Physics Data Analysis with Neural Networks

Daniel Lersch

July 6, 2021

Overview

- This presentation and jupyter notebooks are available at:

  http://hadron.physics.fsu.edu/~dlersch/GlueXPID_with_Networks/

- Content:

1.) Introduction to neural networks

2.) Implementation in physics data analysis:
   - Scikit
   - Tensorflow / Keras

3.) Performance evaluation

4.) Hyper parameter optimization (optional)

Search for $\eta \to e^+e^-\gamma$ in GlueX

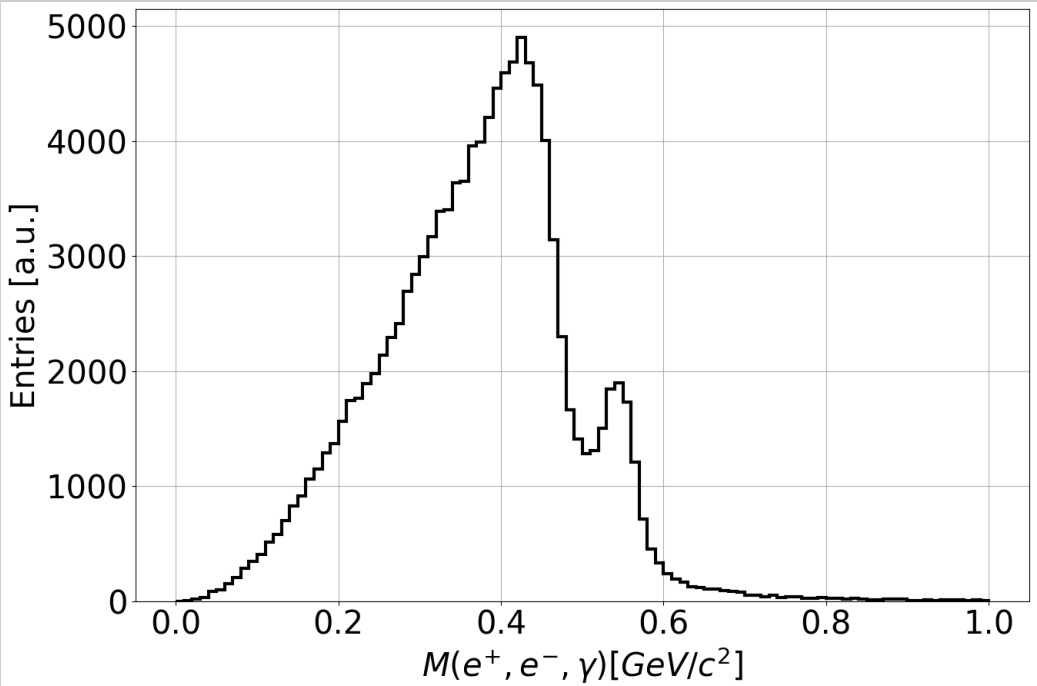
- Given a (toy) data set:
  - i) $\eta \to e^+e^-\gamma$
  - ii) $\eta \to \pi^+\pi^-\gamma$ ($6\times$ more likely than i))

Search for $\eta \to e^+e^-\gamma$ in GlueX

- Given a (toy) data set:
  - i) $\eta \to e^+e^-\gamma$
  - ii) $\eta \to \pi^+\pi^-\gamma$ ($6\times$ more likely than i))
- **Task:** Reconstruct the reaction: $\eta \to e^+e^-\gamma$

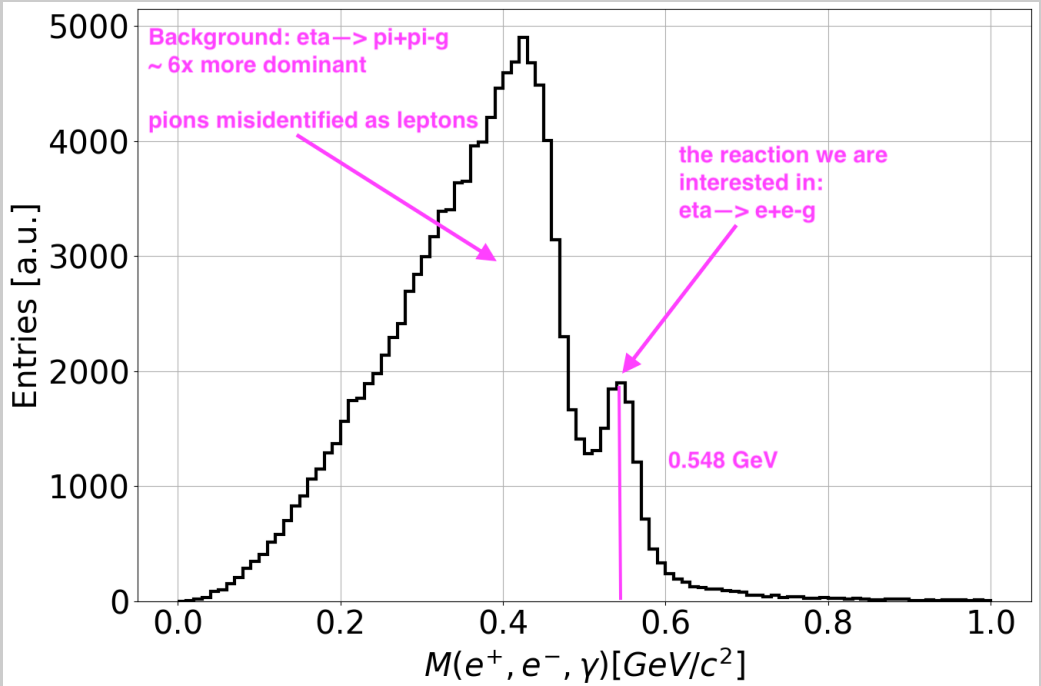
Search for $\eta \to e^+e^-\gamma$ in GlueX

- Given a (toy) data set:
  - i) $\eta \to e^+e^-\gamma$
  - ii) $\eta \to \pi^+\pi^-\gamma$ (6$\times$ more likely than i))
- **Task:** Reconstruct the reaction: $\eta \to e^+e^-\gamma$



Search for $\eta \to e^+e^-\gamma$ in GlueX

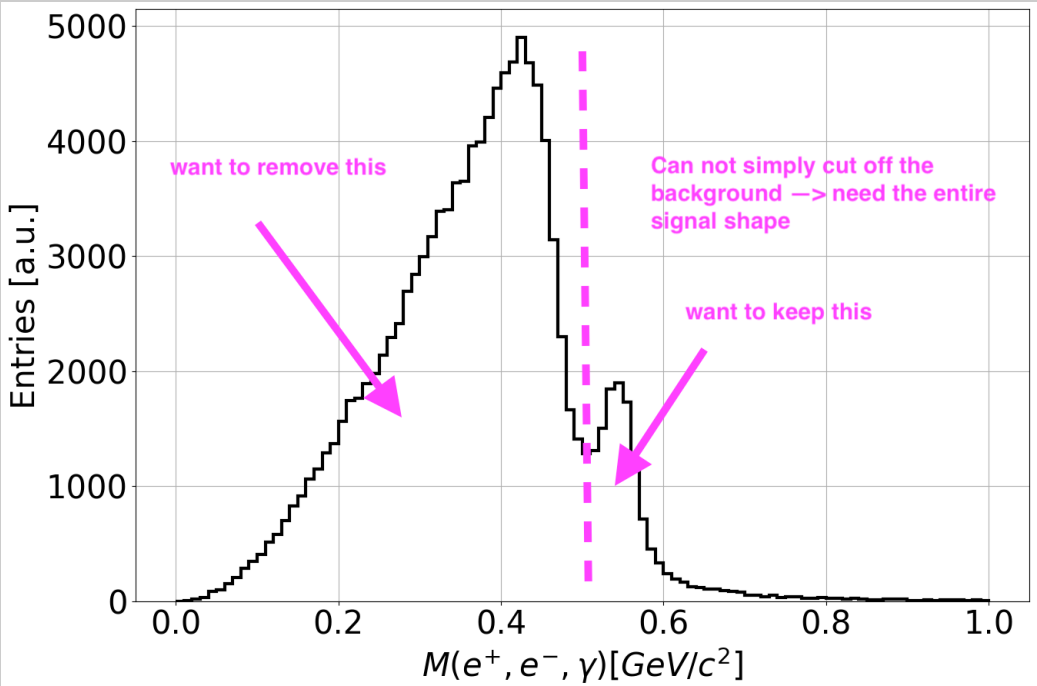
- Given a (toy) data set:
  - i) $\eta \to e^+e^-\gamma$
  - ii) $\eta \to \pi^+\pi^-\gamma$ (6× more likely than i))
- **Task:** Reconstruct the reaction: $\eta \to e^+e^-\gamma$
  i.e. $M(e^+, e^-, \gamma) = m_\eta = 0.548\,\mathrm{GeV/c^2}$



- **Problem:** Pions are misidentified as electrons: $\eta \to \pi^+\pi^-\gamma$ treated as $\eta \to e^+e^-\gamma$

Search for $\eta \to e^+e^-\gamma$ in GlueX

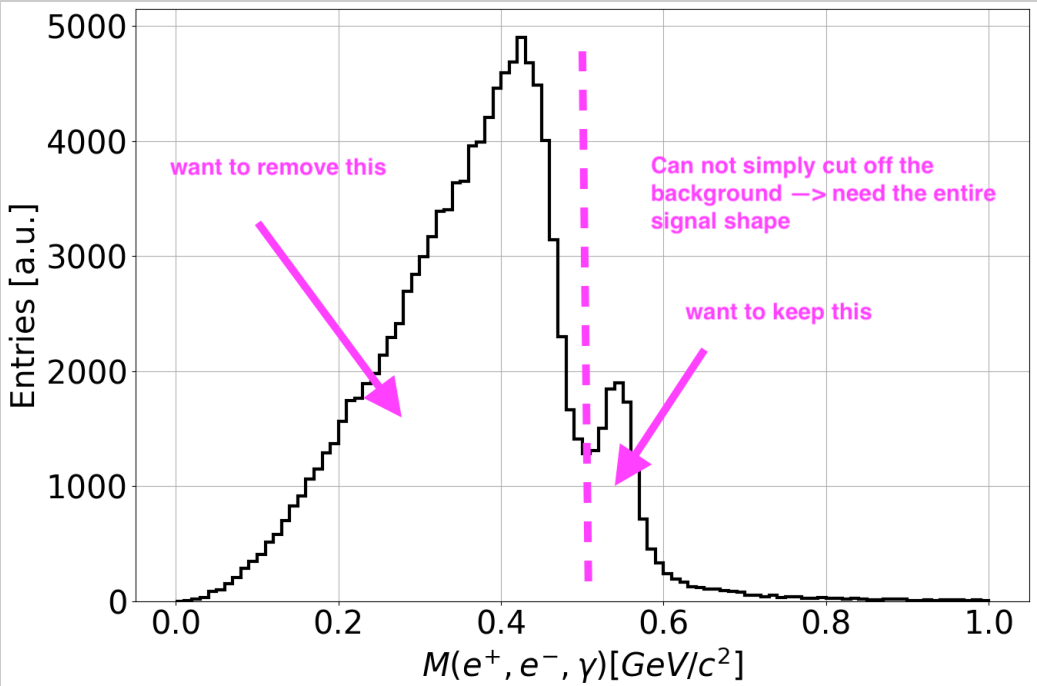
- Given a (toy) data set:
  - i) $\eta \to e^+e^-\gamma$
  - ii) $\eta \to \pi^+\pi^-\gamma$ (6× more likely than i))
- **Task:** Reconstruct the reaction: $\eta \to e^+e^-\gamma$
  i.e. $M(e^+, e^-, \gamma) = m_\eta = 0.548\,\mathrm{GeV/c^2}$



- **Problem:** Pions are misidentified as electrons: $\eta \to \pi^+\pi^-\gamma$ treated as $\eta \to e^+e^-\gamma$
- **Constraint:** Have to preserve the $M(e^+, e^-, \gamma)$-signal shape

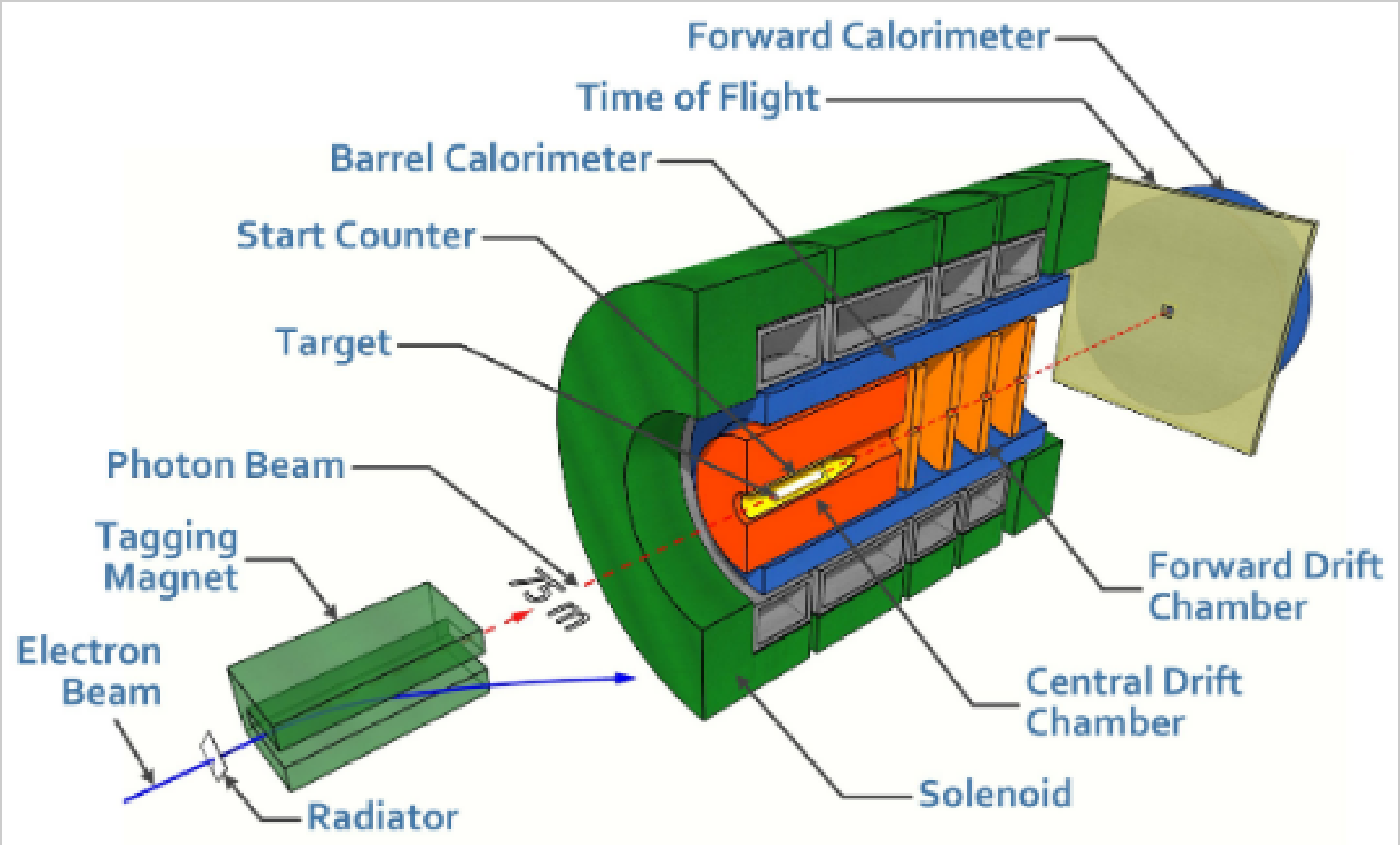
Search for $\eta \to e^+e^-\gamma$ in GlueX

- Given a (toy) data set:
  - i) $\eta \to e^+e^-\gamma$
  - ii) $\eta \to \pi^+\pi^-\gamma$ (6× more likely than i))
- **Task:** Reconstruct the reaction: $\eta \to e^+e^-\gamma$
  i.e. $M(e^+, e^-, \gamma) = m_\eta = 0.548\,\mathrm{GeV/c^2}$



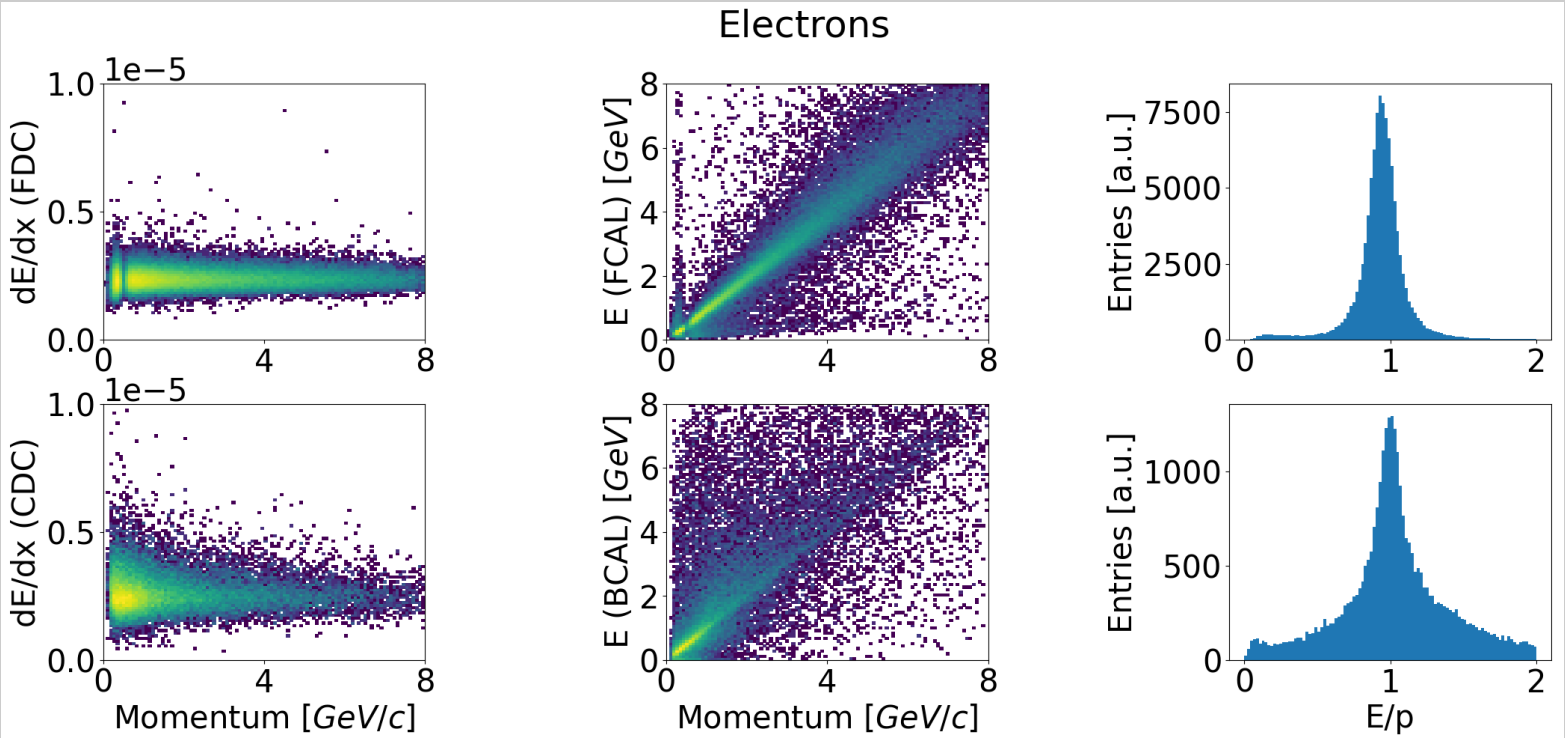
- **Problem:** Pions are misidentified as electrons: $\eta \to \pi^+\pi^-\gamma$ treated as $\eta \to e^+e^-\gamma$
- **Constraint:** Have to preserve the $M(e^+, e^-, \gamma)$-signal shape
- **Approach:** Use neural networks to suppress $\eta \to \pi^+\pi^-\gamma$ background

The Gluonic Excitation Experiment (GlueX)



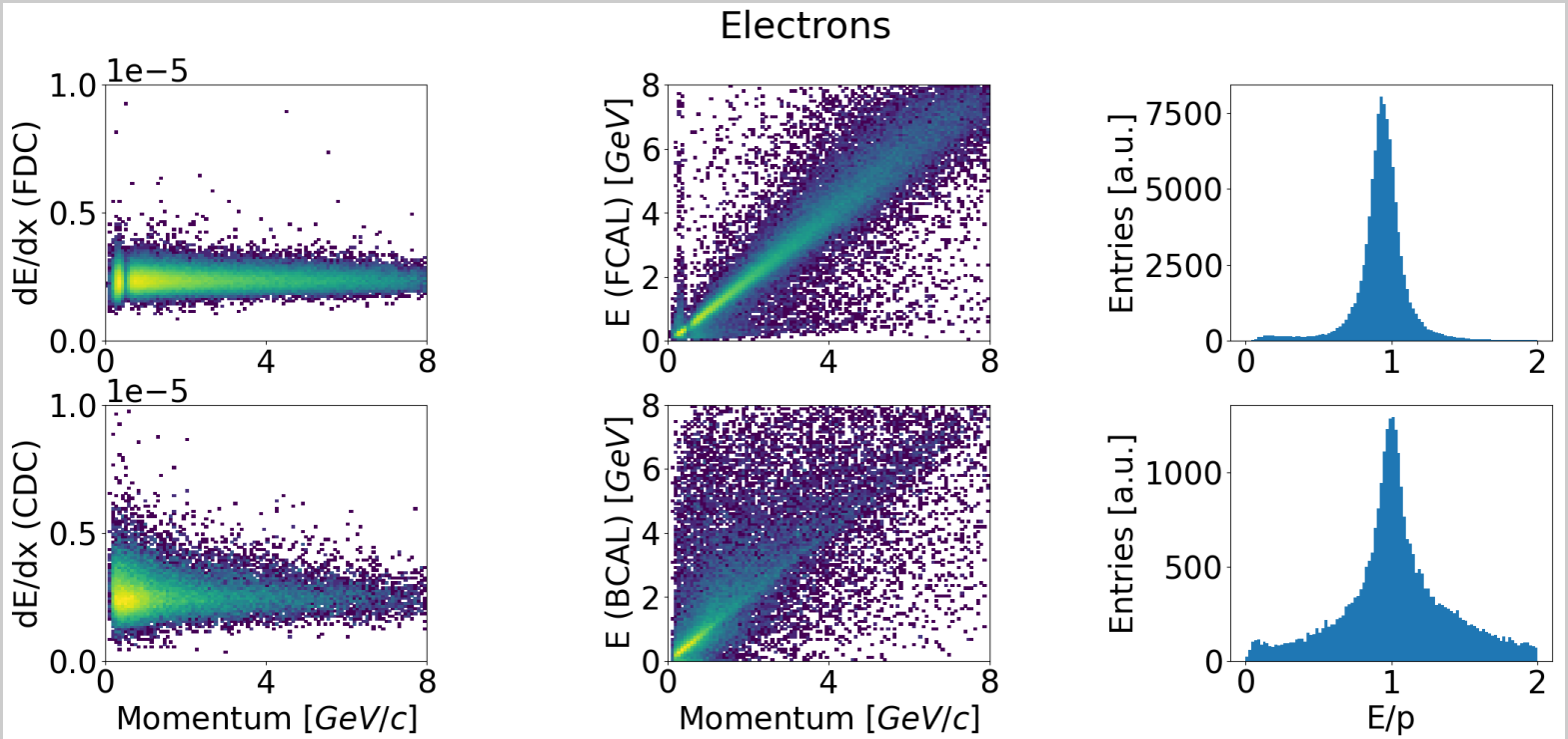
Understanding the Data

- Look at simulations



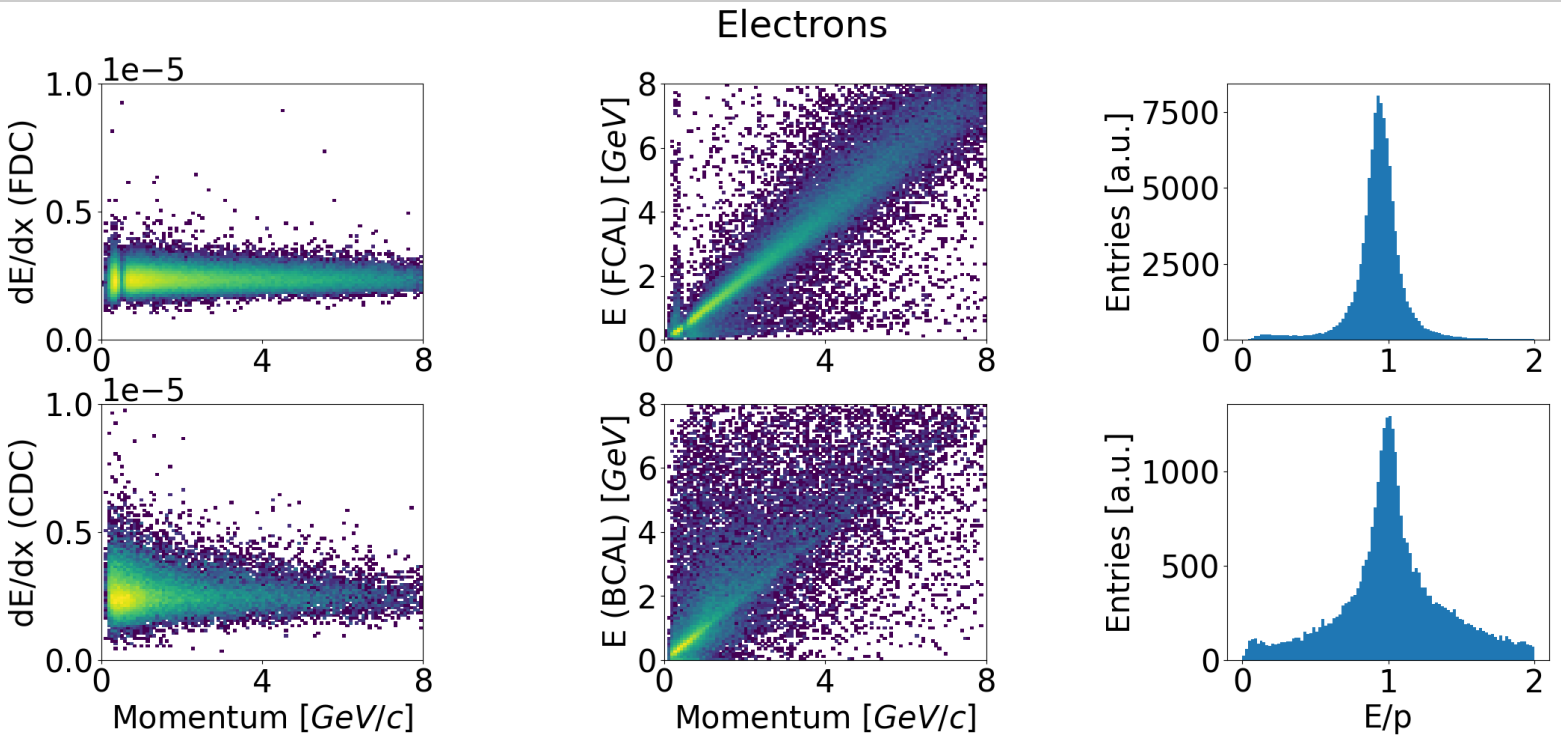
Understanding the Data

- Look at simulations
- Number of electrons = Number of pions



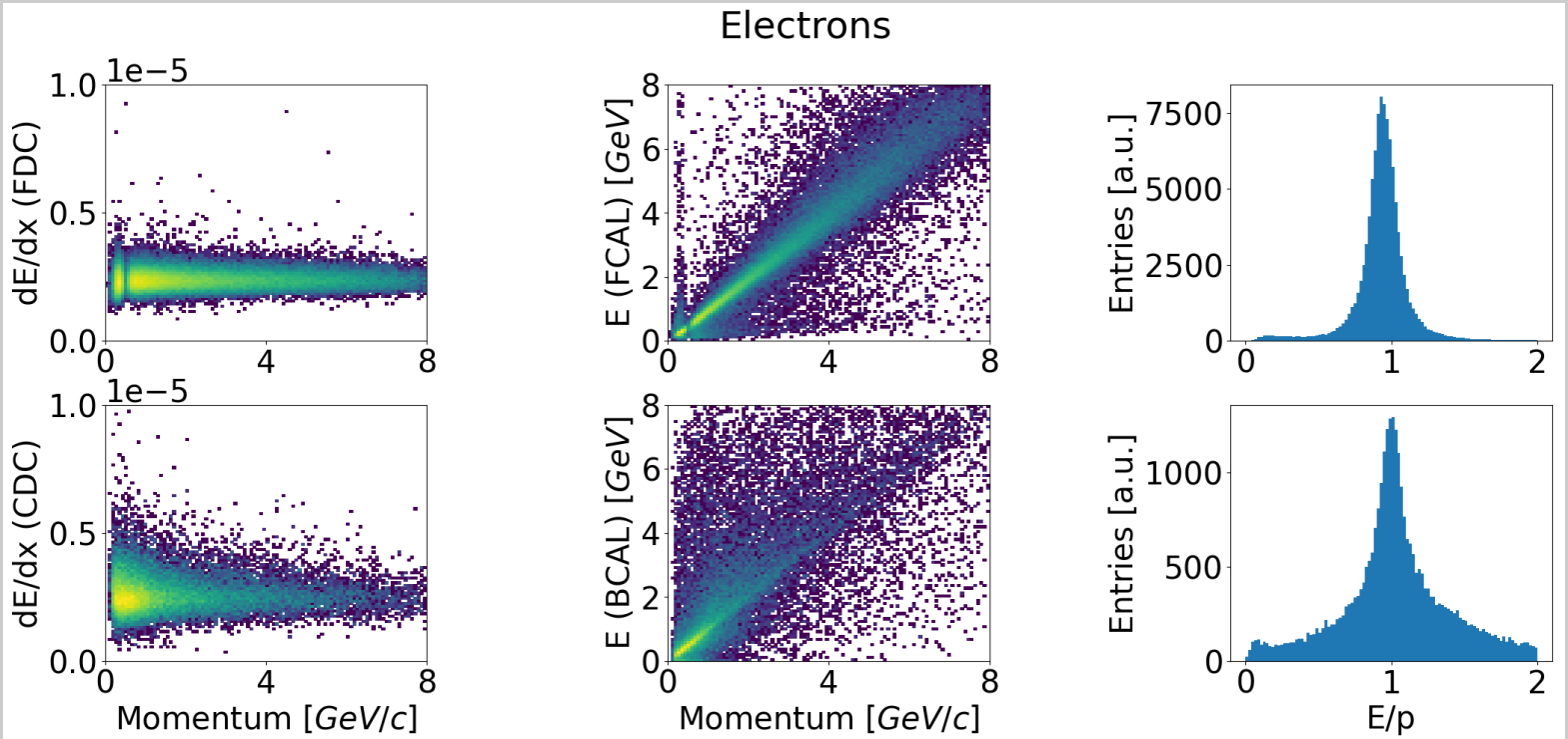
Understanding the Data

- Look at simulations
- Number of electrons = Number of pions
- Events are labeled: $e^{\pm}$: 1 and $\pi^{\pm}$: 0



