Author: Umberto Borso

SN: 19112316

Classification of Neutrino Interactions with Convolutional Neural Networks

Project Report

I. INTRODUCTION

One of the main challenges in high-energy particle physics (HEP) is the correct classification of particle interactions recorded in detectors. The Fermilab's NOvA experiment is focused on the detection and categorization of neutrinos, to better understand the quantum mechanical phenomena of neutrino oscillations which is not predicted by the Standard model of Particle Physics.

Neutrinos are electrically neutral fermions which interact only via the weak interaction and gravity, and whose rest mass is much smaller that of other elementary particles. Since the weak force has a very short range, and gravitational interaction is very weak at subatomic scales, neutrinos interact rarely with other particles and for this reason they are extremely difficult to detect[1].

Neutrinos exist in three different flavours: electron neutrinos ν_e, muon neutrino ν_μ and tau neutrino ν_τ, associated with the corresponding leptons. The phenomenon of neutrino oscillation is observed when a neutrino, which was created with a specific flavour, is later measured to be in a different flavour[1]. This strange behaviour of neutrinos cannot be explained by the Standard Model and it is therefore of great research interest, since it could provide insights on how to improve our description and understanding of fundamental particles and interactions.

The NOvA experiment involves the creation of an intense beam of neutrinos, which is fired trough two large detectors for a long period of time. After being created, the neutrinos are sent through the near detector, located at Fermilab in Illinois (USA), they then travel for 500 miles through the Earth, without the need of a tunnel since their interaction with matter is very rare, and they finally reach the far detector located Minnesota (USA). Most of the neutrinos located detected at Femilab are expected to be muon neutrinos, however by the time they get to Minnesota ($< 3\ ms$), some of them might have changed flavour to electron or muon neutrinos[2]. The difference ratios of neutrino flavours detected at the two different laboratories represent relevant experimental data to better understand the phenomenon of neutrino oscillations. It is therefore of great importance to develop and efficient and high performing strategy to correctly classify neutrinos.

This project is looking at simulated images of neutrino detections, very similar to those generated by the detectors of the NOvA experiment. In the simulated data, the neutrino beam is travelling in the z-direction, and for each neutrino interaction the images consist of two 100×80 pixels images: an X-view from above ($x \times z$ plane) and a Y-view from the side ($y \times z$ plane) of the tracks of particles in the detector. The aim of the project is to develop a machine learning classifier to identify ν_μ charged-current events, and to test how the classifier efficiency depends on some metavariables associated with each detection event, such as neutrino energy, lepton energy and interactions type. More details about the metavariables will follow. A complete description of the machine learning techniques employed in the development of the classifiers, including Deep Learning and Convolutional Neural Networks (CNN), follow in this project report.

II. Machine Learning and Neural Networks:

Machine Learning (ML) is a branch of Artificial Intelligence which enables computer systems to learn and adapt without following a specific set of instructions, by using statistical and mathematical models to extract information from data.

In this project we will make use of Deep Learning, a subfield of ML focused on the development of algorithms which are inspired by the functions and structure of the human brain[3]. This class of algorithms is known as neural networks (NN) which are comprised of an input layer, one or more hidden layers, and an output layer. Each layer of the neural network is made of artificial neurons which are the basic entity of any NN model. A neural network with an infinite number of neurons is able to represent an arbitrarily complex function with exact precision. The mathematical principle upon which each neuron operates is that an n dimensional input $\boldsymbol{x}$ is modified by applying a n dimensional weight $\boldsymbol{w}$ and a scalar bias b and then fed to an activation function ϕ. The output of the neuron will therefore be[4]:

$$y = \phi(z) = \phi\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b\right) \qquad (1)[4]$$

Where $z = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$ is the argument of the activation function ϕ which applies to it a non-linear transformation. Of particular interest for this project are the Rectified Linear Unit function (ReLU) and the Sigmoid function. The ReLU function[5] returns a value of 0 if the argument $z \leq 0$ or a linear output if the argument $z > 0$:

$$y = \phi(z) = max(z, 0) \qquad (2)[5]$$

The Sigmoid function instead for a given argument z returns the following[6]:

$$y = \phi(z) = \frac{1}{1 + e^{z}} \qquad (3)[6]$$

In order to build a neural network, connections between neurons need to be established. A simple structure for a NN is a fully connected network, in which every neuron is connected with all the neurons in the previous layer, which provide the input values, and all the neurons in the successive layer, which receive the output of the activation function. An example of such NN, with two hidden layers, is shown in Figure 1.

The input to the model shown in Figure 1 is processed from left to right, meaning that information flows forward through the network, and the equation describing the output is given by:

$$f(x) = \phi\left(\boldsymbol{w}_3^T \phi(\boldsymbol{z}_2) + \boldsymbol{b}_3\right) \qquad (4)$$

Where:

$$\boldsymbol{z}_2 = \boldsymbol{w}_2^T\, \phi(\boldsymbol{z}_1)\ + \boldsymbol{b}_2 \qquad (5)$$

$$\boldsymbol{z}_1 = \boldsymbol{w}_1^T x\ + \boldsymbol{b}_1 \qquad (6)$$

And where the elements of the vectors $\boldsymbol{b}_j, \boldsymbol{w}_j$ and $\boldsymbol{z}_j$ represent the bias weight and output of each neuron in the j^{th} hidden layer. In order to find the optimal value of these parameters the model needs to undergo some training, which in this case will be supervised, meaning that the model is fed with a set of input values x and their corresponding expected output $f(x)$. During the training process, the NN learns to approximate the function relating x to $f(x)$, so that, during the testing phase, it will use this approximation to predict the output of new inputs.

