# Introduction to Machine Learning: Part IV

Prof. Sean Dobbs[1] & Daniel Lersch[2]

April 23, 2020

[1] (sdobbs@fsu.edu)
[2] (dlersch@jlab.org)

# About this Lecture

- **Part I:**
  - Introduction to DataFrames
  - Basic concepts of machine learning

    (with focus on feedforward neural networks)
- **Part II:**
  - Machine learning in (physics) data analysis
  - Performance evaluation
- **Part III:**
  - Algorithm tuning
  - Hyper parameter optimization
- **Part IV:**
  - Custom neural networks with Tensorflow
  - Transition to Deep Learning

**The individual contents might be subject to change**

# This Lecture will...

... NOT turn you into a machine learning specialist

# This Lecture will...

... NOT turn you into a machine learning specialist

... NOT cover all aspects of machine learning

# This Lecture will...

... NOT turn you into a machine learning specialist

... NOT cover all aspects of machine learning

... give a (very) brief overview only (i.e. further reading is definitely required)

# This Lecture will...

... NOT turn you into a machine learning specialist

... NOT cover all aspects of machine learning

... give a (very) brief overview only (i.e. further reading is definitely required)

... introduce a few machine learning algorithms

# This Lecture will...

... NOT turn you into a machine learning specialist

... NOT cover all aspects of machine learning

... give a (very) brief overview only (i.e. further reading is definitely required)

... introduce a few machine learning algorithms

... utilize the **scikit-learn** library

# This Lecture will...

... NOT turn you into a machine learning specialist

... NOT cover all aspects of machine learning

... give a (very) brief overview only (i.e. further reading is definitely required)

... introduce a few machine learning algorithms

... utilize the **scikit-learn** library

... most likely contain several errors ($\rightarrow$ Please send a mail to `dlersch@jlab.org`)

# Homework and Literature

- Machine learning can be learned best by simply doing it!

# Homework and Literature

- Machine learning can be learned best by simply doing it!
- Homework (most likely posted on Thursday) aims to perform a simple analysis and getting familiar with machine learning

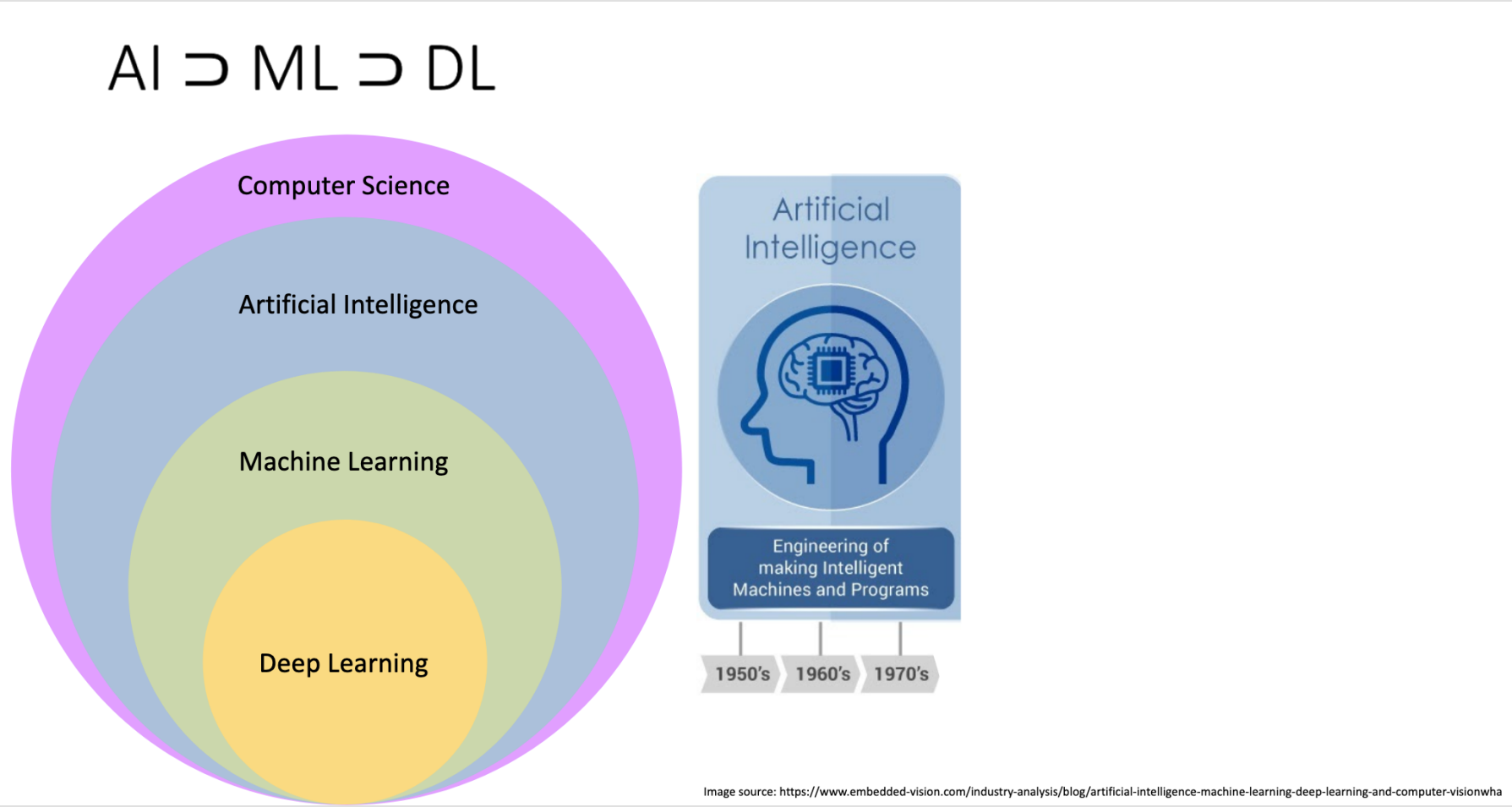
# Homework and Literature

- Machine learning can be learned best by simply doing it!
- Homework (most likely posted on Thursday) aims to perform a simple analysis and getting familiar with machine learning
- Helpful literature:
  - The **scikit-learn** documentation
  - Talks from the **deep learning for science school 2019**[3]
  - "Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow", by Aurélien Géron
  - The internet is full of good (but also very bad!) literature[4] $\rightarrow$ browse with caution
  - The slides of the lecture are available at: `http://hadron.physics.fsu.edu/~dlersch/ml_slides/`

[3] Very good and detailed explanation of (deep) neural networks

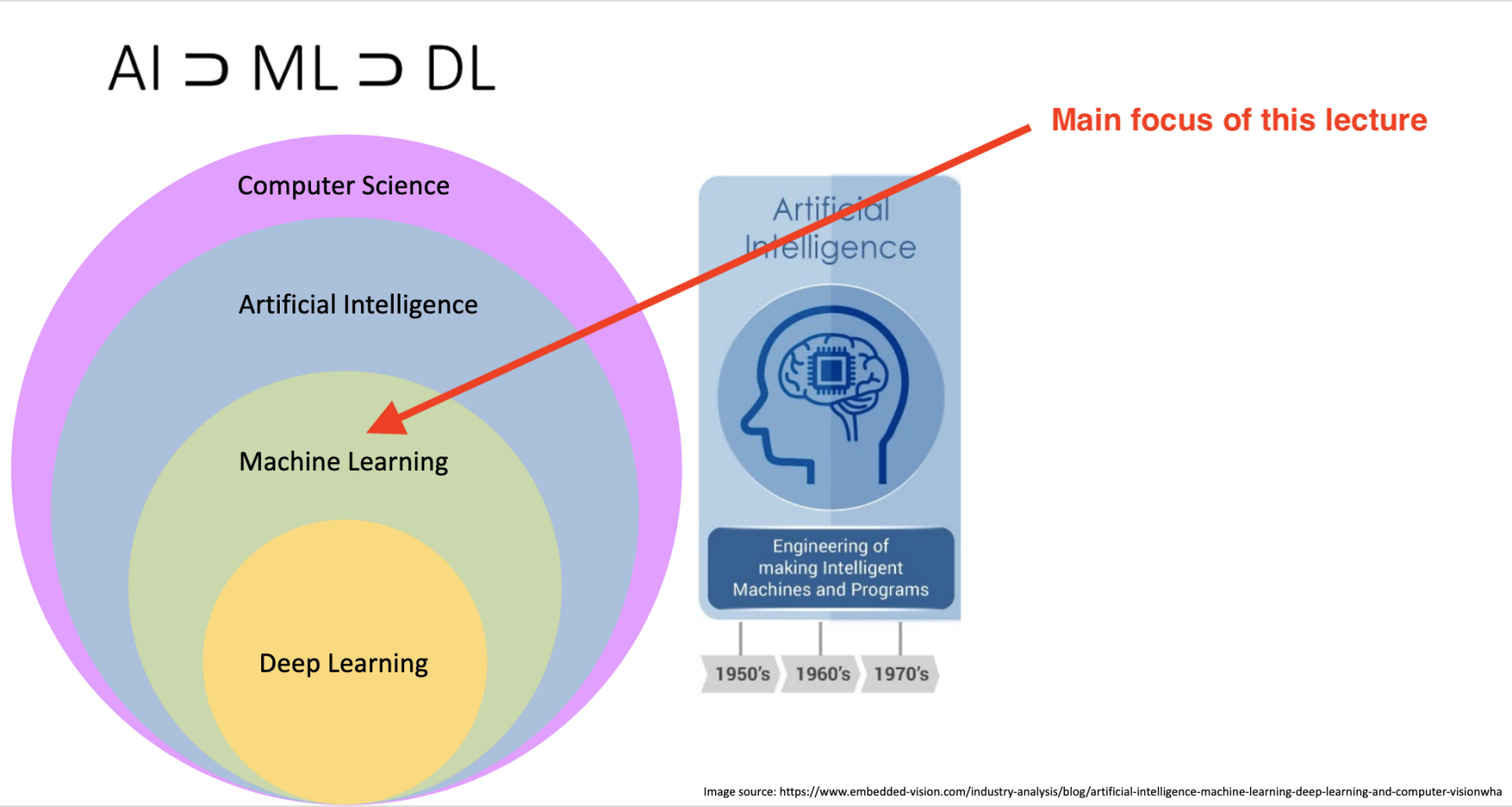
[4] Any document claiming that there is a quick way to understand machine learning without any theory / math is considered as bad

# AI, ML and DL



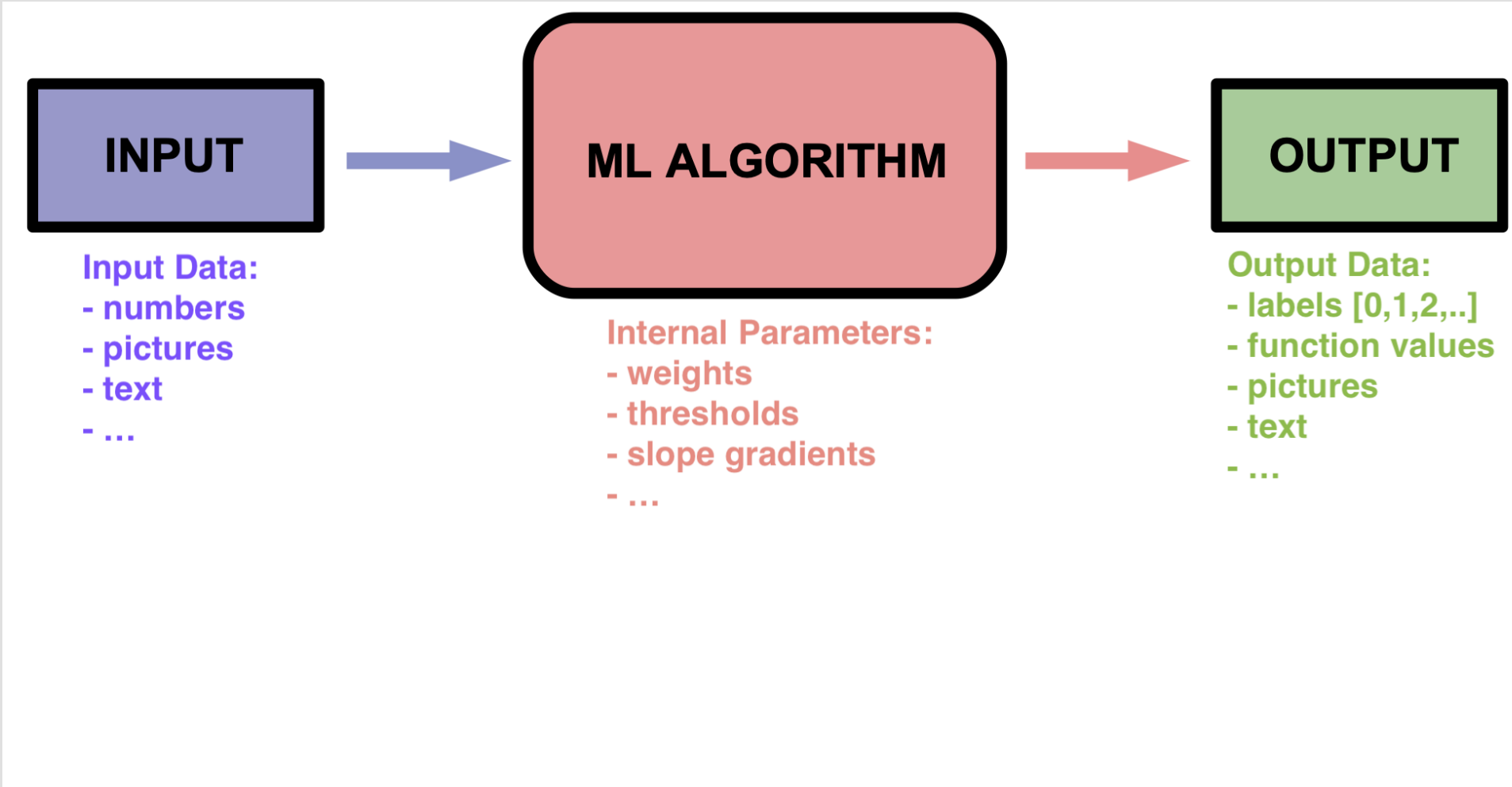

Slide taken from Brenda Ngs introductory talk at the: **deep learning for science school 2019**

