Using deep learning for particle identification and energy estimation in CMS HGCAL L1 trigger

AUTHOR(S):
Anwesha Bhattacharya

June-August 2019

SUPERVISOR(S):
Emilio Meschi

ABSTRACT

In run 4 of the LHC, the extreme high luminosity is expected to generate an enormous pileup of up to 200 proton-proton collisions for each bunch crossing. This has to be read out at 750 kHz with a maximum latency of 12.5μs. In order to disentangle the energy from pileup collision, the upgraded CMS detector for Run-4 will feature a new High Granularity Calorimeter (HGCAL) with unprecedented lateral and longitudinal segmentation. The total number of channels read out into the Level-1 trigger processor will be of the order of 10^6. To process this data with such small latency, we need to develop sophisticated algorithms. In this report, we aim to use machine learning techniques for electron-photon identification and energy estimation in the L1 Trigger. The idea is to implement the architectures on FPGA boards that will have fast inference, enough to cope with the requirements of the HGCAL.

TABLE OF CONTENTS

1. INTRODUCTION

For the fourth run of the LHC, CMS will undergo an update in its detector with the endcap calorimeters being replaced by the High Granularity Calorimeter (HGCAL) **[1]**. The new calorimeter will feature extremely fine transverse and longitudinal segmentation, for both electromagnetic and hadronic compartments. This is necessary to counter the effects of high pileup that will be generated due to the higher luminosity of 7.5×10^{34} $cm^{-2}s^{-1}$ expected from the increased bunch population and focussing in the LHC. Novel approaches are required to enhance pileup rejection and perform particle identification and energy estimation in the detector.

The HGCAL detector will consist of 52 layers, of which 28 layers are for the electromagnetic portion (ECAL) and 24 for the hadronic portion (HCAL) [**1, 2**]. The electromagnetic layers and a part of the hadronic layers will consist of hexagonal silicon sensors with 0.5-1 cm^2 cell size. The remainder of the hadronic portion is made of highly segmented scintillators with SiPM readout. The L1 trigger only uses alternate layers, so that the data collection is kept manageable. Hence, we have 14 effective layers from ECAL.

The latency at the L1 trigger level is 12.5μs, of which, the calorimeter only energy estimation and particle identification will have only ~1μs. To have fast inference, we explore the possibility of employing the powerful tools of deep learning and deploying these architectures on FPGAs. For this, we are using Micron's SB852 board which features Xilinx Virtex Ultrascale+ FPGA [4]. The compiler and the necessary drivers have been provided by Micron. Currently, the Micron development environment is optimized to support Convolutional Neural Networks (CNN) and taking this into account, we develop models that can perform regression and classification.

This report highlights the approaches taken and is arranged in the following order: Section 2 covers the dataset generation process, followed by the proposed models in Section 3. Section 4 reports the performance of the model on the board and we conclude with the summary and future work in Section 5.

2. Dataset

For this study. we use a simulated sample of 100,000 single particle events comprising of 50% photons and 50% electron-positron pairs generated at the interaction region. Forty percent of the photons convert to electron-positron pairs before reaching the surface of the detector. These events are excluded from the sample and will be studied separately at a later stage. This sample was generated with no pileup. The goal here is to develop an architecture with optimal efficiency in terms of latency, computational resource required and identification power. To create the dataset, we use constrained topological clustering that generates 20x20 grids around the simulated hit which corresponds to the maximum energy deposition across all 38 channels. The 3D representation of the energy deposits is shown in Figure 1(a). Initially, the idea was to use the first 14 layers that correspond to the electromagnetic part of the detector. However, it was observed that the electromagnetic layers alone are not sufficient to contain the energies of the most energetic particles and that these particles penetrate into the hadronic layers as well. This can be seen in Figure 1(b). Therefore, we consider all 38 layers for our training.

