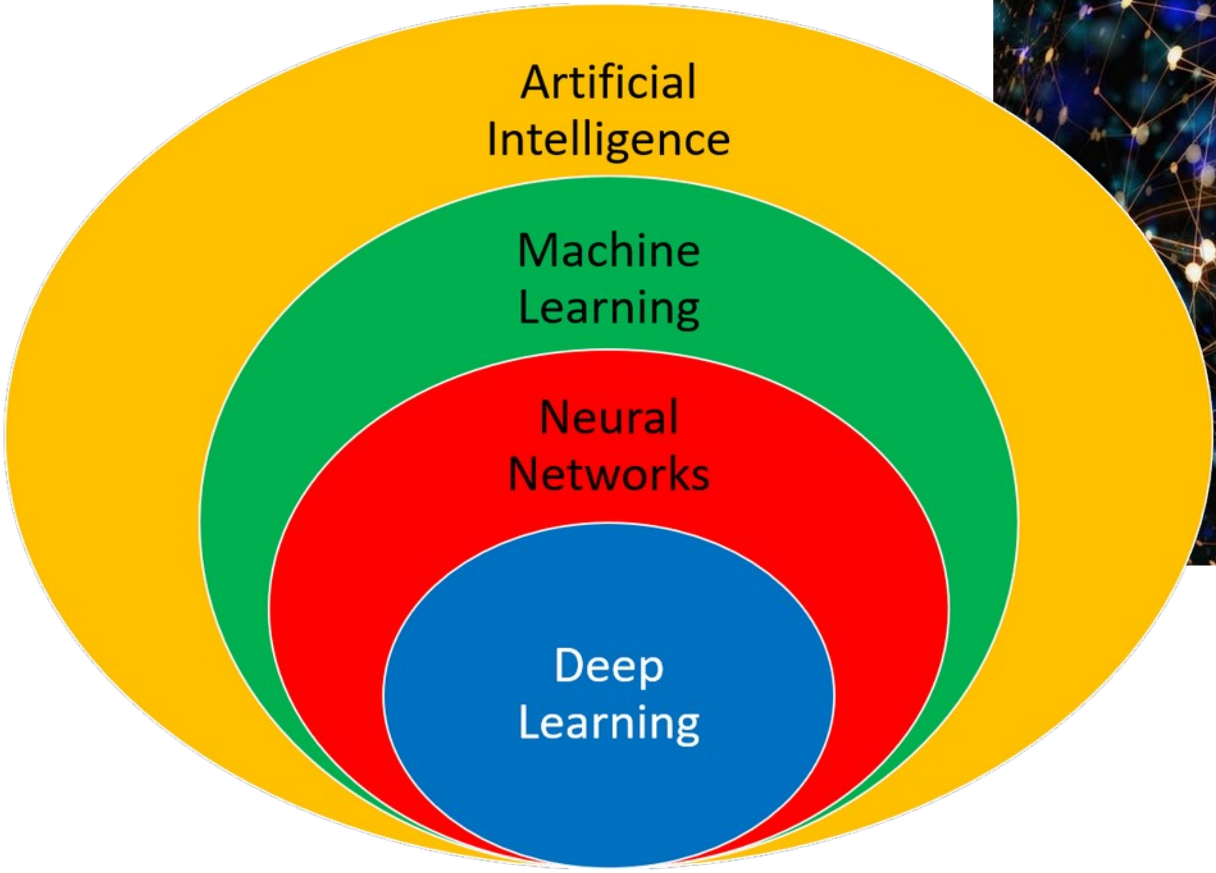
Deep Neural Networks



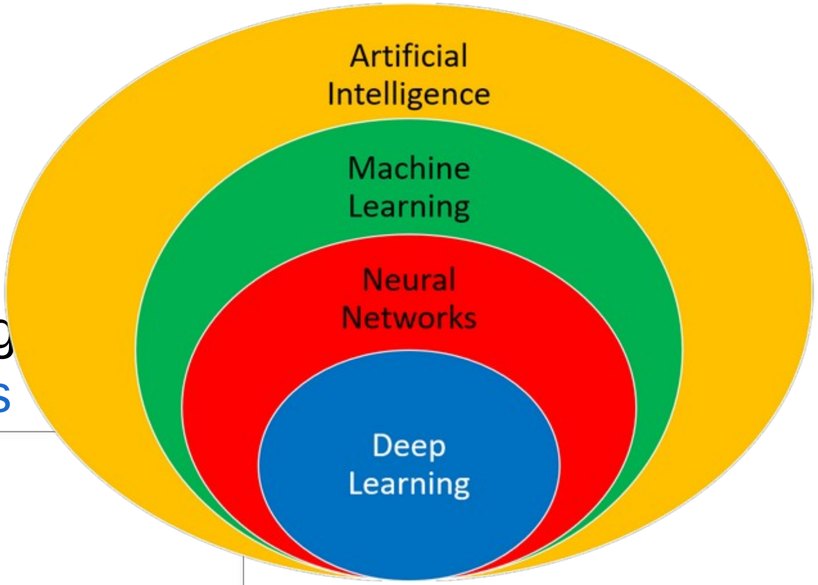
Marcin Wolter
IFJ PAN

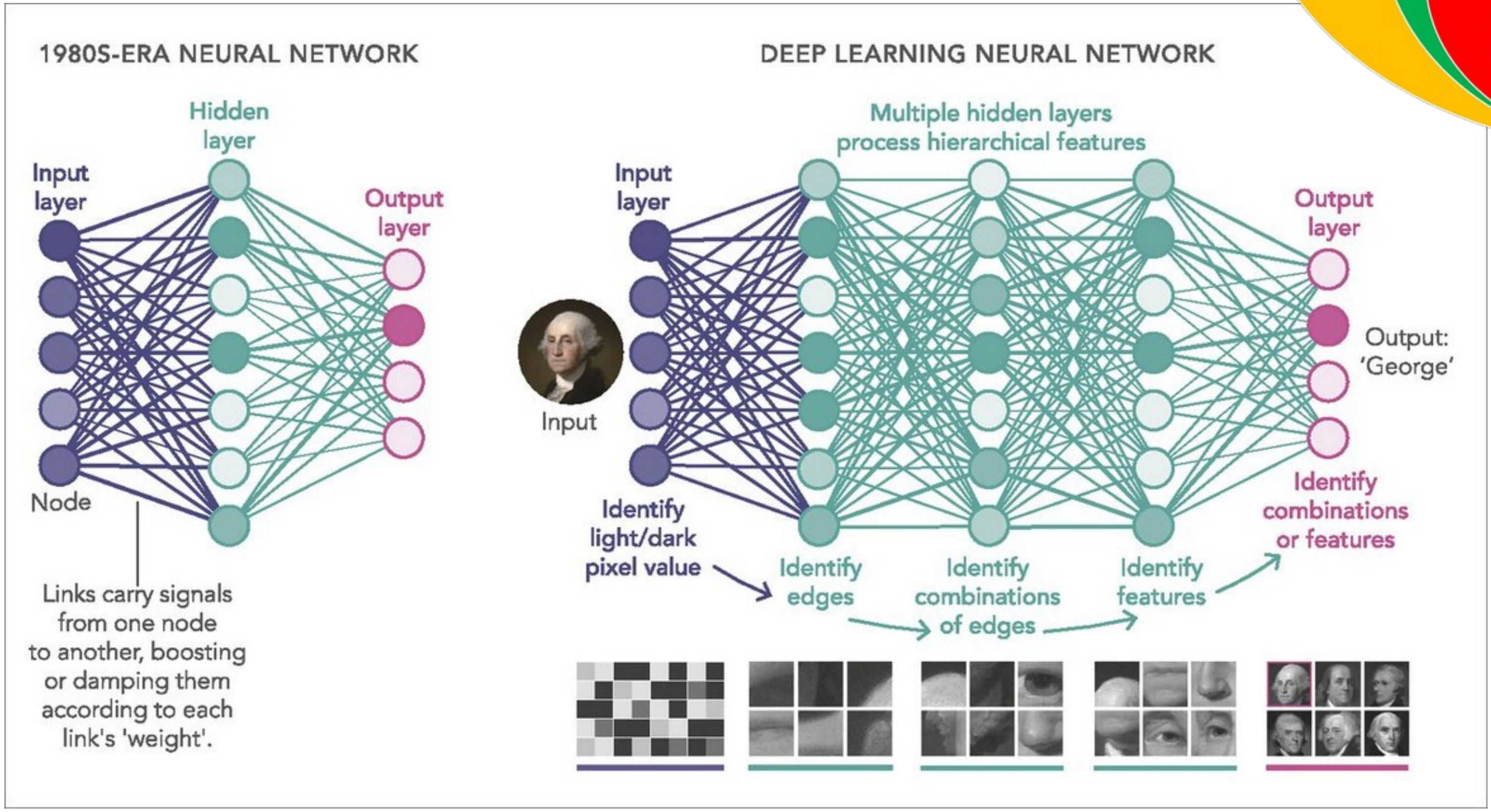
16 December 2019

Deep Neural Networks

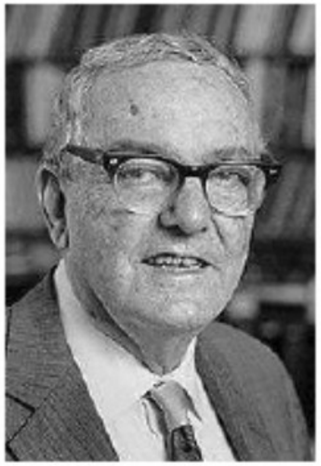
- How are the Deep Neural Networks built?
- What can we do with them?
- Some simple examples in python

All examples from this lecture are available on g
https://github.com/marcinwolter/DNN_examples




1980S-ERA NEURAL NETWORK
Hidden layer
Input layer
Output layer
Node
Links carry signals from one node to another, boosting or damping them according to each link's 'weight'.
DEEP LEARNING NEURAL NETWORK
Multiple hidden layers process hierarchical features
Input layer
Output layer
Input
Output: 'George'
Identify light/dark pixel value
Identify edges
Identify combinations of edges
Identify features
Identify combinations or features

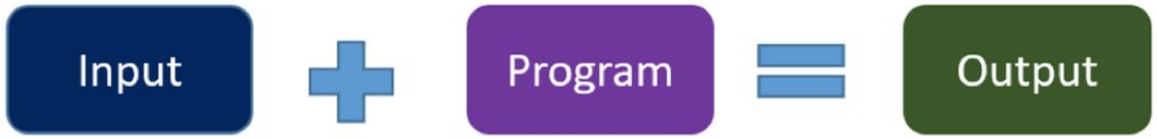
What is Machine Learning?



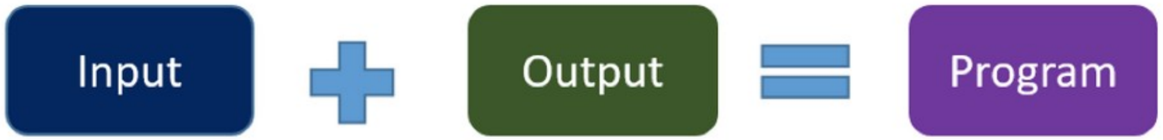
Herbert Simon
Turing Award 1975
Nobel Prize in Economics 1978

- **Herbert Alexander Simon**: "Learning is any process by which a system improves performance from experience."
- "Machine Learning is concerned with computer programs that automatically improve their performance through experience. "

Traditional approach



Machine Learning



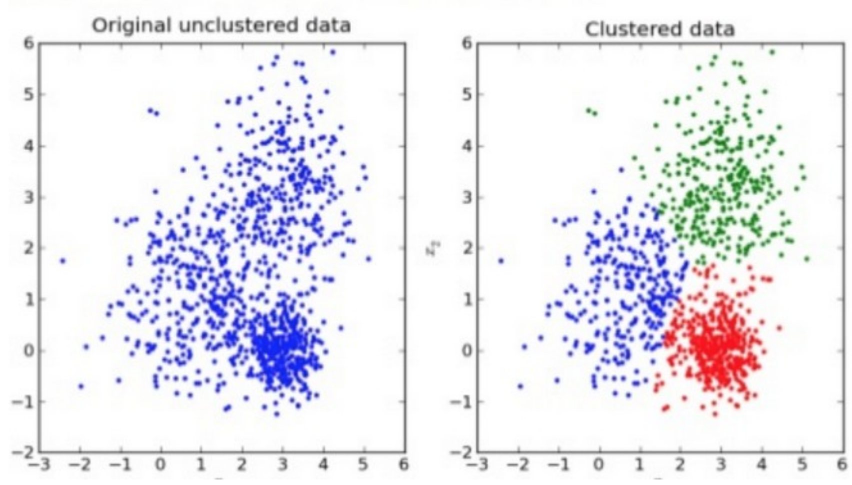
***Example**: Having input and the desired output (eg. calorimeter jet and it's energy) we can create a program (train our algorithm) to give us the desired output (the energies of other jets).*

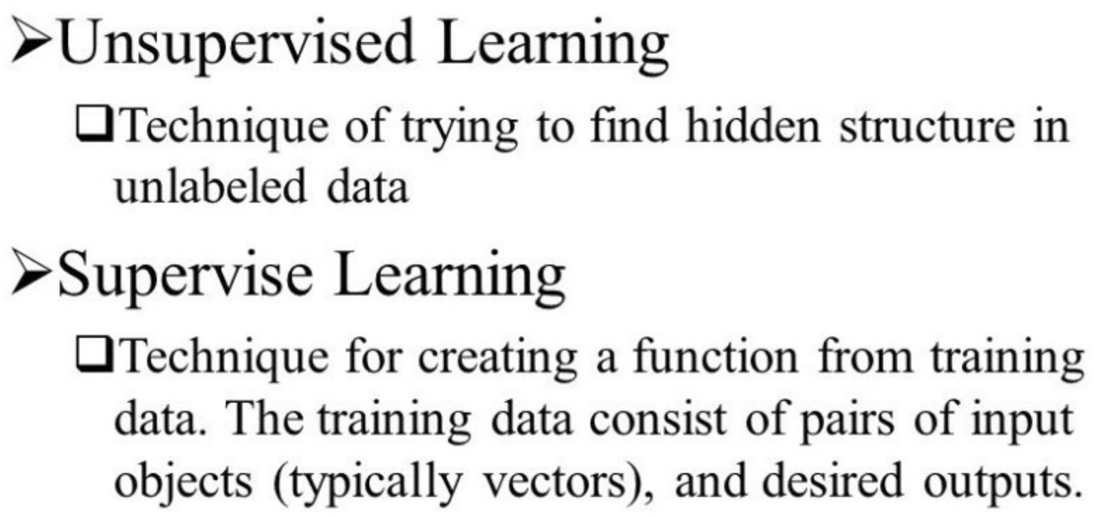
Tom Mitchell

Carnegie Mellon University

What does “machine learning” mean?

- **Machine learning** is a field of computer science that gives computer systems the ability to "learn" (i.e. progressively improve performance on a specific task) with data, without being explicitly programmed.
- **Problems to which it’s applied:**
  - Supervised learning (classification & regression)
  - Clustering (unsupervised learning)
  - Dimensionality reduction
  - Reinforcement learning
  - Many others….





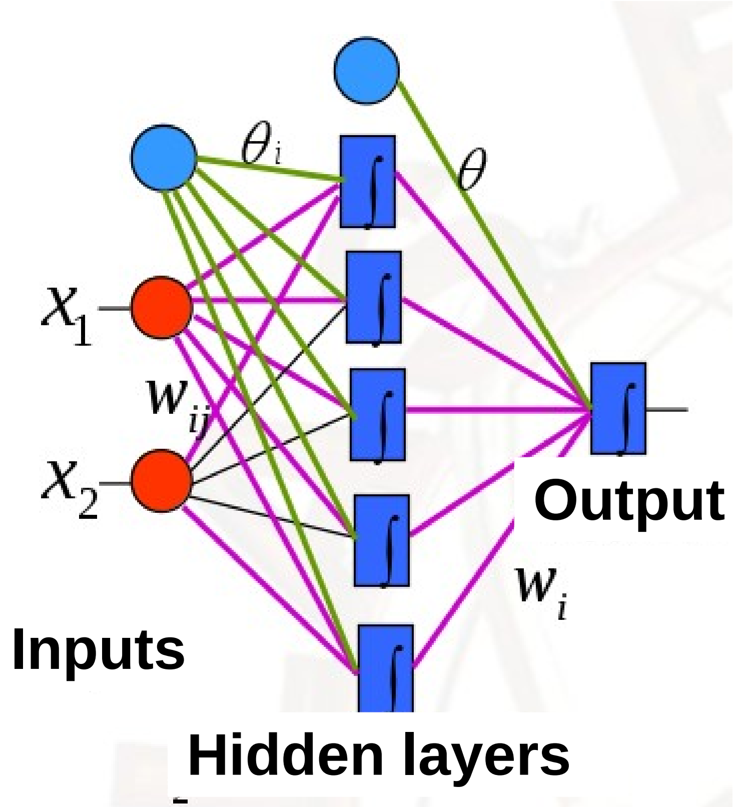
➢Unsupervised Learning

❑Technique of trying to find hidden structure in unlabeled data

➢Supervise Learning

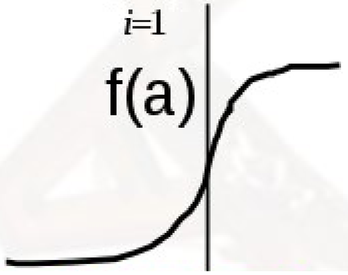
❑Technique for creating a function from training data. The training data consist of pairs of input objects (typically vectors), and desired outputs.

Neural networks



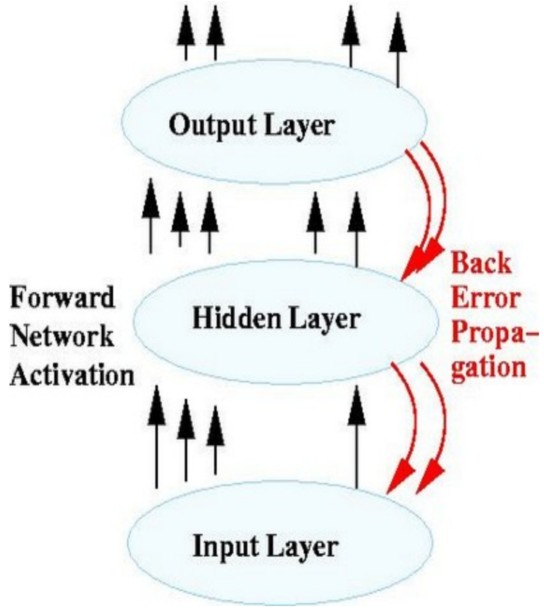
$$a_i = \sum_{j=1}^{2} w_{ij} x_j + \theta_i \quad \rightarrow f(a_i)$$

$$n(x,w) = f\left(\sum_{i=1}^{5} w_i f(a_i) + \theta\right)$$



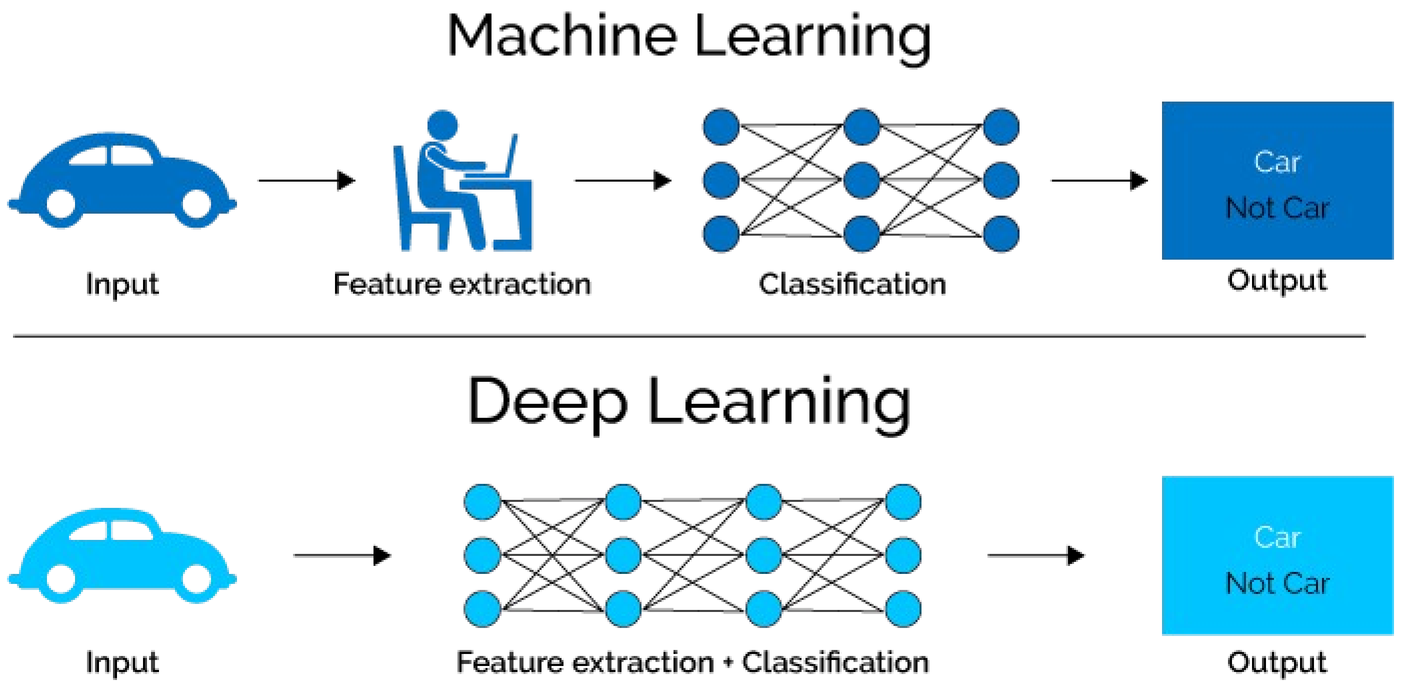
Activation function

- At the input of each node a weighted sum of inputs is given. It is transformed by the activation function (typically sigmoid) and later send to the output.
- How to train the multilayer network? How to tune the weights in the hidden layers? This problem was unsolved for a long time...
- Solution – **backpropagation**. An error y-*f(x,w)* is propagated backward through the net using the actual weights („revolution" of '80'ies).