# AI, ML and DL

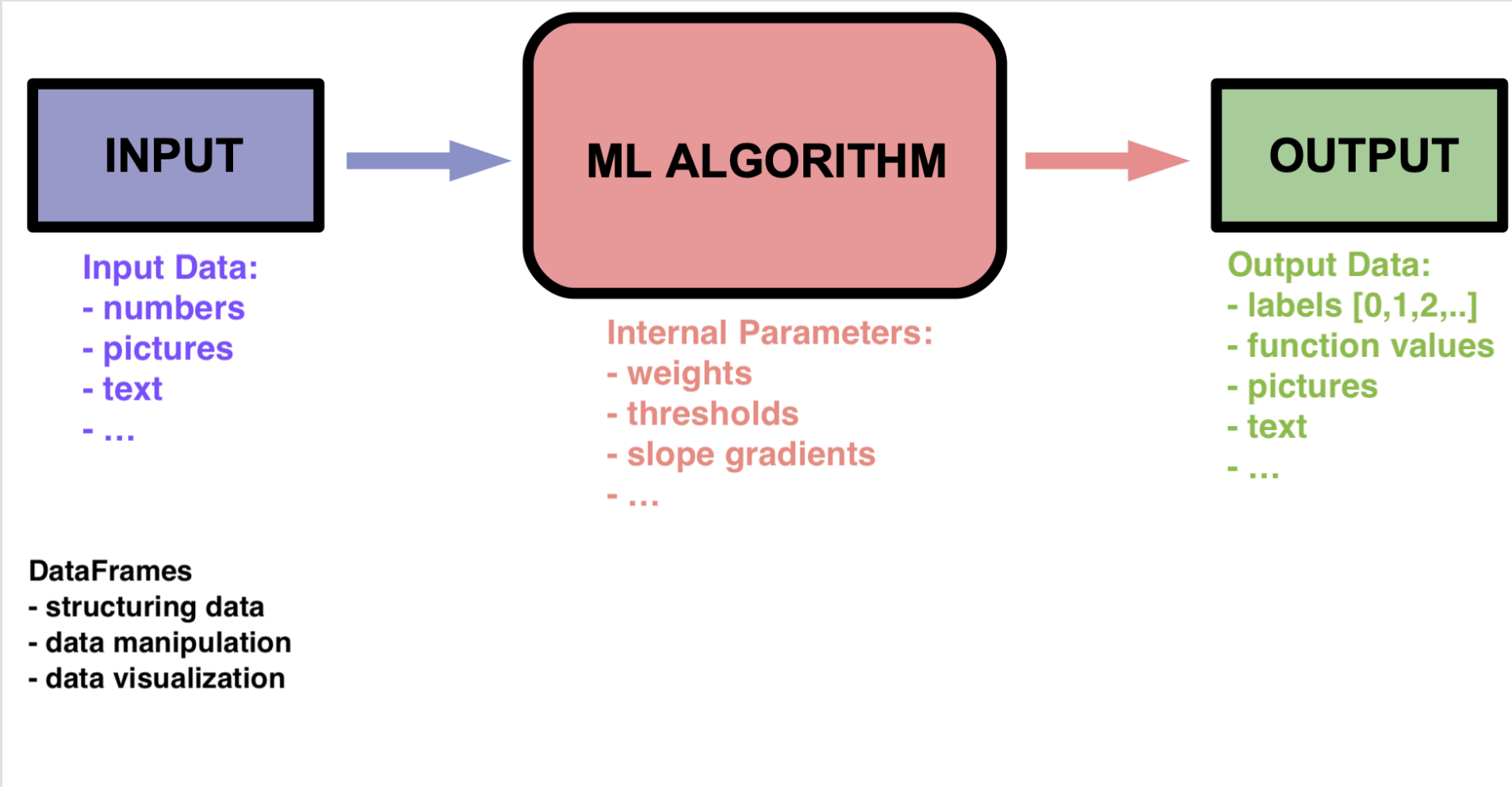



Slide taken from Brenda Ngs introductory talk at the: **deep learning for science school 2019**

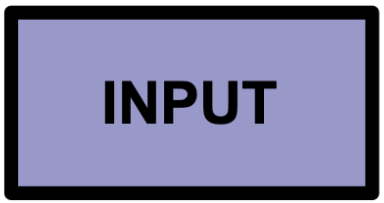
# Basic Components of Machine Learning

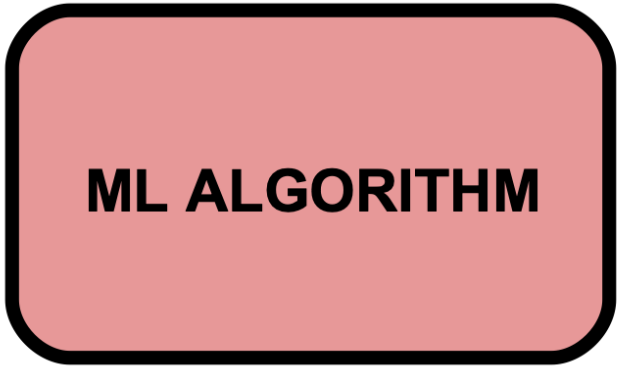


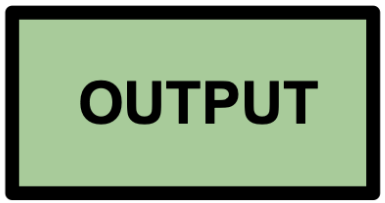
# Basic Components of Machine Learning



# Basic Components of Machine Learning




**Input Data:**
- numbers
- pictures
- text
- ...

**Internal Parameters:**
- weights
- thresholds
- slope gradients
- ...

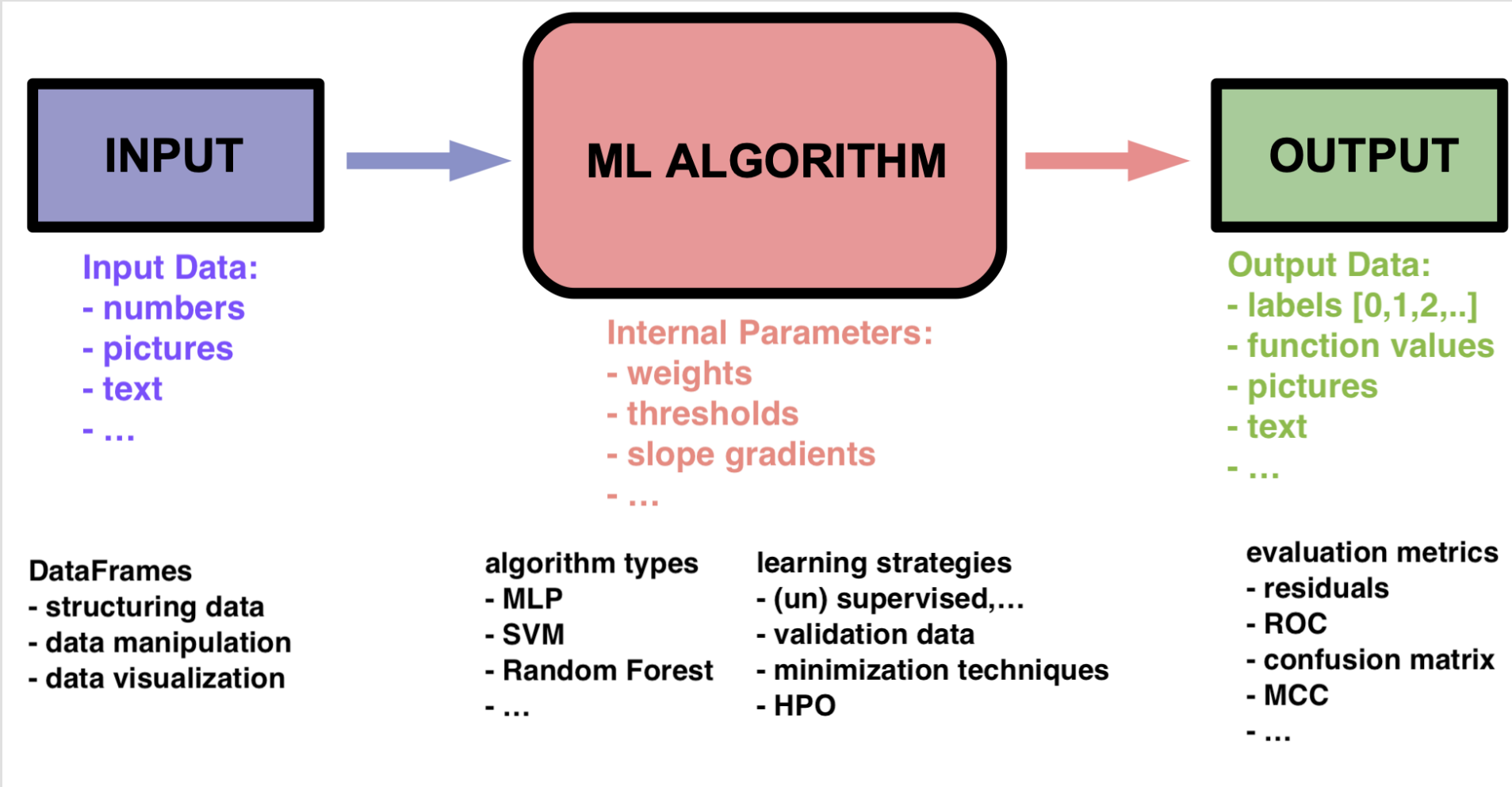
**Output Data:**
- labels [0,1,2,..]
- function values
- pictures
- text
- ...

**DataFrames**
- structuring data
- data manipulation
- data visualization

**algorithm types**
- MLP
- SVM
- Random Forest
- ...

**learning strategies**
- (un) supervised,...
- validation data
- minimization techniques
- HPO

# Basic Components of Machine Learning



# Random Hyper Parameter Search

- In part III: Introduced HPO

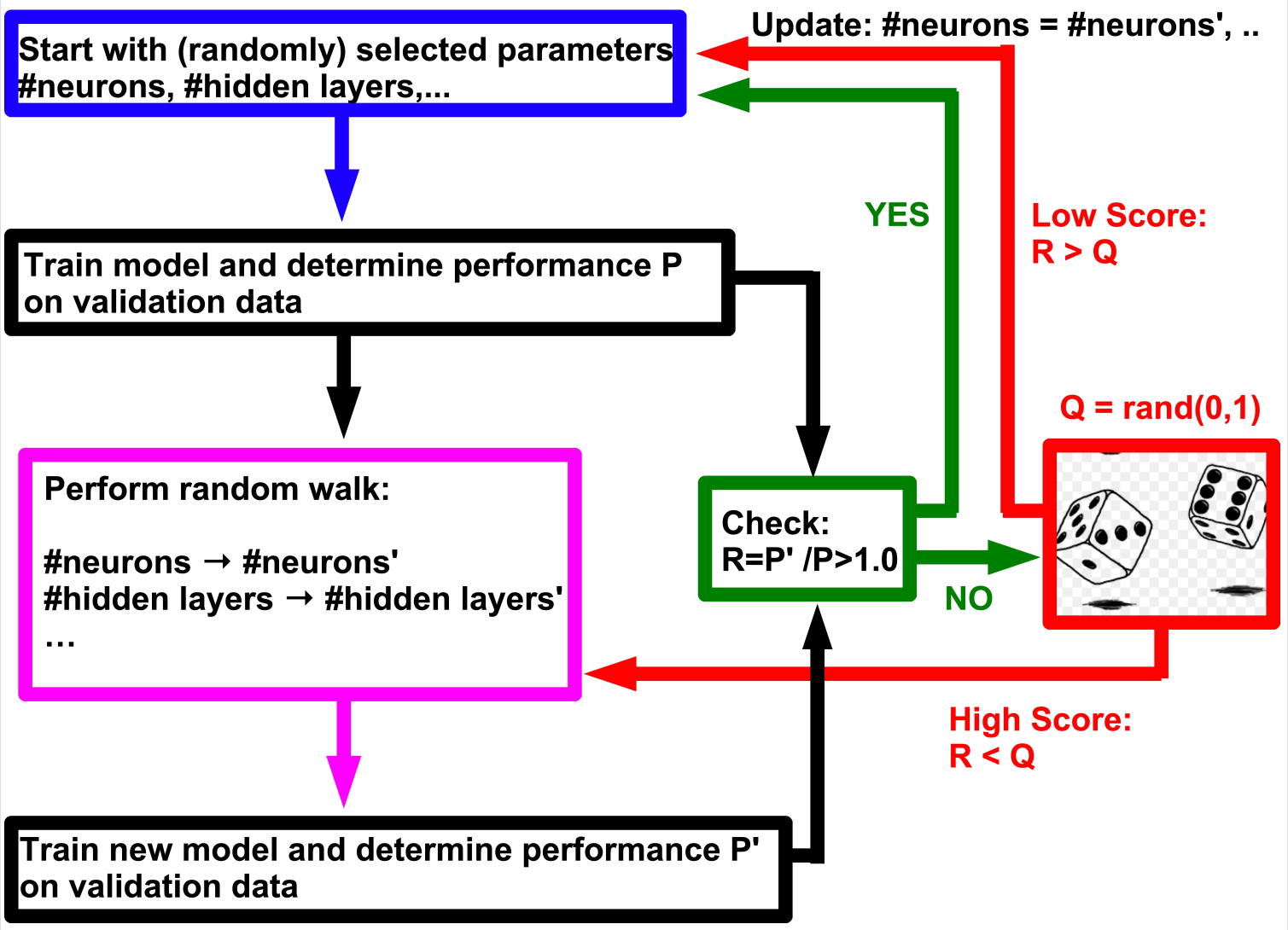
# Random Hyper Parameter Search

- In part III: Introduced HPO
- Tried grid parameter search $\rightarrow$ fixed parameter ranges
  - Might have to try many parameter combinations
  - No guarantee that optimum is found

