

**IceCube Final Report**

**PHY/CS Research**

**Summer 2019**

Dr. Frank McNally

 Tyler Sledge

Roy Wood

**8/2/2019**

*Introduction*

Dr. McNally had tasked Tyler Sledge and Roy Wood with an ongoing research project in collaboration with the IceCube Neutrino Observatory. The project involved high-level machine learning programs and high-energy particle showers. The goal of this project was to develop an algorithm, or model, that utilizes high-level machine learning programs, deep neural networks, and convolution to reconstruct certain parameters of a charged particle entering the Earth’s atmosphere.

*IceCube Neutrino Observatory*

The model uses data provided by the IceCube Neutrino Observatory. The data are made up of events; these events are the stimulation of an array of Digital Optical Modules (DOMs) from the Cherenkov Radiation that results from high-energy particles interacting with the surrounding ice. Hence, each event is a map of the charge distribution with respect to the time of occurrence.

*Data Features & Sanitization*

The events were processed to restructure the input data and remove any unnecessary information; this improves the model’s ability to interpret the input data and streamlines data propagation through the model. The data was also sanitized, normalized, and randomized in order to reduce the potential for the model to overfit the data and to improve the learning rate of the model. The effects of data processing are characterized in the *Assessment* section.

From the raw data, two distinct features are extracted: *charge* and *time*, measurements of the strength of the signals and relative occurrence of the signals from the sensor array, respectively. The table contains the potential feature shapes for the event maps.

Table 1: Potential Feature Shapes for Convolutional Neural Network

|  |  |
| --- | --- |
| Feature Shape | Description |
| 1 layer  (station total charge) | The charge of both tanks is summed to produce the station total charge |
| 1 layer  (average tank charge) | The charge of both tanks is summed then divided by two to produce average tank charge |
| 2 layers  (station total charge  and earliest time) | The charge of both tanks is summed to produce the station total charge and the time of the earlier charge deposit is used |

Table 1 Continued: Potential Feature Shapes for Convolutional Neural Network

|  |  |
| --- | --- |
| Feature Shape | Description |
| 2 layers  (average tank charge  and earliest time) | The charge of both tanks is summed then divided by two to produce average tank charge and the time of the earlier charge deposit is used |
| 2 layers  (station total charge  and average time) | The charge of both tanks is summed to produce the station total charge and the time of both charge deposits are summed and divided by two to produce average time |
| 2 layers  (average tank charge  and average time) | The charge of both tanks is summed then divided by two to produce average tank charge and the time of both charge deposits are summed and divided by two to produce average time |
| 2 layers  (charge by tank) | The charge of each tank is kept separate. |
| 4 layers (charge and timing  by tank) | The charge and timing of each tank is kept separate. |
| Additional layers  (pulse or high-level parameters) | The addition of another layer that contains higher-level information, such as pulse width. |

Sanitization is important as to train the model with reliable data. The data from IceCube Neutrino Observatory contained outliers and noise that could negatively influence the efficacy of the model. The sensor array also has physical limitations that causes a bias in the distribution of the data; specifically, the IceCube DOM array does not record a uniform distribution of events at each energy level. A cut was applied to the dataset, similar to [1], in an attempt to isolate reliable data. Reliable data was classified by the following criteria:

Table 2: Pass Criteria for Data Sanitization

|  |  |  |
| --- | --- | --- |
| Criteria | Description | Reasoning |
| Qmax > 6 | The maximum charge during the event was measured to be greater than six muon energy units. | This reduces the likelihood of a false event from signal noise while maintaining a large enough batch size. |
| Qmax NOT on edge | The maximum charge during the event was NOT measured to be on the edge of the event map. | This reduces the likelihood of an event with incomplete data. |

Normalization scales the data between 0 and 1, reducing the load on the machine learning program. Propagations through the neural network have the potential to generate very large numbers from unscaled datasets which are a burden to the computer.

Randomization is beneficial throughout the training of the model. The order of the events is randomized between epochs (the propagation of all events of a batch from the input dataset) in order to reduce the potential that the model overfits the data. Randomizing the data also reduces the likelihood of the model converging to a local minimum as the model has a different sequence of data to learn from.

*Network Architecture*

From the documents recommended by Dr. McNally, the most successful models consisted of convolutional layers followed by hidden dense layers [1][2]. Convolution-based neural networks utilize the relative location of data within the event map to determine a result. The convolutional neural network used by [2] utilized an event map paired with additional shower features to determine the initial energy of a charged particle entering earth’s atmosphere. There is more information available to the team that describes the architecture used by DECO; hence, the team structured the model after DECO’s architecture. The DECO model has an input shape of 64 x 64 array [1]. The input shape that the team will be feeding the model is a 10 x 10 array; hence, the team simplified the architecture in order to scale the architecture to the input data. The baseline architecture used by the team can be seen in Figure 1, on the next page.



Figure 1: Baseline Model in Keras

The baseline consists of two convolution layers with 32 features and a 3x3 kernel window followed by a max pooling layer with a 2x2 window size. Then two more convolution layers with 64 features and a 3x3 kernel window followed by a max pooling layer with a 2x2 window size. After the second max pooling layer, a flatten layer feeds the data into three dense layers with 512 features and ReLU activation. A final dense layer contains one feature which is the model’s prediction of the result.

*Layer Features*

The team implemented the neural network using the high-level machine learning program Keras. Keras allows for greater functionality of the model without the need for the direct knowledge or understanding of the inner workings of the program. The model is able to yield a prediction either as a categoric or regression result. To be categoric, the result consists of a probability distribution of a set number of potential outputs; conversely, to be regression, the results consists of a single value that gets closer to the specified output. The model chosen by the team is a regression model. The following section outlies the additions or modifications employed to better the efficacy of the model.