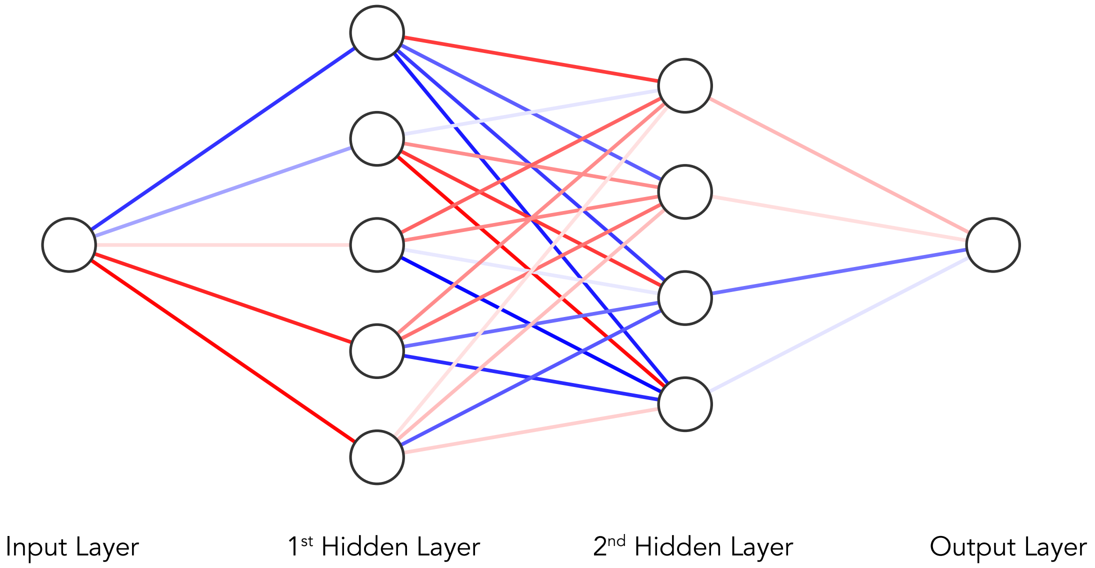

Figure 1. Visualization of Fully Connected (or Dense) NN with 2 hidden layers[7]. *Each circular node represents a neuron, and the lines represent the connection between neurons. The red lines represent positive weights, whereas the blue lines represent negative weights. The opacity of the lines is proportional to the absolute value of the associated weight.*

An effective approach to set a learning rule for a NN model is by defining an error (or loss) function, which determines the difference between the model output and the expected output for a given input. In order to find the optimal parameters which minimise the error function the model also needs an optimization procedure.

Loss functions are chosen depending on the specific task that the NN is required to perform. If the NN is built for a regressions task, i.e. for predicting a numerical output value given a certain input, then deviation of the model output from the actual value needs to be calculated, so that the model can be trained accordingly. A common loss function for such task is the Mean Squared Error (MSE) which calculates the squared deviations of the model outputs from the actual output values and computes its mean[8]:

$$MSE = \frac{1}{N}\sum_{i}^{N}\left(y_i^{actual} - y_i^{network}\right)^2 \qquad (7)$$

Where N is the number of different input values on which the model is trained.

On the other hand, a classification task consists in approximating a function that maps a variable input to a discrete output, often called category, class or label. The expected output on which the model is trained, can be encoded as a one-hot vector $y_{actual} = (0,0,\dots,1,\dots,0)$, holding the probability that the input belongs to each specific class. The position of the number 1 corresponds to the position of the correct class. The model is trained at generating a probability vector, by producing N different outputs, where N is the number of possible classes. For this kind of learning task, a common strategy is to use a cross entropy loss function which measures the difference of the output probabilities from the actual one-hot vector[9]:

$$H(x_i, y, p) = -\sum_{i=1}^{N} y_i \log\left(p(y_i|x_i)\right) \qquad (8)$$

Were x_i is the input fed to the NN model, y is the one-hot vector described above and $p(y_i|x_i)$ is the probability that the NN model output is y_i given an input x_i.

The probabilities $p(y_i|x_i)$ are generated using a SoftMax as the activation function of the output layer in the NN. This function allows to normalise the outputs of the model and turn them into probabilities associated with each class[8]. An example of a SoftMax function is the sigmoid in equation (3). The optimization procedures employed to minimise the loss function, are often based on stochastic gradient descent algorithms[8]. Gradient descent consists in a calculating the derivative of the loss function for a specific input value, with respect to the parameters of the NN (weights and biases), and then taking successive steps in the negative gradient direction to find the minimum of the function. The attribute stochastic is due to the fact that instead of exactly calculating the gradient of the loss function from all the input values, which would be computationally expensive, the actual gradient is replaced with a stochastic approximation calculated with a randomly selected subset of the original input.

The efficiency of gradient descent methods for training NNs is made feasible by the backpropagation algorithm[10]. This procedure works by calculating the gradient of the loss function, using the chain rule, starting from the last layer in the network and proceeding backwards up to the first layer. Backpropagation allows to speed up the gradient calculation process by avoiding redundant calculations of intermediate steps, which makes the calculation in the forward direction computationally more expensive.

