Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Simulation based on Garfield++ and Hyperparameter Optimization for Deep Learning Model Using High Performance Computing

INFN Istituto Nazionale di Fisica Nucleare

Speaker : Muhammad Numan Anwar
PhD Student at POLIBA INFN, Bari

Institute of High Energy Physics Chinese Academy of Sciences

Bari-Lecce Meeting 05 July 2024

Outline

- Simulation based on Garfield ++ for 2023 data
- Optimization of Hyperparameters for Long Short Term Memory (LSTM) Model Using HPC Resources
- Long Short Term Memory (LSTM) Model for Peak Finding Algorithm
- Optimization of Hyperparameters for Convolutional Neural Network(CNN) Model Using HPC Resources
- Convolutional Neural Network(CNN) Model for Clusterization Algorithm
- Future planing

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 etc by using HPC resources such as memory requests, job duration, and CPU usage etc. 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**

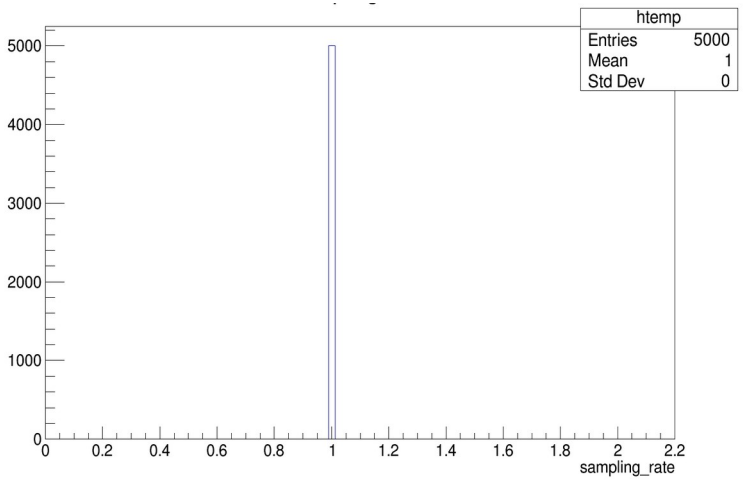
- **Concurrently, a Convolutional Neural Network model is utilized to determine the number of primary ionization clusters based on the primary detected peaks, dealing with a regression problem**

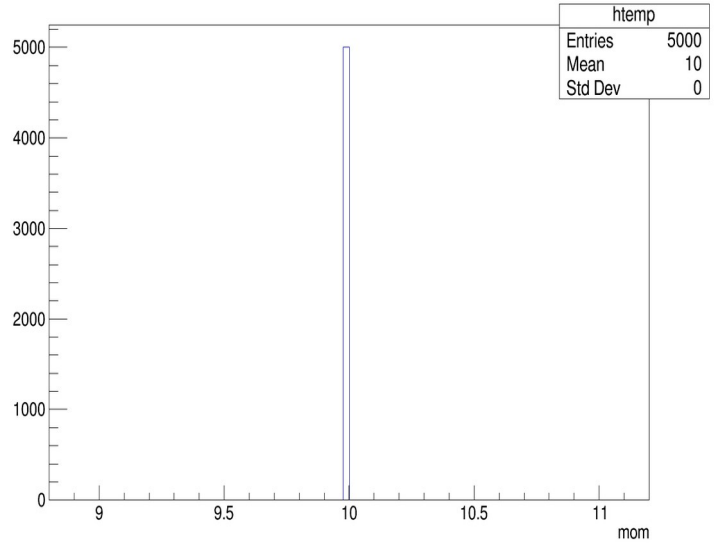
- **It should be noted that the trained models (LSTM and CNN) are applied to simulations based on Garfield++**

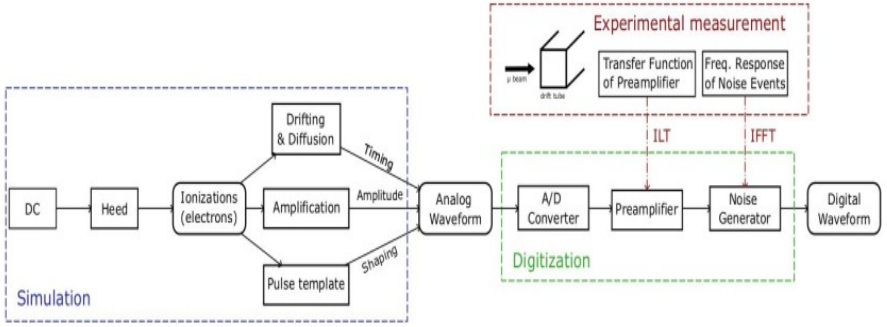
Simulation Based on Garfield ++ for 2023 data

Simulation Based on Garfield ++

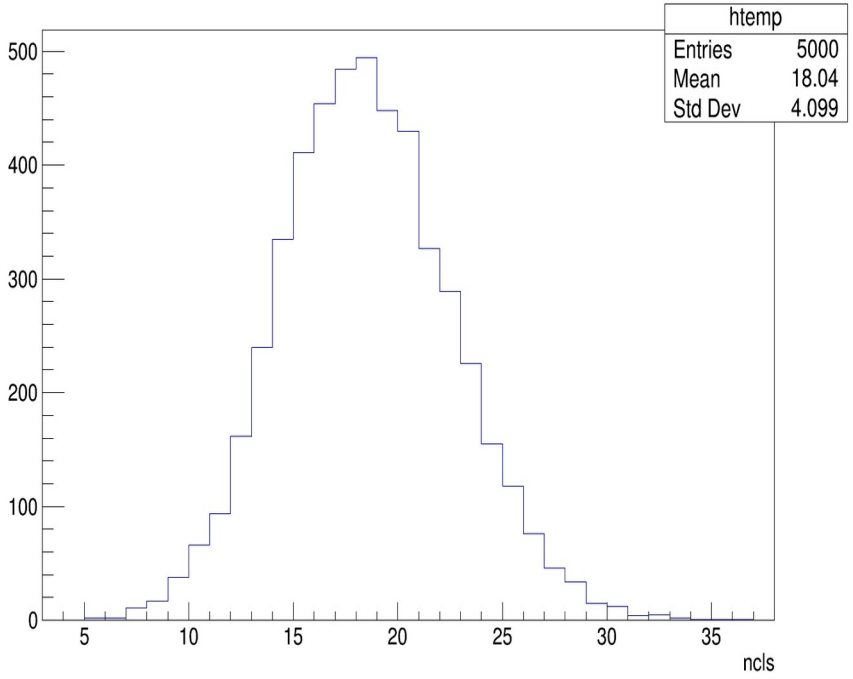
- **Muon particles is passed through mixture of gas having 90% He and 10% Isobutane C_4H_{10} by using a geometry of drift tubes mimicking what was used for the beam test at CERN in 2023**
- **The simulation parameters included a cell size of 0.8 cm, a sampling rate of 1.0 GHz, a time window of 800 ns, 45 angle between the z axis of drift tube chamber and track of the muon particle, and momentum muon particles with momentum 10 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**

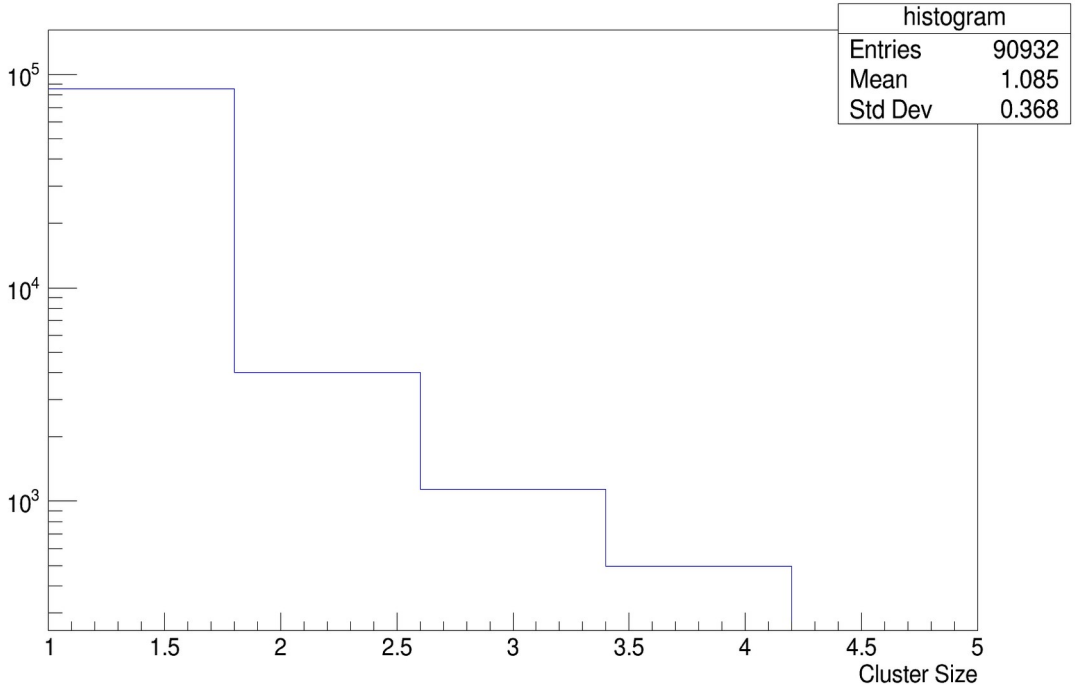






Simulation Based on Garfield ++





The above distribution shows the number of primary ionization clusters with mean value 18.04

The above distribution shows the number of ionized electrons per cluster with mean value 1.085

