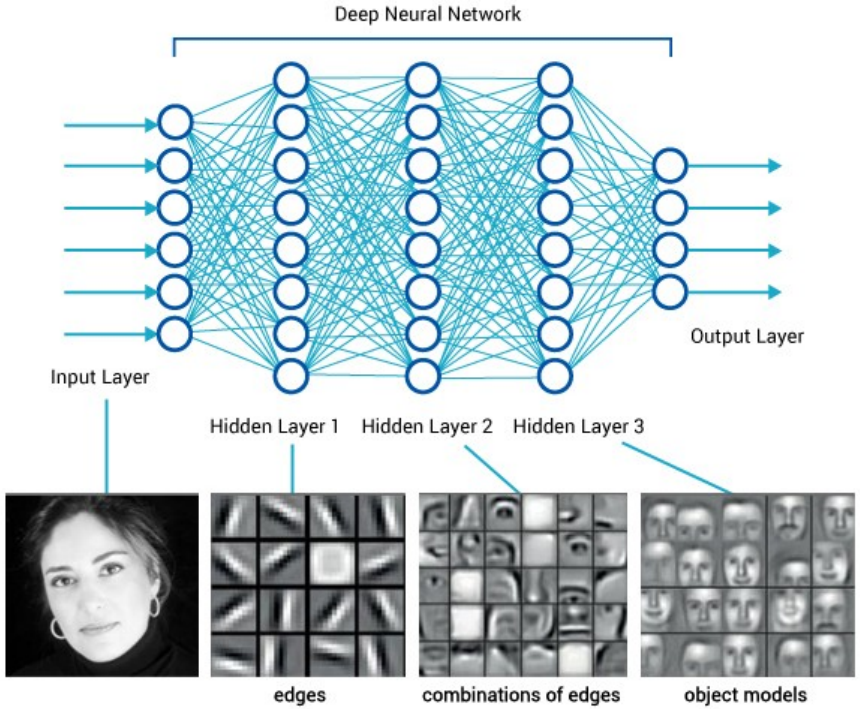
Machine learning

Lecture 5



Marcin Wolter

IFJ PAN

5 March 2019

- Application to the Higgs searches
- Deep learning
- Convolution network
- Track reconstruction using DNN

Successes in deep learning - NN with multiple hidden layers

- **Success due to the combination of factors:**
  1) speeding up the stochastic gradient descent algorithm with graphics processors GPU,
  2) using much larger training sets,
  3) using new learning algorithms, including randomized algorithms such as dropout[1,2] (dropping out random neurons while training)
  4) pre-training the initial layers of the network with unsupervised learning methods such as autoencoders[3,4].
- The approach 4) attempts to learn a **useful layered representation** of the data **without having to backpropagate** through a DNN; standard gradient descent is only used at the end to fine-tune the network.
  With these methods, it became common to train **DNNs of five or more layers.**

[1] Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. Preprint at http://arxiv.org/abs/1207.0580 (2012).
[2] Baldi, P. & Sadowski, P. The dropout learning algorithm. Artif. Intell. 210, 78–122 (2014).
[3] Hinton, G. E., Osindero, S. & Teh, Y.-W. A fast learning algorithm for deep belief nets. Neural Comput. 18, 1527–1554 (2006).
[4] Bengio, Y. et al. in: Advances in Neural Information Processing Systems 19 (MIT Press, 2007).

A very nice review

Deep learning, *Yann LeCun, Yoshua Bengio, Geoffrey Hinton, doi:10.1038/nature14539*

http://pages.cs.wisc.edu/~dyer/cs540/handouts/deep-learning-nature2015.pdf

REVIEW

doi:10.1038/nature14539

Deep learning

Yann LeCun[1,2], Yoshua Bengio[3] & Geoffrey Hinton[4,5]

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning.

Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input.

Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification. Deep-learning methods are representation-learning methods with multiple levels of representa-

intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition[1–4] and speech recognition[5–7], it has beaten other machine-learning techniques at predicting the activity of potential drug molecules[8], analysing particle accelerator data[9,10], reconstructing brain circuits[11], and predicting the effects of mutations in non-coding DNA on gene expression and disease[12,13]. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding[14], particularly topic classification, sentiment analysis, question answering[15] and language translation[16,17].

We think that deep learning will have many more successes in the near future because it requires very little engineering by hand, so it can easily take advantage of increases in the amount of available computation and data. New learning algorithms and architectures that are currently being developed for deep neural networks will only accelerate this progress.

Supervised learning

The most common form of machine learning, deep or not, is supervised learning. Imagine that we want to build a system that can classify

nature COMMUNICATIONS

ARTICLE

Received 19 Feb 2014 | Accepted 4 Jun 2014 | Published 2 Jul 2014

DOI: 10.1038/ncomms5308

Searching for exotic particles in high-energy physics with deep learning

P. Baldi[1], P. Sadowski[1] & D. Whiteson[2]

Collisions at high-energy particle colliders are a traditionally fruitful source of exotic particle discoveries. Finding these rare particles requires solving difficult signal-versus-background classification problems, hence machine-learning approaches are often used. Standard approaches have relied on 'shallow' machine-learning models that have a limited capacity to learn complex nonlinear functions of the inputs, and rely on a painstaking search through manually constructed nonlinear features. Progress on this problem has slowed, as a variety of techniques have shown equivalent performance. Recent advances in the field of deep learning make it possible to learn more complex functions and better discriminate between signal and background classes. Here, using benchmark data sets, we show that deep-learning methods need no manually constructed inputs and yet improve the classification metric by as much as 8% over the best current approaches. This demonstrates that deep-learning approaches can improve the power of collider searches for exotic particles.

Exotic Higgs decays

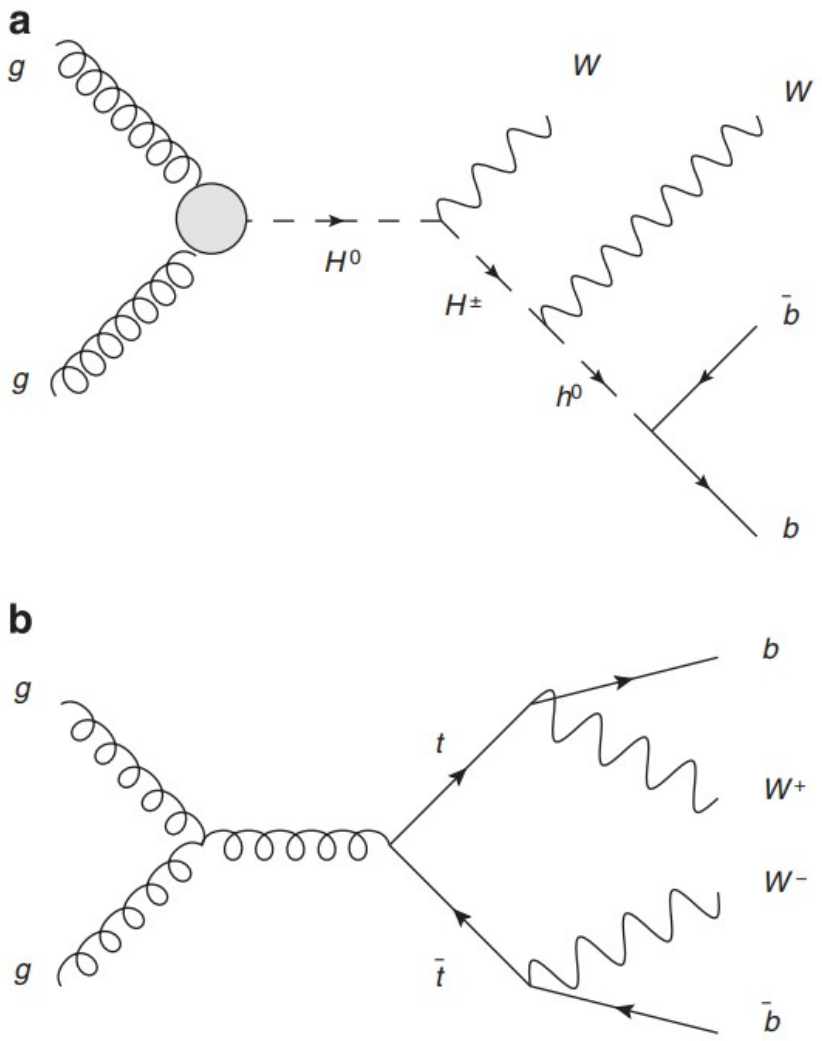


Figure 1 | Diagrams for Higgs benchmark. (**a**) Diagram describing the signal process involving new exotic Higgs bosons H^0 and $H^\pm$. (**b**) Diagram describing the background process involving top quarks (t). In both cases, the resulting particles are two W bosons and two b-quarks.

Exotic Higgs boson searches (simulated CMS data)

- Low level variables – 22 variables (here just few plotted)

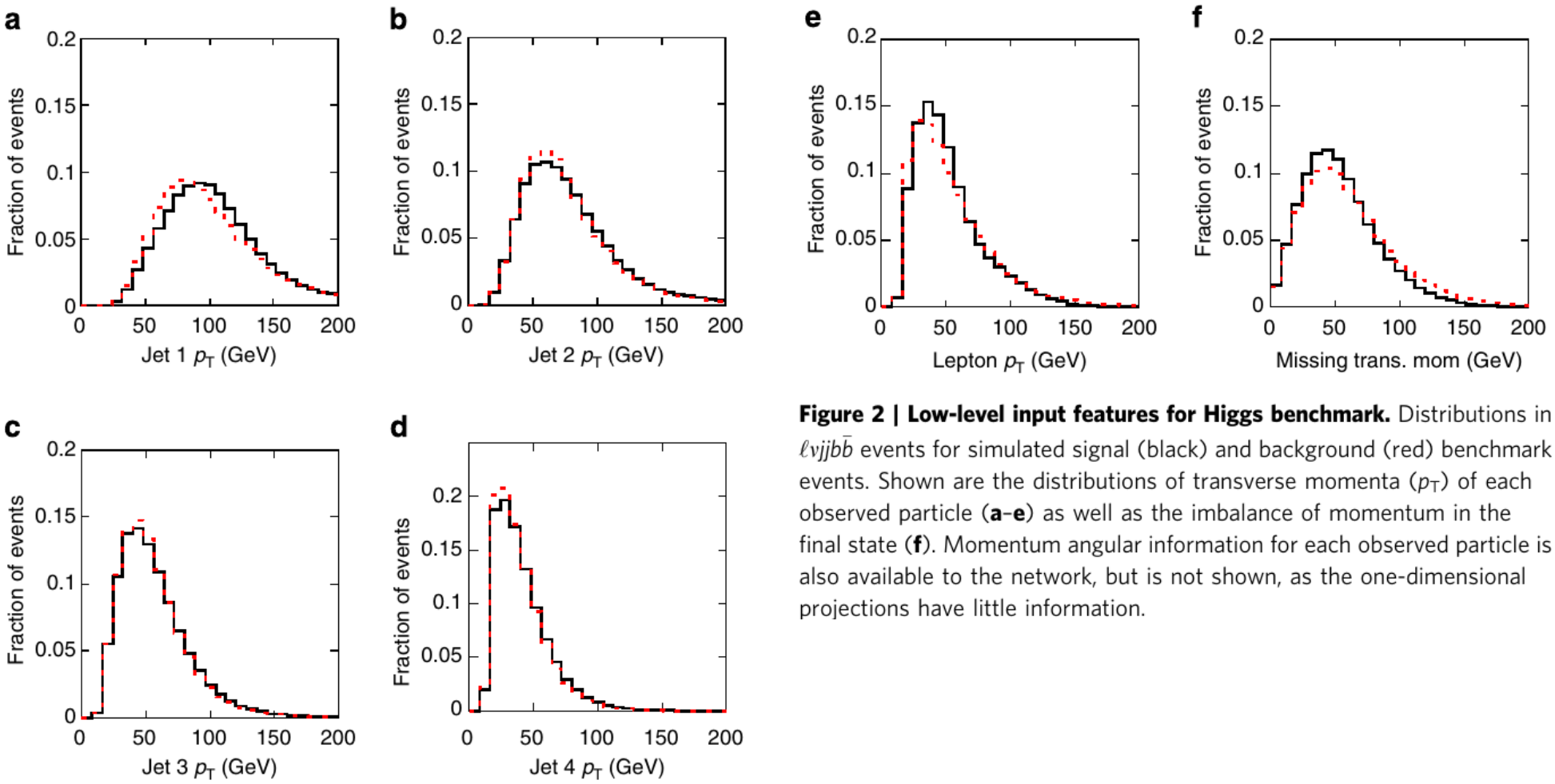


Figure 2 | Low-level input features for Higgs benchmark. Distributions in $\ell\nu jjb\bar{b}$ events for simulated signal (black) and background (red) benchmark events. Shown are the distributions of transverse momenta (p_T) of each observed particle (**a**–**e**) as well as the imbalance of momentum in the final state (**f**). Momentum angular information for each observed particle is also available to the network, but is not shown, as the one-dimensional projections have little information.

Exotic Higgs searches

- Physicists use the well discriminating high-level variables – they are built out of low-level variables and do not contain any additional information (7 variables).

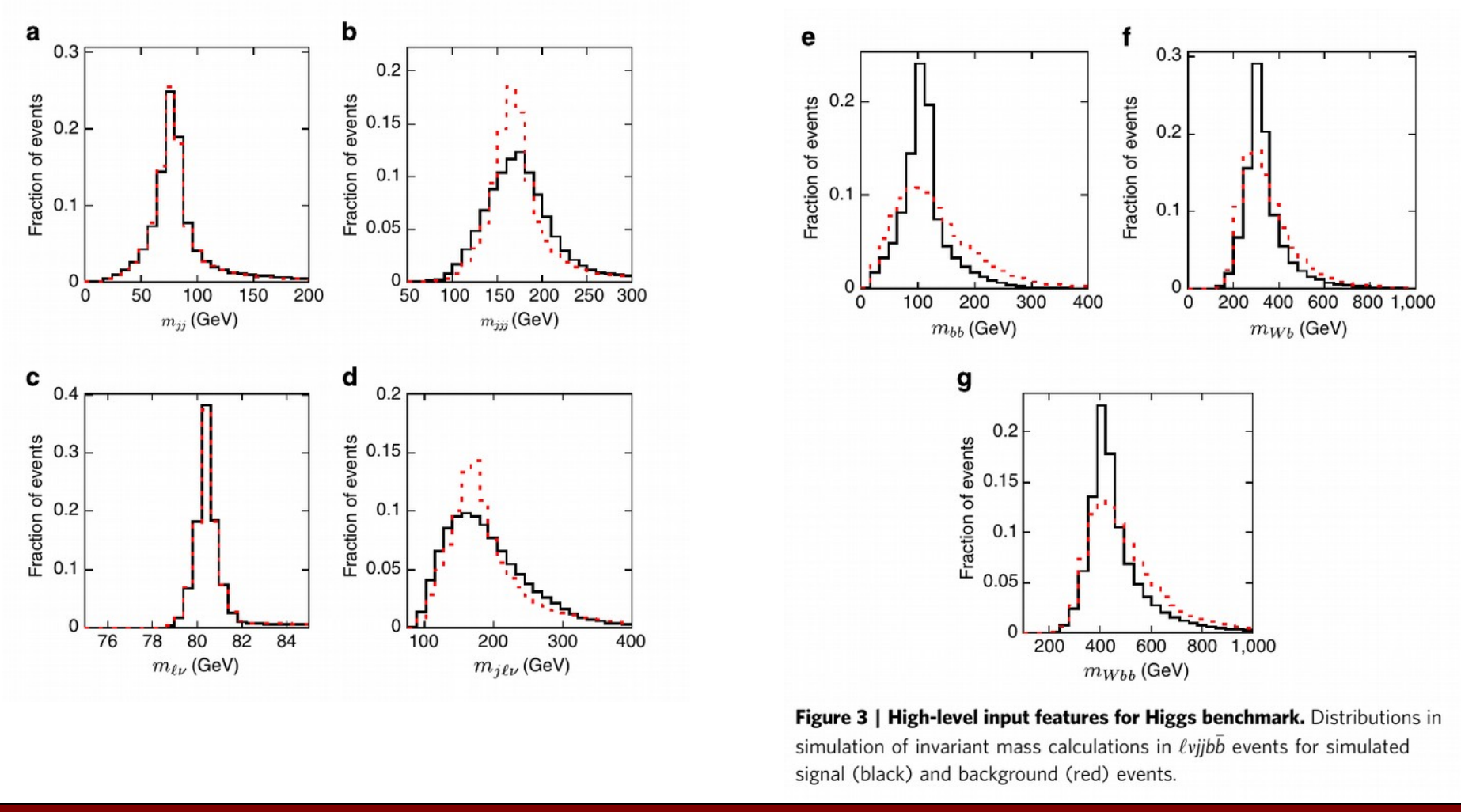


Figure 3 | High-level input features for Higgs benchmark. Distributions in simulation of invariant mass calculations in $\ell\nu jjb\bar{b}$ events for simulated signal (black) and background (red) events.