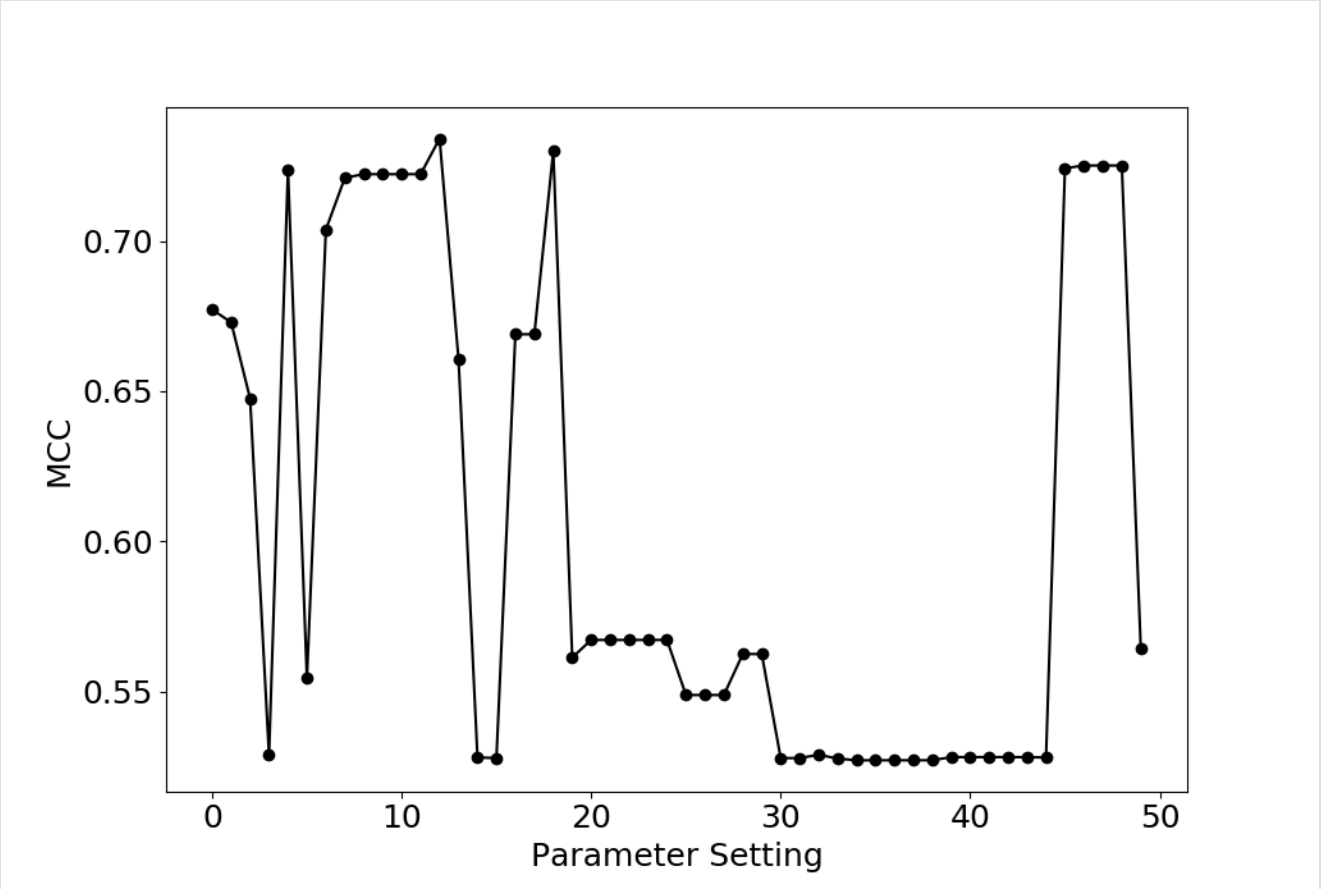
# Random Hyper Parameter Search

- In part III: Introduced HPO
- Tried grid parameter search $\rightarrow$ fixed parameter ranges
  - Might have to try many parameter combinations
  - No guarantee that optimum is found
- (One) alternative: try a random walk search
  - Explore parameter space by random selection
  - Add selection rules, e.g. mlp shall grow in depth, rather than in width

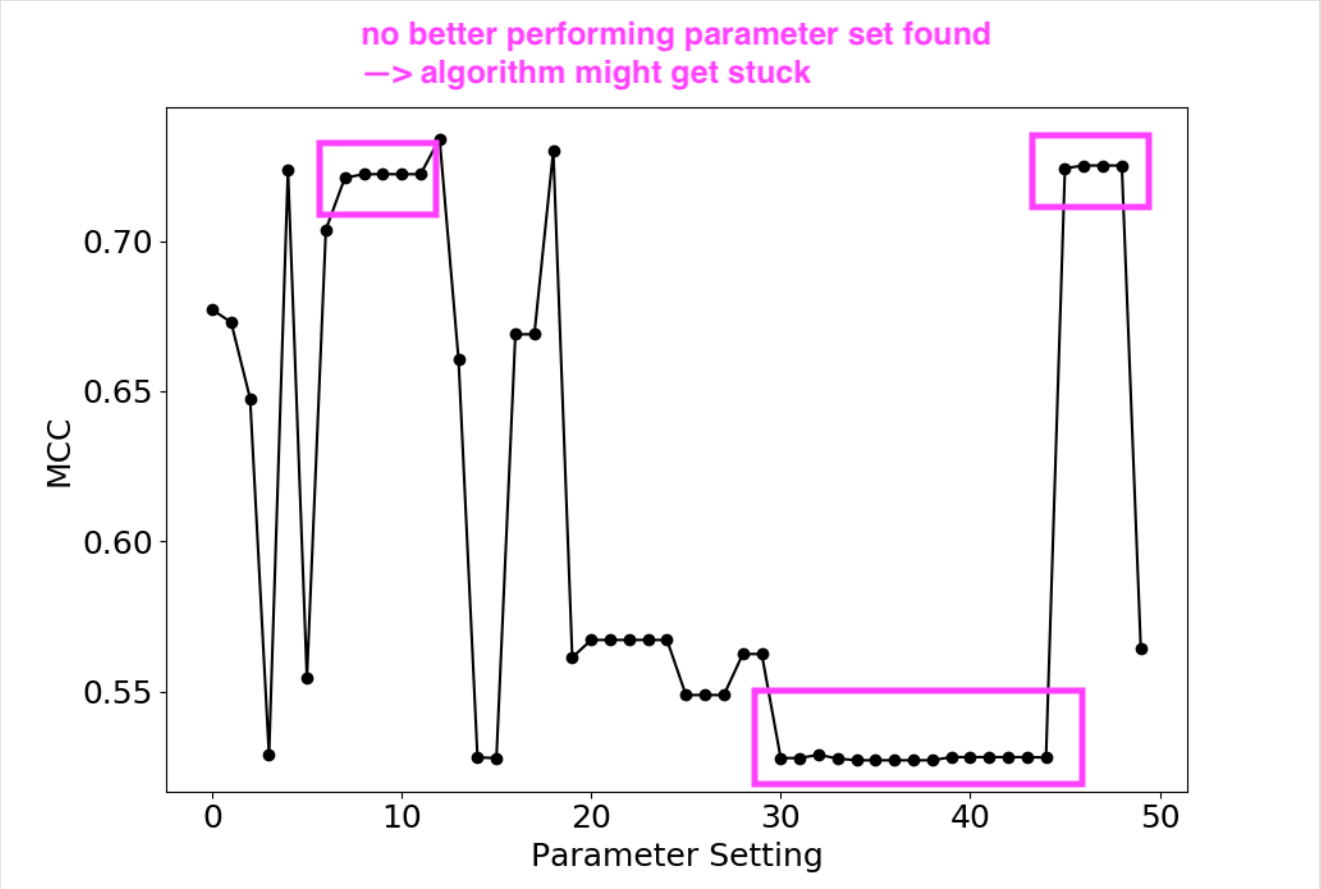
# Random Hyper Parameter Search



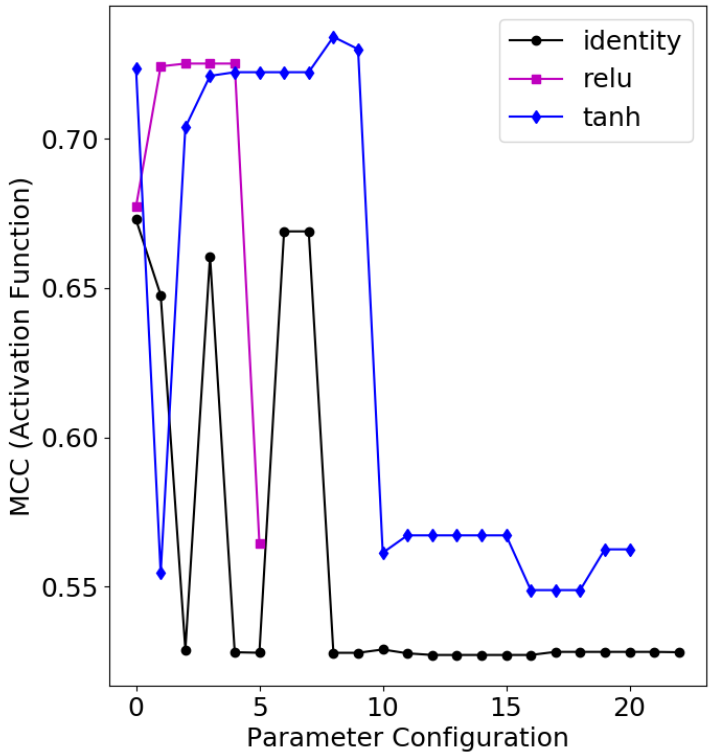
# Results Random Search

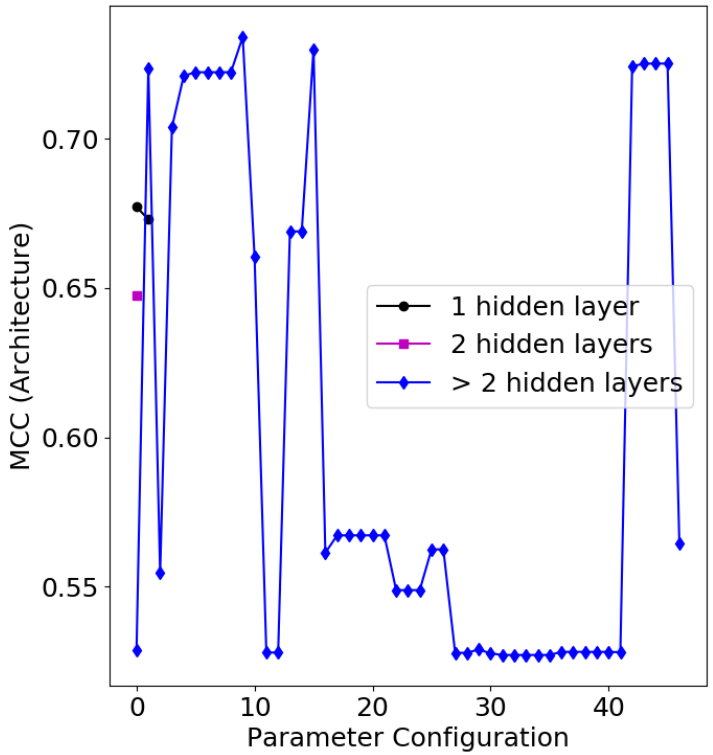


# Results Random Search

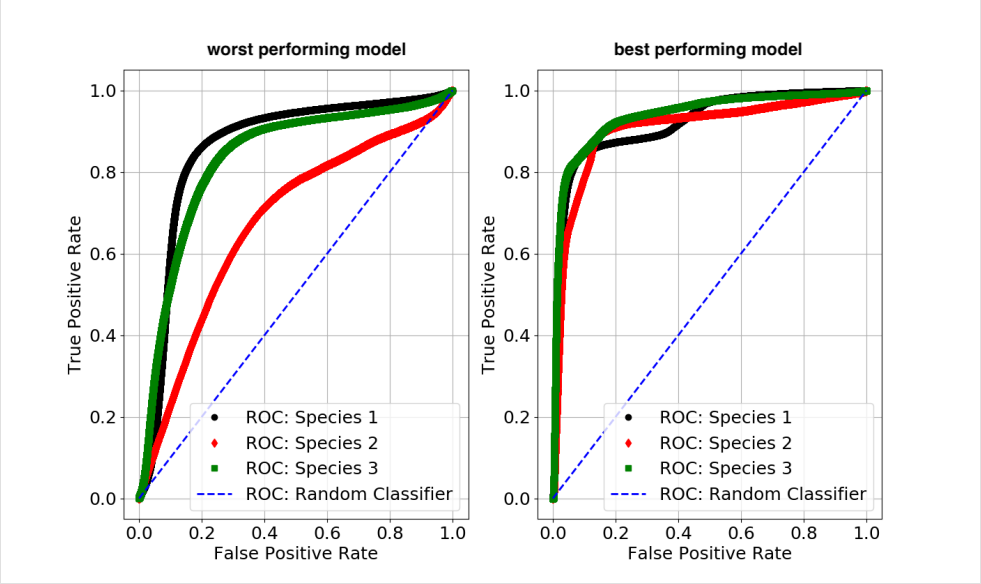


# Results Random Search



# Results Random Search



# Summary Random Search

- Found winning model (MCC = 0.73) after 50 parameter searches

# Summary Random Search

- Found winning model (MCC = 0.73) after 50 parameter searches
- Performances are comparable to grid search results

