Deep Learning techniques for signal processing and event reconstruction in DUNE

Supervisor: Manuel Rodriguez
E-mail: manuel.jesus.rodriguez.alonso@cern.ch
Supervisor: Lorenzo Uboldi
E-mail: lorenzo.uboldi@cern.ch
Supervisor: Paola Sala
E-mail: paola.sala@cern.ch

Openlab Project by:
Laura Accorto
E-mail: laura.accorto@gmail.com

Summer 2021

Abstract

DUNE - *Deep Underground Neutrino Experiment* - will be an experiment based in the USA whose main goal will be to study long-baseline neutrino oscillations from an accelerator beam. At CERN, a smaller prototype of the DUNE far detectors has been created, ProtoDUNE, with the objective to test and validate the technology required for DUNE far detectors. The signal captured by ProtoDUNE needs to be processed to clean it from noise. To do so, a Deep Learning tool has been developed at CERN. However, the costs of this tool imply the necessity to evaluate its performance comparing to more classical techniques, which involve lower costs. For this reason, a statistical algorithm has been considered, which is the one currently used: the results of the two solutions are compared in this project.

Introduction

DUNE - *Deep Underground Neutrino Experiment* - will be a large experiment that will be based in the USA and whose main objective is to detect neutrinos and analyze their behaviour. At CERN, a prototype of the DUNE far detector has been built: ProtoDUNE is a 20 times smaller version of the DUNE far detector, whose components are scaled 1:1 and have the same design; the role of this prototype is to test and validate the technology required for DUNE far detectors.

This experiment gathers a large amount of data, which needs to be lightened in order to save memory space. For this reason, a technique to flexibly distinguish particle signal from everything else is necessary. Two possibilities are analysed in this work: the first, a neural network for image segmentation used to classify the signal; the second, a statistical approach - the *Hit finding algorithm* - which is the currently used one. Each of the two techniques considered has its strengths and its weaknesses, and for this reason, an analysis that compares the results obtained has been conducted in this work.

First of all, we're going to introduce the ProtoDUNE data and its structure. Then we will describe the two techniques proposed to process the data and classify the signal in a deeper way. A comparison between the results obtained is then made to show each technique's advantages and disadvantages.

The ProtoDUNE Data

ProtoDUNE is made up of two drift volumes, which are separated by a vertical cathode plane. Two anode planes are positioned at the opposite sides of the drift volumes with respect to the cathode. The detector is located inside a cryostat which insulates the detector volume from the outside. ProtoDUNE detector is then filled with Liquid Argon (LAr). The cathode plane is designed to be held at -180 kV, providing a 500 V/cm drift electric field in each of the two

opposite horizontal directions: this field allows an electron to travel the entire drift length in slightly less than 2.5 μs. Ionization electrons, produced by traversing charged particles, are drifted to the anode plane by the electric field. A closed circle purification system continuously processes the LAr, eliminating contamination and refilling the active volume so that electrons are not captured during their way.

On each of the two anode planes of ProtoDUNE, we can find three APAs - *Anode Plane Assembly* - for a total of six APAs, performing the signal readout. Each APA is made by three planes of evenly spaced wires: the first two planes - *induction planes* - are transparent to the drifting electrons and, being traversed, read out the local modification of the electric field. The third one - *collection plane* - is at the higher potential and collects the electrons, detecting the total charge. The wires of the three planes are placed at different angles enabling a two-dimensional reconstruction of the event when the information of the three planes is combined.

The signal is collected over time for each channel. In particular, ProtoDUNE data are split into "events": the time scale has been discretized with a sampling rate of 2MHz, where each tick is a sample; the event is then a 3ms window length, translating to 6000 ticks. We actually look at 5888 time-ticks over all 2080 channels from the three planes. Channels from 1 to 800 belong to the first induction plane, channels from 801 to 1600 belong to the second induction plane, and channels from 1601 to 2080 belong to the collection plane.

Data is loaded using PyTorch's *DataLoader*: each event is divided into 23 subsequent batches, containing 256 time-ticks each. Each batch is then a 2080x256 array: by concatenating all 23 batches in the correct order; then one can obtain the complete 2080x5888 array, which spaces along all time-ticks of the event.

