Track finding with Deep Neural Networks

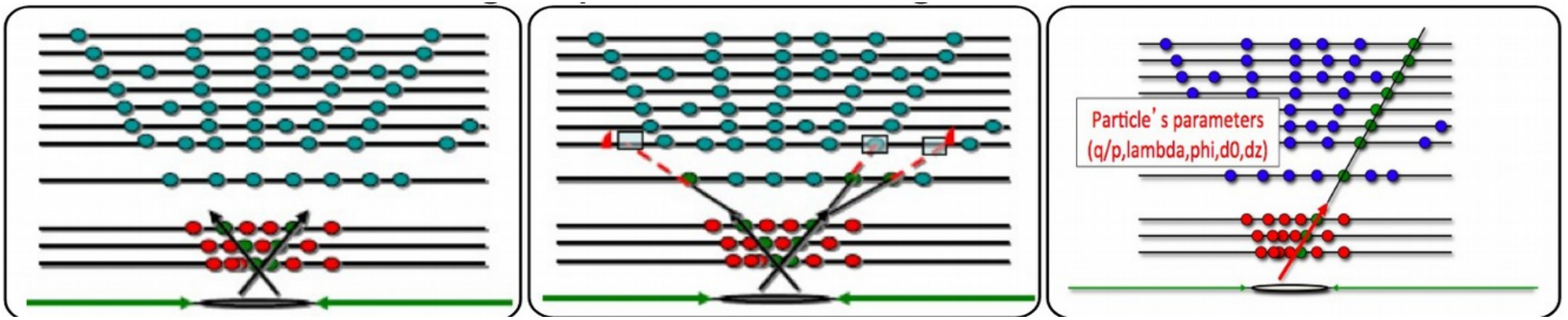
Marcin Kucharczyk, Marcin Wolter

Institute of Nuclear Physics PAN, Kraków

KUKDM 2019

Zakopane, 7 March 2019

Pattern Recognition in High Energy Physics

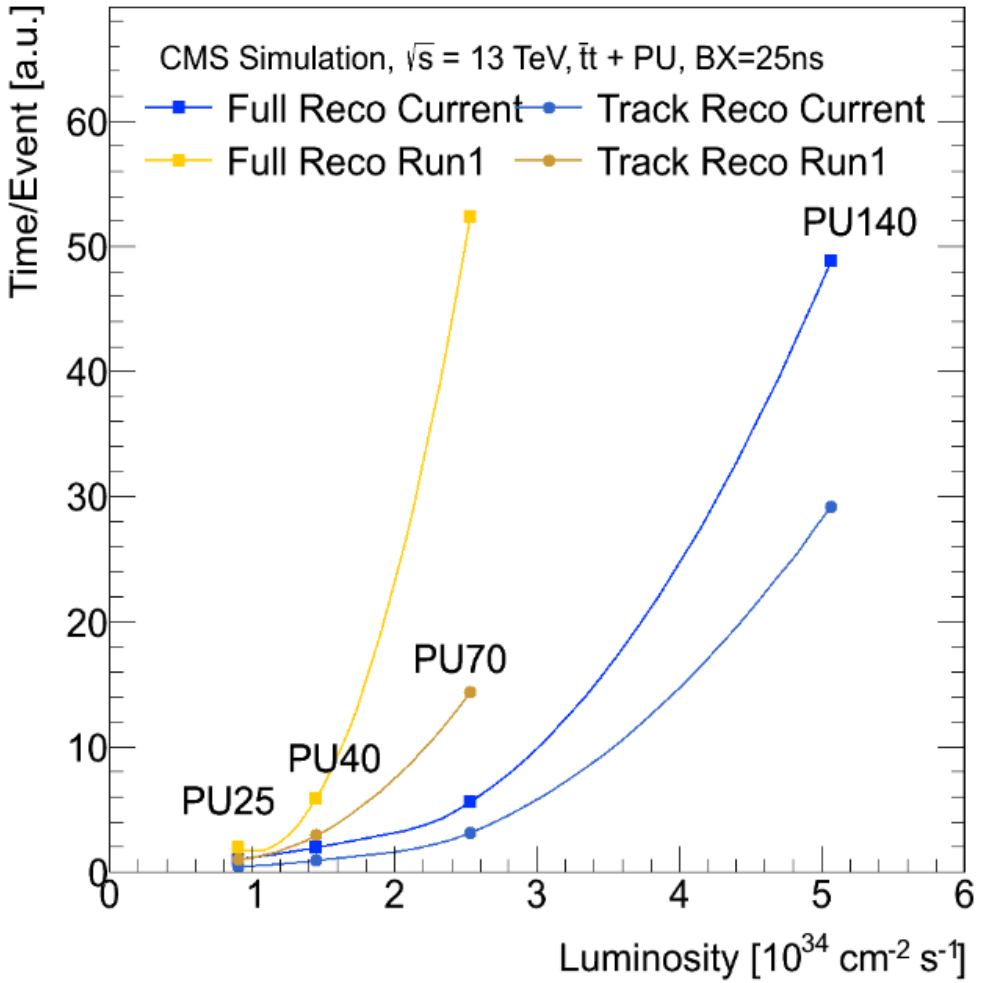


Seeding **Track Building** **Track Fitting**

- **Track seeding** – finding the seeds (initial sets of hits) from which the track starts
- **Track building** ≡ pattern recognition HEP jargon
  - Creating a 2- and 3-dimensional lines and assigning to them all the hits within a certain window
  - Fitted frequently with “robust fit”
- **Track fitting** – final fitting of the track parameters (usually a Kalman filter used for tracking)

Usually this method works fine, is robust and efficient!

So, where is the problem?



CMS experiment simulation

J.-R. Vlimant, Machine Learning for Charged Particle Tracking, MIT, 2018

- The time needed to process one event grows quickly (worse than quadratic) with luminosity (number of collisions).
- Huge part of CPU consumption is the track finding.

Deep Neural Networks (DNN)?

- Fast, parallel, in principle do pattern recognition “at once”, without looping over hits.
- Also experiments with lower occupancy might profit from DNN’s – higher precision and efficiency.
- There is a HEPTrkx group working on tracking for HEP experiments: *https://heptrkx.github.io/*

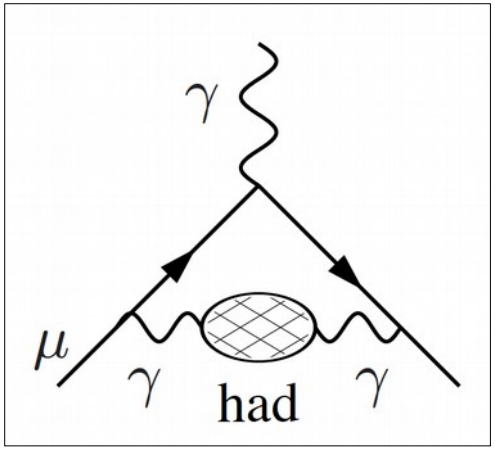
Our goal - application to MuonE

but first to the test-beam data

- Implement DNN to MUonE experiment at SPS, CERN.
- MUonE dedicated to measure a hadronic correction to the anomalous muon magnetic moment.

