Machine learning at SuperNEMO

Adam Mendl

Institute of Experimental and Applied Physics, Czech Technical University in Prague
Faculty of Mathematics and Physics, Charles University

Analysis meeting; March 23 2023

Objectives

- Cluster tracker hits into tracks without a need to fit event.
- Help Tomáš assign closest calorimeter hit to track - fit relies on time information of closest calohit, but assigning closest calohit relies on track (faster track reconstruction algorithm).
- Possibly develop ML fitting algorithm as well.

Outline

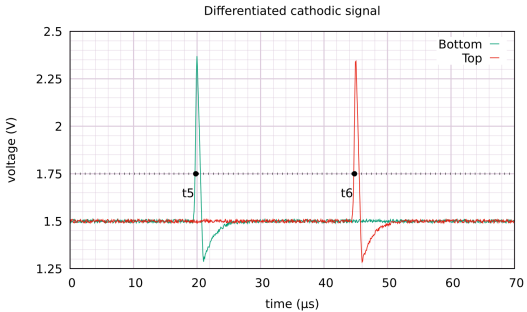
Timestamps

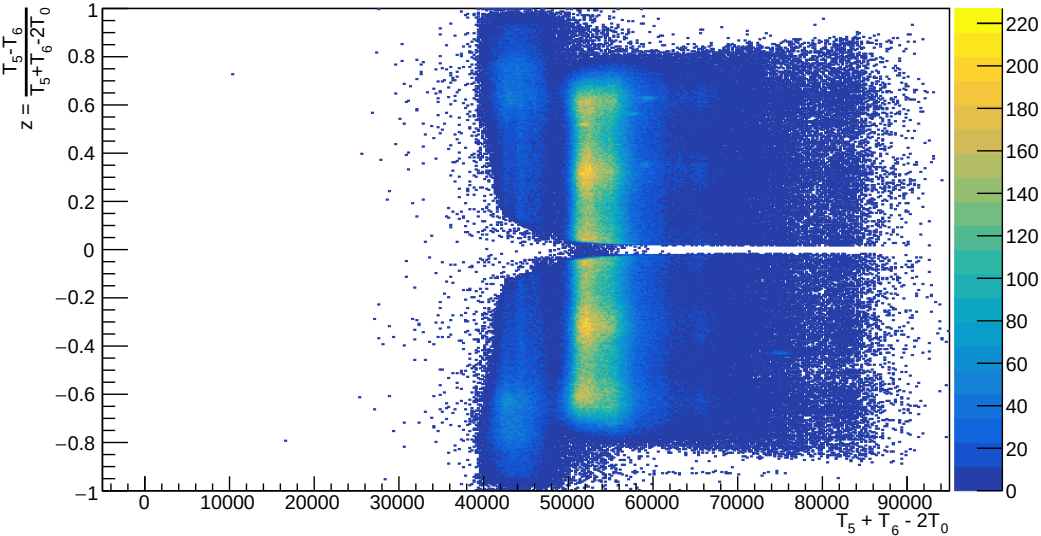
Introduction

- Tracker data has a lot of redundat information - usually used Top/Bottom cathode timestamp
- **How to replace missing (corrupted) timestamps in the most efficient manner?**
- Small project at that time but now it is becoming relevant.
- Github repository:
  `https://github.com/amendl/SuperNEMO-timestamps-analysis`
- run #728 - AREA "Tracker HV0"; 207Bi sources deployed

Introduction

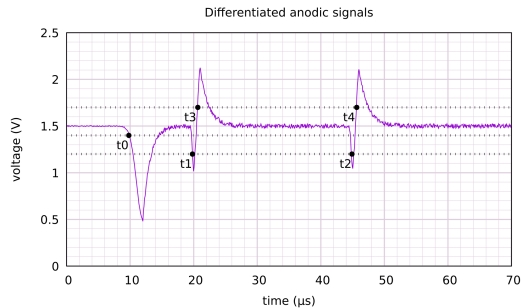
- Cathode gives us full information about z position.
- Analysed data (and physics of geiger cells) imply that propagation time of plasma doesn't have to be constant.
- relative z value
  $z = \frac{T_5 - T_6}{T_5 + T_6 - 2T_0} \in (-1, 1)$



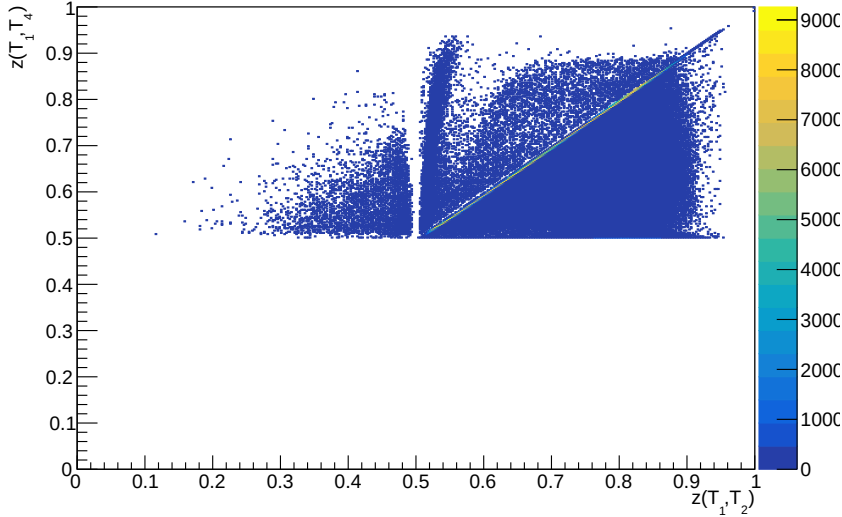

$z = \frac{T_5 - T_6}{T_5 + T_6 - 2T_0}$
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
$T_5 + T_6 - 2T_0$
220
200
180
160
140
120
100
80
60
40
20
0

