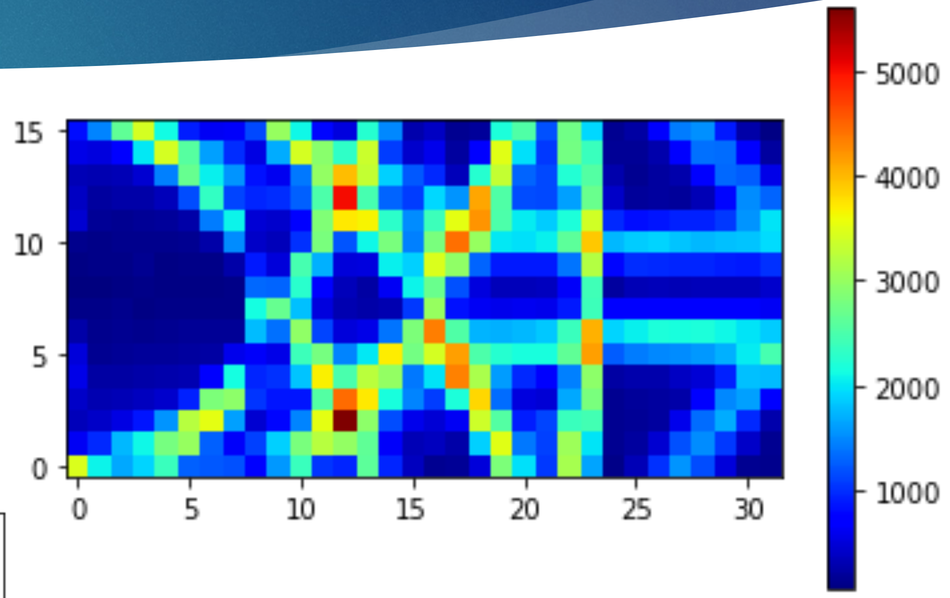
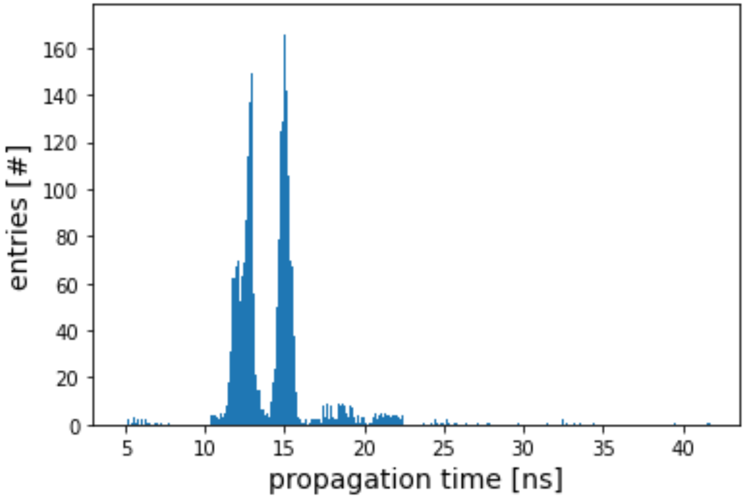
Machine learning algorithms for PID

Parameters

- Simulations provide space and time coordinates of photons
- can be extended to bigger parameter space with e.g. angles

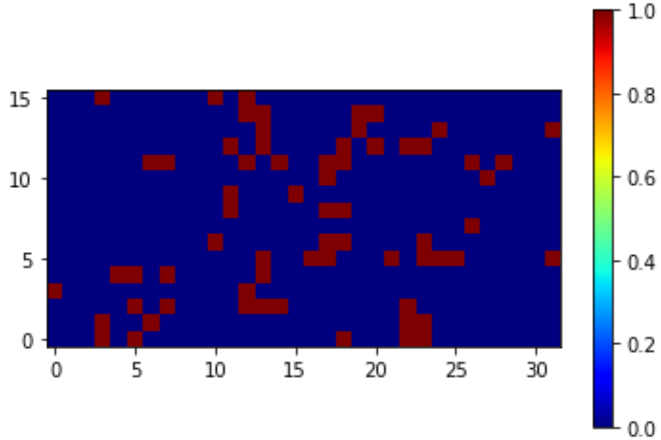




Data representations

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{511} \\ x_{512} \end{pmatrix}$$