Understanding the Data

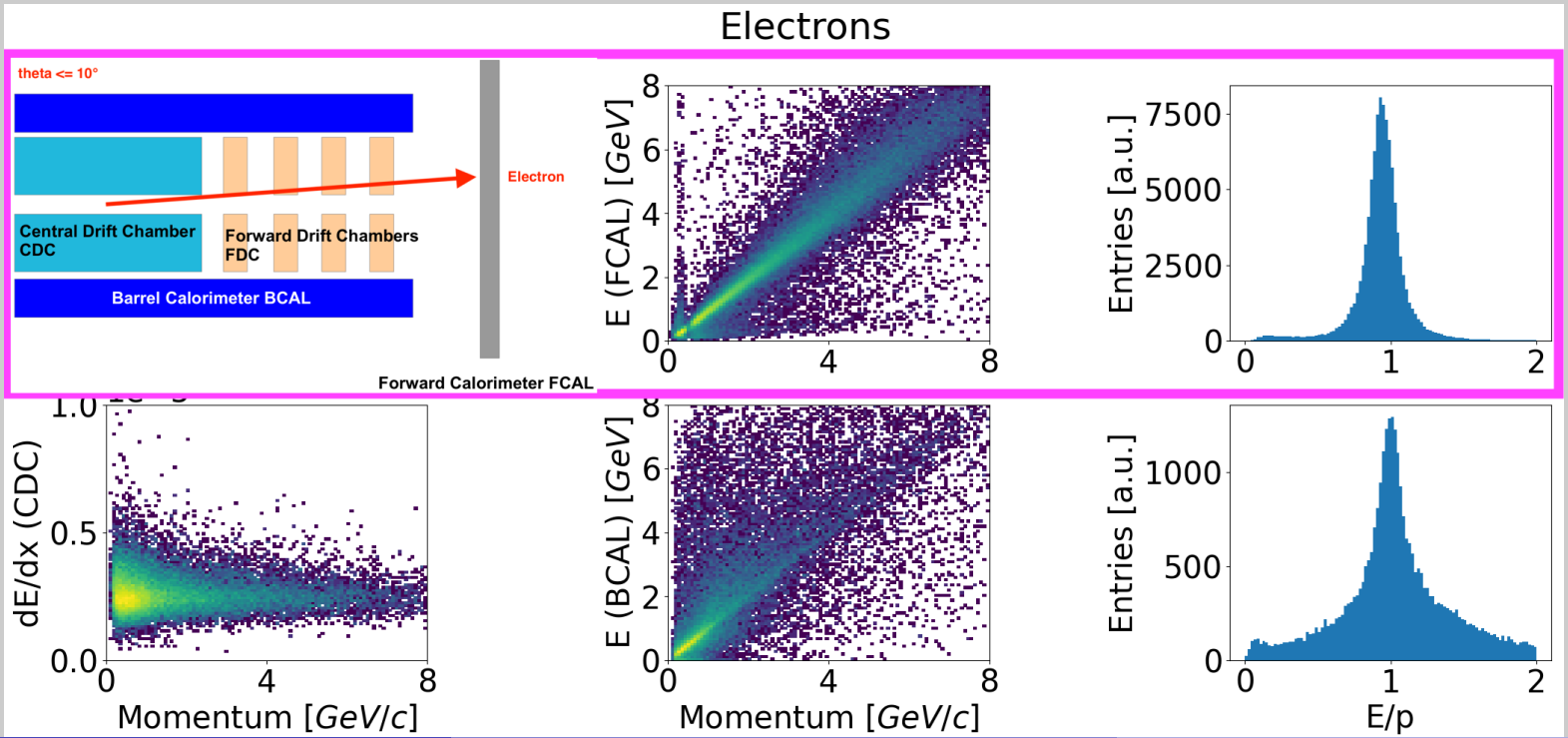
- Look at simulations
- Number of electrons = Number of pions
- Events are labeled: $e^{\pm}$: 1 and $\pi^{\pm}$: 0
- E/p = Energy deposit in calorimeter / Particle momentum



Understanding the Data

- Look at simulations
- Number of electrons = Number of pions
- Events are labeled: $e^{\pm}$: 1 and $\pi^{\pm}$: 0
- E/p = Energy deposit in calorimeter / Particle momentum
- Azimuthal angle:

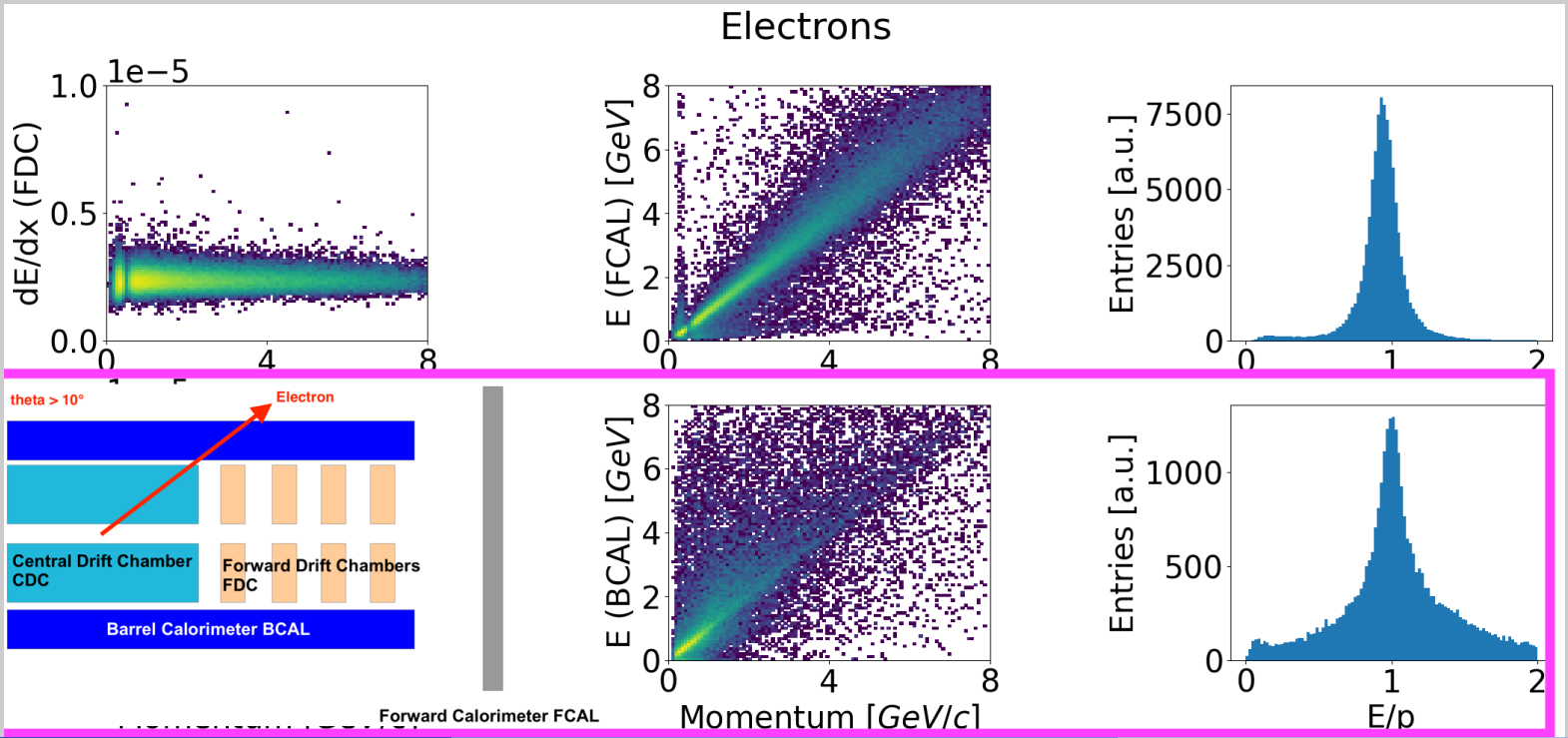
$$\theta < \begin{cases} \leq 10^{\circ} \text{ particle detected in GlueX forward part (FDC, FCAL)}, \\ > 10^{\circ} \text{ particle detected in GlueX central part (CDC, BCAL)} \end{cases}$$



Understanding the Data

- Look at simulations
- Number of electrons = Number of pions
- Events are labeled: $e^{\pm}$: 1 and $\pi^{\pm}$: 0
- E/p = Energy deposit in calorimeter / Particle momentum
- Azimuthal angle:

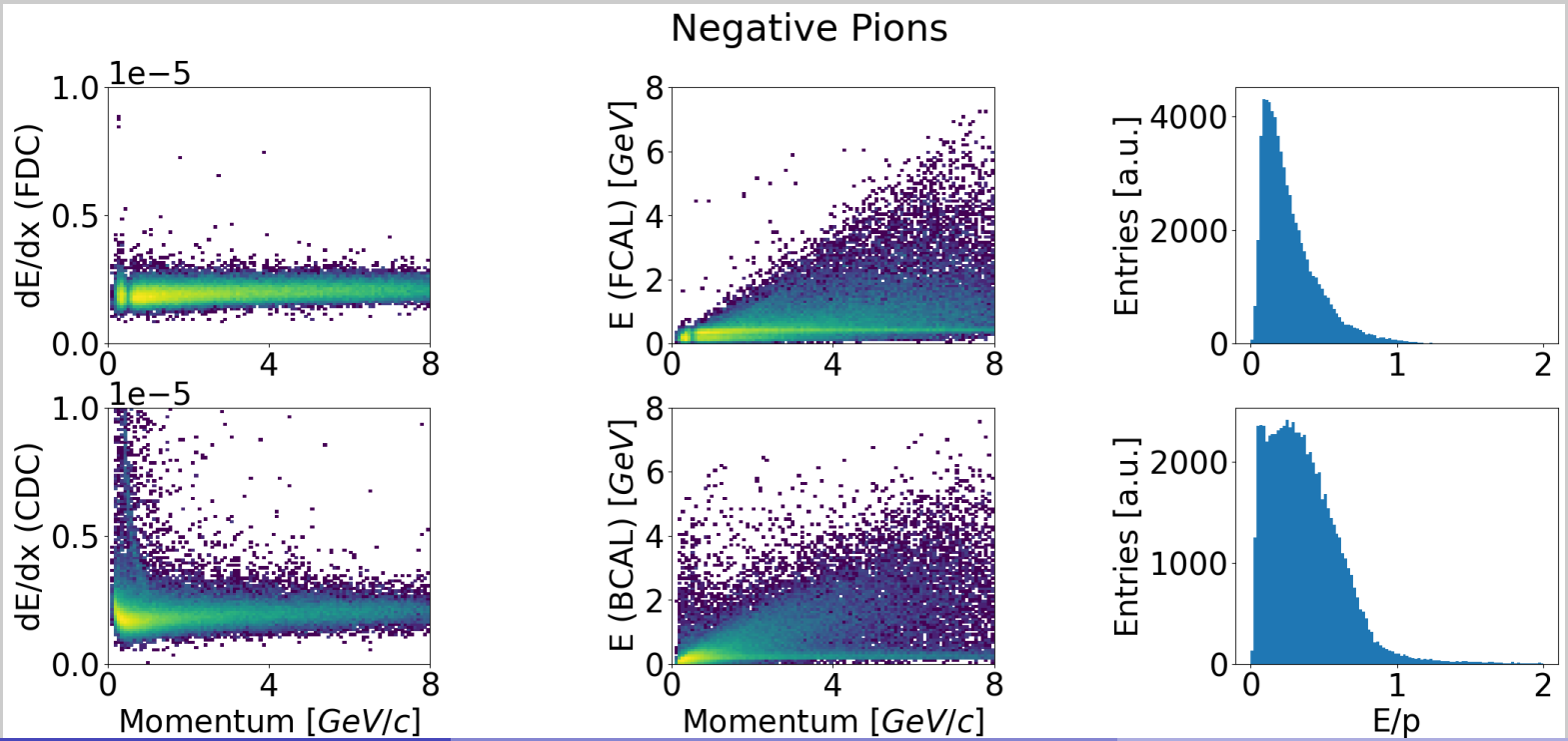
$$\theta < \begin{cases} \leq 10^{\circ} \text{ particle detected in GlueX forward part (FDC, FCAL),} \\ > 10^{\circ} \textbf{ particle detected in GlueX central part (CDC, BCAL)} \end{cases}$$



Understanding the Data

- Look at simulations
- Number of electrons = Number of pions
- Events are labeled: $e^{\pm}$: 1 and $\pi^{\pm}$: 0
- E/p = Energy deposit in calorimeter / Particle momentum
- Azimuthal angle:

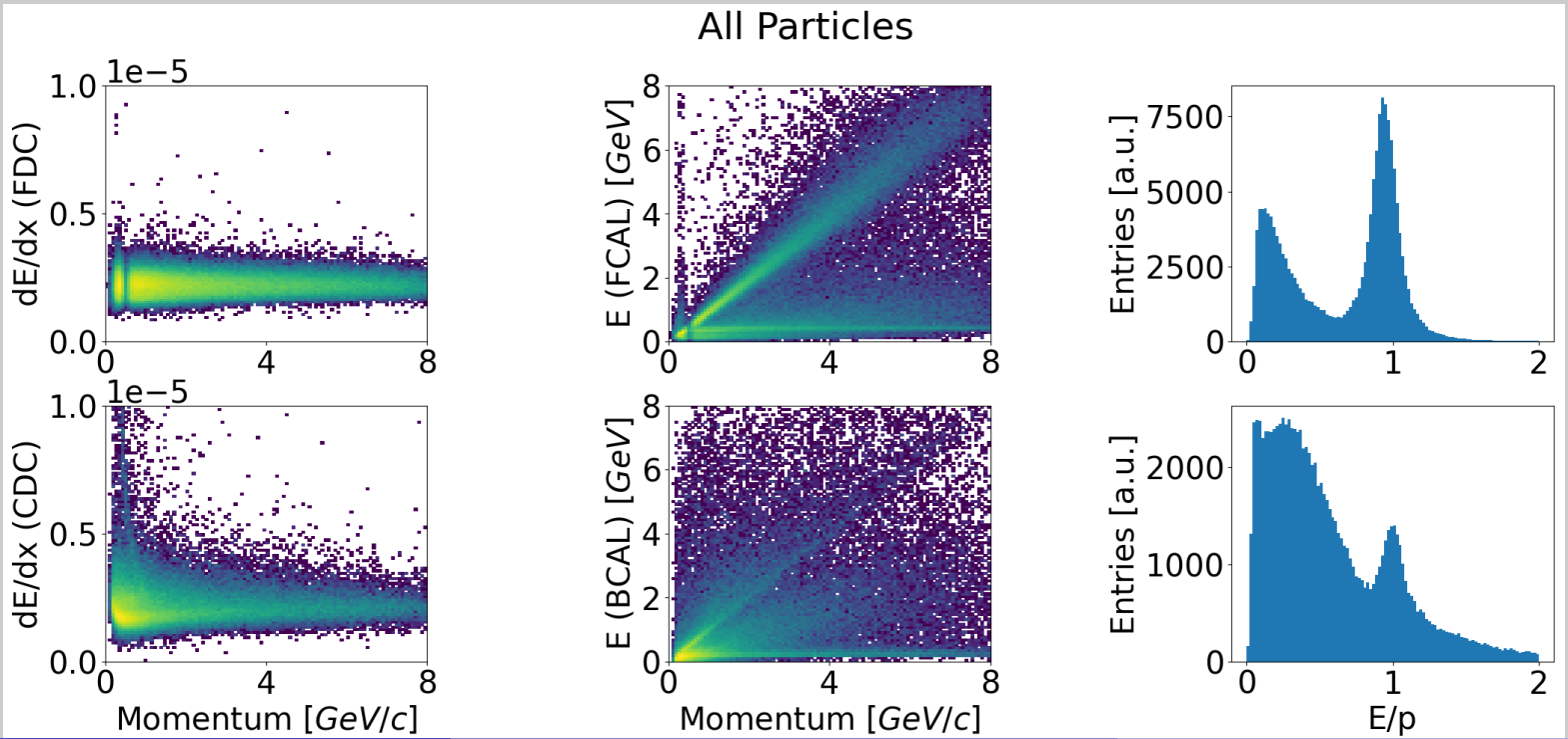
$$\theta < \begin{cases} \leq 10^{\circ} \text{ particle detected in GlueX forward part (FDC, FCAL),} \\ > 10^{\circ} \text{ particle detected in GlueX central part (CDC, BCAL)} \end{cases}$$



Understanding the Data

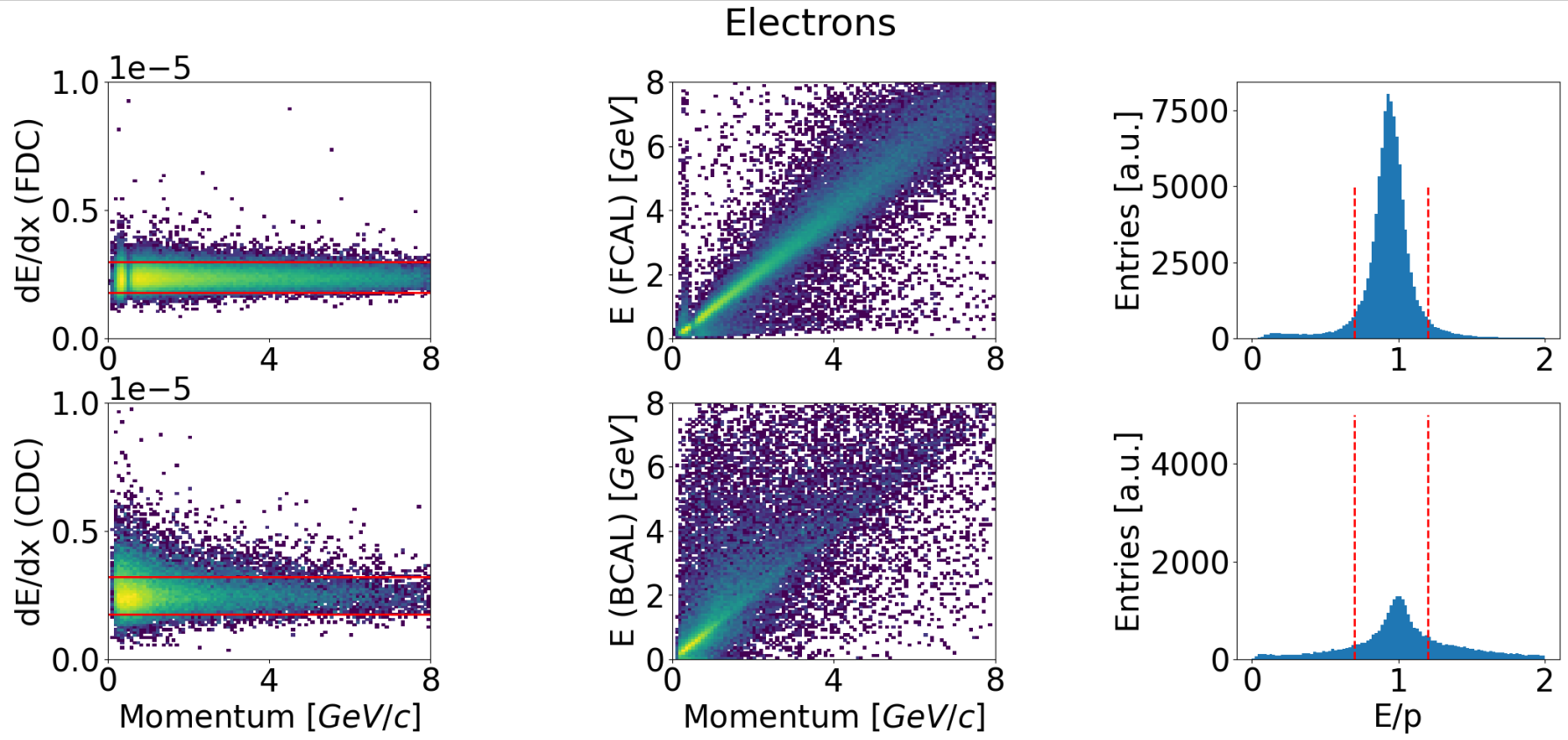
- Look at simulations
- Number of electrons = Number of pions
- Events are labeled: $e^{\pm}$: 1 and $\pi^{\pm}$: 0
- E/p = Energy deposit in calorimeter / Particle momentum
- Azimuthal angle:

$$\theta < \begin{cases} \leq 10^{\circ} \text{ particle detected in GlueX forward part (FDC, FCAL),} \\ > 10^{\circ} \text{ particle detected in GlueX central part (CDC, BCAL)} \end{cases}$$



Reference Analysis

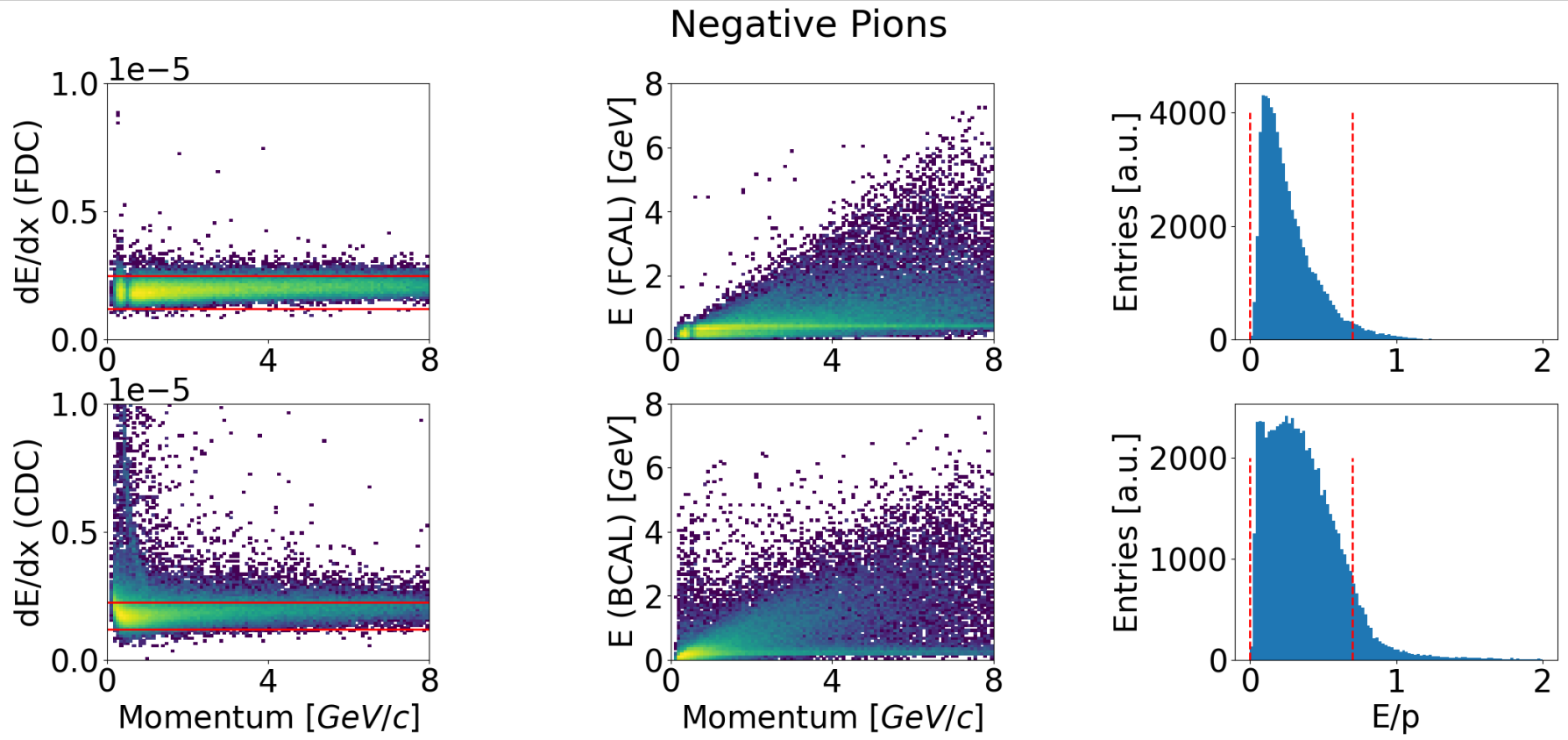
Always have one!



- Solid / dashed red lines indicate event selection
- Event is labeled $\begin{cases} 1 : \text{Event passes selection criteria,} \\ 0 : \text{Event does NOT pass selection criteria} \end{cases}$

Reference Analysis

Always have one!



- Solid / dashed red lines indicate event selection
- Event is labeled
  - 0 : Event passes selection criteria,
  - 1 : Event does NOT pass selection criteria

Performance Evaluation

- How do we know if our event selection was good / bad?

Performance Evaluation

- How do we know if our event selection was good / bad?
- Luckily, our data is labeled

Performance Evaluation

- How do we know if our event selection was good / bad?
- Luckily, our data is labeled
- Compare true labels with those from our event selection

Performance Evaluation

- How do we know if our event selection was good / bad?
- Luckily, our data is labeled
- Compare true labels with those from our event selection
- **Ideally:** True and predicted labels are identical

Performance Evaluation

- How do we know if our event selection was good / bad?
- Luckily, our data is labeled
- Compare true labels with those from our event selection
- **Ideally:** True and predicted labels are identical
- **Define:**

i) True Positive Rate $= \frac{\#\text{Events CORRECTLY labeled as 1 (0)}}{\#\text{Events truly labeled as 1 (0)}}$

ii) False Positive Rate $= \frac{\#\text{Events FALSELY labeled as 1 (0)}}{\#\text{Events truly labeled as 0 (1)}}$

Performance Evaluation

- How do we know if our event selection was good / bad?
- Luckily, our data is labeled
- Compare true labels with those from our event selection
- **Ideally:** True and predicted labels are identical
- **Define:**

i) True Positive Rate $= \frac{\#\text{Events CORRECTLY labeled as 1 (0)}}{\#\text{Events truly labeled as 1 (0)}}$

ii) False Positive Rate $= \frac{\#\text{Events FALSELY labeled as 1 (0)}}{\#\text{Events truly labeled as 0 (1)}}$

- These are the building blocks for nearly all classification performance metrics

Identification Rates

Particle	True Positive Rate (TPR)	False Positive Rate (FPR)
Electrons	0.85	0.11
Negative Pions	0.89	0.15

- **Ideally:** TPR = 1.0 and FPR = 0.0
- TPR(Electron / Pion) + FPR(Pion / Electron) = 1.0
  $\Rightarrow$ Number of particles is conserved!

Identification Rates

Particle	True Positive Rate (TPR)	False Positive Rate (FPR)
Electrons	0.85	0.11
Negative Pions	0.89	0.15

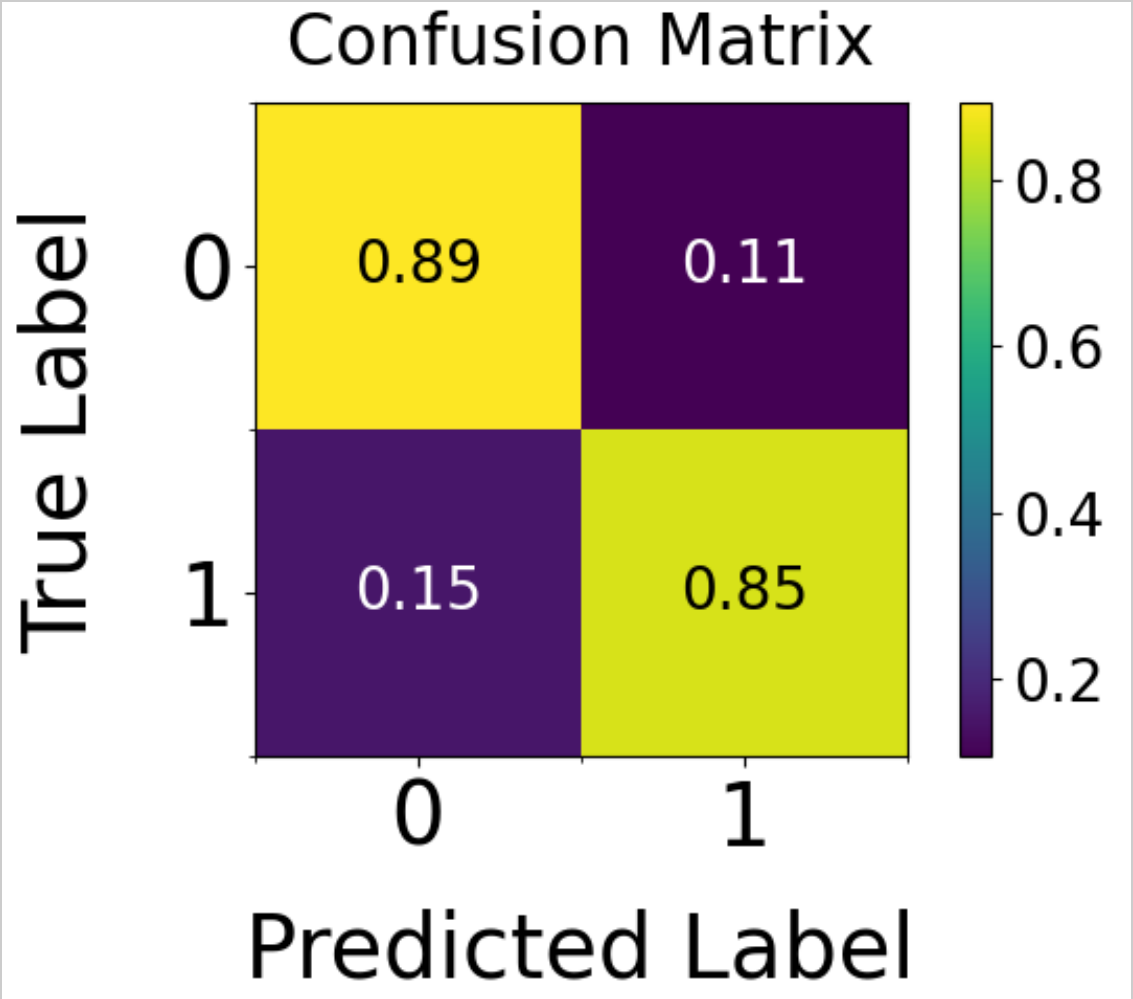
- **Ideally:** TPR = 1.0 and FPR = 0.0
- TPR(Electron / Pion) + FPR(Pion / Electron) = 1.0
  $\Rightarrow$ Number of particles is conserved!

Identification Rates

Particle	True Positive Rate (TPR)	False Positive Rate (FPR)
Electrons	0.85	0.11
Negative Pions	0.89	0.15

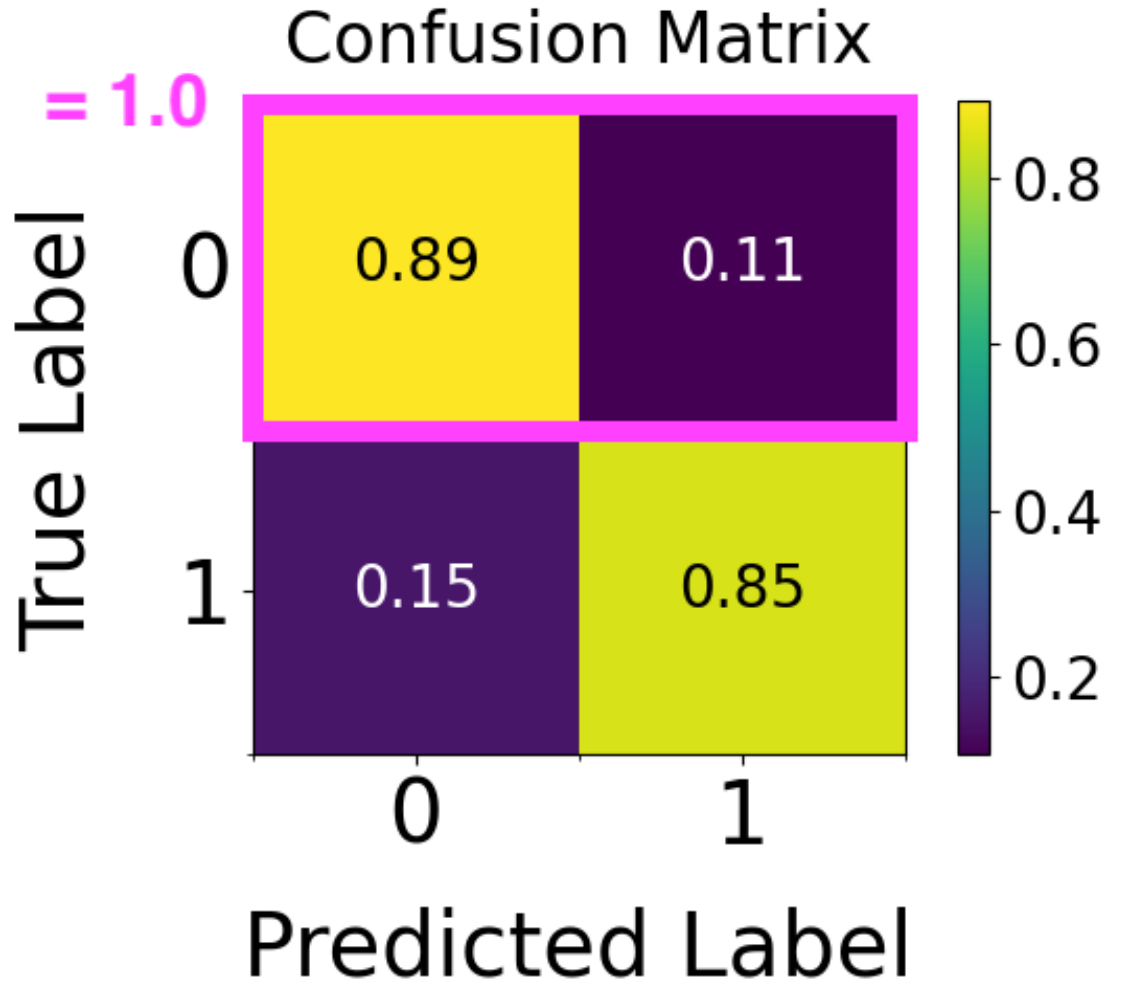
- **Ideally:** TPR = 1.0 and FPR = 0.0
- TPR(Electron / Pion) + FPR(Pion / Electron) = 1.0
  $\Rightarrow$ Number of particles is conserved!