# Summary Random Search

- Found winning model (MCC = 0.73) after 50 parameter searches
- Performances are comparable to grid search results
- ROC curves looks slightly more reasonable

# Summary Random Search

- Found winning model (MCC = 0.73) after 50 parameter searches
- Performances are comparable to grid search results
- ROC curves looks slightly more reasonable
- However, winning model ended up having 7 hidden layers $\rightarrow$ Set the probability to grow in depth / width to 75% / 50% $\rightarrow$ Seems to be an unreasonable setting

# Summary Random Search

- Found winning model (MCC = 0.73) after 50 parameter searches
- Performances are comparable to grid search results
- ROC curves looks slightly more reasonable
- However, winning model ended up having 7 hidden layers $\rightarrow$ Set the probability to grow in depth / width to 75% / 50% $\rightarrow$ Seems to be an unreasonable setting
- BUT: One can easily include constraints into the random search, or simply change the initial settings

# Summary Random Search

- Found winning model (MCC = 0.73) after 50 parameter searches
- Performances are comparable to grid search results
- ROC curves looks slightly more reasonable
- However, winning model ended up having 7 hidden layers $\rightarrow$ Set the probability to grow in depth / width to 75% / 50% $\rightarrow$ Seems to be an unreasonable setting
- BUT: One can easily include constraints into the random search, or simply change the initial settings
- This was just an example on how a HPO random search could look like $\rightarrow$ There are many optimization algorithms, that are by far more sophisticated

# Summary Random Search

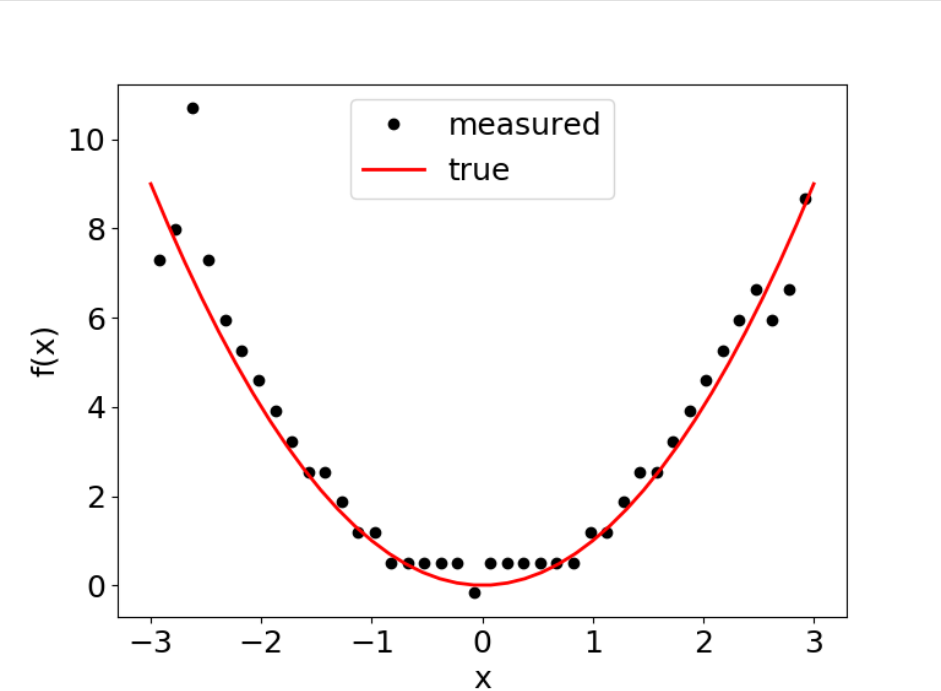
- Found winning model (MCC = 0.73) after 50 parameter searches
- Performances are comparable to grid search results
- ROC curves looks slightly more reasonable
- However, winning model ended up having 7 hidden layers $\rightarrow$ Set the probability to grow in depth / width to 75% / 50% $\rightarrow$ Seems to be an unreasonable setting
- BUT: One can easily include constraints into the random search, or simply change the initial settings
- This was just an example on how a HPO random search could look like $\rightarrow$ There are many optimization algorithms, that are by far more sophisticated
- For example **Spearmint Bayesian Optimization**

# From Machine to Deep Learning: An Example

- Remember our example from the beginning

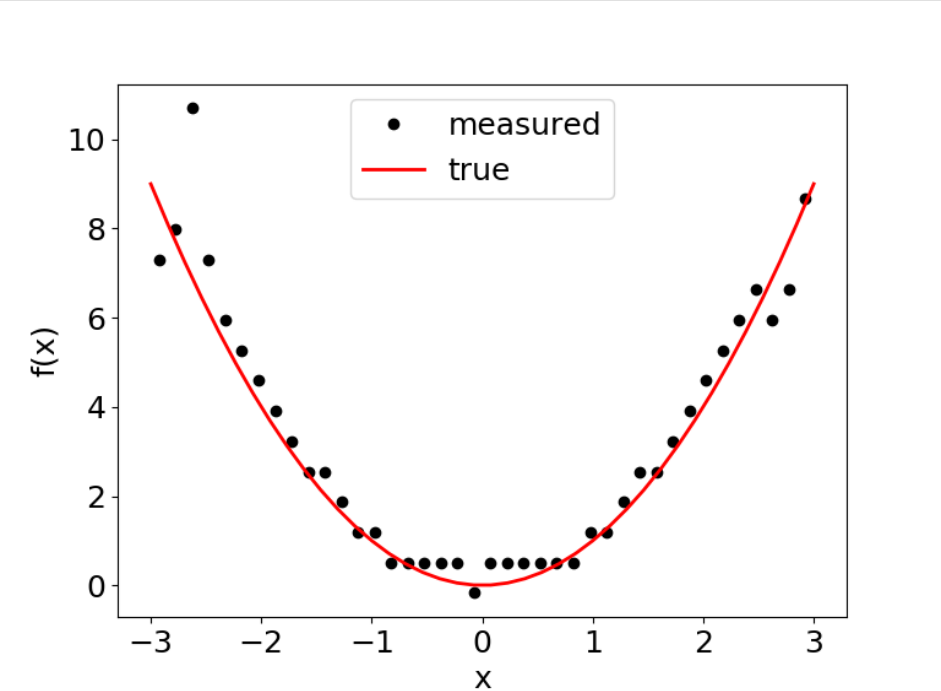
# From Machine to Deep Learning: An Example

- Remember our example from the beginning
- We have a set of measured points and try to determine the underlying function (e.g. via mlp, Gaussian processor,..)



# From Machine to Deep Learning: An Example

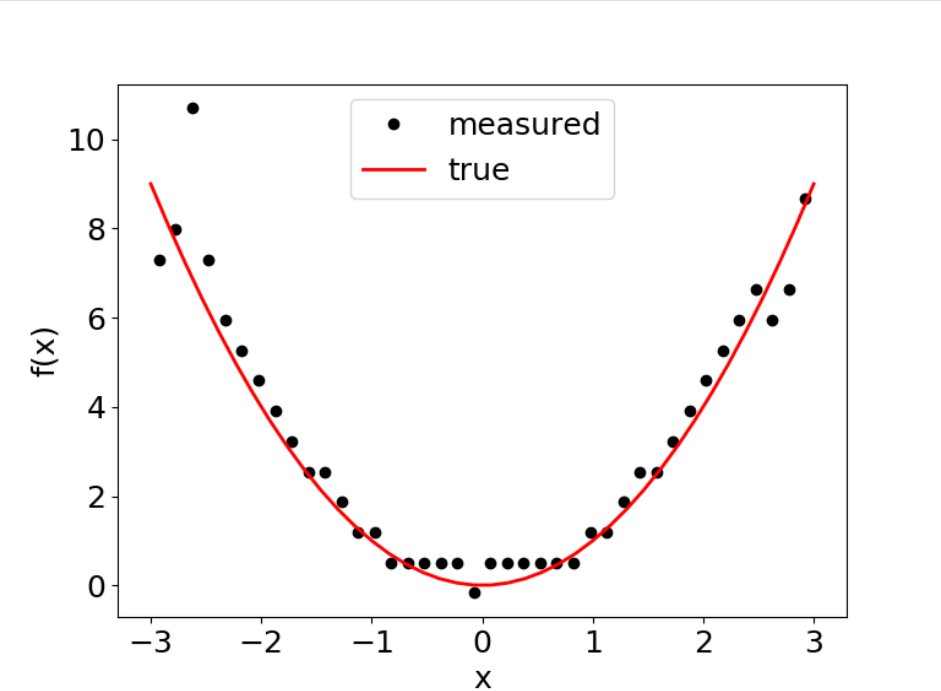
- Remember our example from the beginning
- We have a set of measured points and try to determine the underlying function (e.g. via mlp, Gaussian processor,..)




- This data might come from a measurement after some preprocessing (e.g. calibration, noise supression,...) $\rightarrow$ Information has already been provided

# From Machine to Deep Learning: An Example

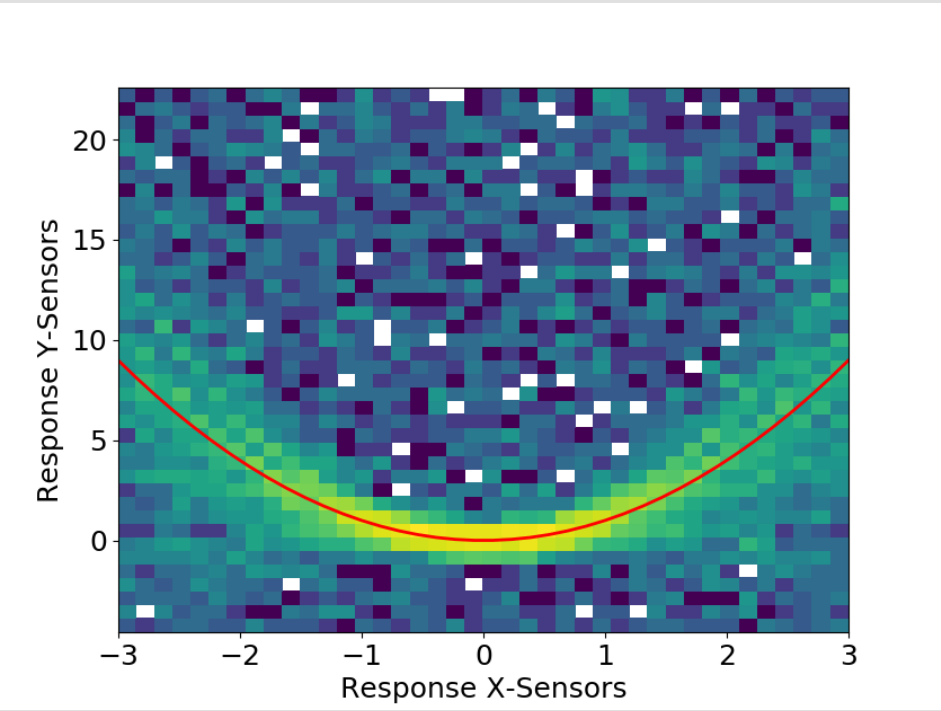
- Remember our example from the beginning
- We have a set of measured points and try to determine the underlying function (e.g. via mlp, Gaussian processor,..)




- This data might come from a measurement after some preprocessing (e.g. calibration, noise supression,...) $\rightarrow$ Information has already been provided
- What happens if we leave the preprocessing step out? $\rightarrow$ Remove one level of information

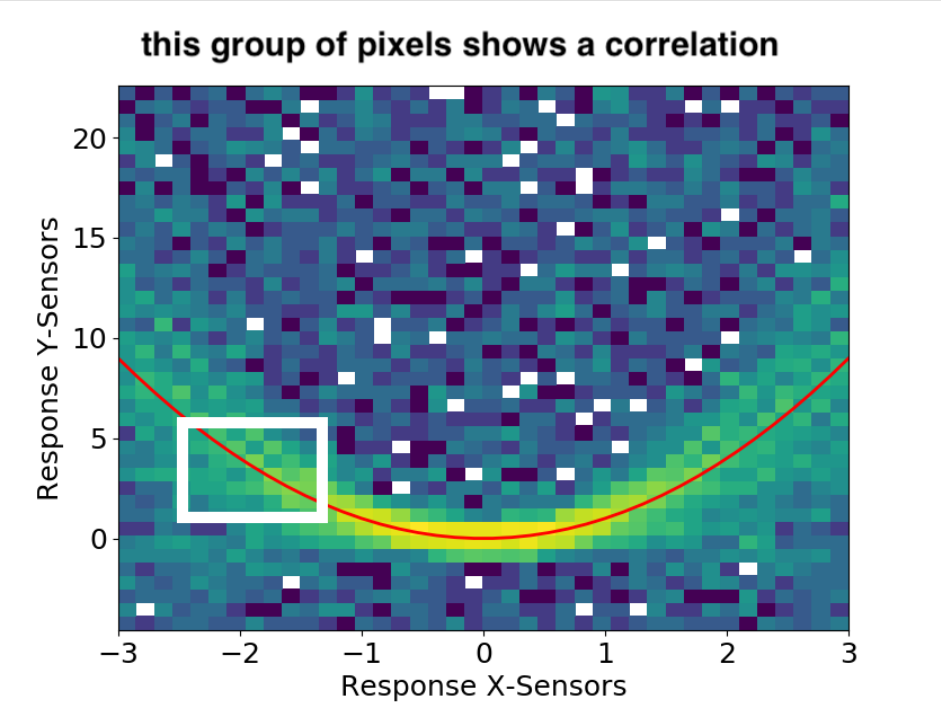
# From Machine to Deep Learning: An Example

