Exploring supernova neutrinos using machine learning techniques

Patel, J
Dept. of Physics and Astronomy
(University College London)
London, UK

This project investigates the detection of supernova neutrinos using machine learning, focusing on the impact of electronic noise on classification accuracy. By simulating noise and training binary classifiers, the study emphasizes the importance of noise mitigation strategies. Additionally, regression CNNs are explored for neutrino energy reconstruction.

I. NEUTRINOS

Neutrinos are nearly massless, electrically-neutral fundamental leptons. There are three flavours of neutrino $(\nu_e, \nu_\mu, \nu_\tau)$ each paired with a charged lepton (e^-, μ^-, τ^-). Neutrinos interact only via the weak nuclear force, making them difficult to detect, requiring large-scale detectors and sophisticated techniques to capture the rare interactions that occur when neutrinos collide with ordinary matter. These interactions fall into two categories: charged-current (CC) processes and neutral-current (NC) processes. In CC, the neutrino is turned into its charged partner (for example, an electron-neutrino ν_e turns into an electron e^-), and in NC, the neutrino scatters but remains as a neutrino (and does not change flavour). In a charged-current interaction, to balance charge, a neutron in the nucleus will also be converted to a proton. Interactions with heavy nuclei, such as argon, with around 40 nucleons, can be complex, and produce many particles, but they will always include a charged lepton or a neutrino.

Supernovae are the most powerful cosmic source of supernova MeV neutrinos [1]. Supernova neutrinos are produced during the core-collapse supernova of massive stars and are crucial to the evolution of a massive star to a neutron star or a black hole and subsequent supernova. Neutrinos and anti-neutrinos are the primary carriers of gravitational energy (~99%) away from a star's collapsing core and help to drive the core's evolution from the initial hot state to the final cold state [2]. These neutrinos are able to escape the dense and opaque stellar medium due a very low interaction rate as they only interact via the weak force.

These neutrino emissions can hold information about the innermost core dynamics and energetics and can function as a warning signal for a supernova event. Neutrinos are detected on Earth before the main photon burst, as neutrinos can escape the stellar envelope before the main photon-carrying shockwaves can escape the stellar envelope [3]. The only observation of SN neutrinos is from the event SN1987A, where two to three hours before the visible light from SN1987A reached Earth, a burst of neutrinos was observed at three neutrino observatories. While only twenty-five neutrinos in total were detected, it was enough to pinpoint and observe SN1987A.

Today collaborations like SNEWS, a global network of neutrino experiments sensitive to supernova neutrinos, have been set up to provide an alert for a core-collapse event, such that observations of supernovae can be set up across a variety of different detectors (electromagnetic, gravitational and neutrino).

II. MODERN NEUTRINO DETECTORS

Modern neutrino detectors come in various forms and sizes, each tailored to different research goals and detection techniques. The majority of detectors consist of a detection medium that when a neutrino passes through, is excited, which is then detected.

Water Cherenkov detectors use large volumes of water as the detection medium. When a neutrino interacts with the water, it produces secondary charged particles that travel faster than the speed of light in water, emitting Cherenkov radiation. This light is detected by photomultiplier tubes around the detector, allowing the reconstruction of the direction and energy of the incoming neutrinos. An example is the Super-Kamiokande detector in Japan.

Liquid scintillator detectors consist of a volume of liquid scintillator, which emits light when charged particles pass through it. Neutrinos interact with the scintillator, producing secondary particles that generate scintillation light. This light is detected by sensor, enabling the reconstruction of neutrino interactions. Examples include the KamLAND detector in Japan and the Daya Bay Reactor Neutrino Experiment in China.

Liquid argon detectors use large volumes of liquid argon as the detection medium. When a neutrino interacts with the argon, it produces charged particles that ionize the argon atoms, releasing electrons. The most common interaction is electron neutrino capture on Argon nuclei:

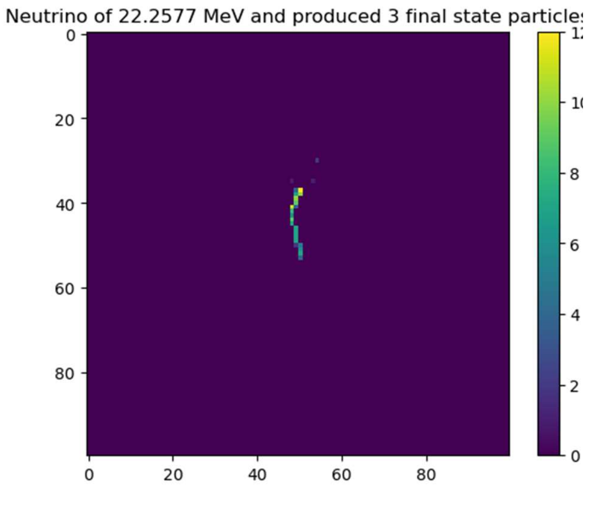


Figure 1. Sample image of simulated LArTPC data

$$\nu_e + Ar_{18}^{40} \rightarrow e^- + K_{19}^{40}$$

These electrons drift through the argon, producing a signal that is detected by wire planes or other readout systems. Examples include the MicroBooNE and DUNE experiments in the United States.