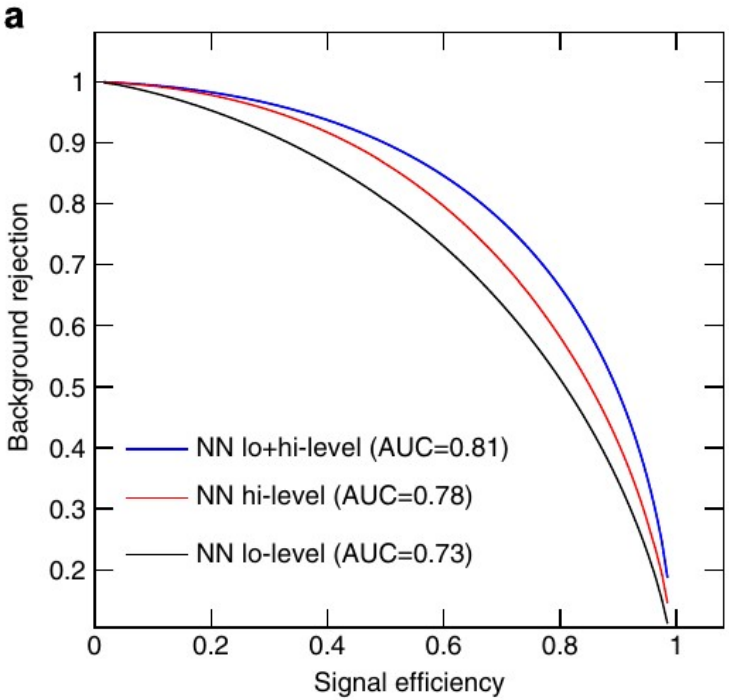
Higgs searches

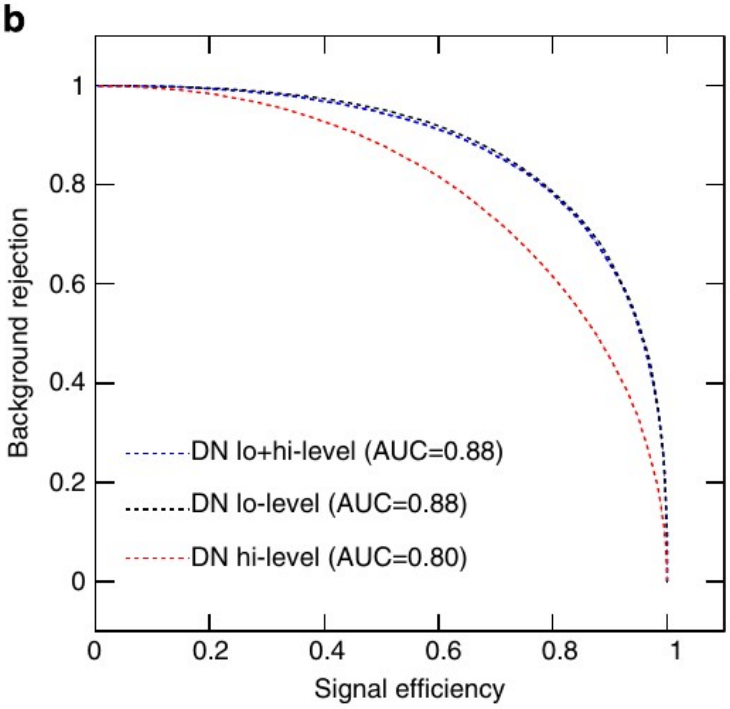
- Data: 2 600 000 events for training, 100 000 for validation
- Deep NN: 5 layers, 300 nodes in each layer, fully connected

Table 1 | Performance for Higgs benchmark.

Technique	Low-level	High-level	Complete
AUC			
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	0.880 (0.001)	0.800 (<0.001)	0.885 (0.002)
Discovery significance			
NN	2.5σ	3.1σ	3.7σ
DN	4.9σ	3.6σ	5.0σ

Comparison of the performance of several learning techniques: boosted decision trees (BDT), shallow neural networks (NN), and deep neural networks (DN) for three sets of input features: low-level features, high-level features and the complete set of features. Each neural network was trained five times with different random initializations. The table displays the mean area under the curve (AUC) of the signal-rejection curve in Fig. 7, with s.d. in parentheses as well as the expected significance of a discovery (in units of Gaussian σ) for 100 signal events and 1,000 ± 50 background events.





Summary of this exercise

- Deep NN found the features allowing to recognize the Higgs boson better, than the skilled physicists with their whole knowledge.
- This might allow in the future to automatize the physics analysis….
- Does it mean unemployment for us?
- Problem: We must trust the Monte Carlo...

The Automated HEP Physicist?

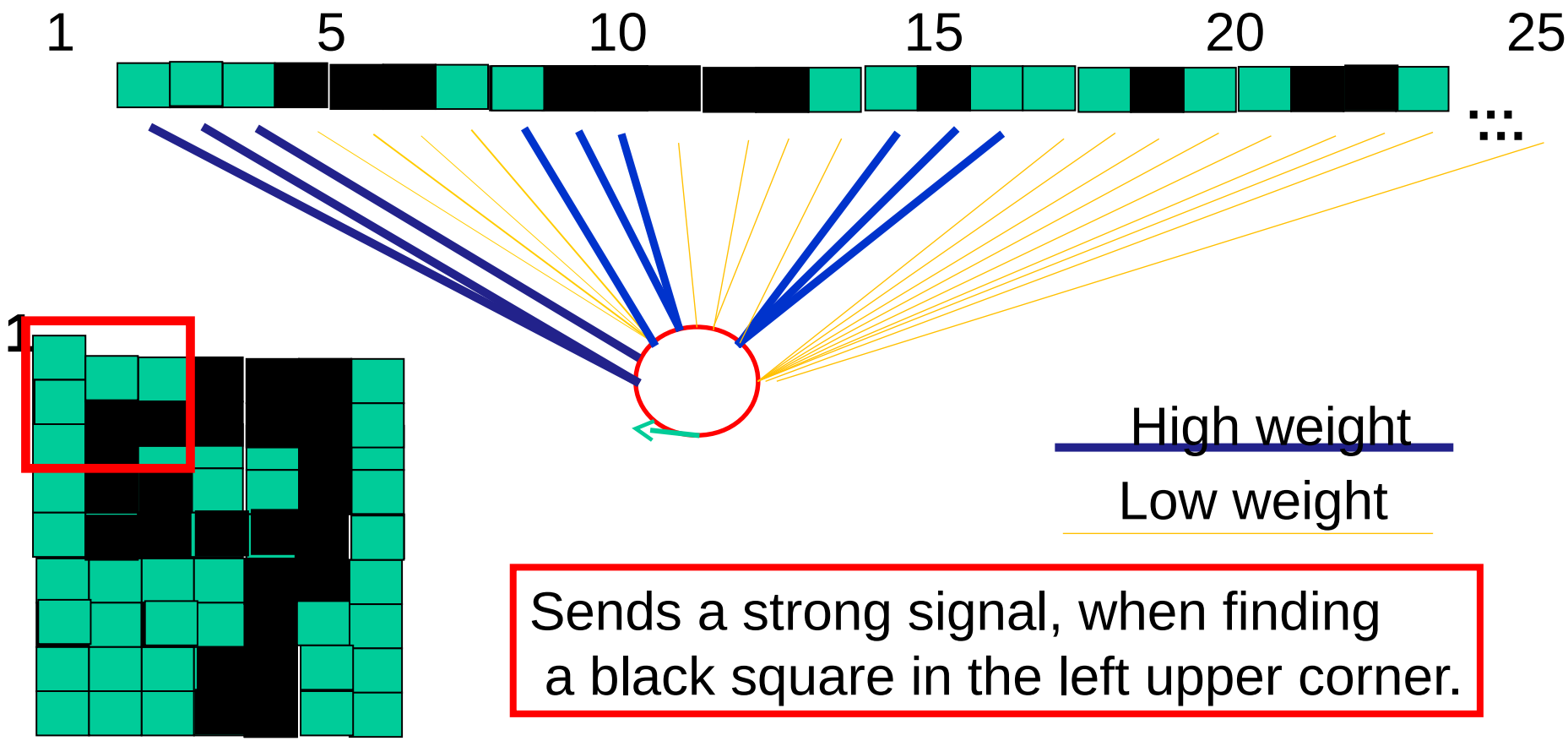
A few years from now our automaton could do on our behalf:

- Automatically determine the set of characteristics that distinguish particles from the primary vertex from those from other vertices and automatically classify particles based on this information.
- Automatically reduce particle event data into a smaller fixed set of numbers, say N ∼ 500 – which may be thought of as “pixelized images” – that can be the basis of further analysis.
- Automatically classify these “images” into two sets: those that look like simulated events and those that don’t.
- Find more sets and classify the events according to MC classes.

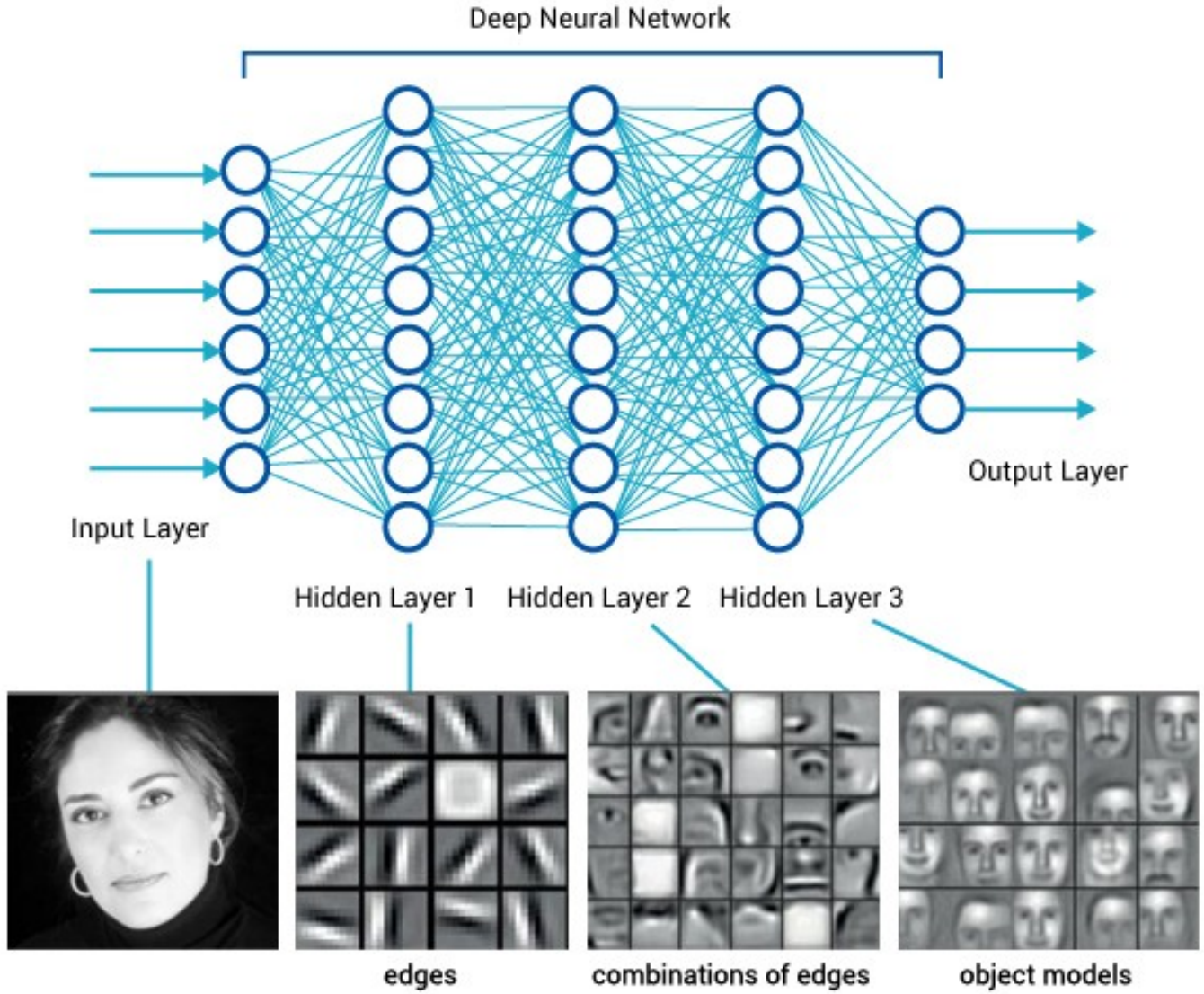
Conclusions inspired by H.B. Prosper “Deep Learning and Bayesian Methods”, https://indico.cern.ch/e/conf12

Deep learning for pattern recognition (a reminder)

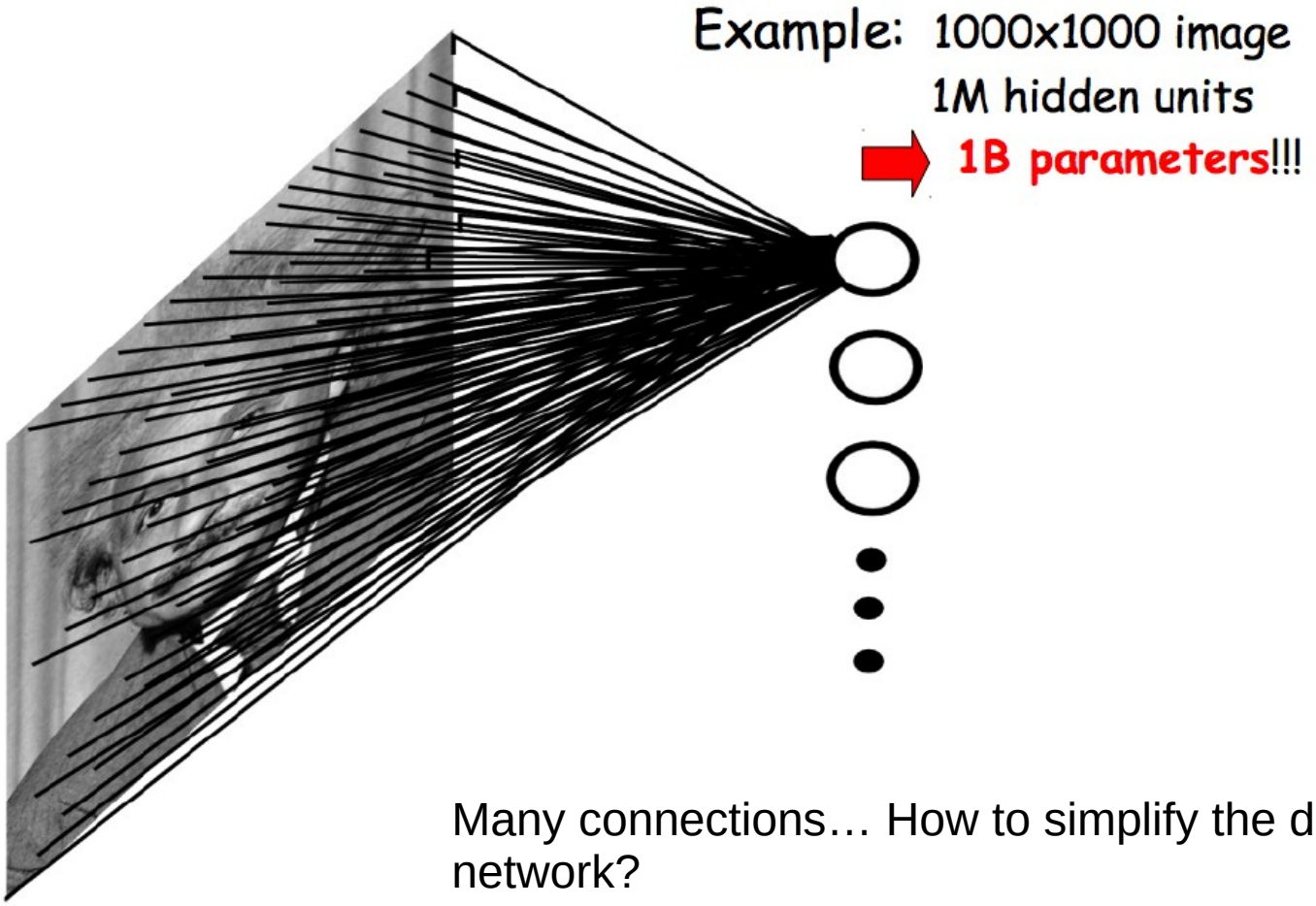
Individual layers are trained to recognize the “features” - from simple to more complex.



Deep Neural Network works like that...

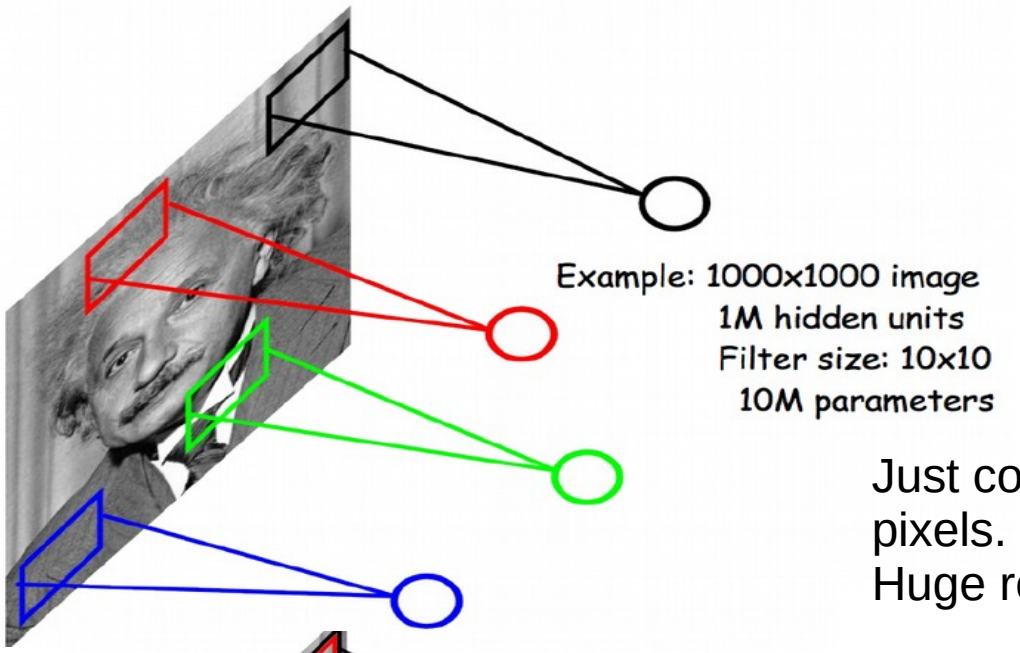


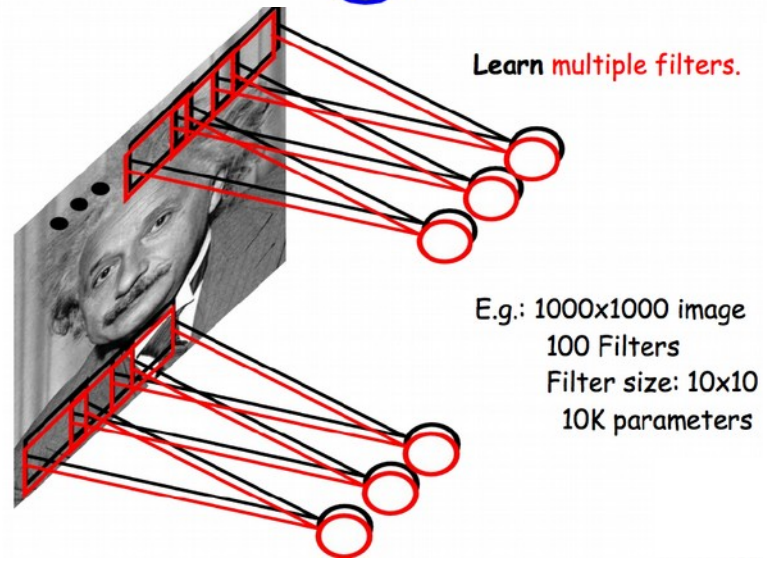
Convolutional NN
Pattern recognition



Many connections… How to simplify the deep neural network?