III. CONVOLUTIONAL NEURAL NETWORKS:

CNNs are a class of Deep Learning algorithms in which every neuron is connected to all the neurons of the successive layer. Training NNs consisting of fully connected layers can quickly result in a very computationally expensive task, since each connection between neurons corresponds to a parameter in the network. CNNs provide a solution to this problem by implementing local connectivity, meaning that each neuron is only connected to a limited number of nearby neurons in the next layer[11]. The main components of a CNN are: convolutional layers, pooling layers, flattening layers and fully connected layers.

Convolutional layers have a similar goal to ordinary fully connected layers, which is to convert the original input into a more abstract representation. In order to efficiently do so, all the neurons in the layer share common weights, which are stored in a matrix element called kernel. The kernel, which typically has a much smaller size than the input image, is applied to the pixels of the image via a convolution operation starting from the top right corner of the image (in a 2D case) and shifting along the image width and length. The objective of successive convolution operations is to extract high-level features from the input image. In CNN the first convolutional layers are responsible for the extraction of low-level features, and as more layers are added to the structure, the complexity of the extracted features increases.

Another key component of CNNs are pooling layers, which are responsible for the dimensionality reduction of the image. Max pooling for example returns the maximum value from the portion of the image covered by the kernel, and therefore helps removing noise from the original input. Since pooling operations returns a single value for a wide region of the input image, they allow the CNN to develop invariancy to spatial translations of the input, which is an extremely powerful property if the model aims to determine whether a certain feature is present more than where it is spatially located[8].

The output of successive layers of convolution and pooling is usually flattened into a mono-dimensional output and fed to a fully connect layer which is responsible for the non-linear Softmax classification of the extracted high-level features. The structure of a traditional CNN including all the key steps and layers described above is show in Figure 2.

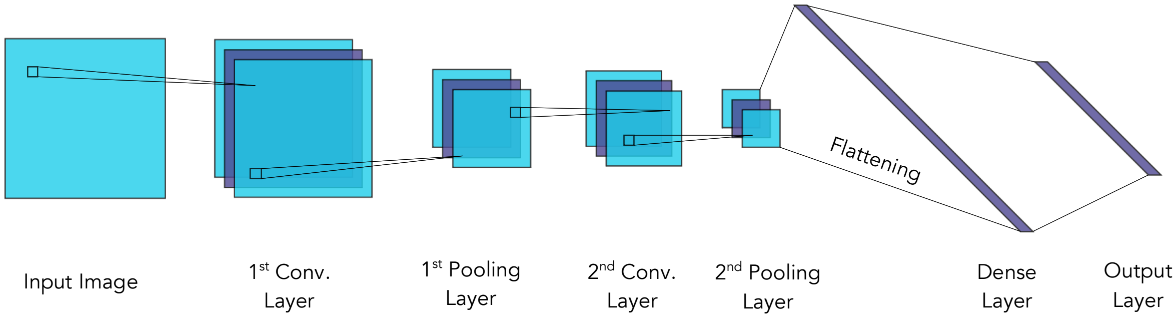

Figure 2. Visualization of the typical structure of a Convolutional NN *[7]. The input images goes through 2 steps of convolution and pooling to extract high-level features and reduce the network dimensionality. The resulting outcome is then flattened and fed to the dense layer which is responsible for the final classification of the image.*

IV. Application to Neutrino Classification:

In the first part of this project a CNN classifier was built to identify ν_μ charged-current events. The input dataset provided to the model consists in pair of images with shape 100×80 pixels and the corresponding labels are binary truth values:

$$1 \rightarrow \nu_\mu \text{ charged-current events}$$
$$0 \rightarrow \text{other events}$$

The image dataset contains simulated interactions for all the three flavours of neutrinos (ν_μ, ν_e *and* ν_τ). The possible interactions are either charged current (CC) or neutral current (NC). In CC events it is possible to determine the flavour of the neutrinos since they are turned into their corresponding charged lepton in the interaction: a muon for ν_μ, an electron for ν_e and a tau for ν_τ. Instead in NC events neutrinos are not turned into their corresponding lepton and hence it is not possible to determine their flavour.

CC interaction can be further categorised into quasi-elastic (QE), resonant (RES) and deep inelastic scattering (DIS) events, which vary depending on the complexity of the neutrino interaction. CC QE are the simplest category of interactions, in which the nucleon remains intact after the scattering of a lepton (for a muonic neutrino[12]: $\nu_\mu + n \rightarrow \mu^- + p$), in CC RES interactions the nucleon is excited into a resonant state, which then decays back to the original state[11], and finally, in higher energy CC DIS interactions, the nucleon is broken up and the neutrino scatters of a quark[13]. An example of a pair of images corresponding to a simulated CC QE interaction is shown in Figure 3.

