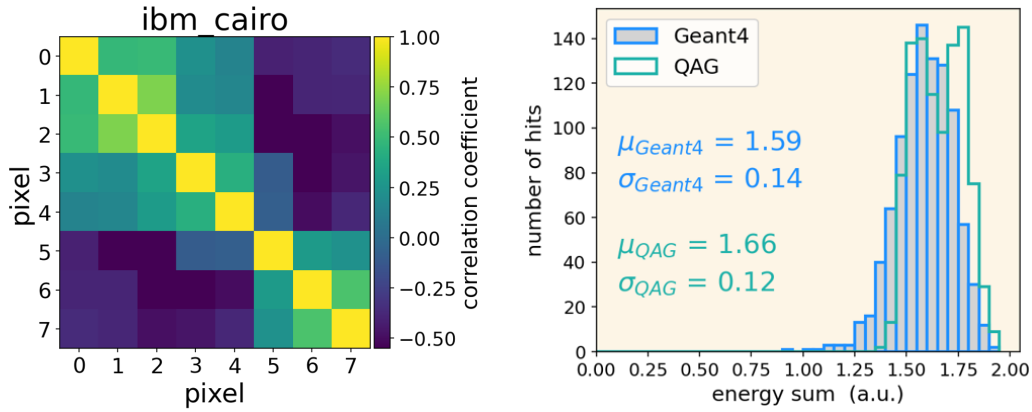


Figure 9.26: (left) The correlation plot for the `ibmq_cairo` training and (right) the energy sum histogram.

9.5 Number of Shots Dependency

The number of shots represents a crucial factor of the QAG model. As explained earlier, it determines the degree of discretization or quantization of the generated data. For our use case, the discretization of the QAG output can be interpreted as the energy resolution, which constitutes an important quantity for physics calorimeters [293]. In this subsection, the influence of the number of shots on the training and inference accuracy of the QAG model is assessed.

The training is carried out with a variable number of shots. The accuracy of the trained models is measured using the previous introduced MSE. In figure 9.27, the average MSE and its standard deviation are visualized in the form of error bars for five tri-

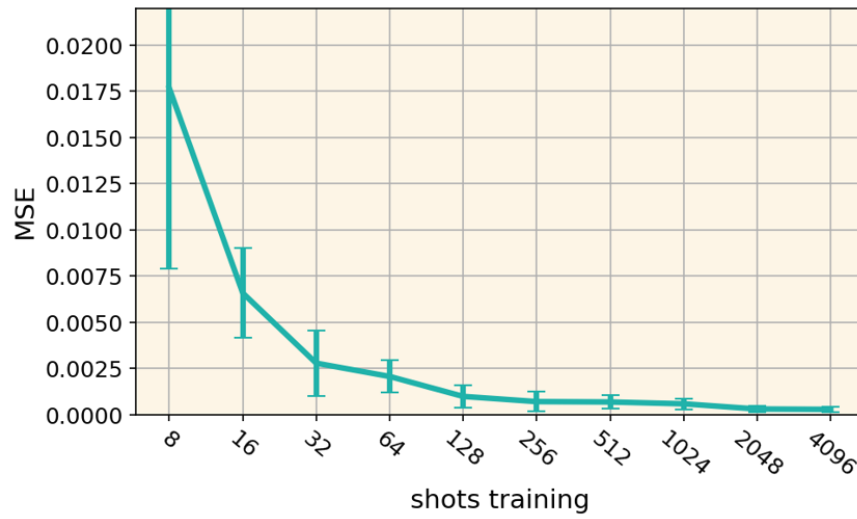


Figure 9.27: Dependency of the MSE as accuracy measure of an increasing number of shots for the QAG training.

als run with the same hyperparameter configuration. With an increasing number of shots, the accuracy strongly increases (lower MSE) up to about 128 shots. At a higher numbers of shots, the MSE saturates and improves only slightly. The error bars are small, resulting to stable training convergence, as already indicated in the training evaluation in figure 9.12.

Furthermore, the influence of the number of shots in inference is tested. For this, the best models trained with 64, 256, 1 024 and 4 096 shots are selected. For these models trained with a fixed number of shots in training, inference is run with different number of shots. The results are shown in figure 9.28. It is visible that the MSE decreases rapidly for larger numbers of shots. Similar to training, the MSE starts to saturate around 128 shots and remains at a constant accuracy level for a larger number of shots. All models trained with a different number of shots demonstrate a similar behavior during inference. This indicates that 128 shots are sufficient to achieve a good level of accuracy in inference. The training and inference tests earlier in this chapter were executed with 512 shots, which is well above the 128 shots threshold.

In classical computing, the impact of reduced precision computing to ML use cases is actively investigated, as demonstrated in chapter 6 by the own investigations within this thesis. There, the accuracy of the three-dimensional images could be maintained while only operating in 256 discrete energy levels. Reducing the number of shots is the quantum analogue for the QAG model. With only 128 shots for training and inference, combined with an only 8 qubit quantum circuit, this is computationally inexpensive compared to other quantum models, which represents an enormous advantage of the

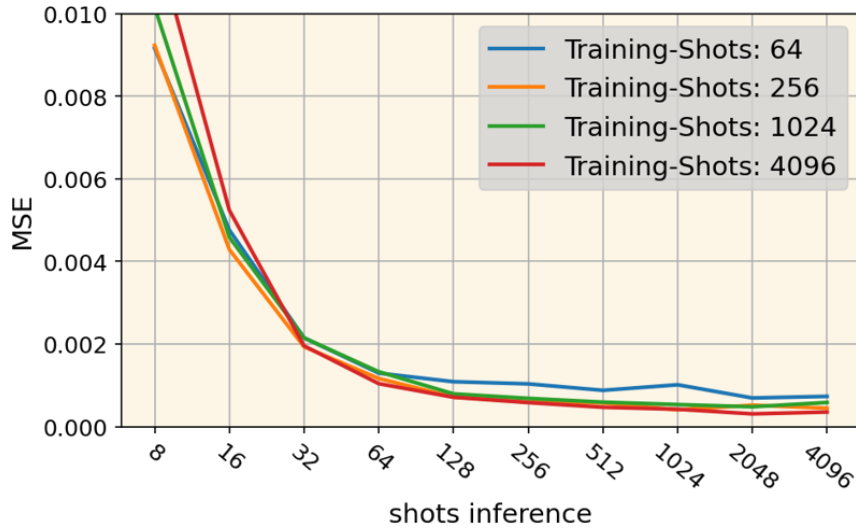


Figure 9.28: Inference accuracy measured from the MSE with an increasing number of shots. The various lines displayed correspond to models trained with varying numbers of shots, as indicated in the legend. For each inference measurement, 980 images are generated and compared to Geant4. No uncertainty bands are given to keep the clarity in the figures.

QAG model. If required, the number of shots can be certainly increased from 128 to more than thousands on current quantum devices and simulators to increase the level of discretization.

9.6 Discussion and Future Work of the Quantum Angle Generator

In this chapter, the newly developed full quantum QAG model was presented and evaluated. Advancements over previous generative models are that it can generate realistic images instead of probability distributions and it allows a lightweight and smooth training. The results clearly demonstrate that the QAG model is capable of generating images of similar precision to the training images on a variety of validation metrics. In the noiseless case, the QAG model perfectly reproduces all accuracy metrics. With a clever intra-qubit entanglement strategy in the MERA-up quantum circuit together with the specific Corr loss term, the pixel-wise correlation was correctly reproduced in the generated image set.

The effect of quantum noise was studied on the training and inference process on

real quantum devices. When including simulated quantum hardware noise in the inference, the results remain stable up to approximately 1.5% of noise. This could be improved to almost 3% by directly training the model with noise. Using realistic simulations with loading the noise model from the physical hardware, the model is theoretically capable to maintain a similar level of accuracy. In practice, performing inference of the noiseless trained model onto a physical quantum device leads to a decrease in accuracy. However, if the model is trained on hardware, it can learn to overcome the underlying noise behavior. To the author's knowledge, this is the first time that an entire training of a generative model on a real quantum device has been successfully performed with precise accuracies.

Regarding quantum noise, it could be investigated whether a model trained with a particular noise configuration is more robust to other noise configurations of a similar magnitude. This means that the average noise level remains approximately constant and only the individual qubit or gate noise levels vary. Because on real quantum devices there are noise fluctuations due to regular calibrations. In this context, it could be investigated in the future what would happen if the noise levels were lower during inference than during training. If lower inference noise levels maintain or even improve the accuracy, then training models with higher noise levels than applied in inference would be a beneficial approach.

To make the QAG model more realistic, conditioning can be implemented in the future to allow the generation of calorimeter shower images of specific primary particle energies, similarly to the classical ML models in chapter 4 and 5. This can be achieved, for example, by implementing an auxiliary loss and adding a specific energy value to the rotational gates in the state preparation function. In calorimetric aspects, increasing the number of pixels would make the models more realistic. However, the additional phase space could also be used to include timing in addition to the energy in the pixels. For the HL-LHC, the use of the timing can be helpful to identify and remove pile-up. The additional opportunity to use timing would be a great advantage in favor of quantum computing.

Since only 256 shots are required for achieving a good level of accuracy and the number of shots can be chosen much larger, the encoding technique could be extended. In this thesis, only the polar angle θ on the Bloch sphere is used, as visualized in figure 7.2. This could be extended by also utilizing the azimuthal angle ϕ . In this way, the number of pixels can be doubled while the number of qubits remains the same. To implement this, the number of shots must be increased, and for half of the shots, the outcomes measured on a different computational basis.

Another straight-forward step of improvement would be to increase the size of the generated images. It would be valuable to investigate whether the hierarchical quantum circuits used provide enough entanglement for larger use cases, e.g., images with

more pixels, to represent the pixel-wise correlations in the data.

As circuits scale up in terms of the number of qubits, another problem arises that would be needed to be investigated. Although the number of qubits on quantum devices is growing rapidly, they are constrained by limited qubit-to-qubit connections. This can restrict the choice of the circuit architectures. Due to complex entanglement strategies, a large number of swapping operations are required to implement the circuits on the quantum devices. These swapping operations consist of noisy entanglement gates that make the circuits even more error-prone to gate level noise.

Doubling the size to sixteen qubits can be still done on quantum simulators. Nevertheless, for even larger models in the future, testing on classical simulators would be impossible due to the exponential scaling of quantum circuit simulations. However, the improving quantum devices and increasing quantum hardware access in the future will reform the way of quantum algorithm deployment, as for example communicated by the IBM 2022 roadmap [294]. Currently, models are first tested on simulators before eventually executing them on physical quantum devices. In the future, the models should be directly implemented and tested on quantum devices. This causes difficulties, especially for QML researches, because model training requires numerous quantum circuit executions. Therefore, models with a lightweight training, such as the QAG model, are likely to dominate in the next couple of years.

Thesis Objective Fulfilled:

A full quantum model which generates accurate individual images with distinct pixel intensities was successfully developed. Moreover, it was demonstrated that it can be executed on current imperfect and noisy quantum devices with achieving satisfying accuracy levels.

Chapter 10

Full Quantum Generative Adversarial Networks

In this chapter, the QAG approach from the previous chapter 9 is converted into a full quantum generative adversarial network model. The model utilizes the same encoding technique and generator, however, it is trained by a quantum discriminator circuit instead of loss functions. Both the generator and the discriminator are quantum circuits. This sets the full qGAN model from this chapter apart from the hybrid qGAN model in chapter 8, which utilizes a classical discriminator neural network. Furthermore, the qGAN here uses angle encoding, in contrast to the previous hybrid model that utilizes amplitude encoding.

In the subsequent section, the objectives for developing a full qGAN model are motivated. This is followed by an introduction to the quantum circuits and the adversarial training. Next, the model results are evaluated. To compare the expressive power of quantum circuits to classical neural networks, the full qGAN is compared to a hybrid model. The results of this chapter were first published in the paper from Ref. [295].

10.1 Motivation for Adversarial Training

Training generative models with the help of loss functions, as for example implemented in the QAG model in the previous chapter 9, represents an already known concept for classical ML applications [296]. For simple use cases, training models with loss functions is more efficient and reasonable accuracies are achieved. However, for more complex applications, limitations are reached. For example, the QAG model

exclusively trained by the MMD loss was merely able to represent the moments – in particular the average shower distribution – present in the training data. But it failed to learn more complex image correlations, prior including an auxiliary loss. However, for learning more complex correlation patterns of the training data, it is challenging to determine appropriate loss functions.

For the generation of complex image patterns, in classical ML the adversarial training approach of two neural networks – the GAN approach – was developed. GANs currently outperform other generative model approaches, as provided in section 3.2. Compared to training with loss functions, the adversarial approach results typically in better generalization and less overfitting [297] and to better representations and more accurate predictions of the target data [298]. In GANs, the discriminator neural network is more adaptable to the training data and takes over the purpose of the loss function. The discriminator network is able to efficiently learn the complex correlations present in the data and employs this knowledge for the generator training.

At the current stage of quantum computing, due to the limited number of qubits, only small and simplified models are simulated. For this, training them with ordinary loss functions achieves satisfying accuracies. In the future, increasingly complex problems can be simulated on more powerful quantum devices with more qubits. Similar as for classical models, for more complex use cases, at some point the adversarial training will probably outperform training with loss functions. Therefore, a full qGAN model is developed and tested. While hybrid quantum-classical GAN models have been well studied, successfully operating full qGAN models are still rare in the literature due to the complexity of the implementation and difficulties in training towards convergence. For example, Ref. [299] proposed a full qGAN model that can generate quantum states with a high fidelity. Another work, Ref. [300] introduced a full qGAN that can be trained with a limited number of samples. This study represents the first successfully implemented full qGAN model for calorimeter simulations.

10.2 The Generator and Discriminator Quantum Circuits

The full qGAN uses a quantum generator circuit and a quantum discriminator circuit. The QAG principle from chapter 9.2 and the MERA-up circuit architecture are employed for the generator. Similar to the QAG model, eight pixel images and angle encoding are utilized.

The discriminator uses the MERA-downsampling (MERA-down) architecture, as illustrated in figure 10.1. Opposite to the generator, the information in the discriminator becomes compressed or downsampled from multiple qubits to one qubit and eventually measured into classical data. The measurement output of the discriminator qubit

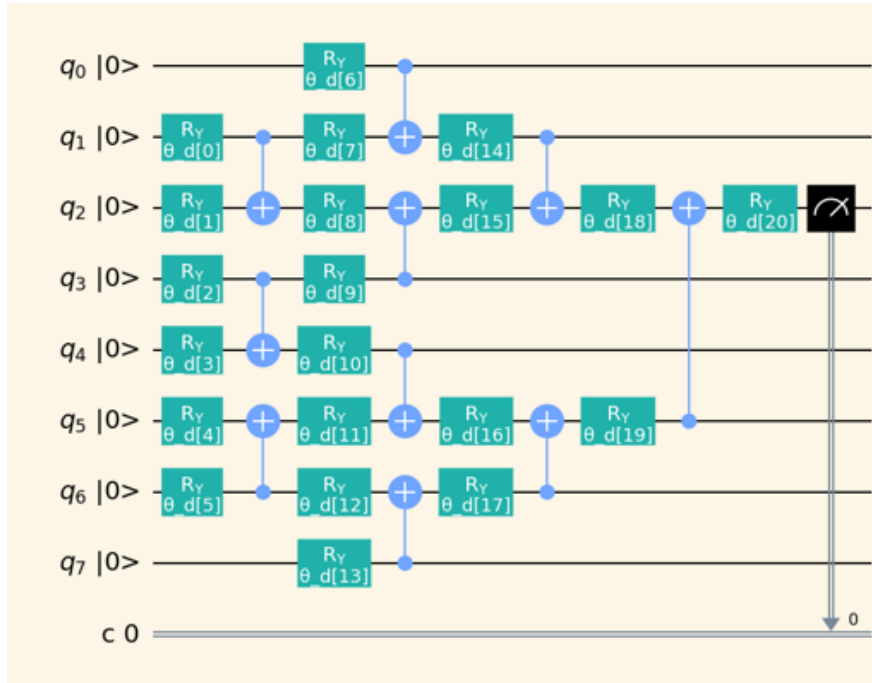


Figure 10.1: The MERA-downsampling quantum circuit architecture employed as discriminator.

corresponds to the True/Fake probability. If $|0\rangle$ is measured, the discriminated image corresponds to a fake image, and if $|1\rangle$ is measured, it corresponds to a true image. The True/Fake probability is determined by measuring the discriminator output for multiple shots and averaging the result.

In the implementation of the full qGAN model, the two circuits – generator and discriminator – are directly connected to each other. Both circuits operate on the same qubits and can be combined to one large circuit. Therefore, no intermediate measurement is required between the generator and discriminator, and the generated quantum state is the direct input into the discriminator. This represents an enormous advantage of the full qGAN model compared to hybrid approaches, since intermediate measurements can distort the result and thus the generated image. This can happen, for example, due to readout errors or due to the fact that the measurement operation is only an approximation of the probability vector of a quantum state instead of providing a precise result, as explained in chapter 7. On the other hand, as a consequence of this direct generator-discriminator connection, the quantum images generated during training are not visible, since any intermediate measurement would destroy the quantum information. Therefore, an adopted quantum circuit is required for inference. For the implementation of the training and inference of the full qGAN model, three distinct circuits are utilized which are introduced in the following.

1. Training with True Images

A first circuit is required for training the discriminator on the true images from the training set. Unlike the hybrid qGAN case, here, the training images have to be encoded from classical into quantum images. For this, angle encoding with the rotation angles ϵ is employed. In the QAG model, only the decoding from the generated quantum image back into a classical image is utilized. Opposite to the decoding, in encoding, the angle ϵ can continuously rotate around the Bloch sphere, implying continuous energy values.

The implementation of the encoding on quantum devices is: first, a H -gate is applied to initiate the $|+\rangle$ state, which corresponds to the equal superposition between the $|0\rangle$ and $|1\rangle$ state. Next, one RY -gate is placed. The rotational angle ϵ of the RY -gate correlates to the encoded pixel energy value. The transformation of the pixel energy E into the angle ϵ is performed with:

$$\epsilon = \frac{2 \cdot \epsilon_{\max}}{E_{\max}} \cdot E - \epsilon_{\max} . \quad (10.1)$$

The factor $2 \cdot \epsilon_{\max}$ in the nominator represents the entire angle range $[\epsilon_{\min}, \epsilon_{\max}]$. According to the definition, $\epsilon_{\max} = -\epsilon_{\min}$. Therefore,

$$\epsilon_{\max} - \epsilon_{\min} = \epsilon_{\max} - (-\epsilon_{\max}) = 2 \cdot \epsilon_{\max} . \quad (10.2)$$

The entire pre-factor $\frac{2 \cdot \epsilon_{\max}}{E_{\max}}$ in front of the energy E corresponds to the linear slope of the transformation function. The factor $\epsilon_{\max}$ is subtracted to shift the angle range from $[0, 2 \cdot \epsilon_{\max}]$ to $[-\epsilon_{\max}, \epsilon_{\max}] = [\epsilon_{\min}, \epsilon_{\max}]$, as required by the definition of the angle ϵ on the Bloch sphere in figure 9.2.

The complete quantum circuit for training the discriminator on the true images is visualized in figure 10.2. The quantum gates left of the gray vertical line in the middle of the image are responsible for encoding the training image into quantum states. Right of the gray vertical line, the MERA-down architecture is placed. Lastly, only qubit two is measured into a classical register consisting of one number providing the True/Fake probability. When $|0\rangle$ is measured, the image corresponds to a fake image, and when $|1\rangle$ is measured, it corresponds to a true image. The True/Fake probability is obtained by measuring the discriminator output for multiple shots and averaging the result. For calculating the GAN loss value, the classical GAN binary cross entropy loss from equation 4.1 is applied. The true images from the training data are labeled with the value 1.

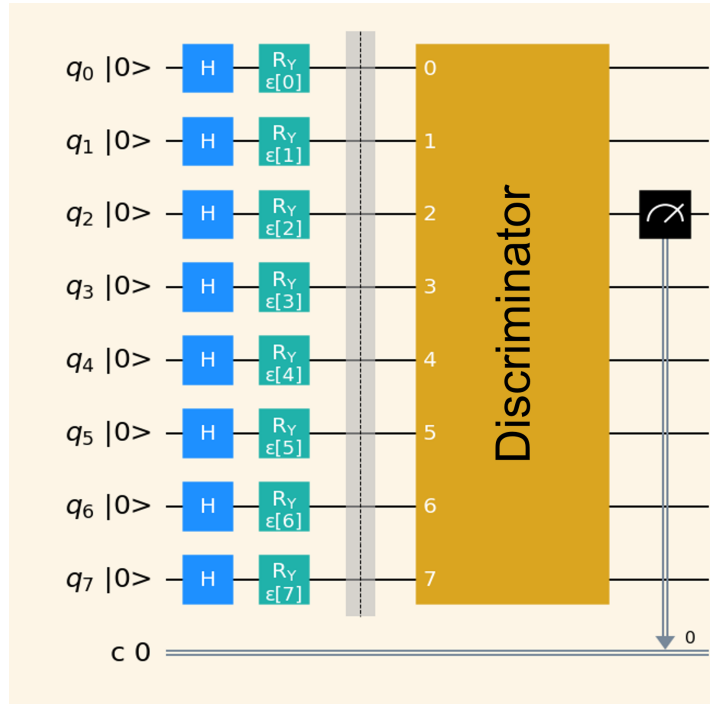


Figure 10.2: The quantum circuit employed for training the discriminator with the true training images.

