A Systematics Analysis of CVN for Neutrino Event Classification

A thesis submitted in partial fulfillment of the requirement
for the degree of Bachelor of Science in
Physics from the College of William and Mary in Virginia,

by

Isabela Gnasso
Advisor: Dr. Tricia Vahle

Williamsburg, Virginia
May 2020

Contents

Acknowledgments

I would like to thank Dr. Tricia Vahle, her wonderful research group at W&M, and all of the scientists at NOvA who helped me with this project. I would especially like to thank Dr. Vahle for devoting her time in taking on this project with me, and for challenging me to accomplish things I did not know I could do. I am so grateful to have had the opportunity to work closely with an advisor who is such an incredible role model, as a scientist and overall as a person.

List of Figures

List of Tables

Public Abstract

A Systematics Analysis of CVN for Neutrino Event Classification

Isabela Gnasso

Dr. Tricia Vahle

May 2020

Introduction and Background

The bulk of this project involved learning to train a version of the Convolutional Visual Network (CVN) created by scientists at NOvA for neutrino event classification. The goal was to test the CVN for sensitivities to systematic uncertainties in the simulated data it was trained on.

Summary of Results

The comparison between the nominal trainings and the weighted training indicated that the CVN may be affected by systematic uncertainties in the simulated data. The weighted run resulted an ν_e event classification distribution that was discrepant from the mean of the nominal runs.

Intellectual Merit

Applying deep learning techniques to High Energy Physics (HEP) has provided oppor to analyze and classify data with higher efficiency and purity than before. The CVN's ability to process enormous visual datasets makes it a very suitable tool for classifying particle interactions.

Broader Impacts

This project involves the culmination of machine learning techniques and HEP, there are broader impacts in both areas. For example, CNN's are being used for the development of fully automated vehicles, medical diagnoses, stock market prediction, entertainment, and security. The broader impact of the physics side of this project relates to our understanding of the origins and behavior of the Universe.

Abstract

With the rapid advancement in machine learning over the last two decades, Convolutional Neural Networks (CNN's) are being implemented in many industries to solve complex image recognition problems. In the context of this project, their image classification capabilities make them very useful for the large and information-dense datasets in High Energy Physics (HEP). This project works with a CNN called the Convolutional Visual Network (CVN), created for the classification of neutrino events measured by the NOvA experiment. Scientists at NOvA have succeeded in training their CVN to classify simulated data with sufficient accuracy. The ultimate goal of this project was to test how the CVN might be affected by systematic uncertainties in simulation data, in order to to evaluate its reliability when classifying measured events. To do this, I trained a verision of the CVN with simulated data from a nominal training sample. Once this baseline performance of the model was established with five nominal runs, I trained a model on a dataset that had a specific event type removed. Results showed that the weighted model produced a different ν_e event classification distribution compared to the mean of the nominal models. It was also found that the model performed with higher efficiency in ν_e event classifications after the cut was applied to the data. In short, these results provide evidence that the model is sensitive to systematic uncertainties in the training.

Chapter 1

The NOvA Experiment

1.1 Introduction

When the NOvA (NuMI Off-axis ν_e Appearance) experiment became operational in 2014, one of its primary goals was to observe muon-neutrino (ν_μ) to electron-neutrino (ν_e) oscillations in order to obtain a more complete understanding of the mechanisms that govern the behavior of these particles. Scientists at NOvA created their own Convolutional Visual Network (CVN) to distinguish between classes of neutrino events to recognize oscillations. After training with a hybrid of real and simulated data, the CVN flavor classifications enabled the identification of ν_μ disappearance oscillations and ν_e appearance oscillations [4] [14].

1.2 Neutrino Oscillations

One significant motivation for particle physics in general is understanding behaviors and interactions between fundamental particles in order to explain the matter that they comprise in the Universe. Developments in neutrino physics are bringing scientists closer to obtaining a full picture of these behaviors, but there is still much unknown about these particles. With the discovery that neutrinos can oscillate between flavors came the proof that these particles must have mass. Neutrino

oscillations involve the mixing of three possible mass states and flavor states. The flavor states that neutrinos can oscillate back and forth between are ν_e, ν_μ, and tau (ν_τ), corresponding to the charged leptons they are associated with. This experiment involves ν_μ's that are produced through pion decay [1] [17] [9].

1.3 Experimental Setup

Since neutrinos have no charge and they rarely interact with matter, their direction of travel cannot be influenced easily. However, there are charged particles that can be directed magnetically before they decay into these leptons. This is how neutrinos are steered and focused in the NuMI (Neutrinos at the Main Injector) beam, illustrated in Fig. 1.1. This beam is produced as protons are fired at a graphite target at Fermilab. The collisions of protons with the graphite produces pions, which can then be steered with magnets into a collected beam [11]. These pions eventually decay via the weak interaction into muons and ν_μ [6].

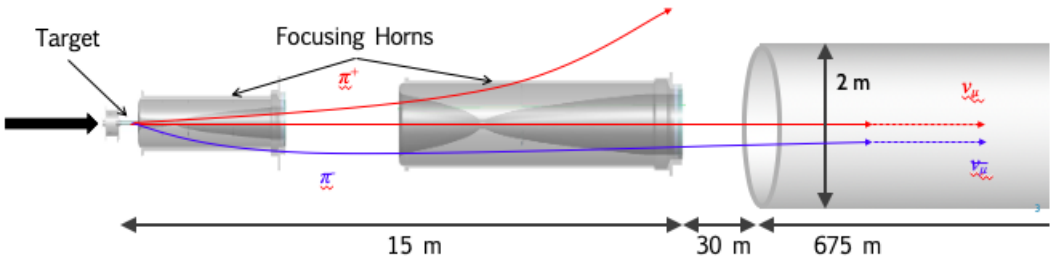

Figure 1.1: NuMI Horn. The NuMI Horn is the device used to direct and focus a beam that eventually decays into ν_μ [22].

This beam diverges as it travels, and is studied in two detectors, the Near Detector (ND) and the Far Detector (FD), which are shown in Figures 1.2 and 1.3 [11]. The detectors are oriented off-center within the halo so that they can receive a flux of neutrinos at an energy level of 2 GeV, where the $\nu_\mu \rightarrow \nu_e$ oscillations are expected

to be highest. The ND, located 1 km from the graphite target, weighs 290 tons and has dimensions of 3.8m x 3.8m x 12.8m. The FD is 810 km away from the target in Ash River, Minnesota, weighing 14 kton with dimensions of 15m x 15m x 60m. Oscillation measurements are conducted by comparing neutrino interactions in both of these detectors [4] [11].

Figure 1.2: Near Detector. NOvA's ND, located at Fermilab [8].

Neutrino interactions are recorded as they occur across the 344,064 liquid scintillator-filled PVC cells in the detectors. The cells have dimensions of 6.6 cm x 3.9 cm and their lengths alternately cover the height and width of either the ND or FD. These cells are arranged to measure two views of all events, across X-Z and Y-Z planes. As neutrinos pass through the cells of the detectors, they may interact with a nucleus of an atom in the liquid scintillator or the surrounding plastic, emitting energy that

Event	Associated Particles	Topology
ν_μ CC	μ^- + hadron	long track with low, steady energy deposition
ν_e CC	e^- + hadron	shower with wide scattering
ν NC	ν + hadron	neutrino produced passes by undetected, so flavor is uknown

Table 1.1: Event Topologies. This table provides a general summary of the spatial patterns and particles used to identify types of neutrino interactions measured by the ND and FD detectors.

is observed as a pattern of light in the fibers connected to photo-detectors in each cell [11]. This process is illustrated in Fig. 1.4. The spatial patterns associated with different types of neutrino interactions, exemplified by Fig. 1.5, are used to categorize these events. The only measurable category of events is charged-current (CC), which are marked by the charged leptons that can be seen at each interaction. Of the CC events, NOvA only measures ν_μ and ν_e interactions. The other category of events, neutral-current (NC), are characterized by a remaining neutrino which moves on undetected. Table 1.1 provides a short summary of these patterns. [11] [6] [4].

Machine learning becomes incredibly useful after interactions have been recorded. The data collected by the detectors becomes a compatible input for a CNN that uses image recognition [4]. A brief summary of CNN's is provided below before discussing NOvA's CVN.