Training LSTM and CNN Model for Two-Step Reconstruction Algorithm_

First Model

Optimization of Hyperparameters for Long Short Term Memory (LSTM) Model Using HPC Resources

Optimization of Hyperparameters for Long Short-Term Memory Models Using HPC Resources

- **Currently, I designed a task involving the simultaneous submission of several jobs using local HPC Resources**
- **The purpose of this task is to train Long Short-Term Memory (LSTM) models to classify signals from background, a process known as a classification task.**
- **To achieve this task, I utilized various hyperparameters, including activation functions, optimizer Epochs, batch size, patience, and dropout rates etc**
- **Additionally, I managed different resources such as memory requests, Job duration, and CPU Usage etc**
- **Then, I selected the best model based on evaluation metrics such as the highest AUC value among all configurations**

```
# Arrays defining different configurations
vminimizer=("sgd"  "rmsprop" "adam")
vneuron=("relu selu" "relu sigmoid"  "selu sigmoid")
vpatiences=("30")
vbatches=("150"  "250")
vtopologies=("96 128 1" "32 64 1" "16 32 1" "8 16 1")
dropout=("0.1"  "0.0")
vepochs=(100)
```

Different hyperparameters for LSTM peak finding model are shown in the screen shot

Structure of Best LSTM Model Using Different Hyperparameters

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 96)                37632
_________________________________________________________________
flatten (Flatten)            (None, 96)                0
_________________________________________________________________
dense (Dense)                (None, 128)               12416
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 50,177
Trainable params: 50,177
Non-trainable params: 0
```

- **The above screen shots tell us about the structure of best LSTM Model with different number of neuron as well as parameters**

Selected Best Long Short-Term Memory Models Using HPC Resources

- **The table shows us different hyperparameters to train best LSTM model for the classification task which are:**
- **Sgd was used as an optimizer stands for "Stochastic Gradient Decent which update parameters based on the learning rate**
- **”96, 128, 1” neurons were used for input layer, dense_0 & dense_1 respectively in LSTM model**
- **Relu and Selu activation functions were used for the dense_0 & dense_1 layers**
- **70% training and 30% validation data were used**
- **Bach size (150), patience (30) and Number of Epochs (100) were used**

Optimizer	sgd
Topology	[96 128 1]
Bach size	[150]
Number of Epochs	100
Activation function	Relu, selu
Train/Validation Split	0.7
Dropout rate	0.1

Criteria to Select Best Long Short Term Memory Model by Using HPC Resources

AUC Score	**0.9699088**

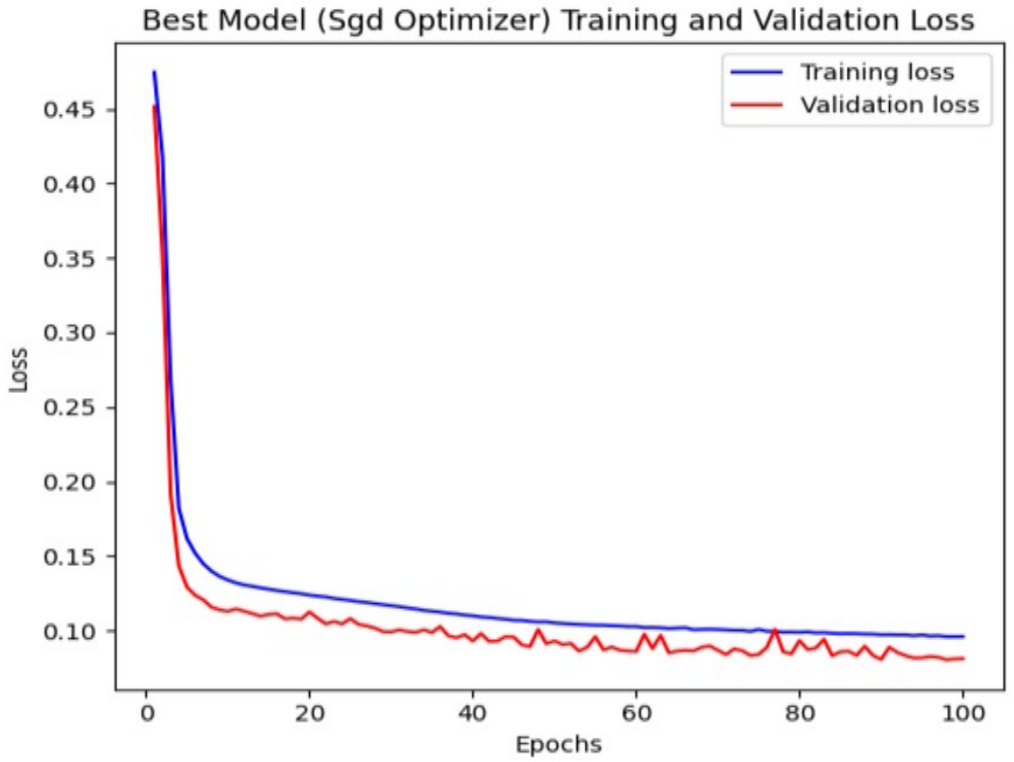
- **I selected best long short term memory (LSTM) model based on the highest Area under the curve value among all the configurations**