Once we have concatenated all batches for each channel, we can then visualize a time series data representing the signal recorded at time t, $t = 1, \ldots, 5888$. Data regarding the single event from a specific APA is then a collection of 2080 time series. For each APA, we have around 3580 events, a total of 3580 x 2080 (7446400) time series. Figure 1 shows the signal from channels coming from induction and collection planes: it can be seen that data recorded by the collection plane behaves very differently compared to data recorded by the induction planes. Over the induction plane, the signal has a bipolar shape, while the collection plane signal has a positive unipolar shape. This difference is due to the fact that the electrons pass through the induction wires while the collection ones completely absorb them. In the collection plane, the signal to noise ratio is much higher than in the induction planes, and, for this reason, it is much easier to detect anomalies and distinguish the signal from noise when working over the collection plane.

It is possible to classify the signal through some different techniques. We will call a *hit* everything which has to be classified as signal. In the next section, the two techniques we consider to find the hits are presented and explained in more detail. In the end, for each event, we will obtain a

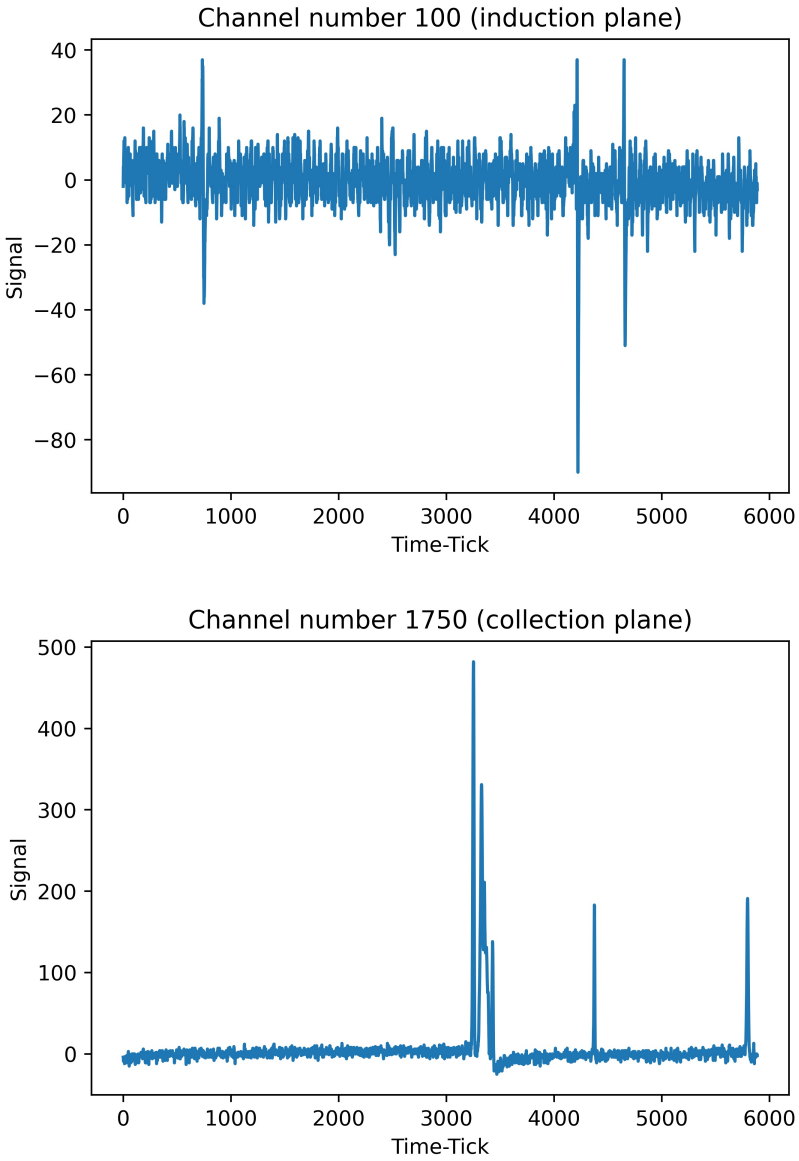

Figure 1: A signal from a chosen event of APA 5 over induction plane and collection plane

2080x5888 binary array, in which 1 denotes a hit found in the selected channel and in that time tick, and 0 denotes noise.

RoI finding in ProtoDUNE

Our objective is to find the hits: a hit means everything which has to be denoted as signal. Once we find the hits, we can detect the *regions of interest* (RoI), which are made up by the adjacent hits in the array space.