Liquid argon time-projection chambers (LArTPCs) are an example of liquid argon detectors where an electric field is applied across the volume of argon. These electrons drift towards the positively charged anode and are collected in a collection plane. The relative times of electrons reaching different parts of the plane allow a high-resolution 3D picture of the particles' paths to be built.

This projects' dataset consists of image slices of simulated neutrino interactions in a LArTPC. Each image has an associated PDG code that denotes each particle in the interaction along with its energy, and its momentum in x, y, z directions. For this simulated dataset, every image corresponds to an interaction where one of the two initial particles is always an electron neutrino, and one of the final particles is an electron (they are mostly all charged current interactions). There are also various other final particles described in each image's PDG code, but they consist of mainly photons and various other light mesons.

III. Neural networks

Image recognition is a core task in computer vision, as it enables various applications from classifying photos, identifying types of pastries, or identifying cancer cells to name a few [4]. It has since become a standard task on which to benchmark machine learning algorithms. Neural networks are loosely based on human neuron brain structure and are often employed for images recognition as they are capable of learning complex patterns and relationships from data.

Every feed-forward neural network consists of three main layers: input, hidden and output. Each layer consists of nodes, called neurons, which receive input signals, process them using an activation function, and then produces an output signal that is passed on to other neurons in the network.

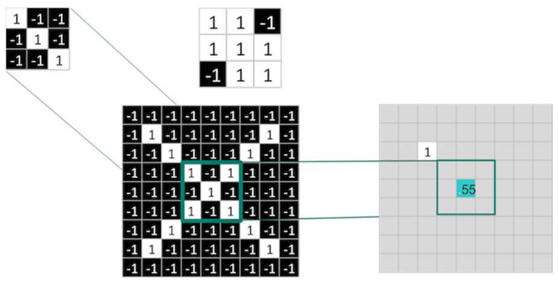
The output of each neuron is calculated by:

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

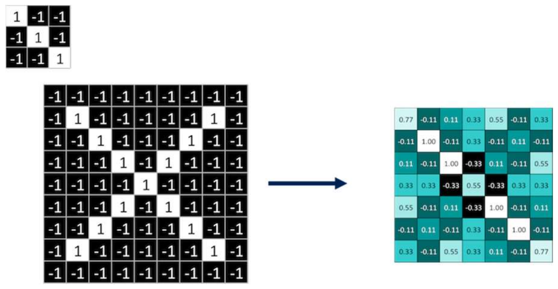
where f is the activation function applied to the summation of all weighted and biased inputs ($w_i x_i + b$).

The input layer receives the initial input data (some structured dataset, images, pixels, etc.) where each neuron in that layer corresponds to the dimensions of the input data. Next, there is at least one hidden layer where the actual computation occurs. Each neuron in a hidden layer takes an input from the previous layer, applies a set of weights and biases, and passes the result through an activation function to produce an output. Finally, the output layer of a neural network produces the final predictions/outputs of the model, where the number of neurons in the output layer depends on the task at hand.

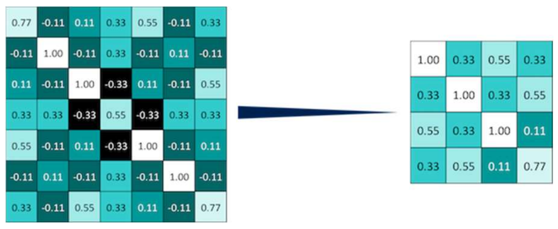
This kind of 'fully connected' neural network (perceptron) has limitations that make it unsuitable for image classification. The network will process each pixel separately and will lose spatial information about the image. For a 100x100 pixel grayscale image, the input layer will require a 10,000-neuron input layer, and each subsequent layer will increase this number. Perceptrons are not suited to image classification due to



Convolution – kernel sliding across image. Score is generated by the number of similar pixels in the kernel



Convolution – whole image as a feature map for this specific kernel. For every kernel there is a different feature map



Pooling – a 2x2 tile pools the maximum value across the feature map

Figure 2. Demonstration of convolution and pooling layer on an image [5]. When convolution and pooling layers are stacked, convolution and pooling is repeated on the feature maps.