Combinatorial approach

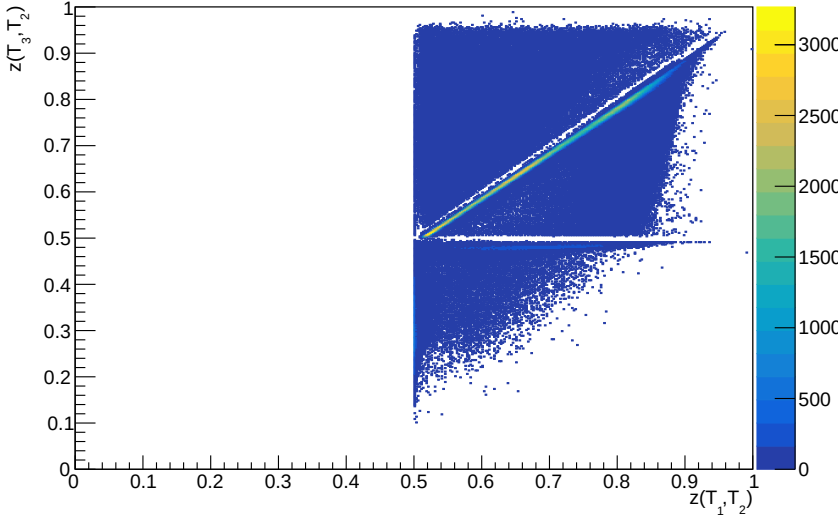
- T_0, T_1, T_2, T_3, T_4, Min = min$\{T_5, T_6\}$, Max = max$\{T_5, T_6\}$
- $T_{First} \in \{T_1, T_3, \text{Min}\}$
- $T_{Second} \in \{T_2, T_4, \text{Max}\}$
- "relative z value": $z = \frac{T_{Second} - T_0}{T_{First} + T_{Second} - 2T_0}$
- Many ways how to obtain $z(T_{First}, T_{Second})$.



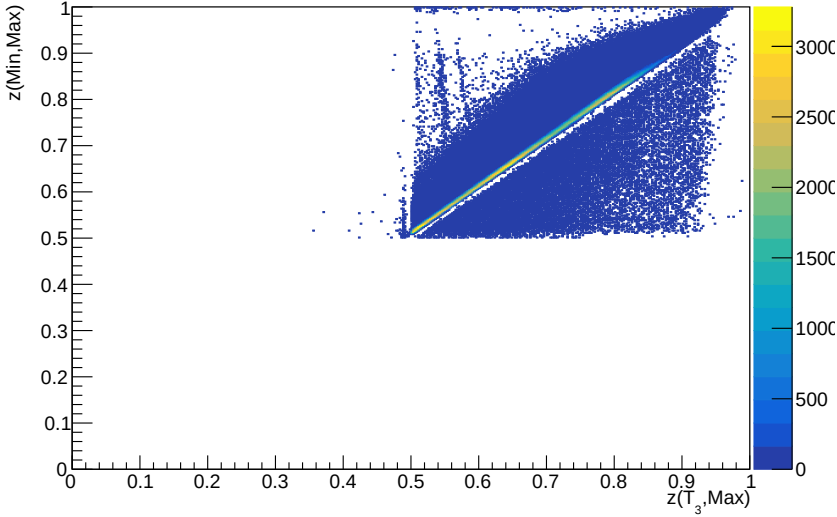
Combinatorial approach



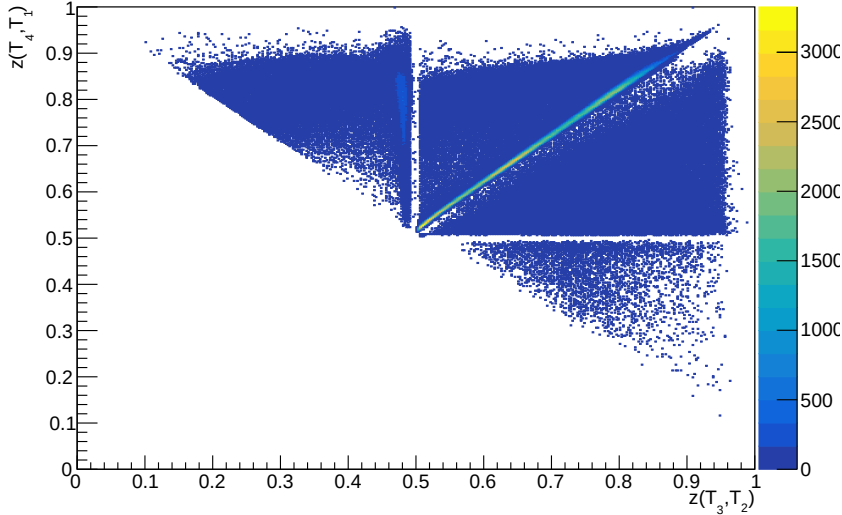
Combinatorial approach



Combinatorial approach

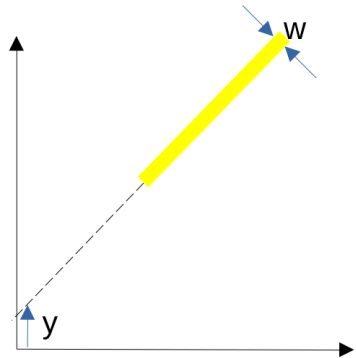


Combinatorial approach



Conclusion

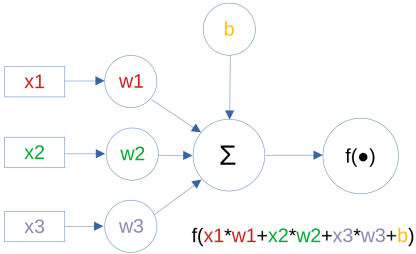
- Find offset for slope in each correlation histogram.
- Find the most reliable combination of timestamps
  - how many percents of events are in correlation line and
  - width of correlation line.