Batch Normalization, as per [3], serves to improve the rate at which the neural network is able to learn without risking divergence by reducing the internal covariate shift of the data. This concept has known since 1998 [3]. Internal covariate shift is defined as “the change in the distributions of internal nodes of a deep network, in the course of training” [3]. Batch Normalization is accomplished by fixing both the mean and the variance of data. According to [3], Batch Normalization also has beneficial effects on gradient flow through a neural network; it “reduc[es] the dependence of gradients on the scale of the parameters or of their initial values.” Out of my scope, normalization also allows the “use [of] saturating nonlinearities by preventing the network from getting stuck in the saturated modes” [3]. This normalization technique can be implemented at each activation layer in order to reduce the internal covariate shift of the data. From [3], as data propagates through a neural network, the distribution of internal nodes can become biased and cause the variance of the data to shift unfavorably causing extended learning times for the machine learning program; normalizing the internal nodes and their distribution maintains desirable parameters for data propagation [3]. The data provided by the IceCube Neutrino Observatory consists of hundreds-of-thousands of events. The most effective process would use the statistics of the entire population to generate the parameters for normalization; however, the process would be very computationally expensive to implement. To overcome this, an estimate of the population will be used based on the statistics of the batch.

Dropout helps reduce the likelihood that model overfits the data. According to [4], Dropout drops a random selection of nodes from a layer in the model. Dropout should be implemented at each layer of the model. Utilizing Dropout yields dramatic improvement for all types of architectures tested in [4], “without using hyperparameters that were tuned specifically for each architecture” [4].

LeakyReLU is the activation type for all layers in the DECO model with the exception of the final layer which is softmax [1]. The team chose to implement LeakyReLU on all the layers for the model. LeakyReLU outputs between a set, negative value and one [1].

*Model Callbacks*

To improve the user experience of the model during training, the team incorporated checkpoint and early stop callbacks. Checkpoints allow the team to save the model at the end of each epoch; this improves the user experience as during a malfunction the model can be continued from where it left off instead of starting over. Early Stop is an important feature as it checks for an improvement in a specified parameter at the end of each epoch, and if the model does not improve, it stops and saves. Early Stop can also be modified to include a patience parameter that checks for an improvement over the course of a defined number of epochs before it stops and saves the model.

*List of Tunable Model Parameters*

The following table contains the list of all tunable parameters within the model and their function.

Table 3: Tunable Parameters of the Model

|  |  |
| --- | --- |
| Parameter | Description |
| # of Convolution Layers | The number of convolutional layers in the model before the dense layers. |
| # of Convolutions per Layer | The number of convolutional layers before a max pooling layer is used. |
| Convolution Feature Count | The number of feature maps per convolutional layer. |
| Convolution Kernel Shape | The shape of the convolutional window used to process the input. |
| Convolution Padding Type | The amount of padding added to the input to manipulate the output shape of the layer. |
| Convolution Bias | Whether bias is implemented on a convolutional layer. |
| # of Dense Layers | The number of dense layers used after the convolutional layers. |
| Dense Feature Count | The number of nodes at a dense layer. |
| Dense Bias | Whether bias is implemented on a dense layer. |

Table 3 Continued: Tunable Parameters of the Model

|  |  |
| --- | --- |
| Activation Layer Type | The type of activation associated with a convolutional or dense layer. |
| Activation Parameter(s)  (based on Activation Layer Type) | Adjusting the shape of the activation function of the activation layer. |
| Max Pooling Shape | The shape of the pooling window used to process the input. |
| Dropout Percentage | Adjusting the percentage of nodes that are neglected at the dropout layer. |
| Model Optimizer Type | The method of reducing loss and training the model. |
| Model Optimizer Parameter(s)  (based on Model Optimizer Type) | Adjusting the behavior of the optimizer type. |
| Model Loss Metric | The measurement of how good or how bad the model is doing. |
| Early-Stop Callback Patience Parameter | The number of epochs to stop if improvement not shown. |
| Data Quality Qmax Cut | Any event map with a maximum charge below 6 muon energy units is discarded. |
| Data Quality Loudest-Station  on Edge Cut | Any event map with a maximum charge along the outer-most edges is discarded. |

*Assessment and Results of Neural Network*

The following section shows the results of each model and comparisons between the baseline model and different variations of the model. The models are analyzed by plotting their error distribution in both one-dimension, where the error is independent of the energy, and two-dimensions, where the error is dependent on the energy. The results of each model contain the mean squared error and mean error of both the training and testing data.

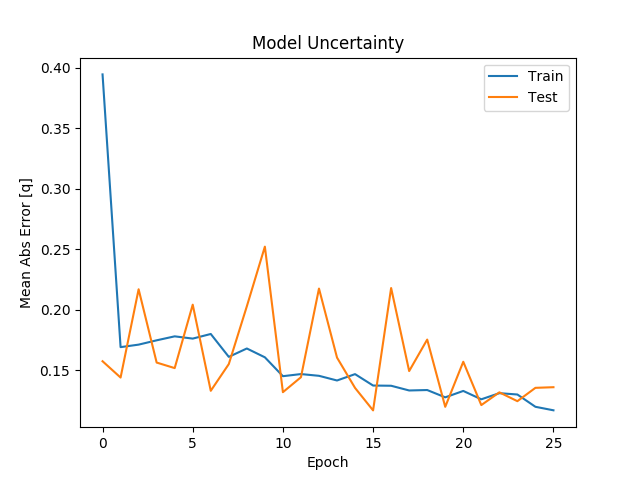
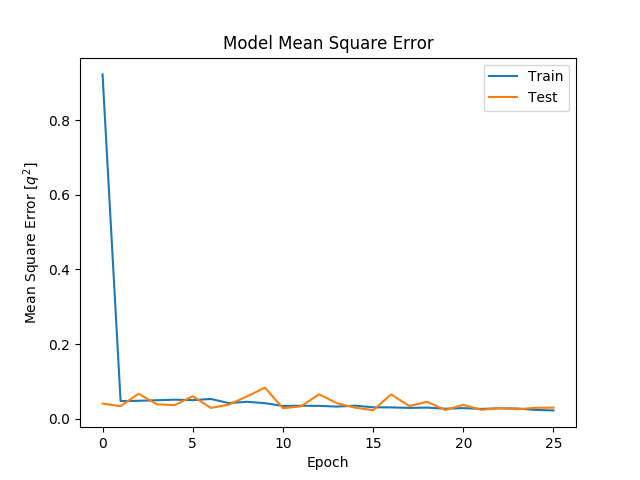