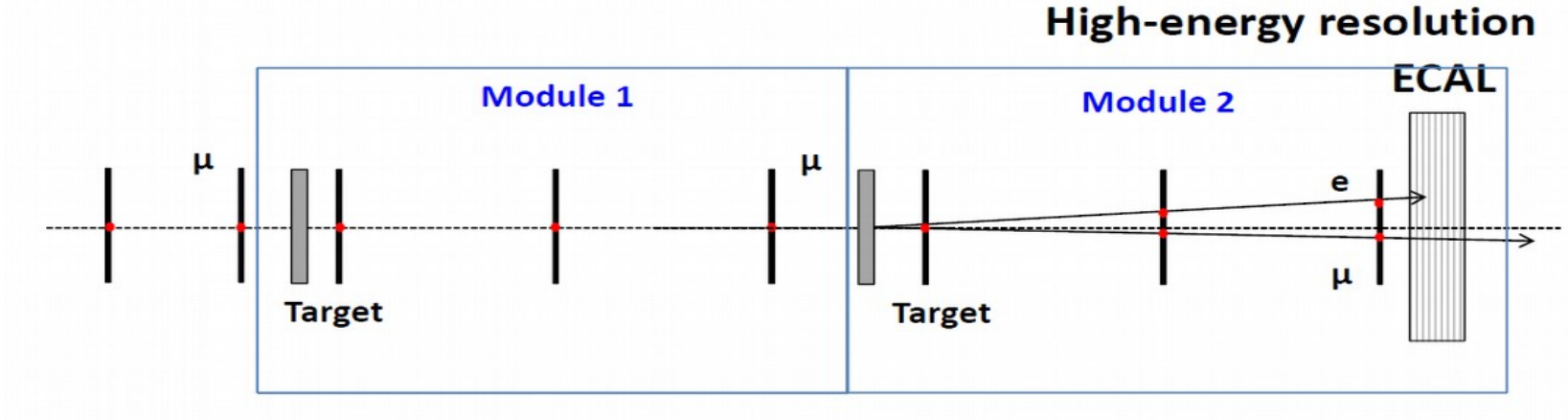
  →in order to increase sensitivity to the potential New Physics phenomena which may cause an observed discrepancy with respect to the Standard Model predictions.

hadronic vacuum polarization



- **Need very good precision and tracking efficiency!**
- **Apply DNN first to the experimental test-beam data taken in 2017 and 2018.**
- DNN may provide fast and efficient pattern recognition.

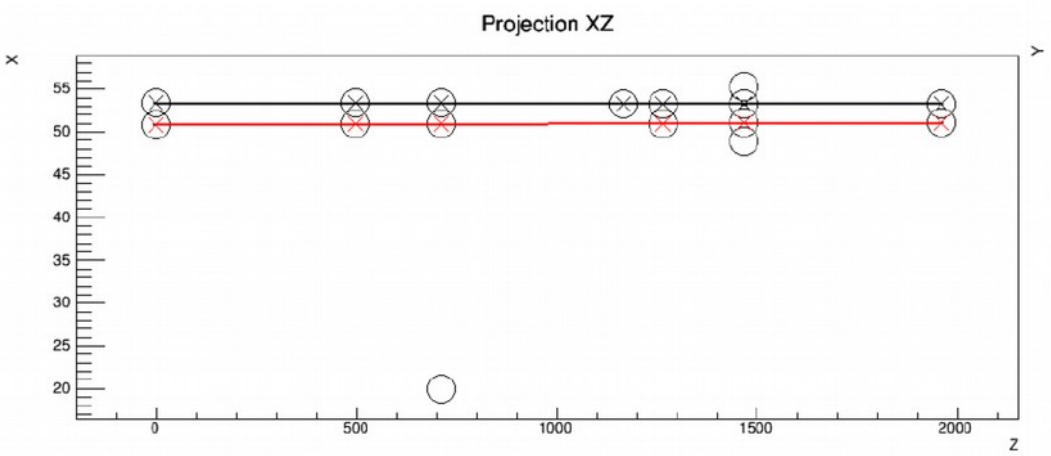
  → the most crucial step in the track reconstruction procedure.