1D-Vector with time information



Image

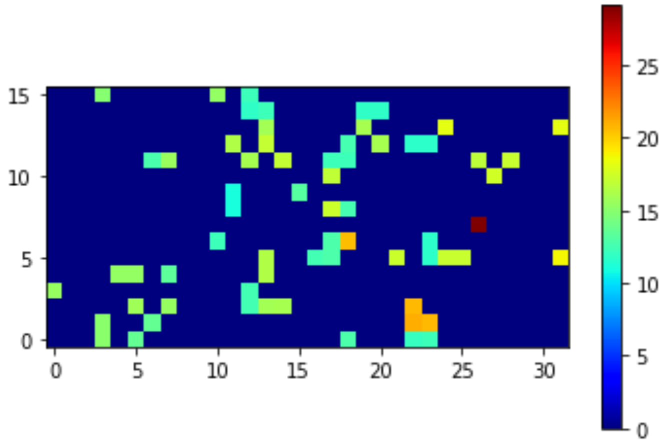
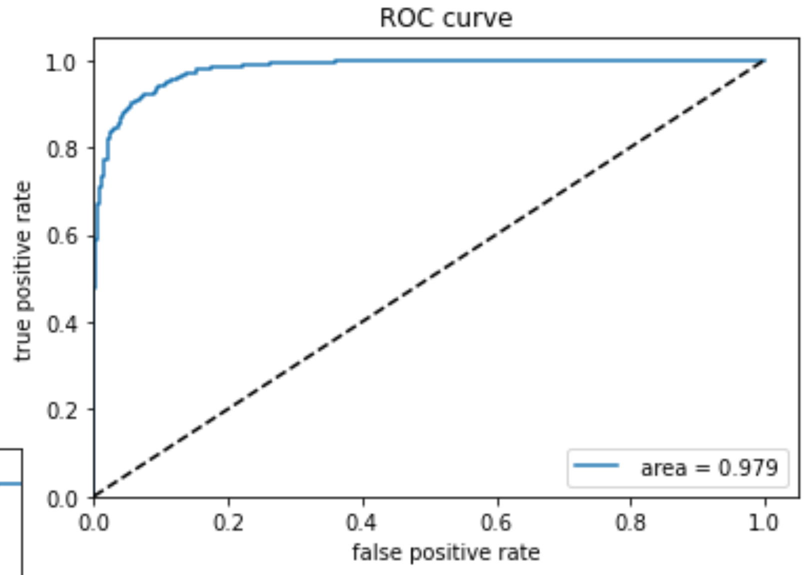
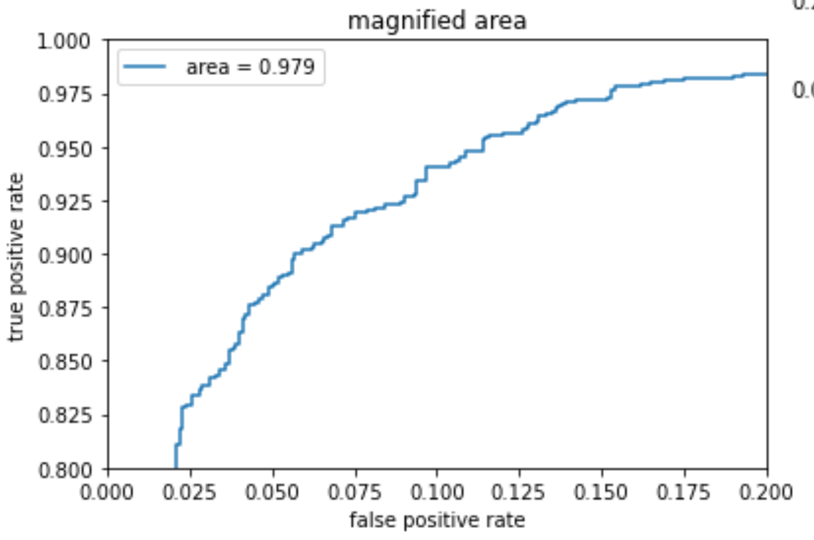