The Confusion Matrix



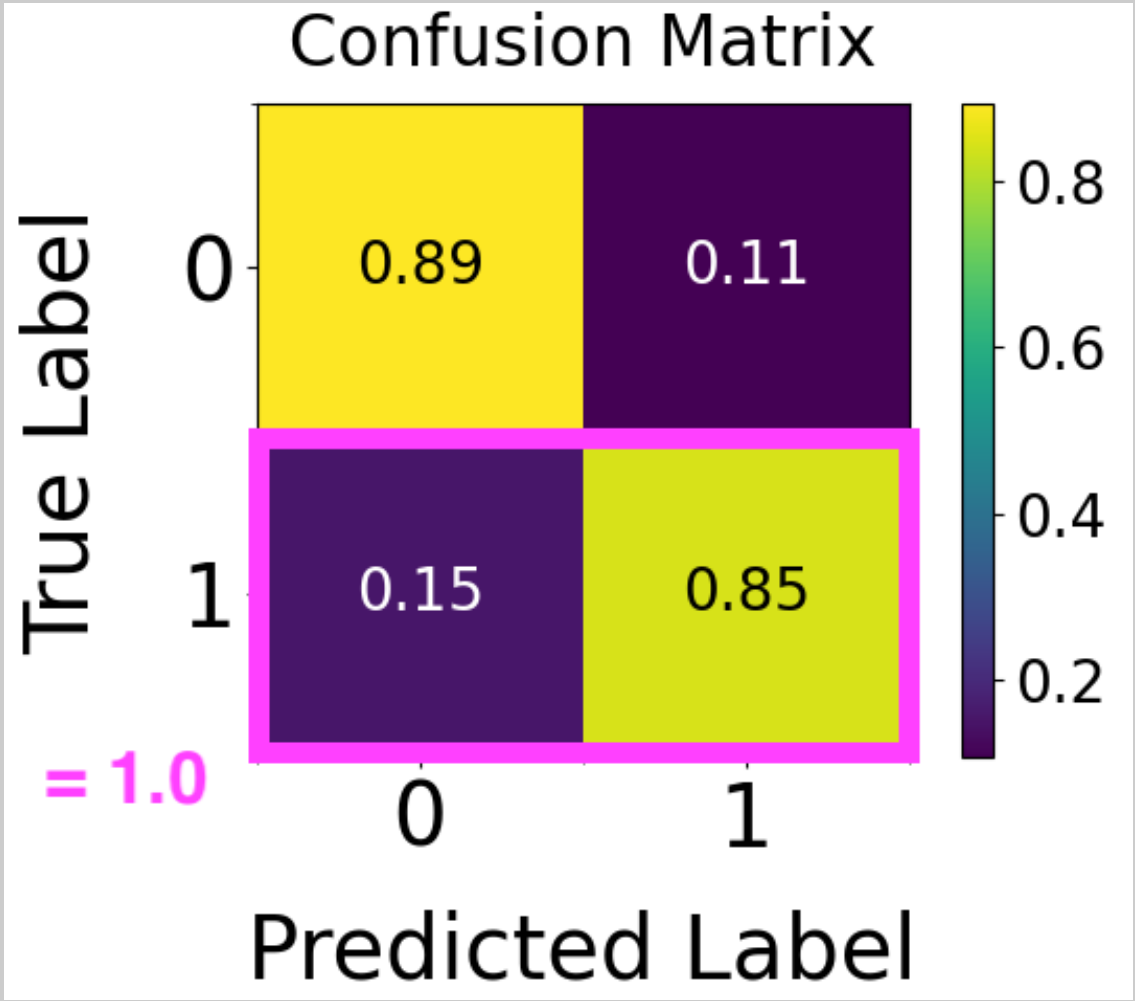
- The confusion matrix is another way to display identification rates
- True positive rates are shown along the diagonal
- **Note:** No rule for which axis holds true / predicted labels

The Confusion Matrix



- The confusion matrix is another way to display identification rates
- True positive rates are shown along the diagonal
- **Note:** No rule for which axis holds true / predicted labels

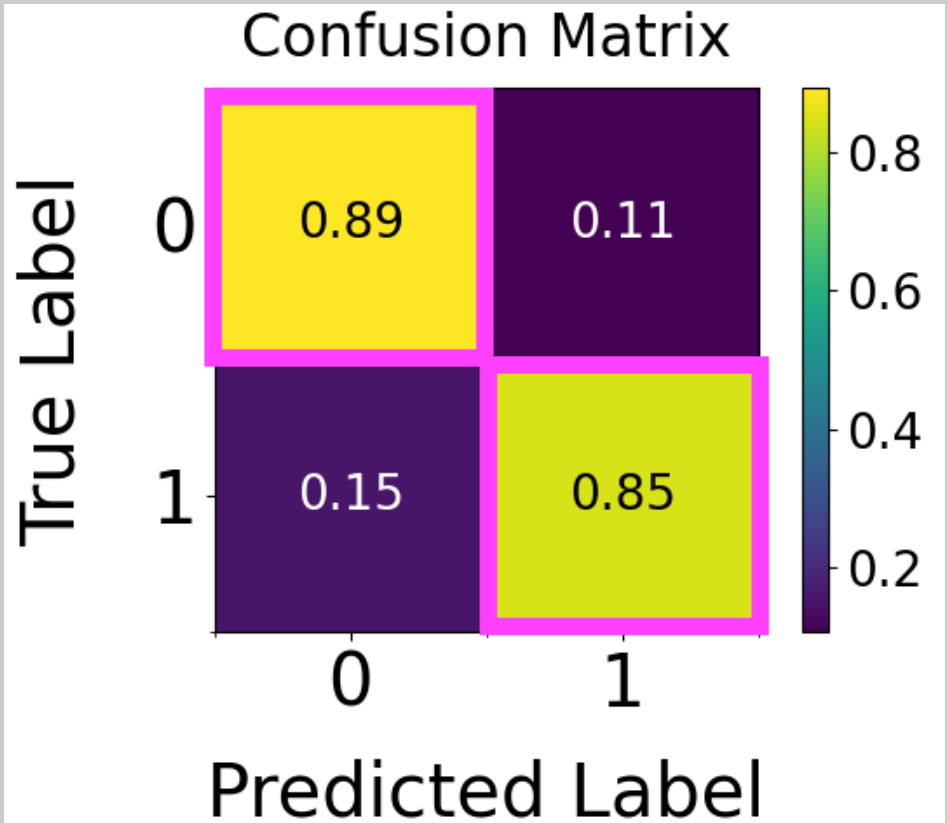
The Confusion Matrix



- The confusion matrix is another way to display identification rates
- True positive rates are shown along the diagonal
- **Note:** No rule for which axis holds true / predicted labels

The Accuracy

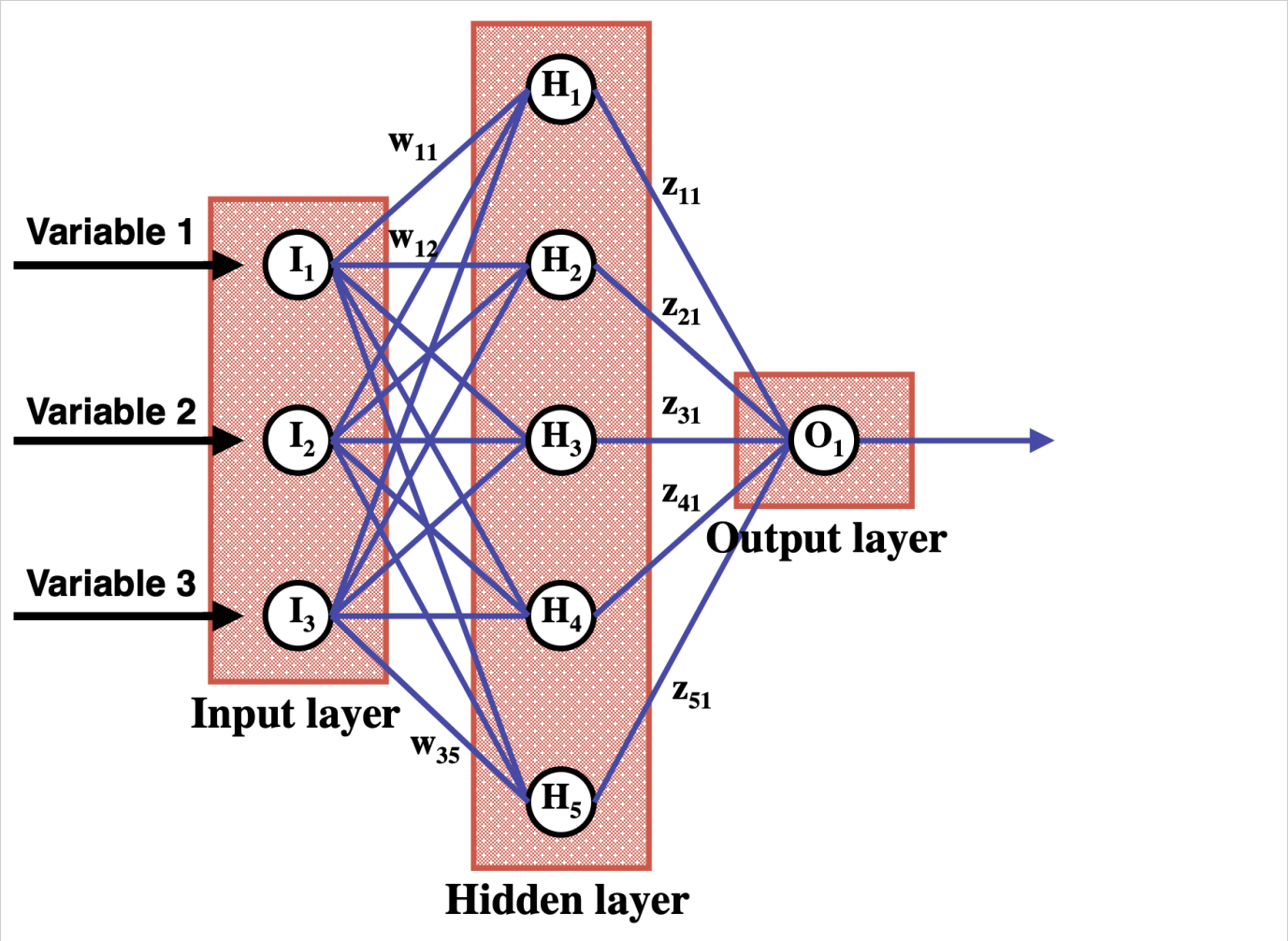
- Many performance metrics can be (directly) derived from the confusion matrix
- **Example: Accuracy** $= \vec{diag} \cdot \vec{R} = \sum_{i}^{\#species} TPR(i) \cdot R(i)$
- R(i) denotes abundance of each particle species in the data
- Balanced data (each species equally represented): R(i) = 1 / number species
- **Here: R(i) = 0.5 and Accuracy** $= 0.89 \cdot 0.5 + 0.85 \cdot 0.5 = 0.87$



Can we do better?

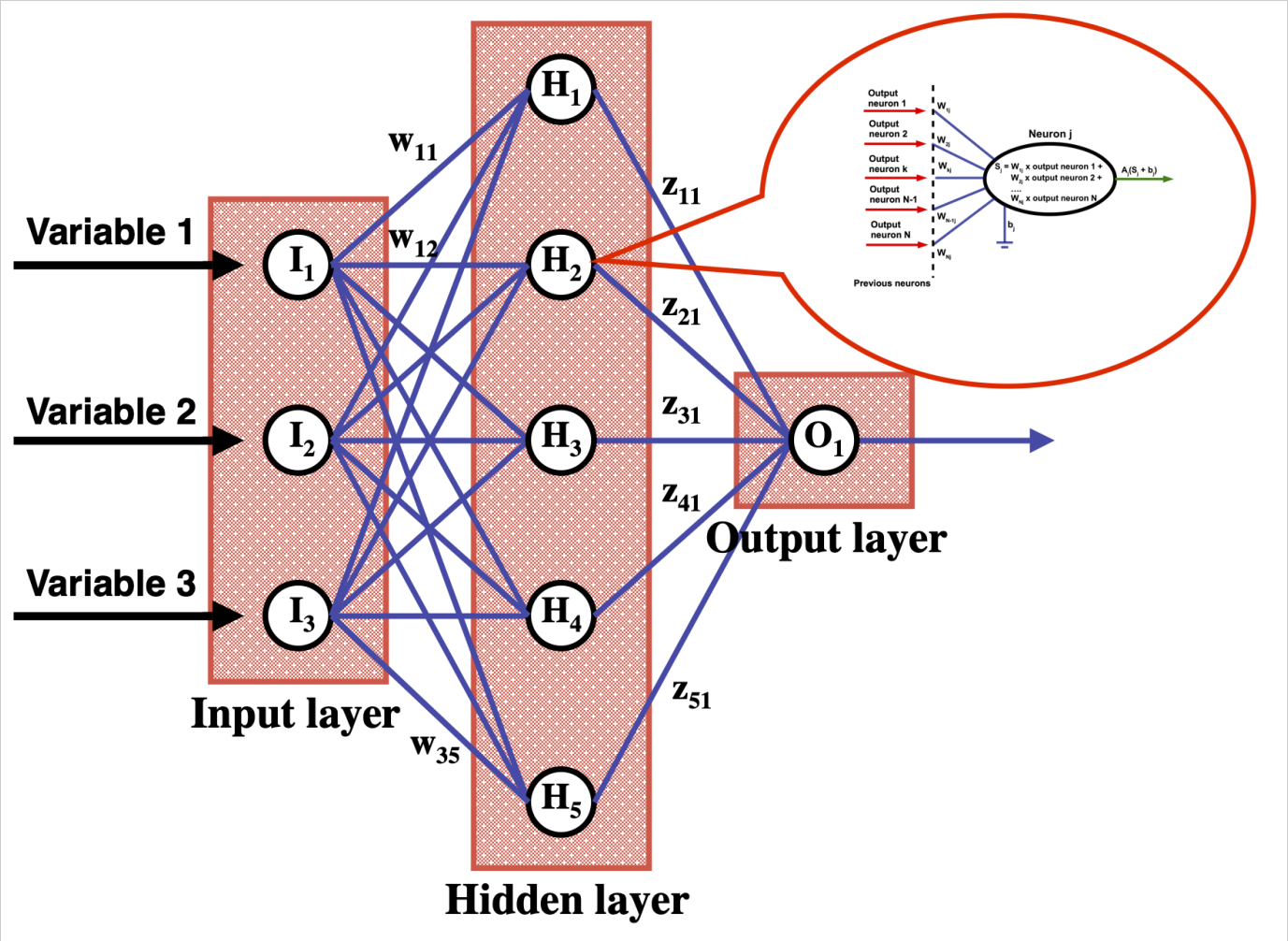
- Would like to optimize PID $\Rightarrow$ TPR = 1 / FPR = 0
- Could tune E/p by hand $\rightarrow$ Tedious and does not use all information
- Input features:
  1.) Is particle detected in forward / central part of GlueX $\equiv \theta$
  2.) Particle Momentum
  3.) Information from forward drift chamber (FDC)
  4.) Information from forward calorimeter (FCAL)
  5.) Information from central drift chamber (CDC)
  6.) Information from central calorimeter (BCAL)
- **Stone-Weierstrass-Theorem**(1990): "[...] there are no nemesis functions that cannot be modeled by neural networks"

The Multilayer Perceptron



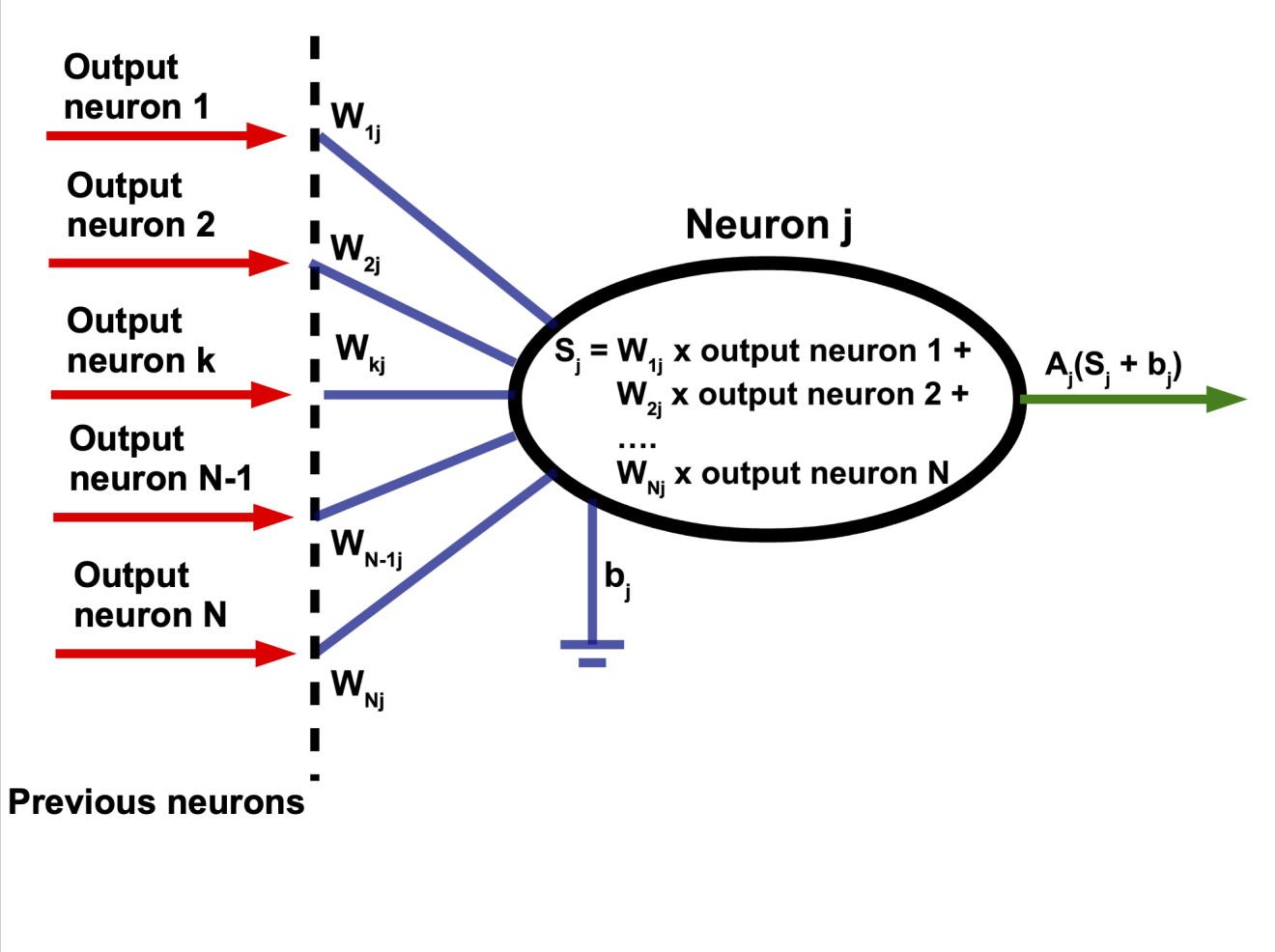
- Most popular example for machine learning algorithms
- **Architecture:** Hidden layers with a set of neurons

The Multilayer Perceptron



- Most popular example for machine learning algorithms
- **Architecture:** Hidden layers with a set of neurons

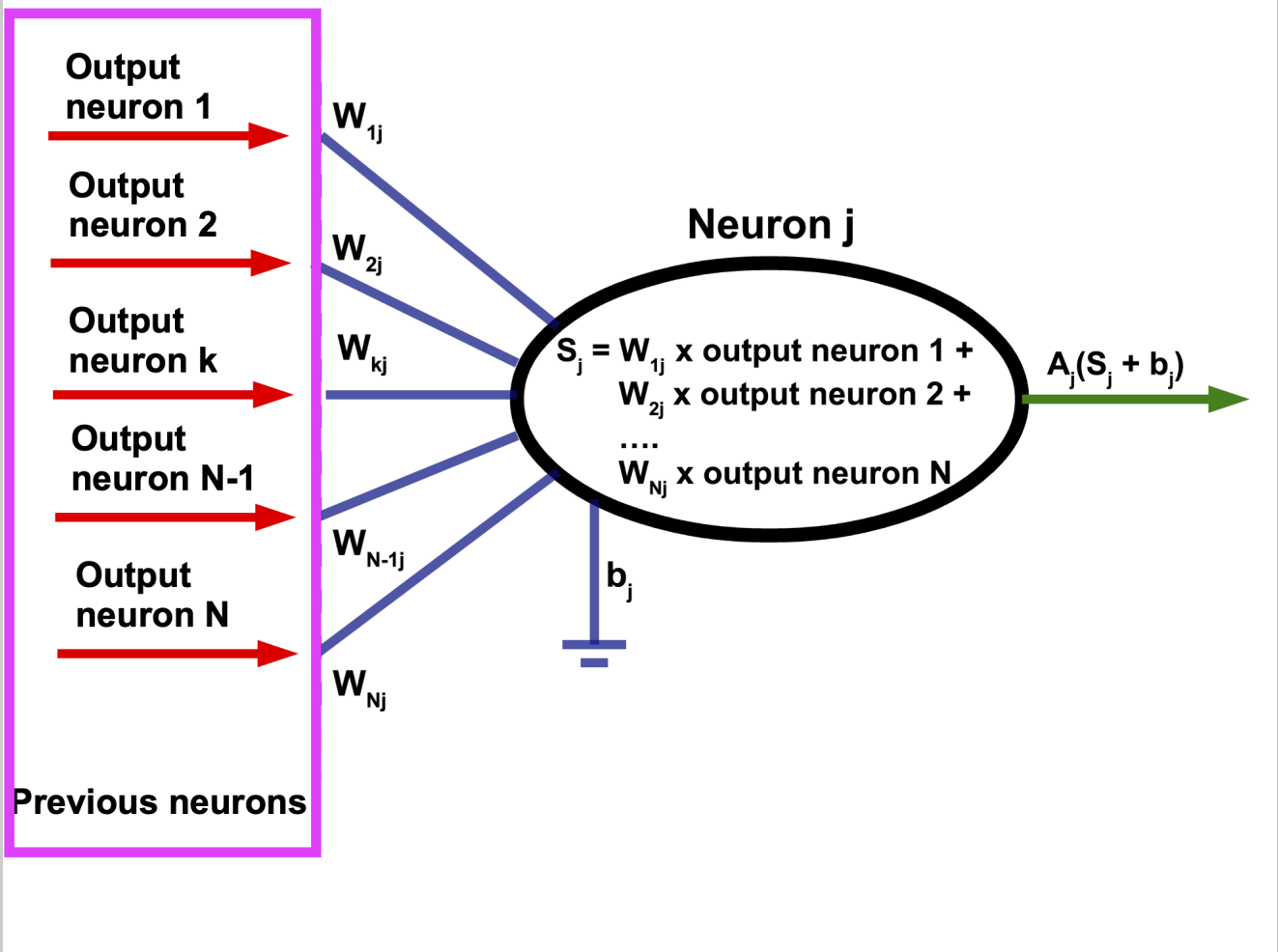
A single Neuron



Basic ingredients:

- Information from previous neurons
- Weights and bias
- Activation function

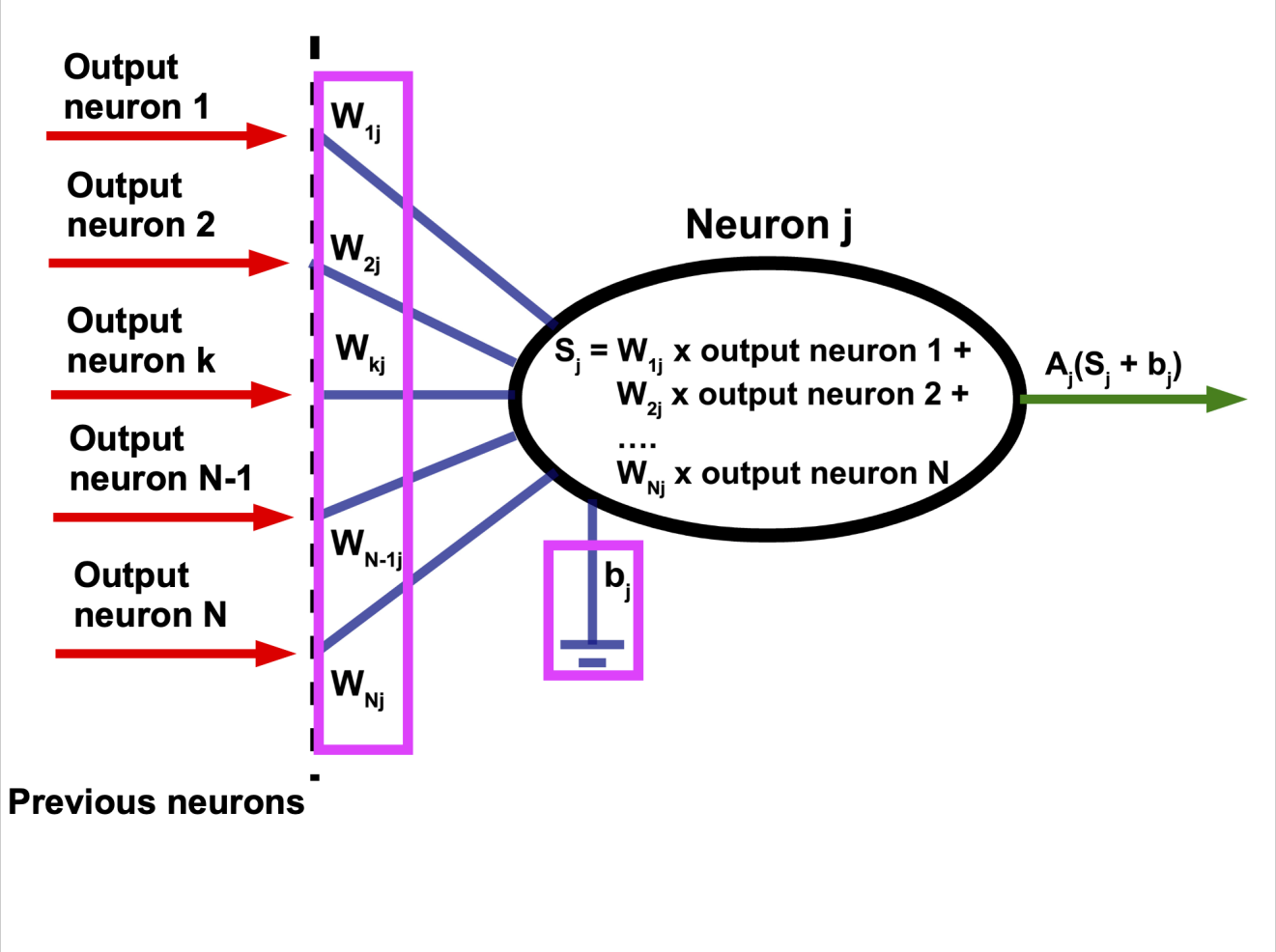
A single Neuron



Basic ingredients:

- **Information from previous neurons**
- Weights and bias
- Activation function

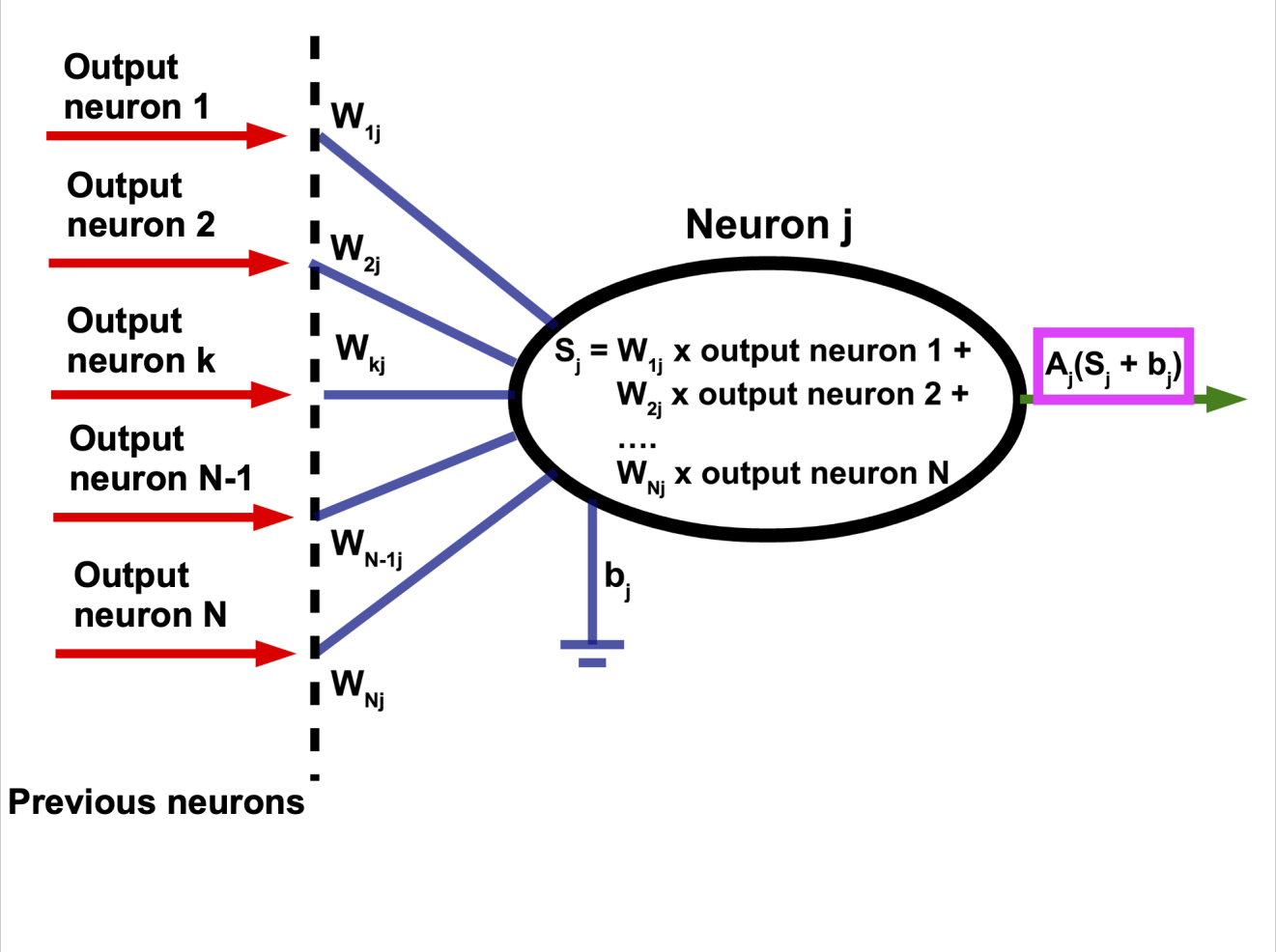
A single Neuron



Basic ingredients:

- Information from previous neurons
- **Weights and bias** $\Rightarrow$ Adjusted via training
- Activation function

A single Neuron



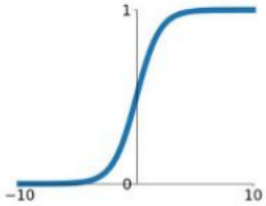
Basic ingredients:

- Information from previous neurons
- Weights and bias
- **Activation function**

Activation Functions

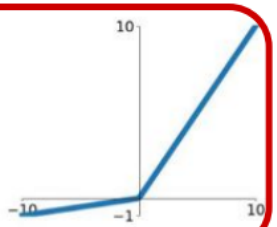
Sigmoid

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



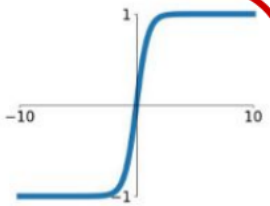
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

