Improving Fast Inference Jet Detection Neural Networks in the CMS Trigger System

Author: Mohammad Abdollah Chalaki
Supervisor: Adrian Alan Pol
CERN Summer Student Programme Report - September 2021

Abstract
The Large Hadron Collider (LHC) is already one of the largest sources of data in the world. By redevelopments before the end of 2027, its upgrade named High-Luminosity Large Hadron Collider (HL-LHC) will gather data by a factor of ten beyond LHC's initial specifications. The objective is to allow observation of rare processes when increasing luminosity. Therefore, some enhancements must be made to current algorithms to satisfy further extremely low-latency requirements. The main aim of this project, carried out through a total of 8 weeks, is to contribute to the efforts for designing fast inference neural networks to be implemented in Compact Muon Solenoid's (CMS) trigger system. For this purpose, Residual Neural Network (ResNet) architectures were exploited for jet detection. Furthermore, MorphNet like schemes were developed to automate the design of Single Shot MultiBox Detector (SSD) neural networks with specific resource constraints.

Keywords
Neural Networks; Jet Detection; SSD; MorphNet; ResNet

1 Introduction

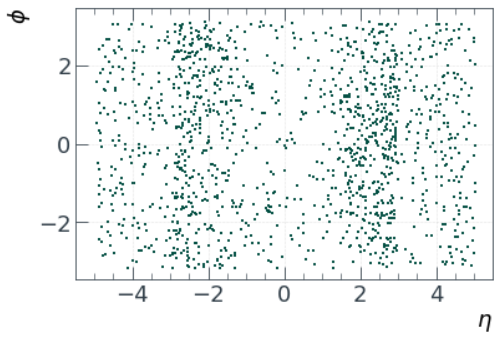
The Large Hadron Collider is the world's largest and most powerful particle accelerator. Within its twenty-seven-kilometers ring, particles reaching near light speed complete thousands of turns every second. Then, more than a billion collisions per second between these particles are detected by LHC's four experiments of ATLAS, ALICE, CMS, and LHCb. Operating at the rate of $40\,\mathrm{MHz}$, each detector is used to collect data for further studies. Without appropriate filtering, there would be petabytes of physics data for the detectors to scrimmage. Not only providing storage facilities is insurmountable for data collection with a 2^{50} order of magnitude, but also, despite tens of thousands of processors at the CERN computing centre, direct processing of data at this rate is impossible. Therefore, exploitation of efficient trigger systems is obligatory in such systems. Triggers are filters with a hierarchical set of algorithms implemented to decrease data rates to a more manageable level by only selecting interesting events. At first, CMS [1] operates a Level 1 (L1) trigger [2] to reduce the $40\,\mathrm{MHz}$ input to a $100\,\mathrm{kHz}$ output on FPGAs and ASICs with approximately $3.2\,\mu\mathrm{s}$ to decide. Then the High Level Trigger (HLT) scales the $100\,\mathrm{kHz}$ stream from the L1 trigger rate down to $1\,\mathrm{kHz}$ [3].

The challange is that the LHC is pushing for higher energy of collisions to explore a rather extensive range of physics. HL-LHC is expected to gather ten times more data compared to its former design [4]. This means that further complex algorithms must be developed to target new constraints for the event selection process. Engineers and scientists at CERN are deploying machine learning algorithms to satisfy these strict low-latency requirements of HL-LHC. Also, some work has already been conducted for designing suitable fast inference neural networks to be implemented in the CMS trigger infrastructure [5, 6]. Although, there is still so much more to be achieved.