2. Training with Fake Images

The second circuit is required for training the discriminator with the fake images provided by the generator and for the generator training. The total circuit is a composition of the generator circuit and the discriminator circuit, as visualized in figure 10.3. Similar as for training the discriminator with the fake images, only the output of one qubit is measured to determine the True/Fake probability.

To train the discriminator, the fake images are labeled with value **0**. Similar to the QAG model, **H**- and **RY**-gates are employed for the state preparation to implement superposition and random noise to the qubits. During the discriminator fake training, the generator weights are frozen and not trainable.

For training the generator, the same circuit as for the discriminator fake training from figure 10.3 is operated. In this case, the discriminator parameters are frozen and the generator parameters trainable. To fool the discriminator during the generator training, the labels are switched, and the fake images are labeled with **1**. Similarly, the True/Fake loss is determined for updating the generator parameters.

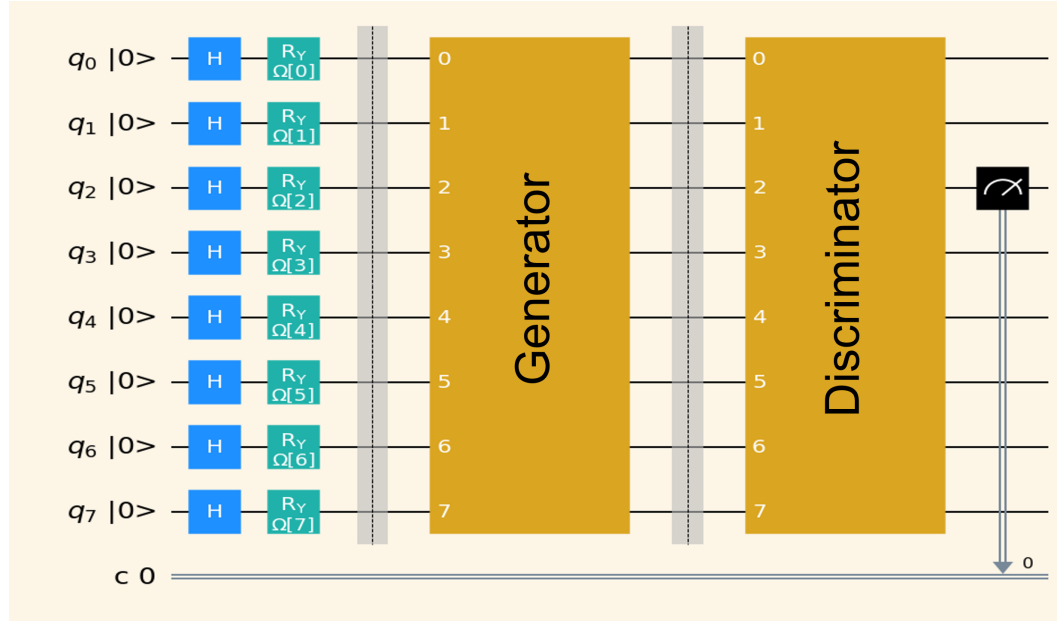


Figure 10.3: The quantum circuit employed for training the discriminator with the fake images of the generator circuit. Furthermore, the same circuit is utilized for the generator training. In the discriminator training, the generator weights are frozen. Opposite, in the generator training, the discriminator weights are frozen.

3. The Inference Process

The last circuit is required for generating shower images and measure them into classical data to make them visible. For this, the quantum states of all qubits are measured to determine the pixel energies according to the procedure developed for the QAG model. Therefore, the classical register consists of eight values, the same number as qubits present in the quantum circuit. The decoding of the measurements into energy values follows the similar principle as introduced for the QAG model in section 9.2.

10.3 The Adversarial Training Optimization

This subsection contains the configuration and hyperparameters chosen for training the full qGAN model. Similar as for the classical ML GAN training in chapter 5, separate generator and discriminator learning rates are employed. For achieving the best results, the generator learning rate $lr_{\text{gen}} = 0.02$ and the discriminator learning rate $lr_{\text{disc}} = 0.04$ are chosen. In order to obtain stable losses towards the end of the training, the generator and discriminator are operated with two distinct exponential learn-

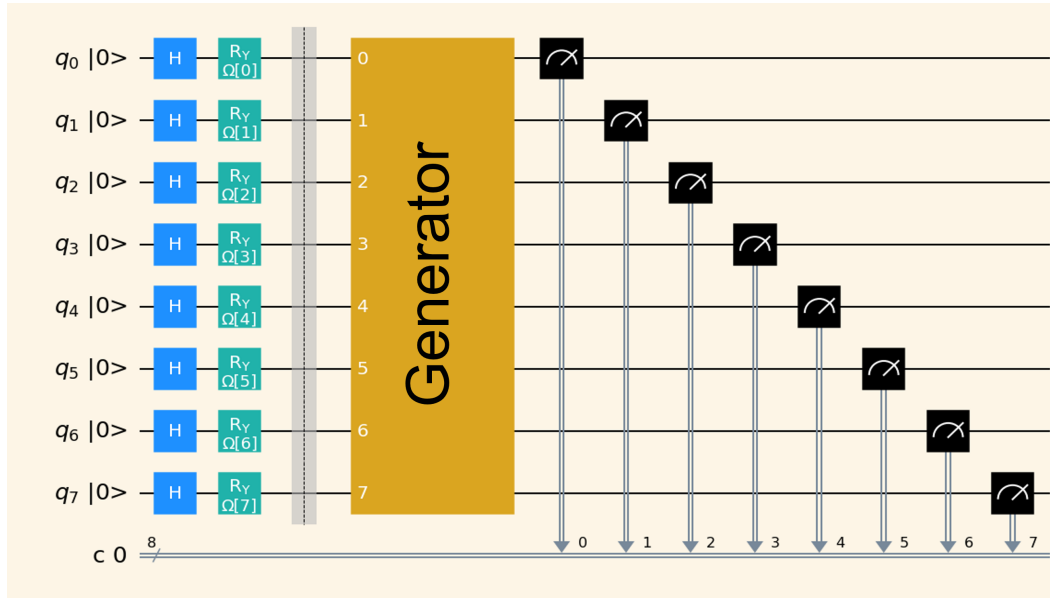


Figure 10.4: The quantum circuit for generating events in inference. The quantum states of all qubits must be measured and decoded to classical data to determine the pixel energies.

ing rate decays. The learning rate decay for the generator is $lr_{\text{dec}_{\text{gen}}} = 0.006$ and for the discriminator $lr_{\text{dec}_{\text{disc}}} = 0.007$. The learning rate decay starts at epoch ten and is visualized in figure 10.5, where both learning rates converge to low values. The model is trained for 1 000 epochs, where one epoch contains eight optimization steps (batch size of eight). In empirical tests, it is observed that the training converges much better if the discriminator is trained more frequent than the generator, similar as for the hybrid qGAN in chapter 8. In the full qGAN model, at each optimization step the discriminator is trained five times with the true and fake images, followed by one generator training step. In training and inference, all introduced circuits use 1 024 shots. The hyperparameters are found by hyperparameter searches using the Optuna [163] library and empirical testing.

As in the classical GAN case, in training, label smoothing is applied. We apply two-sided label smoothing, where the true labels 1 are smoothed to 0.9 and the fake labels 0 to 0.1. This improves training convergence and the achieved accuracy. All trainable rotational parameters are initialized with zeros to provide a stable training start point and to speed-up training compared to random initialization. The SPSA optimizer [288] is used for calculating the generator and discriminator parameter update during training.

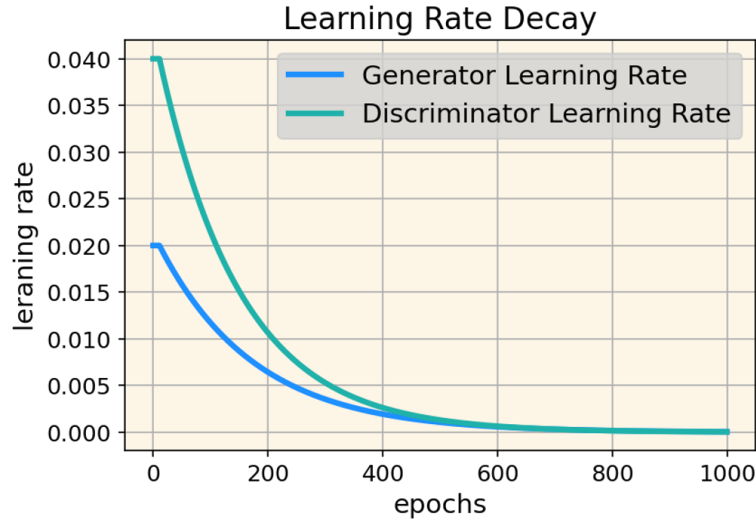


Figure 10.5: The exponential decay of the generator and discriminator learning rate during training.

10.4 Training Evaluation

This section provides the evaluation of the training process. In figure 10.6, the training statistics of 20 trials is visualized, which means that the training is repeated multiple times with the same hyperparameters. The MSE as accuracy metric is pixel-wise calculated between the average training and generated images. Due to the hyperparameter optimization, the mean (blue solid line) converges quickly and remains at a stable level. The Standard deviation (light blue band) is narrow, and the best trial (green solid line) has only minor oscillations. All these factors indicate a smooth, stable and successful training.

Next, the characteristic training losses of the best trial are shown in figure 10.7 (left). Plotted are: the generator loss (red), the discriminator true loss (blue), the discriminator fake loss (orange) and the composite discriminator losses (green), which corresponds to the average of the discriminator true and fake loss. Due to the zero initialization of the circuit parameters, the composite discriminator loss is lower than the generator loss in the first epochs. In the following epochs, the losses converge to increasingly similar values, resulting in a good training convergence. Moreover, the losses remain stable towards the end of the training due to the learning rate decay. Between the single epochs, the losses oscillate a lot, probably because of the small number of batches. A larger number of events per epoch would potentially smooth the losses.

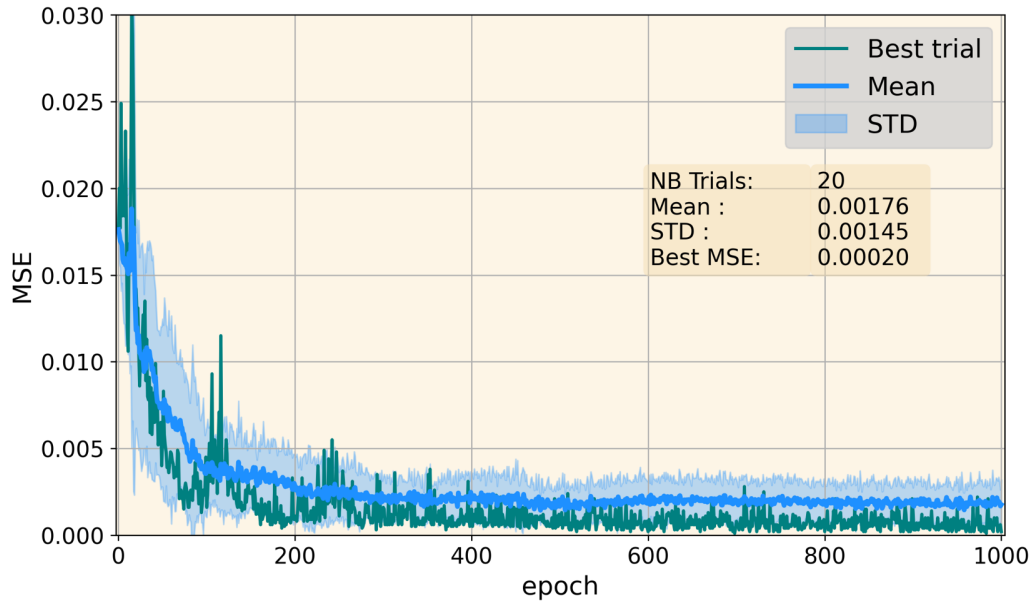


Figure 10.6: The training statistics of repeating the training for 20 trials. The MSE is plotted as accuracy metrics as a function of the training epochs.

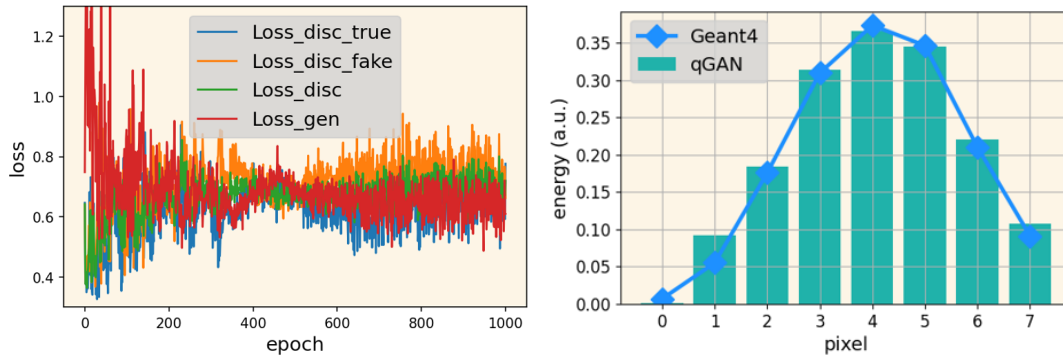


Figure 10.7: (left) The characteristic training losses of the best trial. (right) The generated average shower shape.

10.5 Inference Evaluation

To evaluate the inference accuracy, 980 images are generated and compared to the full test set of Geant4. As a first accuracy metric, the average shower shape of the best trial is provided in figure 10.7 (right). It can be observed that the full qGAN model can precisely reproduce the average shower image of Geant4.

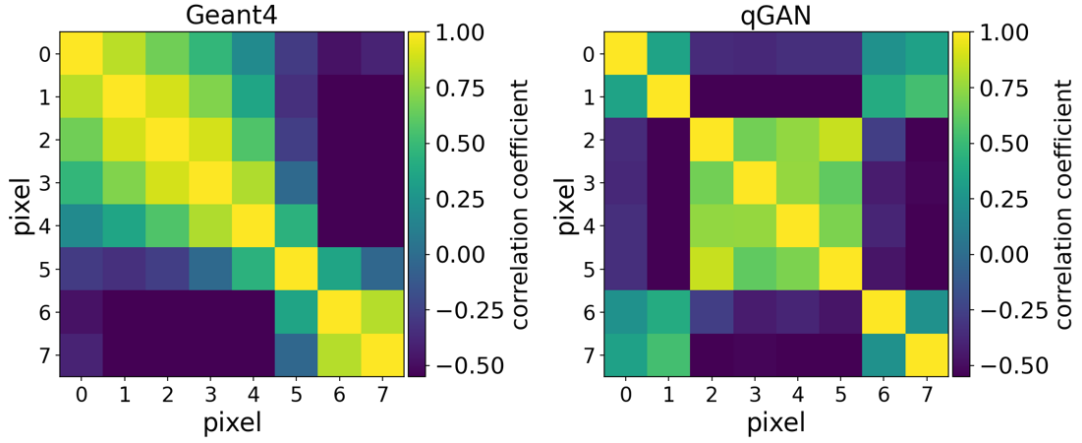


Figure 10.8: The pixel-wise correlation plots. Left the correlation of the Geant4 training data is provided, and right of the full qGAN model.

However, the pixel-wise correlation plot of the generated images, visualized in figure 10.8, does not agree well with Geant4. Furthermore, also the other more sophisticated accuracy metrics, as shown for the QAG model evaluation in section 9.3.2, are also not sufficiently accurate (see appendix B.9).

Therefore, the full qGAN model is only able to reproduce the average shower shape metric. Potential reasons are discussed in the following. The first reason could be the discriminator strength. A deeper or more complex discriminator circuit with more entangling operations would be beneficial. For the generator circuit, it has already been practically demonstrated for the QAG model in chapter 9 that the generator circuit is capable of reproducing the complex physics accuracy metrics. The discriminator MERA-down circuit, however, only achieved a relatively low entangling capability of $E \approx 0.5$ in the circuit study in section 9.2.3, as indicated in figure 9.8. The entanglement enables the circuit to learn correlations between qubits. The fact that the MERA-down architecture maintains a low entanglement capability might be the reason why the pixel-wise correlations cannot be learned correctly, what was further evidenced by the manipulated gate metrics in figure 9.11.

Another possible reason could lie in the training process. The training hyperparameters were optimized to give good results for the MSE as accuracy metric. However, the MSE only considers the average shower shape and not more complex metrics. It could also be that the more complex physics patterns are not present as pronounced as expected. Therefore, they are not easily learnable by the model. Lastly, auxiliary losses, such as the Corr loss for the QAG model, could potentially improve the accuracy.

The training of the full qGAN model remains hardware costly and time consuming

due to the two quantum models trained alternately. Therefore, neither training on quantum hardware nor noise studies on simulators will be performed.

10.6 Hybrid Model Comparison

A last test considers a hybrid qGAN, combining the QAG generator to a classical discriminator. The goal is to compare the full and the hybrid qGAN model to observe possible advantages of the quantum behavior over the classical behavior, or vice versa. For example, quantum circuits are assumed to provide a higher expressive power than classical neural networks [301], meaning that the same accuracy can be achieved with fewer parameters. On the other side, it is investigated if the classical model can perform better on the pixel correlation accuracy metric.

The classical discriminator is constructed using Dense or fully connected layers as parameterized layers. Three discriminator sizes are tested: small (S), medium (M) and large (L), determined by the number of layers and the number of trainable parameters. The three architectures are visualized in figure 10.9. The input layer has size eight and contains the pixels of the input image. The hidden layers are followed by LeakyReLU activation functions. The output represents one neuron, which outputs the True/Fake probability. The configuration of the three discriminator neural networks are arbitrary chosen. The number of trainable parameters are:

- S: 153 trainable parameters
- M: 433 trainable parameters
- L: 1 889 trainable parameters

The training hyperparameters of the hybrid qGAN model are chosen to be similar for all models and were found by a hyperparameter search with Optuna and are: the generator learning rate of 0.01, the discriminator learning rate of 0.006 and a common learning rate decay of 0.006. The training comprises 1 000 epochs, each containing one optimization step. The generator and discriminator are equally optimized once per training step. The discriminator neural network parameters are randomly initialized using a $[0, 1]$ uniform distribution.

Training Evaluation

In the following, the training of the hybrid qGAN models is evaluated. For each of the three models, ten training trials are executed with the same hyperparameters. The results are visualized in figure 10.10. The mean MSE converges to stable results. Towards the end of the training, the hybrid M model (orange) converges to the lowest

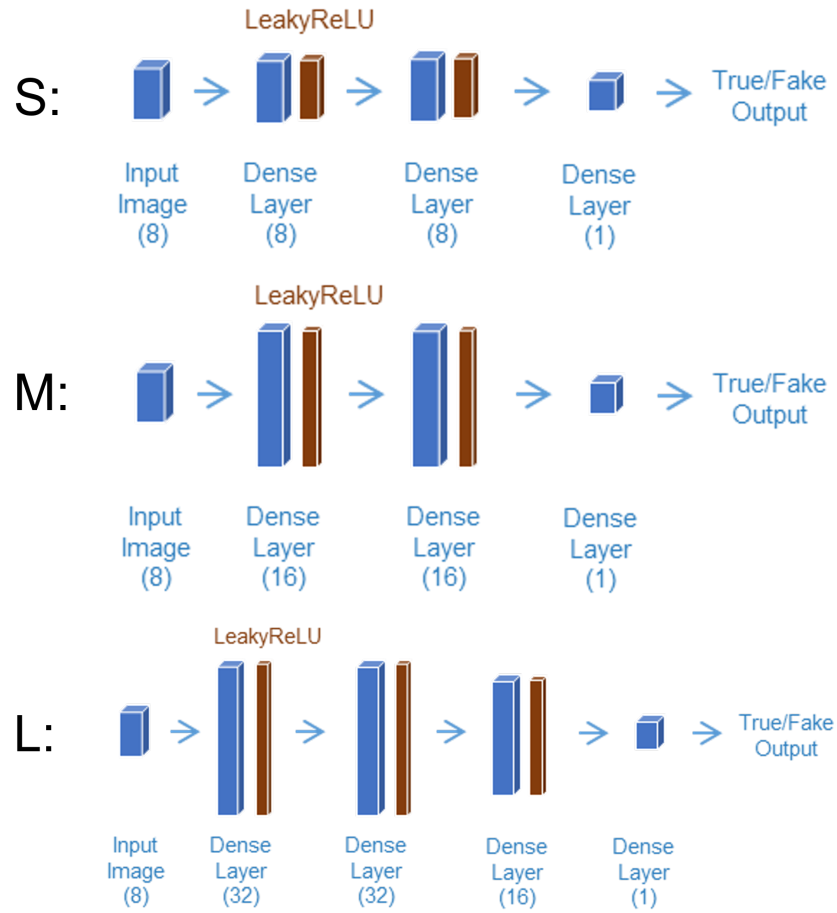


Figure 10.9: The three hybrid qGAN model discriminator architectures: small (S), medium (M) and large (L).