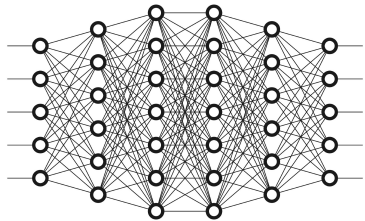
Overview of Machine Learning and its applications for SuperNEMO

Neural networks

Artificial neuron

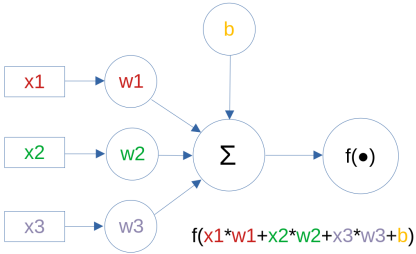


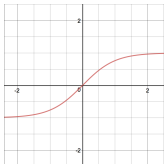
Deep fully connected neural network



Neural networks

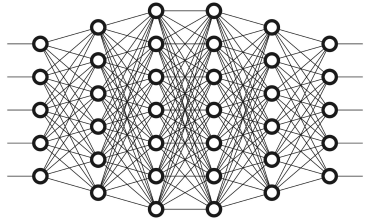
Artificial neuron





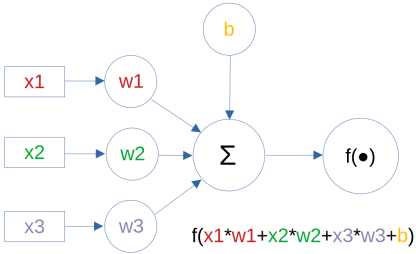
Hyperbolic tangent

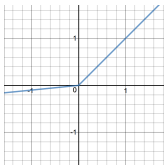
Deep fully connected neural network



Neural networks

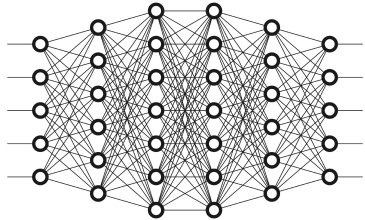
Artificial neuron





Leaky ReLU

Deep fully connected neural network



Neural networks

Loss function

- Matching output $\vec{y}$ to ground thruth $\vec{t}$

$$\ell = \sum_{i=1}^{n} (y_i - t_i)^2$$

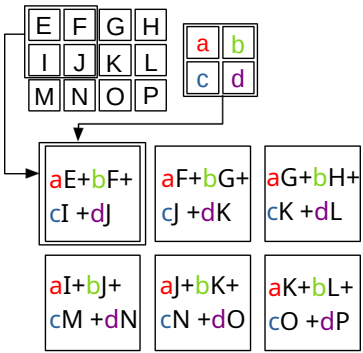
- Training reduced to optimization problem - minimizing loss function.

Backpropagation

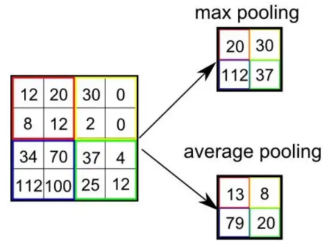
- Calculate predictions.
- Calculate loss function.
- Use chain rule to walk from last layer to first, propagating error and updating weights to make smaller errors next time.
- This is repeated many times.

Convolutional neural networks

Convolutional fiters



Pooling (max, average)



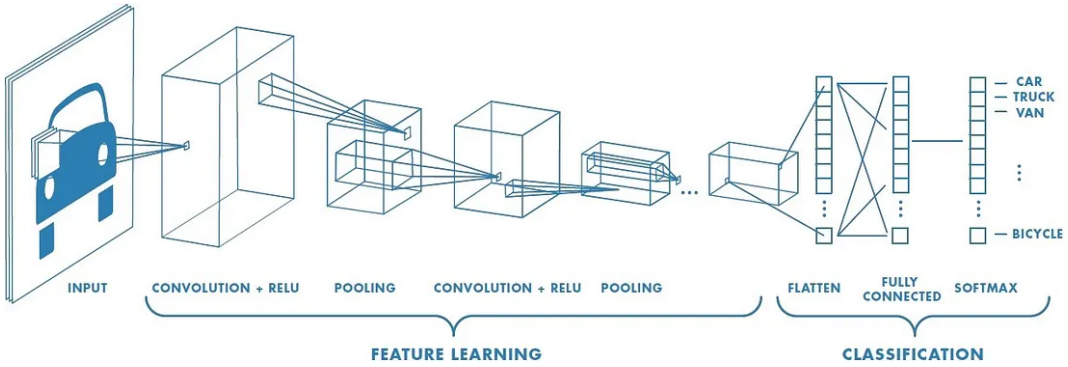
Classification tasks - One hot encoding

- n different classes for classification
- Neural network with exactly n output neurons $\{y_i\}_i^n$

$t_1 = (1, 0, 0)$
$t_2 = (0, 1, 0)$
$t_3 = (0, 0, 1)$

- Apply **Softmax** function $S(\vec{y})_i = \frac{e^{y_i}}{\sum_j e^{y_j}}$.
- Treat $\{S(\vec{y})_i\}_i^n$ as probability distribution.
- **Cross-Entropy** loss function

$$\ell_{CE} = -\sum_i^n t_i \log S(\vec{y})_i$$

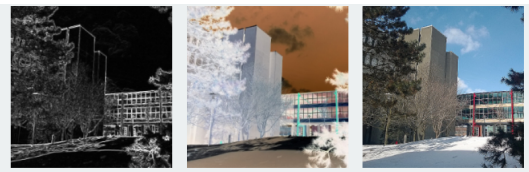

CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

- Inspiration in structure of visual cortex in human brain.
- Translation invarian, not rotation invaritant.
- 1. convulotional layer: edge detection, line detection, . . .
- 2. convolutional layer: combining features from previous layers

