Cluster Counting Technique

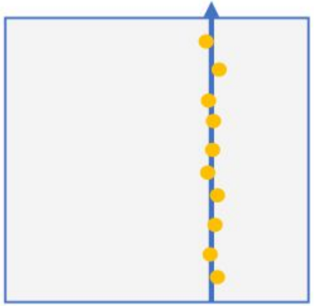
Muhammad Numan Anwar
Department of Physics
Polytechnic University of Bari
INFN, Italy

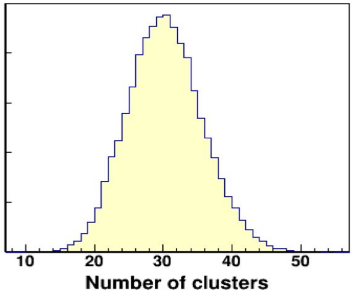
Outline

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**

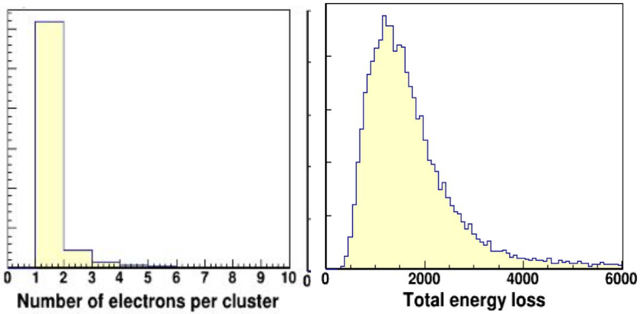
Cluster Counting vs dE/dx

Primary Ionization

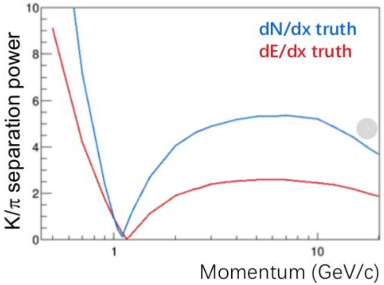




Secondary Ionization



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



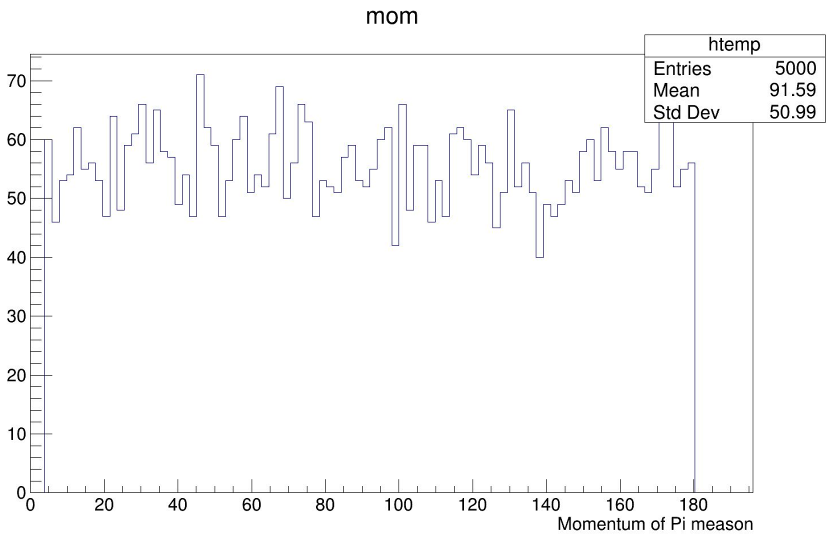
- **Energy loss per unit length(dE/dx) would follow Landau distribution**
- **Number of primary ionization clusters per unit length (dN/dx), Poisson distribution —> Cluster Counting Technique**

