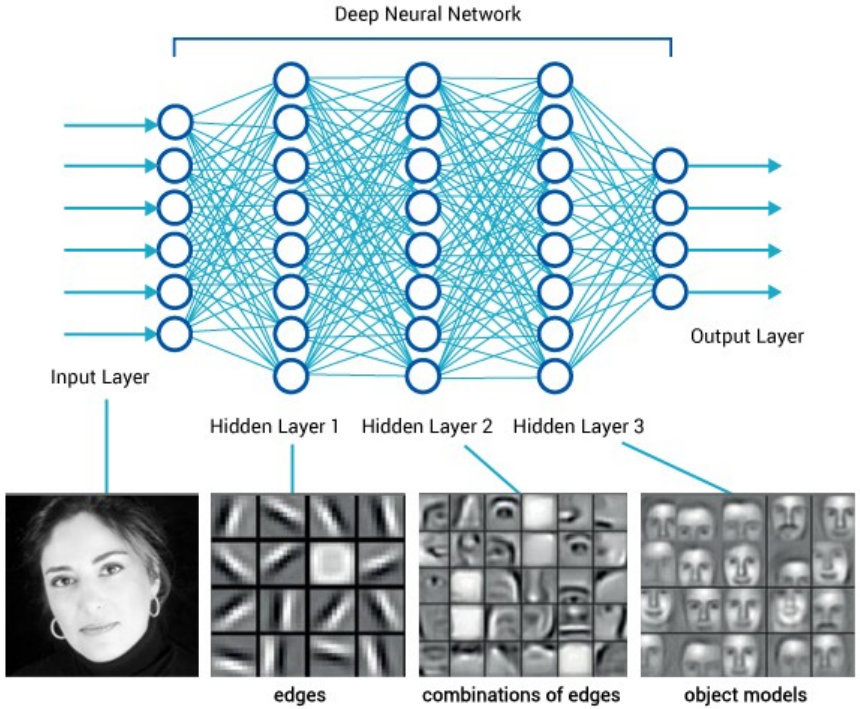
Machine learning

Lecture 5



Marcin Wolter

IFJ PAN

28 April 2017

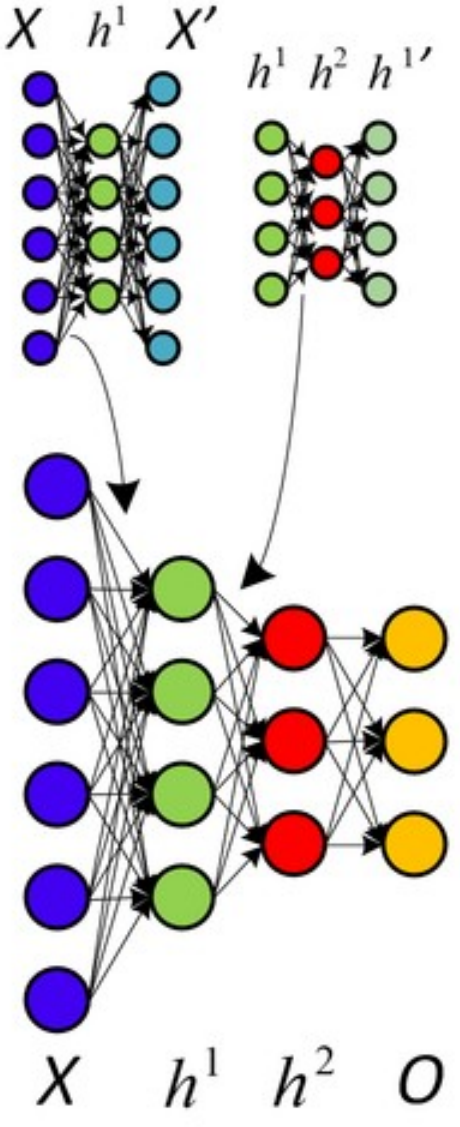
- Application to the Higgs searches
- Deep learning
- Convolution network

Make a network deeper?

Methods with many processing steps (or equivalently with hierarchical feature representations of the data) are more efficient for difficult problems than shallow methods (which two-layer ANNs or support vector machines are examples of)[1]

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?



[1]Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. Advances in neural information processing systems, 19, 153.

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.
  - **We can train a DNN on MNIST as an exercise!**
- 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/

So why did supervised learning with backpropagation not work well in the past?

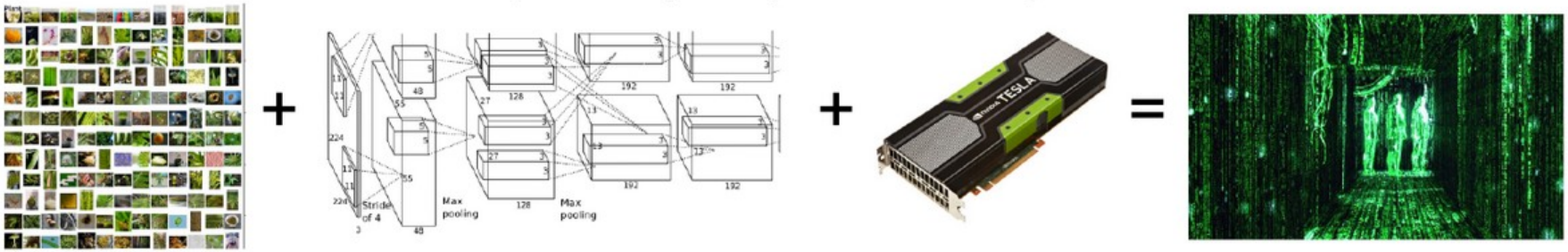
1) Our labeled datasets were thousands of times too small.
2) Our computers were millions of times too slow.
3) We initialized the weights in a stupid way.
4) We used the wrong type of non-linearity.

Geoffrey Hinton on youtube lecture

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

> ***Deep Learning =***
> ***Lots of training data + Parallel Computation + Scalable, smart algorithms***

The Deep Learning "Computer Vision Recipe"

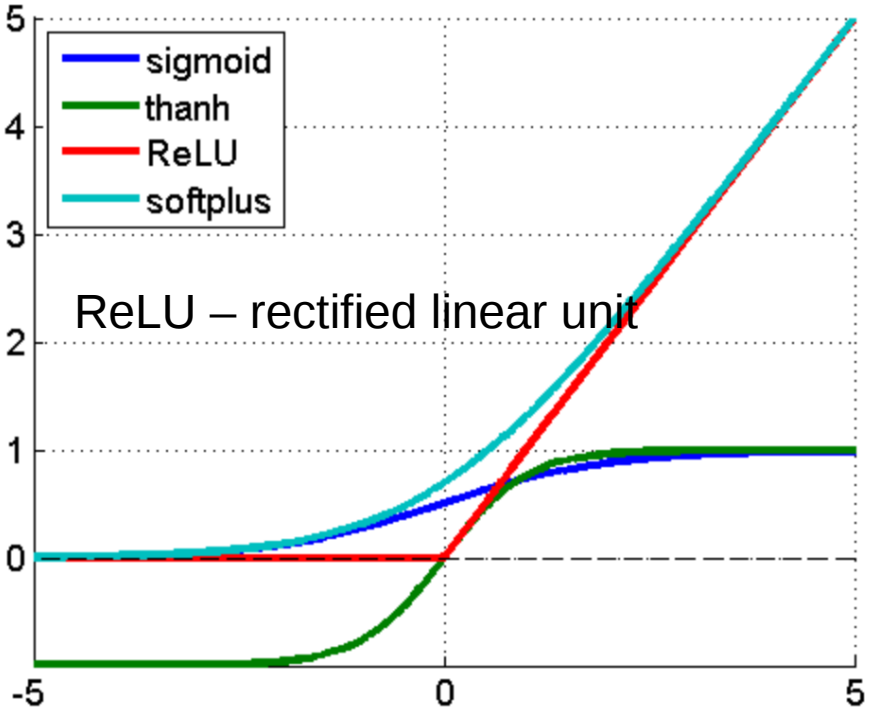


Comments

We used the wrong type of non-linearity.

It was found, that the very much non-differentiable and very simple function ReLU f(x)=max(0,x) tends to be the best.

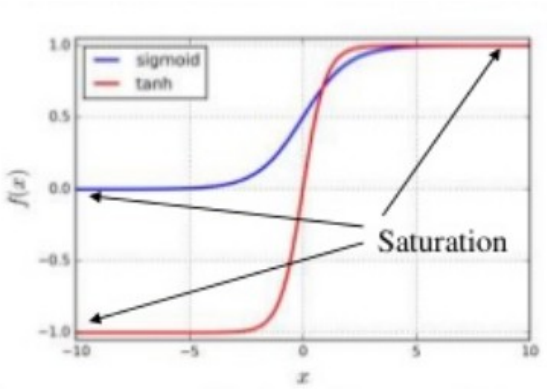
1) Leads to sparse representations, meaning not many neurons need to output non-zero values.
2) Simplicity of the function, and its derivatives, => faster to work with.
3) Helps avoid the vanishing gradient problem that was the bane of backpropagation.

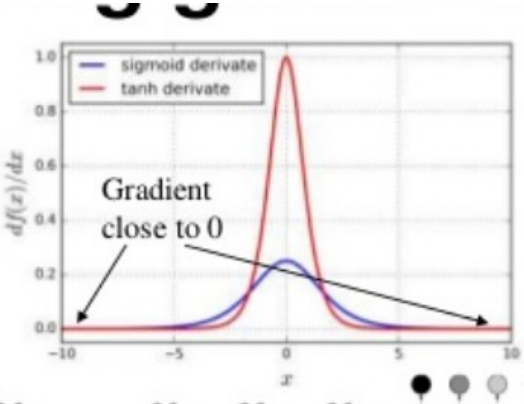


- *Jarrett, K., Kavukcuoglu, K., Ranzato, M. A., & LeCun, Y. (2009, September). What is the best multi-stage architecture for object recognition?. In Computer Vision, 2009 IEEE 12th International Conference on (pp. 2146-2153). IEEE.*
- *Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10) (pp. 807-814).*
- *Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In International Conference on Artificial Intelligence and Statistics (pp. 315-323).*

Vanishing gradient

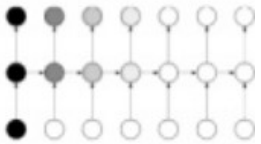
- **Vanishing gradients**—In case of deep networks, for any activation function, abs(dW) will get smaller and smaller as we go backwards with every layer during back propagation. The earlier layers are the slowest to train in such a case.
- The weight update is minor and results in slower convergence. This makes the optimization of the loss function slow. In the worst case, this may completely stop the neural network from training further.





Saturated neurons gradients → 0

$$\frac{\partial h_t}{\partial h_0} = \frac{\partial h_t}{\partial h_{t-1}} \cdot \ldots \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial h_0}$$



Drive previous layers gradients to 0
(especially for far time-stamps)

- Decays exponentially
- Network stops learning, can't update
- Impossible to learn correlations
- between temporally distant events

Known problem for deep feed-forward networks.
For recurrent networks (even shallow) makes **impossible to learn long-term** dependencies!

Comments

We initialized the weights in a stupid way.

- It was not so much choosing random weights that was problematic, as choosing random weights without consideration for which layer the weights are for. The old vanishing gradient problem happens, basically, because backpropagation involves a sequence of multiplications that invariably result in smaller derivatives for earlier layers.
- **Unless weights are chosen with difference scales according to the layer they are in - making this simple change results in significant improvements.**
- We use a heuristic to initialize the weights depending on the non-linear activation function.