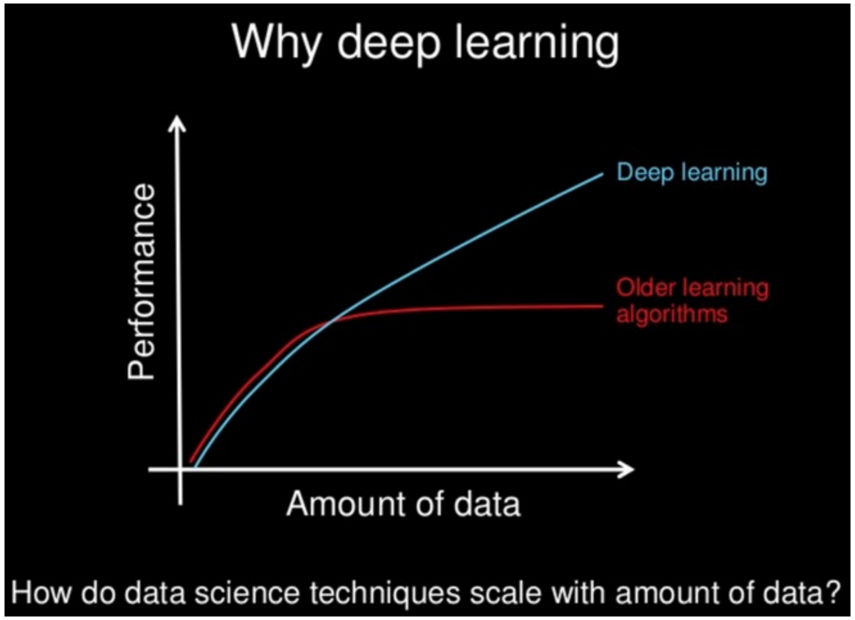
Machine Learning vs Deep Learning

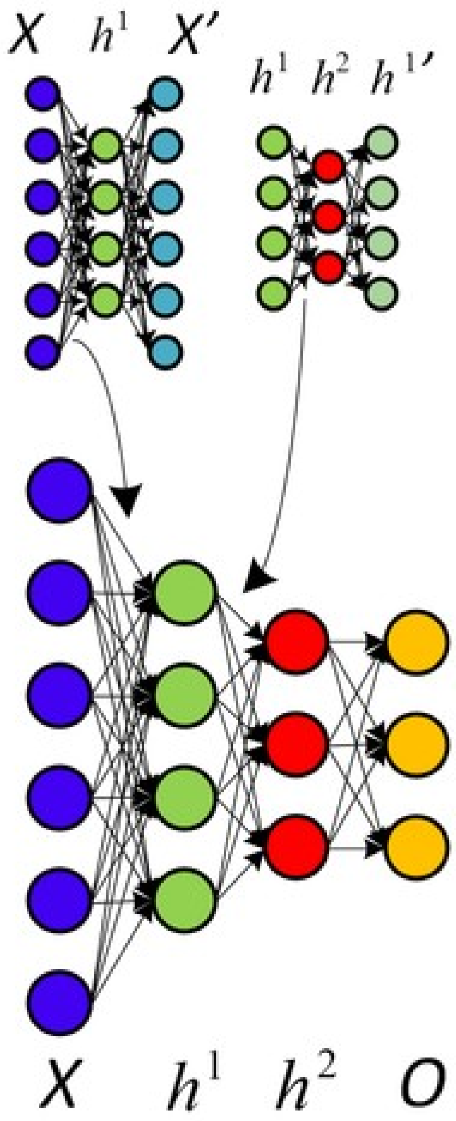
- **Traditional ML (BDT, NN etc)** – the scientist finds good, well discriminating variables (~10), called “features”, and performs classification using them as inputs for the ML algorithm.
- **Deep Learning** – thousands or millions of input variables (like pixels of a photo), the features are *automagically* extracted during training.



Deeper network?

Traditional Neural Networks have one or two hidden layers.
Deep Neural Network: a stack of sequentially trained **auto encoders**, which recognize different features (more complicated in each layer) and automatically prepare a new representation of data. This is how our brains are organized.
But how to train such a stack of layers?





http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning-part-4/

Training a Deep Neural Network

- In the early 2000s, attempts to train deep neural networks were frustrated by the apparent failure of the well known back-propagation algorithms (backpropagation, gradient descent). Many researchers claimed NN are gone, only Support Vector Machines and Boosted Decision Trees should be used!

- In 2006, Hinton, Osindero and Teh[1] first time **succeeded in training a deep neural network** by first initializing its parameters sequentially, layer by layer. Each layer was trained to produce a representation of its inputs that served as the training data for the next layer. Then the network was tweaked using gradient descent (standard algorithm).

- **There was a common belief that Deep NN training requires careful initialization of parameters and sophisticated machine learning algorithms.**

[1]Hinton, G. E., Osindero, S. and Teh, Y., A fast learning algorithm for deep belief nets, Neural Computation 18, 1527-1554.

Training with a brute force

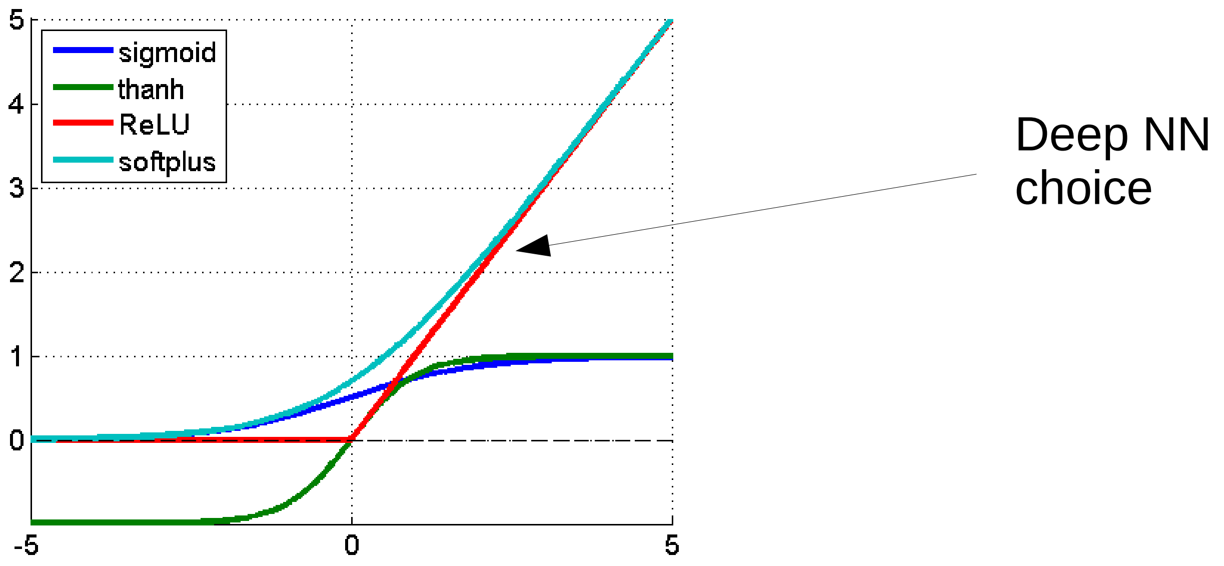
- In 2010, a surprising counter example to the conventional wisdom was demonstrated[1].
- Deep neural network was trained to classify the handwritten digits in the MNIST[2] data set, which comprises 60,000 28 × 28 = 784 pixel images for training and 10,000 images for testing.
- They showed that a plain DNN with architecture (784, 2500, 2000, 1500, 1000, 500, 10 – **HUGE!!!**), trained using standard stochastic gradient descent (Minuit on steroids!), outperformed all other methods that had been applied to the MNIST data set as of 2010. The error rate of this ∼12 million parameter DNN was 35 images out of 10,000.

  The training images were randomly and slightly deformed before every training epoch. The entire set of 60,000 undeformed images could be used as the validation set during training, since none were used as training data.

[1] Cireşan DC, Meier U, Gambardella LM, Schmidhuber J. ‚Deep, big, simple neural nets for handwritten digit recognition. Neural Comput. 2010 Dec; 22 (12): 3207-20.
[2] http://yann.lecun.com/exdb/mnist/

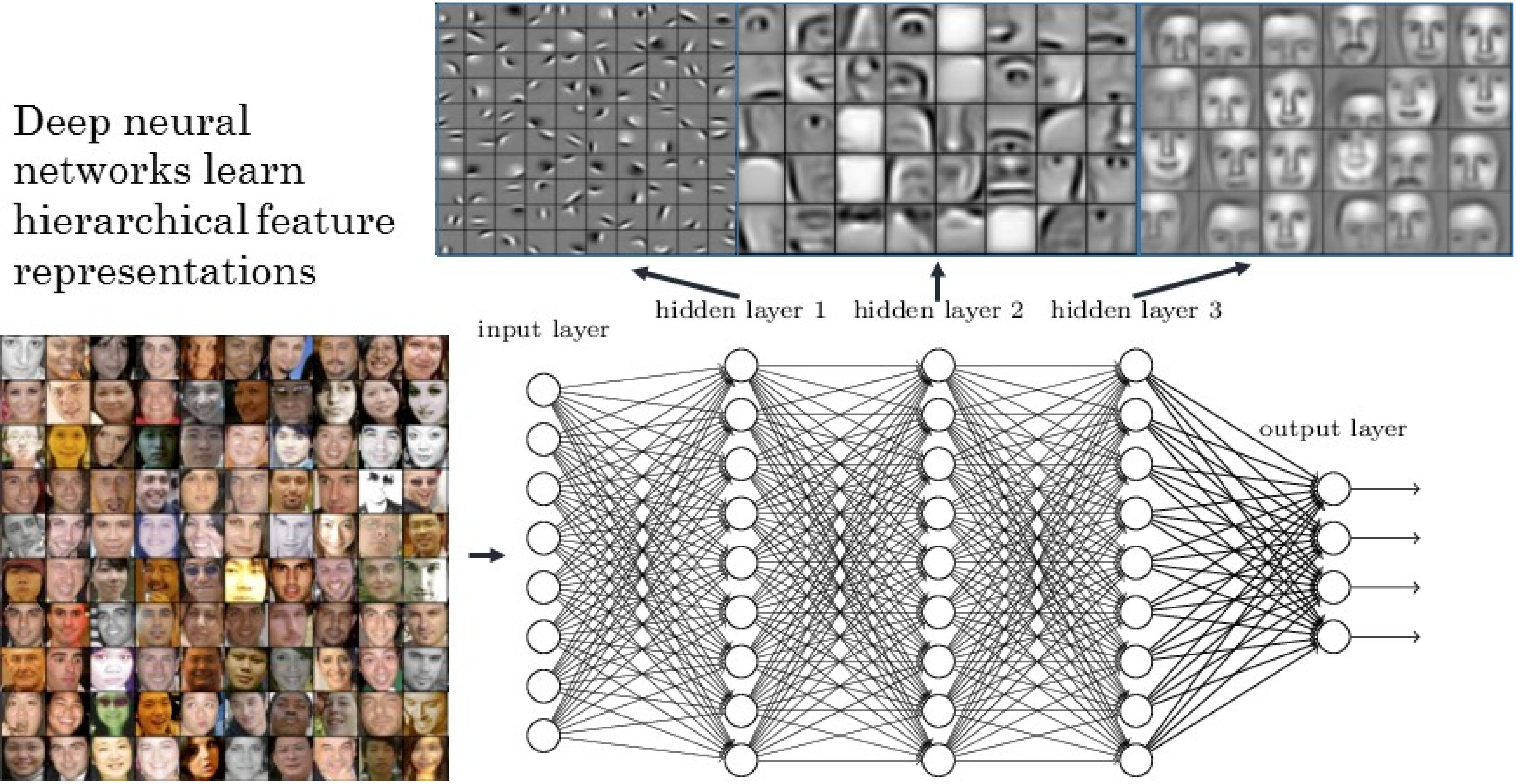
Why it didn't work before?

- More data, clusters of GPU/CPU (computing power!)
- The particular non-linear activation function chosen for neurons in a neural net makes a big impact on performance, and the one often used by default is not a good choice.
- The old vanishing gradient problem happens, basically, because backpropagation involves a sequence of multiplications that invariably result in smaller derivatives for earlier layers: that is, unless weights are chosen with different scales according to the layer they are in! Making this simple change results in significant improvements.



DNN – hierarchical feature extraction

Deep neural networks learn hierarchical feature representations



Hand-writing recognition

MNIST database of hand-written digits.

- With small and very simple MLP after a short training we got an **accuracy of 98.3%**
- Network has just two hidden layers - very simple!

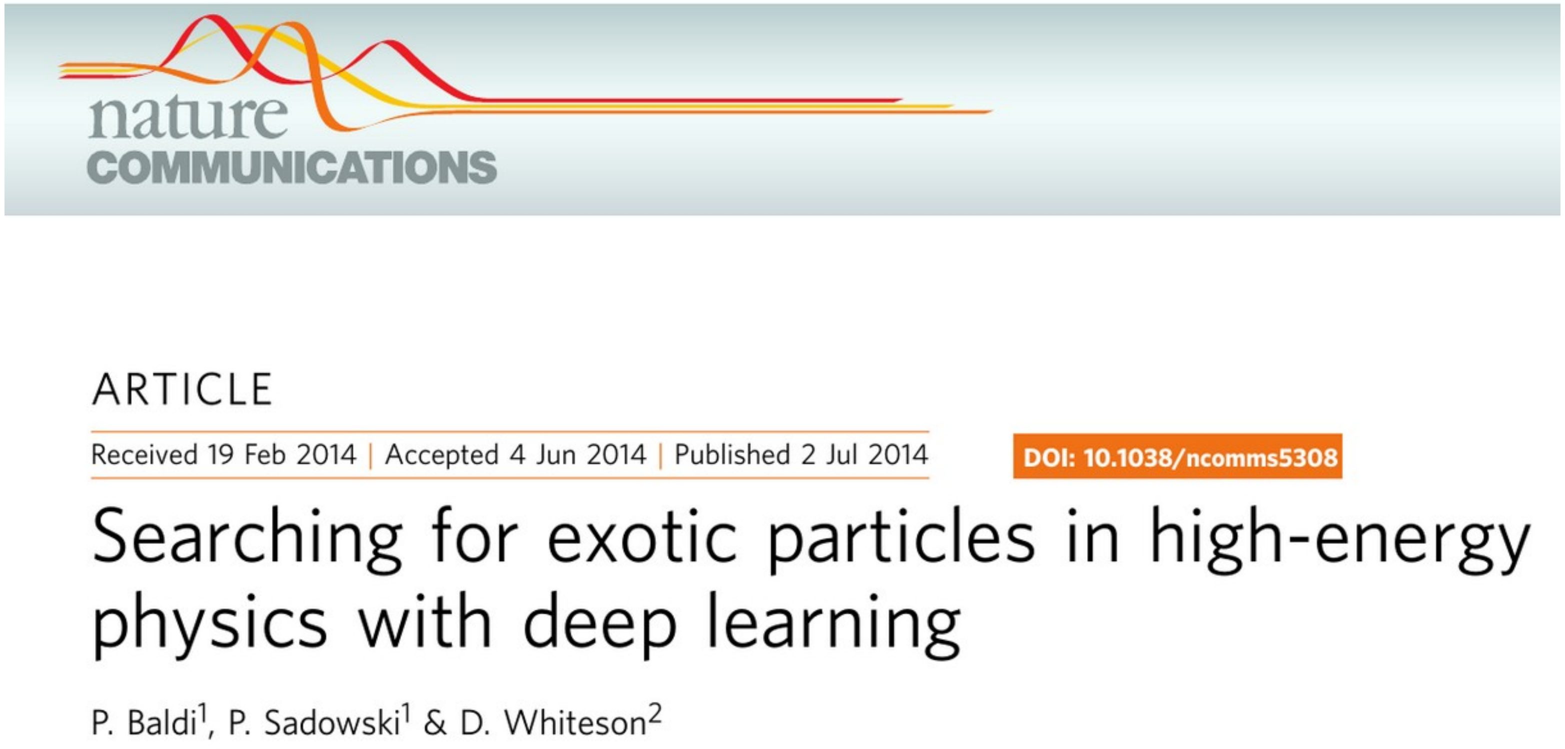
```
60000 train samples
10000 test samples
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_8 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_9 (Dense)              (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
```

Example:
https://github.com/marcinwolter/DNN_examples/blob/master/mnist_mlp.ipynb

Slightly updated example from net.

Automated physicist?

- Use of deep neural network to extract H->ττ and SUSY signals in simulated CMS data[1,2].



nature COMMUNICATIONS

ARTICLE

Received 19 Feb 2014 | Accepted 4 Jun 2014 | Published 2 Jul 2014 DOI: 10.1038/ncomms5308

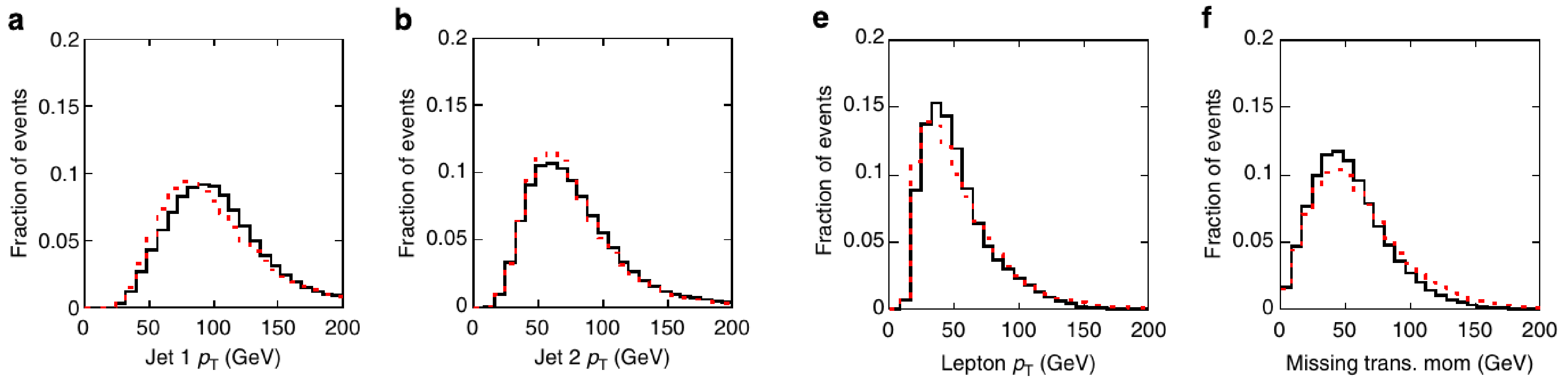
Searching for exotic particles in high-energy physics with deep learning

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

[1] P. Baldi, P. Sadowski, D. Whiteson, Searching for Exotic Particles in High-Energy Physics with Deep Learning, Nature Commun. 5 (2014) 4308
[2] P. Baldi, P. Sadowski, D. Whiteson, Enhanced Higgs Boson to ττ Search with Deep Learning, PRL 114, 111801 (2015)

H->τ τ searches

- Low-level variables – 22, here just few are shown



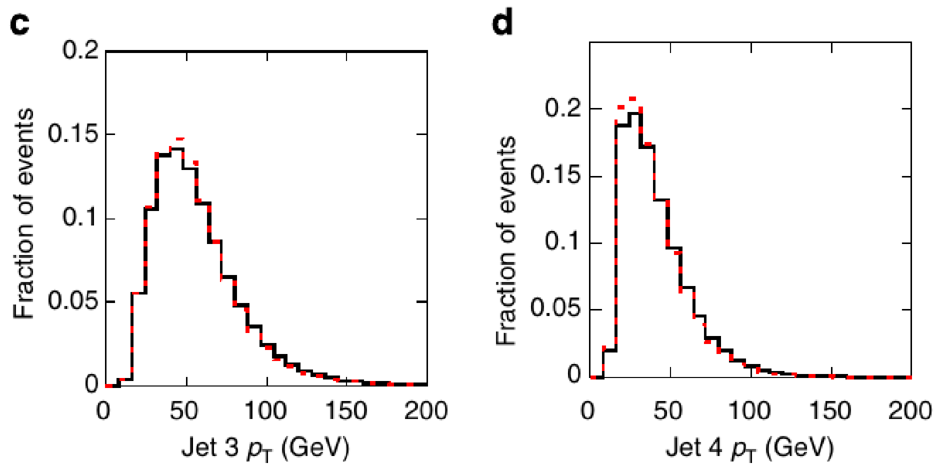


Figure 2 | Low-level input features for Higgs benchmark. Distributions in $\ell\nu jjb\bar{b}$ events for simulated signal (black) and background (red) benchmark events. Shown are the distributions of transverse momenta (p_T) of each observed particle (**a**–**e**) as well as the imbalance of momentum 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.

Higgs search

- Physicists have built 7 well discriminating high-level variables (derived from 22 low level variables).