- Remember our example from the beginning
- We have a set of measured points and try to determine the underlying function (e.g. via mlp, Gaussian processor,..)
- This data might come from a measurement after some preprocessing (e.g. calibration, noise supression,...) $\rightarrow$ Information has already been provided
- What happens if we leave the preprocessing step out? $\rightarrow$ Remove one level of information
- Look at raw data and analyze it pixel by pixel



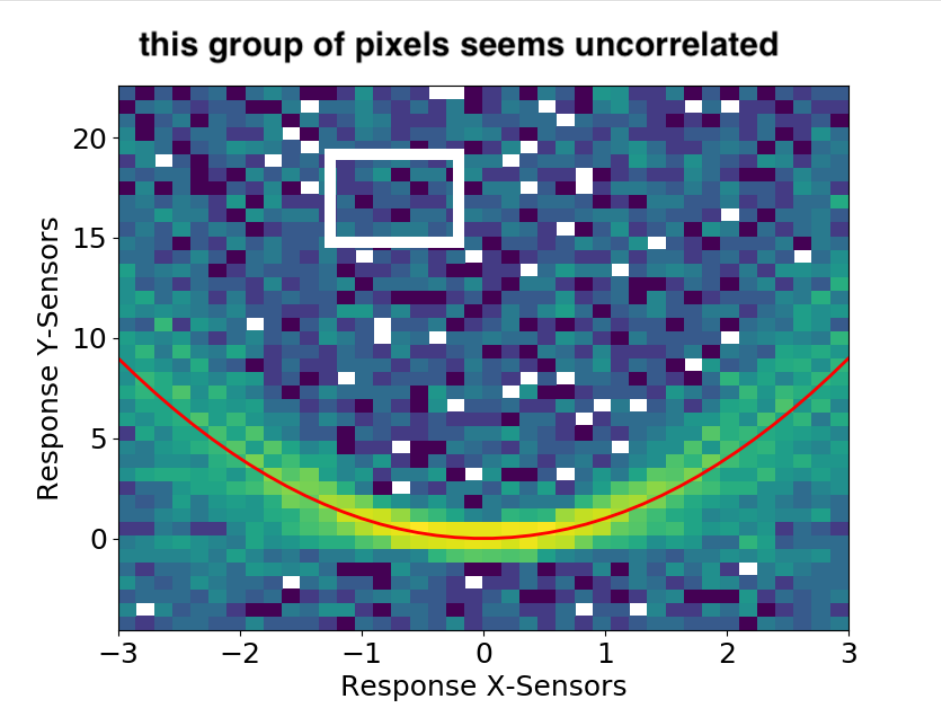
# From Machine to Deep Learning: An Example

- Remember our example from the beginning
- We have a set of measured points and try to determine the underlying function (e.g. via mlp, Gaussian processor,..)
- This data might come from a measurement after some preprocessing (e.g. calibration, noise supression,...) $\rightarrow$ Information has already been provided
- What happens if we leave the preprocessing step out? $\rightarrow$ Remove one level of information
- Look at raw data and analyze it pixel by pixel



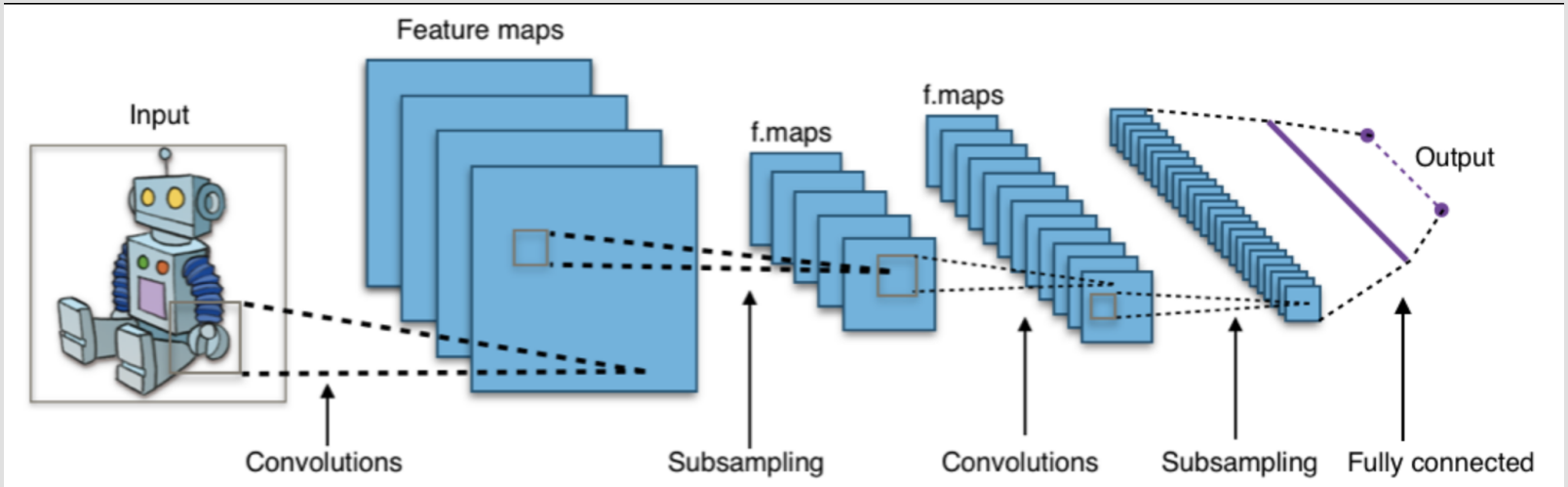
# From Machine to Deep Learning: An Example

- Remember our example from the beginning
- We have a set of measured points and try to determine the underlying function (e.g. via mlp, Gaussian processor,..)
- This data might come from a measurement after some preprocessing (e.g. calibration, noise supression,...) $\rightarrow$ Information has already been provided
- What happens if we leave the preprocessing step out? $\rightarrow$ Remove one level of information
- Look at raw data and analyze it pixel by pixel



# From Machine to Deep Learning: An Example

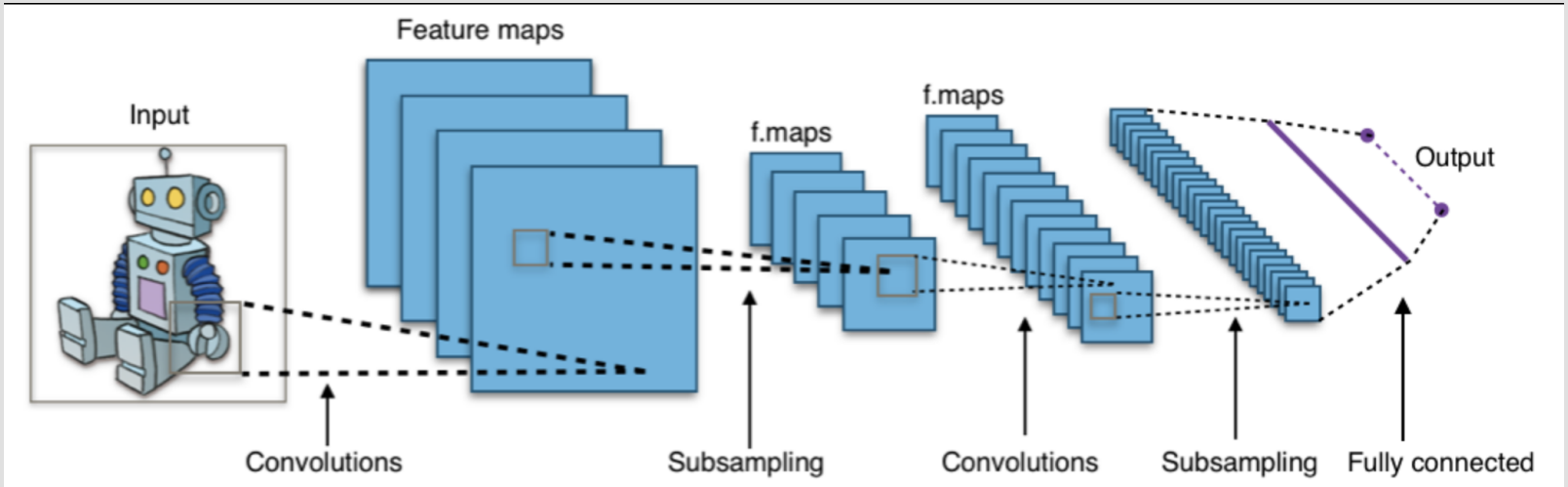
- Remember our example from the beginning
- We have a set of measured points and try to determine the underlying function (e.g. via mlp, Gaussian processor,..)
- This data might come from a measurement after some preprocessing (e.g. calibration, noise supression,...) $\rightarrow$ Information has already been provided
- What happens if we leave the preprocessing step out? $\rightarrow$ Remove one level of information
- Look at raw data and let the machine figure everything out




**Picture taken from** Aphex34 @ wikipedia

# From Machine to Deep Learning: An Example

- Remember our example from the beginning
- We have a set of measured points and try to determine the underlying function (e.g. via mlp, Gaussian processor,..)
- This data might come from a measurement after some preprocessing (e.g. calibration, noise supression,...) $\rightarrow$ Information has already been provided
- What happens if we leave the preprocessing step out? $\rightarrow$ Remove one level of information
- Look at raw data and let the machine figure everything out




**Picture taken from Aphex34 @ wikipedia**

- Many parameters ($\sim 1000\,\mathrm{k}$) to tune!

# From Machine to Deep Learning

- In a very naive picture:

# From Machine to Deep Learning

- In a very naive picture:
  - → No prior information (e.g. calibration, event selection, variable selection) is folded into the data

# From Machine to Deep Learning

- In a very naive picture:
  - → No prior information (e.g. calibration, event selection, variable selection) is folded into the data
  - → (Raw) Data is most likely larger than pre-analyzed data

# From Machine to Deep Learning

- In a very naive picture:
  - → No prior information (e.g. calibration, event selection, variable selection) is folded into the data
  - → (Raw) Data is most likely larger than pre-analyzed data
  - → The model has to do the pre-processing (e.g. figure out correlations)

# From Machine to Deep Learning

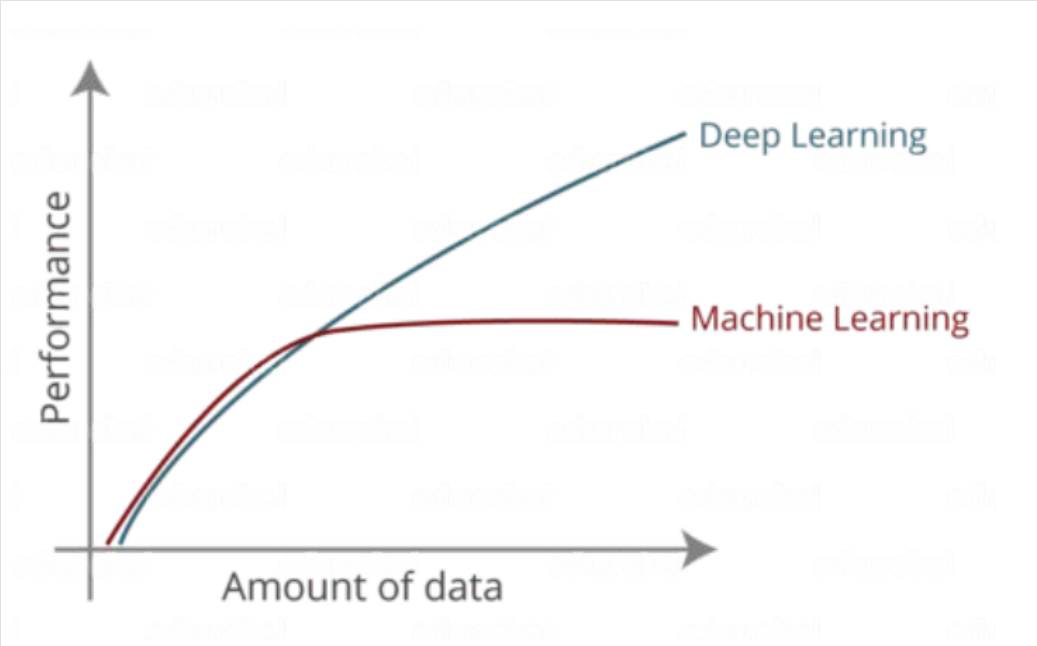
- In a very naive picture:
  - → No prior information (e.g. calibration, event selection, variable selection) is folded into the data
  - → (Raw) Data is most likely larger than pre-analyzed data
  - → The model has to do the pre-processing (e.g. figure out correlations)
  - → Use a neural network and add more (data processing) hidden layers

# From Machine to Deep Learning

- In a very naive picture:
  - → No prior information (e.g. calibration, event selection, variable selection) is folded into the data
  - → (Raw) Data is most likely larger than pre-analyzed data
  - → The model has to do the pre-processing (e.g. figure out correlations)
  - → Use a neural network and add more (data processing) hidden layers
  - → The model becomes deep

# From Machine to Deep Learning

- In a very naive picture:
  - → No prior information (e.g. calibration, event selection, variable selection) is folded into the data
  - → (Raw) Data is most likely larger than pre-analyzed data
  - → The model has to do the pre-processing (e.g. figure out correlations)
  - → Use a neural network and add more (data processing) hidden layers
  - → The model becomes deep




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

# From Machine to Deep Learning

- In a very naive picture:
  - → No prior information (e.g. calibration, event selection, variable selection) is folded into the data
  - → (Raw) Data is most likely larger than pre-analyzed data
  - → The model has to do the pre-processing (e.g. figure out correlations)
  - → Use a neural network and add more (data processing) hidden layers
  - → The model becomes deep
- Deeper model means more parameters which leads to extensive computing time

