Neural Networks for Single-Channel Dual-Readout Calorimetry

Murali Saravanan[1] and Federico De Guio[2]

[1] Cornell University, Ithaca, NY, USA
[2] Texas Tech University, Lubbock, TX, USA

Abstract. This repo contains work done with the Texas Tech group for the CMS collaboration at CERN, specifically in close collaboration with Dr. Federico De Guio. It is a brief study of the feasibility of using machine learning techniques, specifically shallow neural networks, in hadronic dual-readout calorimetry, which simultaneously measures scintillation light and Cherenkov light. The electromagnetic (em) shower fraction can be calculated by comparing these two signals on an event-by-event basis. We show that machine learning has the potential to allow both scintillation and Cerenkov to be read out from a single channel while still allowing for an accurate calculation of the em shower fraction. This short report validates single-channel dual-readout calorimetry as a feasible technique in future high energy particle detector calorimeters. Moreover, there are possible latency advantages if these neural networks are hardwired into FPGAs, which could possibly aid a Level 1 trigger.

Keywords: Neural Networks

Table of Contents

1 Introduction

Over the last twenty years, Dual Readout Calorimetry has emerged as a possible technique for measuring the energy of hadronic jets that offer certain advantages over other calorimetry techniques. (see [1] for a more extensive review of this recent work). In particular, dual readout calorimetry allows us to obtain the electromagnetic portion of the hadronic energy on an event-by-event basis, due to the ability to collect both Cerenkov and Scintillation radiation. Since the em portion of the hadronic energy can fluctuate severely between events, it is the main obstacle for achieving high hadronic energy resolution. Dual-readout calorimetry is a promising technique for use in hadronic calorimeters of future colliders, such as the ILC, CLIC, and FCC.

Single-channel dual readout calorimetry entails combining both of these radiation types into one single channel. For example, a calorimeter with a combination of clear and scintillating fibers might only have one silicon photomultiplier collecting data from both types of radiation. Since these two different types of radiation have different pulse signatures, it is possible that they could be analyzed separately after data collection, which would potentially cut overhead hardware costs by nearly a half. Although a fitting procedure could be utilized to discriminate between Cerenkov and Scintillation, this is a time-consuming procedure which has latency disadvantages. This project focuses on the possibility of using neural networks to conduct this event-by-event analysis as they can be used for **fast** inference after training has been completed. This would allow it to be potential uses in a Level 1 Trigger.

1.1 Understanding the Pulse Signal

Note: The the y-axis is in arbitrary units for the following subsection.

First, consider a sample pulse of only Cerenkov radiation. Due to the nature of the radiation, it would resemble a sharp peak as shown in red in the figure below.

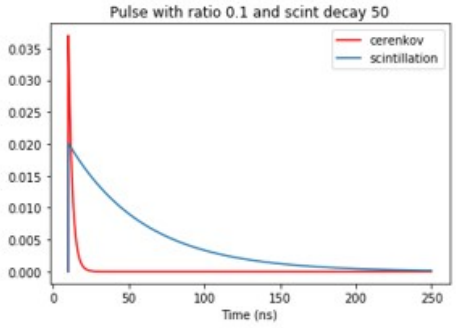

Fig. 1: Example Pulses

We generate this signal as a product of two exponentials, one with a rise rate that characterizes the exponential increase of the signal and another with a decay rate that characterizes the exponential decay of the signal.

Now, consider a sample scintillation-only signal from a scintillating fiber, shown in blue above. It has a much slower decay time, but is also generated similarly.

In a single-channel, these two signals would be superimposed leading to a pulse that would ideally look like the one below.

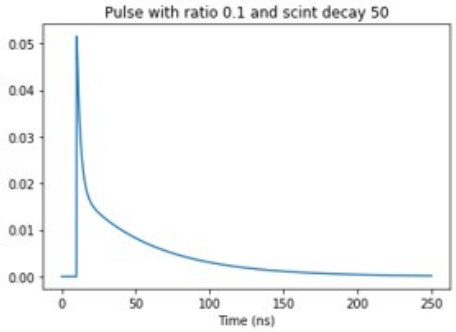

Fig. 2: Example Pulse

We can write this superimposed pulse in the form of the following equation.