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

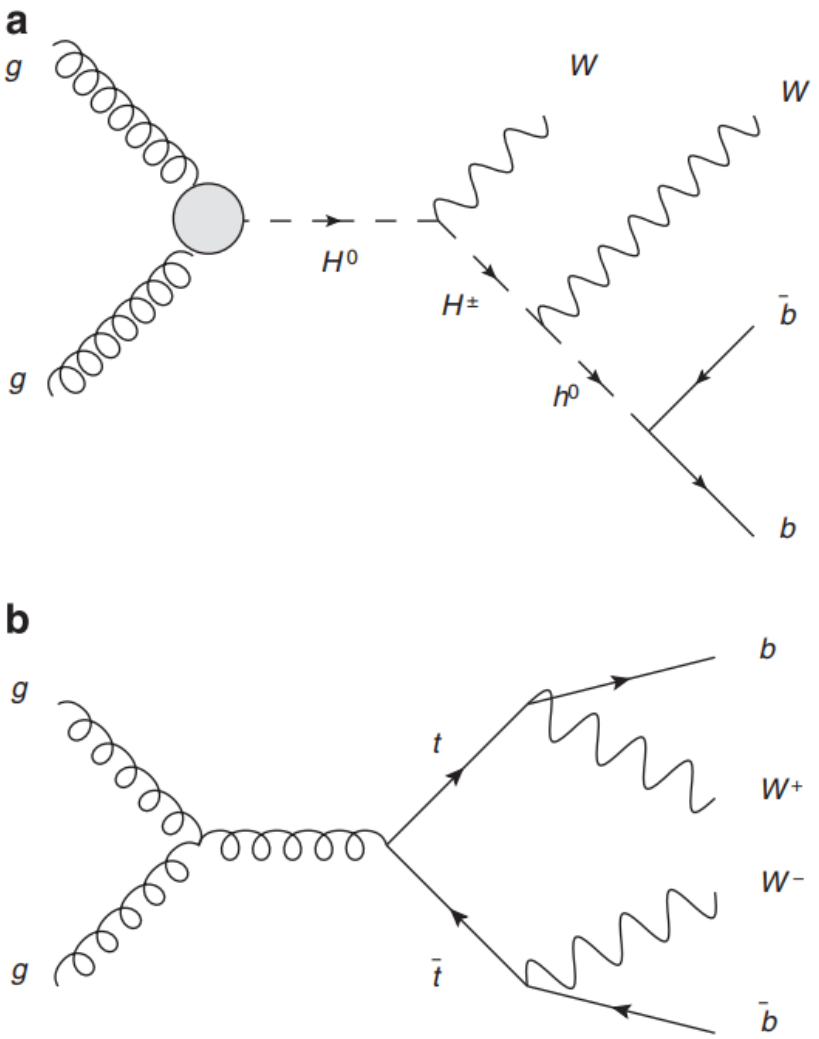


Figure 1 | Diagrams for Higgs benchmark. (**a**) Diagram describing the signal process involving new exotic Higgs bosons H^0 and $H^\pm$. (**b**) Diagram describing the background process involving top quarks (t). In both cases, the resulting particles are two W bosons and two b-quarks.

Exotic Higgs boson searches (simulated CMS data)

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

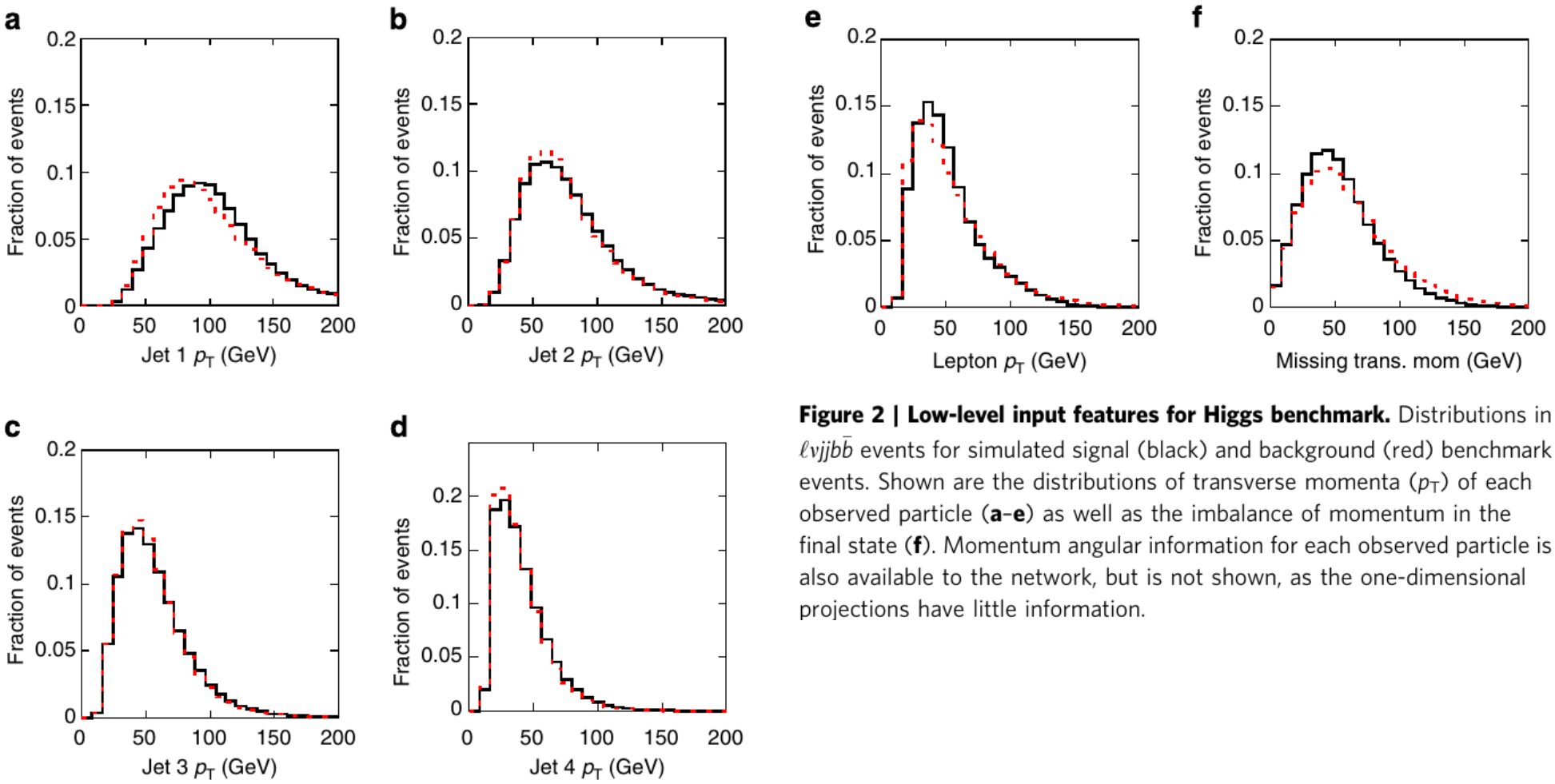


Figure 2 | Low-level input features for Higgs benchmark. Distributions in $\ell\nu jjb\bar{b}$ events for simulated signal (black) and background (red) benchmark events. Shown are the distributions of transverse momenta (p_T) of each observed particle (**a**–**e**) as well as the imbalance of momentum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

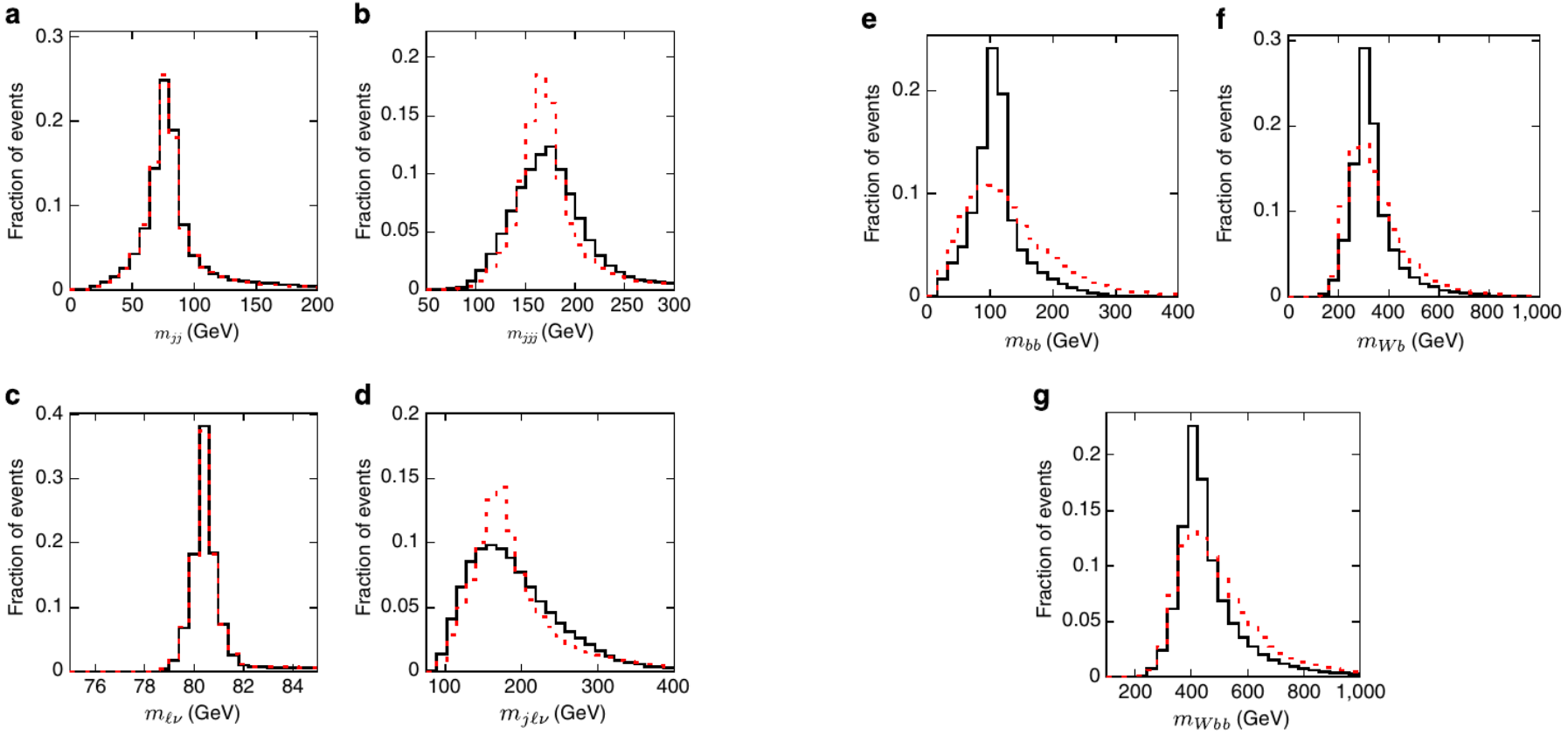


Figure 3 | High-level input features for Higgs benchmark. Distributions in simulation of invariant mass calculations in $\ell\nu jjb\bar{b}$ events for simulated signal (black) and background (red) events.

