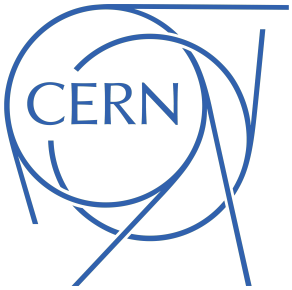
Low Level Machine Learning for the ALPHA-g Detector

Summer student report

Bruno Boato

Advisor: Lukas Golino, Niels Madsen

August 26, 2024



Abstract

This project focuses on the development of a machine learning model designed to trigger the Data Acquisition (DAQ) mechanism for the ALPHA-g experiment. Employing a machine learning model to quickly distinguish if the event was generated by a cosmic ray or an anti-hydrogen annihilation before acquiring the data would increase the signal-to-noise ratio of the system.

1 Introduction

1.1 Background and Motivation

The study of antimatter is pivotal in understanding the fundamental structure and origins of the universe. According to the standard model of cosmology, the universe began with equal amounts of matter and antimatter, however, the observable universe today is predominantly composed of matter. This discrepancy, known as matter-antimatter asymmetry or baryon asymmetry, has profound implications for cosmology and particle physics.

To investigate this asymmetry, physicist study fundamental symmetries such as the Charge, Parity, and Time Reversal (CPT) symmetry, which combines three transformations: charge conjugation (C), parity inversion (P), and time reversal (T). Charge conjugation transforms a particle into its antiparticle, so if C-symmetry were perfect, the laws of physics would remain unchanged for particles and their corresponding antiparticles. While parity inversion reflects spatial coordinates as if in a three-dimensional mirror, which implies that if P-symmetry holds, the laws of physics should be identical when viewed in a mirror. Time reversal, as the name suggests, involves reversing the flow of time in a system. Initially, these transformations were thought to be fundamental symmetries preserving the laws of physics. However, discoveries in particle physics have revealed that the weak interaction violates both C and P symmetries individually, and even their combination, the Charge-Parity (CP) symmetry, has been observed to be violated in certain particle decays.

By examining the combined CP symmetry, instances where this symmetry is not conserved can be explored, leading to CP violation. these CP violations occur when the laws of physics differ for particles and their antiparticles in a way that is not simply a mirror image. Studying CP violation is essential for understanding why the universe contains much more matter than antimatter.

The ALPHA (anti-hydrogen Laser Physics Apparatus) experiment at CERN plays a crucial role in addressing the fundamental questions of antimatter research, including the baryon asymmetry problem. By precisely measuring the spectral lines of anti-hydrogen and comparing them to hydrogen, the experiment tests fundamental symmetries, such as CPT invariance, by producing, trapping and analyzing anti-hydrogen atoms, ALPHA aims to answer fundamental questions about antimatter through precise spectroscopic measurements.

While theoretical arguments suggest that antimatter should behave identically to matter under gravity, experimental verification remains an important goal. Such experiments not only test the weak equivalence principle for antimatter but also probe the foundations upon which a quantum theory of gravity might be built.

ALPHA-g is an extension of the ALPHA experiment designed to specifically explore the gravitational behavior of anti-hydrogen, utilizing a vertical magnetic trap configuration, which allows the observation of the free fall motion of anti-hydrogen atoms, the goal of this experiment is to determine whether antimatter experiences gravity in the same way as normal matter.

1.2 Overview of the Detector

The Radial Time Projection Chamber (rTCP) (radial Time Projection Chamber) is a three dimensional particle tracking detector designed to reconstruct the anti-hydrogen location from the charged pions released in during the annihilation process. The detector consists of a cylindrical structure. Charged pions pass through the gas medium in the rTCP, as they pass through they ionize gas molecules, creating ionization electrons along their path. The ionization electrons created are influenced by the radial electric field E_r, these electrons drift radially outward though the outer walls of the detector due to this electric field. During their drift the electrons spiral around the magnetic field lines (because of the axial magnetic field B_z), resulting in a helical path. The outer walls of the detector are equipped with cathode pads which collect the drifting electrons, as these are reached by the electrons, they induce signals that can be measured and generate a read out signal. The signals are induced on the outer cathode pads, which have 576 fold segmentation in the Z axis, and 32 folds

in the ϕ axis (azimutal angle) adding up to 18432 readout channels.

The reconstruction is not done with ML, it is standard fitting techniques. But the event classification (which is ML) relies on the reconstruction which can be slow, and computationally expensive.

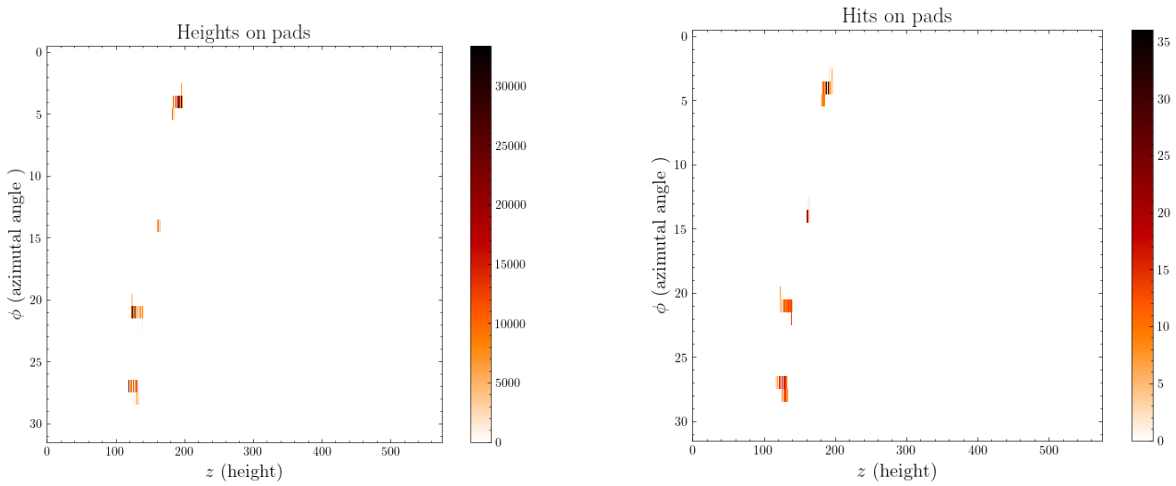
The existing event classification relies on a reconstruction algorithm which makes use of standard fitting techniques, and can be slow and computationally expensive.