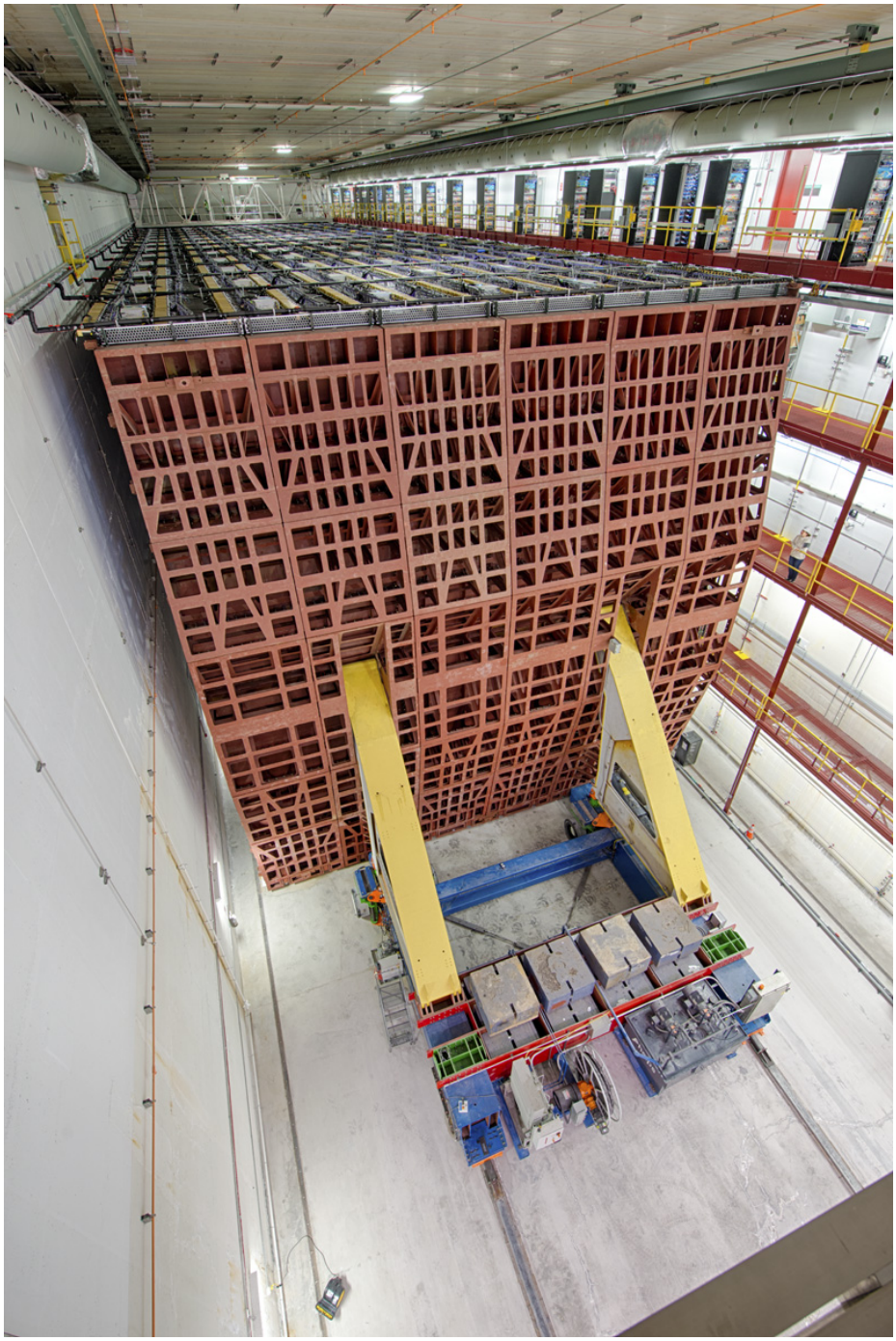

Figure 1.3: Far Detector. This figure demonstrates the immense scale of the FD [7].

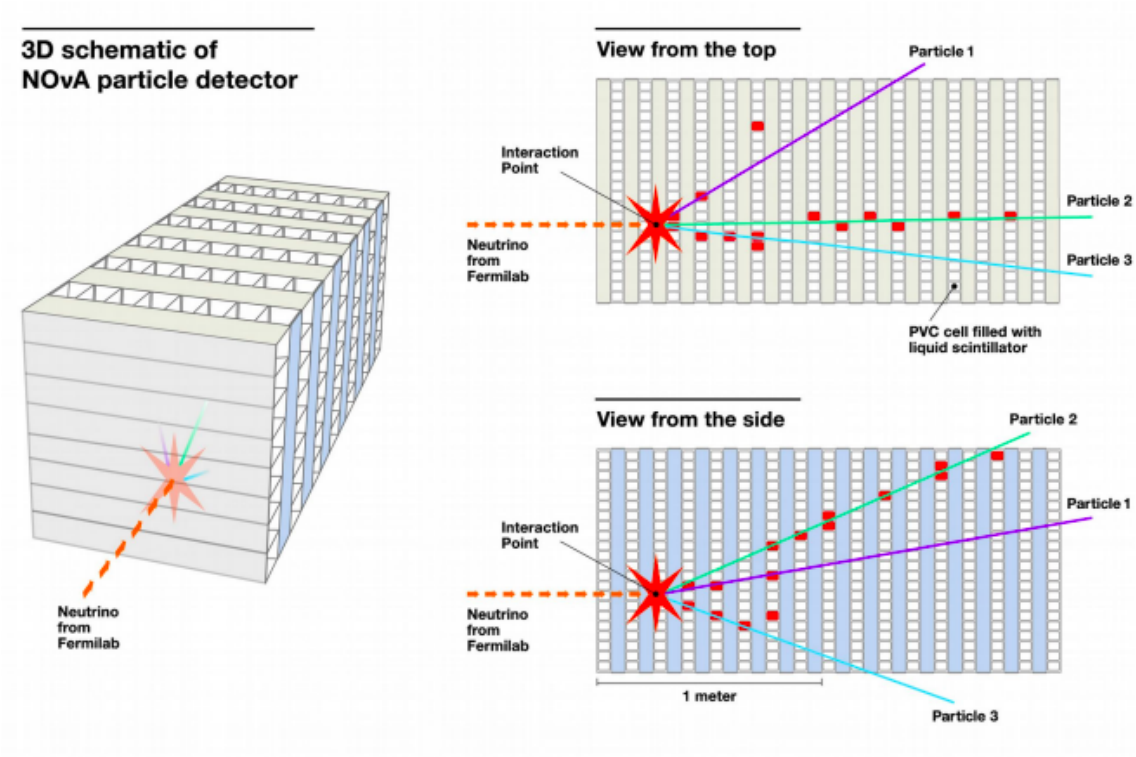

Figure 1.4: Detector Illustration. This figure illustrates the three dimensional layout of the NOvA detectors. Events are recorded as they leave traces of energy throughout extruded cells that fill the structure of the detectors, leaving a topology unique to different types of events [4].