MSE, corresponding to the best accuracy. The L (green) and S (blue) models follow. The reason why the S and L models perform worse is due to the GAN principle. In the adversarial training, the best accuracy is achieved when the generator and discriminator model are balanced. When one of the models outperforms the other one, the accuracy becomes worse. Therefore, the results of the hybrid models can be interpreted as the following: for the S model, the discriminator is weaker than the generator, therefore, worse results are achieved. For the L model, the discriminator is too powerful and outperforming the generator. The M model exhibits a representational power that best meets the one of the generator. Due to the imbalance in the S and L models, the MSE even increases for a few epochs at the start of the training until it starts decreasing. Further, it can be noted that the L model maintains the largest standard deviation band, indicating a training instability.

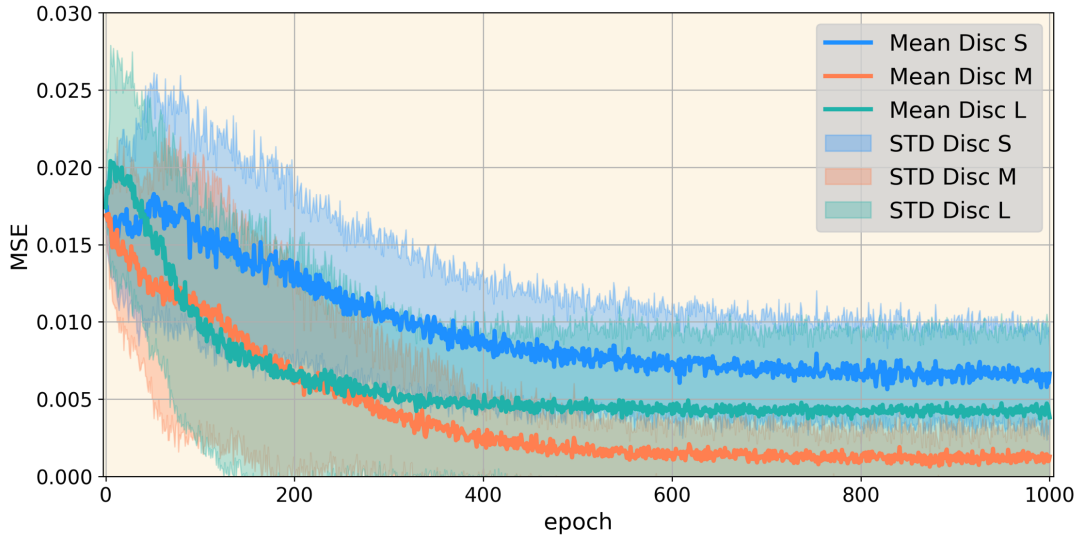


Figure 10.10: The training convergence statistics plot of the three hybrid qGAN models. The MSE is plotted as accuracy metric for the entire training.

More fine-tuning in the discriminator architectures and sizes could potentially lead to better results. Discriminator architectures utilizing 1D convolutional layers also did not improve the accuracy, probably because they were too powerful for the quantum generator circuit.

Inference Evaluation

In figure 10.11 (left) the average shower shape for the best hybrid M model is plotted. It agrees well with the Geant4 prediction. The pixel-wise correlation plot of the same model is illustrated in figure 10.11 (right), which is compared to the correlation pattern of the Geant4 data in figure 9.5. The model fails to correctly reproduce the correlation pattern. Similar as for the full qGAN model, also the other trials – beside the best trial – (not shown) do not achieve better pixel-wise correlations.

The other hybrid models also cannot correctly reproduce the pixel-wise correlation plot. The validation plots for the S and L models are provided and discussed in the appendix B.10. Similar as for the full qGAN model, more complex physics accuracy metrics cannot be correctly represented by the hybrid models.

The training and inference accuracies of the various quantum models from this thesis which use angle encoding are listed in table 10.1. The training accuracy corresponds to the statistical mean MSE values at the last training epoch for multiple trials. The inference accuracy is the calculated MSE of 980 generated images of the model of the best trial in each case. Due to the generation and evaluation of multiple images, the inference accuracy may deviate from the training accuracy. This is also a test of

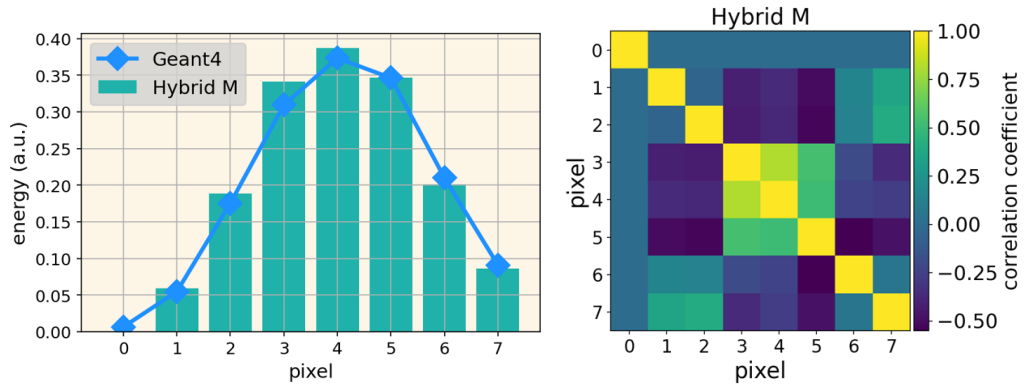


Figure 10.11: (left) The average shower shape plot and (right) the pixel-wise correlation generated with the best trial of the hybrid M qGAN model.

overfitting.

In table 10.1, it can be observed that the newly developed QAG model achieves by far the lowest MSE, resulting in the best accuracy. It is followed by the hybrid M model, which slightly outperforms the full qGAN model. The hybrid L and S models perform worst, with significantly larger MSE values. Only the QAG model is able to correctly reproduce the more complex physics accuracy metrics, as for example the pixel-wise image correlation.

Table 10.1: The various quantum models tested in this thesis including their accuracy in training and inference. The pixel-wise computed MSE is employed as accuracy metrics. The MSE values are multiplied by 10^3 . The lower the MSE, the better the accuracy.

Model	MSE Training	MSE Inference
QAG	0.67 ± 0.56	0.58 ± 0.37
Full qGAN	1.76 ± 1.45	1.77 ± 1.46
Hybrid qGAN S	6.64 ± 3.28	6.65 ± 3.11
Hybrid qGAN M	1.28 ± 2.22	1.50 ± 2.15
Hybrid qGAN L	3.83 ± 4.42	3.80 ± 4.32

The comparison between the hybrid and full qGAN models leads to one interesting outcome. Since the accuracy of the full qGAN model matches best the one of the hybrid M model, the representational power of the two discriminator models, the full quantum and the hybrid, can be considered approximately similar within the given statistics. This means that the MERA-down discriminator quantum circuit architecture with only 20 trainable parameters performs equally well to a classical neural network with more than 400 trainable parameters. This rises the question if variational

quantum circuits requiring far less trainable parameters are able to reach the same representational power than neural networks? Ultimately, the operation of fewer numbers of parameters can lead to faster training convergence. Due to many uncertainties, the aforementioned comparison should be interpreted as a hint instead of a proof. Furthermore, in this study only highly simplified toy models are compared. Thus, a general statement for larger circuits with more qubits cannot be made. On the other side, similar outcomes are observed in related studies. For example, in Ref. [301] and Ref. [302], it is claimed that quantum circuits have a higher storage and modelling capacity as well as a better computational efficiency than neural networks because of the employed quantum principles.

10.7 Discussion of the Full Quantum Generative Adversarial Training

In this chapter, a full qGAN model was developed and trained as a proof of concept. For this purpose, the required quantum circuits and the training process were introduced. In the primary physics evaluation metric, the MSE, the full qGAN model performed well, but could not reach the accuracy of the QAG model from the previous chapter. In more complex physics accuracy metrics, the model performed less well.

A hybrid qGAN was trained with three discriminator sizes. Only the hybrid model with the medium discriminator outperformed the full qGAN in terms of MSE slightly. However, the accuracy of the QAG model could not be reached. Training the model with a small and a large discriminator led to worse results due to an imbalance of the power between the generator and the discriminator models. Similar to the full qGAN model, all hybrid qGAN models failed to correctly reproduce the more complex physics accuracy metrics.

Why the more complex physics metrics could not be reproduced by the qGAN models is the subject of future research. Possible explanations are that the discriminators are not powerful enough to learn the complex patterns present in the training data, or that the training hyperparameters were chosen primarily to optimize for the MSE metric. Probably, the utilization of an auxiliary loss would improve on this, similarly as for the QAG model. Furthermore, the Wasserstein GAN [131] approach can be tested for the full qGAN model.

Thesis Objective Fulfilled:

A full quantum Generative Adversarial Network model was developed and successfully trained to convergence for calorimeter simulations for the first time. The full quantum model with only 20 trainable parameters accomplished a statistically similar good accuracy to the hybrid M model with 433 parameters. This result encourages a potential advantage in representational power of quantum circuits over classical neural network architectures. The further metrics beside the MSE could not be satisfactorily fulfilled, neither by the full qGAN nor by the hybrid versions. These deficits currently set limits to the developments and must be addressed in future studies.

Chapter 11

Summary, Future Work and Conclusion

The research in this thesis is aimed to develop and investigate several generative models to potentially replace the current resource demanding calorimeter detector simulations performed with conventional computing. In the following, the major results of this thesis are summarized and compared to the initial thesis objectives. Afterwards, possible approaches for future improvements and researches are presented, before conclusions are drawn. Finally, related publications and workshop presentations are listed.

11.1 Summary

Since the two main topics deep learning and quantum computing are two different computing approaches, separate summaries are provided for each topic.

Deep Learning Generative Models

Multiple generative models have been presented and evaluated. The Generative Adversarial Network approach was selected as the most appropriate model for the calorimeter use case. Since 3D convolutional layers represent the state-of-the-art for 3D image applications, an initial 3D convolutional neural network architecture was developed and operated as a proof of concept. However, 3D convolutional layers remain computing resource demanding, which led to the development of the novel and more hardware-efficient 2D convolutional neural network architectures. The best introduced 2D architecture was evaluated against eight physics accuracy metrics and

it performed better than previous models, achieving accuracies very close to those of Geant4.

In a further optimization step, the classical model was quantized to an int8 reduced precision, resulting in a further 3x speed-up. Without quantization, a 158 000x speed-up was achieved on GPUs compared to the current Geant4 simulation, and with the quantized model a 120 000x speed-up on CPUs. Thus, the thesis objectives to evaluate more efficient alternatives to Geant4 calorimeter simulations have been achieved.

Quantum Computing Generative Models

For future detector simulation requirements, quantum-based machine learning models were developed and evaluated, since the quantum based models have the potential for an exponential advantage over their classical counterparts. However, the executions of realistic simulations on real quantum devices and quantum simulators represent a bottleneck today. Therefore, models for simplified training data are developed as a first step.

A first hybrid quantum-classical Generative Adversarial Network model was developed as a proof of concept. It achieved good results for 1D and 2D calorimeter simulations. Quantum hardware error rates are a limitation of current quantum devices. Therefore, the influence of noise on the model accuracy was explored. Training with low noise levels actually slightly improved the achieved accuracy, indicating robustness of Quantum Machine Learning against modest levels of noise.

The initial hybrid model experiences limitations, such as only producing average energy distributions and resource-intensive training that does not allow training on physical quantum devices. Therefore, an enhanced full quantum model has been developed. The Quantum Angle Generator employs loss functions and is capable of producing extremely precise images, as proven by several detailed physical accuracy metrics. Due to its demonstrated robustness to quantum hardware noise and its lightweight training, it was able to successfully execute an entire training on real quantum hardware while achieving good accuracies. Even noise level changes due to hardware calibrations can be compensated during training. With the Quantum Angle Generator, the thesis objective of a full quantum model that produces precise results on current quantum devices has been accomplished.

In a complementary research, the Quantum Angle Generator was combined with a quantum discriminator circuit and trained in an adversarial approach. This model represents one of the first successfully trained full quantum Generative Adversarial Network models. Compared to various classical discriminator neural network sizes, it was demonstrated that the quantum circuit with only 20 trainable parameters exhibits a similar representational power as a classical neural network with more than 400 parameters. This suggests that quantum computing could outperform classical

computing approaches in the future with improved quantum hardware and more sophisticated quantum circuits to mirror the correlations in the data.

11.2 Future Work

In both classical and quantum computing, there are many open questions to be answered and many goals to be achieved in the coming years. A brief overview related to detector simulations is provided below.

Classical Computing

In the area of classical computing, alternative detector simulations must be ready for the High Luminosity phase of the Large Hadron Collider by the end of the 2020s. For enhancing the most computationally intensive detector part, the calorimeter simulations, some possible research priorities are provided in the following. The dimensionality of the model needs to be extended to handle the full image size of $51 \times 51 \times 25$ pixels. In the end, the high granularity calorimeter will provide the energy signals and the timing of three million pixels in each direction. In addition, primary particles with various incident angles, different particle types and multiple simultaneous collisions (pile-up) should be examined.

Once a Deep Learning model has been successfully developed, its computing efficiency can be improved for the large-scale use case at the High-Luminosity Large Hadron Collider. This can be accomplished by quantizing the weights of the neural networks into a reduced precision with employing accuracy driven quantization tools that almost maintain a similar level of accuracy.

Quantum Computing

Quantum computing is a promising candidate for overcoming the computing constraints of High Energy Physics in the long term. Quantum computing is a rapidly developing field: the quantum computers steadily increase in number of qubits, become more reliable in terms of hardware noise, and also the utilized computing algorithms improve. To achieve quantum advantage for real applications, hardware and software algorithms must be further evolved.

In particular, the generative models introduced here could be further improved. For example, the models should learn to generate conditional images, in which, a given primary particle energy leads to a corresponding particle shower. In this way, the energy range of the primary particles can be extended to more realistic values. In the near term, when quantum hardware access is still limited, the developed Quantum Angle Generator represents a promising model. It generates precise images despite the noisy hardware behavior and requires only a limited number of training steps.

Moving further into the future, when more qubits are available and hardware is more accessible, adversarially trained models might potentially achieve better accuracies for larger models and images.

In this thesis, simplified models were implemented with a limited number of qubits. With more qubits on future quantum devices, the scalability of these models needs to be tested, which depends strongly on the circuit architecture and the number of gates operated. Furthermore, since there is currently a lack of suitable quantum hardware devices, research is needed to evaluate if current quantum circuits are able to solve high dimensional tasks with sufficient accuracy. Another open question is which data encoding method will dominate in the future. At present, each encoding technique has its specific advantages and limitations. However, achieving speed-up with quantum computing over conventional computing depends strongly on the utilized data encoding method.

11.3 Conclusion

Both Deep Learning and Quantum Computing are rapidly evolving areas. Sure enough, the approaches accomplished and presented in this thesis will be improved by new innovations developed by the community. As improved quantum devices become available in the future, pure quantum algorithms will likely outperform the classical ones and their importance to High Energy Physics will increase.

By developing and analyzing several alternative Deep Learning and Quantum Computing methods, this thesis supports the development of future faster detector simulation approaches. For the High Luminosity Large Hadron Collider, based on Deep Learning Generative Adversarial Network models, multiple optimizations resulted in more than five orders of magnitude speed-up over the current Monte Carlo approach, while maintaining a similar level of physics accuracy. To achieve this, more complex neural network architectures and reduced precision techniques were developed and applied. To address the increasing hardware requirements for detector simulation on time scales far into the future, the potential of quantum computing was explored. The developed Quantum Angle Generator achieved extremely precise results for simplified calorimeters, despite current quantum hardware limitations.

11.4 List of Publications Related to this Thesis

Several papers have been published in an effort of this thesis. The contents of the following publications comprise a substantial fraction of the results of this thesis.

- **F. Rehm, et al.**
“Validation of Deep Convolutional Generative Adversarial Networks for High Energy Physics Calorimeter Simulations”, *Combining Artificial Intelligence and Machine Learning with Physical Sciences (AAAI-MLPS)*, AAAI-MLPS 2021, Ref. [154]
The author of this thesis was the main contributor to the paper.
- **F. Rehm, et al.**
“Reduced Precision Strategies for Deep Learning: A High Energy Physics Generative Adversarial Network Use Case”, *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, SCITEPRESS - Science and Technology Publications 2021, Ref. [176].
The author of this thesis was the main contributor to the paper by providing the model and its evaluation. The co-authors from Intel significantly contributed by providing the quantization tool and the required software framework.
- **F. Rehm, et al.**
“Physics Validation of Novel Convolutional 2D Architectures for Speeding Up High Energy Physics Simulations”, *25th International Conference on Computing in High-Energy and Nuclear Physics (vCHEP)*, EPJ Web Conf. 2021, Ref. [152].
The author of this thesis was the main contributor to the paper.
- **F. Rehm, et al.**
“Benchmark of Generative Adversarial Networks for Fast HEP Calorimeter Simulations”, *The 9th International Conference “Distributed Computing and Grid Technologies in Science and Education” (GRID)*, Meshcheryakov Laboratory of Information Technologies (MLIT) 2021, Ref. [153].
The author of this thesis was the main contributor to the paper.
- **F. Rehm, et al.**
“Reduced Precision Research of a GAN Image Generation Use-Case”, *Lecture Notes in Computer Science (LNCS)*, Springer 2023, Ref. [177].
The author of this thesis was the main contributor to the paper by providing the model and its evaluation. The co-authors from Intel contributed by providing the quantization tool and the required software framework.

- **F. Rehm, et al.**
“Quantum Machine Learning for HEP Detector Simulations”, *The 9th International Conference “Distributed Computing and Grid Technologies in Science and Education” (GRID)*,
 Meshcheryakov Laboratory of Information Technologies (MLIT) 2021, Ref. [255].
 The author of this thesis was the main contributor to the paper.
- **K. Borrás, et al.**
“Impact of quantum noise on the training of quantum Generative Adversarial Networks”, *20th International Workshop on Advanced Computing and Analysis Techniques in Physics Research 2021 (ACAT)*,
 IOP Science, Ref. [244].
 A collaborative work, where the workload was shared between multiple people.
- **F. Rehm, et al.**
“Quantum Angle Generator for Image Generation”, *The First ACM/IEEE International Workshop on Quantum Computing*,
 IEEE Computer Society, Ref. [268].
 The author of this thesis was the main contributor to the paper.
- **F. Rehm, et al.**
“Full Quantum GAN Model for High Energy Physics Simulations”, *21th International Workshop on Advanced Computing and Analysis Techniques in Physics Research 2022 (ACAT)*,
 ACAT Conference 2022, Ref. [295].
 The author of this thesis was the main contributor to the paper. Review process ongoing.
- **F. Rehm, et al.**
“Precise Image Generation on Current Noisy Quantum Devices”, *Quantum Science and Technology*,
 IOP Science 2023, Ref. [269].
 The author of this thesis was the main contributor to the paper. Paper accepted, but submission process ongoing.

11.5 List of Workshop Presentations

To increase the scientific impact of the research results gained in this thesis, the results were presented at multiple workshops, primarily in the High Energy Physics community.

- *NFS 2020 Conference for Accelerated Artificial Intelligence for Big-Data Experiments Conference*,
Poster presentation, 19.10.2020, **Reduced Precision Strategies in Deep Learning: A 3D GAN use case**
- *IML CERN 2020: 4th Inter-experiment Machine Learning Workshop*,
20 min presentation, 21.10.2020, **Reduced Precision Strategies for Deep Learning: 3DGAN Use Case**
- *Physics Fridays Seminar at the RWTH Aachen University*,
45 min presentation, 22.01.2021, **Deep Generative Adversarial Networks for Faster High Energy Physics Calorimeter Simulations**
- *DPG-Frühjahrstagung 2021: German Physics Society*,
20 min presentation, 18.03.2021, **Deep Learning for Accelerating High Energy Physics Simulations**
- *DESY-CMS group meeting 2021*,
20 min presentation, 23.09.2021, **Quantum Machine Learning for HEP Detectors Simulations**
- *Terascale 2021: 14th Annual Meeting of the Helmholtz Alliance "Physics at the Terascale"*,
20 min presentation, 23.11.2021, **Quantum Machine Learning for HEP Detectors Simulations**
- *Snowmass 2021: 2021 Snowmass workshop on quantum computing for high-energy physics*,
20 min presentation, 02.12.2021, **Quantum Machine Learning for HEP Detectors Simulations**
- *CERN openlab Technical Workshop 2022*,
10 min presentation, 23.03.2022, **Quantum GAN**
- *CDCS Opening Symposium 2022: Center for Data and Computing in Natural Sciences (CDCS)*,
Poster presentation, 27.04.2022, **Impact of Quantum Noise on QGAN Training**

- *FSP D-CMS Workshop 2022: Forschungsschwerpunkt der Deutschen CMS Gruppe*, 15 min presentation, 29.09.2022, **Quantum Angle Generator for Image Generation**
- *Cyprus Quantum Computing Workshop 2022*, 45 min presentation, 20.10.2022, **Quantum Image Generation**
- *CERN openlab Technical Workshop 2023*, Poster presentation, 17.03.2023, **Quantum Angle Generator for Image Generation**
- *DPG-Frühjahrstagung 2023: German Physics Society*, 15 min presentation, 23.03.2023, **Quantum Angle Generator for Image Generation**

Appendix A

Appendix Deep Learning

A.1 Hyperparameters for training the GAN Models

The hyperparameters for training the various GAN models of section 5.4 are listed in table A.1. The hyperparameters of the Conv2D G1, G2 and G3 models and the Conv3D model are provided. The 3DGAN model was trained by an external source and not in this thesis. Therefore, the exact training hyperparameters are unknown.