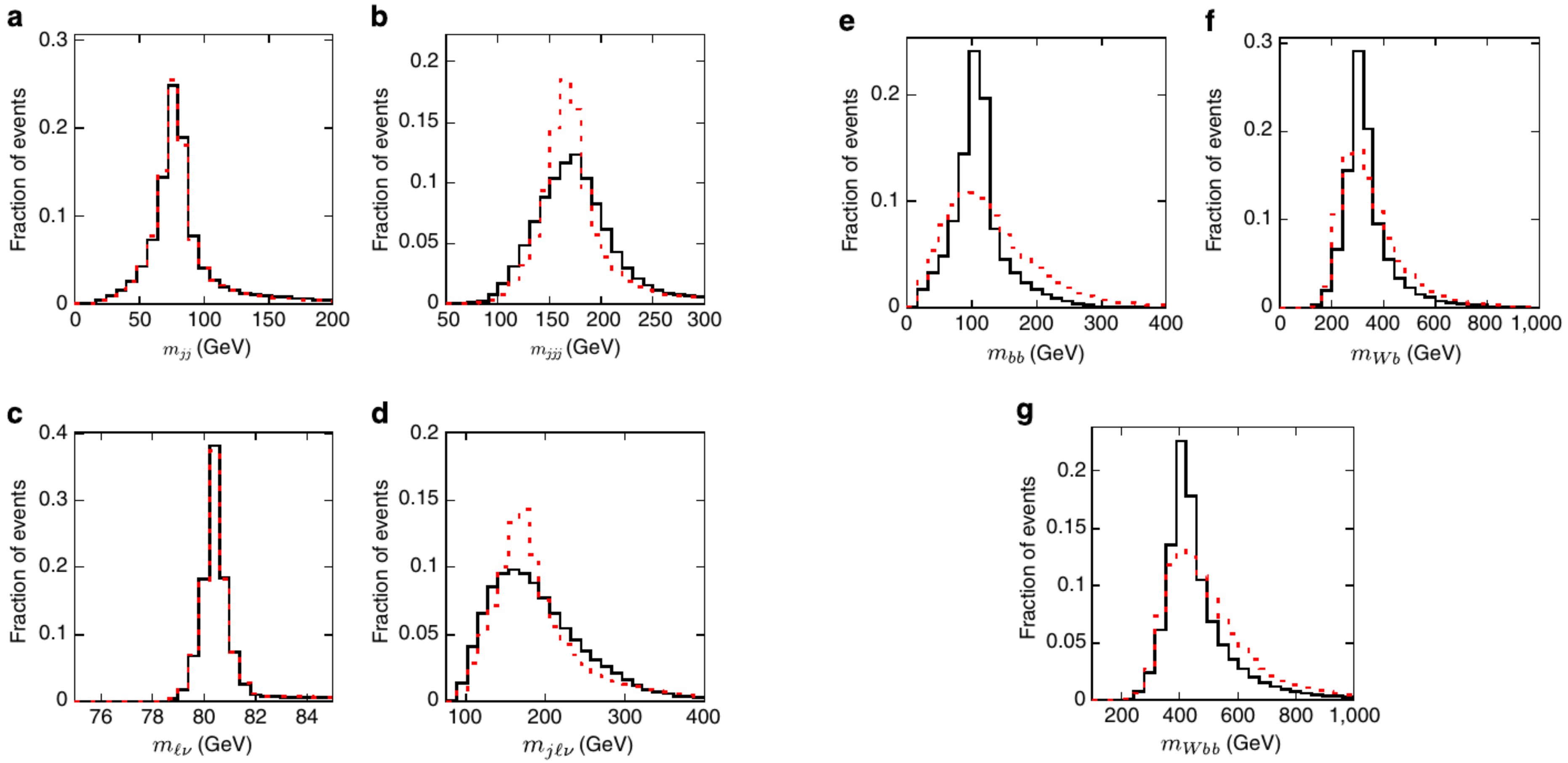


Figure 3 | High-level input features for Higgs benchmark. Distributions in simulation of invariant mass calculations in $\ell\nu jjb\bar{b}$ events for simulated signal (black) and background (red) events.

Higgs searches

Low level – 22 variables
High level – 7 variables

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

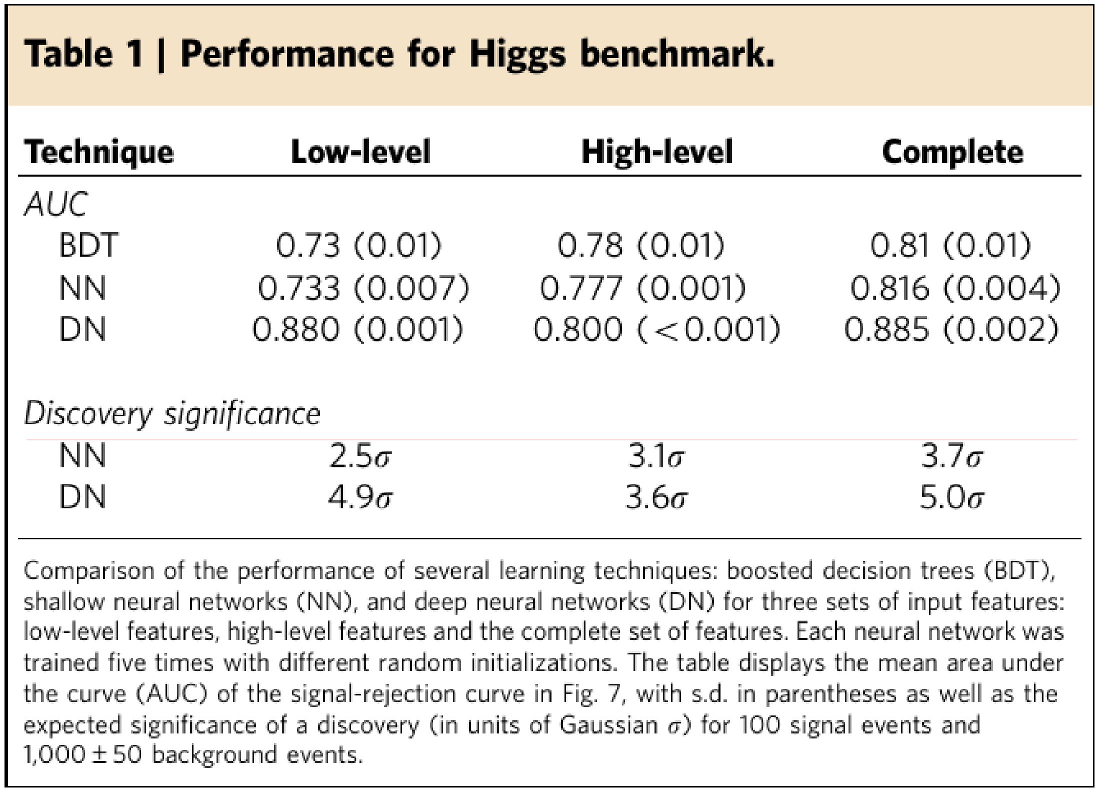
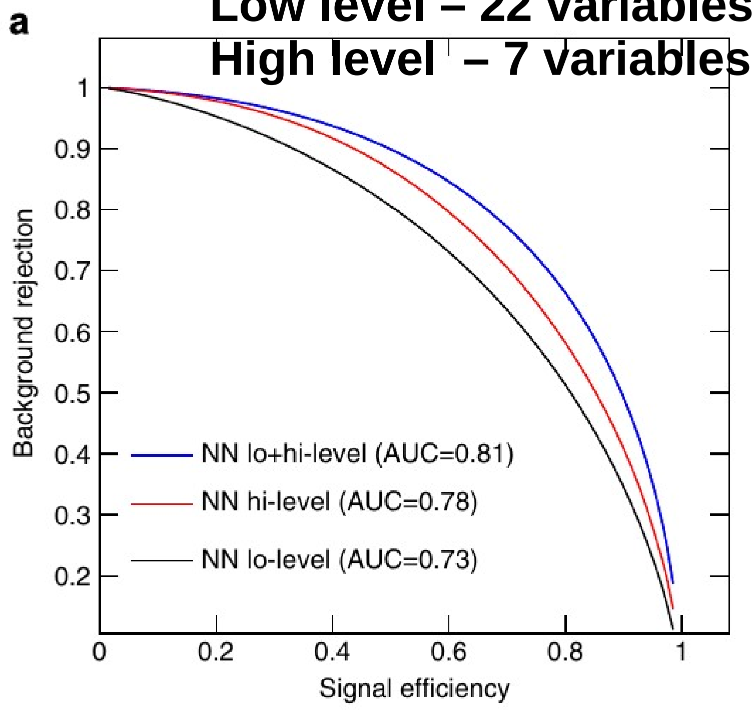
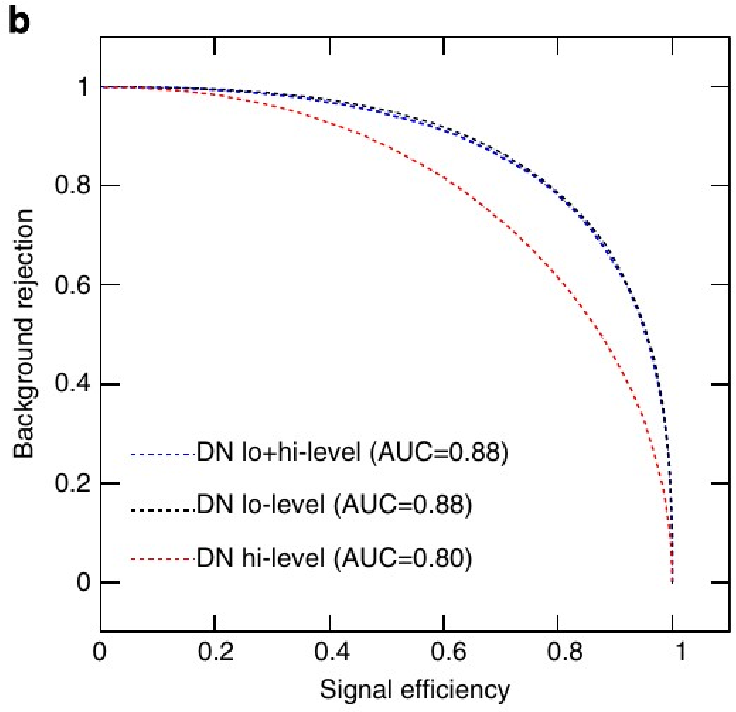


Table 1 | Performance for Higgs benchmark.

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

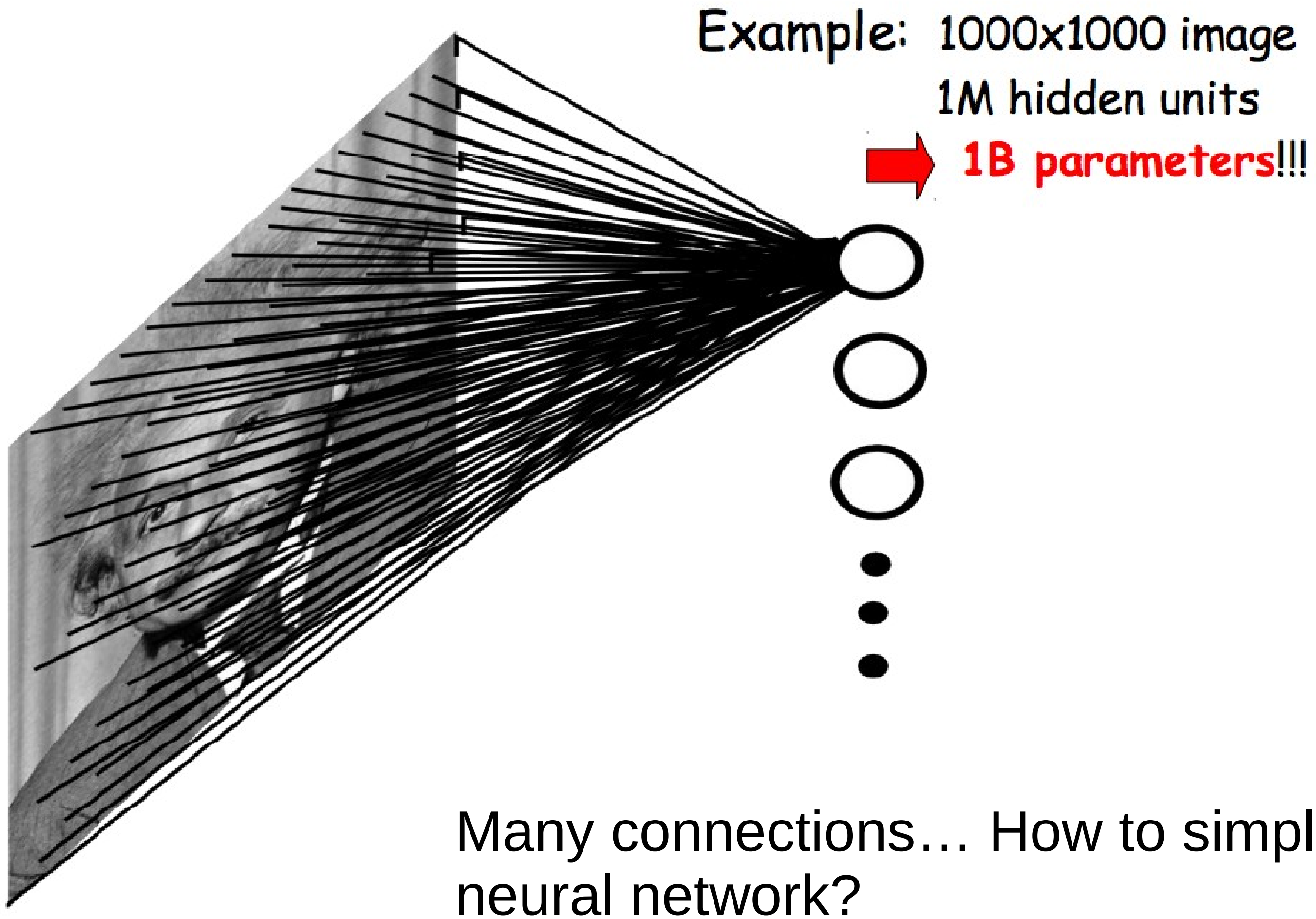
Comparison of the performance of several learning techniques: boosted decision trees (BDT), shallow neural networks (NN), and deep neural networks (DN) for three sets of input features: low-level features, high-level features and the complete set of features. Each neural network was trained five times 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 $1,000 \pm 50$ background events.





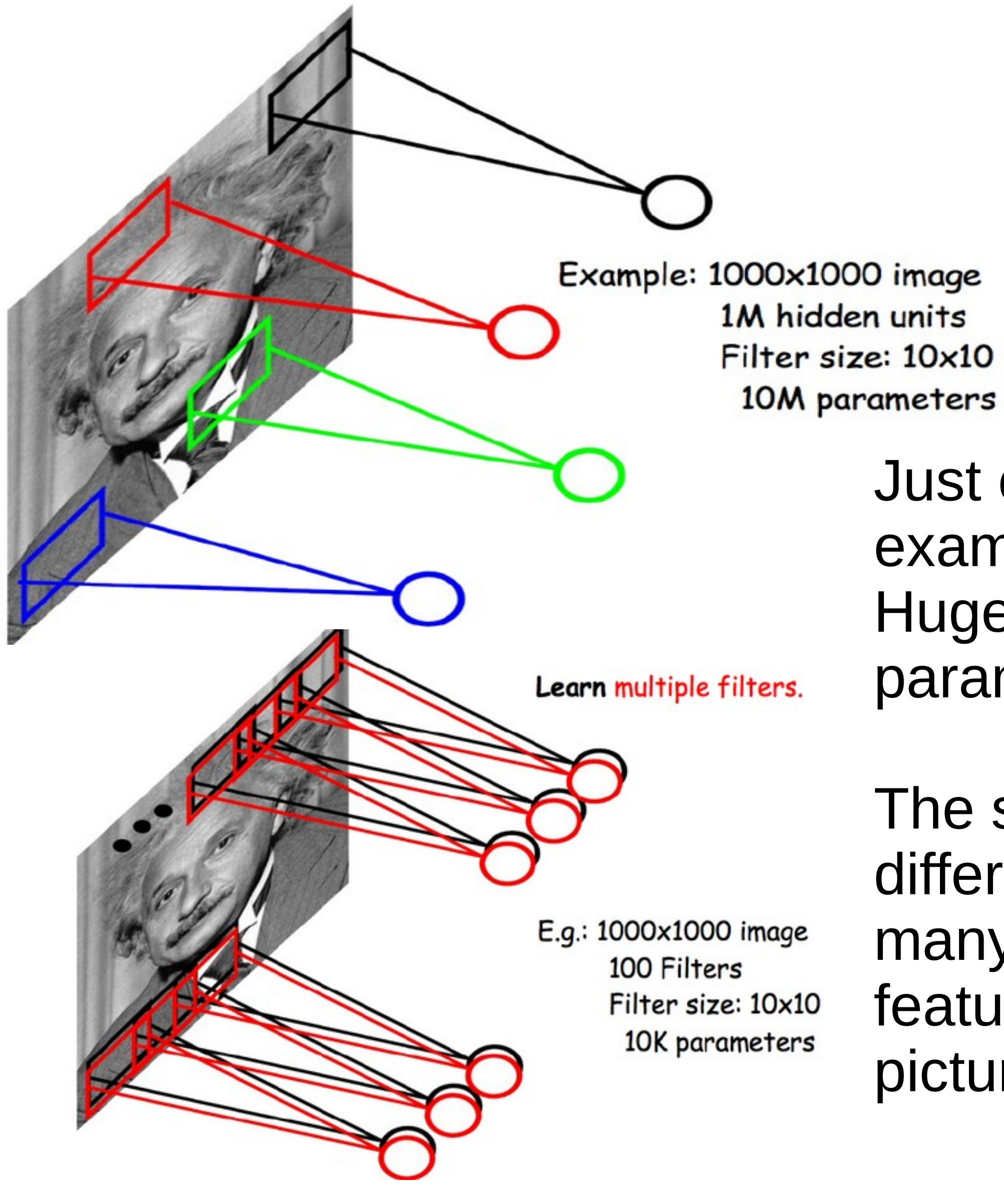
doi:10.1038/ncomms5308

Convolutional Neural Network for 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!

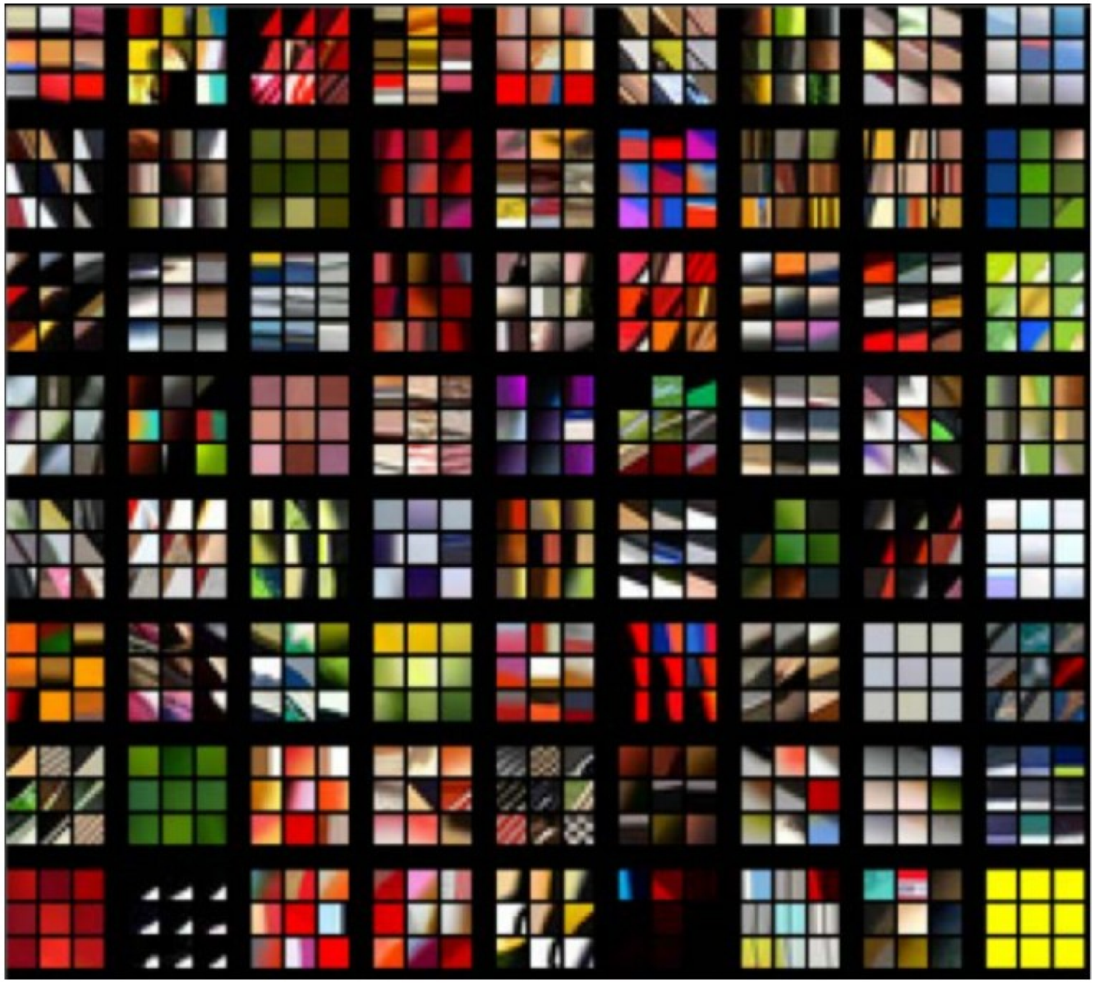
The same features might be found in different places => so we could train many filters, each recognizing another feature, and move them over the picture.

