Hyperparameter Optimization for Deep Learning Models Using High Performance Computing

Muhammad Numan Anwar
Department of Physics
Polytechnic University of Bari
INFN, Italy

Outline

1. **Cluster Counting**
2. **Simulation based on Garfield ++**
3. **Peak Finding Algorithm**
4. **Long Short Term Memory (LSTM) Model**
5. **Clusterization**
6. **Convolution Neural Network (CNN) Model**
7. **Future Planning**

Main Goal of the Talk

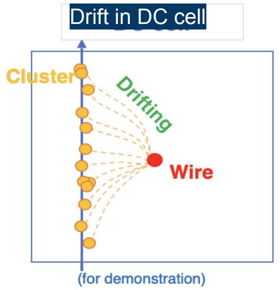
- The main goal of the talk is to train neural network models, such as the Long Short-Term Memory (LSTM) Model and Convolutional Neural Network (CNN) Model, using various hyperparameters like loss functions, activation functions, different numbers of neurons, batch sizes, and varying numbers of epochs. These models are trained for a two-step reconstruction algorithm, which involves peak finding and clusterization

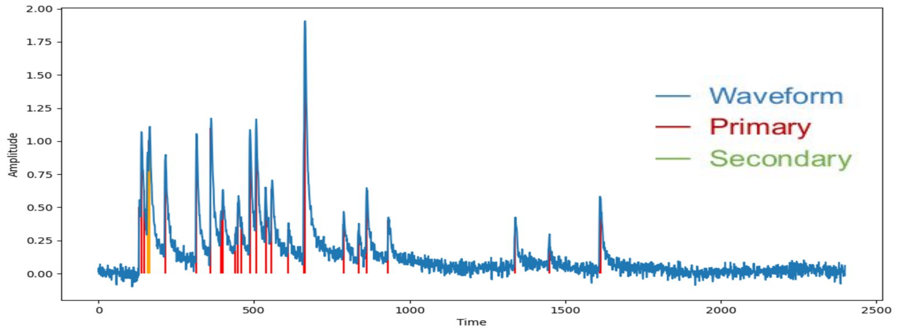
- For the peak finding algorithm, a trained LSTM model is used to discriminate between ionization signals (primary and secondary peaks) and noise in the waveform, addressing a classification problem

- Concurrently, a Convolutional Neural Network model is utilized to determine the number of primary ionization clusters based on the detected peaks, dealing with a regression problem

- It should be noted that the trained models (LSTM and CNN) are applied to simulations based on Garfield++

Cluster Counting in Drift Chambers





- **A charged particle is passed through mixture of gases (10% He and 90% C_4H_{10}) generate electron-ion pairs causing a read out signal (induce current)**

• Task:

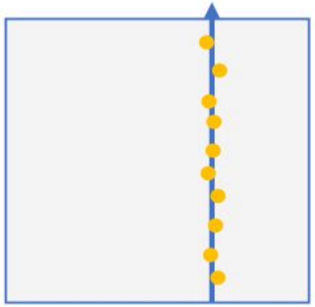
- **Both primary electrons and secondary electrons contribute to peaks in the waveform**
- **Find the number of peaks from primary electrons**

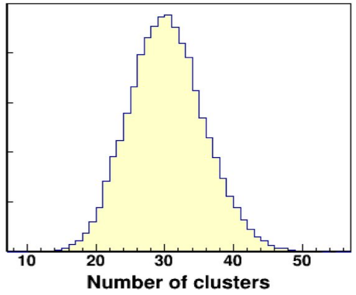
• Two step reconstruction algorithm:

- ***Peak finding:* Find all peaks (primary and secondary) in the waveform**
- ***Clusterization:* Determine the primary peaks from the founded peaks in step 1**

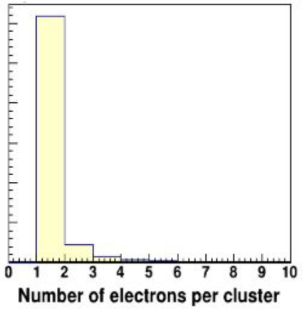
Cluster Counting vs dE/dx

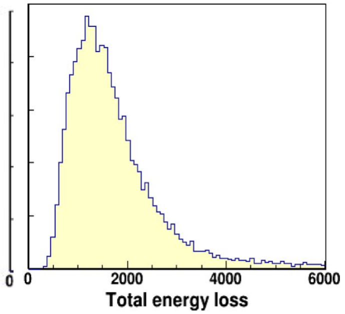
Primary Ionization



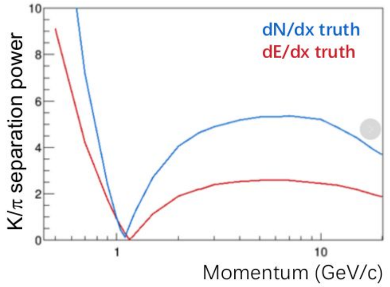


Secondary Ionization





K/p separation power
dN/dx vs dE/dx



- Energy loss per unit length (dE/dx) by the charge particle along the track would follow Landau distribution —> traditional method
- Number of primary ionization clusters per unit length (dN/dx) generated by the charge particle along the track would follow poisson distribution —> cluster counting technique
- We should use cluster counting techniques rather than traditional method. Because the separation power (distinguish between different particles based on their characteristics) in cluster counting tecnique is three time greater than dE/dx for the two particles (K/π)

Deep-Learning-Based Algorithm

- **Deep learning:** Learn rules from large amount of datasets

- Specifically, for cluster counting reconstruction:
  - Machine learning can make full use of the waveform information, not only information of pulse rising edge (e.g. derivative algorithm).
  - Machine learning can learn the hidden relationship in data (signal/noise characteristics, timing structure of primary/secondary peaks).
  - The reconstruction can easily be defined as classification and regression $\Rightarrow$ apply mature ML tools like TensorFlow, Keras etc.