We make the assumption that a hypothetical Silicon Photomultiplier would respond to both types of radiation identically. The rise rate of incoming signal is determined by the photomultiplier (PMT) collecting photoelectron data. Since photoelectrons from both types of radiation hit the PMT simulataneously, the assumption that the pulses rise identically is justified.

In reality, a real-world calorimeter would record a variety of pulses with fluctuations in the em and non-em portion of the hardonic energy. To represent this, in this study we vary the ratio between Cerenkov and scintillation radiation. We define **ratio** as the area under the Cerenkov pulse divided by the area under the scintillation pulse. In our study, we vary ratio between $\frac{1}{25}$ and 25. This range was chosen based off sample test data taken from prototype dual-readout calorimeters shown in [1].

We also consider different scintillation decay rates. Since the decay rate for the scintillating radiation is determined by the material properties of the scintillating fiber, this is a property that can be determined when the calorimeter is constructed. Therefore, we want to consider a wide range of scintillation decay rates. We define this to be **scint_decay**, which we vary between 15ns and 50ns. These values were also chosen from looking at example data from prototype calorimeters [1]. Below is a few example pulses that show various **ratio** and **scint_decay**.

Note that the above pulses are devoid of any background noise.

1.2 Problem Statement

The goal of this project is to use neural networks to predict the amount of Cerenkov and scintillation radiation. (For a basic overview of how Neural Networks work, see [2].) The input for our neural network is essentially our pulse signal. The output is two nodes, one predicting the percentage of the signal that is Cerenkov radiation, and the other node prediction the percentage of the signal that is scintillation.

So for example, if the input signal has a **ratio** = 4, the output node predicting Cerenkov should be 0.8 and the output node predicting scintillation should be 0.2. To make this trickier, this should be true regardless of the value of **scint_decay**. The output prediction of the network should depend only on **ratio**.

2 Part I: Feasibility on Monte-Carlo Simulated Data

Real data collected from any test beam would have copious amounts of noise. Therefore, as a proof of concept, we studied whether we could preform discrimination between Cerenkov and Scintillation radiation on Monte-Carlo Simulated data.

This data is generated by taking the ideal pulses, as shown in Figures 2 and 4, and treating them as a probability density function (pdf). We sample these pdfs on the order of thousands of times. Each sample represents a photoelectron hitting the PMT. We then place these samples into a histogram with various binnings. The binning of the histogram corresponds to the hypothetical digitizer frequency of our calorimeter. Therefore the input layer of our neural network should equal the chosen binning of our histogram, with each node representing the value in each bin. Below is an example histogram.

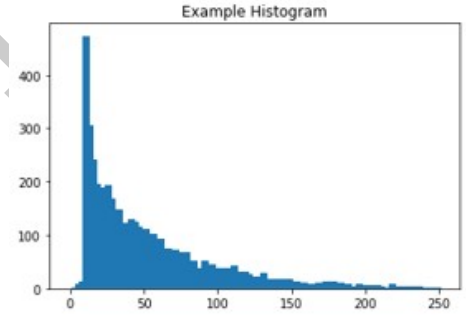

Fig. 3: Example Histogram

This histogram would be the input to our neural network, with the output being two nodes that predict the percentage of Cerenkov and scintillation radiation.

2.1 Design Choices and Implementation

Before results are discussed, the design choices made for this codebase are presented, in an effort to make it easy for the reader to build on these results.

This codebase is written entirely in `Python3`, using the `Jupyter` notebook environment to present results. The package used for the Neural Networks is `tensorflow` with the `Keras` wrapper.

The two driving forces of this code base are `pulse.py` and `nn.py`. `pulse.py` is responsible for the Monte-Carlo Data Generation and `nn.py` provides the necessary tools for building, compiling, training, and testing the model.

2.1.1 Data Generation

`pulse.py` contains two important classes, `Pulse` and `PulseLibrary`. `Pulse` takes in three inputs: **ratio**, **scint_decay**, and photoelectrons. Photoelectron count details how many times to sample the pdf created by **ratio** and **scint _decay**. This class has many useful methods, such as `graph_both_ pulses()` and `graph_pdf()`. The most important method is `output()`, which returns the histogram, Cerenkov radiation percentage, and scintillation radiation percentage.