Simulation Based on Garfield ++

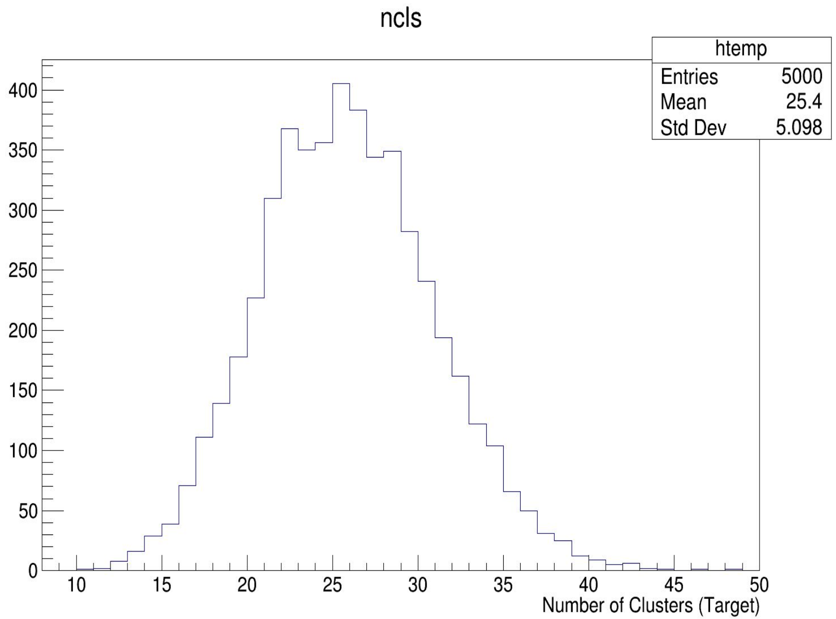
- **Pi meson is passed through mixture of gas having 10% He and 90% Isobutane C_4H_{10} by using a geometry of drift tubes mimicking what was used for the beam test at CERN in 2023**

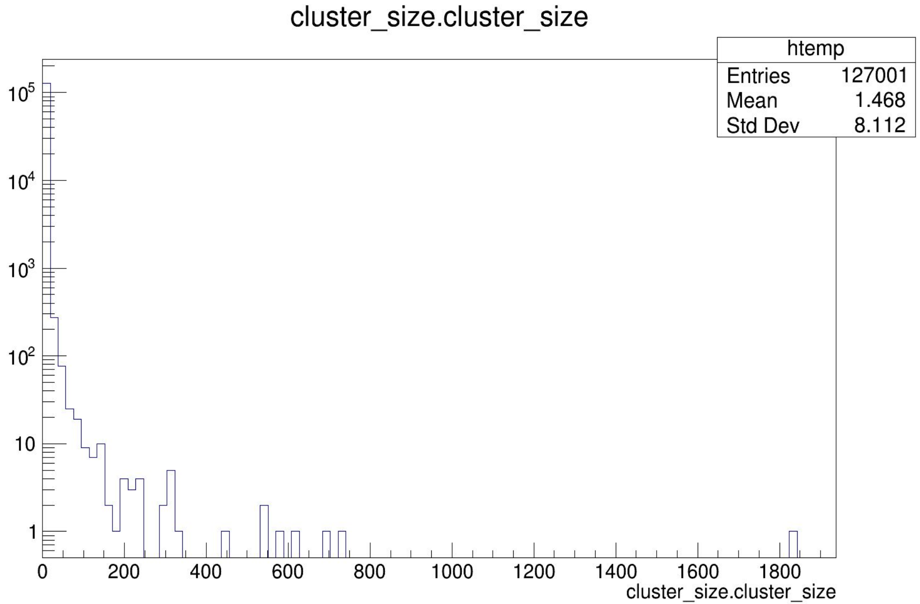
- **The simulation parameters included a cell size of 1.5 cm, a sampling rate of 1.2 GHz, a time window of 2000 ns, and momentum pi-meson particles ranging from 4 to 180 GeV/c. The simulation was conducted using Garfield++**