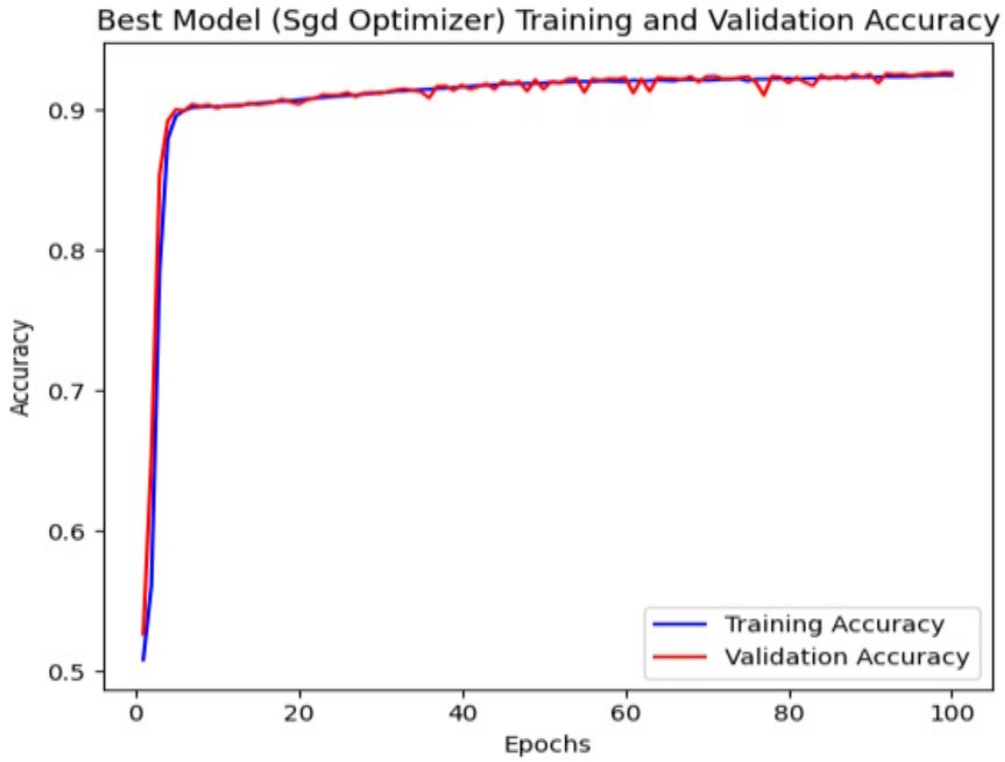
- **The above table shows us the highest value of Area under the curve to choose best LSTM model among all configurations**

Partionable Resources	Usage	Request
CPUS	1.66	4
Memory (MB)	826	5000
Run Remote Usage	14min 22sec	2hr/job

- **The above table shows us different HPC Local Resources of the RECAS like CPUS, Memory Usage and Run remote Usage**

Plots of the Peak Finding LSTM Model

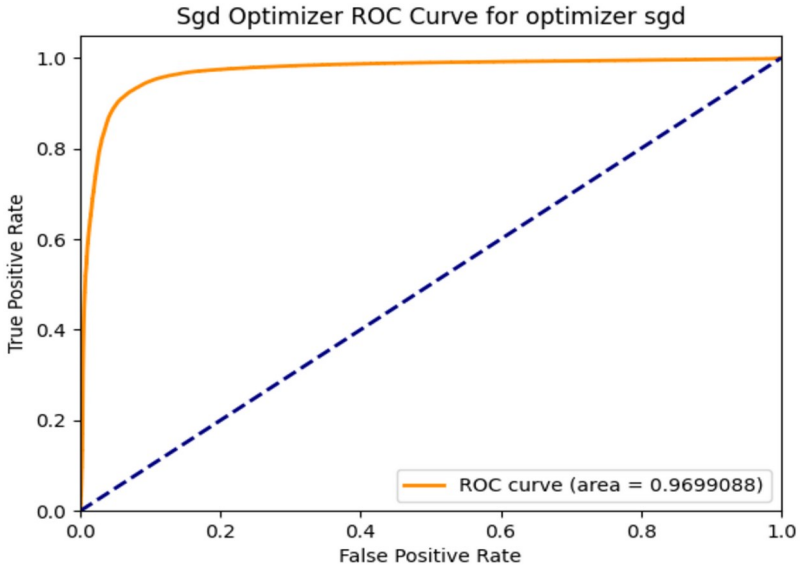




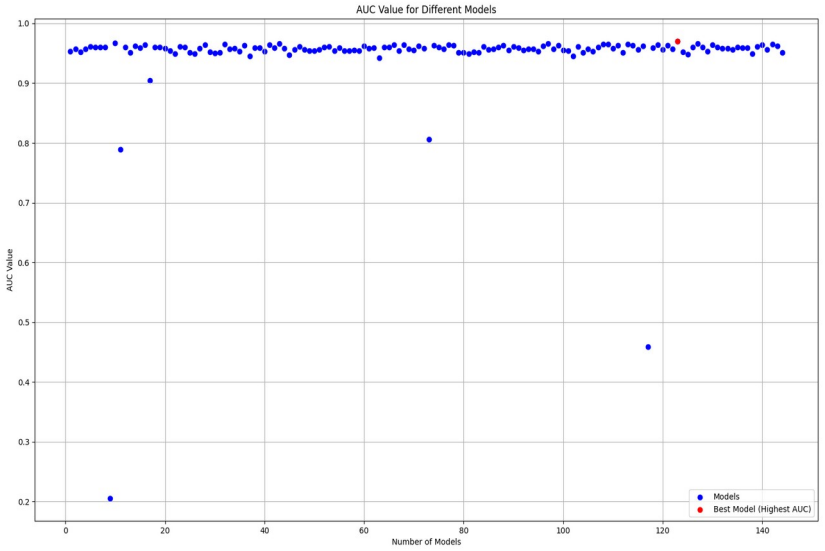
- **The upper left sided plot loss VS epoch show us that the training and validation loss decreases over the epochs and then it become approximately constant which shows a best trained model**

- **The upper right sided plots 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**

Plots of the Peak Finding LSTM Model



		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 0.9699088 with threshold value 0.5 which show a best classification to discrimate sugnal from background
- **TPR = TP/(TP+FN)**
  FPR = FP/(FP+TN)

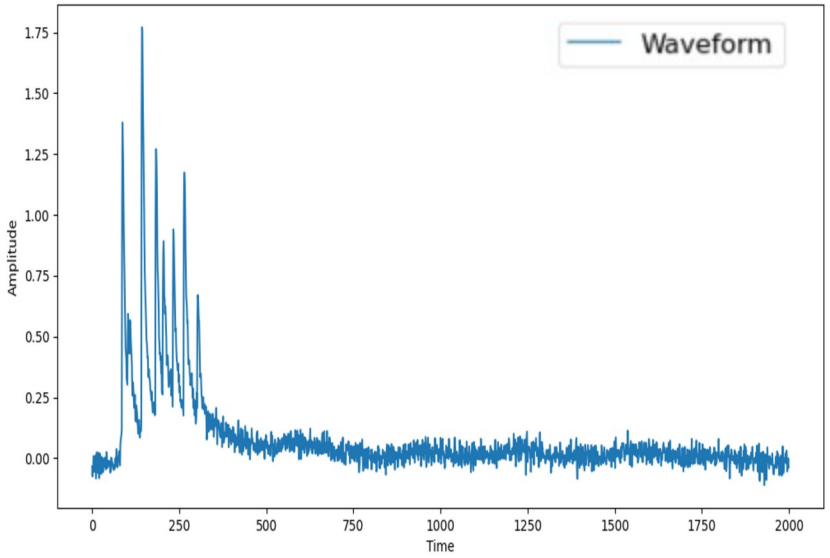
- The above table tell us about the concept of classification (TP, TN) and misclassification (FP, FN)

- **The above plot shows us different configuration models with Area under the curve value**
- **The red dot shows us the best model among all**

Applying Best LSTM Model for Peak Finding Algorithm

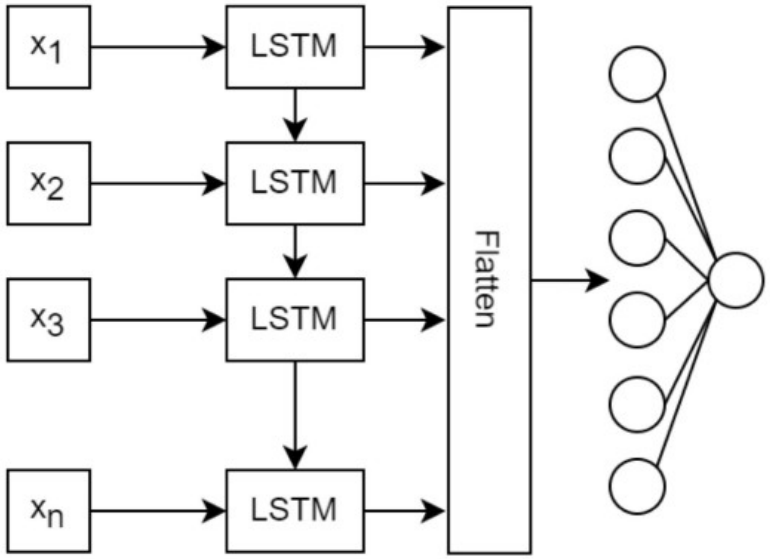
Two-Step Reconstruction Algorithm

Waveform



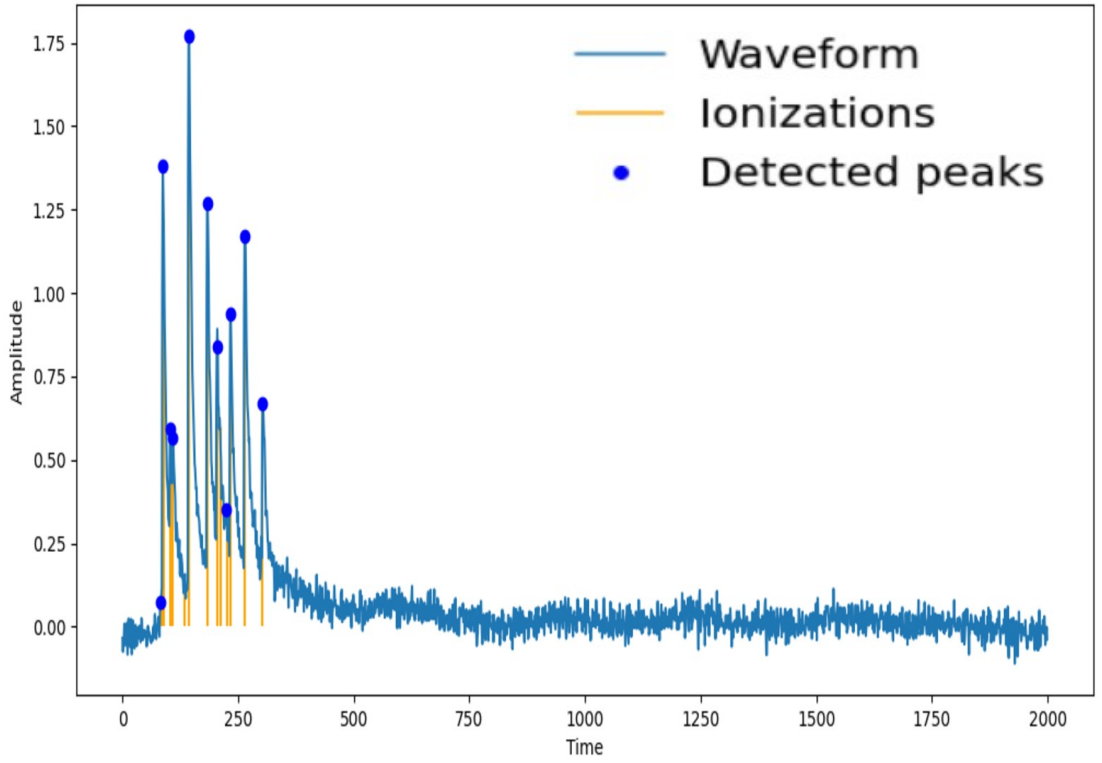
- A classification problem to classify ionization signals (Primary and Secondary Ionizations) and noises in the waveform by using Long Short Term Memory (LSTM) model

Step1: Peak Finding



- 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



- **We applied a Long Short-Term Memory (LSTM) model to the waveform to classify signals (primary ionization and secondary ionization) from the background using a peak-finding algorithm known as classification**

Second Model

Optimization of Hyperparameters for Convolutional Neural Network(CNN) Model Using HPC Resources

Optimization of Hyperparameters for Convolutional Neural Network(CNN) Model Using HPC Resources

- Currently, I designed again task involving the simultaneous submission of several jobs using local HPC Resources
- The purpose of this task is to train convolutional neural network models to detect number of primary ionization clusters based on the detected peaks, a process known as a regression task
- To achieve this task, I utilized various hyperparameters, including activation functions, optimizer Epochs, batch size, patience, and dropout rates etc
- Additionally, I managed different resources such as memory requests, Job duration, and CPU Usage etc
- Then, I selected the best model based on evaluation metric such as the mean square error (mse)