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

Example of pattern recognition

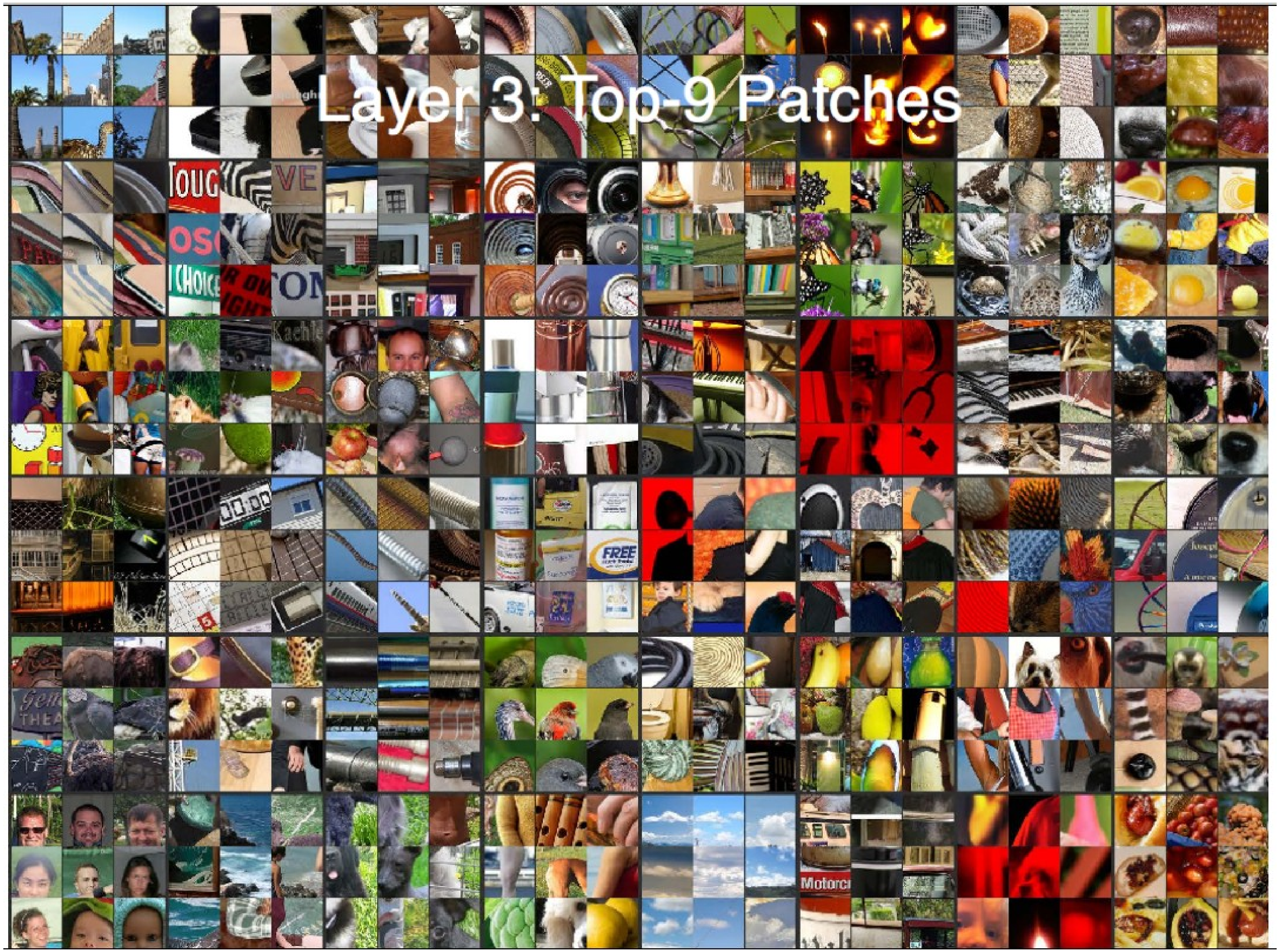
Top 9 patches that activate each filter in layer 1

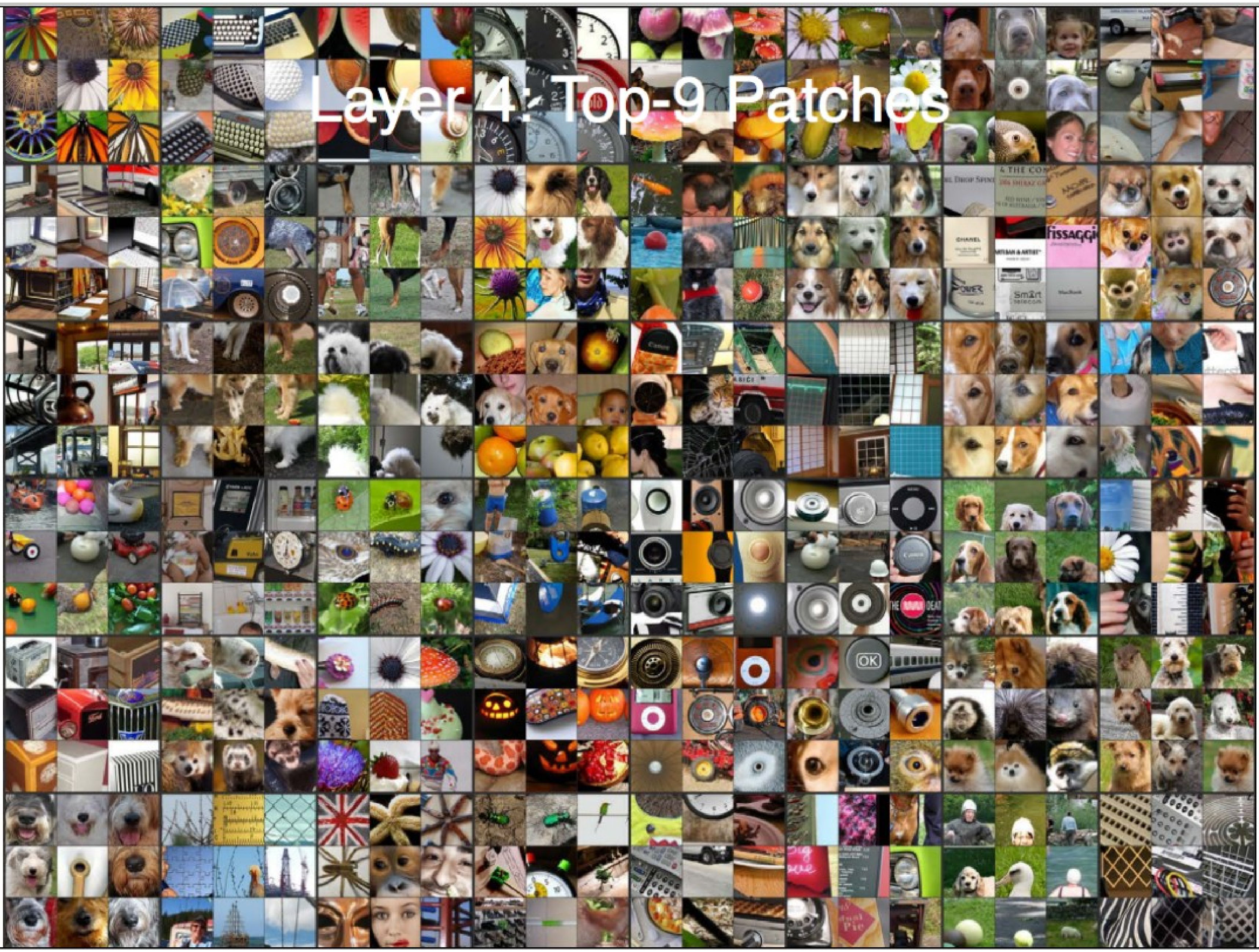
Each 3x3 block shows the top 9 patches for one filter.



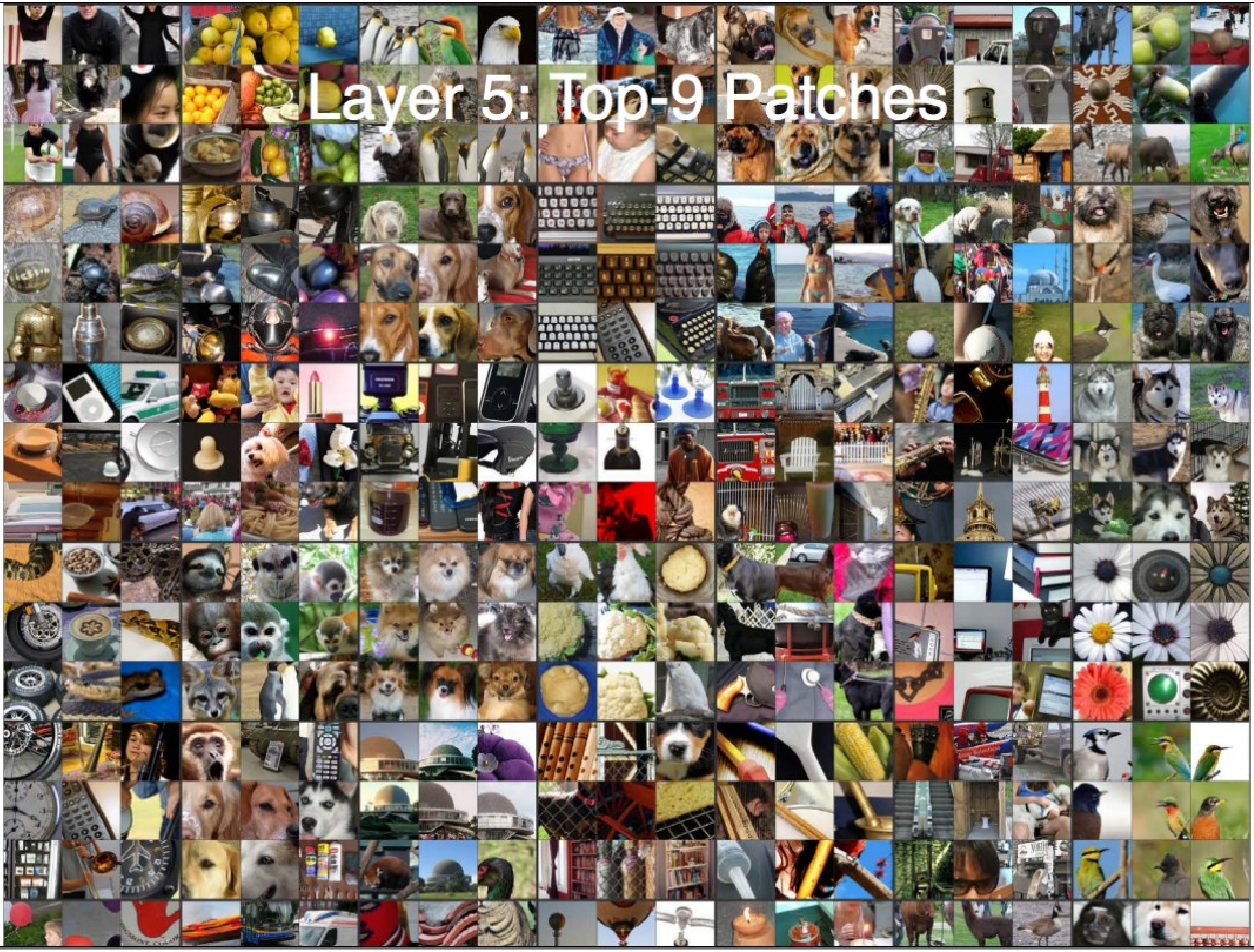
Layer 2: Top-9 Patches

Layer 3: Top-9 Patches

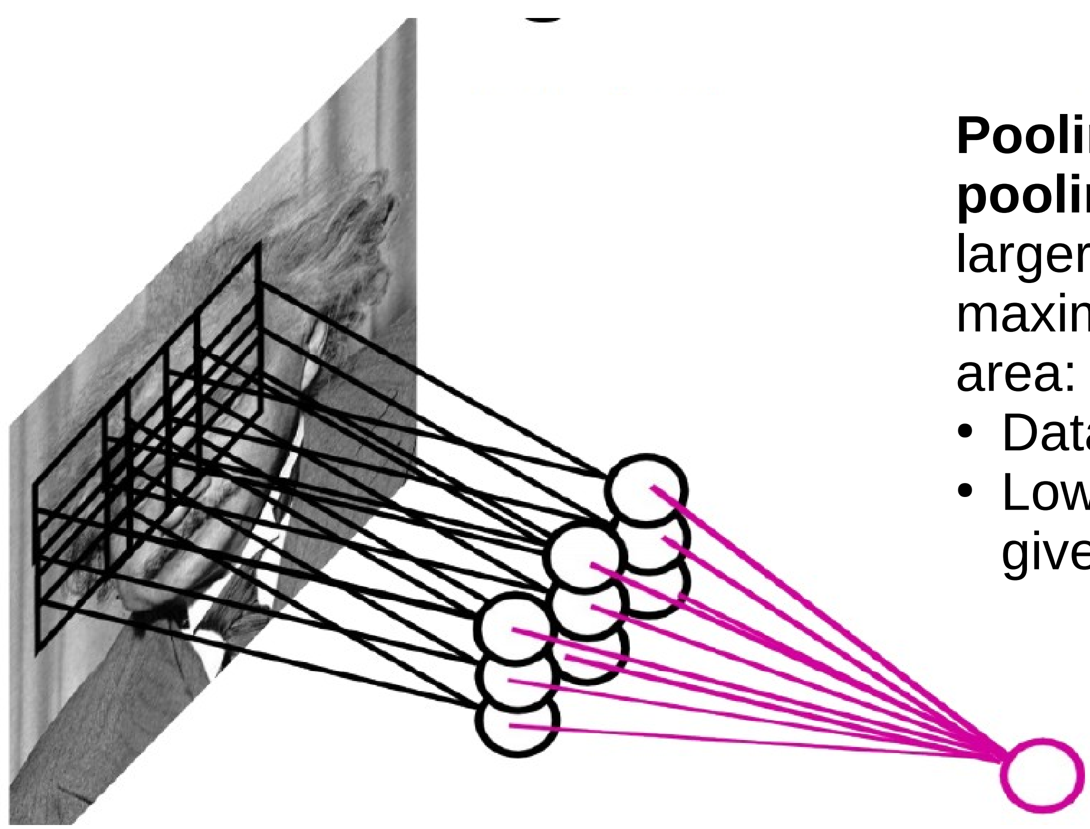



Layer 4: Top-9 Patches

Layer 5: Top-9 Patches

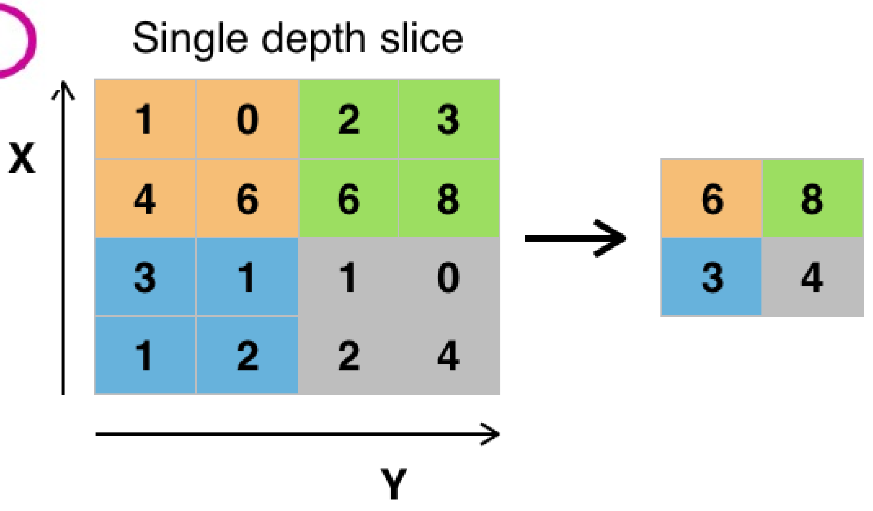


Pooling



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:

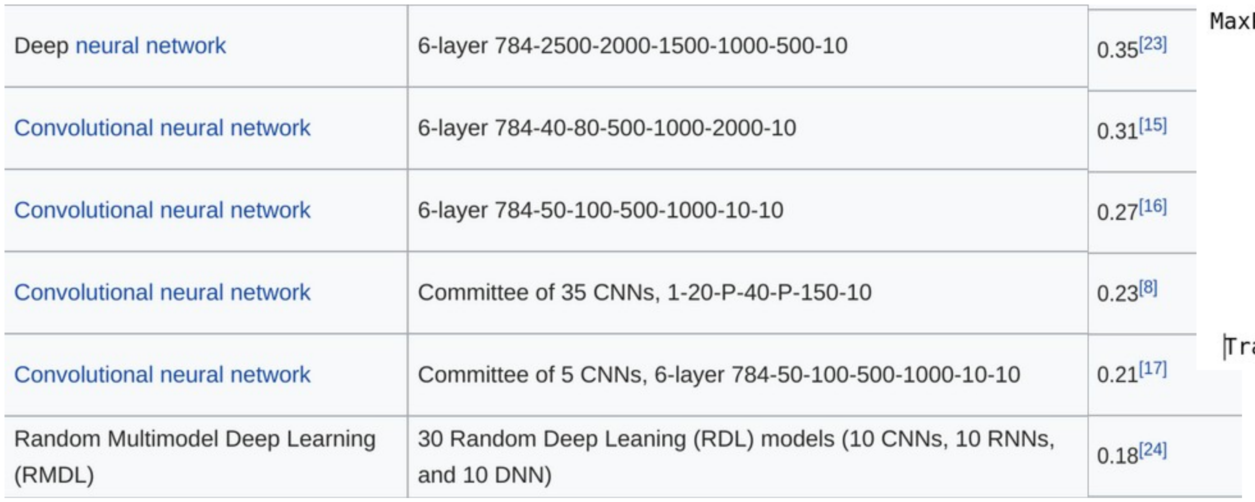
- Data reduction,
- Lower sensitivity for the position of a given feature.



Quick test of hand-writing recognition

MNIST database of hand-written digits.

Best results (Wikipedia):

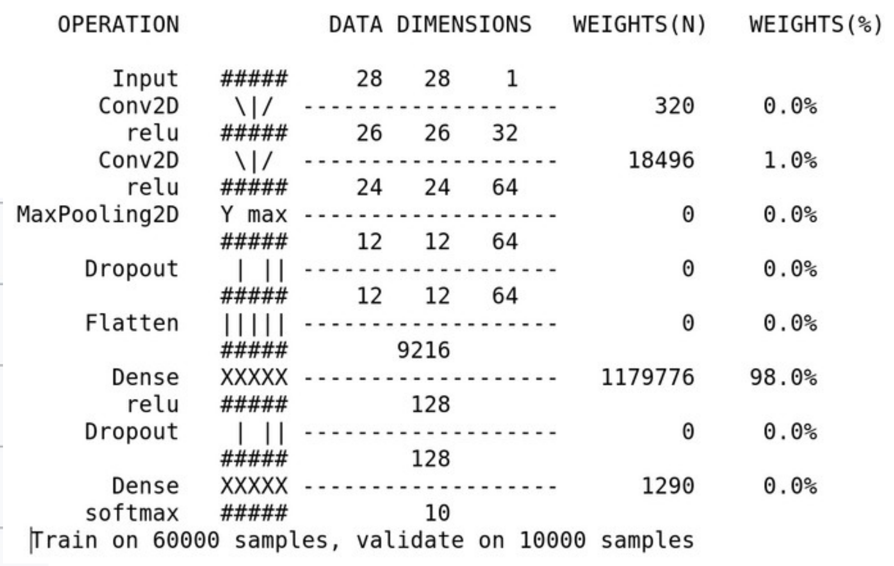


Deep neural network	6-layer 784-2500-2000-1500-1000-500-10	0.35[23]
Convolutional neural network	6-layer 784-40-80-500-1000-2000-10	0.31[15]
Convolutional neural network	6-layer 784-50-100-500-1000-10-10	0.27[16]
Convolutional neural network	Committee of 35 CNNs, 1-20-P-40-P-150-10	0.23[8]
Convolutional neural network	Committee of 5 CNNs, 6-layer 784-50-100-500-1000-10-10	0.21[17]
Random Multimodel Deep Learning (RMDL)	30 Random Deep Leaning (RDL) models (10 CNNs, 10 RNNs, and 10 DNN)	0.18[24]

- With **CNN** we got after a short training an ***accuracy of 99.21%***
- Our CNN neural net (small comparing to the winning nets):

  https://github.com/marcinwolter/DNN_examples/blob/master/mnist_cnn.ipynb

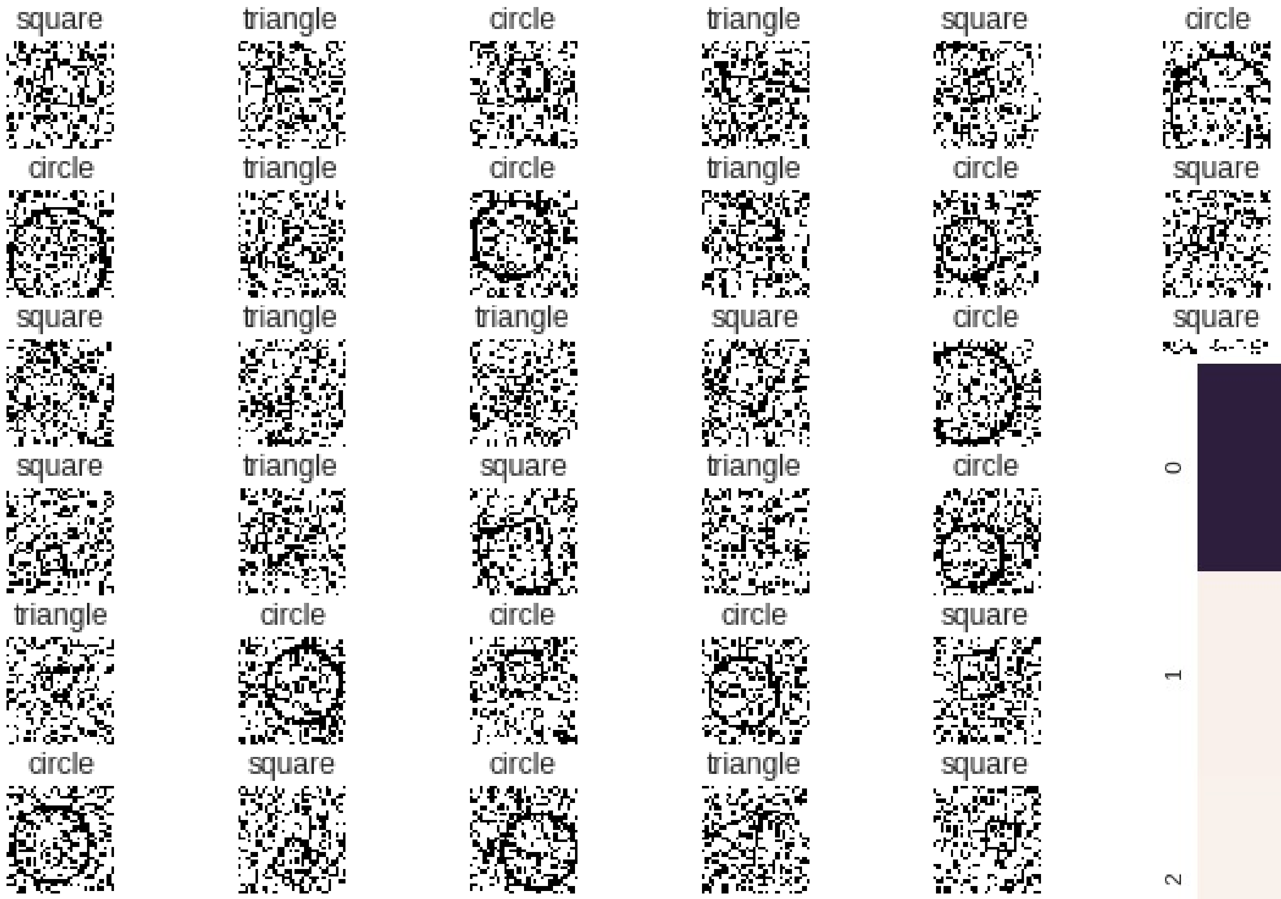
  Slightly updated example from net.