Table A.1: The hyperparameters for training the different GAN models. The learning rate lr represents the generator network learning rate, which is three time the discriminator learning rate. Further, the learning rate decay lr_{dec} , and the loss weight of the True/Fake loss w_{TF} , the auxiliary loss w_{AUX} and the lambda loss w_{E_λ} are provided.

Model	G1	G2	G3	Conv3D
lr	0.005	0.005	0.00025	0.00091
lr_{dec}	0.012	0.01	0.004	0.011
w_{TF}	4.55	5.60	3.91	3.65
w_{AUX}	1.05	0.21	0.48	0.46
w_{E_λ}	0.16	0.12	0.41	0.52

A.2 The GAN Training Losses

In the table A.2 the training losses of the last training epoch for the various GAN models are listed. It can be seen that the Conv3D and the G1 model achieve almost similar

loss values, while the G2 model losses are slightly larger. The G3 model achieves a low AUX and Lambda loss, but the True/Fake loss is clearly larger.

Table A.2: The training losses of the last training epoch for the GAN models. The weighted total loss and the loss components: True/Fake Loss, AUX loss and Lambda loss are provided.

Model	Total	True/Fake	AUX	Lambda
Conv3D	6.12	0.78	5.77	5.64
G1 3-branches	6.17	0.78	5.43	5.41
G2 25-branches	6.59	0.80	6.64	6.28
G3 1-branches	13.12	1.00	5.12	5.12

A.3 SSIM Index: Determination of the Adjustable Parameter

In figure A.1 the SSIM index is plotted for several values of the adjustable parameter L . For smaller L values, the SSIM index stabilizes as the lines for each model no longer change. The SSIM index was chosen to stabilize at $L = 0.0001$.

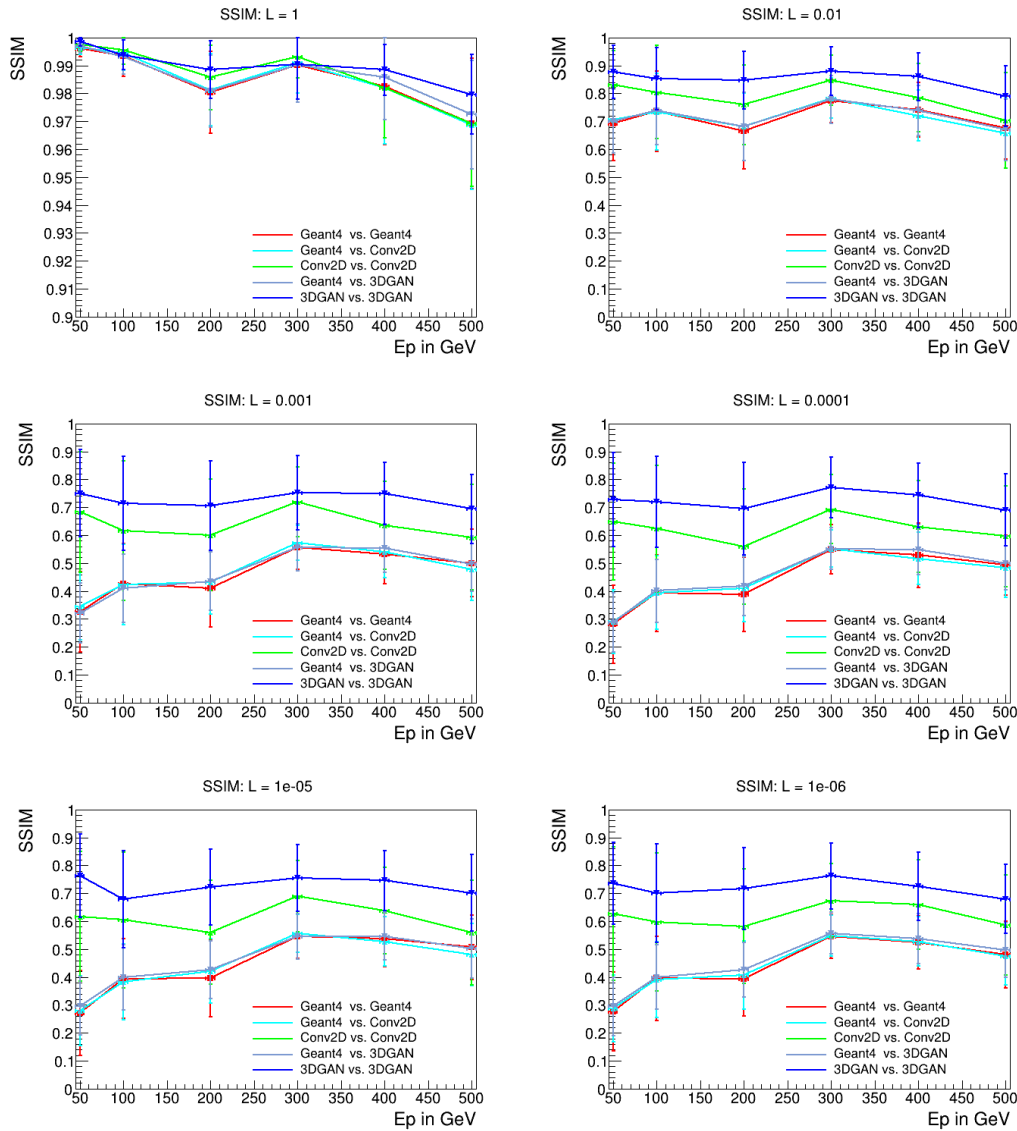


Figure A.1: The SSIM index is visualized for various values of the adjusting parameter L from 1.0 down to 10^{-6} . Different data set configurations between the Geant4, the Conv2D and the 3DGAN data are provided.

A.4 Pixel-wise Correlation Plots

Figure A.2 (top) shows the pixel-wise correlation plots for the 20 000 Geant4 images of test data set for the entire 10-510 GeV range. The correlation evaluates when one pixel energy of one pixel changes, how much this affects the other pixels. The correlation coefficient is in the range of $[-1, 1]$, where -1 corresponds to a perfect negative correlation, 1 to a perfect positive correlation and zero to no correlation. On the left side the correlations on the x-axis are shown, in the middle on the y-axis and on the right on the z-axis. The x- and y-axis correlation plots look similar because the calorimeter is designed to be rotationally invariant with respect to the shower axis. Except for the zero pixel, there exists a nice continuous correlation pattern. The z-axis correlation plot along the longitudinal shower direction shows a different pattern: pixels close together are much more correlated than pixels farther away, as indicated by the yellow diagonal line. It can be noted that the yellow diagonal narrows around pixel 17, resulting in the two positively correlated clusters at the downsampled eight pixel correlation plot, for example shown in figure 9.5. The interpretation of the correlations present in the shower images is very difficult, and in this thesis only the agreement between Geant4 and the generative models is evaluated.

In figure A.2 (bottom) the pixel-wise correlation plots for the Conv2D (G1 three-branches) model are shown, as described in section 5.3.2. Similar as the Geant4 plot, the Conv2D plot is created by 20 000 images for the full energy range. The x- and y-axis correlations look very similar, except for the zero pixel. Further investigations should explore why the zero pixel is off. In the greener area in the upper right and lower left for the long range correlations, the granularity of the Conv2D plots in the x- and y-axis looks slightly different, which might be artefacts from the convolution operations. Perhaps other kernel/filter sizes would improve on this. Also in the z-axis plot, the overall correlation pattern is learned correctly. However, the Conv2D model performs slightly worse on the first three pixels in the upper left corner, as indicated by abrupt cell color changes. Overall, the Conv2D model performance on the correlation plot is good and satisfying. This metric was first developed for the downsampled quantum models and only applied to the 25x25x25 pixel images after the classical GAN studies of this thesis were completed. In further GAN tests, this metric should be considered as an important accuracy measure for evaluating the GAN models.

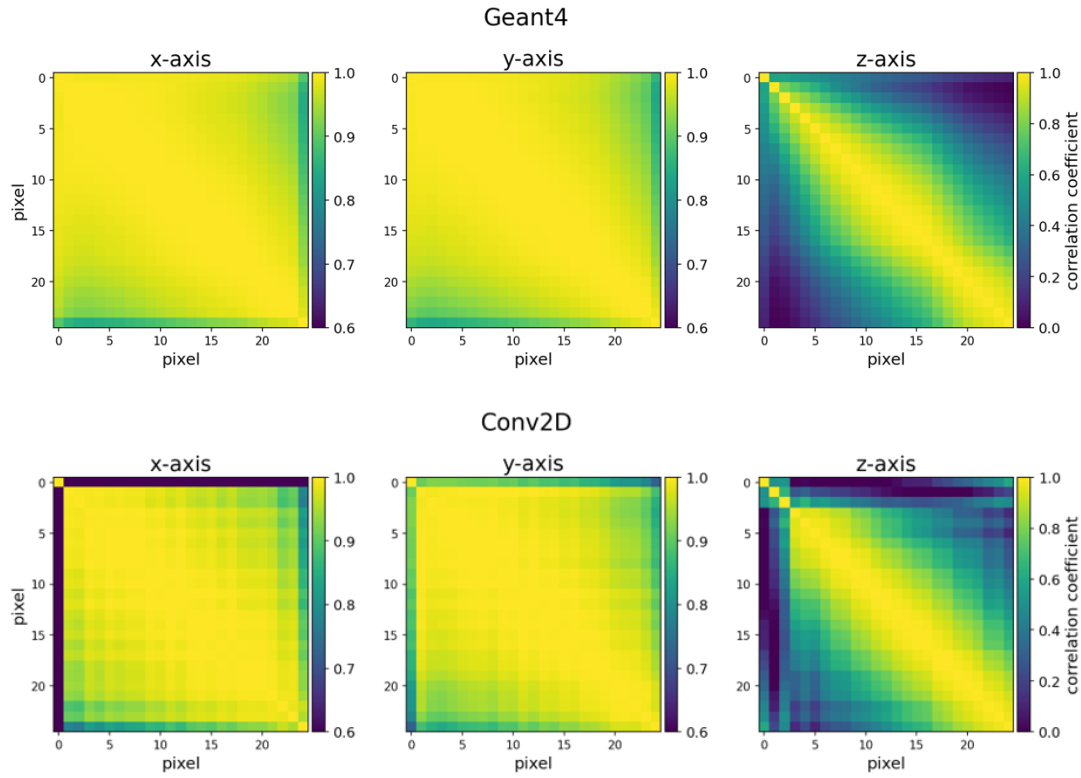


Figure A.2: The pixel-wise correlation plots for the Geant4 test data (top) and for images generated from the Conv2D model (bottom) for 10-510 GeV primary particles. The correlations for the calorimeter x-axis are shown in the left plot, for the y-axis in the middle and for the z-axis on the right. Note: the x- and y-axis correlation plots use a different color scale for the correlation coefficient than in the z-axis plot.

Appendix B

Appendix Quantum Computing

B.1 The CZ-Entangling Gate

This section demonstrates that the control and target qubit in the CZ -gate can be swapped. Three possible visualization methods of the CZ -gate are shown in figure B.1.

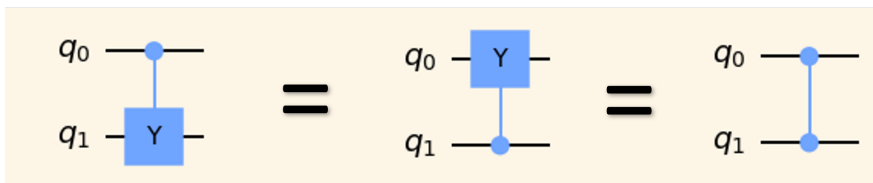


Figure B.1: The different visualization methods of the CZ -gates.

For the left visualization it can be calculated:

- $|0\rangle \otimes |0\rangle \mapsto |0\rangle \otimes |0\rangle$
- $|0\rangle \otimes |1\rangle \mapsto |0\rangle \otimes |1\rangle$
- $|1\rangle \otimes |0\rangle \mapsto |1\rangle \otimes |0\rangle$
- $|1\rangle \otimes |1\rangle \mapsto |1\rangle \otimes (-|1\rangle)$.

By applying the CZ -gate, only the sign of qubit q_1 changes when both qubits are in state $|1\rangle$.

For the middle visualization, where the control qubit is swapped, it can be calculated:

- $|0\rangle \otimes |0\rangle \mapsto |0\rangle \otimes |0\rangle$
- $|0\rangle \otimes |1\rangle \mapsto |0\rangle \otimes |1\rangle$
- $|1\rangle \otimes |0\rangle \mapsto |1\rangle \otimes |0\rangle$
- $|1\rangle \otimes |1\rangle \mapsto (-|1\rangle) \otimes |1\rangle$.

Here, only the sign of qubit q_0 changes when both qubits are in state $|1\rangle$. Due to bilinearity of the tensor product operation:

$$|1\rangle \otimes (-|1\rangle) = (-|1\rangle) \otimes |1\rangle. \quad (\text{B.1})$$

Therefore, both expressions are the same, and it does not matter which qubit controls the other.

B.2 QAG Inference Algorithm

The algorithm 1 shows the pseudo code for generating one image using the QAG model from section 9.2.1. Here the MERA-up architecture is used as an example. The standard deviation σ_{pixel} for the energy distribution of each pixel is obtained from the training data.

Algorithm 1 Quantum Angle Generator Inference

Input: nb_{shots} ▷ Number of measurements
 σ_{pixel} ▷ Array of the pixel standard deviations

Output: E ▷ Array of pixel energies forming the image

Begin

Initialize all 8 qubits to state $|0\rangle$
Randomly draw value v from uniform distribution $[-0.25, 0.25]$
for all index, qubits **do**
 Apply Hadamard gate H to qubit
 Randomly draw Ω from uniform distribution $[-1, 1]$
 $\Omega_{\text{pixel}} = \Omega \cdot v \cdot \sigma_{\text{pixel}}[\text{index}]$
 Apply rotation $RY(\Omega_{\text{pixel}})$ to qubit
end for
Apply the MERA-up unitary transformation to all qubits
Measure the qubits nb_{shots} times
Count number of times state $|0\rangle$ is measured $\rightarrow$ counts($|0\rangle$)
Compute θ according to equation 9.2
Compute pixel energies E according to transformation 9.4
Return E

End

B.3 QAG Circuit Architecture Search

In table B.1 all tested circuit architectures from the circuit study in section 9.2.3 are listed and its composition verbally described.

Table B.1: A verbal description of all circuits used in the circuit study. The plain architectures are visualized in figure 9.7.

Circuit Name	Description of the Circuit Name
Linear	A simple architecture with a linear entanglement strategy.
TTN	The plain Tree Tensor Network (TTN) architecture with RY gates.
TTN_Rz	The TTN architecture with additional RZ gates after each RY gate.
MERA	The plain Multi-scale Entanglement Renormalization Ansatz (MERA).
MERA_Rz	The MERA architecture with additional RZ gates after each RY gate.
MERA-up	The plain MERA-upsampling architecture.
MERA-up_d2	The MERA-upsampling architecture with depth of two (d2). Depth two means that the entire circuit is repeated a second time.
MERA-up_Rz	The MERA-upsampling architecture with additional RZ gates after each Ry gate.
MERA-up_d2_Rz	The MERA-upsampling architecture with depth of two (d2) and RZ gates after each RY gate.

B.4 QAG Circuit Study MMD Loss

The final MMD losses of all circuits evaluated in the QAG circuit study from section 9.2.3 are shown in figure B.2. No systematical pattern regarding the number of parameters N_p , the expressibility X and the entanglement capability E can be observed.

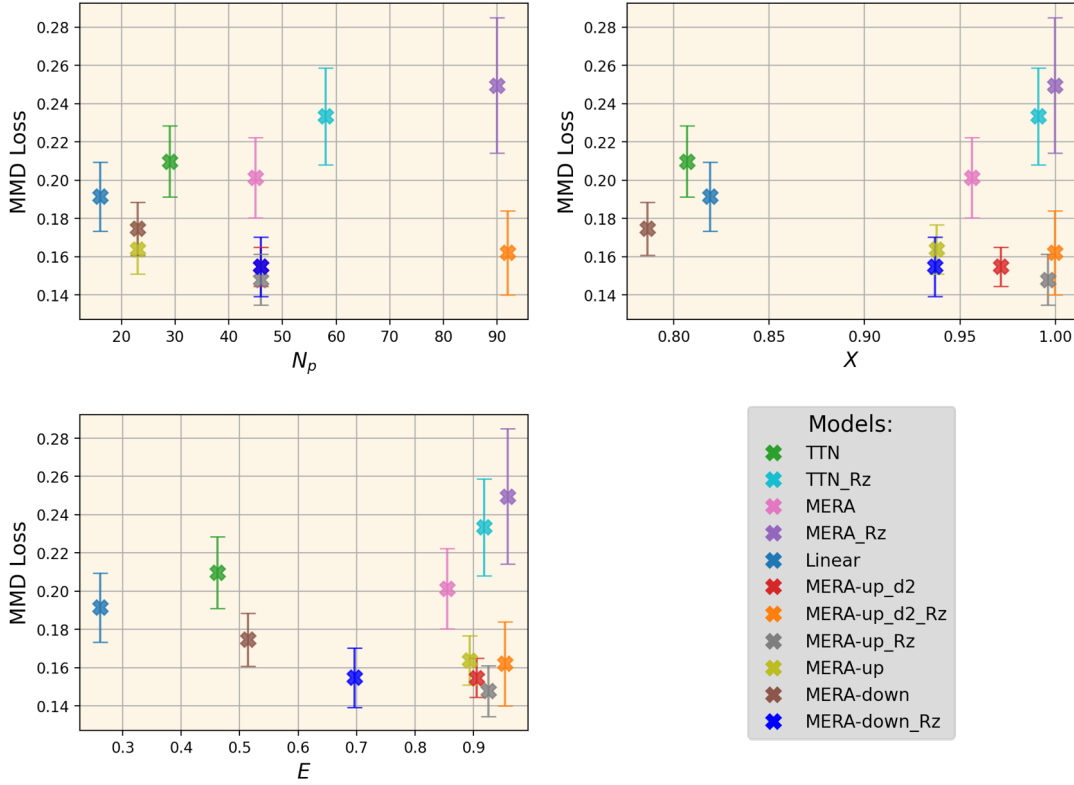


Figure B.2: The MMD losses of the various trained circuit architectures as a function of the circuit characteristic numbers are provided. The MMD loss is plotted as a function of N_p (top left), X (top right) and E (bottom left).

B.5 Mathematical Definitions of the Accuracy Metrics

In this section, the detailed formulas for the inference accuracy metrics discussed in section 9.3.2 are provided.

B.5.1 MSE of the Average Shower Shape

The Mean Squared Error (MSE) for the average calorimeter shower shape from section 9.3.2 is derived from the difference between the average images of Geant4 and the QAG model, calculated over 980 generated inference images and the corresponding 980 Geant4 images. In the subsequent equations, k iterates over the $N = 980$ images and i denotes the i^{th} pixel of each image. Hence, $E_{k,i}^{\text{Geant4}}$ (Geant4) and $E_{k,i}^{\text{QAG}}$ (QAG model) correspond to the energy of pixel i from image k . The average pixel energies $\overline{E}_i^{\text{Geant4}}$ and $\overline{E}_i^{\text{QAG}}$ are computed as follows:

$$\overline{E}_i^{\text{Geant4}} = \frac{1}{N} \sum_{k=1}^N E_{k,i}^{\text{Geant4}} \quad \text{and} \quad \overline{E}_i^{\text{QAG}} = \frac{1}{N} \sum_{k=1}^N E_{k,i}^{\text{QAG}}. \quad (\text{B.2})$$

The MSE is then calculated by iterating over the $M = 8$ individual pixels by the index i :

$$\text{MSE} = \frac{1}{M} \sum_{i=1}^M \left(\overline{E}_i^{\text{Geant4}} - \overline{E}_i^{\text{QAG}} \right)^2. \quad (\text{B.3})$$

As the MSE corresponds to an error, the smaller the value, the better the accuracy of the QAG model.

B.5.2 Energy Sum Histogram

The equations to calculate the terms μ and σ from section 9.3.2 are provided in the following. Each histogram entry (or hit) is the total energy $\hat{E}$ of a shower image:

$$\hat{E}^{\text{Geant4}} = \sum_{i=1}^M E_i^{\text{Geant4}} \quad \text{and} \quad \hat{E}^{\text{QAG}} = \sum_{i=1}^M E_i^{\text{QAG}}, \quad (\text{B.4})$$

where i is iterating over the $M = 8$ pixels.

The means of the distributions for our data sets, Geant4 and QAG, are denoted as μ^{Geant4} and μ^{QAG} , respectively. Similarly, the standard deviations are represented by σ^{Geant4} and σ^{QAG} .