Figure 2: Baseline

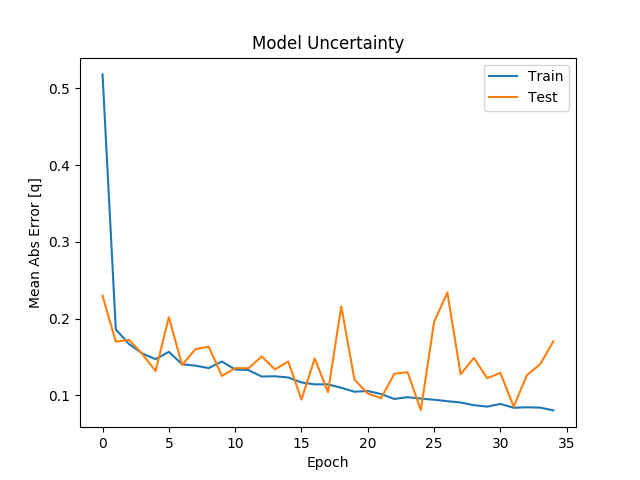
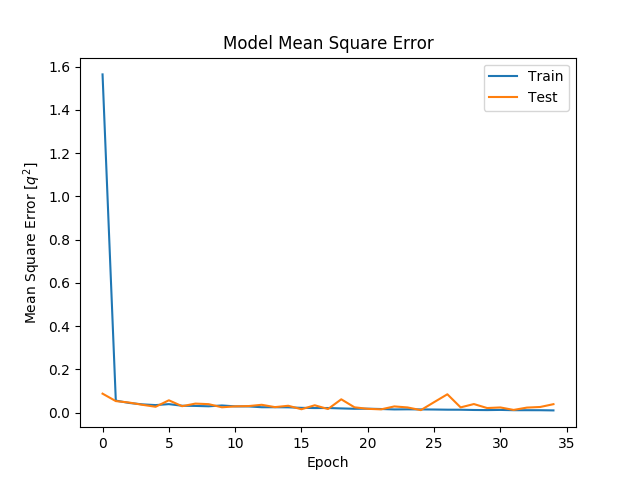


Figure 3: Baseline w/ Batch Normalization

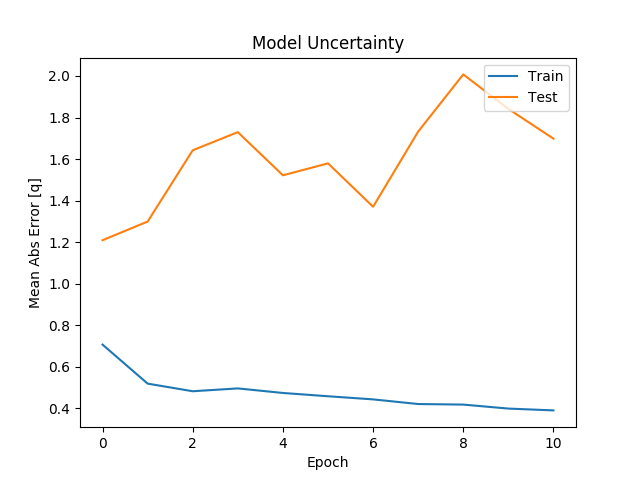


Figure 4: Baseline w/ Dropout

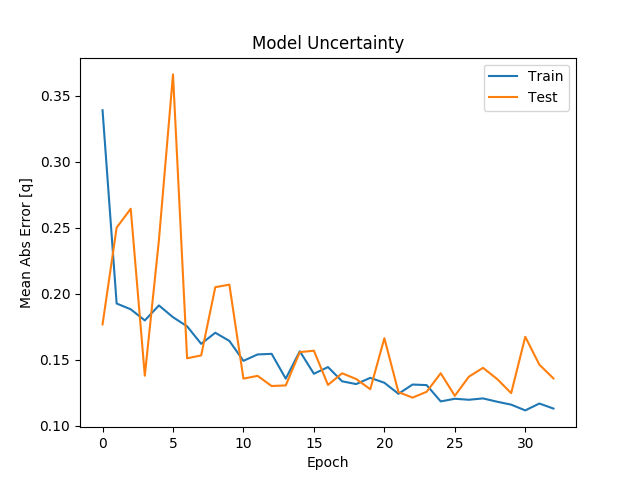
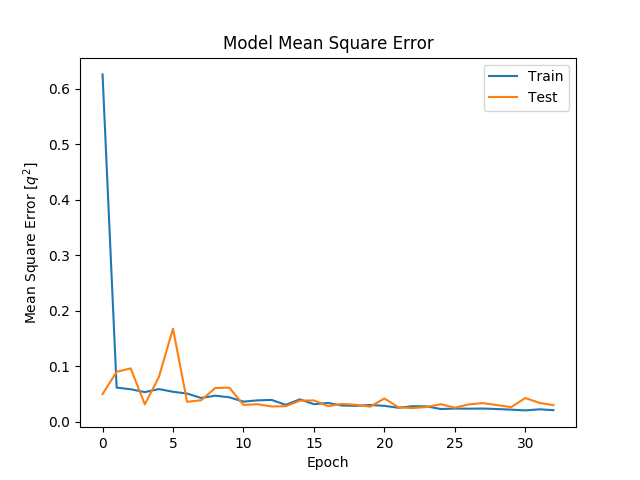