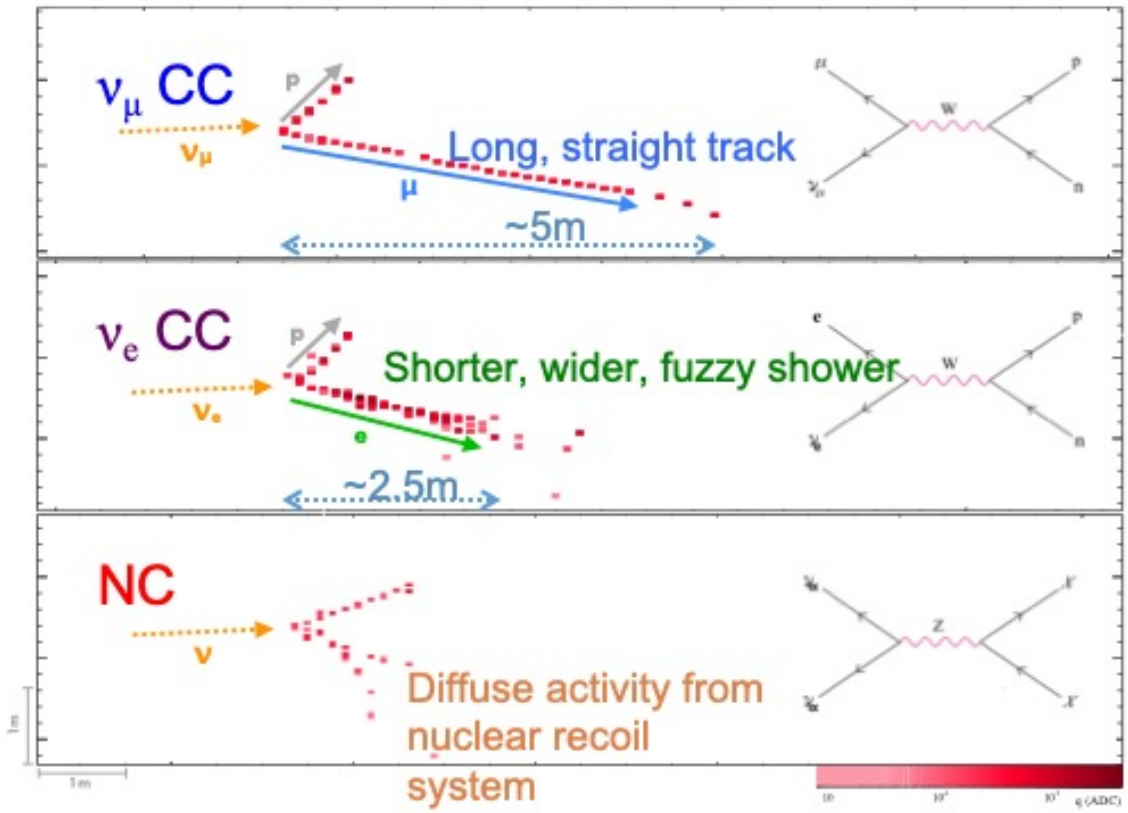

Figure 1.5: Flavor Topologies. This figure contains examples of ν_μ CC, ν_e CC, and NC patterns produced by possible interactions with a nucleus in cells of the detectors [22].

Chapter 2

Neural Networks

2.1 Introduction

As the name suggests, a machine learning algorithm can progressively "learn" to improve the accuracy of its predictions or classifications based on feedback that it receives through many iterations of training. Neural networks are one example of Machine Learning algorithms that are changing the way humans and artificial intelligence interact with the world. In the context of this project, they are crucial in allowing scientists to efficiently analyze patterns in large datasets. They are time saving, customizable, and can handle problems beyond the scope of what human senses are able to do [10].

2.2 Basic Structure

The inspiration for the structures that make up neural networks comes from the way information is processed in biological cognition [20]. In a neural network, there are synapses that carry inputs to the neuron(s). These neurons make up the hidden layers of the neural network. The hidden layers can be defined as intermediary components of the network, between the input and output layers. Each neuron contains an activation function which allows for the modeling of systems with complicated

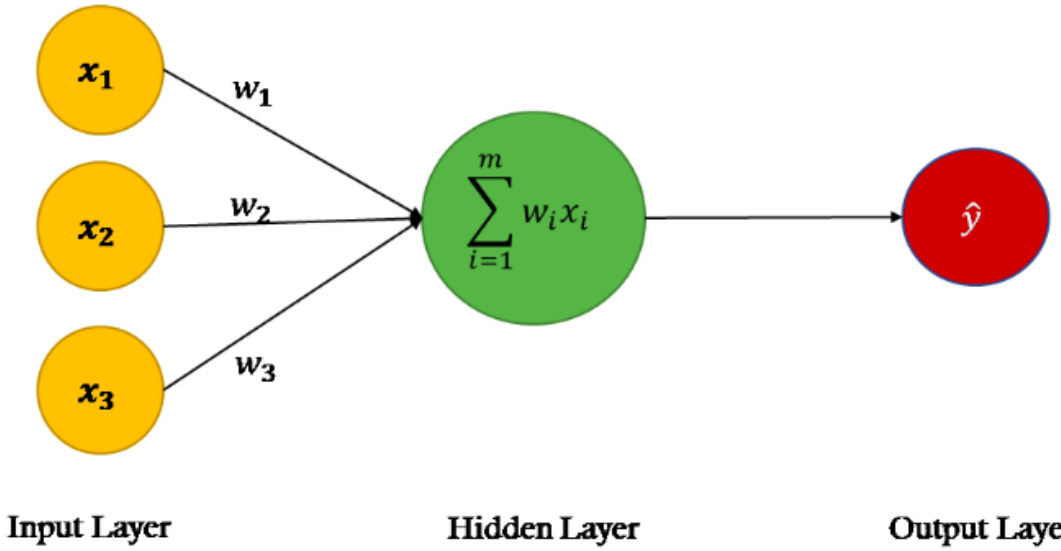

Figure 2.1: Neural Network. Above is an illustration of fundamental components of the neural network structure. It includes an input layer, which passes information along weighted synapses to the hidden layer containing the neuron. Within the neuron, an activation function is applied and the weighted sum of the input is then passed to the output layer. Any number of hidden layers containing a variety of neurons can be combined to create deep learning models [15].

input-output relationships. After the algorithms within the neurons are executed, the result is passed to the output layer [20] [10].

2.3 Convolutional Neural Networks

A CNN is a neural network that performs a convolution before or after at least one of its hidden layers. Essentially, the convolution process involves taking an image and converting it into a numerical representation of lower dimension. More specifically, a convolution is a linear transformation performed on two functions. This linear transformation can be represented by the application of a filter of specified dimensions over an image. As this filter moves a pre-specified number of steps, called the stride, across cells in the grid, it compiles the data into a new cell of what is

called the activation layer. Each filter can be designed to focus on a particular aspect of the input data, and weights can be chosen to emphasize the importance of certain features over others, to create a prediction or classification based on specific quantities of interest in the input data. After the entire image has been transformed into a new, lower-dimension grid called a feature map in the activation layer, it will get passed through the activation function of the neuron. This process can be repeated for any desired number of iterations, ideally stopping when the accuracy of the algorithm is optimized [20].

2.4 Training

Training a neural network begins with randomly assigned weights. After each iteration through rows of data, the error associated with the output is calculated based on known true values, and weights are redefined in an effort to optimize model parameters. Models are initially trained with data that already has a known classification, so that the performance of the algorithm can be assessed after every iteration of training. Training a neural network to be robust and generalizable to adapt to a wide variety of data often requires much repetition before optimal parameters are identified [12].

2.4.1 Back Propagation and Stochastic Gradient Descent

Stochastic gradient descent is a common method used to optimize a neural network when the best model parameters are not predetermined. Steps involved in this process include the following: Running the model from input layer to output, calculating a cost function, running through the network with back propagation, and reassigning weights. The cost function is calculated based on the error associated with the model's output. Then, stochastic gradient descent uses back propagation

to run through the network in reverse, changing all of the weights in an effort to improve results during the next iteration. Calculated values of the cost function associated with different weight coefficients are plotted, and stochastic gradient descent essentially works by "stepping" along the cost function in the direction of a downward slope. The amount that the descent steps in each repetition is called the learning rate. As the descent reaches a minimum where few changes occur between cost function values, it converges at the optimal parameters. Similarly, if the cost function reaches zero, this represents a place where parameters are producing a completely accurate result. The cost function calculations at different parameter values are represented in the 3D cost surface shown in Fig 2.2, and optimal parameters exist where the surface is at a minimum. One drawback of stochastic gradient descent is that it can get stuck in local minima rather than finding the true minimum. Stochastic descent has customizable hyperparameters, or features of the model that can be chosen by the user, that include the learning rate, batch size, and number of epochs [5] [19] [20].

The learning rate, as previously mentioned, is associated with the distance the gradient descent will cover from one iteration through the data to the next. It relates to how quickly the model will reach convergence. The batch parameter represents how many rows of data the model will run through before making a cost function calculation. In stochastic gradient descent, the batch size is traditionally one training sample. This means that the cost function is calculated and weights are redefined after each row of data is processed. All batches are run within a given epoch. The epoch value corresponds to the number of times the model will run through the entire dataset. Usually, it will iterate through the data many times before reaching a sufficient accuracy. [5] [19] [20].

