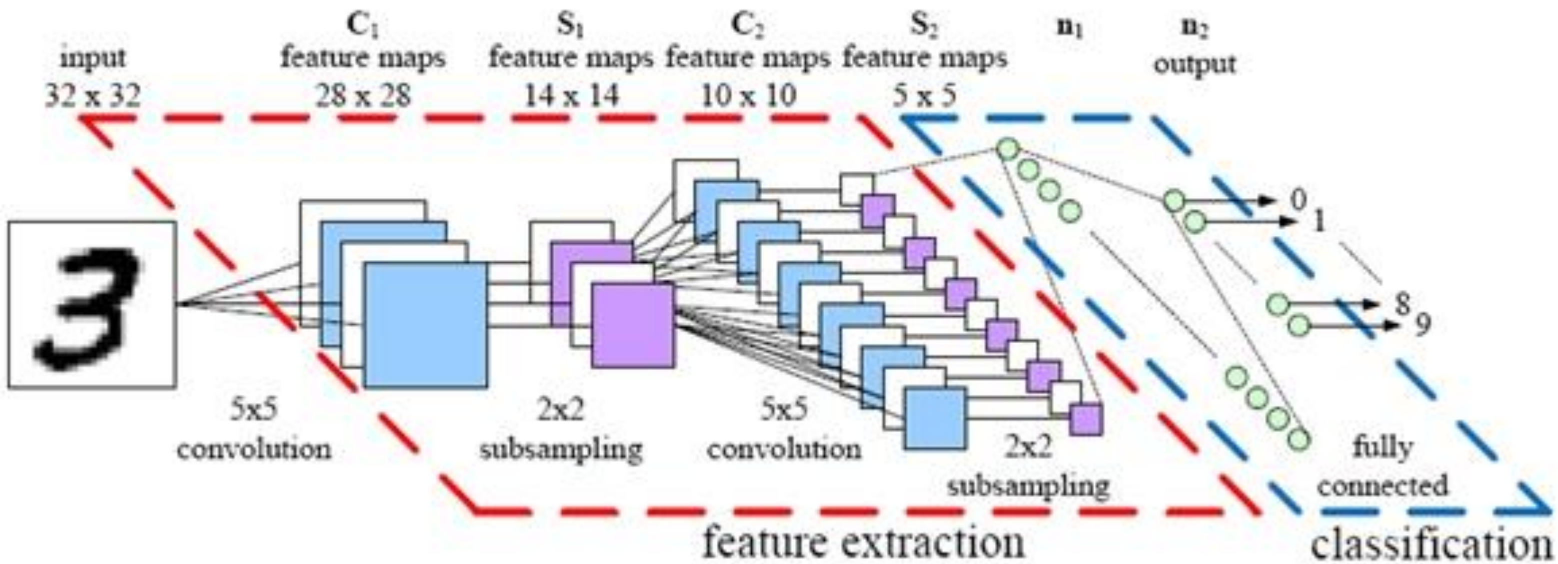
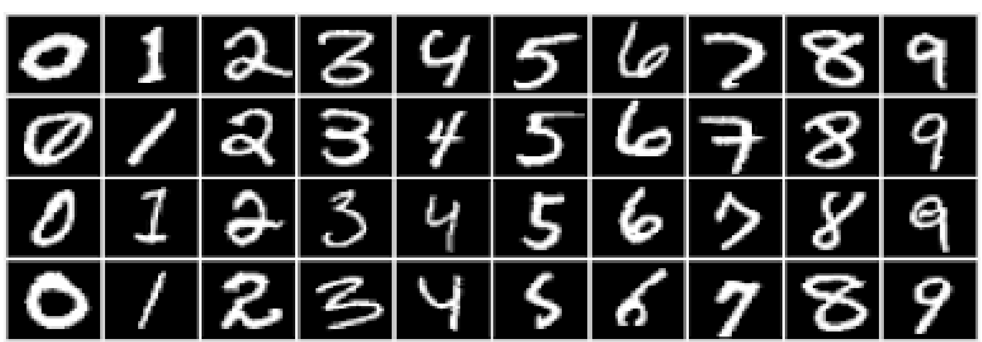
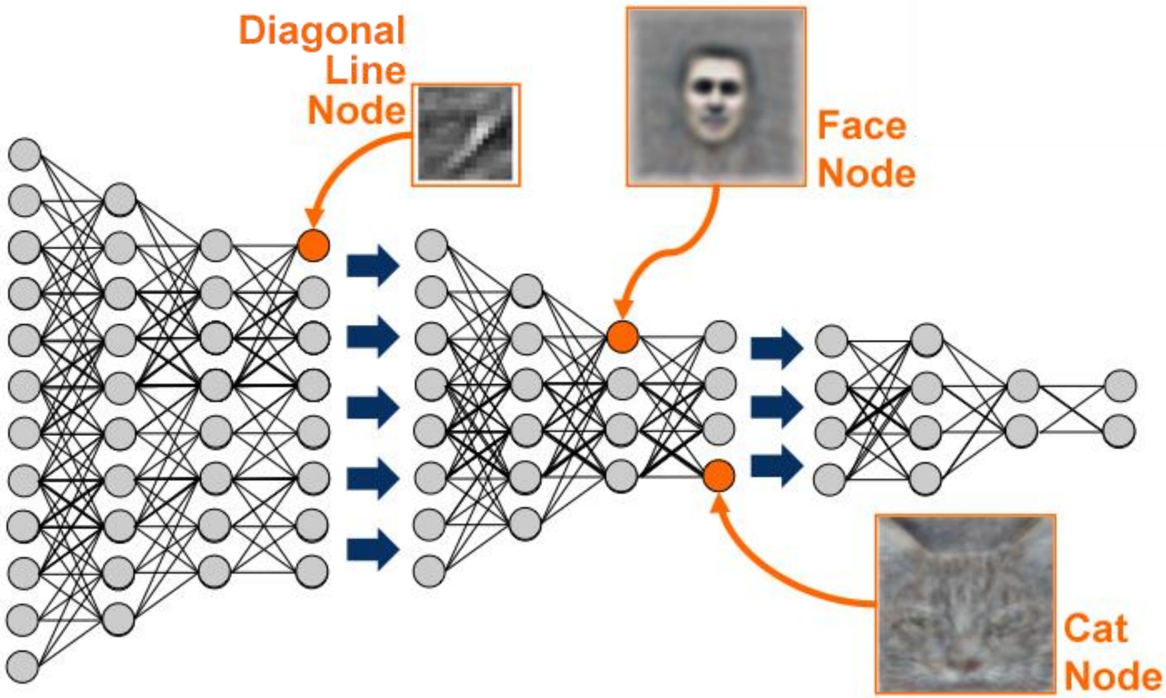
DEEP LEARNING

