SNN Project Report

Multi-Track Events Analysis with Unsupervised Delay Learning

Emanuele Coradin

07/06/2024

1 Objectives

The purpose of this report is to summarize the recent development in the project branch that investigated the possibility of using a Spike Time Dependent Plasticity (STDP) learning rule for synaptic delays, testing the model also with 2-track events.

2 Setup description

2.1 Learning Rule for Delays

In some previous experiments, we tried to introduce the synaptic delays and set them to some fixed values derived from our prior knowledge about the temporal hit representation of the different types of tracks, obtaining interesting performances in terms of signal detection efficiency and neuron specialization.

These results let us think about the possibility of looking for an unsupervised learning method also for the delays. After a brief look at the literature, we came up with the simple idea of applying the Spike Time Dependent Plasticity (STDP) rule, which we had already used as a learning method for the weights. In the following section, we briefly review it.

2.1.1 STDP Principle

In hebbian STDP, the synaptic strength is adjusted on the basis of the relative timing of pre- and postsynaptic spikes. If a presynaptic spike precedes a postsynaptic spike within a certain time window, the synapse is potentiated (strengthened). Conversely, if the post-synaptic spike precedes the pre-synaptic spike, the synapse is depressed (weakened).

2.1.2 Adaptation to Delays

In this study, a similar principle is applied to the delays:

- **Potentiation (Delay):** If a pre-synaptic spike arrives shortly before a post-synaptic spike, the delay is increased (the pre-synaptic spike is delayed).
- **Depression (Advance):** If a post-synaptic spike occurs before the pre-synaptic spike, the delay is decreased (the pre-synaptic spike is advanced).

The delay update rules are given by:

$$\Delta d_j = \begin{cases} d_+ \cdot \exp\left(\frac{t_j - t_i}{\tau_{d_+}}\right) & \text{if } t_j \leq t_i \,(\text{LTP}) \\ -d_- \cdot \exp\left(\frac{t_i - t_j}{\tau_{d_-}}\right) & \text{if } t_j > t_i \,(\text{LTD}) \end{cases} \quad (1)$$

where t_j is the arrival time of the presynaptic spike, t_i is the arrival time of the postsynaptic spike, d_+ and d_- are the learning rates, and τ_{d_+} and τ_{d_-} are the time constants for potentiation and depression, respectively. For reference, see the figure 1, which represents the well-known case of learning weights.

As a regularization technique, the sum of the delays is forced to be constant throughout the process thanks to a renormalization phase applied after each update.

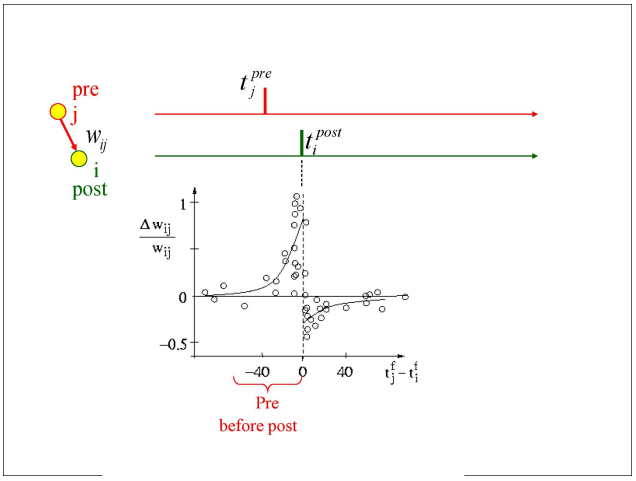

FIGURE 1: Illustration of hebbian STDP weights rule.

2.2 Neural Network Architecture

2.2.1 Quick recap

The neural network is organized into two layers: L0, L1. The neurons belonging to L1 receive incoming signals from L0, and both L0 and L1 are connected to the tracking layers (see Fig.2). All $N_{L0} = 10$ ($N_{L1} = 10$) neurons of L0 (L1) are characterized by the activation threshold T0 (T1). An incoming signal at time t_j from synapse j results in an increase in the neural membrane potential. When an L0 neuron exceeds its activation threshold, it sends an impulse and its potential decreases. It also exclusively inhibits other neurons in its layer according to an Inhibitory Post-Synaptic Potential (IPSP).