Traditional CNNs work with RGB images with 3 input channels, with each channel processed by dedicated kernels. In other words, the channels do not share weights. Since our representation has 38 channels and they are correlated (because the energy deposits across the layers are from the same shower), the natural choice is to use approaches that are either independent of the input shape or use 4D inputs, for example, convolutional LSTMs, 3D-CNNs or GNNs.

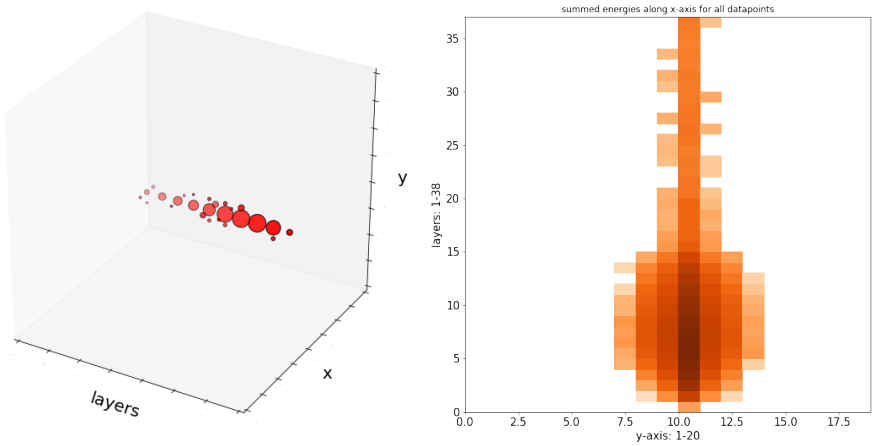


Fig 1.(From left to right) a) 3D visualization of energy depositions. The size of the spheres is proportional to the energy in the cell. b) Total energy of all trigger cells when summed across 38 layers.

3. DL model architectures

Model 1 - 3D CNN architecture

The selection of the model for inference requires the consideration of model performance and its implementation on hardware. As explained in the earlier sections, taking into account the Micron toolset characteristics and in order to optimize the performance, we settle on using CNN models. We start by using 3D-CNNs to incorporate inter-channel correlation. The proposed model is shown in Figure **2.** The input to the network is a matrix of dimension 38x20x20x1. This is convolved twice with 32 3D kernels of size 3x3x3 with relu activation, followed by a 3D max-pooling layer with pool size of 3x3x3. This is followed in series by a 0.25 dropout layer, flatten layer, a fully connected (FC) layer with 64 nodes, a 0.5 dropout layer, and finally a 2 node output using cross-entropy loss for classification.

Training using ~23k samples with learning rate of 1e-3 for around 80 iterations and early stopping, we get an accuracy of 85%, which means that 85% of the time the test data is categorized correctly. The confusion matrix is shown in Table **1**. However, while the performance is acceptable, the time taken for inference is quite large at around 0.8 ms per test image on a GTX 1080Ti GPU**.** It is also quite a memory-intensive process because of the storage requirements of 4D kernels. It was therefore decided to explore the use of 2D CNNs.

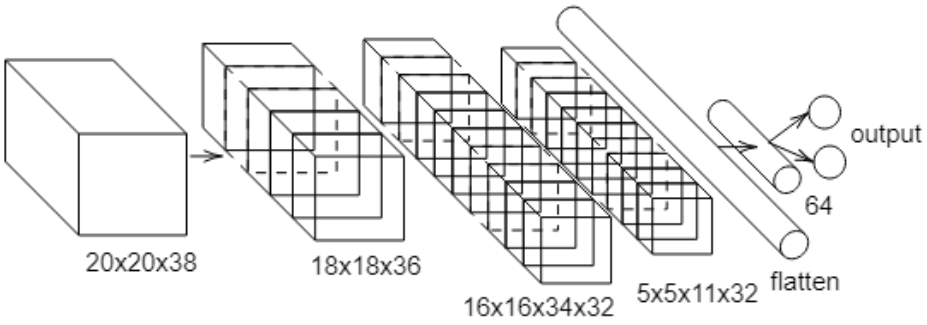


Fig. 2 : Classification network with 3D convolution.