Image with time information

Evaluation of networks

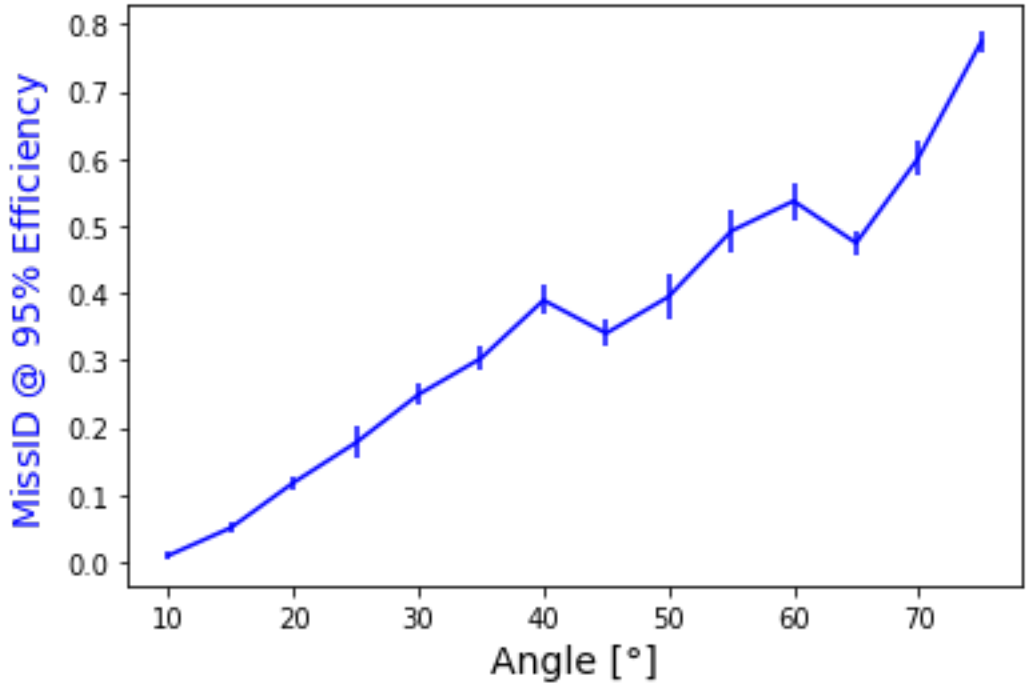
- Modify parameters to maximize performance
- Result in sep. power not directly comparable
- Results shown as ROC-Curve
- Extract Efficiency/MissId