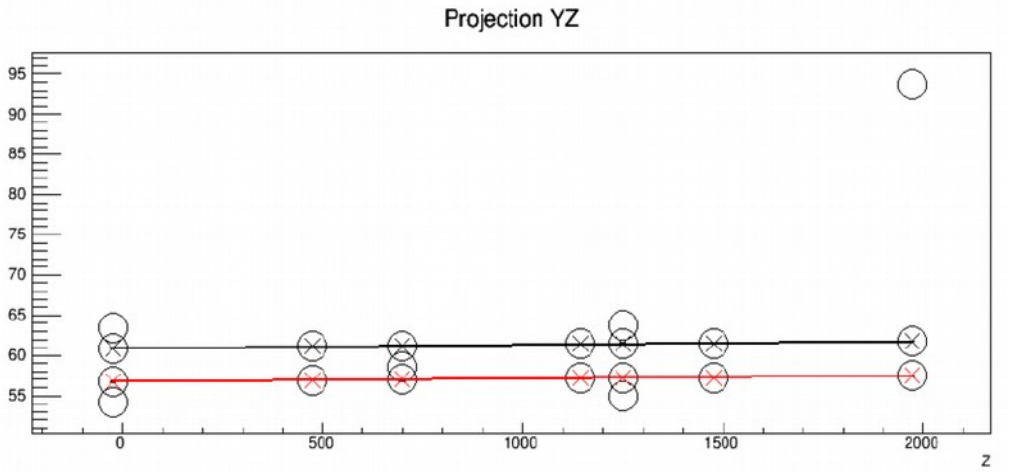


Our goal - application to MUonE

Pattern recognition - our 'classical' approach

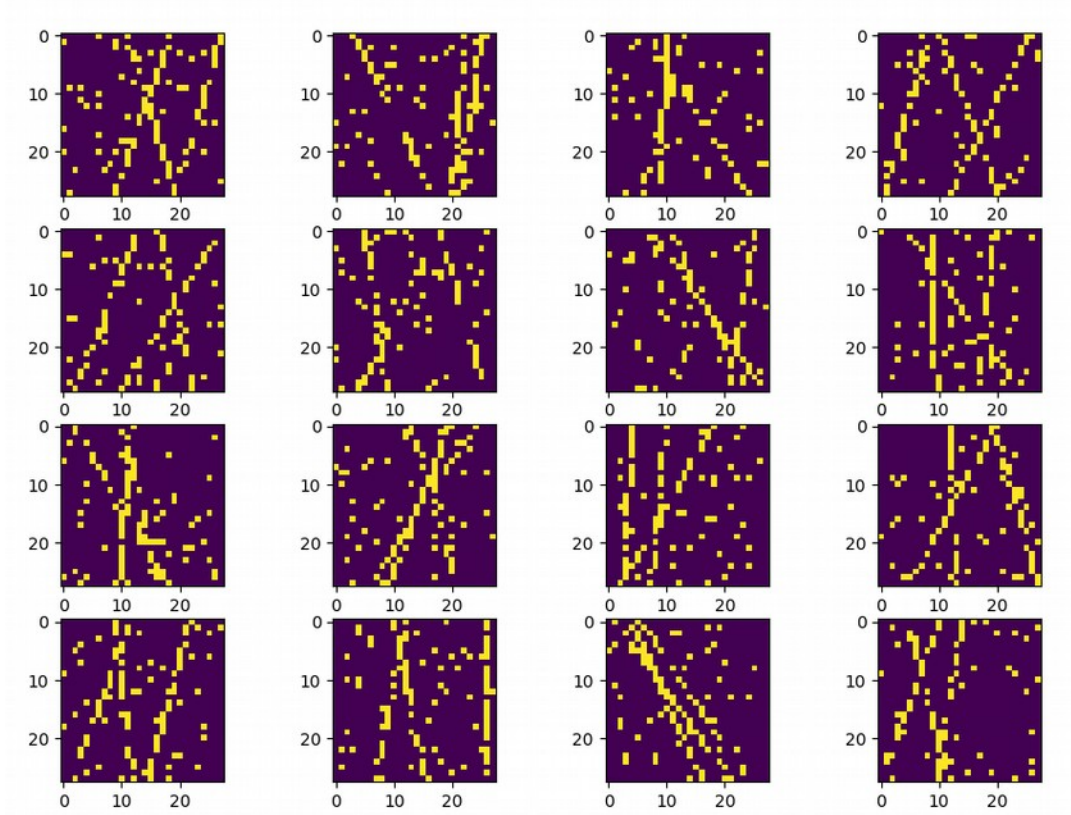
- construct pairs from all the combinations of hits ***(CPU TIME CONSUMING)***
- clusterization of two adjacent hits ***(COMPLICATED ALGORITHMS)***
  → reduces number of clones
  → improves reconstruction of two close tracks
- for each hit pair construct a line and collect all the hits within a certain window
  → for all combinations of hits a fit is performed (CPU TIME CONSUMING)
- **potential problems with final precision and tracking efficiency**





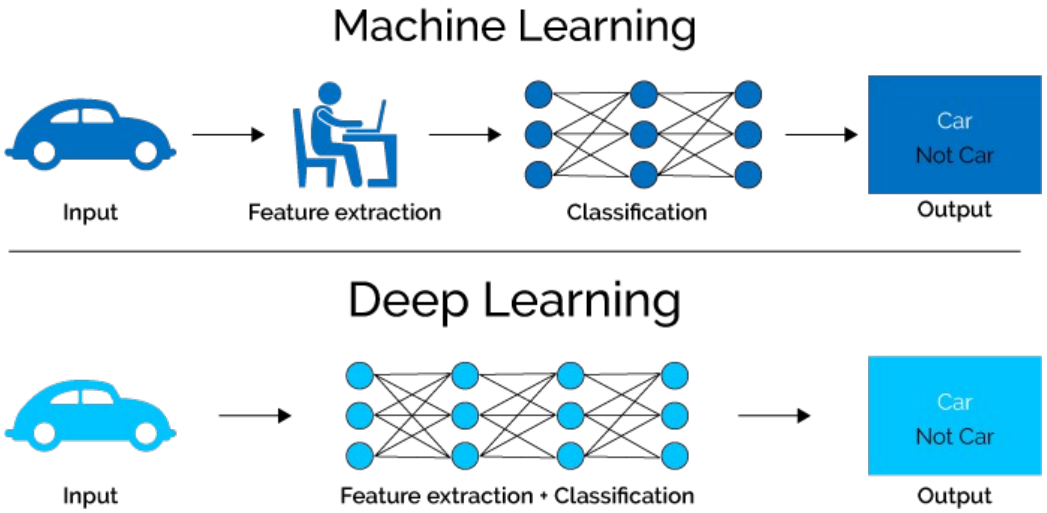
Using DNN → hope to improve all bottlenecks!

Toy model of track finding using Deep Neural Networks



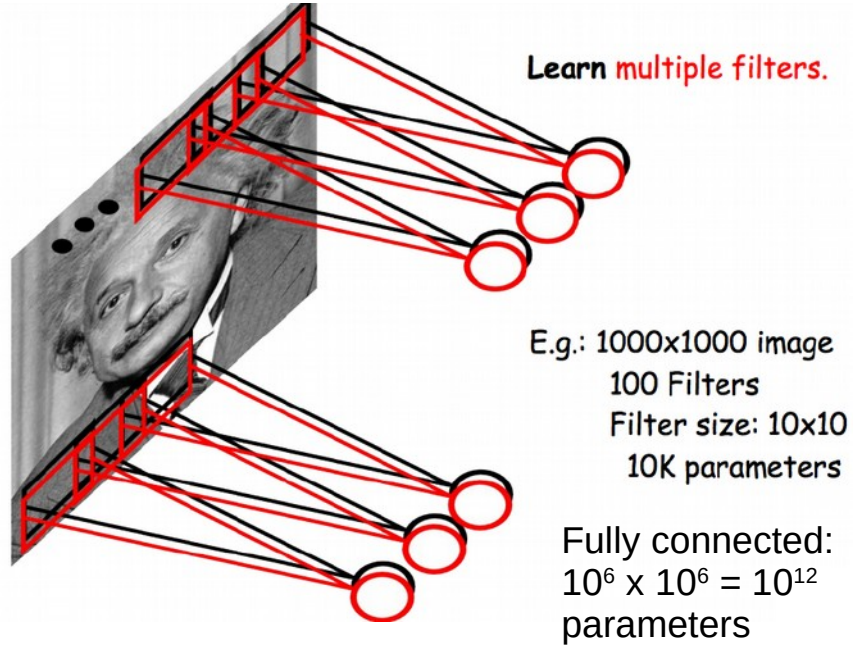
- Toy model for feasibility studies (small, fast training).
- Characteristics of the toy MC:
  - 2-dimensional data to reduce the training time
  - Straight line tracks (no magnetic field) on the 28x28 pixel plane
  - Finite hit efficiency and random noise
- Reconstruction method: training Convolutional Deep Neural Network to identify a track and return track parameters (one of the propositions of the HEPTrkx group)

Deep Learning



Deep Learning – thousands or millions of input variables (like pixels of a photo), the features are *automatically* extracted during training.