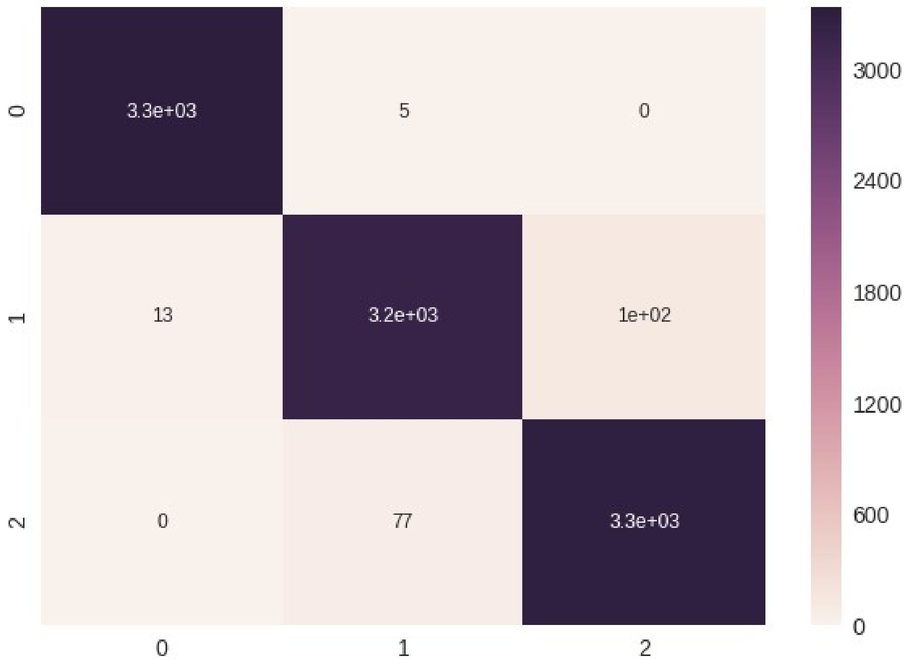


```
     OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

         Input   #####     28   28    1
        Conv2D    \|/  -------------------       320     0.0%
          relu   #####     26   26   32
        Conv2D    \|/  -------------------     18496     1.0%
          relu   #####     24   24   64
  MaxPooling2D   Y max -------------------         0     0.0%
                 #####     12   12   64
       Dropout    | || -------------------         0     0.0%
                 #####     12   12   64
       Flatten   ||||| -------------------         0     0.0%
                 #####        9216
         Dense   XXXXX -------------------   1179776    98.0%
          relu   #####         128
       Dropout    | || -------------------         0     0.0%
                 #####         128
         Dense   XXXXX -------------------      1290     0.0%
       softmax   #####          10
 Train on 60000 samples, validate on 10000 samples
```

Figure recognition





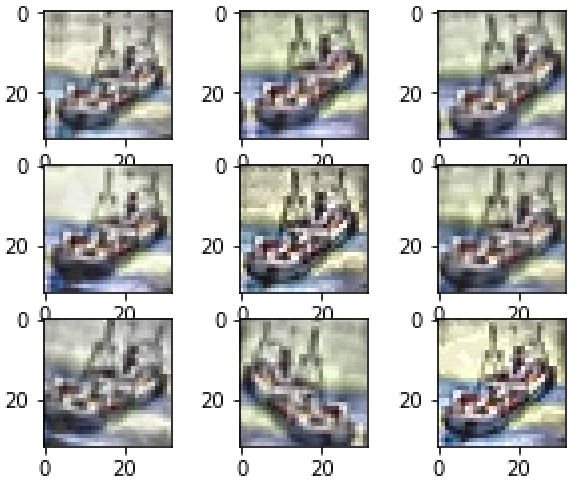
- Qualification project for IFJ PAN Ph.D. students
- https://github.com/marcinwolter/DNN_examples/blob/master/figure_cnn.ipynb

CIFAR image classification Bonus example

- Classifies images into 10 classes

  Very nice and advanced example, uses **data augmentation** (artificial replication)

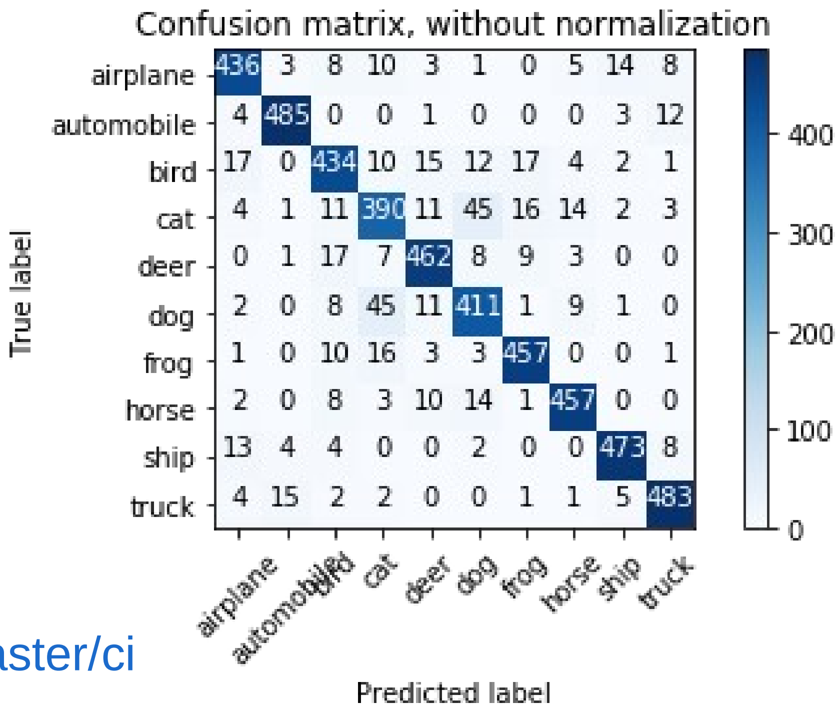
  runs on google Graphic Processor Units (GPU)



Example: cifar_classifier_gpu_aug.ipynb

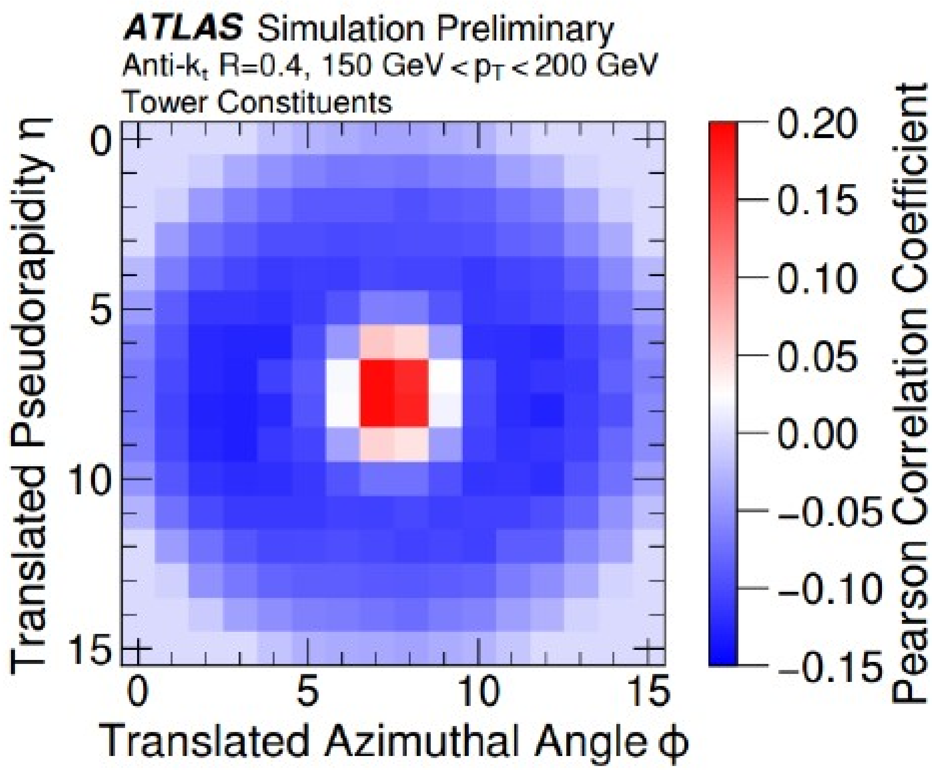
Source from net:
https://colab.research.google.com/drive/1zTYNJ3xtPeNsa5ARBw4Ufj8crPZJx-cp

https://github.com/marcinwolter/DNN_examples/blob/master/cifar_classifier_gpu_aug.ipynb

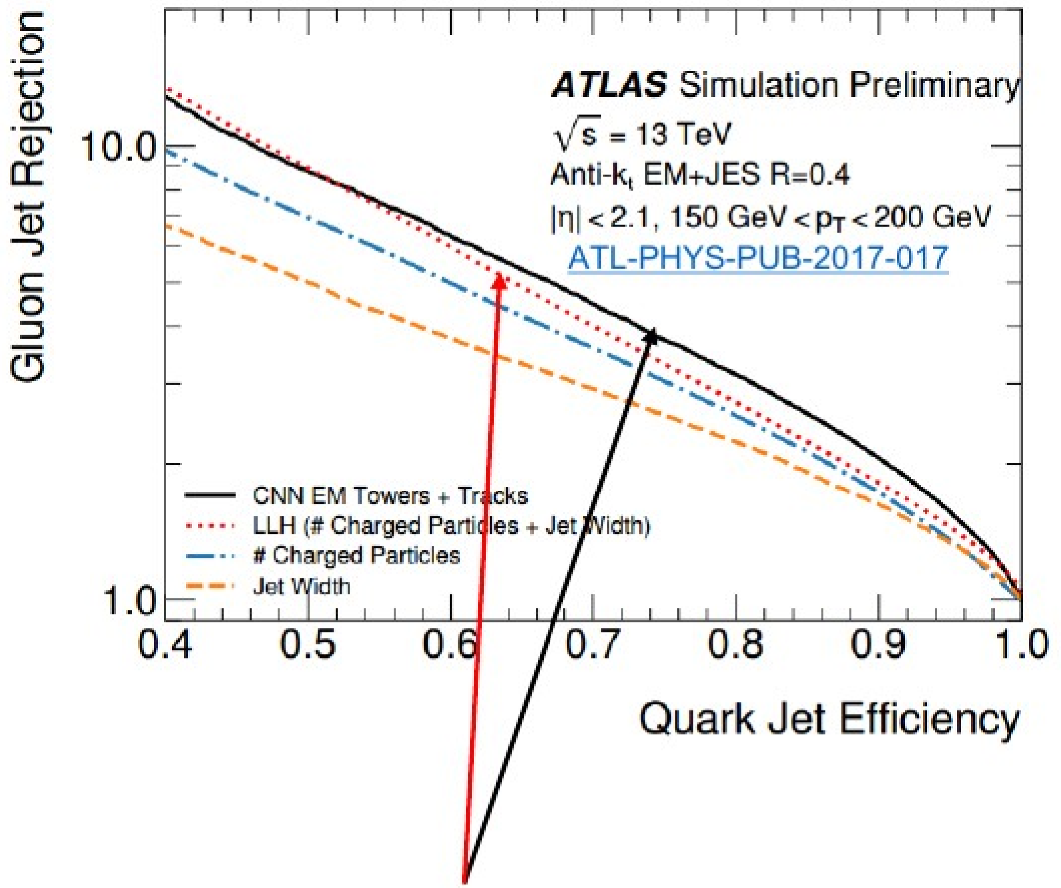


	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	436	3	8	10	3	1	0	5	14	8
automobile	4	485	0	0	1	0	0	0	3	12
bird	17	0	434	10	15	12	17	4	2	1
cat	4	1	11	390	11	45	16	14	2	3
deer	0	1	17	7	462	8	9	3	0	0
dog	2	0	8	45	11	411	1	9	1	0
frog	1	0	10	16	3	3	457	0	0	1
horse	2	0	8	3	10	14	1	457	0	0
ship	13	4	4	0	0	2	0	0	473	8
truck	4	15	2	2	0	0	1	1	5	483

CNNs – quark-gluon jet classification



ATL-PHYS-PUB-2017-017



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

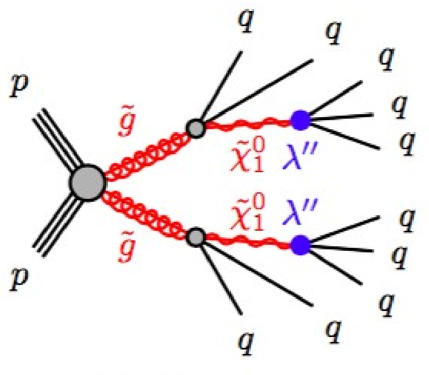
“The image-based approach described in this note is a promising avenue for future research to confront a variety of tagging challenges.”

Example of CNN effort

ATLAS search for R-parity-violating SUSY gluino decays

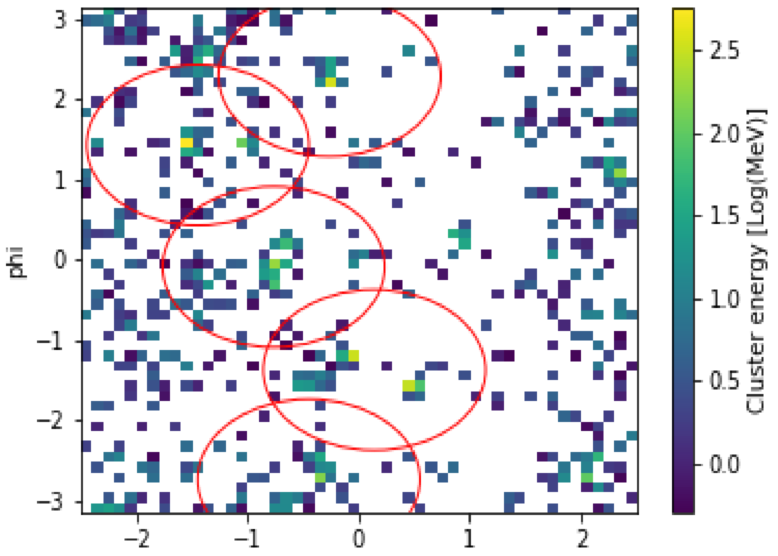
Analysis from ATLAS-CONF-2016-057 used as a benchmark

From ATLAS-CONF-2016-057:



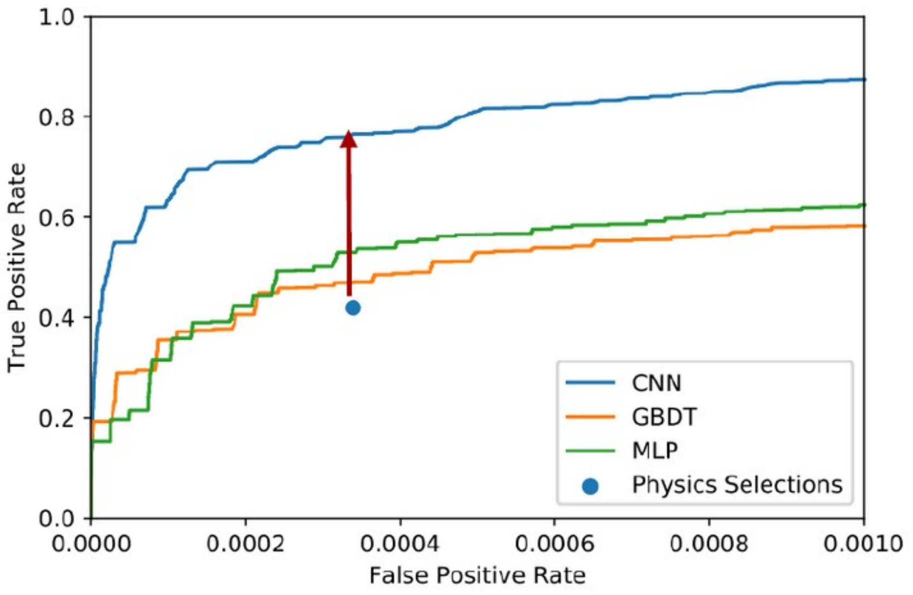
p
p
$\tilde{g}$
$\tilde{g}$
$\tilde{\chi}_1^0$ λ''
$\tilde{\chi}_1^0$ λ''
q

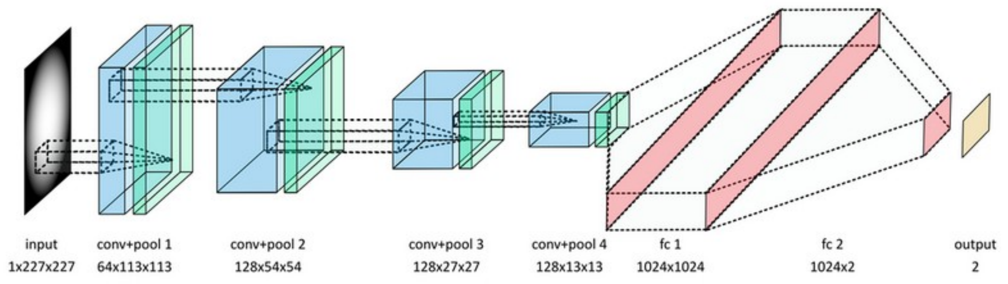
(b) gluino cascade decay



AntiKt R=1.0 trimmed algorithm used in the standard analysis.

CNN outperforms other ML classifiers

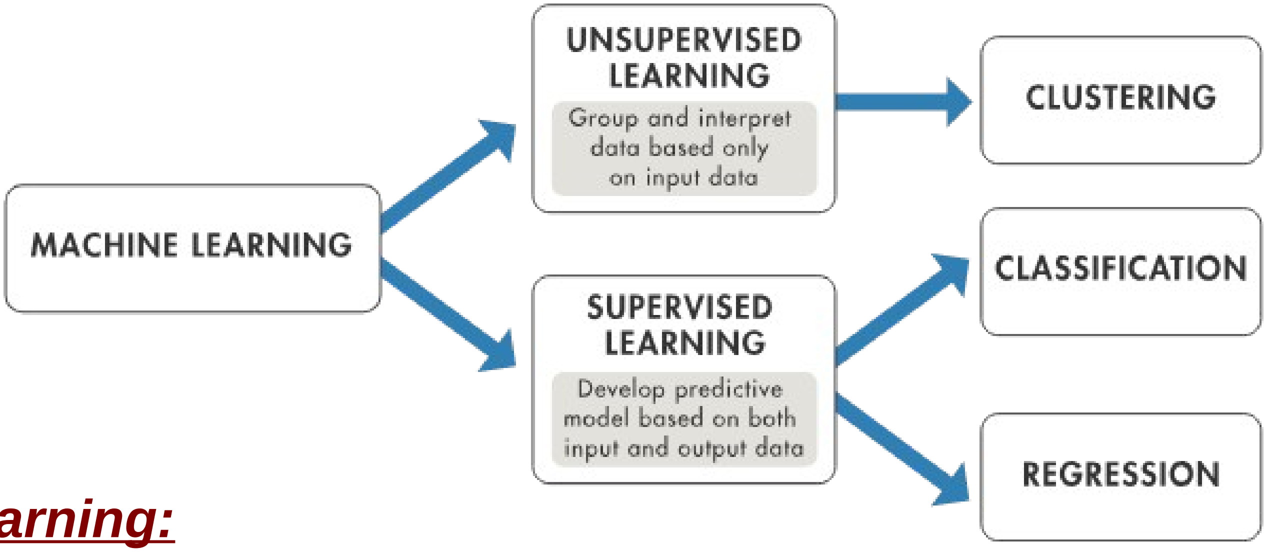




https://indico.cern.ch/event/567550/contributions/2629673/attachments/1510135/2355667/ACAT-DeepNetworksForPhysicsAnalysis-1.pptx

How could we get rid of simulated data? Great problem, at least in physics

- The Monte Carlo simulation used for training Machine Learning methods always differs to some extend from data.
- So, the best would be to train algorithms on data…
- … => towards **unsupervised learning**?
- … or maybe only **weakly supervised**?


MACHINE LEARNING
UNSUPERVISED LEARNING
Group and interpret data based only on input data
SUPERVISED LEARNING
Develop predictive model based on both input and output data
CLUSTERING
CLASSIFICATION
REGRESSION

<u>Unsupervised learning:</u>

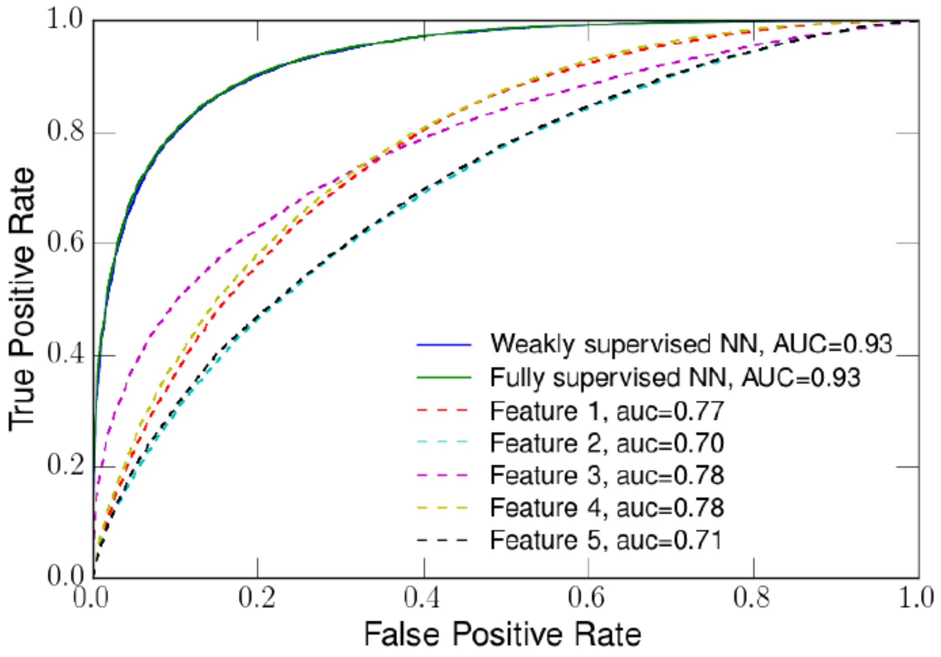
No training datasets are provided, the data is clustered into different classes based on similarity.

Weakly Supervised Classification

- A new approach in Machine Learning: **weakly supervised** classification in which **class proportions are the only input** into the machine learning algorithm.
- **Example problem from particle physics:**
  - **Quark versus gluon tagging** - weakly supervised classification can match the performance of fully supervised algorithms.
  - By design, the new algorithm is insensitive of MC mis-modelling – trained on data.