Figure 5: Baseline w/ LeakyReLU

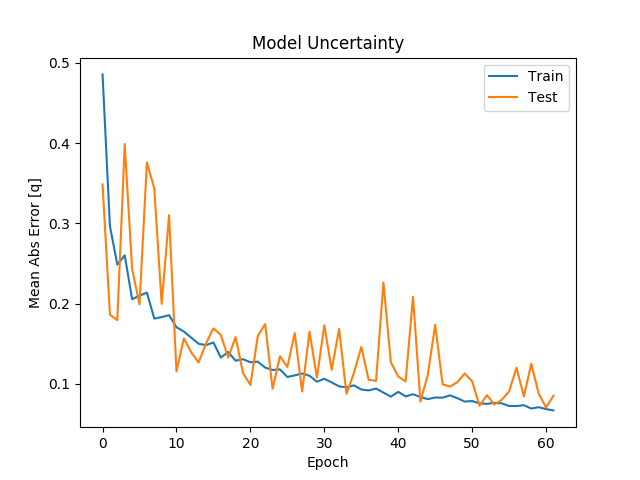
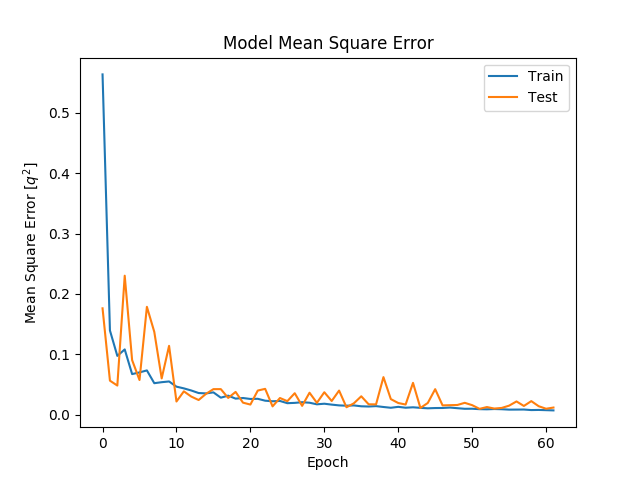


Figure 6: Baseline w/ Batch Normalization and LeakyReLU

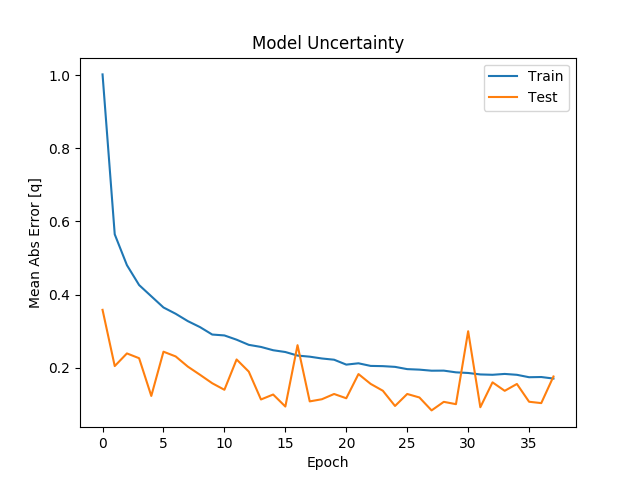
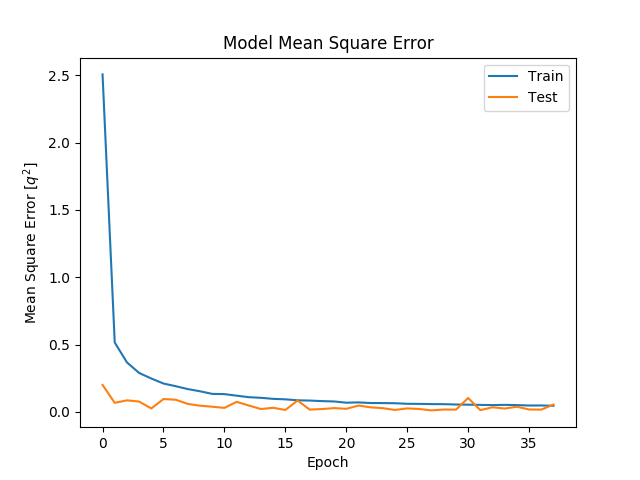


Figure 7: Baseline w/ Batch Normalization, Dropout, and LeakyReLU

The one-dimensional plot of error distribution illustrates the tendency of the model overestimate or underestimate the energy of the event. The optimal shape of the distribution is a tall peak with symmetric, quickly decaying tails on both sides; this shows the model is not biased toward over/under-estimating the energy.

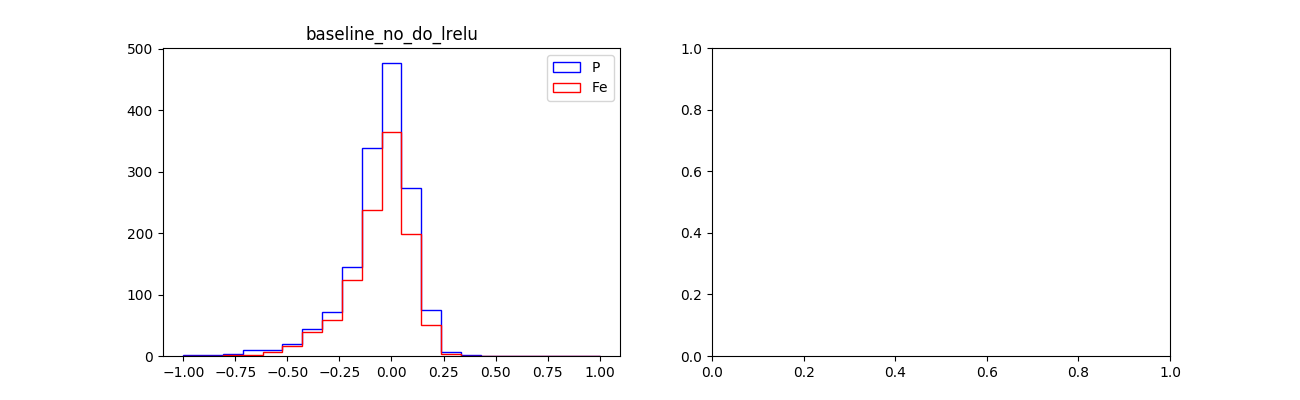
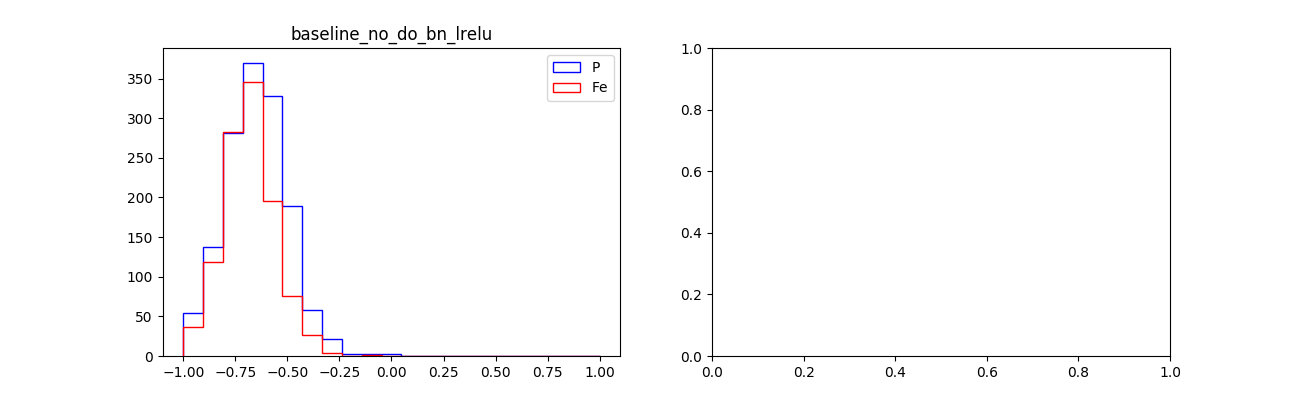