To find hits in ProtoDUNE, we considered two algorithms:

- the *Deep Learning algorithm* that exploits neural network for semantic segmentation, to classify the hits;

- the *Hit finding algorithm* that makes use of statistical techniques to classify the hits; this is the currently used algorithm in the ProtoDUNE experiment.

Deep Learning algorithm

The *Deep Learning algorithm* exploits semantic segmentation theory, considering, for each event, the 2080x5888 array as an image and classifying each pixel of it based on what object should be the pixel part of, that is, either signal or noise. In this case, the output has the same dimension as the input: a possible approach is to use the encoder-decoder architectures for the neural network. The dimensionality of the data is firstly compressed and then restored to the original one to make the pixel-wise classification.

We define a two-class segmentation task in our setting, where one class represents signal and the other electronic noise. We can train the network to classify, pixel by pixel, a raw event of ProtoDUNE identifying with precise localization of both signal and noise.

LinkNet (Chaurasia & Culurciello (2017)) is a neural network structured with encoder and decoder blocks, and it is a lightweight and fast solution that solves the usual problem related to semantic segmentation. In this approach, networks are generally deep and slow, and the encoder-decoder structure tends to be difficult to train. Authors of *LinkNet* showed that it outperforms many other semantic segmentation networks both in terms of speed and in terms of accuracy. For this reason, this architecture has been chosen as a starting point: the *Deep Learning algorithm* used for ProtoDUNE is a tiny and simplified version of *LinkNet*. Having only two classes, a sigmoid has been used instead of a softmax as an activation function for the last layer. For every pixel, the output is $p \in [0, 1]$ representing the signal probability. The output is converted to binary at the end, putting each pixel to 1 if the probability of being signal is more than 0.5, to 0 otherwise.

Hit finding algorithm

The currently used solution, *Hit finding algorithm*, exploits the *Frugal algorithm* (Ma et al. (2014)) to create a threshold based on which the classification is conducted: data is classified as signal wherever it lays above this threshold; otherwise, it is considered as noise.

In this case, data is considered in its time-series shape: for each event, for each channel, the time-series containing the intensity of the signal at time t is analyzed.

In order to "smooth" the time series, a modification of the *Frugal algorithm* is considered.

Frugal algorithm can use only one unit of memory per group to compute a generic quantile for each group in a stream of data. For stochastic streams where data items are drawn from a distribution independently, it is proved that the algorithm finds an approximation to the quantile rapidly and remains stably close to it. Through this algorithm, the generic quantile is

estimated by following the direction suggested by the data stream: if it increases, the quantile does it too. At the very beginning, the quantile estimate is set to 0; then, for each step of the data stream, if data is higher than the quantile estimation, the quantile estimation is increased by 1; otherwise, it is decreased by 1. The detailed algorithm for the generic quantile is shown in Ma et al. (2014).

In *Hit finding algorithm* the *Frugal algorithm* is used to compute 25%, 75% quantiles and the median (50% quantile), but with a modification of the algorithm. The modification proposed and used throughout the *Hit finding algorithm* uses accumulators to evaluate the median and the quantiles. Instead of updating the quantile estimation, $\tilde{m}$, time-by-time, it is made only when an accumulator is over its limit. For example, if the limit is 10, then it has to happen that $s_i > \tilde{m}$ 10 times subsequently to increment $\tilde{m}$ (similarly for the decrementation). This variation produces smoother estimations of the quantiles (the higher the accumulator limit, the smoother the estimation will be).

The median, the 25% and the 75% quantiles are calculated for each time-series data (that is, for each channel and each event). The median is considered as a pedestal, and the 25% and 75% quantiles allow to estimate the pedestal's variance at time t, σ_t; then, the threshold at time t is calculated as $K \cdot \sigma_t$, where $K = 5$ for all channels in the collection plane, and $K = 3$ for all channels in the two induction planes.

The threshold is calculated over the raw data but applied over data in which a *Firwin filter* [1] with 7 taps and a 0.1 cutoff has been applied in order to remove some noise. A hit is identified every time the filtered data lays over the threshold.

An example of all quantities calculated by the *Hit finding algorithm* for channel 1750 over a single event of APA 5 is presented in figure 2.

This procedure is repeated for each channel and each event, obtaining an output which is analogue to the *Deep Learning algorithm* one, that is, a binary 2080x5888 array for each event and each APA.