Convolutional NN



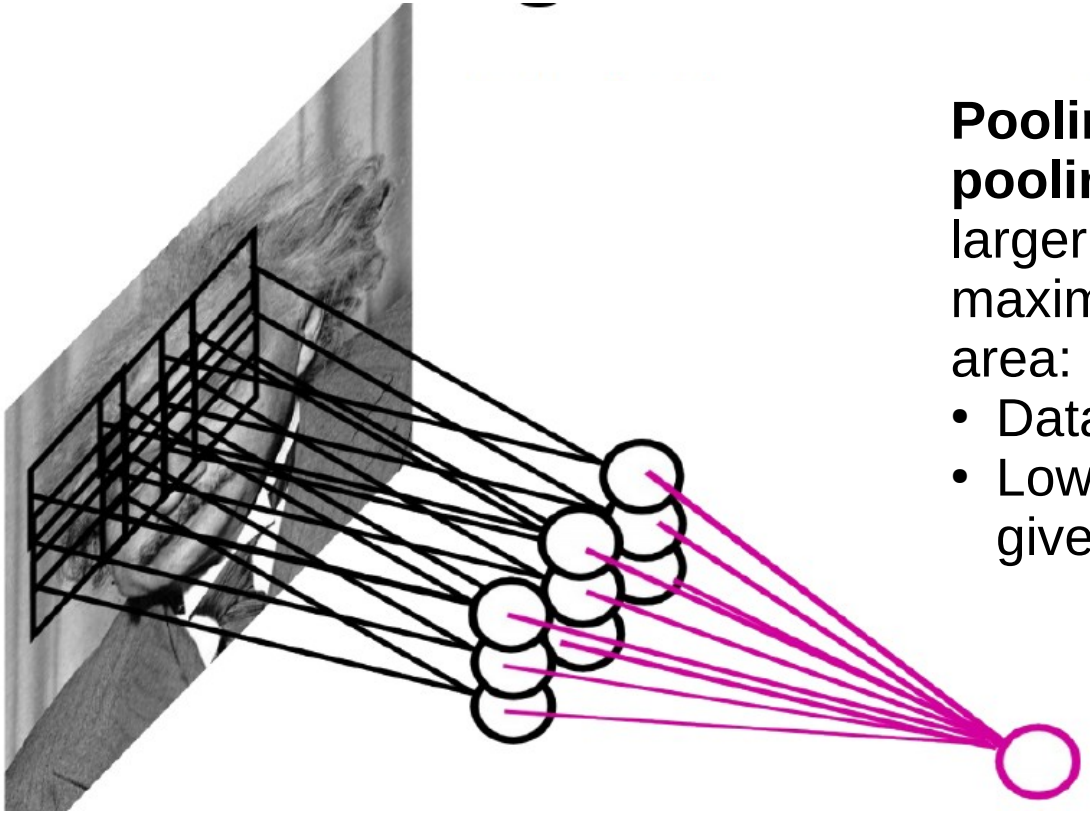


Just connect only local areas, for example 10x10 pixels.
Huge reduction of the number of parameters!

The same features might be found in different places => so we could train many filters, each recognizing another feature, and move them over the picture.

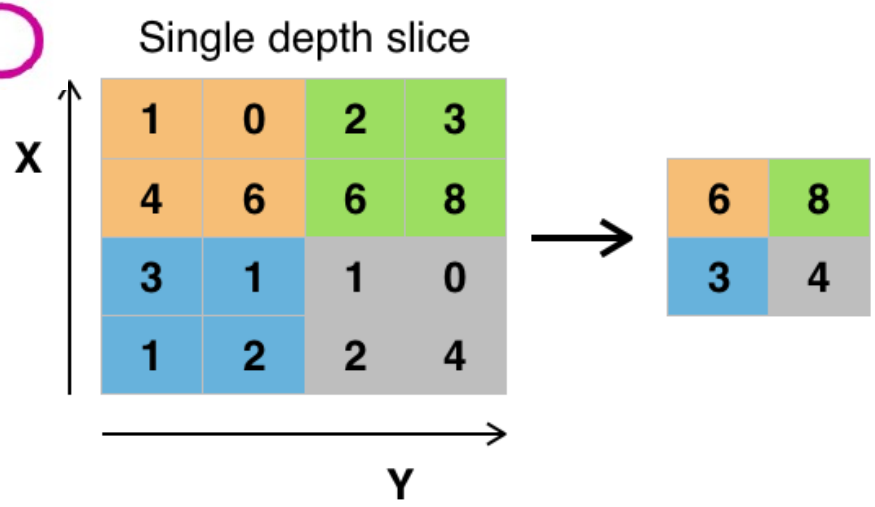
LeCun et al. "Gradient-based learning applied to document recognition" IEEE 1998

Pooling



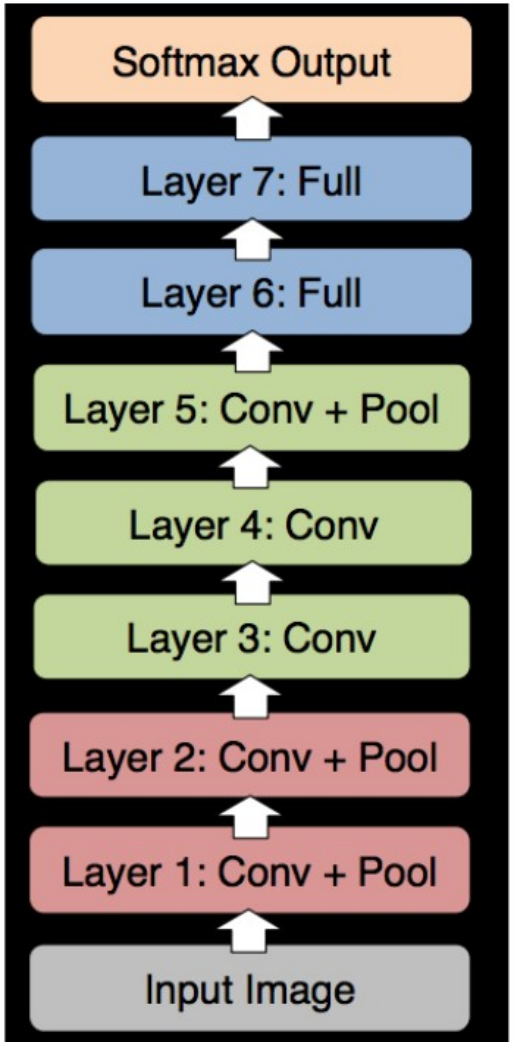
Pooling – (in most cases **max pooling**) the group of outputs for a larger input area is replaced by a maximum (or average) for this given area:

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



Architecture of Alex Krizhevsky et al.

- 8 layers total.
- Trained on Imagenet Dataset (1000 categories, 1.2M training images, 150k test images)
- 18.2% top-5 error
  - Winner of the ILSVRC-2012 challenge.

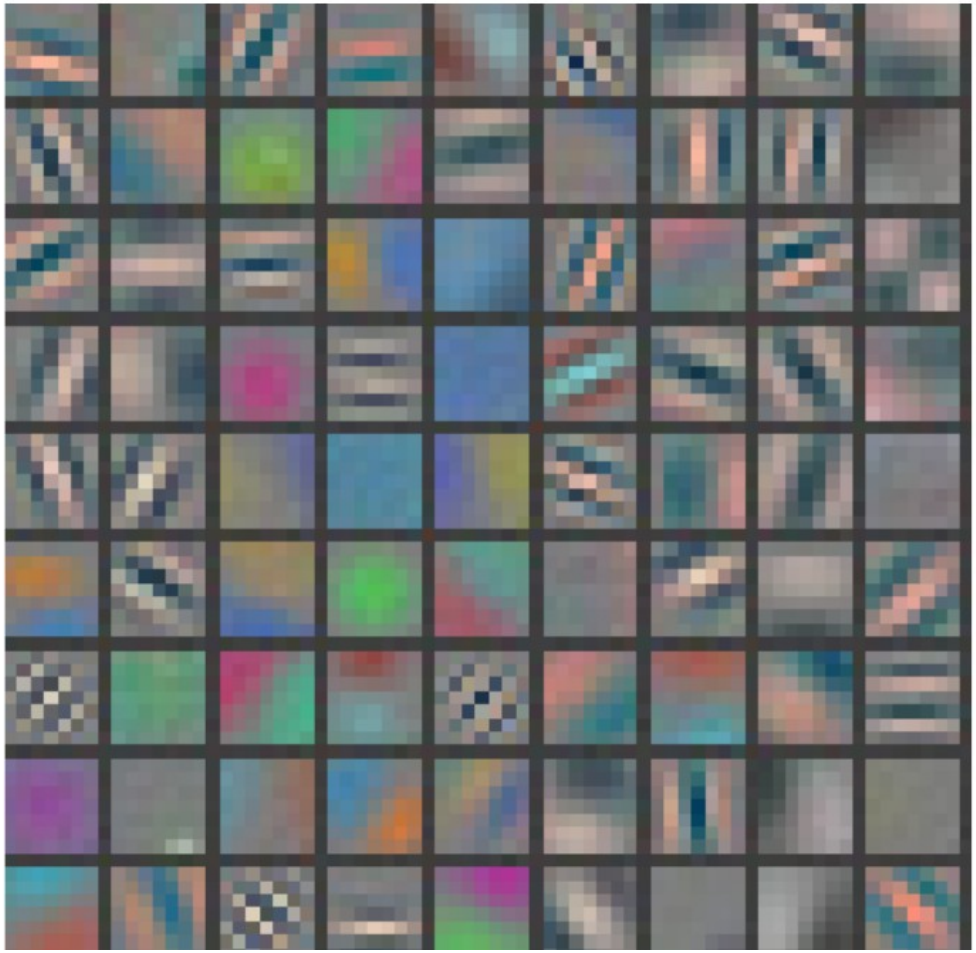


Slide: R. Fergus

First layer filters

Showing 81 filters of 11x11x3.

Capture low-level features like oriented edges, blobs.

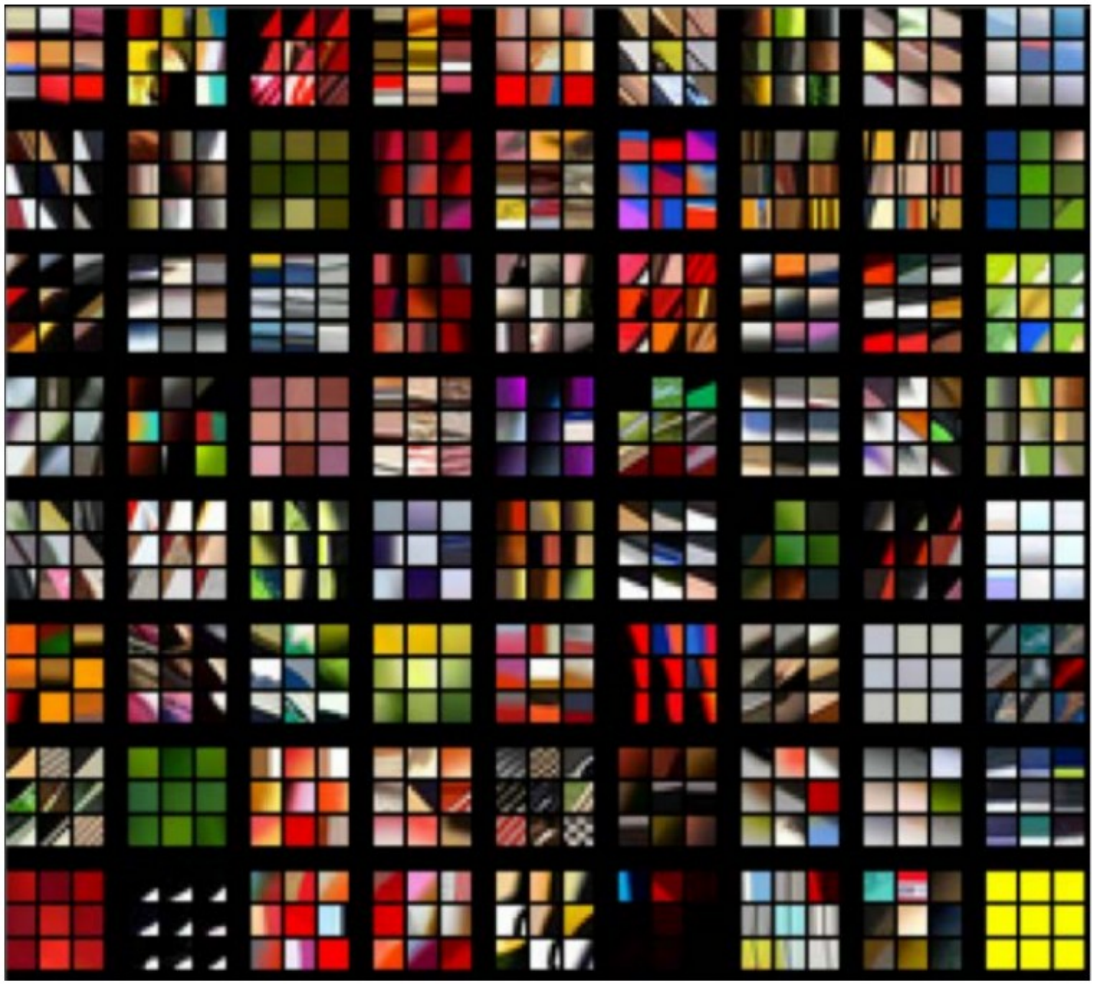


Note these oriented edges are analogous to what SIFT uses to compute the gradients.

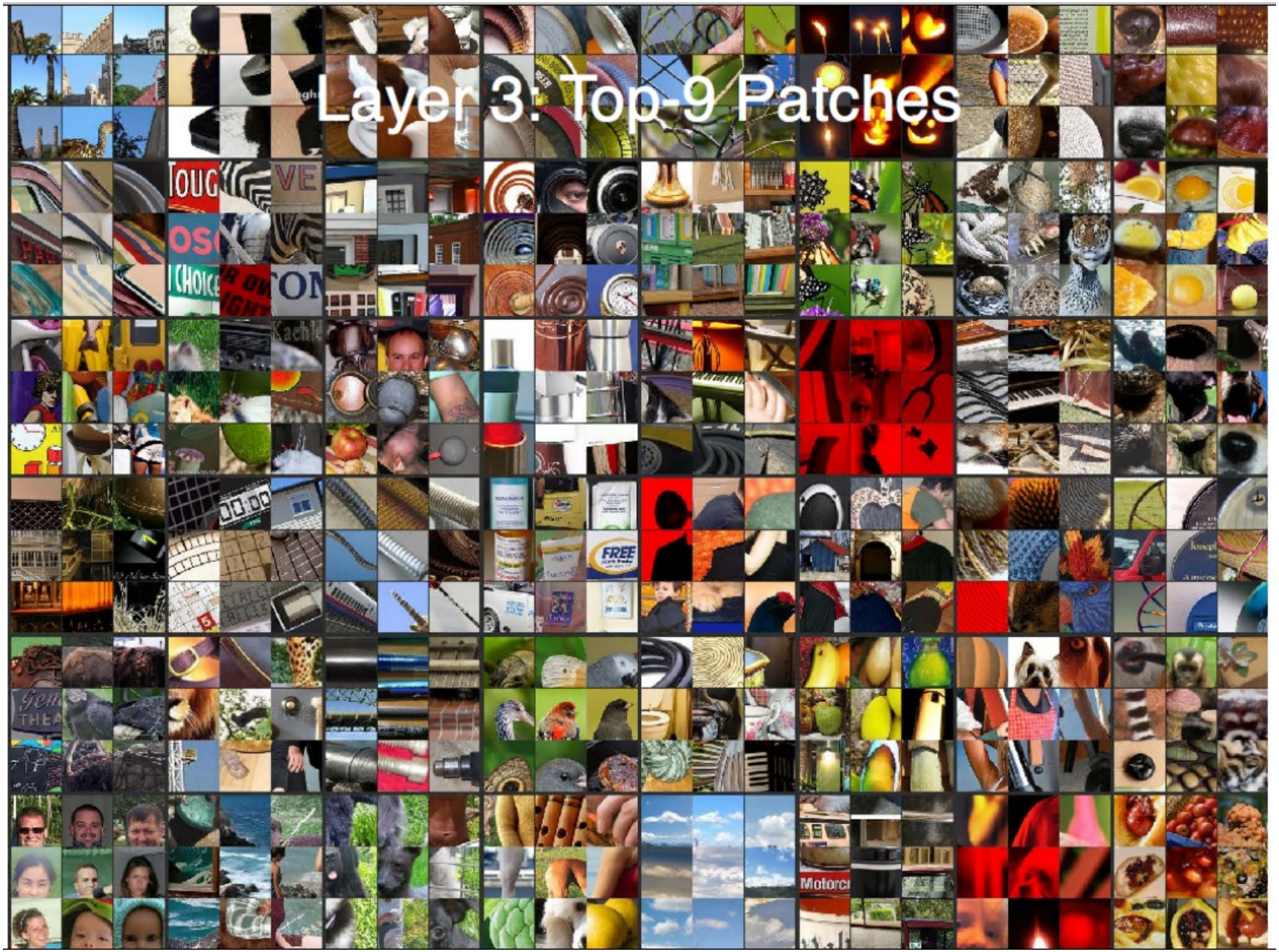
SIFT - scale-invariant feature transform, algorithm published in 1999 roku by David Lowe.