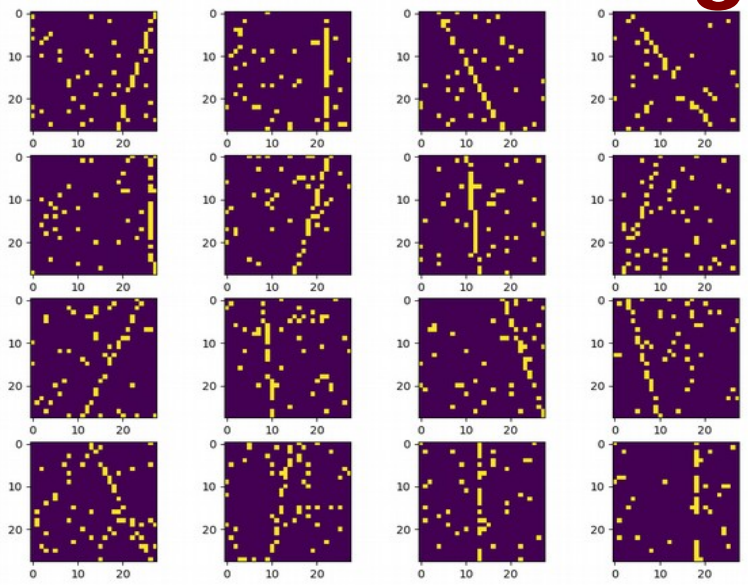
Convolutional Deep Neural Network (CNN)



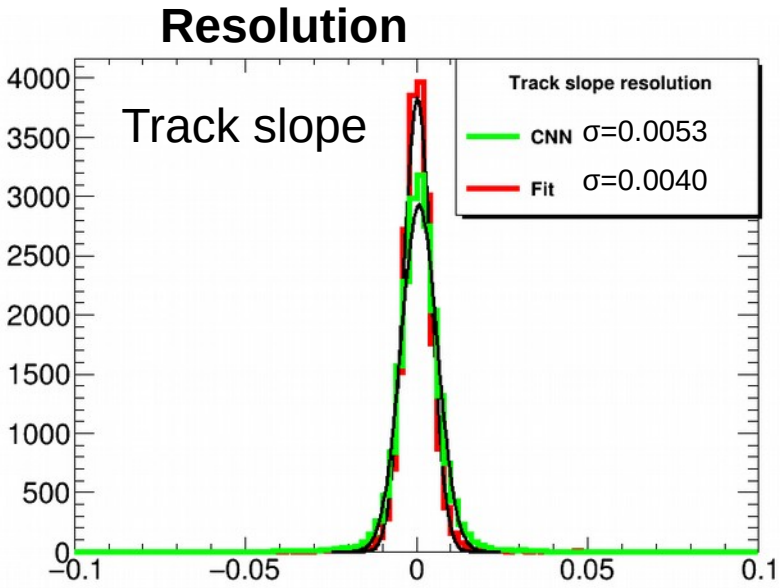
Finding the features:

- the same features might be found in different places
- feature filters might be small (like 10x10 pixels)
- we could train many filters, each recognizing another feature, and move them over the picture
- Huge reduction of the number of parameters!

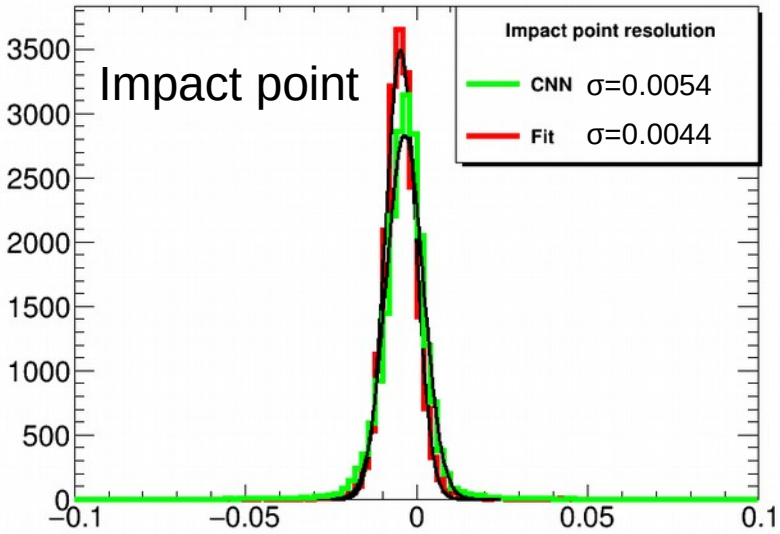
Single track events



70% hit efficiency, 5% noise. Nice, clean events.



- After patter recognition by CNN the track is fitted using the ROOT package robust fit (removing the outliers).
- Results: track parameter resolution only slightly worse for neural network than for the fit.
- **Efficiency: over 99%** *(reconstructed track found within 5 pixels from the true track)*.

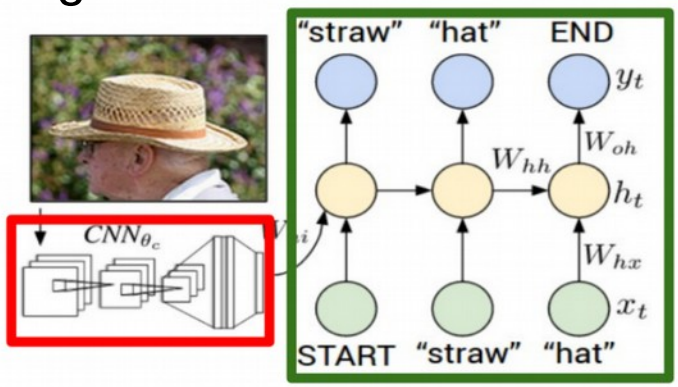


Software used:

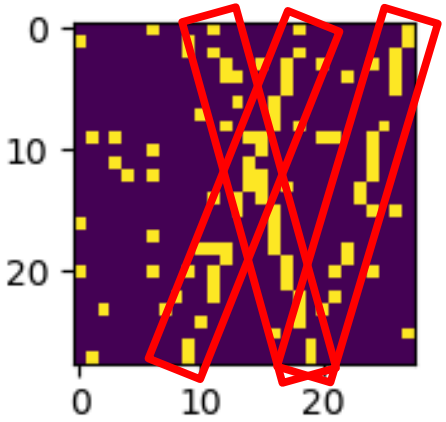
- *Tensorflow - https://www.tensorflow.org/*
- *Keras - https://keras.io/*
- *ROOT - https://root.cern.ch/*

What about many tracks?