The workhorse for data generation is the `PulseLibrary` class. It uses the `Pulse` to create a library of pulses that is then stored as a hdf5 file using the h5py package. (The `name` input parameter becomes the name of the file.) It intakes a list of **ratios** and **scint_decays**. Each combination of a ratio and scint _decay is considered one experiment. In order to create a large amount of data to train our model, we specify how many Monte Carlo simulations should be run with each experiment. This is specified by the input parameter `eventsperexperiment`. (We need more training data than the number of nodes in our network to avoid over-fitting.) Note that creating the pulse library can take an extended amount of time if 10 ratios, 10 sint_decays, and 500 eventsperexperiment are specified. The best practice to create these training and testing sets would to submit them to HTCCondor through lxplus and letting them run overnight.

For each neural network built, we generate a training data library and a testing data library. As expected from the name, the first is used to train the neural network and the second is used to test the accuracy of the network's predictions. This is done because we don't want to test on the same data as we trained on. The random seed is changed between the creation of these two libraries to ensure there are numerous random differences between the two.

The format of the hdf5 file that contains this Monte-Carlo Generated Data is as follows. Each 'experiment' is a group inside the hdf5 file. It is labeled as ratioXXscint_decayXX. Each event is a subgroup within the group, labeled as ratioXXscint_decayXXeventXX. Each subgroup contains two datasets. One is the

histogram, with the label ratioXXscint_decayXXeventXXinput, and the other is the Cerenkov and scintillation percentages, with the label ratioXXscint_decayXX eventXXoutput. Each histogram is stored as a list with the value of each bin as an entry in the list. The output is also stored as a list of length 2, with the first value being the percentage of Cerenkov and the second being scintillation.

2.1.2 Model Architecture

`nn.py` holds only one class, `NeuralNet`. This class takes in a python dictionary that determines the model architecture. (See the `__init__()` method specifications to learn more.) Note that all the Neural Networks used in this study are 'shallow' with only three layers. We will see why this is important later on in Section 4. The class method `train_model` has two inputs. `pulse_library` is the string that specifies the hdf5 file that contains the training dataset. It also takes in an input, `model_name` that gives a name to the network and stores the weights of the network as a hdf5 file. This allows the network to be reloaded without having to be retrained as training can take an extended amount of time.

The `NeuralNet` also contains the method `verify_model()` which produces a profile that is a measure of the effectiveness of the network. This class method takes in the model and a testing library. It runs the network on every histogram in the testing library and then calculates the percent difference between the predicted value and the actual percentages used to generate that Monte-Carlo simulation. For each 'experiment', the value in the profile is the average percent error for all the events simulated for that 'experiment.' The standard deviation of these percent errors is also stored as a object variable.

The rest of the methods in the `NeuralNet` create plots that allow us to visualize how effective our neural network.

The initial model architecture studied was a neural network with only one hidden layer. The input layer was 100 nodes (one for each bin in the histogram), the hidden layer had 10 nodes, and the final layer simply had two nodes. The input was normalized by the absolute max of the library (the largest value of all the histogram bins was normalized to 1).

2.2 Varying Photoelectron Count

First, we investigated how various photoelectron (phe) counts would affect the performance of the neural network. We trained and evaluated five different neural networks, one for each: 1000 phes, 2000 phes, 3000 phes, 4000 phes, and 5000 phes. Below are figures that show the absolute percent difference for each of these neural networks for each combination of ratio and scint_decay. The plot on the left shows percent difference for Cerenkov prediction and the plots on the

right show it for scintillation prediction. The color map gives a heat map of the area where the network is effective.