Comparison and analysis of the two RoI finding techniques

Both algorithms shown in the previous section allow us to find hits and, consequently, the regions of interest. Still, one has to analyse the results in order to see the main differences, highlighting the advantages and disadvantages of each technique.

An advantage of the *Hit finding algorithm* is that being a statistical algorithm, everything it does is easy to interpret and to understand. In contrast, the *Deep Learning algorithm* is actually a black box. On the other hand, the *Deep Learning algorithm* could allow higher performance in terms of accuracy. Therefore one should evaluate if it is worth it to apply this technique to data.

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.firwin.html

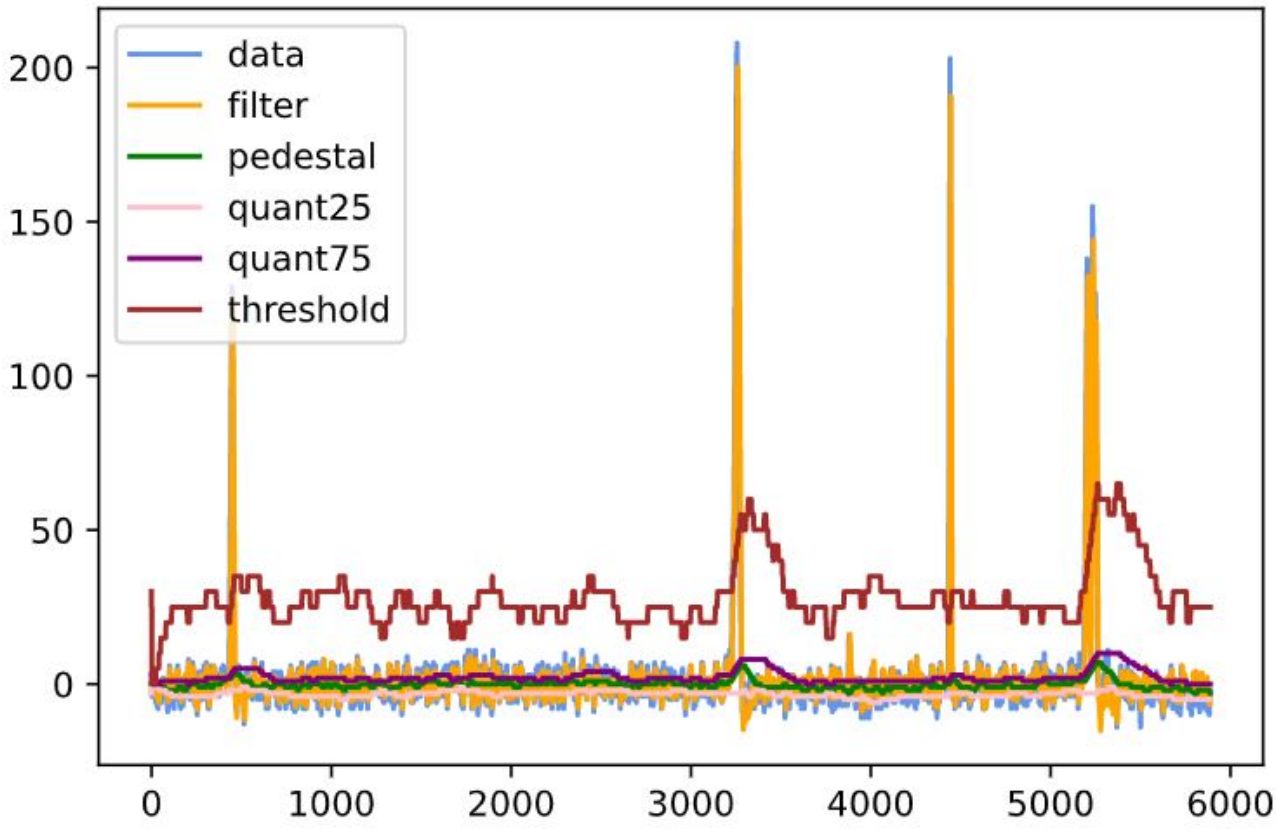

Figure 2: *Hit finding algorithm* applied over a single event of APA 5, channel 1750.

To analyse the main differences between the results proposed by the two algorithms, we start counting the average *percentage of covered image* (PCI) for results given by each algorithm. PCI is calculated by counting the percentage of pixels classified as a hit on each event and then averages over all the events. We obtain $PCI = 2.92\%$ for the *Deep Learning algorithm* and $PCI = 0.74\%$ for the *Hit finding algorithm*, that is, *Deep Learning algorithm* predicts almost 4 times more hits than the *Hit finding algorithm*.