HANDWRITING RECOGNITION



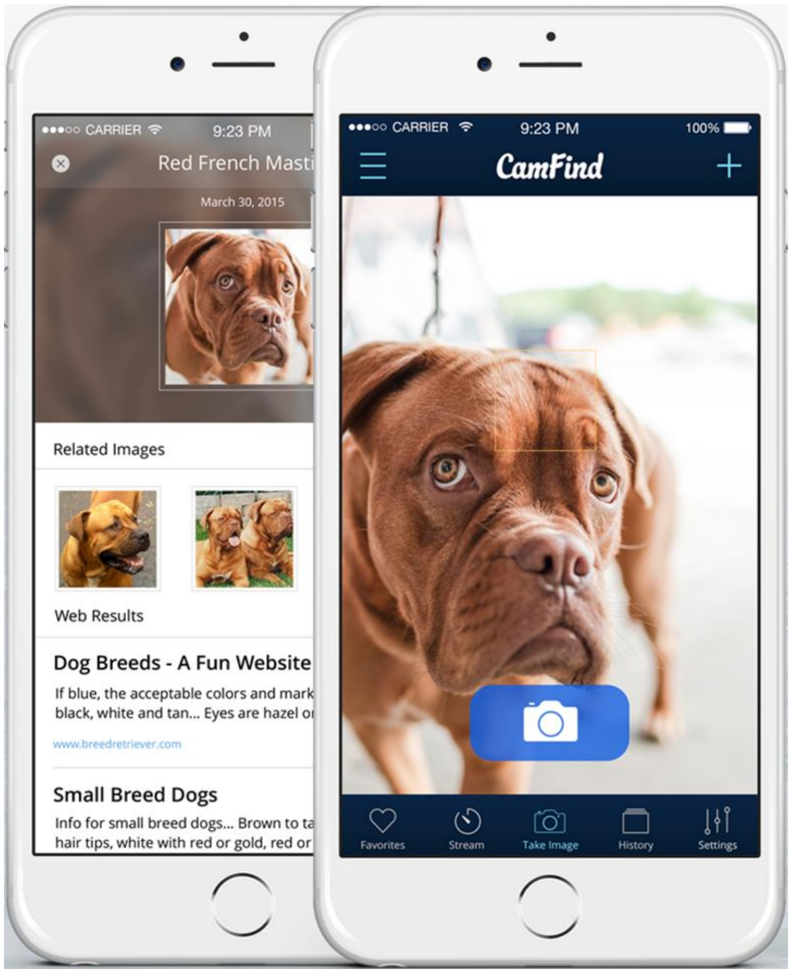


GOOGLE CAT VIDEOS



Deep Learning with GPUs

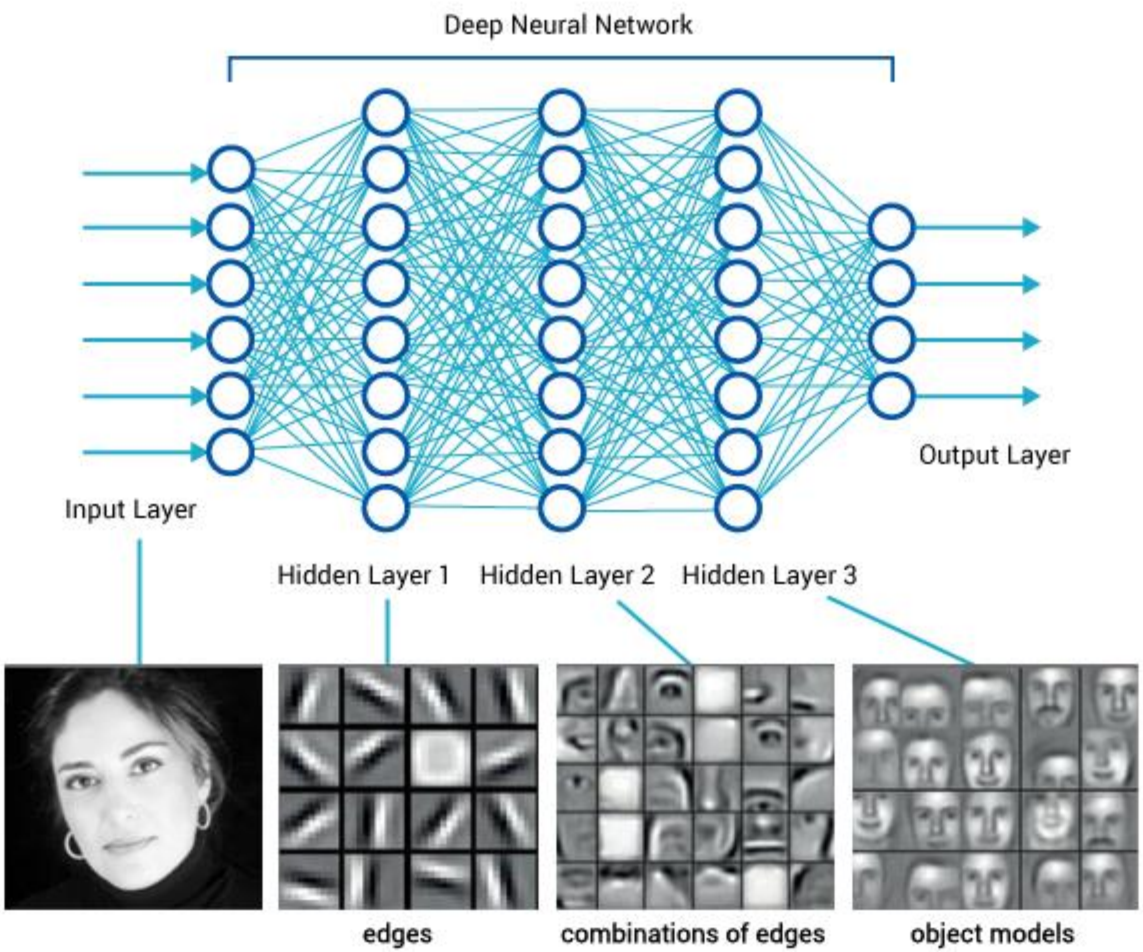
IMAGE RECOGNITION



CamFind

Visual Search Engine
(available on iOS, Android)

FACE RECOGNITION

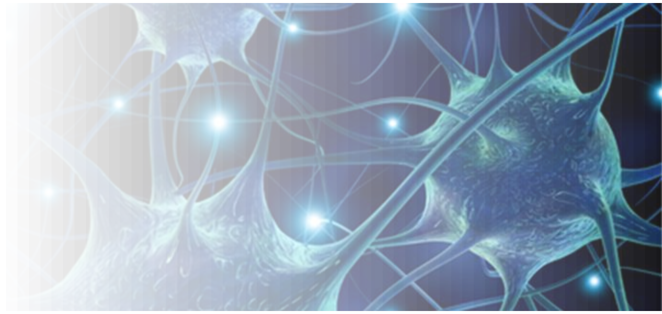


SPEECH TRANSLATION

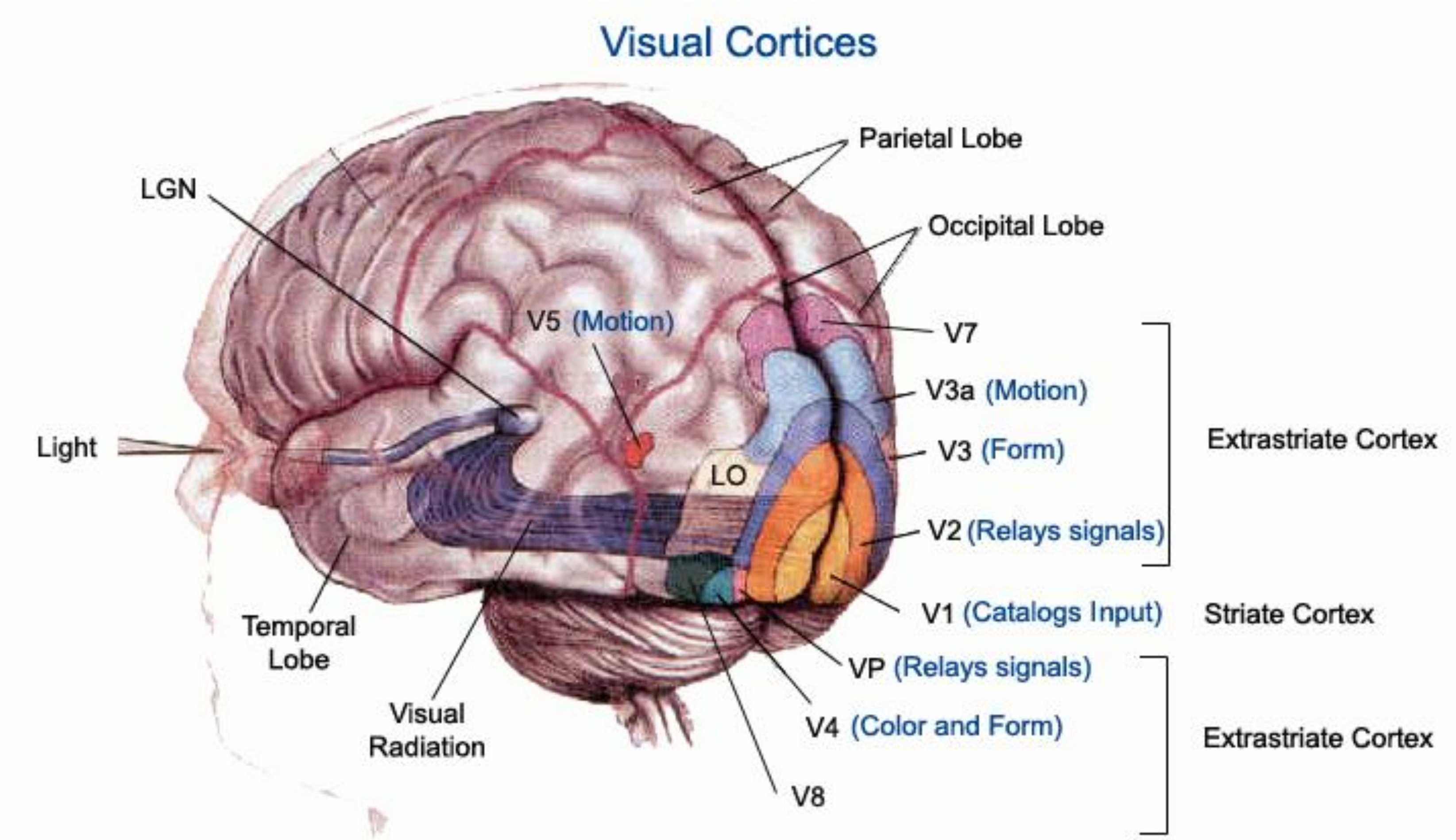
From Hidden Markov Models to Recurrent Neural Networks

https://www.youtube.com/watch?v=mWTySUGXR2k

THEORY

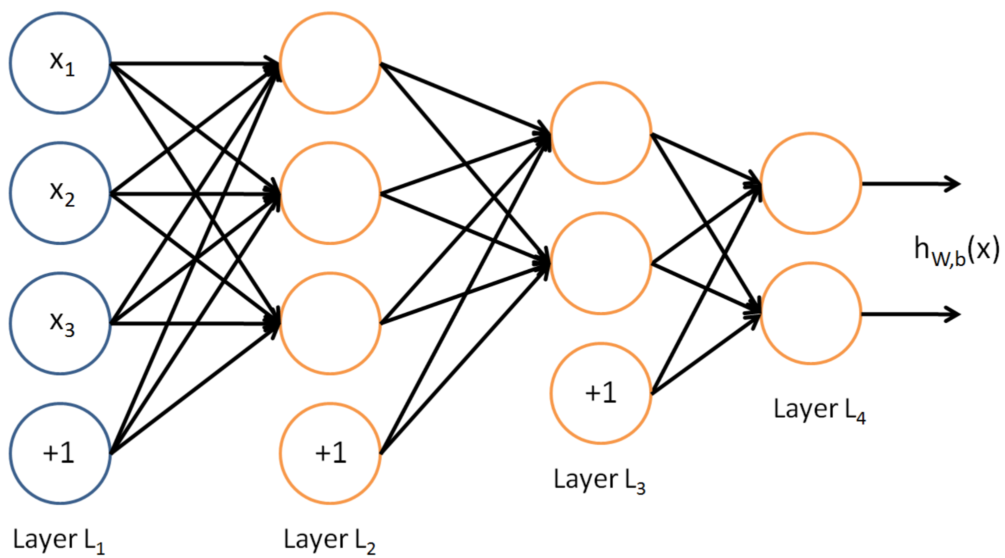


WHAT IS DEEP LEARNING?