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Convolutional filter for edge detection

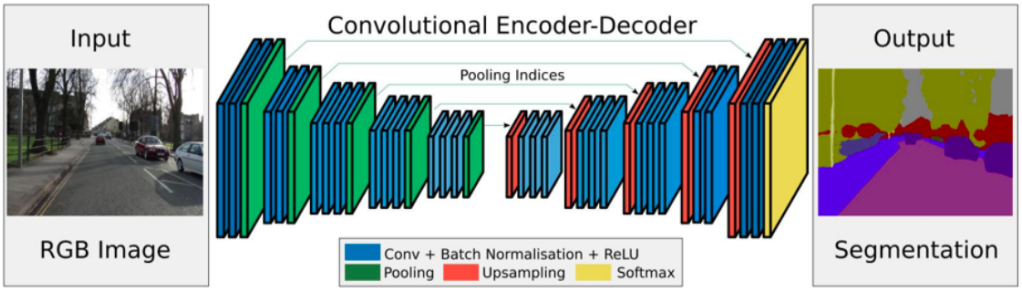


Several convolutions applied to an image

Convolutional autoencoders

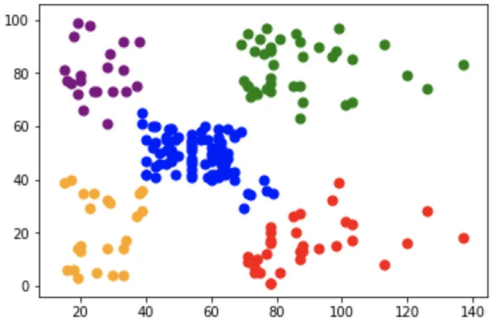
As similar as possible (minimize reconstruction error)

Input ⟵ ⟶ **Reconstruction**

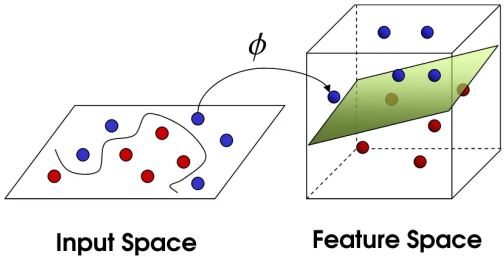


Clustering

Agglomerative clustering



Kernel trick



Number of tracks classification

		PREDICTED			
		0-TRACK	**1-TRACK**	**2-TRACK**	**3-TRACK**
TRUE	**0-TRACK**	100%	-	-	-
	1-TRACK	0.02%	99.25%	0.73%	-
	2-TRACK	-	0.38%	99.37%	0.25%
	3-TRACK	-	-	1.06%	98.94%

- Overall (unweighted) CNN accuracy: 99.58%

- AC accuracy:

  89.72%

- Overall accuracy:

  89.34%

Clustering – without neural networks

- Agglomerative clustering
- Don’t know which algorithms were used.
- Don’t know if kernel trick was used by Matteo.
- But it seems that it doesn’t work well.

Clustering – convolutional autoencoder

input

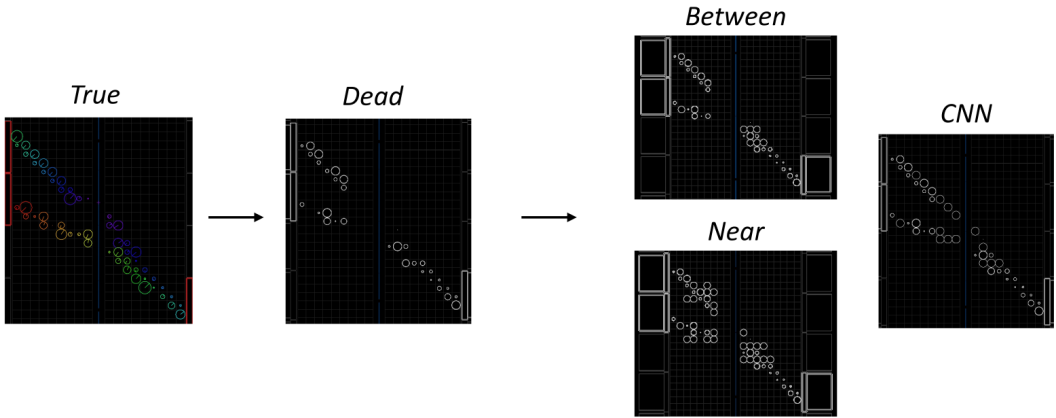
desired output

actual output

Results from Matteo

Resurrection of dead tracker cells

- Might be useful for clustering algorithm if known tracker cells don't work



Other things

Proposed by Matteo; but not sure what the results are !

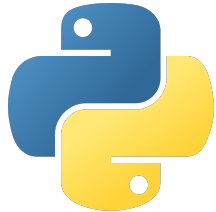
- Particle identification
- Event ID classification
- CNN for $0\nu\beta\beta/2\nu\beta\beta$ discrimination
- Vertex reconstruction

Sources from Matteo

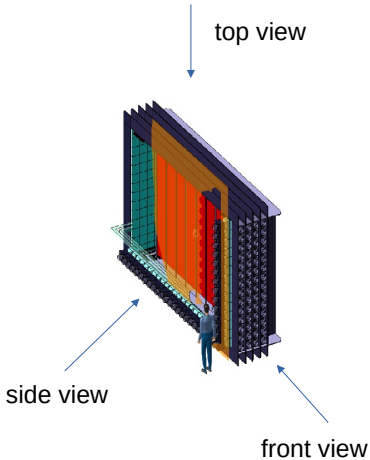
- `NemoDocDB-doc-5595-v2` – General information
- `NemoDocDB-doc-5370-v1` – General information
- `NemoDocDB-doc-5441-v1` – CNN for event ID classification
- `https://github.com/SuperNEMO-DBD/CNNforReconstruction` – still empty
- Chapter on dead cell reconstruction from draft of his thesis

My work & plans

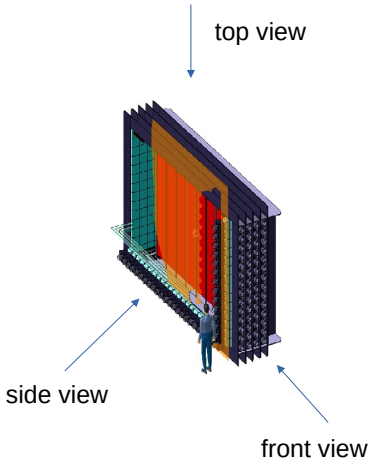
- Python
- Tensorflow; Keras
- `SN-IEGenerator` - Yorck
- Finally, I was able to train models on GPU last week.