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



Simulation Based on Garfield ++





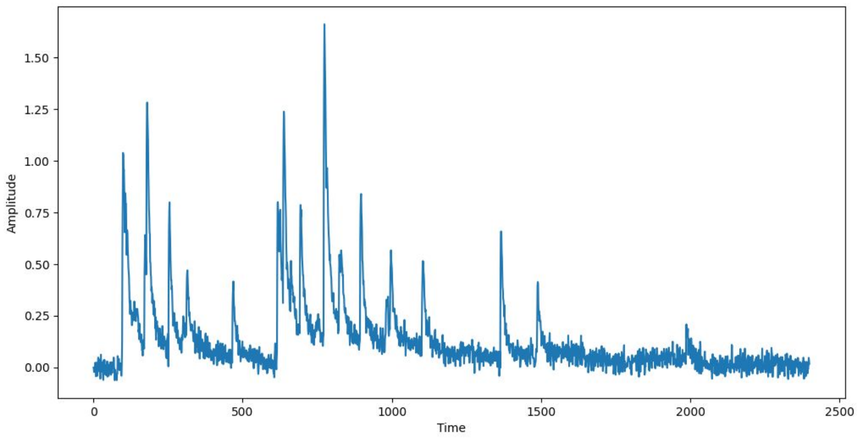
The above distribution shows the number of primary ionization clusters with mean value 25.5

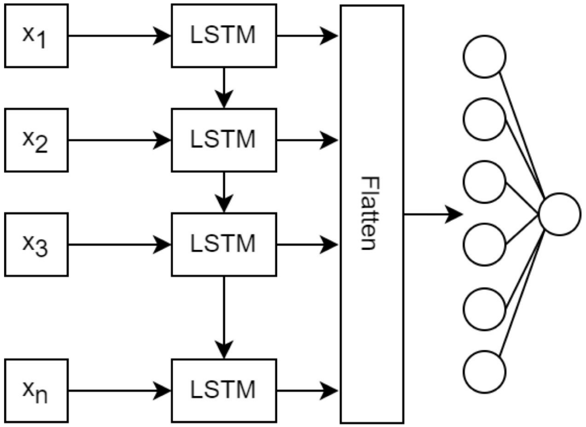
The above distribution shows the number of primary electrons per cluster with mean value 1.468

Two-Step Reconstruction Algorithm

Step1: Peak Finding

Waveform

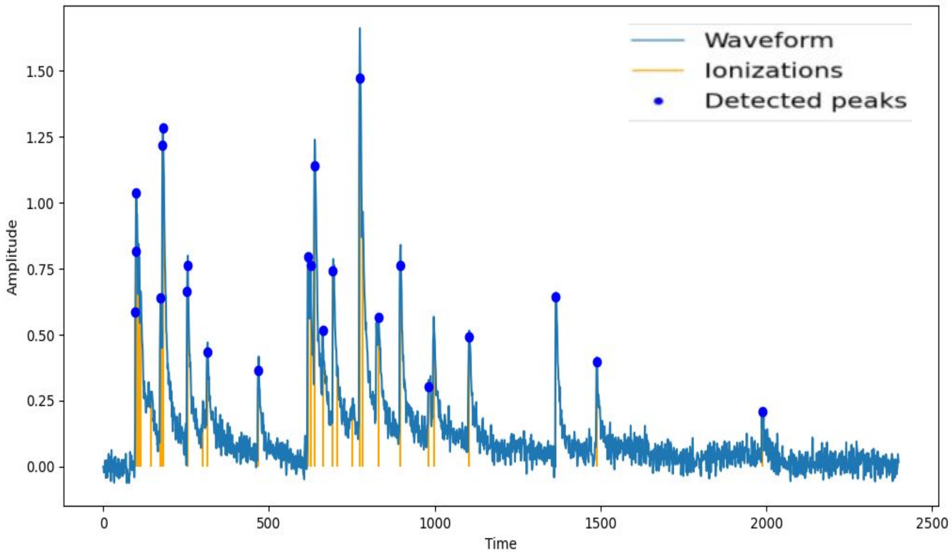




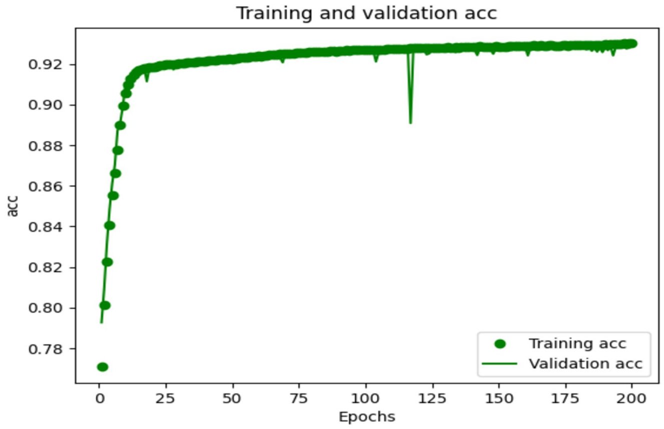
- **A classification problem to classify ionization signals (Primary and Secondary Ionizations) and noises in the waveform**