Model 2 - 2D CNN architecture

1 input architecture

To reduce the inference time, we tested a 2D CNN approach. The difference is that the kernels used here have 2 dimensions, such that they slide over individual channels of the input image instead of convolving across them. To see the effect of using a 2D model, we use the architecture shown in Figure **3.** For this, we have developed two models. In the first model, we use one projected image as shown in Figure 4. For the second model, we use the full image with 38 channels. As expected, the full image performs better at identification than the projected images. In fact, its predictive power is close to that of the 3D CNN architecture. We further explore the possibility of using projected images and to improve the performance. To this end, we use the 2 input architecture discussed in the next section.

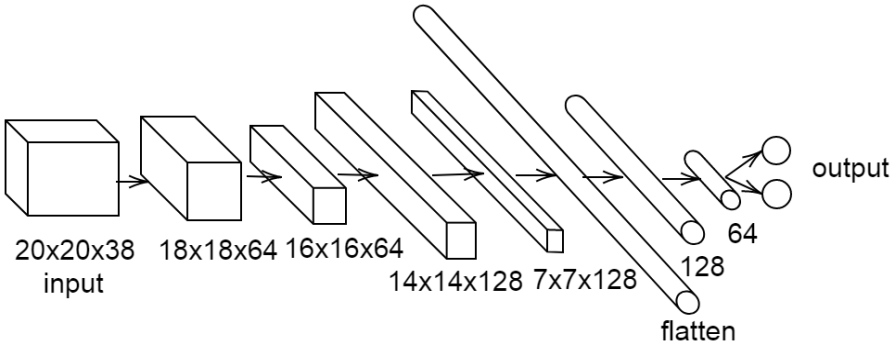


Fig. 3 : 2D classification network with 1 input.

2 input architecture

While 3D CNNs used weight sharing between different channels of the input image, thereby learning any existing relationship between them, using independent 2D filters for each channel does not capture this relation. Therefore, we develop our 2D CNN by using 2 inputs that together can be expected to capture the information across the channels. These inputs are generated by projecting the 3D image along the x and y axes resulting in a pair of arrays of size 38x20x1 each. These inputs, while retaining the layer information, also have information from the x and y dimensions. The image pair is shown in Figure **4** with the color intensity being proportional to the energy deposition magnitude.

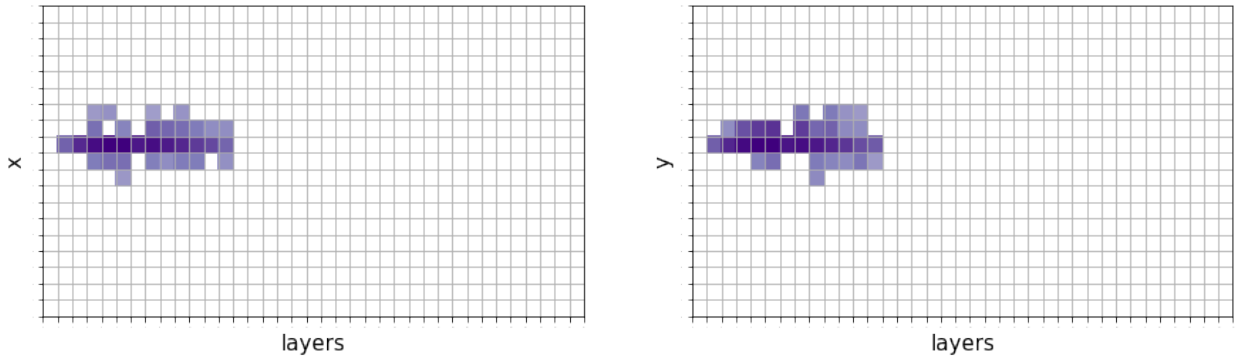


Fig. 4 : Energy deposits with layer vs x axes on summing along the y-axis. Energy depositions in layer vs y axis on summing along x-axis. The color intensity is representative of the energy content in the cell with darker cells having more energy.