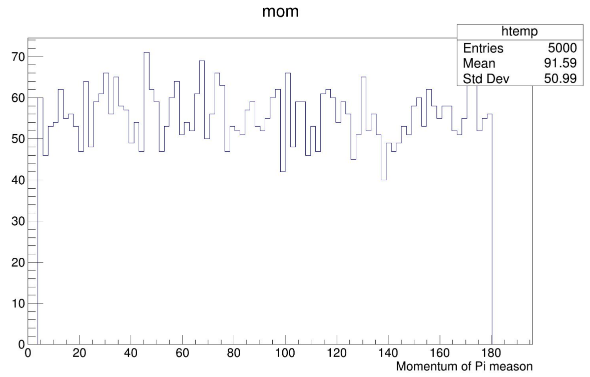
Simulation Based on Garfield ++

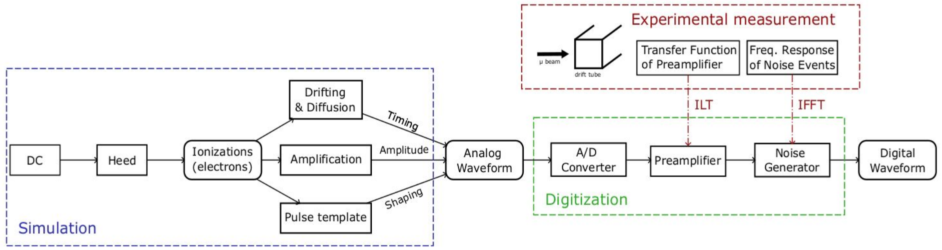
- **Pi meson is passed through mixture of gas having 90% He and 10% Isobutane C_4H_{10} by using a geometry of drift tubes mimicking what was used for the beam test at CERN in 2023**

- **The simulation parameters included a cell size of 1.5 cm, a sampling rate of 1.2 GHz, a time window of 2000 ns, and momentum pi-meson particles ranging from 4 to 180 GeV/c. The simulation was conducted using Garfield++**

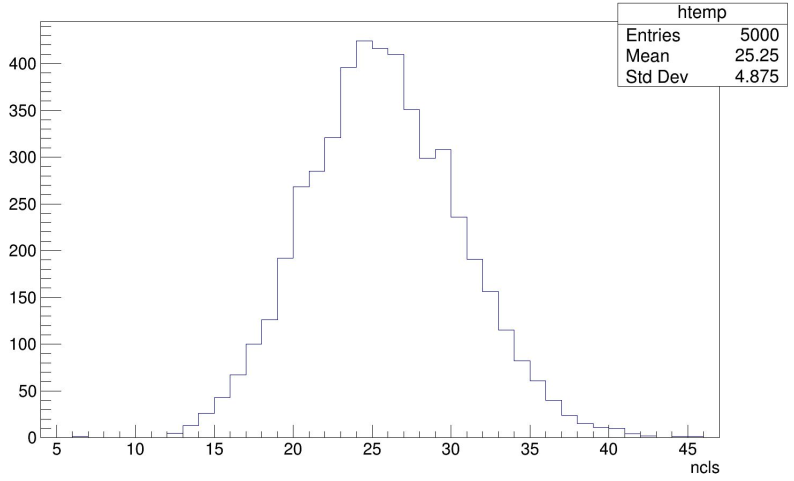
- **Following the simulation in Garfield++, I proceeded to plot various results for the study of the cluster counting techniques**

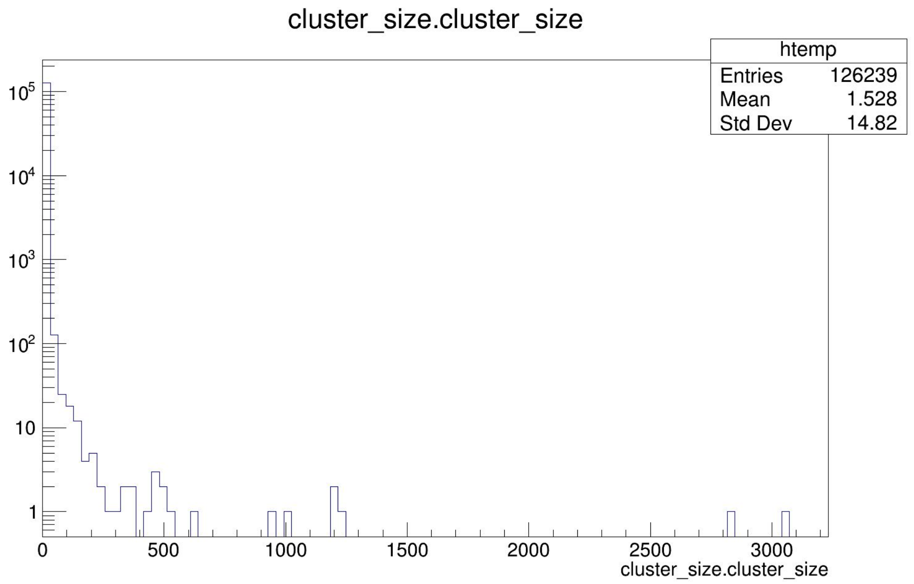
- **The simulation package creates analog induced current waveforms from ionizations. The digitization package incorporates electronics responses taken from experimental measurements and generates realistic digital waveforms**