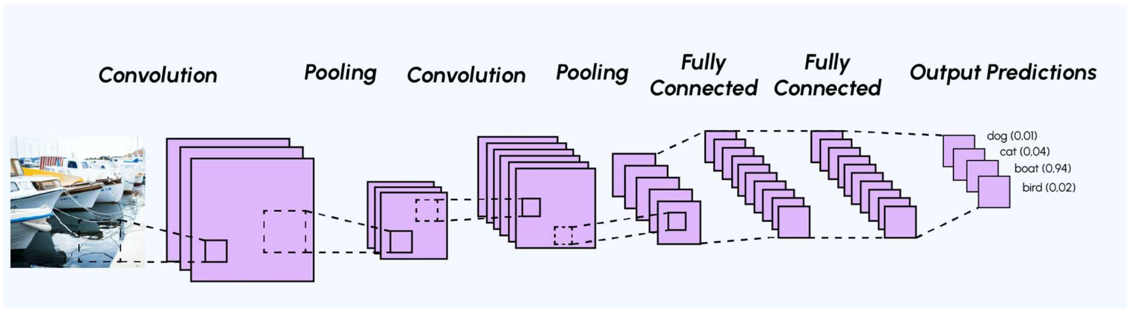


Figure 3. Example of CNN architecture. For binary classification, the very last layer only contains 1 neuron [9]

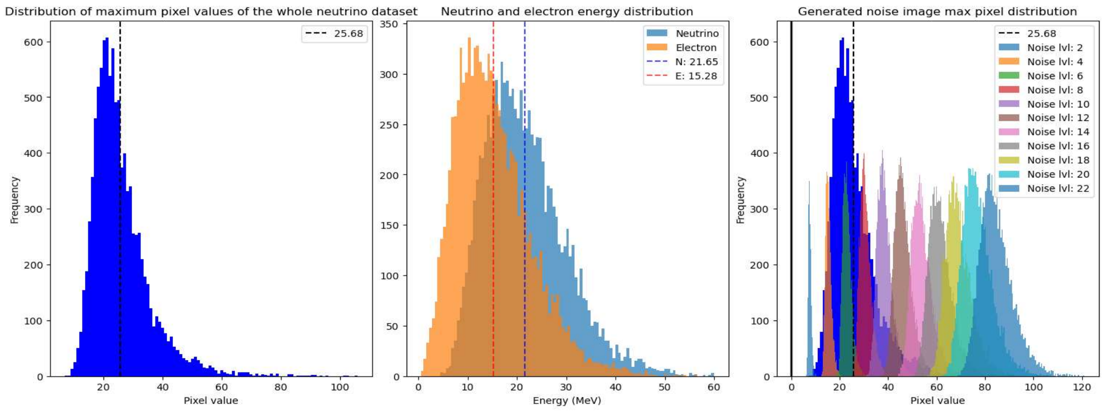


Figure 4. Plot of histograms of the maximum pixel value images in the neutrino dataset. Pixel brightness very roughly corresponds to MeV. The third chart contains the distribution of maximum pixel value from images of noise at different noise levels when compared to the neutrino image distribution.

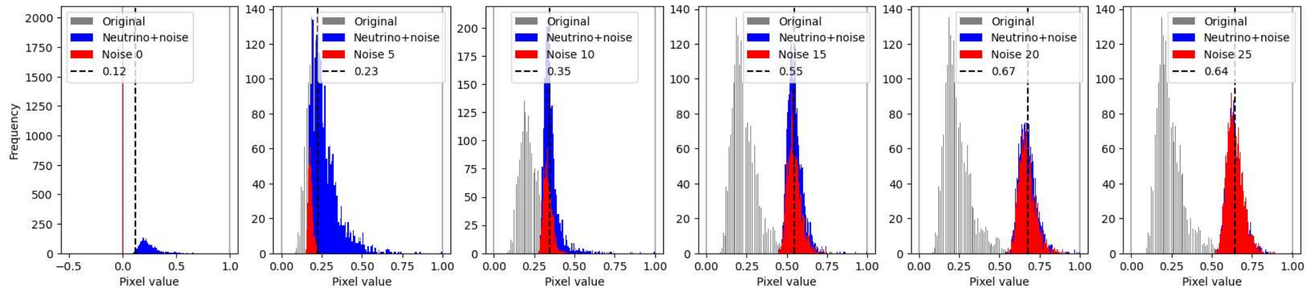


Figure 5. Histograms of neutrino images with noise overlaid, empty noise images, and the original neutrino dataset. It can be easily seen that the addition of noise (red) to neutrino images (blue) shifts the average maximum pixel value of neutrino images higher.

being very computationally expensive and the lack of spatial information being preserved. Instead, a convolution neural network (CNN) can be used as they 'look' for local features and have pooling layers to reduce the number of parameters and increase translation invariance of features. In the early 2010s, the availability of large-scale image datasets led to significant advancements in CNNs. CNN architectures with many layers, such as AlexNet, VGG, GoogLeNet, and ResNet, have achieved state-of-the-art performance on image classification benchmarks, surpassing traditional computer vision methods [6].