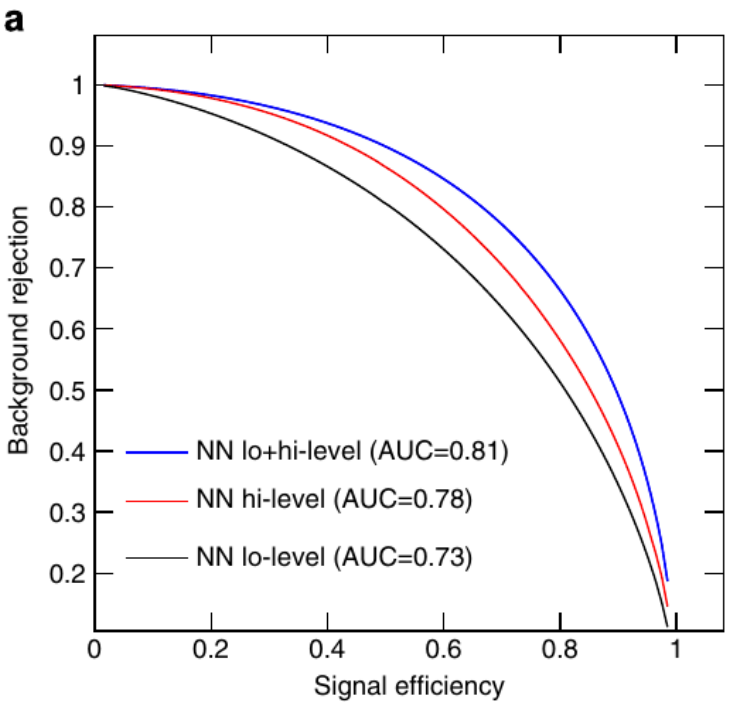
Higgs searches

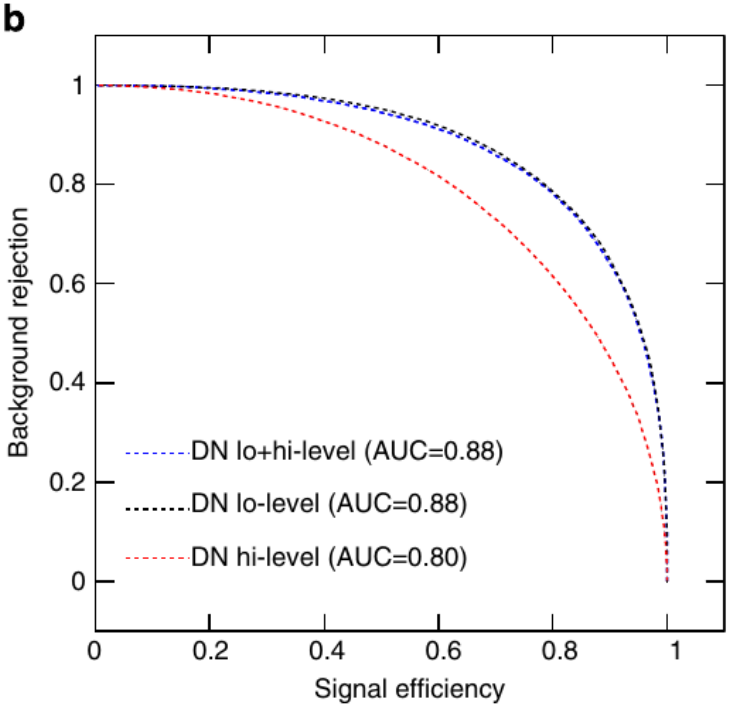
- Data: 2 600 000 events for training, 100 000 for validation
- Deep NN: 5 layers, 300 nodes in each layer, fully connected

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.



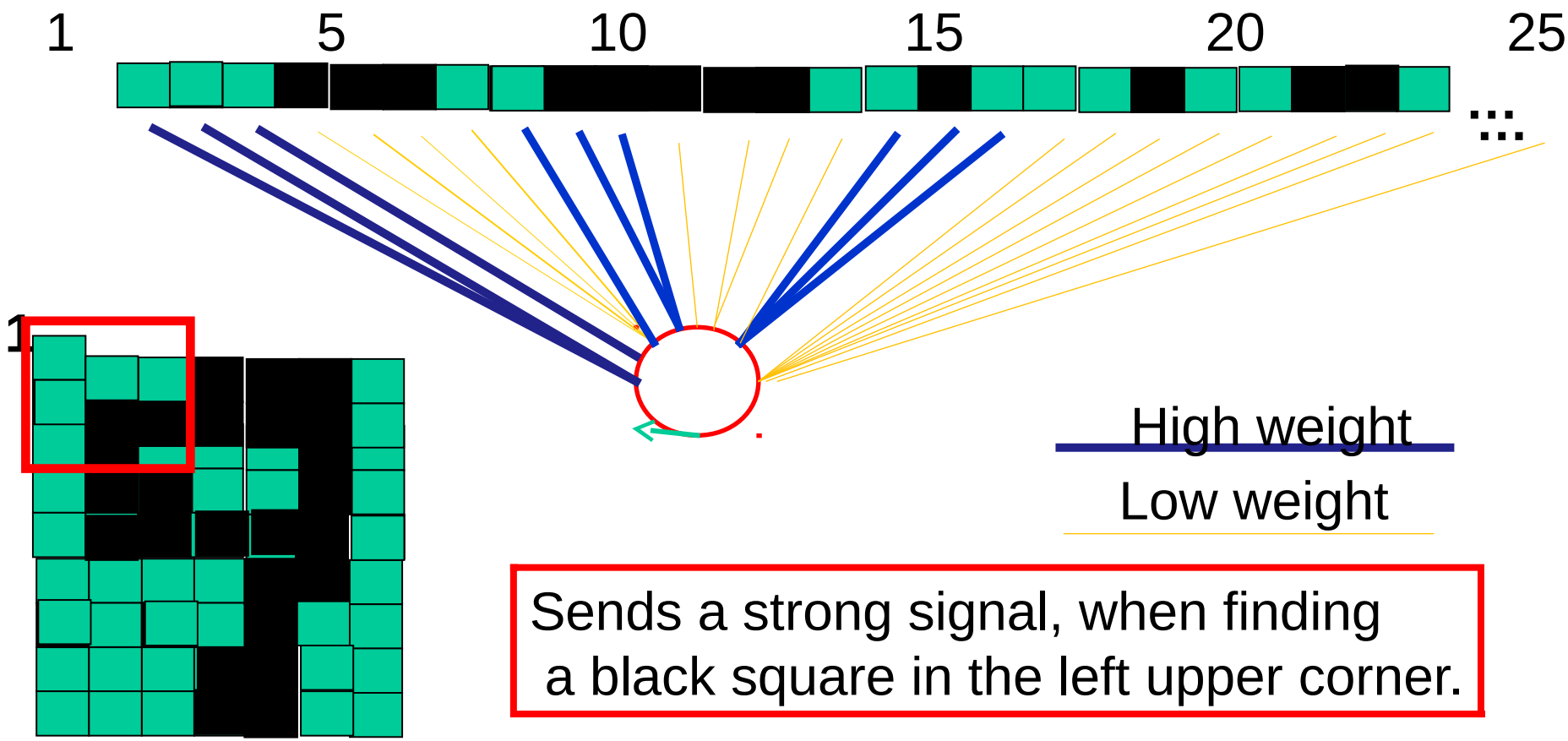


Summary of this exercise

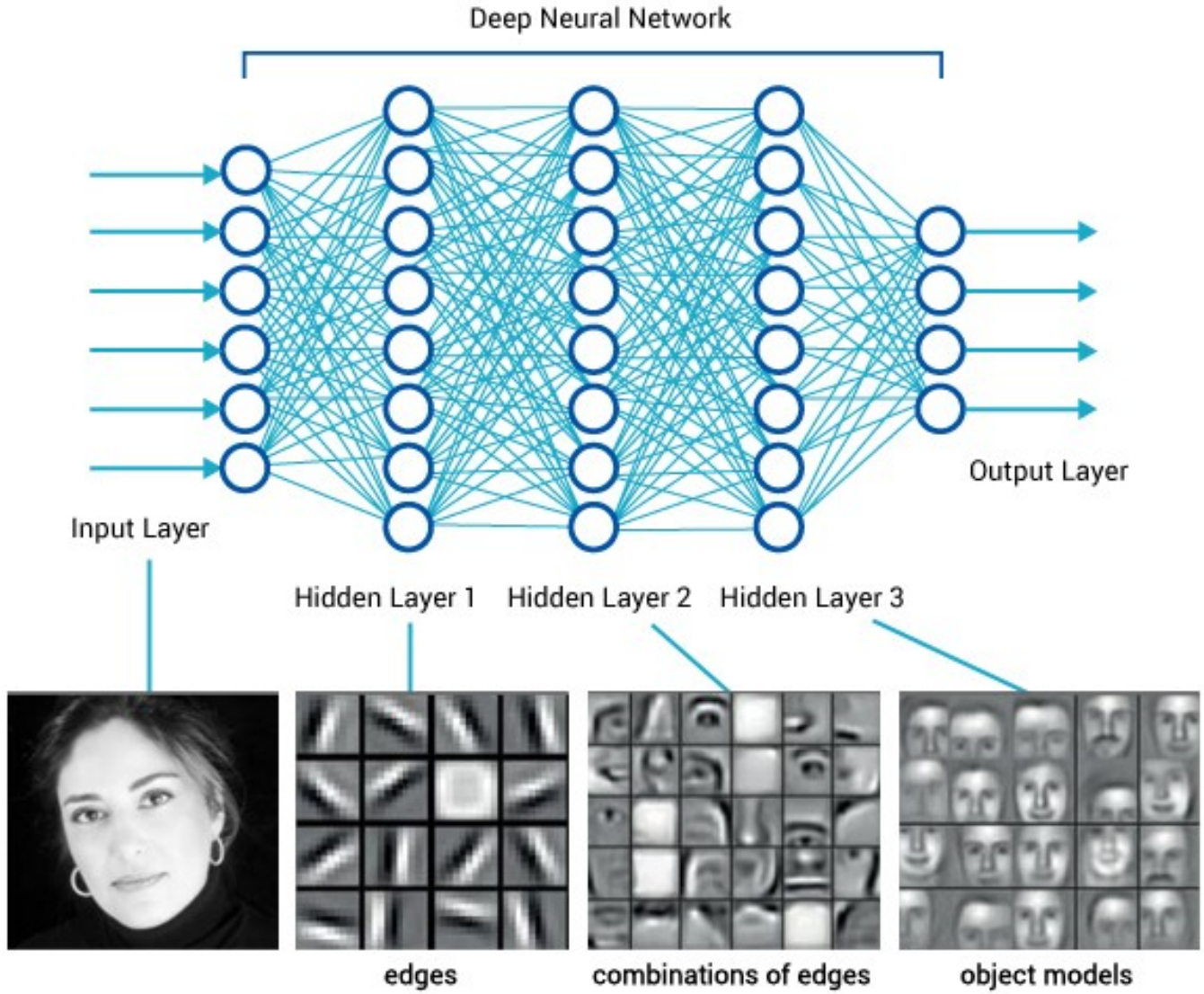
- Deep NN found the features allowing to recognize the Higgs boson better, than the skilled physicists with their whole knowledge.
- This might allow in the future to automatize the physics analysis….
- Does it mean unemployment for us?
- Problem: We must trust the Monte Carlo...

