Deep Learning Applications for Supernova Neutrino Data

James Henderson

University College London, Department of Physics and Astronomy

March 5, 2022

Contents

Background

Supernova Neutrinos

When a massive star reaches the end of its life and undergoes supernova, around 99% of the energy is transmitted in the form of neutrinos [1]. The neutrino is a neutrally charged subatomic particle belonging to the family of leptons. Neutrinos only interact via the weak force due to having virtually no mass and a lack of charge, this causes them to have a low interaction probability. This means that neutrinos created by a supernova can escape the dying star easier than photons produced. As a result, the neutrinos from a supernova can reach Earth up to $2-3$ hours before it can be seen optically.

Modern neutrino detectors, such as Fermilab's MicroBooNE [2] and DUNE [3], use large quantities of liquid argon to observe neutrinos; charged particles produced through neutrino interactions ionise the argon atoms, causing them to lose electrons. A high voltage applied to the argon causes electrons to drift towards a positive anode where the energy deposited in a given time interval can be recorded. These detectors are designed for neutrinos in the range of GeV, however they can also double as supernova neutrino detectors.

Neural Networks

A neural network is a collection of nodes called neurons. These can be arranged into layers, connected to one another and controlled via sets of weights, w_i, biases, b, and activation functions, f, which allow a set of inputs, x_i, to be converted into set of outputs, $\hat{y}$:

$$\hat{y} = f(b + \sum_{i=1}^{n} x_i w_i) \tag{1}$$

Increasing the number of layers in a network can be used to allow it to perform complex tasks. In addition to layers of nodes, convolutional layers can be used to increase the power of a neural network. A convolutional layer applies a set of filters to input data, allowing key features to be picked out and learnt from. An example of a network called a multilayered perception and a set of convolutional filters is shown in Figure 1:

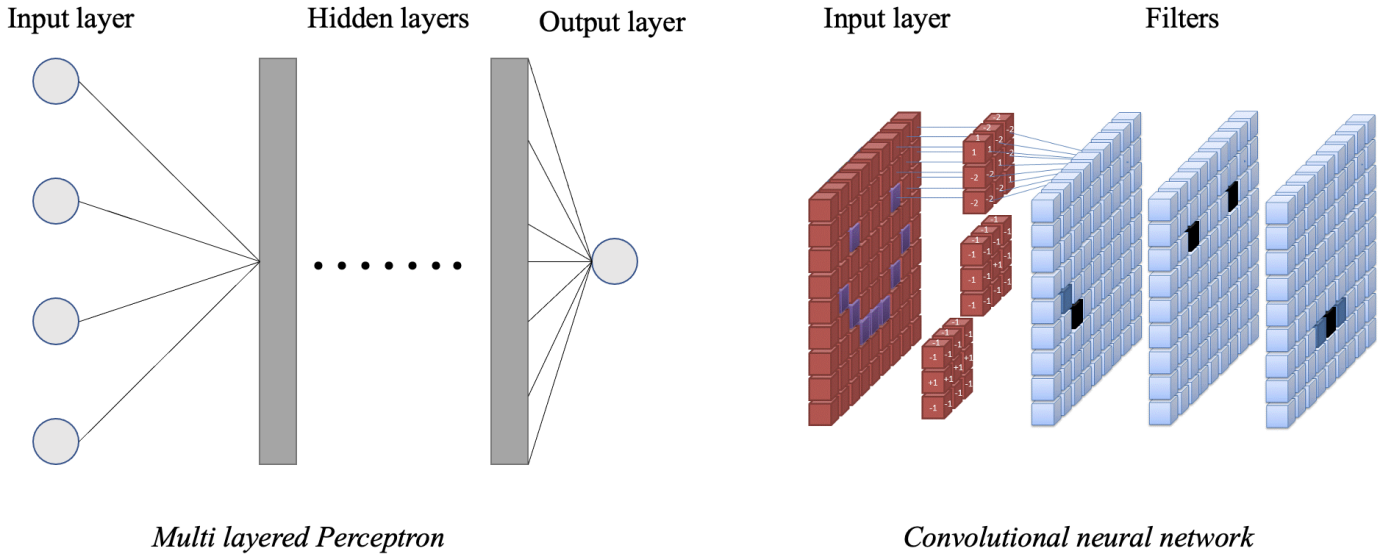

Figure 1: *An example perception and convolutional network [4]*

The output of a network can be compared to a desired result, with the difference between these two values parametrised by what is called a loss function. Through a process of minimising this loss function, a network can be trained to perform a desired task.

Introduction

This project explored the application of neural networks in the identification and investigation of supernova neutrinos. The data set for this project consisted of 10000 simulated 100 by 100 pixel images. These represented the energy deposited in a liquid argon detector in a small slice of space and time. Accompanying the images was meta data describing the particles contained within an image. An example of one of the images is shown in Figure 2:

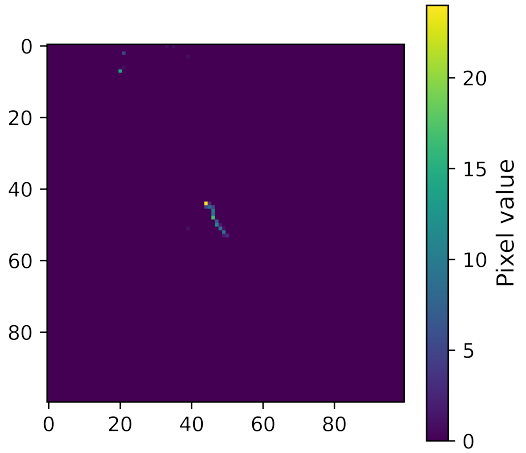

Figure 2: *Image 1 in the neutrino images data set*

80% of the data was used for training networks while the other 20% was used for testing. The aim of this project was to develop a range of neural networks to perform the following tasks:

1 Classify images containing neutrinos from blank images containing electronic noise

2 Classify images containing neutrinos with electronic noise from blank images containing electronic noise

3 Remove the noise from a set of neutrino and blank images containing electronic noise

4 Predict the energy of the electron and neutrino contained within an image

For each task, some preliminary analysis was used to set the scope of the task and to aid in network design. The finalised network designs and their performance are discussed in the following sections. All networks were produced using TensorFlow [5].

Classifying neutrinos from blank images with added noise

The first task undertaken was to build a model which could identify if an image contained a neutrino or if it was a blank image with added electronic noise.

Simulating Electronic Noise

Electronic noise was simulated by adding a value randomly sampled from a Gaussian distribution to each pixel in a given image. The probability of sampling a value, x, from a Gaussian distribution is given by [6]:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

Where μ is the mean of the distribution and σ is the standard deviation. The value of μ was fixed at 0 while the value of σ could be varied to alter the level of *noise* applied to the image. σ was defined as:

$$\sigma = \frac{noise}{4} \tag{3}$$

This was done so that the maximum pixel value in a blank image with added electronic noise was roughly equal to the value of *noise*. The absolute value of the resulting image was taken so that only positive pixel values were possible. An example of the electronic noise applied to a blank image is shown in Figure 3:

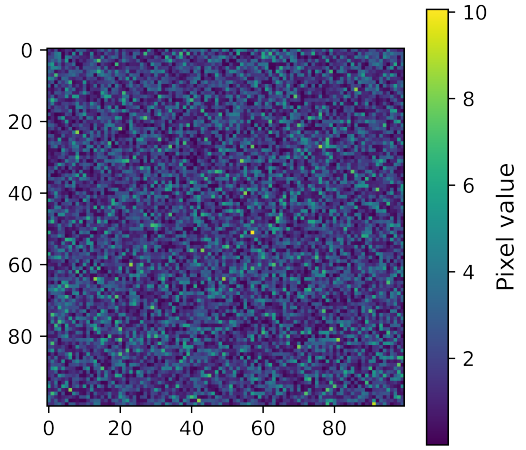

Figure 3: *Blank image with added electronic noise of 10*

Model Design

To identify if an image contained a neutrino, the network produced needed to be able to take in an image and return a 0 if a neutrino wasn't identified or 1 if a neutrino was. This kind of problem is called binary classification and hence the model developed was a Binary Classifier. Images containing neutrinos were assigned the class of 1 while sets of training and testing blank images were produced and assigned the class of 0. The blank images were then mixed with the neutrino images so that both the testing and training set had 50% neutrino images and 50% blank images.