A CNN is a specific type of deep neural network where convolution layers alternate with pooling layers followed by some fully connected layers. There are two primary features to a CNN: producing feature maps, and classification. The convolution and pooling layers generate feature maps, extracting features of images such as edges, textures and shapes, and the fully-connected layer carries out the classification.

Convolution layers apply a kernel to an image by sliding it across the input image and learning specific features in the image in different locations. Pooling layers reduce the dimensionality of a feature map by tiling the convolution feature map and taking the maximum value of each tile (fig. 2). Stacking layers of convolution and pooling layers enable a CNN to extract low-level features (edges, textures) in earlier layers, and learn progressively higher-level features (shapes, objects) in deeper layers.

IV. BINARY CLASSIFICATION

The main task of this project, is the investigation of the effect of electronic noise on the accuracy of a binary classifier. In machine learning, binary classification is a supervised learning algorithm that categorizes data into one of two classes (0 or 1, True or False, etc.). The

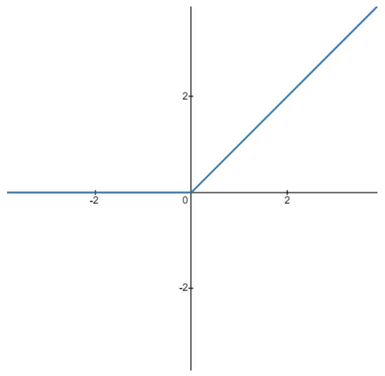
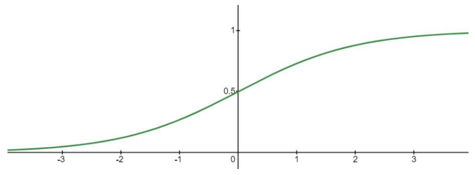


Figure 6. Chart of the ReLU function (top) and sigmoid function (bottom). ReLU does not activate for negative inputs. Sigmoid saturates at large positive or negative inputs.



network architecture of a binary classifier is very close to a CNN with the only difference being that the output layer only has a single neuron with the sigmoid activation function.

The chosen architecture for this project is based on Krizhevsky's AlexNet and consists of 7 layers: 3 convolution and pooling layers, 2 'dense' fully-connected layers, 1 dropout layer to avoid overfitting and a single output neuron with sigmoid activation [7]. All other layers utilise the ReLU activation function to keep the CNN model above 0.

V. Electronic Noise and Image Preprocessing

LArTPCs are sensitive to noise from electronics used to power and control the equipment whether it's from power supplies or voltage regulators, and it's important to mitigate or remove these noise sources as they can interfere with the detection of neutrino events [8]. For the classifier to successfully separate images of neutrino events from just noise we have to train it on images with noise added.

To simulate electronic noise, randomly samples are taken from a Gaussian distribution centred at 0, which is then added over empty 2D 100x100 slices. The images are 'clipped' at 0 such that there are no negatively bright pixels in the image. To increase the amount of noise in an image, σ is increased to increase the standard deviation of the Gaussian distribution that is being randomly sampled. In fig. 4, the histograms illustrate the distribution of increasing noise levels. The majority of neutrino images have a maximum pixel value of 25~26 and so by adding noise at a level above 10 will obscure the captured data in the images and so the model can't be expected to perform very well (fig. 6).

The training datasets are made of 2 parts. The first half of the training dataset is filled with neutrino images with some level of noise overlaid and the second half is filled with 'empty' noisy images. Whenever these dataset are generated, the appropriate training label must also be generated in the form of 0s for an empty image, and 1s for images with a neutrino in them.

For binary classification, the output neuron must have the sigmoid activation function, but this is scale-sensitive and only works in the range [0, 1]. To ensure this model can work effectively, the images in the training dataset are normalised using the min-max normalisation technique.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

This linearly scales the pixel values to between [0, 1] without altering pixel's relative positions ensuring that the overall structure and characteristics of the image remain unchanged, helping to preserve relative pixel brightnesses and other variations that could hold information about energies of interactions. This happens after noise is overlaid onto the images and is also applied at the same scale to the 'empty' noise images in the second half of the training dataset.