Deep learning for pattern recognition (a reminder)

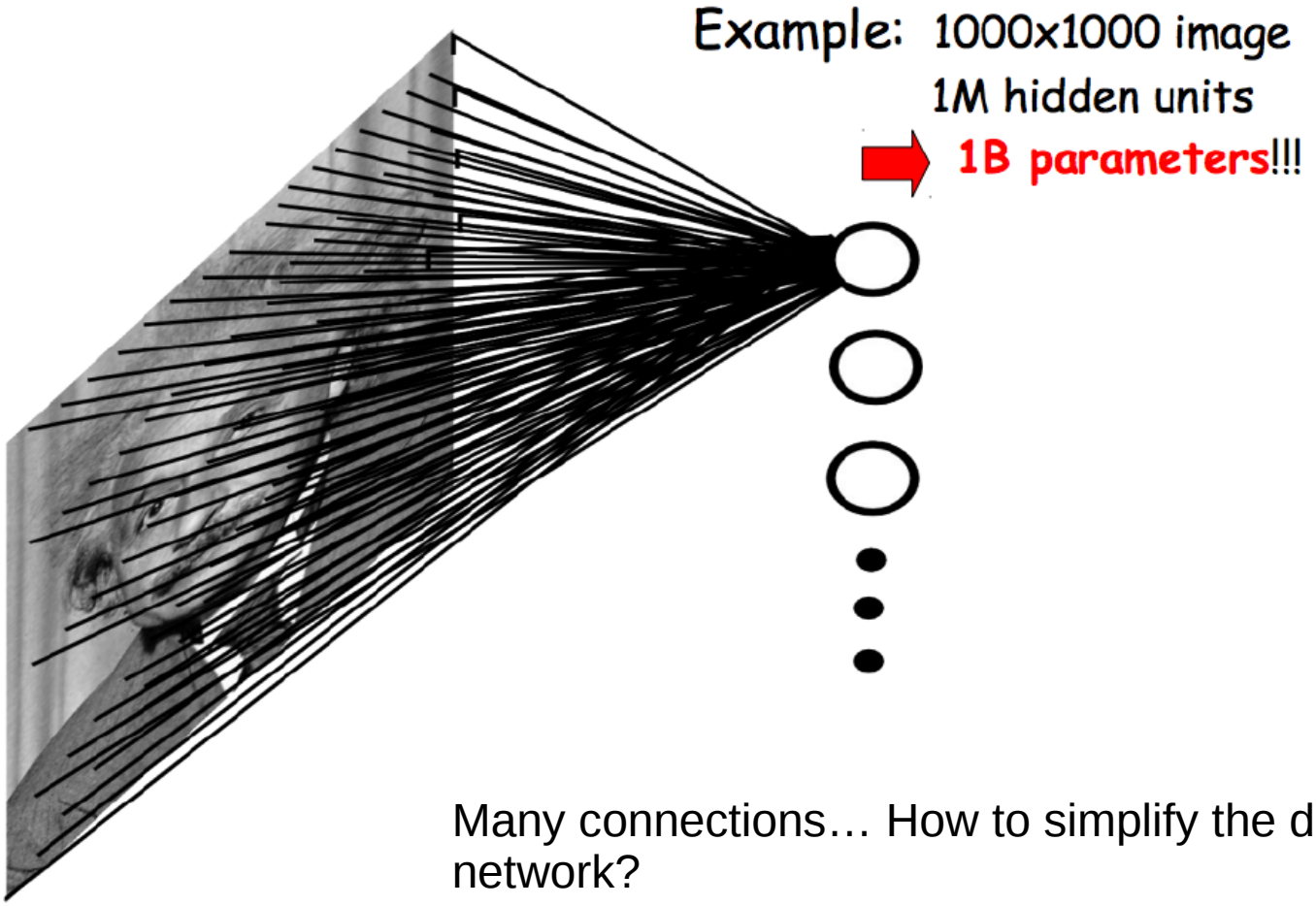
Individual layers are trained to recognize the “features” - from simple to more complex.



Deep Neural Network works like that...

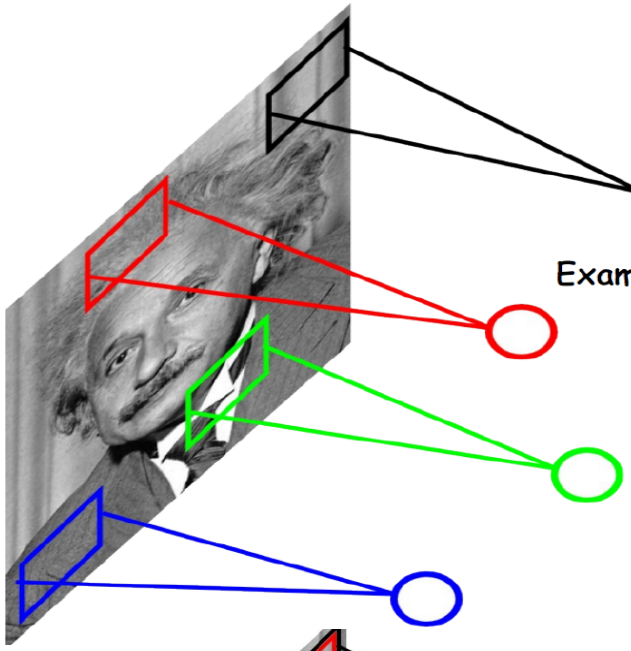


Convolutional NN
Pattern recognition

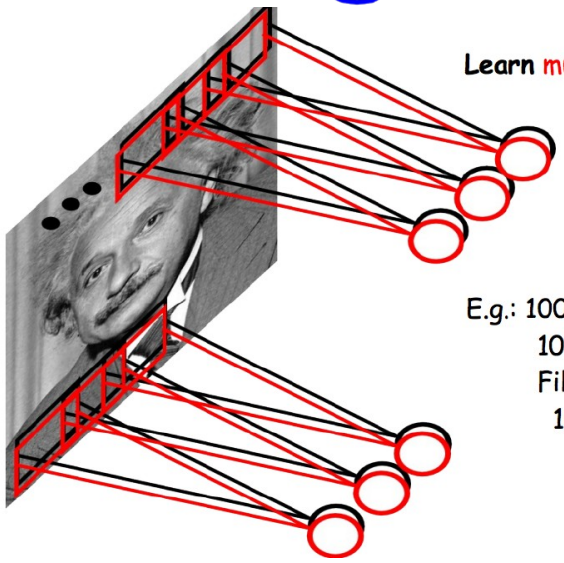


Many connections… How to simplify the deep neural network?

Convolutional NN



Example: 1000x1000 image
1M hidden units
Filter size: 10x10
10M parameters



Learn multiple filters.

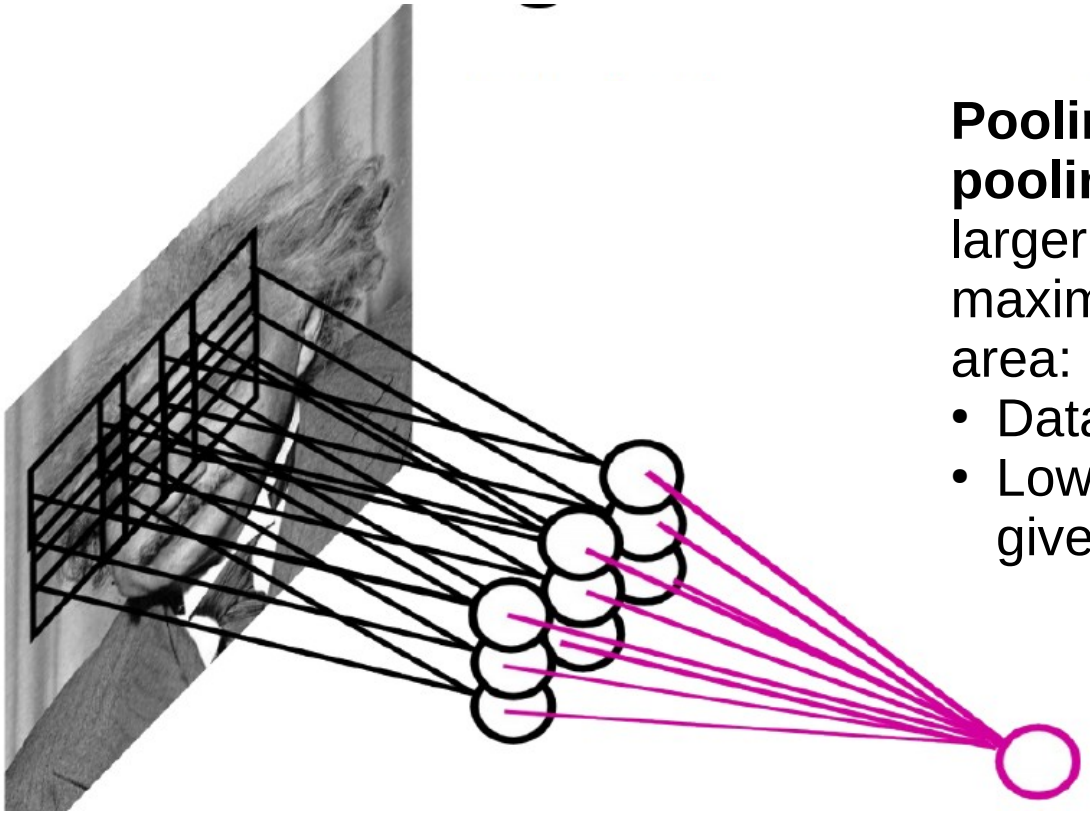
E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

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.

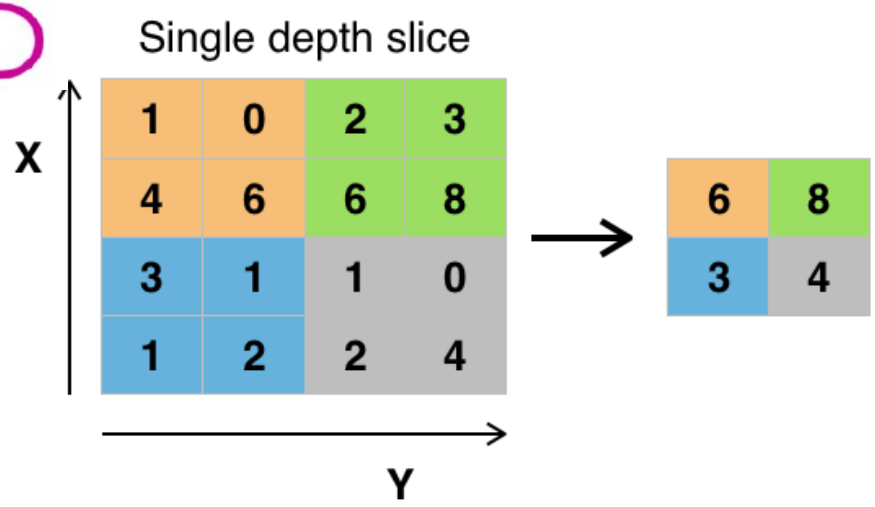
LeCun et al. "Gradient-based learning applied to document recognition" IEEE 1998