To see the spatial distribution of the signal for each algorithm, we can consider *2D-Histograms* over a sample of randomly chosen events. These histograms are made by overlapping events so that for each pixel, we have the frequency of predicted hit on that pixel, that is on that channel and on that time-tick. In figure 3 histograms for both *Deep Learning algorithm* and Hit finding algorithm are presented.

From the histogram, we can see that, while the *Deep Learning algorithm* tends to predict hits uniformly all over the channels without any significant difference, the *Hit finding algorithm* is more likely to predict hits over channels belonging to the collection plane, that is, from channel 1600 to 2080. This fact could be due to the statistical algorithm being not flexible enough for the induction planes classification task.

We confirm this feature if we also calculate the average PCIs over the different planes, which are collected in the table 1. It is clear that the *Deep Learning algorithm* performance is stable over

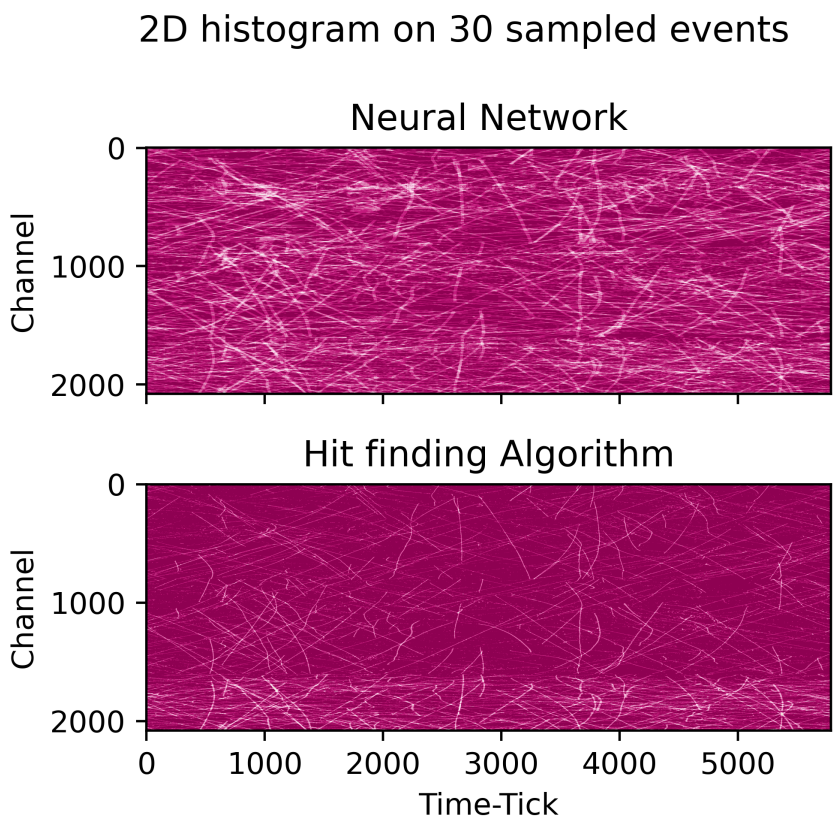

Figure 3: 2D histograms over a sample of randomly chosen events, both for Neural Network (DL algorithm) and Hit finding algorithm.

all the planes, while it changes a lot for the *Hit finding algorithm* when passing from induction to collection.

	1st Induction	2nd Induction	Collection
Neural Network	2.87%	2.79%	3.25%
Hit finding algorithm	0.54%	0.62%	1.25%

Table 1: PCIs over different planes for both algorithms.

To see how the prediction of the two algorithms is matching pixel-by-pixel, we can consider, over a single event, typical Machine Learning metrics, that is, true positives, true negatives, false positives and false negatives. In particular, we consider the *Deep Learning algorithm* prediction as target and check how much the *Hit finding algorithm* prediction is matching. In this way, we can compute all the metrics for each event and average them over all the events. More precisely:

- False Negatives (FN) are pixels classified as hits by the *Deep Learning algorithm* but not by the *Hit finding algorithm*;
- False Positives (FP) are pixels classified as hits by the *Hit finding algorithm* but not by the *Deep Learning algorithm*;
- True Negatives(TN) are pixels classified as noise by both algorithms;

- True Positives (TP) are pixels classified as hits by both algorithms.