The existing machine learning algorithm responsible for event reconstruction operates at a high level, relying of the reconstruction algorithm. Consequently, this algorithm's execution can be slow and computationally expensive, given that the determination of whether an event is classified as cosmic or signal is made only upon the completion of the reconstruction process. This dependency of the event classification on the reconstruction algorithm means that any event can trigger reconstruction, thus reducing the signal-to-noise ratio, given that every time a cosmic event triggers the event readout and reconstruction, we can not process an annhilation event for the remaining 5ms (1/200Hz) it takes to readout.

To enhance the signal-to-noise ratio, we propose the implementation of a two-tiered approach. Initially, a more expedient and lightweight algorithm could be employed to evaluate incoming events prior to triggering the detector readout and primary reconstruction algorithm. By utilizing a faster model, specialized model, which could be run in an FPGA (Field Programmable Gate Array), we could significantly reduce the number of random events that unnecessarily activate the detector. This preemptive filtering would optimize the use of computational resources by ensuring that only likely signal events proceed to the more compute-intensive reconstruction stage.

2 Data Handling

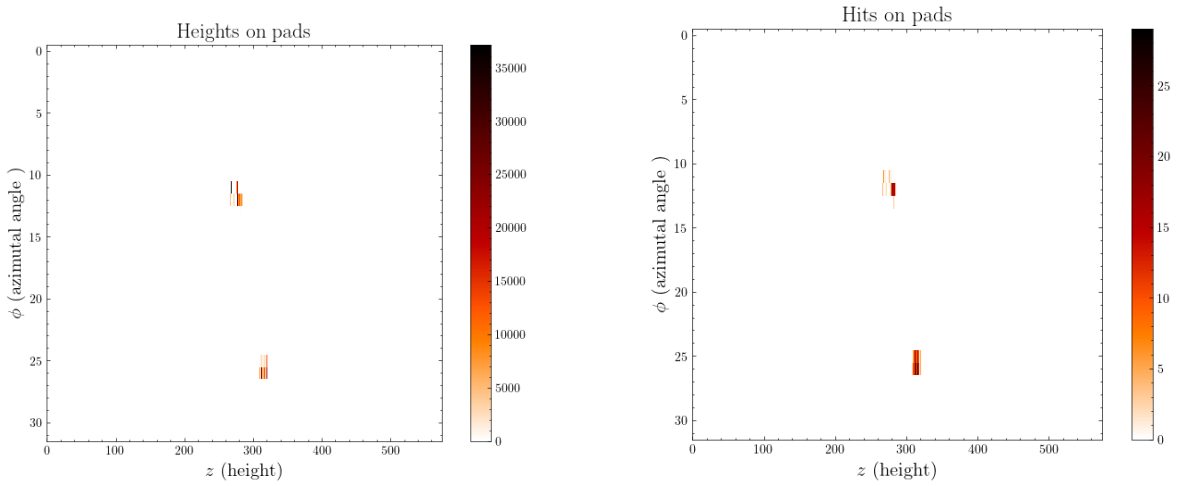
The data from the readouts was stored in a vector of 18432 elements, corresponding to the concatenation of all the folds, so in order to extract more intuitive information, we plot the 2d heatmap in the rolled out cylinder of the sensors, from there we can obtain images of the sensor activations with two channels for each event, one for the numbers hits on the pads and another for the amplitude heights.



(a) Heatmap of heights

(b) Heatmap of hits

Figure 1: Reshaped signal event images



(a) Heatmap of heights

(b) Heatmap of hits

Figure 2: Reshaped background event images

2.1 Exploratory Data Analysis

Now reformulate this, changing as little as possible to make it fitting of a paper: To gain a better understanding of the data beyond analysing a single sample, we average across events to generate a visual representation of the data. By plotting the average over all the samples we can better illustrate the hit and amplitude height distributions, and use this plots to identify differences between these distributions in cosmic events vs hit events.

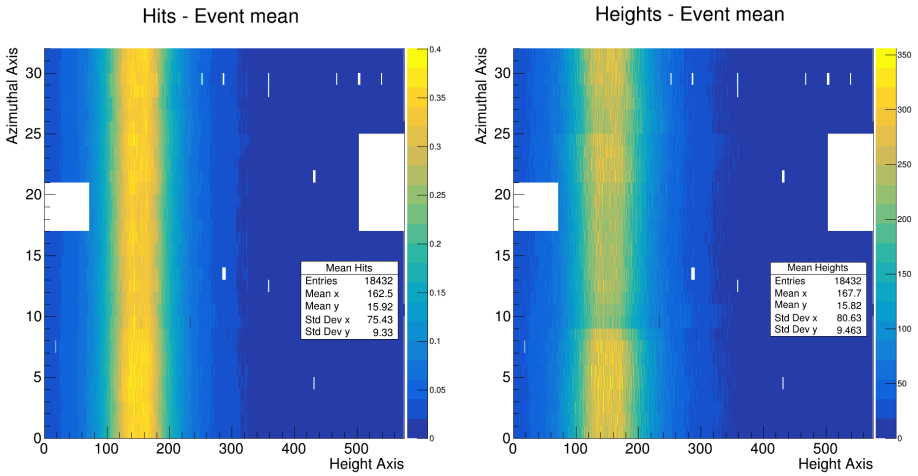


Figure 3: Average Heatmap over all signal events

One observation we can derive from the heatmaps is the absence of measurements in regions corresponding to faulty pads. This lack of data can potentially skew the results and is an important factor to consider when training models using this dataset. It is crucial to account for these gaps since future datasets are unlikely to have such defects, and ignoring them could lead to models that are biased or less accurate in predicting events under normal conditions.