# From Machine to Deep Learning

- In a very naive picture:
  - → No prior information (e.g. calibration, event selection, variable selection) is folded into the data
  - → (Raw) Data is most likely larger than pre-analyzed data
  - → The model has to do the pre-processing (e.g. figure out correlations)
  - → Use a neural network and add more (data processing) hidden layers
  - → The model becomes deep
- Deeper model means more parameters which leads to extensive computing time
- Need sophisticated algorithms / software to handle this challenge

# From Machine to Deep Learning

- In a very naive picture:
  - → No prior information (e.g. calibration, event selection, variable selection) is folded into the data
  - → (Raw) Data is most likely larger than pre-analyzed data
  - → The model has to do the pre-processing (e.g. figure out correlations)
  - → Use a neural network and add more (data processing) hidden layers
  - → The model becomes deep
- Deeper model means more parameters which leads to extensive computing time
- Need sophisticated algorithms / software to handle this challenge
- Tensorflow / Keras are such tools to handle the building and training of deep network models

# From Machine to Deep Learning

- In a very naive picture:
  - → No prior information (e.g. calibration, event selection, variable selection) is folded into the data
  - → (Raw) Data is most likely larger than pre-analyzed data
  - → The model has to do the pre-processing (e.g. figure out correlations)
  - → Use a neural network and add more (data processing) hidden layers
  - → The model becomes deep
- Deeper model means more parameters which leads to extensive computing time
- Need sophisticated algorithms / software to handle this challenge
- Tensorflow / Keras are such tools to handle the building and training of deep network models
- The following slides will show a simple example on how to setup a mlp in Tensorflow

# Setting up the Classification MLP in Tenserflow / Keras

- In Keras, a model is defined as a sequence of layers through which you pass the data

# Setting up the Classification MLP in Tenserflow / Keras

- In Keras, a model is defined as a sequence of layers through which you pass the data

```
import tensorflow as tf

mlp_model = tf.keras.Sequential()
```

# Setting up the Classification MLP in Tenserflow / Keras

- In Keras, a model is defined as a sequence of layers through which you pass the data
- The flexibility of Keras comes into play when setting up the individual layers

# Setting up the Classification MLP in Tenserflow / Keras

- In Keras, a model is defined as a sequence of layers through which you pass the data
- The flexibility of Keras comes into play when setting up the individual layers

```
#First hidden layer:
first_hidden_layer = tf.keras.layers.Dense(
    units=(5),
    input_shape=(3,), #---> we have three input variables
    #no need to define shape for following layers
    activation='tanh',
    kernel_initializer='glorot_normal', #--> Very important!
    bias_initializer='glorot_normal'
)

another_hidden_layer = tf.keras.layers.Dense(...)

final_layer = tf.keras.layers.Dense(
    units=(3), #---> We have three outputs
    activation='softmax', #---> Same function as used in scikit
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros'
)
```

# Setting up the Classification MLP in Tenserflow / Keras

- In Keras, a model is defined as a sequence of layers through which you pass the data
- The flexibility of Keras comes into play when setting up the individual layers
- Add layers to the sequence and define optimizer

# Setting up the Classification MLP in Tenserflow / Keras

- In Keras, a model is defined as a sequence of layers through which you pass the data
- The flexibility of Keras comes into play when setting up the individual layers
- Add layers to the sequence and define optimizer

```
mlp_model.add(first_hidden_layer)
mlp_model.add(another_hidden_layer)
mlp_model.add(final_layer)

#Define loss minimization technique
my_optimizer = tf.keras.optimizers.SGD(learning_rate=0.05)

#Compile the model:
mlp_model.compile(
    optimizer=my_optimizer,
    loss="categorical_crossentropy",
    metrics=['accuracy']
)

mlp_model.summary() #--> Print model setup
```

# Setting up the Classification MLP in Tenserflow / Keras

- In Keras, a model is defined as a sequence of layers through which you pass the data
- The flexibility of Keras comes into play when setting up the individual layers
- Add layers to the sequence and define optimizer

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 5)                 20
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 18
=================================================================
Total params: 38
Trainable params: 38
Non-trainable params: 0
```

# Massaging the Data for the Model

- In scikit
  - Number of outputs set according to number of species
  - Translation of network response to label done internally

# Massaging the Data for the Model

- In scikit
  - Number of outputs set according to number of species
  - Translation of network response to label done internally
- Need to do this manually in Keras

# Massaging the Data for the Model

- In scikit
  - Number of outputs set according to number of species
  - Translation of network response to label done internally
- Need to do this manually in Keras

```
def get_label_vec(label):
    vec = [0,0,0]
    vec[int(label)] = 1 #--> label 1 = (0,1,0), label 2 = (0,01),..

    return vec

data_df['label_vec'] = data_df['label'].apply(
    lambda x: get_label_vec(x)
)
```

# Massaging the Data for the Model

- In scikit
  - Number of outputs set according to number of species
  - Translation of network response to label done internally
- Need to do this manually in Keras

| | var1 | var2 | var3 | label | label_vec |
|---|---|---|---|---|---|
| 0 | 2.140464 | 0.871710 | 1.634352 | 2.0 | [0, 0, 1] |
| 1 | 1.788192 | 1.992385 | 2.423125 | 1.0 | [0, 1, 0] |
| 2 | 0.602616 | -0.480471 | 4.399315 | 0.0 | [1, 0, 0] |
| 3 | 1.354940 | 1.914728 | 2.849413 | 1.0 | [0, 1, 0] |
| 4 | 3.008098 | 0.649694 | 1.575176 | 2.0 | [0, 0, 1] |
| 5 | 1.241234 | 0.621577 | 2.915842 | 0.0 | [1, 0, 0] |
| 6 | 1.013267 | -0.695762 | 4.493476 | 0.0 | [1, 0, 0] |
| 7 | 1.576466 | 2.001228 | 6.066541 | 0.0 | [1, 0, 0] |
| 8 | 0.920714 | 2.119301 | 2.783376 | 1.0 | [0, 1, 0] |
| 9 | 0.128073 | 1.348074 | 3.360470 | 0.0 | [1, 0, 0] |

# Training the Model

- Like in scikit, call the fit-function

# Training the Model

- Like in scikit, call the fit-function

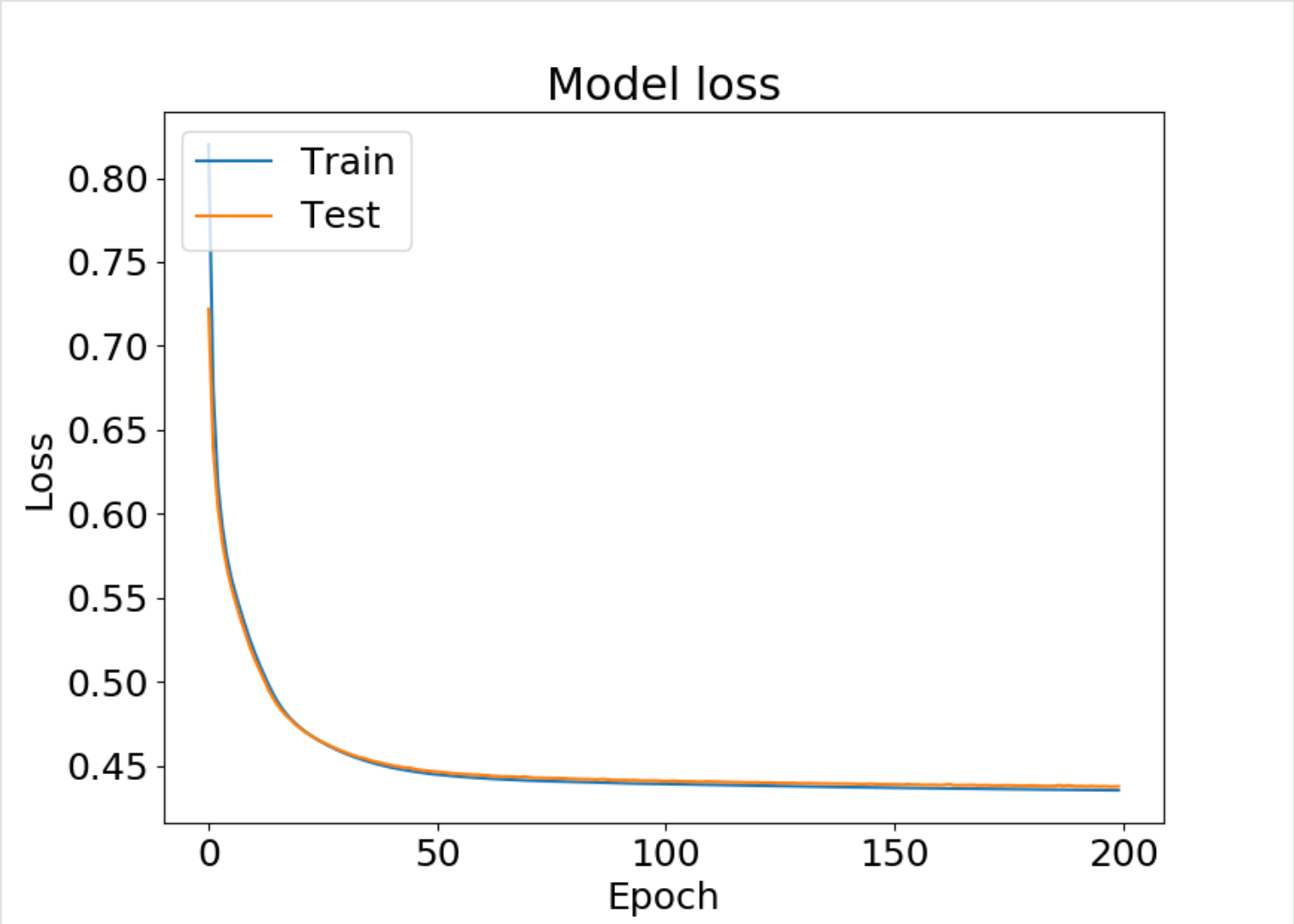
```
#Train model:
history = mlp_model.fit(
      X, #---> The three features
      Y, #---> The label vectors
      epochs=100, #--> number of epochs
      batch_size=200,
      validation_split=0.25
 )

#history summarizes the training history:
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')

plt.show()
```

# Training the Model

- Like in scikit, call the fit-function
- Check the training curve



# Obtaining the Labels

- Like in scikit, we label the event according to the largest mlp output value

# Obtaining the Labels

- Like in scikit, we label the event according to the largest mlp output value

```
def get_predicted_label(x):
    max_val = max(x) #---> Get the maximum output
    max_i = x.index(max_val) #--> Get the node index
    #which corresponds to the label