```
# Arrays defining different configurations
vminimizer=("rmsprop" "sgd" "Adam")
vneuron=("relu selu" "selu selu" "relu relu")
vpatiences=("30")
vbatches=("150")
vtopologies=("32 16" "16 32" "32 64" "8 16")
dropout=("0.1" "0.0")
vepochs=(50)
```

Different hyperparameters for CNN clusterization model are shown in the screen shot

Structure of Best CNN Model Using Different Hyperparameters

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 1021, 32)	160
max_pooling1d (MaxPooling1D)	(None, 510, 32)	0
conv1d_1 (Conv1D)	(None, 507, 64)	8256
max_pooling1d_1 (MaxPooling1	(None, 253, 64)	0
flatten (Flatten)	(None, 16192)	0
dense (Dense)	(None, 32)	518176
dense_1 (Dense)	(None, 1)	33

Total params: 526,625
Trainable params: 526,625
Non-trainable params: 0

- **The above screen shots tell us about the structure of CNN Model with different number of filters as well as parameters**

Best CNN Model Using Various Hyperparameter Optimization Techniques

- **The table shows us different hyperparameters to train CNN model for the regression task which are:**
- **Root mean square propagation(Rmsprop) was used as an optimizer**
- **[32 64] filters were used for two Convolution 1D layers respectively in CNN model**
- **Selu and Selu activation functions were used for the two Convolution 1D layers respectively**
- **70% training and 30% validation data were used**
- **Bach size (150) and Number of Epochs (50) were used**
- **[32 1] neurons were used for the two dense layers respectively**

Optimizer	Rmsprop
Number of Filters	[32 64]
Filter Size	4
Bach size	[150]
Number of Epochs	50
Activation function	Selu, Selu
Train/Validation Split	0.7
neurons	[32 1]

Criteria to Select Best CNN Model Based on the Lowest Mean Absolute Error (MAE)

Mean Square Error (MSE)	3.72629

- **I selected best CNN model based on the lowest mean square error (MSE) value among all the configuration**