The comparison between the average heatmaps for both signal and cosmic ray events reveals a

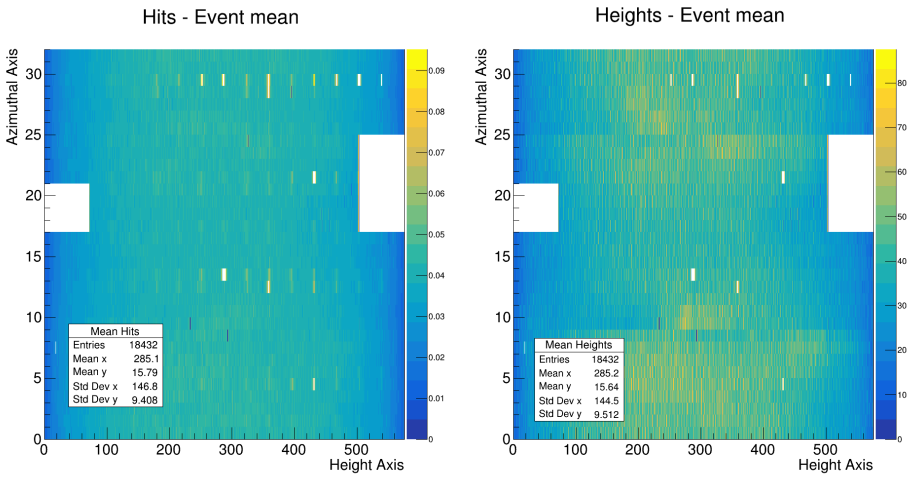
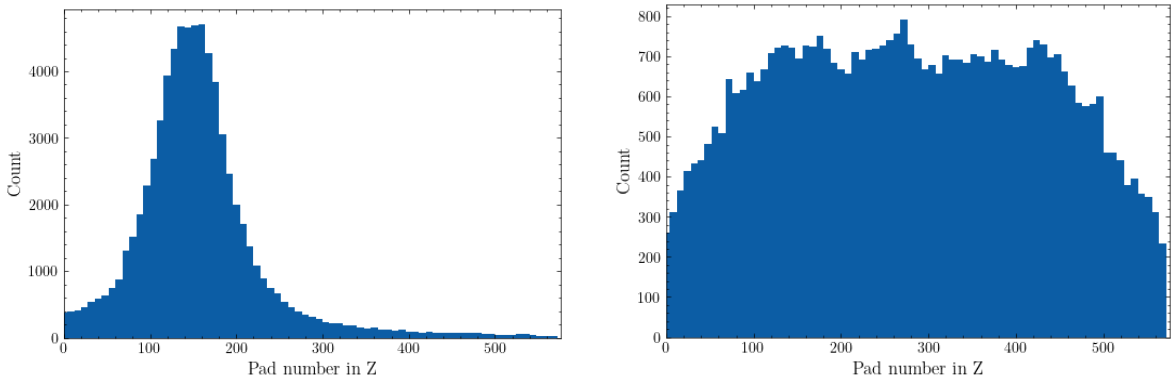


Figure 4: Average Heatmap over all background events

distinct difference in the distribution of hits, as illustrated in Figures 3 and 4. In the case of signal events, there is a notable concentration of hits and increased amplitude heights along a specific section of the Z-axis. This pattern can be attributed to the confinement of anti-hydrogen happening in the same section of the trap, which significantly increases the bias in the data, thus the probability of an annihilation event occurring in one specific area of the trap is much higher compared to other areas. In contrast, cosmic ray events exhibit a more uniform distribution of hits across a broader section of the trap, indicating that cosmic rays have an equal likelihood of generating events throughout the entire trap.

To explore this further we integrated the measured signals along the Z-axis over 10,000 samples to produce histograms which (Figure 5) illustrate the distribution of events along the Z-axis.



(a) Z Counts for signal events

(b) Z Counts for background events

Figure 5: Histograms of events along the Z axis (bin size of 8)

2.2 Data Preparation

A data preparation pipeline is necessary to be able to manage our large-scale datasets efficiently, for this we employ the uproot library, a powerful tool for reading and writing ROOT files in pure Python

and NumPy. Uproot enables the creation of iterators that return batches of filtered entries as NumPy arrays, allowing us to load data into memory in manageable chunks for processing and training.

Building upon this efficient data loading mechanism, we implement TensorFlow dataset generators. These generators create a seamless pipeline between data loading and model training, allowing for on-the-fly data generation and augmentation of data. Our custom generator function takes two key inputs: a signal event iterator and a background event iterator, which will be processed separately, and later concatenated with their corresponding labels to allow for supervised learning.

From each entry in our datasets, we extract two primary features: hit counts, representing the number of hits registered in each sensor pad, and amplitude heights. The iterator returns a numpy array structured as a 2×18432 matrix. We subsequently transform this matrix into a two-channel image representation. Each channel corresponds to one of the key features and is reshaped into a 32×576 format. which corresponds to an unrolled representation of the cylindrical arrangement of the sensors in the trap. By maintaining the spatial relationships between data points, we enable our subsequent machine learning models to potentially learn and exploit spatial patterns that may be characteristic of signal or background events.