The architecture we use is shown in Figure 5 and takes as inputs a pair of 38x20x1 images generated by summing along the x and y axes. The model develops along two symmetrical branches. The first layer after the input is a convolution layer with kernel size 3x3, stride 1, followed by another 2D Conv layer with 64 filters and a kernel size of 3x3. All convolution layers have relu activation functions. This is followed by a 2D max pooling layer with window size 2x2 and stride size 2x2. The output from the two branches are concatenated and then flattened at this point to generate a vector of length 17408. We then apply two FC layers of sizes 128 and 64, each followed by a 0.5 dropout. Categorical cross-entropy is used as a loss function for both the 2D and 3D models. The inference time for this model is 0.09ms which is much faster compared to the 3D model.

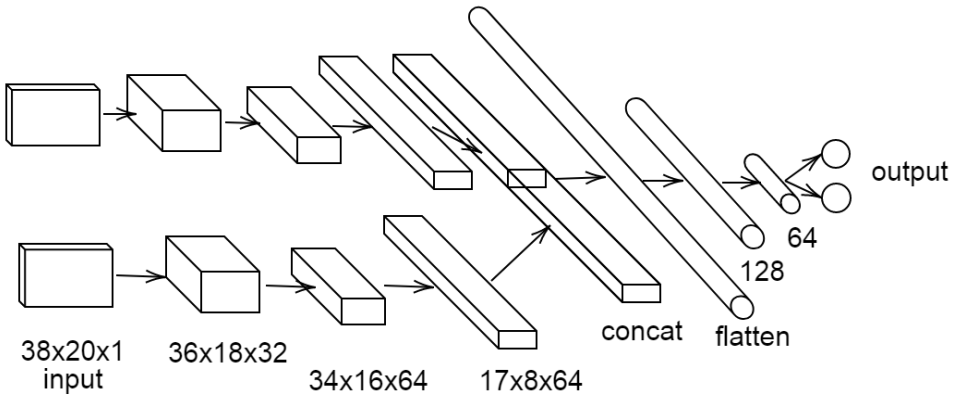


Fig. 5 : 2D classification network with 2 inputs.

Classification performance of the models.

We tested our model on ~14k samples and achieved 85% accuracy with both 3D and 2D models. The confusion matrix is given in Table 1.

			Pred Photons	Pred e^+/e^-
True photons	2D conv	1 input - X1	0.78	0.22
		1 input - X2	0.81	0.19
		1 input - X	0.84	0.16
		2 input	0.82	0.18
	3D conv	3D conv	0.85	0.15
True e^+/e^-	2D conv	1 input - X1	0.11	0.89
		1 input - X2	0.12	0.88
		1 input - X	0.15	0.85
		2 input	0.13	0.87
	3D conv	3D conv	0.15	0.85

Table 1 : Classification performance of the models in terms of confusion matrix. The different row colors represent different model's performance.

For a better understanding of the performance of the 3D and 2D models, a visual comparison is provided with a classification plot in Figure **6**. From these, we can see that the 3D model is able to distinguish the two types of particles based on their energy deposition more accurately than the 2D model.