    return max_i

data_df['predicted_label'] = ...
```

# Obtaining the Labels

- Like in scikit, we label the event according to the largest mlp output value
- Quickly check whether we did everything properly

# Obtaining the Labels

- Like in scikit, we label the event according to the largest mlp output value
- Quickly check whether we did everything properly

| | var1 | var2 | var3 | label | label_vec | predicted_label_vector | predicted_label |
|---|---|---|---|---|---|---|---|
| 0 | 2.140464 | 0.871710 | 1.634352 | 2.0 | [0, 0, 1] | [0.027873897925019264, 0.011295211501419544, 0... | 2 |
| 1 | 1.788192 | 1.992385 | 2.423125 | 1.0 | [0, 1, 0] | [0.04109284281730652, 0.9282639622688293, 0.03... | 1 |
| 2 | 0.602616 | -0.480471 | 4.399315 | 0.0 | [1, 0, 0] | [0.9697380065917969, 0.013557250611484 05, 0.01... | 0 |
| 3 | 1.354940 | 1.914728 | 2.849413 | 1.0 | [0, 1, 0] | [0.0637781098484993, 0.877092957496643 1, 0.059... | 1 |
| 4 | 3.008098 | 0.649694 | 1.575176 | 2.0 | [0, 0, 1] | [0.033749695867300034, 0.01577608287334442, 0.... | 2 |
| 5 | 1.241234 | 0.621577 | 2.915842 | 0.0 | [1, 0, 0] | [0.9168961048126221, 0.037820685654878616, 0.0... | 0 |
| 6 | 1.013267 | -0.695762 | 4.493476 | 0.0 | [1, 0, 0] | [0.9510762095451355, 0.016439627856016 16, 0.03... | 0 |
| 7 | 1.576466 | 2.001228 | 6.066541 | 0.0 | [1, 0, 0] | [0.3349720239639282, 0.365845441818237 3, 0.299... | 1 |
| 8 | 0.920714 | 2.119301 | 2.783376 | 1.0 | [0, 1, 0] | [0.09435451775789261, 0.819707810878753 7, 0.08... | 1 |
| 9 | 0.128073 | 1.348074 | 3.360470 | 0.0 | [1, 0, 0] | [0.6537267565727234, 0.259109765291214, 0.0871... | 0 |

# Obtaining the Labels

- Like in scikit, we label the event according to the largest mlp output value
- Quickly check whether we did everything properly

| | var1 | var2 | var3 | label | label_vec | predicted_label_vector | predicted_label |
|---|---|---|---|---|---|---|---|
| 0 | 2.140464 | 0.871710 | 1.634352 | 2.0 | [0, 0, 1] | [0.027873897925019264, 0.011295211501419544, 0... | 2 |
| 1 | 1.788192 | 1.992385 | 2.423125 | 1.0 | [0, 1, 0] | [0.041092842817306652, 0.9282639622688293, 0.03... | 1 |
| 2 | 0.602616 | -0.480471 | 4.399315 | 0.0 | [1, 0, 0] | [0.9697380065917969, 0.013557250611484 05, 0.01... | 0 |
| 3 | 1.354940 | 1.914728 | 2.849413 | 1.0 | [0, 1, 0] | [0.0637781098484993, 0.877092957496643 1, 0.059... | 1 |
| 4 | 3.008098 | 0.649694 | 1.575176 | 2.0 | [0, 0, 1] | [0.033749695867300034, 0.01577608287334442, 0.... | 2 |
| 5 | 1.241234 | 0.621577 | 2.915842 | 0.0 | [1, 0, 0] | [0.9168961048126221, 0.037820685654878616, 0.0... | 0 |
| 6 | 1.013267 | -0.695762 | 4.493476 | 0.0 | [1, 0, 0] | [0.9510762095451355, 0.016439627856016 16, 0.03... | 0 |
| 7 | 1.576466 | 2.001228 | 6.066541 | 0.0 | [1, 0, 0] | [0.3349720239639282, 0.365845441818237 3, 0.299... | 1 |
| 8 | 0.920714 | 2.119301 | 2.783376 | 1.0 | [0, 1, 0] | [0.09435451775789261, 0.819707810878753 7, 0.08... | 1 |
| 9 | 0.128073 | 1.348074 | 3.360470 | 0.0 | [1, 0, 0] | [0.6537267565727234, 0.259109765291214, 0.0871... | 0 |

# Obtaining the Labels

- Like in scikit, we label the event according to the largest mlp output value
- Quickly check whether we did everything properly

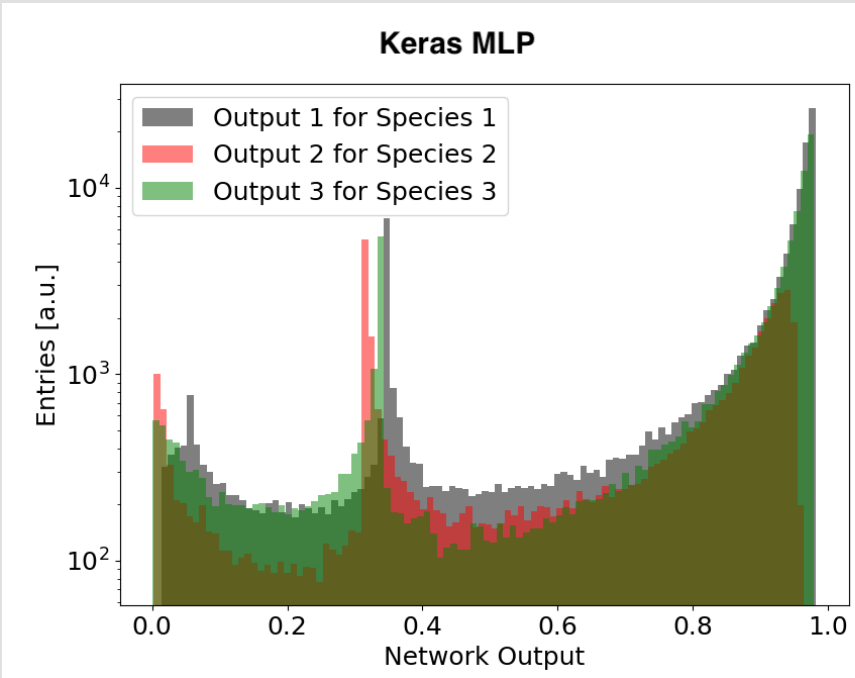
| | var1 | var2 | var3 | label | label_vec | predicted_label_vector | predicted_label |
|---|---|---|---|---|---|---|---|
| 0 | 2.140464 | 0.871710 | 1.634352 | 2.0 | [0, 0, 1] | [0.027873897925019264, 0.011295211501419544, 0... | 2 |
| 1 | 1.788192 | 1.992385 | 2.423125 | 1.0 | [0, 1, 0] | [0.04109284281730652, 0.9282639622688293, 0.03... | 1 |
| 2 | 0.602616 | -0.480471 | 4.399315 | 0.0 | [1, 0, 0] | [0.9697380065917969, 0.013557250611484405, 0.01... | 0 |
| 3 | 1.354940 | 1.914728 | 2.849413 | 1.0 | [0, 1, 0] | [0.0637781098484993, 0.877092957496643, 0.059,... | 1 |
| 4 | 3.008098 | 0.649694 | 1.575176 | 2.0 | [0, 0, 1] | [0.033749695867300034, 0.015776082873344421, 0.... | 2 |
| 5 | 1.241234 | 0.621577 | 2.915842 | 0.0 | [1, 0, 0] | [0.9168961048126221, 0.037820685654878616, 0.0... | 0 |
| 6 | 1.013267 | -0.695762 | 4.493476 | 0.0 | [1, 0, 0] | [0.9510762095451355, 0.016439627856016164, 0.03... | 0 |
| 7 | 1.576466 | 2.001228 | 6.066541 | 0.0 | [1, 0, 0] | [0.33497202396392822, 0.3658454418182373, 0.299... | 1 |
| 8 | 0.920714 | 2.119301 | 2.783376 | 1.0 | [0, 1, 0] | [0.09435451775789261, 0.8197078108787537, 0.08... | 1 |
| 9 | 0.128073 | 1.348074 | 3.360470 | 0.0 | [1, 0, 0] | [0.6537267565727234, 0.259109765291214, 0.0871... | 0 |

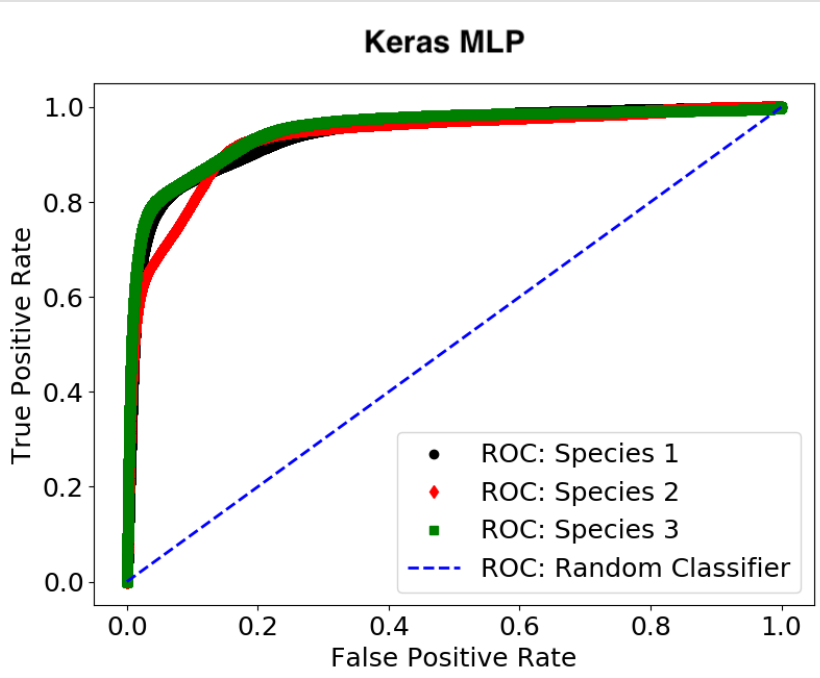
# Model Performance

- Inspect usual plots: ROC, network output, confusion matrix,..

# Model Performance

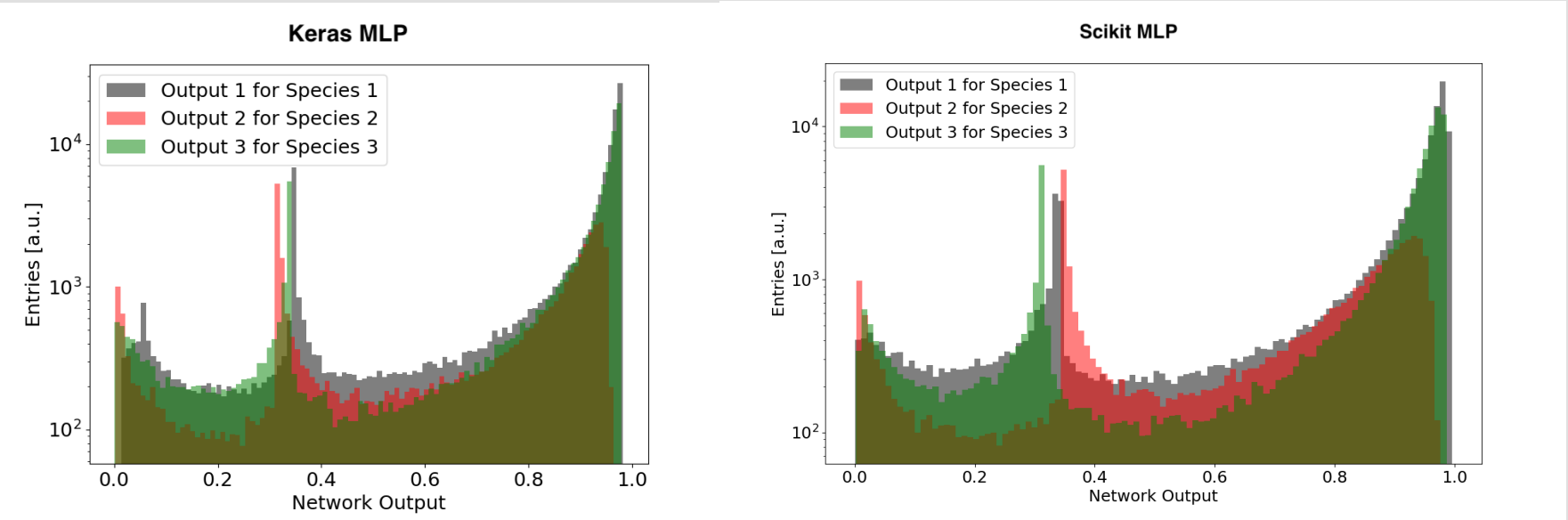
- Inspect usual plots: ROC, network output, confusion matrix,..
- And compare them to the Scikit mlp



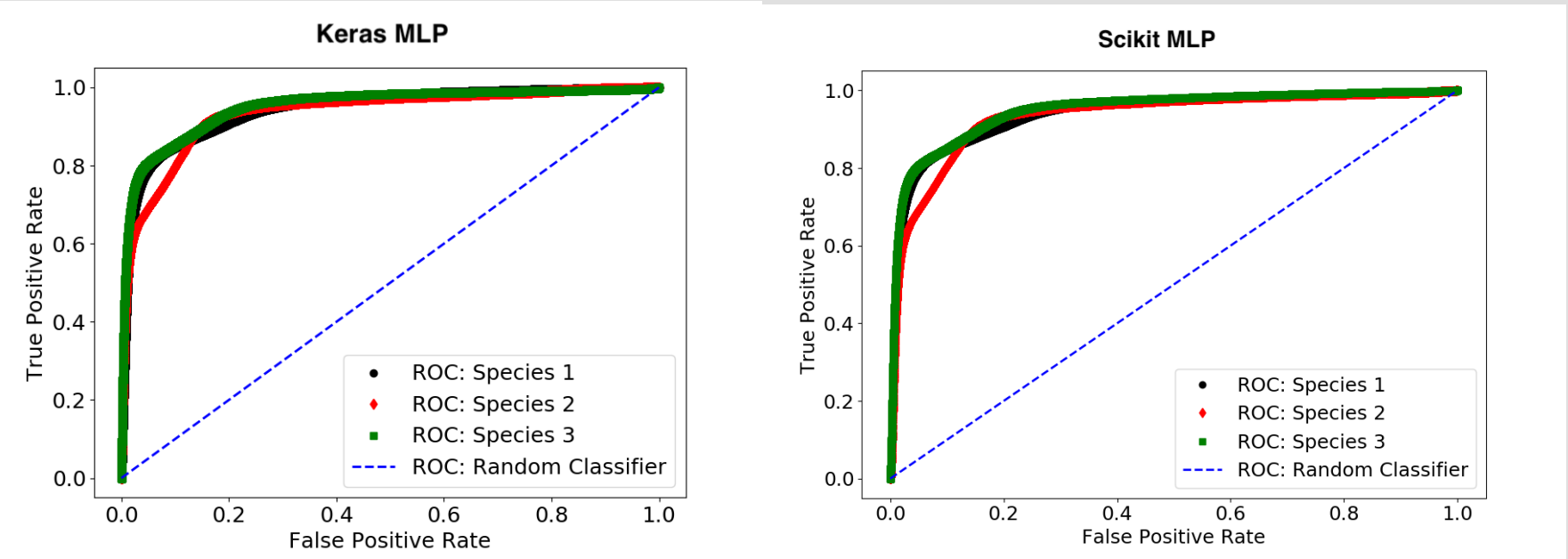
# Model Performance

- Inspect usual plots: ROC, network output, confusion matrix,..
- And compare them to the Scikit mlp



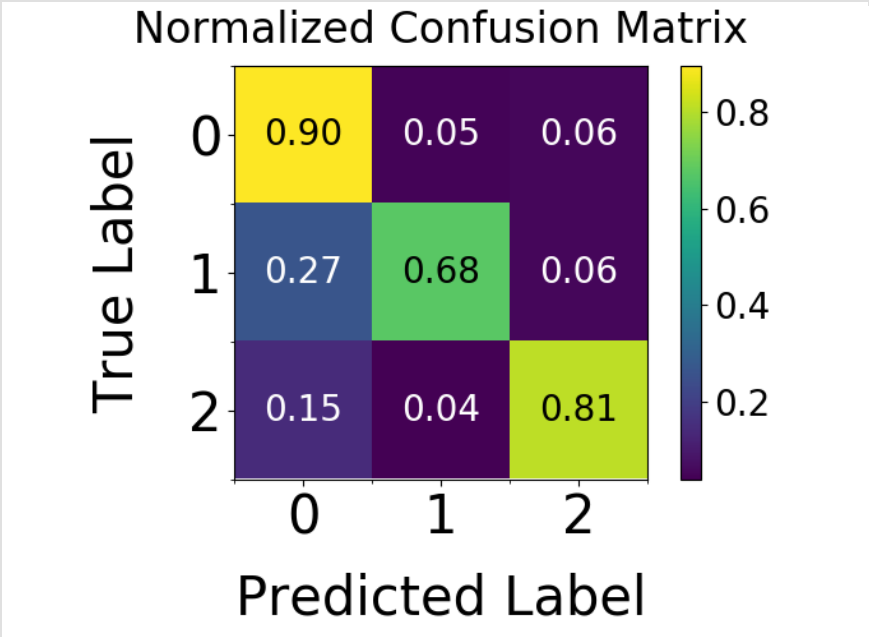
# Model Performance

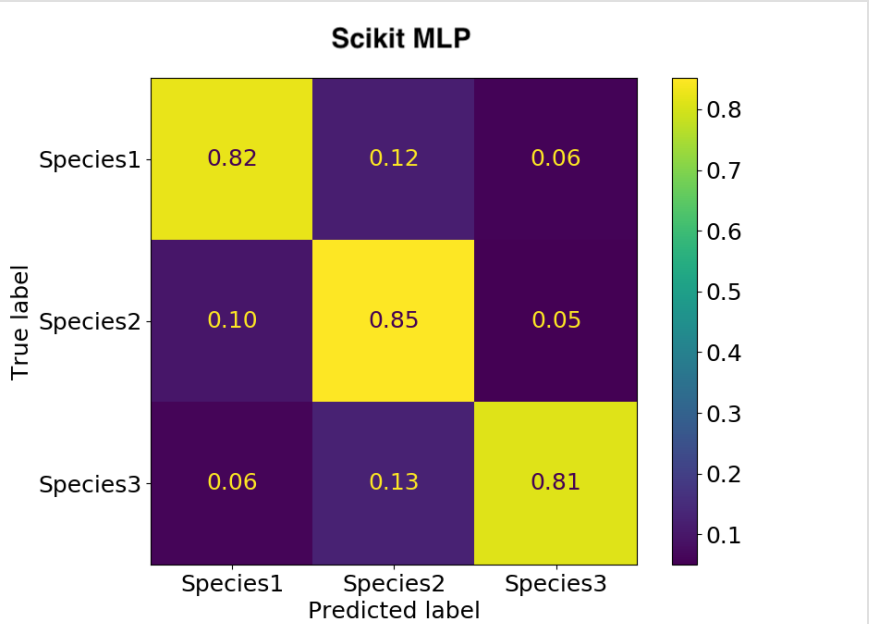
- Inspect usual plots: ROC, network output, confusion matrix,..
- And compare them to the Scikit mlp



# Model Performance

- Inspect usual plots: ROC, network output, confusion matrix,..
- And compare them to the Scikit mlp



# Model Performance

- Inspect usual plots: ROC, network output, confusion matrix,..
- And compare them to the Scikit mlp

| Model | F1-Score | Accuracy | MCC |
|---|---|---|---|
| Scikit MLP | 0.8 | 0.82 | 0.72 |
| Keras MLP | 0.81 | 0.82 | 0.72 |

# Model Performance

- Inspect usual plots: ROC, network output, confusion matrix,..
- And compare them to the Scikit mlp

| Model | F1-Score | Accuracy | MCC |
|---|---|---|---|
| Scikit MLP | 0.8 | 0.82 | 0.72 |
| Keras MLP | 0.81 | 0.82 | 0.72 |

- Both models show similar performance $\rightarrow$ Not surprising, as they have nearly identical settings

# Model Performance

- Inspect usual plots: ROC, network output, confusion matrix,..
- And compare them to the Scikit mlp

| Model | F1-Score | Accuracy | MCC |
|---|---|---|---|
| Scikit MLP | 0.8 | 0.82 | 0.72 |
| Keras MLP | 0.81 | 0.82 | 0.72 |

- Both models show similar performance $\rightarrow$ Not surprising, as they have nearly identical settings
- It is sometimes worth to compare different frameworks on the same problem $\rightarrow$ Check for consistency

# A few Words on Tensors

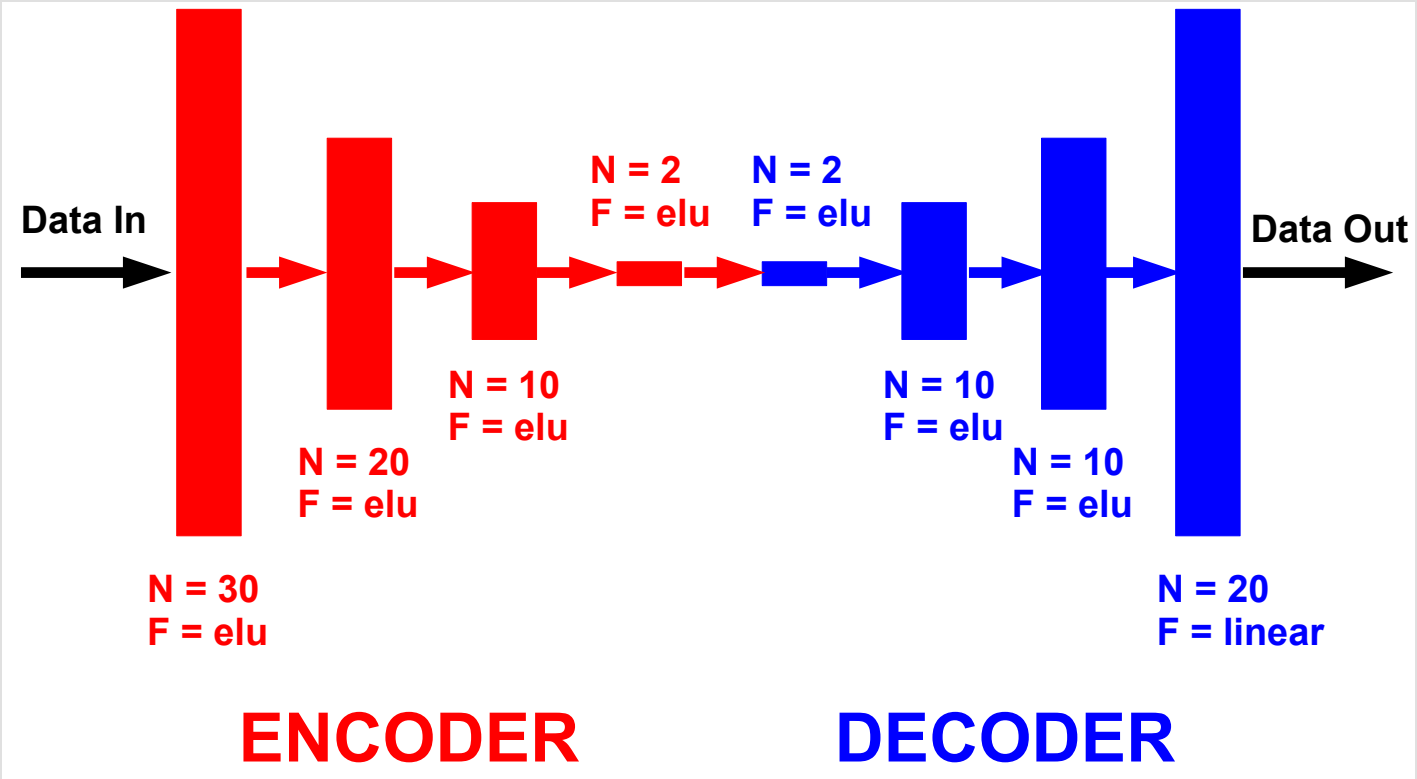
- As the name suggests, Tensorflow uses tensors which are multidimensional objects with rank and shape
- The following examples are taken from the **Tensorflow web-page**

| Rank | Shape | Dimension number | Example | |
|---|---|---|---|---|
| 0 | [] | 0-D | A 0-D tensor. A scalar. | **a single number** |
| 1 | [D0] | 1-D | A 1-D tensor with shape [5]. | **a 5-dim vector** |
| 2 | [D0, D1] | 2-D | A 2-D tensor with shape [3, 4]. | **a 3x4 matrix** |
| 3 | [D0, D1, D2] | 3-D | A 3-D tensor with shape [1, 4, 3]. | **a cube** |
| n | [D0, D1, ... Dn-1] | n-D | A tensor with shape [D0, D1, ... Dn-1]. | **I am lost...** |

- This allows for a high flexibility regarding data input and mathematical operations (i.e. weight multiplication in a deep dense neural net)
- BUT: You have to ensure that the shape of your data and your input layers match!!!

# Deep Learning in GlueX Physics

- Use autoencoder for electron / pion identification
- Features: $p$, $\theta$, $dE/dx(CDC)$, $E(BCAL)$ and $E(FCAL)$
- Number of input nodes = 5 = Number of output nodes