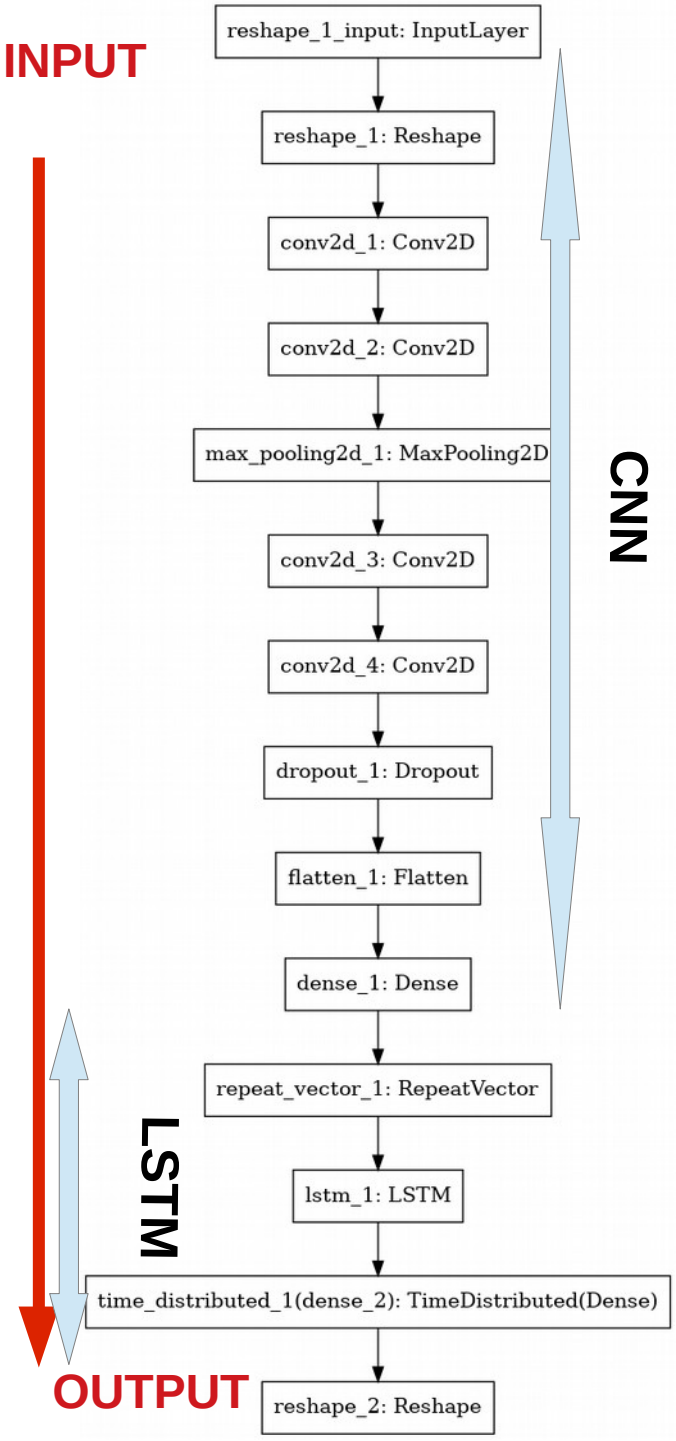
- Solution: add Long Short-Term Memory (LSTM) layer to process a sequence of tracks (not only a single one).
- LSTM allows labeling images, so why not to use it for labeling tracks? **Recurrent Neural Network**



Convolutional Neural Network



CNN extracts the features from our input image. The LSTM network is than trained as a language model on the feature vector.



Network architecture

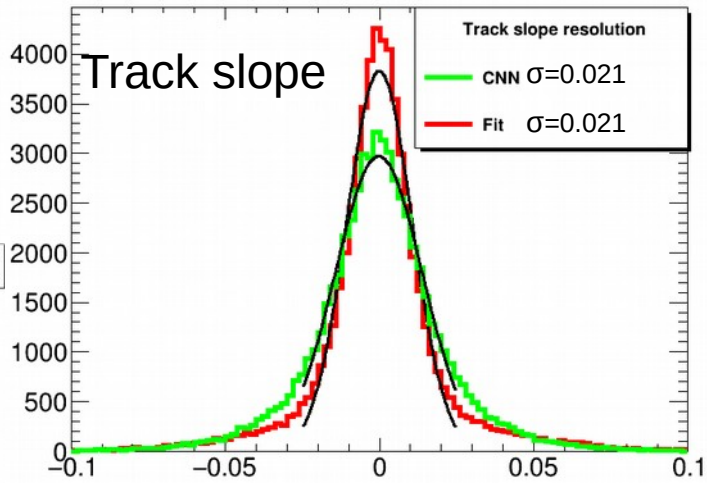
Tests with three tracks and higher noise

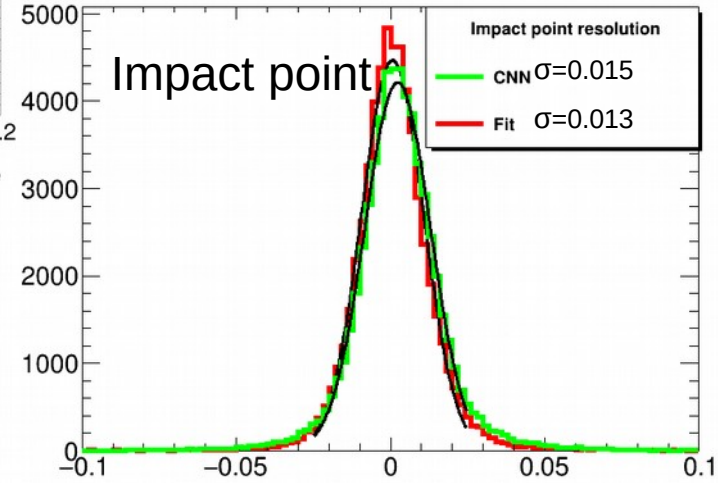
- **3 tracks**
- **70% hit efficiency**
- **5% random noise**

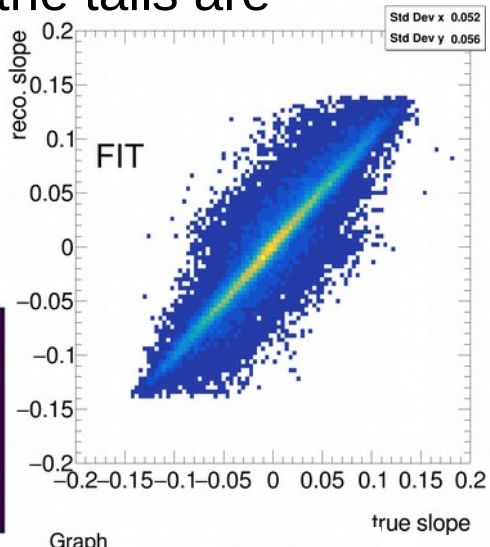
Slightly worse resolution achieved by CNN than the fit performed afterwards. Fit gives narrower resolution in the central part, but the tails are comparable to the CNN.