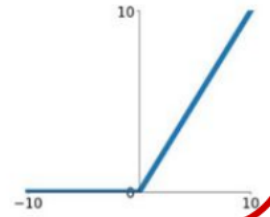


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

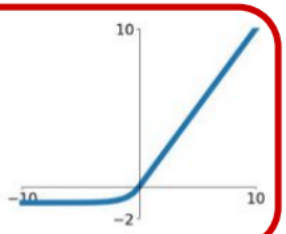
ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



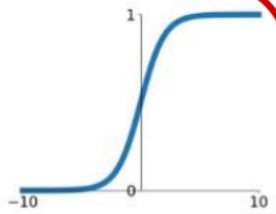
Most commonly used in modern networks as hidden layer activations

Activation function plots taken from Mustafa Mustafas lecture at the: deep learning for science school 2019

Activation Functions

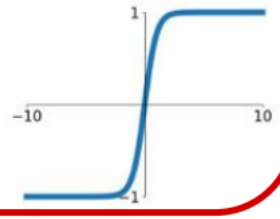
Sigmoid

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



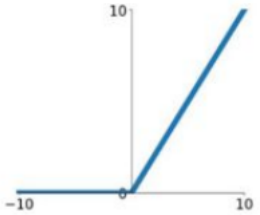
tanh

$$\tanh(x)$$



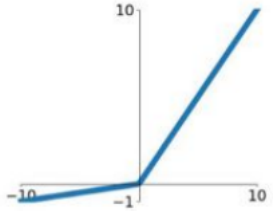
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

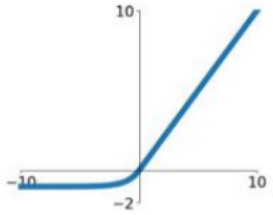


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

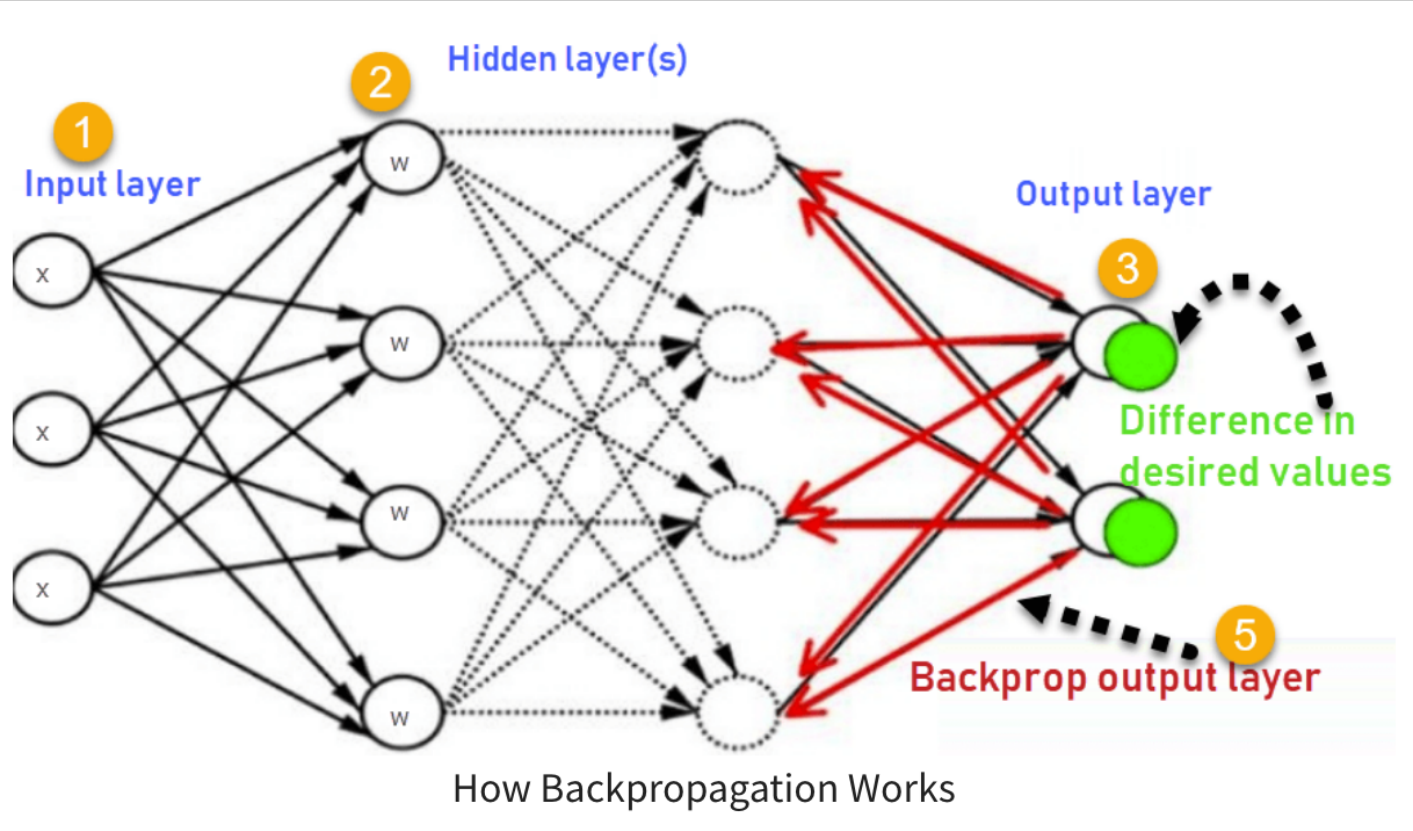
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Often used for output layers

Activation function plots taken from Mustafa Mustafas lecture at the: deep learning for science school 2019

Training a Neural Network: Backpropagation

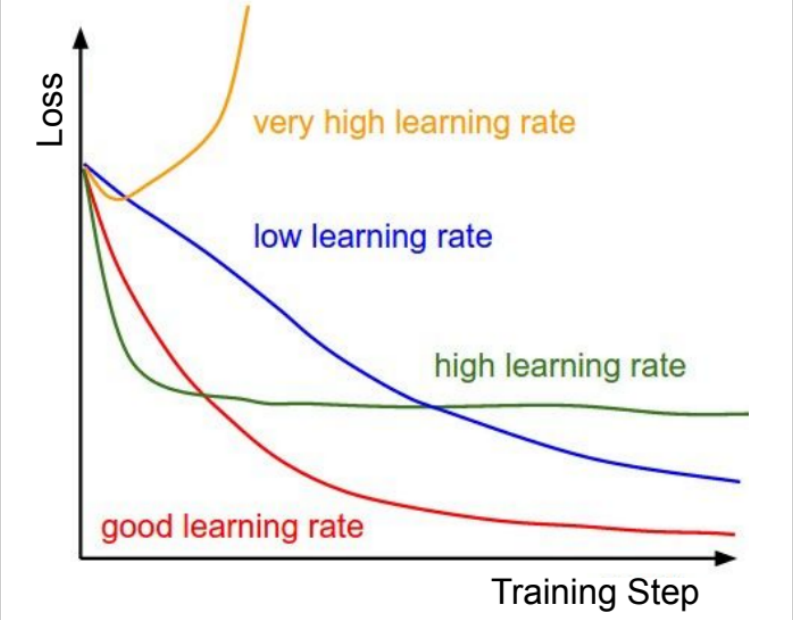


How Backpropagation Works

- Pass data forward
- Calculate error
- Pass error backward to update weights

Picture taken from here

Training a Neural Network: Updating Weights and Convergence



- **Goal:** Find minimum error (loss) as fast as possible:

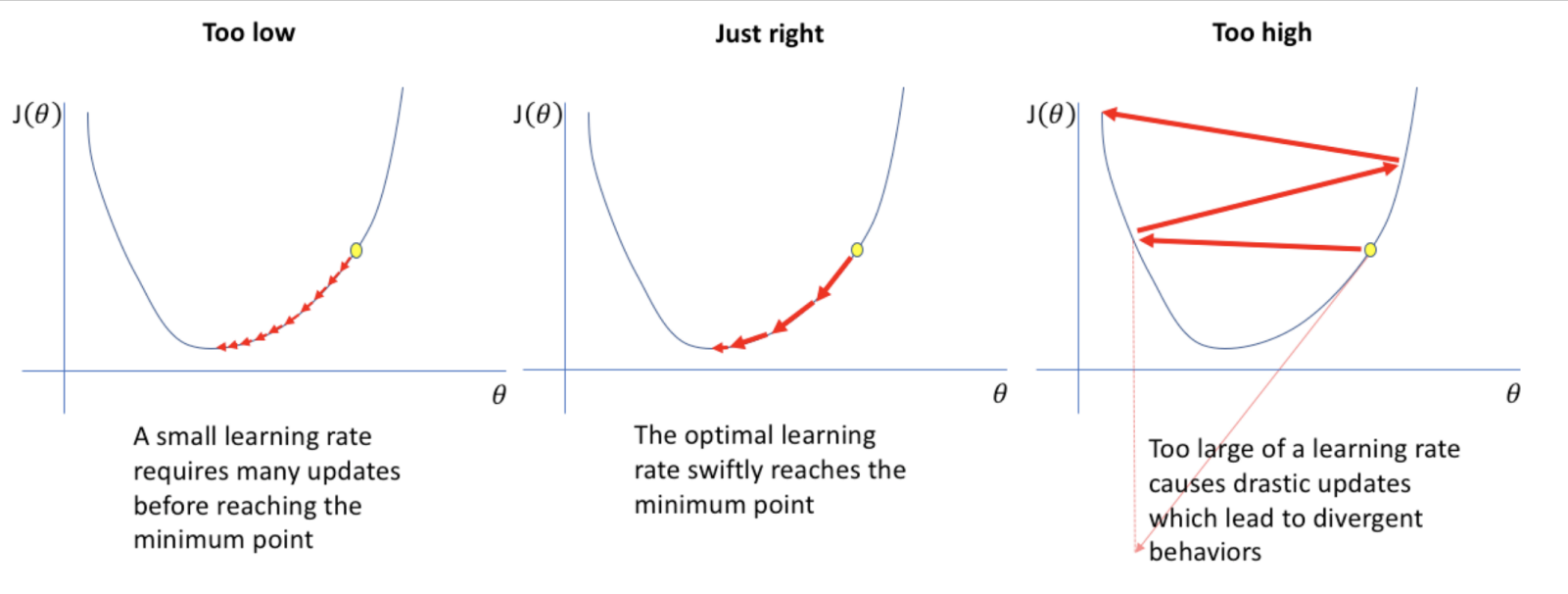
$$0 \stackrel{!}{=} d(Error) \propto \sum_i \sum_j \frac{\partial \text{Error}}{\partial w_{ij}} dw_{ij} \tag{1}$$

- Use:

$$\textbf{Gradient Descent: } \boldsymbol{w_{k+1} = w_k - \eta \nabla \text{Error}(w_k)} \tag{2}$$

- Error $\propto$ prediction - truth
- η : Learning rate

Training a Neural Network: Updating Weights and Convergence



- **Goal:** Find minimum error (loss) as fast as possible:

$$0 \stackrel{!}{=} d(Error) \propto \sum_i \sum_j \frac{\partial \text{Error}}{\partial w_{ij}} dw_{ij} \tag{1}$$

- Use:

$$\textbf{Gradient Descent: } \boldsymbol{w_{k+1} = w_k - \eta \nabla \text{Error}(w_k)} \tag{2}$$

- Error $\propto$ prediction - truth
- η : Learning rate

Validation Data and Generalizability

- Want to enable network to abstract / generalize on unknown data AND avoid overfitting (i.e. avoid that network reproduces features from training data only)

Picture taken from Brenda Ngs introductory talk at the: deep learning for science school 2019

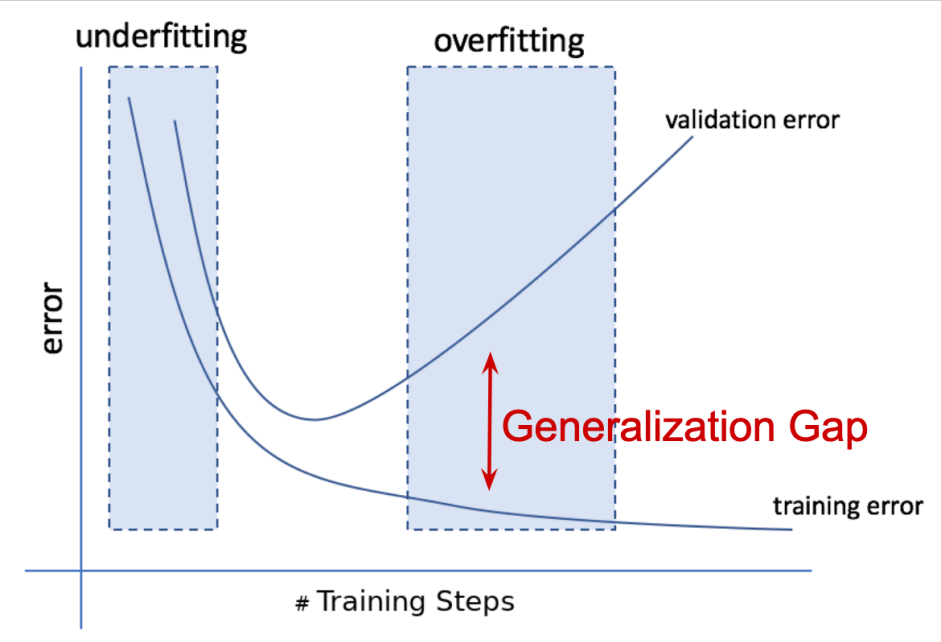
Validation Data and Generalizability

- Want to enable network to abstract / generalize on unknown data AND avoid overfitting (i.e. avoid that network reproduces features from training data only)
- **Validation Data:** Part of training data that is NOT used to update internal parameters[1], but used to determine when training is complete

[1] This data is "unseen" by the algorithm during the training stage

Validation Data and Generalizability

- Want to enable network to abstract / generalize on unknown data AND avoid overfitting (i.e. avoid that network reproduces features from training data only)
- **Validation Data:** Part of training data that is NOT used to update internal parameters[1], but used to determine when training is complete

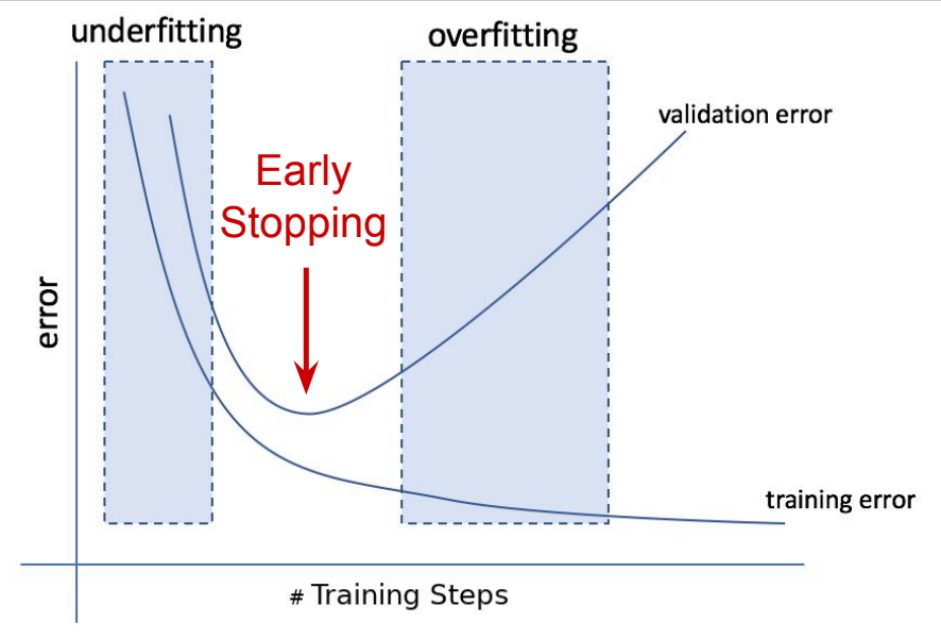


Picture taken from Mustafa Mustafas talk at the: deep learning for science school 2019

[1] This data is "unseen" by the algorithm during the training stage

Validation Data and Generalizability

- Want to enable network to abstract / generalize on unknown data AND avoid overfitting (i.e. avoid that network reproduces features from training data only)
- **Validation Data:** Part of training data that is NOT used to update internal parameters[1], but used to determine when training is complete



Picture taken from Mustafa Mustafas talk at the: deep learning for science school 2019

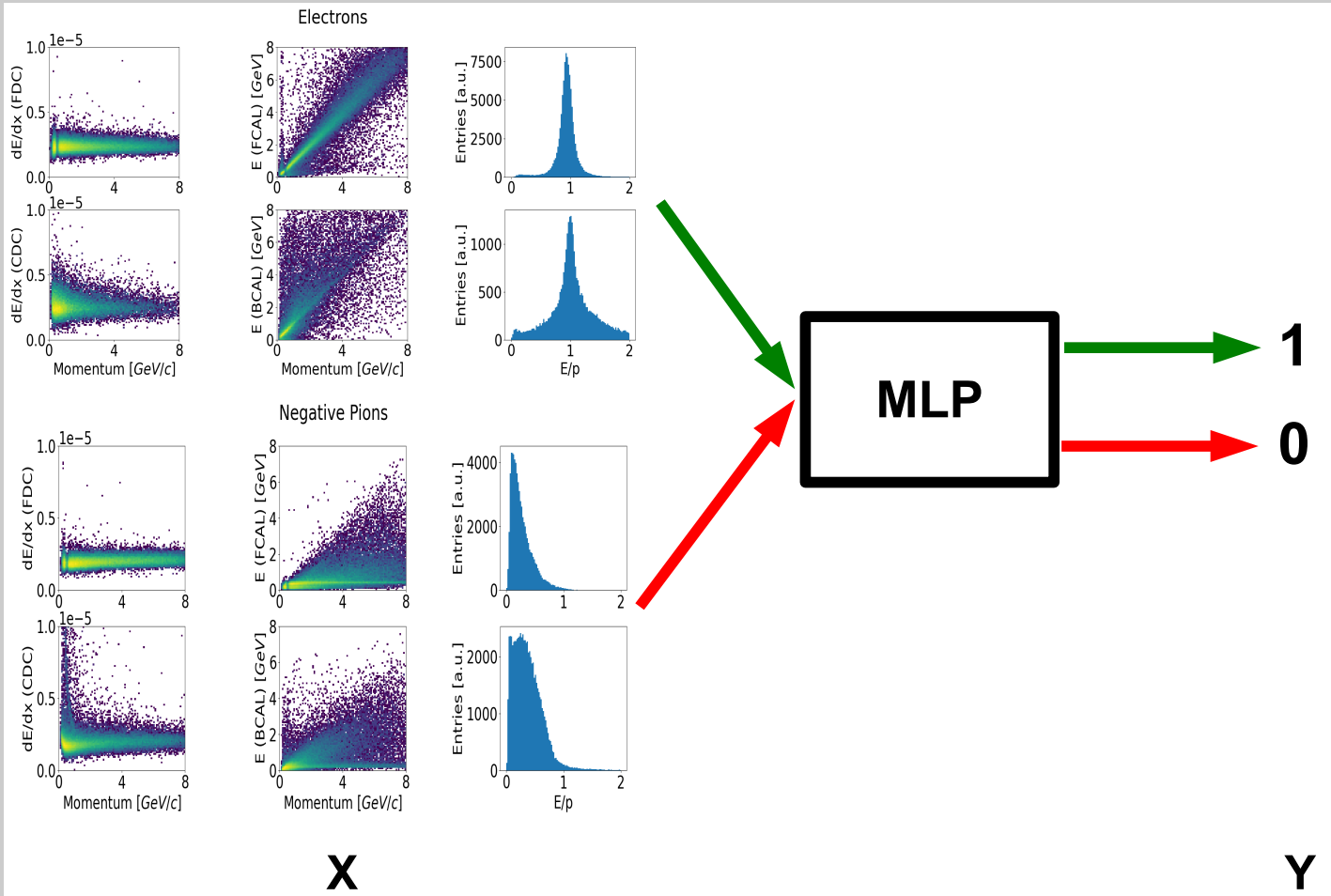
[1] This data is "unseen" by the algorithm during the training stage

Setting up and training a Neural Network in Scikit

```
from sklearn.neural_network import MLPClassifier

#Set up the network:
 my_mlp = MLPClassifier(
     hidden_layer_sizes=(8,5), #two hidden layers with 8/5 neurons
     activation='relu', #rectified linear unit function
     solver='sgd', #stochastic gradient descent optimizer #--> minimize error
     max_iter = 200, #maximum number of learning epochs
     validation_fraction=0.5, #--> Use 50$\%$ of data for validation
     early_stopping=True,#--> Use early stopping
     learning_rate_init = 0.05 #step size for the gradient
  )
```

Setting up and training a Neural Network in Scikit

```
#Train the network:
my_mlp.fit(X,Y)
#X = input data (energies, momenta,..)
#Y = target labels (0: pions / 1: leptons)
```



Setting up and training a Neural Network in Scikit

```
from sklearn.neural_network import MLPClassifier

#Set up the network:
 my_mlp = MLPClassifier(
     hidden_layer_sizes=(8,5), #two hidden layers with 8/5 neurons
     activation='relu', #rectified linear unit function
     solver='sgd', #stochastic gradient descent optimizer #--> minimize error
     max_iter = 200, #maximum number of learning epochs
     validation_fraction=0.5, #--> Use 50$\%$ of data for validation
     early_stopping=True,#--> Use early stopping
     learning_rate_init = 0.05 #step size for the gradient
  )

  #Train the network:
  my_mlp.fit(X,Y)
  #X = input data (energies, momenta,..)
  #Y = target labels (0: pions / 1: leptons)

  #Get the loss curve and validation score:
  loss_curve = my_mlp.loss_curve_
  val_score = my_mlp.validation_score_
```

Setting up a Neural Network in TF/Keras

```
import tensorflow as tf

#1.) Set up the sequential model:
my_mlp = tf.keras.Sequential(name='keras_mlp')
```

Setting up a Neural Network in TF/Keras

```
import tensorflow as tf

#1.) Set up the sequential model:
my_mlp = tf.keras.Sequential(name='keras_mlp')
#2.) Define the input layer:
input_layer = tf.keras.layers.Input(shape=(6,),name='input')
#3.) Define hidden layer(s):
dense_layer_1 = tf.keras.layers.Dense(8,'relu',name='dense_1')
dense_layer_2 = tf.keras.layers.Dense(5,'relu',name='dense_2')
#5.) Define the output layer:
output_layer = tf.keras.layers.Dense(1,'sigmoid',name='output')
#6.) Put the all the layers together:
my_mlp.add(input_layer)
my_mlp.add(dense_layer_1)
my_mlp.add(dense_layer_2)
my_mlp.add(output_layer)
```

Setting up a Neural Network in TF/Keras

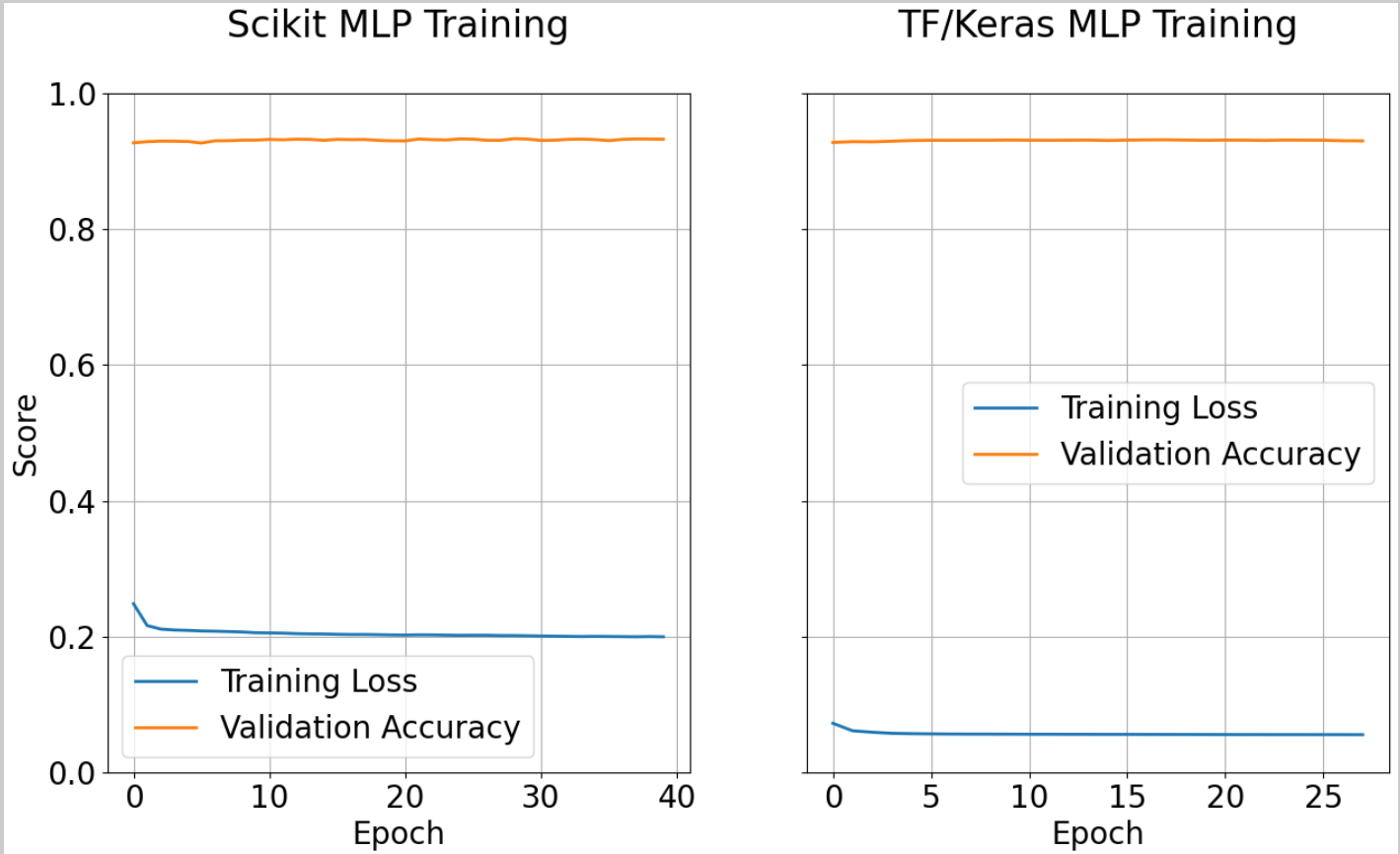
```
import tensorflow as tf

#1.) Set up the sequential model:
my_mlp = tf.keras.Sequential(name='keras_mlp')
#2.) Define the input layer:
input_layer = tf.keras.layers.Input(shape=(6,),name='input')
#3.) Define hidden layer(s):
dense_layer_1 = tf.keras.layers.Dense(8,'relu',name='dense_1')
dense_layer_2 = tf.keras.layers.Dense(5,'relu',name='dense_2')
#5.) Define the output layer:
output_layer = tf.keras.layers.Dense(1,'sigmoid',name='output')
#6.) Put the all the layers together:
my_mlp.add(input_layer)
my_mlp.add(dense_layer_1)
my_mlp.add(dense_layer_2)
my_mlp.add(output_layer)
#7.) Define the loss-function:
mlp_loss = tf.keras.losses.binary_crossentropy
#8.) Define the optimizer:
mlp_optimizer = tf.keras.optimizers.SGD(learning_rate=0.05)
#9.) Compile the model and get a summary:
my_mlp.compile(optimizer=mlp_optimizer,loss=mlp_loss,metrics=['accuracy'])
my_mlp.summary()
```

Training a Neural Network with TF/Keras

```
Model: "keras_mlp"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_0 (Dense)              (None, 8)                 56
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 45
_________________________________________________________________
output (Dense)               (None, 1)                 6
=================================================================
Total params: 107
Trainable params: 107
Non-trainable params: 0
_________________________________________________________________
```

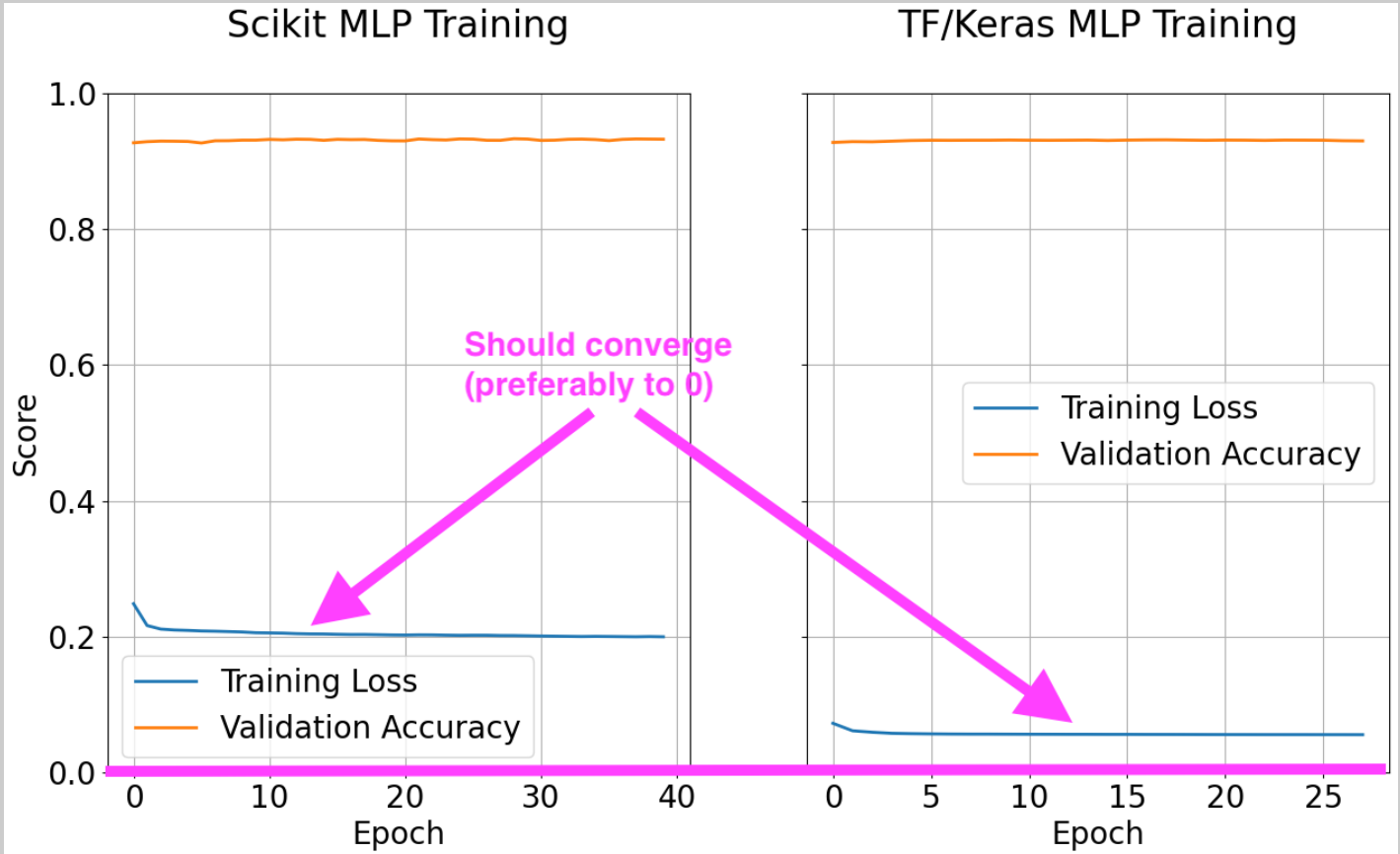
```
#Train the network
training_history = my_mlp.fit(
      x=X, #X = input data (energies, momenta,..)
      y=Y,  #Y = target labels (0: pions / 1: leptons)
      validation_slpit=0.5, #--> Use 50$\%$ of data for validation
      callbacks=tf.keras.EarlySopping() #--> Customize the
      #early stopping function
)

#Get the training loss and validation score:
loss_curve = training_history.history['loss']
val_score = training_history.history['val_accuracy']
```