WHAT IS DEEP LEARNING?

Biologically-inspired multilayer neural networks



Both supervised and unsupervised

Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.

WHAT IS DEEP LEARNING?

Example. Face recognition (Facebook)

Edges   Eyes, Noses, Mouths   Faces

Deeper layers learn higher-order features

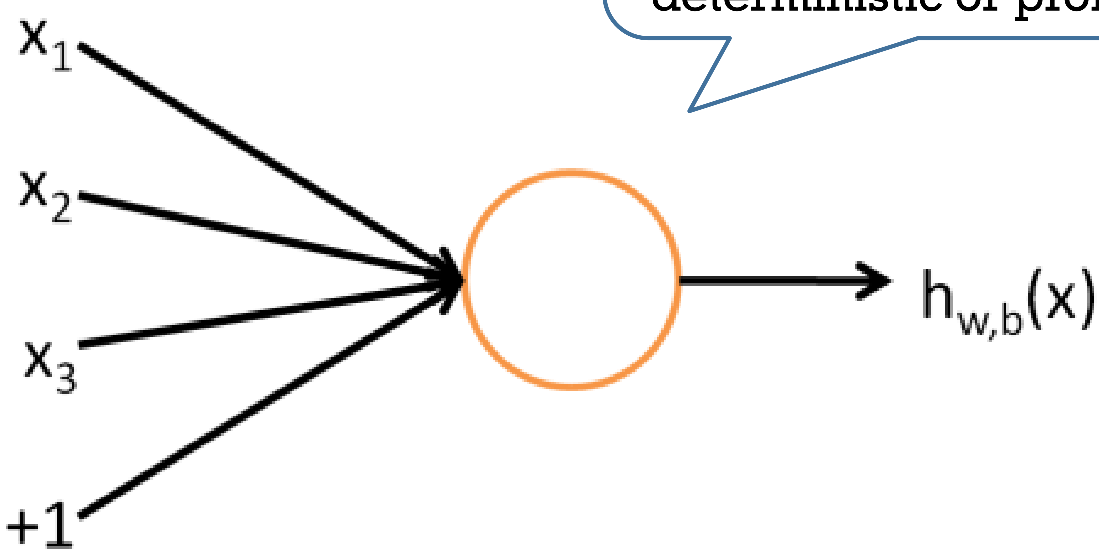
LISA lab, "Deep Learning Tutorials." (http://www.deeplearning.net/tutorial/) accessed 2014.

Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009.

NEURON

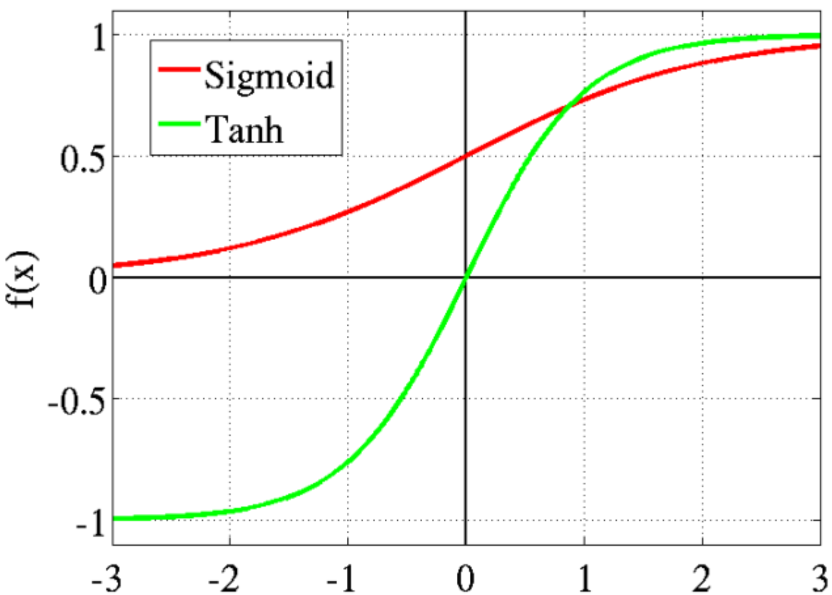
Depending on function f,
neurons can be:
real-valued or binary-valued;
deterministic or probabilistic.

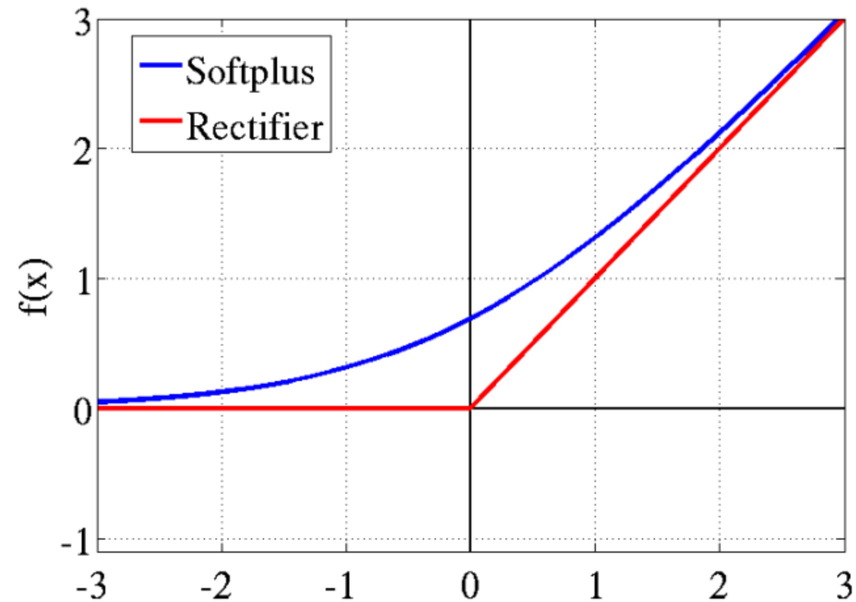
Perceptron.