Pooling



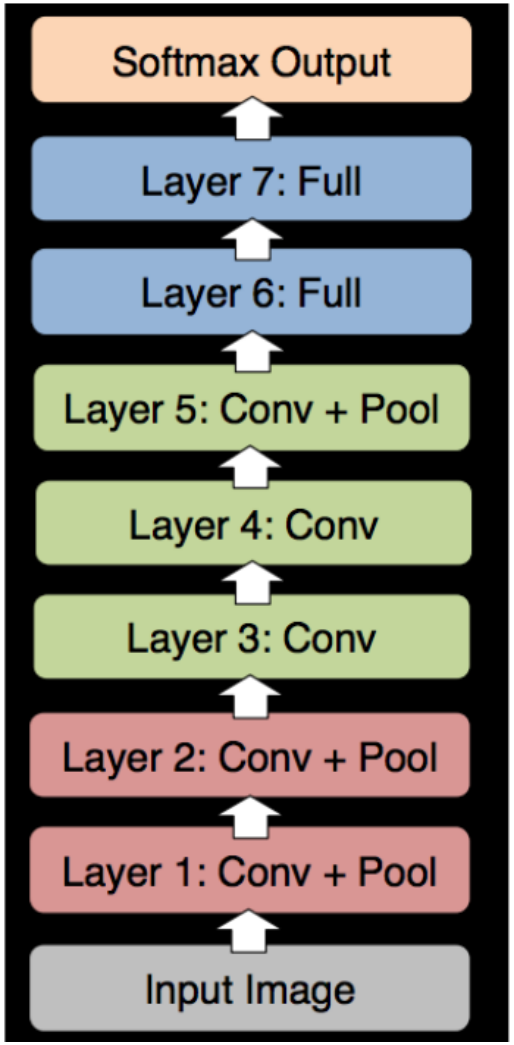
Pooling – (in most cases **max pooling**) the group of outputs for a larger input area is replaced by a maximum (or average) for this given area:

- 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.

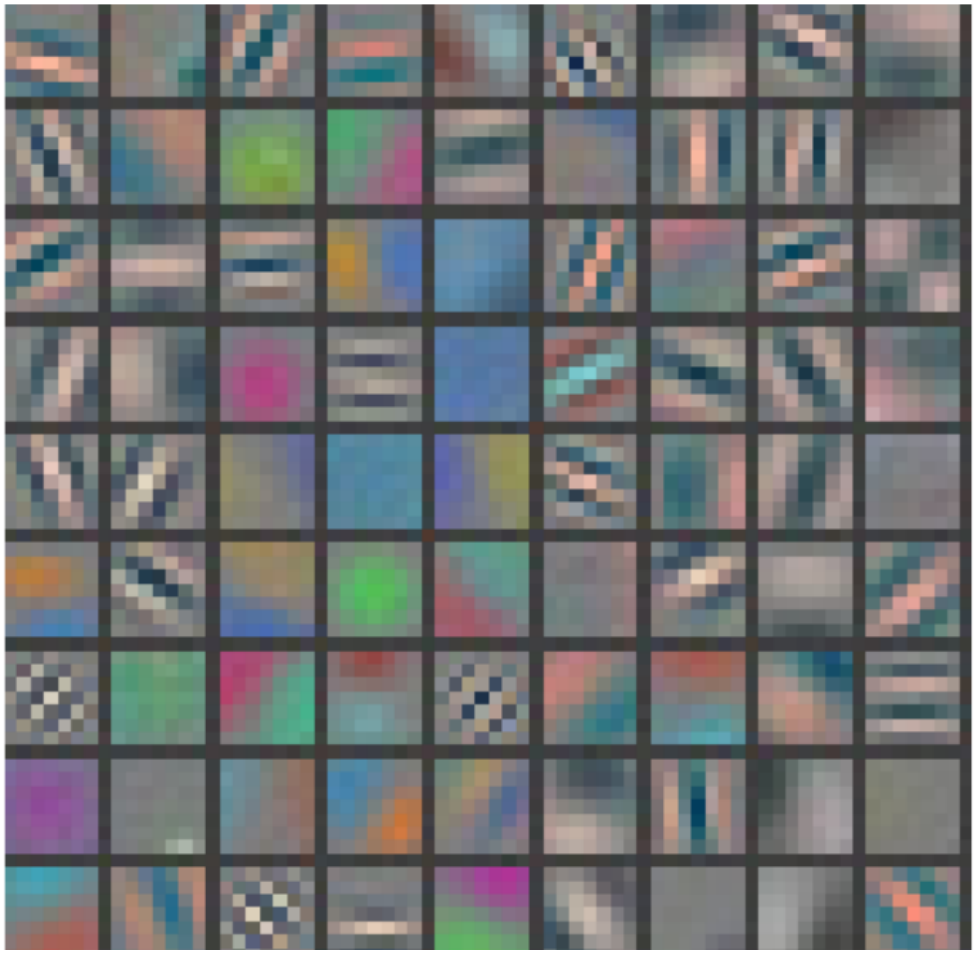


Slide: R. Fergus

First layer filters

Showing 81 filters of 11x11x3.

Capture low-level features like oriented edges, blobs.



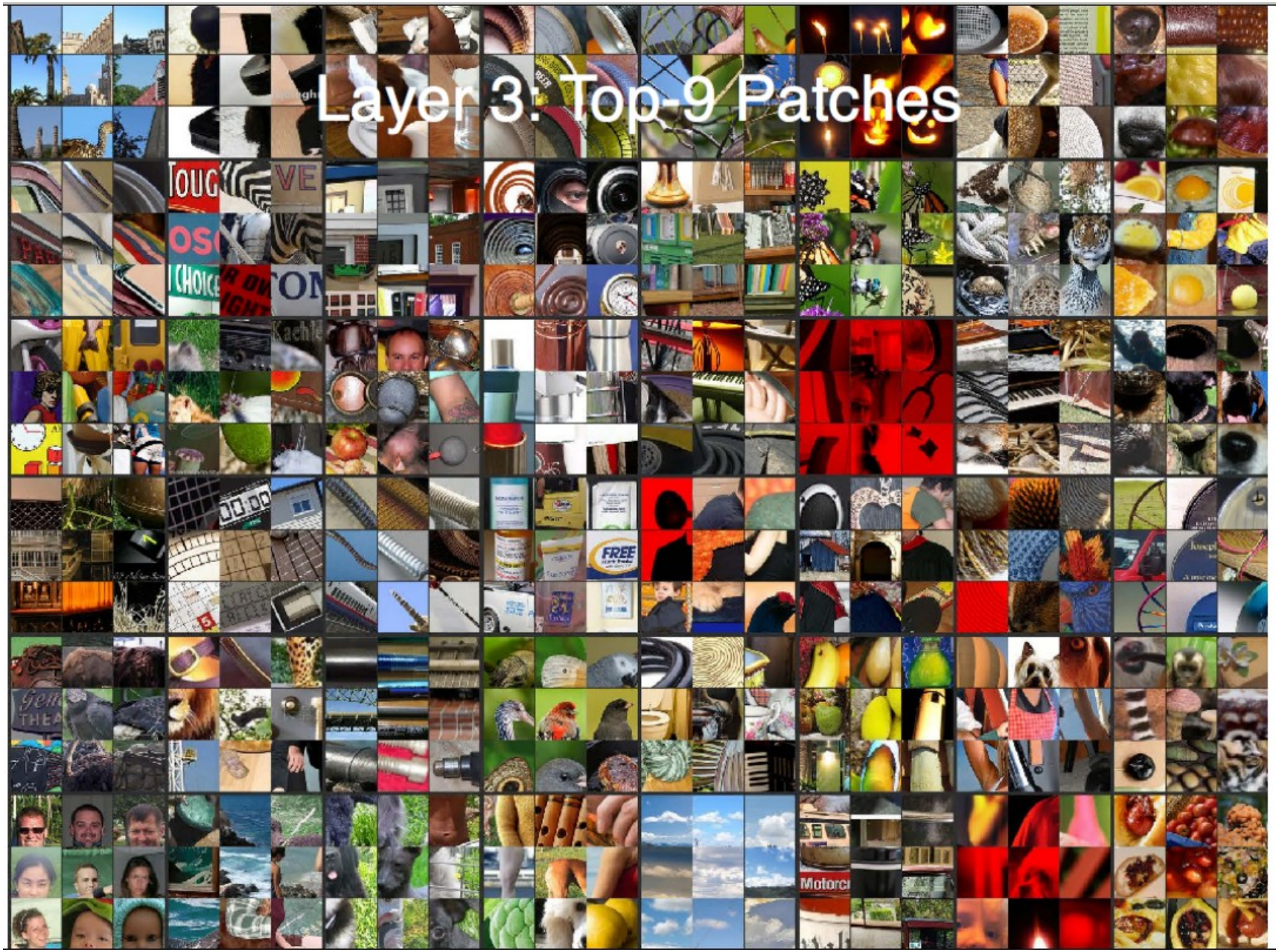
Note these oriented edges are analogous to what SIFT uses to compute the gradients.

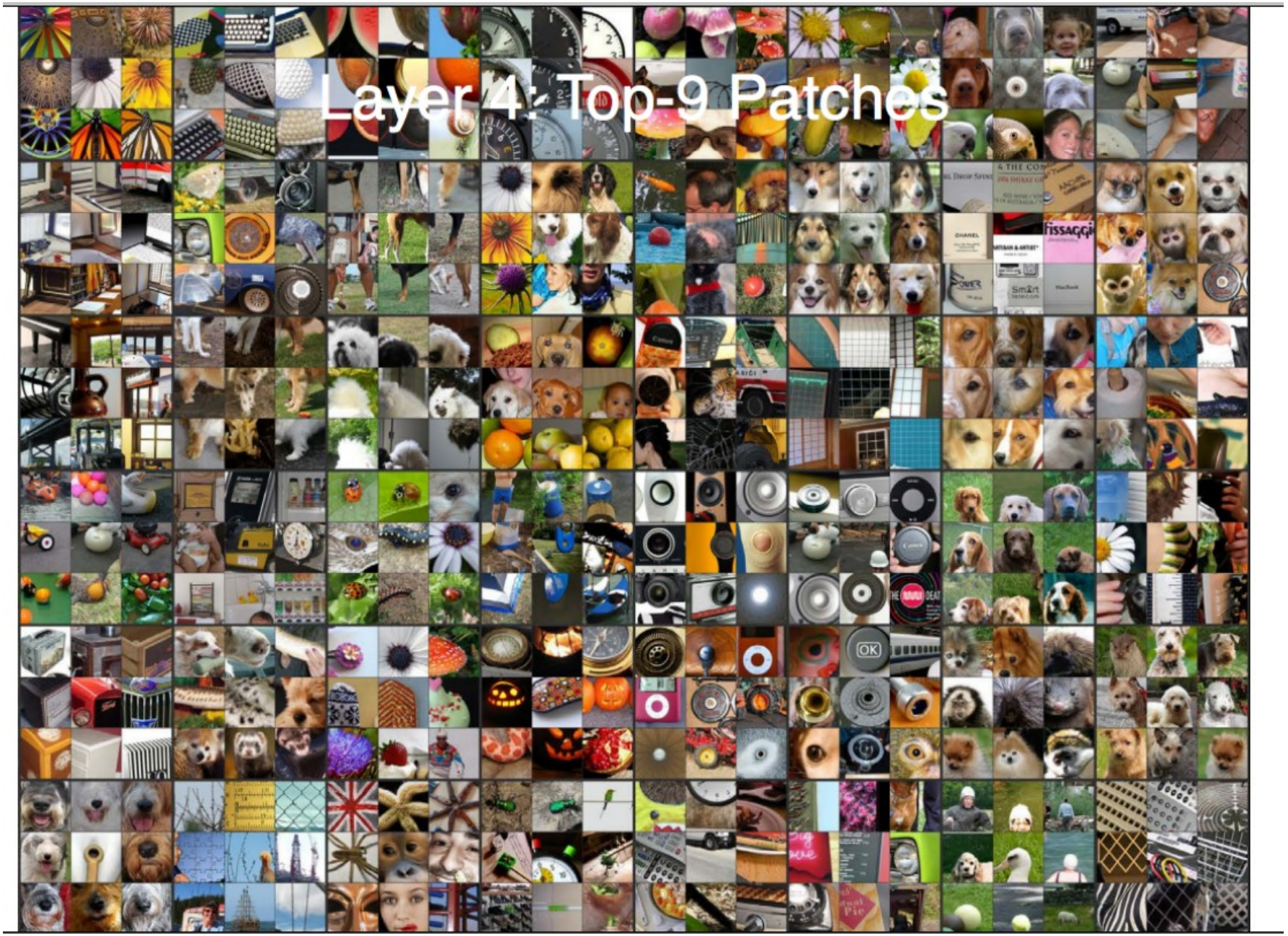
SIFT - scale-invariant feature transform, algorithm published in 1999 roku by David Lowe.

