Brain-Inspired Learning Machines

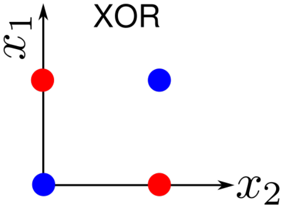
Deep Neural Networks

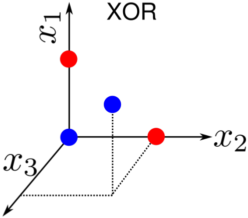
Emre Neftci

Department of Cognitive Sciences, UC Irvine,

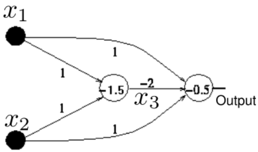
October 27, 2016

XOR can be solved with an intermediate perceptron



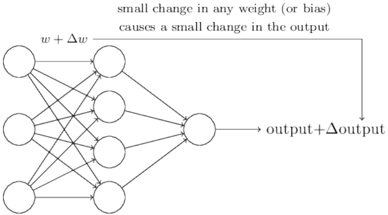


- We need an intermediate unit that is on only when x_1 and x_2 are both on.
- XOR gate with two perceptrons



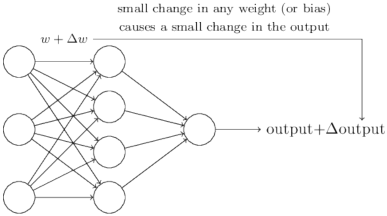
Last week: learning *shallow* networks

Multilayer (deep) networks are more powerful: is it possible to train a multilayer Perceptrons?



CREDIT ASSIGNMENT PROBLEM: which hidden unit weight should we modify to reach a target output?

Multilayer (deep) networks are more powerful: is it possible to train a multilayer Perceptrons?



CREDIT ASSIGNMENT PROBLEM: which hidden unit weight should we modify to reach a target output?

Solution: Back-propagation (Demo next week) http://playground.tensorflow.org/

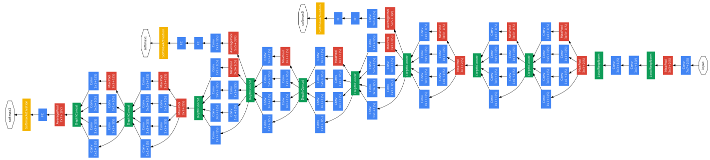
Deep neural networks

Before:

> *"How many hidden layers and how many units per layer do we need? The answer is at most two"*

Hertz, Krogh, and Palmer,, 1991

Now:



Szegedy et al., *arXiv preprint arXiv:1409.4842*, 2014

What happened between 1990s and now?

Good old online backpropagation for plain multilayer perceptrons yields a very low 0.35% error rate on the MNIST handwritten digits benchmark. All we need to achieve this best result so far are many hidden layers, many neurons per layer, numerous deformed training images to avoid overfitting, and graphics cards to greatly speed up learning.

Cireşan, Meier, Gambardella, and Schmidhuber, *Neural computation*, 2010

Better hardware, bigger data and tricks!

neon_mlp_extract.py

```
# setup model layers
layers = [Affine(nout=100, init=init_norm, activation=Rectlin()),
          Affine(nout=10, init=init_norm, activation=Logistic(shortcut=True))]

# setup cost function as CrossEntropy
cost = GeneralizedCost(costfunc=CrossEntropyBinary())

# setup optimizer
optimizer = GradientDescentMomentum(
        0.1, momentum_coef=0.9, stochastic_round=args.rounding)
```

How many hidden units?