Inspecting the learning Curves



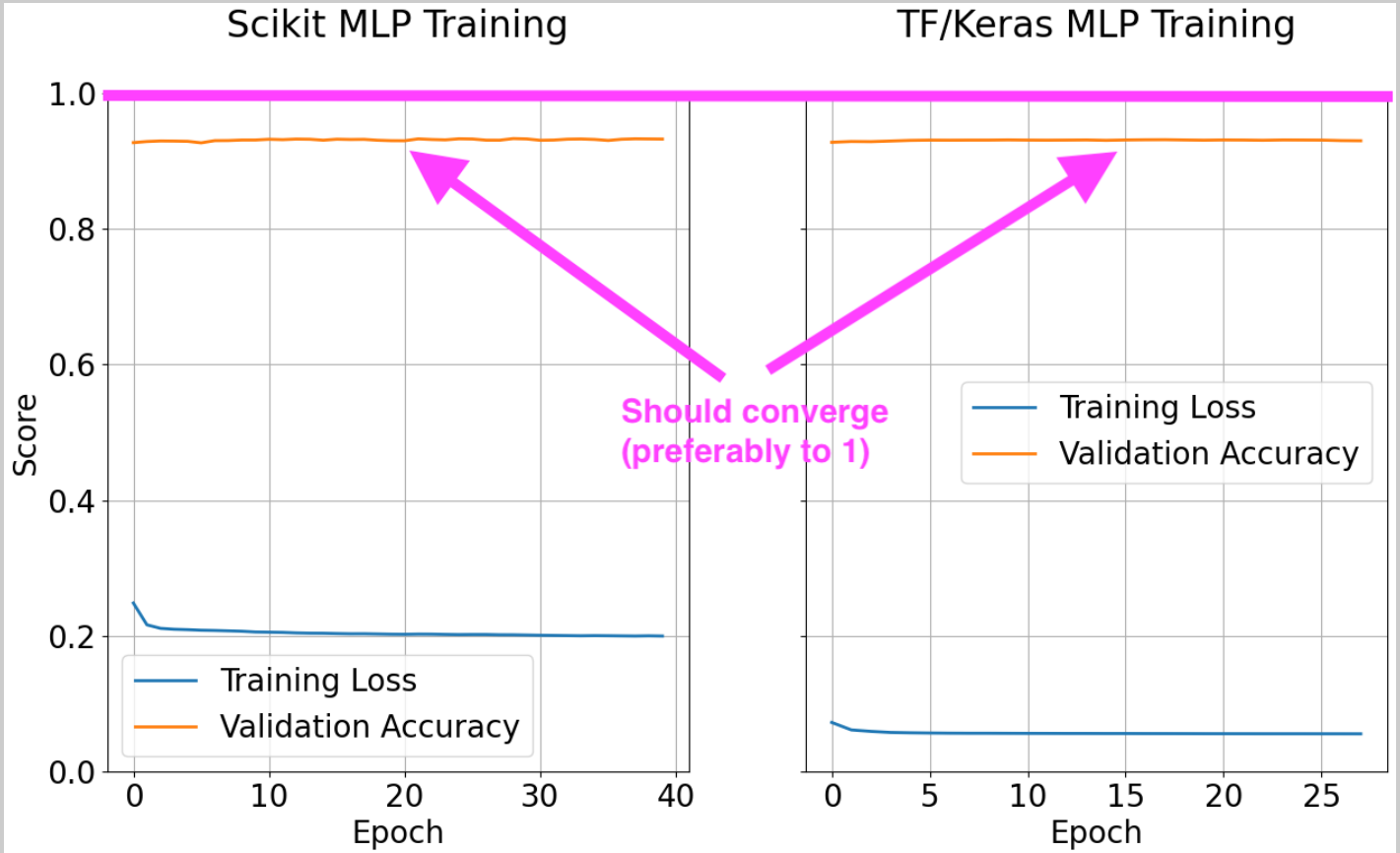
- **NOTE(S):**
  - Most settings (e.g. architecture, loss,..) are similar for both networks
  - Did not tune individual parameters
- Both networks show similar accuracy

Inspecting the learning Curves



- **NOTE(S):**
  - Most settings (e.g. architecture, loss,..) are similar for both networks
  - Did not tune individual parameters
- Both networks show similar accuracy

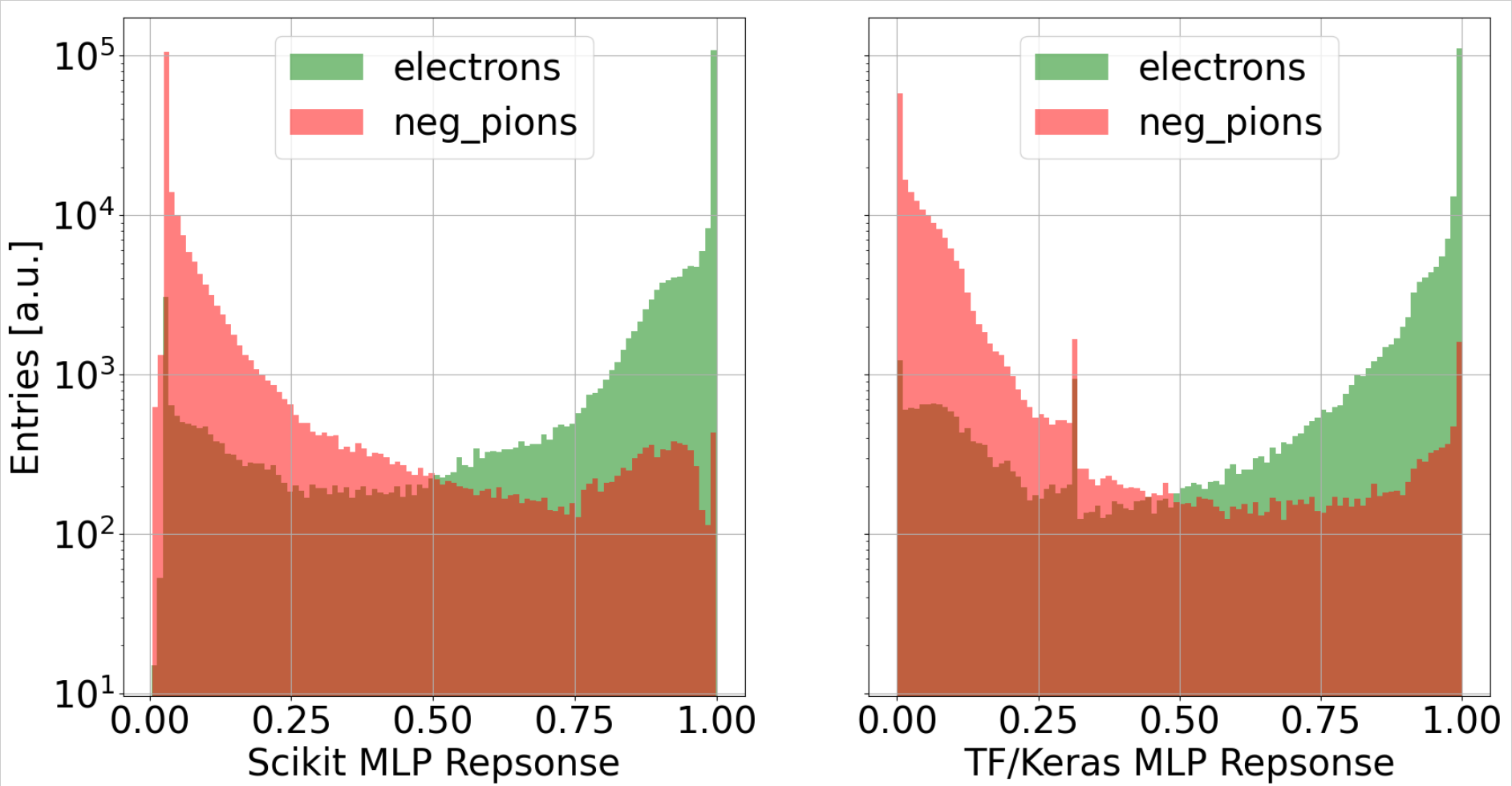
Inspecting the learning Curves



- **NOTE(S):**
  - Most settings (e.g. architecture, loss,..) are similar for both networks
  - Did not tune individual parameters
- Both networks show similar accuracy

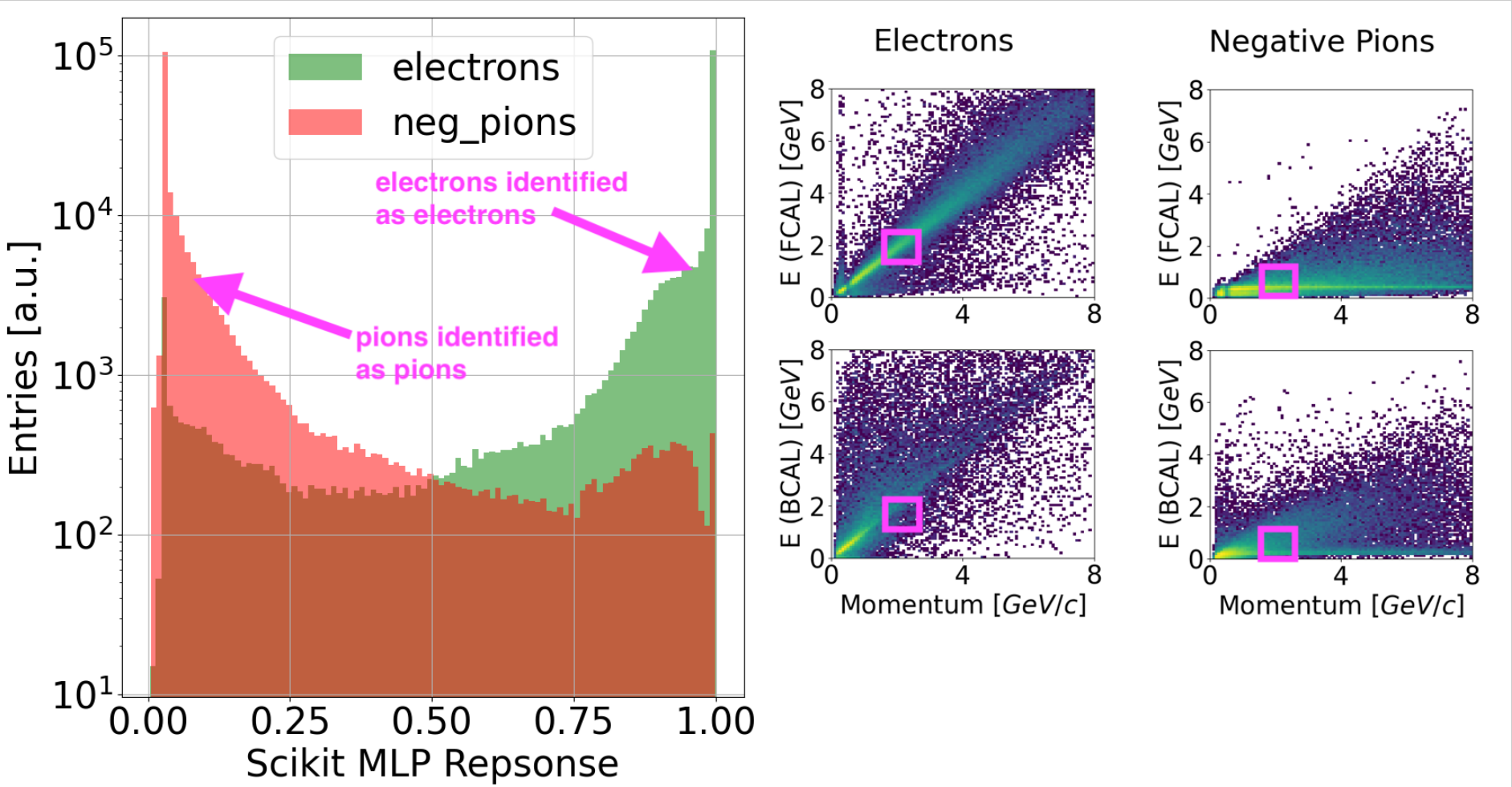
Network Responses

One of the very first things to check!



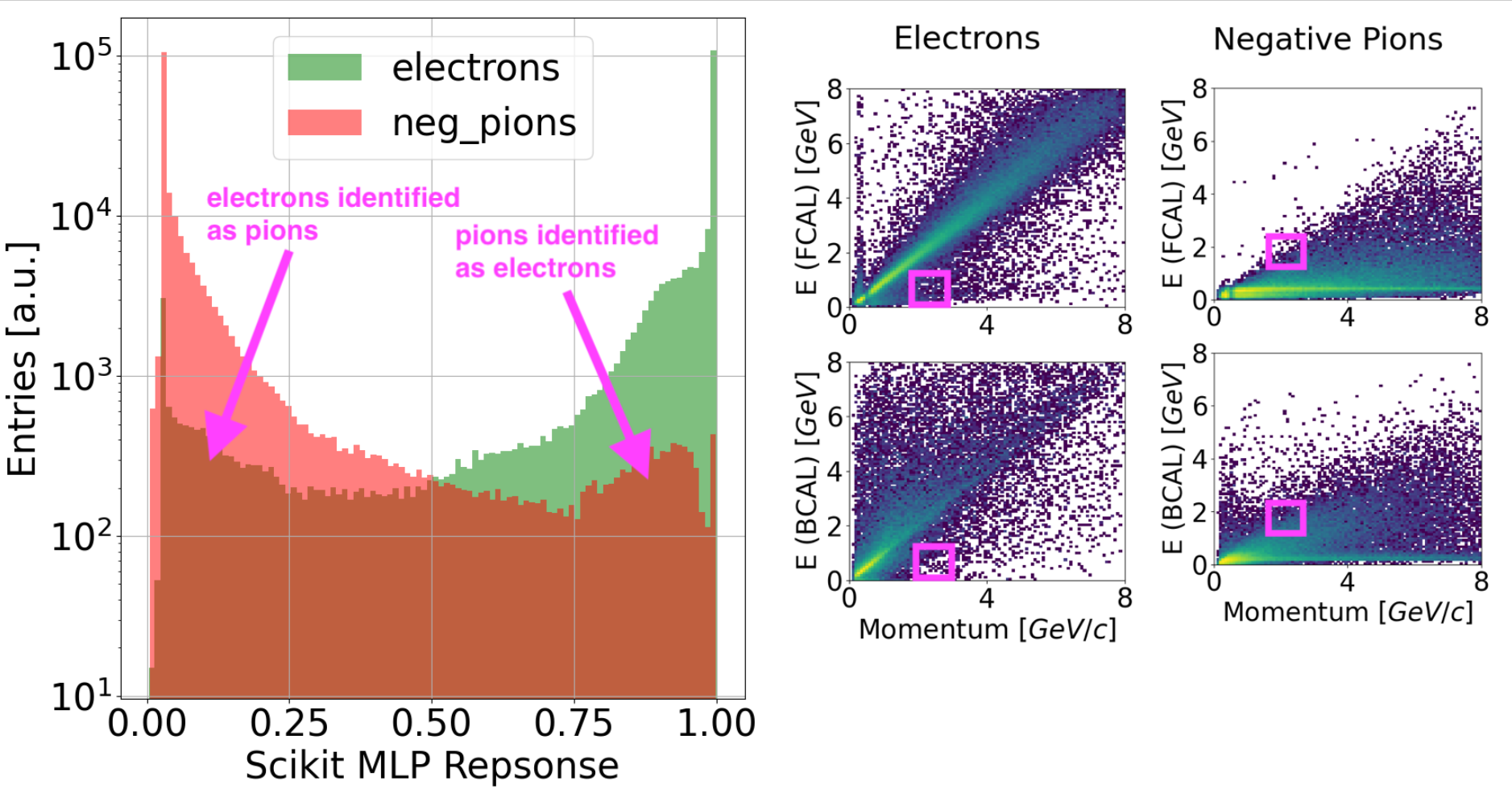
Network Responses

One of the very first things to check!



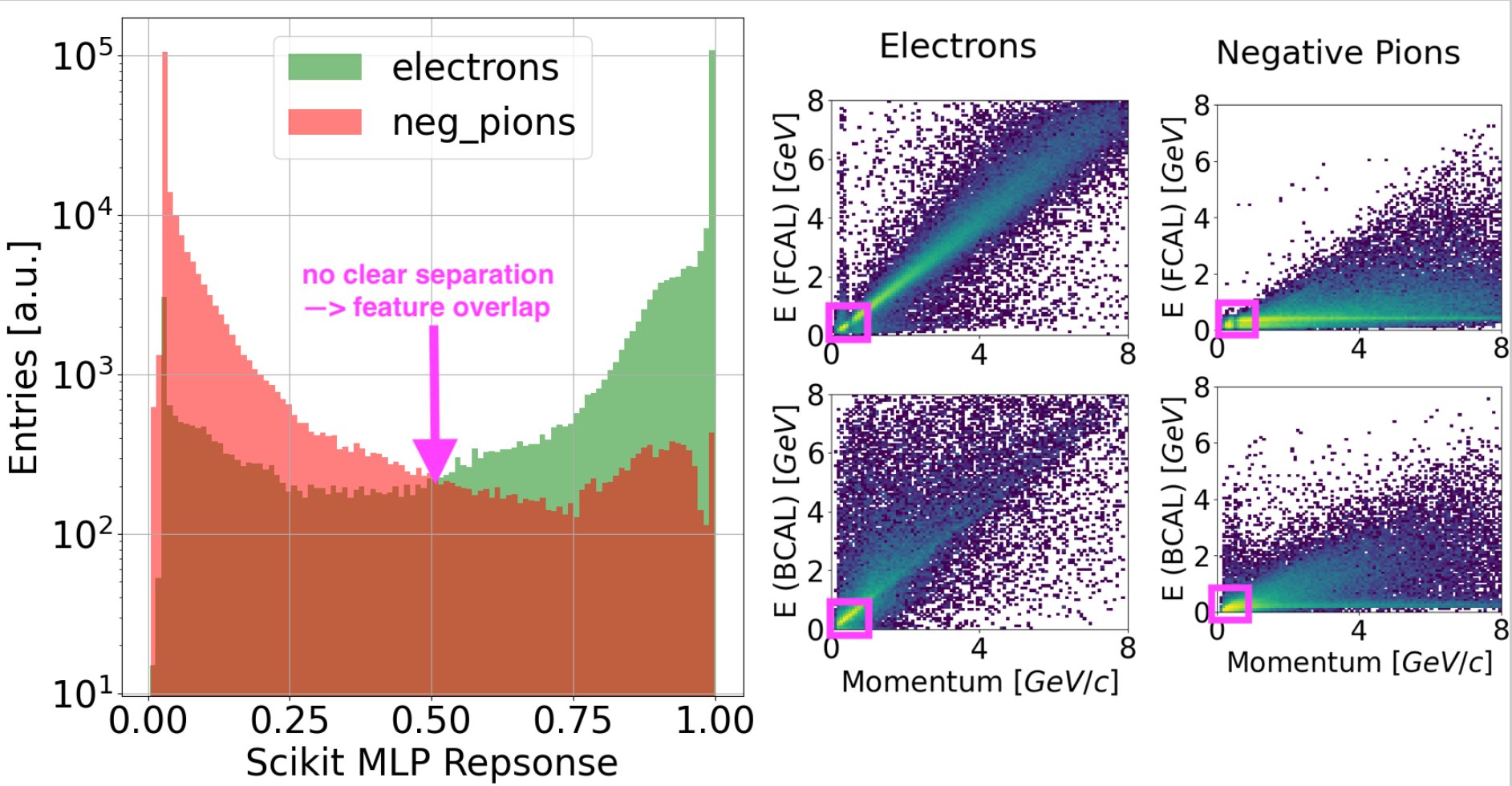
Network Responses

One of the very first things to check!



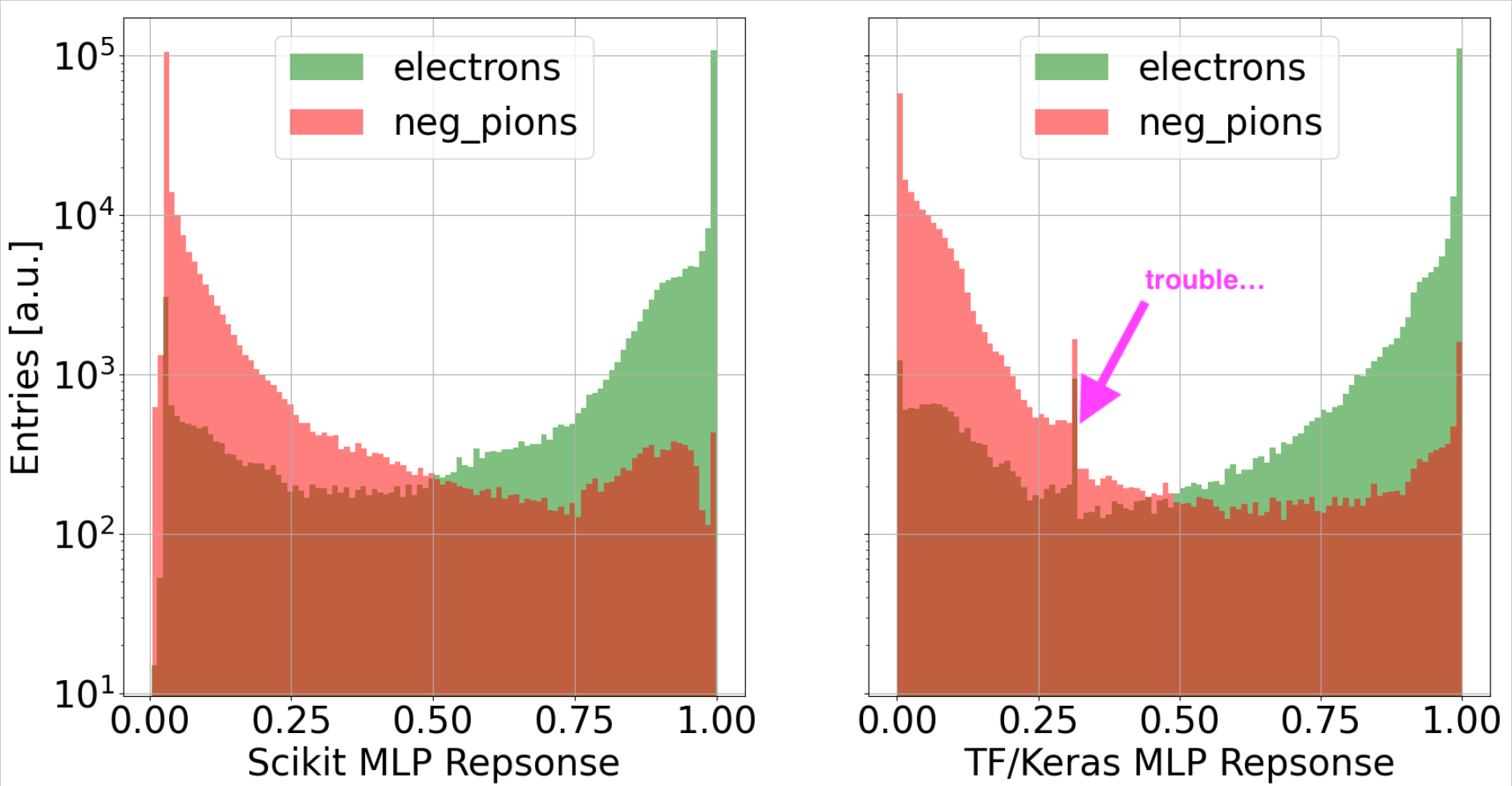
Network Responses

One of the very first things to check!

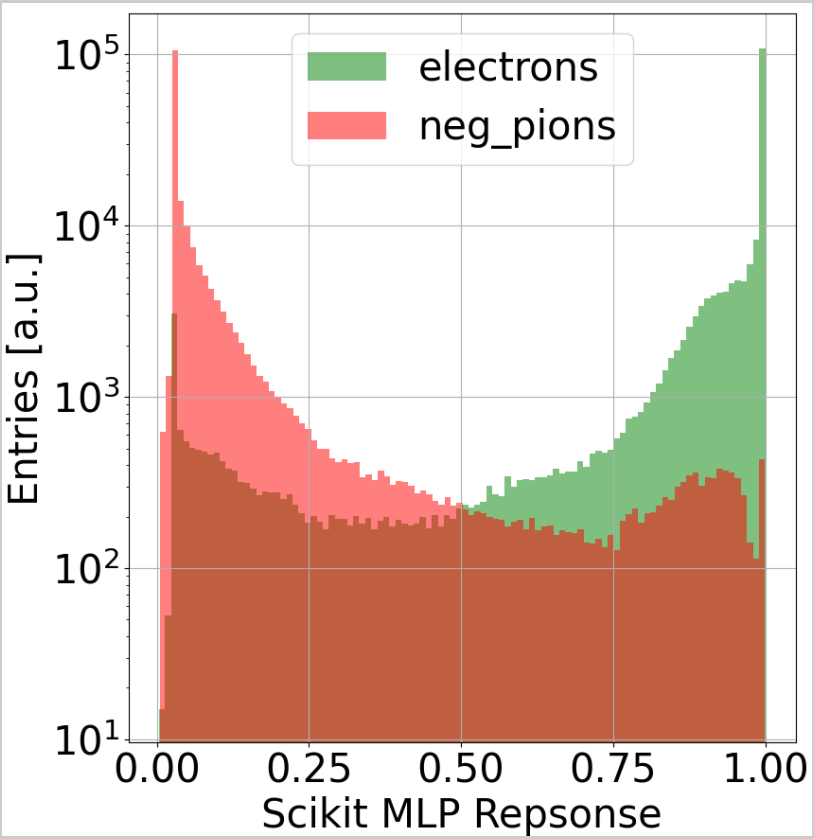


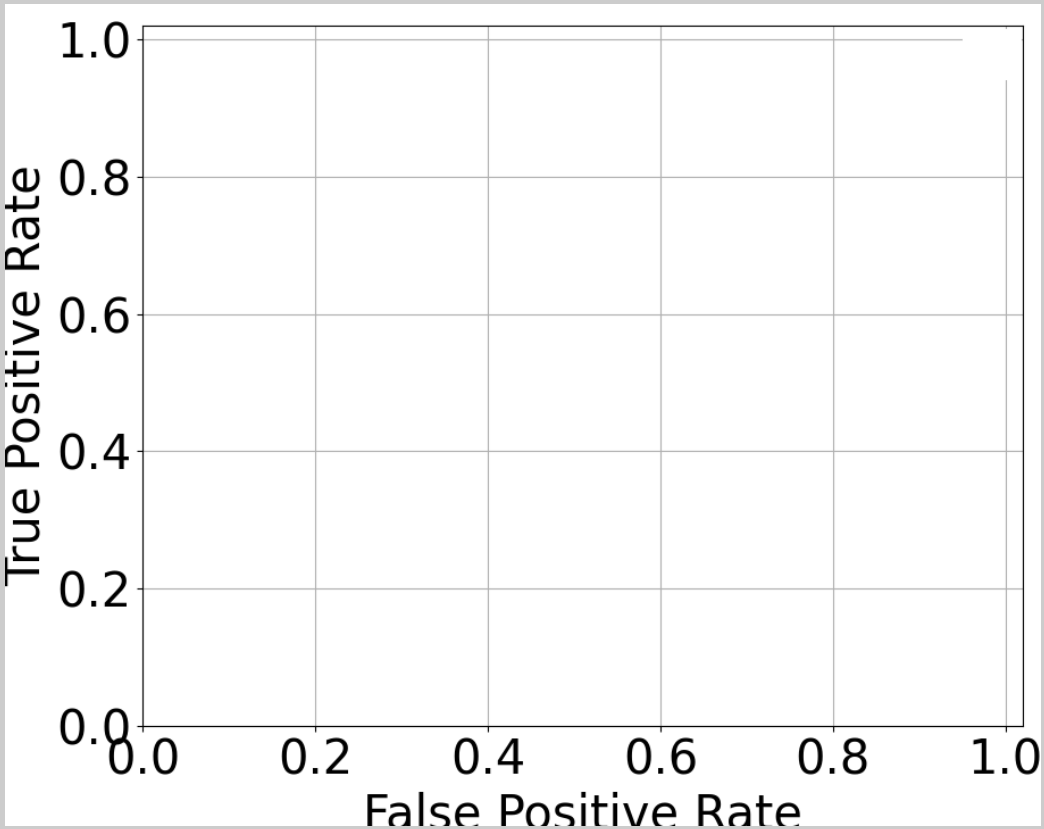
Network Responses

One of the very first things to check!