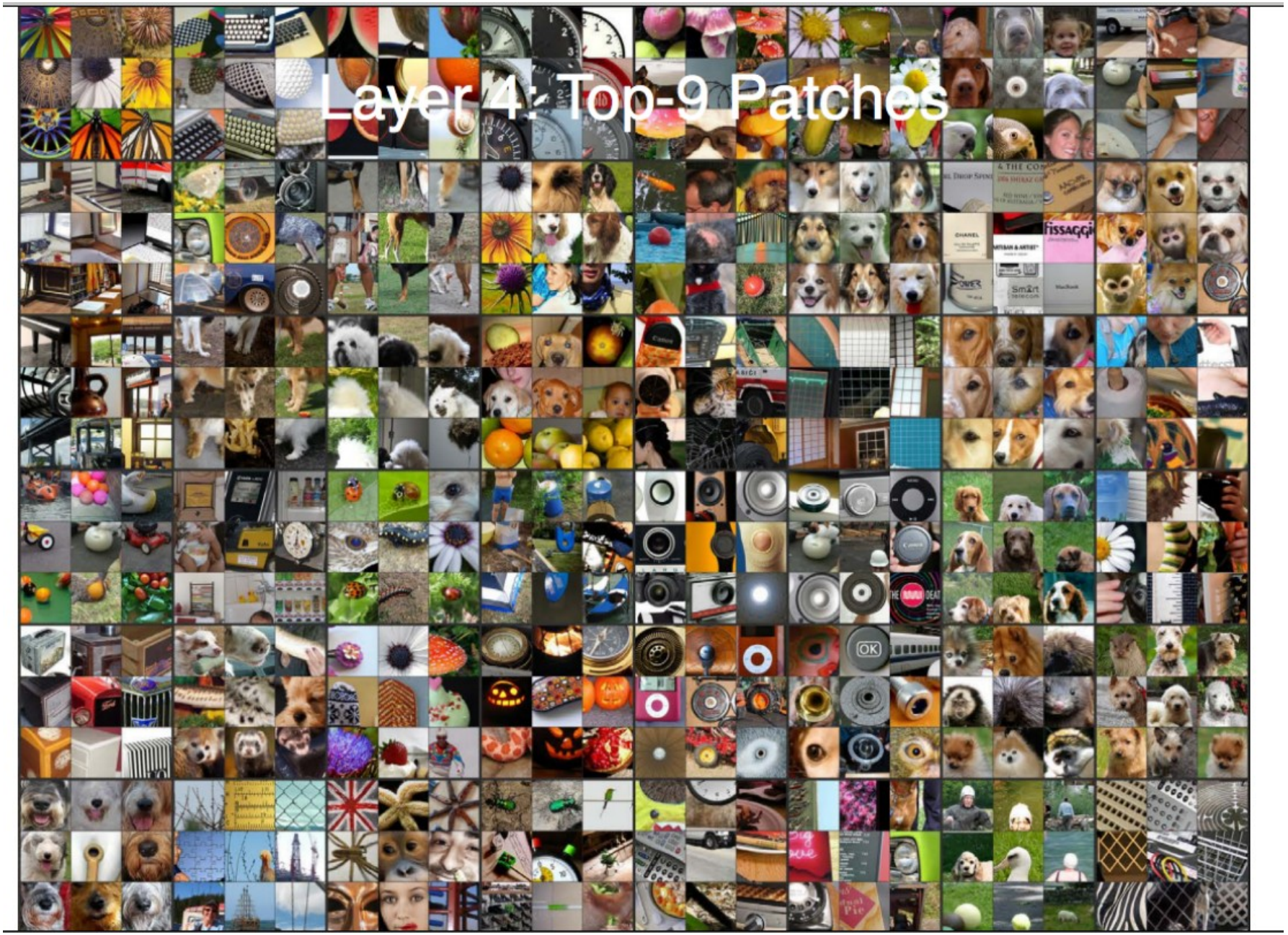
Top 9 patches that activate each filter in layer 1

Each 3x3 block shows the top 9 patches for one filter.




Layer 2: Top-9 Patches


Layer 3: Top-9 Patches


Layer 4: Top-9 Patches


Layer 5: Top-9 Patches

Few properties of Deep Neural Networks

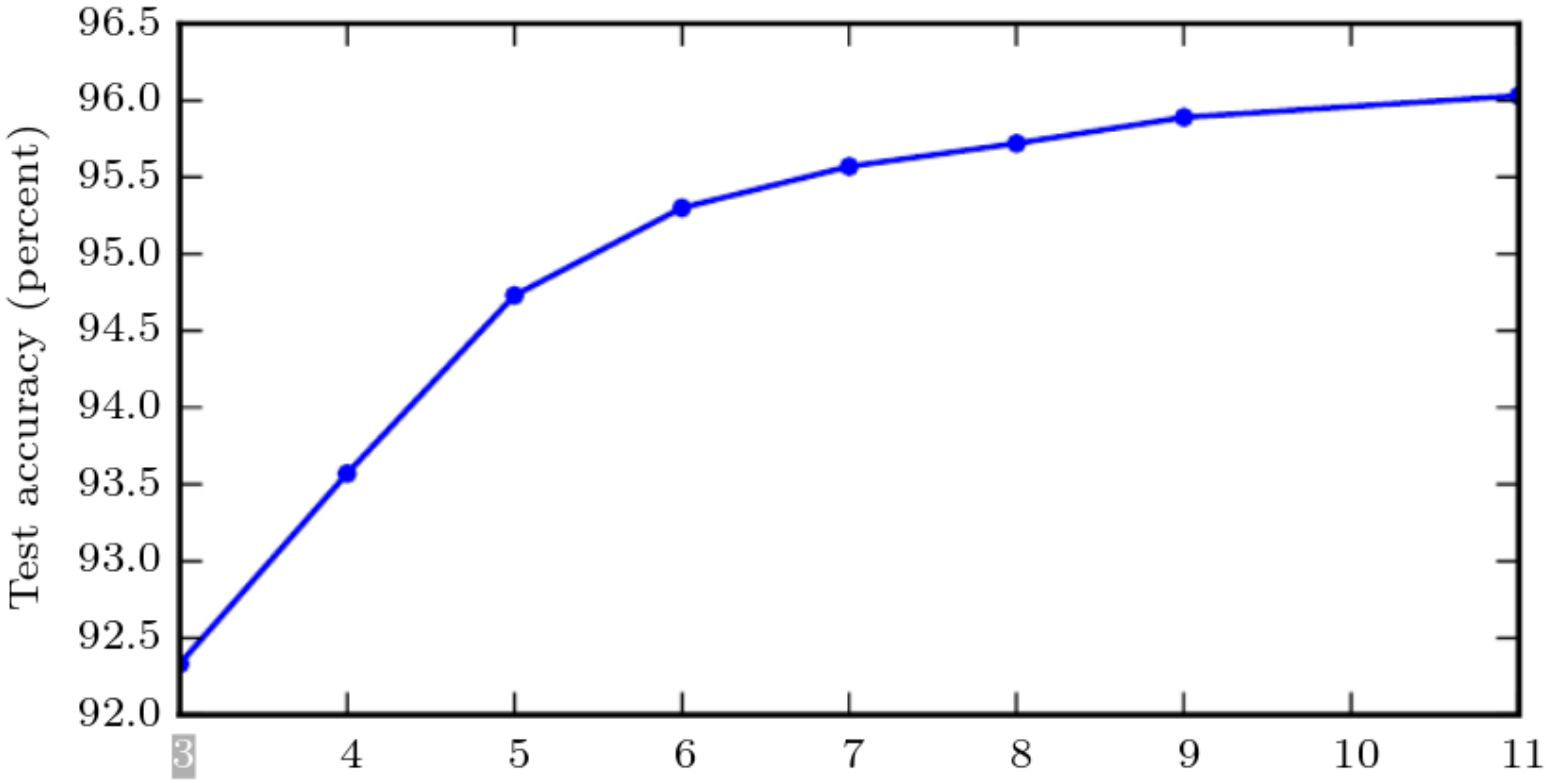


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from Goodfellow *et al.* (2014d). The test set accuracy consistently increases with increasing depth. See figure 6.7 for a control experiment demonstrating that other increases to the model size do not yield the same effect.

http://www.deeplearningbook.org

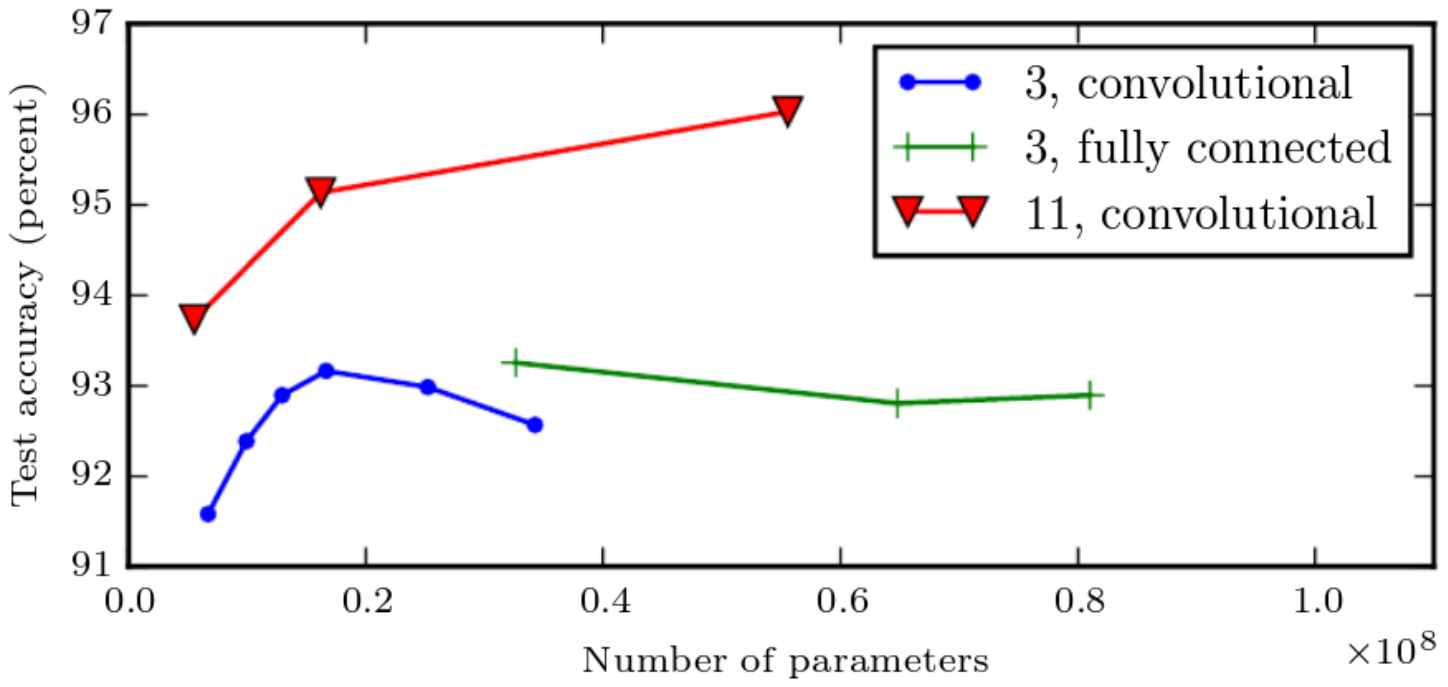
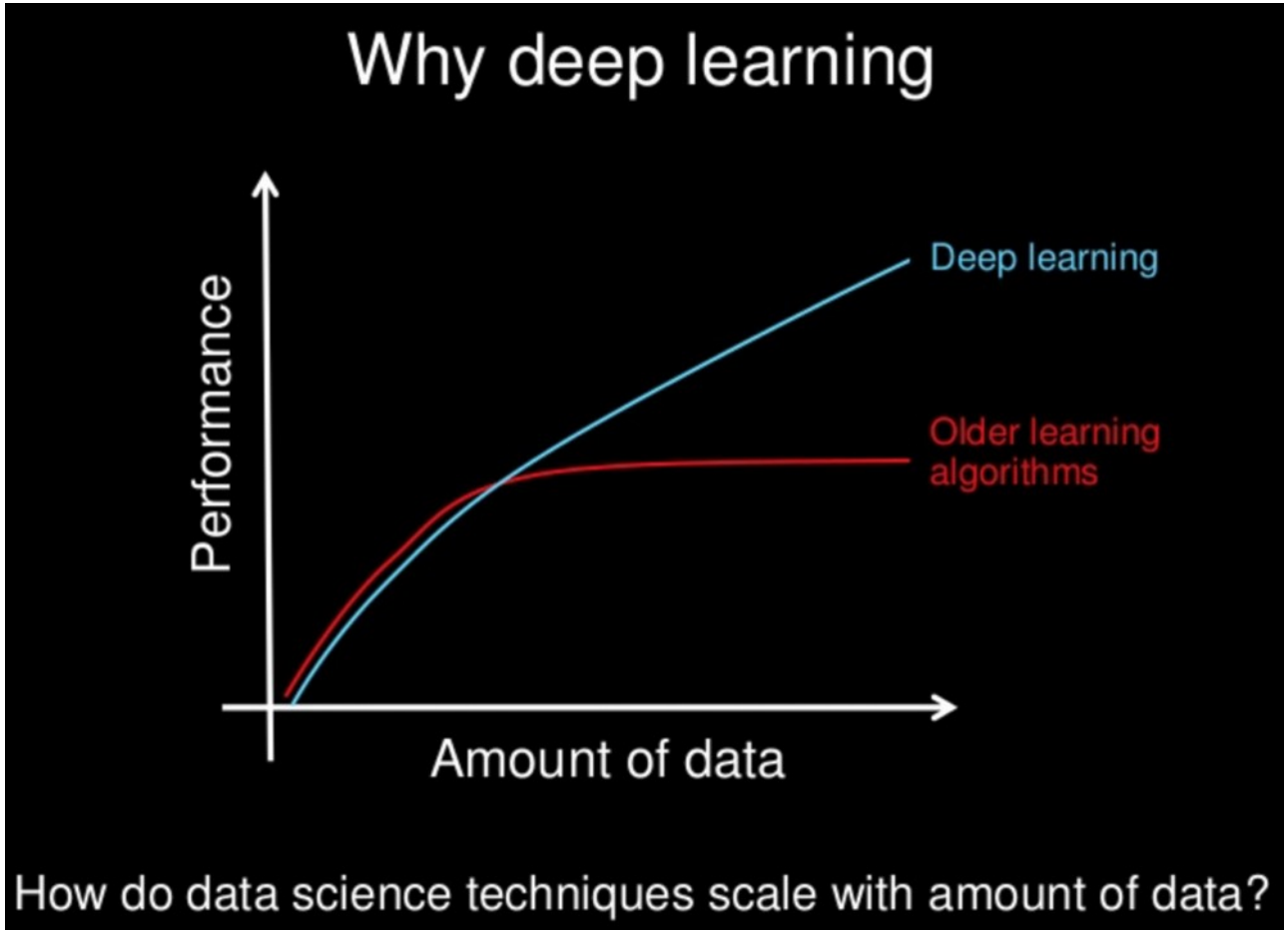


Figure 6.7: Deeper models tend to perform better. This is not merely because the model is larger. This experiment from Goodfellow *et al.* (2014d) shows that increasing the number of parameters in layers of convolutional networks without increasing their depth is not nearly as effective at increasing test set performance. The legend indicates the depth of network used to make each curve and whether the curve represents variation in the size of the convolutional or the fully connected layers. We observe that shallow models in this context overfit at around 20 million parameters while deep ones can benefit from having over 60 million. This suggests that using a deep model expresses a useful preference over the space of functions the model can learn. Specifically, it expresses a belief that the function should consist of many simpler functions composed together. This could result either in learning a representation that is composed in turn of simpler representations (e.g., corners defined in terms of edges) or in learning a program with sequentially dependent steps (e.g., first locate a set of objects, then segment them from each other, then recognize them).