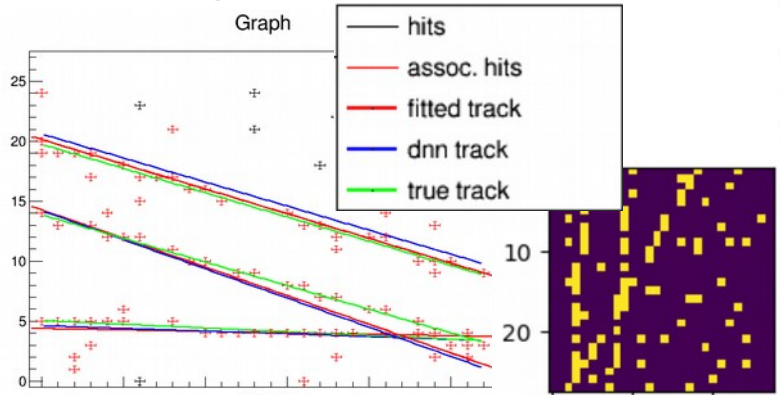
Efficiency: ~87%

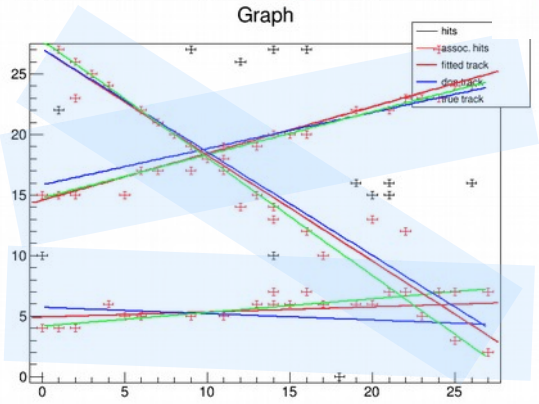
Resolution

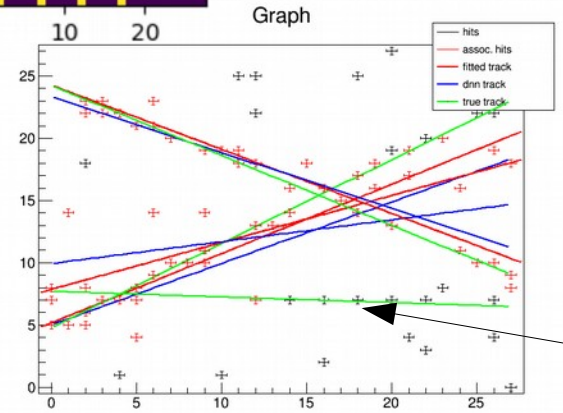












Failure

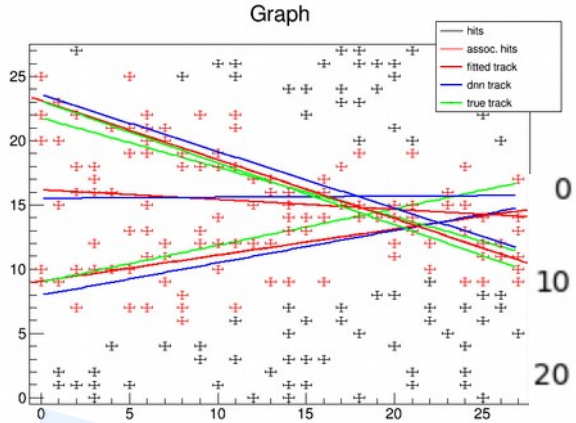
Tests with more tracks and higher noise

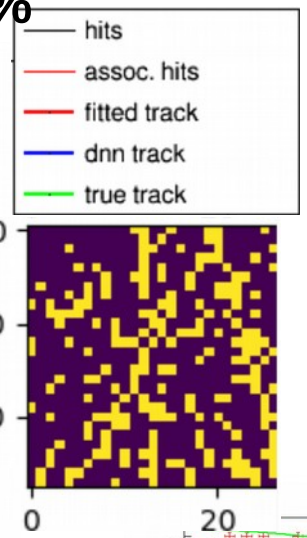
- **3 tracks**
- **70% hit efficiency**
- **20% random noise**

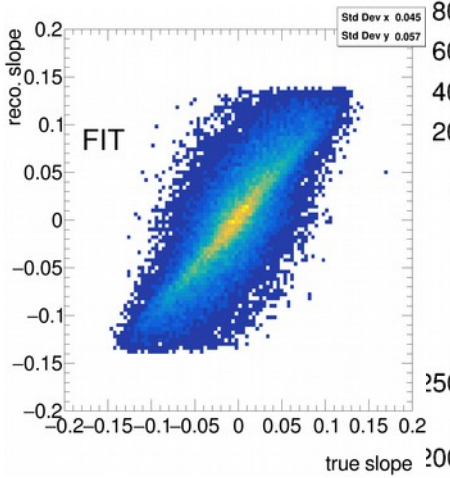
Tracks are hardly visible by eye, but still could be recognized by the network.

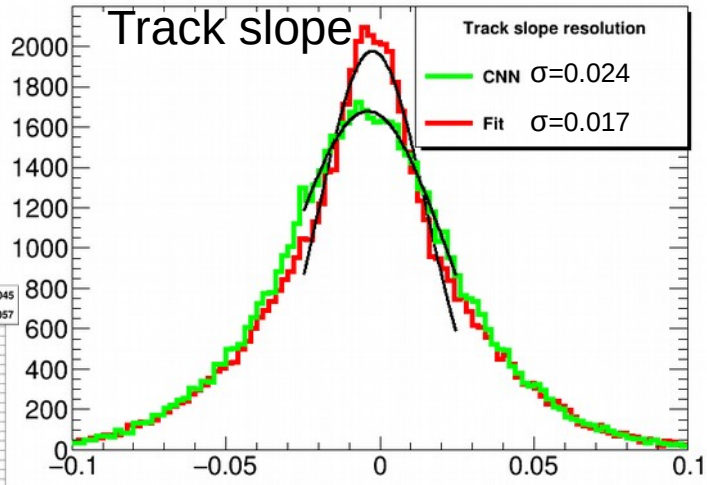
The resolution and efficiency degrade.