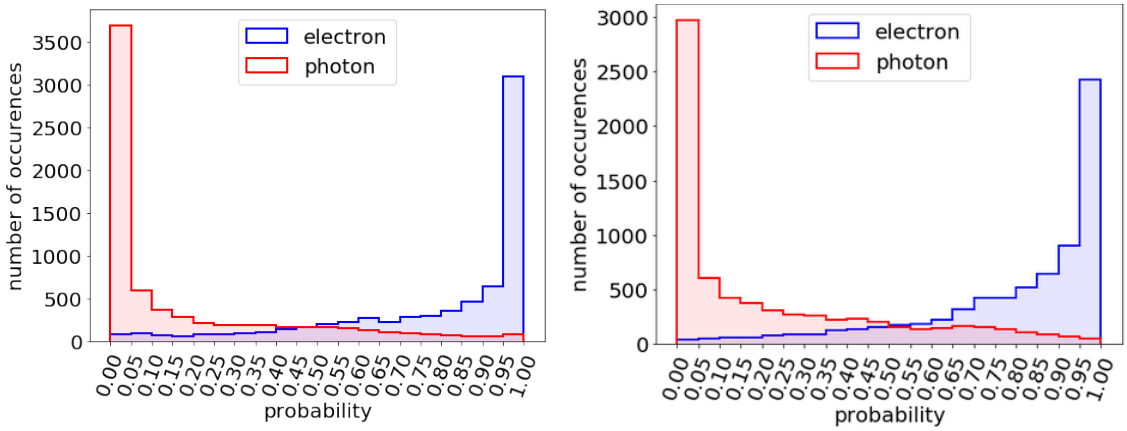
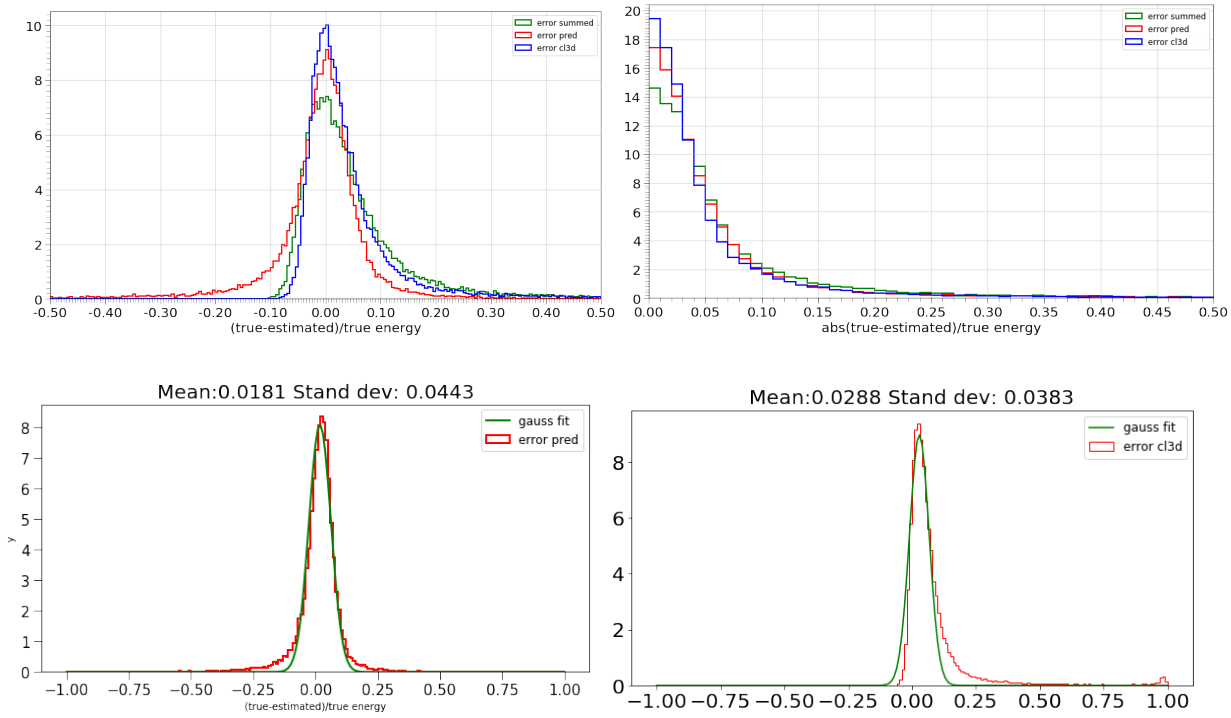


Fig. 6 : Histogram showing the performance of the 3D convolution model (left) and the 2D 2-input classification model (right). The color of the histograms represents the ground truth particle ID and the probability represented on the x-axis is the model's prediction. From the figure we see that the 3D model performs slightly better than the 2D 2-input model.