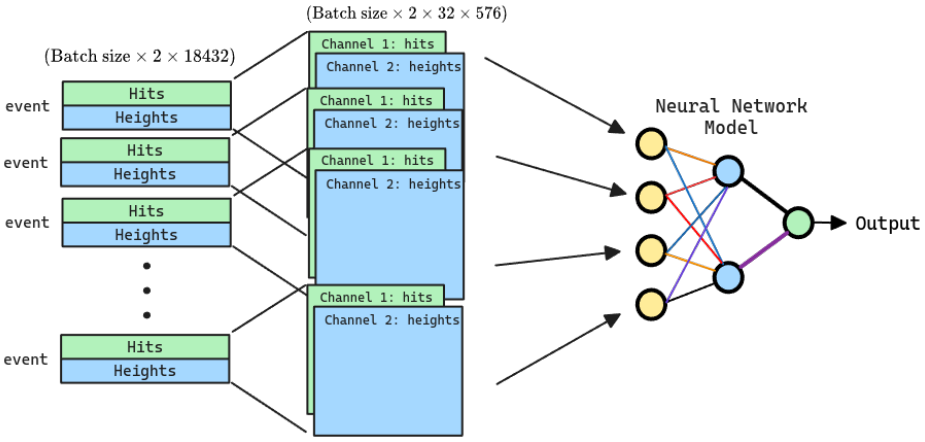


Figure 6: Illustration of training pipeline

Both hit counts and amplitude heights undergo a min-max normalization, by subtracting the minimum value of the dataset and then dividing by the range. This step is crucial for bringing both features to a comparable scale, preventing one feature from dominating the other due to differences in raw magnitudes. Normalization often leads to faster convergence during model training and ensures that the relative importance of features is maintained. The normalization process is applied separately to hit counts and amplitude heights, and individually for signal and background data, preserving any inherent differences between signal and background distributions. To facilitate our supervised learning approach, we generate corresponding labels for each dataset. Signal data is assigned a label of 0, while background data is assigned a label of 1. This binary labeling scheme sets up our problem as a binary classification task.

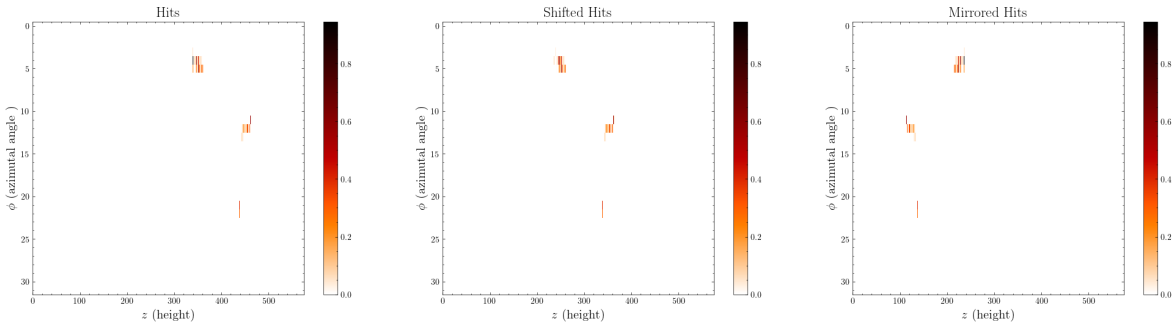
The final step in our feature processing involves stacking the normalized and reshaped hit counts and amplitude heights to create a multi-channel image-like structure. Each "image" in our dataset consists of two channels, one for hit counts and one for amplitude heights. This multi-channel approach allows us to represent multiple aspects of each event simultaneously, providing a richer dataset for our models to learn from. The resulting data structure has a shape of $(\text{batch size}, 2, 32, 576)$. The background and signal events are then concatenated together, with the same amount of entries taken from the iterators from each group to conform the batch, ensuring dataset balancing to prevent bias.

2.3 Data Transformation and Loading for Model Training

antiprotons and positrons are only held in one place, and we only mix in 1 electrode.

1. Random Z-axis Shifts: We introduced random shifts along the Z-axis to simulate a more uniform distribution of anti-hydrogen throughout the trap volume.

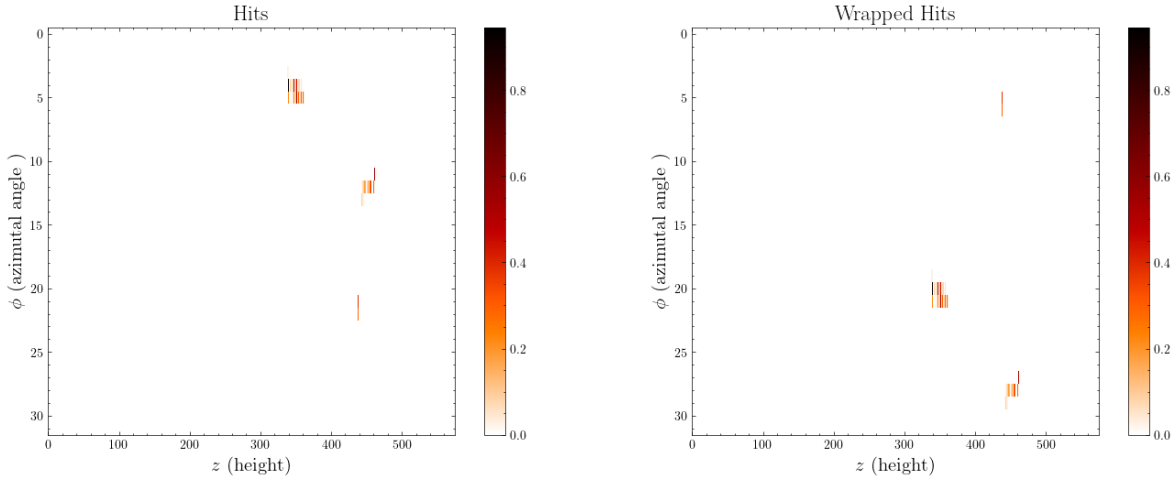
2. Z-axis Mirroring: We implemented a mirroring technique along the Z-axis to further homogenize the spatial distribution of both signal and background events.



(a) Hits without transformation (b) Hits with random z shift (c) Hits with mirroring

Figure 7: Transformed Images from sensors

To address the issue of faulty detector pads resulting in data gaps (white patches), we leveraged the cylindrical geometry of the sensor, applying random rotations around the phi-axis to the entire dataset redistributing the impact of faulty pads across different regions of the detector in our training data, reducing the model's sensitivity to these hardware-specific faults with the data.



(a) Image without transformation (b) Image after random rotation

Figure 8: Reshaped event images

The resulting histogram of the signal data after having applied the forementioned transformation can be seen in figure 9.