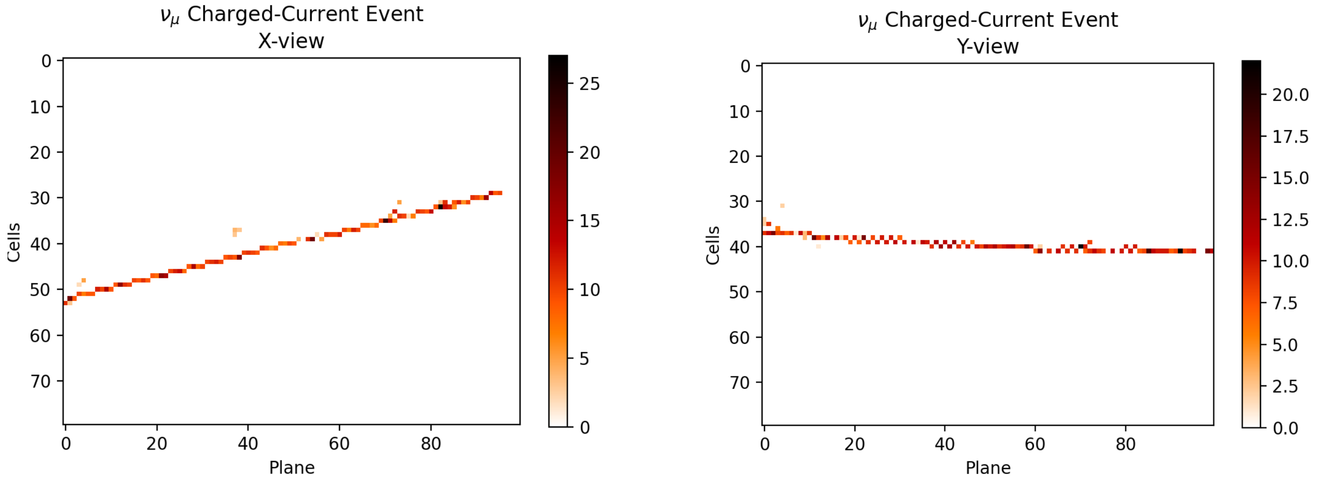

Figure 3. Simulated ν_μ CC QE event. *A simulated pair of* 100 × 80 *pixels images: a view from above (x × z plane) and a view from the side (y × z plane) of the track of a muon in the detector generated by a ν_μ CC QE interaction.*

Even though the dataset contains simulated neutrino interactions of different types and flavours, 88% of the data correspond to ν_μ CC events. Because of this imbalance in the dataset, the challenge for this classification task consists in keeping the NN model from identifying all the events as part of the dominant class. This problem can arise during the training process as the model tries to maximise the accuracy of its predictions. Therefore, if the model were to predict only 1s (were 1 is the binary label associated with the dominant class), it could easily achieve an accuracy of 88%. If this was the case the resulting model would be completely useless for classifying neutrinos, since it would always return the same output for any given input.

The complete dataset of simulated neutrino interactions includes 200 files each containing 7000 pairs of images. In order to train and test the NN model only 4 of these files were selected, since the 28000 pairs of images contained were found to be a good compromise to achieve a good classification performance and a reasonable training time. 80% of the images were used to train the model and the remaining 20% to test its predictions.

V. CNN Classifier Structure:

For this image classification task Convolutional Neural Networks (CNNs) were found to be particularly efficient given their translational-invariance and ability to recognise high-level visual features[14].

Since each interaction is associated with a pair of simulated detection images, the structure of the CNN was divided into 2 main branches, each having a different input image. The first branch is responsible of extracting high-level feature from the X-view image, while the second performs the same task on the Y-view image. After 2 successive layers of 2D convolution and Max pooling, which are responsible for the modelling of the individual images, the two branches merge together and their output is combined. A final convolution-pooling operation is performed and finally a flattening and a dense layer are responsible for the classification of the neutrino interactions. The overall architecture was inspired by the Convolutional Visual Network for neutrino interactions classification developed by A. Aurisano et al. in their paper "A Convolutional Neural Network Neutrino Event Classifier"[11]. The complete structure of the CNN classifier is shown Figure 4.

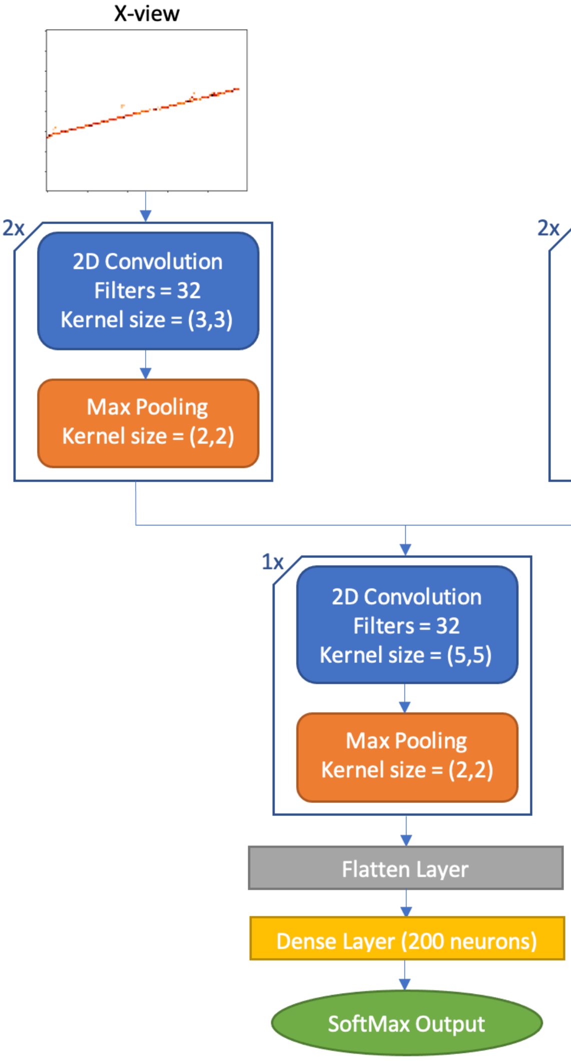

Figure 4. Diagram of CNN classifier structure. *The X and Y views of the neutrino interaction are individually fed to a branch of the CNN to be processed by 2 Convolutional layers and 2 Max Pooling layers, and undergo feature extraction and dimensionality reduction. The 2 branches are than merged and fed to another process of Convolutional and Max Pooling, before being flattened and classified by a final dense layer terminated with a SoftMax output.*