Simulation Based on Garfield ++





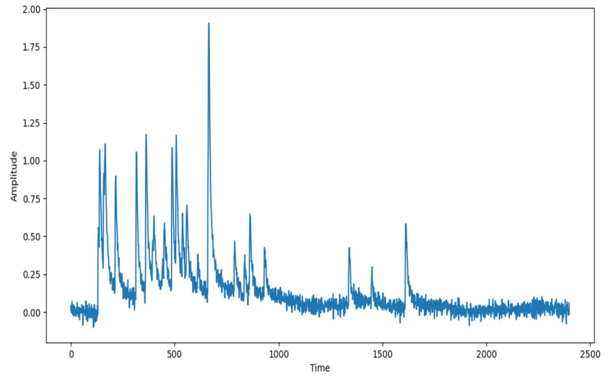
The above distribution shows the number of primary ionization clusters with mean value 25.25

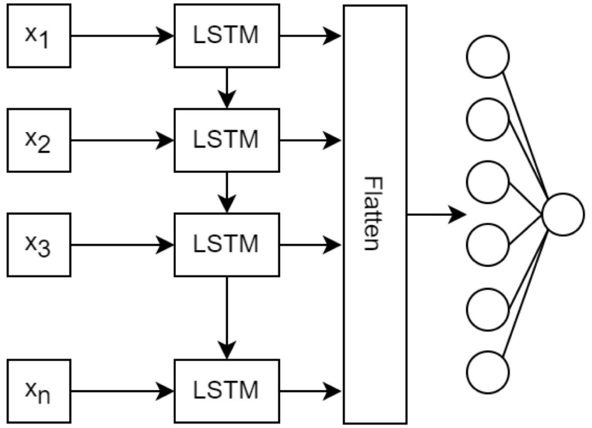
The above distribution shows the number of primary electrons per cluster with mean value 1.528

Two-Step Reconstruction Algorithm

Step1: Peak Finding

Waveform

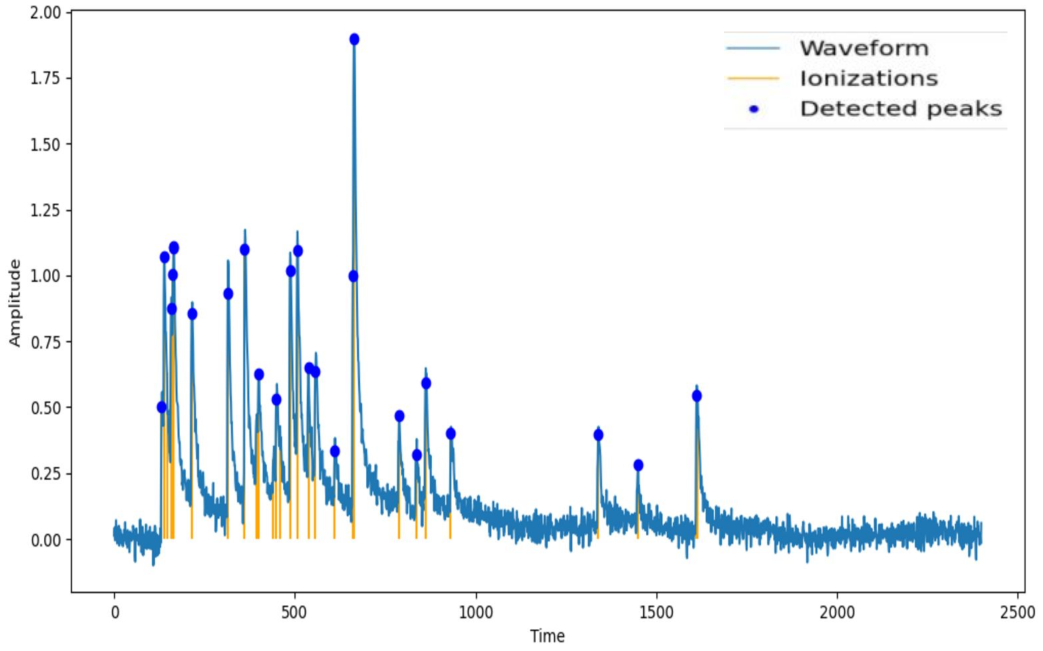




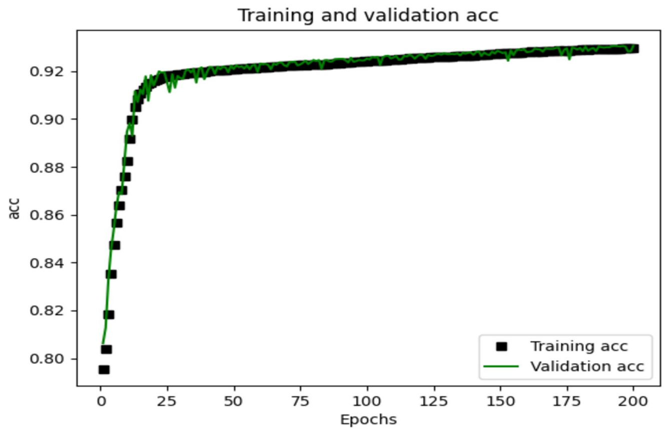
- A classification problem to classify ionization signals (Primary and Secondary Ionizations) and noises in the waveform by using Long Short Term Memory (LSTM) model