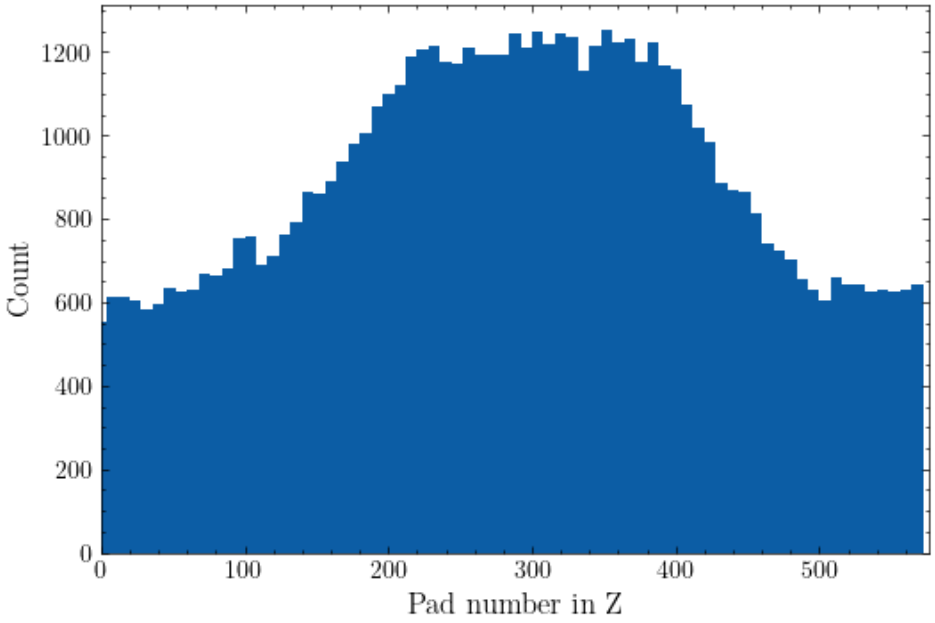


Figure 9: Z histogram after transformations

3 Model Development and Evaluation

3.1 Model Architectures

3.1.1 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a kind of deep learning model primarily used for analyzing visual imagery. Unlike traditional neural networks, which use fully connected layers, s leverage convolutional and pooling layers to capture spatial hierarchies in data, which allows them to process images in a way that is computationally efficient and highly effective at feature extraction.

Convolutional layers, apply a set of learnable filters to the input data. Each filter is convolved across the width and height of the input, performing a dot product between the entries of the filter and the input at any position. This operation produces a two-dimensional activation map that indicates the presence of features (such as edges or textures) in the input image. Convolutional layers exploit local spatial coherence by restricting the receptive field of each filter to a small area of the input, which enables them to learn localized patterns.

Pooling layers reduce the spatial dimensions of the activation maps. Pooling helps to down-sample the data, reducing the number of parameters and computational load, and controls overfitting. The most common form of pooling is max pooling, which takes the maximum value from a small rectangular neighborhood in the input map, effectively summarizing the most prominent features and discarding the less important ones.

CNNs have become a cornerstone in the field of machine learning, particularly in tasks involving spatially structured data like images, and given that we preserved spatial information from the cylindrical arranged sensors by representing the data in the form of two image like arrays, this makes this type of neural network architecture a good candidate for the task at hand.

The proposed CNN model consists of three convolutional layers with max pooling layers in between, followed by a dense layer with ReLu activation before one final linear layer for the binary classification.

3.1.2 Dilated Convolutional Neural Network

Dilated CNNs (CNNs) have emerged as a powerful variant of traditional CNNs, particularly in tasks requiring extensive contextual information. By incorporating a spacing (dilation) between the elements of the convolution filter, effectively "dilating" the filter over the input image, how much we dilate the filter is determined by the dilation rate, a dilation rate of 1 corresponds to standard convolution, while higher rates allow the filter to cover a larger area of the input with the same number of parameters. This process enables the network to focus on both local and global features by progressively increasing the dilation rate across layers.

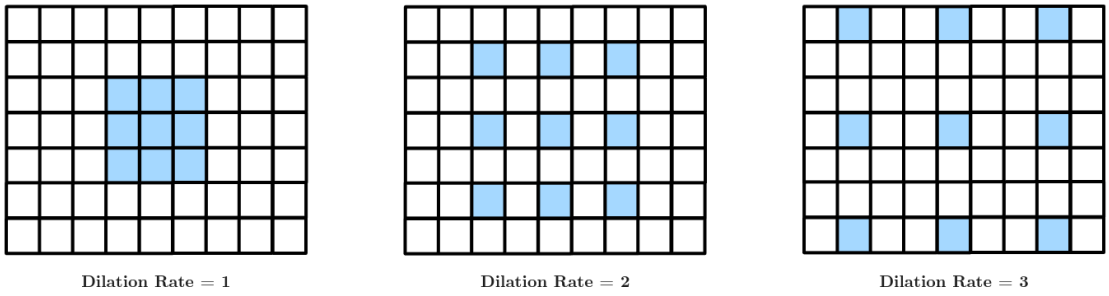


Figure 10: Ilustration of convolution kernel with varying dilation rates

These networks effectively expand the receptive field of their convolutional filters without increasing the number of parameters, allowing them to capture more global features and context from the input data.

Dilated CNNs operate by introducing a spacing (dilation) between the elements of the convolutional filter, effectively "dilating" the filter over the input image. This dilation rate determines how far apart the kernel elements are spread. A dilation rate of 1 corresponds to standard convolution, while higher rates allow the filter to cover a larger area of the input with the same number of parameters. This process enables the network to focus on both local and global features by progressively increasing the dilation rate across layers.

By using dilated convolutions, networks can consider wider areas of the input image without the need for additional layers. This broader view allows the network to understand more complex spatial relationships, which is critical for task such as this one where the interactions between distant pixels matter.