Regression model and performance

We further want to explore the capability of deep learning models for energy estimation. To do this, we use a similar model as that for 2D classification using the nominal energy of the incoming particle as target. In the FC layers, the relu activation is replaced with elu activation function. The last layer has one output node and uses mean squared error (mse) loss for regression instead of categorical cross-entropy that was used for classification. We train both the 2-input and 1-input models for comparison.



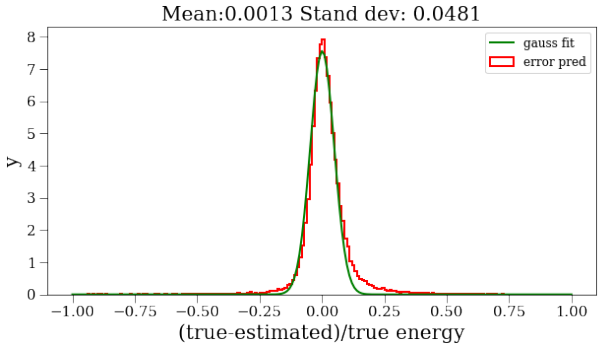


Fig. 7 : Performance of 2D 2-input model in energy estimation. In the top plots, the red curve shows the relative error in the predicted energy, green shows relative error of true energy and energy estimated by summing up energy across all cells, and the blue curve represents the relative energy error with respect to the energy predicted using density based clustering approach (cl3d). Top left: Relative error histogram plot. Top right: Absolute value of the relative error plot. Mid left: Gaussian fit on the relative error from cl3d. Mid right: Gaussian fit on relative error for 2-input 2D model estimation. Mid left: Gaussian fit on the relative error from cl3d. Bottom: Gaussian fit on relative error for 1-input 38 channels 2D model estimation.

In Figure 7 we show the relative error distributions for the three energy estimates: the 2D 2-input model predictions, the traditional density-based clustering approach (cl3d) and, as a sanity check, the total energy if we summed up all the trigger cell energy values for a particular cluster. From these plots, we can see that the model has performance close but not competitive with that of cl3d. The last three plots in Figure 7 show the standard deviation based on a Gaussian fit of the cl3d based prediction and the predictions based on both models. Clearly, the cl3d performs better. This shows the scope for further improvement of the regression models to achieve competitive performance.

4. Model on hardware

We tested our models on the SB852 board which is a “full-height, GPU-length, PCIe x16 Gen3 board with a Xilinx® VU9P Virtex Ultrascale+ FPGA, a 2GB Hybrid Memory Cube (HMC), up to 512GB of high-performance memory and two QSFP28 connectors” [4]. To do that, we first convert our model to .onnx format using onnx convert tools for keras models. Then we use Micron’s dedicated compiler [3] to convert the .onnx model to their FPGA bitstream. We compare the output from the hardware with the output from onnxruntime, which is a tool to run the .onnx models. The output from the runtime is identical to the output of an inference made directly using python keras.

Sample outputs from onnxruntime and the board for the 2D classification model with two inputs are given below. *Particle ID* refers to the type of particle for the test case, *output onnxruntime* is the inference using onnxruntime and *output hw* is the board’s output. The individual test cases are separated by horizontal lines.

```
particle ID =  22.0 → output onnxruntime =  0.6614985 ,  output hw =  0.71780354

--------------------------------------------------

particle ID =  22.0 → output onnxruntime =  0.43477285,  output hw =  0.5117166

--------------------------------------------------

particle ID =  22.0 → output onnxruntime =  0.9984659, output hw =  0.9641675

--------------------------------------------------

particle ID =  22.0 → output onnxruntime =  0.74310744, output hw =  0.62062156
```