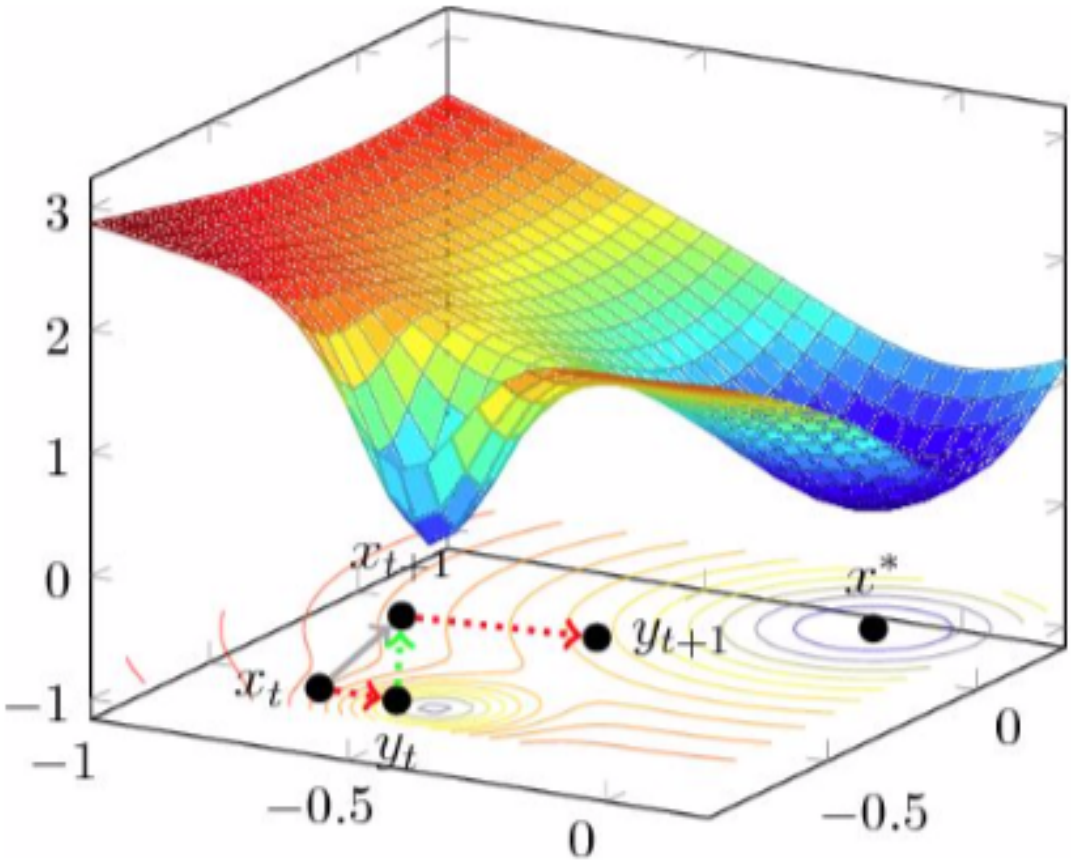

Figure 2.2: Cost Surface. Individual points are plotted for each cost function calculation. As the gradient descent algorithm gravitates towards the minimum, it is pinpointing optimal weights to implement in the neural network [19].

Chapter 3

NOvA's CVN

3.1 Architecture

The NOvA CVN is loosely based on the GoogLeNet architecture, consisting of convolutional layers, pooling layers, inception modules, and a softmax output [4]. The term inception module refers to a specific configuration of layers that are grouped together. In this case, the CVN and GoogLeNet inception modules consist of the same convolutional layers and a pooling component, which are shown in Fig 3.1. The pooling layers work to reduce the dimensions of the image. The CVN uses max pooling and average pooling. Both extract data from a specified n×m region of the image and replace this n×m region with a single value representative of this region. In max pooling, the maximum value from the region remains, and in average pooling the mean of all values replaces the region. The softmax output refers to the exponential function used in evaluation to normalize the network outputs. GoogLeNet and NOvA's CVN utilize local response normalization (LRN), which works to avoid local minima as the model seeks the optimal weights for the network [4].

Two key differences between the CVN and GoogLeNet structures include the CVN's parallel X and Y processing and reduced number of inception modules. The CVN has been built to evaluate the same events as they were recorded in the X and

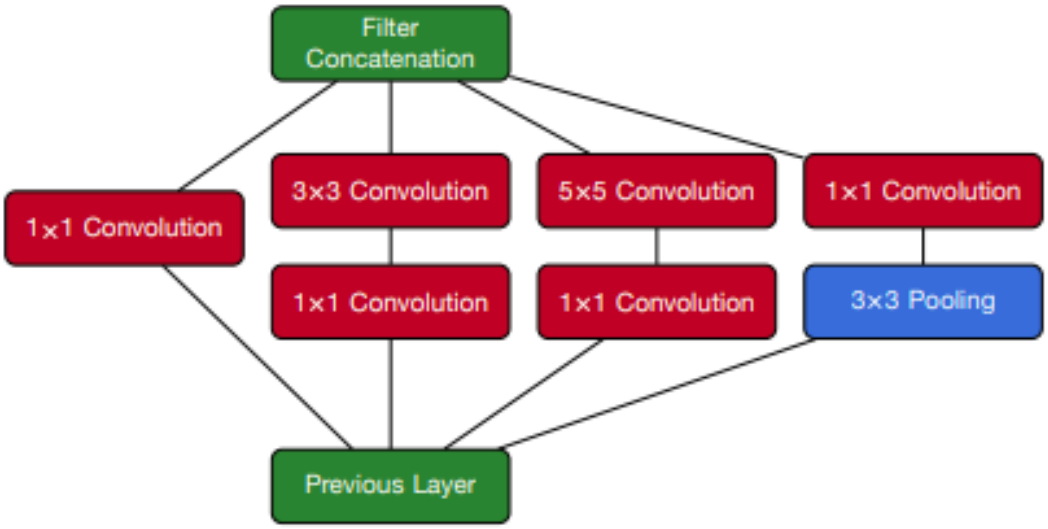

Figure 3.1: Inception Module. The inception module for NOvA's CVN consists of several convolution layers, and a pooling layer. A single inception module represents a structure that is concatenated and repeated throughout the CVN's architecture [4].

Y planes of the Far Detector. To extract information obtained from both views, they are put through identical convolutions, pooling layers, and three inception modules in parallel before being recombined for a final pass through an inception module and pooling layer to isolate common features. The full architecture of the CVN is included in Fig. 3.2 [4].

3.2 Data Simulation and Training Methods

While many CC events are recognized by the spatial patterns that are produced when they interact with a NOvA detector, there is still room for misidentification using the naked eye. CNN's are built to avoid false positives, or in this case, incorrect classifications of events. As they train, they improve their accuracy based on details

that humans may overlook [4].

Using the knowledge that most of the oscillated beam is comprised of ν_μ with 2 percent electron neutrino contamination, NOvA chose to comprise simulated data of equal parts $\nu_\mu \rightarrow \nu_e$ oscillations, $\nu_\mu \rightarrow \nu_\tau$ oscillations, and an oscillated flux of ν_μ. In total, the CVN was trained with 4.7 million FD events simulated by the GENIE package [3] on an 80 percent training and 20 percent testing split. Training the model with 80 percent of the files is meant to optimize the accuracy of the model and avoid overtraining. NOvA's CVN uses stochastic gradient descent in training to find the optimal weights for different parameters. Since stochastic descent is prone to getting stuck in local minima, the step size of the gradient descent was dropped at intervals where the network loss improvement rate plateaued. This, along with training each iteration with 32 training samples produced versions of the CVN with optimal accuracy [4].

NOvA conducted training procedures to create a more generalizable model with sufficient external validity to accurately classify real data. With the simulated data, they utilized the back-propagation process to prevent the model from biasing any one feature of the data too heavily. They also added variety to the data to make the model more adaptable to real conditions that would include noise and asymmetry from the layout of the detectors [4].

3.3 Results

Results from the CVN classifications published in "A Convolutional Neural Network Neutrino Event Classifier," by Aurisano et. al. showed that ν_μ disappearance oscillations and ν_e appearance oscillations were selected from background with an increase in efficiency compared to event identification algorithms previously used in

other NOvA publications. For ν_μ CC events the selection efficiency relative to the true event was 58%, which when compared to previous results of 57%, showed that the model was not underperforming with these classifications. For ν_e CC events were identified with 49% efficiency, which was a significant increase over the previous 35% efficiency of a different event selector algorithm [4]. There were also tests conducted to ensure that the model's sensitivity to systematic uncertainties was not higher than those of previous measurements of event selectors for ν_μ and ν_e interactions. Both ν_μ and ν_e signal selections were sensitive to systematics by less than 2%, which were comparable to previous results [4] [16] [2].