The same reasoning applies to L1.

2.2.2 Learnable paramaters and their initialization

Important note: potentially, this update could train the model by adjusting the values of synaptic weights and delays simultaneously, but to reduce complexity I decided to

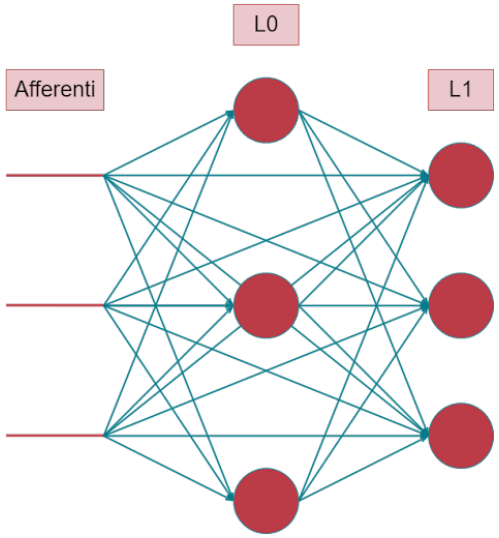

FIGURE 2: Architecture of the neural network. (Instead of 'afferenti' think of the CMS tracking layers)

let the network change only the latter, leaving the former at their initial values.

In particular, I initialized the weights sampling from a Gaussian distribution with mean 1 and standard deviation $\sigma = \frac{2}{\sqrt{N_{\text{Tracking Layers}}}} = \frac{2}{\sqrt{10}} = 0.6$, clipping to 0 the negative values, and then normalizing the sum of the weights to 1.

The synaptic delays related to the CMS tracker are initialized according to a uniform distribution in the interval [1.2 , 1.8]ns, while I forced the synaptic delays of the connections between L0 and L1 to be 0. Furthermore, throughout the training phase, I forced their sum to constantly be equal to the initial one.

2.3 Mini-Grid Search and Pretraining

2.3.1 Considered tracks and background noise modeling

For this research, I restricted the problem to a three-class classification, considering only muons with transverse momentum of 1 GeV, 3 GeV, and 10 GeV. The data sets I used to train and test the model contain 100,000 events, half of which contain only background noise, while the other half also contain signals.

The noise follows a Poisson distribution, an average number of 100 background hits per event, distributed according to a uniform distribution in the radial coordinate of the transverse plane of the CMS tracker.

2.3.2 Training Details

Firstly, the network is trained using just single-particle events. I split the data set into 80,000 events for training and 20,000 for testing. I manually run a quick grid search to find some working hyperparameters. I ended up with a configuration that looks promising for the job, and here I report the command I used to launch the software for reference:

```
./SNNT13.out --Train_fraction 0.8 --N_ev 100000
    --NL0 10 --NL1 10
--TH0 0.65 --TH1 0.35 --L1inhibitfactor 1 --
    IPSP_dt_dilation 0.4
--a_plus 0 --a_minus 0 --alpha 0.5 --split_layer0
     0 --taud_minus 2.e-09
--taud_plus 2.e-09 --d_minus 1.e-12 --d_plus 9.e
    -12
```

So, regarding the parameters that control the update of the delays, I found this:

$d_- = 1$ ps, $d_+ = 9$ ps, $\tau_{d_+} = \tau_{d_-} = 2$ ns.

I mainly experimented with d_+, d_-, so there may be other settings that work even better.

If you have specific questions about the other parameters, feel free to email me.

3 Results

3.1 Test with Single-Particle Events

In Fig.3, the x-axis represents the neuron ID, the y-axis denotes the particle class, and the color map indicates the efficiency. The efficiency is defined as the fraction of events containing a specific type of particle in which the neuron has fired at least once.

Indicating the ID of the neurons $\in [0, 19]$, we can see that:

- Neurons 8 and 19 are more active during events containing 1 GeV particles;
- Neurons 4 and 17 are more active during events containing 3 GeV particles (but also neuron 0, 13, 15, 18 with lower efficiencies);
- Neurons 5 and 10 are more active during events containing 10 GeV particles.