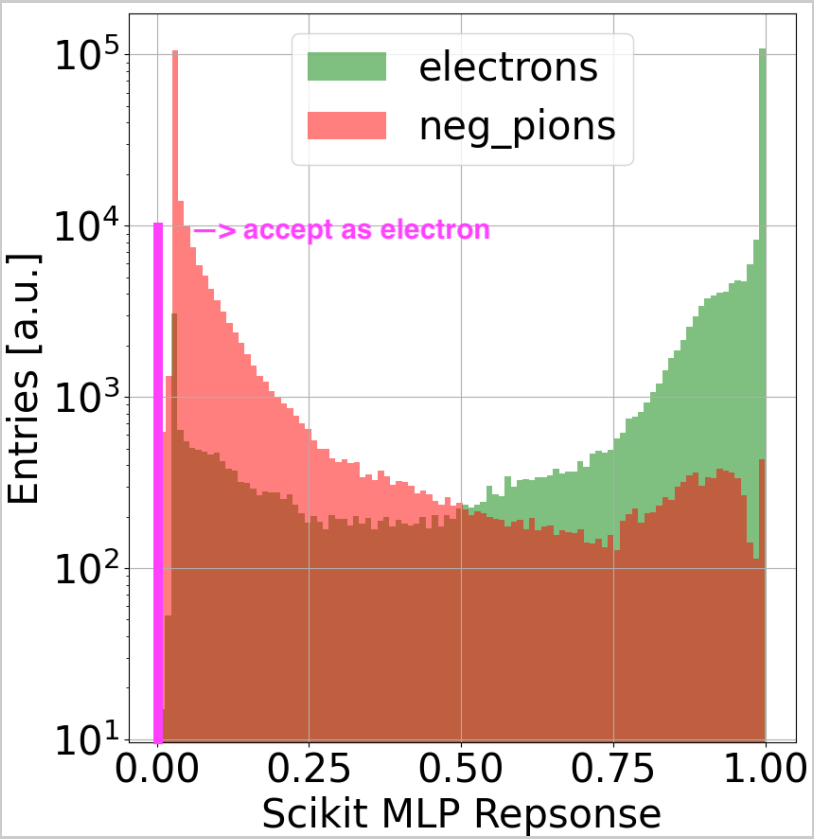
Threshold Scans

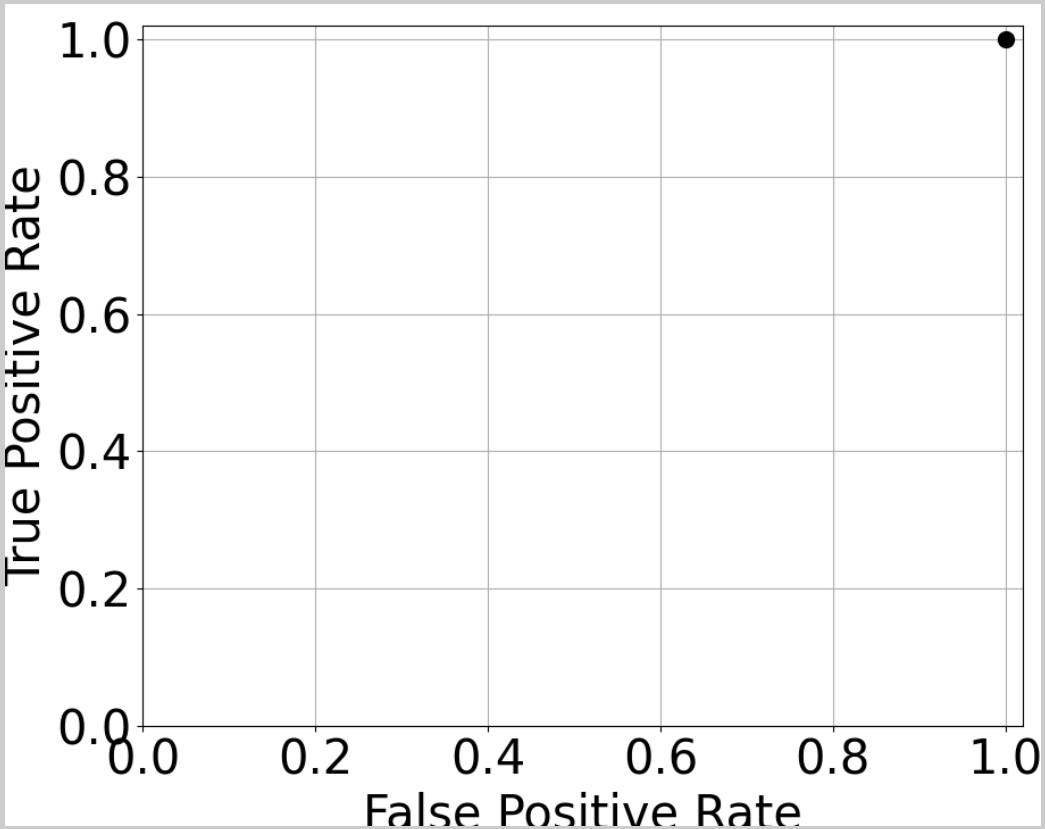




- Scan MLP responds distribution
- Calculate TPR / FPR and compare
- Find optimum threshold

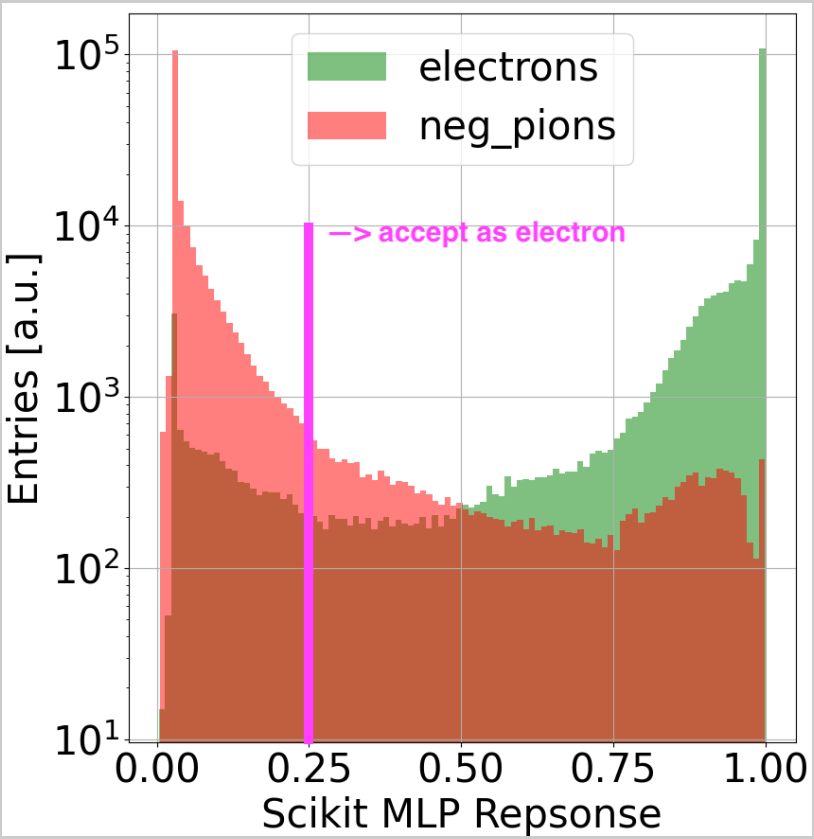
Threshold Scans

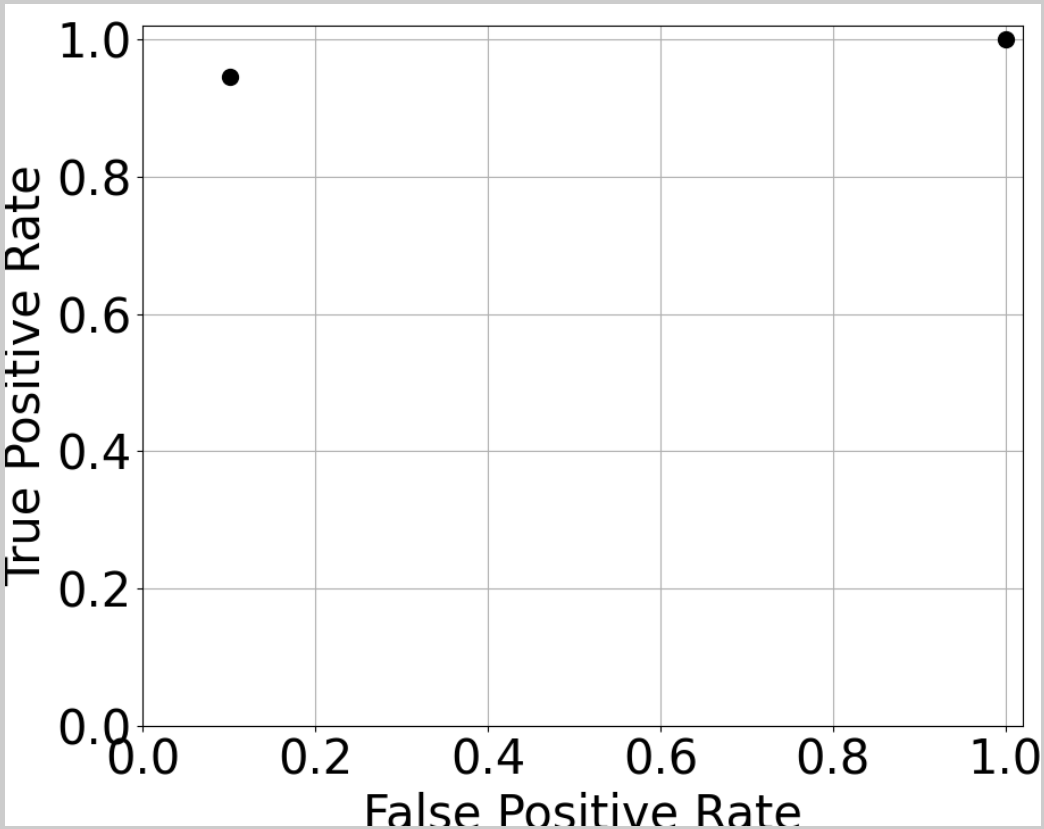




- Scan MLP responds distribution
- Calculate TPR / FPR and compare
- Find optimum threshold

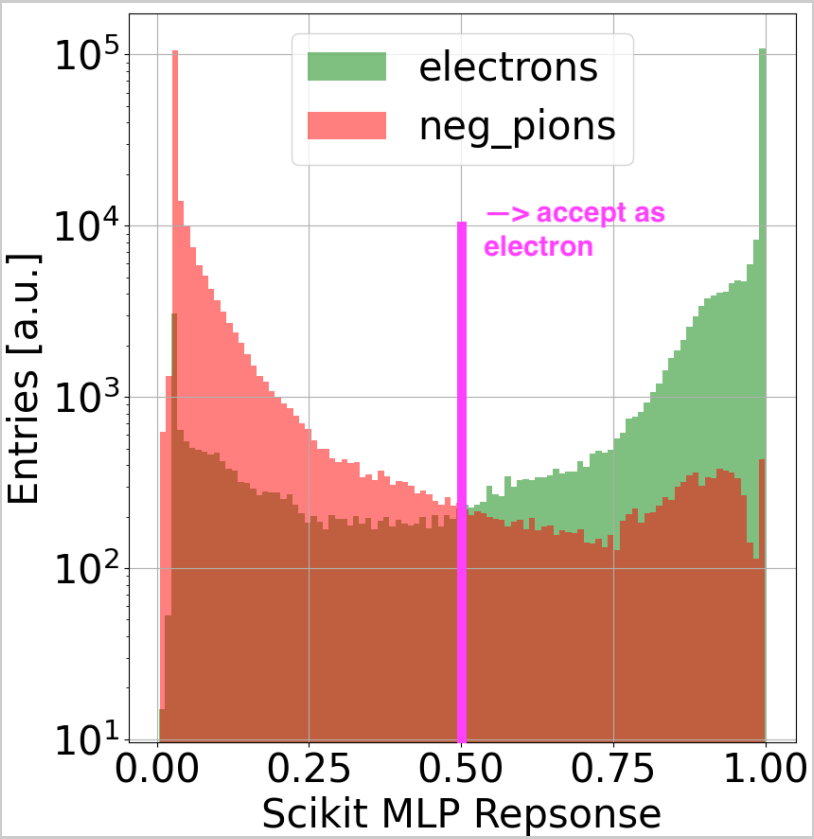
Threshold Scans

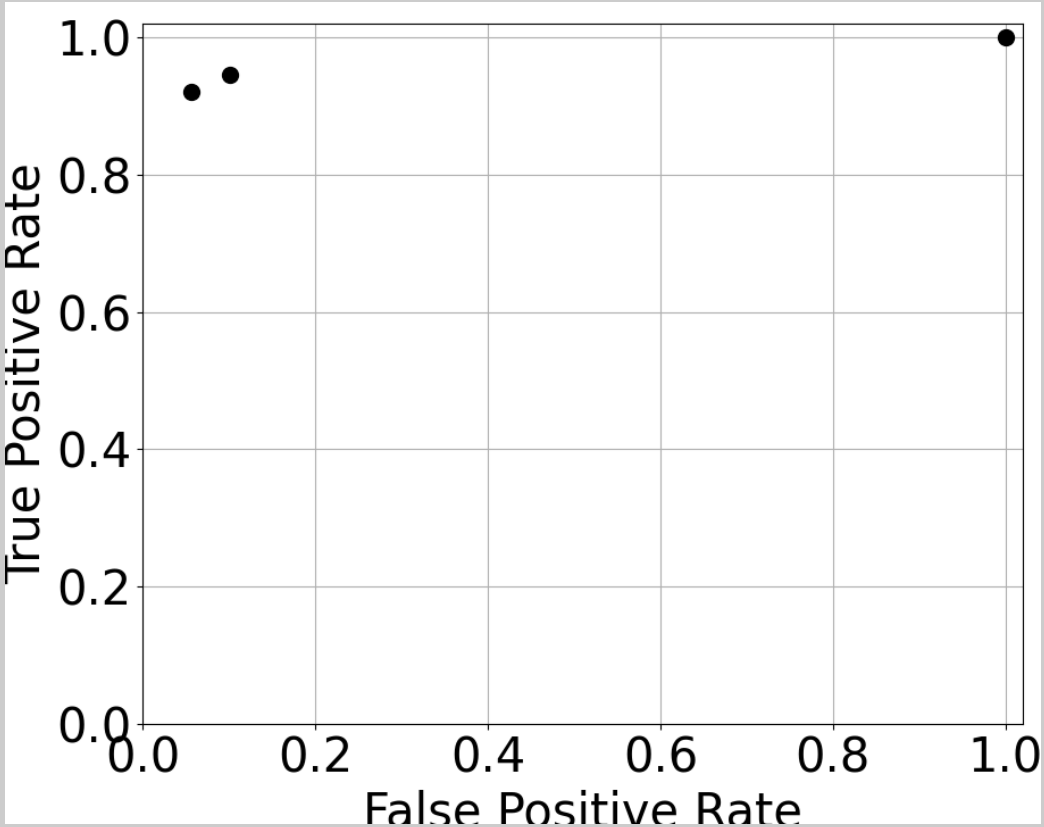




- Scan MLP responds distribution
- Calculate TPR / FPR and compare
- Find optimum threshold

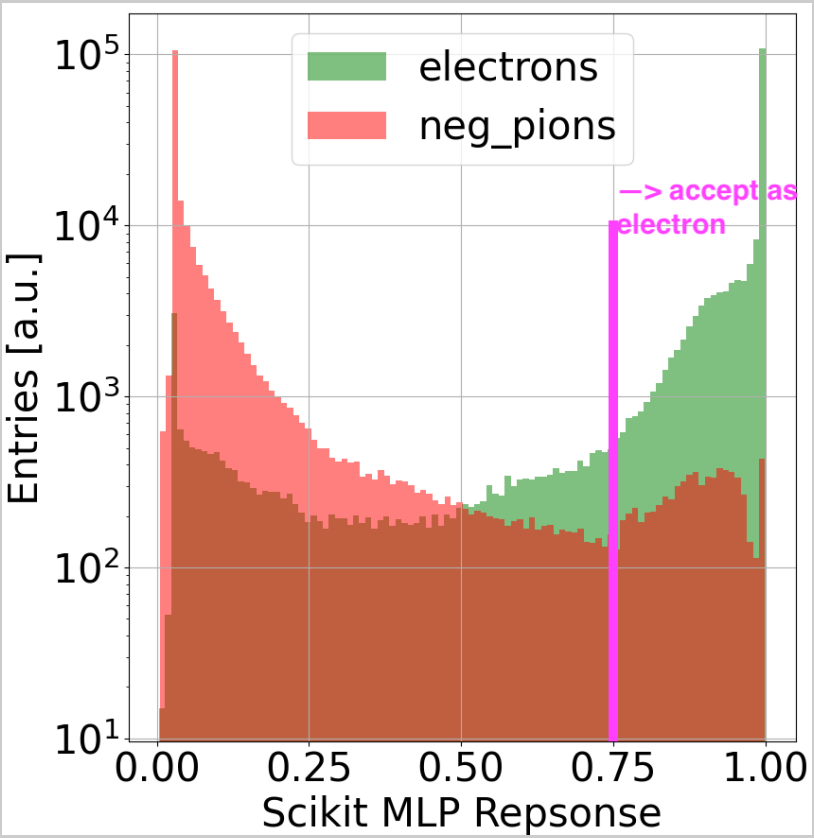
Threshold Scans

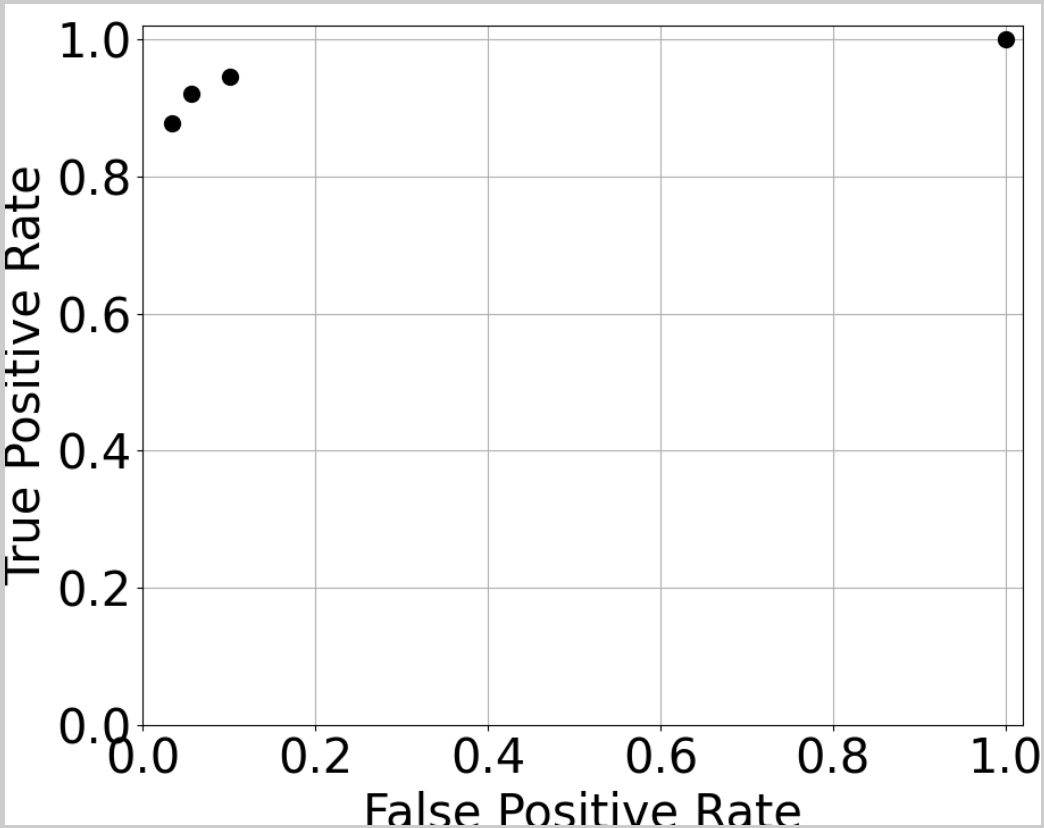




- Scan MLP responds distribution
- Calculate TPR / FPR and compare
- Find optimum threshold

Threshold Scans



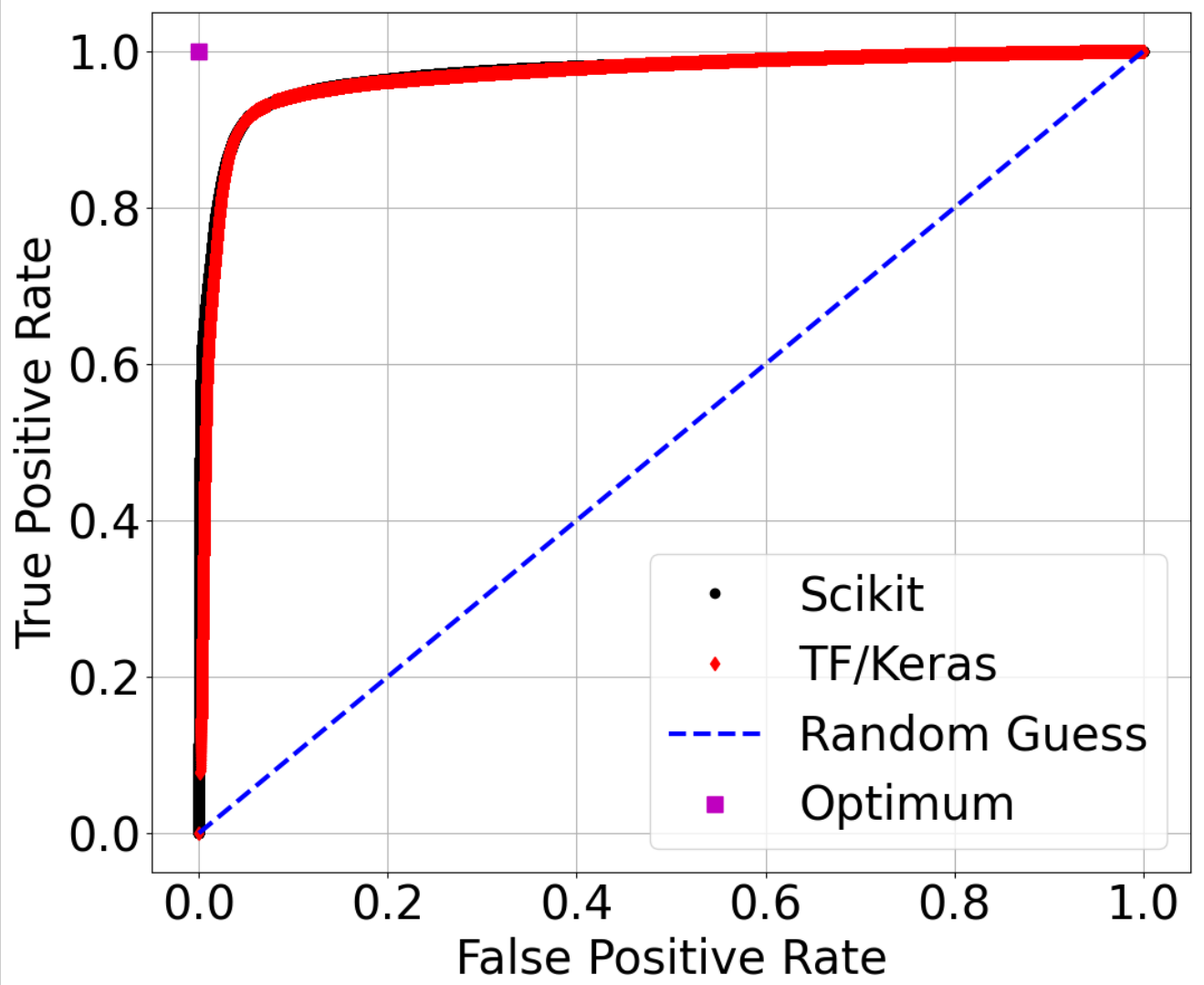


- Scan MLP responds distribution
- Calculate TPR / FPR and compare
- Find optimum threshold
- **ROC- (Receiver Operating Characteristic) curve**

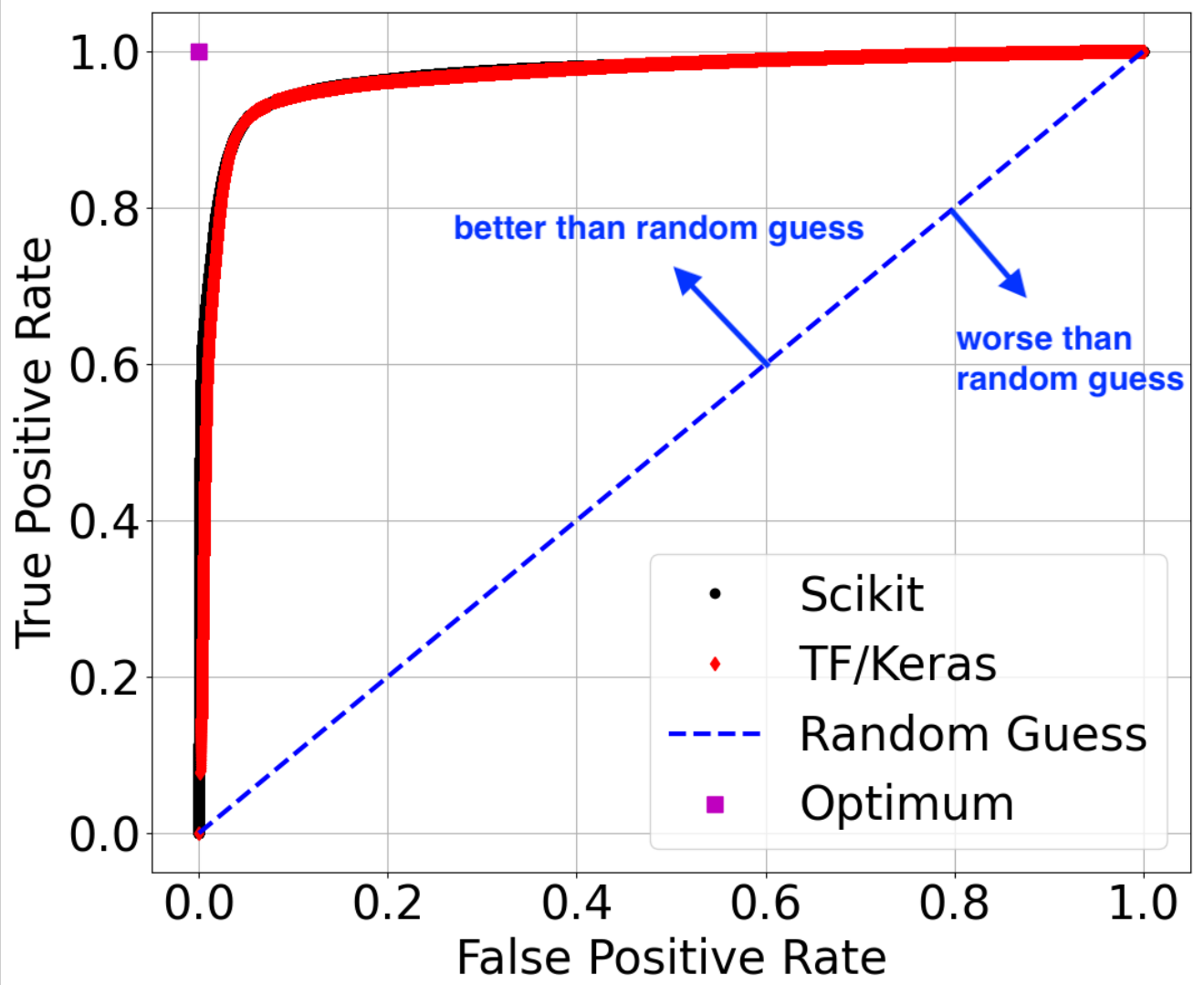
The ROC-Curve

```
from sklearn.metrics import roc_curve

#Get the network response:
scikit_response = scikit_mlp.predict_proba(X)[:,1]
keras_response = keras_mlp.predict(X)

#Get the roc-curve:
fpr_scikit, tpr_scikit,_ = roc_curve(labels,scikit_response,pos_label=1)
fpr_keras, tpr_keras,_ = roc_curve(labels,kreas_response,pos_label=1)
```