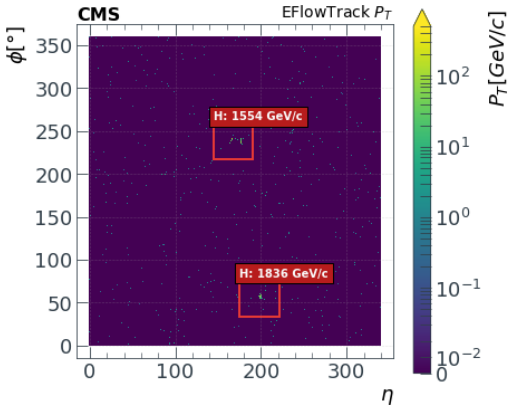
Our aim through this project is to contribute to the enhancement of jet detection in CMS experiment trigger systems. A jet is a collimated spray of particles used to probe the underlying elementary

particle that initiates the cascade of particles [7]. It is detected by the use of tracker data and calorimeter energy deposits. There are two types of calorimeters exploited at CMS: electromagnetic (ECAL) and hadron (HCAL) detectors. Prior is responsible for measuring the energy of electrons and photons, while the latter measures the energy of long-lived charged and neutral hadrons [8]. The calorimeters have a finite segmentation in pseudo-rapidity η and azimuthal angle ϕ, with the coordinate of the resulting energy deposit, tower, computed as the geometrical centre of the cells. An example of calorimeter outputs is depicted in Fig. 1 (left). To filter the desired events by exploitation of image processing algorithms, an image must be extracted from such data. This process consists of mapping defined grid edges of the continuous variables ϕ and η to pixel coordinates. Fig. 1 (right) illustrates an example of a 340×360 pixel image comprised in the training dataset made this way. In our study, we will use a Pythia [9] generated calorimeter dataset [10], with the central region of the CMS events $-3 < \eta < +3$ along using the Delphes library for emulated effects. The goal of Delphes is to allow the simulation of a multipurpose detector for phenomenological studies. The simulation includes a tracking system embedded into a magnetic field, calorimeters, and a muon system [11].

With generated data, identification of different kinds of jets begins. In dataset, the ground truth labels are provided for the jets above threshold momentum ($30\,\mathrm{GeV}/c$ for b and $200\,\mathrm{GeV}/c$ for the jets from boosted heavy particles) using a simple cone algorithm, i.e. associating together particles whose trajectories lie within a circle of radius $R = 0.4$ from the jet centre. In our investigation for efficient tagging of the Higgs boson (H), W boson (W), Z boson (Z), or top quark (t) jets, we will exploit image processing algorithms. With this aim and to provide fast inference solutions for jet detection, first, Jet-SSD open source project [12] is implemented. In which, Single Shot MultiBox Detectors [13] are used for localization of jets and their classification. Further, Residual Neural Networks [14] were exploited to investigate their performance on the dataset in comparison to the previously implemented MobileNet architecture [15]. Finally, the automated design of SSDs is studied with a hardware-oriented approach. The work is influenced by MorphNet [16], a resource-constrained learning structure for the fast and simple design of neural networks. But, since our optimization is based on calculating the cost for SSD architectures rather than the conventional neural networks designed for image classification, some fundamental changes are mandatory to be made to the MorphNet method. By the end of these modifications, approaches targeting Floating Point Operations (FLOPs) or model size are adopted with the main goal to achieve lower latency while also seeking compressed models with preserved or higher accuracies.



Placement of ECAL towers with non-zero readouts



Energy deposits translated to a two-dimensional image

Fig. 1: Example events regarding (left) position of CMS ECAL towers with non-zero readouts and (right) translated two-dimensional neural network input image with the red bounding boxes correspond to ground truth with target label and momentum.

2 The Project

This section covers how this project had been carried out by going through three main phases. First, the implementation of the open-source Jet-SSD repository is discussed, along with providing some information on the SSD architecture. Second, a brief survey is given on the residual neural networks while explaining the extension of the ResNet-18, 34, and 50 to the project. Then, following some solutions that were deployed to ameliorate their performance, we will assess how ResNet performed on the dataset compared to the MobileNet. Finally, the MorphNet is studied for a better understanding, besides mentioning its partial implementation and results.