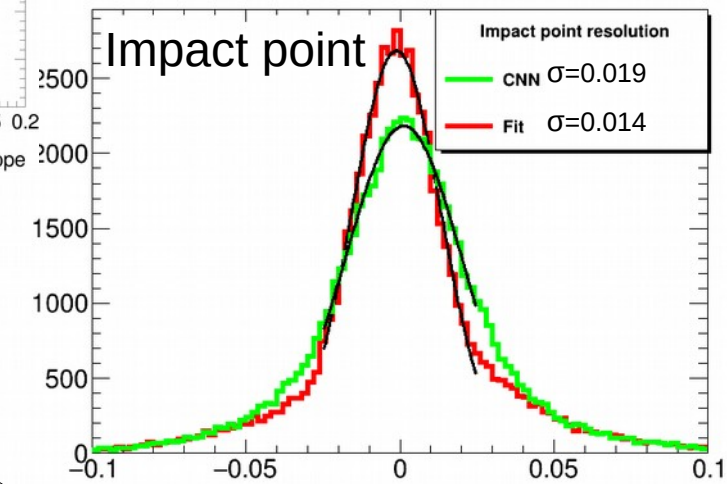
Efficiency CNN: ~66%

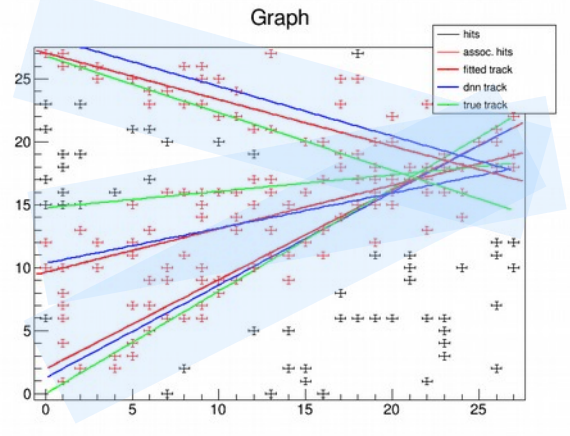


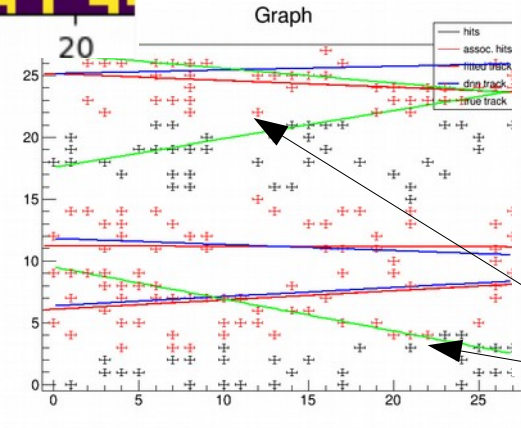












Failure

Results

- Robust and effective track finding in the toy model data
- After quite long initial training the classification itself is quite fast.

  CPU usage (Three tracks, 70% efficiency, 20% noise)**:**
  - Deep NN training: 457 a.u
  - Pattern recognition using NN: **13 a.u** **it is very fast!**
  - Track fitting: 385 a.u

  The pattern recognition is very fast! About 30 times faster than the fitting performed afterwards.
- Resolution obtained by Convolutional Neural Network is not much worse than the one from the robust fit.

Future plans

- We plan to apply the Deep Neural Network techniques to the experimental testbeam data taken in 2017 and 2018 by the MUonE experiment operating at the SPS accelerator at CERN.
- Deep Neural Network might be also used for pattern recognition in MuonE, which should improve it's precision and efficiency. Crucial, since anomalous magnetic muon moment should be measured with precision better than 0.05% to produce a useful result.
- Novel techniques, worth trying on real data (test beam first!).
- The final approach might be not so straight forward, might be a hybrid of neural network and traditional methods (ideas from HEP.TrkX group) :
  - Track seeding with CNN
  - Hit classification with LSTMs
- We will need much more computer power for network training (Prometeus, GPU?)

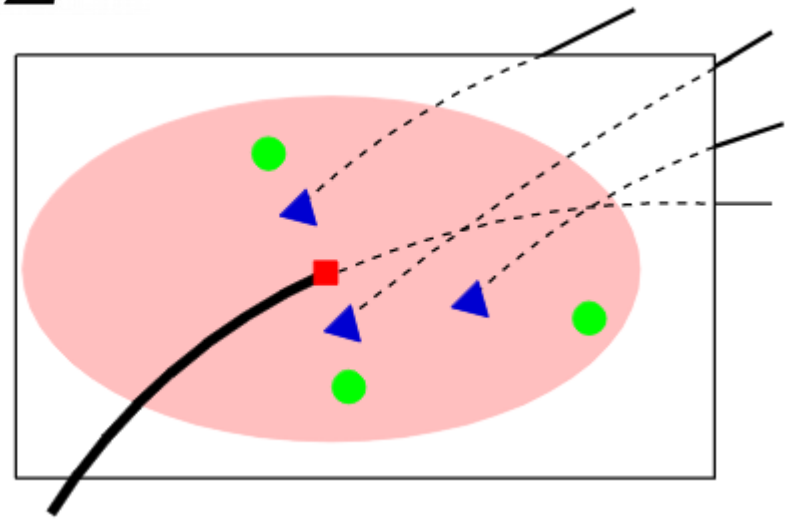
Calculations done on Zeus using LHCb grant no. lhcbflav08

Backup

Kalman filter

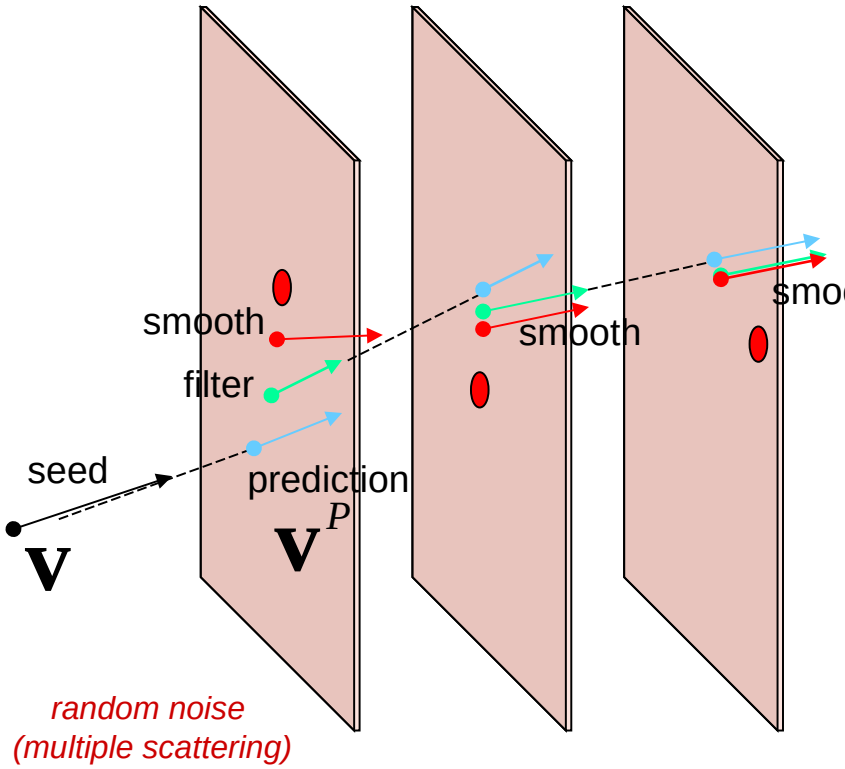
- Enables tracking the moving objects (rockets, planes etc).
- Commonly used for he combined tracking and fitting in HEP.
- **Combinatorial KalmanFilter:**
  - split a track into independent branches
  - propagate them to the next detector layer and split if more than one hit found
  - discard bad branches using track log-likelihood:
    - penalty for missing hits $L = -\sum \Delta\chi^2$