From comparison of the neutrino images with blank images containing added noise, it was noticed that the distributions of activated (non-zero) pixels took on very different forms; the images containing neutrinos had the majority of activated pixels located towards the centre of each image, while blanks containing electronic noise had activated pixels evenly dispersed over the whole image. This can be seen in Figure 4:

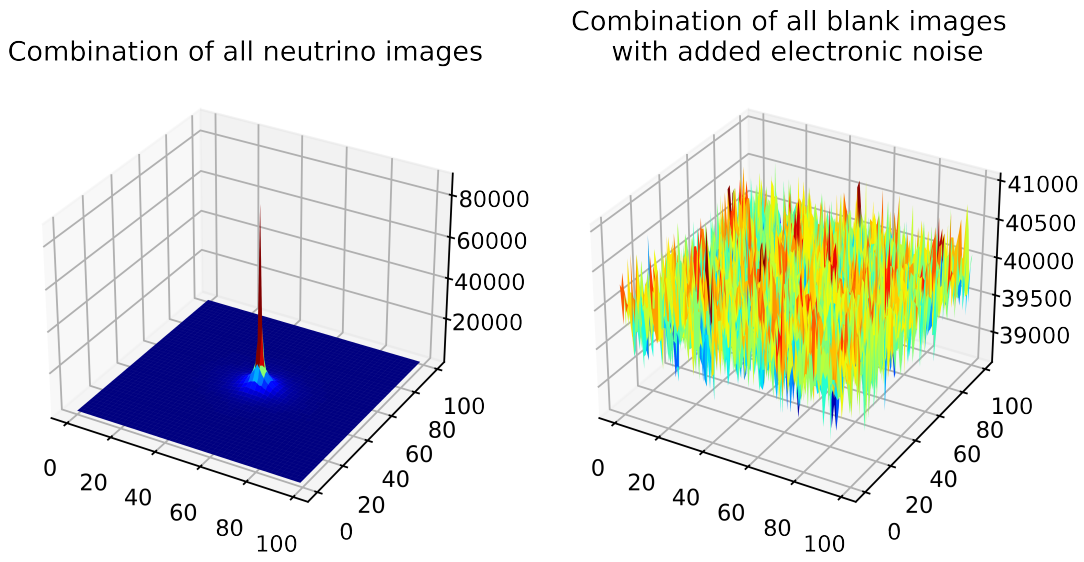

Figure 4: *Plot of pixel values with all neutrino images combined compared to all blank images with an added noise of 20 combined*

Due to this comparison, it was predicted that a fairly simple model design could be used to classify images over a large range of testing noises. The final model chosen contained an input neuron corresponding to each pixel in the input image. These neurons were fully connected to a single output neuron. Initially a sigmoid function was used for the network output as this generally used for binary classification, however, it was found that a linear activation function increased the network's performance and so this was used. The final network design for the binary classifier developed is shown in Figure 5:

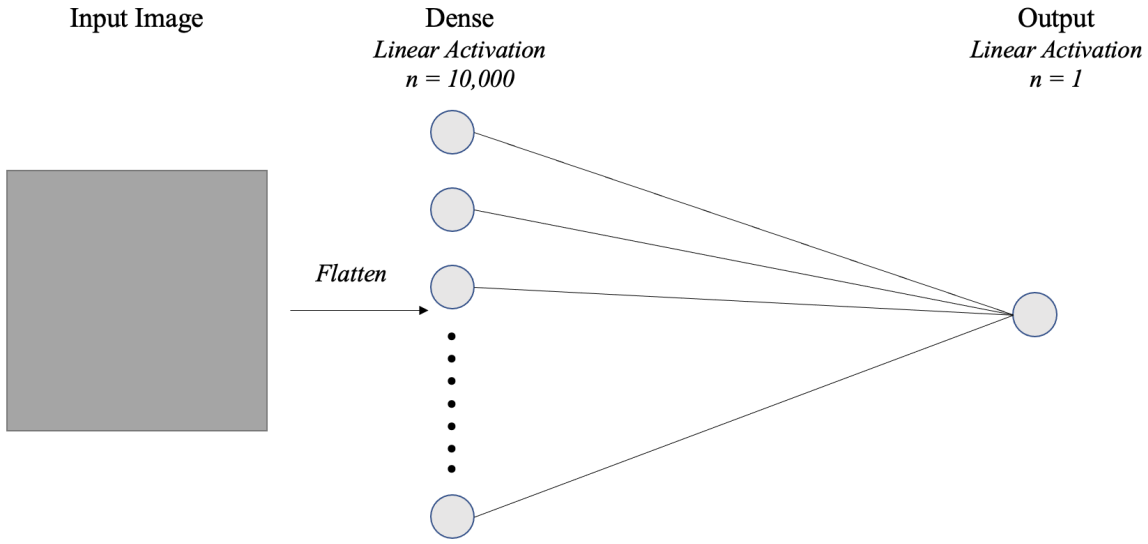

Figure 5: *Binary Classifier design*

This model would return the predicted probability, $p(y_i)$, of an image having the label, y_i =1. If a probability was greater than 0.5 then the network had classified the image as containing a neutrino, class 1, if not then network had classified the image as having no neutrino, the class of 0. The model used binary cross entropy as its loss function as this is generally used for binary classification tasks. This is given by (4), where N is the number of data points [5]:

$$H_p(q) = \frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i)) \tag{4}$$

Model Performance

Initially, the model was trained on data with no added noise. The model was then tested on 10 data sets, each with increasing amounts of noise added to the blank images. These noise values ranged from 0-200. For each testing set, the rate of accurate, false positive (the model classifies an image as having a neutrino when it doesn't) and false negative (the model classifies an image as being a blank when it has a neutrino) predictions were recorded. The results from this first test are shown in Figure 6:

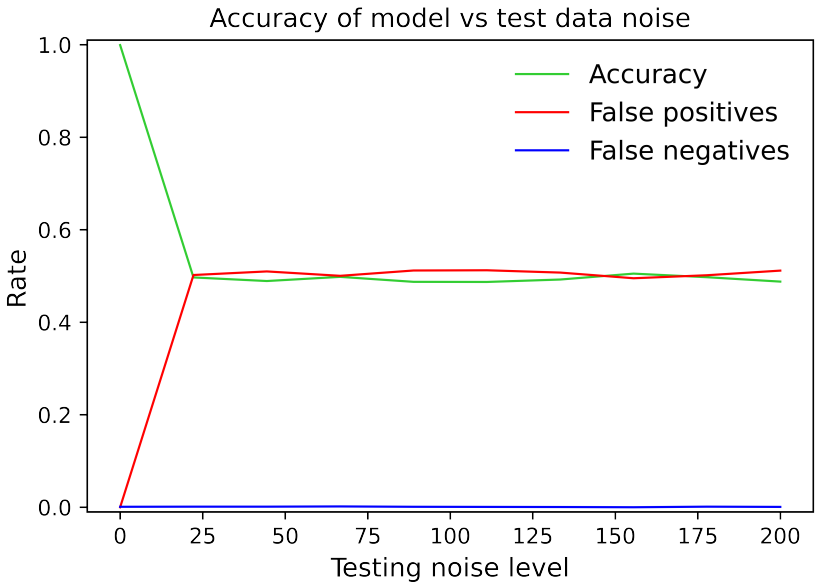

Figure 6: *Results of Binary Classifier trained on no noise tested on increasing noise levels*

The model achieved a 100% accuracy rate when the training set contained no noise, however, this dropped to 50% as soon as noise was present in the training data. From the false positive rate, it can be seen that the model was over sensitive, causing it to identify every image as containing a neutrino. To improve the robustness of the model to noise, a unique network was trained on data containing blank images with added noise for each noise value in the range. The performance of each of these models over the full noise range was then evaluated. The average performance of each model is shown in Figure 7:

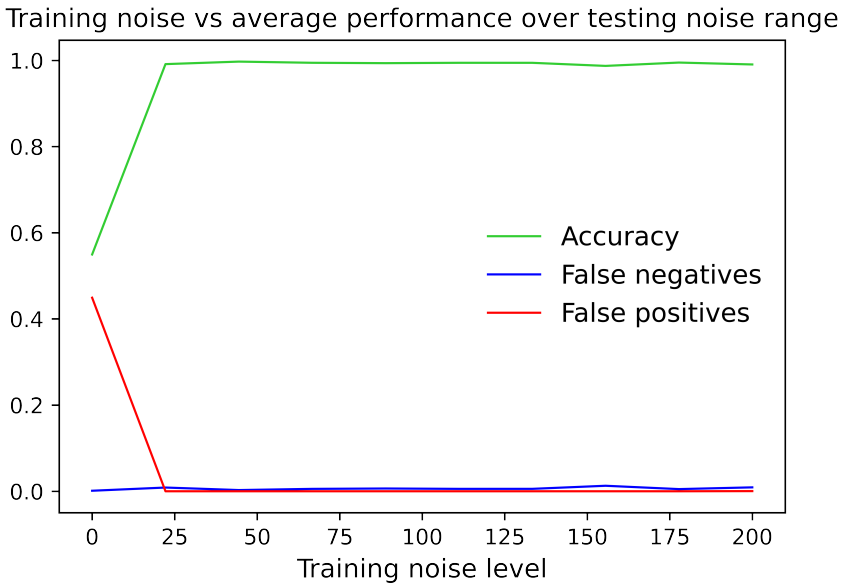

Figure 7: *Testing results for Binary Classifier trained on data with increasing noise values*

It can be seen that the models with noise added to their training data were able to achieve average accuracy rates of over 99%. This shows that the simple model design used is very robust to a large range of noises once it is trained on data containing noise.