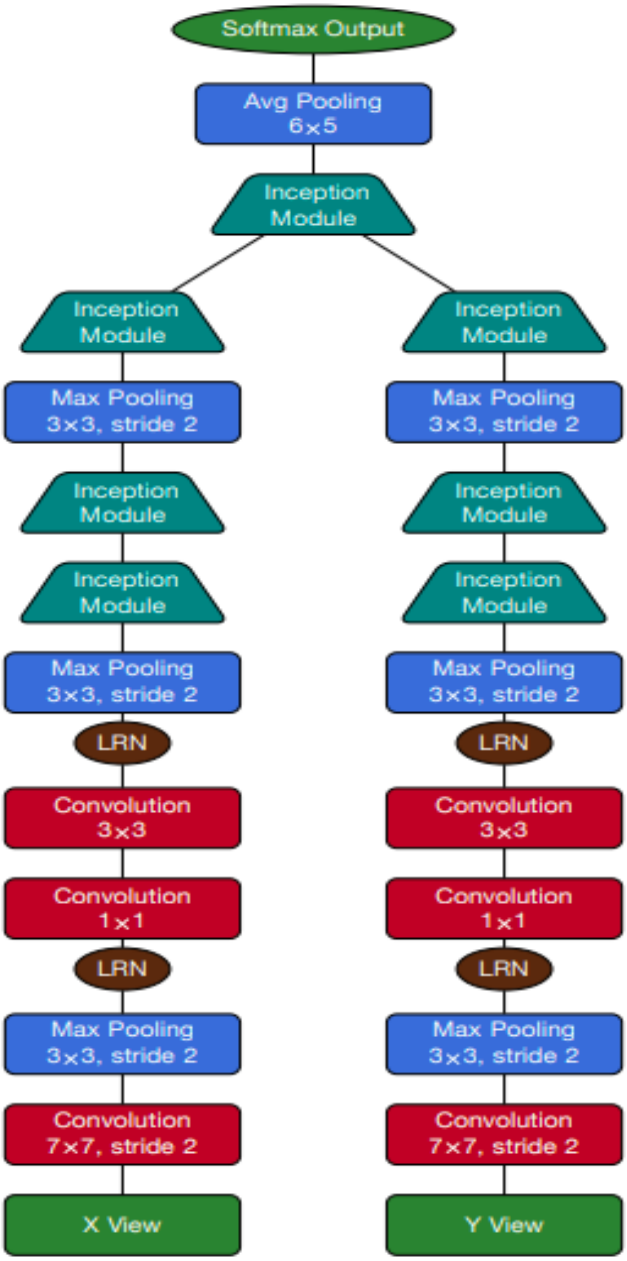

Figure 3.2: CVN Architecture. The CVN parallel architecture takes inputs from the X View and Y View data recorded in the FD detector and performs the same convolution and pooling steps on each before combining the transformed data and using convolution, LRN, and pooling a few more times to extract common features from the datasets [4].

Chapter 4

Systematics Analysis

4.1 Baseline Training

NOvA has made it possible to train versions of the CVN so that it can be applied to various other classification problems. The goal of this project was to train a version of the CVN classifier and evaluate the training sensitivity to systematic uncertainty. Before analyzing possible systematic biases from the model, it was necessary to obtain a baseline for classification. This baseline is made up five nominal models that were created using data made up of all event types, and trained with the default parameters of the classification network. The CVN was run with a 50 percent training, 50 percent testing split of the same source files for the training of every model created in this project. This means 50% of the input data was used to train and fit the model while the other 50% was used to validate its classifications [21]. The nominal baseline was created using the same learning rate, epochs, and number and size of batches within these epochs. The consistency of each model in this baseline was evaluated based on how many events were correctly classified.

A few methods for assessing the consistency of the nominal baseline included comparing event count totals, PID plots, ROC curves, and confusion matrices associated with each model. Fig. 4.1 shows the similarities between the PID plots created by

two nominal models. In general, the model generates a PID score, ranging from 0 to 1.0, that is based on how well a data point fits the model's criteria for a certain type of event. A PID score of 1.0 represents an event that perfectly matches a certain label. The PID plots in Fig.4.1 show how events have been rated by the model based on ν_e characteristics. They demonstrate the quantity of ν_e events classified by the models, as well as their ability to distinguish between event types, or their "class separability" [18].

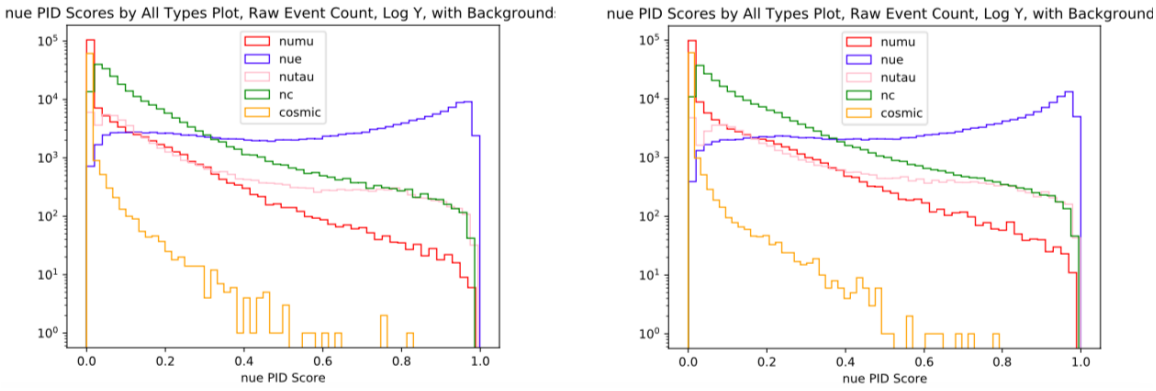

Figure 4.1: Nominal PID Comparison. These are two ν_e PID plots created with nominal models that make up the baseline. In these graphs, the model is scoring more ν_e events as ν_e than any other event type. The shape of both plots is very similar, and they contain similar event counts for each type.

A model's ability to accurately distinguish between event types is also evaluated based on The Receiver Operating Characteristic (ROC) curve and the Area Under the ROC Curve (AUROC). An example of a nominal ROC curve is shown in Fig. 4.2. This plot is very representative of the nominal models, as the vast majority of them have areas of 0.93. The dashed line represents a model that would have no class separability at all, and an ROC Curve of area = 1 corresponds to perfect class distinction [18].

Another important facet of the model's accuracy is its ability to avoid false positives and false negatives. The confusion matrix in Fig. 4.3 was produced with the

