DEEP LEARNING – PART 1

Verena Kaynig-Fittkau
Ray (Thouis) Jones

ComputeFest 2016

Check This (Repository) Out

- https://github.com/vkaynig/IACS_ComputeFest_DeepLearning

- Or go to github and search for IACS_ComputeFest

What We Cover Today:

- Basic theory of deep neural networks (DNN)
- Introduction to Theano
- Training deep neural networks with Theano

- Some useful tips and tricks for training DNNs

Workshop Goal

- Flatten the learning curve for the Theano tutorials
- http://deeplearning.net/tutorial/

Deep Learning

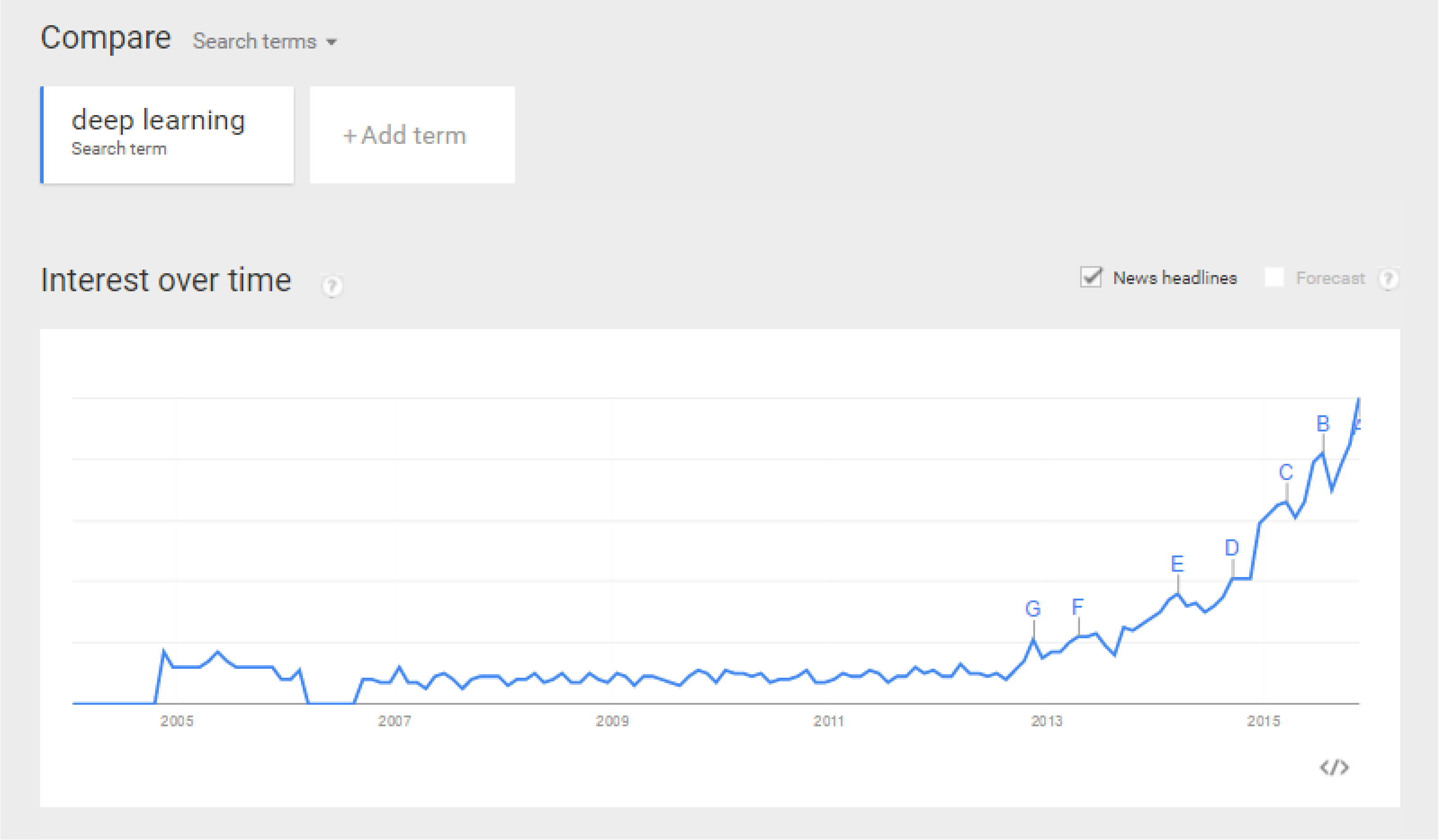
Why do you want to know about deep learning?

Motivation

- It works!
- State of the art in machine learning
- Google, Facebook, Twitter, Microsoft are all using it.

- It is fun!
- Need to know what you are doing to do it well.

Google Trends



Google Brain - 2012

16 000

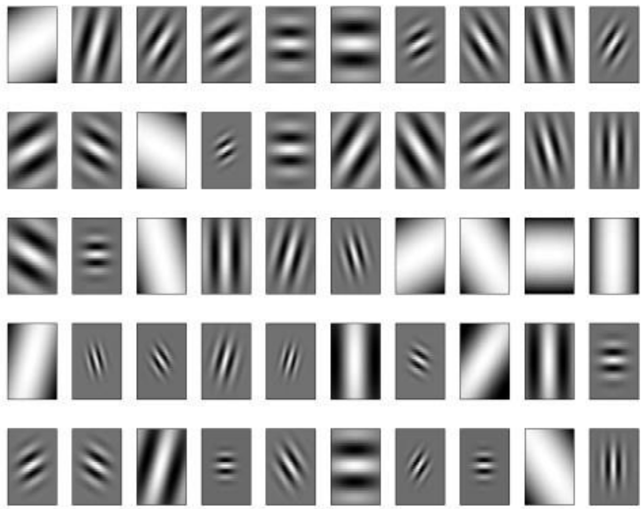
What it learned

http://www.nytimes.com/2012/06/26/technology/in-a-big-network-of-computers-evidence-of-machine-learning.html?pagewanted=all