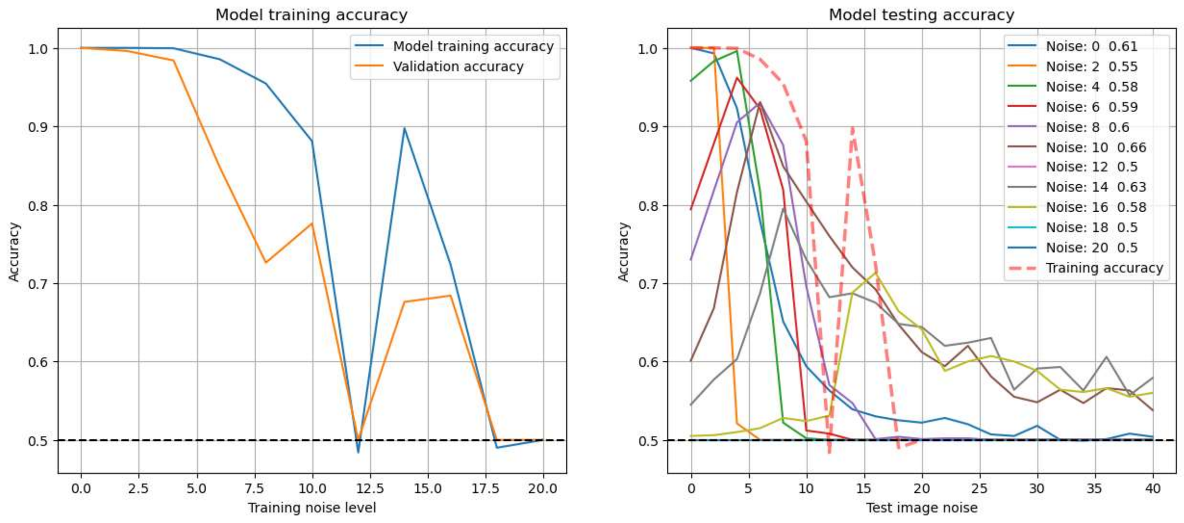


Figure 7. Plot of the model training accuracy as the training images' noise increases (left). Testing accuracy of each trained model on images with increasing noise (right). Each model is trained on a single energy level and is reset before training on a new noise level.

VI. MODEL TRAINING

The model consists of a convolution layer with 16 (3x3) kernels, followed by a (2x2) pooling layer, followed by two 64 (3x3) kernel convolution and (2x2) pooling layers, then two 64 neuron fully-connected layers with a 0.3 dropout layer between them, and a final 1 neuron output layer. This results in 460, 257 trainable parameters, which is quite computationally expensive to train. Trading-off the model's ability to pick up on patterns with computation time, the training dataset consists of only 2000 images half of which contain images from the simulated neutrino dataset, and the testing dataset is similarly made-up but with 500 images. The model is trained over 15 epochs with the '`adam`' optimiser and the '`binary_crossentropy`' loss function.

To investigate the effects of noise on the accuracy of the model, a nested loop structure was implemented to train and test the model with varying levels of noise in the input data. The outer loop iterates over a range of noise levels (0 to 20), loading initial weights for the model and generating training and testing datasets with the specified noise levels. Inside this loop, the model is compiled and trained on the training data, and the accuracy and validation accuracy are recorded. The inner loop iterates over a separate range of noise levels for testing (0 to 40), generating new testing datasets with varying noise levels and evaluating the model's performance on each dataset. The testing accuracy is recorded for each noise level, along with the confusion matrix and additional metrics, such as false positive rate and false negative rate. The results, including training accuracy, validation accuracy, testing accuracy, and confusion matrices, are stored in separate lists for analysis. The model's weights are reset after each round of training and testing. The size of the training dataset and number of epochs have been traded off in the interest of keeping processing times down, so accuracies are likely to be higher on a more powerful computer, but the overarching behaviour of the testing accuracy falling after a noise level of ten will still appear.

Here the accuracy is the metric being used to assess the performance of the model as the testing dataset contains equal numbers of neutrino and non-neutrino images. Measuring the rate of true positives, false negatives etc. of a model is necessary for dataset where there is, for example, a very small number of neutrino images and a large number of non-neutrino images. If the model

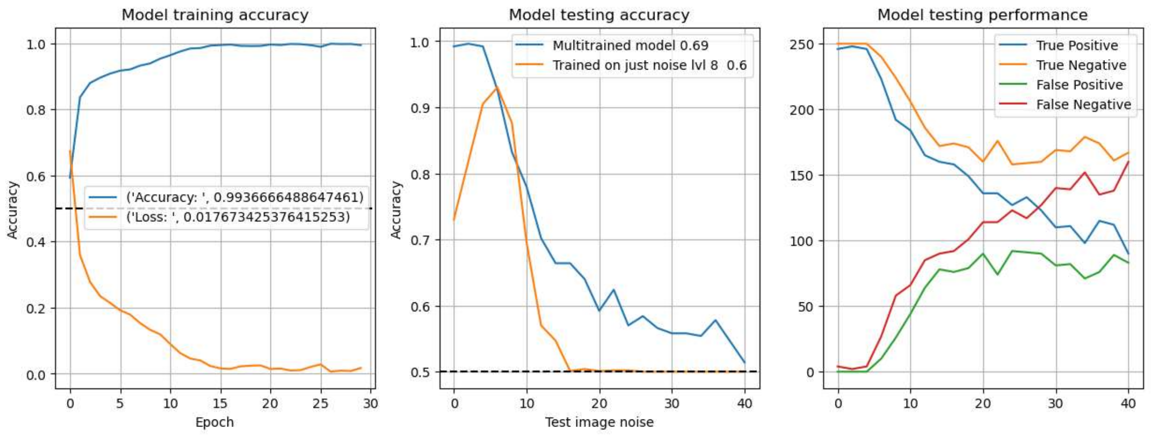


Figure 9. Training accuracy of a model trained over a range of noisy images. It achieves a higher overall accuracy compared to the best performing model from the previous regime.