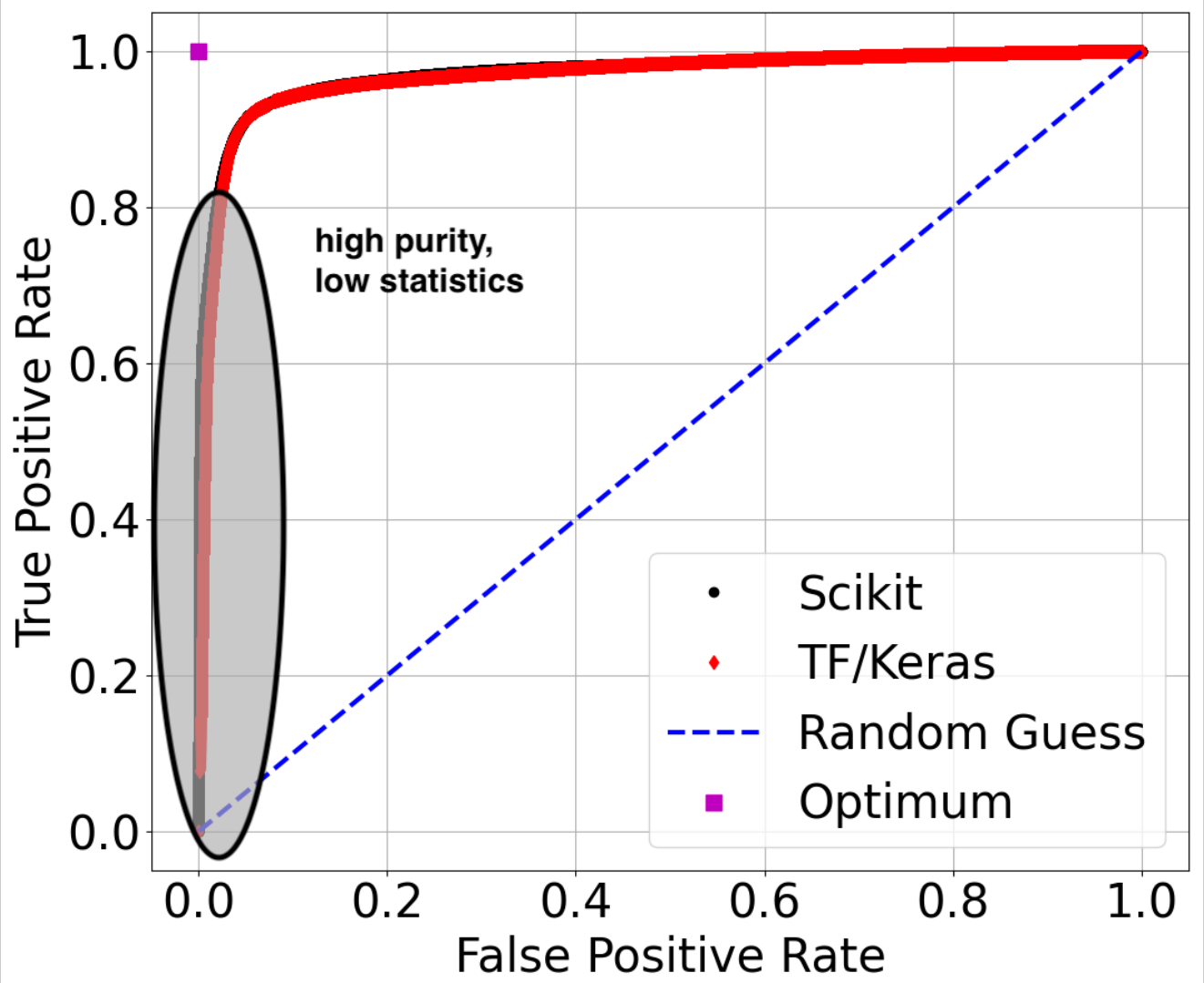
The ROC-Curve



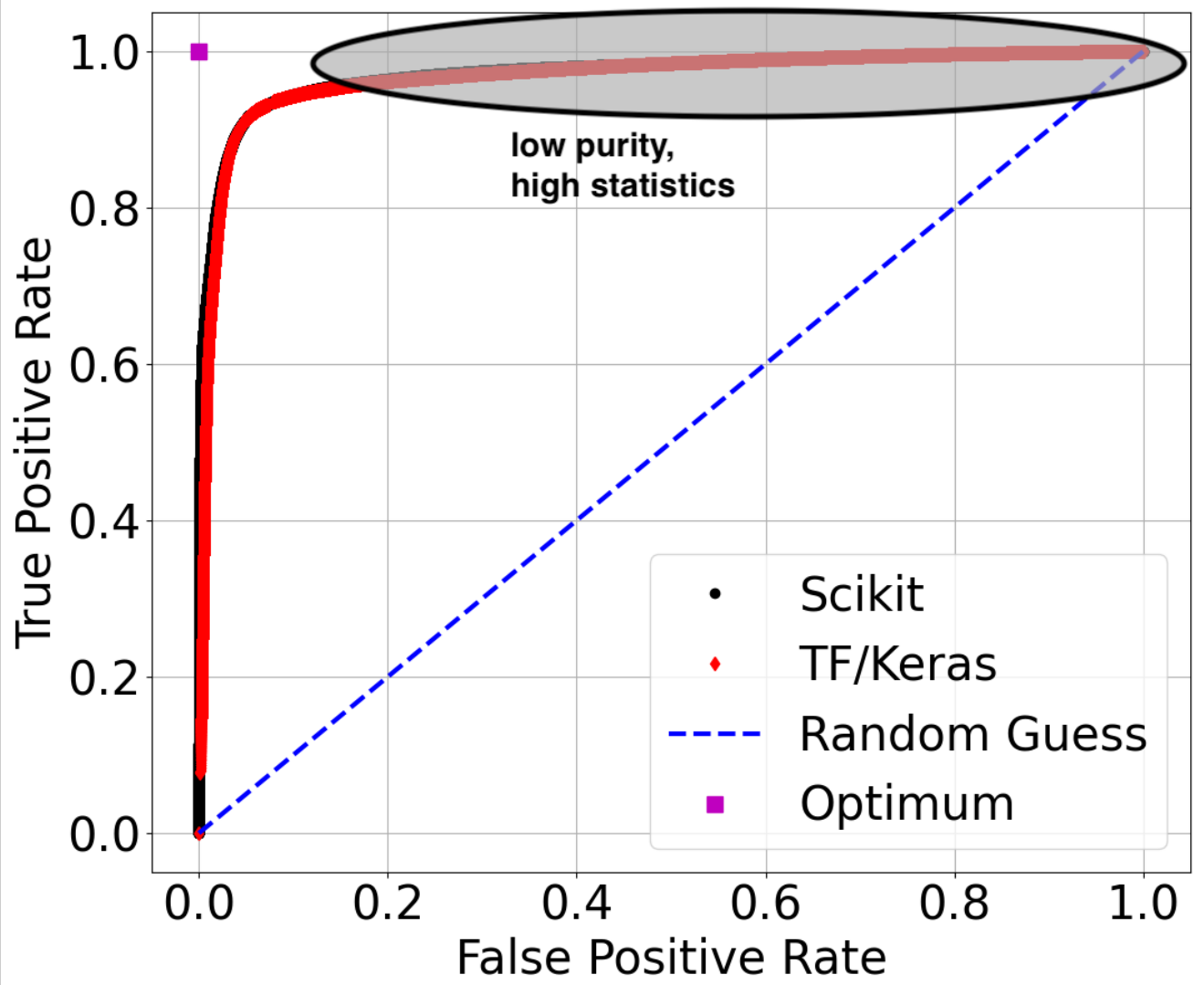
The ROC-Curve



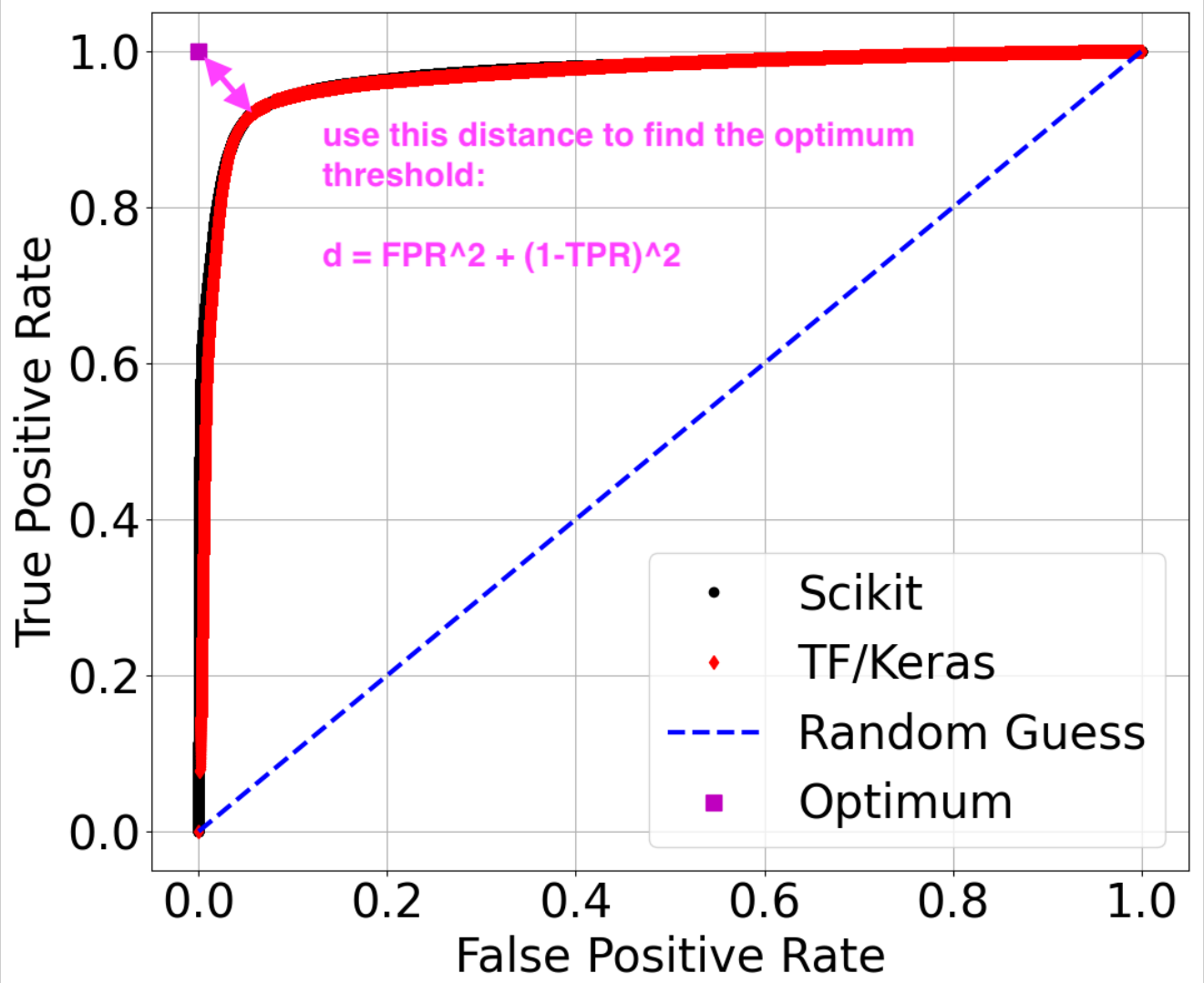
The ROC-Curve



The ROC-Curve



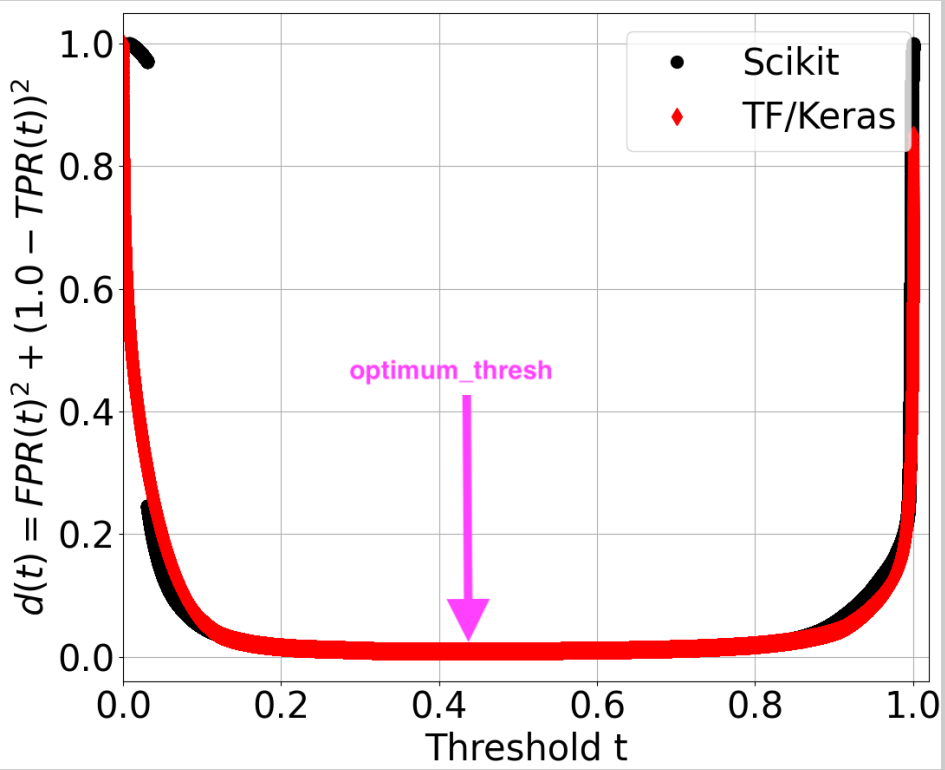
The ROC-Curve



Optimum Thresholds and Labels

- Internally handled for scikit mlp
- Need to find threshold for keras mlp

```
#Labels from scikit:
scikit_labels = scikit_mlp.predict(X)

#Labels from keras:
keras_labels = np.where(keras_response>=optimum_thresh,1.0,0.0)
```



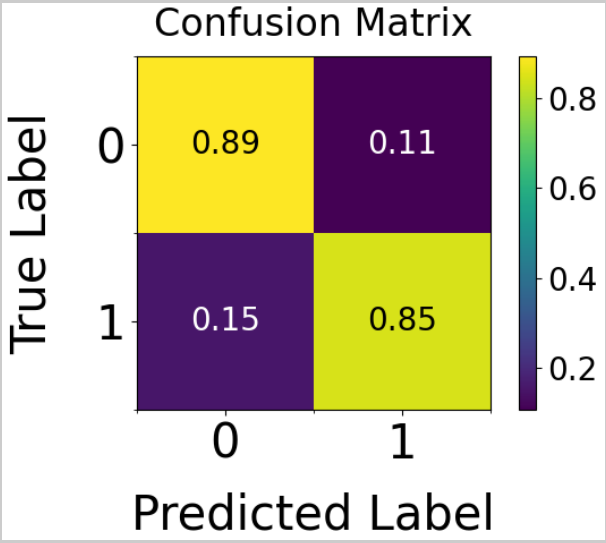
Optimum Thresholds and Labels

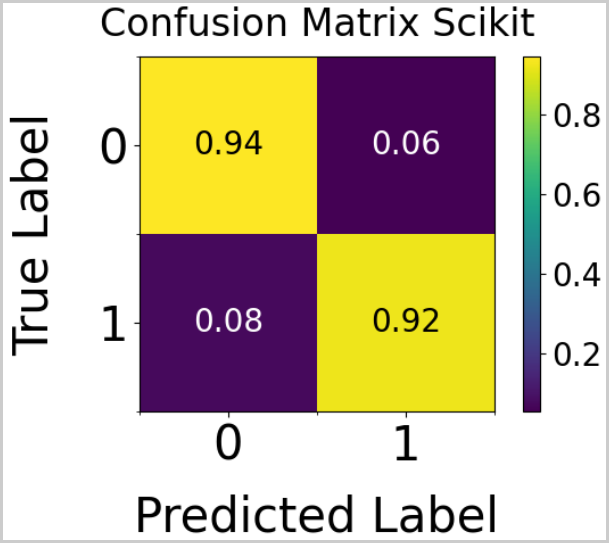
- Internally handled for scikit mlp
- Need to find threshold for keras mlp

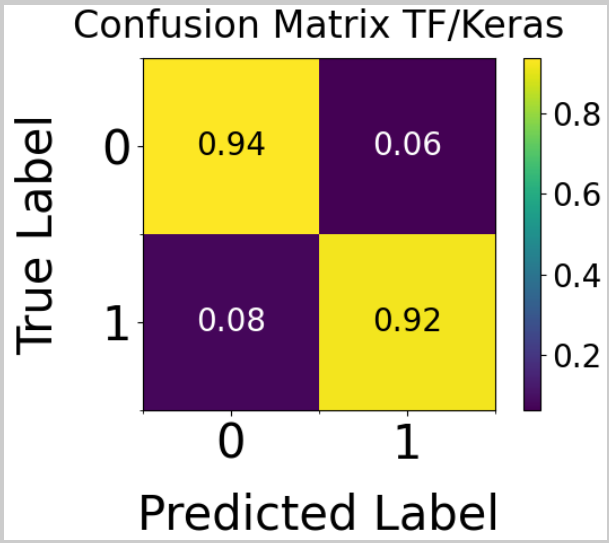
```
#Labels from scikit:
scikit_labels = scikit_mlp.predict(X)

#Labels from keras:
keras_labels = np.where(keras_response>=optimum_thresh,1.0,0.0)
```

p	theta	phi	ddEdx_CDC	dE_BCAL_preshower	...	dE_FCAL	label	scikit_prob_lep	scikit_label	keras_prob_lep	keras_label
3.925110	0.166660	2.531556	0.000002	0.025954	...	0.000000	1.0	0.031089	0.0	0.000447	0.0
1.750640	0.206904	-2.695393	0.000003	0.916215	...	0.000000	1.0	0.787939	1.0	0.830379	1.0
2.336900	0.052576	2.885235	0.000000	0.000000	...	2.173967	1.0	0.996849	1.0	0.998797	1.0
5.825788	0.710650	-2.959795	0.000002	0.925353	...	0.000000	1.0	0.996104	1.0	0.998847	1.0
2.882963	0.143638	-0.763970	0.000002	0.000000	...	1.143092	0.0	0.101160	0.0	0.091617	0.0
0.661498	0.609131	3.100280	0.000002	0.086262	...	0.000000	0.0	0.451565	0.0	0.396868	0.0
0.443637	0.031602	2.455756	0.000000	0.000000	...	0.431946	1.0	0.951643	1.0	0.987271	1.0
2.020184	0.127360	0.079234	0.000002	0.000000	...	0.214426	0.0	0.031089	0.0	0.010049	0.0
1.827218	0.024664	2.837258	0.000000	0.000000	...	0.908476	0.0	0.168505	0.0	0.200512	0.0
6.215446	0.120326	-3.040930	0.000002	0.000000	...	0.436846	0.0	0.031089	0.0	0.000063	0.0

Electron Identification Performances

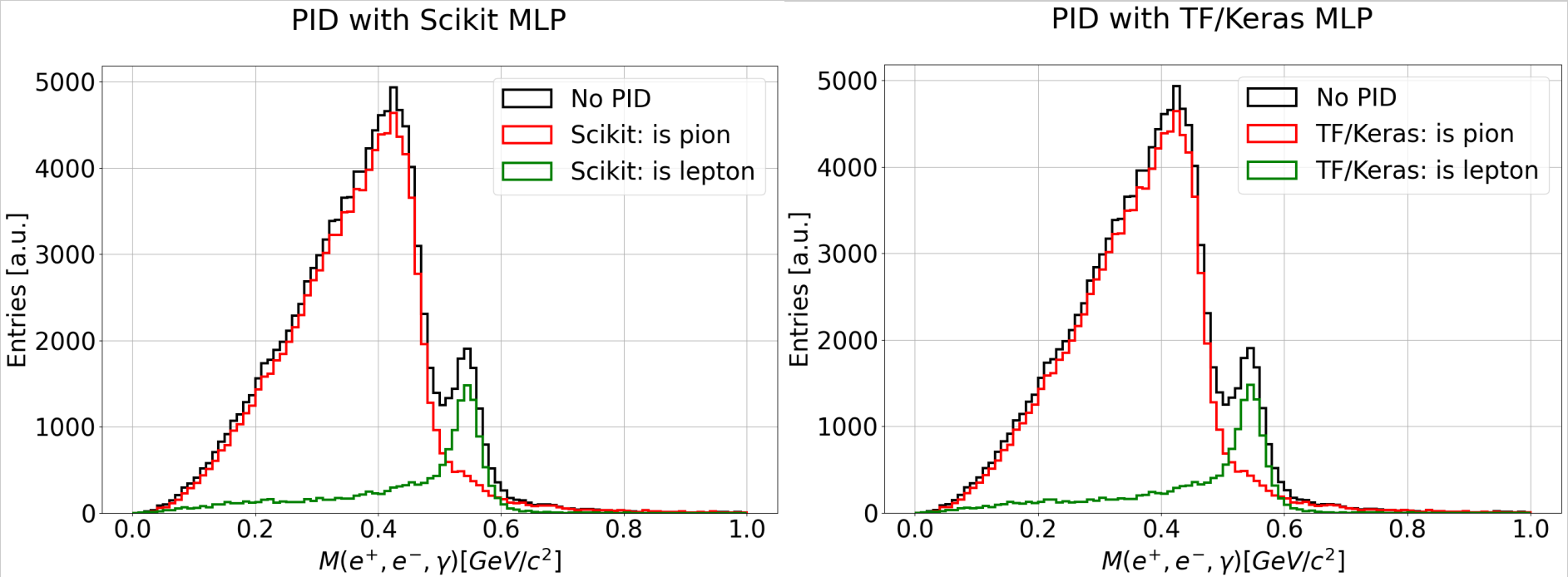






Method	TPR	FPR	min. $d(t)$	threshold t	Accuracy
Reference Analysis	0.85	0.11	na	na	0.87
Scikit MLP	0.92	0.06	0.0094	0.47	0.93
TF/Keras MLP	0.92	0.06	0.0094	0.43	0.93

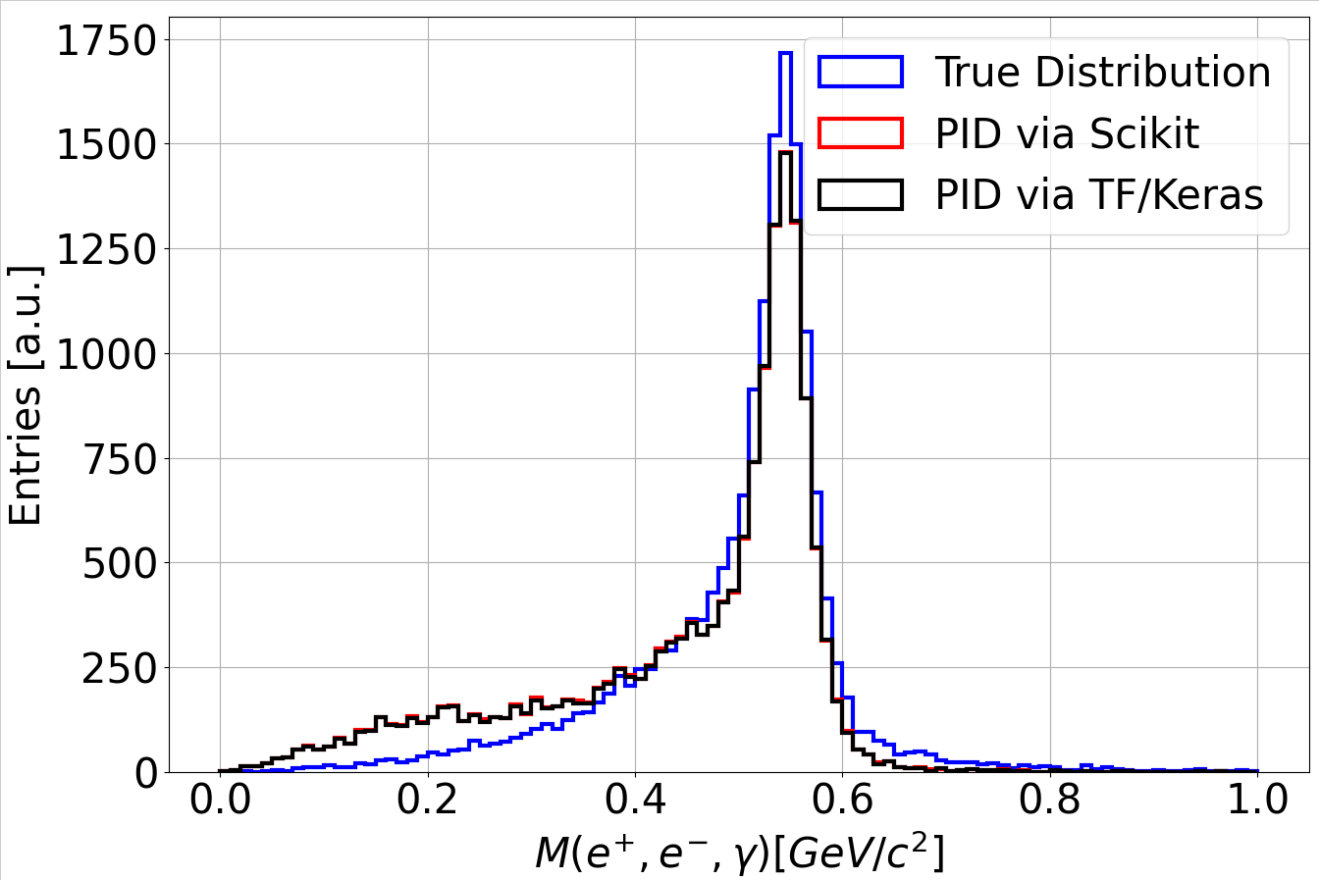
Reconstructing $\eta \to e^+e^-\gamma$ Events



- ☑ Networks separate $\eta \to \pi^+\pi^-\gamma$ from $\eta \to e^+e^-\gamma$
- ☑ $\eta \to e^+e^-\gamma$-signal shape is undisturbed (i.e. continuous and smooth)

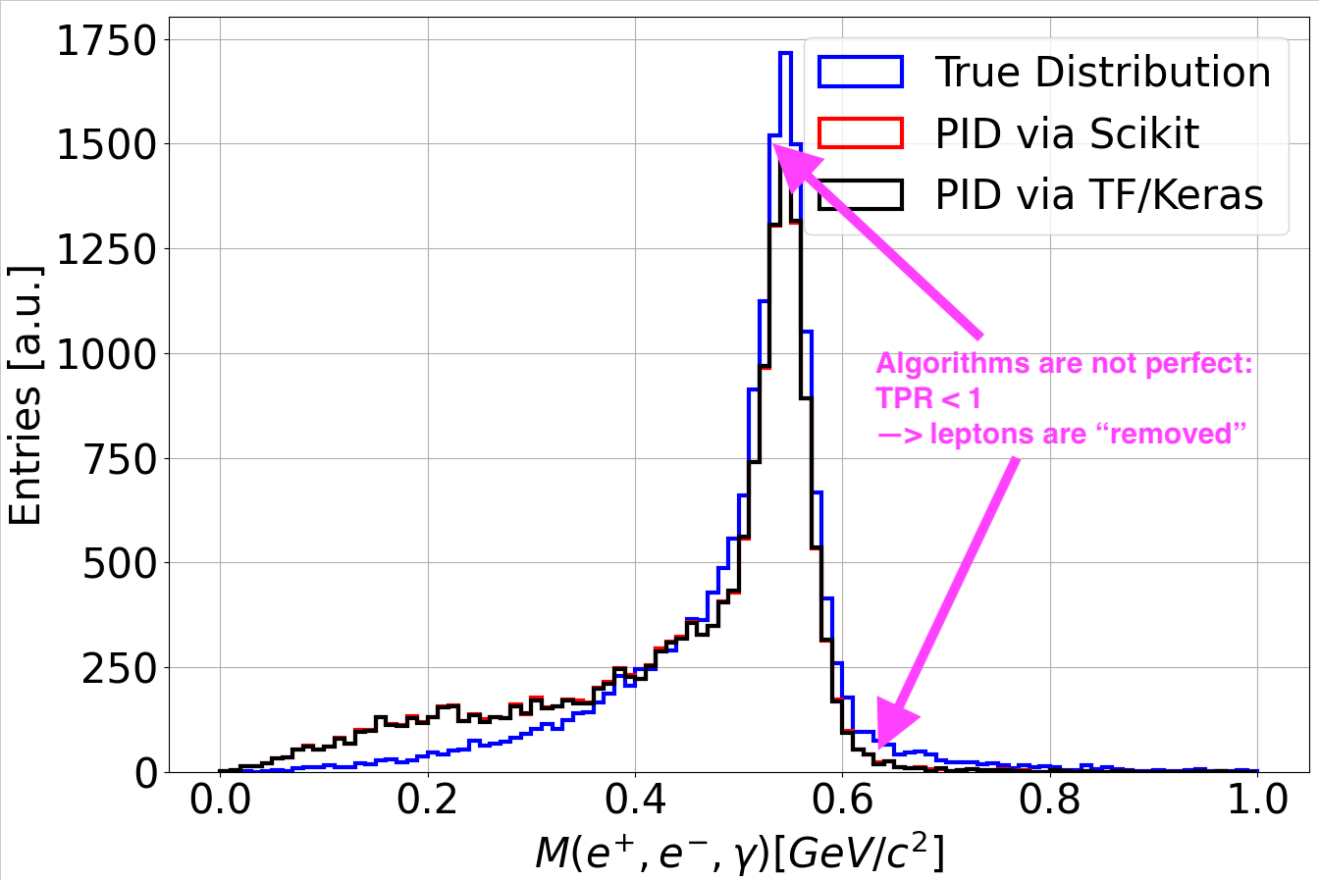
Comparing Signal Shapes

- Networks do not significantly alter signal shape
- Reconstructed shapes are identical for both networks
- **NOTE:** Not always lucky to have true distribution



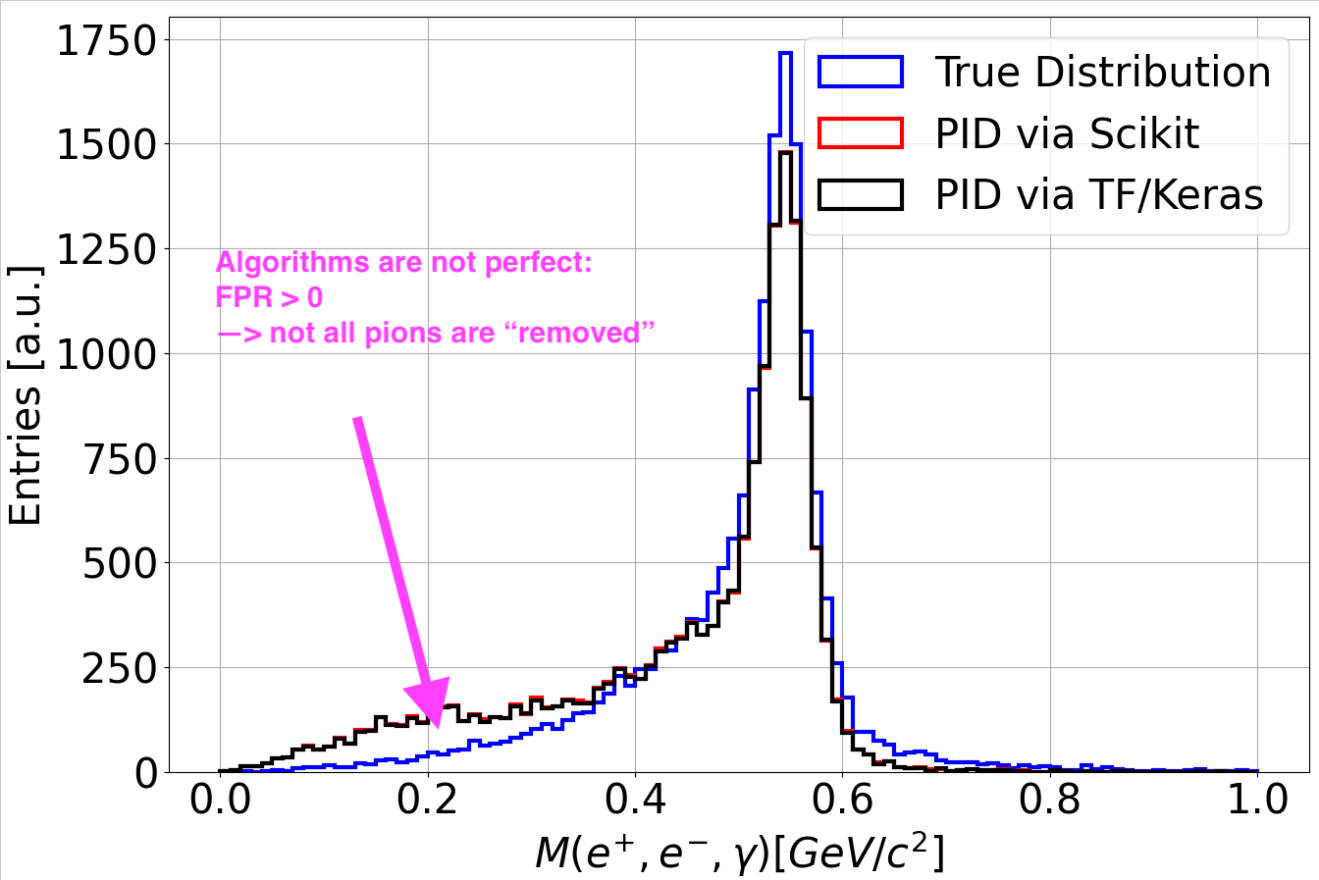
Comparing Signal Shapes

- Networks do not significantly alter signal shape
- Reconstructed shapes are identical for both networks
- **NOTE:** Not always lucky to have true distribution



Comparing Signal Shapes

- Networks do not significantly alter signal shape
- Reconstructed shapes are identical for both networks
- **NOTE:** Not always lucky to have true distribution



Where to go from here?

- Neural network classifiers show "reasonable" performance
  (Accuracy = 93% $\leftrightarrow$ current benchmark)
- **Legit question:** Can one do better?
- **Answer(s):**
  - Hopefully yes
  - Try different models and compare
    (i.e. random forest, Gaussian processes, likelihood estimator,...)
  - Tune network parameters $\rightarrow$ Hyper Parameter Optimization (HPO)

Hyper Parameters

- **Fit Parameters:** Model internal parameters → Set by optimization procedure → driven by data
- **Hyper Parameters:** Determine model architecture / performance → Set by user

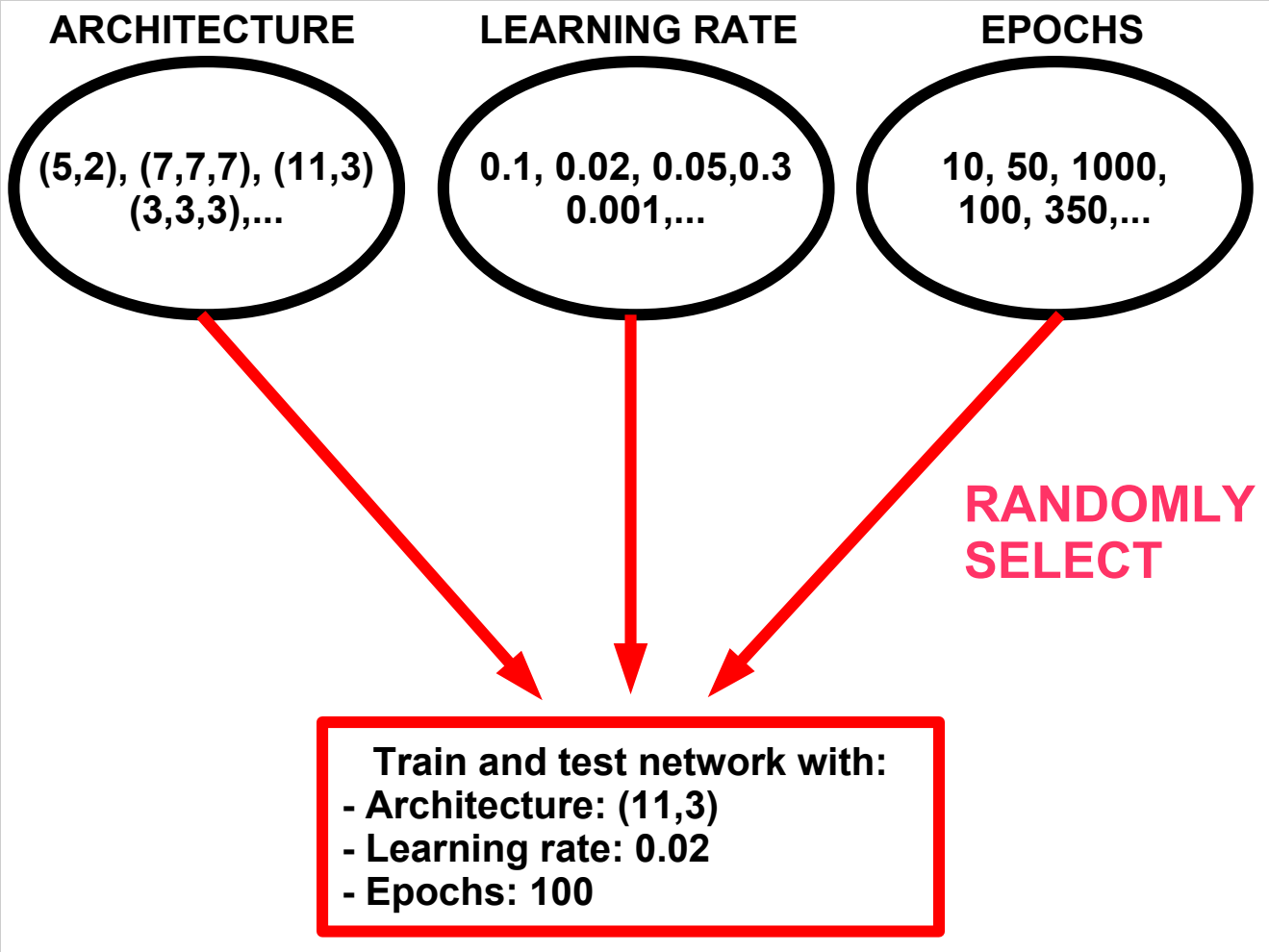
Model	Fit Parameters	Hyper Parameters
$pol(N) = p_N x^N + p_{N-1} x^{N-1} + ... + p_0$	$p_N, .., p_0$	N, minimizer,...
Multilayer Perceptron	weights, biases	#Hidden Layers, #Neurons, #Epochs,...

Hyper Parameter Optimization (HPO)

- **Goal:** Find set of hyper parameters which maximizes the prediction performance $\leftrightarrow$ Minimize prediction error
- **To consider:**
  - Use computational time efficiently
  - Generalizability $\leftrightarrow$ avoid overfitting $\rightarrow$ Use validation data!
  - Actually find optimum
- **How to: HPO?**
  - Test different settings manually $\rightarrow$ can be painful
  - Use algorithm
    - ★ Grid search: simple, but ineffective if parameter ranges are unknown $\#\text{Searches} = \prod_i \#\text{Settings[Parameter(i)]}$
    - ★ Random parameter search
    - ★ Bayesian optimization
    - ★ and many more...