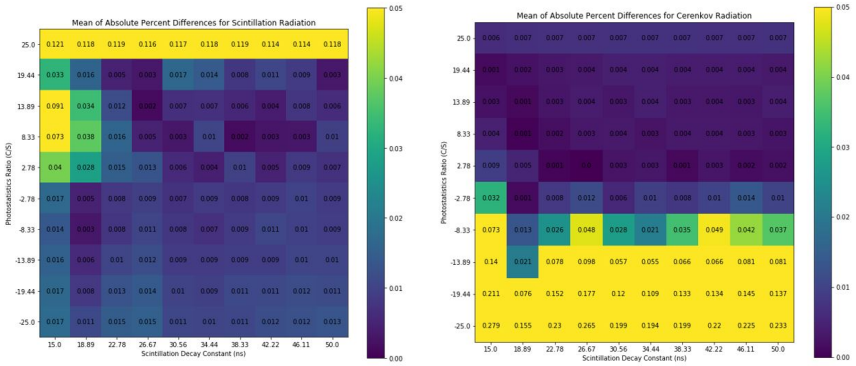

Fig. 4: Profiles for 5000 photoelectrons and 100 bins

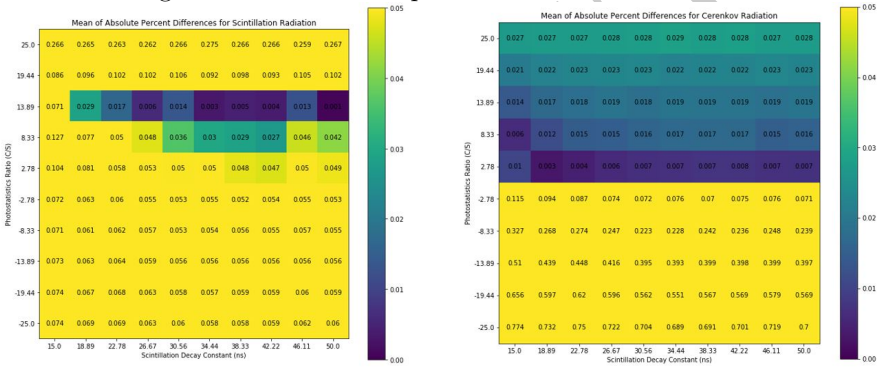

Fig. 5: Profiles for 1000 photoelectrons and 100 bins

2.3 Varying Digitizer Frequency (Input Layer)

We also investigated how varying the digitizer frequency would . This corresponds to changing the number of bins for the histograms . We looked into 300 bins, which corresponds to increasing the digitzer frequency, and 30 bins, which corresponds to decreasing the digitzer frequency. Corresponding to this, the input layer was changed to 300 nodes and 30 nodes respectively. The results of this varied digitizer frequency for 1000 and 5000 photoelectrons are shown below in a similar format.

2.4 Conclusions

From the Monte-Carlo Simulations, we can draw some important conclusions about the efficacy of Neural Networks in predicting Cerenkov and scintillation radiation percentages from hypothetical signals from a single-channel dual-readout