Manual Feature Design

Yes **No**

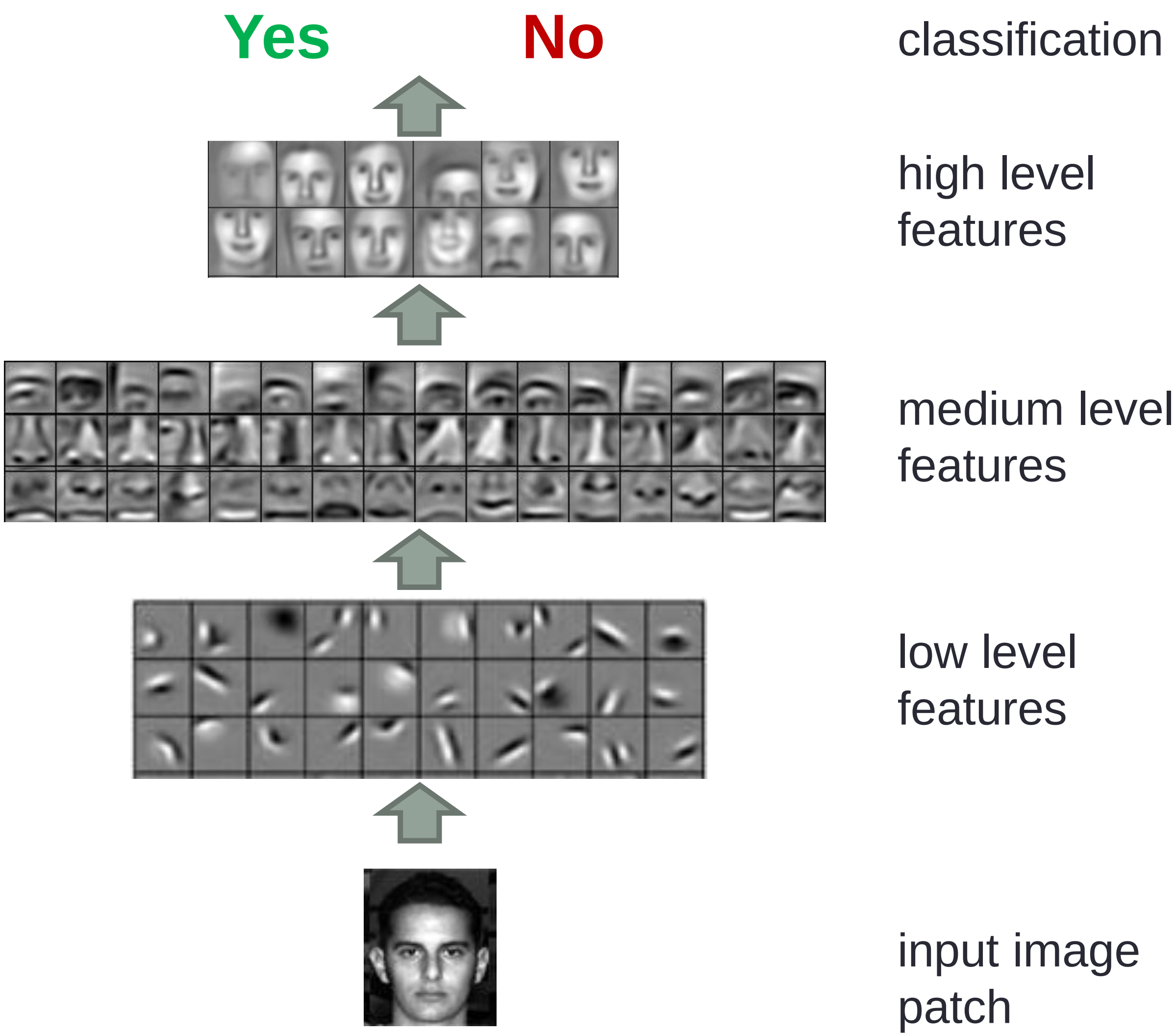
classification



hand designed features

input image patch

Learned Feature Hierarchy



[Honglak Lee]

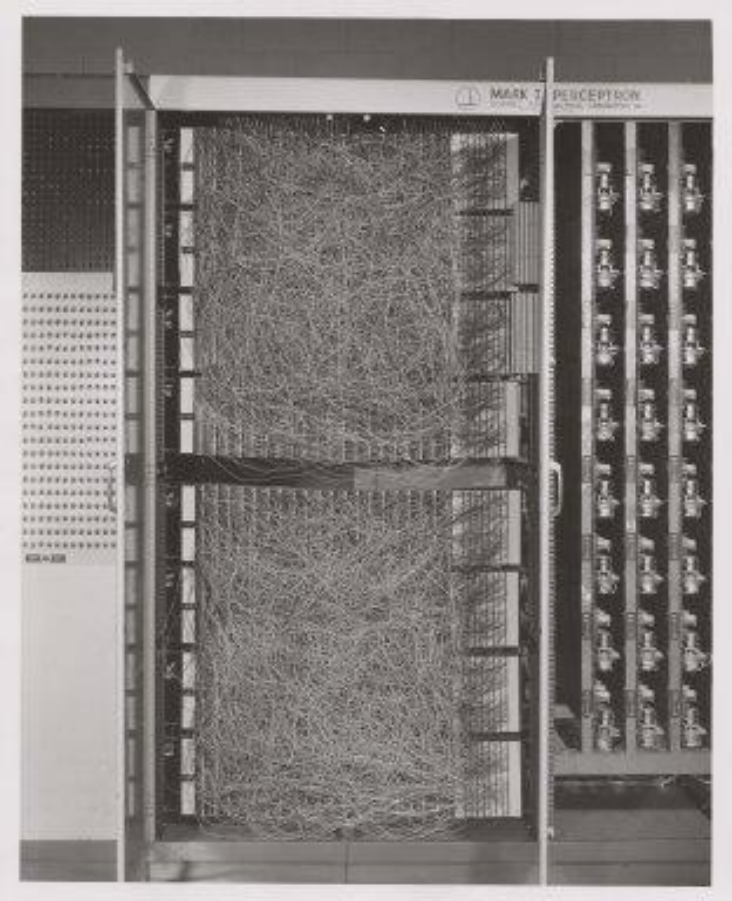
Deep Learning Techniques

- Artifical neural network
  - Introduced in the 60s
  - Perceptron in the old days

- Convolutional neural network
  - Introduced in the 80s

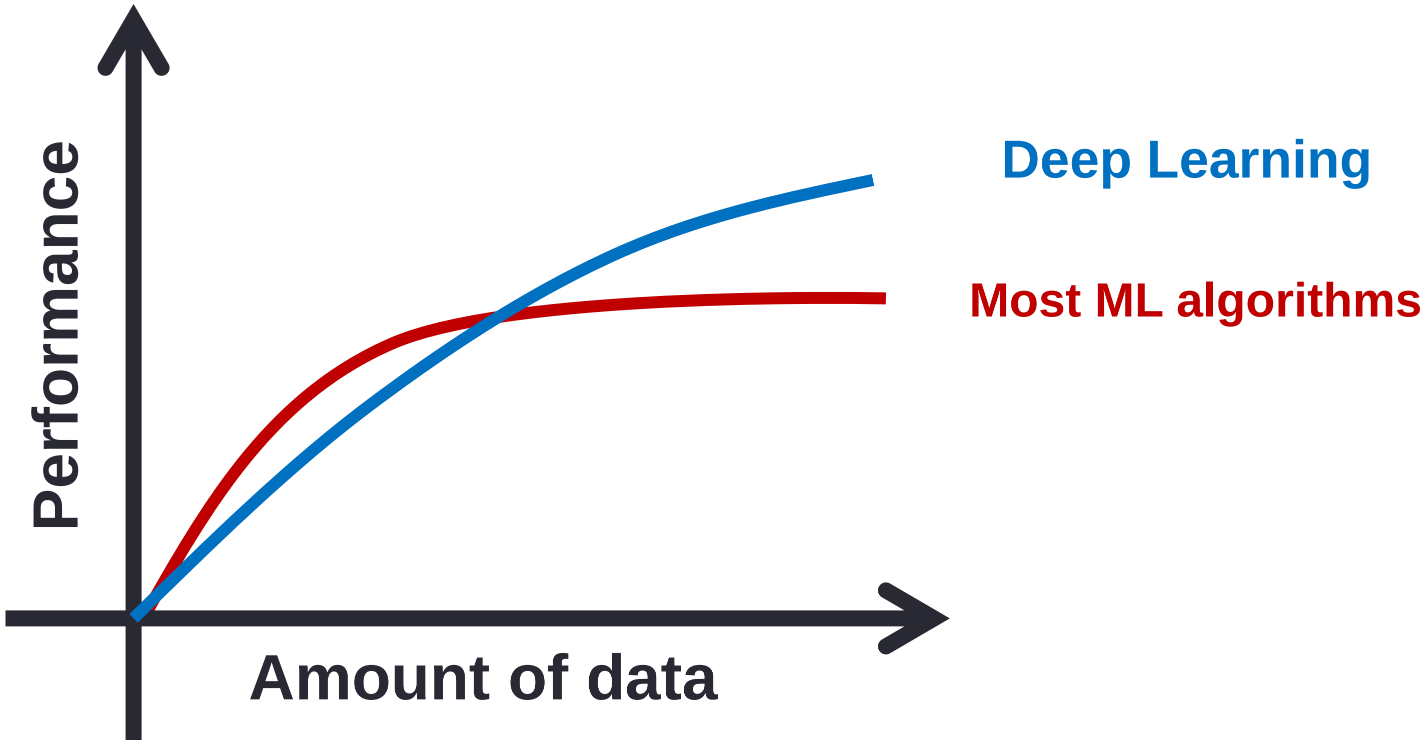
- Recurrent neural network
  - Introduced in the 80s