All indexes are normalized to the target, which in our case is the neural network prediction.

We obtain $FN = 81.8\%$ and $FP = 0.16\%$, so it is clear that *Hit finding algorithm* predicts much less compared to the *Deep Learning algorithm*. Still, when *Hit finding algorithm* predicts a hit, at least it is coherent with the *Deep Learning algorithm* prediction since false positives are very low.

In order to have a better idea of the matches and their location, we plot these metrics over a single event image: an example is presented in figure 4.

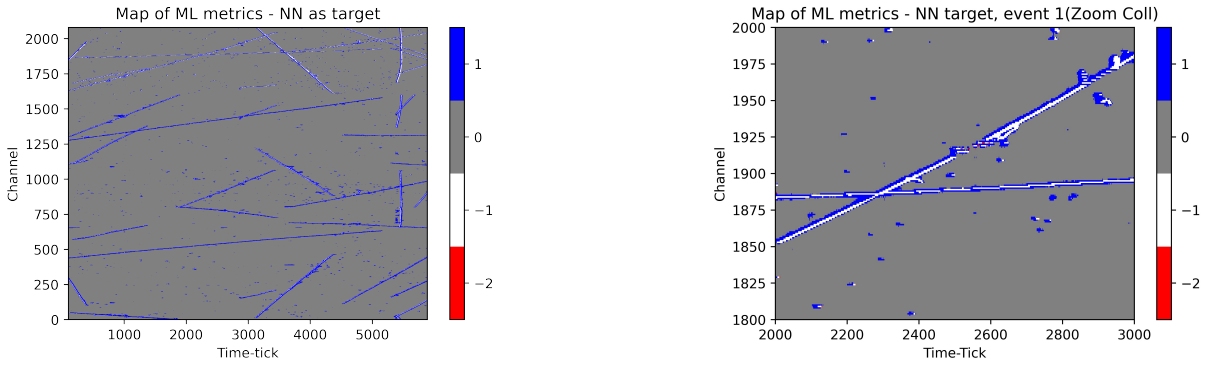

Figure 4: ML metrics mapped over a single event considering Neural Network as a target; on the right, a zoom over a single track on the collection plane.

True negative pixels are represented in grey, true positive pixels in white, false negative pixels in blue and false positive pixels in red, always considering the neural network as a target and the *Hit finding algorithm* as the prediction.

From the images and the zooms over single tracks, we can see many false negatives, especially over the induction plane. However, the false negatives are usually next to some true positives. In many cases, the *Hit finding algorithm* doesn't miss the region of interest but predicts a smaller version of it. Isolated false negatives, which are the problematic ones, are more frequent over the induction planes. At the same time, from the zoom, we can see that, while the *Deep Learning algorithm* tracks are continuous lines, on the other hand, the *Hit finding algorithm* tends to predict interrupted lines.

The comparisons made up to now are simply showing the main differences between the two methods. Still, there is no possibility to evaluate the predictive performance of each algorithm through these metrics. In order to do so, a common ground truth is necessary, but it is not available. For this reason, we perform a Monte Carlo simulation. For computing the metrics, we use the *Monte Carlo reconstruction*. This reconstruction is obtained by using the official reconstruction chain of ProtoDUNE, that is, *Pandora* (Marshall & Thomson (2015)). Future work could be done by producing the metrics with the ground truth obtained by the Monte Carlo simulation.

We simulate two datasets:

- the *cosmic dataset*, in which both cosmic rays and radiological events are simulated;
- the *radiological dataset*, in which only radiological events are simulated.

By considering the radiological dataset, we can see also the performances of the two algorithms when considering very low energy events.

The Monte Carlo reconstructions are compared firstly to the predictions made by the *Deep Learning algorithm* and secondly to the predictions made by the *Hit finding algorithm*, both for the cosmic and the radiological dataset.

Over the cosmic dataset, we can see the main behaviours of both algorithms. An example of their predictions compared to the Monte Carlo reconstruction over the cosmic dataset is shown in figure 5. In table 2 false negatives and false positives for each algorithm over the Monte Carlo cosmic dataset are shown.