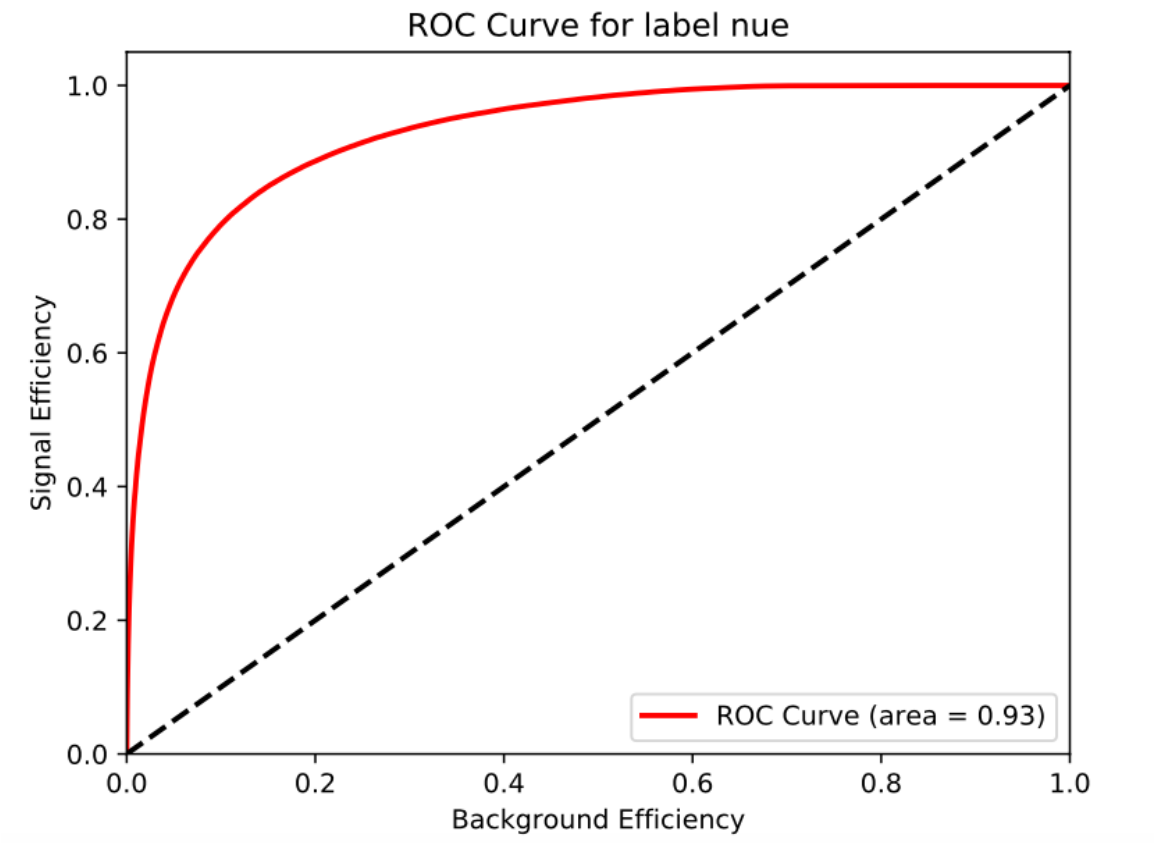

Figure 4.2: Nominal ROC Curve. This curve gives insight into the model's sensitivity and ability to distinguish between event types. Nearly all of the nominal models had an ROC value of 0.93, with only one of them being 0.94. Both values demonstrate excellent class separability.

first nominal run. It gives more insight into how well the model accurately predicts each event type by revealing which events were most often mistaken for others. The area of interest here is mainly on how often ν_e events were correctly classified. In the case of this nominal model had 73% efficiency. The average ν_e classification efficiency was $75 \pm 5\%$ for all nominal models.

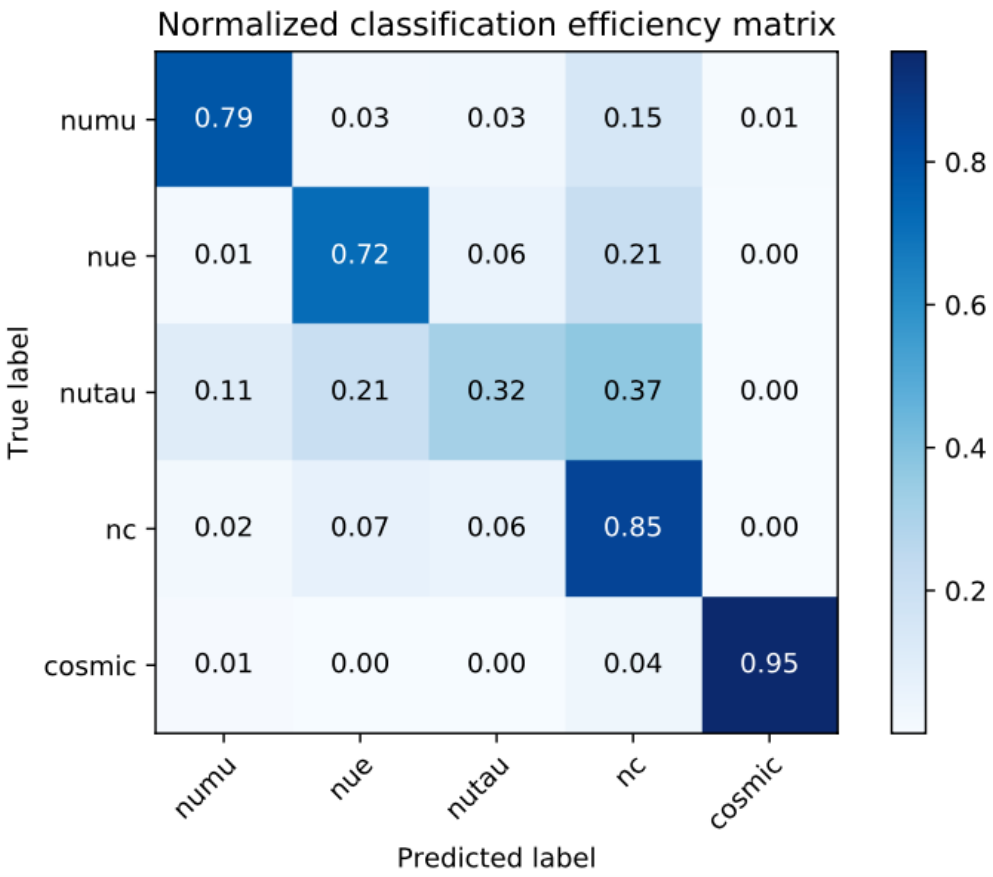

Figure 4.3: Nominal Confusion Matrix. The efficiency confusion matrix shows exactly which events were correctly classified by the model as ν_e. It also reveals of those that were classified incorrectly, what the model labeled them as, and compares this to their true label.

4.2 Weighted Training

As previously mentioned, when the model trains in a CNN, it identifies certain features in the image data through the use of filters to make classifications. It is possible for the model to emphasize certain features that give it accurate classifications on the simulated events, but lead to less accurate results for real datasets. In order to obtain reliable results of classifications for real, measured events, it is crucial to ensure that the model is not susceptible to systematic uncertainties. One of the

biggest sources of systematic uncertainty exists in the form of neutrino interactions with matter once they have been created [22]. Removing an event type associated with this uncertainty from the model's training reveals how it responds to systematics in the training process.

The weighted model, also referred to as the cut model, was created with the same ratio of testing and training of the original source files as the nominal baseline. The model also included initial parameters identical to the ones listed previously for the nominal models: learning rate, epochs, and number and size of batches within these epochs. Maintaining these parameters was necessary in order to observe how the addition of weights might alter the model's classifications. In the configuration of this cut run, the model was told to throw out events labeled as CC resonant interactions and then the model was trained via the same processes as the nominal models.

The weighted model's class separability and event count for ν_e classifications is shown in the PID plot in Fig. 4.4. It has a noticeably higher event count for ν_e events than the two nominal examples. The ROC curve for the weighted model, shown in Fig. 4.5, had an area of 0.94. It is a slight improvement over four of the nominal runs by 0.01, and matches the value of the remianing nominal. The confusion matrix in Fig.4.6 contains the composition of all of the weighted model's classifications. Most notably, it was able to identify ν_e events with 84% efficiency, which was higher than the $75 \pm 5\%$ mean efficiency of the nominal models.

4.3 Results and Comparisons

When comparing each nominal model to the weighted one, four of them had event counts that were lower than the weighted event count by thousands. However, there was one nominal run, called test2, that was very similar in shape and event count to the weighted run. Fig. 4.7 shows the mean and rms error of all five nominal ν_e PID