Classifying neutrinos with added noise from blank images with added noise

The next task was to train a model to classify neutrino images from blank images when noise was added to both the neutrinos and blanks. This processes made differentiating between the two classes of images harder as the electronic noise would distort the neutrino images as shown in Figure 8:

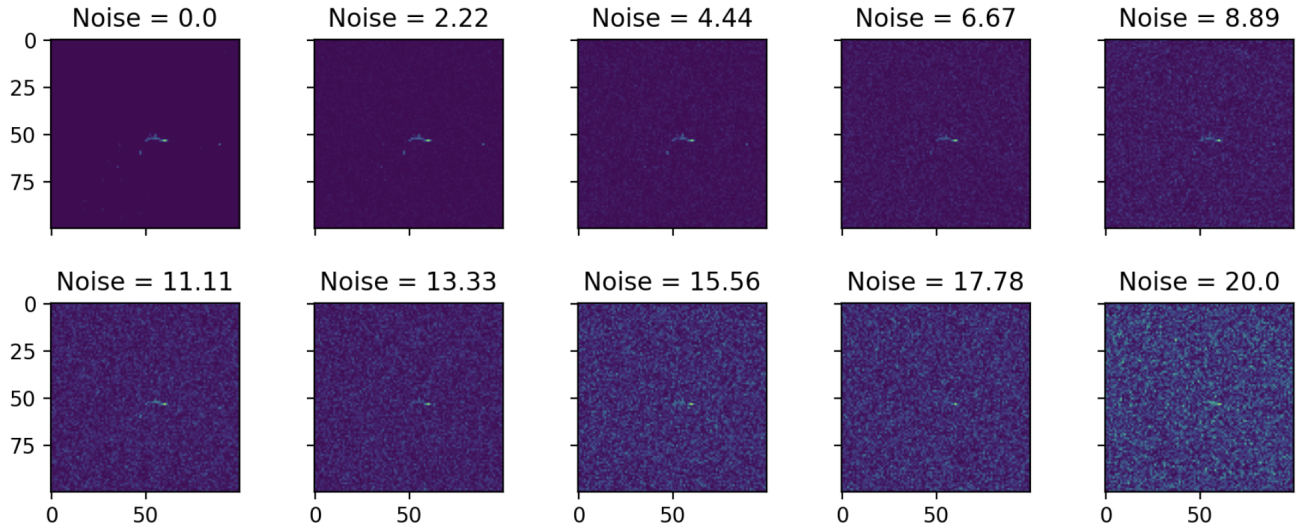

Figure 8: *Neutrino images with an increasing level of electronic noise added*

Model Design

To enhance the model, a convolutional layer was first used to extract key features of a given image. It was hoped that this would allow the model to pick out neutrinos covered by electronic noise. Alongside the convolutional layer, max pooling was used to decrease the dimensionality of the data. The output from this layer was then

flattened and fed into a series of dense layers which finally connected to an output neuron controlled by a sigmoid function to classify the image. Like the previous model, binary cross entropy was used to train the model. The final model design is shown in Figure 9:

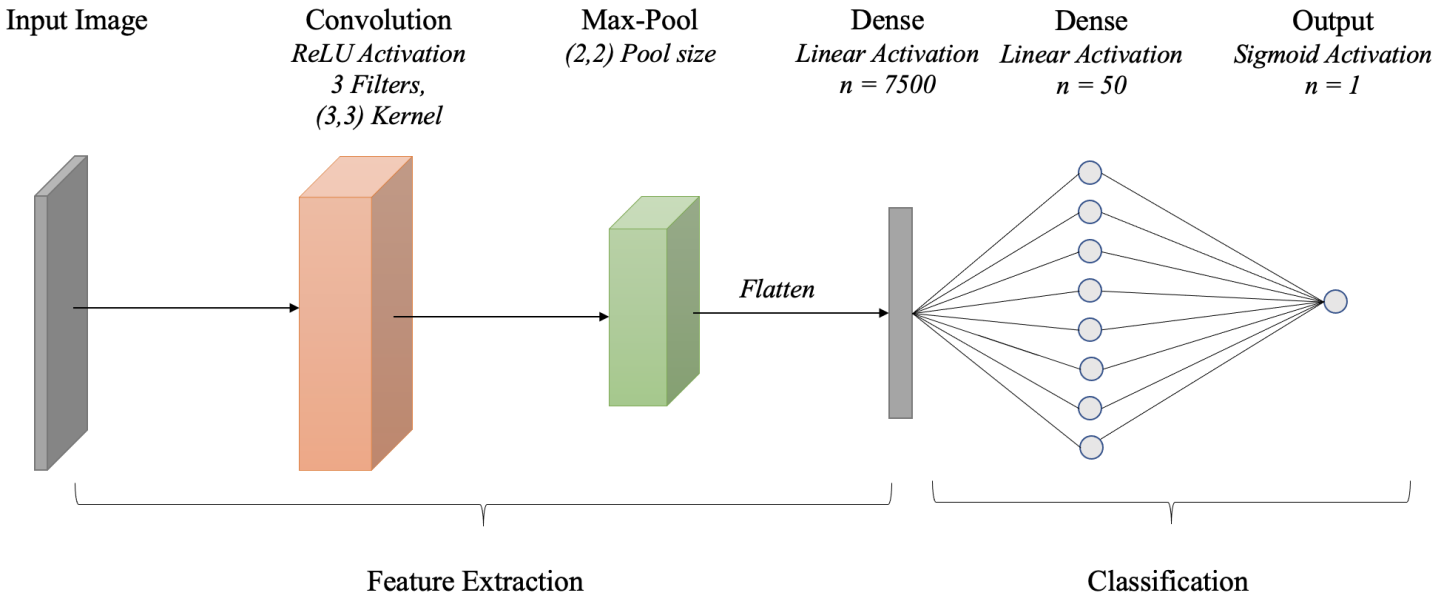

Figure 9: *Improved Binary Classifier design*

Model Performance

From investigation of the neutrino images, it was found that the modal maximum pixel value for the image set was around 21. It was expected that noise around and above this value would cause issues with images classification. The distribution of maximum pixel values can be seen in Figure 10:

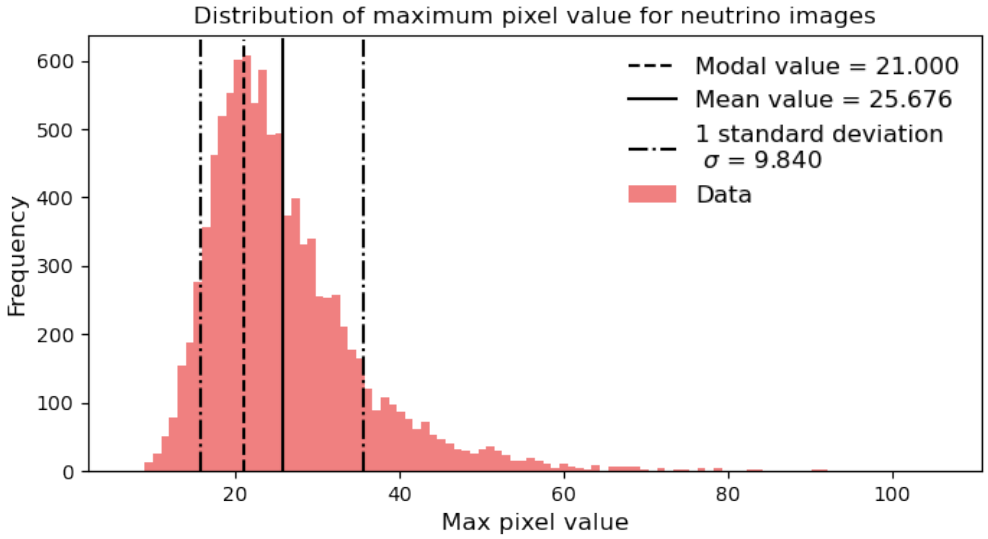

Figure 10: *Distribution of maximum neutrino image pixel values. It was expected that noise values around and above mode would cause issues with model performance*