- **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 LSTM**

Evaluation by Waveform

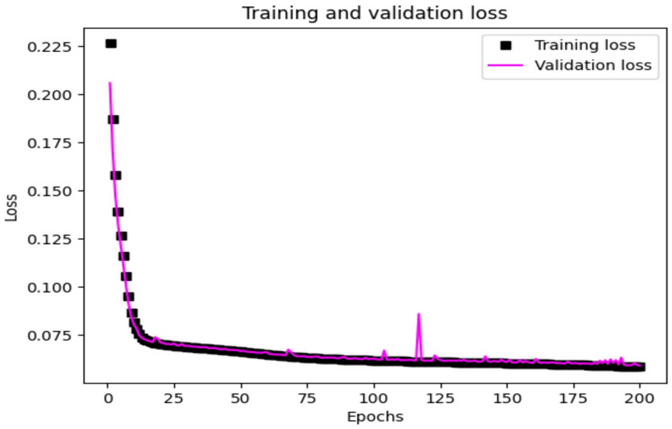


Performance of the LSTM Model for 5000 events



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

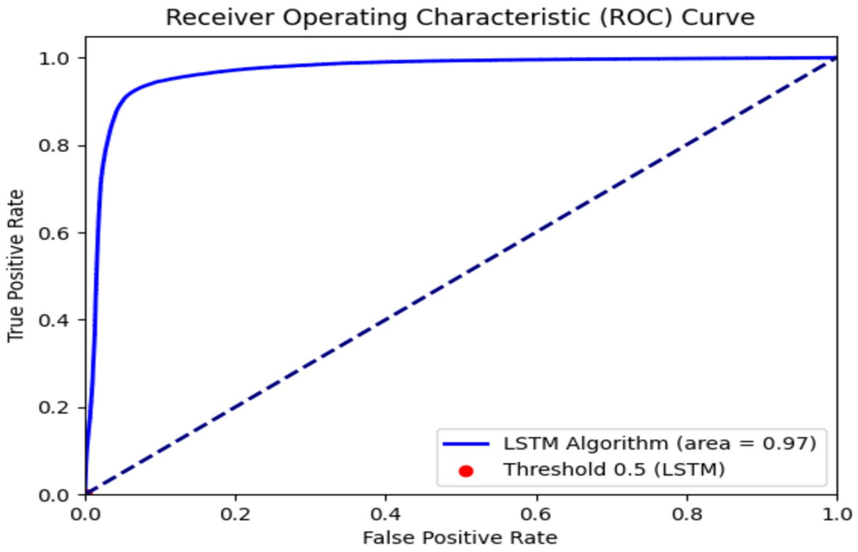
$$MSE(y^{(i)}, y^{(i)}_{pred}) = \frac{\left(y^{(i)} - y^{(i)}_{pred}\right)^2}{n}$$