Figure 8: Baseline Figure 9: Baseline w/ Batch Normalization

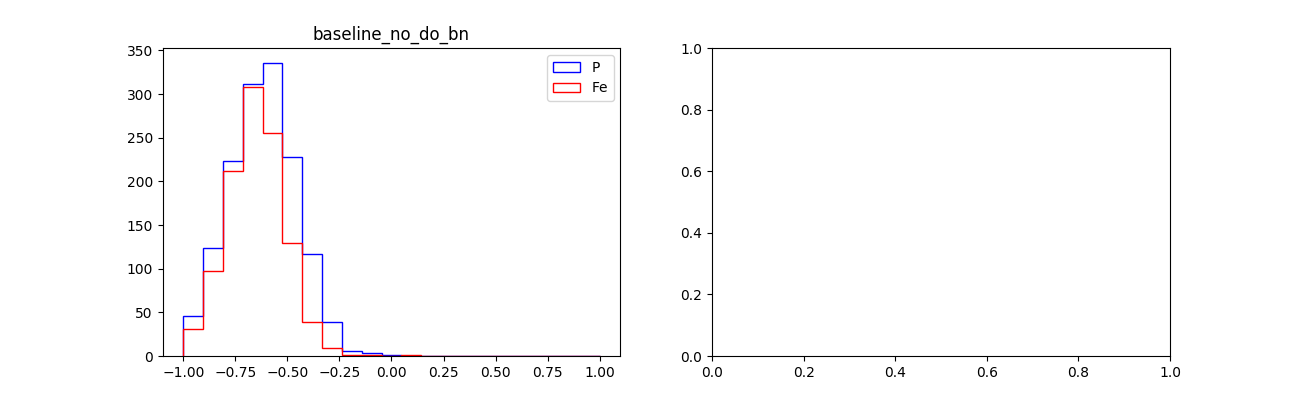
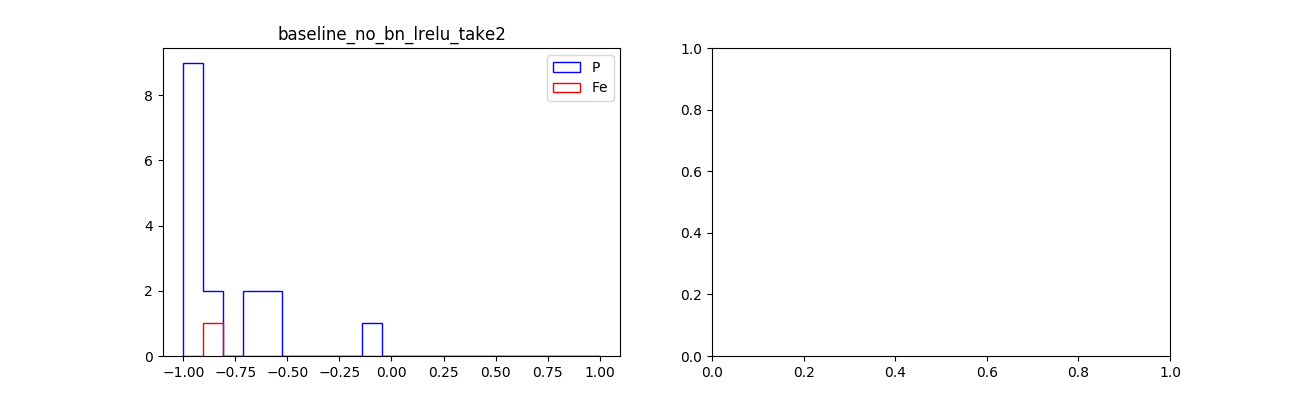