incorrectly categorises all of the images as noise, this will still result in a high accuracy as the small number of false negatives is small compared to the number of true negatives. The accuracy can be calculated from:

$$accuracy = \frac{TP + FN}{TP + TN + FP + FN}$$

but for this project, the `model.evaluate` function is used to calculate this.

Plotting the results as in fig. 7 shows that the training accuracy of the model falls dramatically after noise level ten which is due to the average noisy image being added to the average neutrino image has a higher average pixel brightness (fig 6.) thus obscuring the original pixel information.

The model trained on clean neutrino images achieves a validation and test accuracy of 1.0 but this quickly falls to 0.5 at higher levels of noise. The best performing model is the one trained at a noise level of 10, maintaining an average test accuracy of 0.66, but this same model only has a test accuracy of 0.6 on clean neutrino images.

What this shows is that electronic noise affects the accuracy of a binary classifier but training a model on images with a variety of noise levels might be able to mitigate this.

The hybrid training set consists of 3000 images where every 200 images has a consistently higher noise added to them (0 to 15). This training set still retains the form where half of the image set are empty noise images. This model is trained over 30 epochs and achieves a higher overall accuracy when testing over the same noise range (0 to 40) even though it has only been trained on a smaller noise range compared to the previous model (fig. 7). Again the testing accuracy of the model still drops as the level of noise is increased as the noise begins to obscure the relative pixel brightness in neutrino images (fig. 6).

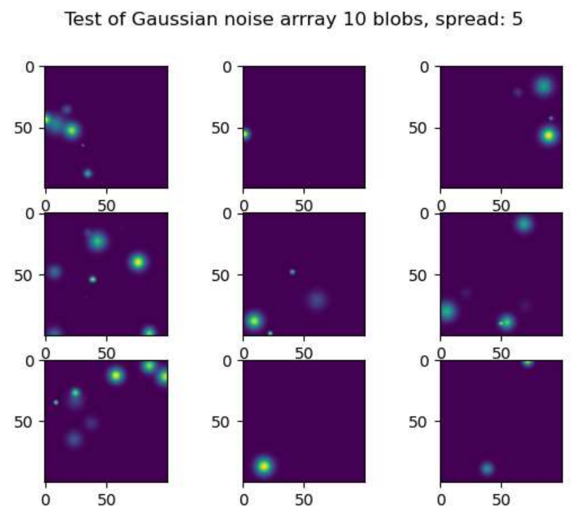


Figure 8. Sample of images with randomly generated Gaussian blobs

VII. Simulating radioactive noise

Another issue that faces LArTPCs is contamination by radioisotopes, like $^{39}Cl,\ ^{39}Ar,\ ^{35}S$ which are low-energy β emitters with a decay energy of a few MeV. They can be modelled by the addition of a 3D Gaussian 'blob' to an empty image, the equation for which is:

$$f(x,y) = A \exp\left(\frac{((x-x_0)^2 + (y-y_0)^2)}{2(\sigma_X^2 + \sigma_Y^2)}\right)$$

where x_0 and y_0 is the co-ordinate of the centre of the blob, A is the height (pixel brightness), and σ_X and σ_Y are the height and width of the blob. Given that pixel brightness very roughly corresponds to MeV in these slices, the heights (pixel brightness) of the Gaussian blobs are generated in the range [0, 5] (fig. 9). From fig. 9 the radioactive noise isn't expected to affect the models accuracy too much as the majority of neutrino images' maximum pixel value is above that of the radioactive noise.

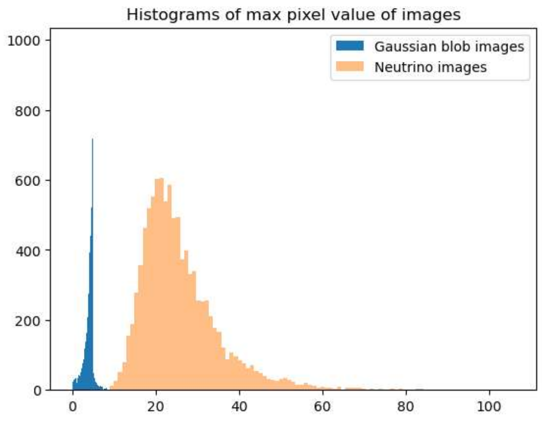


Figure 11. Histogram of the Gaussian blob noise images vs neutrino images. This is at a σ level of 5. In training and testing, $\sigma = 20$.