- Labels: Signal or Noise
- Features: Slide windows of peak candidates, with a shape of (15, 1)
- The data of waveform is time sequence data, which is suitable for Long short Term Memory (LSTM) model

Evaluation by Waveform

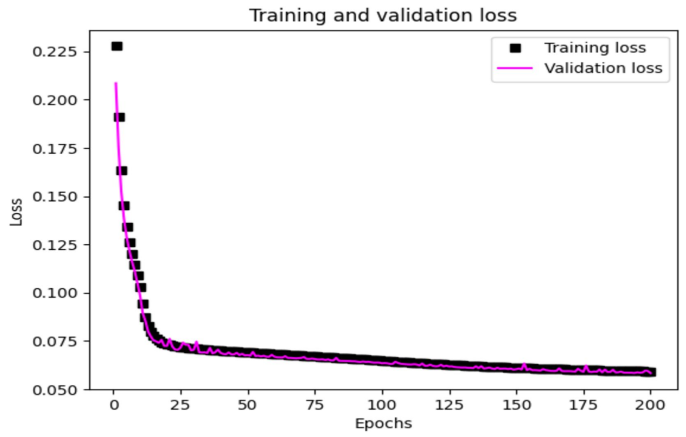


Performance of the LSTM Model



$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}}$$

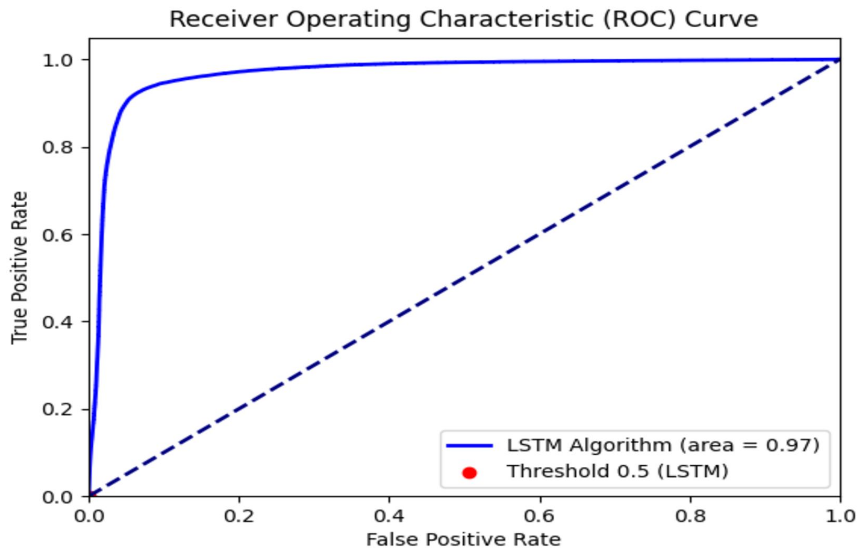
$$MSE(y^{(i)}, y^{(i)}_{pred}) = \frac{\left(y^{(i)} - y^{(i)}_{pred}\right)^2}{n}$$



The above plot Accuracy VS Epoch show us that the training and validation accuracy increases over the epochs and then it become approximately constant which shows a best trained model

The above plot loss VS epoch show us that the training and validation loss decreases over the epochs and then it become approximately constant which shows a best trained model

Performance of the LSTM Model

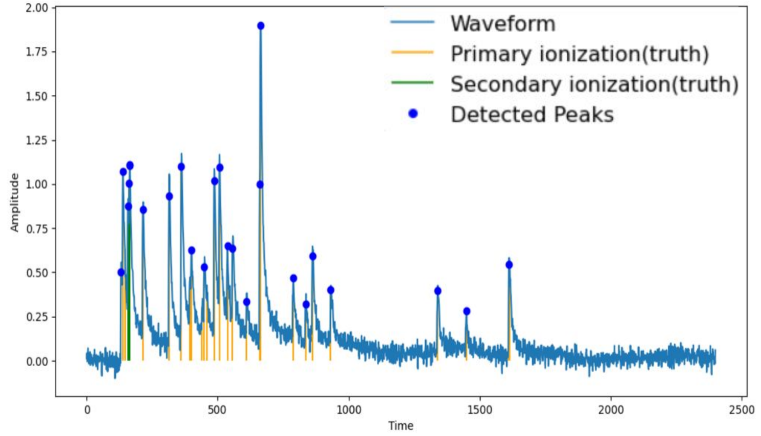


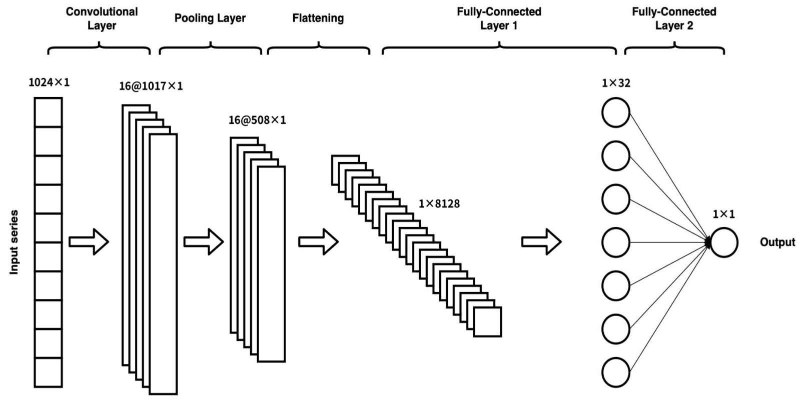
TPR = TP/(TP+FN)
FPR = FP/(FP+TN)