For the most interesting neurons, the numerical values of the efficiencies are displayed in the plot.

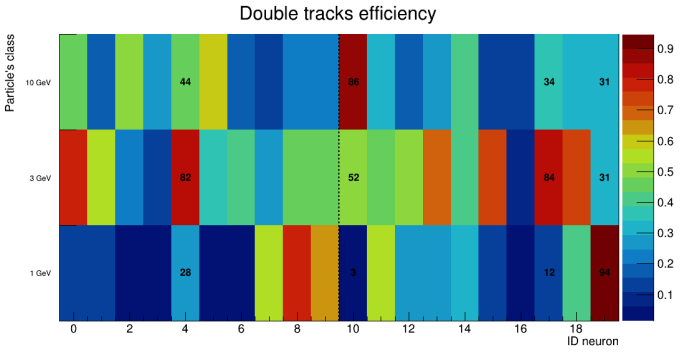

FIGURE 3: Single-Particle Events Efficiency Map. A vertical dashed line separates neuron belonging to layers L0 and L1

For the fake rate, defined as the fraction of background events in which a neuron fired at least once, see the plot in Fig.7 in the following section.

Furthermore, I provide in Figure 4 a plot showing the numerical values of the learned delays, and in Figure 5 the difference between these values at the end of the training and their initial values.

The index of a tracking layer is proportional to its radial distance from the beam. This means that layers farther from the beam are assigned higher indices. By analyzing the delays of the most active and specialized neurons, we can see how the system naturally adapts to the timing characteristics of the hits from the negative particle tracks. Specifically,

neurons adjust their delays to align the hits within a narrower time window, increasing the probability of being activated.

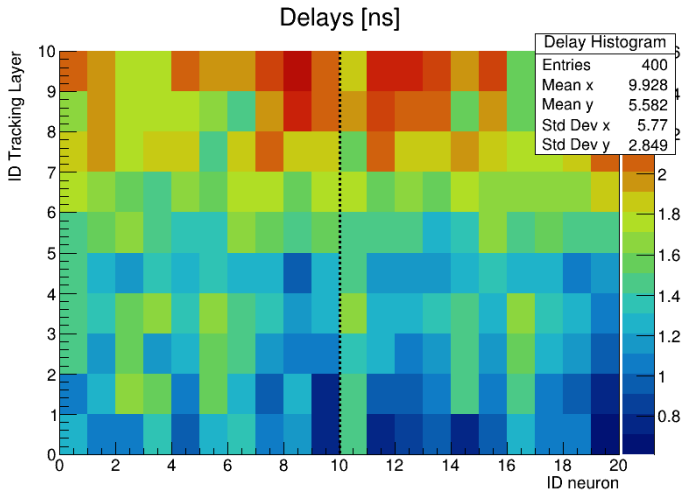

FIGURE 4: Learned synaptic delays for each neuron. A vertical dashed line separates neuron belonging to layers L0 and L1

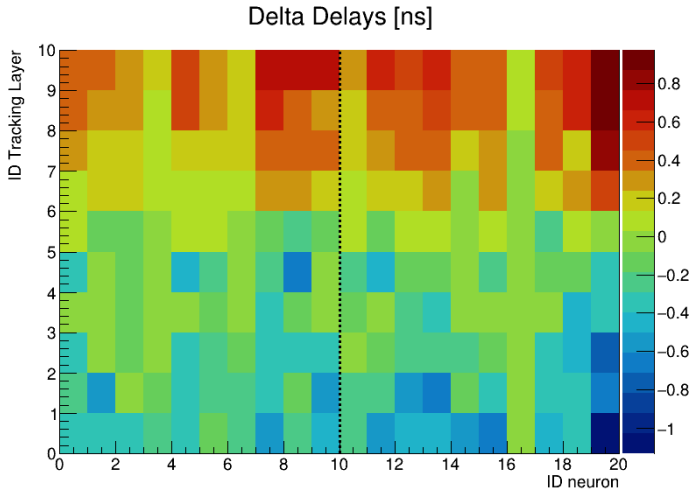

FIGURE 5: Difference between the learned synaptic delays for each neuron and their initial values. A vertical dashed line separates neuron belonging to layers L0 and L1

3.2 Test with Two-Particle Events

I also proceed to test the network with 50,000 events, half of them containing just noise, while the other half contains two particles.