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

Performance of the LSTM Model for 5000 events

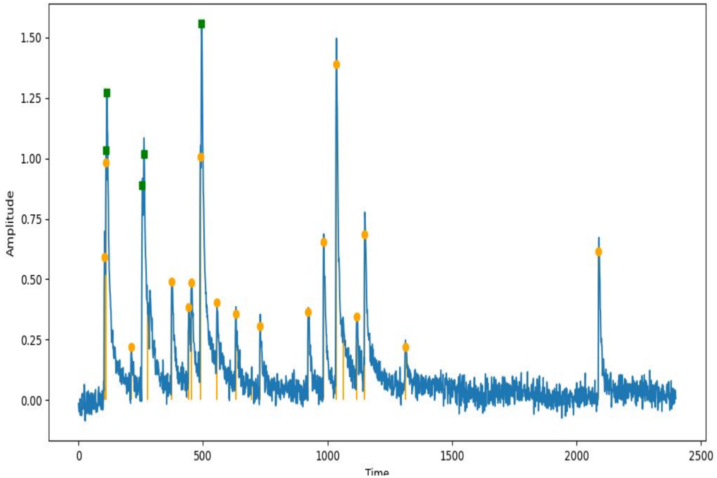


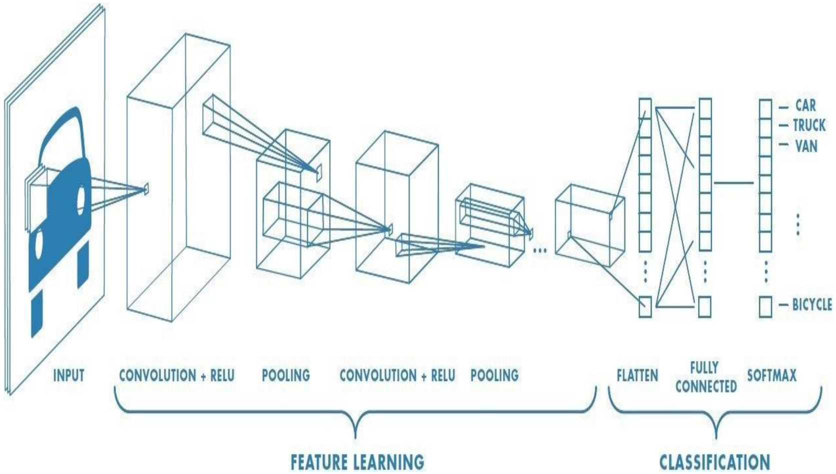
The above plot show ROC curve for the LSTM model with Area under the curve value 0.97 with threshold value 0.5 which show a best classification

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

		Prediction	
		Sig	Noise
Truth	Sig	TP	FN
	Noise	FP	TN

Step2: Clusterization for 5000 Events

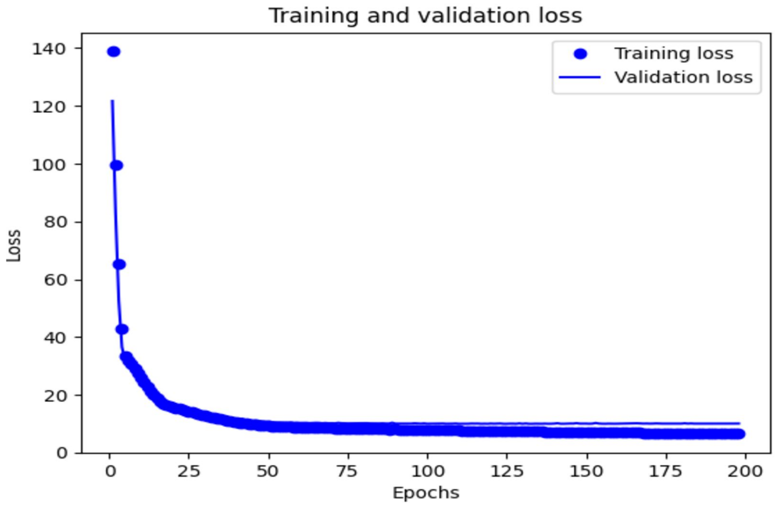




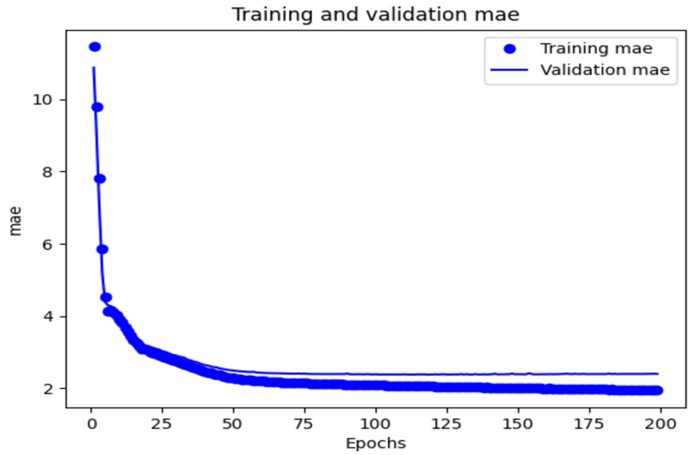
- Determine the number of clusters from the detected peaks(Yellow dots represents primary and Red dots shows secondary ionizations) is knowns as Regression Problem in ML