What Changed -Computational Power



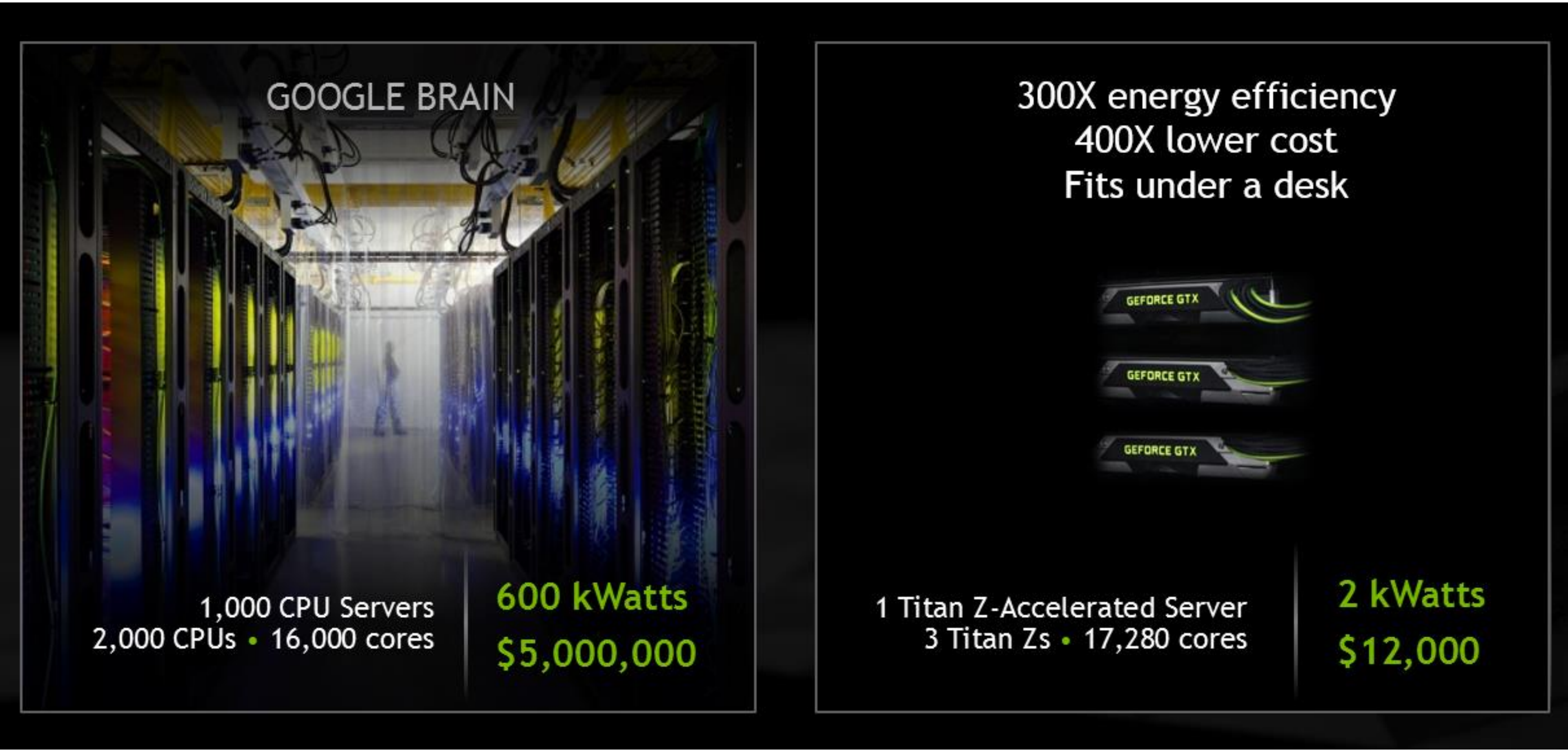
What Changed – Data Size

Scaling with Data Size



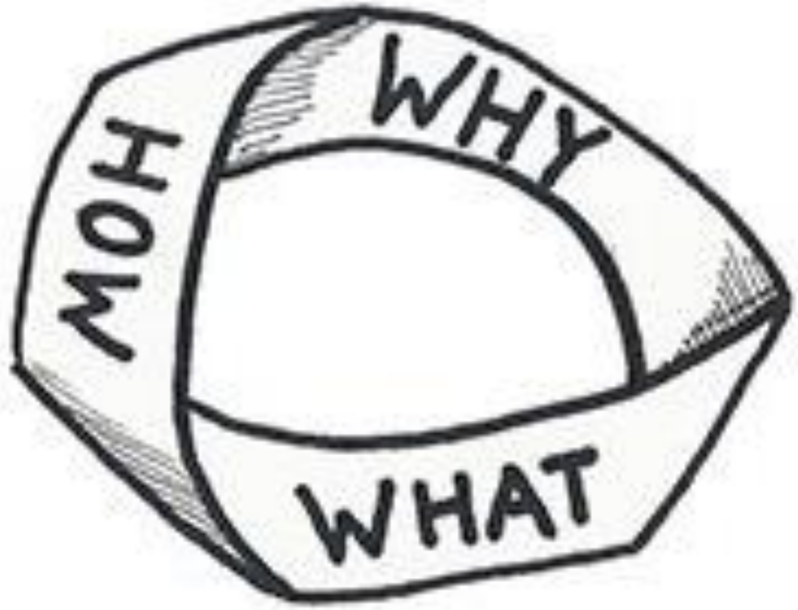
[Andrew Ng]