The same binary classifier can be used here as used in the previous section. The low energy of the radioactive noise poses very little challenge to model after it has been trained on a multitude of scenarios. The model is trained on datasets that are structured similar to before but instead the number of Gaussian blobs ($\sigma = 20$) is increased with each new dataset. The testing sets each consist of 500 images each with an increasing number of blobs.

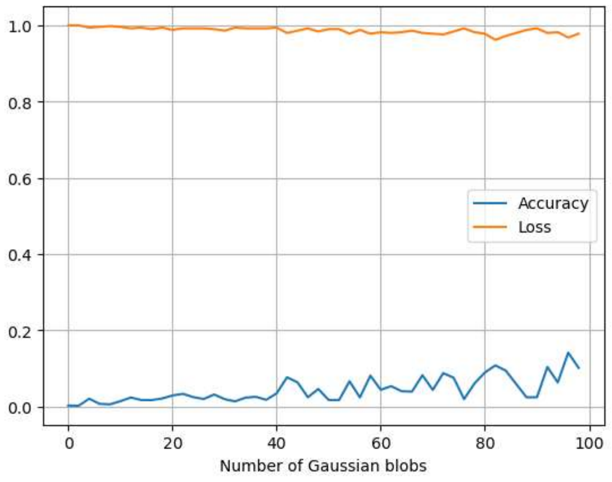


Figure 10. Testing accuracy and loss of the binary classifier on images with increasing number of Gaussian blobs.

From fig. 11 it can be seen that to affect the model's accuracy will require a large amount of Gaussian blobs to be added to the image, as the accuracy remains very high when 100 blobs are added, even at a higher σ level of 20.

VIII. Reconstructing energies

To reconstruct energies, a regression CNN is required, as the energy labels for this task are continuous and not discrete (neutrino or non-neutrino). This new model is mostly the same as the one used for binary classification, but it needs different optimisers and loss functions. To ensure as many underlying patterns as possible have been captured, the number of kernels and neurons in both convolution and dense layers have both been increased respectively. The model architecture follows the same shape except from the output layer where there are now 2 output neurons with activation function sigmoid corresponding to electron energy and neutrino energy.

For each image, 2 energy labels need to be extracted from the image PDG codes: the electron energy and the neutrino energy. Both images and energy labels are

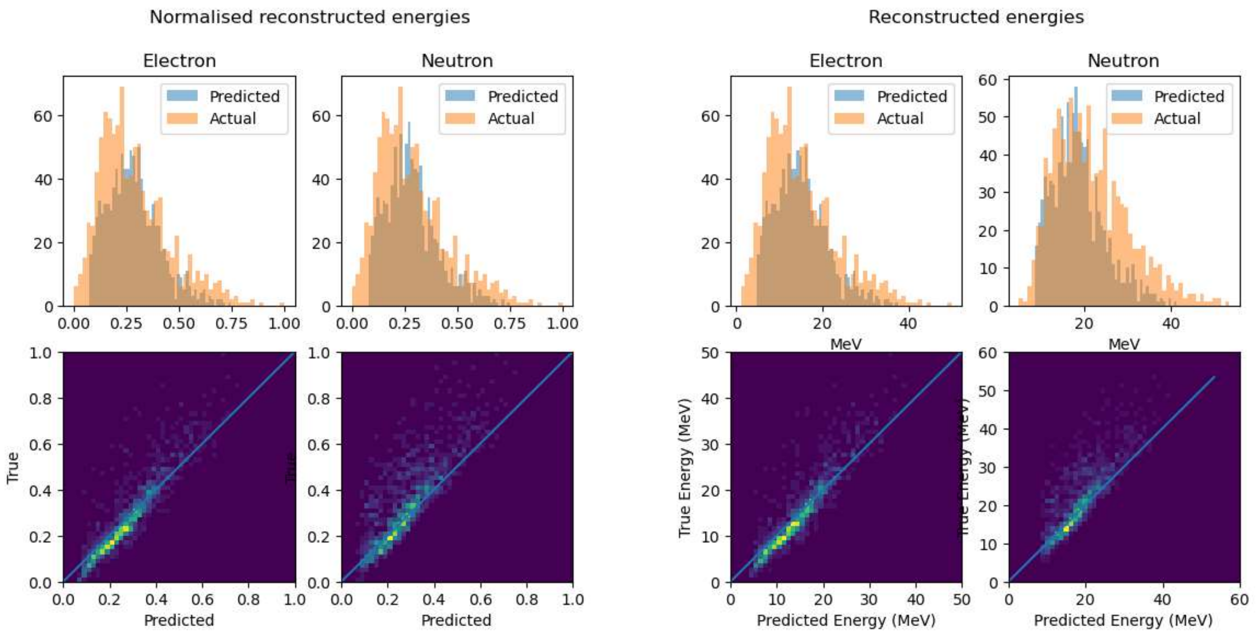


Figure 12. Histograms of true energy value and predicted energy value for reconstructed electron and neutrino energies. Diagonal blue line is linear relationship a good model will reach.

normalised using min-max normalisation but each at their respective scales. This means that the output of the model will also be normalised and so to compare the true energies at scale, they need to be scaled back up.