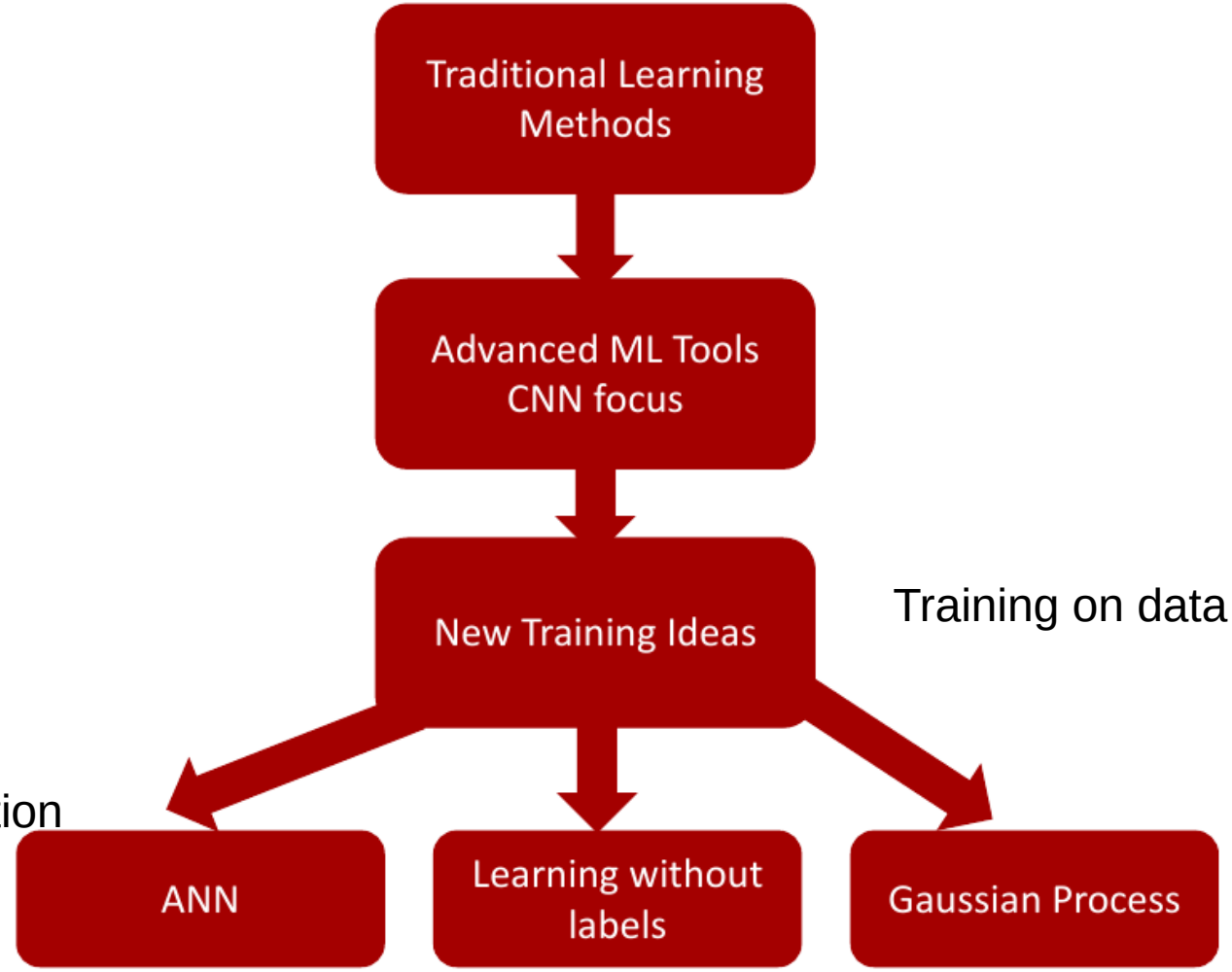
http://www.deeplearningbook.org

Deep learning



- Scales up for huge amount of inputs

Road Map
from Zihao Jiang presentation

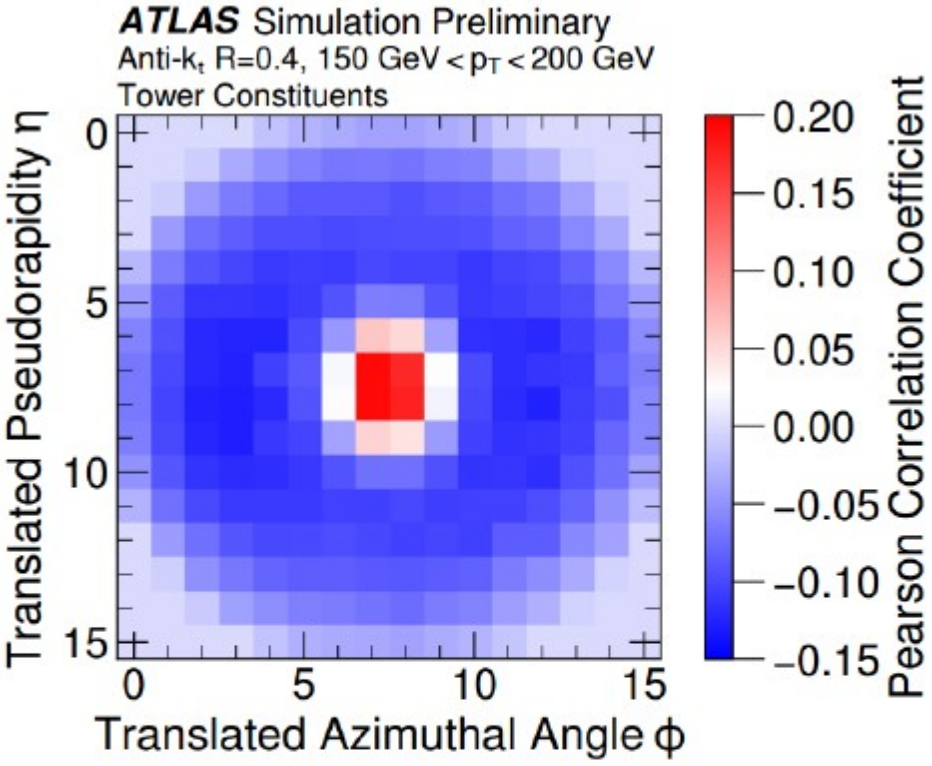


Training on data (not MC!)

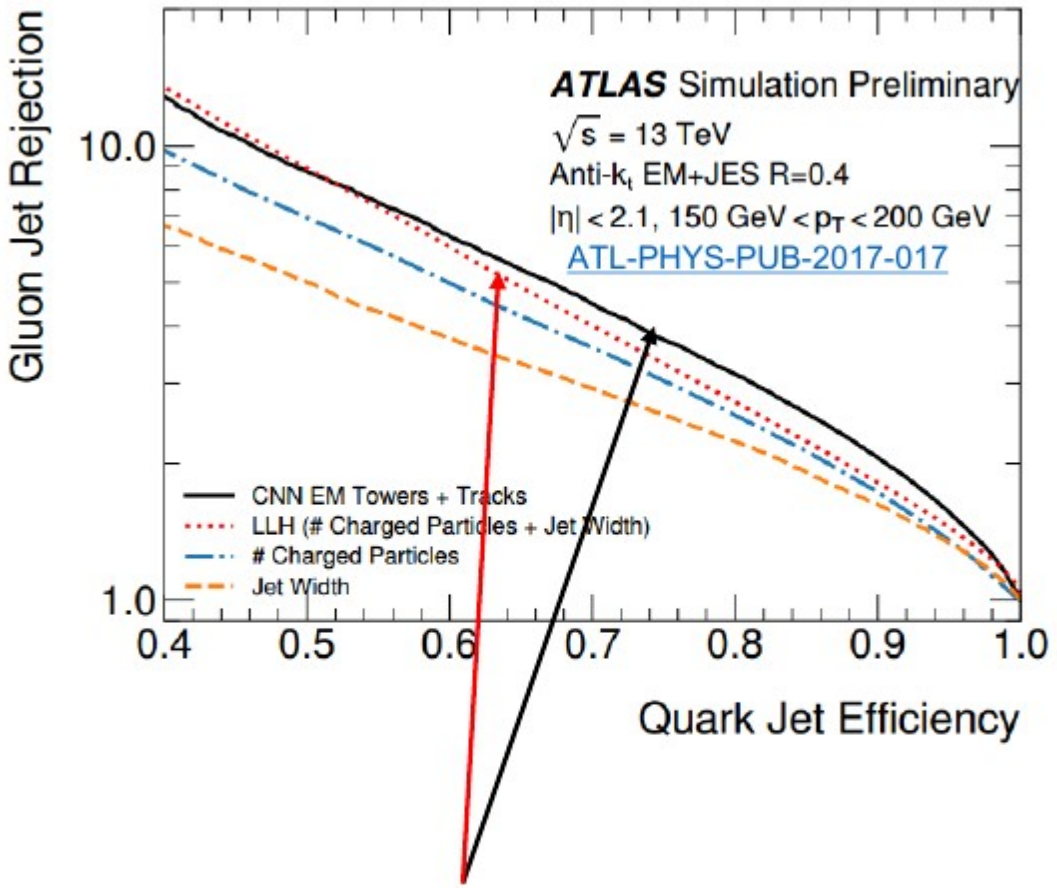
Punish for correlation with some, poorly modeled variables

2017/12/4 Zihao Jiang 3

CNNs

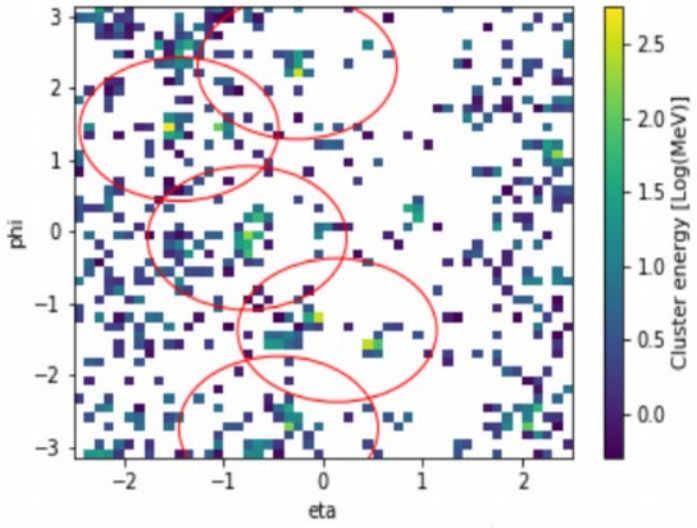


Per pixel correlation between image intensity and CNN output.
The four pixels at the core is highly correlated with jet being a quark jet



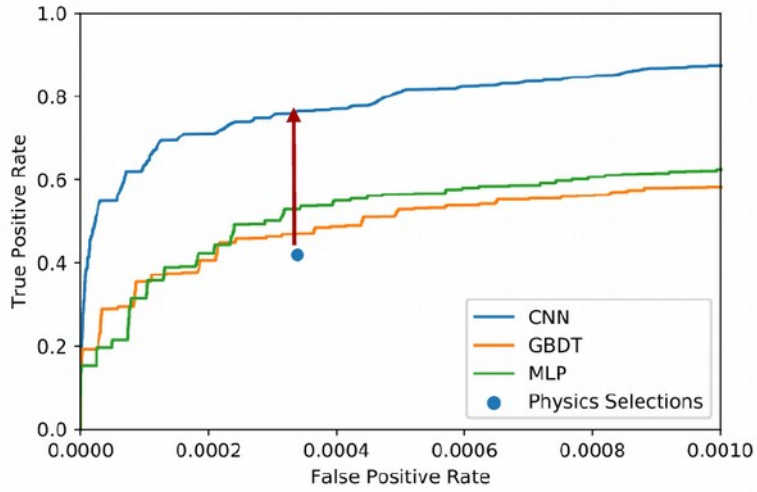
CNN as an entirely different approach than building likelihood from high level quantities show improvement of quark vs. gluon classification

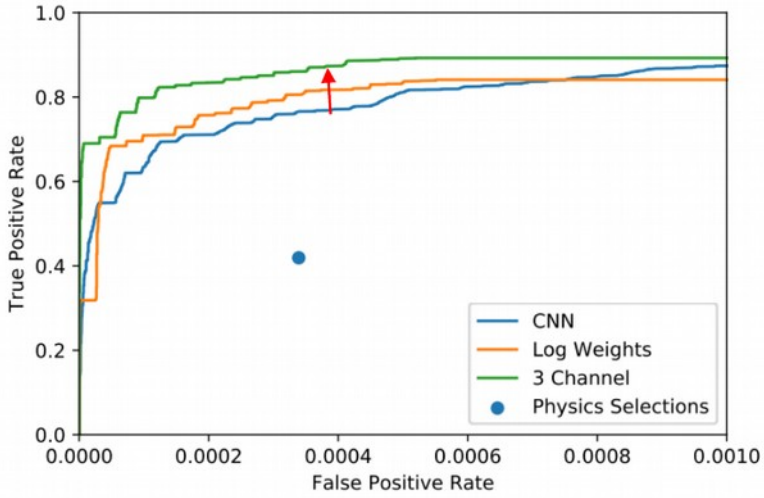
CNNs



Event Classification:
Search for RPV SUSY gluino decays

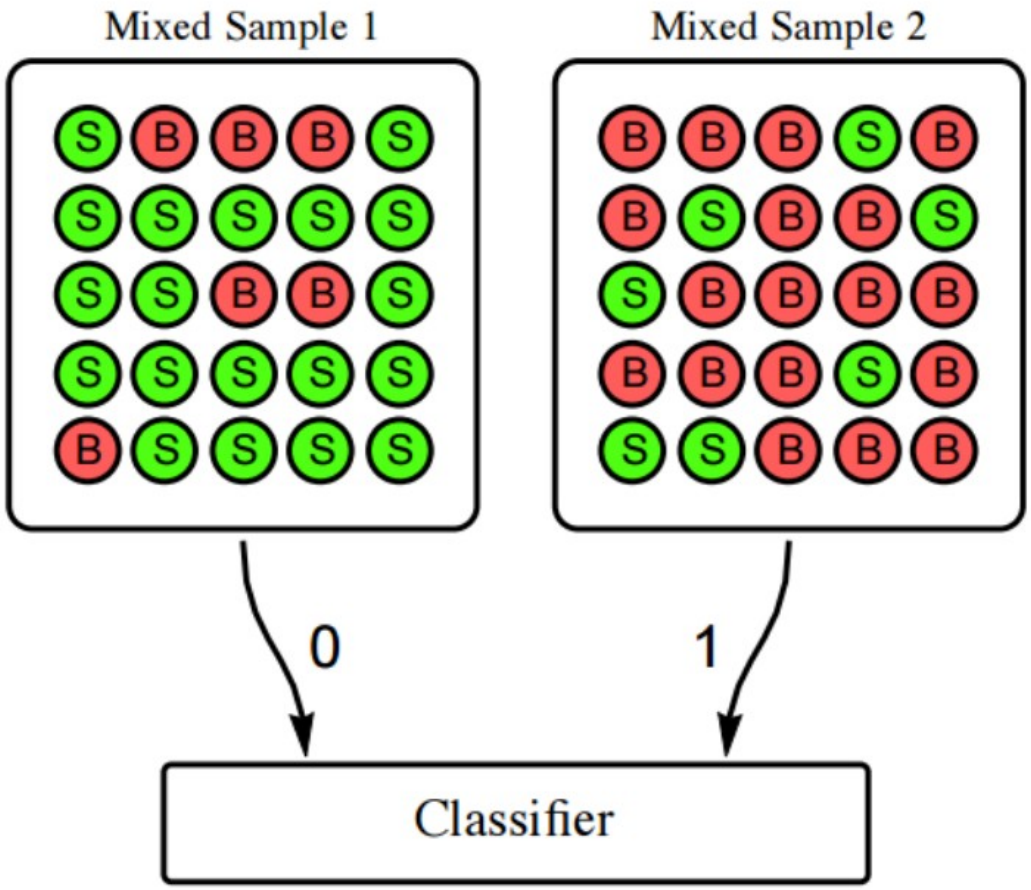
- Image content = calo tower energy
- CNN outperforms typical classifiers
- 3-channel image (Ecal, Hcal, Track) further boosts the performance
- NOTE: soft activity systematics missing





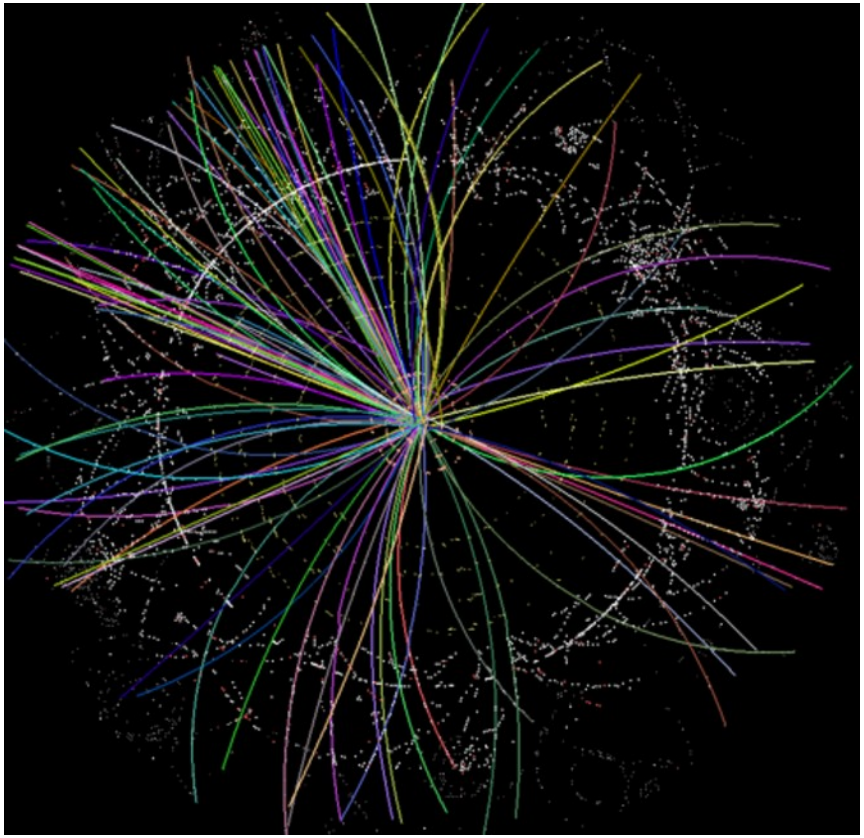
Learning from Data
--Classification w/o Labeling

- A step even further is classification w/o labeling (CWoLa, 1708.02949) https://arxiv.org/abs/1708.02949
- A classifier is trained to distinguish sample 1 from sample 2 which are mixture of signal and background with different fractions
- Such a classifier is optimal for distinguishing signal from background



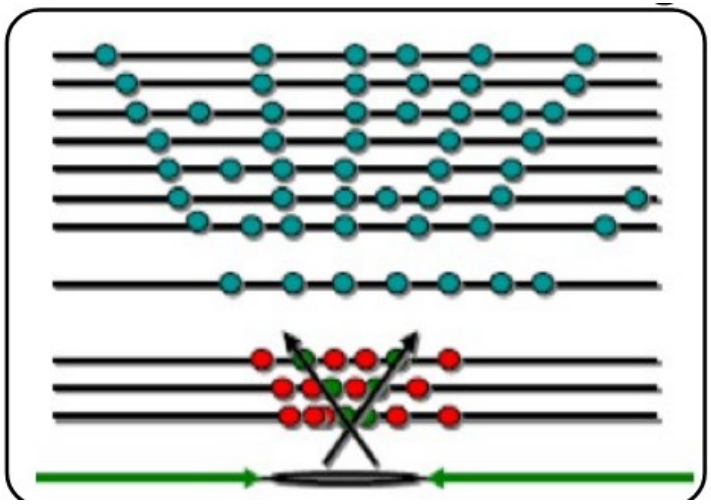
Tracking – the current status

- Tracking in High Energy Physics – finding the tracks of charged particles from the hits they leave in detectors.
- Number of collisions per event increases – for upgraded CERN experiments (High Luminosity LHC) we expect thousands of tracks.