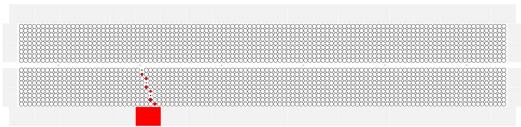


Strategy

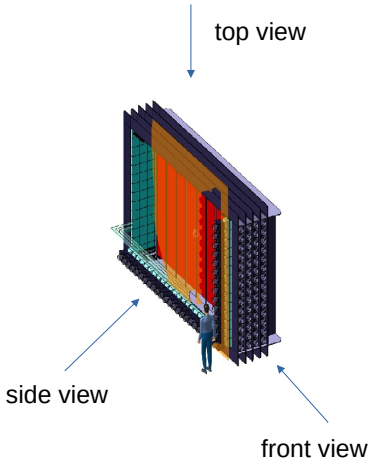


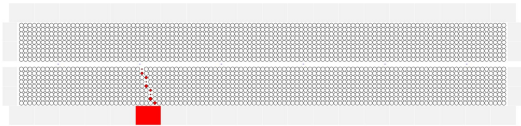
Strategy





Strategy





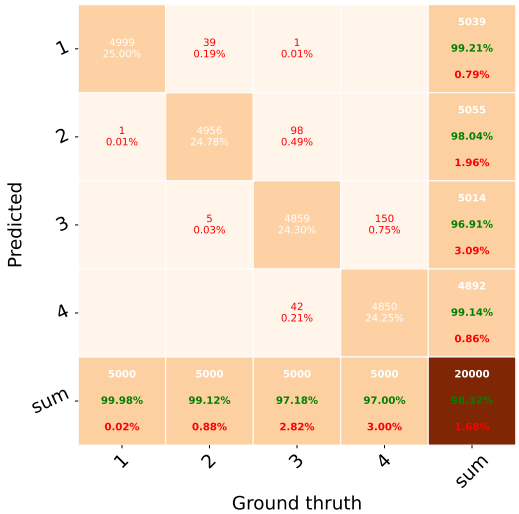
- We need to introduce discretization in z direction as well.

NN Architecture – top view

```
i = keras.Input(shape=(9,113))
img = layers.Reshape((9,113,1))(i)
x = layers.Conv2D(16,(3,15),activation = 'relu',padding="same")(img)
x = layers.MaxPooling2D(pool_size = (1,2),strides=(1,2))(x)
x = layers.Conv2D(128,(3,7),activation = 'relu',padding="same")(x)
x = layers.MaxPooling2D(pool_size = (1,2),strides=(1,2))(x)
x = layers.Conv2D(256,(3,3),activation = 'relu',padding="same")(x)
x = layers.MaxPooling2D(pool_size = (2,2))(x)
x = layers.Conv2D(256,(3,3),activation = 'relu',padding="same")(x)
x = layers.MaxPooling2D(pool_size = (2,2))(x)
x = layers.Conv2D(256,(3,3),activation = 'relu',padding="same")(x)
x = layers.MaxPooling2D(pool_size = (2,2))(x)
x = layers.Dropout(0.2)(x)
x = layers.Flatten()(x)
x = layers.Dense(128,activation='relu',use_bias=True)(x)
x = layers.Dense(64,activation='sigmoid',use_bias=True)(x)
x = layers.Dense(4)(x)
x = layers.Softmax()(x)
model = keras.Model(inputs = i, outputs = x)
```

Top view training

Trained on 180k events and tested on 20k events.

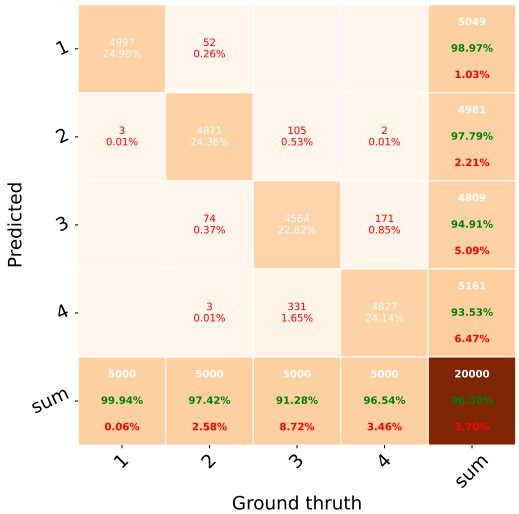


NN architecture – side view

```
i = keras.Input(shape=(30,113))
img = layers.Reshape((30,113,1))(i)
x = layers.Conv2D(16,(3,7),activation = 'relu',padding="same")(img)
x = layers.MaxPooling2D(pool_size = (1,2),strides=(1,2))(x)
x = layers.Conv2D(64,(3,3),activation = 'relu',padding="same")(x)
x = layers.MaxPooling2D(pool_size = (2,2))(x)
x = layers.Conv2D(256,(3,3),activation = 'relu',padding="same")(x)
x = layers.MaxPooling2D(pool_size = (2,2))(x)
x = layers.Conv2D(256,(3,3),activation = 'relu',padding="same")(x)
x = layers.MaxPooling2D(pool_size = (2,2))(x)
x = layers.Dropout(0.2)(x)
x = layers.Flatten()(x)
x = layers.Dense(256,activation='relu',use_bias=True)(x)
x = layers.Dense(64,activation='sigmoid',use_bias=True)(x)
x = layers.Dense(4)(x)
x = layers.Softmax()(x)
model = keras.Model(inputs = i, outputs = x)
```

Side view training

Trained on 180k events and tested on 20k events.