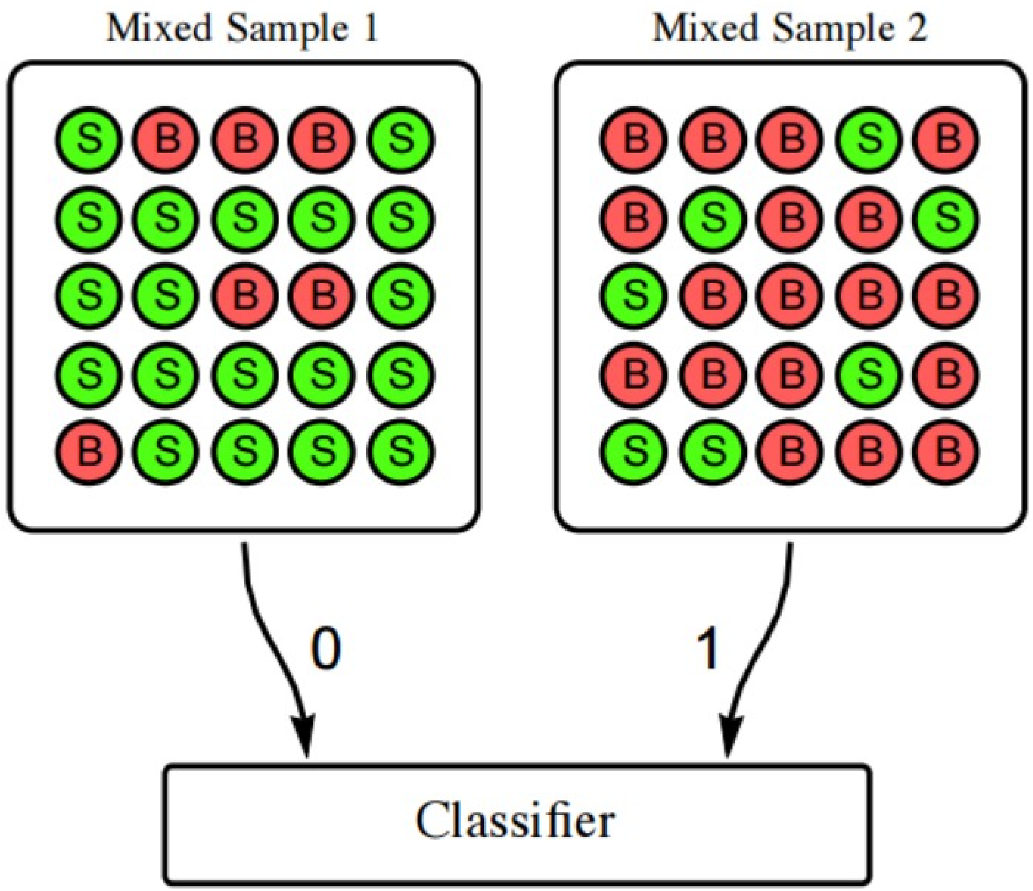
- **Problem:** we have to find in data what is the proportion of gluon and quark jets.
- Maybe template fits in one/some variable/s using again MC?

arXiv:1702.00414



Learning from Data
Classification w/o Labeling

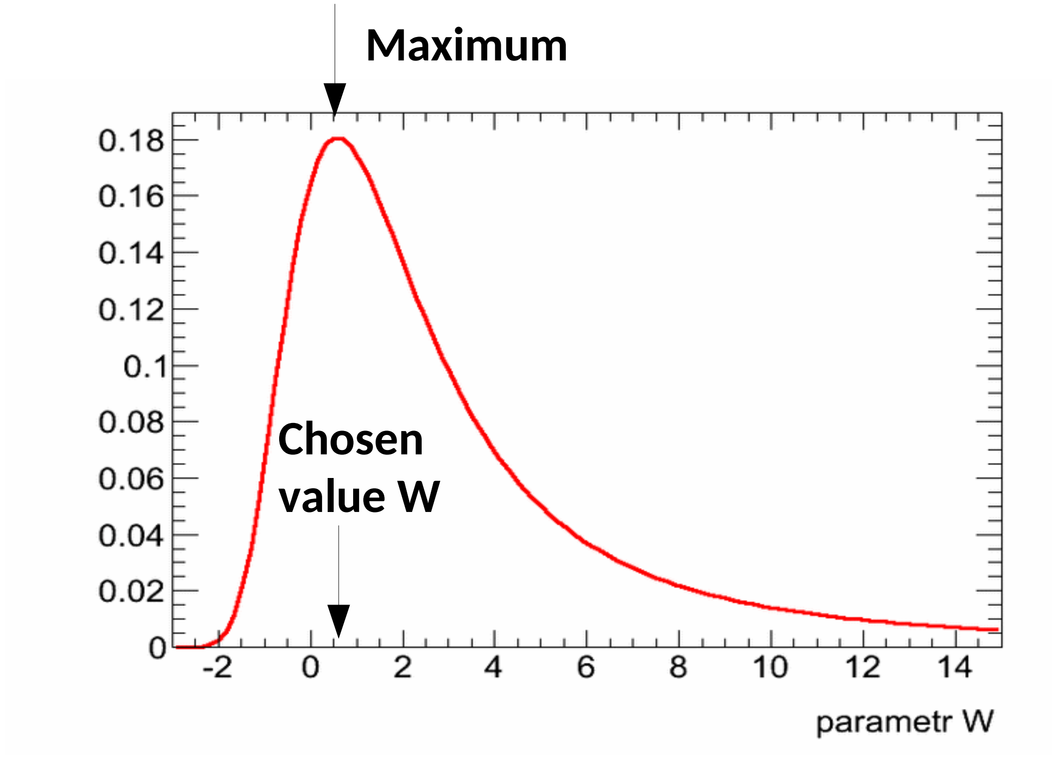
- A step even further is classification w/o labeling (CWoLa) https://arxiv.org/abs/1708.02949
- A classifier is trained to distinguish sample 1 from sample 2, which are mixtures of signal and background with different (and unknown) fractions.
- Such a classifier is optimal for distinguishing signal from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

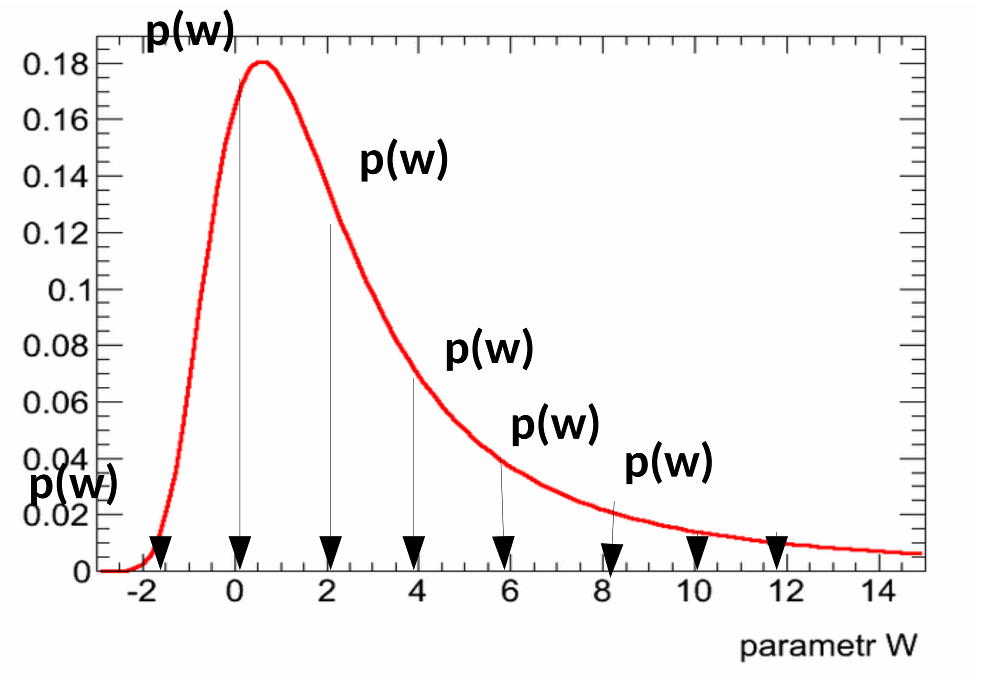


Bayesian learning

- All regression tools (neural networks, decision trees…) described here were returning the most probable output value.
- But a better thing is….
  - **Probability distribution**

Machine vs. Bayesian learning





Machine learning

We chose just one value of a parameter (or just one function).

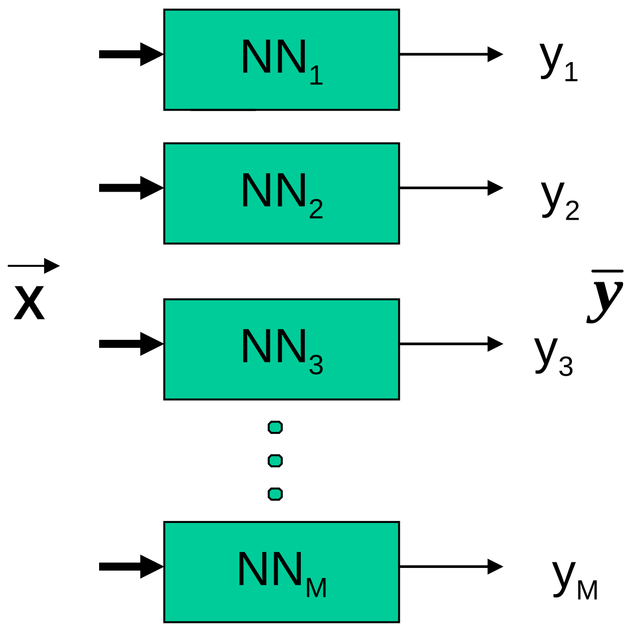
Bayesian learning

Each value of a parameter (or function) has a given probability.

Implementation: Bayesian Neural Networks

Instead of one use many neural networks.

Having many networks we can get the probability distribution.



$$\bar{y} = \sum_i a_i y_i(x)$$

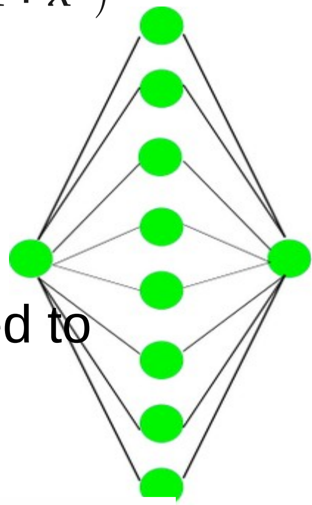
C.M. Bishop
“Neural Networks for Pattern Recognition”,
Oxford 1995

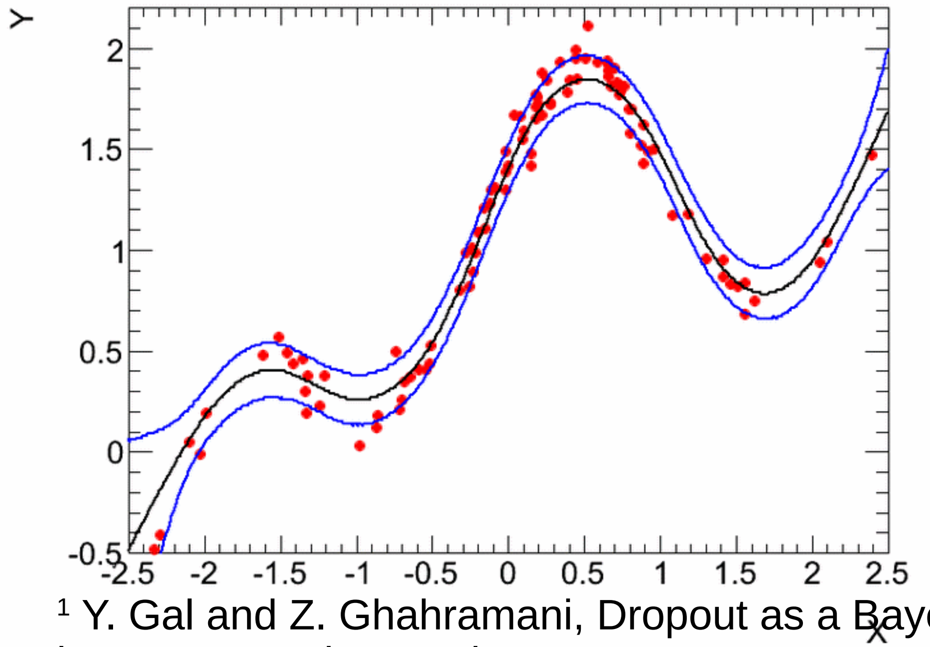
Free software (used by D0 Collaboration):
Radford Neal, http://www.cs.toronto.edu/~radford/fbm.software.html

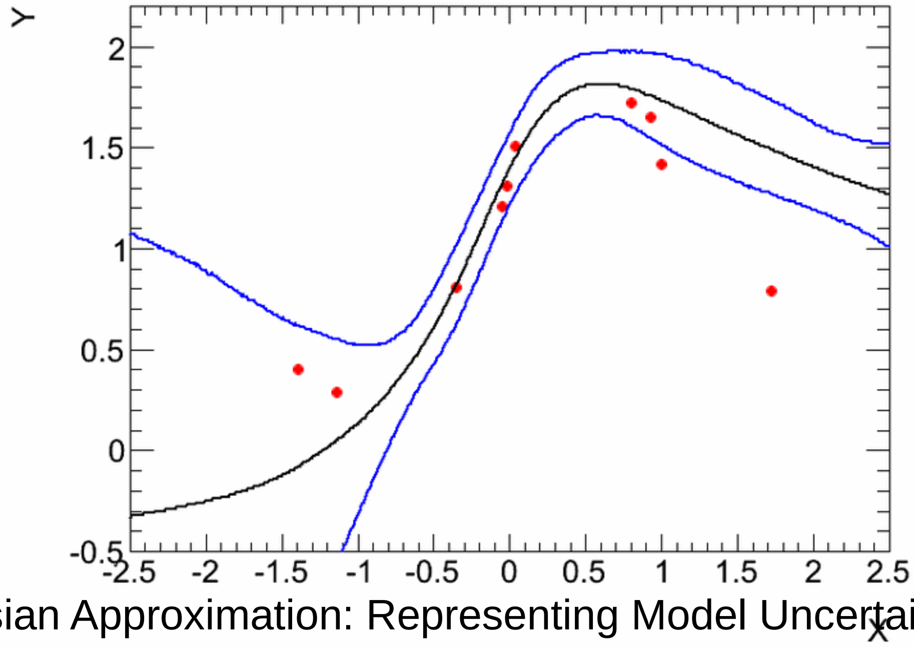
Example of BNN

How does the 8-node, 1-hidden layer BNN works?

- Data generated from function: $y = 0.3 + 0.4x + 0.5\sin(2.7x) + 1.1/(1+x^2)$
  with gaussian noise with б=0.1
- 400 neural networks, distribution => median and 10% Qnt i 90% Qnt
- Gal and Ghahramani (U. of Cambridge)[1] demonstrated that a DNN, in which nodes are randomly dropped during training (a procedure referred to as dropout), approximates variational inference in Bayesian neural networks.



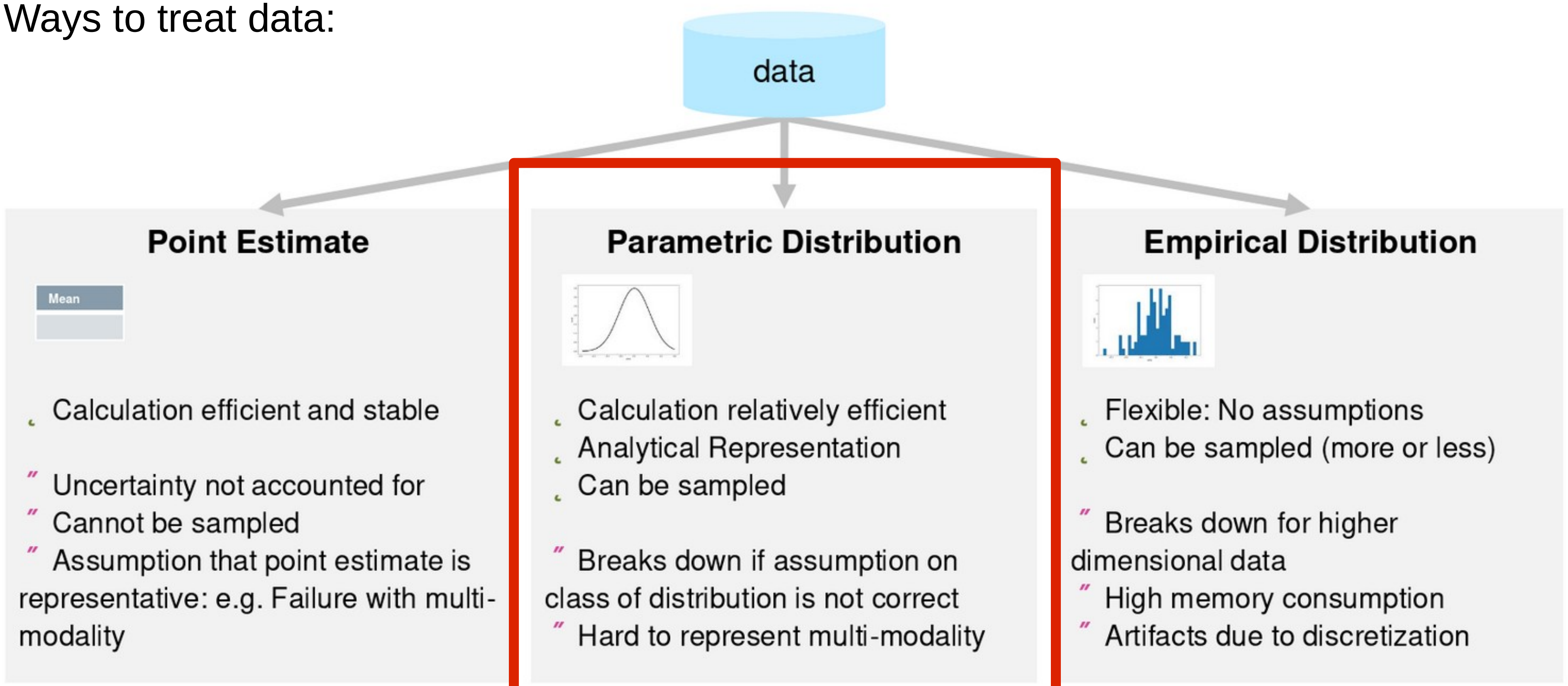




[1] Y. Gal and Z. Ghahramani, Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning, arXiv:1506.02142v5, 25 May 2016

Mixture Density Network (MDN)

Ways to treat data:



Point Estimate

‚ Calculation efficient and stable

″ Uncertainty not accounted for
″ Cannot be sampled
″ Assumption that point estimate is representative: e.g. Failure with multi-modality

Parametric Distribution

‚ Calculation relatively efficient
‚ Analytical Representation
‚ Can be sampled

″ Breaks down if assumption on class of distribution is not correct
″ Hard to represent multi-modality

Empirical Distribution

‚ Flexible: No assumptions
‚ Can be sampled (more or less)

″ Breaks down for higher dimensional data
″ High memory consumption
″ Artifacts due to discretization

Classic Neural Network

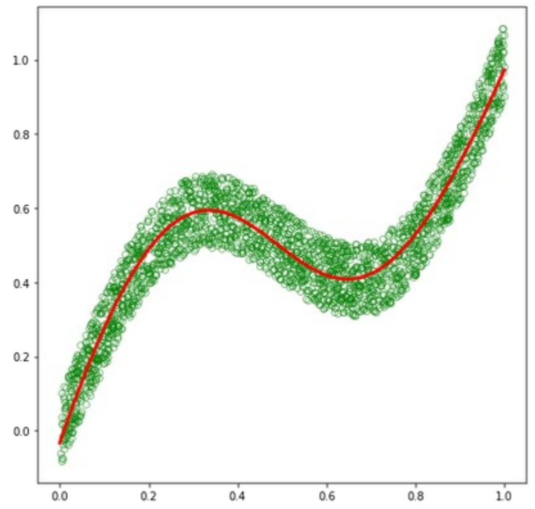
Mixture Density Neural Network

Bayesian Networks???

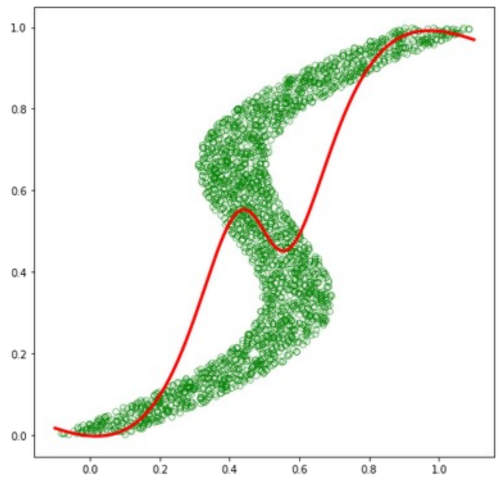
Mixture Density Network (MDN)

- MDN is an interesting model formalism working on supervised learning problems in which the target variable cannot be easily approximated by a single standard probability distribution.
- Conditional probability distribution *p(y|x)* is modeled as a mixture of distributions (few Gaussians), in which the individual distributions and the corresponding mixture coefficients are parametrized by functions of the inputs *x*.