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

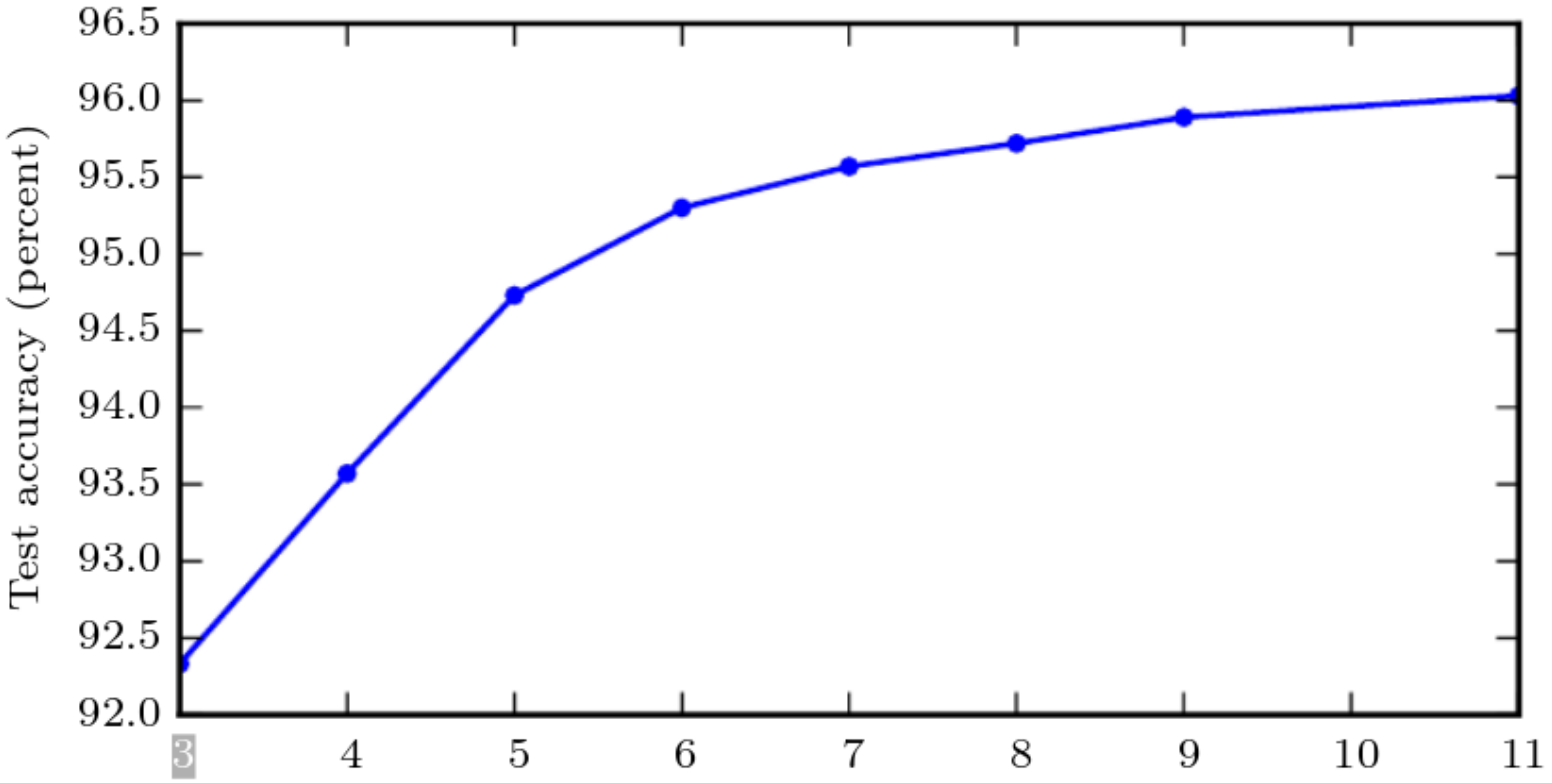


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from Goodfellow *et al.* (2014d). The test set accuracy consistently increases with increasing depth. See figure 6.7 for a control experiment demonstrating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

http://www.deeplearningbook.org

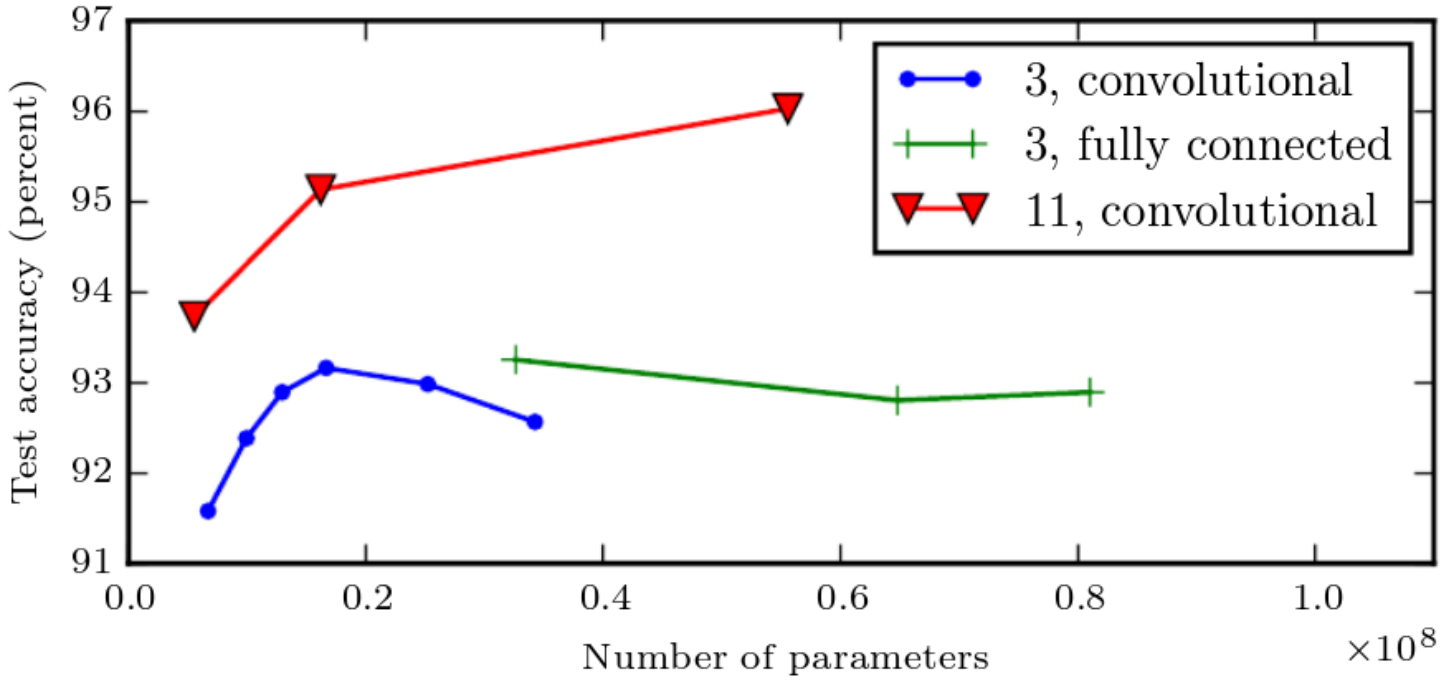
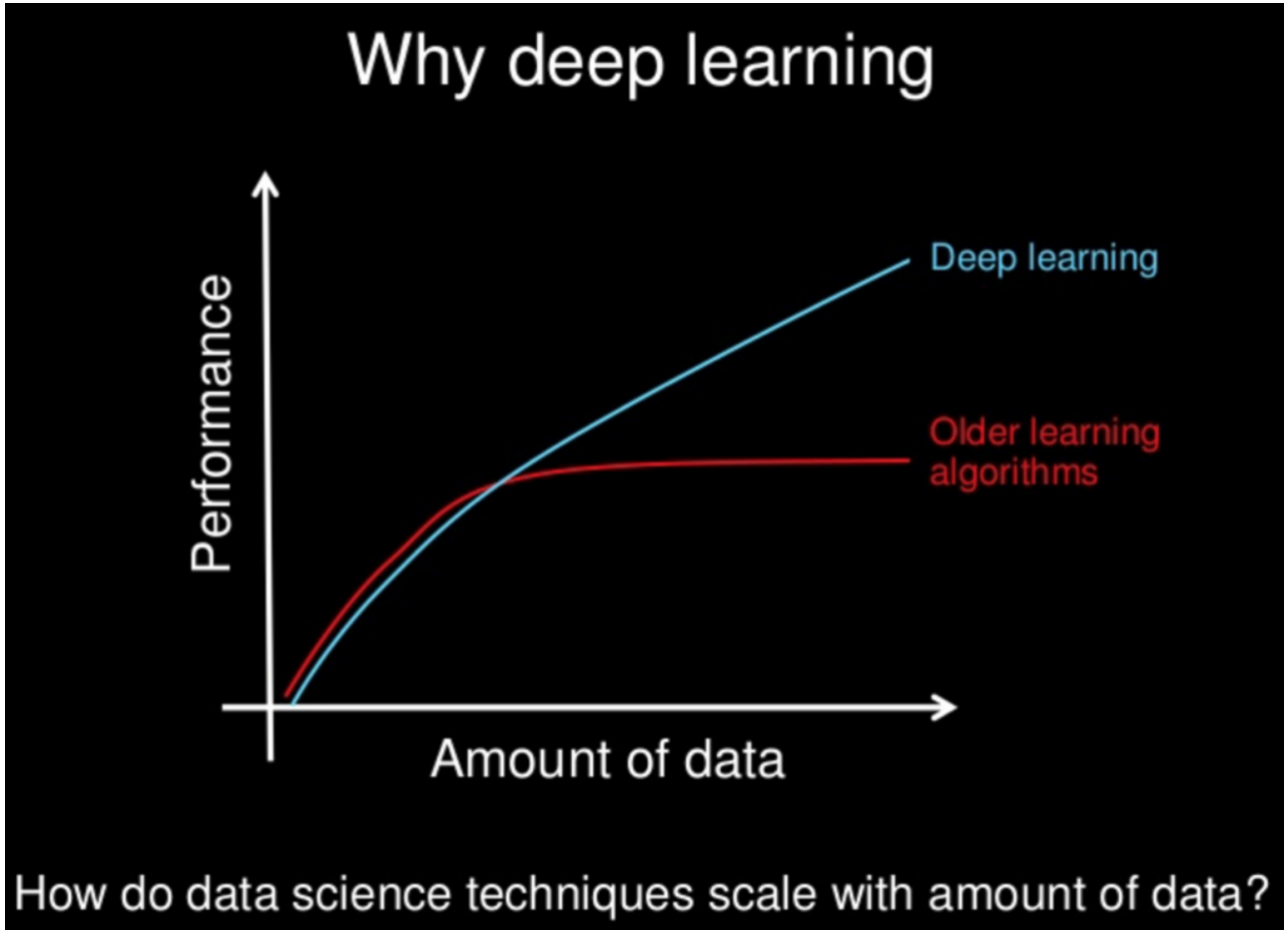


Figure 6.7: Deeper models tend to perform better. This is not merely because the model is larger. This experiment from Goodfellow *et al.* (2014d) shows that increasing the number of parameters in layers of convolutional networks without increasing their depth is not nearly as effective at increasing 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).