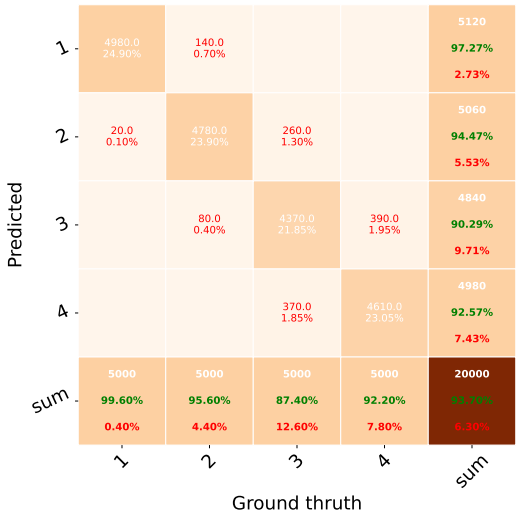


NN architecture – front view

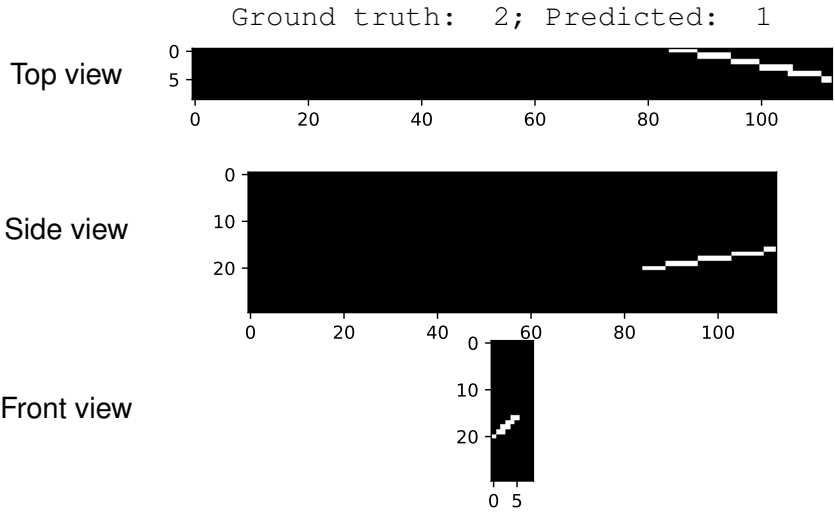
```
i = keras.Input(shape=(30,9))
img = layers.Reshape((30,9,1))(i)
x = layers.Conv2D(16,(6,2),activation = 'relu',padding="same")(img)
x = layers.Conv2D(64,(6,2),activation = 'relu',padding="same")(x)
x = layers.MaxPooling2D(pool_size = (3,1),strides=(6,2))(x)
x = layers.Conv2D(128,(2,2),activation = 'relu',padding="same")(x)
x = layers.Dropout(0.2)(x)
x = layers.Flatten()(x)
x = layers.Dense(256,activation='relu',use_bias=True)(x)
x = layers.Dense(64,activation='sigmoid',use_bias=True)(x)
x = layers.Dense(4)(x)
x = layers.Softmax()(x)
model = keras.Model(inputs = i, outputs = x)
```

Front view training

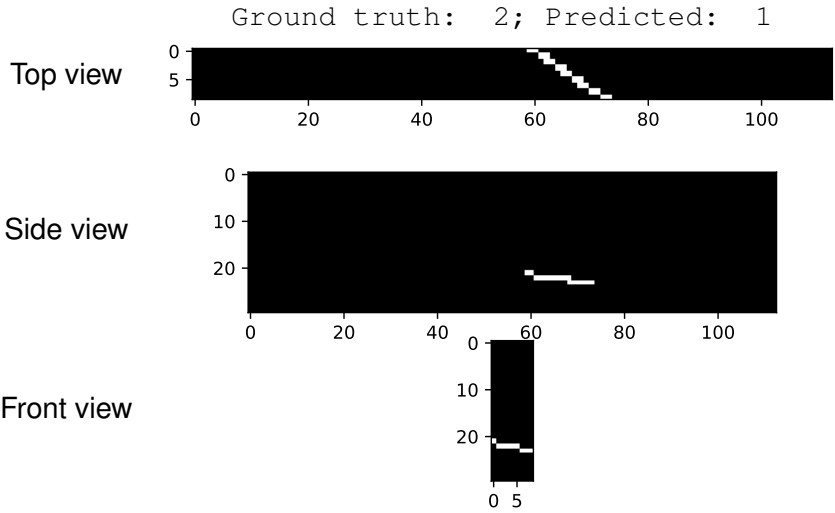
Trained on 180k events and tested on 20k events