- **The above table shows us the value of different evaluation metrics to choose best CNN model among all configurations**

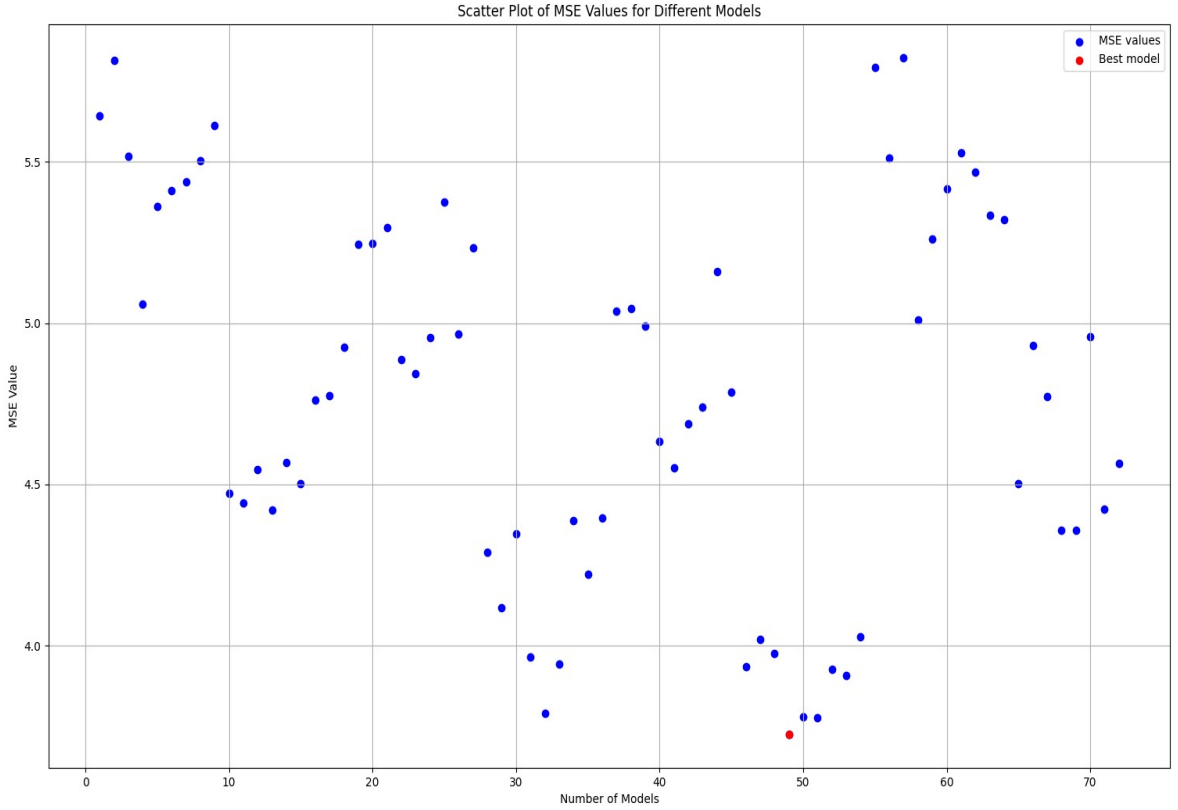
- Mean Squared Error(MSE)/ Quadratic Loss/ L2:

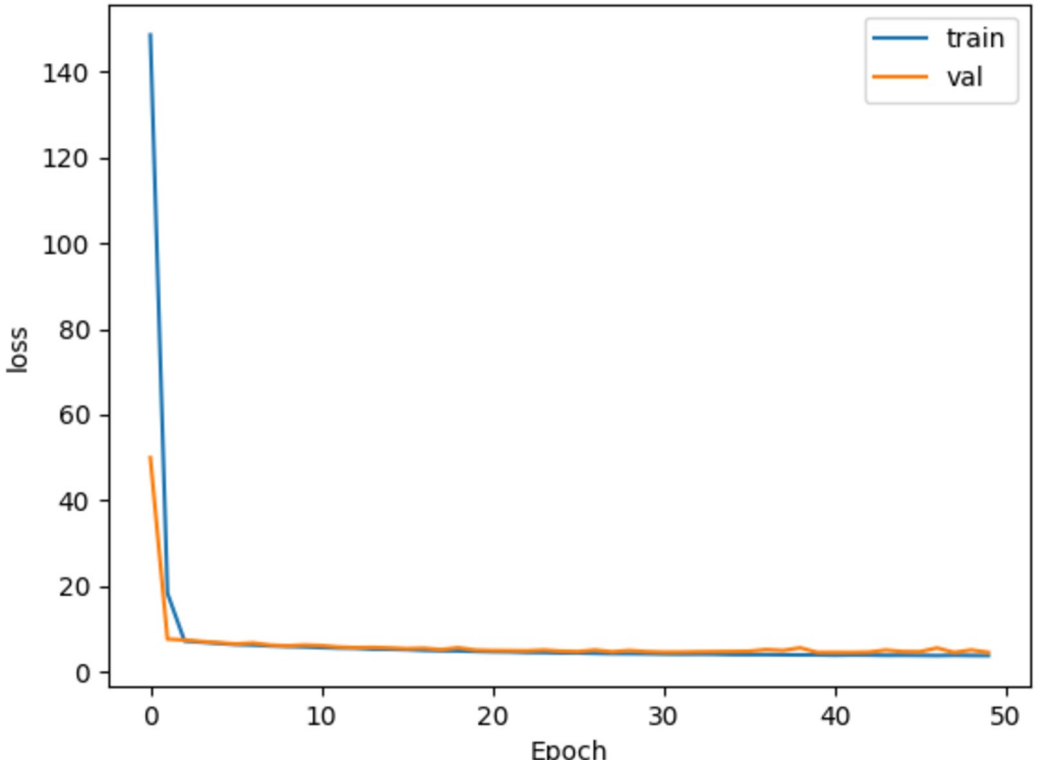
$$MSE(y^{(i)}, y_{pred}^{(i)}) = \frac{\left(y^{(i)} - y_{pred}^{(i)}\right)^2}{n}$$

Partionable Resources	Usage	Request
CPUS	1	4
Memory (MB)	135	5000
Run Remote Usage	1 min 6 sec	2hr/job

- **The above table shows us different HPC Local Resources of the RECAS like CPUS, Memory Usage and Run remote Usage**

Plots of the Best CNN Regression Model



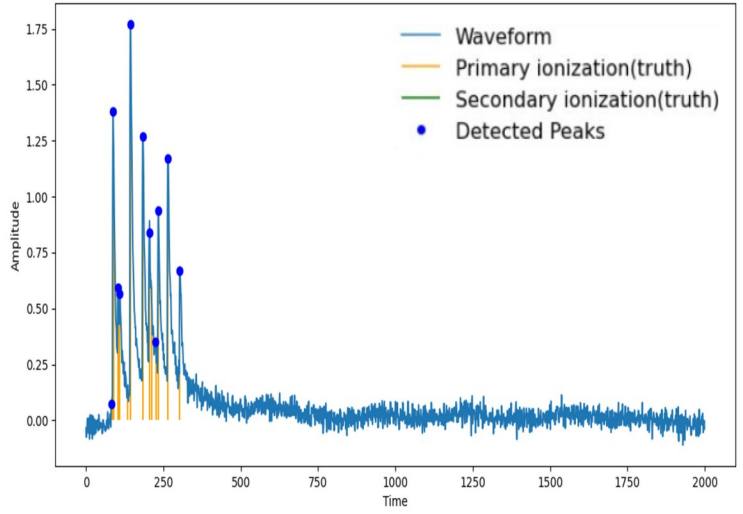


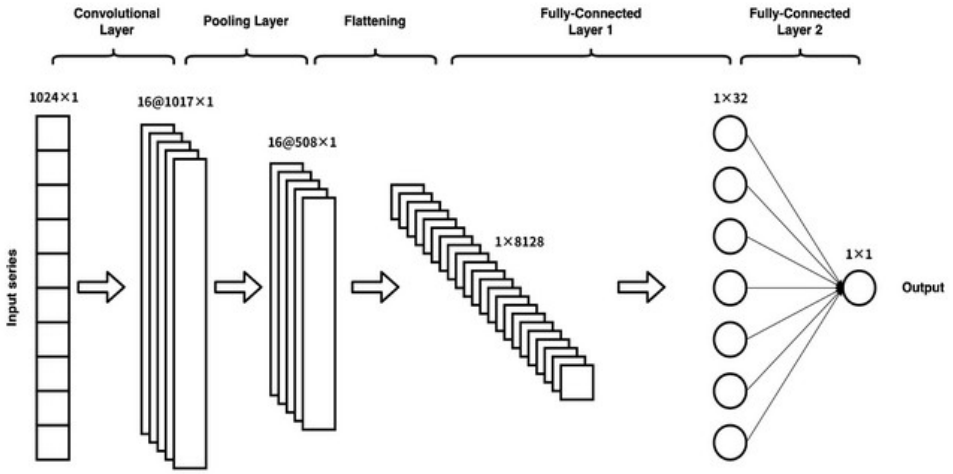
The above plot shows us different configuration models with Mean Absolute error value. The red dot shows us the best model among all

- **The plots show us that the training and validation loss mean square error decreases over the epochs and then it become constant which show us the best result**

Applying Best CNN Model for Clusterization Algorithm

Step2: Clusterization

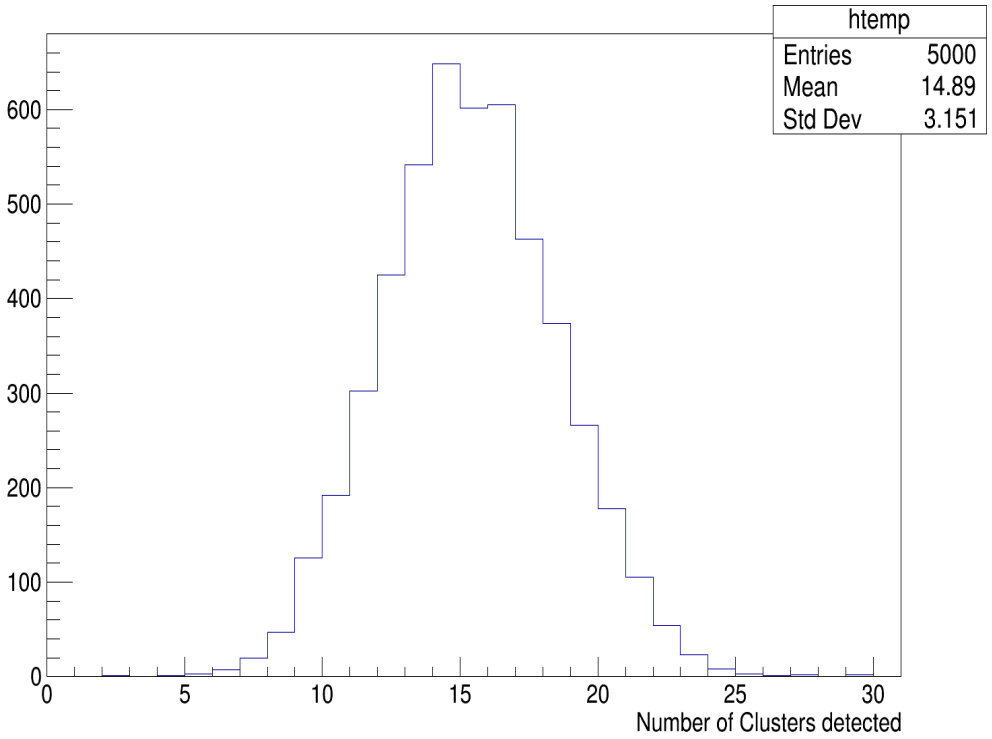


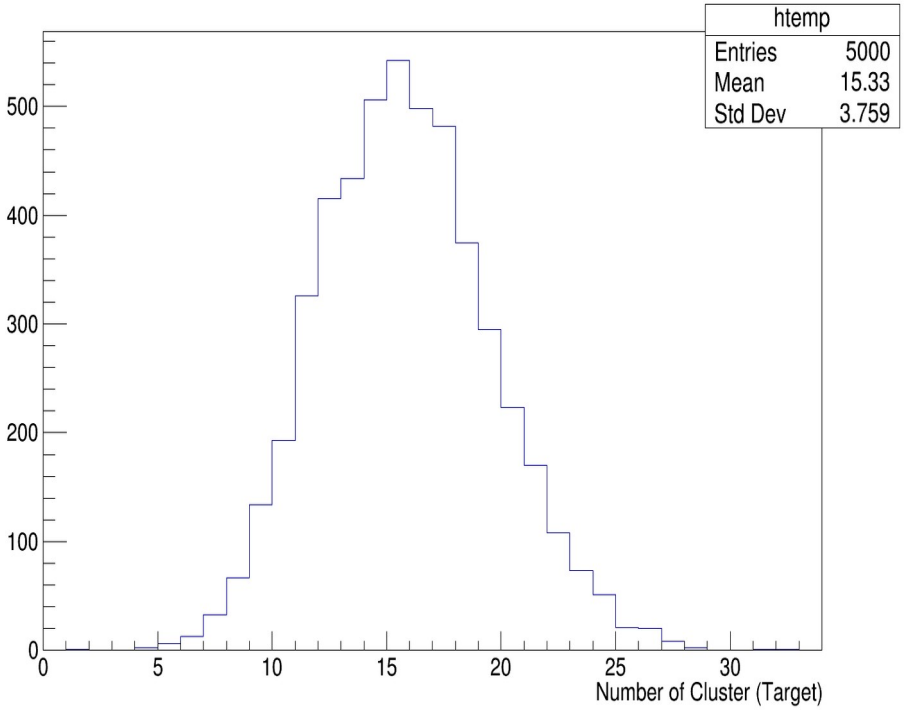


- A regression problem to predict Number of primary ionization clusters based on the primary detected peaks by using Convolutional Neural Network (CNN) model
- The peaks found by peak finding Algorithm would be training sample of this algorithm

- 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

Final Results





- **Number of Primary ionized clusters with mean value (14.89) detected by CNN Model based on the detected primary peaks with mean value (15.33)**
- **Good Gaussian distribution**

Summary

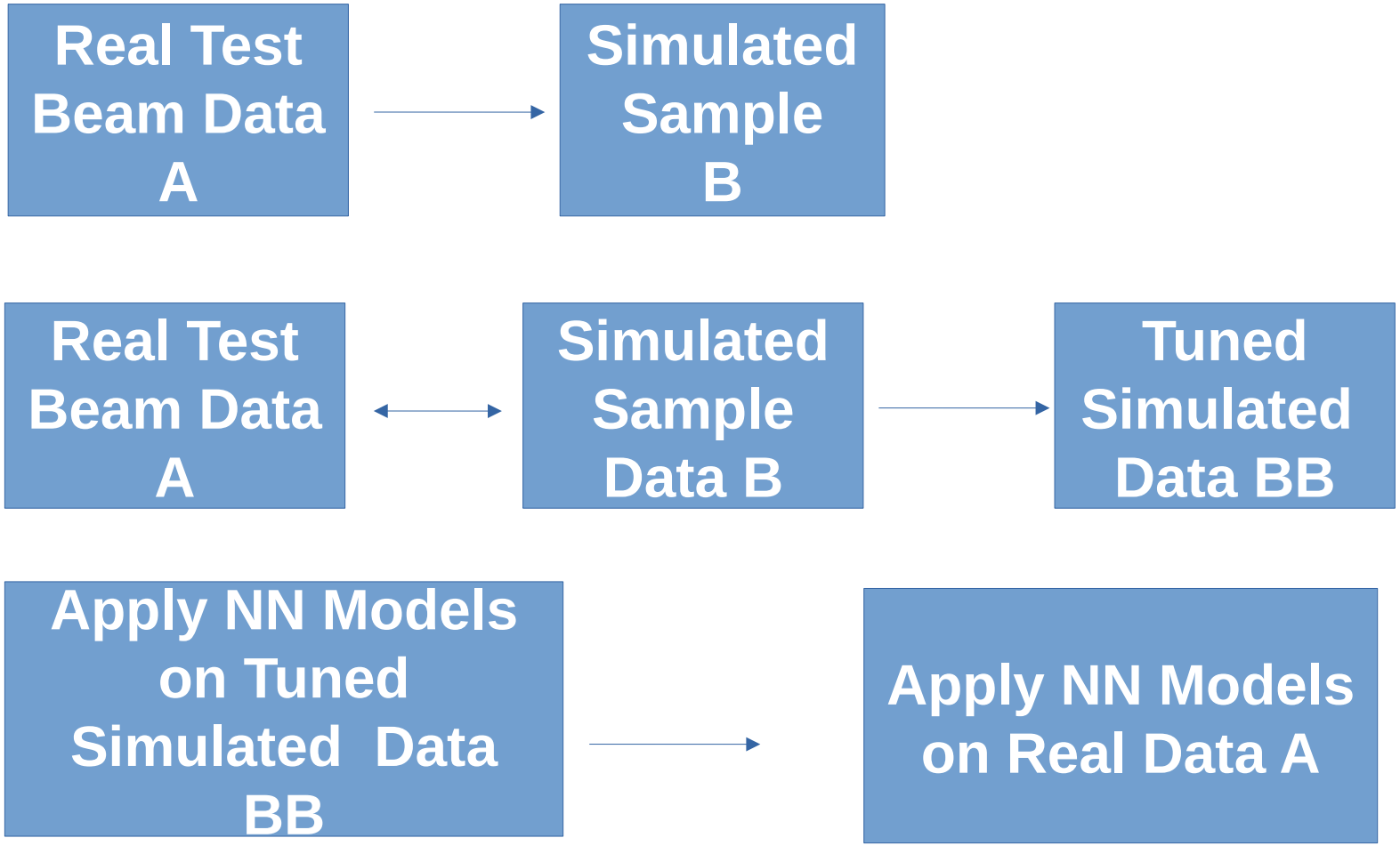
- **Particle identification (PID) is essential in most particle experiments**
- **Cluster counting (CC) in gaseous detector is the most promising breakthrough in PID due to potential of 3 times better resolution than traditional method**