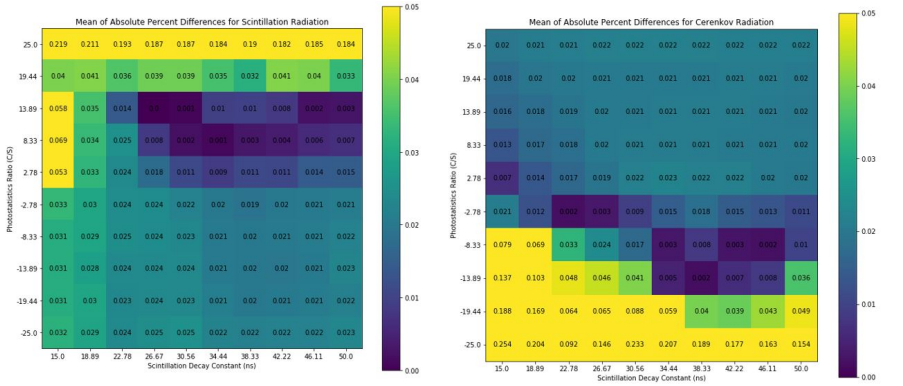

Fig. 6: Profiles for 5000 photoelectrons and 300 bins

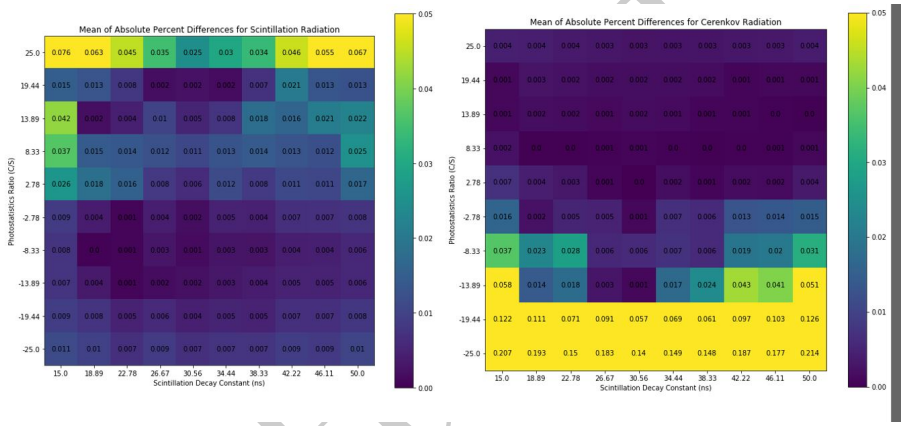

Fig. 7: Profiles for 5000 photoelectrons and 30 bins

calorimeter. From the plots in the previous two sections we see that the profile from our neural network depends mostly on the **ratio** and not on **scint_decay**. We also see that a higher photoelectron count gives better results but the change is gradual. Over 1k-5k photoelectrons range studied there is no tipping point where the neural net vastly improves. The more phes, the better the random fluctuations are hidden and the neural network makes better predictions.

We also note that a higher digitizer frequency does not result in better prediction ability. We conclude that a lower digitzer frequency hides the effects of fluctuations, which increases the overall performance of the network. We also note that scintillation prediction is much more stable than Cerenkov prediction in that, even with reduced photoelectrons or higher digitizer frequency, scinitllation prediction will not worsen as much as Cerenkov prediction.

While these conclusions may seem obvious, we note that neural networks are very capable in a certain range of parameter space. In this proposed parameter space these networks are effective with an error less than 5%. We have shown that in Monte Carlo generated data, neural networks are a viable technique in

determining the em fraction of hadronic energy in single-channel dual-readout calorimetry on an event-by-event basis within a 5% error.

3 Part II: Feasibility on Real Data

We've established from the Monte-Carlo simulations that using neural networks to measure Cerenkov and scintillation radiation ratios from single-channel dual-readout calorimeters and can be done with an error of less than 5% in a vast range of parameter space. The next logical step is to see if similar results can be achieved with real data.

3.1 Preliminary Results

In a study conducted by Cova et al., high-energy electrons in the 20-200GeV scale from a test beam were shown at silica fibers and the Cerenkov and scintillation response were recorded. This data allows us to answer these important questions: What does a realistic pulse look like? What is the actual ratio of Cerenkov radiation to scintillation radiation? What is the expected photoelectron count? Below is a plot included in this paper.

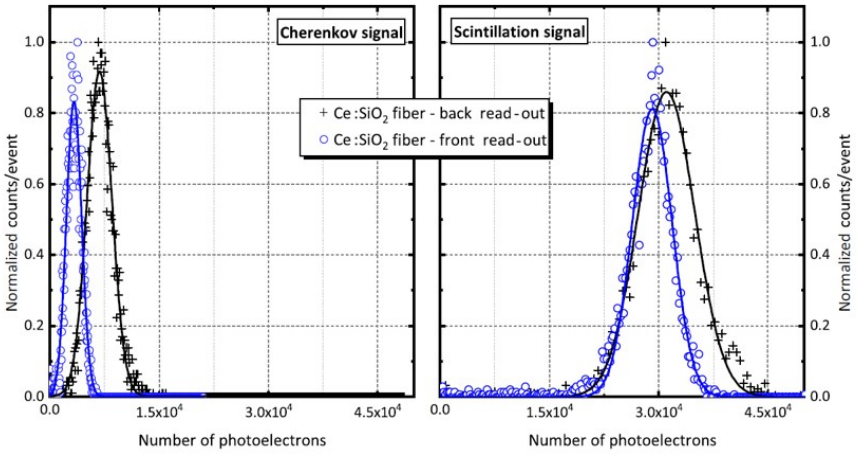

Fig. 8: Reprinted from [3]

As we can see from this figure, we can expect on average 7.5e3 phe for Cerenkov radiation and 30e3 phe for Scintillation radiation. This corresponds to a ratio=-4. Monte-Carlo results done with 5000 phes in this portion of parameter space show a possibility of of less than 1% error with a binning of 100 for the histogram. If our Monte-Carlo simulations are to be believed, we can be hopeful about applying this technique to real data as well.

3.2 Future Work

Unfortunately, not enough time was available to fully investigate the feasibility of this application of machine learning on real data. The authors of the paper mentioned before did share some raw data from the test beams. The obvious next step is to clean this data and generate a a training library with which neural networks can be trained and tested on.