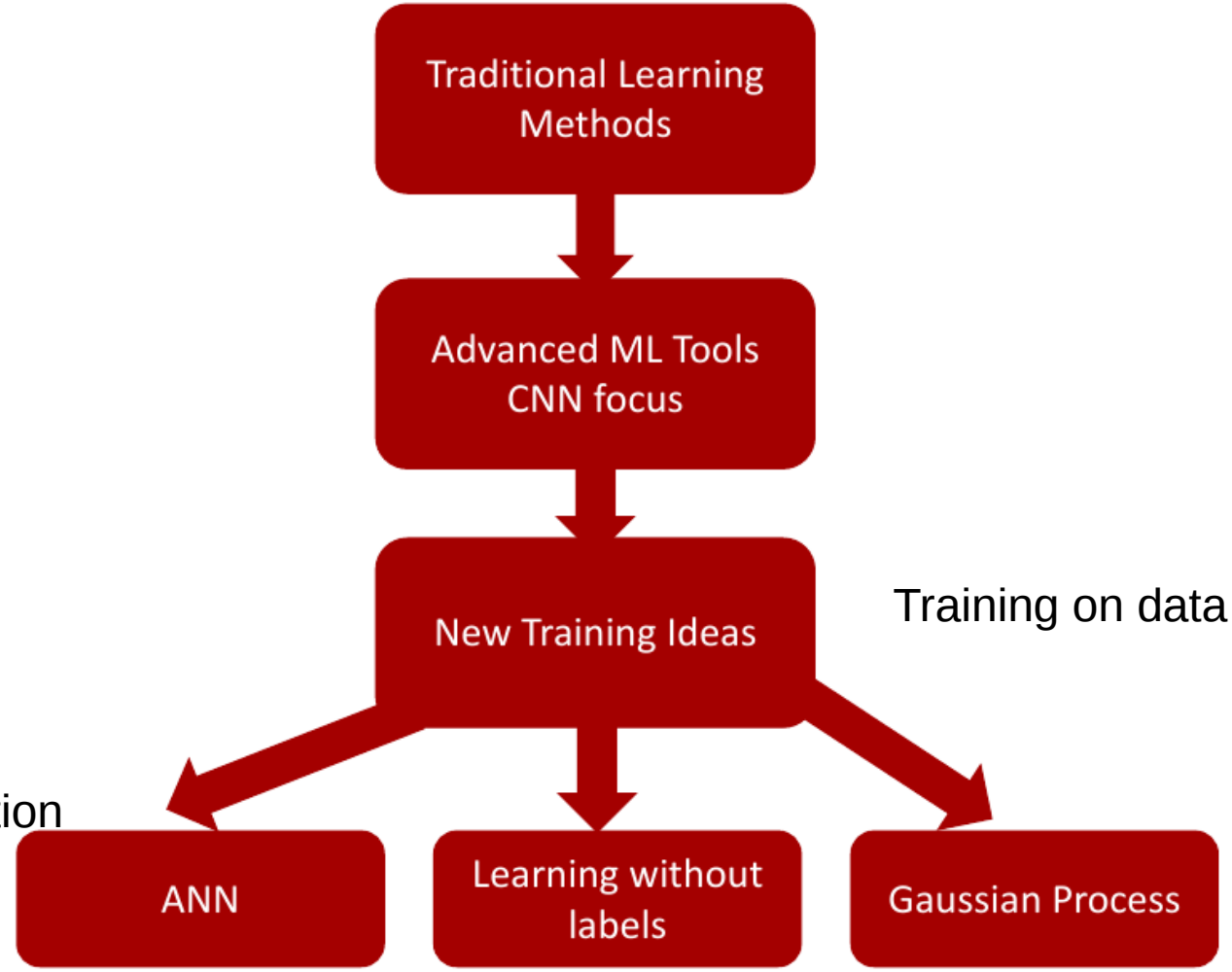
http://www.deeplearningbook.org

Deep learning



- Scales up for huge amount of inputs

Road Map
from Zihao Jiang presentation

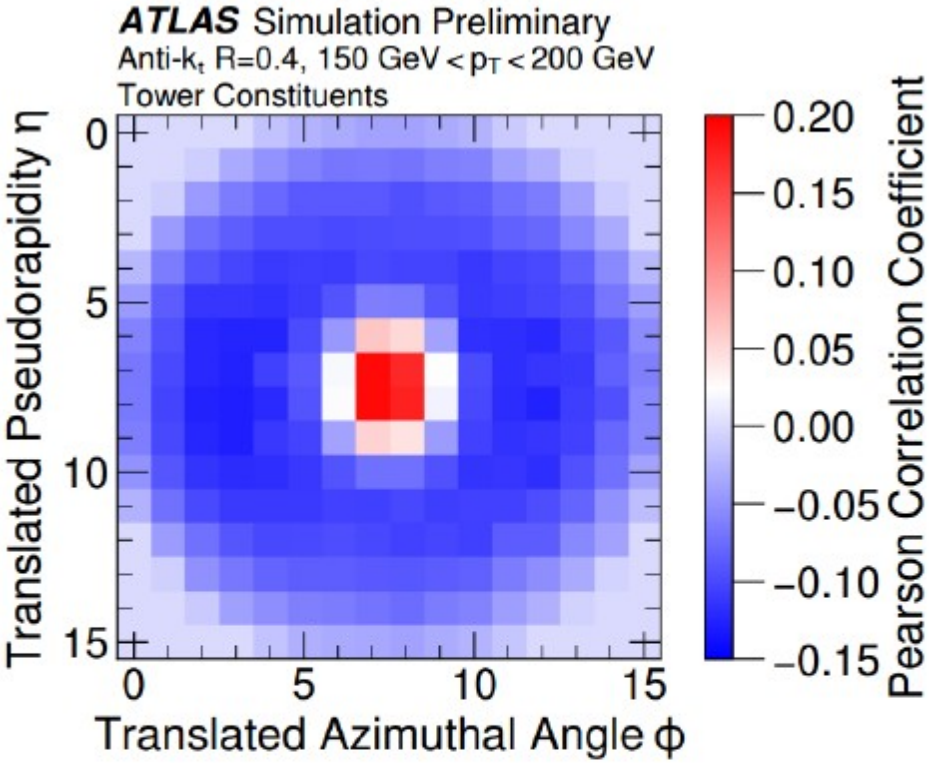


Training on data (not MC!)

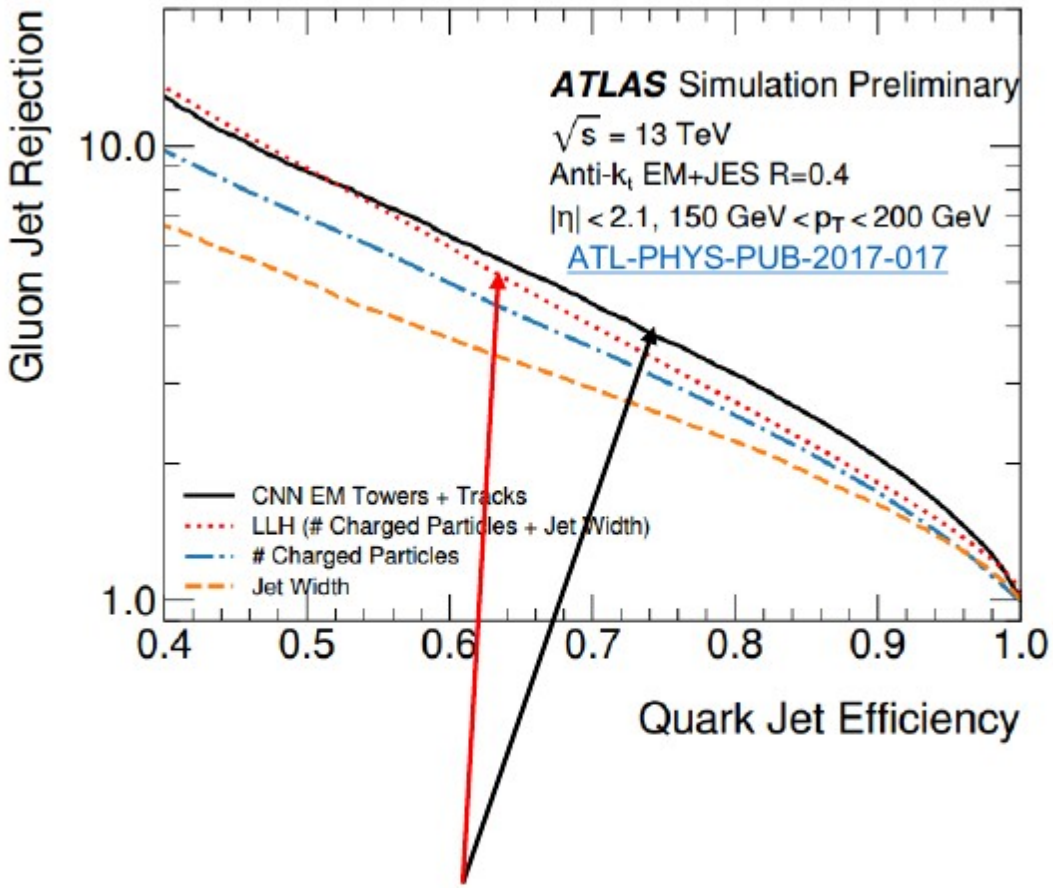
Punish for correlation with some, poorly modeled variables