The two-dimensional convolutional layers in the branches have a 3 × 3 kernel size. An odd number as kernel size is usually preferred to an even number, such as 2 × 2 or 4 × 4. This is because odd size kernels symmetrically divide the pixels in the previous layer around the output pixel. Without this symmetry it would be necessary to account for the distortions across the layers which take place when an even kernel size is used[15]. Furthermore, two successive 3 × 3 kernel size convolutions were implemented, instead of a single larger size convolution. This is because stacking multiple layers of smaller size convolutions is found to be more efficient than having a larger size convolution[16].

However after the two branches merge a 5×5 kernel size convolutional layer was implemented. Even though this might not be as efficient as two smaller convolutions, larger kernel sizes are better at identifying spatially wider features that might have been previously missed by smaller kernels.

Activation Functions:

All the layers in the CNN use ReLU as activation function, except for the final SoftMax output layer, which uses a Sigmoid function. The ReLU, described in equation (2), is the most popular activation function for classification tasks, as it ensures better accuracy and network convergence speed, when compared to other activation functions, such as Tanh and Sigmoid[17].

Loss Function, optimizer and metrics:

The chosen loss function for this first classification task is binary cross entropy, which corresponds to an adaptation of the general cross entropy loss function in equation (8), to a classification task with only 2 possible classes. The implementation of this loss function also allows to introduce the number of true negatives as a metric of validation of the model. This metric helps identifying whether the model is always predicting the dominant class, as in that case the number of true negatives is always 0. The overall accuracy of the model is used as a second metric of evaluation.

To to minimise the loss function during the training process, the optimizer chosen is "adam", a computationally efficient algorithm based on stochastic gradient descent[18].

Number of epochs, steps and batch size:

The model was trained for 20 epochs, and 400 steps, using a batch size of 30 images. These were found to be the optimal values to fully exploit the data in the training dataset and to achieve a high training accuracy $\sim 95\%$, while avoiding overfitting.

VI. CNN CLASSIFIER PERFORMANCE:

In the plots in Figure 5 it is possible to see how the model performs in terms of the two metrics defined above (accuracy and true negatives) for both the training and testing dataset. The training dataset, which represents 80% of the original data, includes 22264 pairs of images, while the testing data includes the remaining 5566 pairs of images.

As it can be seen, the accuracy and the number of true negatives follow very different trends when plotted against the epoch. While the accuracy of the training dataset is increasing, as expected, as the training epochs increase, the validation accuracy has a flat trend. At a first analysis this might point out to overfitting since the validation accuracy does not improve with more training. However looking at the true negatives trend lines, it can be seen that the model progressively improves at identify true negatives. Even though the validation accuracy does not rise as the training goes on, the overall predicting ability of the network improves. As a matter of fact the model which initially predicts 0 true negatives, progressively gets better at classifying events belonging to the minority class.

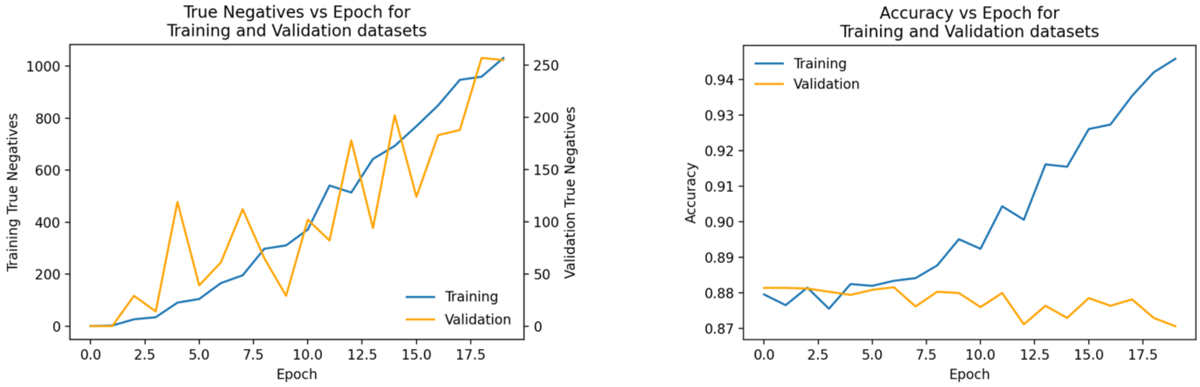

Figure 5. True negatives and accuracy of the model plotted against training epoch. *The plot on the left is showing the number of correctly identified true negatives against the training epoch. The plot on the right is showing the accuracy of the model against the training epoch. The blue line corresponds to the training dataset and the orange line to the validation dataset.*

Note the true negatives metric is using a threshold probability value of 0.5, which leads to an accuracy of 39% in the identification of true negatives (non ν_μ CC events) and an accuracy of 94% in identification of true positives (ν_μ CC events), summing up to an overall accuracy of 87%.