All the combinations of 1 GeV, 3 GeV, and 10 GeV muons are equally probable.

In Fig.6, the x-axis represents the neuron ID, the y-axis denotes the particle classes, and the color map indicates the efficiency.

Analyzing the results, we can see that:

- Neurons 8 and 19 are more active for events containing at least one 1 GeV particle.
- Neurons 0, 4, 12, 15, 17, and 18 aare more active for events containing at least one 3 GeV particle.

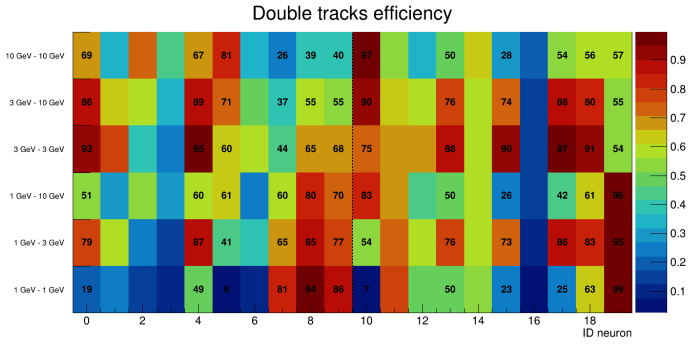

FIGURE 6: Two-Particles Events Efficiency Map. A vertical dashed line separates neuron belonging to layers L0 and L1.

- Neuron 10 is more active for events containing at least one 10 GeV particle (and neuron 5 in particular in 10 GeV - 10 GeV events).

The plot in Fig.7 shows the fake rates for each neuron, using data from single and double track events. In both cases, the fake rate has an order of magnitude at the percent level, and we can see that there is a systematic increase for the two-track case.

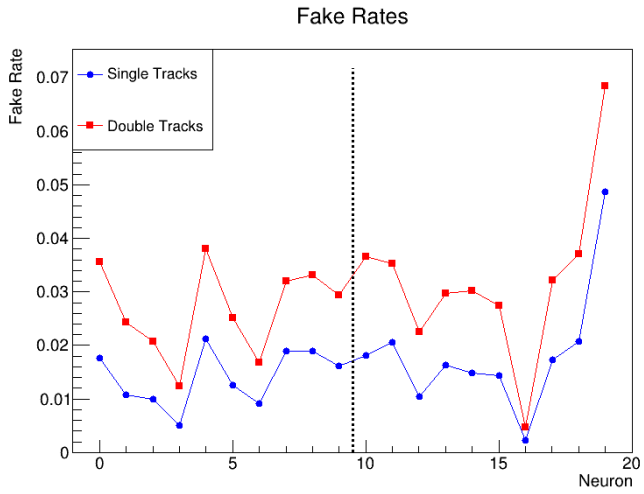

FIGURE 7: Fake rates for each neuron, comparing single and double track events. A vertical dashed line separates neuron belonging to layers L0 and L1.

4 Conclusions

In conclusion, the STDP method has successfully enhanced the specialization of neurons. Remarkably, without any further modifications, we are now able to apply the model to simple multitrack events, achieving comparable results for the first time.

4.1 Next Steps

- **Improved Testing:**
  - Implement a more stringent testing method by analyzing hits that cause neuron activation, as requested in the past by Mia. I have already made some developments toward this direction, but I need some other work to complete it.
  - In multitracks events, estimate the efficiency as a function of the angular distance of the tracks in the transverse plane.
  - Provide a quantitative estimate of the probability of observing a specific class of events given the list of active neurons.
- **Function Adjustments:** Improve the plot function for potentials due to changes in procedures caused by delays.
- **Anti-muon Management:** Encode the management of anti-muons by mirroring learned delays or using a set of neurons to scan the tracker in the opposite direction.
- **Hyperparameter Optimization:** Enhance performance and tolerable background levels through a more accurate hyperparameter search, possibly using a genetic algorithm, maybe allowing again the weights learning.

References