I don’t Have a Cluster at Home



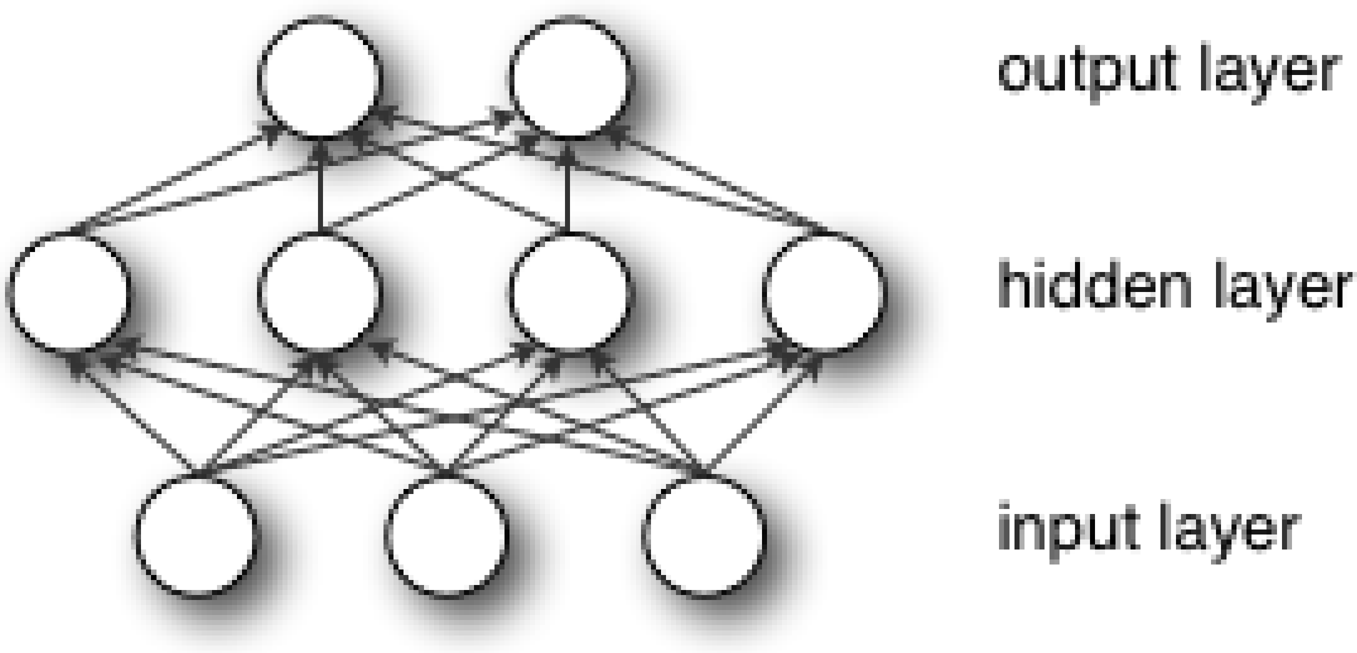
http://www.geforce.com/whats-new/articles/introducing-nvidia-geforce-gtx-titan-z

Deep Learning

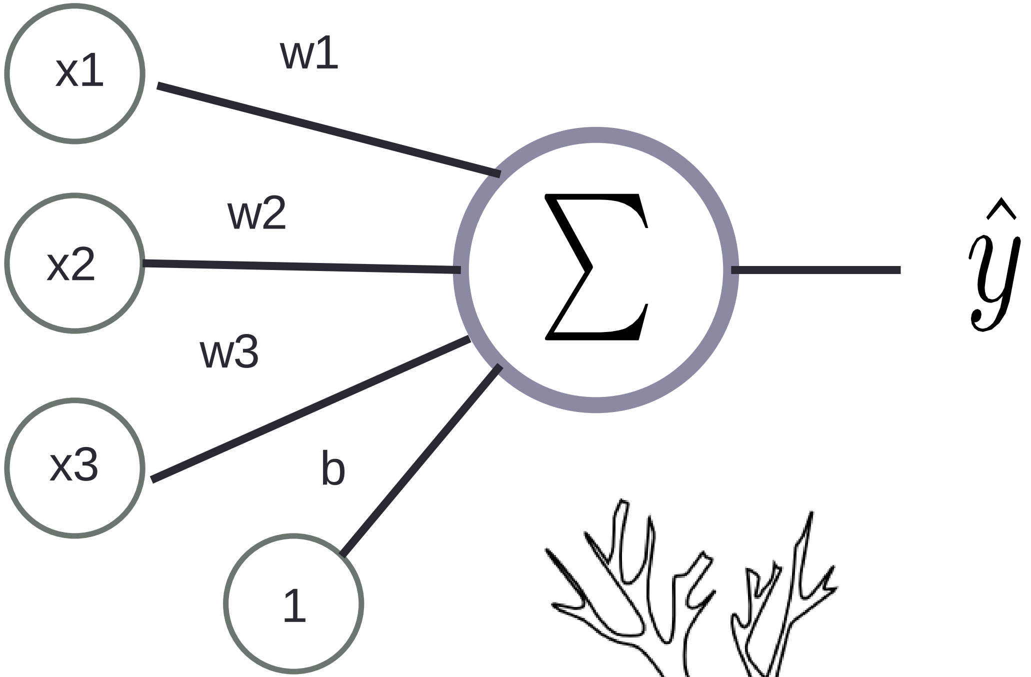


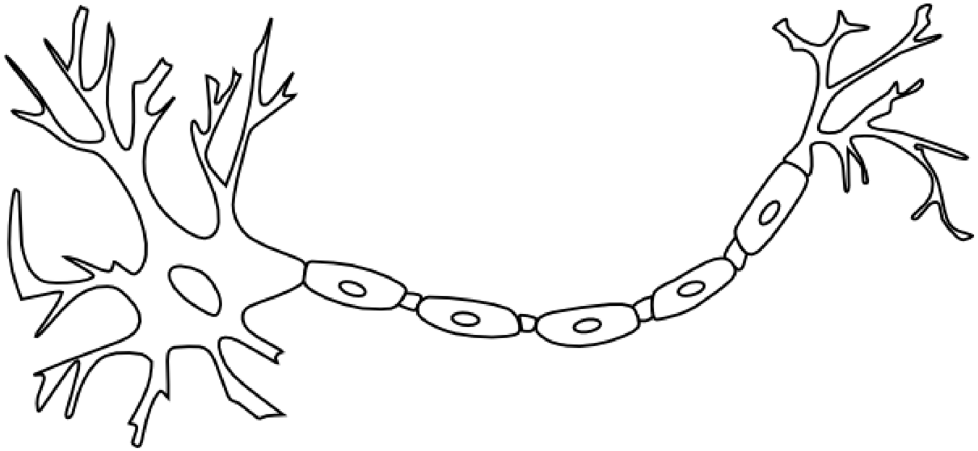
What is deep learning?