The means μ of the total energy for the $N = 980$ images are then calculated using the well-known equation:

$$\mu^{\text{Geant4}} = \frac{1}{N} \sum_{i=1}^N \hat{E}_i^{\text{Geant4}} \quad \text{and} \quad \mu^{\text{QAG}} = \frac{1}{N} \sum_{i=1}^N \hat{E}_i^{\text{QAG}}. \quad (\text{B.5})$$

And the standard deviations, denoted by σ , are computed as:

$$\sigma^{\text{Geant4}} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\hat{E}_i^{\text{Geant4}} - \mu^{\text{Geant4}})^2}$$

and (B.6)

$$\sigma^{\text{QAG}} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\hat{E}_i^{\text{QAG}} - \mu^{\text{QAG}})^2}$$

A closer agreement between the means and standard deviations of the two distributions indicates a higher accuracy of the QAG model.

B.6 Training Data Covariance Compared to Correlation

The pixel-wise calculated correlation plot for the downsampled eight pixel Geant4 training data is in figure B.3 compared to the pixel-wise calculated covariance plot. The covariance describes similarly to the correlation how much two variables change in mutual dependence. However, the correlation measures the proportional change whereas the covariance measures the absolute change [303]. The covariance is calculated as described in Ref. [304] with the command `cov` [305] from the NumPy library [287]. Covariance can be used to examine the relationship between two pixels. A positive covariance says that if the energy of one pixel increases, also the energy of the other pixel increases. A negative covariance states that when the energy of one pixel increases, the energy of the other pixel decreases. The comparison between the covariance plot and the correlation plot in figure B.3 reveals similar patterns. This confirms the pixel-correlation patterns in the training data and makes the pixel-wise correlation plot to an interesting metric for physics accuracy.

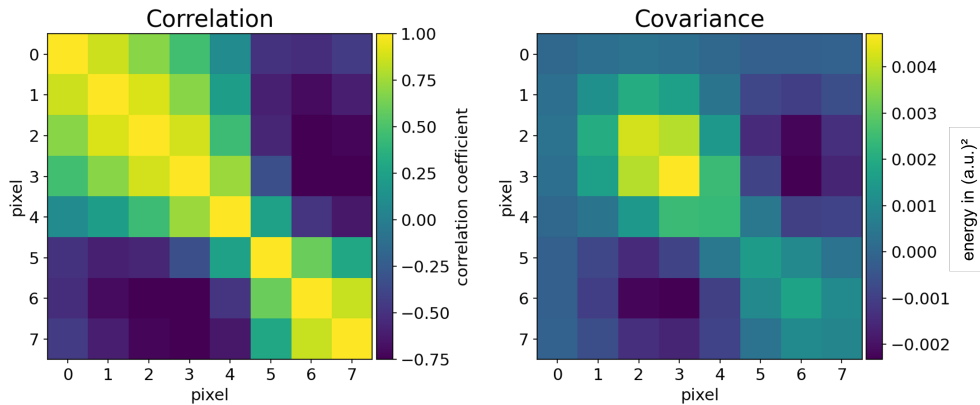


Figure B.3: (left) The pixel-wise calculated correlation plot for the downsampled eight pixel Geant4 training data compared to the (right) covariance plot.

B.7 QAG Inference Noise Study Shower Shapes

The generated shower images of the QAG inference fake noise study from chapter 9 are presented in figure B.4. Similar to the hybrid qGAN model in chapter 8, it can be seen that the shower patterns approach a random uniform distribution at larger noise levels, because the large energies decrease and the smaller ones increase. For exclusively readout noise, the shower image correctly reproduces Geant4 up to 5% of noise. From the same study, the generated shower images of applying hardware noise in inference and for executing inference on the real `ibmq_montreal` quantum device

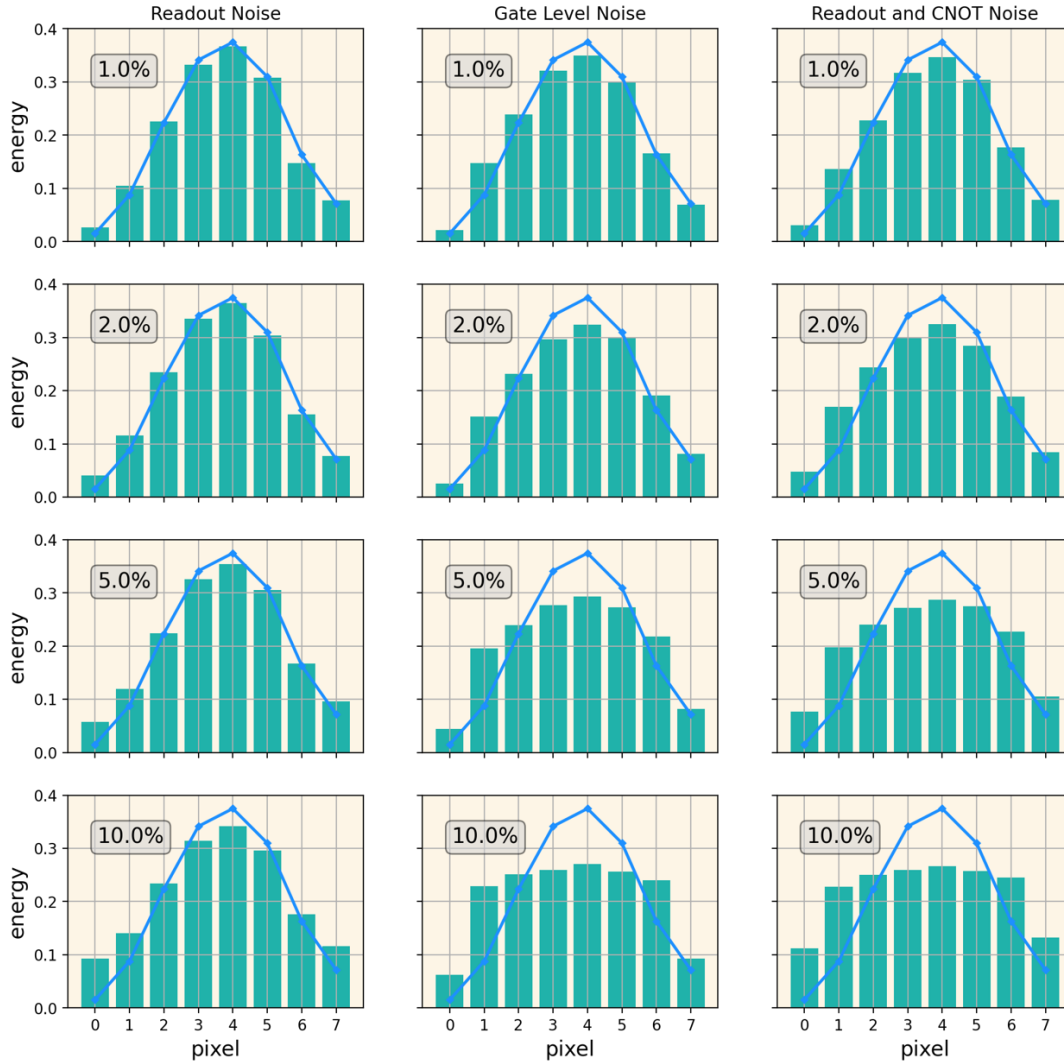


Figure B.4: Example shower images generated with various noise models and noise levels. The left row contains shower images generated with only readout noise, the middle row with only two-qubit gate level CNOT noise and the right row with both, readout and CNOT noise. From top to bottom, the noise levels in inference are increased. The noise levels of each plot are provided in the gray box inside the plot. The blue line indicates the Geant4 data as reference, and the green bars the averaged generator output.

are provided in figure B.5. The shower shape with hardware noise on the simulator deviates only slightly from Geant4. On the other side, on the real `ibmq_montreal` quantum hardware, the generated shower shape is worse.

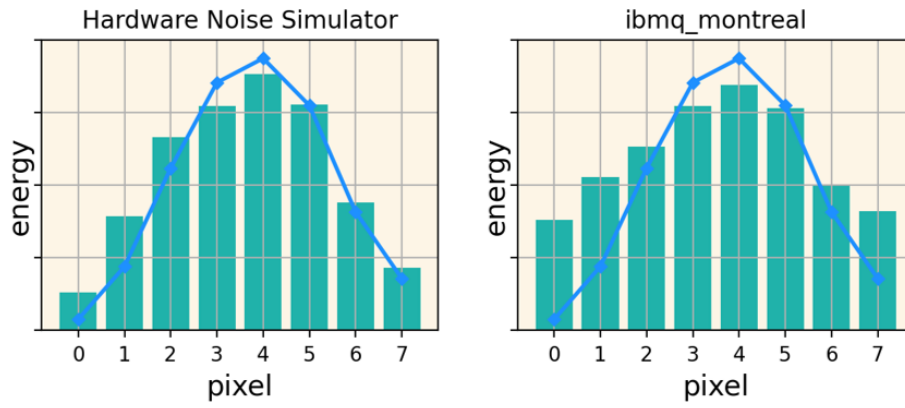


Figure B.5: The generated shower images for employing inference with hardware noise loaded into the simulator (left) and inference run with the same noise level on the real `ibmq_montreal` quantum device (right).

B.8 QAG Quantum Hardware Training

This chapter provides more information to the QAG hardware training from chapter 9. During the QAG training on `ibmq_montreal`, a calibration caused a significant change in the noise levels. The noise levels before the calibration are visualized in figure B.6 on the top and after the calibration on the bottom. It can be recognized, especially the readout noise of qubit number two in the quantum circuit architecture (indexed as qubit five on the real hardware) increased from about 2% to over 8%, while the other qubits experienced only minor noise level changes.

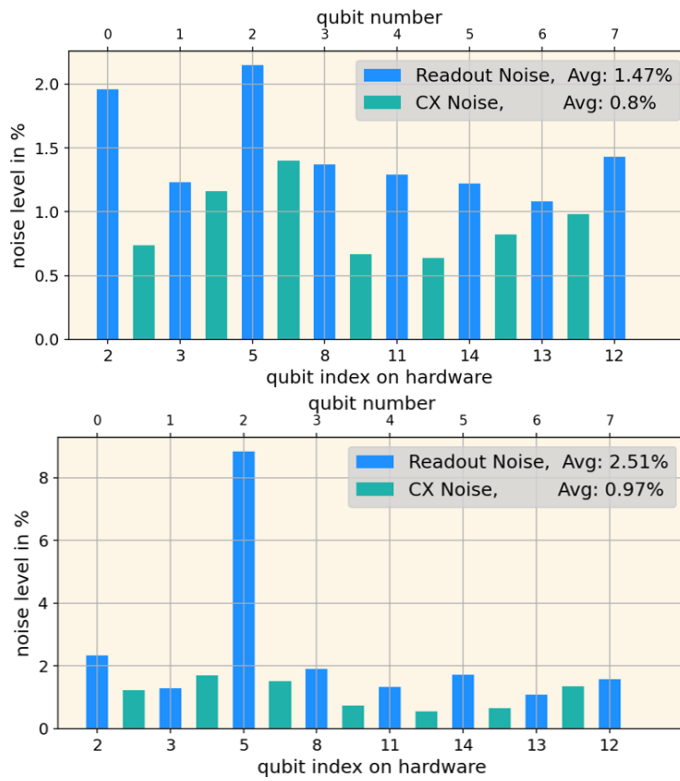


Figure B.6: On the top, the noise levels of the `ibmq_montreal` device are shown before, and on the bottom after the calibration change during training. Inference is performed with the bottom noise model. The plots have different scales on their y-axes.

The noise levels of the QAG training on the `ibmq-cairo` device are visualized in figure B.7. As it can be seen, the noise levels of all qubits have only a small variation.

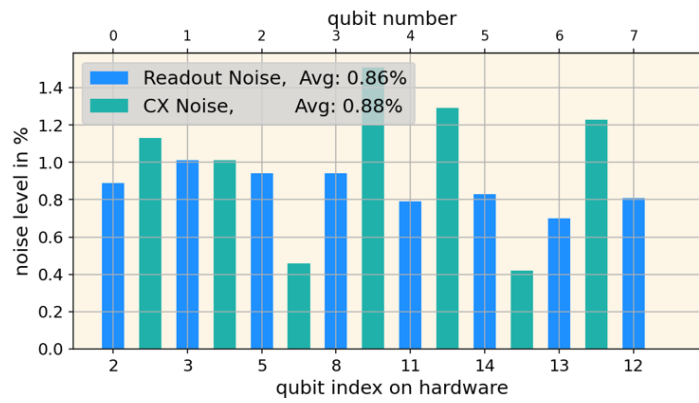


Figure B.7: The noise levels of the training on the `ibmq_cairo` device.

B.9 Full qGAN Model Accuracy Metrics

Further accuracy metrics of the best trial of the full qGAN model from chapter 10 are provided. In figure B.8 (left), the mean energy histogram of the full qGAN model differs from the one of Geant4, confirmed by the mean μ and the standard deviation σ . With k-means clustering, four clusters are found in the generated data in figure B.8 (right). Opposite to the QAG model, the shapes of the clusters do not differ considerably, especially for pixel zero, one, six and seven, there are no differences. This weak pixel energy variation is underlined by the pixel-wise energy deposition histograms in figure B.9.

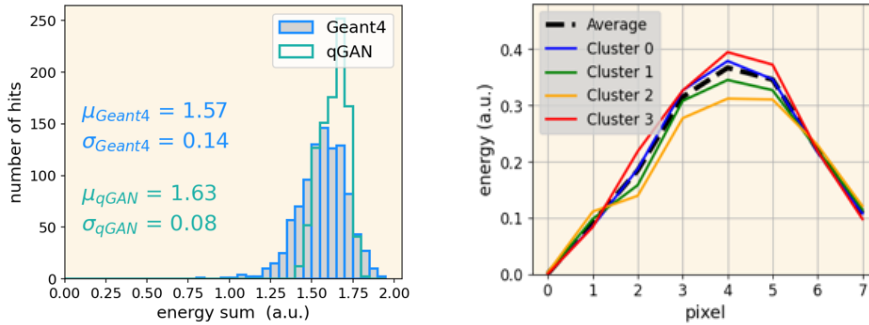


Figure B.8: (left) The pixel-wise correlation and (right) the k-means clustering metric generated by the best trial of the full qGAN model.

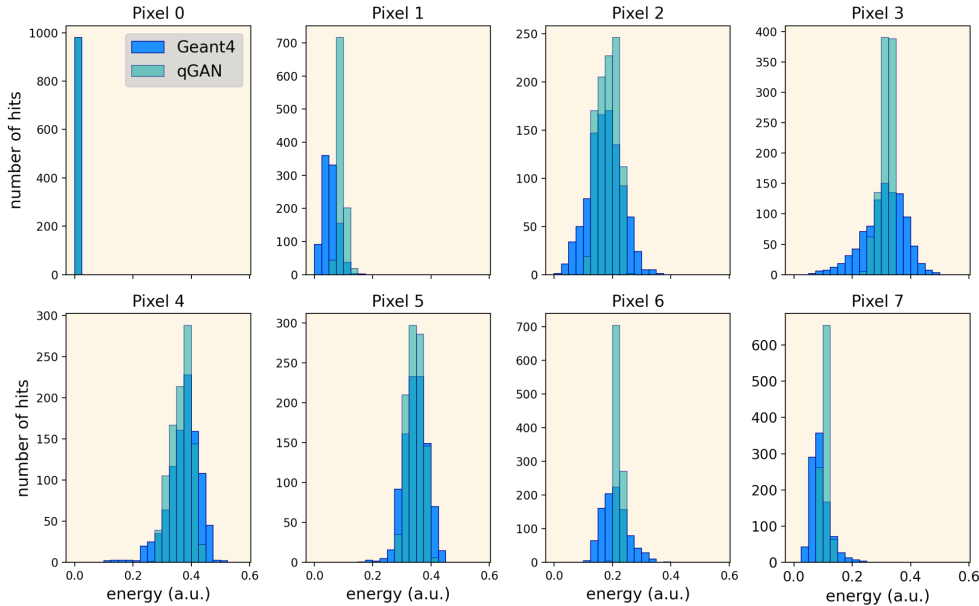


Figure B.9: The pixel-wise energy deposition histograms of the full qGAN model.

B.10 Hybrid S and Hybrid L qGAN Model Accuracy Metrics

For completeness, the accuracy metrics of the hybrid S (small discriminator) and the hybrid M (medium discriminator) qGAN model from chapter 10 are provided. The average shower shapes are visualized in figure B.10. The hybrid L model (right) is still able to visually correctly reproduce the shower shape, whereas, the hybrid S model (left) is off for some pixels. These visual results agree with the measured accuracy values from table 10.1.

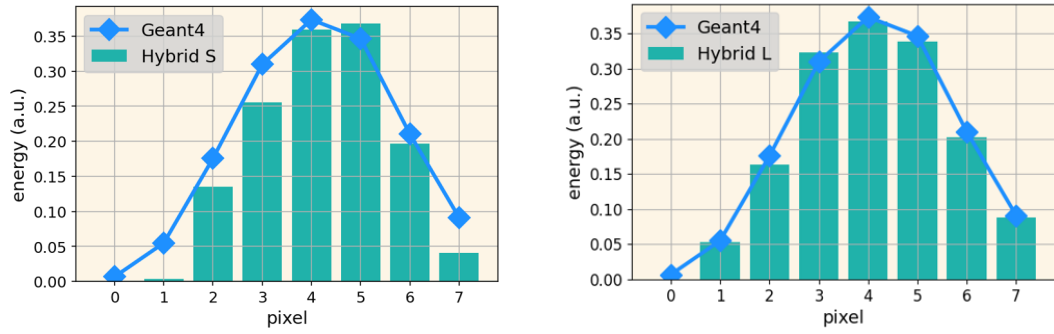


Figure B.10: (left) The average shower shape plot of the best trial of the hybrid S qGAN model and (right) the average shower shape of the best trial of the hybrid L qGAN model.

The pixel-wise correlation plots are presented in figure B.11. Both models are unable to correctly reproduce the correlation pattern from the Geant4 training data.

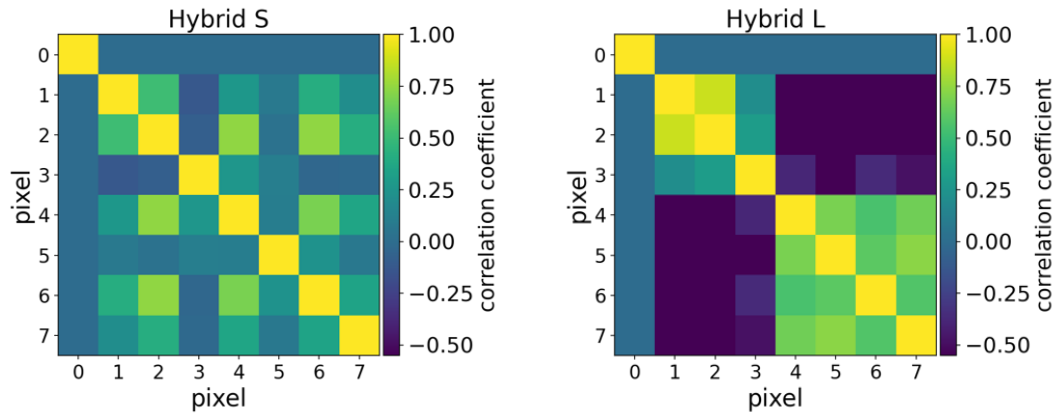


Figure B.11: (left) The pixel-wise correlation plot of the best trial of the hybrid S qGAN model and (right) the best trial of the hybrid L qGAN model.