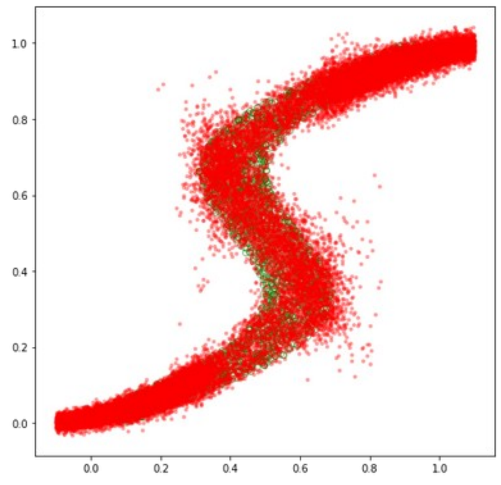
Regression:



Regular network - OK



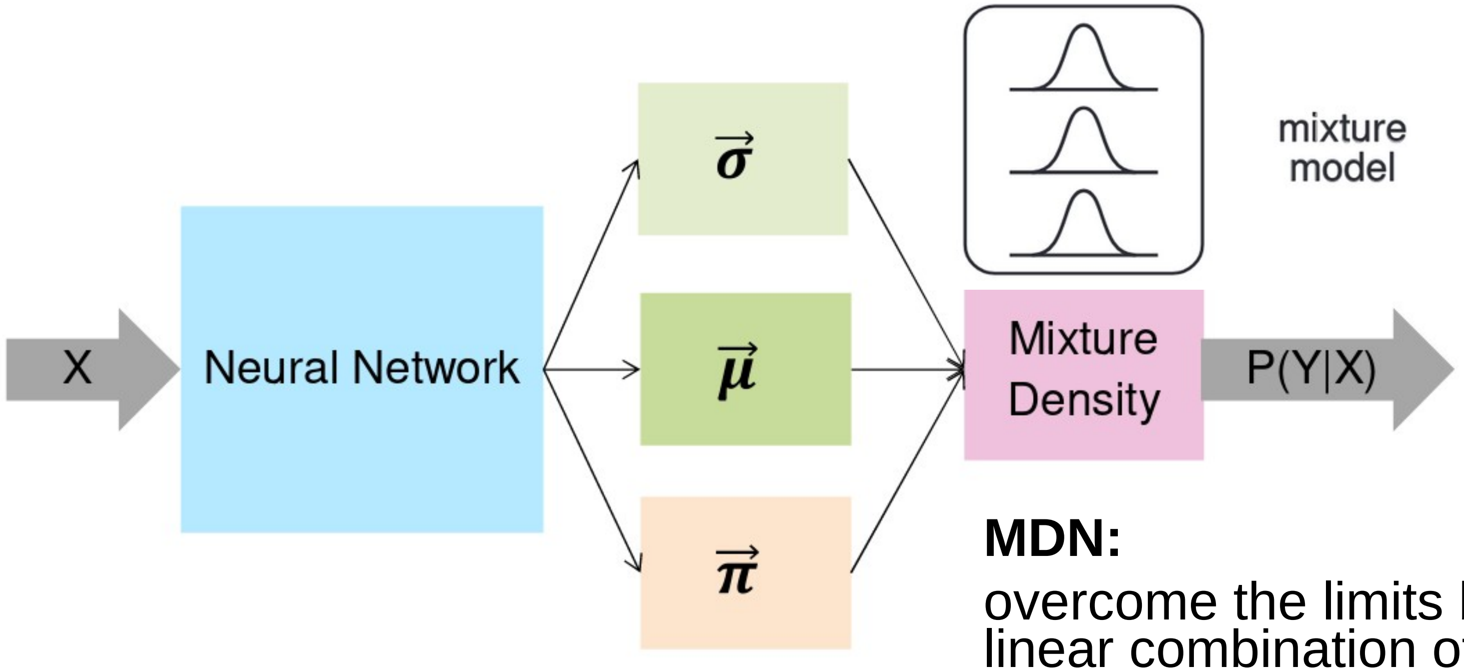
Regular network - problems



MDN sampling - OK

Nice presentation:
http://www.dbs.ifi.lmu.de/Lehre/DLAI/WS18-19/script/06_uncertain.pdf

Mixture Density Network (MDN)



$\vec{\sigma}$: standard deviations
$\vec{\mu}$: means
$\vec{\pi}$: cluster weights

Loss: $-\log P(\mathbf{Y}|\mathbf{X})$

MDN:
overcome the limits by using a linear combination of kernel function

MDN return not only maximum probability value, but the probability distribution: returns errors!

Example from https://github.com/cpmpercussion/keras-mdn-layer
Google Colaboratory runable adapted example:
https://github.com/marcinwolter/DNN_examples/blob/master/MDN_1D_sine_prediction.ipynb

Reference: Bishop, Christopher M. Mixture density networks. Technical Report NCRG/4288,Aston University, Birmingham, UK, 1994
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.120.5685&rep=rep1&type=pdf

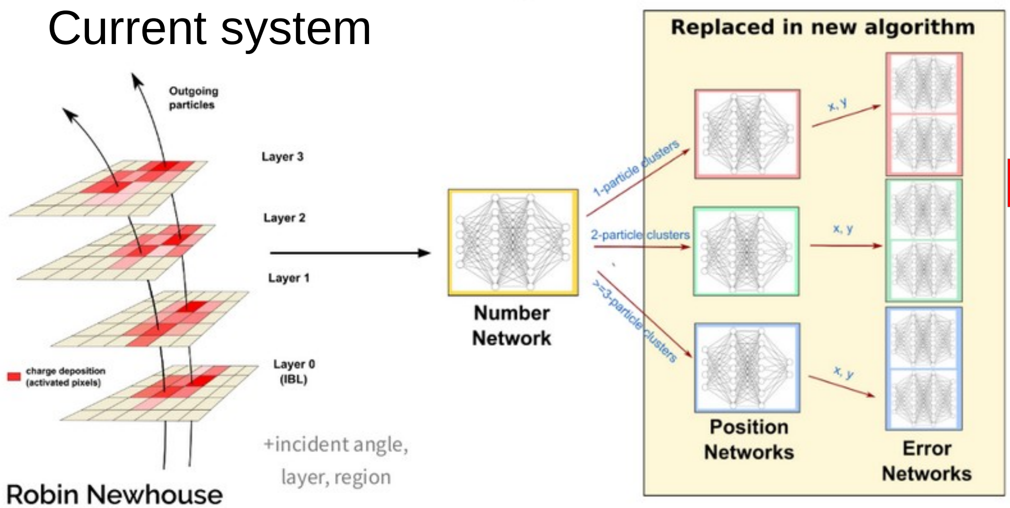
MDM example
- ATLAS cluster position calculation

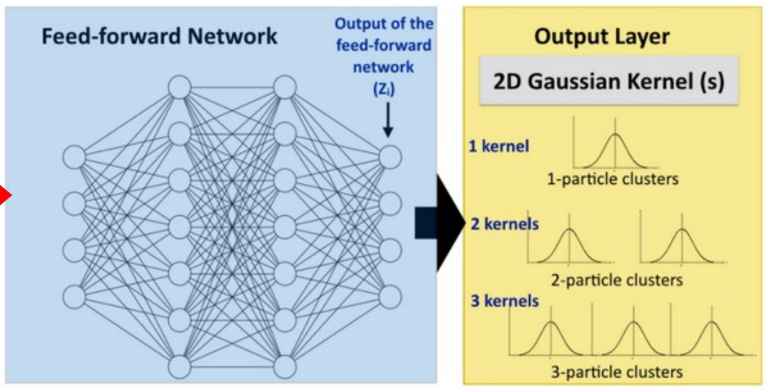
Currently uses a three-step process

1. Number Network: Determine the **number of particles** in this hit
2. Position Network: Determine the **x-y position** of each particle
3. Error Network: Estimate the **error in x and y** for each particle

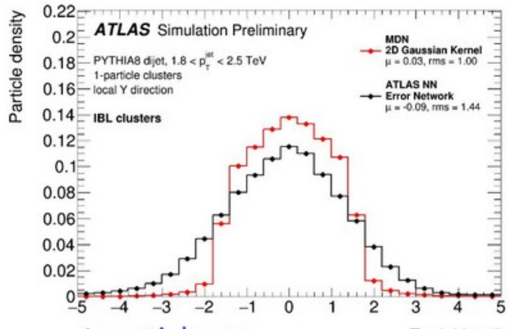
MDN: Feedforward network + Gaussian kernel(s)

Current system

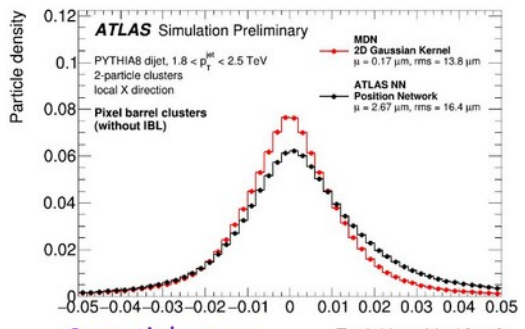




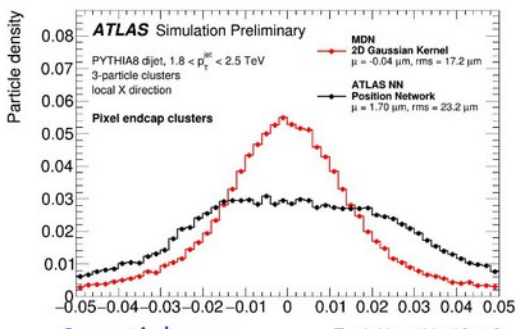
ATLAS studies of Elham E Khoda:
https://indico.cern.ch/event/795039/contributions/3432383/attachments/1856640/3049881/Machine_Learning_in_CTIDE_1.pdf



1 particle err.

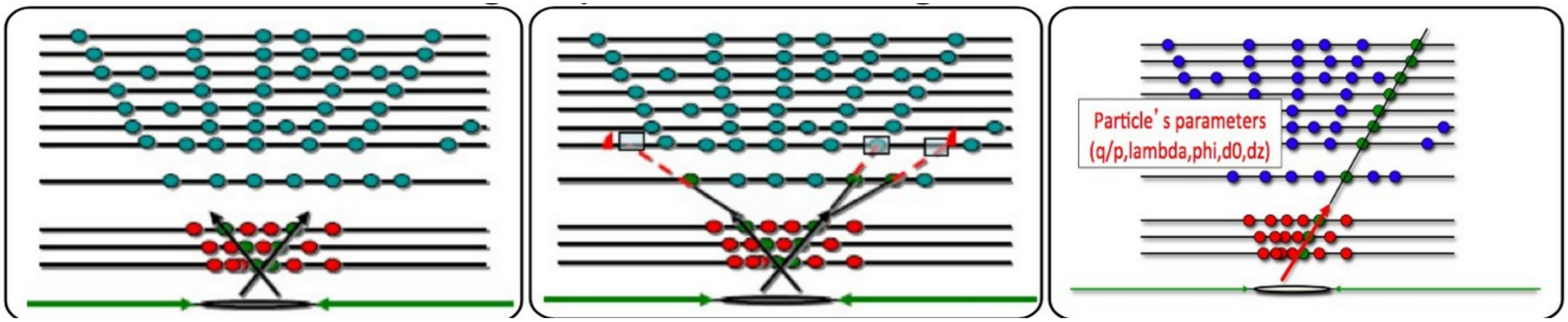


2 particle pos.



3 particle pos.

Pattern Recognition in High Energy Physics

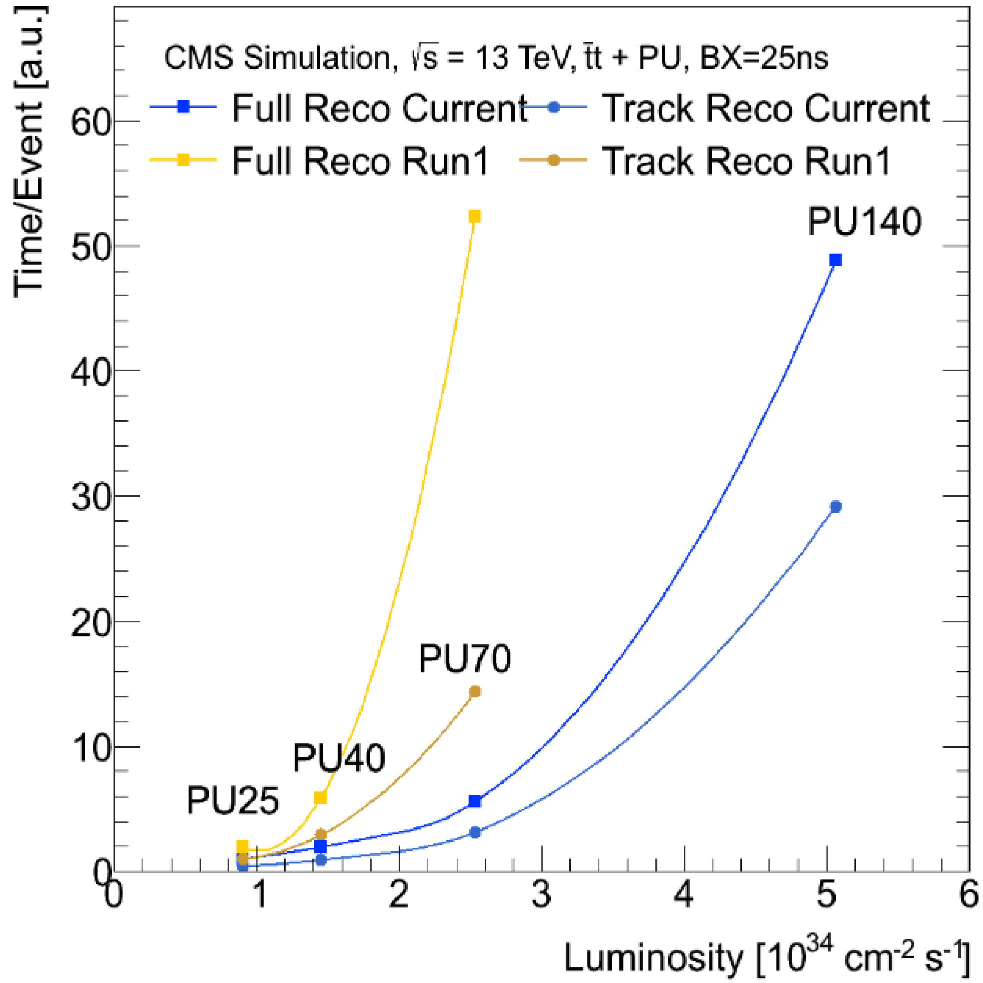


Seeding **Track Building** **Track Fitting**

- **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

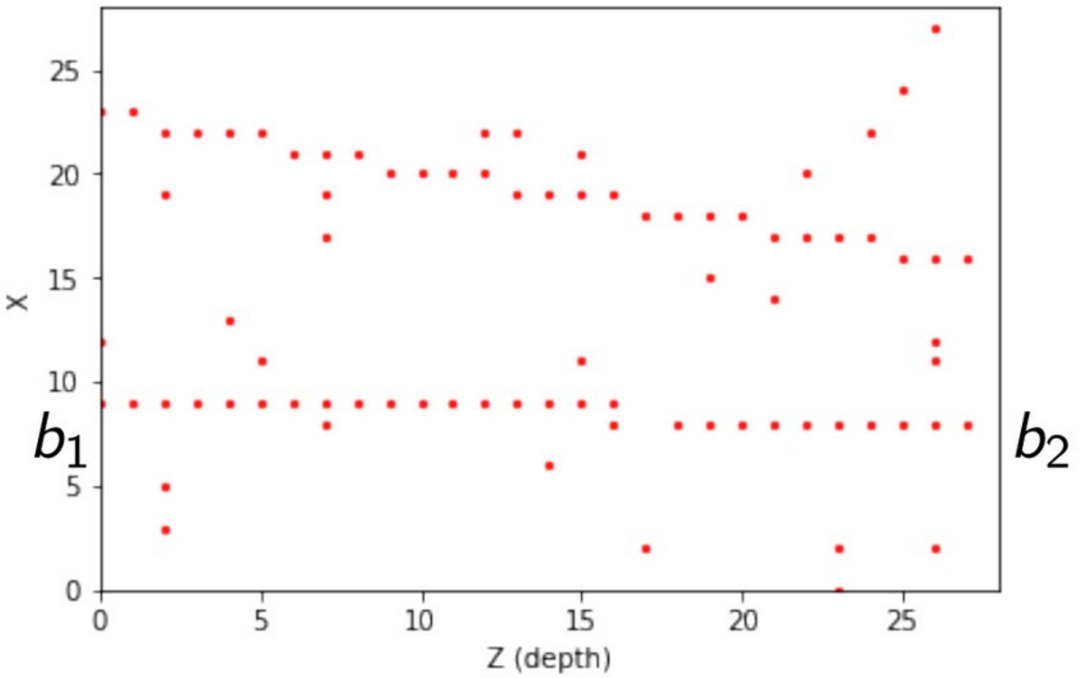
- The time needed to process one event grows quickly (worse than quadratic) with luminosity (number of collisions).
- Huge part of CPU consumption is the track finding.

Deep Neural Network (DNN)?

- Fast, parallel, in principle does 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/*

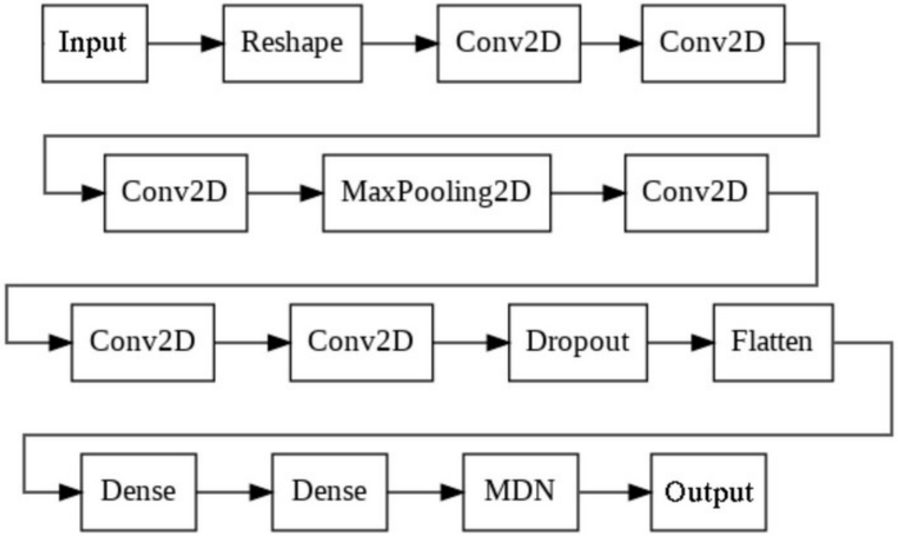
Tracking in 2D toy model

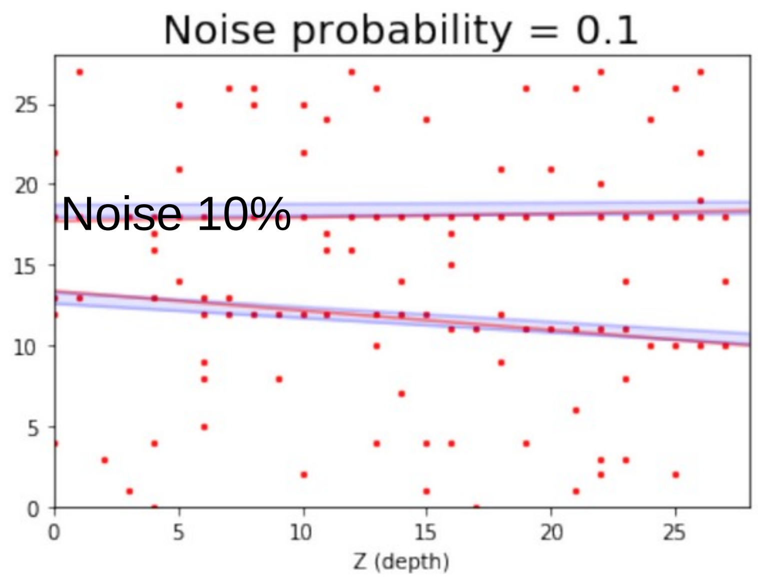
- **CNN returns track parameters (regression)**
- What about many tracks?
  - Solution: add **Mixture Density Network(MDN)** layer to process many tracks.
  - Straight tracks – described by two parameters. Each parameter has associated MDN Gaussian
  - If a number of tracks lower than expected some MDN outputs have very low amplitude.
  - **Important** - we are getting errors of track parameters

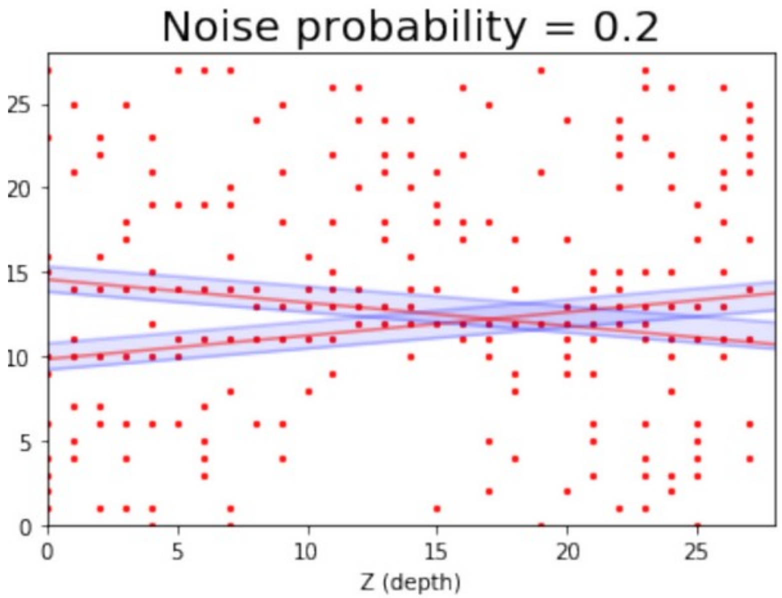


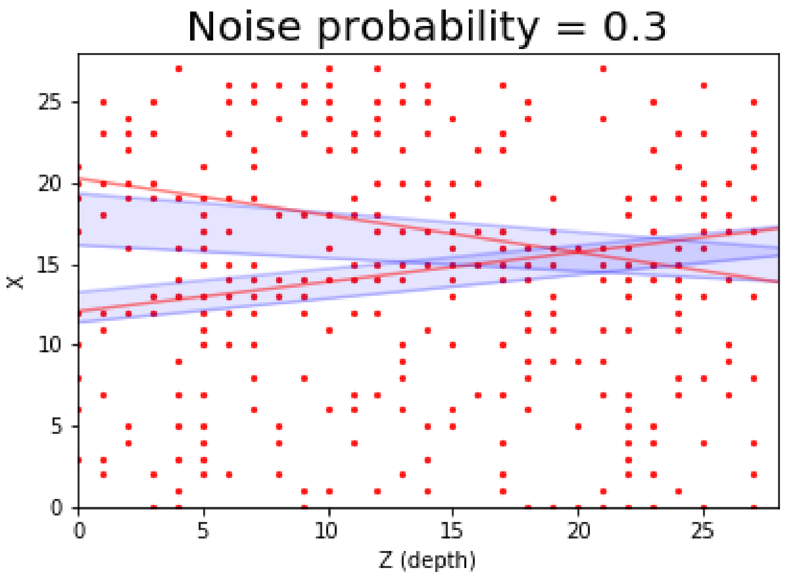
Track described by to interception parameters b_1 and b_2.