What – Neural Network Architecture

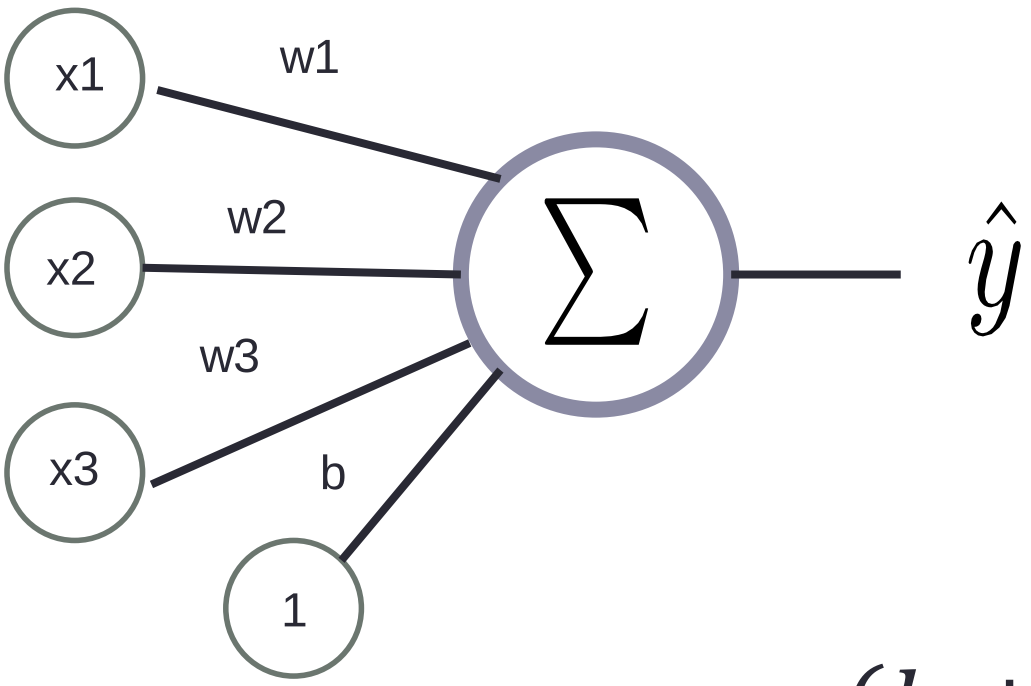


Artificial Neuron



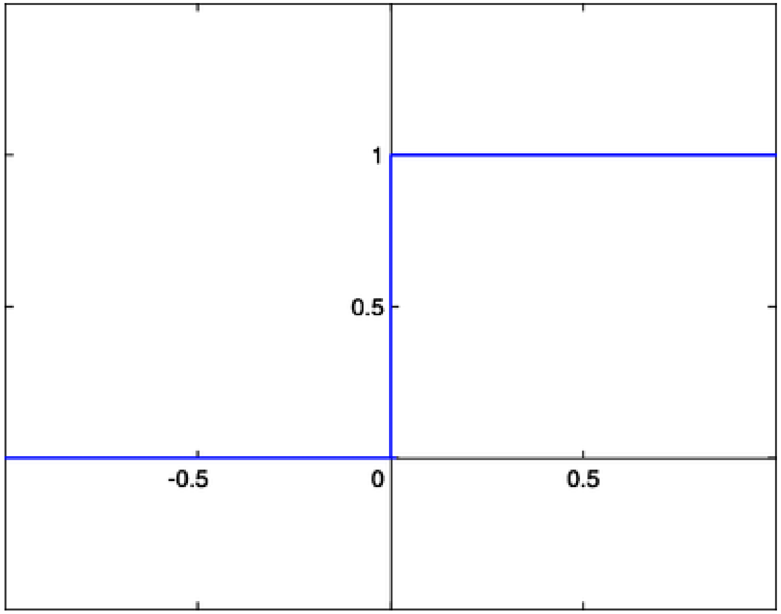


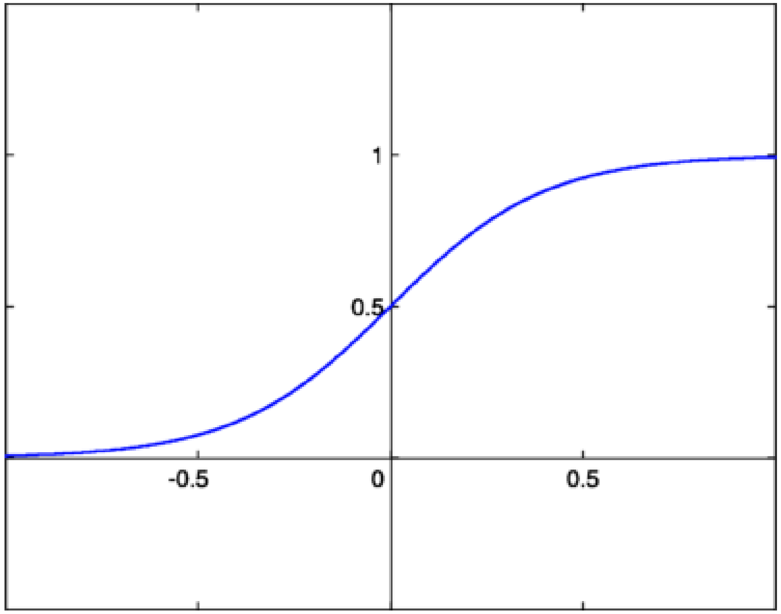
Artificial Neuron



$$s(b + w^T x)$$

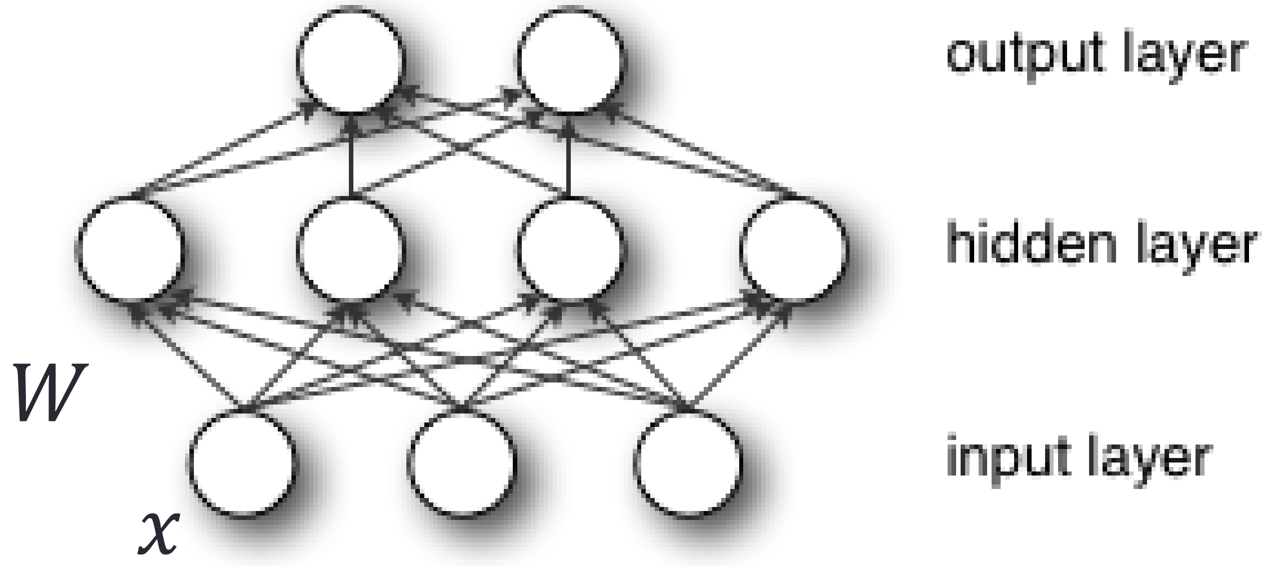
Step vs Sigmoid Activation





$$s(x) = \frac{1}{1 + e^{-cx}}$$

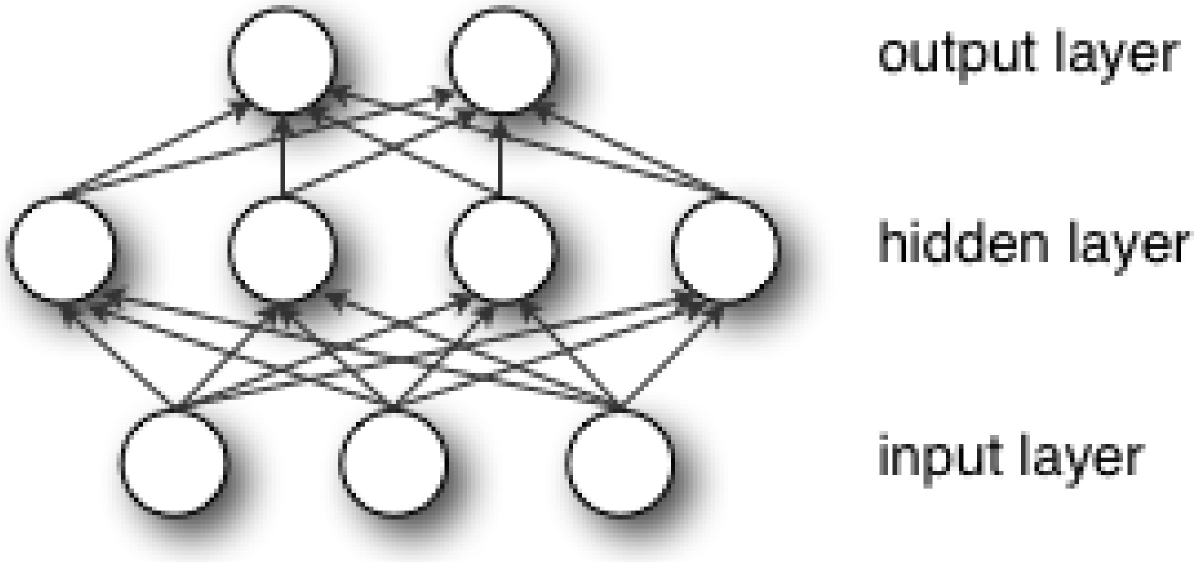
Multi-Layer Perceptron



$$s\left(b^{(1)} + W^{(1)}x\right)$$

http://deeplearning.net/tutorial/mlp.html

Multi-Layer Perceptron



$$f(x) = G\left(b^{(2)} + W^{(2)}\left(s\left(b^{(1)} + W^{(1)}x\right)\right)\right)$$

G: logistic function, softmax for multiclass

http://deeplearning.net/tutorial/mlp.html

Logistic Regression (Multiclass)

- Probabilistic classifier

$$P(Y = i|x, W, b) = softmax_i(Wx + b)$$
$$= \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}}$$

$$y_{pred} = \mathrm{argmax}_i P(Y = i|x, W, b)$$

How - Optimization

- As so many machine learning methods, also DNNs are about optimizing a loss function:

training samples

regularizer

$$\underbrace{\arg\min_{\boldsymbol{\theta}}}_{\text{model parameter}} \frac{1}{T} \sum_t l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)}) + \lambda\Omega(\boldsymbol{\theta})$$

training labels

model parameter

Loss Function

- Classification error would be good, but it's not smooth
- Need to find smooth proxy

- Last layer is logistic regression
- The whole network estimates class probabilities

$$P(Y = i|x, W, b) = softmax_i(Wx + b)$$

- Maximize the correct class assignment probabilities in the training set

Minimize Negative Log Likelihood

- Maximize the likelihood of the data set given the model parameters
- Take the log to simplify for numerical stability and math simplicity
- Take negative log likelihood to cast into minimization

training data