Since this classification task is a simulated version of the categorization happening in the NOvA experiment, an important consideration was made in order to adjust the threshold probability value. In the NOvA experiment scientists aim at identifying whether muon neutrinos change their flavour while travelling from the Fermilab detector to the Minnesota detector, and if so, what percentage of them have undergone this transformation. For this reason it is equally important to correctly identify muon neutrinos from ν_μ CC events and neutrinos of different flavours from other CC events. Therefore it is believed that the optimal classification model for this scenario should achieve an equal accuracy in the identification of both the considered classes.

In order to do so the probability threshold value was adjusted, a posteriori, to 0.93, meaning that events with a predicted score > 0.93 are classified as 1, otherwise as 0. This threshold value yields an accuracy of 80% in the identification of true negatives (non ν_μ CC events) and an accuracy of 81% in identification of true positives (ν_μ CC events), summing up to an overall accuracy of 81%.

VII. CNN Classifier Dependence on Metavariables:

Neutrino energy:

The plot of the observed neutrino energy distribution, for the testing dataset, is shown in figure in Figure 6. It is possible to see that the histogram distribution follows an exponential probability distribution. In order to test the model accuracy dependence on the energy of the neutrinos, the images in the testing dataset were divided into 50 different batches depending on the energy of the neutrino in the simulated interaction. Each batch contains ~ 100 interactions, the first batch includes interactions with neutrino energies in the range $[0,1.10]\ GeV$, and the last batch includes interactions with neutrino energies in the range $[38.40,77.60]\ GeV$. The last batches cover a wider energy range since fewer interactions are detected at high energies, as it can be seen in the histogram distribution. The accuracy of the model in determining the class of the interactions in a specific batch is represented as a datapoint in the orange trendline. The x-coordinate of each datapoint corresponds to the average of the neutrino energies of the interactions in the batch.

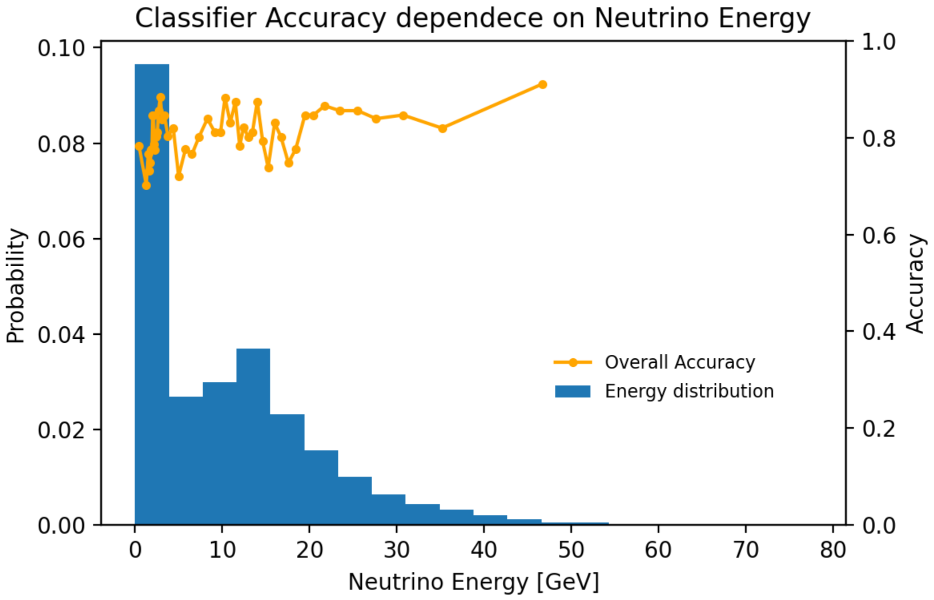

Figure 6. ***Accuracy of the model plotted against the neutrino energy for the classified interaction.*** *The energy distribution is also plotted as a histogram distribution. The accuracy does not follow a specific trend, while the histogram is following an exponential probability distribution.*

The accuracy of the model does not follow a particular trend, and therefore is not strongly dependent on neutrino energy, however the model was found to be very accurate at classifying high energy interactions $[38.40,77.60]\ GeV$. Overall it can be said that the model is consistent over the full range of neutrino energies $[0,77.60]\ GeV$, as the accuracy slightly oscillates around 0.8 as the considered neutrino energy increases.

Lepton energy:

The model dependence on the energy of the lepton involved in the interaction was also tested. Similarly to neutrino energy, also the lepton energy distribution follows an exponential distribution shown in

Figure 7. As before the images in the testing dataset were divided into 50 different batches depending on the energy of the lepton in the simulated interaction.