Due to this finding, the range of testing noises was reduced. Initially, 10 testing sets were created with added noise ranging from 0-20. For each noise value, a model was trained on data with that level of added noise and then tested over the full range of testing sets. The performance of each model over the noise range can be seen in Figures 11 and 12:

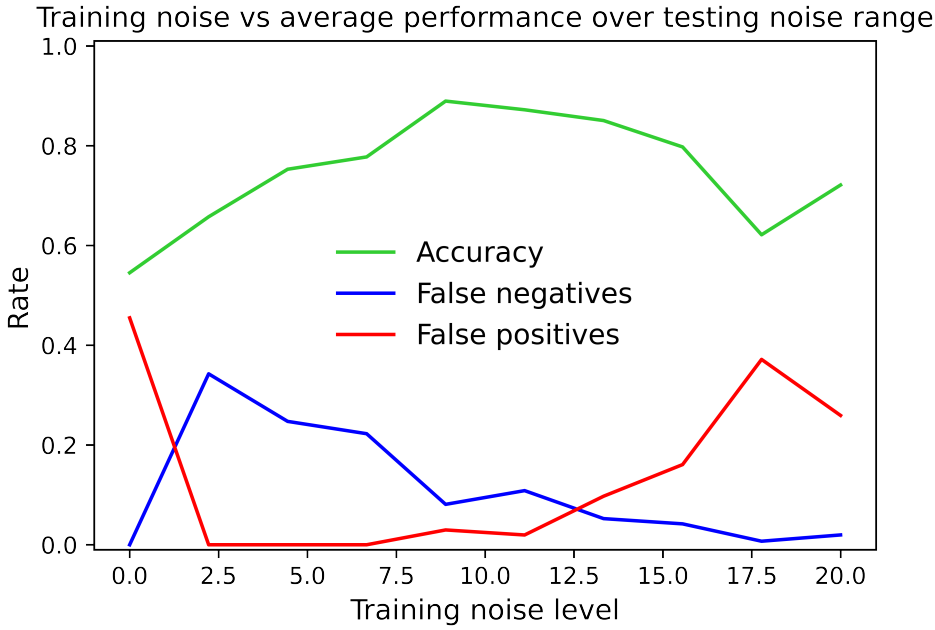

Figure 11: *Average results of Binary Classifiers trained on increasing noise levels for noise range of 0-20*

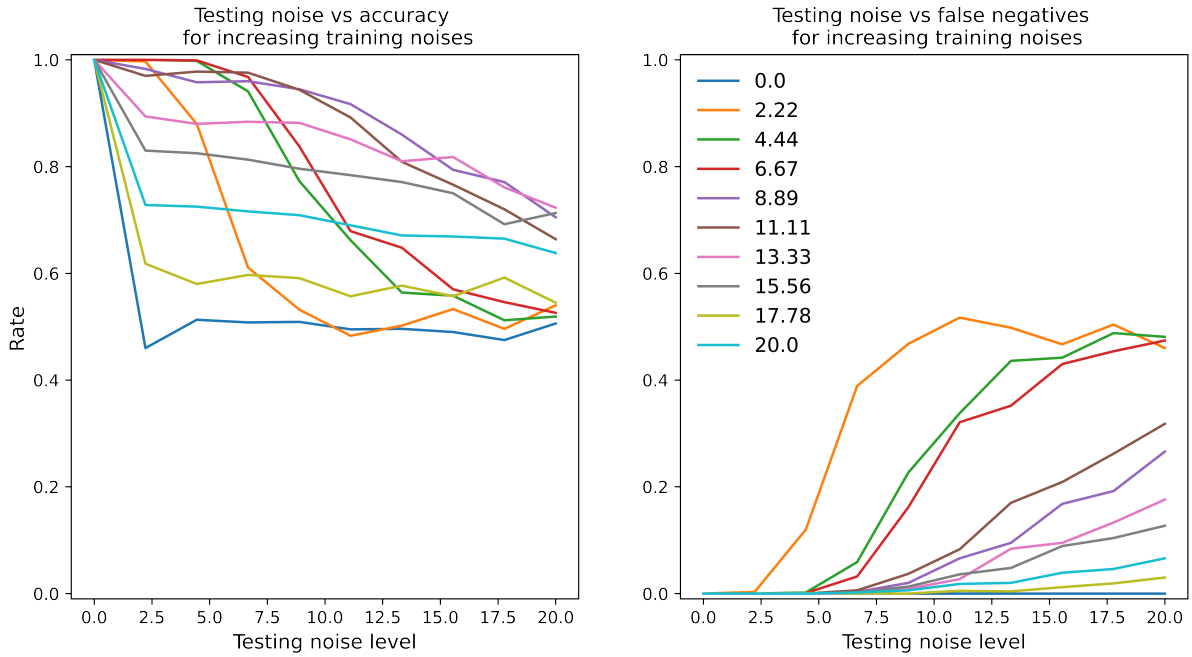

Figure 12: *Results of Binary Classifiers trained on increasing noise levels for noise range of 0-20*

Although there are some fluctuations, the general trend is that as the training noise increases the model's average accuracy increases. The false positive rate is high for the model trained on a noise of 0, however, once noise is introduced to the training data, this becomes a high false negative rate which gradually falls as the training noise is increased. As the training noise increases, the level of false positives begins to rise again. Models trained on low amounts of noise tend to work well over the low noise range, however, quickly drop off once the noise becomes higher, whereas models trained on higher noise values tend to work at a reduced accuracy, but do so over a large range of testing noises.

Models trained on lower noise values tended to experience a drop off between a testing noise of 7 and a testing noise of 10. Models trained on noises within this drop of range, such as the model trained on a noise of 8.89, tended to achieve a higher accuracy over the full noise range.

To test the model on higher noises, another 10 testing sets were created, this time ranging from 20-40. The same training process was repeated as shown in Figures 13 and 14:

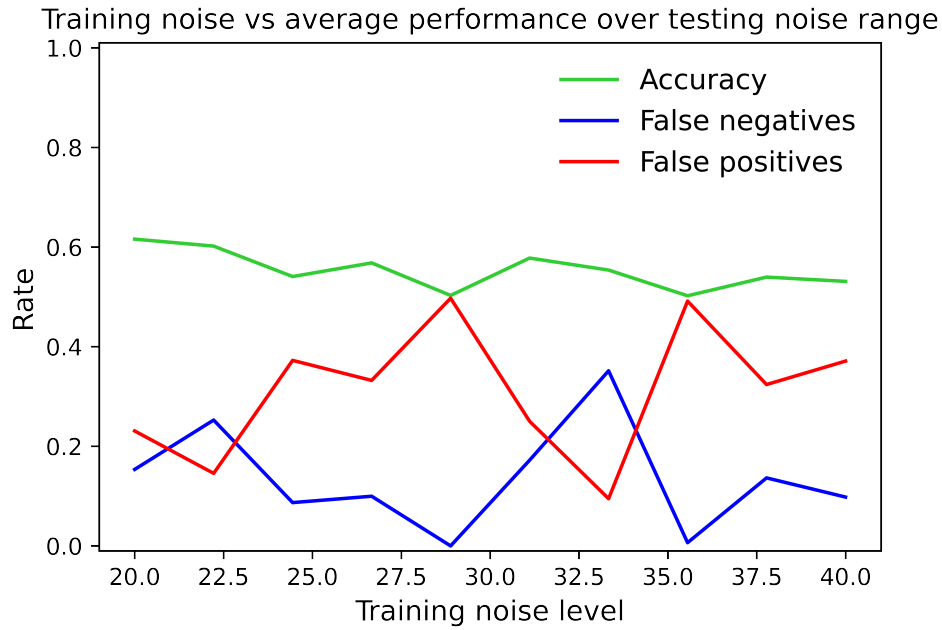

Figure 13: *Average results of Binary Classifiers trained on increasing noise levels for noise range of 20-40*