$$\mathcal{L}(\theta, \mathcal{D}) = \sum_{i=0}^{|\mathcal{D}|} \log P(Y = y^{(i)} | x^{(i)}, \theta)$$

Parameter W, b

Stochastic Gradient Decent (SGD)

- Take small steps downward on the surface of the loss function

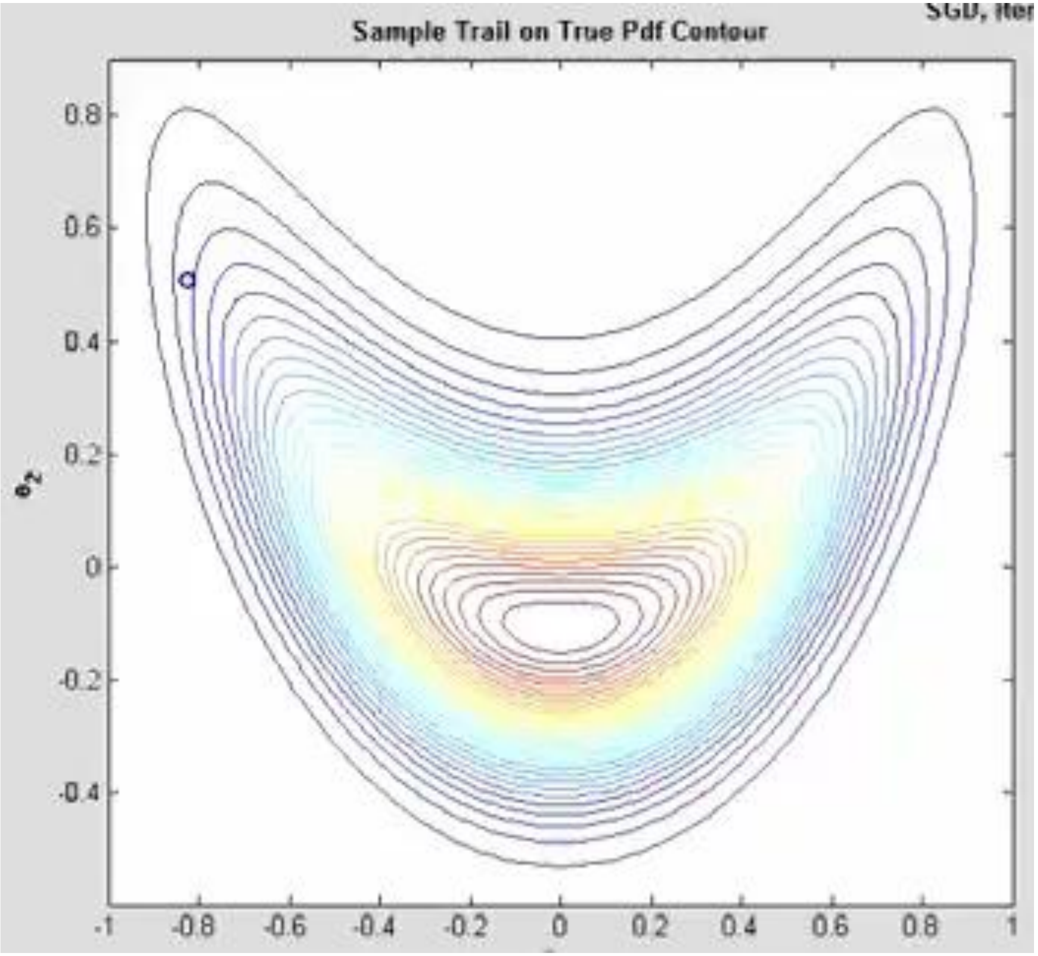
- For each training sample:
  - Estimate gradient
  - Update Parameters

$$\Delta = -\nabla_{\boldsymbol{\theta}} l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \underset{\substack{\uparrow \\ \text{learning rate}}}{\alpha}\, \Delta$$

- We minimize negative log likelihood
- Need to compute gradient – Ugh
- Theano does it for us!!

Stochastic Gradient Decent



https://www.youtube.com/watch?v=HvLJUsEc6dw

Libraries

- Theano
- Torch
- Caffe

- TensorFlow
- …

Theano

- Full disclosure: My favorite
- Python
- Transparent GPU integration
- Symbolical Graphs
- Auto-gradient
- Low level – in a good way!
- If you want high-level on top:
  - Pylearn2
  - Keras, Lasagne, Blocks
  - …

Torch

- Lua (and no Python interface)
- Very fast convolutions
- Used by Google Deep Mind, Facebook AI, IBM
- Layer instead of graph based

https://en.wikipedia.org/wiki/Torch_(machine_learning)

Caffe

- C++ based
- Higher abstraction than Theano or Torch
- Good for training standard models
- Model zoo for pre-trained models

Tensorflow

- Symbolic graph and auto-gradient
- Python interface
- Visualization tools
- Some performance issues regarding speed and memory

https://github.com/soumith/convnet-benchmarks/issues/66

We are going to use Theano today

- Python library for efficient evaluation of mathematical expressions
- **tight integration with NumPy** – Use *numpy.ndarray* in Theano-compiled functions.
- **transparent use of a GPU** – Perform data-intensive calculations up to 140x faster than with CPU.(float32 only)
- **efficient symbolic differentiation** – Theano does your derivatives for function with one or many inputs.