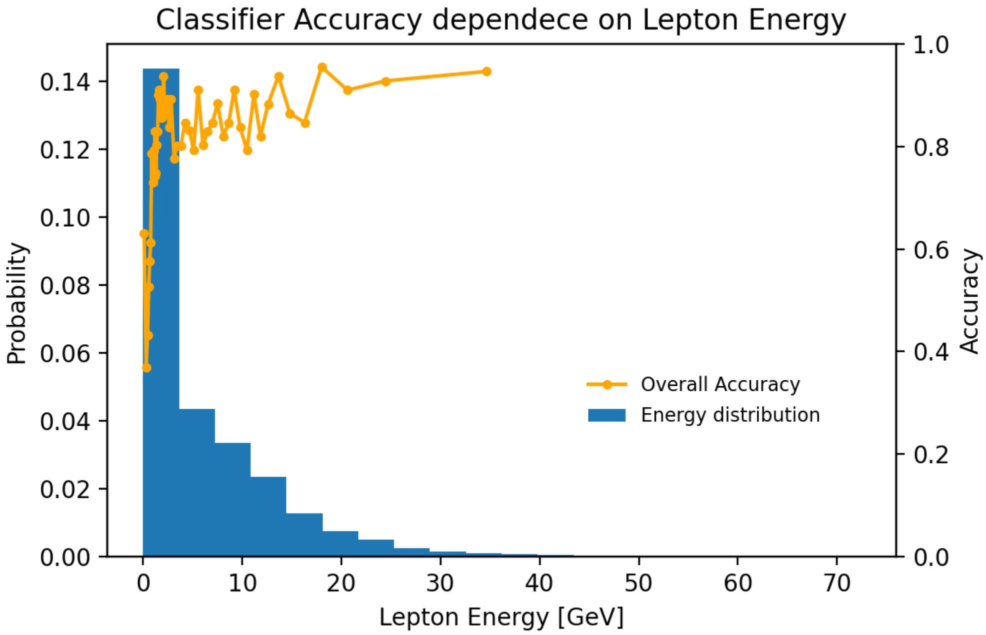

Figure 7. Accuracy of the model plotted against the lepton energy for the classified interaction. *The energy distribution is also plotted as a histogram distribution. The histogram is clearly following an exponential probability distribution.*

The accuracy of the model is low (< 0.6) for lepton energies up to $0.71\ GeV$, and it progressively increases as the lepton energy grows. The model poorly performs at classifying interactions with low lepton energy, however it is able to achieve high accuracies (> 0.8) in the identification of interactions with lepton energies, above $0.8\ GeV$. Overall it can be said that the model is consistent for lepton energies above $0.8\ GeV$, however the model predictions are not reliable below this threshold value. This difference in the model performance can be explained by the physics underlying the detection of neutrinos in the NOvA scintillators. When a neutrino undergoes a charged current interaction (which represents the majority of the simulated interactions) it is turned into the corresponding charged lepton, which deposits its energy and leaves a detectable trace in the scintillator. A higher energy lepton would generate a longer and better defined track when travelling through the scintillator, which in turn would lead to an easier detection of the neutrino interaction. For this reason high energy lepton interactions are better identified by the model. In Figure 8 a simulated low lepton energy CC QE interaction is compared to high lepton energy CC QE interaction.

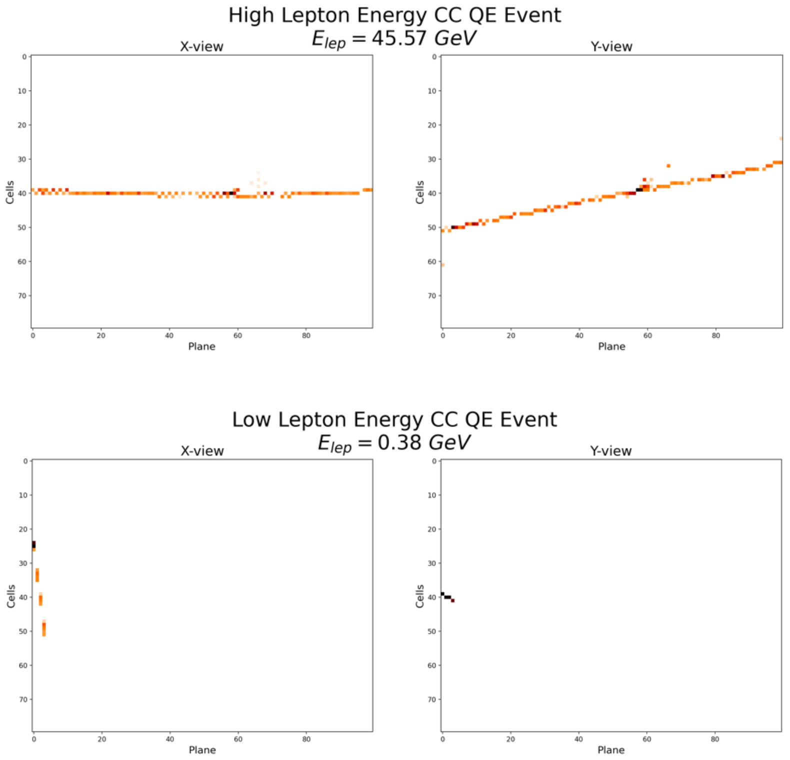

*Figure 8. **Comparison between high and low lepton energy CC QE interactions.** Higher lepton energy interactions produce longer and better defined particle tracks in the scintillator detector, which are more accurately classified by the model.*

Interaction Type:

The accuracy of the model was also analysed for each possible neutrino interaction. Results are shown in Table 1. The model accuracy varied between the three possible CC interactions. In particular the highest accuracy was found for QE interactions, and the lowest accuracy for DIS interactions. This effect could be explained by considering the complexity of each interaction, which is described at page 6-7. As a matter of fact QE events, which yield the highest accuracy, represent the most simple and clear neutrino interactions, while DIS evets, which are more messy, as the nucleon is broken up in the interaction, yield the lowest accuracy. As expected the remaining RES event, whose complexity is higher than QE but lower than DIS events, have an associated accuracy which sits somewhere in between. The model is therefore better performing at classifying simple interactions, whose corresponding images include fewer particle tracks.