- Extracting features from input
- 1D CNN can handle sequence data

Performance of the Model for 5000 Events



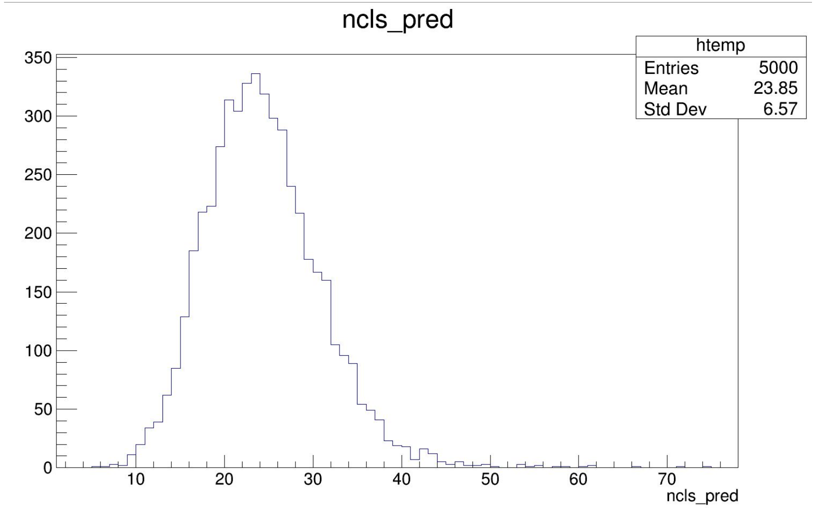
The above plot loss VS epoch show us that the training and validation loss decreases over the epochs and then it become constant shows us a best trained model



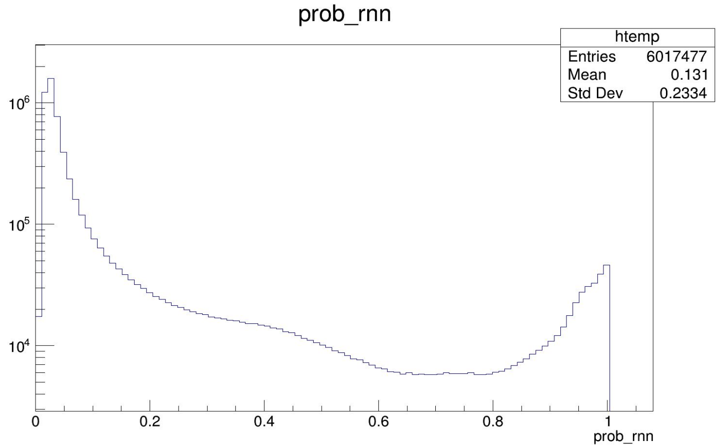
The above mean absolute error VS epoch show us that the training and validation loss decreases over the epochs

Predictions of the Two Models for 5000 events

Number of Primary clusters predicted by CNN Model



Prob of the LSTM Model to Classify Signal and Background



Future Planning

Now, we aim to enhance the performance of these models by using:

- **Different optimizers, loss and activation functions**
- **Adding more layers, different number of filters and kernel sizes**
- **Adjusting different number of training epochs etc.**
- **Once we achieve the best performance, we will apply our model to the test beam data.**