Free and Open Source

- F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley and Y. Bengio

- Giving credit: http://deeplearning.net/software/theano/citation.html#citation

Exercise: clone github repository

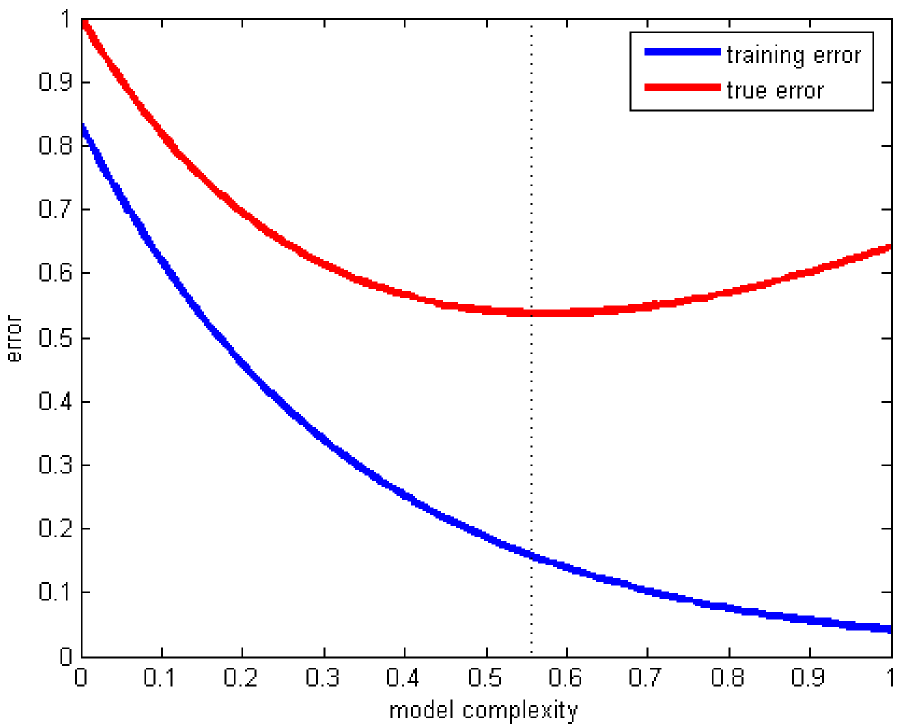
- git clone --recursive https://github.com/vkaynig/IACS_ComputeFest_DeepLearning.git

LET’S GET SOME EXPERIENCE

TIPS AND TRICKS

Number of Layers / Size of Layers

- If data is unlimited larger and deeper should be better
- Larger networks can over fit more easily
- Take computational cost into account



Learning Rate

- One of the most important parameters
- If network diverges most probably learning rate is too large
- Smaller works better
- Can slowly decay over time
- Can have one learning rate per layer

Other tips for SGD:
http://leon.bottou.org/publications/pdf/tricks-2012.pdf

Momentum

- Helps to escape local minima
- Crucial to achieve high performance

$$
\begin{aligned}
v_{t+1} &= \mu v_t - \varepsilon \nabla f(\theta_t) \\
\theta_{t+1} &= \theta_t + v_{t+1}
\end{aligned}
$$

More about Momentum:
http://www.jmlr.org/proceedings/papers/v28/sutskever13.pdf

Convergence

- Monitor validation error
- Stop when it doesn’t improve within n iterations

- If learning rate decays you might want to adjust number of iterations

Initialization of W

- Need randomization to break symmetry
- Bad initializations are untrainable
- Most heuristics depend on the number of input (and output) units
- Sometimes W is rescaled during training
  - Weight decay (L2 regularization)
  - Normalization

Data Augmentation

- Exploit invariances of the data
- Rotation, translation
- Nonlinear transformation
- Adding Noise

Type	Classifier
Neural network	6-layer NN 784-2500-2000-1500-1000-500-10 (on GPU), with elastic distortions
Convolutional neural network	Committee of 35 conv. net, 1-20-P-40-P-150-10, with elastic distortions

Preprocessing	Error rate (%)
None	0.35[17]
Width normalizations	0.23[8]

http://en.wikipedia.org/wiki/MNIST_database

Data Normalization

- We have seen std and mean normalization

- Whitening
  - Neighbored pixels often are redundant
  - Remove correlation between features

More about preprocessing: http://deeplearning.stanford.edu/wiki/index.php/Data_Preprocessing

Non-Linear Activation Function

- Sigmoid
  - Traditional choice
- Tanh
  - Symmetric around the origin
  - Better gradient propagation than Sigmoid
- Rectified Linear (discussed tomorrow)

L1 and L2 Regularization

- Most pictures of nice filters involve some regularization

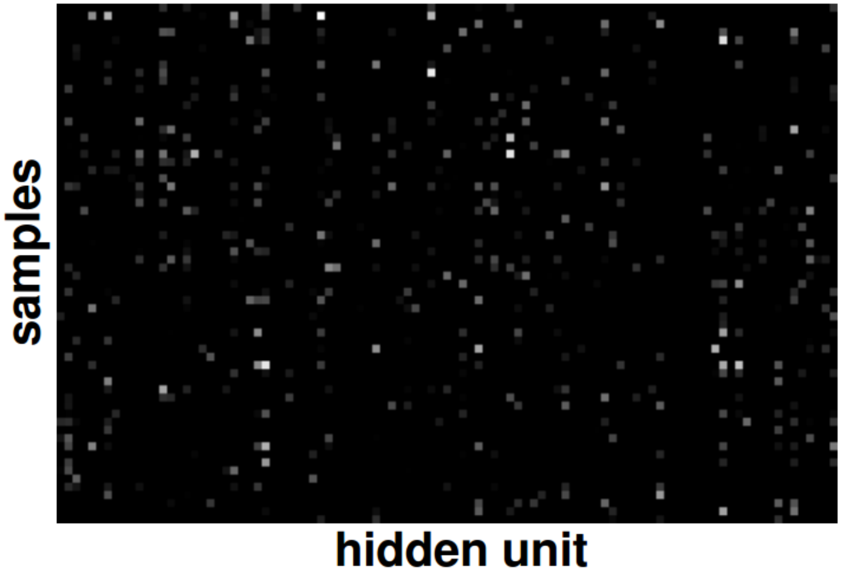
- L2 regularization corresponds to weight decay
- L2 and early stopping have similar effects

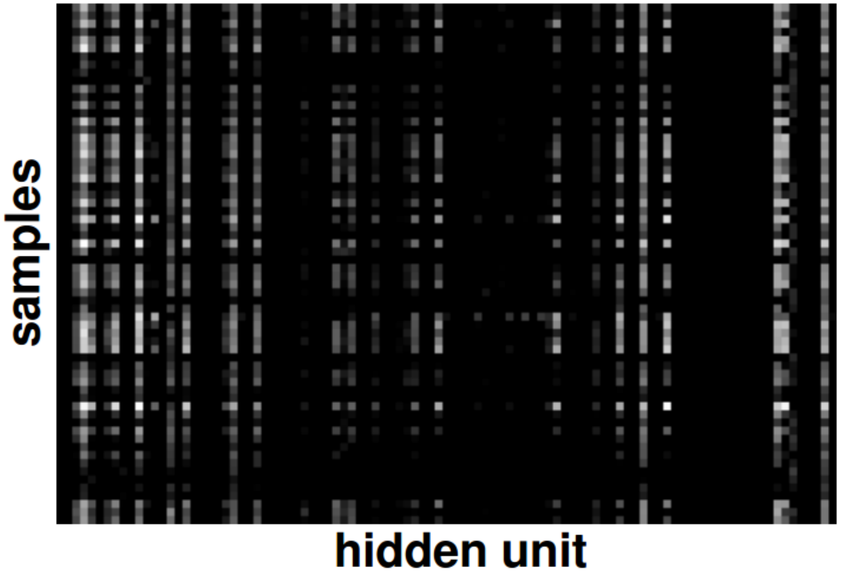
- L1 leads to sparsity

- Might not be needed anymore (more data, dropout)