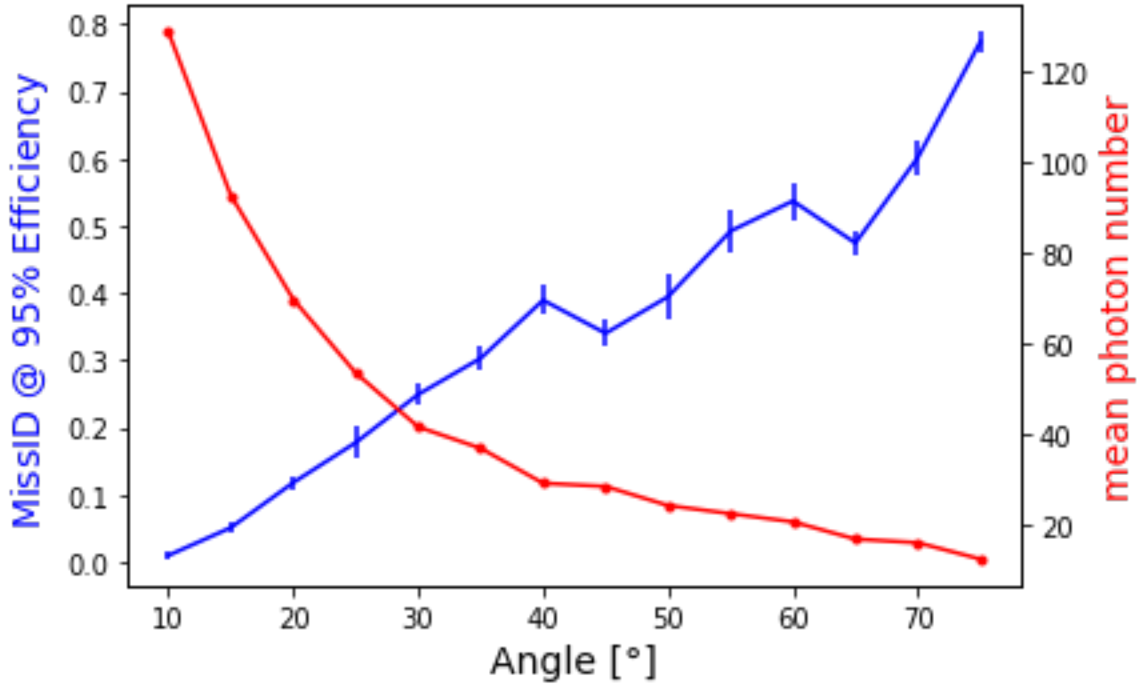
CNN with time information

- Network tested for stability over different angles
- Each angle trained separately
- MissId for 95 % efficiency best at low angles
- Probably connected to photon numbers



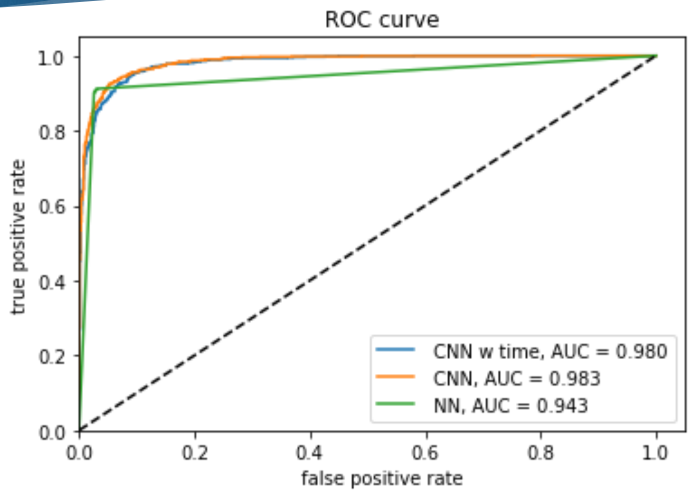
CNN with time information

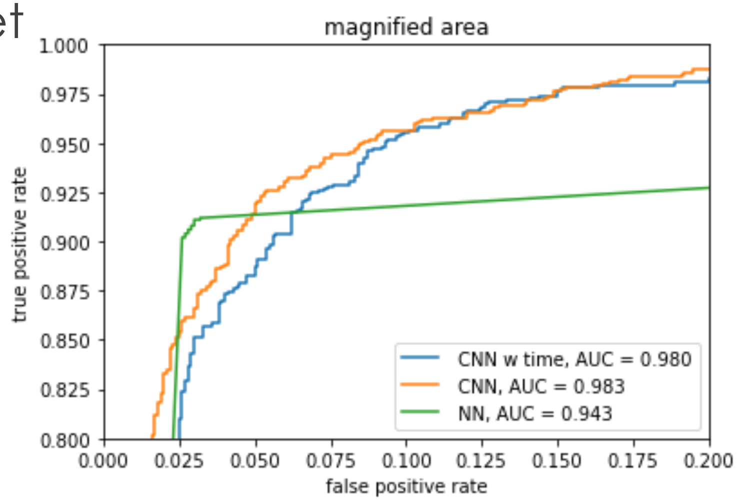
- Network tested for stability over different angles
- Each angle trained separately
- MissId for 95 % efficiency best at low angles
- Probably connected to photon numbers