		Prediction	
		Sig	Noise
Truth	Sig	TP	FN
	Noise	FP	TN

The above plot show ROC curve for the LSTM model with Area under the curve value 0.97 with threshold value 0.5 which show a best classification

Step2: Clusterization

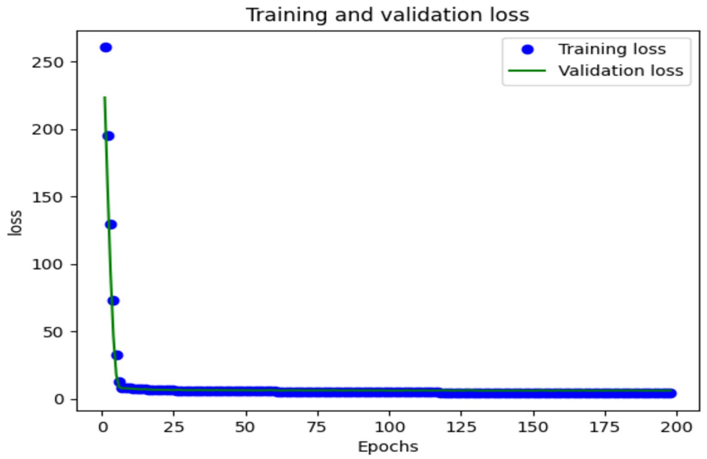




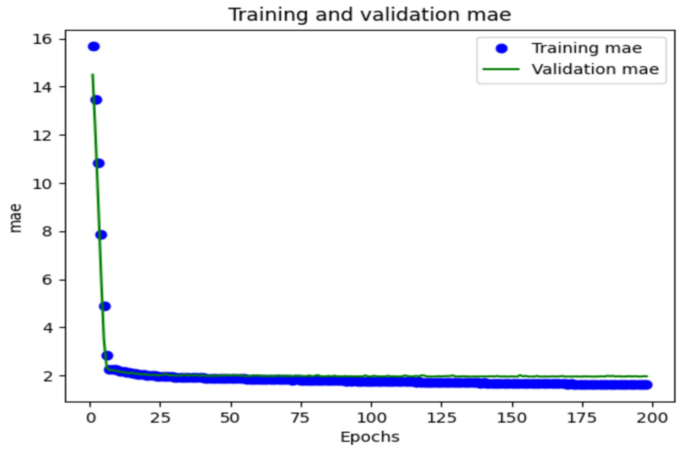
- A regression problem to predict Number of primary ionization clusters based on the primary detected peaks by using Convolutional Neural Network (CNN) model
- The peaks found by peak finding algorithm would be training sample of this algorithm

- Labels: Number of clusters from MC truth
- Features: Time list of the detected times in the previous step encoding in an (1024, 1) array.
- A regression problem

Performance of the CNN Model



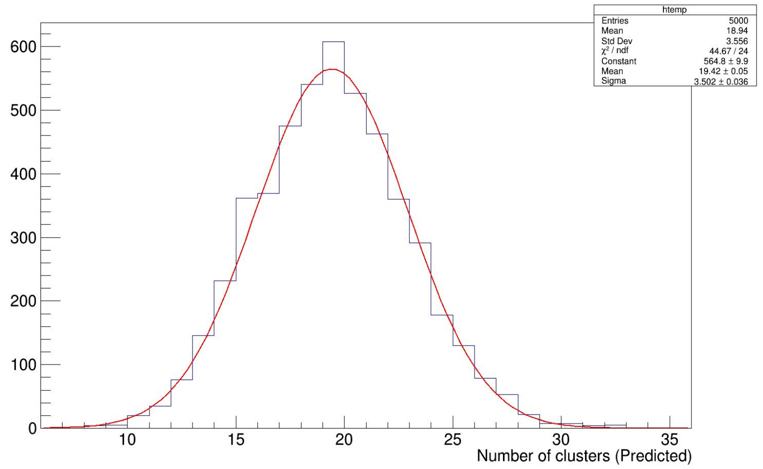
$$MAE(y^{(i)}, y^{(i)}_{pred}) = \frac{\left|y^{(i)} - y^{(i)}_{pred}\right|}{n}$$