- **I executed the code pertaining to the simulation of particles traversing a gas mixture made out of 90% Helium (He) and 10% Isobutane (C4H10) 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 binary cross entropy function as the loss function, scaled exponential linear unit (selu) and rectified linear unit (ReLU) as activation functions, Adaptive Moment Estimation (Adam) as the optimizer, with a batch size of 150 and 100 epochs . Then, I selected this best model based on the evaluation metric such as highest AUC (0.9699088) value among all configurations for training the LSTM model**

Summary

- **For the Clusterization algorithm, I trained convolutional Neural Network (CNN) Model by using Scale Exponential Linear Unit(SELU) as activation functions, Root mean square propagation (rmsprop) as the optimizer, with a batch size of 150 and 50 epochs . Then, I selected this best model based on the evaluation metrics such as mean square error(3.72629) value among all configurations for training the CNN model**

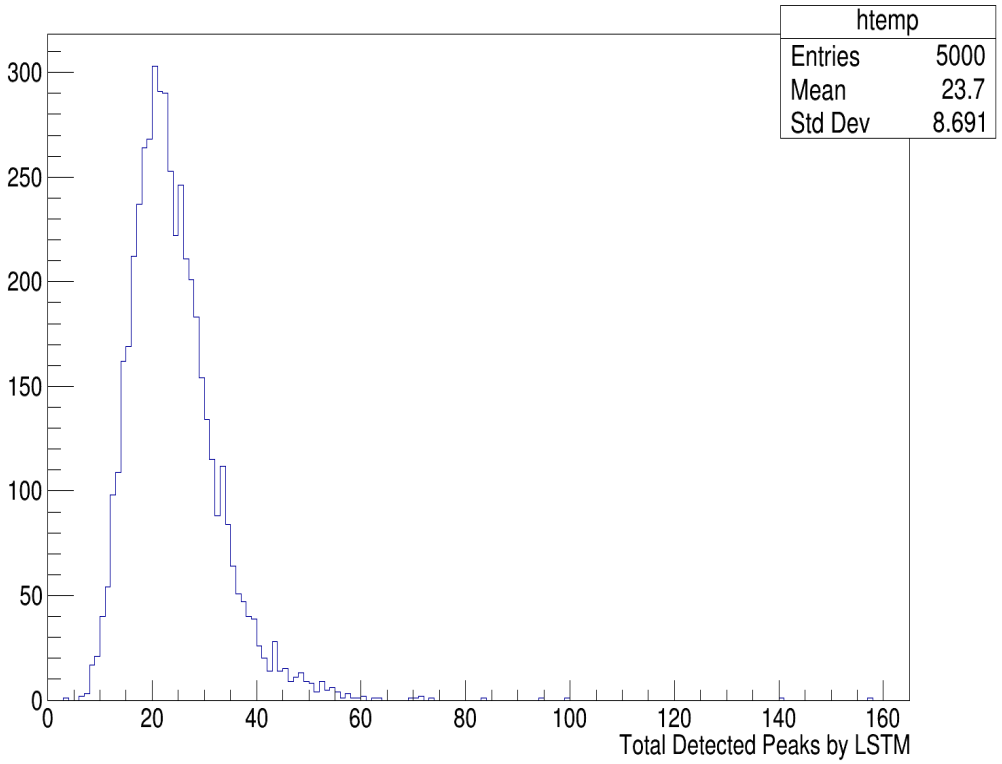
Future Planning

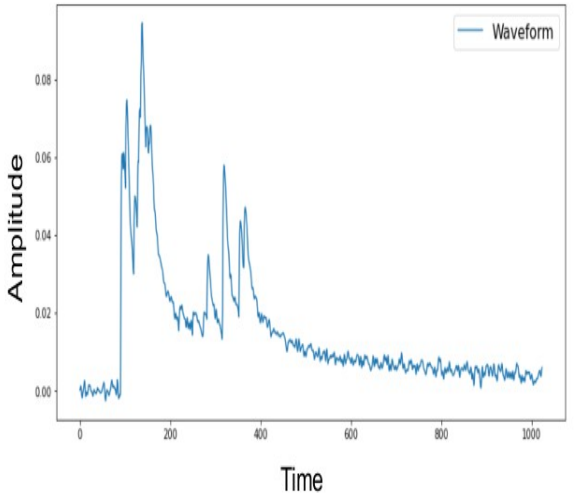
- **Now, we need real test beam data (A) as reference**
- **we would simulate waveform in simulation data (B) with the same parameters as were used in real data (A)**
- **Then we would compare real data (A) and simulated data (B)**
- **Onward, we would tune simulated data (BB) according to real data (A)**
- **After that we should Train Neural Network Models on tune simulated data (BB)**
- **At the end, we would apply Neural Network Models on the real beam test data (A)**

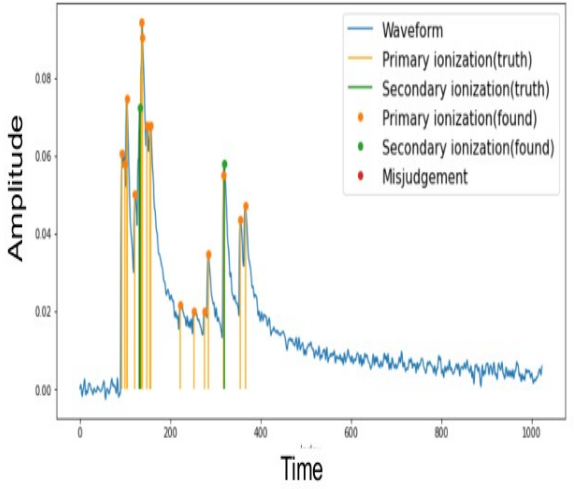


Thank you

Background Slides







Step1. Peak Finding

Discriminate peaks (both primary and secondary) from the noises (classification problem)

Step2. Clusterization:

Determine the number of clusters (N_{cls}) from the detected peaks (regression problem)