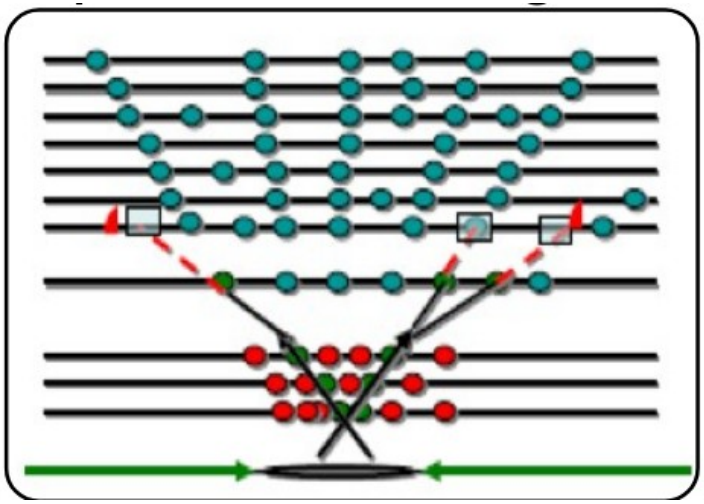


- Current tracking algorithms have been used very successfully in High Energy Physics experiments
- Good efficiency and modeling with acceptable CPU consumption
- Problem: they don't scale so well to expected high luminosity conditions
- Thousands of charged particles, $O(10^5)$ 3D space-points, while the standard algorithms scale worse than quadratic
- Standard algorithms are intrinsically sequential

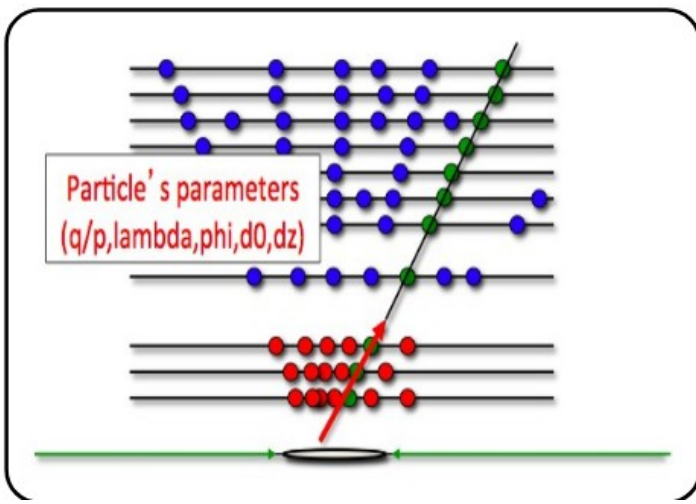
Pattern Recognition in High Energy Physics



Seeding



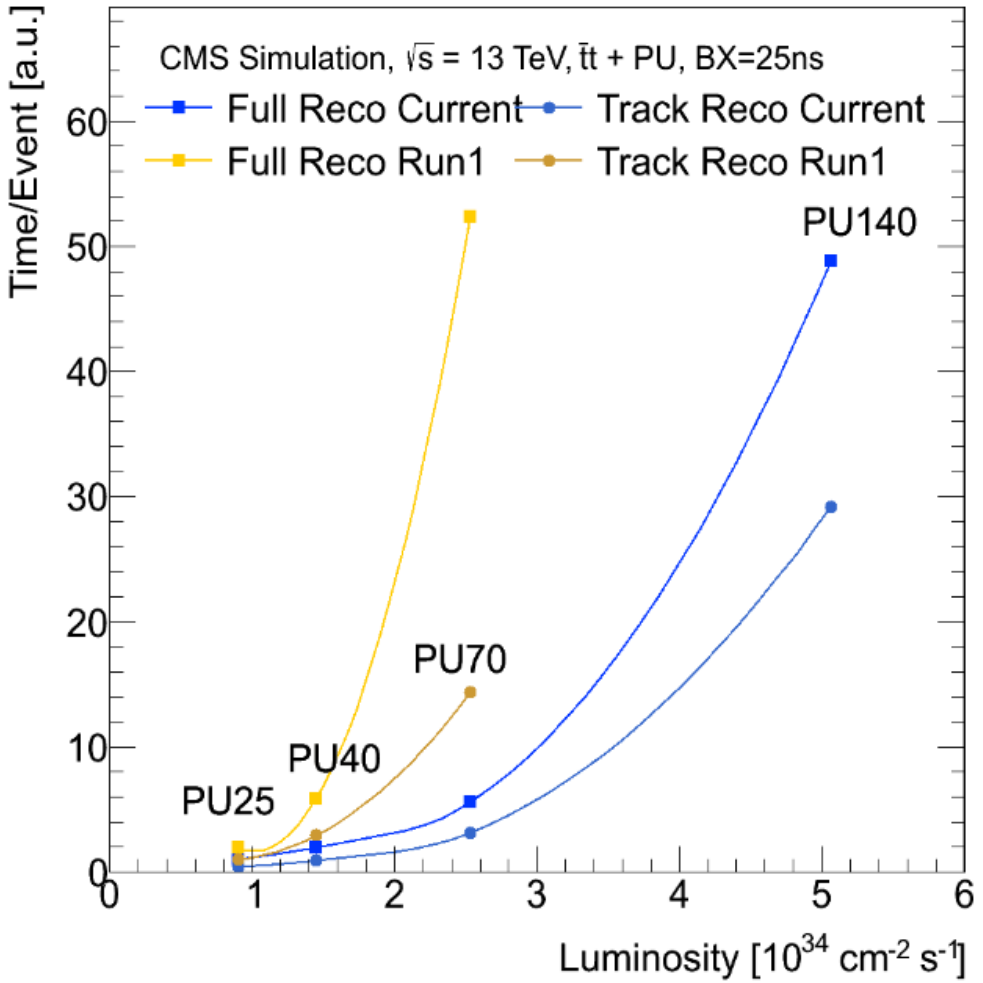
Track Building



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)

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 with luminosity (number of collisions).
- Huge part of CPU consumption is the track finding.
- Will be a bottle-neck for the future experiments.

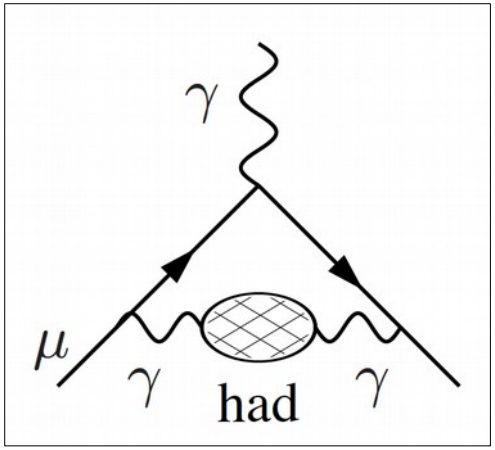
Deep Neural Networks (dnn)?

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

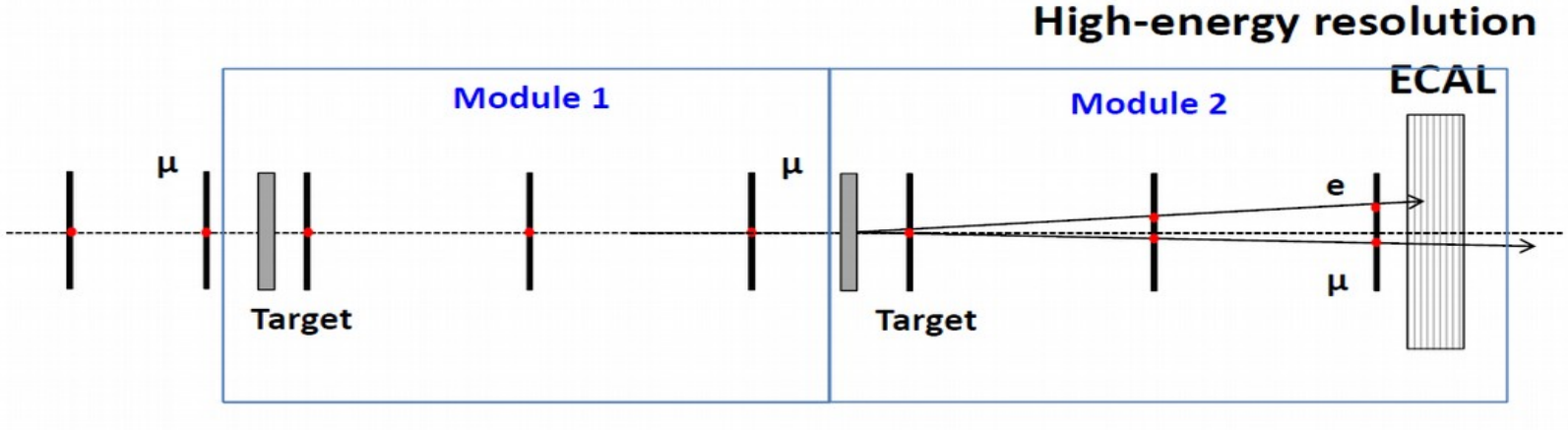
Our goal - application to MUonE

- 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 testbeam 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 - ’classical’ approach

• construct pairs from all the combinations of hits ***(CPU TIME CONSUMING)***

• clusterization of two adjacent hits ***(COMPLICATED ALGORITHMS)***

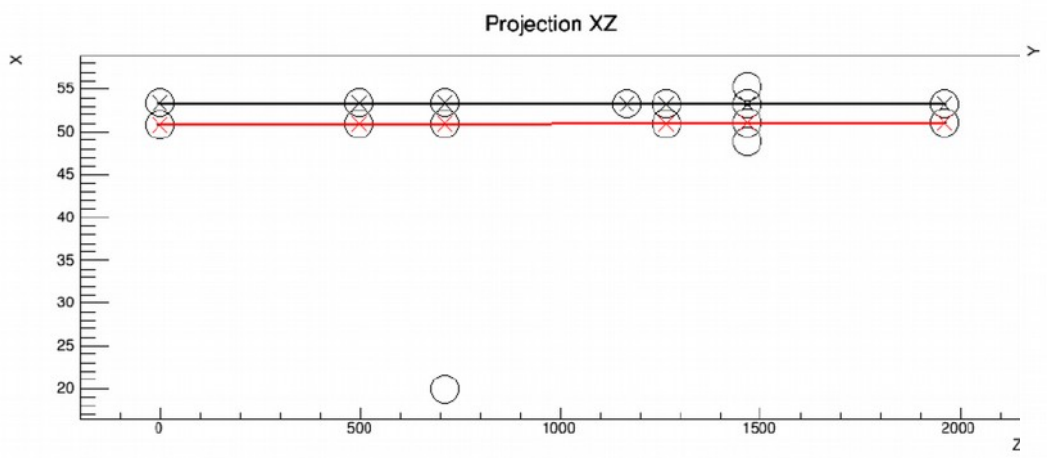
→ reduces number of clones

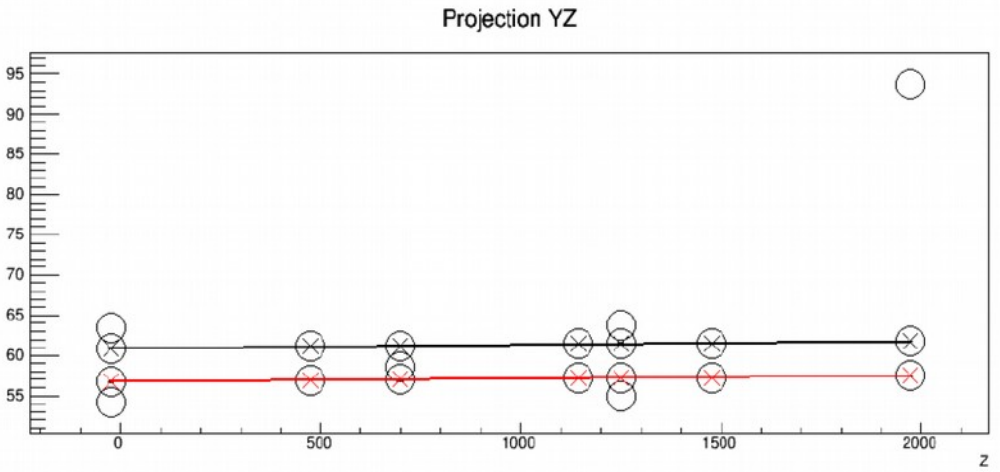
→ improves reconstruction of two close tracks

• for each hit pair construct a line and collect all the hits within a certain window

→ all combinations of hits for a lines are fitted (CPU TIME CONSUMING)

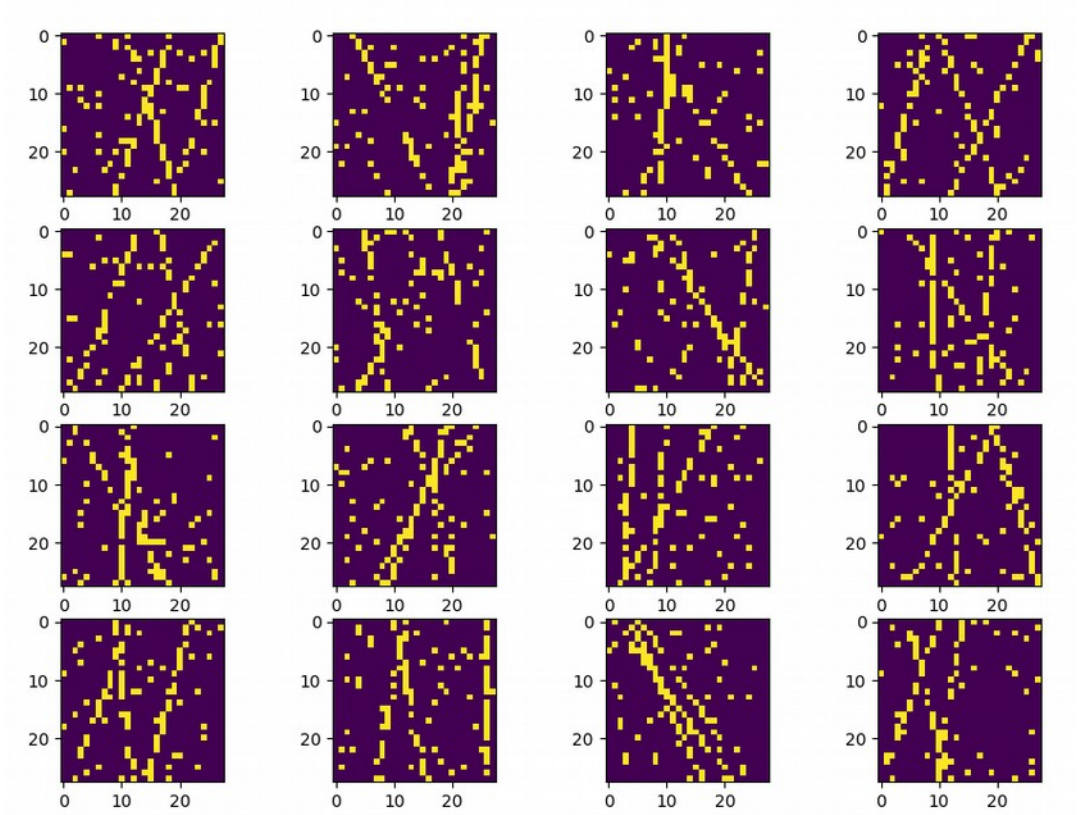
• **potential problems with final precision and tracking efficiency**





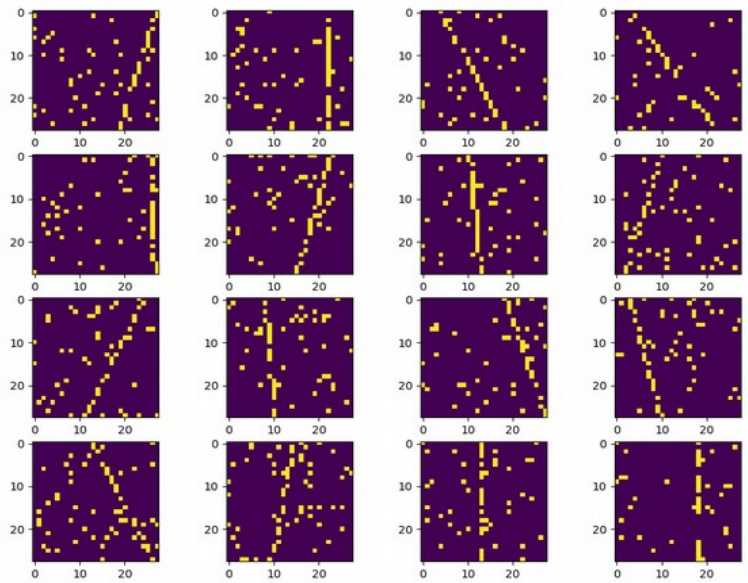
Using DNN → hope to improve all bottlenecks!