$$h_{w,b}(x) = f(w^\top x) = f\left(\sum_{i=1}^{d} w_i x_i + b\right)$$

ACTIVATION FUNCTIONS





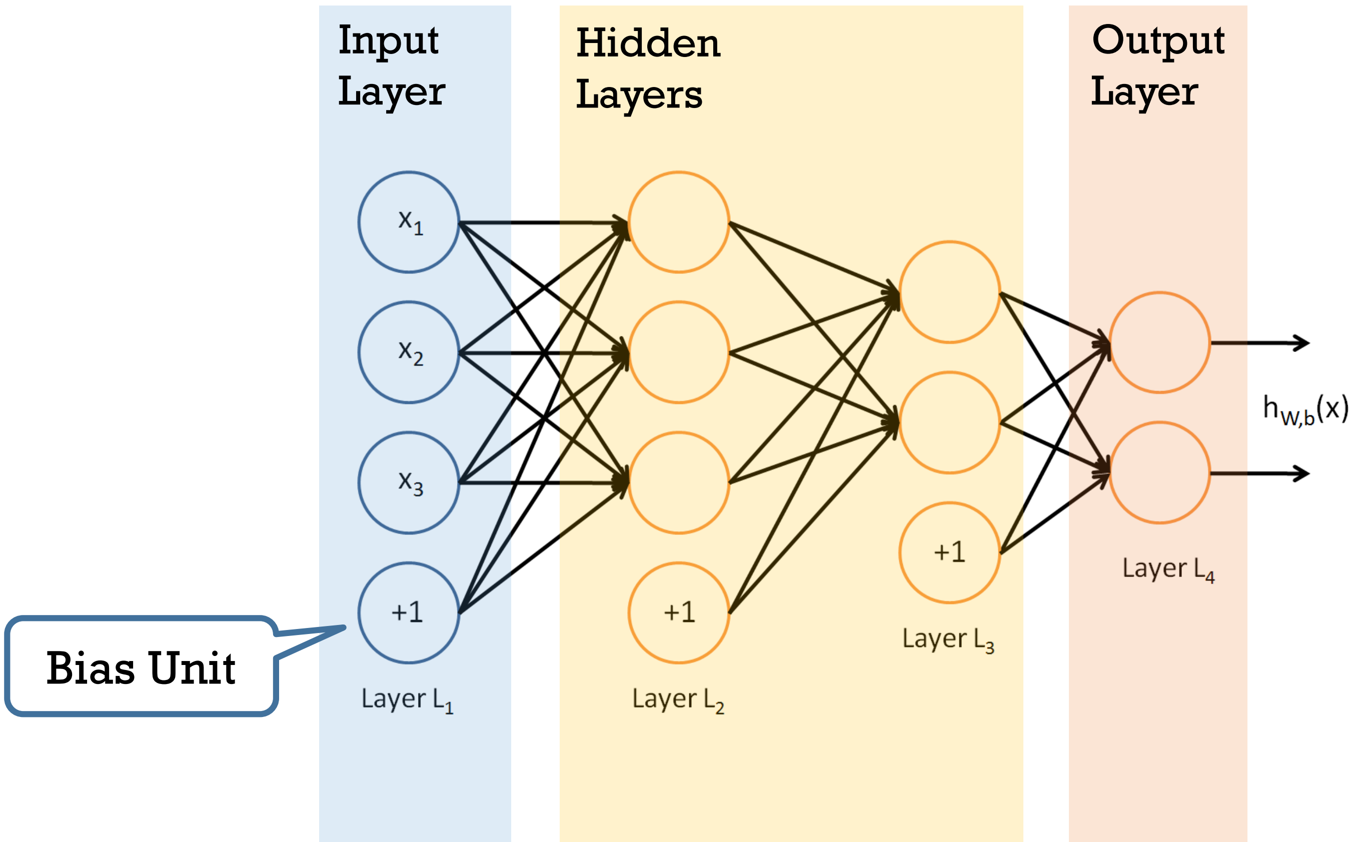
sigmoid $f(z) = \frac{1}{1+e^{-z}}$

tanh $f(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$

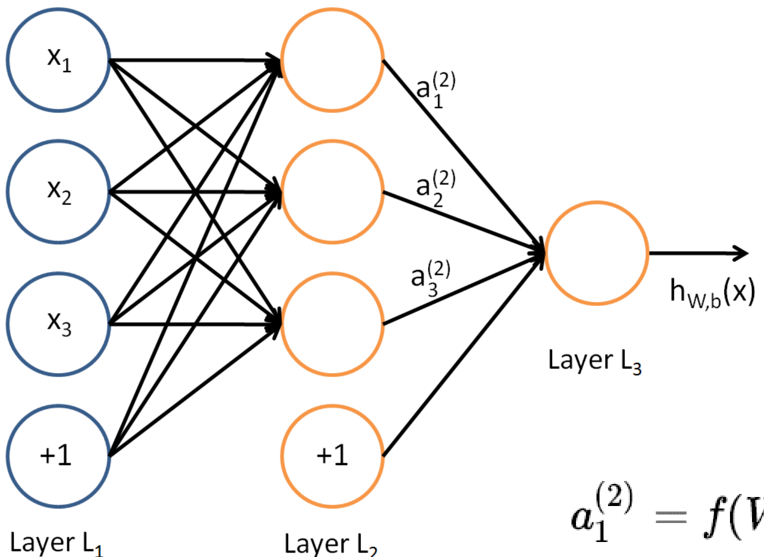
softplus $f(z) = \ln(1 + e^{-z})$

rectified linear unit (ReLU) $f(z) = \max(0, z)$

MULTI-LAYER NEURAL NETWORK

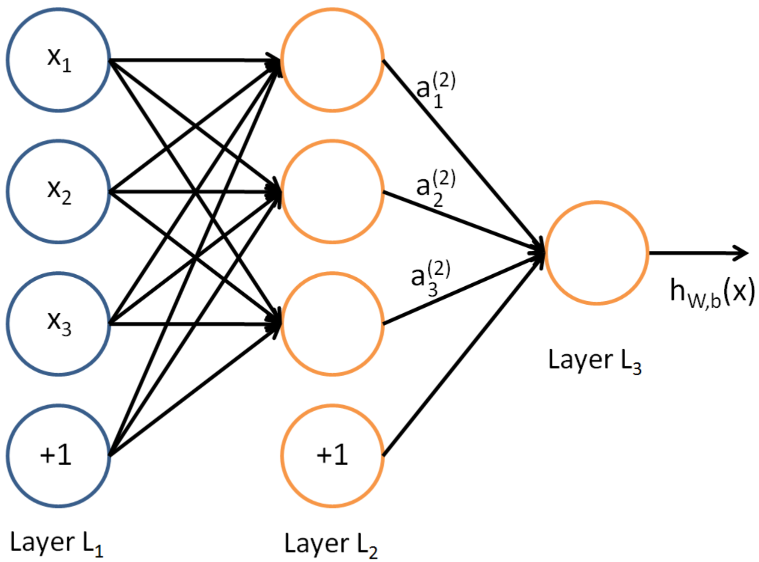


MULTI-LAYER NEURAL NETWORK



$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b}(x) = a_1^{(3)} &= f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})
\end{aligned}
$$

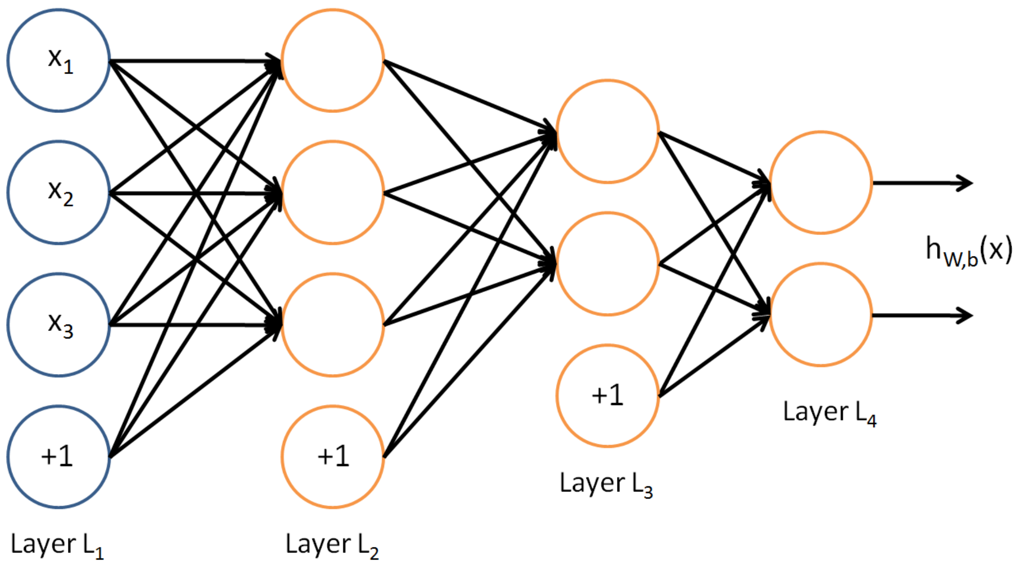
MULTI-LAYER NEURAL NETWORK



Forward Propagation.

$$\begin{aligned}
z^{(2)} &= W^{(1)}x + b^{(1)} \\
a^{(2)} &= f(z^{(2)}) \\
z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\
h_{W,b}(x) &= a^{(3)} = f(z^{(3)})
\end{aligned}$$

MULTI-LAYER NEURAL NETWORK



$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l+1)})$$

Activation

Feed-Forward Neural Network.

Neural Network Architecture.
Arrangement of neurons and their connections.

COST FUNCTION

For binary neurons, other loss functions are used.

Error for one data point

$$J(W,b;x,y) = \frac{1}{2}\left\| h_{W,b}(x) - y \right\|^2$$

Error for training set + Weight decay regularization

$$\begin{aligned} J(W,b) &= \left[\frac{1}{m} \sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)}) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2 \\ &= \left[\frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{2} \left\| h_{W,b}(x^{(i)}) - y^{(i)} \right\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2 \end{aligned}$$

λ weight decay parameter

BACK PROPAGATION

Chain Rule for Neural Networks.

$$\frac{\partial}{\partial W_{ij}^{(l)}}\left(\frac{1}{2}\left\|a^{(n_l)} - y\right\|^2\right) = \left(a^{(n_l)} - y\right)\frac{\partial}{\partial W_{ij}^{(l)}} f\left(z^{(n_l)}\right)$$

$$= \left(a^{(n_l)} - y\right) f'\left(z^{(n_l)}\right)\frac{\partial}{\partial W_{ij}^{(l)}}\left(W^{(n_l-1)} a^{(n_l-1)} + b^{(n_l-1)}\right)$$

$$= \left(a^{(n_l)} - y\right) f'\left(z^{(n_l)}\right) W^{(n_l-1)}\frac{\partial}{\partial W_{ij}^{(l)}} a^{(n_l-1)}$$

$$= \underbrace{\underbrace{\left(a^{(n_l)} - y\right) f'\left(z^{(n_l)}\right)}_{\delta^{(n_l)}} W^{(n_l-1)} f'\left(z^{(n_l-1)}\right)}_{\delta^{(n_l-1)}}\frac{\partial}{\partial W_{ij}^{(l)}} z^{(n_l-1)}$$

BACK PROPAGATION

1. Perform a feed-forward pass, computing the activations layer by layer.

2. For the output layer (layer n_l), set

$$\delta^{(n_l)} = -(y - a^{(n_l)}) \bullet f'(z^{(n_l)})$$

3. For $l = n_l - 1, n_l - 2, n_l - 3, \ldots, 2$, set

$$\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)} \right) \bullet f'(z^{(l)})$$

4. Compute the desired partial derivatives:

$$\nabla_{W^{(l)}} J(W, b; x, y) = \delta^{(l+1)} (a^{(l)})^T + \lambda W^{(l)}$$