```
Processing File:  Model669/model_metrics.txt
Processing File:  Model621/model_metrics.txt
Processing File:  Model109/model_metrics.txt
Processing File:  Model143/model_metrics.txt
Processing File:  Model544/model_metrics.txt
Processing File:  Model458/model_metrics.txt
Processing File:  Model410/model_metrics.txt
Processing File:  Model492/model_metrics.txt
Processing File:  Model23/model_metrics.txt
Processing File:  Model859/model_metrics.txt
Processing File:  Model811/model_metrics.txt
Processing File:  Model333/model_metrics.txt
Processing File:  Model247/model_metrics.txt
Processing File:  Model648/model_metrics.txt
Processing File:  Model600/model_metrics.txt
Processing File:  Model682/model_metrics.txt
Processing File:  Model596/model_metrics.txt
Processing File:  Model122/model_metrics.txt
Processing File:  Model523/model_metrics.txt
Processing File:  Model437/model_metrics.txt
Processing File:  Model471/model_metrics.txt
Processing File:  Model385/model_metrics.txt
Processing File:  Model95/model_metrics.txt
Processing File:  Model838/model_metrics.txt
Processing File:  Model786/model_metrics.txt
Processing File:  Model312/model_metrics.txt
Processing File:  Model226/model_metrics.txt
Processing File:  Model260/model_metrics.txt
Processing File:  Model627/model_metrics.txt
Processing File:  Model661/model_metrics.txt
Processing File:  Model575/model_metrics.txt
Processing File:  Model149/model_metrics.txt
Processing File:  Model101/model_metrics.txt
Processing File:  Model502/model_metrics.txt
Processing File:  Model416/model_metrics.txt
Processing File:  Model498/model_metrics.txt
Processing File:  Model450/model_metrics.txt
Processing File:  Model364/model_metrics.txt
Processing File:  Model817/model_metrics.txt
Processing File:  Model765/model_metrics.txt
Processing File:  Model205/model_metrics.txt
Processing File:  Model287/model_metrics.txt
Processing File:  Model606/model_metrics.txt
Processing File:  Model688/model_metrics.txt
Processing File:  Model640/model_metrics.txt
Processing File:  Model554/model_metrics.txt
Processing File:  Model24/model_metrics.txt
```

	F1 Score	Efficiency*Purity	AUC Score	Recall	Precision	Model
0	0.6681235829212993	0.5016408236934942	0.39	1.0	0.5016408236934942	Model128
1	0.9373194296306269	0.8787180207026415	0.98	0.9252187423337968	0.9497408347847984	Model529
2	0.938201264976603	0.8802547963116932	0.98	0.9324760814457437	0.9439971853722138	Model477
3	0.9325780341978785	0.8702193176503717	0.98	0.9103769727696459	0.9558889819048232	Model343
4	0.9412132192165926	0.8861901537795372	0.98	0.9239921498078338	0.959088401307134	Model99
..	...	...	...	...	...	...
849	0.6681235829212993	0.5016408236934942	0.53	1.0	0.5016408236934942	Model606
850	0.6681235829212993	0.5016408236934942	0.22	1.0	0.5016408236934942	Model688
851	0.6681235829212993	0.5016408236934942	0.52	1.0	0.5016408236934942	Model640
852	0.9303548494746529	0.8656586770118148	0.98	0.9403876032382043	0.9205339096676072	Model554
853	0.6681235829212993	0.5016408236934942	0.80	1.0	0.5016408236934942	Model24

[854 rows x 6 columns]

Meaning of different Hyperparameters

- **SGD (Stochastic Gradient Descent): SGD is an optimization algorithm used to minimize a function by iteratively moving towards the minimum value in small steps.**
- **Batch Size: Batch size refers to the number of training samples used to train a model in one iteration.**
- **Dropout: Dropout is a regularization technique used to prevent overfitting in neural networks. It works by randomly setting a fraction of the input units to 0 at each update during training time, which helps to make the model robust by preventing it from relying too heavily on any one feature.**
- **Dropout Rate: The dropout rate is the fraction of the input units in a neural network that are set to zero during training.**
- **Epochs: An epoch is a single pass through the entire training dataset**
- **Activation Function: An activation function in a neural network is a mathematical operation that is applied to the input coming into a neuron. It decides whether a neuron should be activated or not, helping to add non-linearity to the model, which allows it to learn more complex patterns.**
- **Patience: Patience is a hyperparameter often used in conjunction with early stopping during training.**

Meaning of different Hyperparameters

- **Adam Optimizer:**

- **Adam stands for "Adaptive Moment Estimation." This optimizer combines two other optimizers—RMSprop and SGD with momentum. It keeps track of an exponentially decaying average of past gradients (similar to momentum) and an exponentially decaying average of past squared gradients (similar to RMSprop).**

- **Essentially, Adam adjusts the learning rate for each parameter, combining the benefits of having momentum (which helps to propel the optimizer towards the right direction and smooth out updates) and scaling the gradient by the square root of the recent average of squared gradients. This helps Adam adapt its learning rates based on the properties of data and make it effective and stable in practice..**

- **RMSprop Optimizer:**

- **RMSprop stands for "Root Mean Square Propagation." It was developed to resolve issues where the learning rates either diminish too quickly or too slowly. This optimizer adjusts the learning rate for each parameter by dividing the gradient by a running average of its recent magnitude.**

- **RMSprop makes adjustments to the learning rate during the training process. It does this by keeping track of the average of past squared gradients, and using this average to scale the gradient. This helps in a more balanced and faster convergence, as it prevents the learning rate from becoming too large or too small for certain weights in the network.**

Meaning of Evaluation metrics

- **Precision: Precision is a metric that measures the accuracy of the positive predictions made by a model. In other words, it is the ratio of correctly predicted positive observations to the total predicted positives. It answers the question: "Of all the items labeled as positive, how many actually belong to the positive class?"**

- **Recall: Recall (also known as sensitivity) is the metric that measures the ability of a model to find all the relevant cases within a dataset. It is the ratio of correctly predicted positive observations to all observations in the actual positive class. It answers the question: "Of all the actual positives, how many were identified correctly?**

- **F1 Score: The F1 Score is the harmonic mean of precision and recall. It is a way to combine both precision and recall into a single measure that captures both properties. This score is particularly useful when you need to balance precision and recall and there is an uneven class distribution (large number of actual negatives).**