Toy model of track finding using Deep Neural Networks

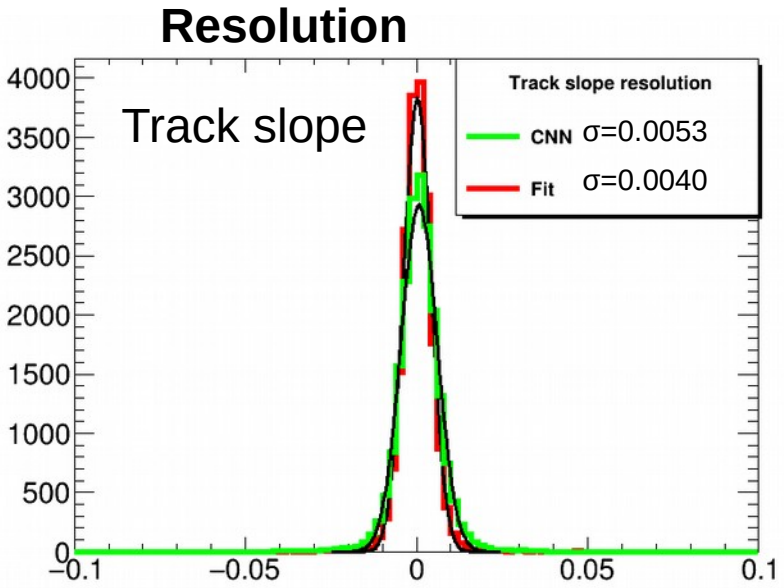


- Characteristics of the toy MC:
  - 2-dimensional data to reduce the training time
  - Straight line tracks (no magnetic field) on the 28x28 pixel plane
  - Finite hit efficiency and random noise
- Reconstruction method: training Convolution Deep Neural Network to identify a track and return track parameters (one of the propositions of the HEPTrkx group)
- LSTM (Long Short-Term Memory units) to process a sequence of tracks (not only a single one).

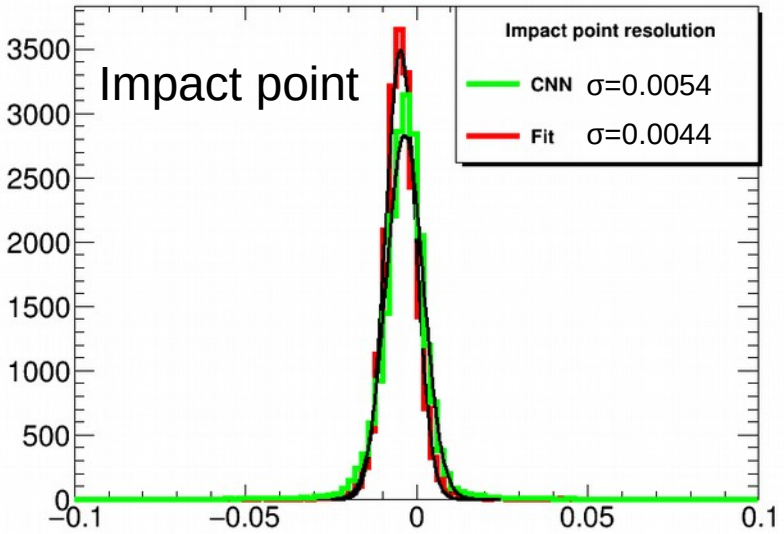
Single track events



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

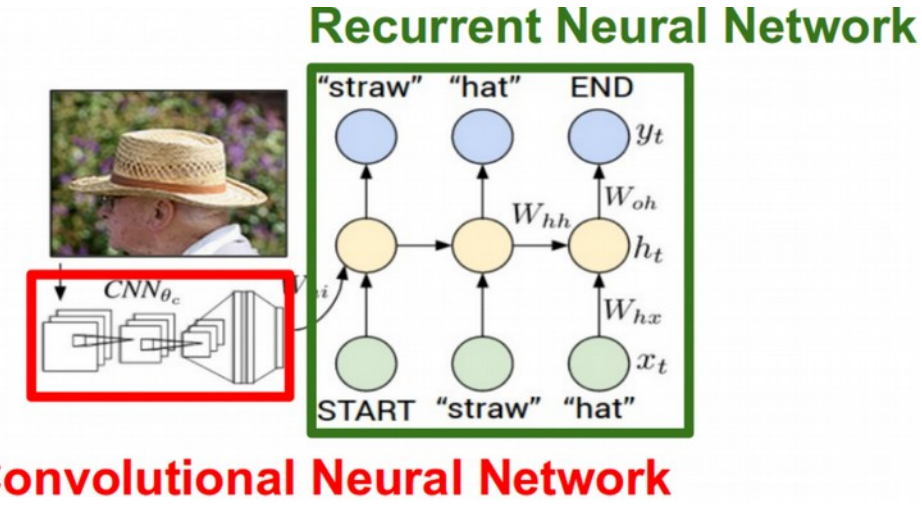


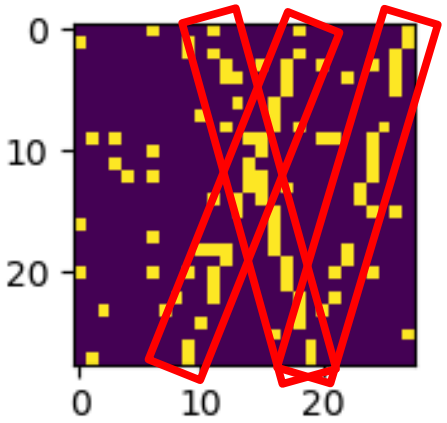
- After patter recognition by CNN the track is fitted using the ROOT package robust fit (removing the outliers).
- Results: track parameter resolution only slightly worse for neural network than for the fit.
- **Efficiency: over 99%** (reconstructed track: found within 5 pixels from the true track).
- *Software used:*
  - *Tensorflow - https://www.tensorflow.org/*
  - *Keras - https://keras.io/*
  - *ROOT - https://root.cern.ch/*



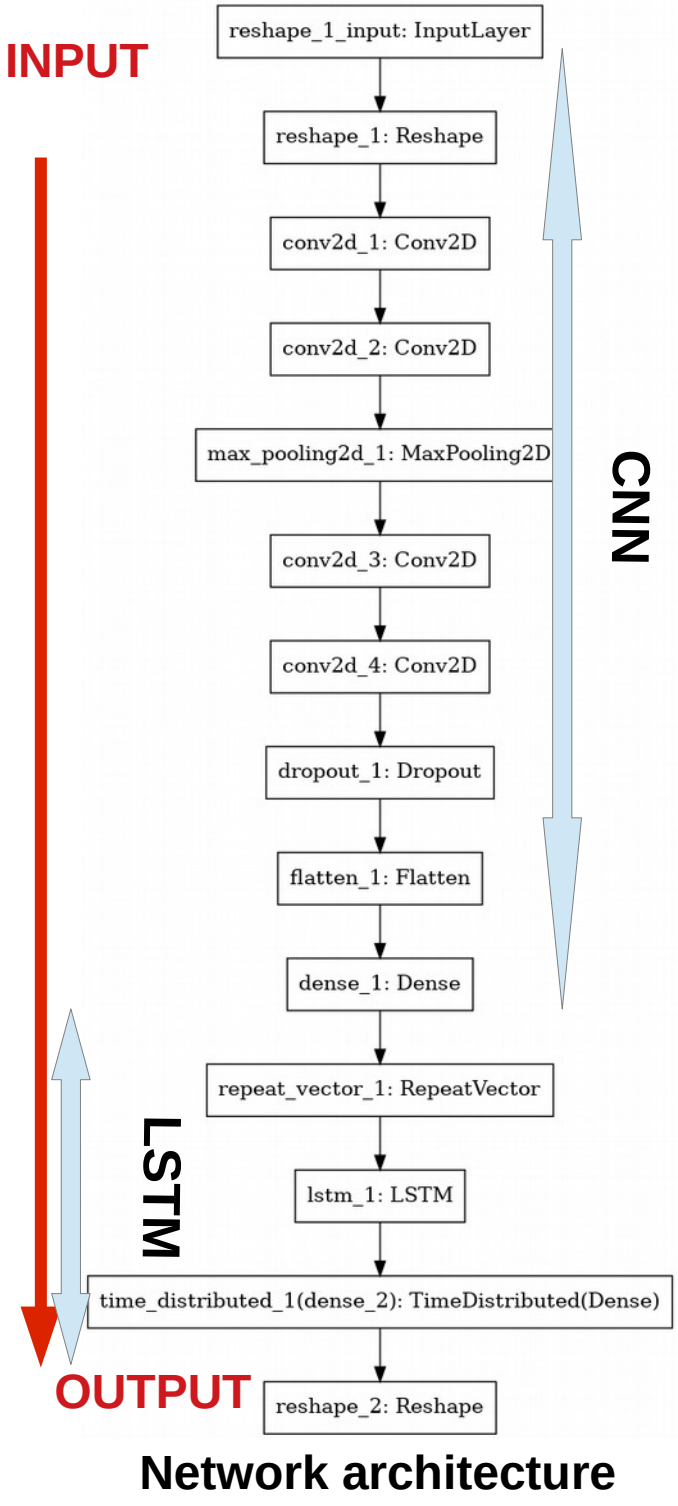
What about many tracks?

- Solution: add Long Short-Term Memory (LSTM) layer.
- LSTM allows labeling images, so why not use it for labeling tracks?





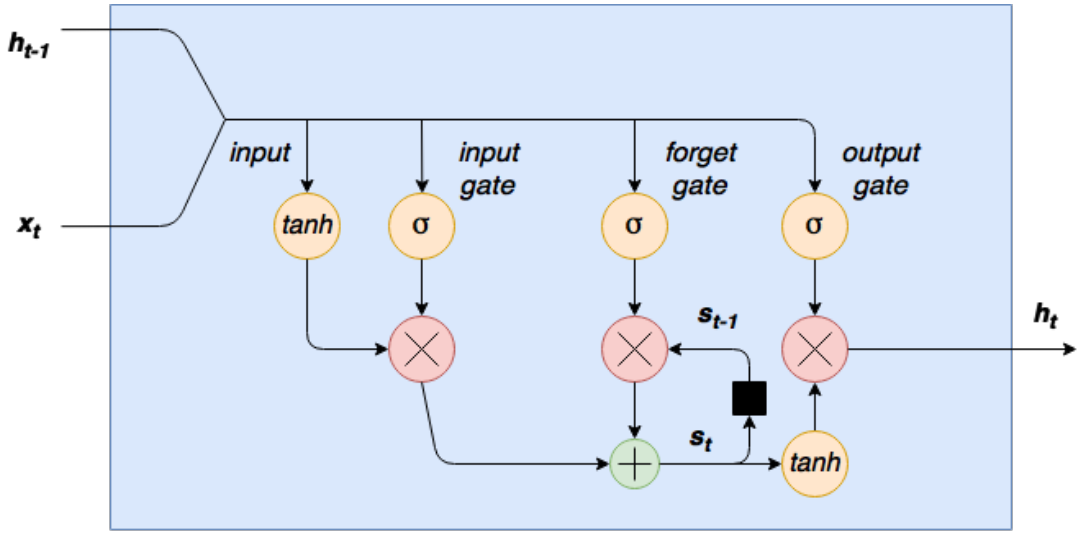
CNN extracts the features from our input image. The LSTM network is trained as a language model on the feature vector.



Network architecture

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).



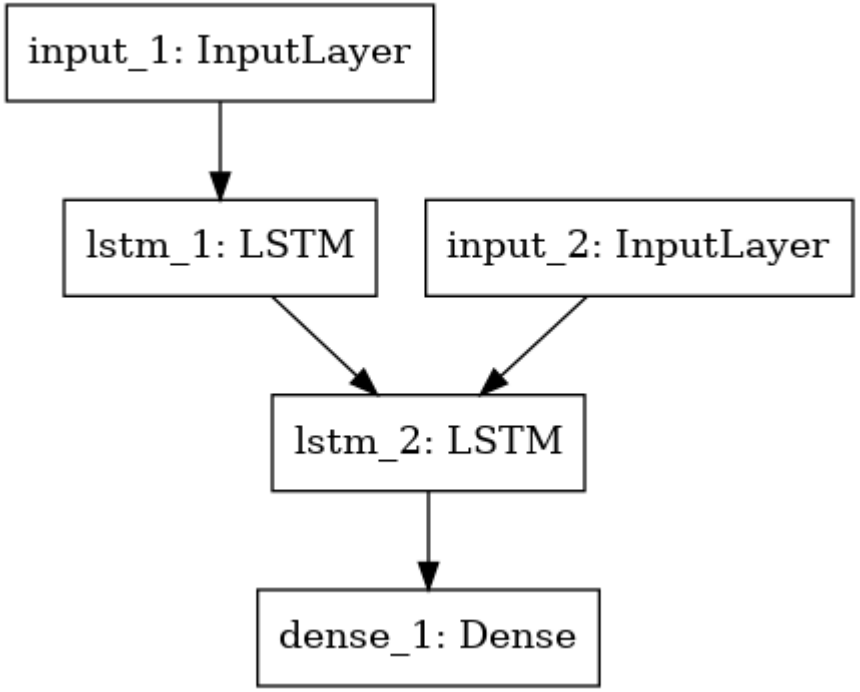
- LSTM is an artificial recurrent neural network . Unlike standard feedforward neural networks, LSTM has feedback connections that make it a "general purpose computer" (that is, it can compute anything that a Turing machine can)!!!
  It can not only process single data points (such as images), but also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

LSTM for language translation

https://github.com/keras-team/keras/blob/master/examples/lstm_seq2seq.py

- Input sentence: Be nice.
- Decoded sentence: Soyez gentille !
- Input sentence: Beat it.
- Decoded sentence: Dégagez-les.
- Input sentence: Call me.
- Decoded sentence: Appellez-moi !
- Input sentence: Call us.
- Decoded sentence: Appelle-nous !
- Input sentence: Come in.
- Decoded sentence: Entre !
- Input sentence: Drop it!
- Decoded sentence: Laissez tomber !

Basic **character-level** sequence-to-sequence model. We apply it to translating short English sentences into short French sentences, character-by-character.
Note that it is fairly unusual to do character-level machine translation, as word-level models are more common in this domain.

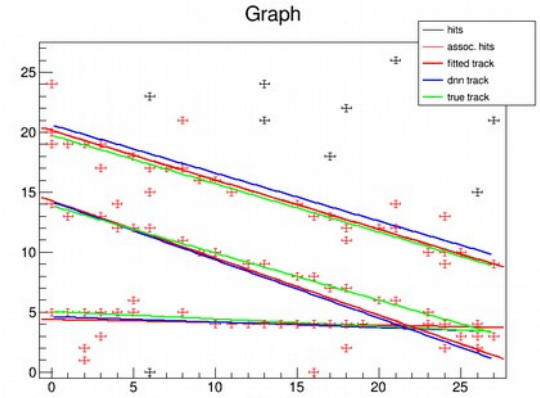


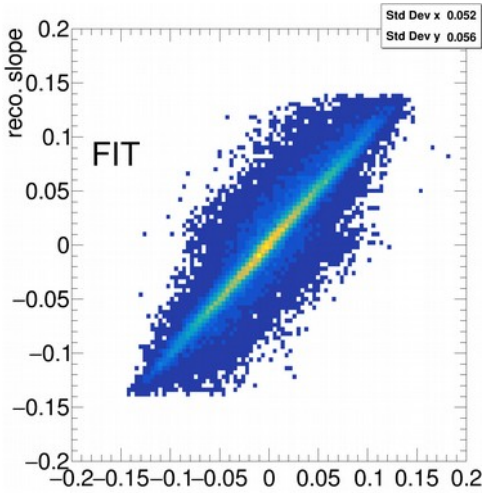
Tests with more tracks and higher noise

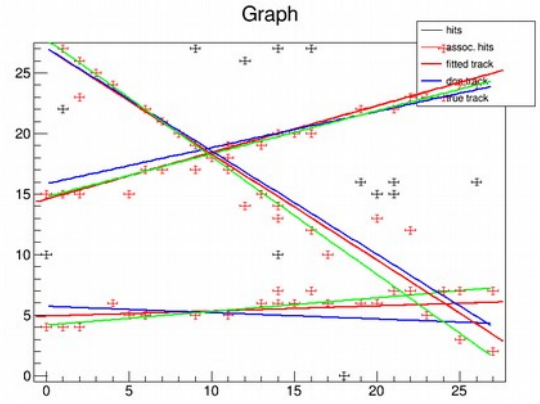
- **3 tracks**
- **70% hit efficiency**
- **5% random noise**

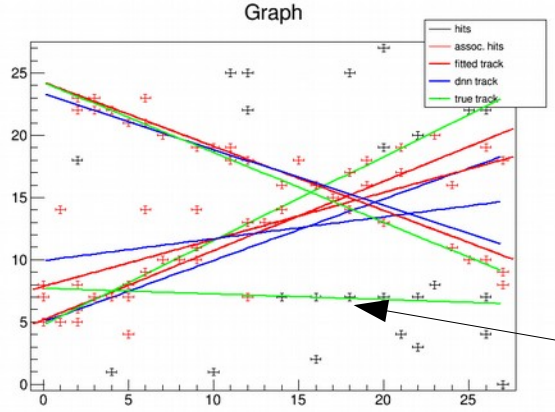
Slightly worse resolution achieved by CNN than the fit performed afterwards. Fit gives narrower resolution in the central part, but the tails are comparable to the CNN.