Mixed Density Network



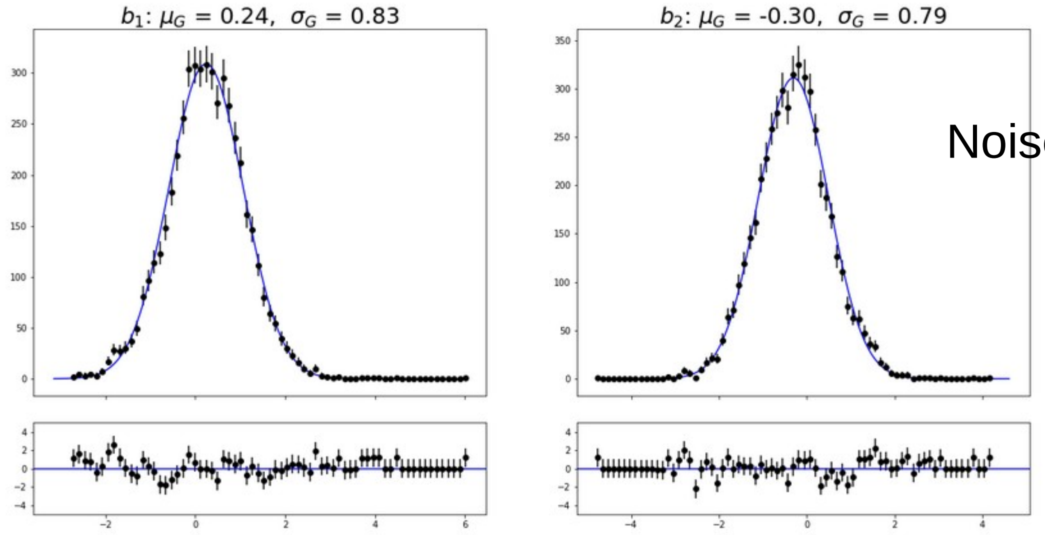




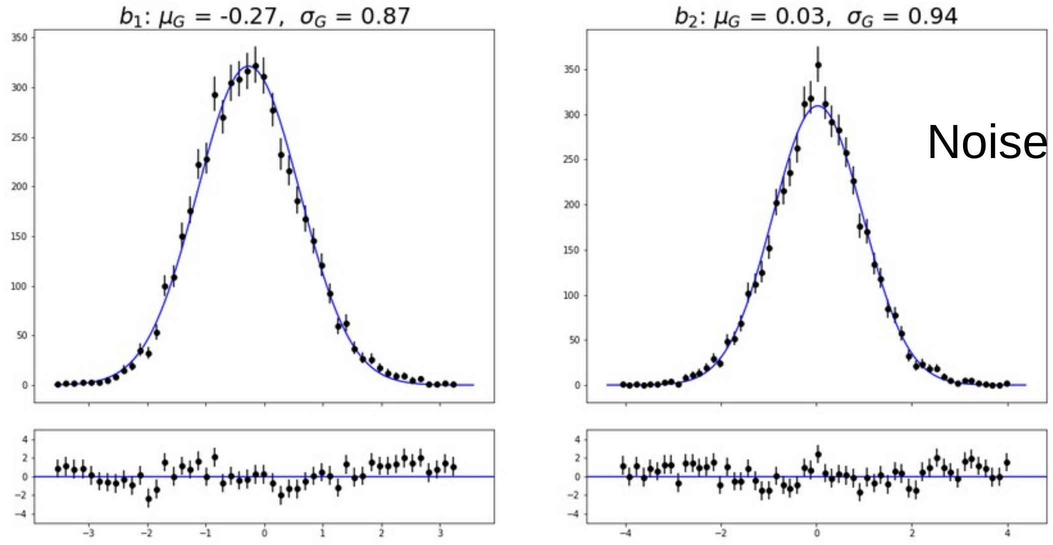


Designed by **Karol Białas** and **Mateusz Słysz** summer students at IFJ
https://github.com/marcinwolter/DNN_examples/blob/master/dnn_tracking_2D_mdn_multimod.ipynb

MDN Tracking – error estimation



Noise 10%



Noise 30%

Pull plots have a width not far from 1.

If a random variable x is generated repeatedly with a Gaussian than the pull distribution:

$$g = \frac{x-\mu}{\sigma}$$

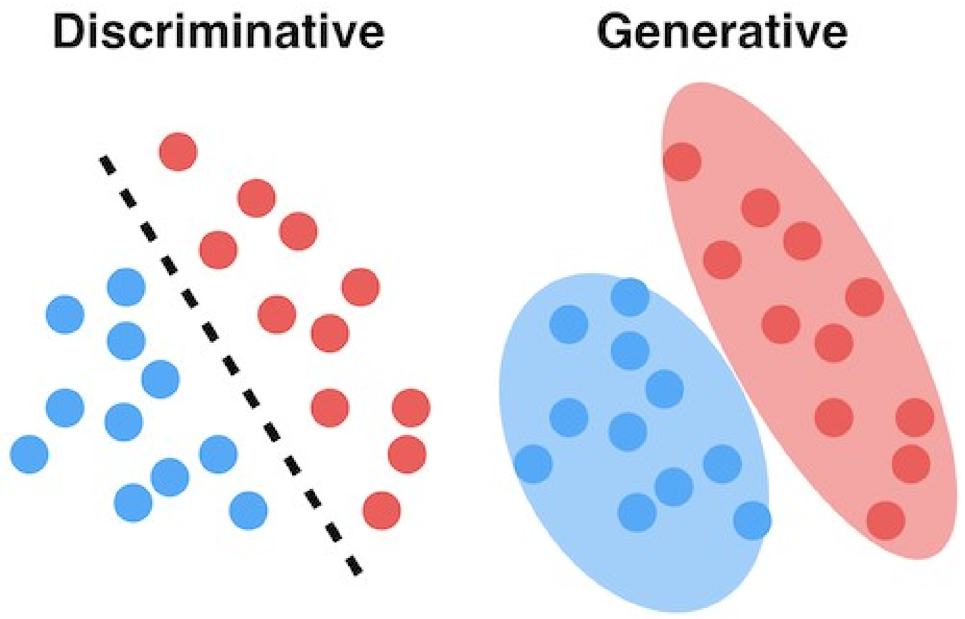
will be distributed as a standard Gaussian with mean zero and unit width.

Generative Adversarial Nets (GANs)

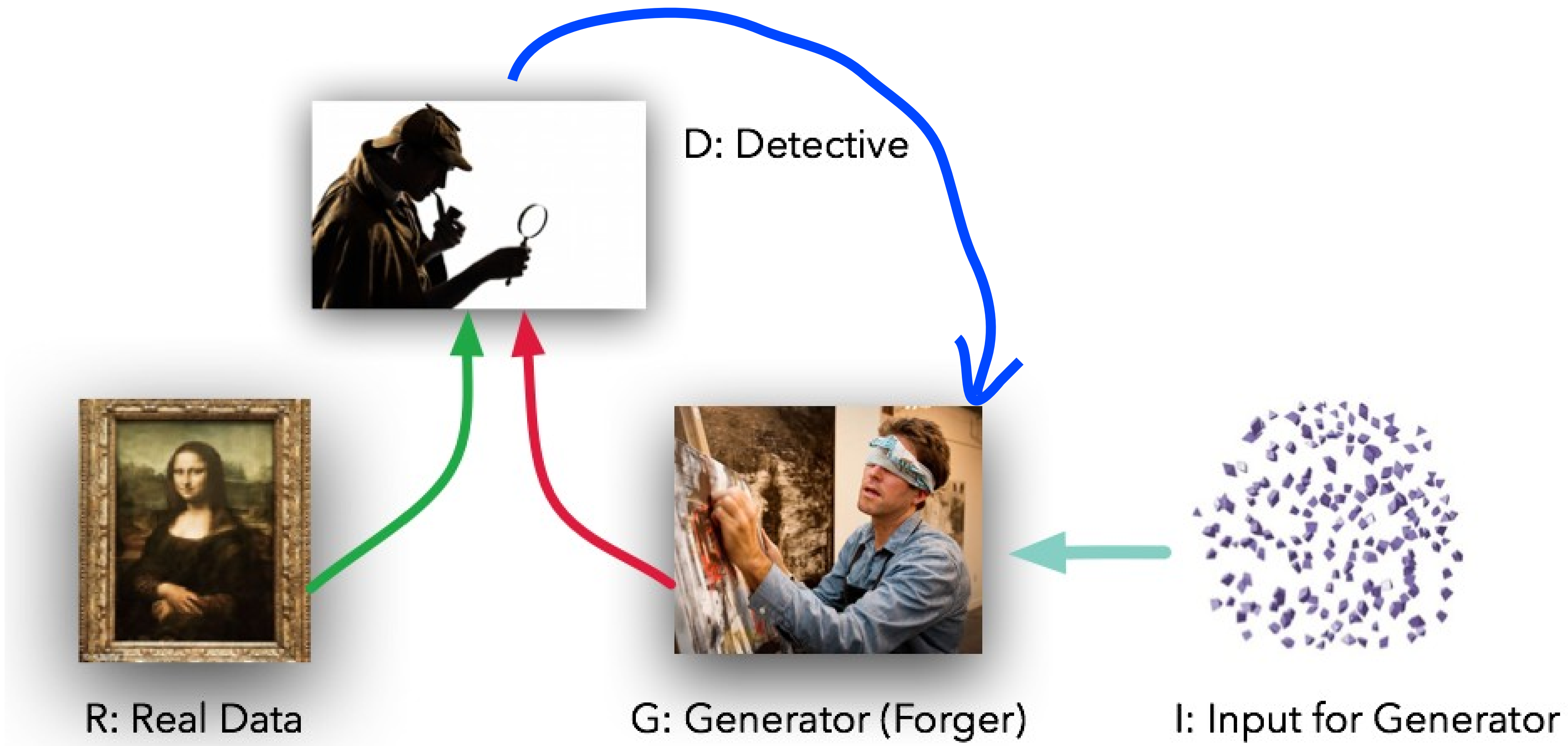
- GANs were introduced Ian Goodfellow and others in 2014 . Yann LeCun called adversarial training “the most interesting idea in the last 10 years in ML.” https://arxiv.org/abs/1406.2661
- GANs’ can learn to mimic any distribution of data. They can be taught to create worlds similar to our own in any domain: images, music, speech, prose. They are robot artists!

How do GANs work?

- **Discriminative algorithms** - classify input data; given the features, they predict a label or category to which that data belongs (*signal* or *background*)
- **Generative algorithms** – do the opposite, assuming the event is *signal*, how likely are these features?
- Another way to distinguish discriminative from generative like this:
  - **Discriminative models** learn the boundary between classes
  - **Generative models** model the distribution of individual classes



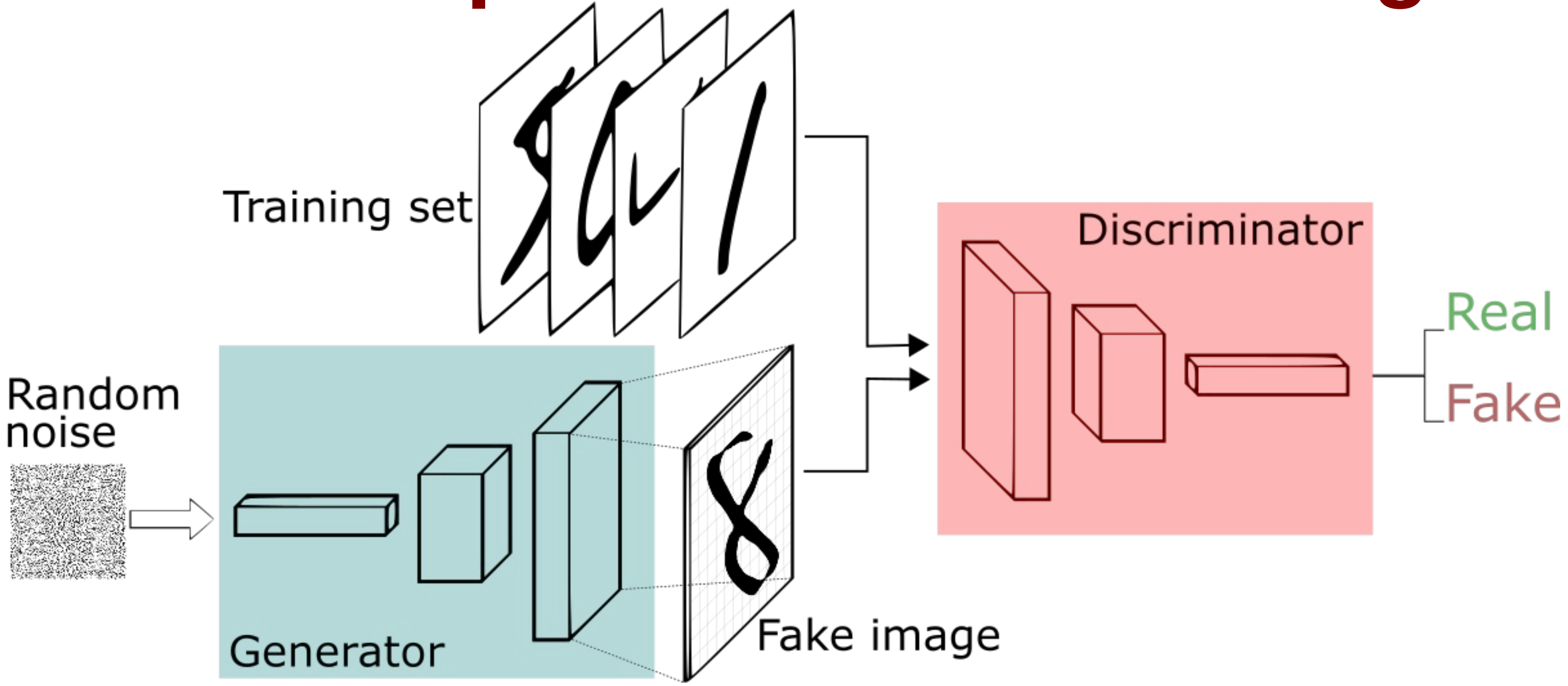
Blind forger and detective



The forger has never seen Mona Lisa, but gets the judgments of detective and tries to fool him (i.e. paint something that looks like Mona Lisa).

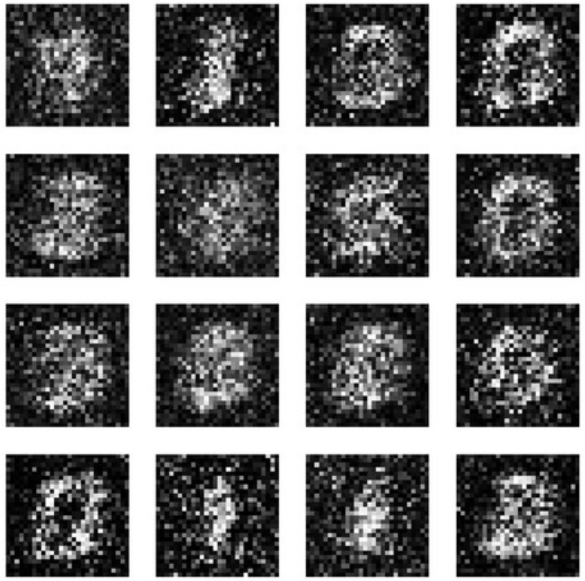
They both (forger and detective) have to train in parallel (important), since if detective is to clever the forger will never paint anything acceptable.

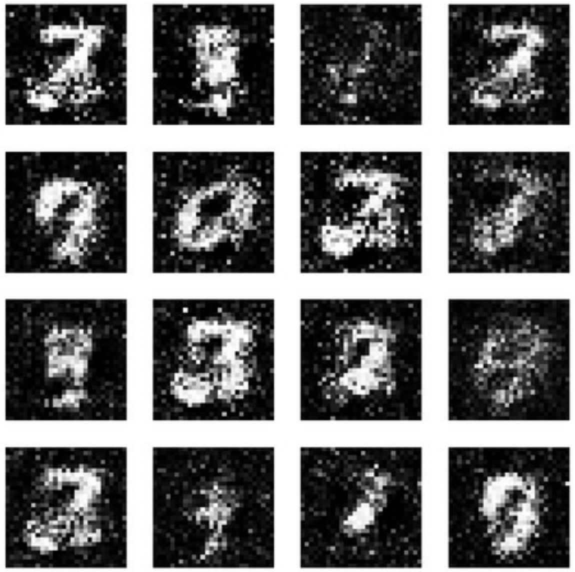
GANs example – hand-written digits

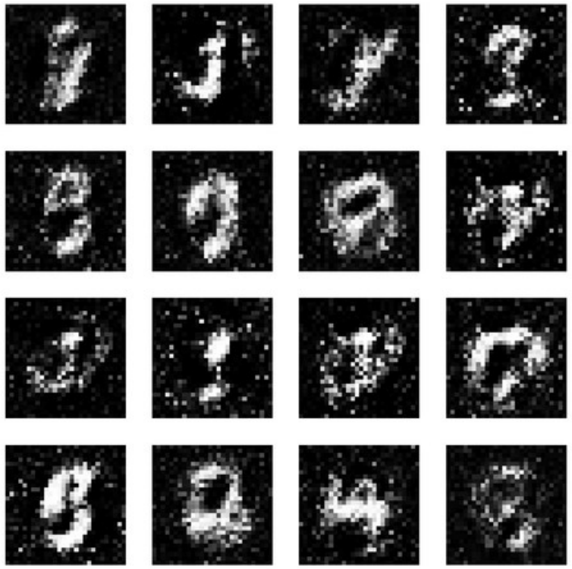


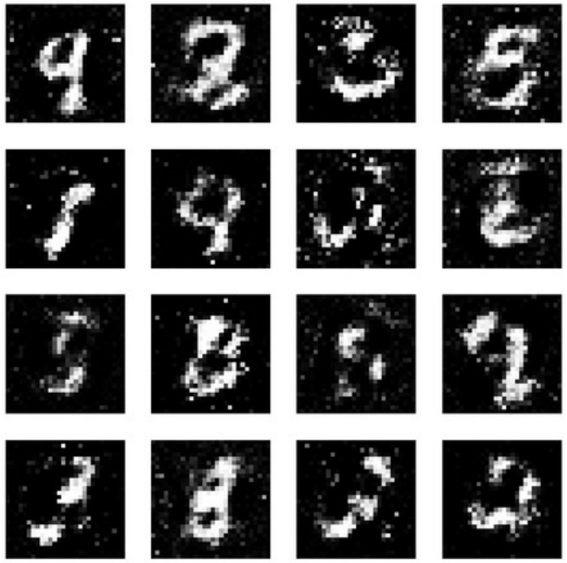
- **Training set** – MNIST: hand-written digits supplied by US post.
- **Discriminator** – convolutional neural network labeling images as real or fake.
- **Generator** - inverse convolutional network (while a standard convolutional classifier takes an image and downsamples it to produce a probability, the generator takes a vector of random noise and upsamples it to an image).

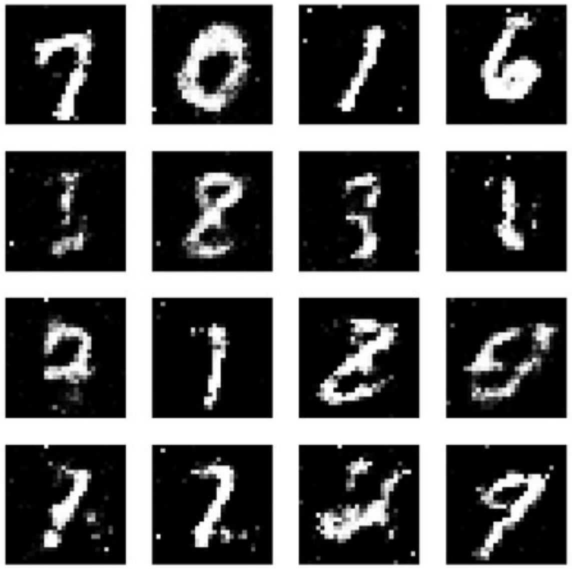
Implementation: Python code using Keras interface and TensorFlow backend.

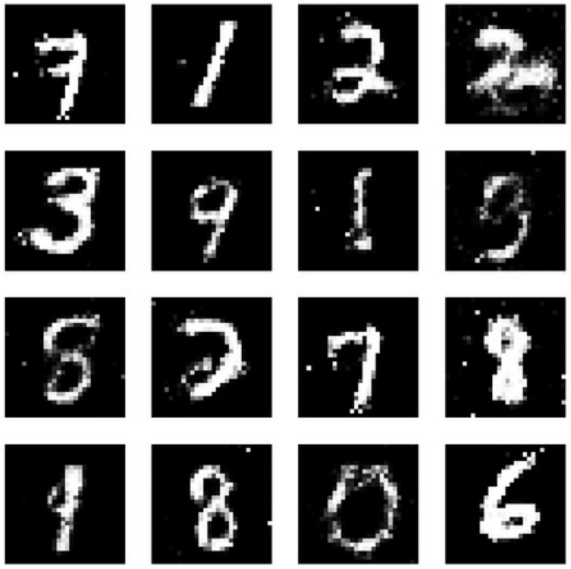

400 cycles


800 cycles


1200 cycles


2400 cycles


8000 cycles

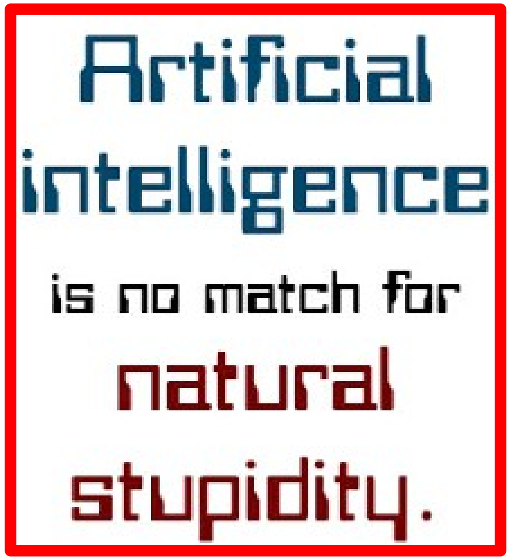

19900 cycles

Each cycle digits look more and more realistic.

Updated code from the net:
https://github.com/marcinwolter/DNN_examples/blob/master/gan_generate_letters.ipynb

Conclusions

- Many new methods were developed recently.
- Machine Learning approach becomes to be used not only for classification, but also for other tasks.
- Each month new application appear!


Artificial intelligence is no match for natural stupidity.

- The development is driven by AI applications (image recognition, autonomous cars etc). But the physics community can profit!
- More and more advanced ML techniques have application in HEP. Try to find a new one!

GANs give profits!!!!

"Edmond de Belamy" is part of a fictitious family created by a "generative adversarial network," of which there's ten other paintings. "Edmond" is one of the most striking of the paintings, and will likely become an important part of art history going forward thanks to its huge selling price. The generator behind the painting created new portraits based on 15,000 from the last 600 years, taking existing art and crafting something wholly original and quite alien.