Using a training set of 8000 neutrino images, over 5 epochs, and using the 'adam' optimiser and 'mse' loss function, the model reaches a minimum loss of 0.0044. When reconstructing energies of 1000 unseen neutrino images, the mean absolute error reaches 0.0507/0.0788 (electron/neutrino) for normalised data and reaches 2.47/3.81 (electron/neutrino) for scaled up data (table 1).

From fig. 8 the accuracy of the model can be better appreciated through the 2D histogram. A well-calibrated model that predicts energies that closely match the true energies, will result in a strong diagonal pattern in the 2D histogram. The model here roughly achieves this diagonal pattern, but it tends to underpredict the higher energy image for both particle types, which could be due to the model not having enough training data at these higher energies.

Table 1 – Energy reconstruction performance metrics

	Normalised		**Scaled**	
	Electron	***Neutrino***	***Electron***	***Neutrino***
Mean Squared Error	0.00568	0.0145	0.0754	0.120
Mean Absolute Error	0.0507	0.0788	2.47	3.81
R² Score	0.790	0.533	0.790	0.533
RMS Error	0.0754	0.120	0.275	0.347

IX. Conclusions

In conclusion, this project has provided insights into the challenges associated with detecting supernova neutrinos using machine learning techniques. By simulating electronic noise and studying its impact on the accuracy of binary classifiers, we have demonstrated the importance of removing sources of noise in ensuring the reliability of neutrino detection systems.

Model that have training that encompass varying levels of noise to enhance their appear to be more robust when analysing images that have characteristics that haven't been trained on (e.g. more noise than what has been previously trained on). The use of different neural network architectures, such as convolutional neural networks (CNNs), has highlighted their efficacy in handling complex image data and extracting relevant features for classification tasks.

The investigation into energy reconstruction using regression CNNs has shown promising results, although there are areas for potential improvement, particularly where they underestimate the energy of higher energy particles.

REFERENCES

[1] H.-T. Janka, "Neutrino Emission from Supernovae," *Handbook of Supernovae*, pp. 1575–1604, 2017, doi: https://doi.org/10.1007/978-3-319-21846-5_4.

[2] K. Scholberg, "Supernova neutrino detection," *Journal of Physics: -Conference Series*, vol. 375, no. 4, p. 042036, Jul. 2012, doi: https://doi.org/10.1088/1742-6596/375/1/042036.

[3] IceCube Collaboration *et al.*, "IceCube Sensitivity for Low-Energy Neutrinos from Nearby Supernovae," *Astronomy & Astrophysics*, vol. 535, p. A109, Nov. 2011, doi: https://doi.org/10.1051/0004-6361/201117810.

[4] M. Turner, "BakeryScan and Cyto-AiSCAN," *Medium*, Nov. 13, 2021. https://towardsdatascience.com/bakeryscan-and-cyto-aiscan-52475b3cb779

[5] S. WASSWA, "How Convolutional Neural Networks Work.," *wasswa-sam.netlify.app*, Feb. 16, 2016. https://wasswa-sam.netlify.app/deeplearning/2017/02/26/how-convnets-work.html (accessed Mar. 20, 2024).

[6] A. Amidi and S. Amidi, "The evolution of image classification explained," *Stanford.edu*, 2012. https://stanford.edu/~shervine/blog/evolution-image-classification-explained

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2012, doi: https://doi.org/10.1145/3065386.

[8] Melanie, "Convolutional Neural Network: Everything You Need to Know," *Data Science Courses | DataScientest*, Sep. 10, 2023. https://datascientest.com/en/convolutional-neural-network-everything-you-need-to-know

[9] P. Baldi, J. Bian, L. Hertel, and L. Li, "Improved energy reconstruction in NOvA with regression convolutional neural networks," *Physical review*, vol. 99, no. 1, Jan. 2019, doi: https://doi.org/10.1103/physrevd.99.012011.

[10] R. Acciarri *et al.*, "Noise Characterization and Filtering in the MicroBooNE Liquid Argon TPC," *Journal of Instrumentation*, vol. 12, no. 08, pp. P08003–P08003, Aug. 2017, doi: https://doi.org/10.1088/1748-0221/12/08/p08003.