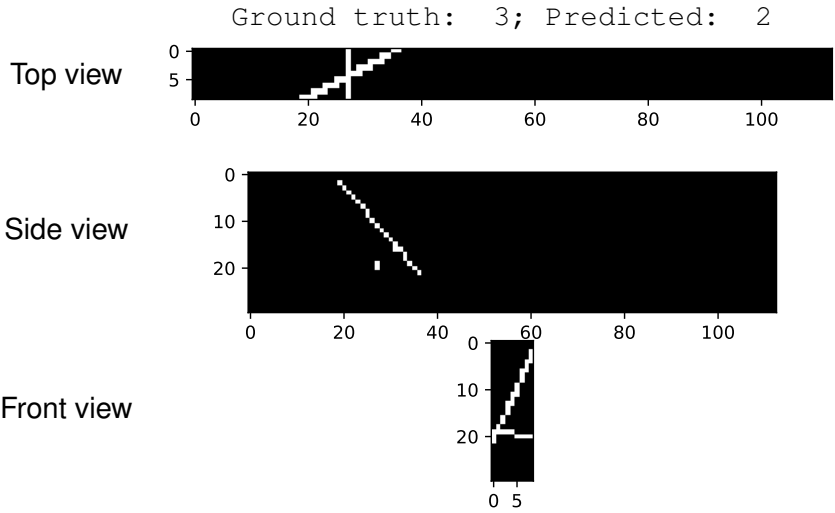
When it doesn't work - output from top model



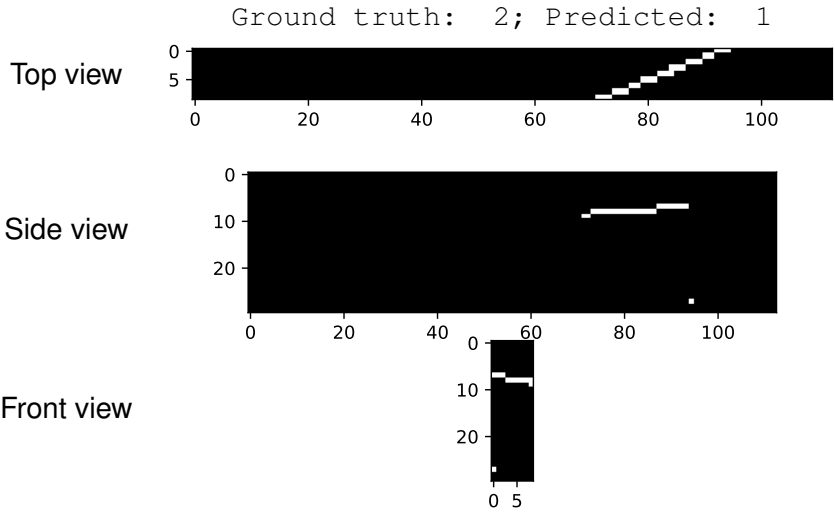
When it doesn't work - output from top model



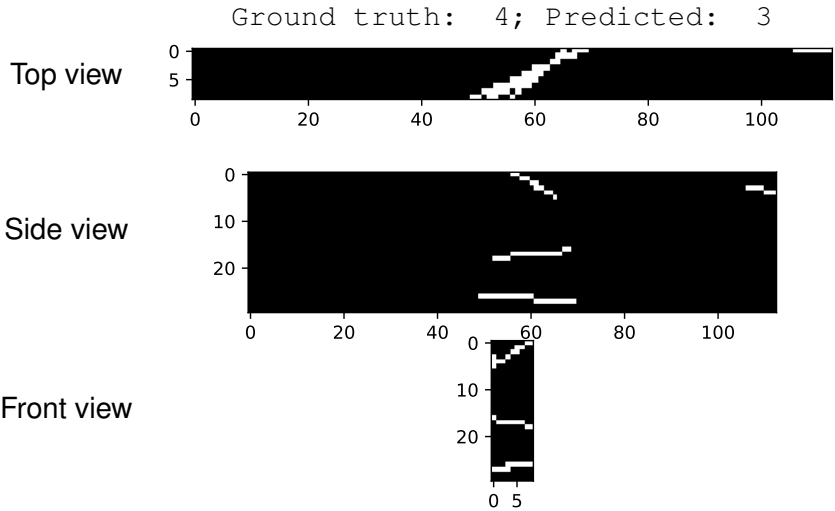
When it doesn't work - output from top model



When it doesn't work - output from top model



When it doesn't work - output from top model

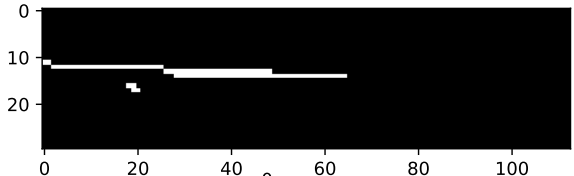


When it doesn't work - output from top model

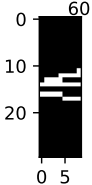
```
Ground truth:  3; Predicted:  [- - 2.7236032e-01
7.2738546e-01]
```

Top view

Side view



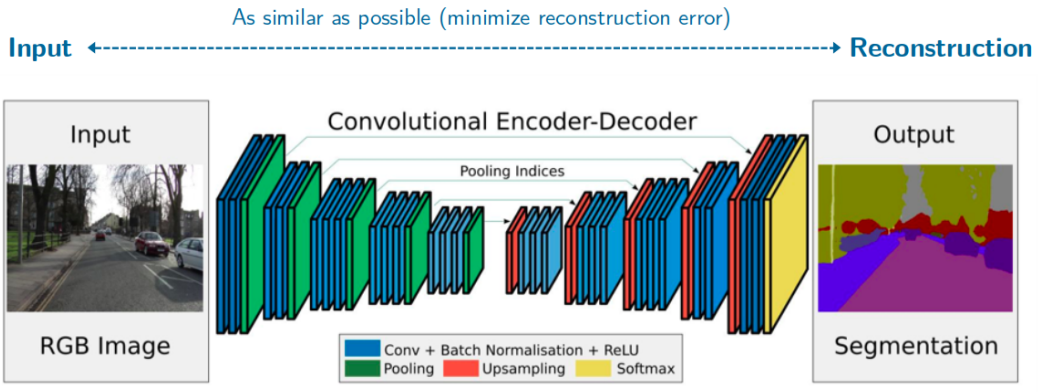
Front view



Future plan and ideas

- Repair generator (or use something else).
- Try it with noise.
- Test it on simulated data.
- Find ideal discretization in z direction.
- `Conv3D` – probably slow.
- Don’t use one hot encoding – natural ordering between classes.