Bibliography

- [1] T. Kibble, "The Standard Model of Particle Physics," *European Review*, vol. 23, 12 2014. [Online]. Available: <https://doi.org/10.1017/S1062798714000520>
- [2] L. Lista, "Practical Statistics for Particle Physicists," *CERN Yellow Reports: School Proceedings*, pp. Vol 5 (2017): Proceedings of the 2016 European School of High-Energy Physics, 2017. [Online]. Available: <https://e-publishing.cern.ch/index.php/CYRSP/article/view/354>
- [3] S. Heinemeyer, S. Jadach, and J. Reuter, "Theory requirements for SM Higgs and EW precision physics at the FCC-ee," 2021. [Online]. Available: <https://arxiv.org/abs/2106.11802>
- [4] L. Evans, "The large hadron collider," *New Journal of Physics*, vol. 9, no. 9, p. 335, 2007. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1367-2630/9/9/335/meta>
- [5] R. Romaniuk, K. Pozniak, C. Collaboration, C. Warsaw, and W. Zabolotny, "A New Boson with a Mass of 125 GeV Observed with the CMS Experiment at the Large Hadron Collider," *Science*, vol. 338, pp. 1569–1575, 12 2012. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0370269312008581>
- [6] L. Evans, "The Large Hadron Collider," *New Journal of Physics*, vol. 9, no. 9, p. 335, Sep 2007. [Online]. Available: <https://dx.doi.org/10.1088/1367-2630/9/9/335>
- [7] J. Apostolakis, *Detector Simulation*. Cham: Springer International Publishing, 2020, pp. 485–531. [Online]. Available: https://doi.org/10.1007/978-3-030-35318-6_11
- [8] V. Daniel Elvira, "Impact of detector simulation in particle physics collider experiments," *Physics Reports*, vol. 695, p. 1–54, Jun 2017. [Online]. Available: <http://dx.doi.org/10.1016/j.physrep.2017.06.002>

- [9] S. Agostinelli *et al.*, “GEANT4—a simulation toolkit,” vol. 506, 2003, pp. 250–303. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0168900203013688>
- [10] M. Lamont. LHC Commissioning with beam. [Online]. Available: <https://lhc-commissioning.web.cern.ch/schedule/images/LHC-nom-lumi-proj-with-ions.png>
- [11] A. Collaboration, “ATLAS Software and Computing HL-LHC Roadmap,” CERN, Geneva, Tech. Rep., 2022. [Online]. Available: <https://cds.cern.ch/record/2802918>
- [12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [13] P.-L. Dallaire-Demers and N. Killoran, “Quantum generative adversarial networks,” *Physical Review A*, vol. 98, no. 1, Jul 2018. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevA.98.012324>
- [14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Networks,” pp. 139–144, 2020. [Online]. Available: <https://papers.nips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html>
- [15] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A Survey of Quantization Methods for Efficient Neural Network Inference,” 2021. [Online]. Available: <https://arxiv.org/abs/2103.13630>
- [16] “Intel® Neural Compressor,” 2021. [Online]. Available: <https://github.com/daisyden/lpot>
- [17] *TensorFlow Lite*, TensorFlow For Mobile & IoT, Aug. 2020. [Online]. Available: <https://www.tensorflow.org/lite>
- [18] T. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. O’Brien, “Quantum computers,” *Nature*, vol. 464, no. 7285, pp. 45–53, Apr. 2010. [Online]. Available: <https://www.nature.com/articles/nature08812>
- [19] A. Delgado, K. E. Hamilton *et al.*, “Quantum Computing for Data Analysis in High-Energy Physics,” 2022. [Online]. Available: <https://arxiv.org/abs/2203.08805>
- [20] A. Di Meglio, M. Doser, B. Frisch, D. M. Grabowska, M. Pierini, and S. Vallecorsa, “CERN Quantum Technology Initiative Strategy and Roadmap,” 2021. [Online]. Available: <https://cds.cern.ch/record/2789149>

- [21] R. P. Feynman, "Simulating physics with computers," in *Feynman and computation*. CRC Press, 2018, pp. 133–153. [Online]. Available: <https://link.springer.com/article/10.1007/BF02650179>
- [22] M. Sajjan, J. Li, R. Selvarajan, S. Sureshbabu, S. Kale, R. Gupta, V. Singh, and S. Kais, "Quantum machine learning for chemistry and physics," *Chemical Society Reviews*, vol. 51, 07 2022. [Online]. Available: <https://pubs.rsc.org/en/content/articlelanding/2022/cs/d2cs00203e>
- [23] J. X. Tong Li, Shibin Zhang, "Quantum Generative Adversarial Network: A Survey," *Computers, Materials & Continua*, vol. 64, no. 1, pp. 401–438, 2020. [Online]. Available: <http://www.techscience.com/cmc/v64n1/39150>
- [24] S. Knutzen and M. Weber, "Measurement of the $pp \rightarrow t\bar{t}$ cross-section and the top-quark mass with the CMS detector," 2013. [Online]. Available: https://www.institut3b.physik.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaajwffp
- [25] K. Lamichhane, "Search for New Bosons in Gluon-Gluon and Vector Boson Fusion Processes at the LHC and Development of Silicon Module Assembly Techniques for the CMS High Granularity Calorimeter," 2020. [Online]. Available: <https://cds.cern.ch/record/2791475>
- [26] S. Druger, A. Kolakowska *et al.*, *University Physics*, 09 2016, vol. 3. [Online]. Available: <https://openstax.org/details/books/university-physics-volume-1>
- [27] M. T. et al. (Particle Data Group), "Particle Physics Booklet," *Review of Particle Physics*, 2018. [Online]. Available: <http://pdg.lbl.gov/>
- [28] M. Fairbairn, A. Kraan, D. Milstead, T. Sjöstrand, P. Skands, and T. Sloan, "Stable massive particles at colliders," *Physics Reports*, vol. 438, no. 1, pp. 1–63, 2007. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0370157306004054>
- [29] M. D. Schwartz, *Quantum Field Theory and the Standard Model*. Cambridge University Press, 2013. [Online]. Available: <https://schwartzqft.fas.harvard.edu>
- [30] J. Häuser and W. Dröscher, "Gravity beyond Einstein? Part I: Physics and the Trouble with Experiments," *Zeitschrift für Naturforschung a*, vol. 72, 08 2017. [Online]. Available: <https://doi.org/10.1515/zna-2016-0479>
- [31] C. Collaboration, "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC," *Physics Letters B*, vol. 716, no. 1, pp. 30–61, 2012. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0370269312008581>

- [32] U. Schubert and C. Williams, “Interplay between SM precision, BSM physics, and the measurements of α_{had} in $\mu - e$ scattering,” *Phys. Rev. D*, vol. 100, p. 035030, Aug 2019. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevD.100.035030>
- [33] V. A. Mitsou, “Overview of searches for dark matter at the LHC,” *Journal of Physics: Conference Series*, vol. 651, p. 012023, Nov 2015. [Online]. Available: <https://doi.org/10.1088/1742-6596/651/1/012023>
- [34] H. Nunokawa, S. Parke, and J. W. Valle, “CP violation and neutrino oscillations,” *Progress in Particle and Nuclear Physics*, vol. 60, no. 2, p. 338–402, Apr 2008. [Online]. Available: <http://dx.doi.org/10.1016/j.ppnp.2007.10.001>
- [35] M. Yoshimura, “Yet another symmetry breaking to be discovered,” *Progress of Theoretical and Experimental Physics*, vol. 2016, no. 7, p. 07B103, Jun 2016. [Online]. Available: <https://doi.org/10.1093%2Fptep%2Fptw056>
- [36] S. P. Martin, “A supersymmetry primer,” in *Perspectives on supersymmetry II*. World Scientific, 2010, pp. 1–153.
- [37] L. Bernard, C. Deffayet, K. Hinterbichler, and M. von Strauss, “Partially massless graviton on beyond Einstein spacetimes,” *Physical Review D*, vol. 95, no. 12, Jun 2017. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevD.95.124036>
- [38] N. van Remortel, “The nature of natural units,” 2016. [Online]. Available: <https://www.nature.com/articles/nphys3950>
- [39] F. Pisano and N. O. Reis, “Natural units, numbers and numerical clusters,” 2001. [Online]. Available: <https://arxiv.org/abs/hep-ph/0112097>
- [40] M. El Naschie, “On the vital role played by the electron-volt units system in high energy physics and mach’s principle of “denkökonomie”,” *Chaos, Solitons & Fractals*, vol. 28, no. 5, pp. 1366–1371, 2006. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0960077905010763>
- [41] E. Lopienska, “The CERN accelerator complex, layout in 2022. Complexe des accélérateurs du CERN en janvier 2022,” 2022, general Photo. [Online]. Available: <https://cds.cern.ch/record/2800984>
- [42] S. Mattei, “H[−] ion source for CERN’s Linac4 accelerator: simulation, experimental validation and optimization of the hydrogen plasma,” 2017, presented 24 Jul 2017. [Online]. Available: <https://cds.cern.ch/record/2277635>
- [43] R. Appleby, B. Goddard, A. Gomez-Alonso, V. Kain, T. Kramer, D. Macina, R. Schmidt, and J. Wenninger, “Beam-related machine protection for the CERN Large Hadron Collider experiments,” *Physical Review Special Topics - Accelerators and Beams*, vol. 20, 06 2010. [Online]. Available: <https://journals.aps.org/prab/abstract/10.1103/PhysRevSTAB.13.061002>

- [44] J. Jowett, "Colliding Heavy Ions in the LHC," p. TUXGBD2. 6 p, 2018. [Online]. Available: <https://cds.cern.ch/record/2648704>
- [45] S. Chatrchyan, G. Hmayakyan *et al.*, "The CMS experiment at the CERN LHC. The Compact Muon Solenoid experiment," *JINST*, vol. 3, p. S08004, 2008, also published by CERN Geneva in 2010. [Online]. Available: <https://cds.cern.ch/record/1129810>
- [46] "Facts and figures about the LHC," 2021. [Online]. Available: <https://home.cern/resources/faqs/facts-and-figures-about-lhc>
- [47] J. Albrecht, A. A. Alves, G. Amadio, G. Andronico, N. Anh-Ky, L. Aphecetche, J. Apostolakis, M. Asai, L. Atzori, and *et al.*, "A Roadmap for HEP Software and Computing R&D for the 2020s," *Computing and Software for Big Science*, vol. 3, no. 1, Mar 2019. [Online]. Available: <http://dx.doi.org/10.1007/s41781-018-0018-8>
- [48] W. Herr and B. Muratori, "Concept of luminosity," in *CERN Accelerator School and DESY Zeuthen: Accelerator Physics*, 9 2003, pp. 361–377. [Online]. Available: <http://cds.cern.ch/record/941318?ln=de>
- [49] G. Apollinari, B. Alonso *et al.*, "High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report V. 0.1," vol. 4/2017, 11 2017. [Online]. Available: <https://cds.cern.ch/record/2749422?ln=de>
- [50] J. B. Dainton, M. Klein, P. Newman, E. Perez, and F. Willeke, "Deep inelastic electron-nucleon scattering at the LHC," *Journal of Instrumentation*, vol. 1, no. 10, pp. P10 001–P10 001, Oct 2006. [Online]. Available: <https://doi.org/10.1088%2F1748-0221%2F1%2F10%2Fp10001>
- [51] H. Wiedemann, *Particle Accelerator Physics*. Springer, Cham, 2015, vol. 4. [Online]. Available: <https://doi.org/10.1007/978-3-319-18317-6>
- [52] K. Wille and J. McFall, *The Physics of Particle Accelerators: An Introduction*, ser. The Physics of Particle Accelerators: An Introduction. Oxford University Press, 2000. [Online]. Available: <https://books.google.fr/books?id=k1STXLP4lkgC>
- [53] H. Kolanoski and N. Wermes, *Particle Detectors: Fundamentals and Applications*. Oxford University Press, 2020. [Online]. Available: <https://books.google.ch/books?id=7-4-zQEACAAJ>
- [54] *LHC Workshop on Experimental Conditions and Beam-Induced Detectors Backgrounds: CERN, Geneva, Switzerland 3 - 4 Apr 2008*. LHC Workshop on Experimental Conditions and Beam-Induced Detectors Backgrounds, CERN. Geneva: CERN, 2009. [Online]. Available: <http://cds.cern.ch/record/1184436>
- [55] D. Griffiths, *Introduction to elementary particles*. John Wiley & Sons, 2020.

- [56] R. Brown and D. Cockerill, "Electromagnetic calorimetry," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 666, pp. 47 – 79, 2012, advanced Instrumentation. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0168900211005572>
- [57] C. W. Fabjan and F. Gianotti, "Calorimetry for particle physics," *Rev. Mod. Phys.*, vol. 75, pp. 1243–1286, Oct 2003. [Online]. Available: <https://link.aps.org/doi/10.1103/RevModPhys.75.1243>
- [58] R. Jost, J. M. Luttinger, and M. Slotnick, "Distribution of Recoil Nucleus in Pair Production by Photons," *Phys. Rev.*, vol. 80, pp. 189–196, 1950. [Online]. Available: <https://journals.aps.org/pr/abstract/10.1103/PhysRev.80.189>
- [59] M. Diemoz, "The electromagnetic calorimeter of the cms experiment," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 581, no. 1, pp. 380–383, 2007, vCI 2007. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0168900207016762>
- [60] F. Ferri, "The CMS ECAL Phase-2 Upgrade for High Precision Energy and Timing Measurements," CERN, Geneva, Tech. Rep., 2019. [Online]. Available: <https://cds.cern.ch/record/2797724>
- [61] B. Webber, "Hadronization," 1994. [Online]. Available: <https://arxiv.org/abs/hep-ph/9411384>
- [62] S. Dris, C. Foudas, and J. Troska, "Performance of the CMS Tracker Optical Links and Future Upgrade Using Bandwidth Efficient Digital Modulation," 04 2010. [Online]. Available: <https://arxiv.org/abs/1004.5574>
- [63] V. Chekalina, E. Orlova, F. Ratnikov, D. Ulyanov, A. Ustyuzhanin, and E. Zakharov, "Generative Models for Fast Calorimeter Simulation: the LHCb case," *EPJ Web of Conferences*, vol. 214, p. 02034, 2019. [Online]. Available: <http://dx.doi.org/10.1051/epjconf/201921402034>
- [64] S. Belforte, A. Fanfani, I. Fisk, J. Flix, J. Hernandez, J. Klem, J. Letts, N. Magini, V. Miccio, S. Padhi *et al.*, "The commissioning of cms computing centres in the worldwide lhc computing grid," in *2008 IEEE Nuclear Science Symposium Conference Record*. IEEE, 2008, pp. 1963–1970.
- [65] CERN. Worldwide lhc computing grid. [Online]. Available: <https://wlcg-public.web.cern.ch/>
- [66] I. Bird, P. Buncic, F. Carminati, M. Cattaneo, P. Clarke, I. Fisk, M. Girone, J. Harvey, B. Kersevan, P. Mato, R. Mount, and B. Panzer-Steindel, "Update of the Computing Models of the WLCG and the LHC Experiments," Tech. Rep., 2014. [Online]. Available: <https://cds.cern.ch/record/1695401>

- [67] L. Clissa, "Survey of Big Data sizes in 2021," 2022. [Online]. Available: <https://arxiv.org/abs/2202.07659>
- [68] by Mélissa Gaillard and S. Pandolfi, "CERN Data Centre passes the 200-petabyte milestone," 2017. [Online]. Available: <https://cds.cern.ch/record/2276551>
- [69] J. Pearlstein, "Big Data Has Arrived at an Almost Unimaginable Scale," 2013. [Online]. Available: <https://www.wired.com/2013/04/bigdata>
- [70] R. C. Tim Berners-Lee, "World-Wide Web," 1989. [Online]. Available: <https://cds.cern.ch/record/245440/files/p69.pdf>
- [71] C. O. Software and Computing, "CMS Phase-2 Computing Model: Update Document," CERN, Geneva, Tech. Rep., 2022. [Online]. Available: <https://cds.cern.ch/record/2815292>
- [72] CMS Offline and Computing Public Results. Accessed in October 2022. [Online]. Available: <https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults>
- [73] O. Mattelaer, "On the maximal use of Monte Carlo samples: re-weighting events at NLO accuracy," *The European Physical Journal C*, vol. 76, no. 12, Dec 2016. [Online]. Available: <http://dx.doi.org/10.1140/epjc/s10052-016-4533-7>
- [74] J. Cruz-Martinez, N. Glover, T. Gehrmann, and A. Huss, "NNLO corrections to VBF Higgs boson production," 2018. [Online]. Available: <https://arxiv.org/abs/1807.07908>
- [75] D. Salamani, S. Gadatsch, T. Golling, G. Stewart, A. Ghosh, D. Rousseau, A. Hasib, and J. Schaarschmidt, "Deep Generative Models for Fast Shower Simulation in ATLAS," 10 2018, pp. 348–348. [Online]. Available: <http://cds.cern.ch/record/2680531>
- [76] G. Aad *et al.*, "AtlFast3: the next generation of fast simulation in ATLAS," *Comput. Softw. Big Sci.*, vol. 6, p. 7, 2022. [Online]. Available: <https://link.springer.com/article/10.1007/s41781-021-00079-7>
- [77] G. J. Erdmann M. and Q. T., "Precise Simulation of Electromagnetic Calorimeter Showers Using a Wasserstein Generative Adversarial Network," *Comput Softw Big Sci* 3, 1 2019. [Online]. Available: <https://doi.org/10.1007/s41781-018-0018-8>
- [78] G. Rukhkhattak, S. Vallecorsa, and F. Carminati, "Three dimensional energy parametrized generative adversarial networks for electromagnetic shower simulation," 2018 *25th IEEE International Conference on Image Processing (ICIP)*, pp. 3913–3917, 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8451587>

- [79] E. Barberio, J. Boudreau, B. Butler, S. L. Cheung, and A. Dell, "Fast simulation of electromagnetic showers in the ATLAS calorimeter: Frozen showers," *Journal of Physics: Conference Series*, vol. 160, p. 012082, Apr 2009. [Online]. Available: <https://doi.org/10.1088/1742-6596/160/1/012082>
- [80] de Oliveira *et al.*, "Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis," vol. 1, no. 1. Springer Science and Business Media LLC, Sep 2017.
- [81] M. Paganini *et al.*, "CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks," vol. 97, no. 1. American Physical Society (APS), Jan 2018. [Online]. Available: <https://doi.org/10.1103/PhysRevD.97.014021>
- [82] M. Sami and I. Mobin, "A Comparative Study on Variational Autoencoders and Generative Adversarial Networks," in *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIIT)*, 2019, pp. 1–5. [Online]. Available: <https://ieeexplore.ieee.org/document/8834544>
- [83] D. Sipio *et al.*, "DijetGAN: a Generative-Adversarial Network approach for the simulation of QCD dijet events at the LHC." Springer Science and Business Media, Aug 2019. [Online]. Available: [https://link.springer.com/article/10.1007/JHEP08\(2019\)110](https://link.springer.com/article/10.1007/JHEP08(2019)110)
- [84] P. Lebrun, L. Linssen, A. Lucaci-Timoce, D. Schulte, F. Simon, S. Stapnes, N. Toge, H. Weerts, and J. Wells, "The CLIC Programme: Towards a Staged e+e- Linear Collider Exploring the Terascale : CLIC Conceptual Design Report," 2012. [Online]. Available: <https://arxiv.org/abs/1209.2543>
- [85] D. Belayneh, F. Carminati, A. Farbin, B. Hooberman, G. Khattak *et al.*, "Calorimetry with deep learning: particle simulation and reconstruction for collider physics," *The European Physical Journal C*, vol. 80, no. 7, Jul 2020. [Online]. Available: <https://doi.org/10.1140/epjc/s10052-020-8251-9>
- [86] M. Pierini and M. Zhang, "CLIC Calorimeter 3D images: Electron showers at Fixed Angle," Jan. 2020. [Online]. Available: <https://doi.org/10.5281/zenodo.3603122>
- [87] Energy measurement in calorimeters. [Online]. Available: <http://www.particles.uni-freiburg.de/dateien/vorlesungsdateien/particledetectors/kap8>
- [88] N. Sharma, R. Sharma, and N. Jindal, "Machine Learning and Deep Learning Applications-A Vision," *Global Transitions Proceedings*, vol. 2, no. 1, pp. 24–28, 2021, 1st International Conference on Advances in Information, Computing and Trends in Data Engineering (AICDE - 2020). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2666285X21000042>

- [89] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to Machine Learning, Neural Networks, and Deep Learning," *Translational Vision Science & Technology*, vol. 9, no. 2, pp. 14–14, 02 2020. [Online]. Available: <https://doi.org/10.1167/tvst.9.2.14>
- [90] J. Chai, H. Zeng, A. Li, and E. W. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100134, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2666827021000670>
- [91] *Google Trends*. Google, <https://trends.google.de/trends/>, Accessed in June 2022.
- [92] X. Wang, Y. Zhao, and F. Pourpanah, "Recent advances in deep learning," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 4, pp. 747–750, 2020. [Online]. Available: <https://link.springer.com/article/10.1007/s13042-020-01096-5>
- [93] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Networks*, vol. 111, pp. 47–63, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0893608018303332>
- [94] D. Shetty, J. Varma, S. Navi, and M. R. Ahmed, "Diving Deep into Deep Learning: History, Evolution, Types and Applications," *The International Journal on Media Management*, vol. Volume-9, pp. 2278–3075, 01 2020.
- [95] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi, "Scaling for edge inference of deep neural networks," *Nature Electronics*, vol. 1, no. 4, pp. 216–222, 2018. [Online]. Available: <https://www.nature.com/articles/s41928-018-0059-3>
- [96] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer. [Online]. Available: <https://www.bibsonomy.org/bibtex/26681a2e990d3d35887d519e2ec3bd7a9/flint63>
- [97] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/0925231293900060>
- [98] A. Oussidi and A. Elhassouny, "Deep generative models: Survey," 04 2018, pp. 1–8. [Online]. Available: <https://ieeexplore.ieee.org/document/8354080>
- [99] H. GM, M. K. Gourisaria, M. Pandey, and S. S. Rautaray, "A comprehensive survey and analysis of generative models in machine learning," *Computer Science Review*, vol. 38, p. 100285, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1574013720303853>

- [100] G. Batzolis, J. Stanczuk, C.-B. Schönlieb, and C. Etmann, "Conditional image generation with score-based diffusion models," 2021. [Online]. Available: <https://arxiv.org/abs/2111.13606>
- [101] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf>
- [102] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, Y. Shao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," 2022. [Online]. Available: <https://arxiv.org/abs/2209.00796>
- [103] J. M. Tomczak, *Flow-Based Models*. Cham: Springer International Publishing, 2022, pp. 27–56. [Online]. Available: https://doi.org/10.1007/978-3-030-93158-2_3
- [104] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using Real NVP," 2016. [Online]. Available: <https://arxiv.org/abs/1605.08803>
- [105] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing Flows: An Introduction and Review of Current Methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3964–3979, Nov 2021. [Online]. Available: <https://doi.org/10.1109%2Ftpami.2020.2992934>
- [106] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, "Neural Spline Flows," 2019. [Online]. Available: <https://arxiv.org/abs/1906.04032>
- [107] D. P. Kingma and P. Dhariwal, "Glow: Generative Flow with Invertible 1x1 Convolutions," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: <https://proceedings.neurips.cc/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf>
- [108] J. Rocca. Understanding Variational Autoencoders (VAEs). [Online]. Available: <https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>
- [109] P. Pandey. (2018) Deep Generative Models. [Online]. Available: <https://towardsdatascience.com/deep-generative-models-25ab2821afd3>
- [110] A. Estiri, M. Sabramooz, A. Banaei, A. Dehghan, B. Jamialahmadi, and M. Jafari Siavoshani, "A Variational Auto-Encoder Approach for Image Transmission in Wireless Channel," 10 2020. [Online]. Available: <https://arxiv.org/abs/2010.03967>