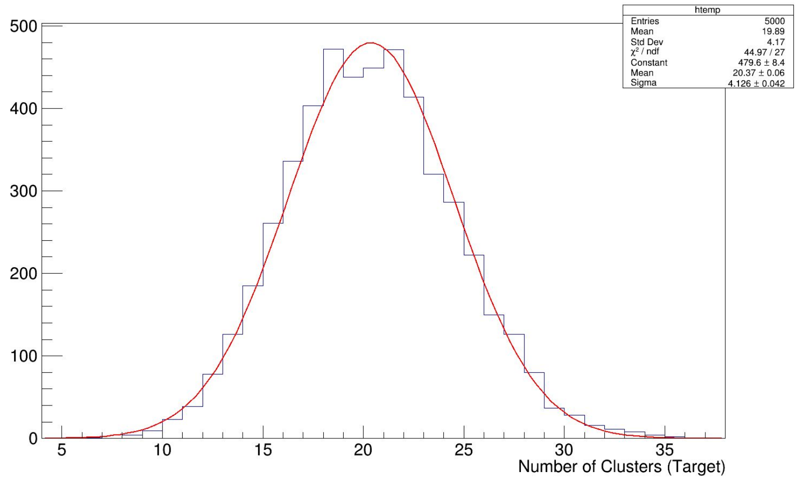


The above plot loss VS epoch show us that the training and validation loss decreases over the epochs and then it become constant shows us a best trained model

The above mean absolute error VS epoch show us that the training and validation loss decreases over the epochs and then it become constant shows us a best trained model

Predictions of the N_{cls} by CNN Models





- Number of Primary ionized clusters with mean value (18.94) predicted by CNN Model based on the detected primary peaks with mean value (19.89)
- Good Gaussian distribution

Summary

- Particle identification (PID) is essential in most particle experiments
- Cluster counting (CC) in gaseous detector is the most promising breakthrough in PID due to potential of 3 times better resolution than traditional method
- I executed the code pertaining to the simulation of particles traversing a gas mixture made out of 90% Helium (He) and 10% Isobutane (C_4H_{10}) filling drift tubes with the same geometry of the one used for the beam test at CERN in 2023
- Following the simulation in Garfield++, I proceeded to plot various results for the study of the cluster counting technique
- A two-step reconstruction algorithm involving peak finding (Discriminate signal from background in the waveform) and clusterization (Primary ionization clusters based on the detected peaks) was used in cluster counting techniques
- For the peak finding algorithm, I trained Long Short Term Memory (LSTM) model by using mean square error (MSE) as the loss function, sigmoid and rectified linear unit (ReLU) as activation functions, stochastic gradient descent (SGD) as the optimizer, with a batch size of 250 and 200 epochs
- Concurrently, I trained Convolutional Neural Network (CNN) Model using mean absolute error (MAE) as a metrics, Root mean square propagation as the optimizer, with a batch size of 250 and 200 epochs to determine the number of primary clusters based on the detected peaks, dealing with a regression problem

Future Planning

- ***Now, we will apply the trained models to the real beam test data to classify signals from noise in the waveform and determine the number of primary clusters based on the detected peaks***

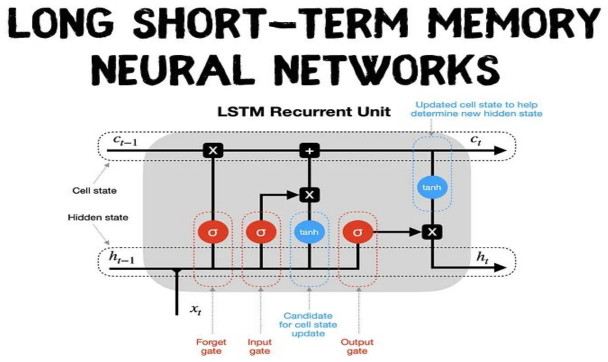
Thank you

ACCURACY and LSTM

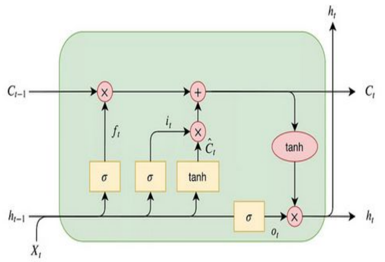
- The accuracy is defined as the ratio between the number of correct predictions to the total number of predictions
- Accuracy values range between 0 and 1. Obviously an accuracy values near to 1 means that our model fits well the datasets

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}}$$

- Forget Gate: This gate determines what information from the previous cell state should be forgotten or retained.
- Input Gate: It controls what new information should be stored in the cell state.
- Output Gate: This gate defines the output of the LSTM cell, considering the current input and the updated cell state


LONG SHORT-TERM MEMORY NEURAL NETWORKS
LSTM Recurrent Unit
Updated cell state to help determine new hidden state
c_{t-1}
c_t
Cell state
Hidden state
h_{t-1}
h_t
x_t
σ
tanh
Forget gate
Input gate
Candidate for cell state update
Output gate

Long SHort Term Memory (LSTM)



$$f_t = \sigma(W_f \cdot [h_{t-1}, X_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, X_t] + b_i)$$

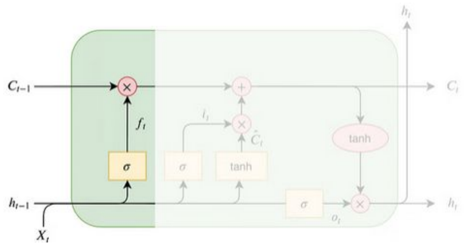
$$o_t = \sigma(W_o \cdot [h_{t-1}, X_t] + b_o)$$

$$\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, X_t] + b_C)$$

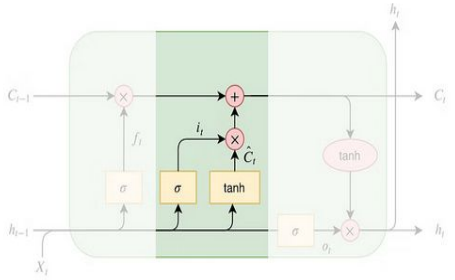
$$C_t = i_t \cdot \hat{C}_t + f_t \cdot C_{t-1}$$

Forget Gate

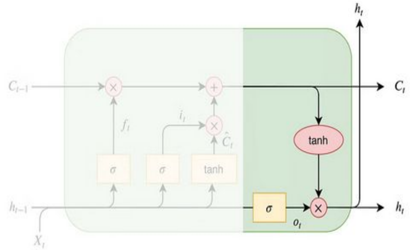
state and the new input data.