2.1 Jet-SSD Implementation

Jet-SSD is an open access Git repository [12] that contains components of the Jet Single Shot Detection article [17]. In particular, its objective is to use an SSD network to perform simultaneous localization and classification, plus additional regression tasks to measure jet features. The exploited SSD architecture is a single-stage method that employs convolutional feature layers to the end of truncated base networks for producing a fixed set of detection predictions [13]. An approach different from previous state-of-the-art object detectors using: hypothesize bounding boxes, resample pixels or features for each box, and applying a high-quality classifier. Although these approaches offer accurate results, they are too slow to tackle real-time applications due to their intensive computations, even for embedded systems with high-end hardware. However, SSD with comparable accuracy to such slower techniques is also significantly faster than its rivals like YOLO [18]. Combined, any better candidate could not be found than the SSD to target fast inference requirements while also promising high performance with smaller input image size generated at the detectors. Hence, this particular architecture was implemented with the MobileNet [15] as the backbone model for training in early work. Even though the wise choices in the Jet-SSD do not stop here, we will postpone the discussion of some further important features to the future sections.

Initially in our work, to implement and reproduce the earlier results, the entire Jet-SSD repository was cloned using Linux Public Login User Service (LXPLUS) to access Virtual Machines (VMs) at CERN. Then, a Conda environment was built for the installation of Python package dependencies. Finally, the whole code was executed on GPUs. Crucial steps to take before proceeding to the next section of this summer project.

2.2 Extension of ResNet to the Jet-SSD

As mentioned before, the Jet-SSD only supported MobileNetV1 as its backbone model for training at first. Concerning the crucial importance of the network depth, exploitation of deeper architectures seemed worthy of a try to us, not only for the evaluation of such structures on our dataset but also to explore a baseline to the work. For choosing a model from a manifold of leading result architectures, we came up with the ResNet due to its substantial gain in accuracy and easier optimization. ResNet is a design that explicitly reformulates the layers as learning residual functions regarding the layer inputs instead of learning unreferenced functions. This aims to address the degradation issue in deeper networks, which is offered to rapid degrades followed by saturated accuracy in the converging process of these structures. A phenomenon that was also witnessed in the training trend of the MobileNetV1 which unexpectedly, does not stem from overfitting, but instead suggests that the solvers might have difficulties approximating identity mappings by multiple nonlinear layers. The problem can be tackled by the residual learning reformulation deployed by ResNet. Furthermore, despite the inspiration of ResNet by the VGG nets [19] for the implementation of the plain network, it has also considerably fewer filters and, therefore, lower complexity compared to the deep convolutional networks. For instance, ResNet-34 has only 18% of the FLOPs in VGG-19 [5], an advantage along with other mentioned capabilities that made the ResNet even a rather promising approach.