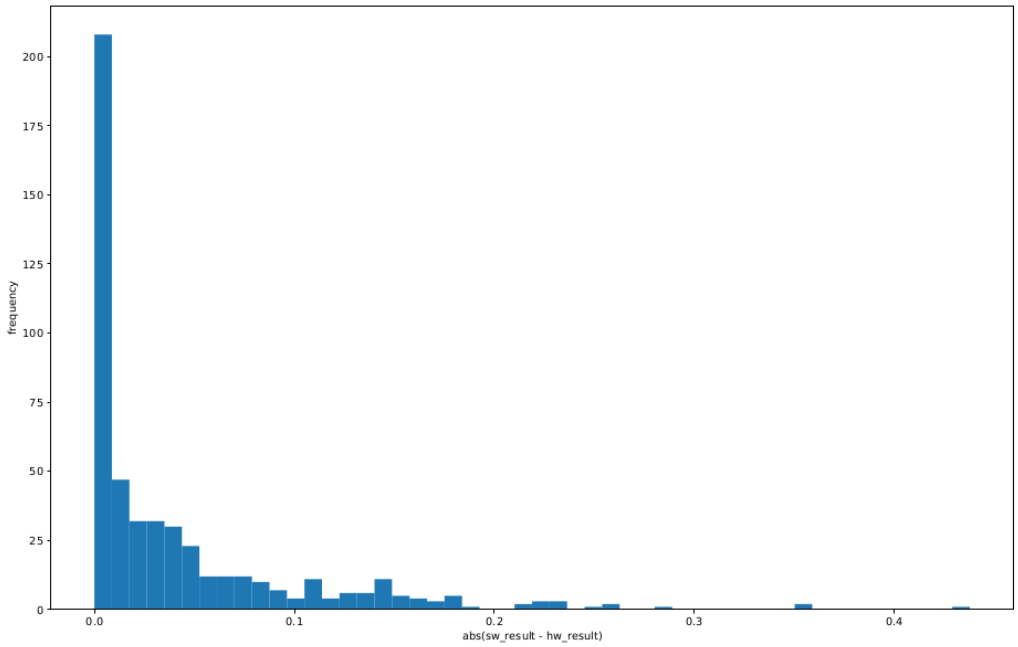


Fig. 8 : Plot of frequency vs absolute value of the relative error of the predictions from the model on SB852.

Figure **8** shows the frequency of the absolute value of the relative difference between the output from onnxruntime and the hardware. We can see that they do not match precisely. For the purpose of particle classification, this might be acceptable, but in general, the L1 requires agreement to better than 0.001%. More investigation is needed to understand these discrepancies. The probable reasons these differences trigger could be overflow or the effect of using fixed-point precision for the board while using floating-point

precision on onnxruntime. A way to validate this would be to check the results from onnxruntime's inference using fixed point precision.

5. Conclusion

We have developed deep learning architectures to perform particle identification and energy regression based on simulated data. The models are based on hardware requirements and the dataset used. Two broad variety of models have been explored- one based on 3D convolution and the other with 2D convolution. Under the 2D convolution model category, we have used single input models and a model having 2 inputs. We see promising results for classification and regression using these deep learning architectures. The current best model gives 85% classification accuracy. Currently, the regression results are not competitive with the traditional density-based clustering approaches for energy estimation. With a more diverse training data and tailored models, the results are expected to improve.

6. Acknowledgments

First and foremost I am very grateful to my supervisor, Emilio Meschi, for guiding me through the whole project. I thoroughly enjoyed our discussions especially when they delved into 'deep' physics. I also thank Thomas, Dejan, and Awais for helping me and answering all my questions. I am also thankful to the Micron team for funding this project and helping our team to resolve the issues we faced with the board. Last but not least, my sincerest thanks to the OpenLab community for giving me this wonderful opportunity to be a part of CERN and have one of the greatest experiences.

7. References

[1] The Phase-2 Upgrade of the CMS Endcap Calorimeter, https://cds.cern.ch/record/2293646

[2] The Phase-2 Upgrade of the CMS L1 Trigger Interim TDR, https://cds.cern.ch/record/2283192?ln=en

[3] FWDNXT SDK https://github.com/FWDNXT/SDK

[4]MicronHardware https://www.micron.com/products/advanced-solutions/advanced-computing-solutions