Comparison of networks

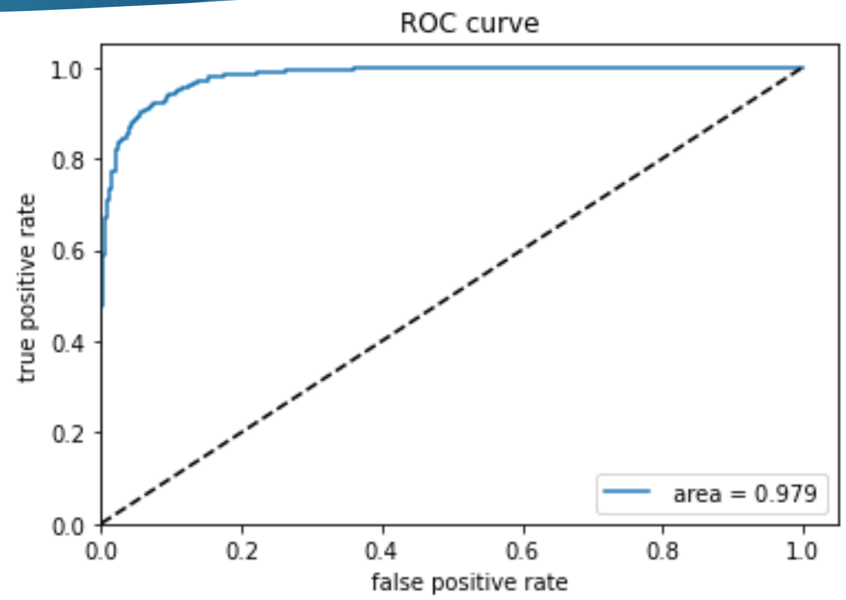
- All networks perform at the same level
- CNN with time information should perform significantly better
  - time information not interpreted
- Best suited candidate not found yet

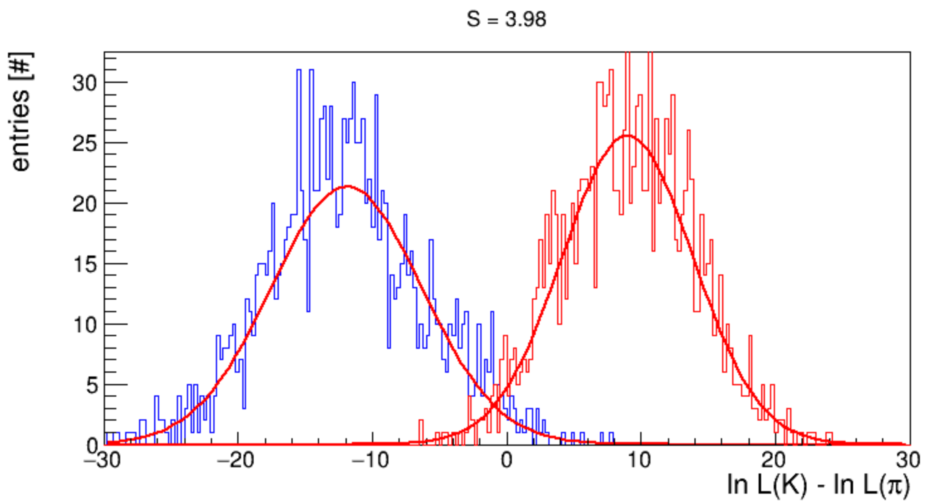




Outlook

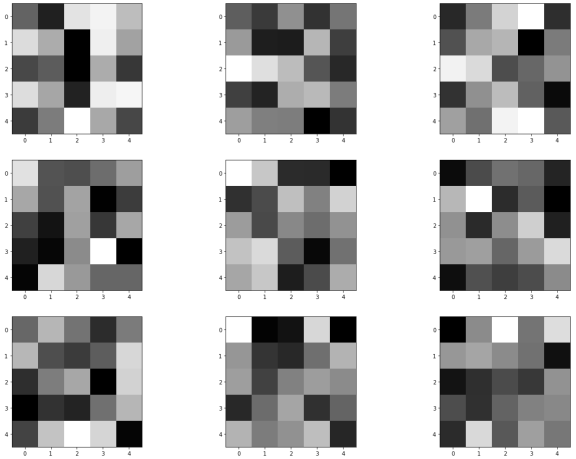
- Increase performance of networks
- Train networks over multiple angles
- Compare networks to classical algorithms