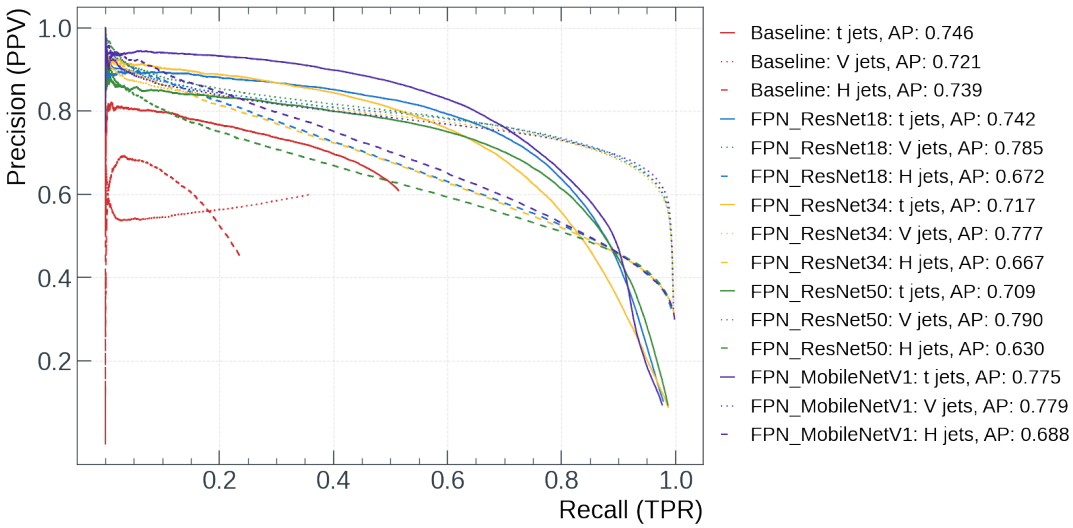


Fig. 2: Result of the implemented ResNet architectures and the MobileNetV1

During this project, we have intended to apply the new set of architecture by appending ResNet-18, 34, and 50 to the Jet-SSD. The programming comprised implementation of the new structures by an object-oriented approach using Pytorch. Indeed, we managed to do this by making some changes to the ResNet class of the TorchVision package to guarantee its well-implementation while working in harmony with other parts and features. Also, some additional modifications were necessary to be made into the SSD configuration document regarding feature maps and steps in the SSD settings. Following these steps, the training process commenced by exploiting the GPUs provided in the CERN facility. The initial results were a drawback. We have witnessed overfitting to some extent with the consequence of a loss value that was considerably greater than the results of MobileNetV1. Due to the hardware constraints and in spite of using Cuda for multi-core training, the addition of new samples was not an available option to us. Instead, we followed alternative methods to enhance the outcome; changes to weight initialization concluded to negligible effects, while luckily the hiring of pre-trained models and data augmentation contributed to a +5% increase in the final accuracy. The precision-recall plot is illustrated in Fig. 2 for the ResNet structures and MobileNetV1. It can be seen that although the results of ResNet-18 and 34 surprisingly surpass the ResNet-50's outcome, they still slightly fall behind figures representing the MobileNetV1. Something contrary to what we were anticipating. This might stem from the uniqueness of our field of work in contrast to the implementation of image processing for conventional picture databases. ResNet however will still be studied in the next step of this project for regularizing the Jet-SSD to explore lower latency models.

2.3 A MorphNet Based Approach

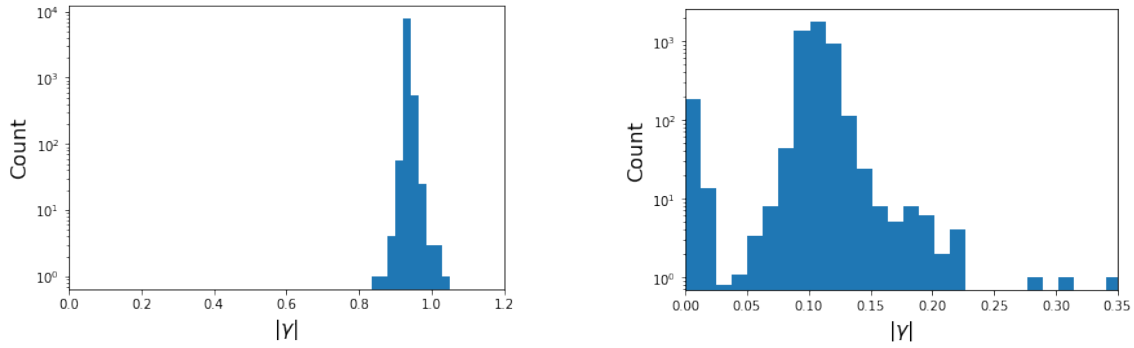
As discussed earlier, the main goal of this project is to contribute to the development of fast inference neural networks. To achieve such an objective, two main approaches of size compression [20–22] and FLOPs reduction [16, 23, 24] were the most favorable techniques in earlier works. While shrinking the size of a model can be sought through straightforward approaches like using weight sharing or Quantization Aware Training (QAT), a feature also supported in the Jet-SSD, targetting FLOPs is usually more complicated. Although the whole process comprises a kind of the same sparsifying regularization scheme, operations like model pruning or the hiring of separable convolutions are a bit tricky to perform. Nonetheless, to tackle latency in hardware-implemented deep neural networks, FLOPs seems to be a more functional constraint to pursue with a substantial effect. The challange in our case is rather relevant to the development of an infrastructure capable of automating the design procedure instead of just opting

for the right technique. In general, a similar platform might be found helpful in any akin application when considering the numerous experiments needed to be done before converging to final architecture. But aiming our escalating problem, the distinction of particle identification from other fields of work, the development of such a program seems like a critical demand. To satisfy this need, MorphNet was a fertile source of inspiration to us. MorphNet is a scalable approach adaptable to specific resource constraints for the automation design of neural networks established by Google. MorphNet utilizes a hybrid strategy alternating between a sparsifying regularizer and a uniform width multiplier. It iteratively shrinks the input network via a resource weighted regularizer on activations followed by uniformly expanding all layers by a multiplicative factor [16]. This approach is well sculptured to find over-parameterized and bottlenecked layers to eliminated the calculated cost for two supporting constraints of model size and FLOPs. Although the developed repository of MorphNet is accessible in GitHub [25], it can not be directly exploited in our work.

Firstly, despite the appearance of the MobileNet, ResNet, or other conventional architectures in MorphNet's ready-to-train structures, it is noteworthy that the deployment of these models is accompanied by modifications in this project. The alternation comprises the addition of BatchNorms, changes in the size of the layers for the implementation of headers, etc. Moreover, these deep neural networks are used as a backbone model within our SSD architecture. To address the prior issue one can alter the source code for using a modified structure instead, but dealing with the latter problem is rather challenging. While MorphNet is designed to target constraints in image classifiers, rewriting its algorithm is necessary for the appropriate regularization and calculation of cost in our SSD-based trigger application. Before starting this process, the approach was successfully tested on a personal classifier using uncommon Channel State Information (CSI) dataset to make sure of its validity. After, the efforts continued by applying an L1 regularizer to the training operation. This regression method that is termed as Least Absolute Shrinkage and Selection Operator (Lasso) adds the absolute value of magnitude of coefficients to the loss function,

$$\text{Loss} = \min_{\theta} \mathcal{L}(\theta) + \lambda \sum_{1}^{n} |w_i| \tag{1}$$

with a suitable scale factor of λ. As depicted in Fig. 3, it can be seen that the deployment of this regularizer on the γ_L variables of BatchNorm layers was surprisingly effective at inducing sparsity. With the clear separation in γ's distribution each associated with a particular neuron determining its scale, those which have been zeroed out can be easily pruned away. In addition, the shifted median also contributes to a more robust training trend. But to use this regularizer for the development of a complete automation designing process, our endeavors remain unfinished.



Primary Gamma Distribution Secondary Gamma Distribution

Fig. 3: A comparision between histogram of $|\gamma|$ distribution in BatchNorm layers of (left) primary trained MobileNetV1 architecture and (right) after applying the L1 regularizer to the training process

3 Conclusion and Future Work

To target strict low-latency requirements of the HL-LHC, trigger systems and algorithms for particle identification must be redeveloped. Addressing this demand, we commenced a process starting by reimplementation of the Jet-SSD, followed by extension of ResNet to the project, and finally proposing a promising approach for hardware-oriented automation design of satisfying deep neural networks.

In future work, the full execution of hardware-restrained automation design is considered to be the first step. Once a primary architecture is achieved with the desired specifications this way, its implementation on FPGA and experimental tests would be the next challenges to tackle. Also, testing other conventional structures as the SSD backbone besides the MobileNet and ResNet can help achieving better detection accuracy.

Acknowledgements

I am thankful to CERN for providing such an opportunity as Summer Student Programme. A chance for the gathering of students from all over the world cooperating on interesting projects, along with attending great lectures and virtual tours at the CERN facilities. I also could not be more grateful to Adrian Alan Pol for his tremendous support and supervision during the course of this project.