- [111] J. M. Joyce, "Kullback-leibler divergence," in *International encyclopedia of statistical science*. Springer, 2011, pp. 720–722. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-3-642-04898-2_327
- [112] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019. [Online]. Available: <https://doi.org/10.1561%2F22000000056>
- [113] Alqahtani *et al.*, "Applications of Generative Adversarial Networks (GANs): An Updated Review," 12 2019. [Online]. Available: <https://link.springer.com/article/10.1007/s11831-019-09388-y>
- [114] T. Mukhiddin, W. Lee, S. Lee, and T. Rashid, "Research issues on generative adversarial networks and applications," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2020, pp. 487–488. [Online]. Available: <https://ieeexplore.ieee.org/document/9070385>
- [115] M. Paganini, L. de Oliveira, and B. Nachman, "Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters," *Physical Review Letters*, vol. 120, no. 4, Jan 2018. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevLett.120.042003>
- [116] G. R. Khattak, S. Vallecorsa, F. Carminati, and G. M. Khan, "Particle Detector Simulation using Generative Adversarial Networks with Domain Related Constraints," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 28–33. [Online]. Available: <https://ieeexplore.ieee.org/document/8999196>
- [117] M. Faucci Giannelli, "FastCaloGAN: a fast simulation for the ATLAS calorimeter system using GANs," 2020. [Online]. Available: <https://cds.cern.ch/record/2746428>
- [118] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*. Ieee, 2017, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/8308186>
- [119] A. Zhang *et al.* Convolutional Neural Networks. [Online]. Available: https://d2l.ai/chapter_convolutional-neural-networks.html
- [120] R. Hachilif, R. Baghdadi, and F. Benhamida, "Graduation Thesis Implementing and Optimizing Neural Networks using Tiramisu," Ph.D. dissertation, 06 2019. [Online]. Available: https://www.researchgate.net/publication/334974839_Graduation_Thesis_Implementing_and_Optimizing_Neural_Networks_using_Tiramisu

- [121] X. Ding, X. Zhang, Y. Zhou, J. Han, G. Ding, and J. Sun, "Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs," 2022. [Online]. Available: <https://arxiv.org/abs/2203.06717>
- [122] M. Abadi, A. Agarwal, P. Barham, E. Brevdo *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015, software available from [tensorflow.org](https://www.tensorflow.org/). [Online]. Available: <https://www.tensorflow.org/>
- [123] A. Paszke, S. Gross, F. Massa, A. Lerer, and J. a. Bradbury, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," pp. 8024–8035, 2019. [Online]. Available: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- [124] R. Kadam. Difference Between Channels and Kernels in Deep Learning. [Online]. Available: <https://medium.com/analytics-vidhya/difference-between-channels-and-kernels-in-deep-learning-6db818038a11>
- [125] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1. Atlanta, Georgia, USA, 2013, p. 3. [Online]. Available: https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf
- [126] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015. [Online]. Available: <https://dl.acm.org/doi/10.5555/3045118.3045167>
- [127] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: <http://jmlr.org/papers/v15/srivastava14a.html>
- [128] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231222008426>
- [129] K. Fukushima, "Cognitron: A self-organizing multilayered neural network," *Biological cybernetics*, vol. 20, no. 3, pp. 121–136, 1975. [Online]. Available: <https://link.springer.com/article/10.1007/BF00342633>
- [130] Vallecorsa *et al.*, "3Dgan Model," <https://github.com/svalleco/3Dgan/tree/master/keras>, 2020.
- [131] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017. [Online]. Available: <https://arxiv.org/abs/1701.07875>

- [132] E. Y. Zhu, S. Johri, D. Bacon, M. Esencan, J. Kim, M. Muir, N. Murgai, J. Nguyen, N. Piseni, A. Schouela *et al.*, “Generative quantum learning of joint probability distribution functions,” p. 043092, 2022. [Online]. Available: <https://journals.aps.org/prresearch/abstract/10.1103/PhysRevResearch.4.043092>
- [133] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, “Generative adversarial networks: introduction and outlook,” *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017. [Online]. Available: <https://ieeexplore.ieee.org/document/8039016>
- [134] F. Facchinei and C. Kanzow, “Generalized nash equilibrium problems,” *Annals of Operations Research*, vol. 175, no. 1, pp. 177–211, 2010. [Online]. Available: <https://link.springer.com/article/10.1007/s10288-007-0054-4>
- [135] F. Farnia and A. Ozdaglar, “Do gans always have nash equilibria?” PMLR, pp. 3029–3039, 2020. [Online]. Available: <http://proceedings.mlr.press/v119/farnia20a>
- [136] J. D. Curtó, I. C. Zarza, F. de la Torre, I. King, and M. R. Lyu, “High-resolution deep convolutional generative adversarial networks,” 2017. [Online]. Available: <https://arxiv.org/abs/1711.06491>
- [137] X. Ying, “An Overview of Overfitting and its Solutions,” *Journal of Physics: Conference Series*, vol. 1168, p. 022022, 02 2019. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022/meta>
- [138] GAN Anatomy. [Online]. Available: <https://developers.google.com/machine-learning/gan>
- [139] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye, “A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications,” 2020. [Online]. Available: <https://arxiv.org/abs/2001.06937>
- [140] A. Aggarwal, M. Mittal, and G. Battineni, “Generative adversarial network: An overview of theory and applications,” *International Journal of Information Management Data Insights*, vol. 1, no. 1, p. 100004, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2667096820300045>
- [141] L. Weng, “From GAN to WGAN,” 2019. [Online]. Available: <https://arxiv.org/abs/1904.08994>
- [142] I. Goodfellow, “NIPS 2016 Tutorial: Generative Adversarial Networks,” 2017. [Online]. Available: <https://arxiv.org/abs/1701.00160>
- [143] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” pp. 2818–2826, 2016. [Online]. Available: <https://ieeexplore.ieee.org/document/7780677>

- [144] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," 2016. [Online]. Available: <https://arxiv.org/abs/1606.03498>
- [145] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," 2017. [Online]. Available: <https://arxiv.org/abs/1710.10196>
- [146] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," 2017. [Online]. Available: <https://arxiv.org/abs/1704.00028>
- [147] Z. Li, Y. Zhang, and S. Arora, "Why Are Convolutional Nets More Sample-Efficient than Fully-Connected Nets?" in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=uCY5MuAxcxU>
- [148] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2016. [Online]. Available: <https://arxiv.org/abs/1603.07285>
- [149] S. Wieluch and D. F. Schwenker, "Dropout Induced Noise for Co-Creative GAN Systems," 2019. [Online]. Available: <https://arxiv.org/abs/1909.04474>
- [150] G. E. Nasr, E. Badr, and C. Joun, "Cross Entropy Error Function in Neural Networks: Forecasting Gasoline Demand," in *FLAIRS Conference*, 2002. [Online]. Available: <https://aaai.org/Papers/FLAIRS/2002/FLAIRS02-075.pdf>
- [151] P. M. Swamidass, Ed., *MAPE (mean absolute percentage error)*. Boston, MA: Springer US, 2000, pp. 462–462. [Online]. Available: https://doi.org/10.1007/1-4020-0612-8_580
- [152] Rehm, Florian, Vallecorsa, Sofia, Borrás, Kerstin, and Krücker, Dirk, "Physics Validation of Novel Convolutional 2D Architectures for Speeding Up High Energy Physics Simulations," *EPJ Web Conf.*, vol. 251, p. 03042, 2021. [Online]. Available: <https://doi.org/10.1051/epjconf/202125103042>
- [153] F. Rehm, S. Vallecorsa, K. Borrás, and D. Krücker, "Benchmark of generative adversarial networks for fast hep calorimeter simulations," *Proceedings of the 9th International Conference in Distributed Computing and Grid-technologies in Science and Education (GRID'2021)*, 2021. [Online]. Available: <http://ceur-ws.org/Vol-3041/310-315-paper-57.pdf>
- [154] F. Rehm, S. Vallecorsa, , K. Borrás, and D. Krücker, "Validation of Deep Convolutional Generative Adversarial Networks for High Energy Physics Calorimeter Simulations," *AAAI-MLPS 2021*, vol. 2964, 2021. [Online]. Available: https://ceur-ws.org/Vol-2964/article_178.pdf

- [155] G. Khattak, "2D Seperable convolutions for 3DGAN, [CERN openlab Internal Report]," 2020, pp. 28–33.
- [156] F. Gomez-Donoso, A. Garcia-Garcia, J. Rodríguez, S. Orts, and M. Cazorla, "LonchaNet: A sliced-based CNN architecture for real-time 3D object recognition," 05 2017, pp. 412–418. [Online]. Available: <https://ieeexplore.ieee.org/document/7965883>
- [157] K. Sarkar, B. Hampiholi, K. Varanasi, and D. Stricker, "Learning 3D Shapes as Multi-Layered Height-maps using 2D Convolutional Networks," Cham, pp. 74–89, 2018. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-01270-0_5
- [158] T. Quast, "Qualification, performance validation and fast generative modelling of beam test calorimeter prototypes for the CMS calorimeter endcap upgrade," p. 236, 5 2020. [Online]. Available: <https://publications.rwth-aachen.de/record/792901>
- [159] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4799–4807. [Online]. Available: <https://ieeexplore.ieee.org/document/8237776>
- [160] M. Feurer and F. Hutter, *Hyperparameter Optimization*. Cham: Springer International Publishing, 2019, pp. 3–33. [Online]. Available: https://doi.org/10.1007/978-3-030-05318-5_1
- [161] P. I. Frazier, "A Tutorial on Bayesian Optimization," 2018. [Online]. Available: <https://arxiv.org/abs/1807.02811>
- [162] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, "Evolutionary algorithms," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 3, pp. 178–195, 2014. [Online]. Available: <https://dl.acm.org/doi/abs/10.1002/widm.1124>
- [163] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," 2019. [Online]. Available: <https://arxiv.org/abs/1907.10902>
- [164] L. Lu, "Dying ReLU and initialization: Theory and numerical examples," *Communications in Computational Physics*, vol. 28, no. 5, pp. 1671–1706, Jun 2020. [Online]. Available: <https://doi.org/10.4208%2Fcicp.oa-2020-0165>
- [165] V. Sofia and C. Federico, "Distributed Training of Generative Adversarial Networks for Fast Detector Simulation," in *High Performance Computing*. Cham: Springer International Publishing, 2018, pp. 487–503. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-02465-9_35

- [166] H. Chen, Y. Alhinai, Y. Jiang, E. Na, and C. Hao, "Bottleneck analysis of dynamic graph neural network inference on cpu and gpu," 2022. [Online]. Available: <https://arxiv.org/abs/2210.03900>
- [167] Nvidia. (2022) Nvidia developer zone: Gpu management and deployment. [Online]. Available: https://docs.nvidia.com/deploy/nvml-api/structnvmlUtilization_t.html#structnvmlUtilization_t
- [168] A. Borji, "Pros and Cons of GAN Evaluation Measures," 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1077314218304272>
- [169] E. Buhmann, "Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed," 5 2020. [Online]. Available: <https://link.springer.com/article/10.1007/s41781-021-00056-0>
- [170] S. Morgan Jr and P. Eby, "Corrections to the bethe-bloch formula for average ionization energy loss of relativistic heavy nuclei close collisions," *Nuclear Instruments and Methods*, vol. 106, no. 3, pp. 429–435, 1973. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0029554X73903030>
- [171] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004. [Online]. Available: <https://ieeexplore.ieee.org/document/1284395>
- [172] SSIM: Structural Similarity Index. [Online]. Available: <https://www.imatest.com/docs/ssim/>
- [173] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014. [Online]. Available: <https://doi.org/10.7717/peerj.453>
- [174] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, and A. Zaborowska. (2022) Fast Calorimeter Simulation Challenge 2022. [Online]. Available: calochallenge.github.io/homepage/
- [175] IEEE, "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2008*, pp. 1–70, 2008. [Online]. Available: <https://ieeexplore.ieee.org/document/4610935>
- [176] F. Rehm, S. Vallecorsa, V. Saletore, H. Pabst, A. Chaibi, V. Codreanu, K. Borras, and D. Krücker, "Reduced Precision Strategies for Deep Learning: A High Energy Physics Generative Adversarial Network Use Case," *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods*, 2021. [Online]. Available: <http://dx.doi.org/10.5220/0010245002510258>

- [177] F. Rehm, V. Saletore, S. Vallecorsa, K. Borrás, and D. Krücker, "Reduced precision research of a gan image generation use-case," in *Pattern Recognition Applications and Methods*, M. De Marsico, G. Sanniti di Baja, and A. Fred, Eds. Cham: Springer International Publishing, 2023, pp. 3–22. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-24538-1_1
- [178] A. Jain, S. Bhattacharya, M. Masuda, V. Sharma, and Y. Wang, "Efficient Execution of Quantized Deep Learning Models: A Compiler Approach," 06 2020. [Online]. Available: <https://arxiv.org/abs/2006.10226>
- [179] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation," 2020. [Online]. Available: <https://arxiv.org/abs/2004.09602>
- [180] H. Wu, "Inference at Reduced Precision on GPUs," 2019. [Online]. Available: <https://developer.download.nvidia.com/video/gputechconf/gtc/2019/presentation/s9659-inference-at-reduced-precision-on-gpus.pdf>
- [181] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Accurate post training quantization with small calibration sets," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 4466–4475. [Online]. Available: <https://proceedings.mlr.press/v139/hubara21a.html>
- [182] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Post-training 4-bit quantization of convolution networks for rapid-deployment," 2019. [Online]. Available: <https://arxiv.org/abs/1810.05723>
- [183] R. Gupta and V. Ranga, *Comparative Study of Different Reduced Precision Techniques in Deep Neural Network*, 01 2021, pp. 123–136. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-15-8377-3_11
- [184] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training Deep Neural Networks with 8-bit Floating Point Numbers," 2018. [Online]. Available: <https://arxiv.org/abs/1812.08011>
- [185] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed Precision Training," 2017. [Online]. Available: <https://arxiv.org/abs/1710.03740>
- [186] S. R. Nandakumar, M. Le Gallo, C. Piveteau, V. Joshi, G. Mariani, I. Boybat, G. Karunaratne, R. Khaddam-Aljameh, U. Egger, A. Petropoulos, and et al., "Mixed-Precision Deep Learning Based on Computational Memory," *Frontiers in Neuroscience*, vol. 14, May 2020. [Online]. Available: <https://dx.doi.org/10.3389/fnins.2020.00406>

- [187] J. Osorio Ríos, A. Armejach, G. Khattak, E. Petit, S. Vallecorsa, and M. Casas, "Evaluating mixed-precision arithmetic for 3d generative adversarial networks to simulate high energy physics detectors," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2020, pp. 49–56. [Online]. Available: <https://ieeexplore.ieee.org/document/9356207>
- [188] Intel, "oneAPI Deep Neural Network Library (oneDNN)." [Online]. Available: <https://github.com/oneapi-src/oneDNN>
- [189] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," pp. 770–778, 2016. [Online]. Available: <https://ieeexplore.ieee.org/document/7780459>
- [190] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," p. 4278–4284, 2017. [Online]. Available: <https://dl.acm.org/doi/10.5555/3298023.3298188>
- [191] J. G. Feng Tian, Haihao Shen and H. Abidi, "Intel® LPOT Key Takeaways," 2021. [Online]. Available: <https://www.intel.com/content/www/us/en/artificial-intelligence/posts/intel-low-precision-optimization-tool.html>
- [192] S. Li, L. D. Xu, and S. Zhao, "The internet of things: a survey," *Information systems frontiers*, vol. 17, pp. 243–259, 2015.
- [193] S. Vallecorsa, F. Carminati, and G. Khattak, "3D convolutional GAN for fast simulation," *EPJ Web of Conferences*, vol. 214, p. 02010, 01 2019. [Online]. Available: <https://doi.org/10.1051/epjconf/201921402010>
- [194] B. Tuomanen, *Hands-On GPU Programming with Python and CUDA: Explore high-performance parallel computing with CUDA*. Packt Publishing Ltd, 11 2018. [Online]. Available: <https://subscription.packtpub.com/search?query=hands%20gpu%20programming%20python%20and%20cuda>
- [195] *Post-training quantization*, TensorFlow Documentation, Jul. 2020. [Online]. Available: https://translate.google.com/translate?hl=de&sl=en&tl=de&u=https%3A%2F%2Fwww.tensorflow.org%2Felite%2Fperformance%2Fpost_training_quantization&anno=2&prev=search
- [196] PyTorch, "Introduction to Quantization on PyTorch," 2020. [Online]. Available: <https://pytorch.org/blog/introduction-to-quantization-on-pytorch/>
- [197] J. P. Dowling and G. J. Milburn, "Quantum technology: the second quantum revolution," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 361, no. 1809, pp. 1655–1674, 2003.
- [198] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information*, vol. 2, no. 1, Jan 2016. [Online]. Available: <https://doi.org/10.1038%2Fnpjqi.2015.23>

- [199] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, no. 6, pp. 595–600, Apr 2018. [Online]. Available: <https://doi.org/10.1038%2Fs41567-018-0124-x>
- [200] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134. [Online]. Available: <https://ieeexplore.ieee.org/document/365700>
- [201] N. Elsayed, A. S. Maida, and M. Bayoumi, "A Review of Quantum Computer Energy Efficiency," in *2019 IEEE Green Technologies Conference(GreenTech)*, 2019, pp. 1–3. [Online]. Available: <https://ieeexplore.ieee.org/document/8767125>
- [202] J. Ikonen, J. Salmilehto, and M. Möttönen, "Energy-efficient quantum computing," *npj Quantum Information*, vol. 3, no. 1, p. 17, 2017. [Online]. Available: <https://ui.adsabs.harvard.edu/abs/2017npjQI...3...17I/abstract>
- [203] DESY. Partnership on quantum computing: DESY joins the IBM Quantum Network. Accessed in January 2023. [Online]. Available: https://www.desy.de/news/news_search/index_eng.html?openDirectAnchor=2326
- [204] R. S. Sutor, *Dancing with Qubits: How quantum computing works and how it can change the world*. Packt Publishing Ltd, 2019. [Online]. Available: <https://www.packtpub.com/product/dancing-with-qubits/9781838827366>
- [205] T. Forcer, T. Hey, D. Ross, and P. Smith, "Superposition, Entanglement and Quantum Computation," *Quantum Information & Computation*, vol. 2, 03 2002. [Online]. Available: <https://core.ac.uk/download/pdf/1502592.pdf>
- [206] D. C. Marinescu and G. M. Marinescu, "Chapter 2 - measurements and quantum information," in *Classical and Quantum Information*, D. C. Marinescu and G. M. Marinescu, Eds. Boston: Academic Press, 2012, pp. 133–220. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780123838742000023>
- [207] P. M. Lokenath Debnath, *Introduction to Hilbert Spaces with Applications*, 10 2005. [Online]. Available: <https://www.elsevier.com/books/introduction-to-hilbert-spaces-with-applications/debnath/978-0-08-045592-1>
- [208] F. T. Arecchi, E. Courtens, R. Gilmore, and H. Thomas, "Atomic coherent states in quantum optics," *Phys. Rev. A*, vol. 6, pp. 2211–2237, Dec 1972. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.6.2211>
- [209] L. M. Narducci, C. A. Coulter, and C. M. Bowden, "Exact diffusion equation for a model for superradiant emission," *Phys. Rev. A*, vol. 9, pp. 829–845, Feb 1974. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.9.829>

- [210] M. Weigold, J. Barzen, F. Leymann, and M. Salm, "Expanding Data Encoding Patterns For Quantum Algorithms," in *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, 2021, pp. 95–101. [Online]. Available: <https://ieeexplore.ieee.org/document/9425837>
- [211] M. Soeken, D. M. Miller, and R. Drechsler, "Quantum circuits employing roots of the pauli matrices," *Physical Review A*, vol. 88, no. 4, p. 042322, 2013.
- [212] H. Weinfurter, "Experimental bell-state analysis," *Europhysics Letters*, vol. 25, no. 8, p. 559, 1994.
- [213] S. Johnstun and J.-F. V. Huele, "Understanding and compensating for noise on ibm quantum computers," *American Journal of Physics*, 2021. [Online]. Available: <https://doi.org/10.1119/10.0006204>
- [214] M. Schuld and F. Petruccione, *Supervised learning with quantum computers*. Springer, 2018, vol. 17.
- [215] "How do quantum computers work?" available at: <https://www.ibm.com/topics/quantum-computing> Accessed in October 2022. [Online]. Available: <https://www.ibm.com/topics/quantum-computing>
- [216] W. F. Vinen, "The physics of superfluid helium," 2004. [Online]. Available: <https://cds.cern.ch/record/808382>
- [217] T. Tai, J. Cai, and S. M. Anlage, "Coherence Time Enhancement of Interacting Two-Level Systems in Aluminum Superconducting Resonators," 2021. [Online]. Available: <https://arxiv.org/abs/2109.11742>
- [218] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," *Physical Review A*, vol. 100, no. 3, Sep 2019. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevA.100.032328>
- [219] Qiskit. (2022) Device backend noise model simulations. [Online]. Available: https://qiskit.org/documentation/tutorials/simulators/2.device_noise_simulation.html
- [220] "Expanding the IBM Quantum roadmap to anticipate the future of quantum-centric supercomputing," available at: <https://research.ibm.com/blog/ibm-quantum-roadmap-2025> Accessed in October 2022. [Online]. Available: <https://research.ibm.com/blog/ibm-quantum-roadmap-2025>
- [221] P. Braccia, F. Caruso, and L. Banchi, "How to enhance quantum generative adversarial learning of noisy information," *New Journal of Physics*, vol. 23, no. 5, p. 053024, May 2021. [Online]. Available: <https://doi.org/10.1088%2F1367-2630%2F2105053024>