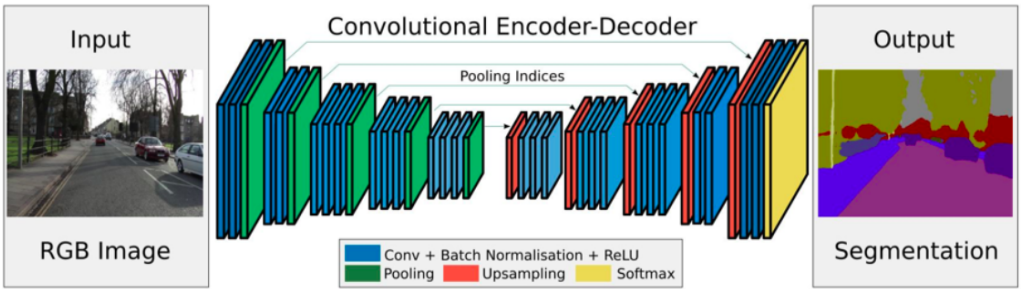
Convolutional autoencoders



Convolutional autoencoders

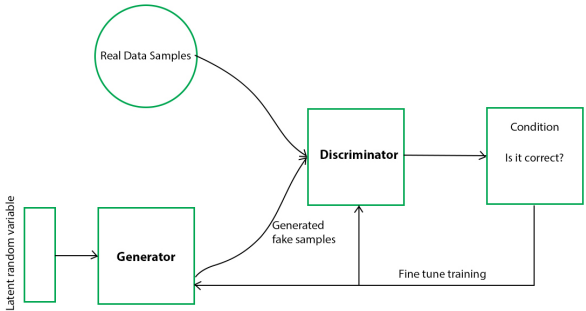
As similar as possible (minimize reconstruction error)

Input ⟷ **Reconstruction**



How to define closeness? How to cope with permutations of tracks?

Generative adversarial networks



- Discriminator tries to distinguish between real data samples and artificially generated samples.

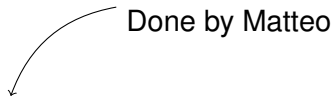
- Generator tries to fool discriminator

Strategy

- Convolutional discriminator
- Generator from convolutional autoencoder
- Two options
  - for every number of tracks train separate model creating all clusters, or
  - try to cluster only one track and continue iteratively.

Strategy

- Convolutional discriminator
- Generator from convolutional autoencoder
- Two options
  - for every number of tracks train separate model creating all clusters, or
  - try to cluster only one track and continue iteratively.



Strategy

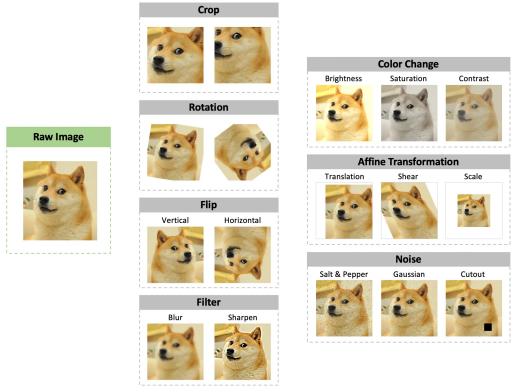
- Convolutional discriminator
- Generator from convolutional autoencoder
- Two options
  - for every number of tracks train separate model creating all clusters, or
  - try to cluster only one track and continue iteratively.

Done by Matteo

Should solve both objective of output and similarity of input and output at once.

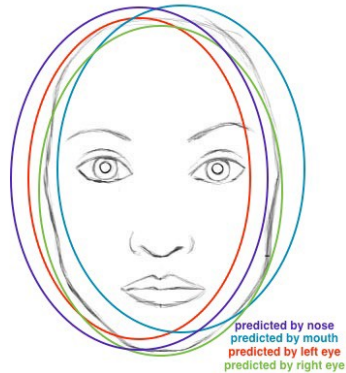
Convolution should not work

- Convolutional filters cannot be rotated nor deformed.
- Sometimes solved by rotating, inverting and deforming training images.



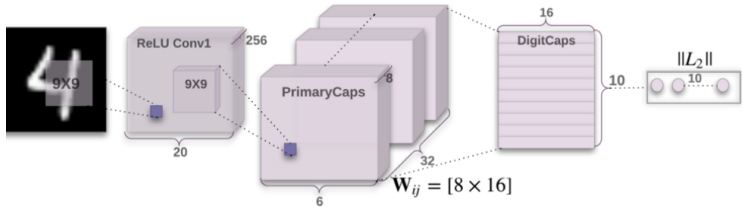
Convolution should not work

- Convolutional filters cannot be rotated nor deformed.
- Sometimes solved by rotating, inverting and deforming training images.
- Capsule network adds affine transformations as parameters.
- Handling slopes of tracks well.
- Less dependent on z discretization.



CapsNet

- Capsule works like filter in CNN, but encodes size and rotation



Architecture with understanding of slope might be great for autoencoders and track reconstruction.

`Series of articles on CapsNet, Preprint on Routing algorithm`

Final remarks

Some issues with ML

- Usualy not able to estimate uncertainties with mainstream ML algorithms
- Obtained uncertainties does not have anything in common with probability in strict mathematical sense.
- For many applications ground thruth is hard to define
- Still not sure about speed of inference on GPU.

Relevant files and links

`[AI] := /sps/nemo/scratch/amendl/AI/my_lib`

- `[AI]/datasets/number_of_tracks.py` – Loading data from toyhaystack root files
- `[AI]/number_of_tracks_classification.py` – Main script for classification tast (top/side/front model)
- `[AI]/combined.py` – Combining and training top, side and front model

Putting code to github:

`https://github.com/amendl/SuperNEMO_ML_applications`

Thank you for your attention