- **AUC Score (Area Under the Curve): The AUC score is used with the ROC curve (Receiver Operating Characteristic curve), which plots the true positive rate (recall) against the false positive rate at various threshold settings. The AUC score represents the degree or measure of separability achieved by the model. It tells how much the model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.**

Meaning of Evaluation metrics

- **Efficiency*Purity: This is a product of two metrics, Efficiency and Purity, which is often used in fields like particle physics but can also be applied in other areas of classification.**

- **Efficiency: It is similar to recall. It measures the proportion of actual positives that are correctly identified.**

- **Purity: This metric is similar to precision. It measures the proportion of positive identifications that were actually correct.**

- **When combined (Efficiency*Purity), this product provides a single measure that reflects how many of the selections were correct (purity) and how many of the correct cases were selected (efficiency). This can be particularly useful in scenarios where it's crucial not only to identify positive cases accurately but also to cover as many of them as possible without introducing too many errors.**

- **The acronym "CPU" stands for Central Processing Unit. It is the primary component of a computer that performs most of the processing inside. A CPU executes instructions from a computer program by performing the basic arithmetic, logical, control, and input/output (I/O) operations specified by the instructions.**

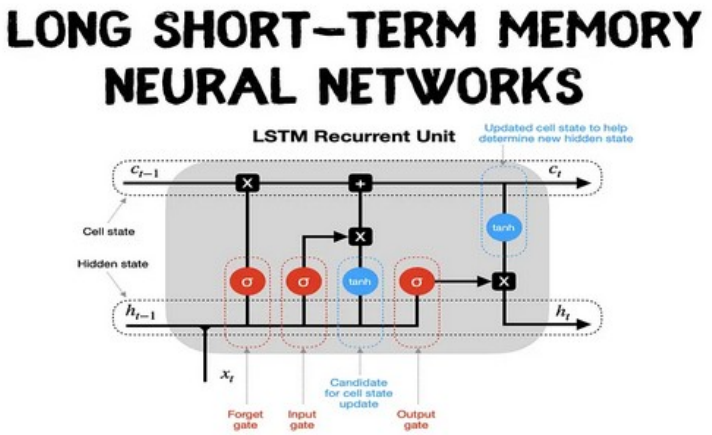
- **When you mention using CPU from RECAS resources, it suggests that you are utilizing CPU computing power provided by the RECAS (REsources for Cloud federated Access Services) project or a similar computing resource. RECAS typically offers infrastructure and computing resources to support scientific research, where CPUs are used to process tasks, run simulations, or analyze data. These resources are often shared or distributed across a network, allowing users to leverage powerful computational capabilities remotely.**

ACCURACY and LSTM

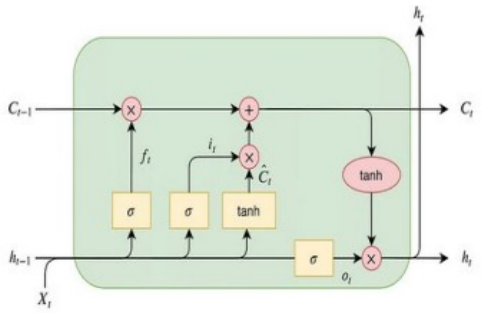
- The accuracy is defined as the ratio between the number of correct predictions to the total number of predictions
- Accuracy values range between 0 and 1. Obviously an accuracy values near to 1 means that our model fits well the datasets

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

- 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 (LSTM)



Forget Gate

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

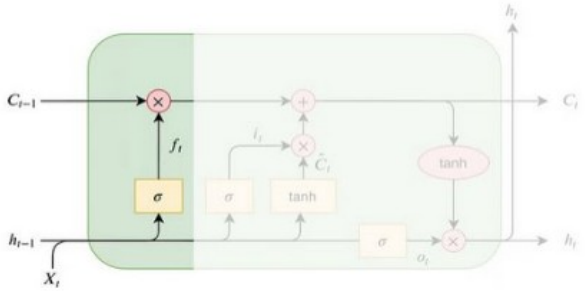
$$i_t = \sigma(W_i \cdot [h_{t-1}, X_t] + b_i)$$

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

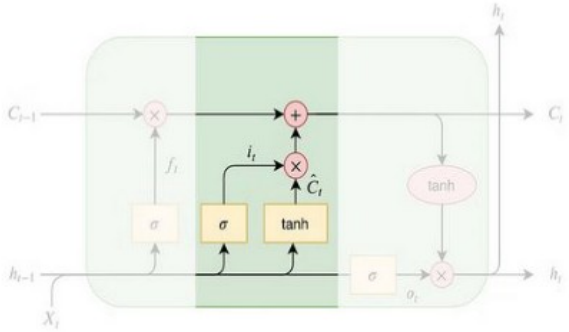
$$\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, X_t] + b_C)$$

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

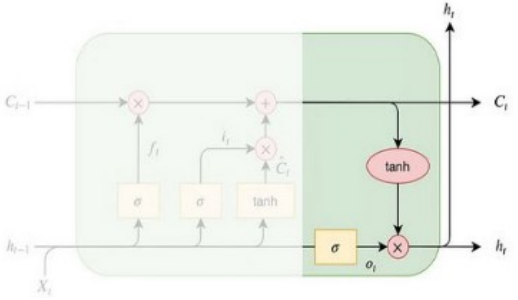
state and the new input data.



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):

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

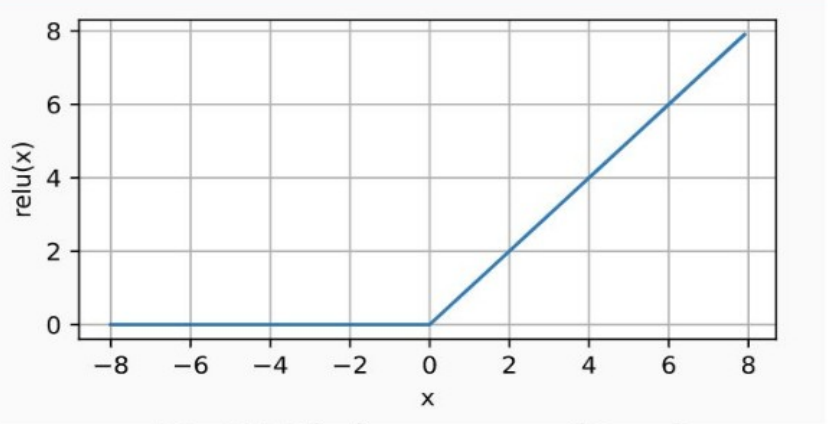
NUMBER OF EPOCHS

- **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.

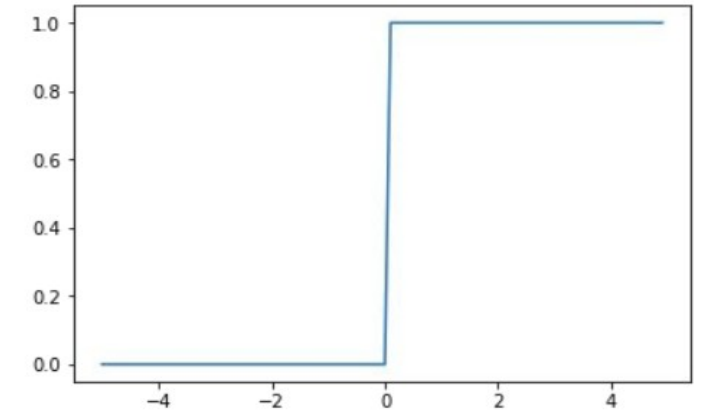
RECTIFIED LINEAR UNIT (RELU)

- 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:

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



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



SCALED EXPONENTIAL LINEAR UNIT (SELU)

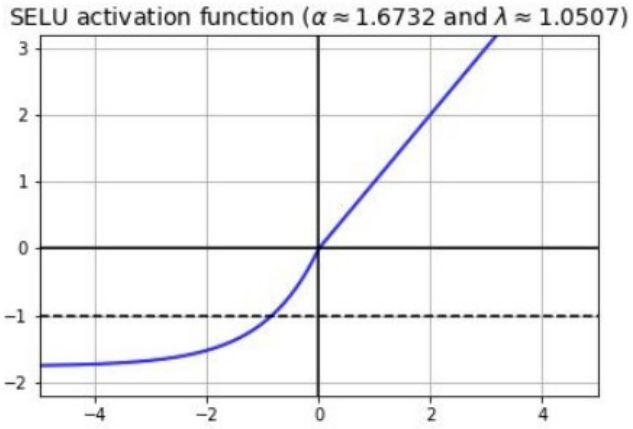
- Another choice is the Scaled Exponential Linear Unit (SELU)

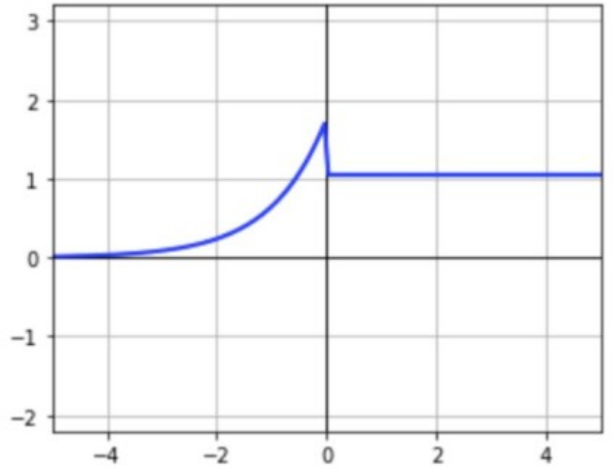
- The functions depends on two parameters and the equation is the following:

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

- The function is not differentiable in zero

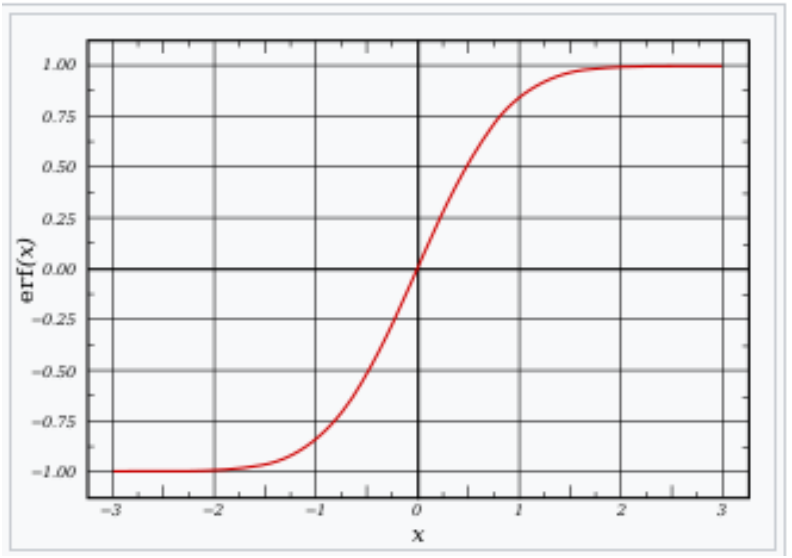
$$\frac{dSELU(x)}{dx} = \lambda \begin{cases} \alpha e^x & x \leq 0 \\ 1 & x > 0 \end{cases}$$



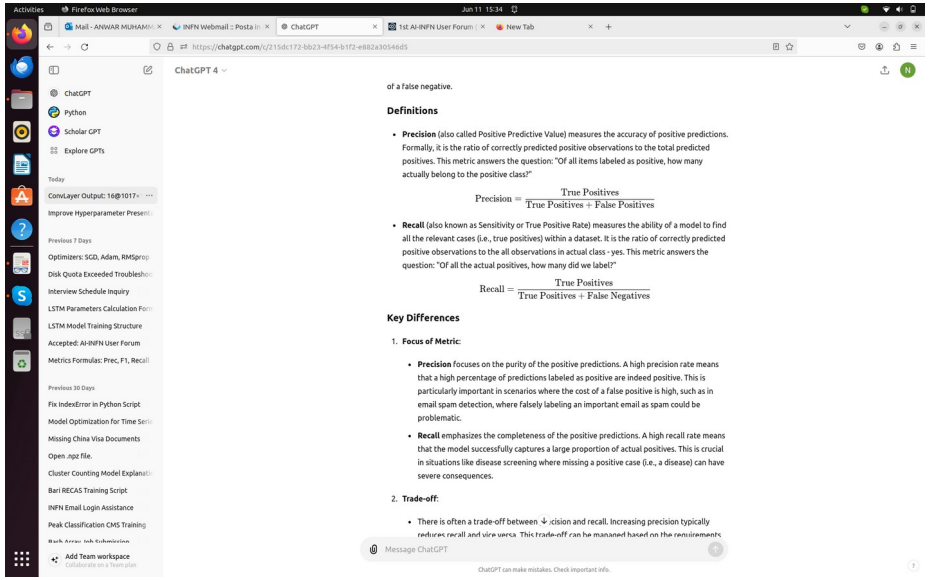


Sigmoid Function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



- **The sigmoid function outputs a value between 0 and 1, making it especially useful for models where you need to predict probabilities that vary between these two limits. The function is S-shaped, providing a smooth gradient as xxx increases or decreases. This characteristic is particularly beneficial during the optimization phase of training a model, as it provides a clear path toward minimizing the loss**.



of a false negative.

Definitions

- **Precision** (also called Positive Predictive Value) measures the accuracy of positive predictions. Formally, it is the ratio of correctly predicted positive observations to the total predicted positives. This metric answers the question: "Of all items labeled as positive, how many actually belong to the positive class?"

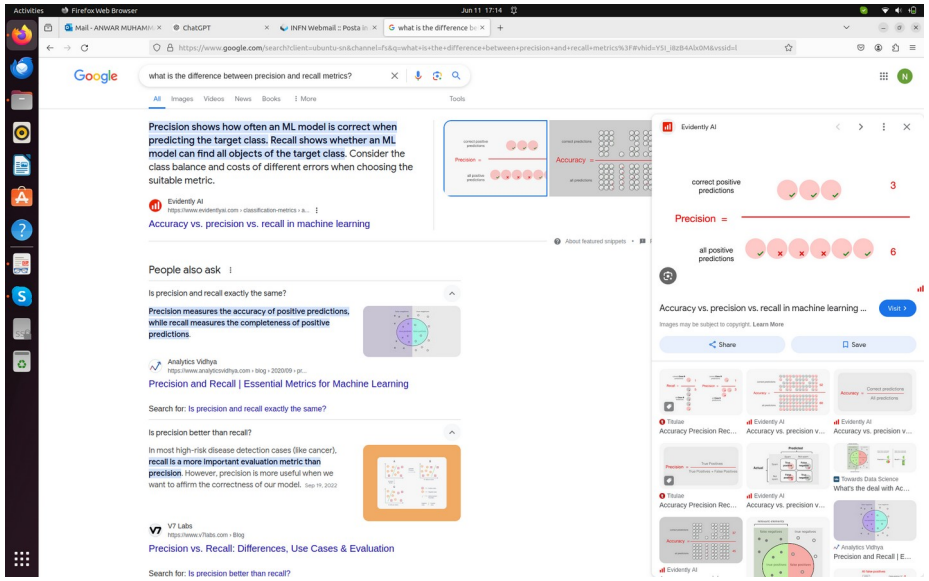
$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall** (also known as Sensitivity or True Positive Rate) measures the ability of a model to find all the relevant cases (i.e., true positives) within a dataset. It is the ratio of correctly predicted positive observations to the all observations in actual class - yes. This metric answers the question: "Of all the actual positives, how many did we label?"

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Key Differences

1. **Focus of Metric**:
   - **Precision** focuses on the purity of the positive predictions. A high precision rate means that a high percentage of predictions labeled as positive are indeed positive. This is particularly important in scenarios where the cost of a false positive is high, such as in email spam detection, where falsely labeling an important email as spam could be problematic.
   - **Recall** emphasizes the completeness of the positive predictions. A high recall rate means that the model successfully captures a large proportion of actual positives. This is crucial in situations like disease screening where missing a positive case (i.e., a disease) can have severe consequences.
2. **Trade-off**:
   - There is often a trade-off between ↓ cision and recall. Increasing precision typically



Precision shows how often an ML model is correct when predicting the target class. Recall shows whether an ML model can find all objects of the target class. Consider the class balance and costs of different errors when choosing the suitable metric.

Accuracy vs. precision vs. recall in machine learning

People also ask

Is precision and recall exactly the same?

Precision measures the accuracy of positive predictions, while recall measures the completeness of positive predictions.

Precision and Recall | Essential Metrics for Machine Learning

Is precision better than recall?

In most high-risk disease detection cases (like cancer), recall is a more important evaluation metric than precision. However, precision is more useful when we want to affirm the correctness of our model.

Precision vs. Recall: Differences, Use Cases & Evaluation