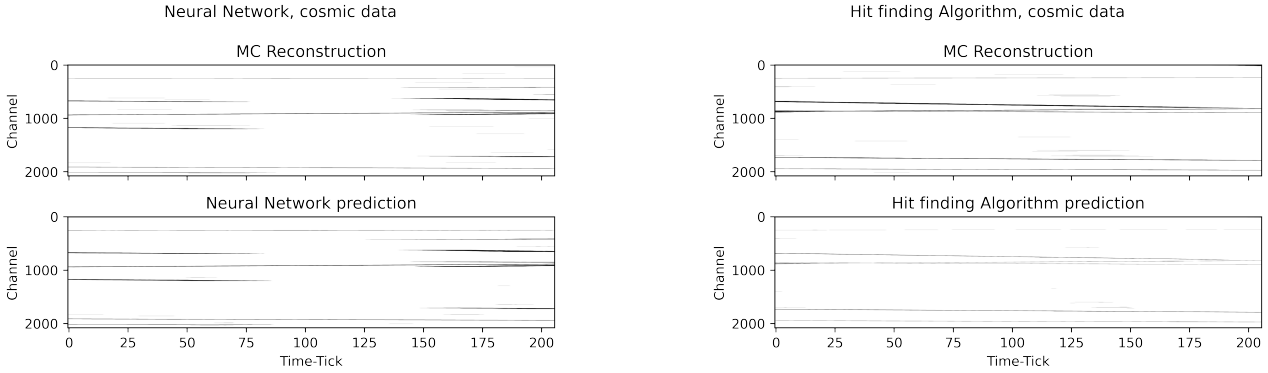

Figure 5: Predictions by both algorithms over the Monte Carlo cosmic dataset.

	Neural Network	Hit finding algorithm
False negatives	15.84%	69.50%
False positives	44.55%	0.70%

Table 2: False negatives and false positives by both algorithms predictions over the cosmic dataset.

From these results, it seems that the *Deep Learning algorithm* tends to overpredict, while the *Hit finding algorithm* tends to underpredict. Anyway, this consideration holds only if one thinks that the Monte Carlo reconstruction is a reliable reconstruction of the ground truth: it could be that the reconstruction itself classifies low energy signal as noise. Therefore some of the false positives given by the neural network may actually be true positives.

Similarly to what we have already done before, we can plot the location of all metrics over a single event, but this time considering the Monte Carlo reconstruction as a target. Figure 6 shows the metrics for both algorithms over a single event from the cosmic dataset. The majority of the false positives produced by the *Deep Learning algorithm* is adjacent to true positives,

meaning that the neural network manages to find the tracks. Still, they are thicker than the reconstructed Monte Carlo ones. On the other hand, the *Hit finding algorithm* produces many isolated false negatives: it often happens that *Hit finding algorithm* completely misses tracks along with the event.

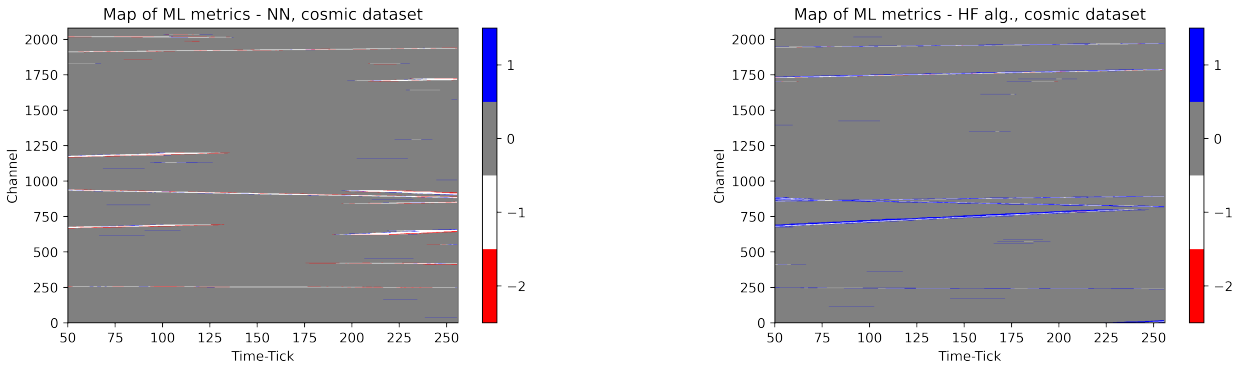

Figure 6: Metrics' location by both algorithms over the Monte Carlo cosmic dataset.