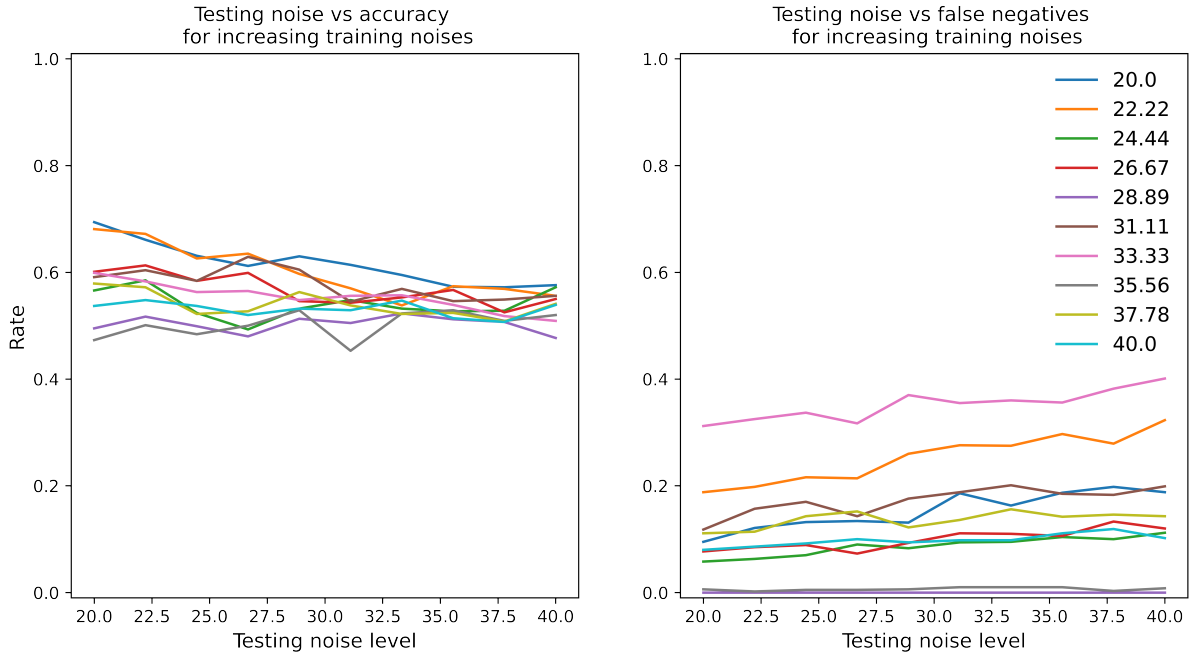

Figure 14: *Results of Binary Classifiers trained on increasing noise levels for noise range of 20-40*

Once again, it can be seen that the average accuracy of the models continue to drop as the training noise is

increased. The accuracy of all of these models also drops to around 50% as the testing noise becomes greater.

These results show that for this problem, increasing the training noise made the model more competent at higher training noises. Models trained on noises around 8-12 achieved the highest average accuracy rates. This process of increasing training noise however had a limit, with average accuracy continuing to drop as the training noise passed 12. As expected, all models began to struggle at higher noises in the range of 20 onwards.

Depending on the desired use of a neutrino classifier, it may be more beneficial to have either a model with a higher false negative rate or a higher false positive rate. For example, a model used for separating data for future analysis may benefit from having a higher false negative rate so that the data isn't contaminated by images containing no neutrinos. On the other hand, for a system used to detect supernovas early, it may be useful to have an overly sensitive model so that potential supernova events aren't missed.

Electronic noise removal

The next goal was to create a model which could be given a set of images containing neutrino and blank images with added electronic noise and would output the images with the noise removed.

Model Design

To remove the noise while also maintaining the quality of the output image, convolutional layers of decreasing numbers of filters were used followed by convolutional transpose layers of increasing number of filters. Then, a final convolutional layer with a single filter was used to output a 100 by 100 pixel image. The final model design is shown in Figure 15:

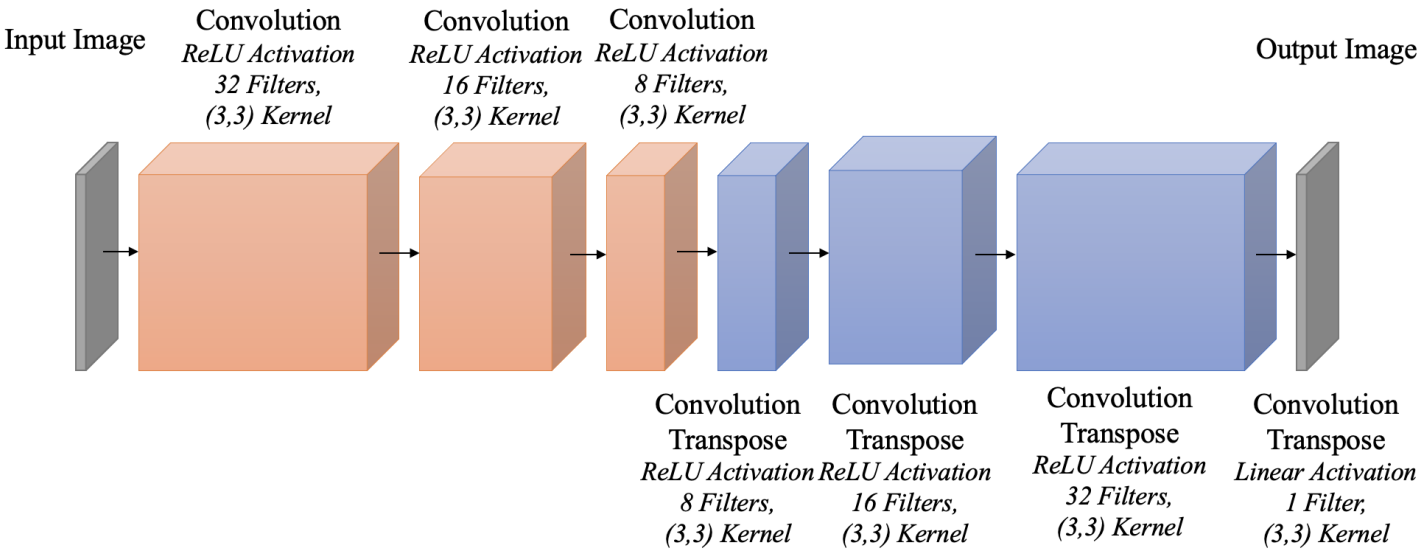

Figure 15: *Image Denoiser design*

Mean squared error was used as the loss function for this model, given by (5), where $\hat{y}_i$ is the predicted value for each pixel and y_i is the true value for each pixel [5]:

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{5}$$

Model Performance

Initially, the model was trained on no noise. It was then evaluated on batches of testing sets containing increasing levels of noise ranging from 0-20. The results and an example of the denoising of the same image with increasing

noise levels is shown in Figure 16:

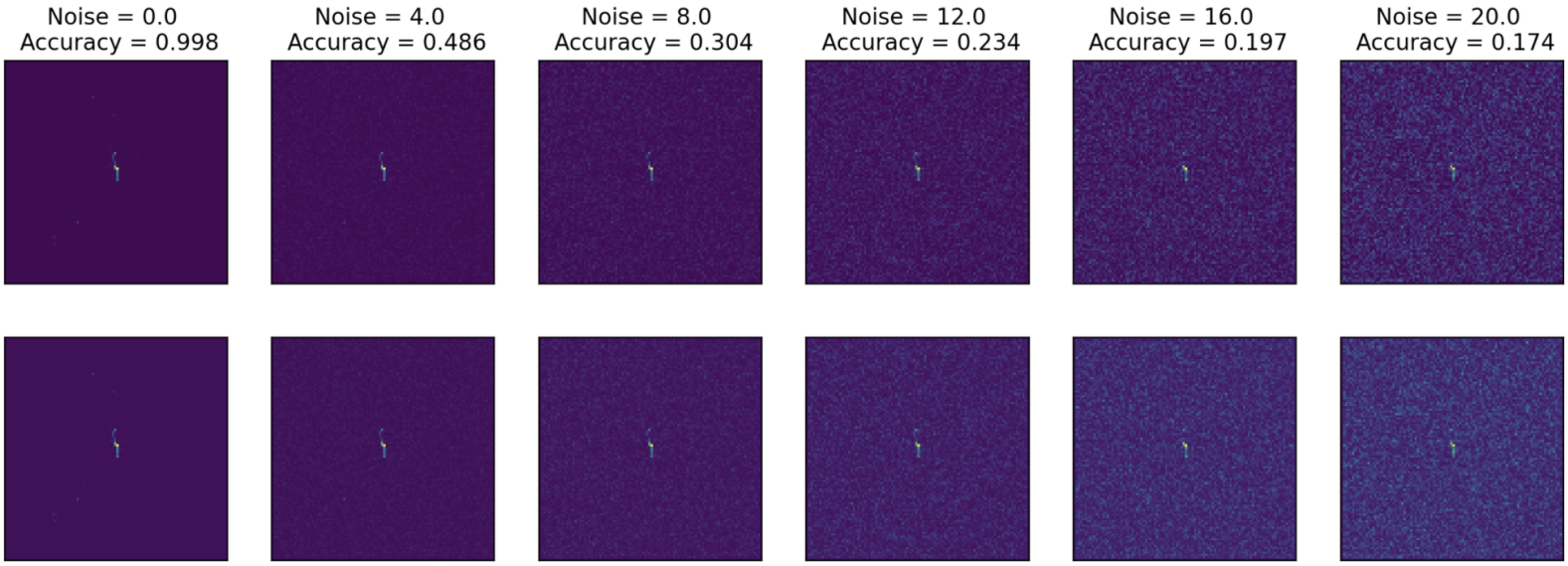

Figure 16: *Denoised images produced by Denoiser trained on noise of 0*

The model was able to accurately reproduce the input images however it was unable to remove the noise. This caused its accuracy to drop as the noise level increased. Once again, to improve the versatility of the model, multiple networks were trained on increasing levels of noise and then tested on batches of the testing sets. The results from this process are shown in Figures 17 and 18:

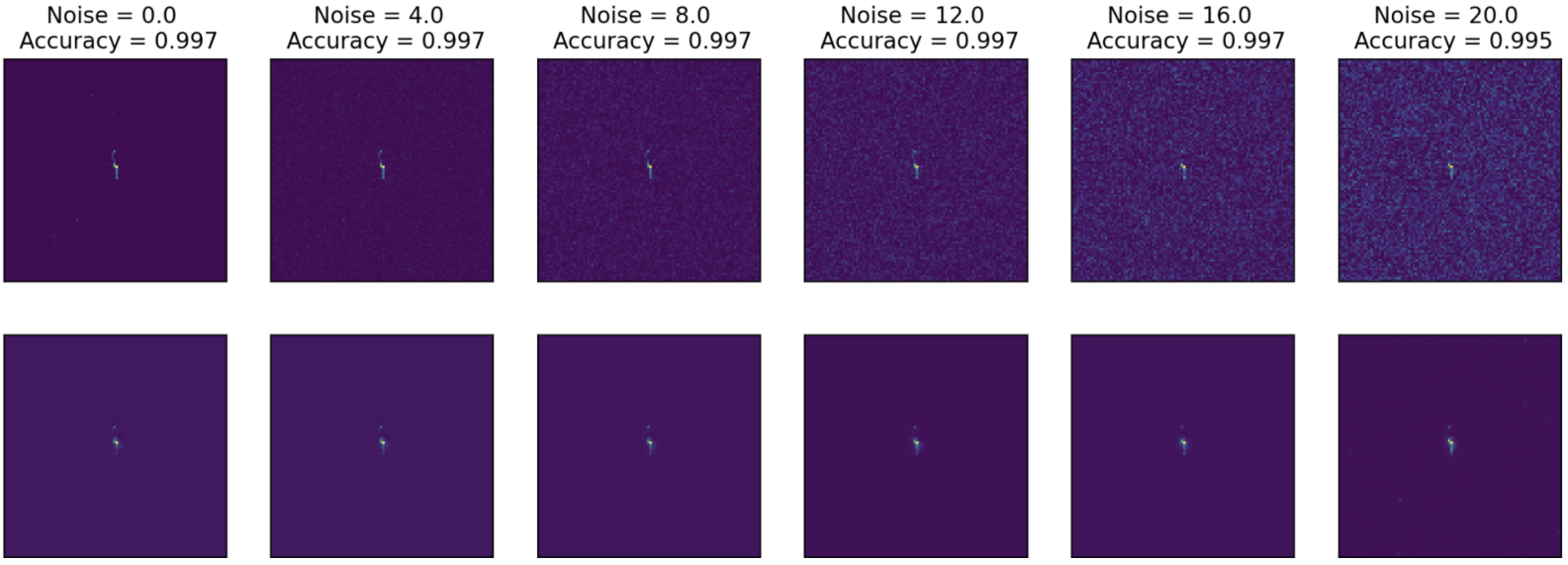

Figure 17: *Denoised images produced by Denoiser trained on noise of 20*

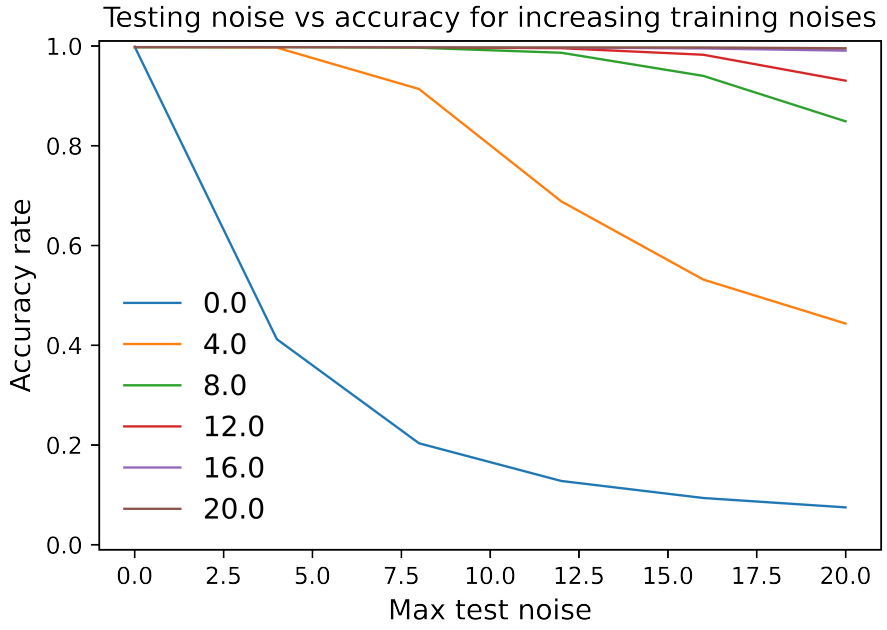

Figure 18: *Denoiser accuracy when trained on increasing noise values*

Increasing the training noise caused the reproduced images to have the majority of added noise removed - resulting in the accuracy over the full testing range to be above 99%. It can be seen that this comes at a slight cost for image detail as the network filters out activated pixels it has associated with electronic noise.

As a further test of the Denoiser, it was used in conjunction with the simple Classifier design. The Classifier was trained on a data set with no added noise. It was then evaluated on testing sets with increasing levels of noise. These sets were then passed through a Denoiser trained on a noise level of 16 . The Classifier was then tested on the denoised data sets. The results from this investigation are shown in Figure 19:

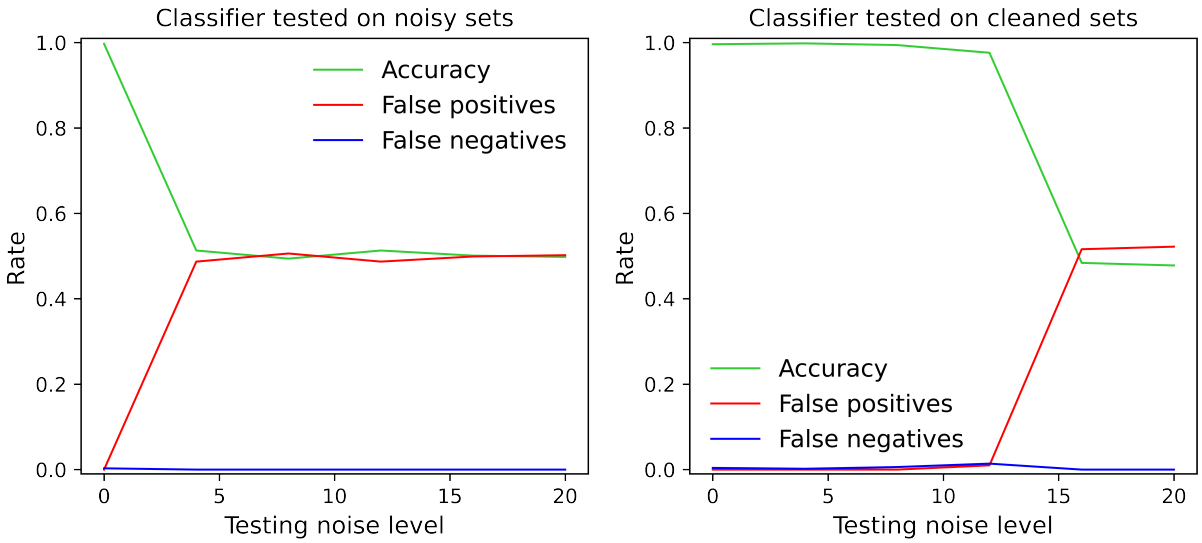

Figure 19: *Results from a Classifier trained on a noise value of 0 when tested on noisy data sets and data sets cleaned by Denoiser trained on a noise of 16*

It can be seen that the Denoiser removed the noise to such a high degree that the Classifier was able to accurately differentiate between the images for a larger range of testing noises.