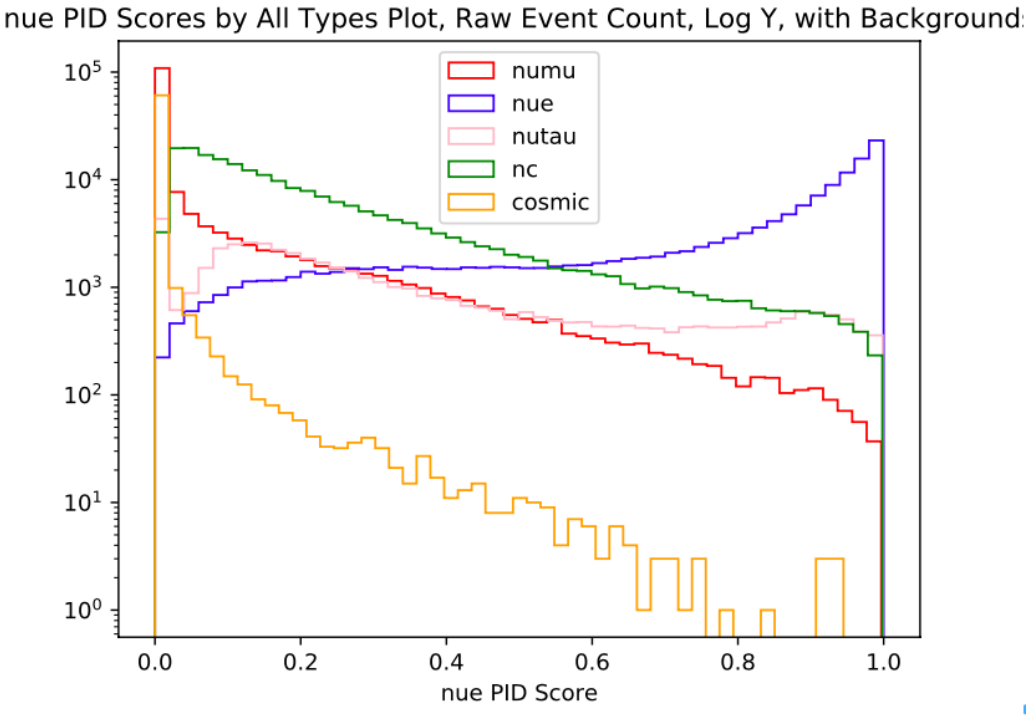

Figure 4.4: Weighted PID Plot. This PID plot shows the number of events classified by the weighted model as ν_e and illustrates its ability to separate between event types. Notice the higher event count for the weighted model compared to the nominal examples.

results compared to the weighted PID values. To show the effect of test2 on this plot, the mean values in Fig. 4.8 exclude those of test2. In both cases, the event count values of the weighted model were outside of the nominal mean values, indicating that applying a systematic change to the data did alter the model's classifications.

4.4 Conclusions

In conclusion, it was crucial to test for systematic bias in the CVN to ensure its reliability when classifying measured events, and when being applied to other problems. With more time, we would comapre PID scores on an event by event basis to hone the source of the differences between the nominal and modified models. Based on the

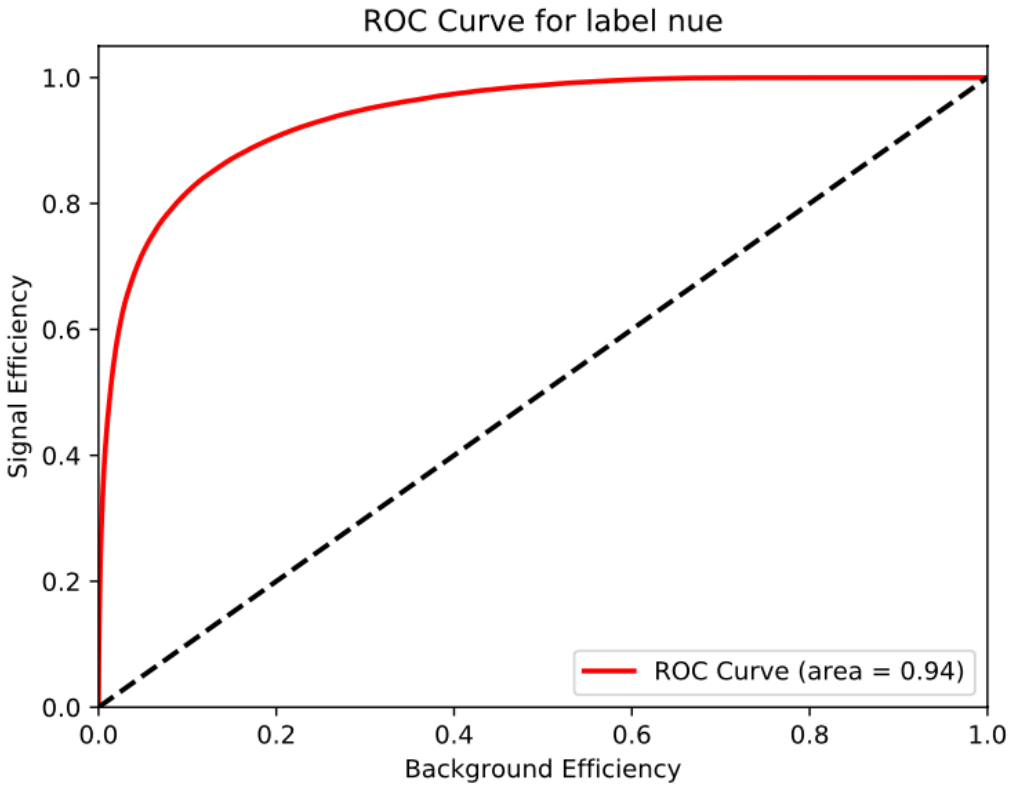

Figure 4.5: Weighted ROC Curve. The area under this curve shows the weighted model's class separability.

comparison between the mean of all nominal runs and the weighted run, it appears that the weighted run was able to classify a higher efficiency and quantity of ν_e events, which shows that the model is sensitive to systematic changes in the data.

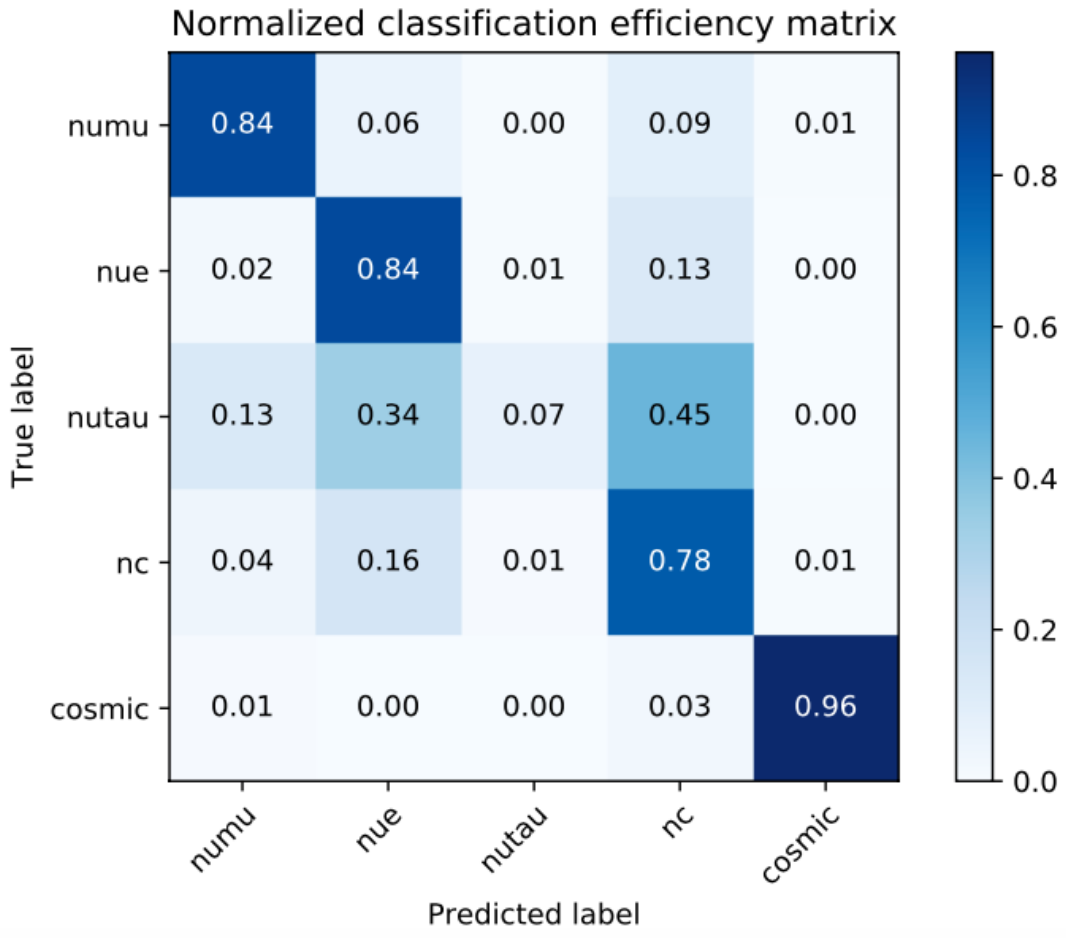

Figure 4.6: Weighted Confusion Matrix. Above is the summary of the weighted model's classifications compared to their true labels. Notice how ν_e events were correctly labeled more often than the nominal models, on average.

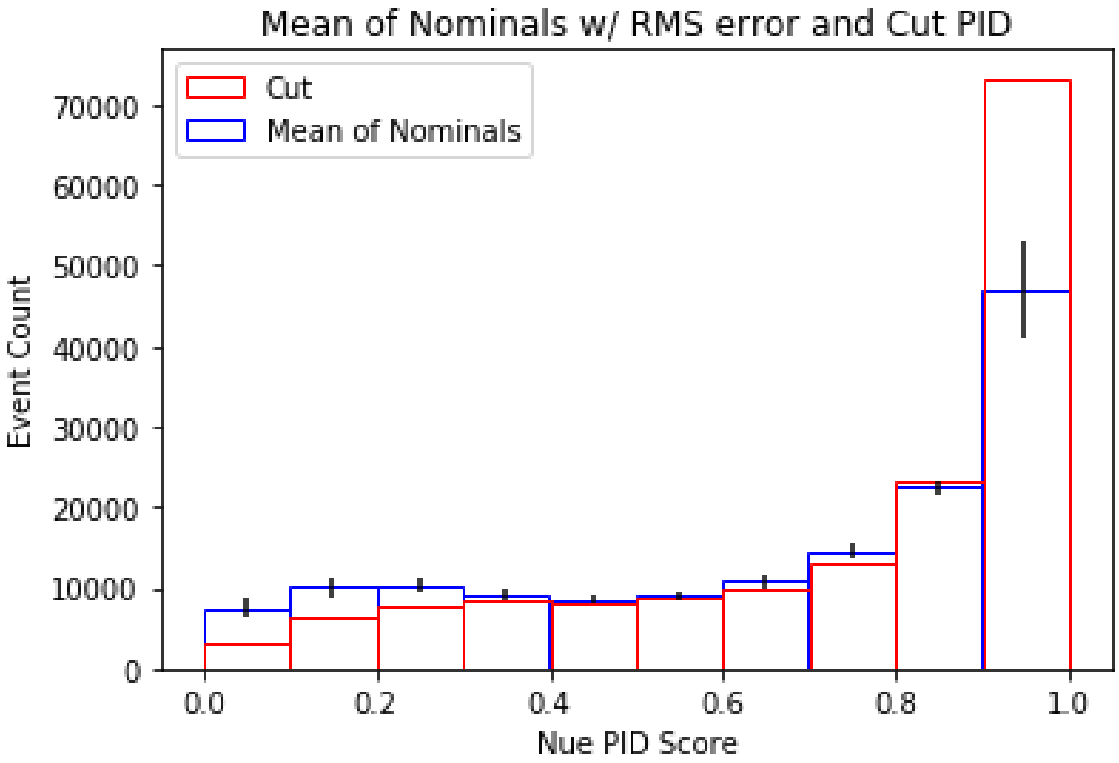

Figure 4.7: Compared PID Scores. This plot includes the mean of all nominal ν_e PID plots with rms errorbars, compared to the PID result of the cut run.

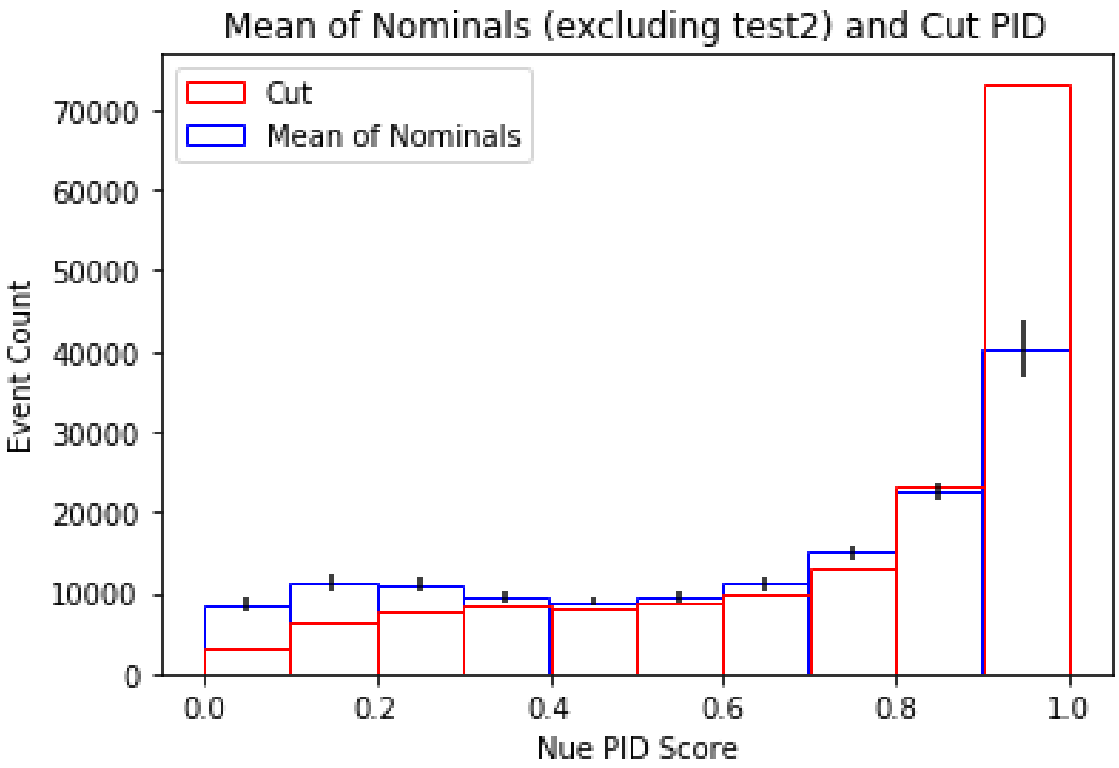

Figure 4.8: Compared PID Scores without test2. Notice how much lower the peak drops after exlcuding test2, which was most similar in shape to the weighted run.

Bibliography

[1] Acero, M. A. (2019). First measurement of neutrino oscillation parameters using neutrinos and antineutrinos by NOvA. Journal of Physics: Conference Series, 1219. doi: 10.1088/1742-6596/1219/1/012021

[2] Adamson, P. et al. "First Measurement of Electron Neutrino Appearance in NOvA." Physical Review Letters 116.15 (2016): n. pag. Crossref. Web. doi: 10.1103/PhysRevLett.116.151806

[3] C. Andreopoulos et al., The GENIE neutrino monte carlo generator, Nucl. Instrum. Meth. A614 (2010) 87–104.

[4] Aurisano, A., Radovic, A., Rocco, D., Himmel, A., Messier, M., Niner, E., ... Vahle, P. (2016). A convolutional neural network neutrino event classifier. Journal of Instrumentation, 11(09). doi: 10.1088/1748-0221/11/09/p09001

[5] Difference Between a Batch and an Epoch in a Neural Network. https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

[6] Energetics of Charged Pion Decay. http://hyperphysics.phy-astr.gsu.edu/hbase/Particles/piondec.html.

[7] Fermilab Creative Services. Far Detector Image https://vms.fnal.gov/asset/detail?recid=1936347

[8] Fermilab Creative Services. Near Detector Image https://vms.fnal.gov/asset/detail?recid=1937176

[9] 2018 Fermilab Physics Slam: Explaining neutrino oscillations. https://www.youtube.com/watch?v=UY1QQr-PZOg

[10] Google's Machine Learning Crash Course. https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/anatomy

[11] How does NOvA work? https://novaexperiment.fnal.gov/how-does-nova-work/.

[12] How do we 'train' a neural network? https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73.

[13] Neutrino Oscillations. https://www.youtube.com/watch?v=tqkZNEQk9l8.

[14] NOvA. https://en.wikipedia.org/wiki/NOvA#History

[15] Power of a Single Neuron. https://towardsdatascience.com/power-of-a-single-neuron-perceptron-c418ba445095

[16] Psihas, Fernanda. The Convolutional Visual Network for Identification and Reconstruction of NOvA Events. United States. doi:10.1088/1742-6596/898/7/072053.

[17] Research Goals. https://novaexperiment.fnal.gov/research-goals/.

[18] ROC Curves. https://acutecaretesting.org/en/articles/roc-curves-what-are-they-and-how-are-they-used

[19] Runfola, D. (2018, November). Williamsburg.

[20] Runfola, D. (2019, November). Williamsburg.

[21] About Train, Validation and Test Sets in Machine Learning
https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

[22] Vahle, P. Long-baseline neutrino Oscillation Experiments (2019)