In the same way, we can consider predictions of both algorithms over the radiological dataset. In this case, the results are more difficult to evaluate since the signal is very mild and problematic to catch, as one can see from the predictions shown in figure 7.

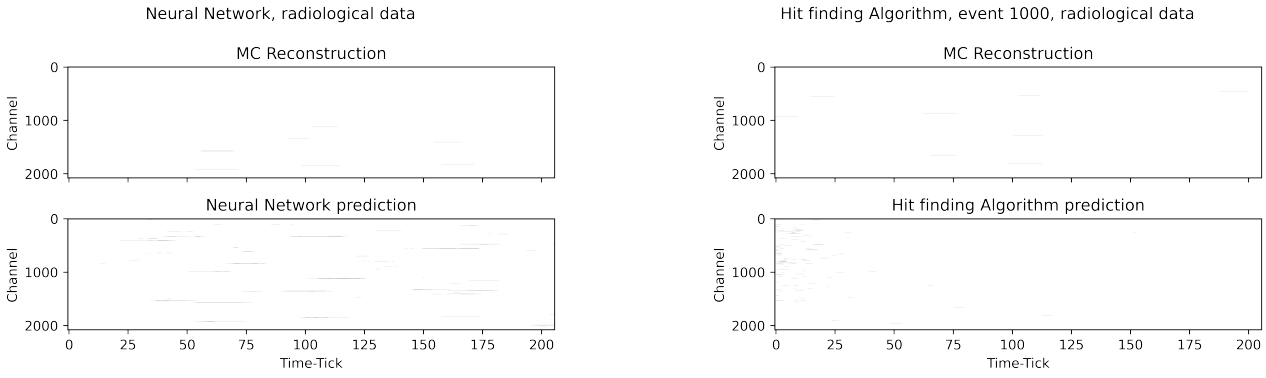

Figure 7: Predictions by both algorithms over the Monte Carlo radiological dataset.

For this reason, we check the true positives and false positives differently: for each track which is in the Monte Carlo reconstruction, we check if it is present also in the prediction given by each algorithm, even if they only partially overlap, we classify it as true positives; at the same way, we check if some tracks which are present in the prediction of the algorithms are completely missing in the Monte Carlo reconstruction, and in this case, we identify the false positives. We obtain that the true positive rate is 95% for the Neural Network, even with a higher false positive rate, but the true positives are only 49% for the Hit finding algorithm. We can conclude that the accuracy of the neural network is higher, but also its purity is much lower than the *Hit finding algorithm*.

Conclusions and Future work

One should choose the algorithm based on which one best fits his necessities and costs in terms of both: computational time and money.

If we consider the results just obtained, we can see that generally, the *Deep Learning algorithm* tends to overpredict, and the *Hit finding* tends to underpredict; in this sense, the choice of the algorithm is guided by the costs and benefits that come with overpredictions and underpredictions.

At the same time, another important aspect that would drive the choice is surely the final purpose that one has: if the objective is to find the single hits at a pixel-wise level, then the *Hit finding algorithm* performance is worse compared to the *Deep Learning algorithm*; on the other hand, if we consider the classification task at a region-level, then the *Hit finding algorithm* performance is not as bad as at the pixel level, but still there are tiny tracks which aren't identified at all and many long tracks are split.

Moreover, it is crucial to understand what is the kind of signal that we want to detect. If we have to handle very low energy events, we have seen that the *Deep Learning algorithm* performs way better than the *Hit finding algorithm.*

A lot of work still needs to be done, particularly regarding the ground truth to compare with; a Monte Carlo truth can be extracted as ground truth to see if the predictive performance changes compared to the one already seen over the Monte Carlo reconstruction. Moreover, some post-processing over the neural network results can be done by removing all classified tracks shorter than a threshold in the time-tick direction. These tracks can be considered false positives since we know that the signal has a minimum length in time.

Bibliography

Chaurasia, A. & Culurciello, E. (2017). LinkNet: Exploiting encoder representations for efficient semantic segmentation. In *2017 IEEE Visual Communications and Image Processing (VCIP).*

Ma, Q., Muthukrishnan, S. & Sandler, M. (2014). Frugal Streaming for Estimating Quantiles:One (or two) memory suffices. *arXiv:1407.1121 [cs]* ArXiv: 1407.1121.

Marshall, J. & Thomson, M. (2015). The Pandora Software Development Kit for Pattern Recognition. *Eur. Phys. J. C* **75**, 439.