Neutrino and Electron Energy Prediction

The final task undertaken was to develop a model capable of identifying the energy of a neutrino in a given image. In this task the training and testing sets no longer contained blank images.

Model Design

Investigation of the images and meta data revealed that the neutrino and electron energies had a correlation with the average pixel value in each image, as shown in Figure 20:

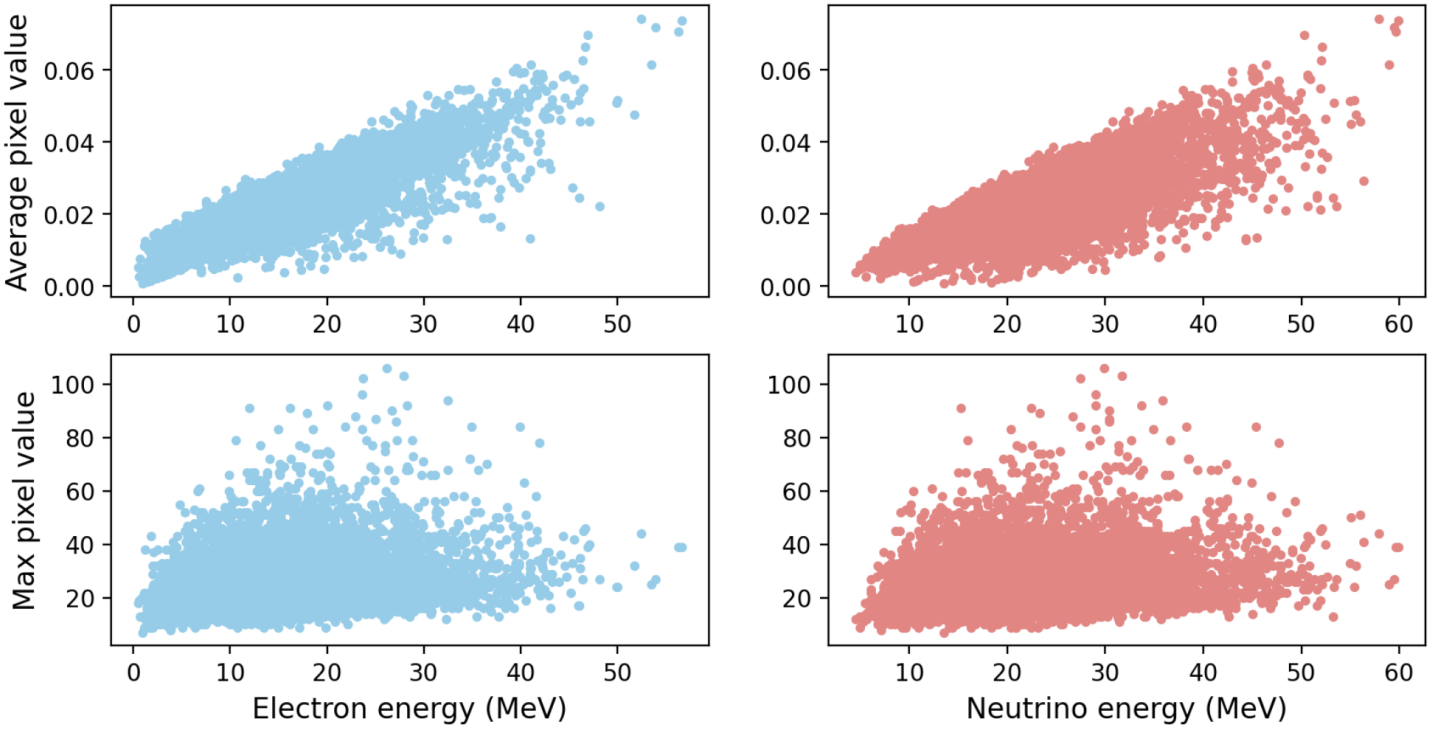

Figure 20: *Comparison of average and max pixel values to neutrino and electron energies with the left two plots showing the results for electrons and the right two showing the results for neutrinos.*

To utilise this trend, convolutional layers of increasing numbers of filters were used in conjunction with average pooling to extract key features of the images. The output from the final pooling layer was then flattened and passed into a dense layer connected to a single output neuron. The output neuron would return the predicted energy of the neutrino or electron. This model also used mean squared error was its loss function where $\hat{y}_i$ was the predicted energy and y_i was the true energy. The final model design is shown in Figure 21:

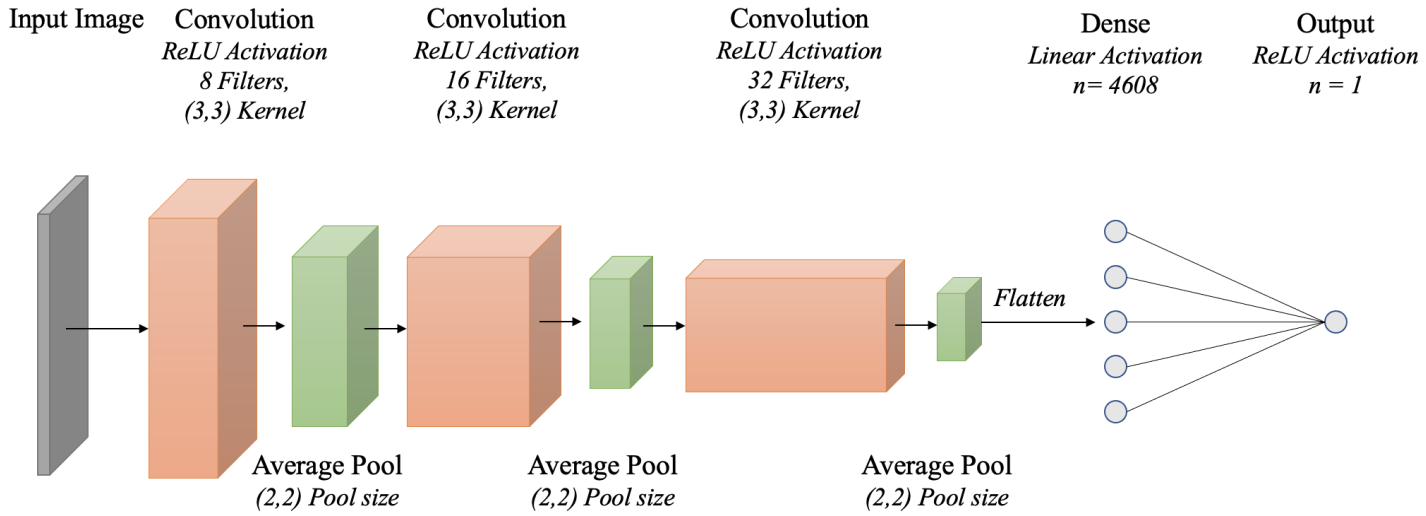

Figure 21: *Energy Predictor model design*

Model Performance

The model was first trained to reproduce the electron energies. The model was then tested on the testing set of images and the differences between the predicted and true energy values were calculated and plotted in a histogram. The result from testing this model are shown in Figure 22:

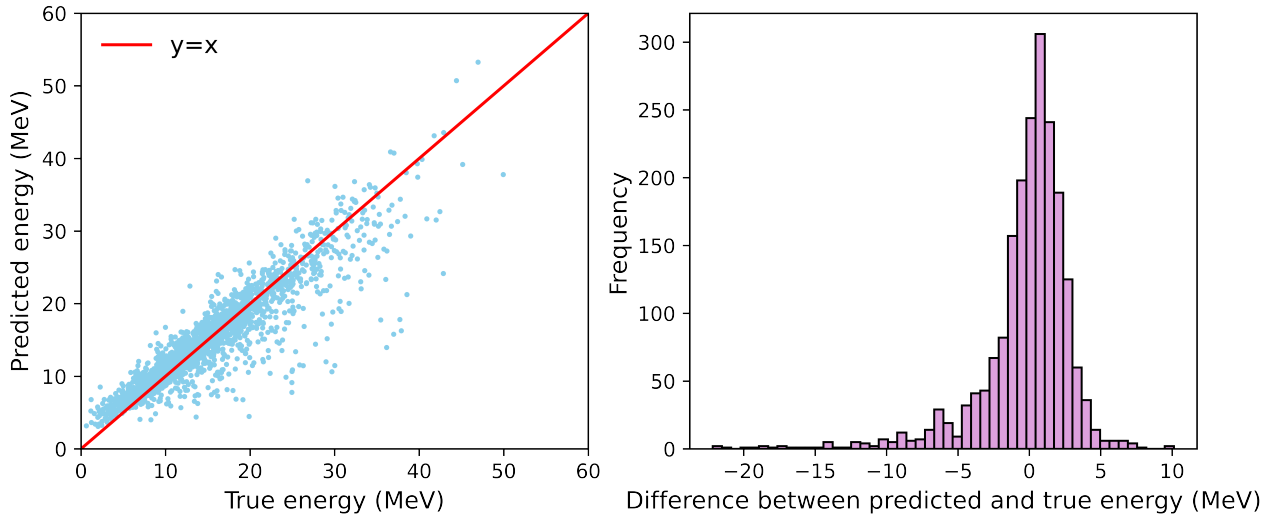

Figure 22: *Results from the energy Predictor trained to obtain the energy of electrons within images*

It can be seen that there is a correlation between the energy predicted by the model and the true energy of the electron in each image, however, there is some deviation. The modal energy difference appears to fall very close to 0.