Input Gate



Output Gate



EXAMPLES of LOSS FUNCTIONS

- Mean Squared Error(MSE)/ Quadratic Loss/ L2:

$$MSE(y^{(i)}, y^{(i)}_{pred}) = \frac{\left(y^{(i)} - y^{(i)}_{pred}\right)^2}{n}$$

- Mean Absolute Error (MAE)/ L1 Loss:

$$MAE(y^{(i)}, y^{(i)}_{pred}) = \frac{\left|y^{(i)} - y^{(i)}_{pred}\right|}{n}$$

- Mean Bias Error (MBE):

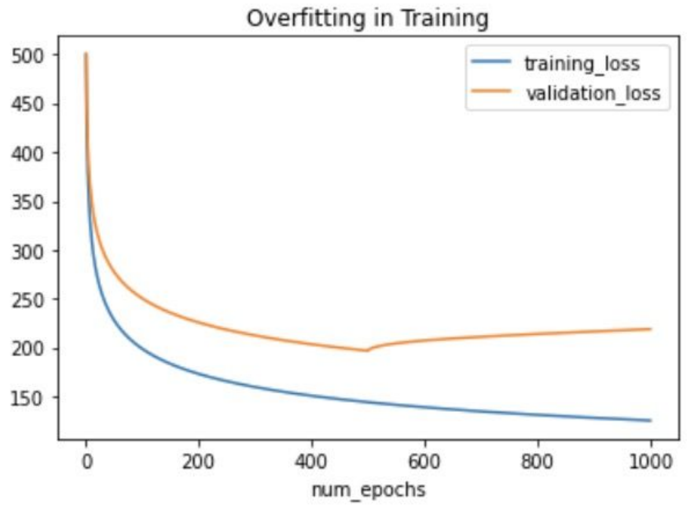
$$MBE(y^{(i)}, y^{(i)}_{pred}) = \frac{\left(y^{(i)} - y^{(i)}_{pred}\right)}{n}$$

NUMBER OF EPOCHS

- **Epoch**: In terms of artificial neural networks, an epoch refers to one cycle through the full training dataset
- Number of epochs is a delicate choice:
  - ❑ A large number of epochs can induce our model to an overfitting problem
  - ❑ Too small number of epochs can lead to an under fitting problem
- To avoid a wrong choice we can use the ' EarlyStopping', also implemented by Keras:
  - ❑ It allows to stop the training when a monitor (set by us and tipically the loss function) has stopped improving.

OVERFITTING PROBLEM

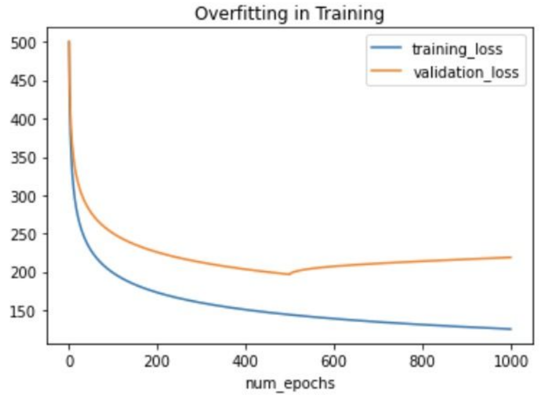
- The more complex the model is, the higher is the risk of overfitting
- Here a clear example of overfittig, the train loss keeps going down while the validation loss get worse. It is always important to split the training in train and validation set and to have a clear picture of the train history
- In order to avoid overfitting and make the training stable we have different approach

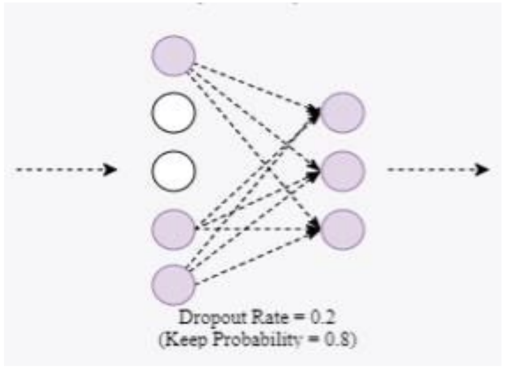


FACING OVERFITTING PROBLEM

- Introduce a callback function that stops the training if the validation loss get worse and restore the best parameters (**Early Stop function**).