4 Part III: Implementation on FPGAs

As mentioned earlier, all of the networks in this study have three layers, including the input and output layer. Although a few other model architectures with more layers were tested, they did not seem to have any major noticeable improvement on the simple three-layer models. A secondary goal of this project was to maintain a smaller size to our neural network as this would allow the neural network to be uploaded on an FPGA. An FPGA, also known as a Field Programmable Gate Array, is an integrable circuit that can be programmed after manufacturing. If a neural network can be uploaded to an FPGA and perform with low latency, then it could potentially be used to give rough em-fraction predictions to assist L1 triggers.

4.1 hls4ml

hls4ml is a package for machine learning inference in FPGAs [4]. It translate open-source machine learning package models, such as the ones created by Keras, and turns them into High Level Synthesis. HLS is a language that is used by Xilinx, a company that manufactures FPGAs, to program their circuits. This package allows us to rapidly see the resource predictions and latency estimates for the networks that we generate.

4.2 Latency Predictions

The main limitation in latency is the number of multiplication units in the FPGA. If we aim for one clock cycle per analysis, he number of nodes in our neural network needs to be at most the number of multiplication units in the FPGA. Fortunately, current high-end FPGAs have about 6,000 multiplier units and the number of weights in our networks is well below that limit at about 1,032. Below is just a screencap from the output of the HLS project generated by hls4ml when we provided the weights of one of our Keras models as an hdf5 file.

We clearly see that a sub 5ns latency can be expected from our neural network. The LHC has bunch collisions every 25ns. If a future collider has a similar

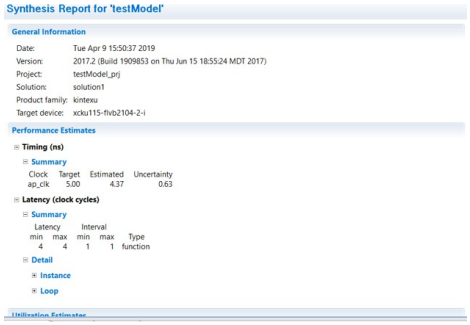

Fig. 9: Latency Predictions from hls4ml

collision frequency, than this FPGA could very easily analyze the output of a single-channel dual-readout calorimeter at a

5 Conclusions and Future Work

In this short technical report, we have shown that using neural networks to perform event-by-event em fraction energy measurement in single-channel dual-readout calorimeters is very possible. In fact, these preliminary results provide a range of parameter space where neural networks are viable within 5% error. This could be used to inform the geometry/construction of the calorimeters and the associated hardware in the future. We have also shown that 'shallow' neural networks with only one hidden layer are adequate in performing this analysis, allowing for low-latency calculations in FPGAs.

There are obviously many avenues for further research. The first obvious step is reproducing similar results on real test beam data. Implementing one of these networks on a real or emulated FPGA would also be useful since the hls4ml results are quite promising.

The codebase and information in this report can be used to continue this analysis to further investigate these and other avenues of future work. If the effectiveness of this computing technique can be shown using real test beam data from prototype calorimeters, one can build a strong argument that dual readout calorimetry should be widely used in the future. As planning starts for the next generation of high energy particle accelerators coming in the next few decades, such as CLIC and the FCC, it is important to utilize calorimetry techniques that are effective at achieving high energy resolutions while also being cost effective. Single-channel dual-readout calorimetry provides us this unique ability to measure hadronic energy with great resolution while also reducing hardware costs by eliminating the need for separate ADC channels for Cerenkov and scintillating fibers. We have shown that Neural Networks could play a vital role in legitimizing single-channel dual-readout calorimetry, in terms of improving hadronic energy resolution, being cost-effective, and meeting latency standards.

6 Acknowledgements

M.S. would like thank F.G. for his patience and encouragement over the course of the few months. M.S. would also like to thank Nural Akchurin as well for his guidance and advice throughout this project. Finally, M.S. would also like to thank the University of Michigan, the UM-CERN Research Abroad Program, and the Richard Lounsebery Foundation for providing funding over the course of this project.

7 References

[1] arXiv:1712.05494 [physics.ins-det]
[2] https://youtu.be/aircAruvnKk
[3] Cova, Francesca, et al. Physical Review Applied 11.2 (2019): 024036.
[4] J. Duarte et al., "Fast inference of deep neural networks in FPGAs for particle physics", JINST 13 P07027 (2018), arXiv:1804.06913.