:



Fitter: Kalman Filter

- Kalman Filter:
  - Used for track fitting by most of HEP experiments
  - Easy to include random noise processes (ms) and systematic effects (eloss)
  - It is a local and incremental fit (dynamic states)
    We can do simultaneously fitting & patter recognition



IPropagator

$$\mathbf{v}^P = f(\mathbf{v}) \xrightarrow{linear} \mathbf{v}^P = \mathbf{F}\,\mathbf{v}$$

transport matrix

$$\mathbf{C}^P = \mathbf{F}\mathbf{C}\mathbf{F}^T + \mathbf{Q}$$

noise matrix (ms)

IProjector

$$\vec{\mathbf{w}} = \vec{h}(\mathbf{v}) \xrightarrow{linear} \mathbf{w} = \mathbf{H}\,\mathbf{v}$$

Projection matrix

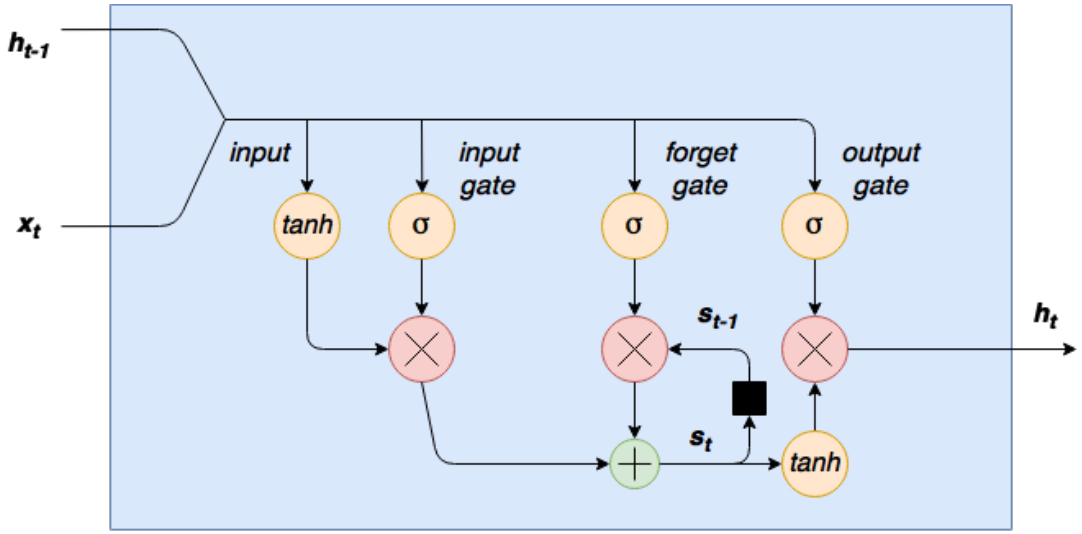
$$\mathbf{r} = \mathbf{w} - \mathbf{m}$$

residual = projection - measurement

Jose Angel Hernando, ACAT 03

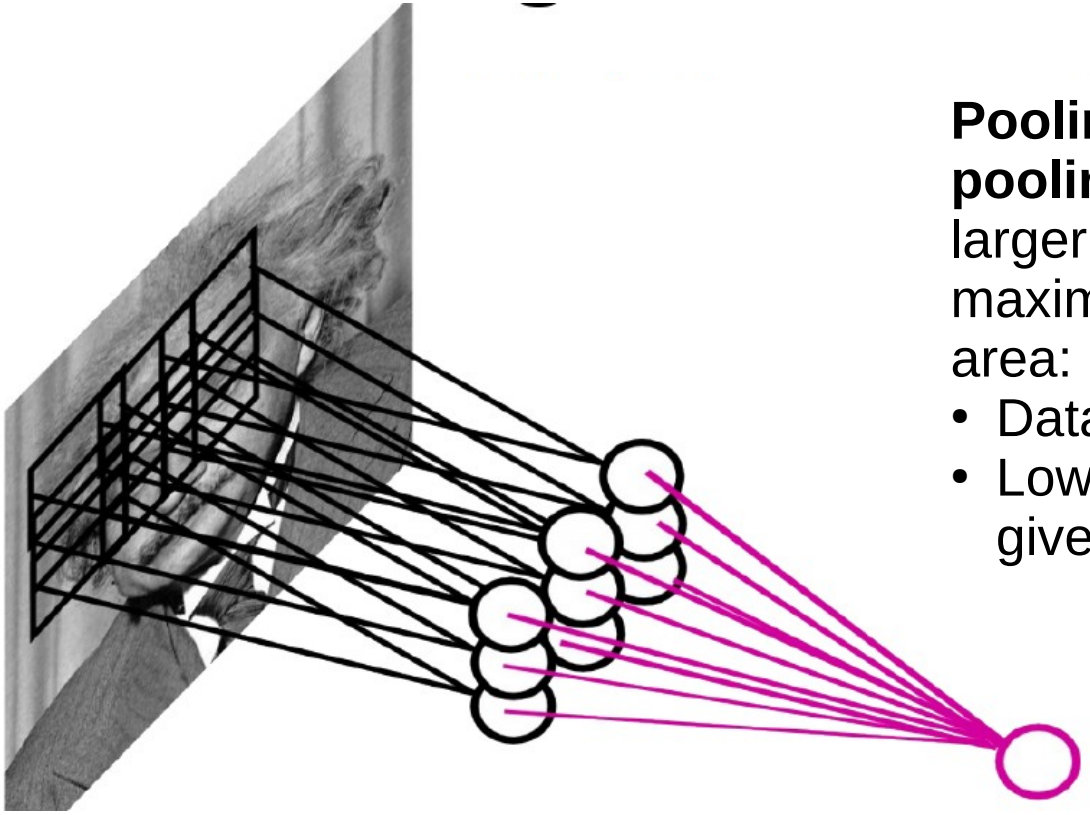
Long Short Term Memory (LSTM)

Adding the LSTM to the network is like adding a memory unit that can remember context from the very beginning of the input. Useful for object labeling on an image or Natural Language Processing (NLP).



- LSTM is an artificial recurrent neural network . Unlike standard feedforward neural networks, LSTM has feedback connections that make it a "general purpose computer" (that is, it can compute anything that a Turing machine can)!!!
  It can not only process single data points (such as images), but also entire sequences of data (such as speech or video).

Pooling



Pooling – (in most cases **max pooling**) the group of outputs for a larger input area is replaced by a maximum (or average) for this given area:

- Data reduction,
- Lower sensitivity for the position of a given feature.