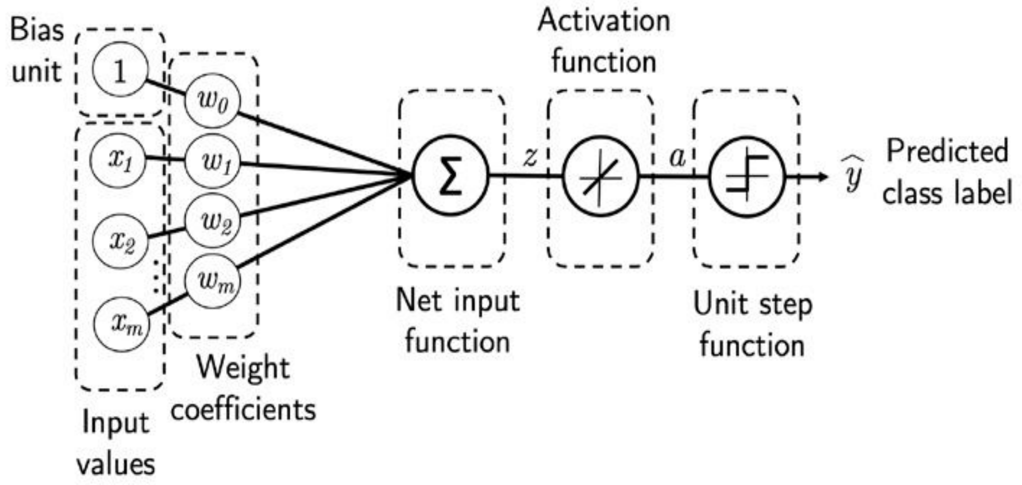
- **<u>Dropout:</u>** it refers to the practice of disregarding certain nodes in a layer at random during training. A dropout is a regularization approach that prevents overfitting by ensuring that no units are co-dependent with one another





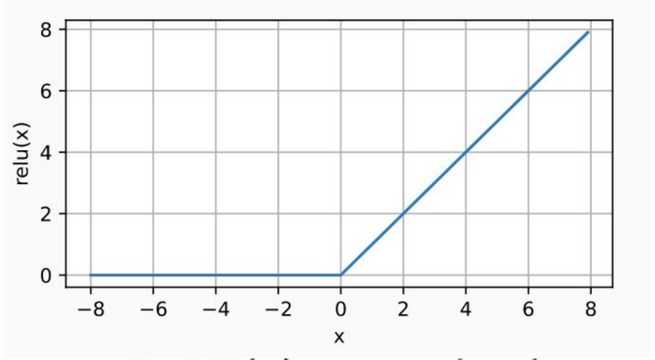
ACTIVATION FUNCTION

- Most of them provides to add non-linearity to the mode
- The activation function σ has as input the weighted sum of the input variables x, added with the bias b
- The functions are in general differentiable operators in order to transform the inputs to outputs

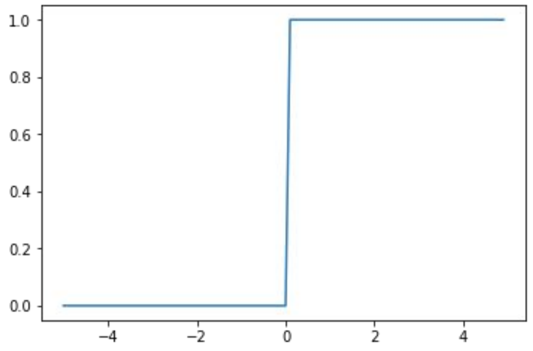


RECTIFIED LINEAR UNIT (RELU)

- One the most popular non-linear activation function is the REctified Linear Unit (ReLU)
- It provides a non-linear transformation and returns the max value between the input x (the argument) and 0
- The ReLU function is also differentiable in as given below:



$$ReLU(x) = max(0, x)$$



$$\frac{dReLU(x)}{dx} = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$$

SCALED EXPONENTIAL LINEAR UNIT (SELU)

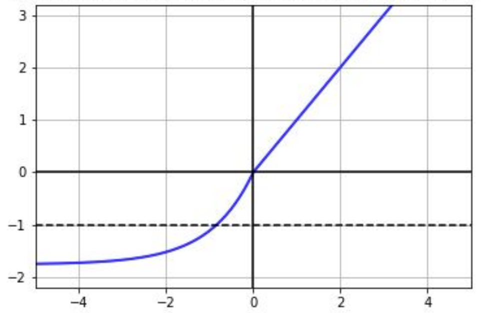
- Another choice is the Scaled Exponential Linear Unit (SELU)
- The functions depends on two parameters and the equation is the following:

$$SELU(x) = \lambda \begin{cases} \alpha(e^x - 1) & x \leq 0 \\ x & x > 0 \end{cases}$$

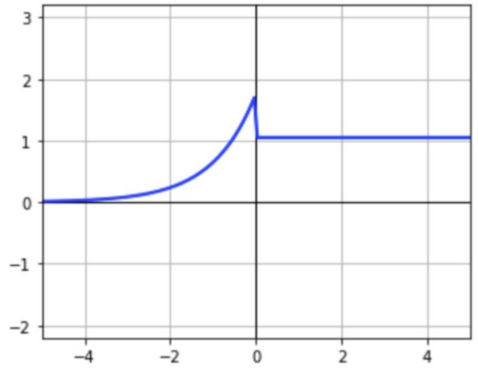
- The function is not differentiable in zero

$$\frac{dSELU(x)}{dx} = \lambda \begin{cases} \alpha e^x & x \leq 0 \\ 1 & x > 0 \end{cases}$$

SELU activation function ($\alpha \approx 1.6732$ and $\lambda \approx 1.0507$)



3
2
1
0
−1
−2
−4
−2
0
2
4



3
2
1
0
−1
−2
−4
−2
0
2
4