2017/12/4 Zihao Jiang 3

CNNs

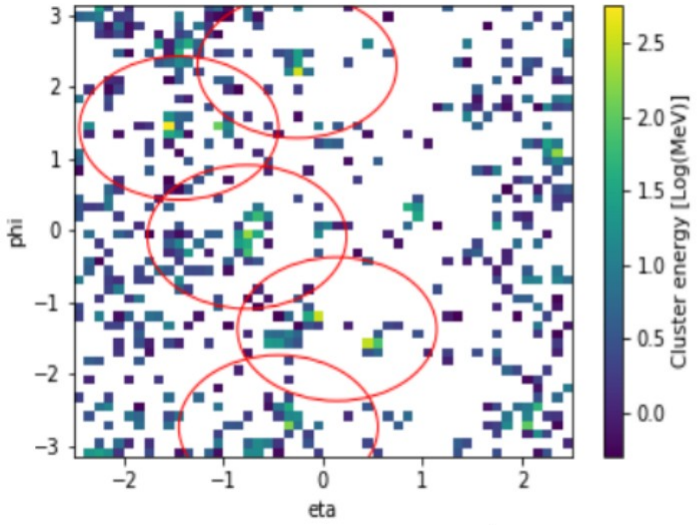


Per pixel correlation between image intensity and CNN output.
The four pixels at the core is highly correlated with jet being a quark jet



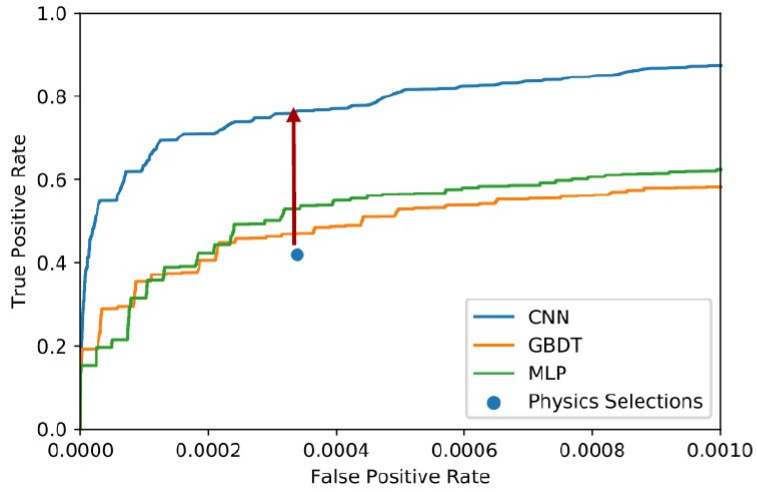
CNN as an entirely different approach than building likelihood from high level quantities show improvement of quark vs. gluon classification

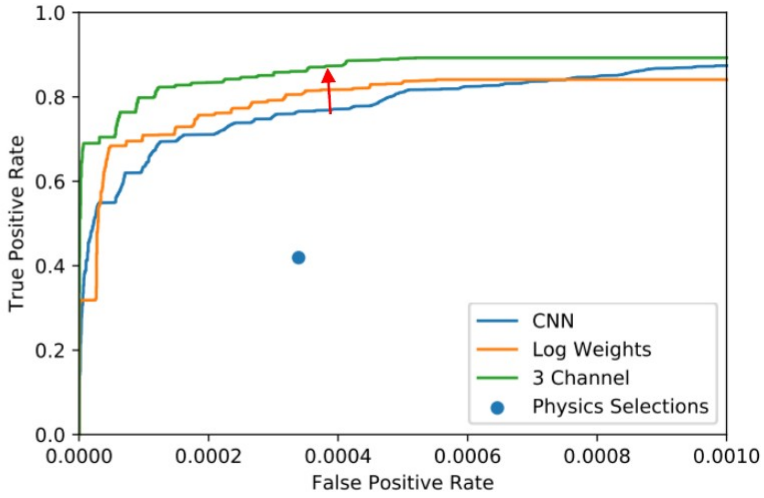
CNNs



Event Classification:
Search for RPV SUSY gluino decays

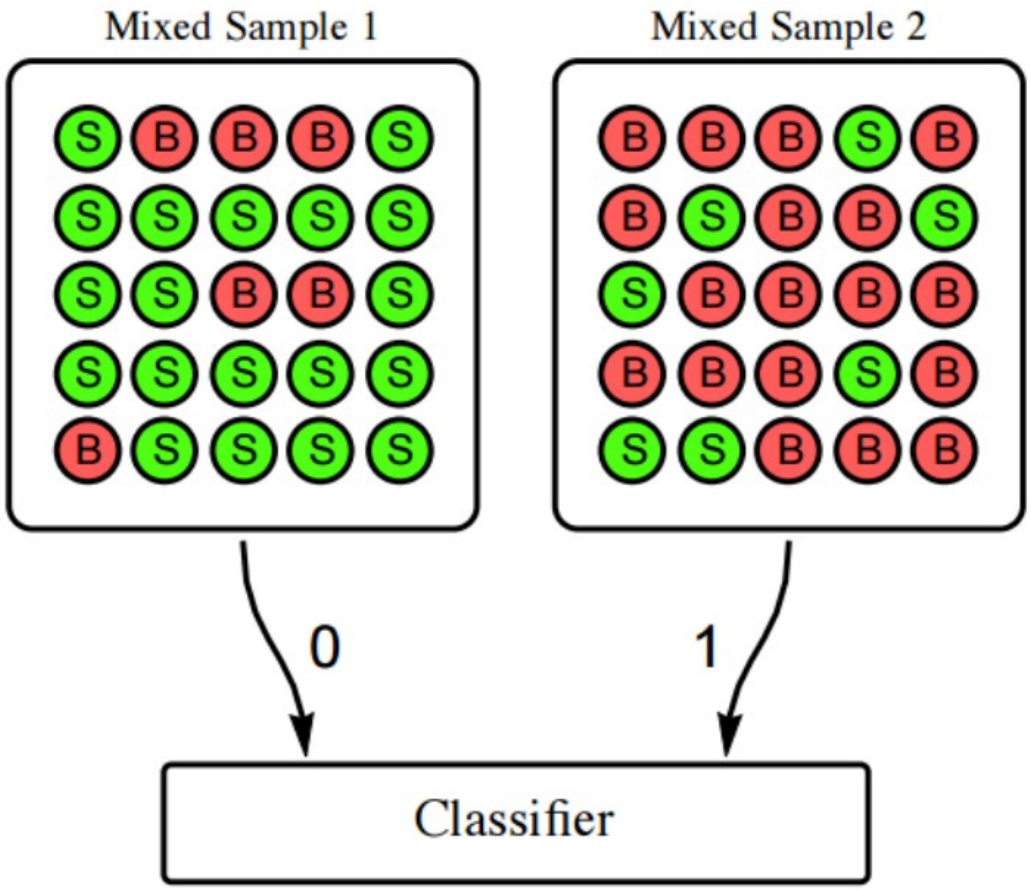
- Image content = calo tower energy
- CNN outperforms typical classifiers
- 3-channel image (Ecal, Hcal, Track) further boosts the performance
- NOTE: soft activity systematics missing



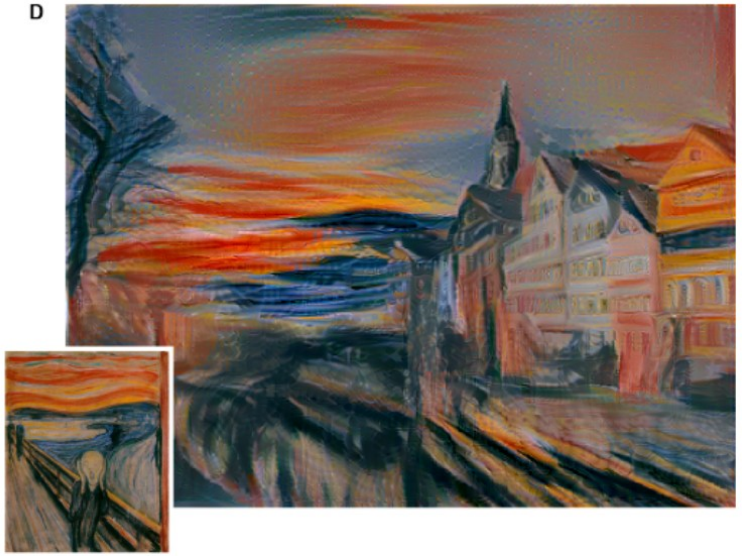


Learning from Data
--Classification w/o Labeling

- 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



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).

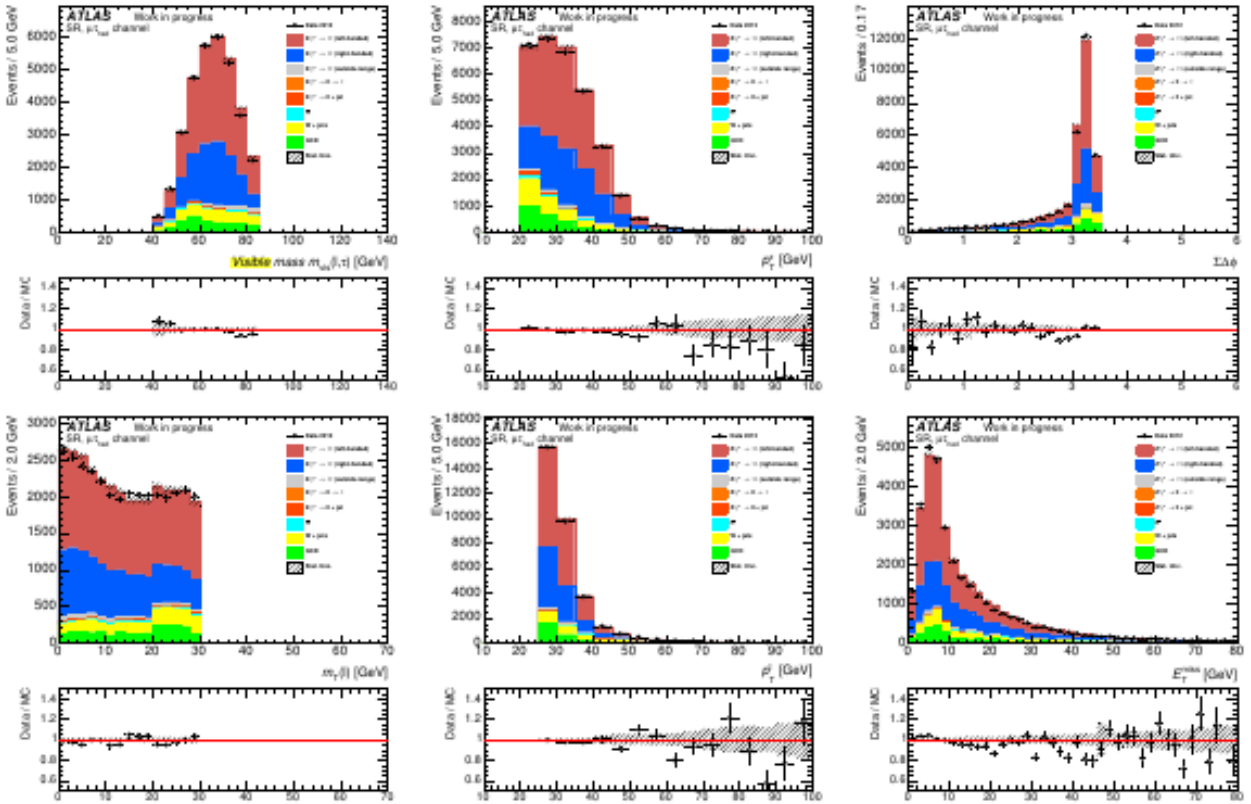


Figure 41: Distributions of variables observed in $Z \to \tau\tau$ (μ-had channel). From top-left: visible mass of τ-lepton system, τ transverse momentum, sum of polar angles between τ and missing-E_T and between lepton and missing-E_T, transverse mass of the lepton-missing-E_T system, lepton transverse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