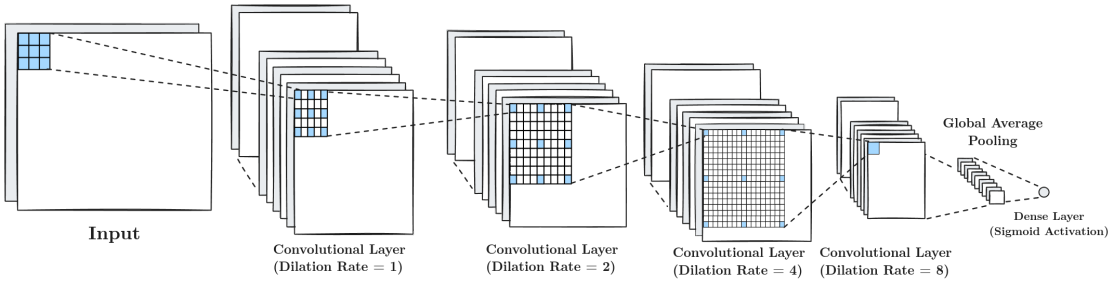


Figure 11: Dilated CNN architecture

The model consists of multiple convolutional layers, each with a different dilation rate. The first convolutional layer applies a standard 3x3 filter, capturing local features with a dilation rate of 1. As the data passes through each subsequent layer, the dilation rate doubles, expanding the receptive field and enabling the model to integrate more global information. By the fourth convolutional layer,

the dilation rate reaches 8, providing a wide perspective while still maintaining the resolution of local details. Following the convolutional layers, the model employs global average pooling to condense the spatial information into a single feature vector, this is a technique in neural networks where the information from different parts of an image is averaged to create a single summary for each feature. Instead of focusing on specific areas, global average pooling looks at the whole image to see the overall presence of that feature in the image. This pooling operation not only reduces the computational burden but also facilitates the transition from spatially structured data to a format suitable for classification. The final layer of the model is a dense layer with a sigmoid activation function, designed for binary classification task.

This model leverages the strengths of dilated convolutions by systematically increasing the dilation rate in successive convolutional layers. This approach allows the model to learn hierarchical representations, starting with fine-grained features and gradually incorporating broader context, while also reducing the number of required parameters.

3.1.3 Attention Based Convolutional Neural Network

Attention based Neural Networks make use of attention blocks, which are mechanisms designed to help the network focus on the most important parts of the input. They achieve this by assigning different levels of importance (attention) to various elements of the feature maps produced by convolutional layers. Attention blocks make use of two kinds of attention:

- Channel-wise attention focuses on which feature channels are most important. It first applies global average pooling to summarize each channel's information across the entire spatial area, capturing the global context of the feature maps. Then, it generates a channel-wise attention vector using fully connected layers. This vector is multiplied with the original feature maps, effectively emphasizing the more important channels.
- Spatial attention determines which spatial locations within the feature maps are most relevant. This is done by applying a 1x1 convolution to create a spatial attention map, which highlights important areas within the feature maps. The spatial attention map is multiplied with the feature maps to emphasize the significant spatial regions.

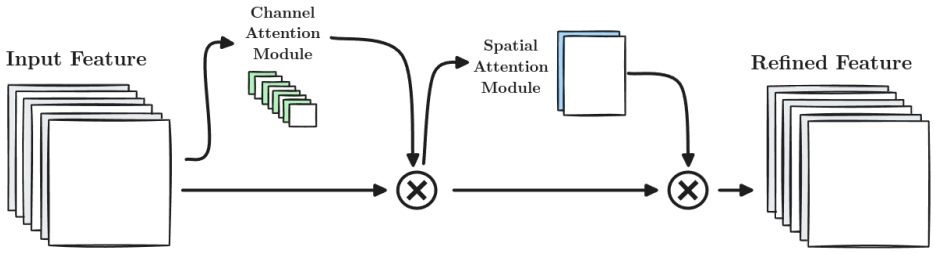


Figure 12: Illustration of Convolutional Attention block based on the original paper [1]

During the project, two attention-based algorithms were tested for image classification tasks. The first model integrates attention blocks within a sequence of diluted convolutional layers, and pooling layers, while the second lightweight model utilizes Separable Convolutions and Squeeze-and-Excite Blocks.

The first model employs a series of convolutional layers with increasing dilation rates, allowing it to capture features at various scales. After each convolutional block, a max-pooling layer reduces the spatial dimensions, helping the network focus on the most relevant features. Attention blocks are then strategically placed after each convolutional operation. These blocks first apply channel attention by using global average pooling to capture the global context of the feature maps. A series of dense

layers generate a channel-wise attention vector, which is multiplied with the input feature maps to emphasize important channels. Next, spatial attention is applied by convolving the feature maps with a 1x1 convolution to generate a spatial attention map. This map highlights important spatial locations, and the feature maps are multiplied with this attention map to enhance relevant areas. The output of the channel and spatial attention mechanisms is combined using an addition operation, resulting in a feature map that incorporates both types of attention. This combined attention map is then passed to the next layer of the network, allowing for more refined feature extraction. The model concludes with a global average pooling layer that aggregates the spatial information into a single feature vector, which then passes through a dense layer with a sigmoid activation function for binary classification.

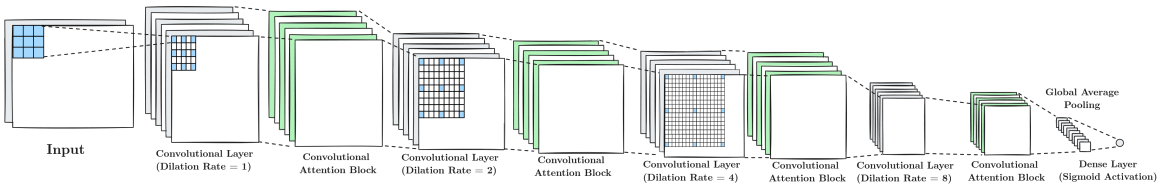


Figure 13: Illustration of model using convolutional attention blocks

The second model adopts a lightweight design, using separable convolutions and squeeze-and-excite blocks to achieve high efficiency and performance.

Squeeze-and-Excite (SE) blocks are a sophisticated type of attention mechanism designed to improve the representational power of a convolutional neural network (CNN) by selectively emphasizing informative feature channels and suppressing less useful ones. SE blocks operate on the principle that not all features are equally important for a given task, and their importance can vary depending on the input. The process begins with the "squeeze" operation, where global average pooling is applied to each channel of the feature map to capture channel-wise statistics, which summarize the information in the channel. The "excite" operation consists of using a small additional network (called a bottleneck) to generate a set of importance values, or modulation weights. These weights tell the network how much attention to give to each feature channel. The original features are then adjusted (or recalibrated) based on these weights, allowing the network to focus more on the most important features.

small feed-forward network consisting of a bottleneck architecture with two fully connected layers. The output of the excite operation is a set of weights, which are applied to the original feature maps through element-wise multiplication, allowing the network to recalibrate the feature responses adaptively.