- Use tied weights during training $\rightarrow$ Reduce number of trainable parameters




**N: Number of neurons in dense hidden layer**
**F: Activation function in dense hidden layer**

# Machine / Deep Learning vs. Fitting a Function

| | ML / DL | Fit Function (e.g Gauss, pol(N),...) |
|---|---|---|
| Parameters | weights, nodes,... | mean, width, $p_i \cdot x^i$, ... |
| Hyper Parameters | learning rate, architecture, learning epochs, tolerance,... | order of pol., include tails, fit iterations, tolerance,... |
| Optimizer | Adam, SGD, L-BFGS,... | $\chi^2$, Log-Likelihood |
| Data Sets | training, validation,.. bootstrapping | same here |
| Performance Evaluation | Mean squared error,... | $\chi^2$/ndf,... |
| Basic question | Which model? | Which fit function? |
| ... | ... | ... |

# Machine / Deep Learning vs. Fitting a Function

- At its heart, using machine (deep) learning is not much different to fitting a function

# Machine / Deep Learning vs. Fitting a Function

- At its heart, using machine (deep) learning is not much different to fitting a function
- Machine / Deep learning algorithms are successful, because they consist of many ($\sim 100$ - $\sim 10^9$) parameters $\Rightarrow$ (Many) hyper parameters to tune $\Rightarrow$ There is no free lunch

# Machine / Deep Learning vs. Fitting a Function

- At its heart, using machine (deep) learning is not much different to fitting a function
- Machine / Deep learning algorithms are successful, because they consist of many ($\sim 100$ - $\sim 10^9$) parameters $\Rightarrow$ (Many) hyper parameters to tune $\Rightarrow$ There is no free lunch
- Provocative statement: Machine / Deep learning is:
  Reduction of N adjustable parameters to M $<<$ N adjustable hyper parameters

# Machine / Deep Learning vs. Fitting a Function

- At its heart, using machine (deep) learning is not much different to fitting a function
- Machine / Deep learning algorithms are successful, because they consist of many ($\sim 100$ - $\sim 10^9$) parameters $\Rightarrow$ (Many) hyper parameters to tune $\Rightarrow$ There is no free lunch
- Provocative statement: Machine / Deep learning is:
  Reduction of N adjustable parameters to M $<<$ N adjustable hyper parameters
- Depending on the data set (complexity / size) $\Rightarrow$ Training of ML / DL is far more challenging than a pol. fit

# Machine / Deep Learning vs. Fitting a Function

- At its heart, using machine (deep) learning is not much different to fitting a function
- Machine / Deep learning algorithms are successful, because they consist of many ($\sim 100$ - $\sim 10^9$) parameters $\Rightarrow$ (Many) hyper parameters to tune $\Rightarrow$ There is no free lunch
- Provocative statement: Machine / Deep learning is:
  Reduction of N adjustable parameters to M $<<$ N adjustable hyper parameters
- Depending on the data set (complexity / size) $\Rightarrow$ Training of ML / DL is far more challenging than a pol. fit
- Do not be afraid to use ML / DL

# Open Questions

- I want to use machine learning in my analysis. Do I have to switch to DataFrames now?

# Open Questions

- I want to use machine learning in my analysis. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.

# Open Questions

- I want to use machine learning in my analysis. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)

# Open Questions

- I want to use machine learning in my analysis. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) analysis

# Open Questions

- I want to use machine learning in my analysis. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) 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. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) analysis
- Which framework is the best for me? Or which is the best to start with?
  - Well, it depends on what you want to do

# Open Questions

- I want to use machine learning in my analysis. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) analysis
- Which framework is the best for me? Or which is the best to start with?
  - Well, 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. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) analysis
- Which framework is the best for me? Or which is the best to start with?
  - Well, 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. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) analysis
- Which framework is the best for me? Or which is the best to start with?
  - Well, 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 $\rightarrow$ Tensorflow

# Open Questions

- I want to use machine learning in my analysis. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) analysis
- Which framework is the best for me? Or which is the best to start with?
  - Well, 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 $\rightarrow$ Tensorflow
  - Of course, there is some overlap between the different frameworks

# Open Questions

- I want to use machine learning in my analysis. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) analysis
- Which framework is the best for me? Or which is the best to start with?
  - Well, 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 $\rightarrow$ 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. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) analysis
- Which framework is the best for me? Or which is the best to start with?
  - Well, 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 $\rightarrow$ 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. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) analysis
- Which framework is the best for me? Or which is the best to start with?
  - Well, 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 $\rightarrow$ 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. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) analysis
- Which framework is the best for me? Or which is the best to start with?
  - Well, 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 $\rightarrow$ 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. Do I have to switch to DataFrames now?
  - No. You can, but you do not have to.
  - Use DataFrames to prepare the training data and train/evaluate your algorithm(s)
  - Once the algorithm(s) are trained you can deploy them to your (ROOT / Java / python / ...) analysis
- Which framework is the best for me? Or which is the best to start with?
  - Well, 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 $\rightarrow$ 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. $\sim 100$ 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**

# 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**
- 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**[5]
- 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**
- 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**
- 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
- Tensorflow / Keras provides a much larger functionality than shown in this lecture (i.e. convolutional networks, regularized training, time dependent 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
- Tensorflow / Keras provides a much larger functionality than shown in this lecture (i.e. convolutional networks, regularized training, time dependent learning,...)
- 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
- Tensorflow / Keras provides a much larger functionality than shown in this lecture (i.e. convolutional networks, regularized training, time dependent learning,...)
- 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!