Monitoring Training

- Monitor training and validation performance
- Can monitor hidden units
- Good: Uncorrelated and high variance



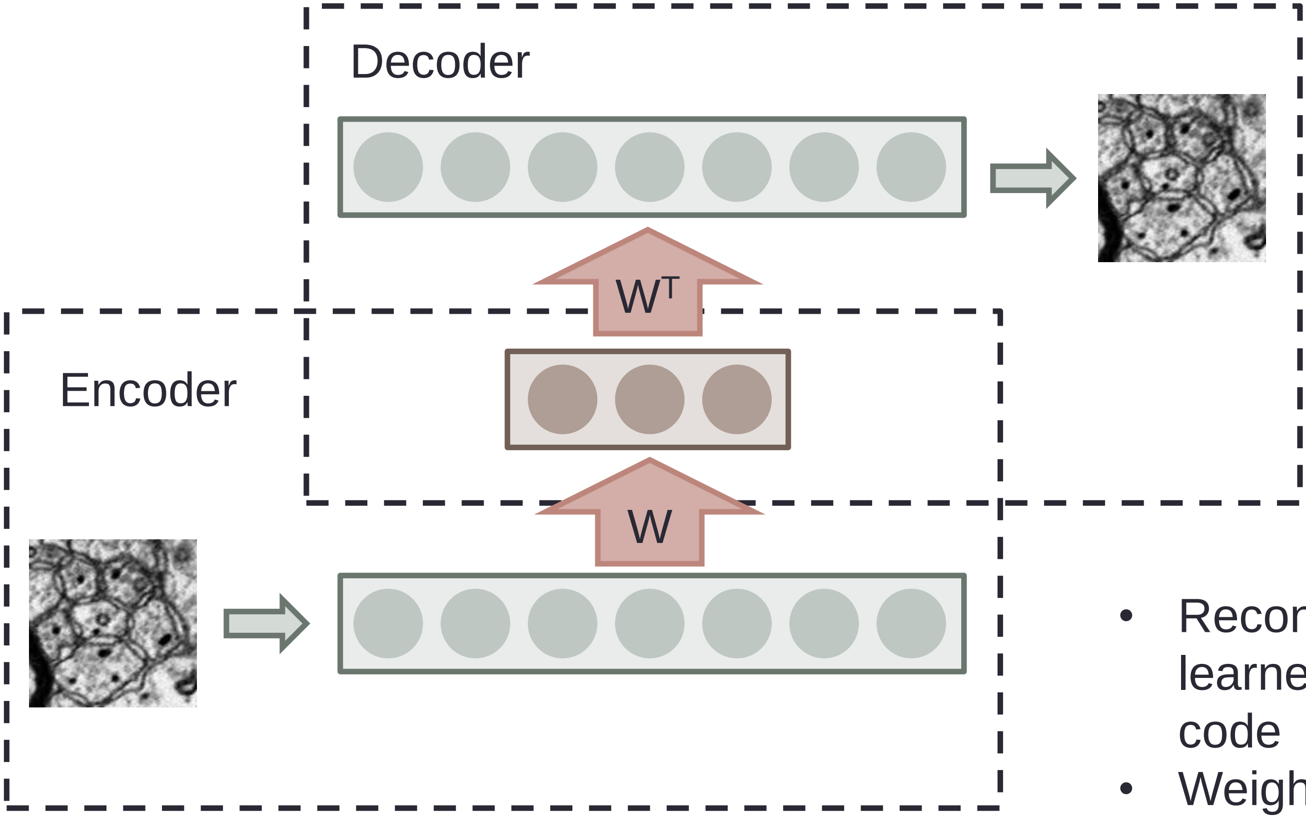


Autoencoder

- This is what Google used for their Google brain
- Basically just a MLP
- Output size is equal to input size

- Popular for pre-training a network on unlabeled data

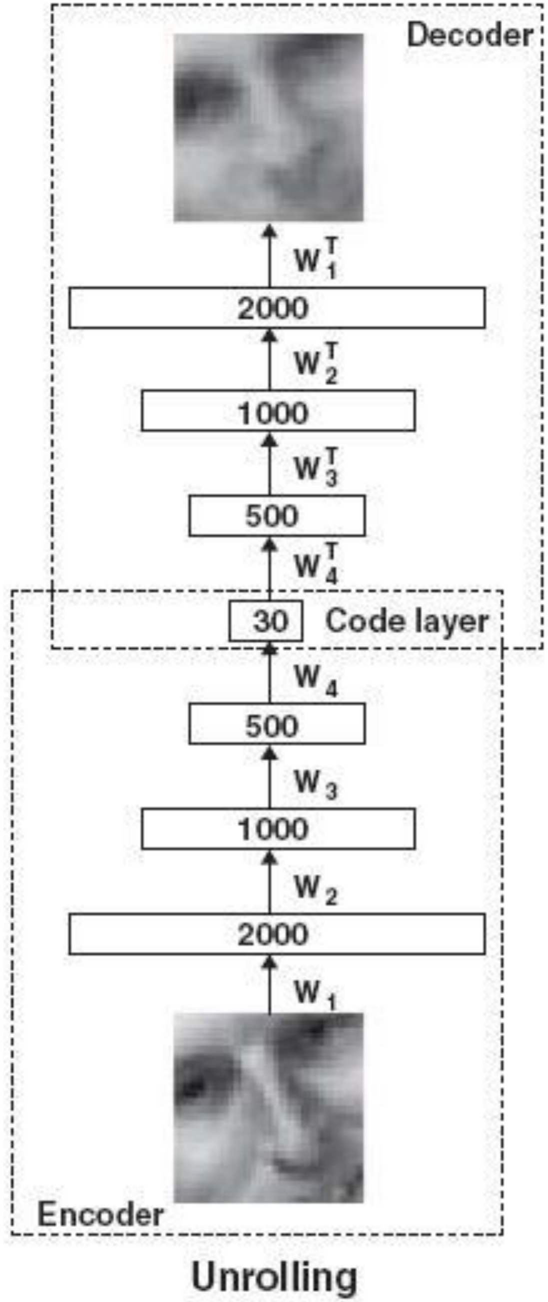
Autoencoder



- Reconstruct image from learned low dimensional code
- Weights are tied
- Learned features are often useful for classification

Deep Autoencoder

- Each Layer trained by itself
- Higher layers build on top of lower layers
- Can do fine tuning in the end



Further Resources

- More about theory:
  - Yoshua Bengio's book:http://www.iro.umontreal.ca/~bengioy/dlbook/
  - Deep learning reading list: http://deeplearning.net/reading-list/

- More about Theano:
  - http://deeplearning.net/software/theano/
  - http://deeplearning.net/tutorial/