	CC QE interaction	CC RES interaction	CC DIS interaction	Other CC interaction	NC interaction	Background events
Accuracy	0.91	0.83	0.78	0.90	0.80	0.92

Table 1. Accuracy of the model for the different types of neutrino interactions.

VIII. FINAL CONSIDERATIONS AND CONCLUSION:

The application of Deep Learning to classify simulated neutrino interactions similar to those observed in the NOvA experiment, proved to be successful. The CNN binary classifier developed in this project was able to identify ν_μ CC events with an overall accuracy of 81%. The major challenge encountered in this classification problem was the imbalance between the two classes in the dataset, and the risk of creating a model identifying every possible input as a member of the majority class. This issue was solved by introducing the number of true negatives as a metric of the neural network, and by adjusting the probability threshold value used to determine output class.

The model was found to be dependent on both the energy of the lepton generated in a neutrino interaction, and the type of neutrino interaction. The underlying reason of the model dependence on these metavariables is directly related to the complexity and clarity of the classified images. The CNN classifier achieved maximum performances in the identification of high lepton energy interaction, and of CC QE interactions.

Further improvements to the developed algorithm could be achieved by training and testing the model on a wider range of images, with special regard to the less represented interactions, such as those involving high energy neutrinos and leptons.

In conclusion, Deep Learning and neural networks proved to be very powerful tools for classifying neutrino interactions, and their high versatility and scalability could provide valuable insights in many other HEP experiments involving image categorization and signal-background classification.

IX. References:

[1] Close, Frank (2010). Neutrinos (softcover ed.). Oxford University Press. ISBN 978-0-199-69599-7.

[2] J. Bian, «The NOvA Experiment: Overview and Status», arXiv:1309.7898 [hep-ex, physics:physics], set. 2013. Available at: http://arxiv.org/abs/1309.7898

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553. Springer Science and Business Media LLC, pp. 436–444, May 27, 2015. doi: 10.1038/nature14539.

[4] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, "An Introductory Review of Deep Learning for Prediction Models With Big Data," Frontiers in Artificial Intelligence, vol. 3. Frontiers Media SA, Feb. 28, 2020. doi: 10.3389/frai.2020.00004.

[5] A. F. Agarap, «Deep Learning using Rectified Linear Units (ReLU)», mar. 2018. Available at: https://ui.adsabs.harvard.edu/abs/2018arXiv180308375A

[6] A. K. Singh e B. U. Shankar, «Multi-Label Classification on Remote-Sensing Images», arXiv:2201.01971 . Available at: http://arxiv.org/abs/2201.01971

[7] NN-SVG, Publication-ready NN-architecture schematics. https://alexlenail.me/NN-SVG/index.html

[8] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "Deep Learning". MIT Press. pp. "5.1 Learning Algortihms" pp. 105, "6.3 Hidden Units" pp. 178–179, "5.9 Stochastic Gradient Descent" pp. 147, "9.3 Pooling" pp. 330-335 . ISBN 9780262035613.

[9] K. P. Murphy, Machine learning: a probabilistic perspective. Cambridge, MA: MIT Press, 2012.

[10] Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. Nature 323, 533–536 (1986). https://doi.org/10.1038/323533a0

[11] A. Aurisano et al., «A Convolutional Neural Network Neutrino Event Classifier», J. Inst., vol. 11, n. 09, pagg. P09001–P09001, set. 2016, doi: 10.1088/1748-0221/11/09/P09001.

[12] K. Saraswat, P. Shukla, V. Kumar, and V. Singh, "Charged current quasi elastic scattering of muon neutrino with nuclei," Indian Journal of Physics, vol. 92, no. 2. Springer Science and Business Media LLC, pp. 249–257, Aug. 29, 2017. doi: 10.1007/s12648-017-1093-0.

[13] D. Grover, K. Saraswat, P. Shukla, and V. Singh, "Charged-current deep-inelastic scattering of muon neutrinos (νμ) off Fe56," Physical Review C, vol. 98, no. 6. American Physical Society (APS), Dec. 13, 2018. doi: 10.1103/physrevc.98.065503.

[14] H. Lee, R. Grosse, R. Ranganath, e A. Y. Ng, «Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations», in Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09, Montreal, Quebec, Canada, 2009, pagg. 1–8. doi: 10.1145/1553374.1553453.

[15] Towards data science https://towardsdatascience.com/deciding-optimal-filter-size-for-cnns-d6f7b56f9363

[16] C. Peng, X. Zhang, G. Yu, G. Luo, e J. Sun, “Large Kernel Matters -- Improve Semantic Segmentation by Global Convolutional Network”, arXiv:1703.02719. Available at: http://arxiv.org/abs/1703.02719

[17] J. Cui, S. Qiu, M. Jiang, Z. Pei, and Y. Lu, “Text Classification Based on ReLU Activation Function of SAE Algorithm,” Advances in Neural Networks - ISNN 2017. Springer International Publishing, pp. 44–50, 2017. doi: 10.1007/978-3-319-59072-1_6.

[18] D. P. Kingma e J. Ba, «Adam: A Method for Stochastic Optimization», arXiv:1412.6980. Available at: http://arxiv.org/abs/1412.6980