Random Hyper Parameter Search

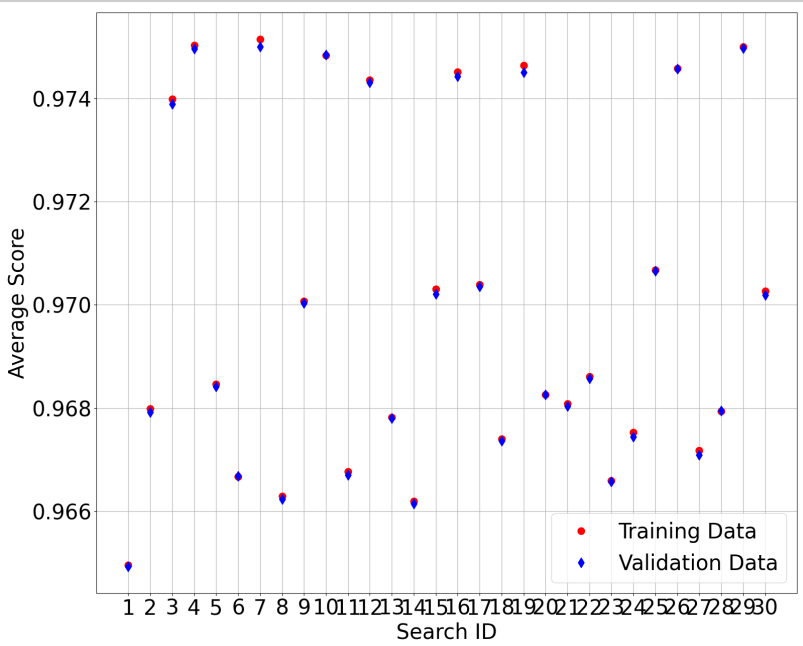
- Draw random parameter samples
- Use training and test data
- Select parameter sample with highest score on test data

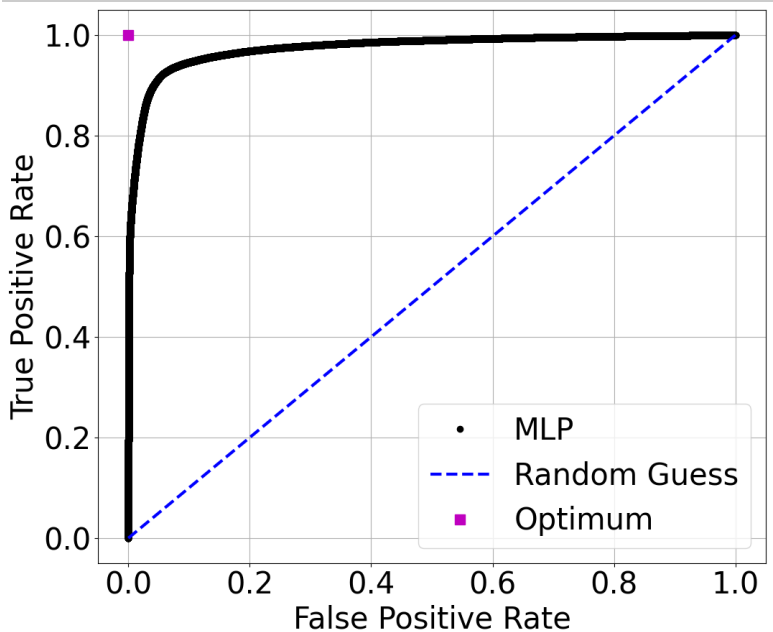


Random HPO in scikit

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.neural_network import MLPClassifier

#Which parameters do you want to tune?
parameters_to_tune_mlp = {
     'hidden_layer_sizes': [(5),(8,3),(3,3,3),...], #--> Architecture
     'learning_rate_init': stats.uniform(0.001,0.1), #--> Learning rate
     'max_iter': stats.randint(100,800) #Epochs
     #(number of training iterations
}

#Set up the search algorithm:
random_search = RandomizedSearchCV(
      MLPClassifier(), #--> The object you want to tune
      param_distributions=parameters_to_tune_mlp,
      n_iter=30, #--> Perform 30 random searches
      scoring='roc_auc' #--> Integral under the ROC-curve
 )

 #Run the search
 random_search.fit(X,Y)
```

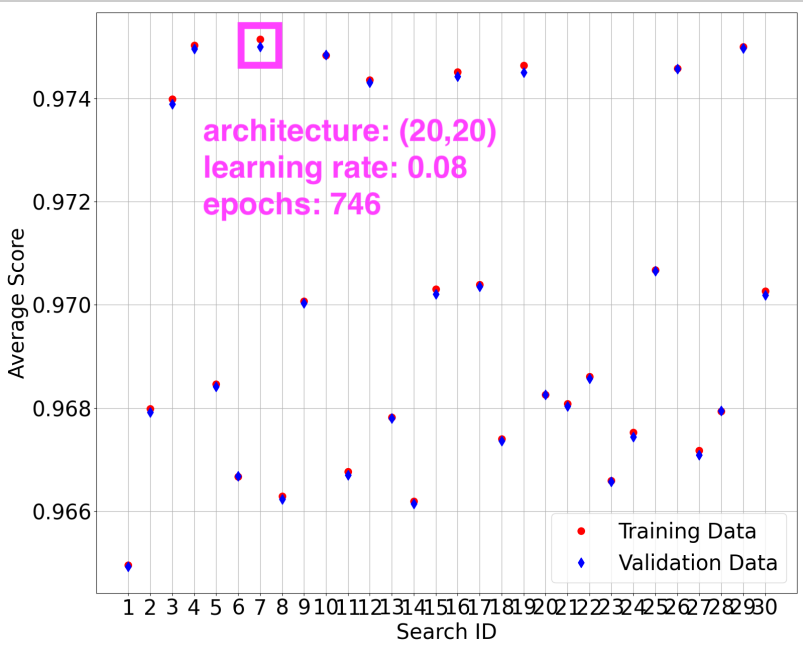
And the Winner is...

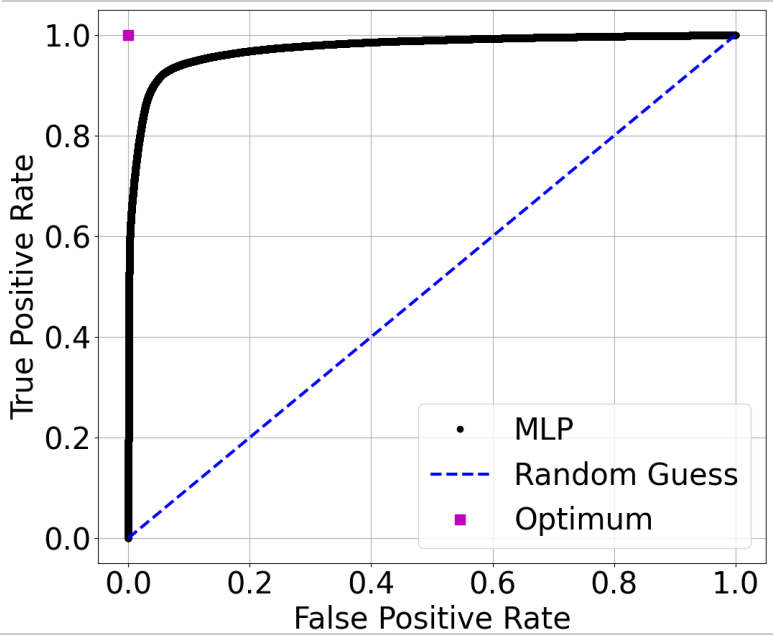




Method	TPR	FPR	min. $d(t)$	threshold t	Accuracy
Reference Analysis	0.85	0.11	na	na	0.87
Scikit MLP	0.92	0.06	0.0094	0.47	0.93
TF/Keras MLP	0.92	0.06	0.0094	0.43	0.93
Scikit MLP (rnd search)	0.93	0.06	0.0092	0.44	0.933

And the Winner is...



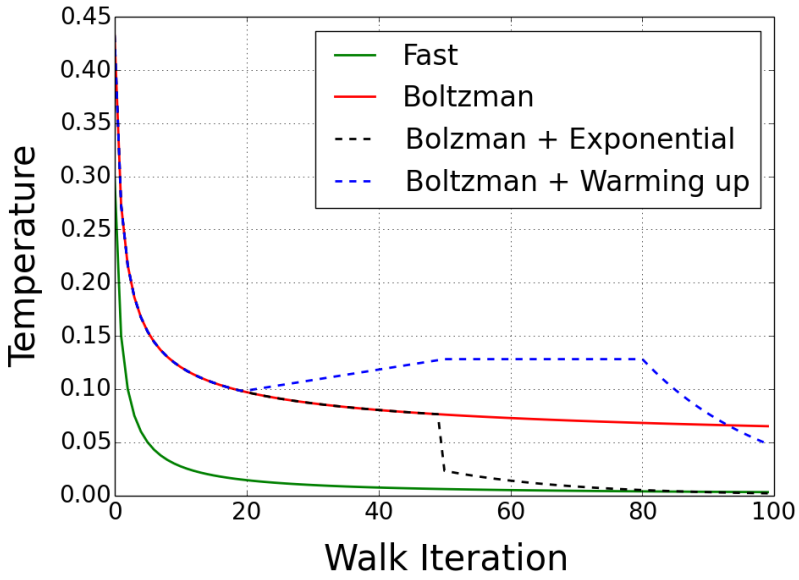


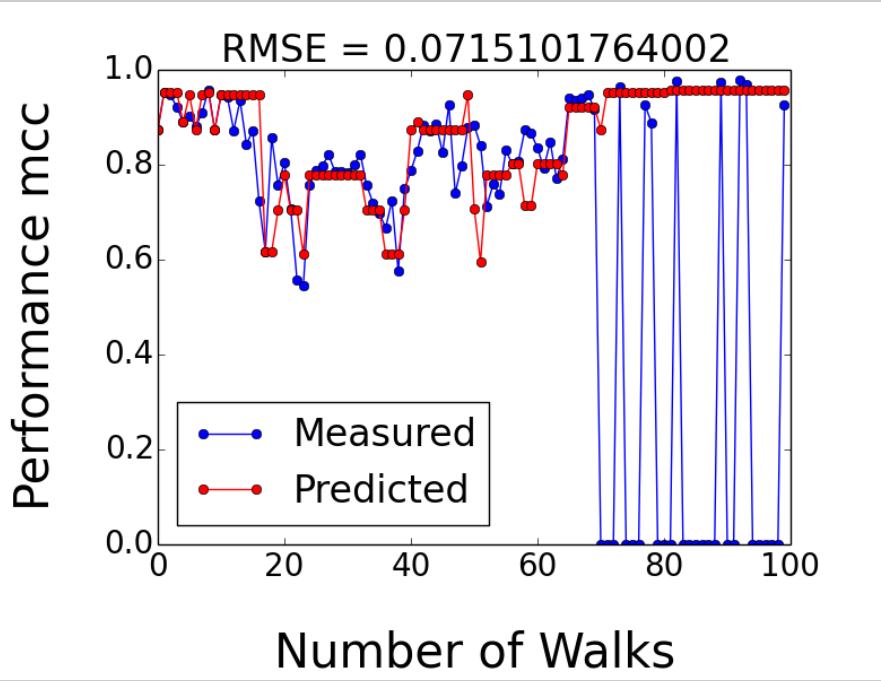
Method	TPR	FPR	min. $d(t)$	threshold t	Accuracy
Reference Analysis	0.85	0.11	na	na	0.87
Scikit MLP	0.92	0.06	0.0094	0.47	0.93
TF/Keras MLP	0.92	0.06	0.0094	0.43	0.93
Scikit MLP (rnd search)	0.93	0.06	0.0092	0.44	0.933

Conclusion

- Obtained minor improvement with HPO
  $\rightarrow$ This happens sometimes
- Use different (more efficient) HPO algorithm
  (Bayesian based optimizer)
- Try an other algorithm (e.g. random forest classifier)

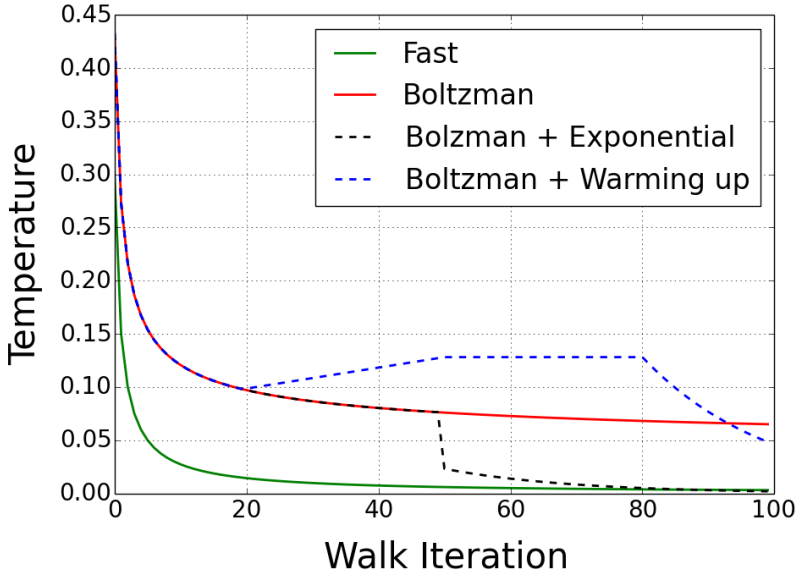
R & D on Hyper Parameter Optimization (HPO)

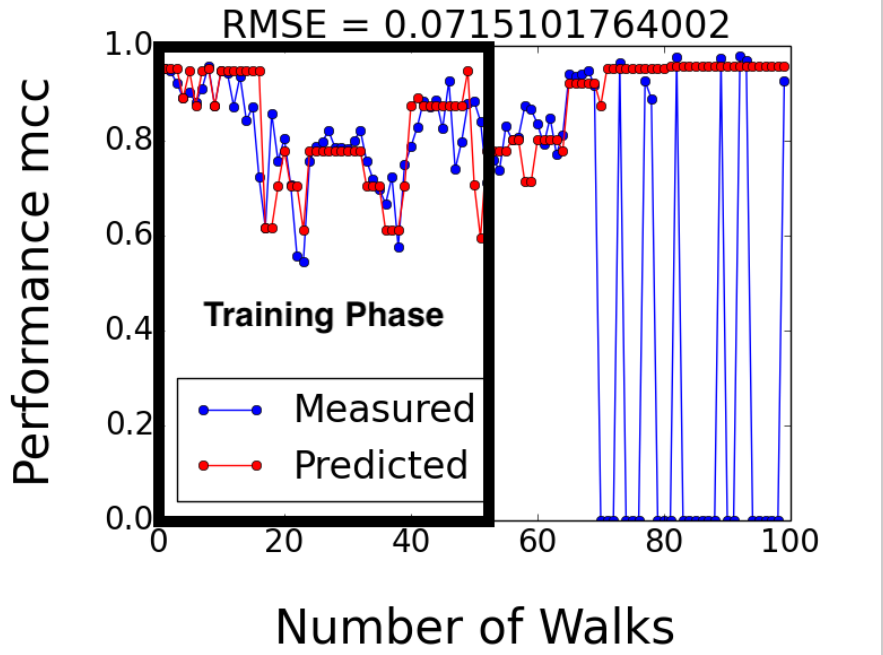




- Inspired by Random-Walk-Fitter for CLAS6 / GlueX data analysis
- Random-Walk:
  - Randomly walk through parameter space
  - Only update if improvement in performance is found
  - But: Random update according to temperature (do not get stuck at local minimum)
- Method is fast + Generates Data $\Rightarrow$ Train ML algorithm in parallel!
- Right: Preliminary results on simple classifier with fast temperature setting

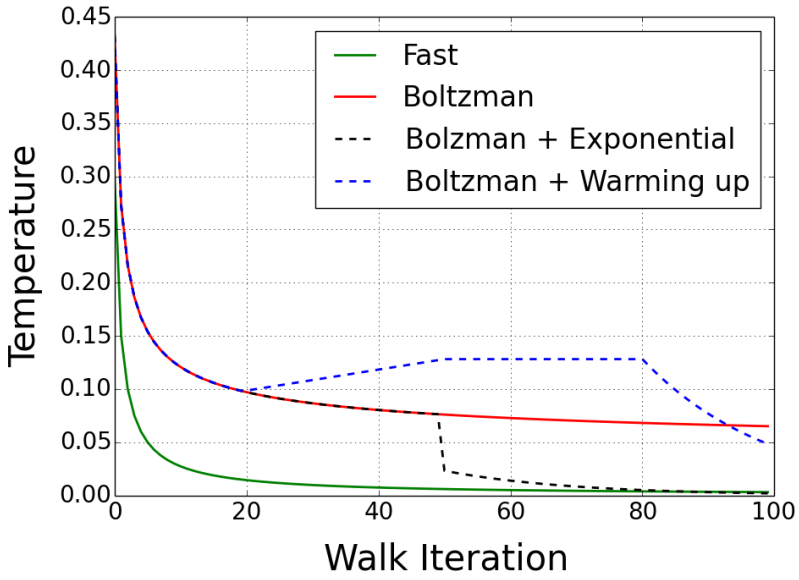
R & D on Hyper Parameter Optimization (HPO)

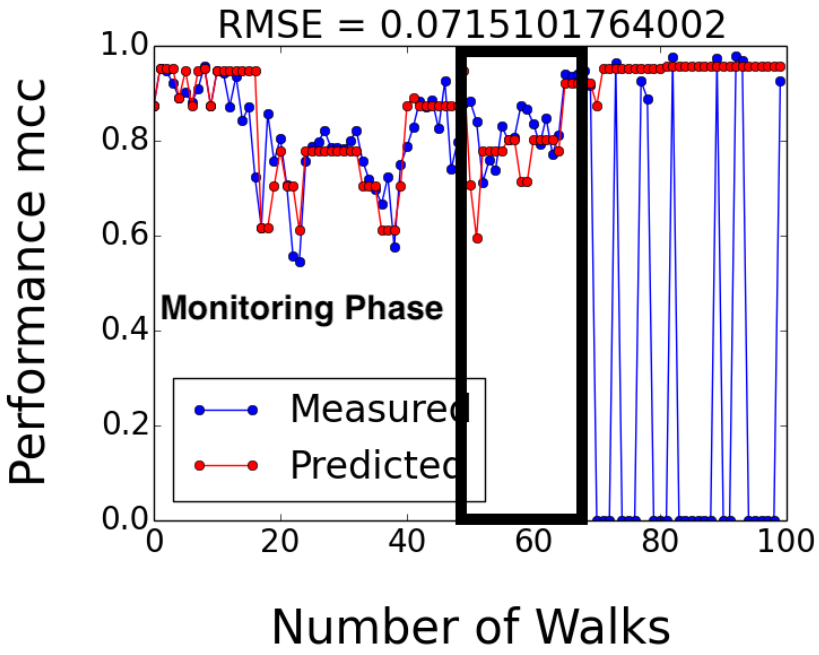




- Inspired by Random-Walk-Fitter for CLAS6 / GlueX data analysis
- Random-Walk:
  - Randomly walk through parameter space
  - Only update if improvement in performance is found
  - But: Random update according to temperature (do not get stuck at local minimum)
- Method is fast + Generates Data $\Rightarrow$ Train ML algorithm in parallel!
- Right: Preliminary results on simple classifier with fast temperature setting

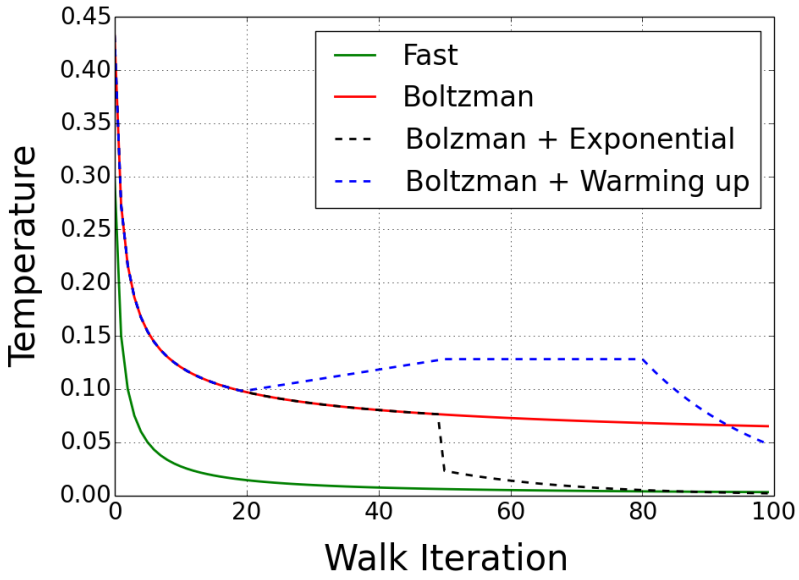
R & D on Hyper Parameter Optimization (HPO)

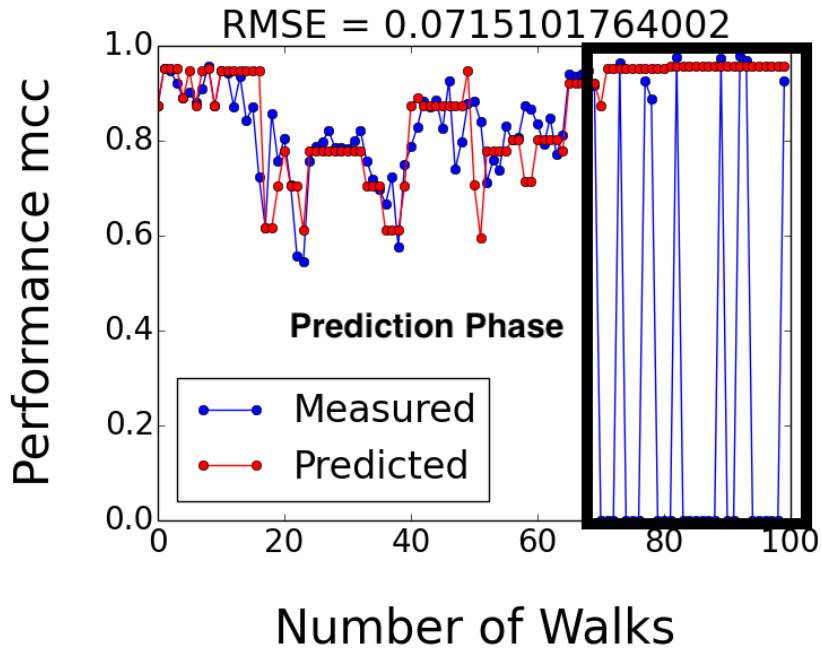




- Inspired by Random-Walk-Fitter for CLAS6 / GlueX data analysis
- Random-Walk:
  - Randomly walk through parameter space
  - Only update if improvement in performance is found
  - But: Random update according to temperature (do not get stuck at local minimum)
- Method is fast + Generates Data $\Rightarrow$ Train ML algorithm in parallel!
- Right: Preliminary results on simple classifier with fast temperature setting

R & D on Hyper Parameter Optimization (HPO)





- Inspired by Random-Walk-Fitter for CLAS6 / GlueX data analysis
- Random-Walk:
  - Randomly walk through parameter space
  - Only update if improvement in performance is found
  - But: Random update according to temperature (do not get stuck at local minimum)
- Method is fast + Generates Data $\Rightarrow$ Train ML algorithm in parallel!
- Right: Preliminary results on simple classifier with fast temperature setting

Open Questions

- I want to use machine learning in my analysis. Which framework is the best for me? Or which is the best to start with?

Open Questions

- I want to use machine learning in my analysis. Which framework is the best for me? Or which is the best to start with?
  - It depends on what you want to do

Open Questions

- I want to use machine learning in my analysis. Which framework is the best for me? Or which is the best to start with?
  - It depends on what you want to do
  - Efficient and quick filtering of large data sets without tuning too much $\rightarrow$ Apache Spark (not discussed in this lecture)

Open Questions

- I want to use machine learning in my analysis. Which framework is the best for me? Or which is the best to start with?
  - It depends on what you want to do
  - Efficient and quick filtering of large data sets without tuning too much $\rightarrow$ Apache Spark (not discussed in this lecture)
  - R & D on many different algorithms with flexibility on parameter tuning $\rightarrow$ Scikit

Open Questions

- I want to use machine learning in my analysis. Which framework is the best for me? Or which is the best to start with?
  - It depends on what you want to do
  - Efficient and quick filtering of large data sets without tuning too much $\rightarrow$ Apache Spark (not discussed in this lecture)
  - R & D on many different algorithms with flexibility on parameter tuning $\rightarrow$ Scikit
  - R & D on analyzing large and complex data sets, various options to customize own model $\rightarrow$ Keras + Tensorflow

Open Questions

- I want to use machine learning in my analysis. Which framework is the best for me? Or which is the best to start with?
  - It depends on what you want to do
  - Efficient and quick filtering of large data sets without tuning too much $\rightarrow$ Apache Spark (not discussed in this lecture)
  - R & D on many different algorithms with flexibility on parameter tuning $\rightarrow$ Scikit
  - R & D on analyzing large and complex data sets, various options to customize own model $\rightarrow$ Keras + Tensorflow
  - Of course, there is some overlap between the different frameworks

Open Questions

- I want to use machine learning in my analysis. Which framework is the best for me? Or which is the best to start with?
  - It depends on what you want to do
  - Efficient and quick filtering of large data sets without tuning too much $\rightarrow$ Apache Spark (not discussed in this lecture)
  - R & D on many different algorithms with flexibility on parameter tuning $\rightarrow$ Scikit
  - R & D on analyzing large and complex data sets, various options to customize own model $\rightarrow$ Keras + Tensorflow
  - Of course, there is some overlap between the different frameworks
  - Good documentation for each of them

Open Questions

- I want to use machine learning in my analysis. Which framework is the best for me? Or which is the best to start with?
  - It depends on what you want to do
  - Efficient and quick filtering of large data sets without tuning too much $\rightarrow$ Apache Spark (not discussed in this lecture)
  - R & D on many different algorithms with flexibility on parameter tuning $\rightarrow$ Scikit
  - R & D on analyzing large and complex data sets, various options to customize own model $\rightarrow$ Keras + Tensorflow
  - Of course, there is some overlap between the different frameworks
  - Good documentation for each of them
- Which model shall I use for my analysis?

Open Questions

- I want to use machine learning in my analysis. Which framework is the best for me? Or which is the best to start with?
  - It depends on what you want to do
  - Efficient and quick filtering of large data sets without tuning too much $\rightarrow$ Apache Spark (not discussed in this lecture)
  - R & D on many different algorithms with flexibility on parameter tuning $\rightarrow$ Scikit
  - R & D on analyzing large and complex data sets, various options to customize own model $\rightarrow$ Keras + Tensorflow
  - Of course, there is some overlap between the different frameworks
  - Good documentation for each of them
- Which model shall I use for my analysis?
  - Again, it depends on what you want to do (classify small data set with few features vs. customized model on individual problem)

Open Questions

- I want to use machine learning in my analysis. Which framework is the best for me? Or which is the best to start with?
  - It depends on what you want to do
  - Efficient and quick filtering of large data sets without tuning too much $\rightarrow$ Apache Spark (not discussed in this lecture)
  - R & D on many different algorithms with flexibility on parameter tuning $\rightarrow$ Scikit
  - R & D on analyzing large and complex data sets, various options to customize own model $\rightarrow$ Keras + Tensorflow
  - Of course, there is some overlap between the different frameworks
  - Good documentation for each of them
- Which model shall I use for my analysis?
  - Again, it depends on what you want to do (classify small data set with few features vs. customized model on individual problem)
  - There is no need to use a deep model with 10^9 parameters, if a random forest classifier shows (after a careful analysis) a close to perfect performance

Open Questions

- I want to use machine learning in my analysis. Which framework is the best for me? Or which is the best to start with?
  - It depends on what you want to do
  - Efficient and quick filtering of large data sets without tuning too much $\rightarrow$ Apache Spark (not discussed in this lecture)
  - R & D on many different algorithms with flexibility on parameter tuning $\rightarrow$ Scikit
  - R & D on analyzing large and complex data sets, various options to customize own model $\rightarrow$ Keras + Tensorflow
  - Of course, there is some overlap between the different frameworks
  - Good documentation for each of them
- Which model shall I use for my analysis?
  - Again, it depends on what you want to do (classify small data set with few features vs. customized model on individual problem)
  - There is no need to use a deep model with 10^9 parameters, if a random forest classifier shows (after a careful analysis) a close to perfect performance
  - Do not try to fine tune a random forest classifier on a complex data set (e.g. ~ 100 different correlated variables), if you can not achieve a reasonable performance

References and Further Reading

- Most of the information / code pieces have been taken from / inspired by the following web-sites: (the blue items are clickable links)
  - **Apache Spark**
  - **Apache Spark ML**
  - **Pyspark documentation**
  - **Python scikit-learn**
  - **Tensorflow**
  - **Tensorflow Keras Models**
  - **Keras**
  - **stackoverflow**
  - **distill.pub: Current problems and issues in machine/deep learning**
  - **A Recipe for Training Neural Networks (Andrej Karpathy)**
  - "Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow", by Aurélien Géron $\rightarrow$ Really good book!
  - Talks from the **deep learning for science school 2019**
  - Talks from the **deep learning for science school 2020**

References and Further Reading

- Most of the information / code pieces have been taken from / inspired by the following web-sites: (the blue items are clickable links)
  - **Apache Spark**
  - **Apache Spark ML**
  - **Pyspark documentation**
  - **Python scikit-learn**
  - **Tensorflow**
  - **Tensorflow Keras Models**
  - **Keras**
  - **stackoverflow**
  - **distill.pub: Current problems and issues in machine/deep learning**
  - **A Recipe for Training Neural Networks (Andrej Karpathy)**
  - "Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow", by Aurélien Géron $\rightarrow$ Really good book!
  - Talks from the **deep learning for science school 2019**
  - Talks from the **deep learning for science school 2020**
- If you are stuck with a problem / framework $\Rightarrow$ Do not spend weeks to solve it $\Rightarrow$ Look it up (stackoverflow, google, yahoo,...), most likely someone has a similar problem

References and Further Reading

- Most of the information / code pieces have been taken from / inspired by the following web-sites: (the blue items are clickable links)
  - **Apache Spark**
  - **Apache Spark ML**
  - **Pyspark documentation**
  - **Python scikit-learn**
  - **Tensorflow**
  - **Tensorflow Keras Models**
  - **Keras**
  - **stackoverflow**
  - **distill.pub: Current problems and issues in machine/deep learning**
  - **A Recipe for Training Neural Networks (Andrej Karpathy)**
  - "Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow", by Aurélien Géron $\rightarrow$ Really good book!
  - Talks from the **deep learning for science school 2019**
  - Talks from the **deep learning for science school 2020**
- If you are stuck with a problem / framework $\Rightarrow$ Do not spend weeks to solve it $\Rightarrow$ Look it up (stackoverflow, google, yahoo,...), most likely someone has a similar problem
- Try not to start from scratch (sometimes not avoidable)

References and Further Reading

- Most of the information / code pieces have been taken from / inspired by the following web-sites: (the blue items are clickable links)
  - **Apache Spark**
  - **Apache Spark ML**
  - **Pyspark documentation**
  - **Python scikit-learn**
  - **Tensorflow**
  - **Tensorflow Keras Models**
  - **Keras**
  - **stackoverflow**
  - **distill.pub: Current problems and issues in machine/deep learning**
  - **A Recipe for Training Neural Networks (Andrej Karpathy)**
  - "Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow", by Aurélien Géron $\rightarrow$ Really good book!
  - Talks from the **deep learning for science school 2019**
  - Talks from the **deep learning for science school 2020**
- If you are stuck with a problem / framework $\Rightarrow$ Do not spend weeks to solve it $\Rightarrow$ Look it up (stackoverflow, google, yahoo,...), most likely someone has a similar problem
- Try not to start from scratch (sometimes not avoidable)
- Again, there is no easy / quick way to learn all the aspects of machine learning

References and Further Reading

- Most of the information / code pieces have been taken from / inspired by the following web-sites: (the blue items are clickable links)
  - **Apache Spark**
  - **Apache Spark ML**
  - **Pyspark documentation**
  - **Python scikit-learn**
  - **Tensorflow**
  - **Tensorflow Keras Models**
  - **Keras**
  - **stackoverflow**
  - **distill.pub: Current problems and issues in machine/deep learning**
  - **A Recipe for Training Neural Networks (Andrej Karpathy)**
  - "Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow", by Aurélien Géron $\rightarrow$ Really good book!
  - Talks from the **deep learning for science school 2019**
  - Talks from the **deep learning for science school 2020**
- If you are stuck with a problem / framework $\Rightarrow$ Do not spend weeks to solve it $\Rightarrow$ Look it up (stackoverflow, google, yahoo,...), most likely someone has a similar problem
- Try not to start from scratch (sometimes not avoidable)
- Again, there is no easy / quick way to learn all the aspects of machine learning
- My personal recommendation: Try it out yourself! (i.e. pick an example data set and start playing around)

Final Remarks

- Tried to give a rough impression on machine learning in physics data analysis

Final Remarks

- Tried to give a rough impression on machine learning in physics data analysis
- Did not cover all aspects in machine learning

Final Remarks

- Tried to give a rough impression on machine learning in physics data analysis
- Did not cover all aspects in machine learning
- Not all shown code snippets are best programming practice $\rightarrow$ They shall simply give you an idea on how to implement various functions

Final Remarks

- Tried to give a rough impression on machine learning in physics data analysis
- Did not cover all aspects in machine learning
- Not all shown code snippets are best programming practice $\rightarrow$ They shall simply give you an idea on how to implement various functions
- scikit provides a much larger functionality than shown in this lecture

Final Remarks

- Tried to give a rough impression on machine learning in physics data analysis
- Did not cover all aspects in machine learning
- Not all shown code snippets are best programming practice $\rightarrow$ They shall simply give you an idea on how to implement various functions
- scikit provides a much larger functionality than shown in this lecture
- Many approaches shown here are based on my own experience, i.e. NOT the ultimate truth $\rightarrow$ Someone else might have tackled the problems differently

Final Remarks

- Tried to give a rough impression on machine learning in physics data analysis
- Did not cover all aspects in machine learning
- Not all shown code snippets are best programming practice $\rightarrow$ They shall simply give you an idea on how to implement various functions
- scikit provides a much larger functionality than shown in this lecture
- Many approaches shown here are based on my own experience, i.e. NOT the ultimate truth $\rightarrow$ Someone else might have tackled the problems differently
- I hope you had fun!