Separable convolutions are an efficient variant of standard convolutional layers that significantly reduce the computational cost and the number of parameters in a CNN. This technique divides a regular convolution operation into two distinct stages: depthwise convolution and pointwise convolution. In the depthwise convolution step, each input channel is convolved separately with a different kernel, capturing spatial features independently for each channel. This is followed by a pointwise convolution, which uses a 1x1 kernel to combine the outputs of the depthwise convolution across channels. By separating the spatial and channel mixing processes, separable convolutions enable the model to achieve similar representational power to standard convolutions while being more computationally efficient.

The lightweight attention model architecture leverages the combination of separable convolutions and squeeze-and-excite blocks to create an efficient CNN. The model starts with an input layer, where the dimensions are rearranged to ensure compatibility with the following operations. It then proceeds with four blocks, each consisting of a separable convolution layer followed by batch normalization and a squeeze-and-excite block, designed to enhance important feature channels dynamically. After each block, max-pooling layers are used to reduce the spatial dimensions of the feature maps, ensuring that the most salient features are retained. The dilation rate of the separable convolutions increases

progressively in each block, allowing the network to capture features at different scales effectively. Finally, the output from the last block undergoes global average pooling to aggregate the spatial information into a single vector, which is then passed through a dense layer with a sigmoid activation function for classification.

3.2 Model Comparison

We evaluated the performance of four models, Classic CNN, Dilated CNN, Attention CNN, and Attention SE CNN, feeding the data in batches of 32768 samples to the model. Conducting testing after every one of 17 epochs.

Each model was trained on the same dataset, but the data generator used a data iterator to load the samples. This meant that each epoch was composed of different subsets of the data, ensuring that the models were exposed to different samples throughout the training process. During training every best performing model was saved to later pick the best performing model on testing data to run on validation data.

The performance of these models was compared based on two key metrics: the number of parameters, which is crucial for their future implementation in an Field Programmable Gate Array (FPGA), and the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC), which plots the True Positive Rate (sensitivity) against the False Positive Rate, showing how well the model distinguishes between the two classes.

Model	**Number of Parameters (Size)**	**AUC**
Classic CNN	4,774,753 (18.21 MB)	0.87
Dilated CNN	28,385 (110.88 KB)	0.96
Attention CNN	9,997 (39.05 KB)	0.99
Attention SE CNN	6,761 (26.41 KB)	0.94

Table 1: Comparison of CNN Models Based on Parameters and AUC in validation data

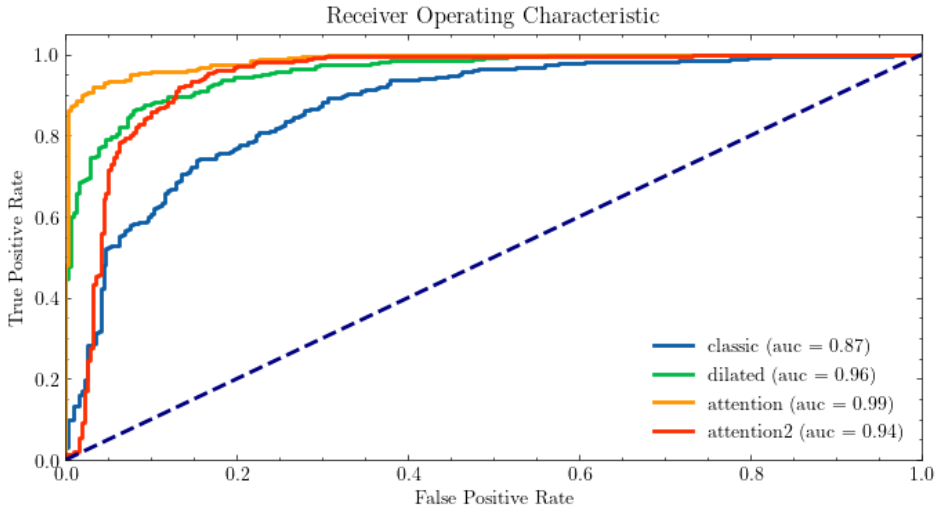


Figure 14: Comparison of ROC AUC Across Different Models

Despite having the largest number of parameters (4,774,753) and size (18.21 MB), the Classic CNN achieved a relatively lower AUC score of 0.87. In contrast, the Dilated CNN demonstrated an impressive balance between model size and performance, achieving a high AUC of 0.96 with only

28,385 parameters (110.88 KB), suggesting that dilated convolutions can effectively capture relevant features with significantly fewer parameters.

The Attention CNN exhibited the best performance among all models, achieving the highest AUC of 0.99 with a modest 9,997 parameters (39.05 KB). This outstanding result highlights the effectiveness of attention mechanisms in enhancing model performance while maintaining a relatively small parameter count. The Attention SE CNN, while having the fewest parameters at 6,761 (26.41 KB), achieved a competitive AUC of 0.94. Although not the highest performing model in this comparison, its ability to achieve strong results with the smallest parameter count demonstrates the potential efficiency of combining attention mechanisms with squeeze-and-excitation blocks.

4 Future Work

4.1 Development of a Quantized Model and Pruning Strategies

Quantization and pruning are two techniques critical to make better use of neural networks in resource constrained environments such as FPGAs. Quantization involves reducing the precision of a model's weights and activations. For this, high precision floating point numbers which are usually used for model weights and activations, are converted into smaller integer representations, such as 8 bit integers. Prunning involves removing redundant or less important parameters, such as very small weights, with the purpose of removing model complexity and making the model smaller without losing much of the accuracy.

These two techniques are important to deploy the model in FPGAs because these devices are designed to operate with limited logic gates and memory. FPGAs excel at parallel processing and computation speed but they are resource constrained in comparison to CPUs or GPUs. By prunning unnecessary parameters from a neural network, we minimize the resource requirements necessary to run the model.