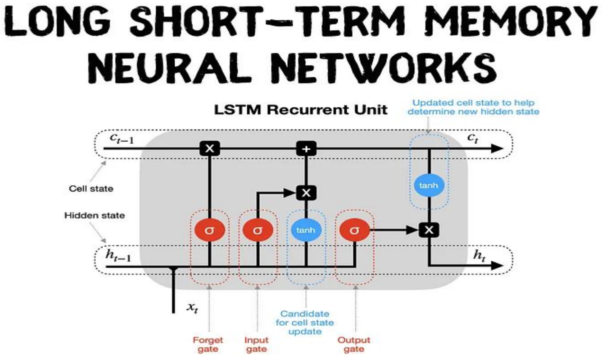

Thank you

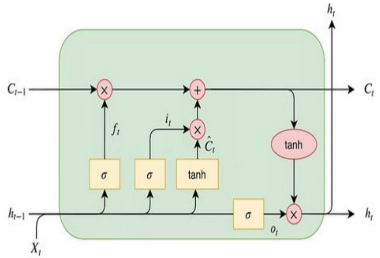
ACCURACY and LSTM

- 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





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

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

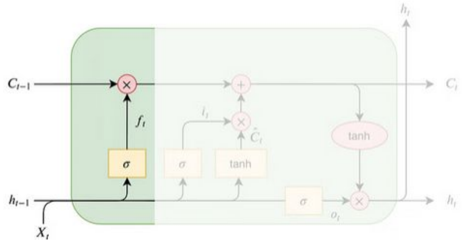
$$o_t = \sigma(W_o \cdot [h_{t-1}, X_t] + b_o)$$

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

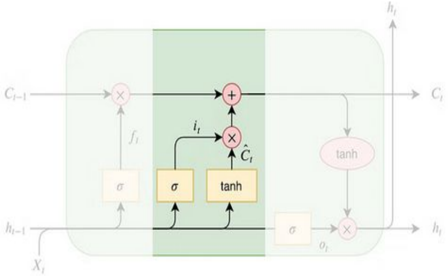
$$C_t = i_t \cdot \hat{C}_t + f_t \cdot C_{t-1}$$

Forget Gate

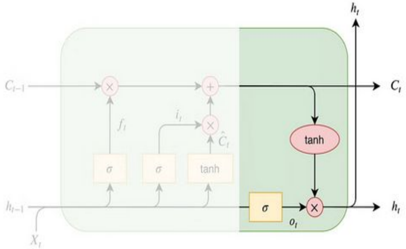
state and the new input data.



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):

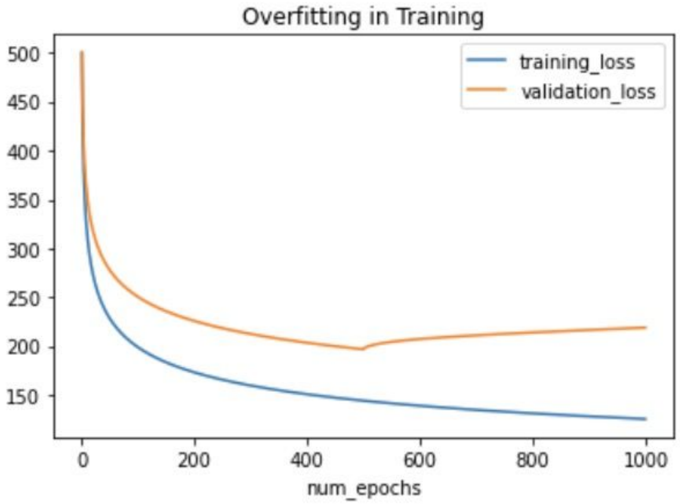
$$MBE(y^{(i)}, y^{(i)}_{pred}) = \frac{\left(y^{(i)} - y^{(i)}_{pred}\right)}{n}$$

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

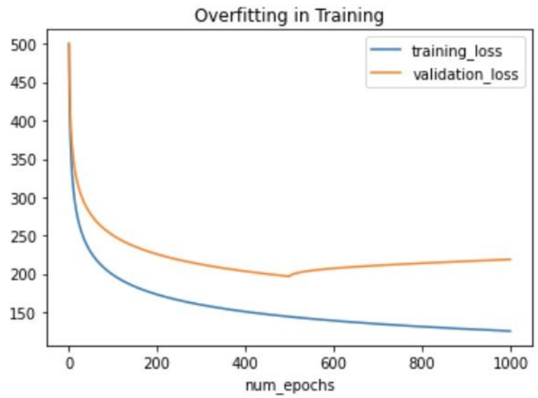
- The more complex the model is, the higher is the risk of overfitting
- Here a clear example of overfittig, the train loss keeps going down while the validation loss get worse. It is always important to split the training in train and validation set and to have a clear picture of the train history
-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