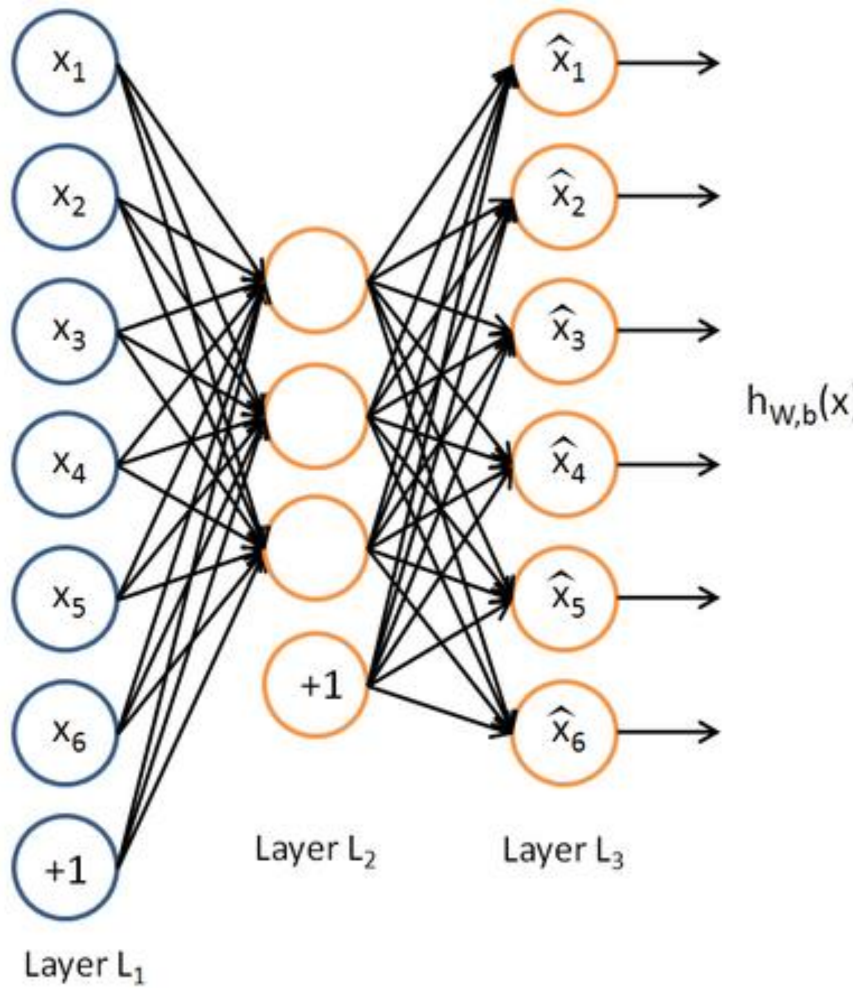
$$\nabla_{b^{(l)}} J(W, b; x, y) = \delta^{(l+1)}.$$

AUTOENCODERS

Training a multilayer neural network to reconstruct the input from a reduced representation.

Strategies for Dimensionality Reduction

- Few hidden neurons
- Sparse activations



SPARSE AUTOENCODER

Sparsity Penalty.

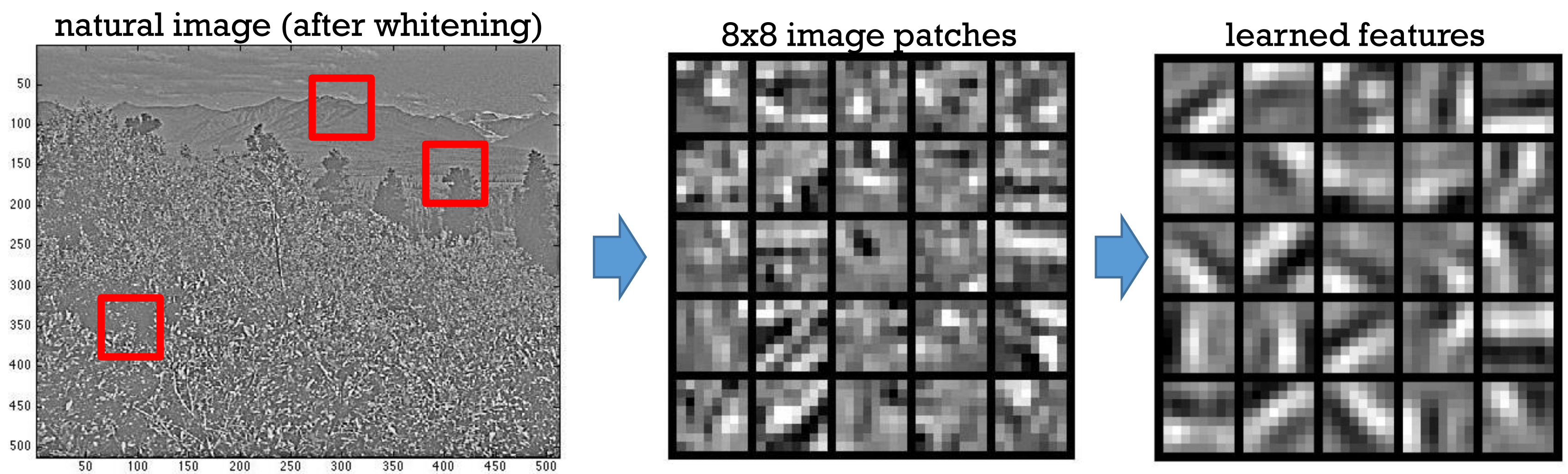
Average activation $$\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}\left[a_j^{(2)}(x^{(i)})\right]$$

$$\mathrm{KL}(\rho||\hat{\rho}_j) = \rho\log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}$$

$$J_{\mathrm{sparse}}(W,b) = J(W,b) + \beta\sum_{j=1}^{s_2}\mathrm{KL}(\rho||\hat{\rho}_j),$$

β sparsity parameter

NATURAL IMAGES

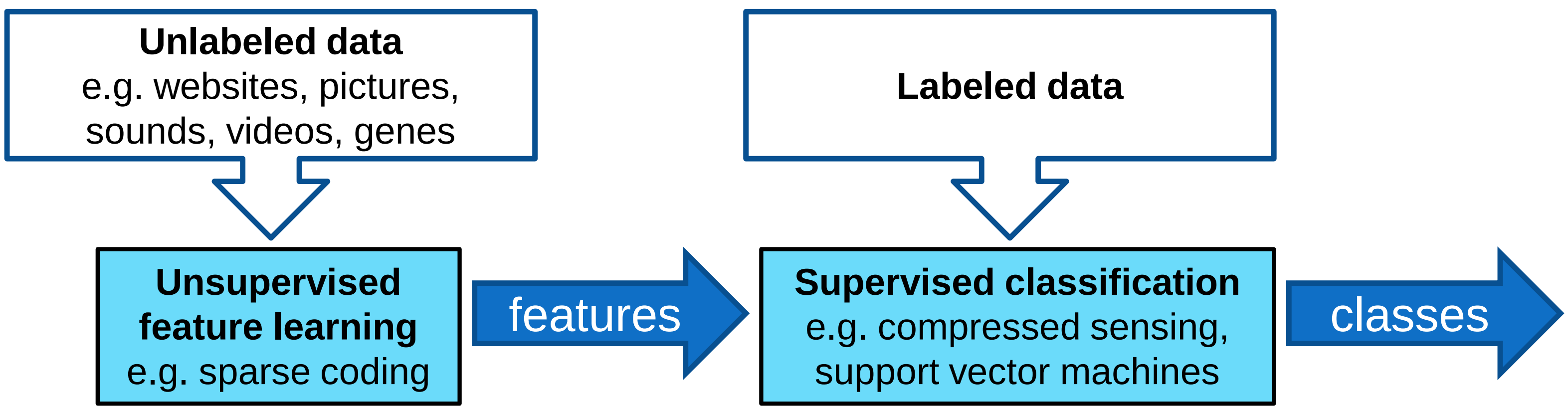


Edge features similar to those from neuroscience experiments (see Hubel & Wiesel Cat Experiment, 1959).

GREEDY INITIALIZATION OF LAYERS

- Initialization of neural weights
  1. Train an autoencoder on input data.
  2. Feed-forward data to hidden layer.
  3. Train a second autoencoder on hidden activations.
  4. Feed-forward data to second hidden layer.
  5. ... and so on.
- Fine tuning of neural weights
  - Backpropagation

SEMI-SUPERVISED LEARNING



WHY DOES DEEP LEARNING WORK?

It's not really about the autoencoders…

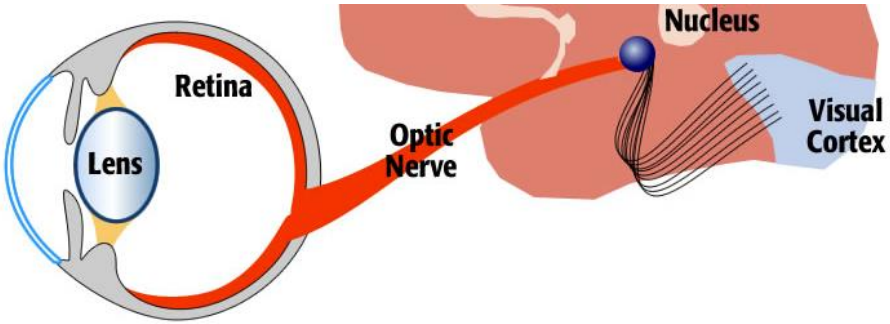
- Greedy layer-wise initialization can be any of the following:
  - Contrastive divergence
  - Sparse autoencoder
  - Sparse coding
  - K-means clustering
  - Random data vectors

Coates, Adam, and Andrew Y. Ng. "Learning feature representations with k-means." Neural Networks: Tricks of the Trade. Springer 2012. 561-580.

Coates, Adam, and Andrew Y. Ng. "The importance of encoding versus training with sparse coding and vector quantization." ICML 2011.

WHY DOES DEEP LEARNING WORK?

… but it has something to do with sparsity.



Compressed Sensing

- If input is sparse in some known basis, then it can be reconstructed from almost any random projections.

Donoho, D.L. (2006). "Compressed sensing". IEEE Transactions on Information Theory 52 (4): 1289.

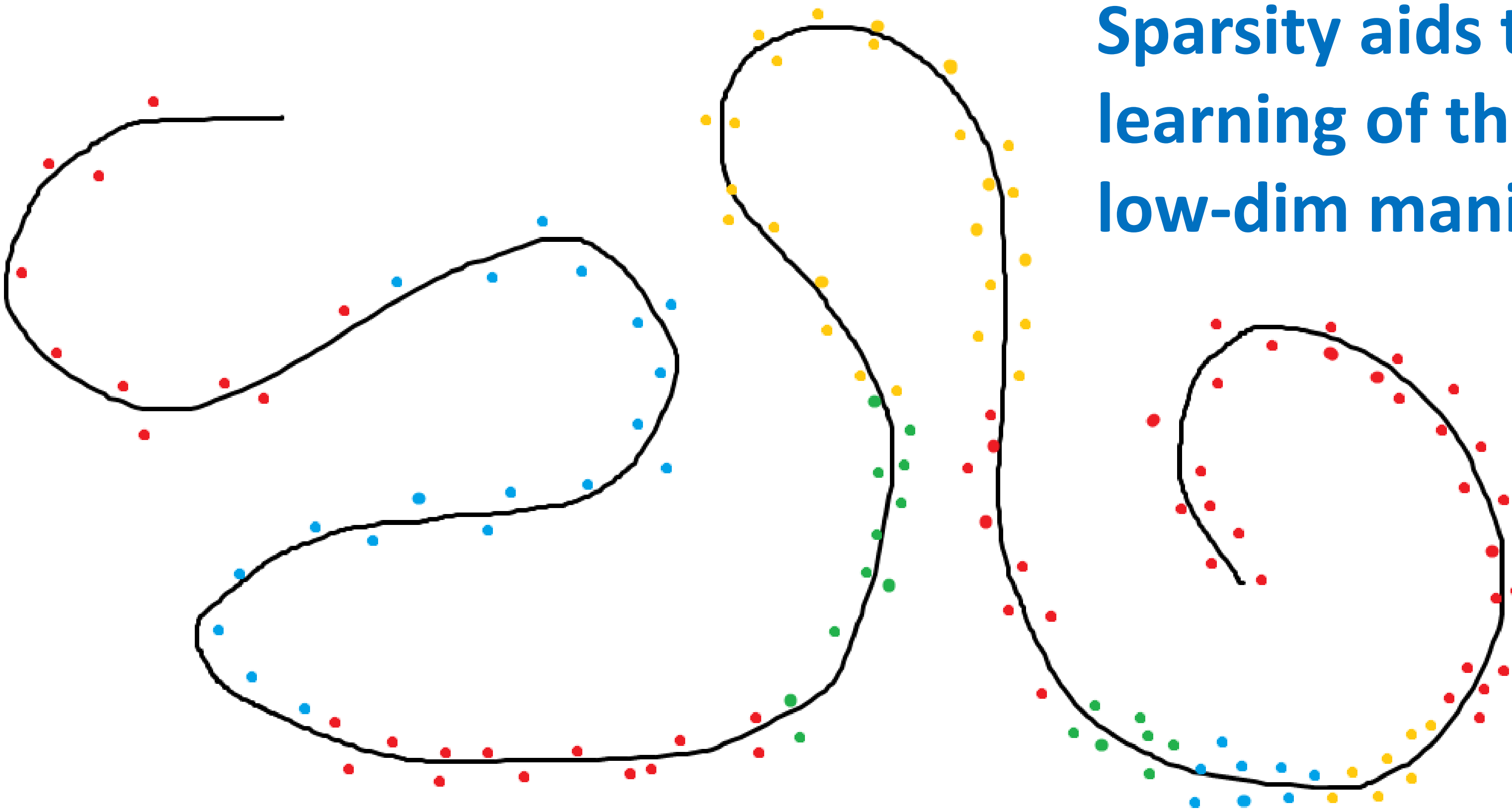
Candès, Emmanuel J.; Romberg, Justin K.; Tao, Terence (2006). "Stable signal recovery from incomplete and inaccurate measurements". CPAM 59 (8): 1207.

Hillar, Christopher, and Friedrich T. Sommer. "When can dictionary learning uniquely recover sparse data from subsamples?." arXiv:1106.3616 (2011).

WHY DOES DEEP LEARNING WORK?

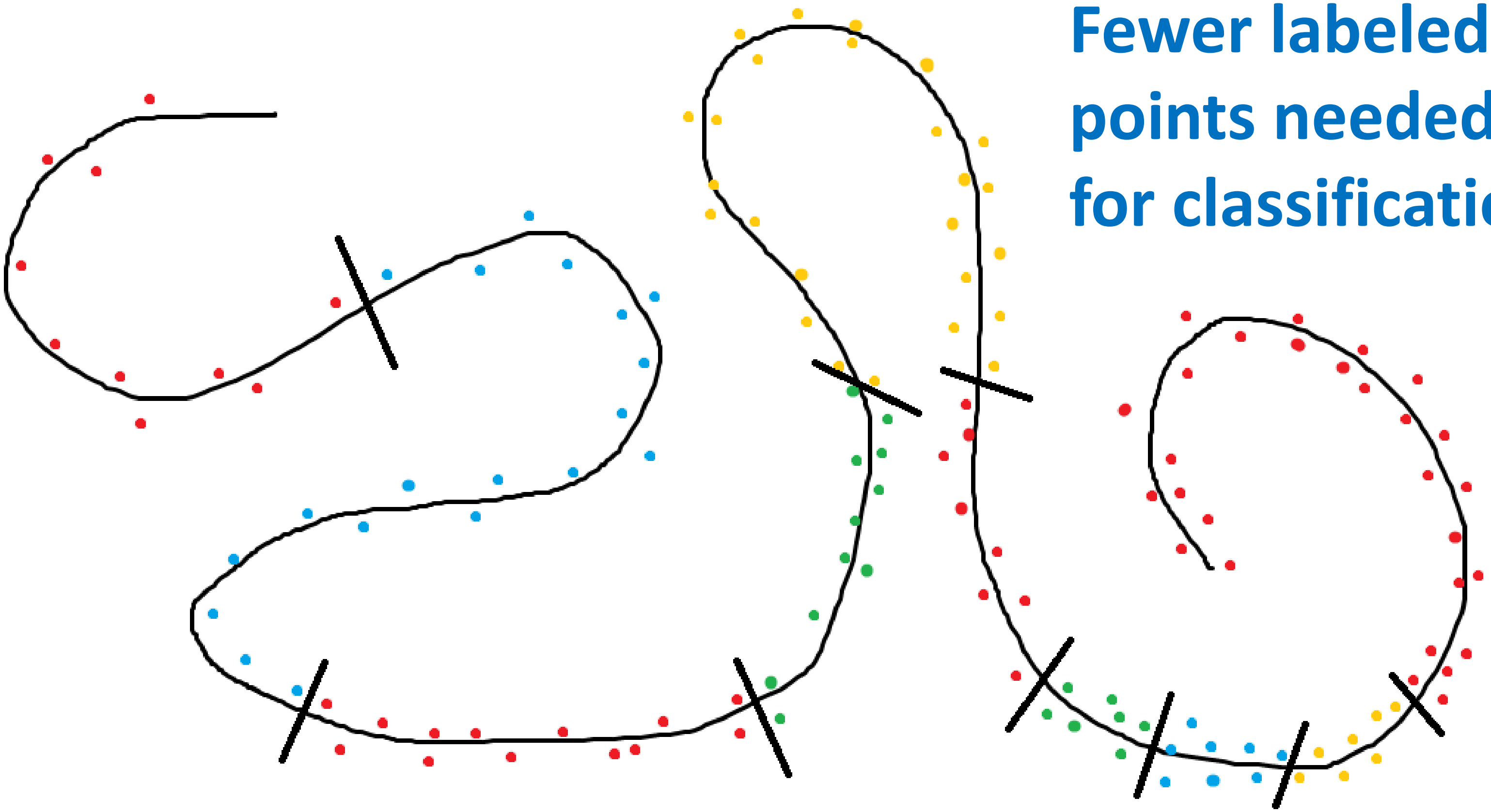
Data is often near low-dim manifold in high-dim space

WHY DOES DEEP LEARNING WORK?



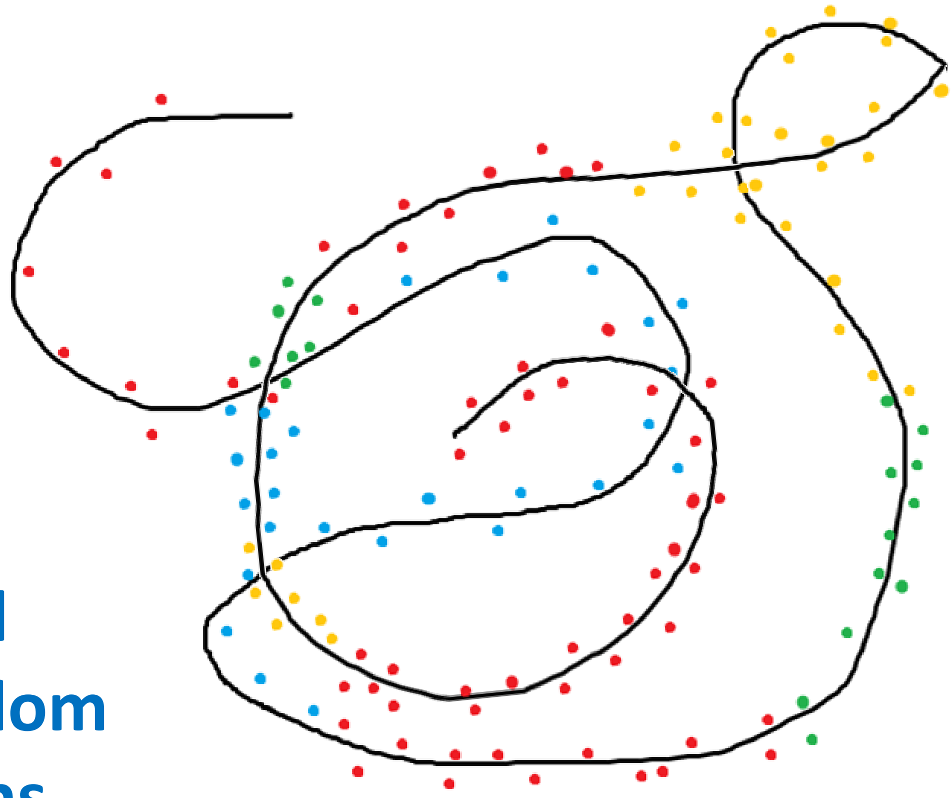
Sparsity aids the learning of the low-dim manifold

WHY DOES DEEP LEARNING WORK?



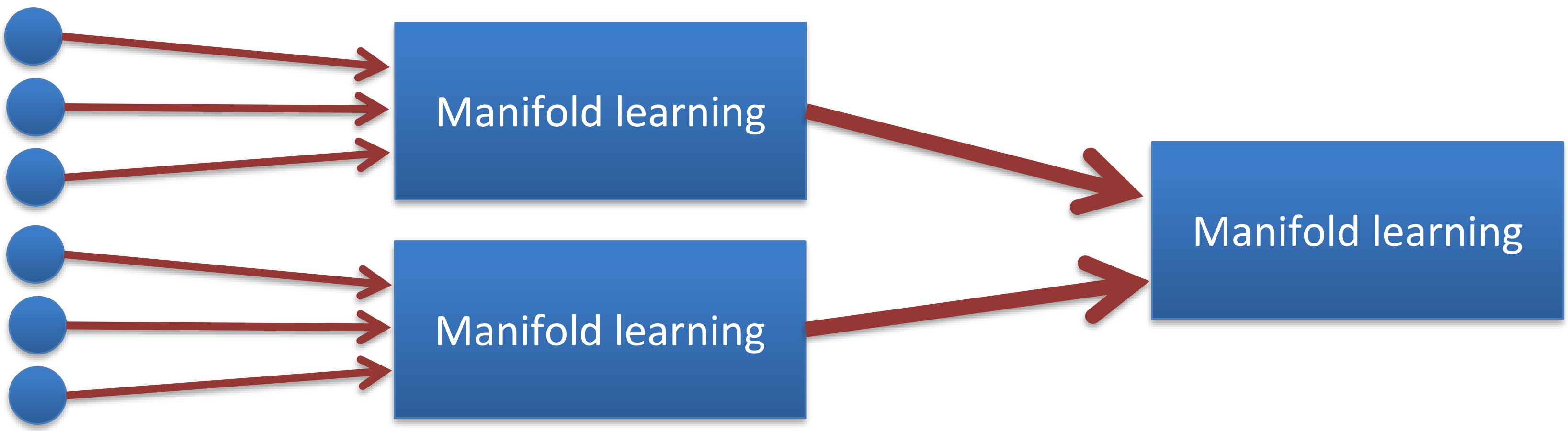
Fewer labeled points needed for classification

WHY DOES DEEP LEARNING WORK?



Structure preserved after random projections

HOW DOES DEEP LEARNING WORK?



Learning a
low-dim manifold
through sparsity

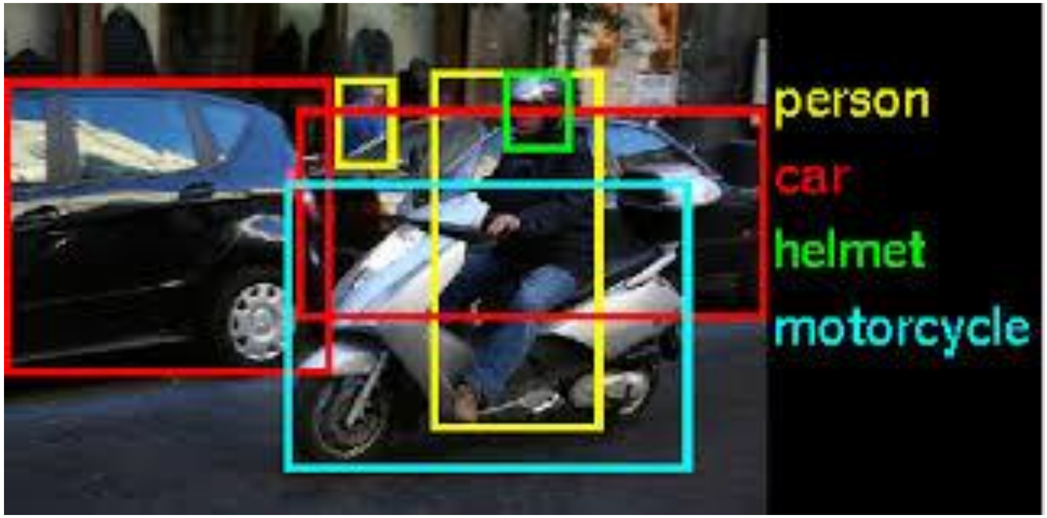
Transport
via random
projections

Information fusion
for discovery of
new structures

Example. Speech recognition – from hidden markov models (one layer) to deep learning (multiple layers)

CONVOLUTIONAL NEURAL NETWORKS

IMAGE RECOGNITION



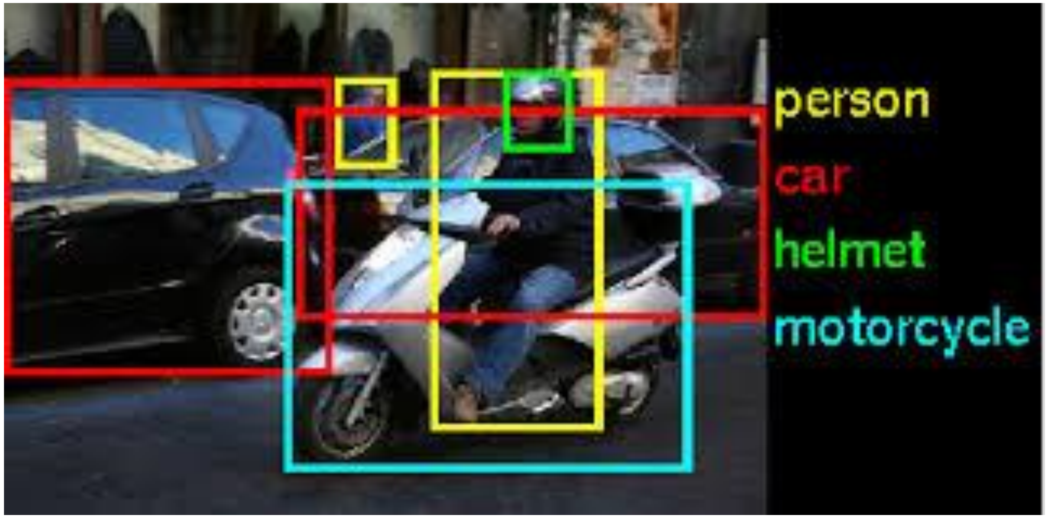
Too many parameters.

- (MNIST 28×28 pixels) * (100 features) = 78,400 params

Translation invariance.

- Statistics of a patch (usually) does not depend on its location.

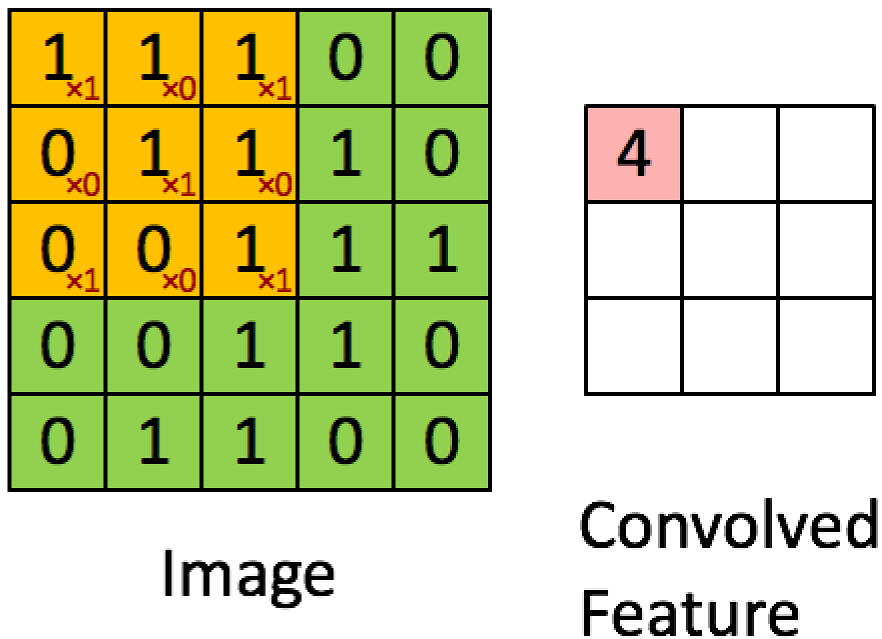
IMAGE RECOGNITION



Solution.

- Learn good features from small image patches.
- Use features to compute better representation of image.
- Down-sample image to reduce dimensionality.
- Repeat.

CONVOLUTION



FEATURE MAP

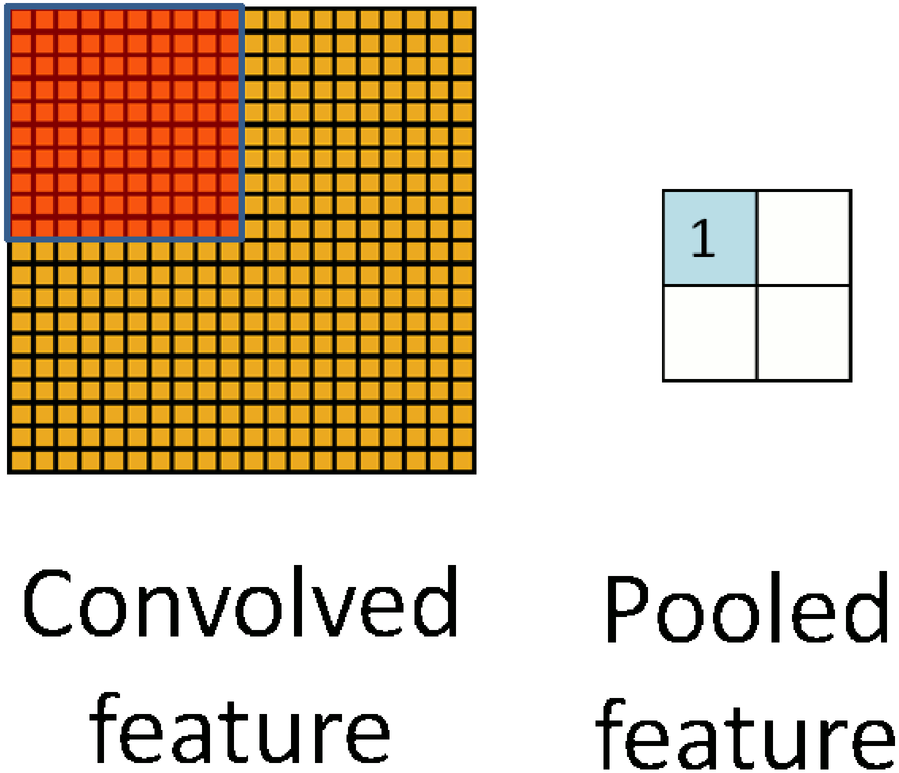
Image

Feature Map

-1	-1	-1
2	2	2
-1	-1	-1

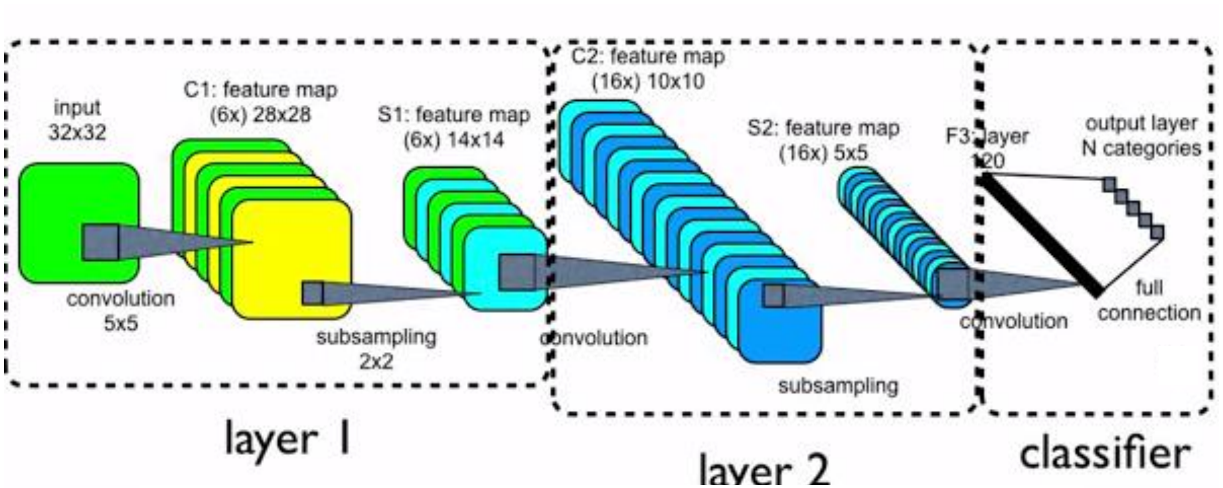
Filter

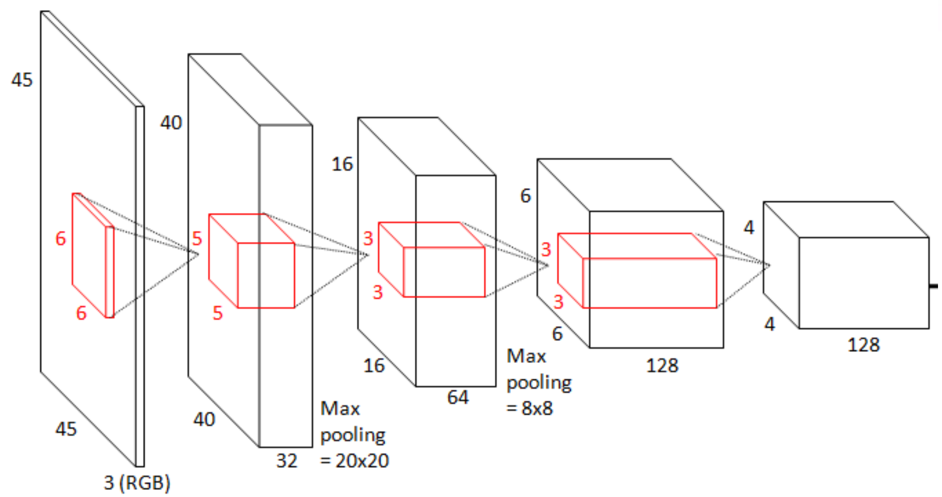
POOLING



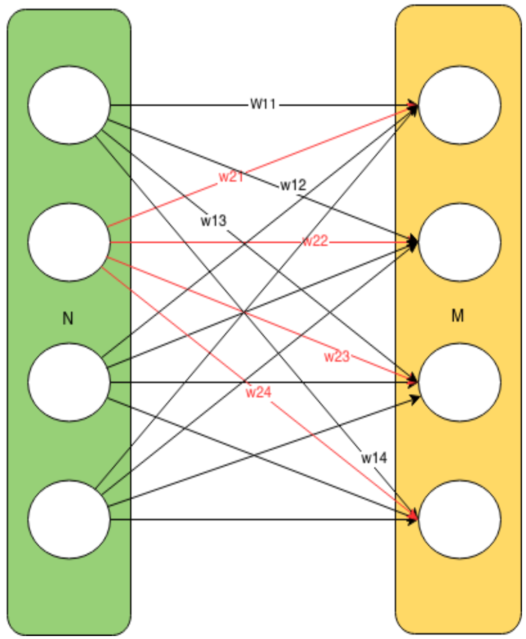
Examples. Mean Pooling, Max Pooling.

CONVOLUTIONAL PYRAMID

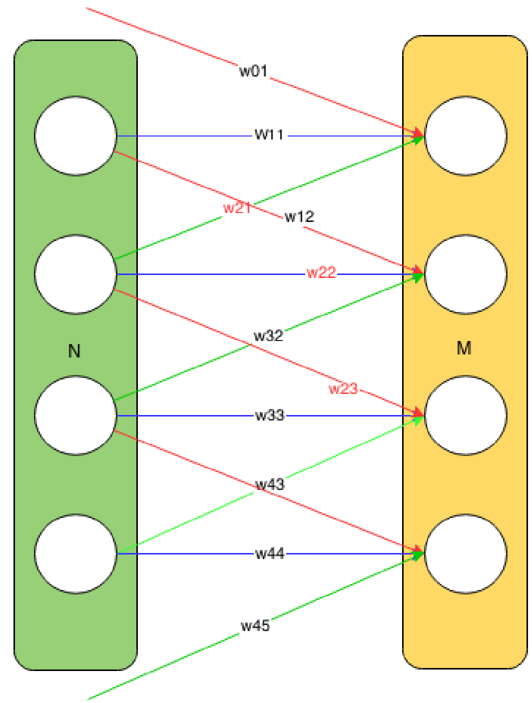




SHARED WEIGHTS



Fully-Connected (Dense)



Locally-Connected with Shared Weights

DISCUSSION

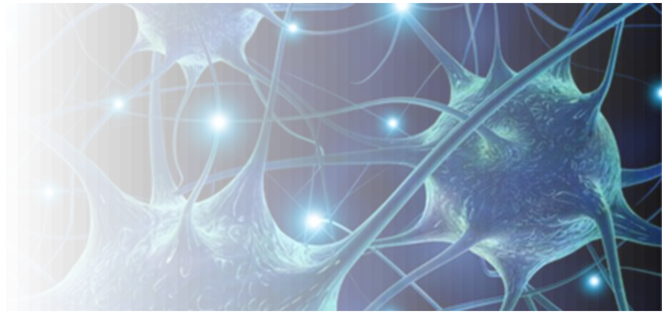
Question 1.

How is backpropagation and fine-tuning performed?

Question 2.

How to introduce rotation, skew, scale invariance?

TENSORFLOW



ONLINE TUTORIAL

Basic Usage.

- https://www.tensorflow.org/versions/r0.11/get_started/basic_usage.html

Simple MNIST Example.

- https://www.tensorflow.org/versions/r0.11/tutorials/mnist/beginners/index.html#mnist-for-ml-beginners

Convolutional Neural Networks.

- https://www.tensorflow.org/versions/r0.11/tutorials/mnist/pros/index.html#deep-mnist-for-experts

SUMMARY

- Multilayer Neural Networks
  - Neuron
  - Activation Function
  - Forward Propagation
- Convolutional Neural Networks
  - Convolution
  - Pooling
  - Shared Weights
- Learning Algorithm
  - Cost Function
  - Backpropagation
  - Sparse Autoencoder
  - Greedy Initialization