Efficiency: 87%



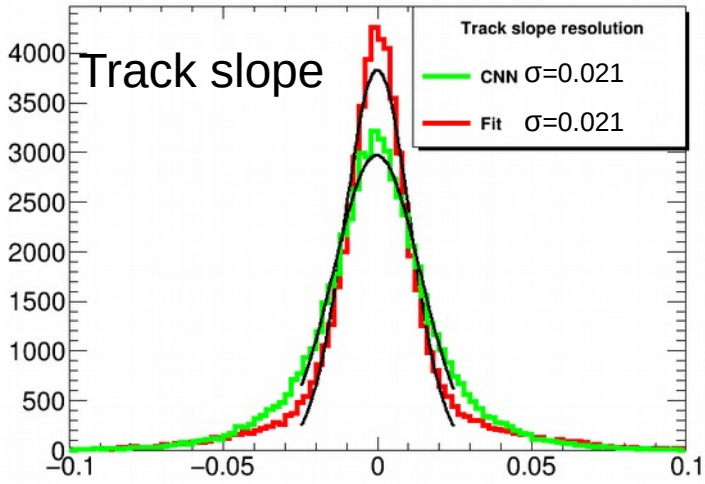


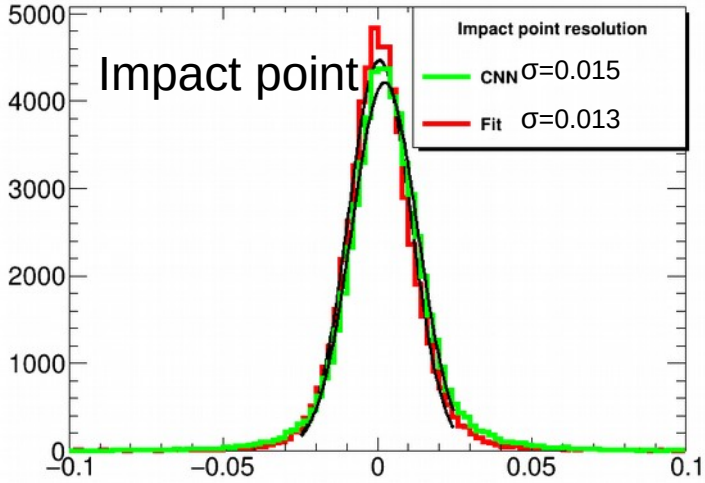




Failure

Resolution

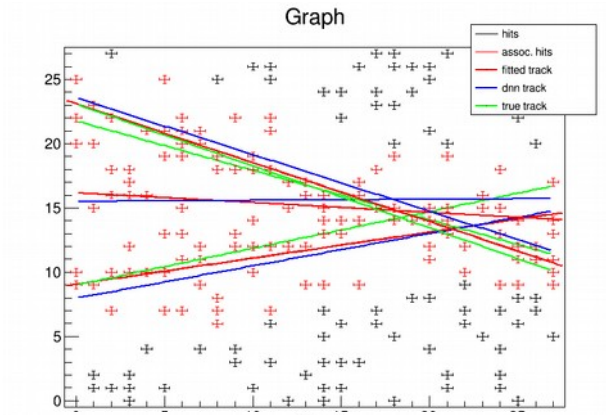


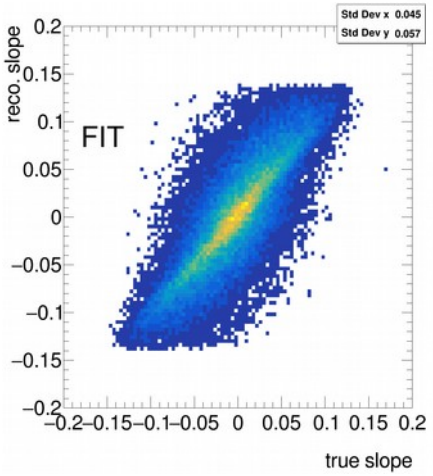


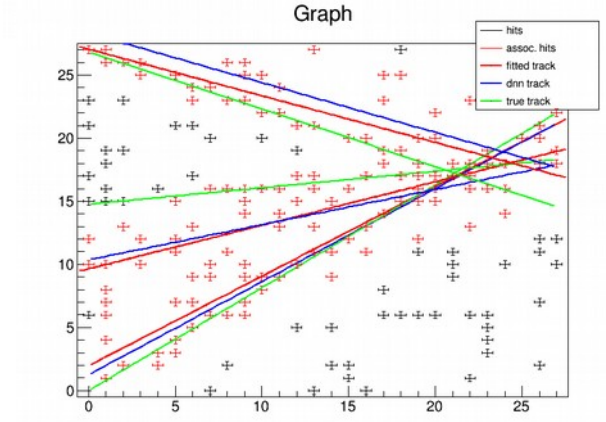
Tests with more tracks and higher noise

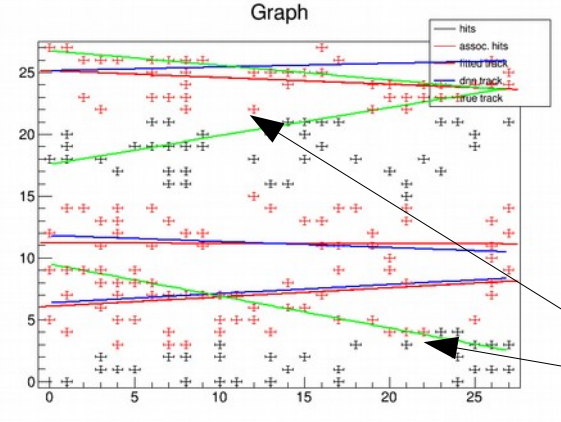
- **3 tracks**
- **70% hit efficiency**
- **20% random noise**

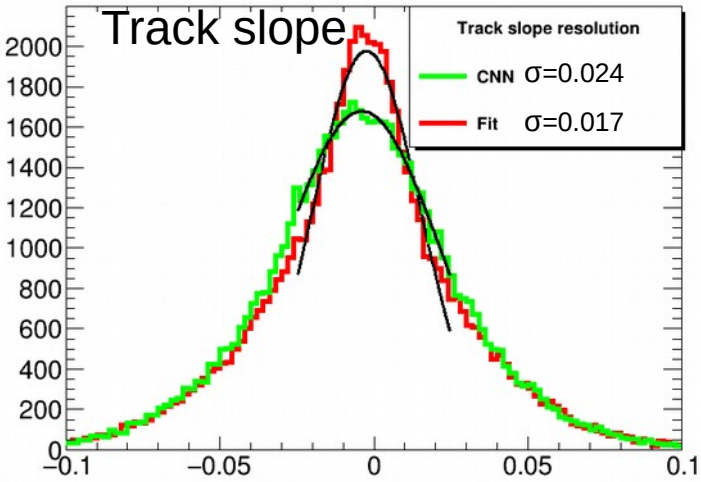
Tracks are hardly visible by eye, but still could be recognized by the network.
The resolution and efficiency degrade.
Efficiency CNN: 66%

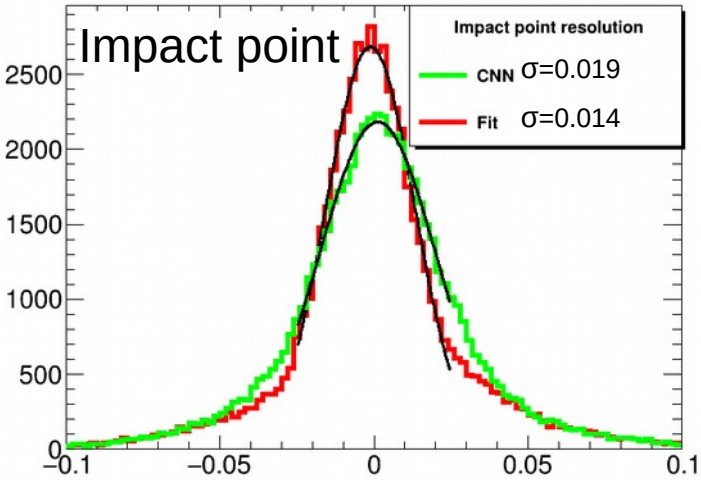












Failure

Results

- Robust and effective track finding in the toy model data
- Quite insensitive for noise and hit inefficiency
- 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

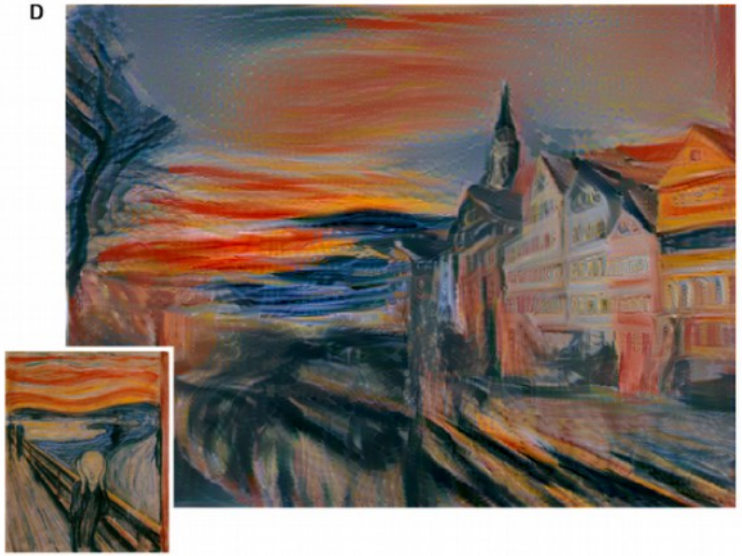
The pattern recognition is very fast! 40 times faster than fitting performed afterwards.
Resolution obtained by Convolutional Neural Network is not much worse, than the one from the robust fit.

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 will be used for pattern recognition in MuonE, which should improve it's precision and efficiency. It's crucial, since anomalous magnetic muon should be measured with precision better than 0.5% to produce a useful result.
- The approach might be not so straight forward, it might be a hybrid of neural network and traditional methods (ideas from HEP.TrkX group) :
  - Track seeding
  - Hit classification with LSTMs
  - Hit classification (track extrapolation) with CNN
- Novel techniques, worth trying on real data (test beam).
- We will need much more computer power for network training (Prometeus, GPU?)

Calculations done on Zeus using LHCb grant no. lhcbflav08

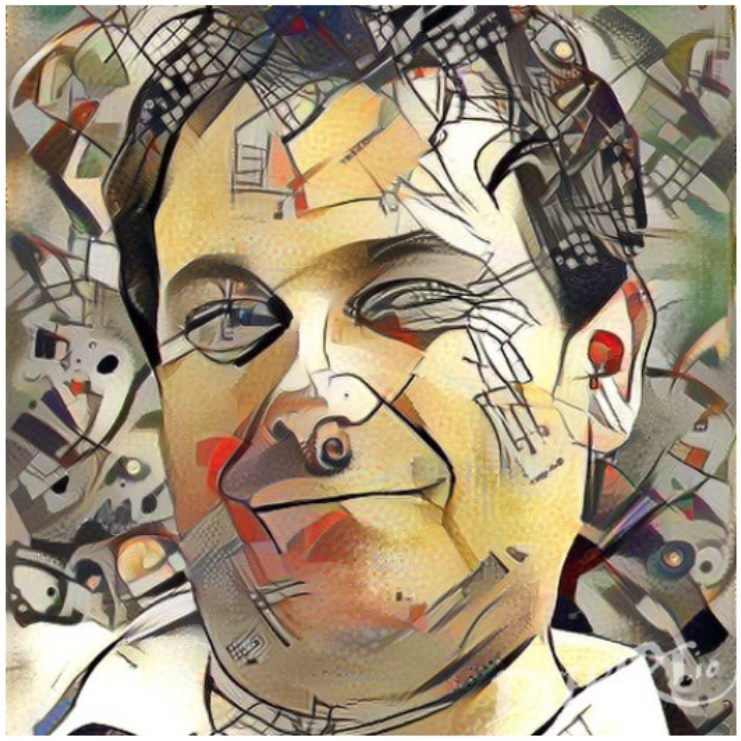
Deep Neural Network for artists :)



DeepArt.io

“A Neural Algorithm of Artistic Style”, arXiv:1508.06576

DeepArt.io




DEEPART.io




The Automated HEP Physicist?

- A few years from now our automaton could do on our behalf:
- Automatically determine the set of characteristics that distinguish particles from the primary vertex from those from other vertices and automatically classify particles based on this information.
- Automatically reduce particle event data into a smaller fixed set of numbers, say N ~ 500 – which may be thought of as “pixelized images” – that can be the basis of further analysis.
- Automatically classify these “images” into two sets: those that look like simulated events and those that don’t.
- Find more sets and classify the events according to MC classes.

Conclusions inspired by H.B. Prosper “Deep Learning and Bayesian Methods”, https://indico.cern.ch/e/conf12

Deep Learning Software (few packages)

- **Theano** is a low-level library that specializes in efficient computation. You'll only use this directly if you need fine-grain customization and flexibility.
- **TensorFlow** is another low-level library that is less mature than Theano. However, it's supported by Google and offers out-of-the-box distributed computing.
- **Lasagne** is a lightweight wrapper for Theano. Use this if need the flexibility of Theano but don't want to always write neural network layers from scratch.
- **Keras** is a heavyweight wrapper for both Theano and Tensorflow. It's minimalistic, modular, and awesome for rapid experimentation. This is our favorite Python library for deep learning and the best place to start for beginners.
- **TMVA/root** is now interfaced to Keras (root 6.08)

Exercises

ATLAS Z → tau tau selection

- Data:
  - mc12/Ztautau.root - signal
  - Powheg_ttbar.root - bckg
  - Wenu.root - bckg
  - Wmunu.root - bckg
  - Wtaunu.root - bckg
  - Zee.root - bckg
  - Zmumu.root - bckg
- Variables:

  preselection:

  if(!(evtsel_is_dilepVeto > 0 && evtsel_is_tau > 0 && fabs(evtsel_tau_eta) < 2.47 && evtsel_is_conf_lep_veto == 1 && evtsel_tau_numTrack == 1 && evtsel_lep_pt > 26 && fabs(evtsel_lep_eta) < 2.4 && evtsel_transverseMass < 70)) continue;

  if (!(evtsel_is_oppositeSign>0 && evtsel_is_mu>0 && evtsel_is_isoLep>0)) continue;

ATLAS Z → tau tau selection

- Variables used for training:
  - *evtsel_tau_et*
  - *evtsel_dPhiSum*
  - *evtsel_tau_pi0_n*
  - *evtsel_transverseMass*
  - *sum_cos_dphi*
- Spectator
  - *vis_mass*
- Program:
  - TMVAClassificationMW.C i TMVAClassificationMW.h
    Performs the basic training.

ATLAS Z → tau tau selection

- Install root & TMVA
- Get data and a sample program:
  - http://nz14-46.4.ifj.edu.pl/cwiczenieATLAS/
- Run a sample code in C++:

  root -l

  .L TMVAClassificationMW.C++

  TMVAClassificationMW t

  t.Loop()
- Modify it:
  - **Try to optimize the parameters of the selected method**
  - **Try to remove or add some variables.**
  - **Try to use individual variables, for example the variables used to build *sum_cos_dphi***
  - **Use all the types of background use the weights *WeightLumi***
- Zaaplikować wyuczony klasyfikator do danych (*data12/Muons.PhysCont.grp14.root*), można się wzorować na przykładzie TMVAClassificationApplication dostępnym na stronie TMVA oraz załączonym przykładzie TMVAClassificationApplicationMW.C.

ATLAS Z → tau tau selection

Make such a plot for visible mass (doesn't need to be so nice).

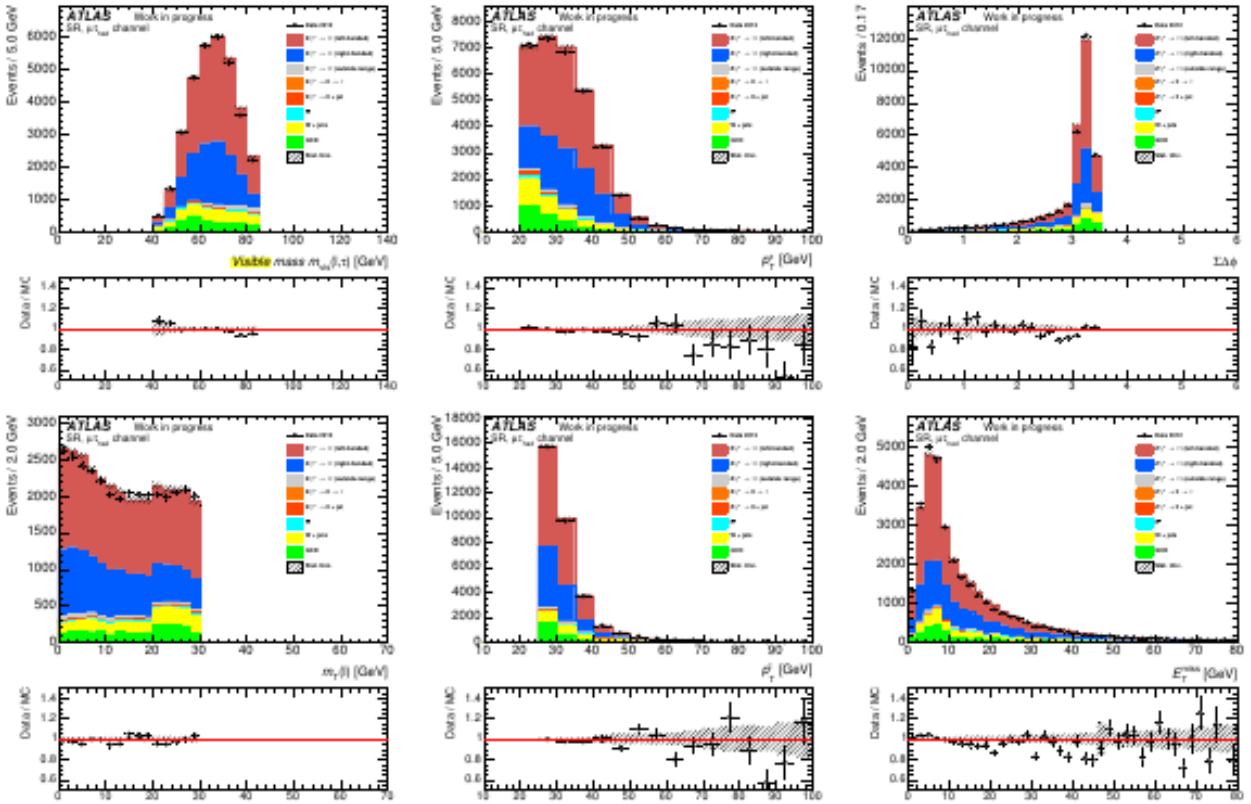


Figure 41: Distributions of variables observed in $Z \to \tau\tau$ (μ-had channel). From top-left: visible mass of τ-lepton system, τ transverse momentum, sum of polar angles between τ and missing-E_T and between lepton and missing-E_T, transverse mass of the lepton-missing-E_T system, lepton transverse momentum and missing-E_T.