Figure 10: Baseline w/ Dropout Figure 11: Baseline w/ LeakyReLU

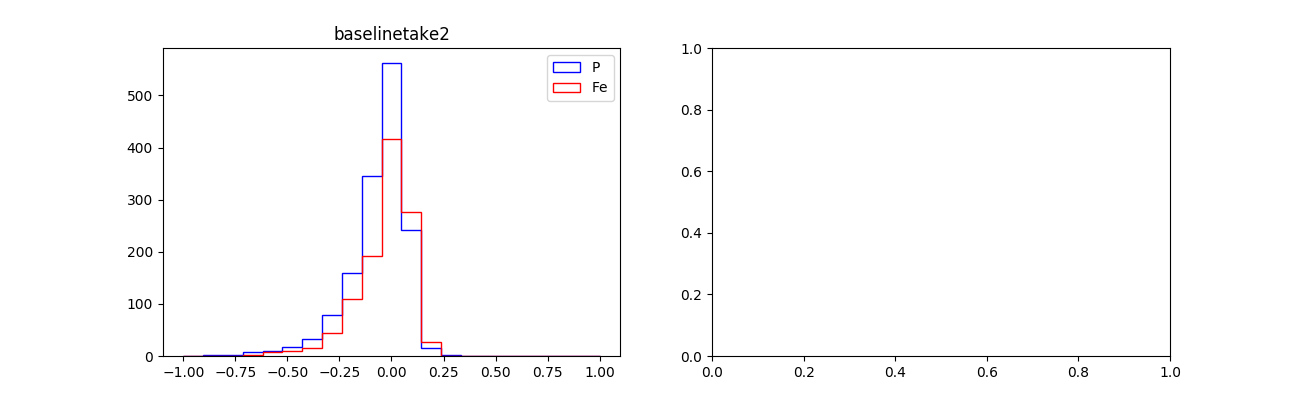
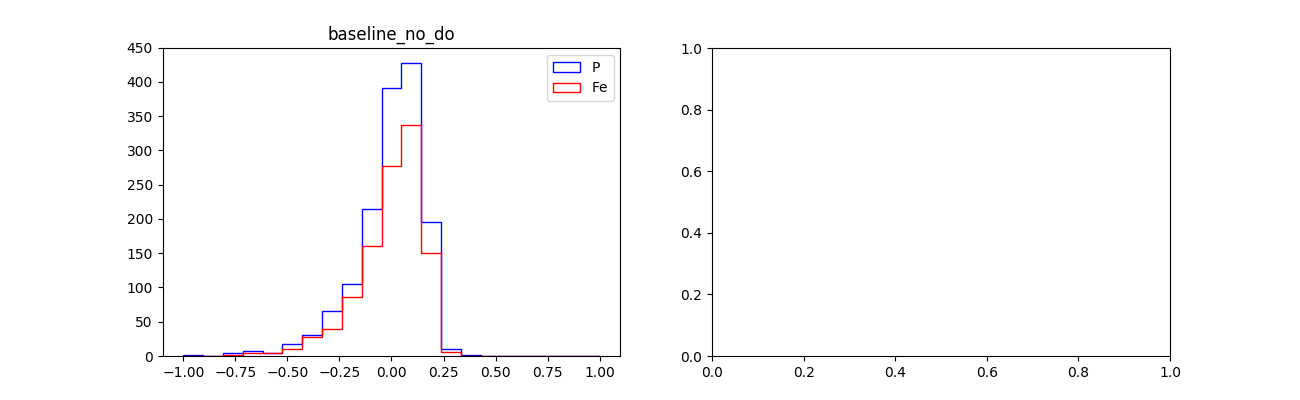


Figure 12: Baseline w/ BN & LeakyReLU Figure 13: Baseline w/ BN, DO, & LeakyReLU

The two-dimensional plot of error distribution gives insight into the model’s dependency on the energy. The optimal shape of the distribution would be a solid yellow line from the bottom left corner to top right corner of the plot, realistically, with two dark-blue lines running immediately parallel to the yellow line; this indicates that the model is able to accurately estimate the energy, independent of the energy level.

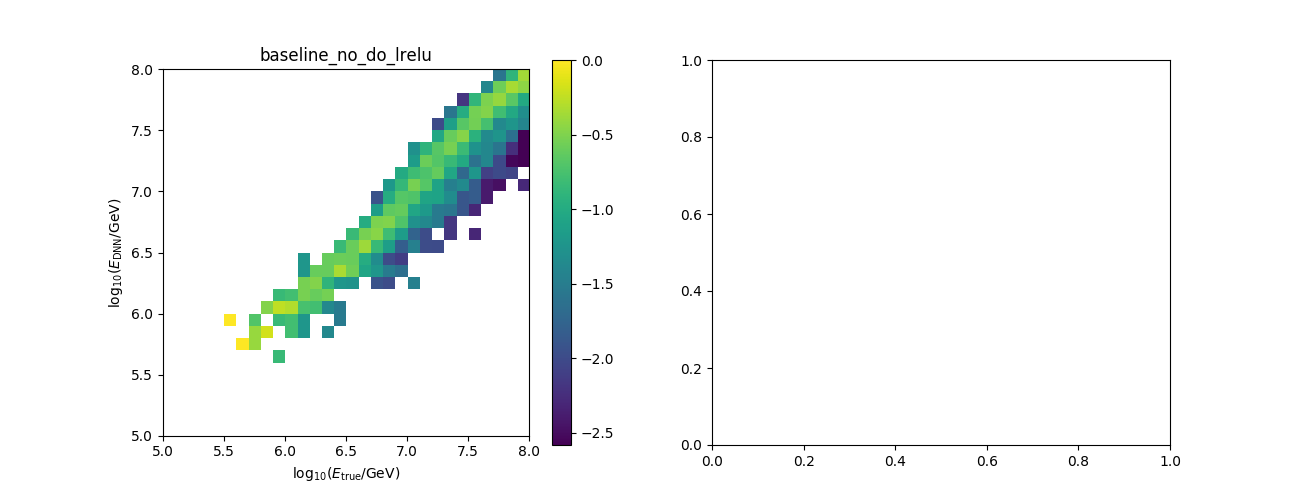
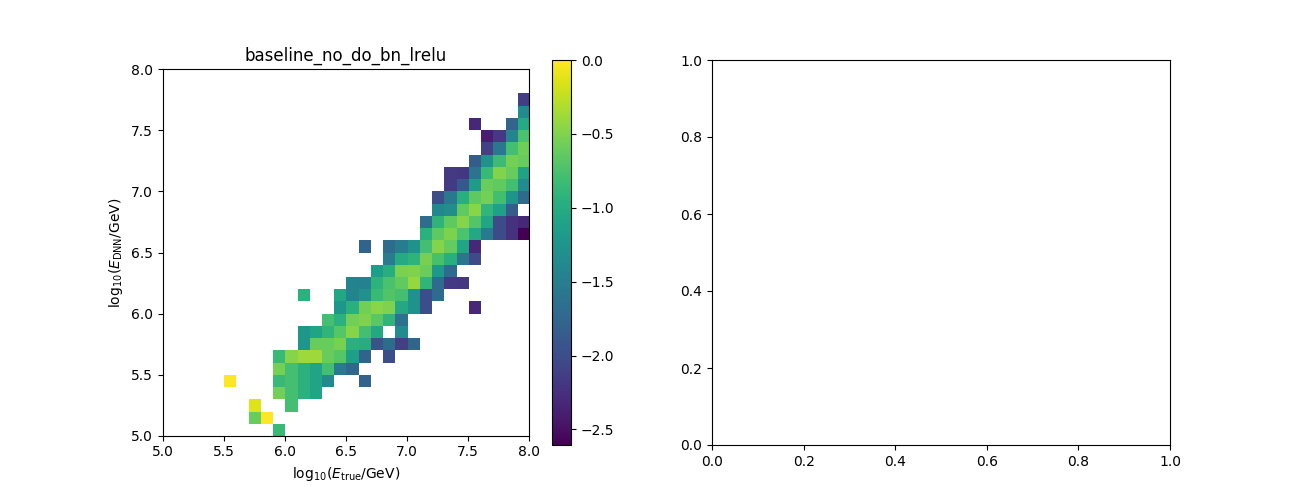