- [222] T. Patel, A. Potharaju, B. Li, R. B. Roy, and D. Tiwari, "Experimental evaluation of nisq quantum computers: Error measurement, characterization, and implications," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020, pp. 1–15.
- [223] J. Roffe, "Quantum error correction: an introductory guide," *Contemporary Physics*, vol. 60, no. 3, pp. 226–245, Jul 2019. [Online]. Available: <https://doi.org/10.1080%2F00107514.2019.1667078>
- [224] C. Piveteau, D. Sutter, and S. Woerner, "Quasiprobability decompositions with reduced sampling overhead," *npj Quantum Information*, vol. 8, no. 1, Feb 2022. [Online]. Available: <https://doi.org/10.1038%2Fs41534-022-00517-3>
- [225] E. Knill, "Quantum computing with realistically noisy devices," *Nature*, vol. 434, pp. 39–44, 04 2005. [Online]. Available: <https://www.nature.com/articles/nature03350>
- [226] K. Temme, S. Bravyi, and J. M. Gambetta, "Error Mitigation for Short-Depth Quantum Circuits," *Physical Review Letters*, vol. 119, no. 18, Nov 2017. [Online]. Available: <https://doi.org/10.1103%2Fphysrevlett.119.180509>
- [227] "Qiskit documentation and source code," accessed: 09-02-2022. [Online]. Available: <https://qiskit.org/textbook/ch-quantum-hardware/measurement-error-mitigation.html>
- [228] "IBM Media Center Gallery," available at: <https://newsroom.ibm.com/media-quantum-innovation?keywords=quantum&l=100> Accessed in October 2022. [Online]. Available: <https://newsroom.ibm.com/media-quantum-innovation?keywords=quantum&l=100>
- [229] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, no. 2, p. 021314, Jun 2019. [Online]. Available: <https://doi.org/10.1063%2F1.5088164>
- [230] H. Zhang, A. Wu, Y. Wang, G. Li, H. Shapourian, A. Shabani, and Y. Ding, "A Compilation Framework for Photonic One-Way Quantum Computation," 2022. [Online]. Available: <https://arxiv.org/abs/2209.01545>
- [231] F. Arute, K. Arya *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct 2019. [Online]. Available: <https://doi.org/10.1038%2Fs41586-019-1666-5>
- [232] G. G. Guerreschi, J. Hogaboam, F. Baruffa, and N. P. D. Sawaya, "Intel Quantum Simulator: a cloud-ready high-performance simulator of quantum circuits," *Quantum Science and Technology*, vol. 5, no. 3, p. 034007, May 2020. [Online]. Available: <https://doi.org/10.1088%2F2058-9565%2Fab8505>

- [233] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.
- [234] M. Wang, R. Huang, S. Tannu, and P. Nair, "TQSim: A Case for Reuse-Focused Tree-Based Quantum Circuit Simulation," 2022. [Online]. Available: <https://arxiv.org/abs/2203.13892>
- [235] Z.-M. He, L. Li, S. Zheng, Z. Huang, and H. Situ, "A Conditional Generative Model Based on Quantum Circuit and Classical Optimization," *International Journal of Theoretical Physics*, vol. 58, pp. 1138–1149, 04 2019. [Online]. Available: <https://link.springer.com/article/10.1007/s10773-019-04005-x>
- [236] F. Phillipson, "Quantum Machine Learning: Benefits and Practical Examples." in *QANSWER*, 2020, pp. 51–56. [Online]. Available: <https://ceur-ws.org/Vol-2561/paper5.pdf>
- [237] H.-L. Huang and Y. D. et al., "Experimental Quantum Generative Adversarial Networks for Image Generation," *Physical Review Applied*, vol. 16, no. 2, Aug 2021. [Online]. Available: <https://doi.org/10.1103/PhysRevApplied.16.024051>
- [238] L. Buffoni and F. Caruso, "New trends in quantum machine learning," *Europhysics Letters*, vol. 132, no. 6, p. 60004, Dec 2020. [Online]. Available: <https://doi.org/10.1209/2F0295-5075%2F132%2F60004>
- [239] G. An, "The Effects of Adding Noise During Backpropagation Training on a Generalization Performance," *Neural Computation*, vol. 8, no. 3, pp. 643–674, 1996. [Online]. Available: <https://dl.acm.org/doi/10.1162/neco.1996.8.3.643>
- [240] R. Reed and R. Marks, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, 1999. [Online]. Available: <https://books.google.fr/books?id=cnbgoQEACAAJ>
- [241] C. M. Bishop, "Training with Noise is Equivalent to Tikhonov Regularization," *Neural Computation*, vol. 7, no. 1, pp. 108–116, 01 1995. [Online]. Available: <https://doi.org/10.1162/neco.1995.7.1.108>
- [242] B. Coyle, "Machine learning applications for noisy intermediate-scale quantum computers," 2022. [Online]. Available: <https://arxiv.org/abs/2205.09414>
- [243] H. Wang, Y. Ding *et al.*, "QuantumNAS: Noise-Adaptive Search for Robust Quantum Circuits," 2021. [Online]. Available: <https://arxiv.org/abs/2107.10845>
- [244] K. Borrás, S. Y. Chang, L. Funcke, M. Grossi, T. Hartung, K. Jansen, D. Kruecker, S. Kühn, F. Rehm, C. Tüysüz, and S. Vallecorsa, "Impact of quantum noise on the training of quantum generative adversarial networks," *Journal of Physics*:

- Conference Series*, vol. 2438, no. 1, p. 012093, Feb 2023. [Online]. Available: <https://dx.doi.org/10.1088/1742-6596/2438/1/012093>
- [245] H. Wang, J. Gu, Y. Ding, Z. Li, F. Chong, D. Z. Pan, and S. Han, "RoQNN: Noise-aware training for robust quantum neural networks," 2022. [Online]. Available: <https://openreview.net/forum?id=wwIBobGFj2V>
- [246] S. Gupta and R. Zia, "Quantum neural networks," *Journal of Computer and System Sciences*, vol. 63, no. 3, pp. 355–383, 2001.
- [247] M. Weigold, J. Barzen, F. Leymann, and M. Salm, "Data Encoding Patterns for Quantum Computing," in *Proceedings of the 27th Conference on Pattern Languages of Programs*, ser. PLoP '20. USA: The Hillside Group, 2020. [Online]. Available: <https://dl.acm.org/doi/10.5555/3511065.3511068>
- [248] S. Lloyd and C. Weedbrook, "Quantum Generative Adversarial Learning," *Physical Review Letters*, vol. 121, no. 4, Jul 2018. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevLett.121.040502>
- [249] J.-G. Liu and L. Wang, "Differentiable learning of quantum circuit born machines," *Physical Review A*, vol. 98, no. 6, Dec 2018. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevA.98.062324>
- [250] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.
- [251] D. Bokhan, A. S. Mastiukova, A. S. Boev, D. N. Trubnikov, and A. K. Fedorov, "Multiclass classification using quantum convolutional neural networks with hybrid quantum-classical learning," *Frontiers in Physics*, vol. 10, Nov 2022. [Online]. Available: <https://doi.org/10.3389/fphy.2022.1069985>
- [252] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, Sep 2017. [Online]. Available: <https://doi.org/10.1038/nature23879>
- [253] M. Vogel, "Quantum Computation and Quantum Information, by M.A. Nielsen and I.L. Chuang," *Contemporary Physics - CONTEMP PHYS*, vol. 52, pp. 604–605, 11 2011. [Online]. Available: <https://www.cambridge.org/highereducation/books/quantum-computation-and-quantum-information/01E10196D0A682A6AEFFEA52D53BE9AE#overview>
- [254] Qiskit, "qGANs for Loading Random Distributions," accessed in March 2021. [Online]. Available: https://qiskit.org/documentation/machine-learning/tutorials/04.qgans_for_loading_random_distributions.html

- [255] F. Rehm, S. Vallecorsa, K. Borras, and D. Krücker, "Quantum machine learning for HEP detector simulations," 2021. [Online]. Available: <http://ceur-ws.org/Vol-3041/363-368-paper-67.pdf>
- [256] S. Ramos-Calderer, A. Pérez-Salinas, D. García-Martín, C. Bravo-Prieto, J. Cortada, J. Planagumà, and J. Latorre, "Quantum unary approach to option pricing," 12 2019. [Online]. Available: <https://journals.aps.org/pr/abstract/10.1103/PhysRevA.103.032414>
- [257] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, 12 2014.
- [258] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, "Evaluating analytic gradients on quantum hardware," *Physical Review A*, vol. 99, no. 3, Mar 2019. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevA.99.032331>
- [259] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*. Elsevier, 1992, pp. 65–93.
- [260] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, "General parameter-shift rules for quantum gradients," 2021. [Online]. Available: <https://arxiv.org/abs/2107.12390>
- [261] L. Funcke, T. Hartung, K. Jansen, S. Kühn, and P. Stornati, "Dimensional Expressivity Analysis of Parametric Quantum Circuits," *Quantum*, vol. 5, p. 422, Mar 2021. [Online]. Available: <http://dx.doi.org/10.22331/q-2021-03-29-422>
- [262] A. tA v, M. S. ANIS, Abby-Mitchell *et al.*, "Qiskit: An open-source framework for quantum computing," 2021.
- [263] Y. I. Bogdanov, N. A. Bogdanova, D. V. Fastovets, and V. F. Lukichev, "Study of the effect of quantum noise on the accuracy of the Schrodinger equation simulation on a quantum computer using the Zalka-Wiesner method," 2022. [Online]. Available: <https://arxiv.org/abs/2201.03068>
- [264] C. Alexandrou, L. Funcke, T. Hartung, K. Jansen, S. Kühn, G. Polykratis, P. Stornati, and X. Wang, "Using classical bit-flip correction for error mitigation including 2-qubit correlations," 2021, arXiv:2111.08551.
- [265] G. S. Ravi, K. N. Smith, P. Gokhale, and F. T. Chong, "Quantum Computing in the Cloud: Analyzing job and machine characteristics," 2022. [Online]. Available: <https://arxiv.org/abs/2203.13121>
- [266] E. Grant, M. Benedetti, S. Cao *et al.*, "Hierarchical quantum classifiers," *npj Quantum Information*, vol. 4, no. 1, Dec 2018. [Online]. Available: <https://doi.org/10.1038/s41534-018-0116-9>

- [267] K. Sharma, S. Khatri, M. Cerezo, and P. J. Coles, "Noise resilience of variational quantum compiling," *New Journal of Physics*, vol. 22, no. 4, p. 043006, Apr 2020. [Online]. Available: <https://doi.org/10.1088%2F1367-2630%2Fab784c>
- [268] F. Rehm, S. Vallecorsa, M. Grossi, K. Borrás, D. Krücker, S. Schnake, and A.-H. Verney-Provatas, "Quantum angle generator for image generation," in *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*. IEEE Computer Society, 2022, pp. 425–429. [Online]. Available: <https://inspirehep.net/literature/2620987>
- [269] F. Rehm, S. Vallecorsa, K. Borrás, M. Grossi, D. Krücker, and V. Varo, "Precise Image Generation on Current Noisy Quantum Devices," 2023. [Online]. Available: <https://arxiv.org/abs/2307.05253>
- [270] S. Y. Chang, S. Herbert, S. Vallecorsa, E. F. Combarro, and R. Duncan, "Dual-parameterized quantum circuit GAN model in high energy physics," *EPJ Web of Conferences*, vol. 251, p. 03050, 2021. [Online]. Available: <https://doi.org/10.1051%2Fepjconf%2F202125103050>
- [271] A. Khoshaman, W. Vinci, B. Denis, E. Andriyash, H. Sadeghi, and M. H. Amin, "Quantum variational autoencoder," *Quantum Science and Technology*, vol. 4, no. 1, p. 014001, Sep 2018. [Online]. Available: <https://doi.org/10.1088%2F2058-9565%2Faada1f>
- [272] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, "Quantum Boltzmann Machine," *Physical Review X*, vol. 8, no. 2, May 2018. [Online]. Available: <https://doi.org/10.1103%2Fphysrevx.8.021050>
- [273] J. Tian, X. Sun, Y. Du, S. Zhao, Q. Liu, K. Zhang, W. Yi, W. Huang, C. Wang, X. Wu, M.-H. Hsieh, T. Liu, W. Yang, and D. Tao, "Recent Advances for Quantum Neural Networks in Generative Learning," 2022. [Online]. Available: <https://arxiv.org/abs/2206.03066>
- [274] O. Kiss, M. Grossi, E. Kajomovitz, and S. Vallecorsa, "Conditional born machine for monte carlo event generation," *Physical Review A*, vol. 106, no. 2, Aug 2022. [Online]. Available: <https://doi.org/10.1103%2Fphysreva.106.022612>
- [275] A. Carrera Vazquez and S. Woerner, "Efficient State Preparation for Quantum Amplitude Estimation," *Phys. Rev. Applied*, vol. 15, p. 034027, Mar 2021. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevApplied.15.034027>
- [276] I. Araujo, D. Park, F. Petruccione, and A. da Silva, "A divide-and-conquer algorithm for quantum state preparation," *Scientific Reports*, vol. 11, 03 2021. [Online]. Available: <https://www.nature.com/articles/s41598-021-85474-1>
- [277] R. LaRose and B. Coyle, "Robust data encodings for quantum classifiers," *Physical Review A*, vol. 102, no. 3, Sep 2020. [Online]. Available: <https://doi.org/10.1103%2Fphysreva.102.032420>

- [278] D. Sierra-Sosa, M. Telahun, and A. Elmaghraby, "TensorFlow Quantum: Impacts of Quantum State Preparation on Quantum Machine Learning Performance," *IEEE Access*, vol. 8, pp. 215 246–215 255, 01 2020.
- [279] F. Rehm, "Downsampled Calorimeter Shower Images to 8 Pixels," 2021. [Online]. Available: <https://doi.org/10.5281/zenodo.7025233>
- [280] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.
- [281] A. Berlinet and C. Thomas-Agnan, *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011. [Online]. Available: <https://link.springer.com/book/10.1007/978-1-4419-9096-9>
- [282] B. Schölkopf, "The Kernel Trick for Distances," *Advances in neural information processing systems*, vol. 13, 2000. [Online]. Available: <https://proceedings.neurips.cc/paper/2000/file/4e87337f366f72daa424dae11df0538c-Paper.pdf>
- [283] A. Gretton, K. Borgwardt, M. J. Rasch, B. Scholkopf, and A. J. Smola, "A Kernel Method for the Two-Sample Problem," 2008. [Online]. Available: <https://arxiv.org/abs/0805.2368>
- [284] Y. Li, K. Swersky, and R. Zemel, "Generative Moment Matching Networks," 2015. [Online]. Available: <https://arxiv.org/abs/1502.02761>
- [285] M.-T. Puth, M. Neuhäuser, and G. D. Ruxton, "Effective use of pearson's product-moment correlation coefficient," *Animal behaviour*, vol. 93, pp. 183–189, 2014.
- [286] NumPy. Pearson Product-Moment Correlation Coefficients. Accessed in January 2023. [Online]. Available: <https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html>
- [287] C. R. Harris, K. J. Millman, S. J. van der Walt *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: <https://doi.org/10.1038/s41586-020-2649-2>
- [288] A. H. Alhabsi, "Improved SPSA optimization algorithm requiring a single measurement per iteration," in *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)*, 2010, pp. 263–265.
- [289] IBMQ, "SPSA Optimizer," 2022, available at: <https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.SPSA.html> Accessed in April 2022.

- [290] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, Oct 2019. [Online]. Available: <https://doi.org/10.1002%2Fqute.201900070>
- [291] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.
- [292] J. A. Hartigan, M. A. Wong *et al.*, "A k-means clustering algorithm," *Applied statistics*, vol. 28, no. 1, pp. 100–108, 1979. [Online]. Available: <https://www.jstor.org/stable/2346830?origin=JSTOR-pdf>
- [293] M. Livan and R. Wigmans, *The Energy Resolution of Calorimeters*. Cham: Springer International Publishing, 2019, pp. 111–140. [Online]. Available: https://doi.org/10.1007/978-3-030-23653-3_6
- [294] "Our new 2022 Development Roadmap," available at: <https://www.ibm.com/quantum/roadmap> Accessed in December 2022. [Online]. Available: <https://www.ibm.com/quantum/roadmap>
- [295] F. Rehm, S. Vallecorsa, M. Grossi, K. Borras, D. Krücker, S. Schnake, and A.-H. Verney-Provas, "Full Quantum GAN Model for HEP Detector Simulations," 2023: To be published.
- [296] S. Czolbe, O. Krause, I. Cox, and C. Igel, "A Loss Function for Generative Neural Networks Based on Watson's Perceptual Model," 2020. [Online]. Available: <https://arxiv.org/abs/2006.15057>
- [297] W. Zhao, S. Alwidian, and Q. Mahmoud, "Adversarial Training Methods for Deep Learning: A Systematic Review," *Algorithms*, vol. 15, p. 283, 08 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0167739X22004332>
- [298] Z. Deng, L. Zhang, K. Vodrahalli, K. Kawaguchi, and J. Zou, "Adversarial Training Helps Transfer Learning via Better Representations," 2021. [Online]. Available: <https://arxiv.org/abs/2106.10189>
- [299] L. Hu, S.-H. Wu *et al.*, "Quantum generative adversarial learning in a superconducting quantum circuit," *Science advances*, vol. 5, 2019.
- [300] P.-L. Dallaire-Demers *et al.*, "Quantum generative adversarial networks," *Physical Review A*, vol. 98, 2018. [Online]. Available: <https://doi.org/10.1103%2Fphysreva.98.012324>
- [301] R. Zhao and S. Wang, "A review of Quantum Neural Networks: Methods, Models, Dilemma," 2021. [Online]. Available: <https://arxiv.org/abs/2109.01840>

-
- [302] C. Zhao and X.-S. Gao, "QDNN: DNN with Quantum Neural Network Layers," 2019. [Online]. Available: <https://arxiv.org/abs/1912.12660>
- [303] K. G. Jöreskog, "Structural analysis of covariance and correlation matrices," *Psychometrika*, vol. 43, no. 4, pp. 443–477, 1978. [Online]. Available: <https://link.springer.com/article/10.1007/BF02293808>
- [304] A. R. Wildt, A. A. Wildt, O. Ahtola, and O. T. Ahtola, *Analysis of covariance*. Sage, 1978, vol. 12.
- [305] NumPy. Numpy Covariance. Accessed in January 2023. [Online]. Available: <https://numpy.org/doc/stable/reference/generated/numpy.cov.html>

Abbreviations

- **1D/2D/3D** 1/2/3-dimensional
- **AC-GAN** Auxiliary Classifier Generative Adversarial Network
- **AE** Autoencoder
- **AI** Artificial Intelligence
- **CMS** Compact Muon Solenoid
- **CNN** Convolutional Neural Network
- **CPU** Central Processing Unit
- **CX** controlled X-gate
- **CY** controlled Y-gate
- **CZ** controlled Z-gate
- **Conv2D/3D** Convolutional 2D/3D layer
- **Corr** Correlation (loss)
- **DL** Deep Learning
- **GAN** Generative Adversarial Network
- **GPU** Graphical Processing Unit
- **Geant4** Geometry and Tracking Version 4
- **HL** High-Luminosity
- **IT** Internet Technology
- **LHC** Large Hadron Collider
- **LPOT** Low Precision Optimization Tool

- **LS** Long Shutdown (related to the LHC)
- **MAPE** Mean Absolute Percentage Error
- **MERA** Multi-scale Entanglement Renormalization Ansatz
- **ML** Machine Learning
- **MMD** Mean Maximum Discrepancy (loss)
- **MSE** Mean Squared Error
- **NC** Neural Compressor
- **NISQ** Noisy Intermediate-Scale Quantum
- **QAG** Quantum Angle Generator
- **QCBM** Quantum Circuit Born Machine
- **RY** Y-rotational gate
- **RZ** Z-rotational gate
- **ReLU** Rectifier Linear Unit Activation Function
- **SSIM** Structural Similarity Index
- **STD** Standard Deviation
- **TTN** Tree Tensor Networks
- **VAE** Variational Autoencoder
- **WGAN** Wasserstein Generative Adversarial Network
- **a.u.** arbitrary unit
- **cGAN** conditional Generative Adversarial Network
- **oneDNNL** oneAPI Deep Neural Network Library
- **qGAN** quantum Generative Adversarial Network