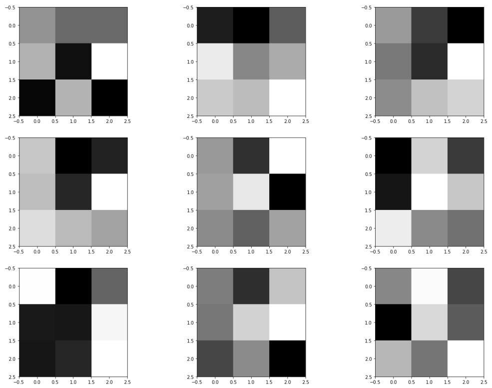




Backup

CNNs





CNNs

- Conv2D(featurenumber, kernel_size, padding)
    - extract complex characteristics
    - reduces dimensionality
- MaxPooling2D(pool_size)
    - takes maximum of given pool size
    - reduces dimensionality
- Flatten()
    - transforms convolution/pooling layer to 1D-Vector
- normal neural net

```
cnnmodel= Sequential()

cnnmodel.add(Conv2D(32,kernel_size=(5,5),activation='relu',input_shape=(16,32,1),padding="same")) #
cnnmodel.add(MaxPooling2D(pool_size=(2,2)))

cnnmodel.add(Conv2D(32,kernel_size=(5,5),activation='relu',padding="same"))
cnnmodel.add(MaxPooling2D(pool_size=(2,2)))

cnnmodel.add(Conv2D(64,kernel_size=(3,3),activation='relu',))
cnnmodel.add(MaxPooling2D(pool_size=(2,2)))

# cnnmodel.add(Conv2D(32,kernel_size=(5,5),activation='relu',input_shape=(16,32,1)))
# cnnmodel.add(MaxPooling2D(pool_size=(2,2)))

cnnmodel.add(Flatten())

cnnmodel.add(Dense(128,activation='relu'))  #128

#cnnmodel.add(Dense(20,activation='relu'))

cnnmodel.add(Dense(2,activation='softmax',name='last_layer'))

cnnmodel.summary()

#learningrate and compiling
learningrate= 0.0005        #0,001
optimizer = keras.optimizers.Adam(learningrate)
cnnmodel.compile(loss='mean_squared_error',optimizer=optimizer,metrics=['accuracy'])
#model.compile(loss='binary_crossentropy',optimizer='Adam',metrics=['accuracy'])
#model.compile(loss='mean_squared_error',optimizer='Adam',metrics=['accuracy'])
✓ 0.1s
```

CNNs

- Conv2D(featurenumber, kernel_size, padding)
  - extract complex characteristics
  - reduces dimensionality
- MaxPooling2D(pool_size)
  - takes maximum of given pool size
  - reduces dimensionality
- Flatten()
  - transforms convolution/pooling layer to 1D-Vector
- normal neural net

```
Model: "sequential_27"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_81 (Conv2D)           (None, 16, 32, 32)        832
_________________________________________________________________
max_pooling2d_81 (MaxPooling (None, 8, 16, 32)         0
_________________________________________________________________
conv2d_82 (Conv2D)           (None, 8, 16, 32)         25632
_________________________________________________________________
max_pooling2d_82 (MaxPooling (None, 4, 8, 32)          0
_________________________________________________________________
conv2d_83 (Conv2D)           (None, 2, 6, 64)          18496
_________________________________________________________________
max_pooling2d_83 (MaxPooling (None, 1, 3, 64)          0
_________________________________________________________________
flatten_27 (Flatten)         (None, 192)               0
_________________________________________________________________
dense_27 (Dense)             (None, 128)               24704
_________________________________________________________________
last_layer (Dense)           (None, 2)                 258
=================================================================
Total params: 69,922
Trainable params: 69,922
Non-trainable params: 0
_________________________________________________________________
```

Layer types

- Conv2D(featurenumber,kernel_size, padding)
- MaxPooling2D(pool_size)
  - takes maximum of given pool size
- Flatten()
  - transforms convolution/pooling layer to 1D-Vector
- Dense(NodeNumber,activation)
- dropout(droupoutpercentage)
  - part of neurons will not be considered for each cycle