Figure 14: Baseline Figure 15: Baseline w/ Batch Normalization

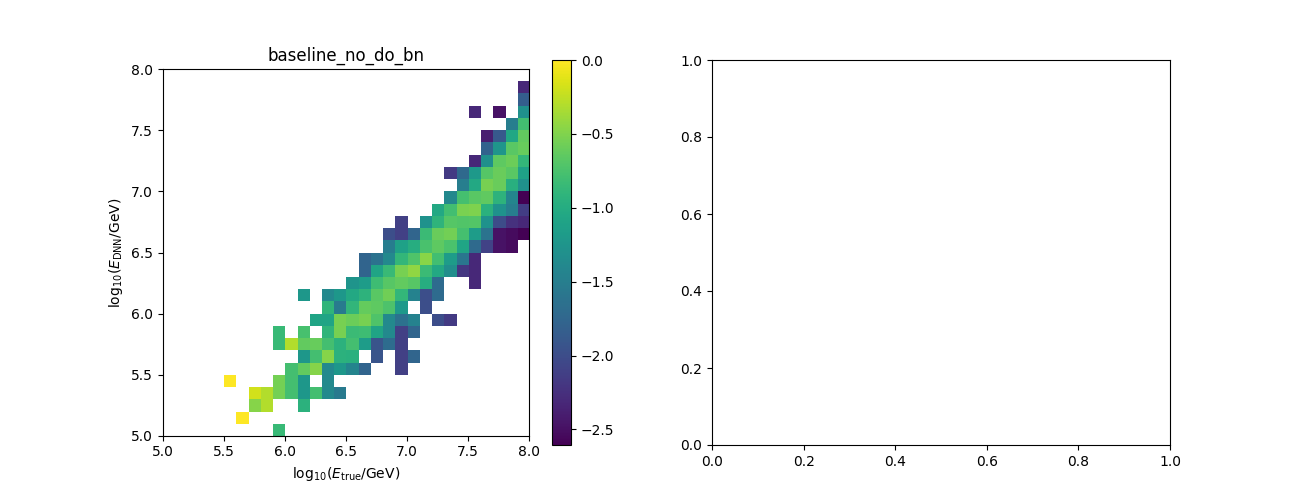
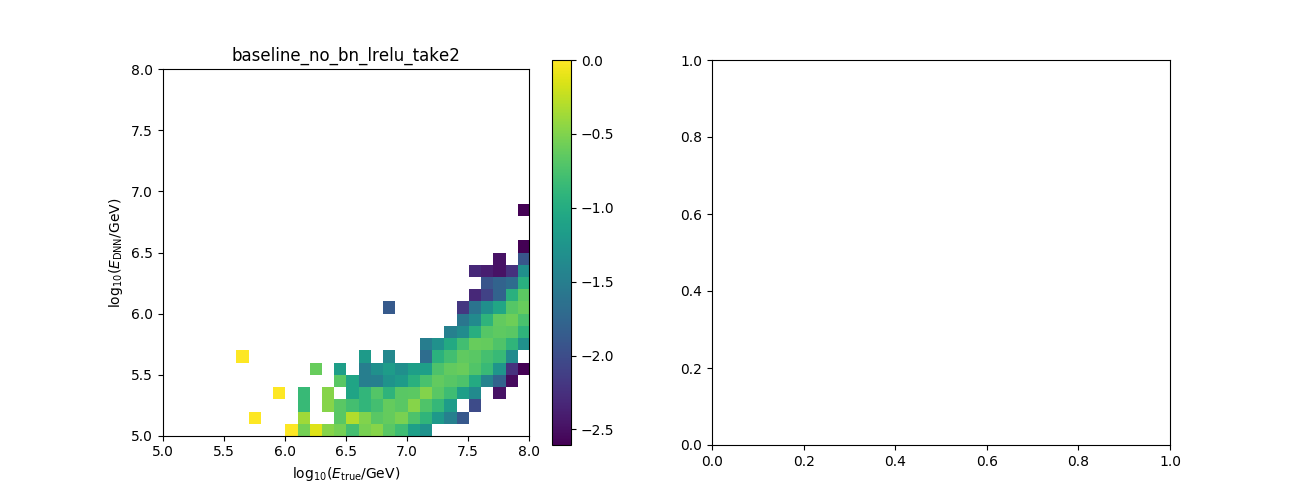