References

[1] S. Chatrchyan et al. The CMS Experiment at the CERN LHC. https://inspirehep.net/files/908cbd9d2558bcd1c536ed2fc72a7483, 2008.

[2] V. Khachatryan, A.M. Sirunyan, A. Tumasyan, W. Adam, E. Asilar, T. Bergauer, J. Brandstetter, E. Brondolin, M. Dragicevic, J. Ero, and et al. The CMS trigger system. https://arxiv.org/abs/1609.02366v2, Jan 2017.

[3] Valentina Gori. The CMS high level trigger. https://arxiv.org/abs/1403.1500, Jan 2014.

[4] The high-luminosity large hadron collider (HL-LHC). https://home.cern/science/accelerators/high-luminosity-lhc.

[5] Ekaterina Govorkova, Ema Puljak, Thea Aarrestad, Thomas James, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, Nicolo Ghielmetti, Maksymilian Graczyk, Sioni Summers, Jennifer Ngadiuba, Thong Q. Nguyen, Javier Duarte, and Zhenbin Wu. Autoencoders on fpgas for real-time, unsupervised new physics detection at 40 mhz at the large hadron collider. https://arxiv.org/abs/2108.03986, 2021.

[6] Claudionor N. Coelho, Aki Kuusela, Shan Li, Hao Zhuang, Jennifer Ngadiuba, Thea Klaeboe Aarrestad, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, and Sioni Summers. Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. https://arxiv.org/abs/2006.10159, Jun 2021.

[7] Huilin Qu and Loukas Gouskos. Jet tagging via particle clouds. https://arxiv.org/abs/1902.08570, Mar 2020.

[8] How a detector works / Calorimeters. https://home.cern/science/experiments/how-detector-works.

[9] Torbjorn Sjostrand, Stephen Mrenna, and Peter Skands. A brief introduction to pythia 8.1. https://arxiv.org/abs/0710.3820, Jun 2008.

[10] The jet single shot detection dataset for simultaneous localization and classification of jets.

[11] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaitre, A. Mertens, and M. Selvaggi. DELPHES 3: a modular framework for fast simulation of a generic collider experiment. https://arxiv.org/abs/1307.6346, Feb 2014.

[12] Open source Jet SSD study. https://github.com/AdrianAlan/jet-ssd.

[13] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. https://arxiv.org/abs/1512.02325, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. https://arxiv.org/abs/1512.03385, 2015.

[15] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. https://arxiv.org/abs/1704.04861, 2017.

[16] Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Tien-Ju Yang, and Edward Choi. MorphNet: Fast & Simple Resource-Constrained Structure Learning of Deep Networks. https://arxiv.org/abs/1711.06798, 2017.

[17] Adrian Alan Pol, Thea Aarrestad, Katya Govorkova, Roi Halily, Anat Klempner, Tal Kopetz, Vladimir Loncar, Jennifer Ngadiuba, Maurizio Pierini, Olya Sirkin, and Sioni Summers. Jet single shot detection. https://arxiv.org/abs/2105.05785, 2021.

[18] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. https://arxiv.org/abs/1506.02640, 2016.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. https://arxiv.org/abs/1409.1556, 2015.

[20] Maxwell D. Collins and Pushmeet Kohli. Memory bounded deep convolutional networks. http://arxiv.org/abs/1412.1442, 2014.

[21] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks, 2015.

[22] Hao Zhou, Jose M. Alvarez, and Fatih Porikli. Less is more: Towards compact cnns, 2016.

[23] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. http://arxiv.org/abs/1611.06440, 2016.

[24] Tom Veniat and Ludovic Denoyer. Learning time-efficient deep architectures with budgeted super networks. http://arxiv.org/abs/1706.00046, 2017.

[25] Ofir Nachum Bo Chen Hao Wu Tien-Ju Yang Edward Choi Ariel Gordon, Elad Eban. MorphNet GitHub Repository. https://github.com/google-research/morph-net, 2017.