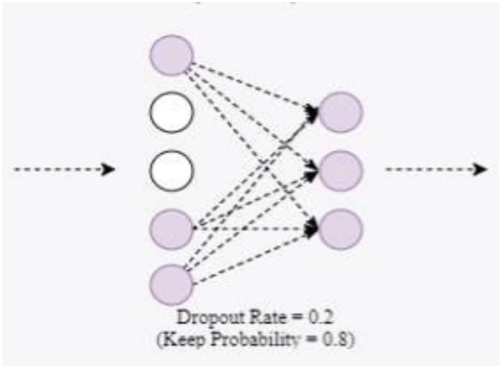


FACING OVERFITTING PROBLEM

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

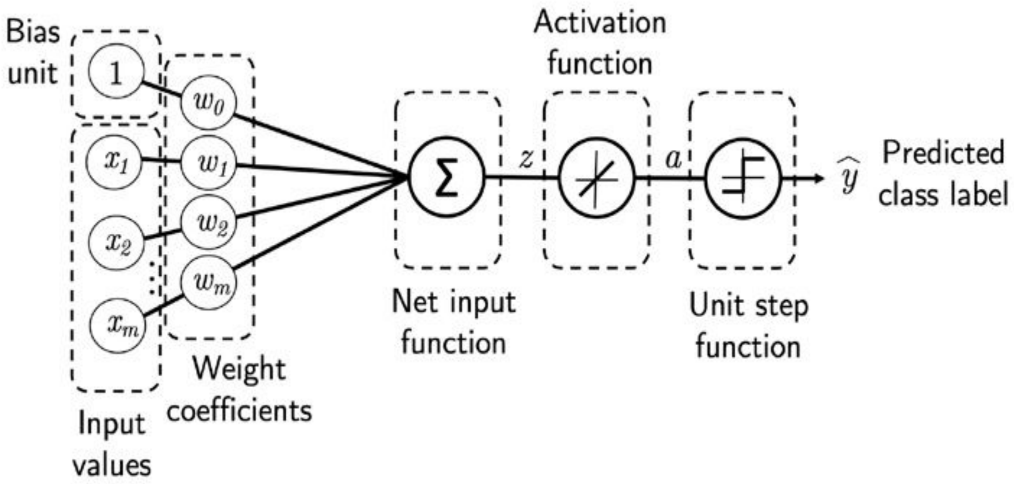
- <u>**Dropout:**</u> it refers to the practice of disregarding certain nodes in a layer at random during training. A dropout is a regularization approach that prevents overfitting by ensuring that no units are co-dependent with one another





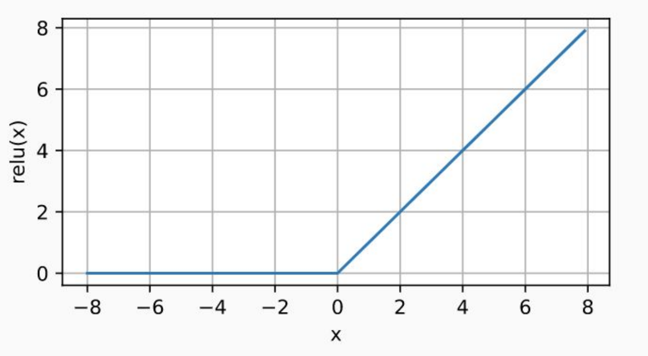
ACTIVATION FUNCTION

- 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



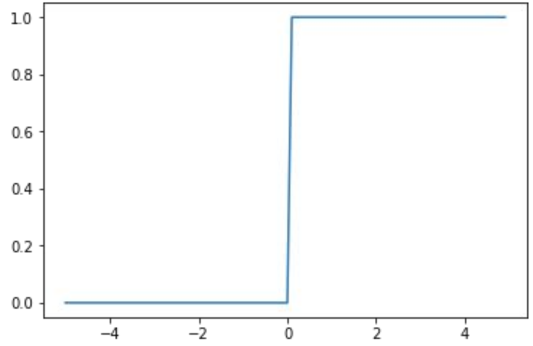
RECTIFIED LINEAR UNIT (RELU)

- 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

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