Figure 16: Baseline w/ Dropout Figure 17: Baseline w/ LeakyReLU

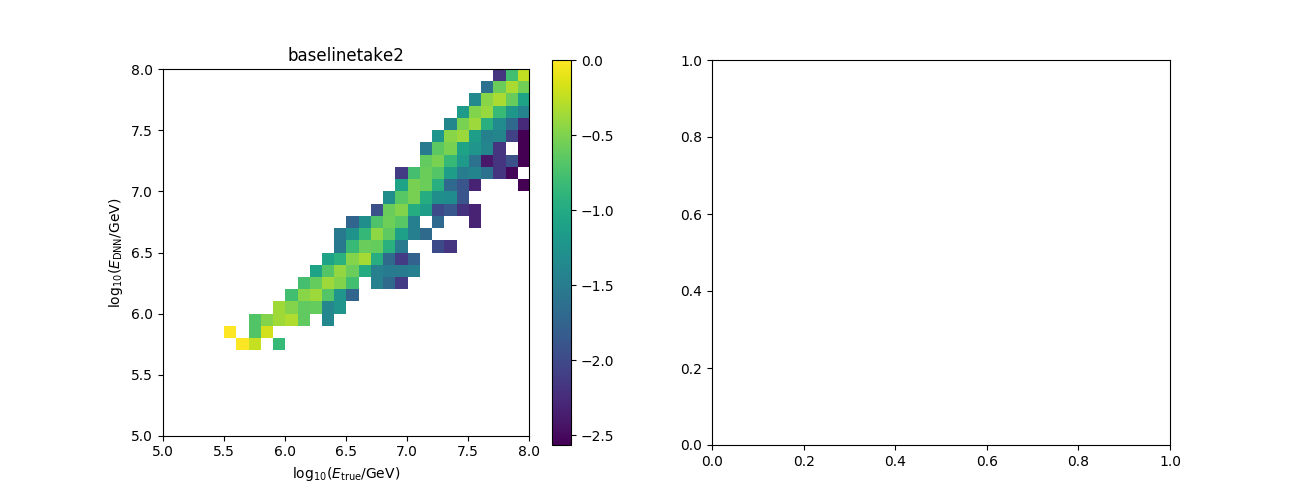
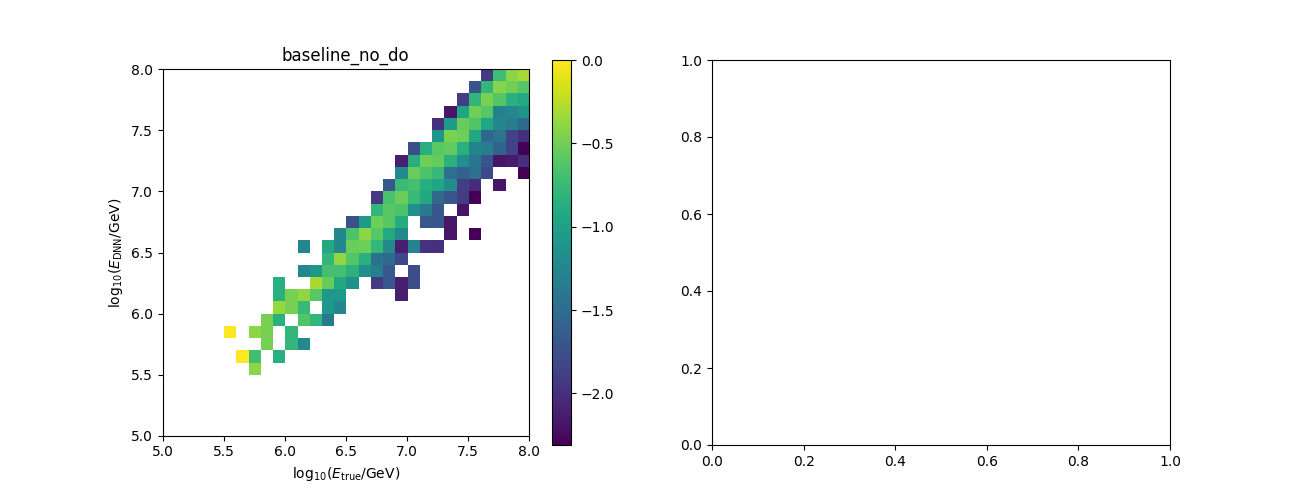


Figure 18: Baseline w/ BN & LeakyReLU Figure 19: Baseline w/ BN, DO, & LeakyReLU

*Potential Improvements*

The model has not been exhausted of potential improvements that could be made. The team recommends looking into the following possible changes for the model: One, including high-level event parameters such as event pulse shape parameters (Time to Peak, Time to 50%), S125, shower center of gravity, shower plane fit, or snow depth at tank. Two, revisiting the feature shapes for the event map found in Table 1. Three, incorporating more outputs, like direction or composition of particle. Four, reevaluating the hexagon to square grid transformation versus hexagonal shaped convolution kernels versus a flexible tank grid. Five, process the input dataset to include rotational copies and/or mirrors of measured events to induce a rotationally/symmetrically invariant model.

*Conclusion*

In summation, the performance of the baseline model improved with the addition of batch normalization and implementation of LeakyReLU, independently; however, when the baseline model was implemented with dropout the model did not converge within 10 epochs. The best results occurred when batch normalization, dropout, and LeakyReLU were all integrated into the baseline model. Although dropout appears to worsen the performance of model alone, dropout is a valuable tool when used in conjunction with batch normalization and LeakyReLU. Two major conflicts are that the loss metrics are viewed in normalized/standardized form that can potentially change between batches of different input datasets and that the training of this specific model was a batch of about twenty-thousand events, after the quality cut is applied.

*Works Cited*

[1] Miles Winter, James Bourbeau, Silvia Bravo, Felipe Campos, Matthew Meehan, Jeffrey Peacock, Tyler Ruggles, Cassidy Schneider, Ariel Levi Simons: “Particle Identification In Camera Image Sensors Using Computer Vision”, 2018; [http://arxiv.org/abs/1803.04493 arXiv:1803.04493]. DOI: [https://dx.doi.org/10.1016/j.astropartphys.2018.08.009 10.1016/j.astropartphys.2018.08.009].

[2] IceCube Collaboration: “The IceCube Neutrino Observatory - Contributions to ICRC 2017 Part V: Solar flares, Supernovae, Event reconstruction, Education & Outreach”, 2017; [http://arxiv.org/abs/1710.01201 arXiv:1710.01201].

[3] Sergey Ioffe: “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, 2015; [http://arxiv.org/abs/1502.03167 arXiv:1502.03167].

[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov: “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, 2014; [http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf].