Machine Learning Classification for Neutrino Interactions

Simran Oodit

April 13, 2024

1 Introduction

Neutrinos are neutral subatomic particles with extremely low mass that rarely interact with matter. As a result, they are difficult to detect. There are three "flavours" of neutrino—electron neutrinos ν_e, muon neutrinos ν_μ , and tau neutrinos ν_τ. Neutrinos interact with matter primarily through the weak interaction which can either be a charged-current (CC) type or neutral-current (NC) type. In a CC interaction, a neutrino emits a charged lepton and changes flavour whereas in a NC interaction, the neutrino does not exchange any charged leptons and maintains its original flavour. The NOvA experiment is a key part of ongoing research which investigates neutrinos.

The aim of this project is to train a machine learning classifier on simulated images resembling the particle tracks made in the NOvA detector, to identify muon neutrino ν_μ charged-current events. The secondary aim is to investigate the relationship between the image metadata, such as neutrino energy, and the performance of the classifier. There were 200 available files of data for this project which contained images of simulated neutrino interactions and their associated metadata.

2 Preprocessing Data

2.1 Binary Classification

Each file of the data contains $\sim$ 7000 different samples where each is a pair of stacked images of dimensions 100×80 pixels. Each pair of images which can be considered as a "top view" and a "side view"—i.e. in the xz and yz plane —for the same neutrino event stacked into one image. The $\sim$ 7000 samples per file were found to be inadequately small for training, so 4 files of data consisting of 27985 samples in total were used for the dataset. Of those samples, 80% was reserved for training, 10% for validation, and 10% for testing.

The metadata associated with each of these stacked images i.e. each event is summarised in Table 1.

Metadata Label	Description
neutrino/nuenergy	Neutrino Energy (GeV)
neutrino/lepenergy	Lepton Energy (GeV)
neutrino/interaction	Interaction
neutrino/finalstate	Final State

Table 1: Table of metadata variables associated with each neutrino interaction image.

The metadata label for interaction takes values between $0-16$ where each number corresponds to one of 17 possible classes of interaction as summarised in Table 2. The first 4 interaction classes $0-3$ are the ν_μ CC type of interest. Attempting to train the model to recognise which specific class an interaction belongs is unnecessary to complete the project aims and would likely require significantly more data and training time to work effectively. Instead, the images are relabelled with a new Boolean variable where:

$$\text{Label} \begin{cases} \text{If interaction is } \nu_\mu \text{ CC} = 1 \\ \text{Else} = 0 \end{cases} \tag{1}$$

Interaction Class Name	Class Label	Interaction Type
kNumuQE	0	ν_μ CC QE
kNumuRES	1	ν_μ CC RES
kNumuDIS	2	ν_μ CC DIS
kNumuOther	3	ν_μ CC Other
kNueQE	4	ν_e CC QE
kNueRES	5	ν_e CC RES
kNueDIS	6	ν_e CC DIS
kNueOther	7	ν_e CC Other
kNutauQE	8	ν_τ CC QE
kNutauRES	9	ν_τ CC RES
kNutauDIS	10	ν_τ CC DIS
kNutauOther	11	ν_τ CC Other
kNuElectronElastic	12	NC Elastic Scattering of $\nu + e^-$
kNC	13	NC interaction
kCosmic	14	Cosmic Background Ray
kOther	15	Other
kNIntType	16	Number of interaction types - analagous to vector size.

Table 2: Table of interaction classes and their assigned numerical labels. Interactions can be Quasi-Elastic (QE), Resonant (RES), Deep Inelastic Scattering (DIS), Charged-Current (CC), Neutral-Current (NC), or miscellaneous.

Hence, any interaction which was originally in classes 0,1,2 or 3 would be relabelled as a positive 1 result and interactions in any of the other classes would be relabelled as a negative 0 result. This makes identifying ν_μ CC type events a binary classification task. The image data consists of pixels which is normalised to values between $0 - 1$ before being input into the model.

After relabelling, analysis of the data reveals that the majority of the samples are ν_μ CC types, i.e. positive results. Whilst only 4 of the 17 classes are ν_μ CC type, 88.24% of the dataset is in the positive class and 11.76% is in the negative class. Training on data with this significant class imbalance may lead to the model reporting artificially high accuracy, which would be a misleading measure of the model's performance. The accuracy describes the number of times the model correctly predicts the class. If the model learns to label every sample as positive, it could still achieve 88% accuracy as this prediction would be correct for $\sim$ 88% of the samples. This is problematic as the model would not be distinguishing a positive result from a negative result, which is the primary purpose of a binary classifier. The model would also not be robust when tested on data with different class distributions. Hence, the model's predictive power on the minority negative class is key to assessing its overall performance.

2.2 Imbalanced Training Data Strategies

There are several strategies to address this high imbalance in the dataset.

1. Resampling could be employed to balance the dataset. Either by removing positive samples to under-represent the majority positive class or including more negative samples to over-represent the minority negative class in the training dataset.

2. The class weights in the loss function could be adjusted to increase the penalty for incorrectly classifying the minority negative class.

3. The model's performance should be evaluated according to multiple metrics.

4. The threshold where the model distinguishes between the two classes could be adjusted. Binary classifiers produce a probability score between $0 - 1$ as output. If the probability score exceeds the threshold value, the output is classed as positive —otherwise, the output is classed as negative. Raising the threshold would make the model more conservative in its determination of a positive result, but this is a trade off against the risk of missing instances of the positive class which could lead to the model being less effective at detecting ν_μ CC events.

The first strategy is less easily implemented and hence less efficient at addressing the imbalance compared to the other strategies. For a real-world application of this classifier, the labour-intensive process of resampling data may be costly for a budget of limited resources so the other strategies are prioritised.

The second strategy is implemented during training, where the class weights are calculated using the following equation.

$$W_{\text{Class i}} = \frac{n_{\text{total samples}}}{n_{\text{total classes}} \times n_{\text{samples of Class i}}} \tag{2}$$

To identify suitable metrics for the third strategy, it is important to understand the type of predictions the binary classifier model can make, listed as follows.

- TP: True Positive, i.e. accurately identifying a ν_μ CC event.
- TN: True Negative,i.e. accurately identifying a non ν_μ CC event.
- FP: False Positive, i.e. misidentifying a non ν_μ CC event as a ν_μ CC event.
- FN: False Negative, i.e. misidentifying ν_μ CC event as a non ν_μ CC event.

This is visualised in the confusion matrix Table 3. False Positives are Type I errors and False Negatives are Type II errors.

Table 3: Confusion Matrix

		Predicted Classes	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

With this categorisation, the following metrics for the binary classifier can be introduced. Considering the accuracy of predictions on the majority positive and minority negative classes separately provides a more informative measure of the classifier's performance compared to the overall accuracy.

$$\begin{aligned} \text{Majority Positive Class Accuracy} &= \frac{TP}{TP + FN} \\ \text{Minority Negative Class Accuracy} &= \frac{TN}{TN + FP} \end{aligned} \tag{3}$$

The positive class accuracy, also known as the true positive rate, is the proportion of the model's positive predictions that are correctly classified as positive, which indicates the precision of the classifier. Sensitivity/Recall is the proportion of actual positives in the data that the model correctly classified as positive which indicates the model's overall ability to detect the positive class.

$$\text{Recall/Sensitivity} = \frac{TP}{TP + FN} \tag{4}$$

Increasing the model's precision can be a trade off against decreasing the model's sensitivity, hence these two metrics are monitored together to observe whether improvement in one is gained at the expense of the other, and to what extent. The fourth strategy is implemented after training. The trained model's predictions for a range of thresholds is categorised into a confusion matrix as shown in Table 3. The optimal threshold is then determined from the threshold at which the accuracies found using Equation 3 are maximised.

3 Classifier Model

3.1 Model Architecture

Because the two stacked images are associated with a single label, each pair of images need to be input into the model simultaneously. To achieve this, a Convolutional Neural Network (CNN) as shown in Figure 1 was originally proposed as these are well-recognised for their ability to classify images.

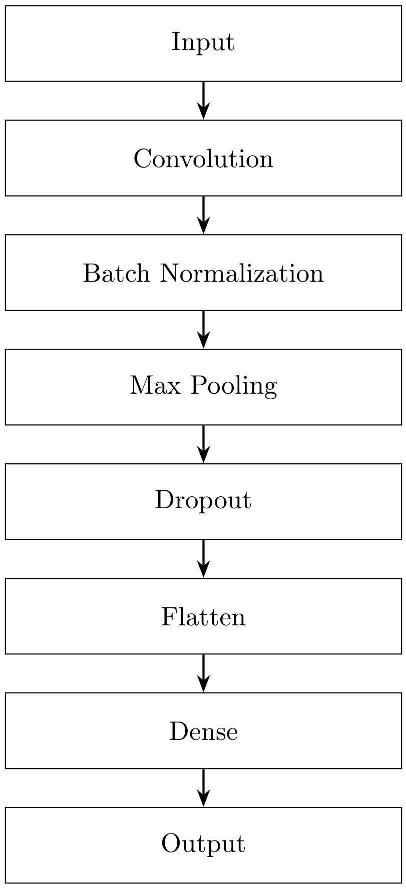
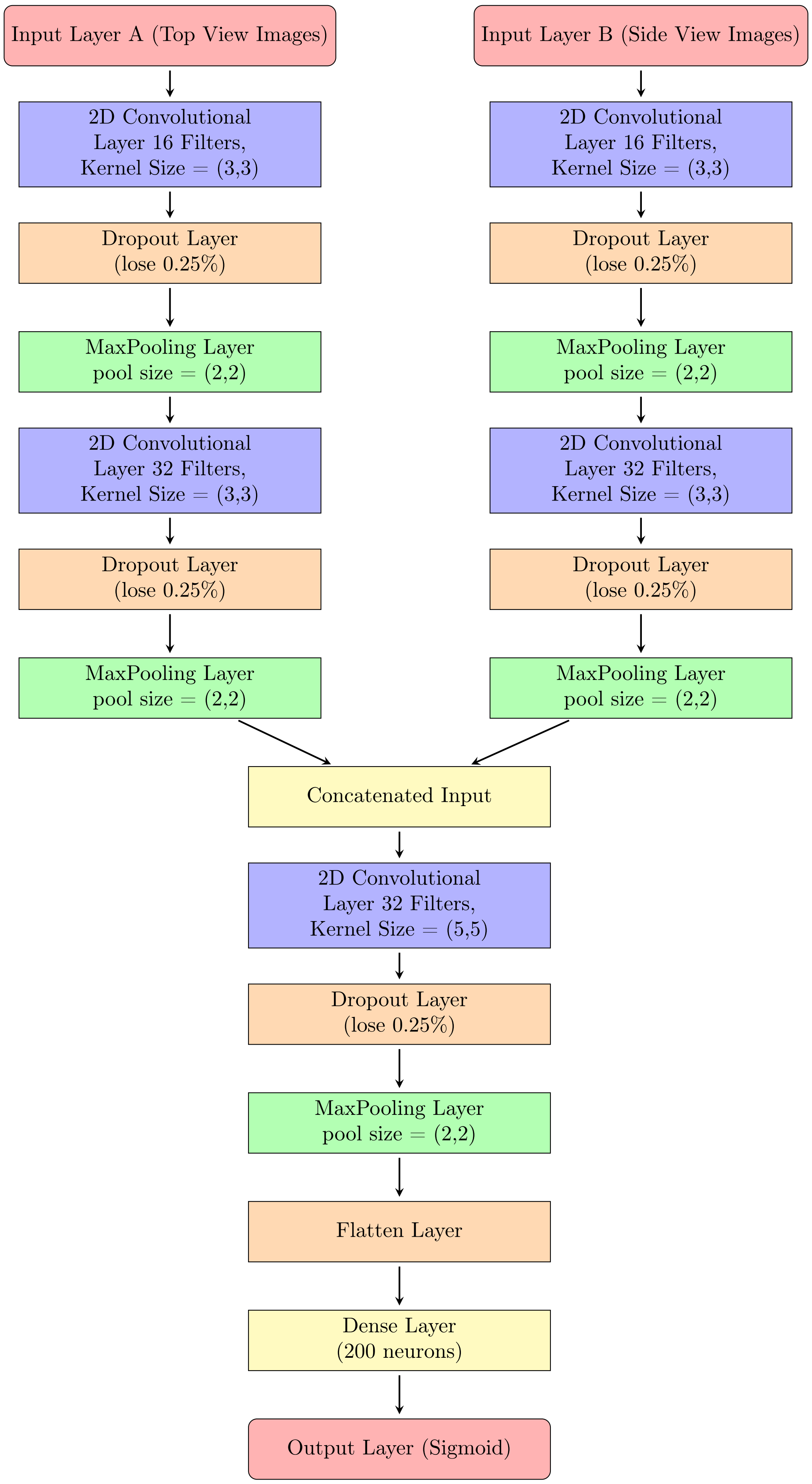


Figure 1: First Sequential Keras CNN Model

This initial CNN model takes input of dimensions $100 \times 80 \times 2$, thereby treating the top and side views as 2 separate channels of the image. The problem with this network design is that the two views are not spatially correlated. This means that the same coordinate in the top and side view arrays are not representing the same point in physical space. The modified model architecture improves on this by taking the top and side view images as separate inputs into two branches of the network that are concatenated to produce a single output, hence matching the single label per pair of images. This approach is inspired by this paper [3] which similarly develops a CNN model to classify neutrino events from the NOvA detector. The modified model architecture is shown in Figure 2.



Input Layer A (Top View Images)
Input Layer B (Side View Images)
2D Convolutional Layer 16 Filters, Kernel Size = (3,3)
2D Convolutional Layer 16 Filters, Kernel Size = (3,3)
Dropout Layer (lose 0.25%)
Dropout Layer (lose 0.25%)
MaxPooling Layer pool size = (2,2)
MaxPooling Layer pool size = (2,2)
2D Convolutional Layer 32 Filters, Kernel Size = (3,3)
2D Convolutional Layer 32 Filters, Kernel Size = (3,3)
Dropout Layer (lose 0.25%)
Dropout Layer (lose 0.25%)
MaxPooling Layer pool size = (2,2)
MaxPooling Layer pool size = (2,2)
Concatenated Input
2D Convolutional Layer 32 Filters, Kernel Size = (5,5)
Dropout Layer (lose 0.25%)
MaxPooling Layer pool size = (2,2)
Flatten Layer
Dense Layer (200 neurons)
Output Layer (Sigmoid)

Figure 2: Two-Input CNN Binary Classifier

A 2D Convolutional Layer applies a set of filters, which are matrices of weights known as kernels. The filter convolves over the image, producing a feature map from the dot product of the weights and the image pixels over which the filter is applied. The output from different filters represent different visual features of the input image. Hence, multiple filters allows the model to extract various features. Smaller kernel sizes focus on capturing fine details whereas larger kernel size aim to capture broader patterns.

For the convolutional layers in the classifier, the number and size of the kernels progressively increase down the layers from 16 (3×3) to 32 (5×5) filters to extract first low and then high-level features. Odd numbers for the kernel sizes are chosen to maintain symmetrical arrangement of the previous layer;s pixels around the output pixel, which avoids creating distortions [2].

The MaxPooling layers downsample the convolutional layer output by representing the feature map from each filter as its maximum value. This reduces dimensionality which speeds up training time but also filters noise and helps prevent overfitting. This improves the model's robustness to small variations in the input data. Dropout layers which randomly remove 25% of the input are also interspersed between the convolutional and pooling layers to prevent overfitting. The proportion of dropped input was chosen by comparing the training and validation data to observe the effect of the dropout layers on overfitting.

The concatenated layer combines the output from each branch which is then flattened, passed to a dense layer and sent to the output layer. The convolutional and dense layers use the ReLU activation function as this was found to achieve the best results. A Sigmoid activation function is applied at the output layer as it is well suited for binary classification on two classes [1].

3.1.1 Model Parameters

$$\text{Binary Cross Entropy} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)) \tag{5}$$

Binary Cross Entropy is the chosen loss function given in Equation 5, where N is the number of samples, y_i is the true label of the sample i, and $p(y_i)$ is the classifier's predicted probability score for the sample i. When the model is compiled, the loss function is weighted using class weights calculated from Equation 2. The model uses the "Adam" optimizer, which is a more sophisticated and highly effective variation of stochastic gradient descent that utilises an adaptive learning rate.

The model is trained on Precision, Recall, and all 4 metrics of the confusion matrix in Table 3. This facilitates monitoring of the False Positives and True Negatives during training which is important for determining whether the model is improving the accuracy of its predictions on the negative minority class. A batch size of 30, with 20 epochs and 450 steps per epoch yields good performance on the chosen metrics whilst maintaining a reasonable training time.

3.2 Model Performance

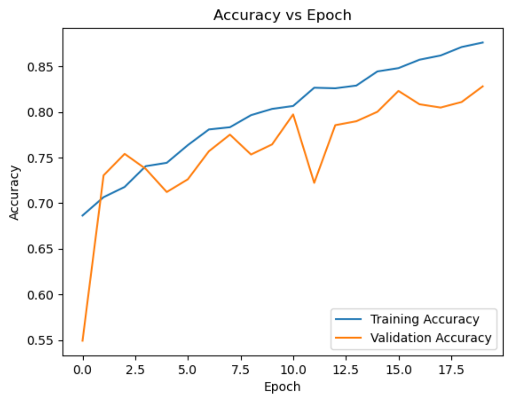


Figure 3: A plot of Accuracy vs Epochs for the Two-input Classifier on Training and Validation data. The model was trained for 20 epochs, on batch size = 32 and steps per epoch = 450

Figure 3 shows the model accuracy and Figure 4 shows the precision and recall plotted against epochs for training and validation data. These plots show that the model achieves $\sim$ 85% on training accuracy and $\sim$ 82% on validation accuracy. The trained model reported $\sim$ 82% testing accuracy. Whilst relatively high, this accuracy is simply the proportion of samples the model correctly identified. The testing precision of $\sim$ 95% and testing recall of 84% give a more reliable indication that the model is sufficiently trained. The model's validation recall increases over the number of epochs whereas its validation precision decreases which demonstrates the expected trade-off between these metrics.

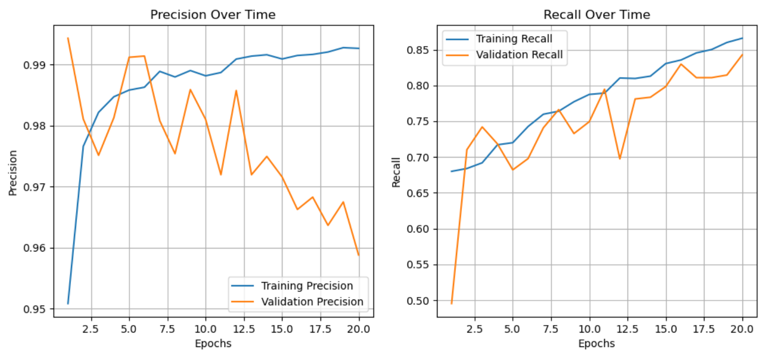


Figure 4: A plot of Precision and Recall vs Epochs for the Two-input Classifier on Training and Validation data. The model was trained for 20 epochs, on batch size = 32 and steps per epoch = 450

Additionally, the model is retrained to report True Positives, True Negatives, False Positives, and False Negatives. The variation of these metrics during training is shown in Figure 5.

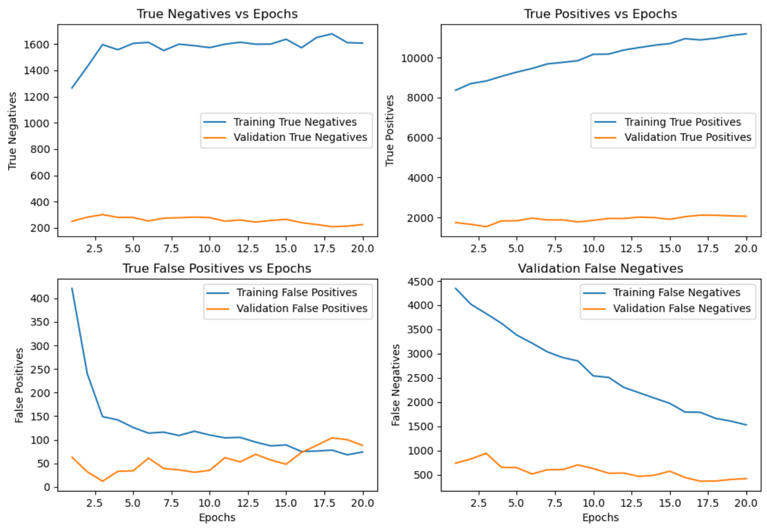


Figure 5: Plots of True Negatives, True Positives, False Positives, and False Negatives vs Epochs for the Two-input Classifier on Training and Validation data. The model was trained for 20 epochs, on batch size = 32 and steps per epoch = 450.

Figure 5 shows that whilst the model improves in training metrics, the validation metrics are largely the consistent. Whilst this may at first suggest that the classifier is not improving its predictions on the negative class, it is important to consider that these metrics are obtained on the default threshold of 0.5, which is not effective for the imbalanced dataset. By implementing the fourth strategy, the accuracy can be recalculated on a chosen threshold from the trained model's predictions. Doing this for a range of thresholds allows an optimal threshold to be determined.

3.3 Optimal Threshold

The model's predictions are probability scores, which are converted to binary labels according to a chosen threshold. The predicted labels are compared against the known labels to find the confusion matrix values as in Table 3 from which the accuracies are subsequently calculated using Equation 3. By performing this procedure iteratively for a range of thresholds, the optimal threshold is identified. The optimal threshold is defined here as the maximum mean of the positive and negative class accuracies. This is visualised by plotting the curve of positive class accuracy i.e. precision against the negative class accuracy over the range of tested thresholds, as shown in Figure 6.

The negative class accuracy decreases as precision increases first gradually and then rapidly once precision surpasses ~ 0.7. The optimal threshold is the point on the curve before the steep drop where both the positive and negative class accuracies are relatively high.

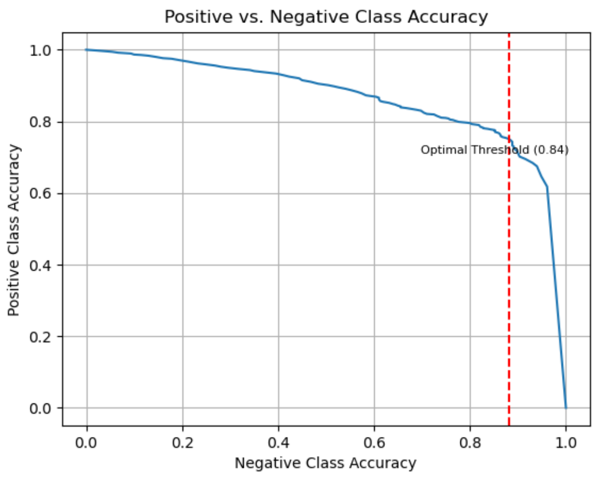


Figure 6: Curve of positive class accuracy against negative class accuracy for a range of classification thresholds between $0 - 1$. The point of intersection between the curve and the red dashed line marks the positive and negative class accuracies at the optimal threshold 0.84.

The difference between the confusion metrics for the default threshold of 0.5 and the optimal threshold 0.84, is shown in Figure 7. The confusion matrix with the optimal threshold improves the classifier's performance on the minority class by decreasing the False Positives and increasing the True Negatives. Applying Equation 3 to the confusion matrix for the optimal threshold yielded a maximum overall accuracy of 82% from a positive class accuracy of 75% and a negative class accuracy of 88% at the optimal threshold 0.84. As expected, the increase in true negatives comes at the expense of a decrease in true positives, but the difference between positive class accuracy and negative class accuracy is reduced, and the model's performance is therefore more balanced over the two classes.

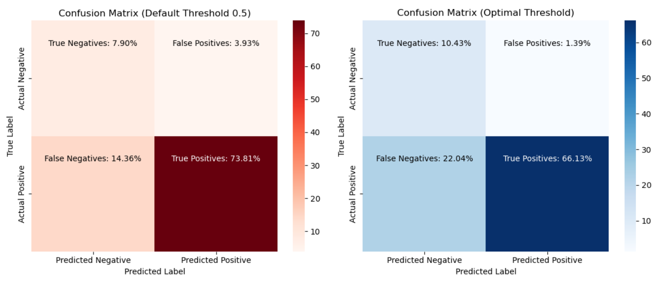


Figure 7: The confusion matrix with metrics as a percentage of total predictions for the default threshold 0.5 (shown in red on the left) and for the optimal threshold 0.84 (shown in blue on the right).

4 Dependence on Metadata

4.1 Energy

4.1.1 Neutrino Energy

The second aim of the project is to see whether the trained classifier's performance varies with any of the metadata. The model is tested on different batches of images containing 55 to 56 samples each, which correspond to the energy bins of the neutrino energy distribution for the samples. The neutrino energies in the testing batch range from $0 - 74.24$ GeV. The model predicts on each batch, and its output probability scores are converted to positive predictions of 1 if they exceed the optimal threshold or negative predictions of 0 if they are equal to or below the threshold. From this, the confusion matrix, and subsequently the positive and negative class accuracies, are calculated with Equation 3 as before. The model's performance is compared against the mean neutrino energy of each testing batch as shown in Figure 8.

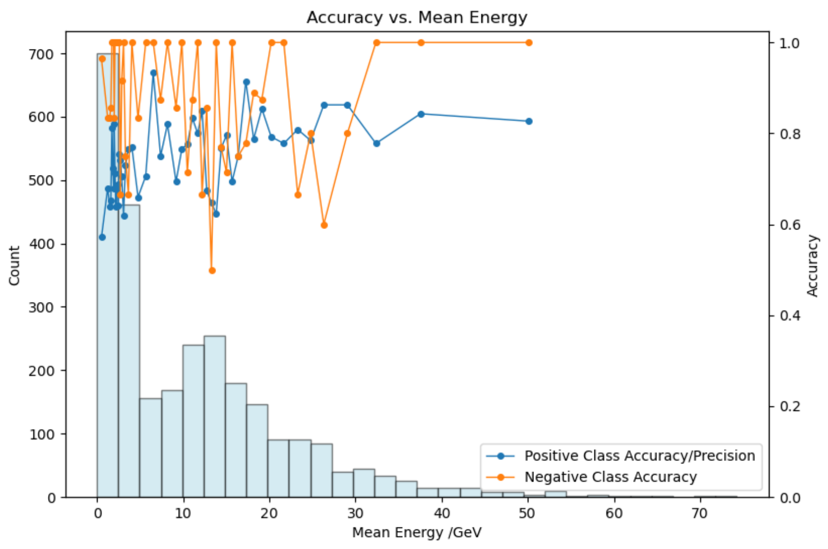


Figure 8: Plot of Positive Class Accuracy and Negative Class Accuracy against the mean neutrino energies of the testing batches and the neutrino energy histogram.

Whilst both accuracies fluctuate irregularly at the lower energies, the negative class accuracy appears to have greater fluctuation. At high neutrino energy, the model performance becomes more stable and is relatively high, surpassing 80% at high energies above 35 GeV. This suggests there may be something characteristic of the high energy testing images which the model is better suited to classifying.

4.1.2 Lepton Energy

As with neutrino energy, the model is tested on different batches of images of roughly 55 to 56 samples each, corresponding to the energy bins of the lepton energy distribution in the testing dataset. The leptons produced in the interactions that form the testing set have energies in the range of $0 - 56.98$ GeV. This performance is shown in Figure 9.

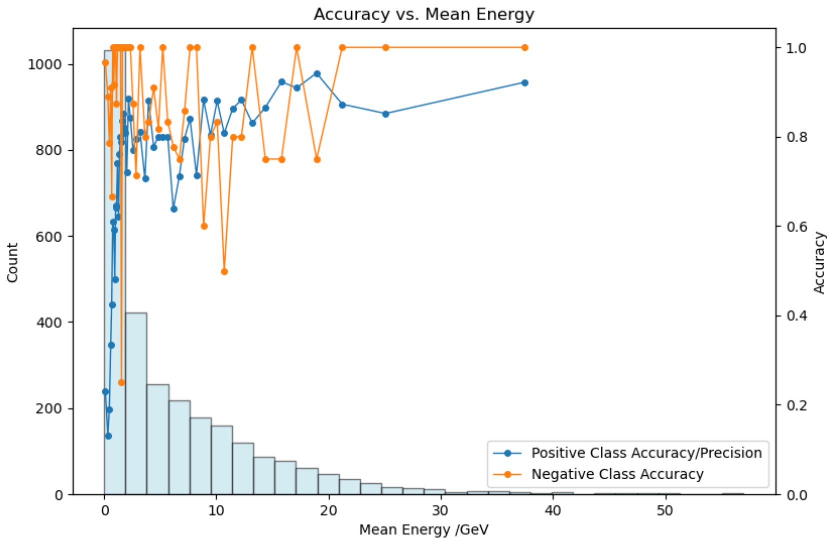


Figure 9: Plot of Positive Class Accuracy and Negative Class Accuracy against the mean lepton energies of the testing batches and the lepton energy histogram..

Similar to neutrino energy, the model's accuracies fluctuate randomly on low lepton energy testing batches, with the negative class accuracy fluctuating to a greater extent. The model performs worse on lower lepton energies compared against the lower neutrino energies. However, there is a general trend of increasing accuracy with increasing lepton energy, which is not a distinctly observable trend for neutrino energy in Figure 8. The model achieves higher accuracies at 80% or above for lepton energies higher than 20 GeV. This further supports the theory that there exists an inherent quality to the high energy event images that the model performs well on.

To gain insight into this observation, images from the high and low energy batches can be visualised and compared. One example is presented in Figure 10, which shows an interaction event at a high 45.63 GeV lepton energy and a low 0.32 lepton GeV energy.

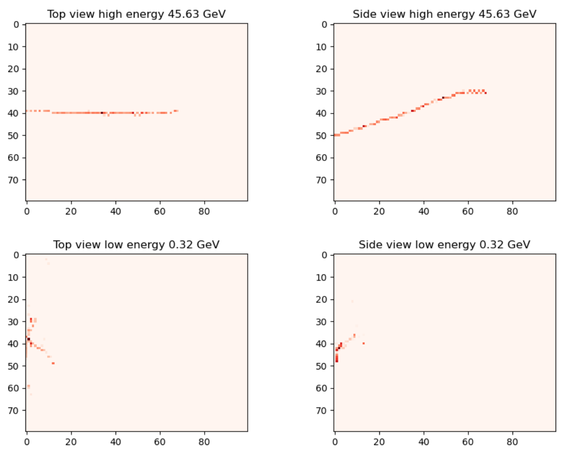


Figure 10: Plot showing the top and side views for a high energy neutrino event (top) and a low energy neutrino event (bottom).

As shown, the track is more clearly distinguished in the high energy image which makes it easier to classify. This trend is consistently observed when comparing multiple different samples from the high energy and low energy batches. This explains why the model accuracy increases for higher lepton energies. Whilst, the model performance has weak dependence on neutrino energy, it has much stronger dependence on lepton energy. This suggest that lepton energy may be a good indicator for identifying particle tracks.

4.2 Interaction Type

In addition to energy, it is interesting to examine how the classifier's performance depends on the interaction type. Table 2 shows that the interactions fall into the categories of CC QE, CC RES, CC DIS, CC Other, and NC or Other types of interaction. This can be constructed as 5 different categories of interactions. Arranging the testing images according to which of the 5 categories the interaction falls under creates 5 batches for the model to be tested on. The accuracy of the model's predictions on each batch indicates how well the classifier identifies different types of interaction. As with neutrino and lepton energy, the predictions are made on the optimal threshold. The NC or Other category is of particular interest as this contains no ν_μ CC types and therefore no positive class results. It is therefore a good example to demonstrate model's robustness when tested on data with a significantly different class distribution to the training dataset. It is also a strong indicator of how accurately the model is able to predict the minority negative class. The mean accuracies the model obtains on each interaction type is summarised in Table 4 . The breakdown of positive and negative accuracies for the interaction types are also visualised in Figure 11.

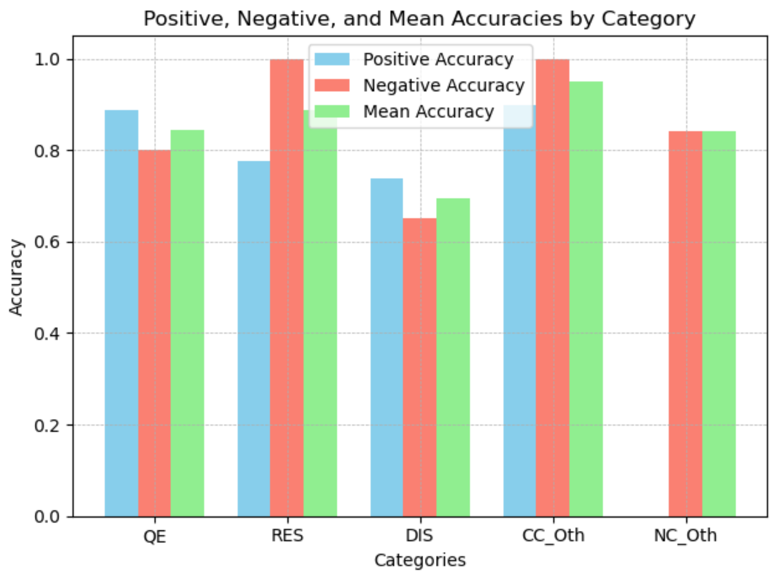


Figure 11: Bar chart showing the Positive Class Accuracy (shown blue), Negative Class Accuracy (shown red), and the Mean Accuracy of both (shown green) for each category of interaction type) As the NC Other category contains no ν_μ CC type events, it has no positive results and hence the accuracy is solely determined from the model's predictions on the negative class.

Category	Overall Accuracy
Charged-Current Quasi-Elastic	0.84
Charged-Current Resonant	0.89
Charged-Current Deep Inelastic Scattering	0.70
Charged Current Other	0.95
Neutral Current or Other	0.84

Table 4: Table of model's overall accuracy for each type of interaction in the dataset. The overall accuracy is the mean of the positive and negative class accuracies.

Encouragingly, the model attains a high accuracy of $\sim$ 84% on the NC or Other types of interaction which are solely negative class samples. This level of accuracy indicates that the model achieves suitable performance on the minority negative class. Notably, the mean accuracy for CC DIS type interactions are relatively low compared to the other interaction types. A possible reason for this is that DIS type events involve the neutrino scattering from a quark inside a nucleon which breaks up, potentially causing many particle tracks that contribute to a "messy" appearance. By comparison, the QE and RES type interactions are less complex which may generate more visually simplistic images for the classifier to identify, leading to the higher accuracies observed for these interaction types.

Overall, investigating the metadata variables reveals that the model performs better at classifying neutrino events which generate more visually clear particle tracks in the detector, which is consistent with expectations.

5 Conclusion

The primary challenge for this project is addressing the imbalanced dataset so that the model is sufficiently accurate at predicting on the minority negative class. By utilising class weight, and calculating the accuracy on an adjusted threshold, the classifier achieves a total 82% accuracy. The 84% accuracy of the model on the NC type interactions which is exclusively negative samples, confirms that the model is able to predict well on the minority negative class, hence overcoming the challenge of the dataset imbalance. The model performance appeared dependent on the produced lepton energy for the neutrino interactions as well the interaction type because these variables controlled the visual simplicity of the images. It's worth noting that common techniques to address data imbalances typically focus on problems where the positive result is in the minority class. Potentially another way to carry out the project would have been to label ν_μ CC type events as negative, and all others positive to utilise the more common techniques. Additionally, the project could be improved by looking at a larger dataset.

References

[1] Gabriel Furnieles. Sigmoid and SoftMax Functions in 5 minutes — towardsdatascience.com. https://towardsdatascience.com/sigmoid-and-softmax-functions-in-5-minutes-f516c80ea1f9. [Accessed 07-04-2024].

[2] Anuja Ihare. Significance of Kernel size — medium.com. https://medium.com/analytics-vidhya/significance-of-kernel-size-200d769aecb1. [Accessed 07-04-2024].

[3] D Rocco, MD Messier, E Niner, G Pawloski, and P Vahle. A convolutional neural network neutrino event classifier. *Journal of Instrumentation*, 11(09):P09001, 2016.