The model was next trained to reproduce the neutrino energy. Once again the model was tested and the differences in energy were calculated. The results from this test are shown in Figure 23:

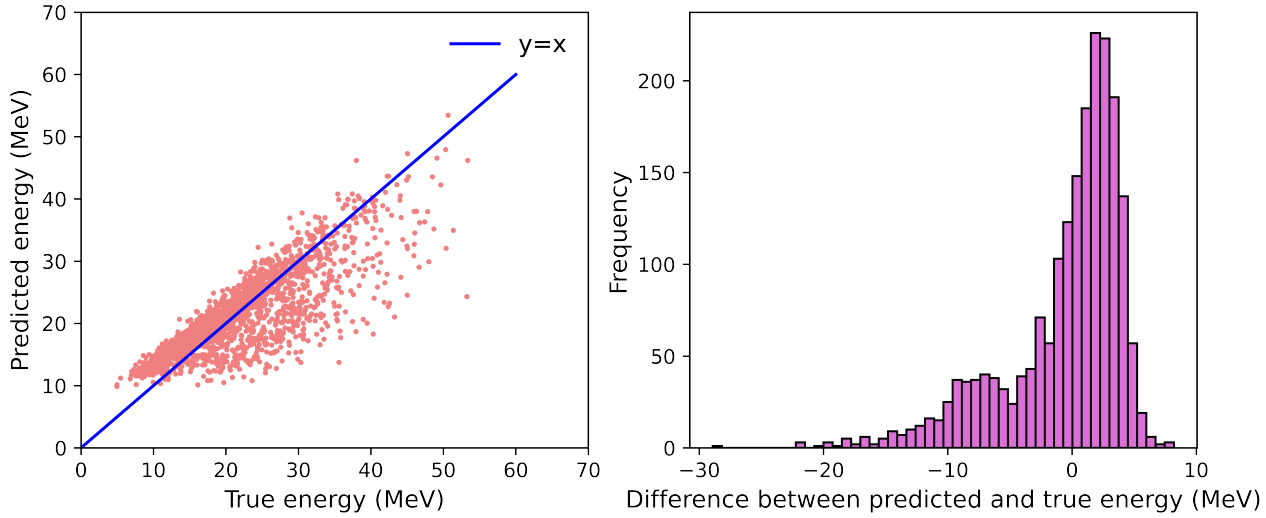

Figure 23: *Results from the energy Predictor trained to obtain the energy of neutrinos within images*

The standard deviation of the difference for both sets was calculated to give the model's resolution for both energies:

$$\text{Electron Energy Resolution} = \pm 3.2\text{MeV}$$
$$\text{Neutrino Energy Resolution} = \pm 4.2\text{MeV}$$

From the distributions for both scenarios, it can be seen that the model is better at predicting the electron energy when compared to the neutrino energy. This may be because the electron energies exhibited a stronger correlation with the average pixel values than the neutrino energies. In addition to this, there are many neutrino energies which are overestimated by the model. This causes the distribution of difference to have a slight skew to the right. There is also a group of neutrinos in which the energy is under estimated by about 7.5MeV, this causes a slightly smaller peak around this value in the neutrino difference distribution. To a lesser extent, this can be observed with the electron energies.

It may be that these results have too high of an error level for the Predictor to be used as an analysis tool for the neutrino and electron energies. However, as supernova neutrinos are expected to have lower energies than the neutrinos many liquid argon detectors are developed to identify, this form of model could be used in conjunction with a classifier to determine if a given neutrino was created during a supernova or not.

Conclusions

This project has outlined machine learning techniques for the analysis of data taken by liquid argon detectors. The findings indicate the effectiveness of neural networks in the process of removing noise from image data sets and in predicting the energy of particles contained within these images. The first two tasks highlight the power of neural networks in image classification and show how a network can be make robust by introducing noise to its training data. The third task has revealed a powerful method of electronic noise removal for liquid argon detector images, while the final task has shown the use of neural networks in extracting numerical information from an image.

The results from this investigation could be improved upon, particularly in the identification of the energies of particles contained within images. Either a completely different network design choice could work better or tweaking of network hyperparameters could bring the energy uncertainty down.

Projects such as the supernova early warning system, SNEWS [7], have been developed to act as alert systems for supernovas by utilising the detection and triangulation of supernova neutrino signals. The results of this

investigation could be used to form multiple steps in an early warning system; through a process of denoising, classification and energy prediction, an automated system could take an input image and determine if a supernova neutrino had been detected or not. This process is outlined in Figure 24:

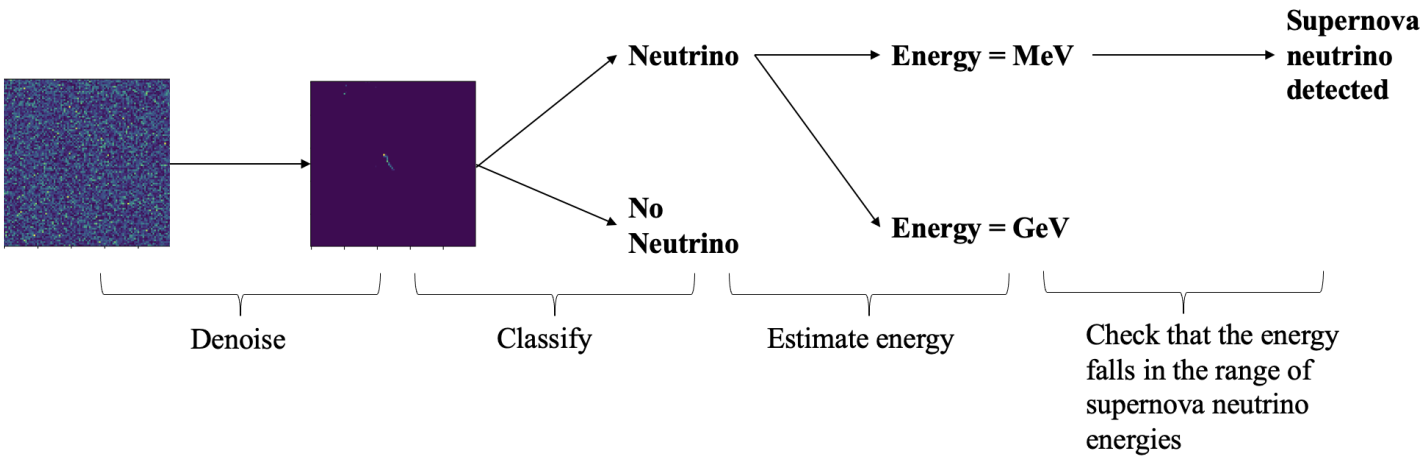

Figure 24: *An example of a possible set of neural networks used for early detection of supernovas*

The results of this investigation could also be built upon; if a network was developed to identify the direction a given neutrino was coming from, then this could be used to automatically align telescopes towards the location of a potential supernova.

In conclusion, this project has met its aims by completing multiple machine learning based tasks on a set of neutrino detector images and has brought forward suggested improvements and further investigations.

References

[1] "Supernova neutrinos — All Things Neutrino." Available at: https://neutrinos.fnal.gov/sources/supernova-neutrinos/ [Accessed: 29/12/2021]

[2] Available at: https://microboone.fnal.gov/ [Accessed: 29/12/2021]

[3] Available at: https://lbnf-dune.fnal.gov/ [Accessed: 29/12/2021]

[4] Available at: https://commons.wikimedia.org/wiki/File:3_filters_in_a_Convolutional_Neural_Network.gif [Accessed: 16/01/2022]

[5] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems", 2015. Available at: tensorflow.org, [Accessed: 16/01/2022]

[6] I. Hughes and T. P. A. Hase, *Measurements and Their Uncertainties: A Practical Guide to Modern Error Analysis*, Oxford: Oxford University Press, 2011.

[7] S. Al Kharusi et al., "SNEWS 2.0: a next-generation supernova early warning system for multi-messenger astronomy". *New Journal of Physics*, 23(3), p.031201, 2021.