Cluster-preserving quantization-aware training (QAT) is a specialized technique that integrates quantization into the training process, allowing the model to learn while considering the effects of lower precision. This method preserves the clusters of similar weights and the sparse structure introduced by pruning, ensuring that the model remains efficient even after quantization. By training the model with quantization in mind, the network learns to operate effectively with lower precision while maintaining the benefits of sparsity. Making use of this approach could in a model that is both highly optimized for the constraints of FPGAs and capable of maintaining high accuracy.

4.2 Model Distillation Techniques for Quantized Representations

Model distillation is a technique used to transfer the knowledge from a large, complex neural network (often called the "teacher" model) to a smaller, simpler one known as the "student" model. The goal of model distillation is to retain the performance and accuracy of the original model while significantly reducing its size and computational requirements. When combined with quantization, model distillation becomes a powerful method for creating efficient, high-performance models that are well-suited for hardware with limited computational power and memory such as FPGAs.

In model distillation, the student model is trained to mimic the outputs of the teacher model rather than simply learning from the raw training data. This process typically involves using the teacher's soft labels, which are the probabilities of the output classes, to guide the training of the student model. By learning from these soft labels, the student model can capture the nuanced knowledge that the teacher model has acquired, such as the relationships between different classes and the importance of various features. When the student model is trained in this manner, it often achieves a level of accuracy that is close to the teacher model, despite having far fewer parameters.

Quantizing models can lead to a loss of accuracy, especially in smaller models that may already be more prone to errors. However, by distilling the knowledge from a high-precision teacher into

a lower-precision student, it's possible to mitigate the accuracy loss. The distillation process helps the student model learn robust representations that are less sensitive to the reduced precision of quantization, resulting in a quantized model that retains high performance.

Model distillation can be adapted specifically for quantized models by incorporating quantization-aware training during the distillation process. In this approach, the student model is trained not only to replicate the teacher's behavior but also to perform well under quantized conditions. This involves simulating quantization effects during training, allowing the student model to adjust to the lower precision while still adhering to the performance benchmarks set by the teacher model.

4.3 FPGA Implementation with High-Level Synthesis using HLS4ML

HLS4ML is an open-source framework designed to translate machine learning models into hardware descriptions that can be implemented on FPGAs and other hardware accelerators. The name "HLS4ML" stands for "High-Level Synthesis for Machine Learning," the objective of this framework is to simplify the deployment of machine learning models onto hardware through high-level synthesis (HLS) tools. This framework allows us to build HLS code from compressed neural networks.

Traditional methods of deploying neural networks on FPGAs often require significant manual effort to optimize and convert the models into a hardware-friendly format. HLS4ML automates much of this process, allowing us to more rapidly deploy machine learning models on FPGAs.

Acronyms

DAQ Data Acquisition

CPT Charge, Parity, and Time Reversal

CP Charge-Parity

rTCP Radial Time Projection Chamber

FPGA Field Programmable Gate Array

CNN Convolutional Neural Network

AUC Area Under the Curve

ROC Receiver Operating Characteristic

References

[1] S. Woo, J. Park, J. Lee, and I. S. Kweon, "CBAM: convolutional block attention module," *CoRR*, vol. abs/1807.06521, 2018.

[2] E. K. Anderson, C. J. Baker, W. Bertsche, N. M. Bhatt, G. Bonomi, A. Capra, I. Carli, C. L. Cesar, M. Charlton, A. Christensen, R. Collister, A. Cridland Mathad, D. Duque Quiceno, S. Eriksson, A. Evans, N. Evetts, S. Fabbri, J. Fajans, A. Ferwerda, T. Friesen, M. C. Fujiwara, D. R. Gill, L. M. Golino, M. B. Gomes Gonçalves, P. Grandemange, P. Granum, J. S. Hangst, M. E. Hayden, D. Hodgkinson, E. D. Hunter, C. A. Isaac, A. J. U. Jimenez, M. A. Johnson, J. M. Jones, S. A. Jones, S. Jonsell, A. Khramov, N. Madsen, L. Martin, N. Massacret, D. Maxwell, J. T. K. McKenna, S. Menary, T. Momose, M. Mostamand, P. S. Mullan, J. Nauta, K. Olchanski, A. N. Oliveira, J. Peszka, A. Powell, C. Rasmussen, F. Robicheaux, R. L. Sacramento, M. Sameed, E. Sarid, J. Schoonwater, D. M. Silveira, J. Singh, G. Smith, C. So, S. Stracka, G. Stutter, T. D. Tharp, K. A. Thompson, R. I. Thompson, E. Thorpe-Woods, C. Torkzaban, M. Urioni, P. Woosaree, and J. S. Wurtele, "Observation of the effect of gravity on the motion of antimatter," *Nature*, vol. 621, p. 716–722, Sept. 2023.

[3] G. Smith, *Characterization and analysis development for the ALPHA-g barrel scintillator*. PhD thesis, University of British Columbia, 2021.

[4] P. Granum, "Measuring the Properties of Antihydrogen," 2022. Presented 19 Sep 2022.

[5] C. N. Coelho, A. Kuusela, S. Li, H. Zhuang, J. Ngadiuba, T. K. Aarrestad, V. Loncar, M. Pierini, A. A. Pol, and S. Summers, "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," *Nature Machine Intelligence*, vol. 3, p. 675–686, June 2021.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[8] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2017.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[10] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," 2018.

[11] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," 2019.

[12] FastML Team, "fastmachinelearning/hls4ml," 2023.

[13] F. Fahim, B. Hawks, C. Herwig, J. Hirschauer, S. Jindariani, N. Tran, L. P. Carloni, G. D. Guglielmo, P. Harris, J. Krupa, D. Rankin, M. B. Valentin, J. Hester, Y. Luo, J. Mamish, S. Orgrenci-Memik, T. Aarrestad, H. Javed, V. Loncar, M. Pierini, A. A. Pol, S. Summers, J. Duarte, S. Hauck, S.-C. Hsu, J. Ngadiuba, M. Liu, D. Hoang, E. Kreinar, and Z. Wu, "hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices," 2021.