- A single layer is enough to approximate any continuous function, but number of hidden units may grow exponentially with the number of inputs.
- More hidden layers improves representational power but can slow down learning (due to vanishing gradients)
- Networks with more hidden nodes are slower to compute
- Too many hidden nodes and layers can overfit (due to increased # parameters)

No general rule for choosing the best number of units and layers

Rule of thumb:

- "Number of hidden units somewhere between input size and output size"

Challenges in computer vision

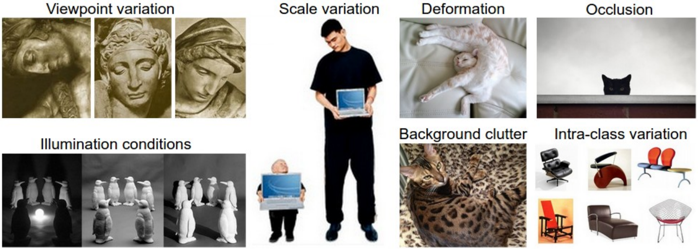
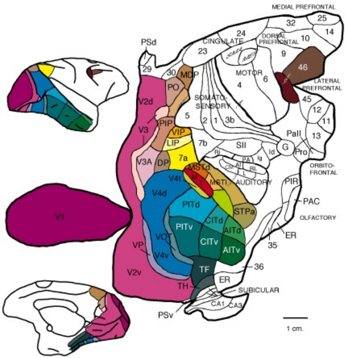
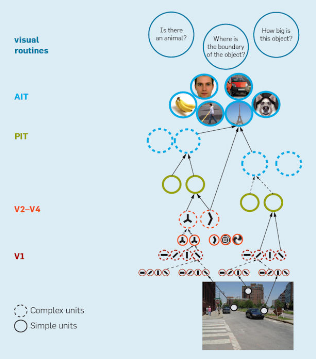


Image from Stanford CS231n Convolutional Neural Networks for Visual Recognition Class

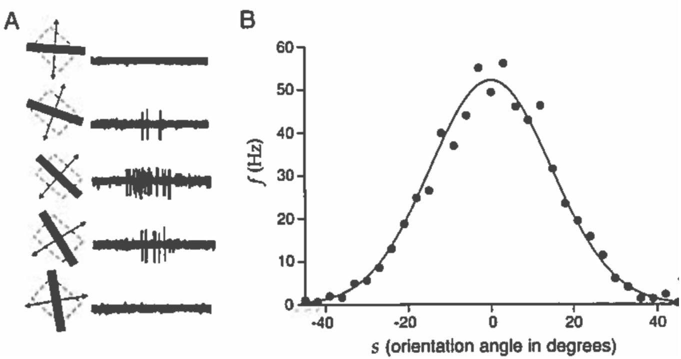
Hierarchical Organization of the Visual Pathway



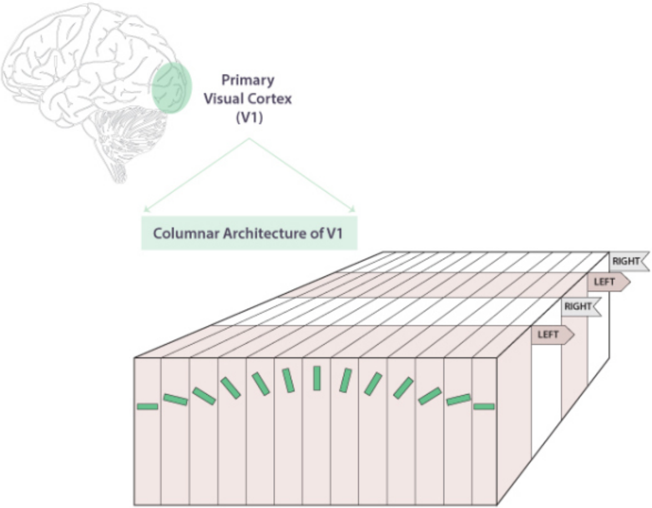


Felleman and Van Essen, 1991 (left), Cerebral Cortex 1:1-47. Serre and Poggio, 2007 (right)

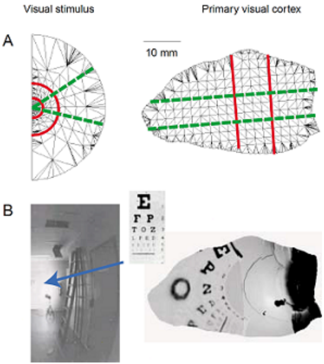
Neurons higher in the hierarchy represent more abstract features



Hubel & Wiesel, 1968


Primary
Visual Cortex
(V1)
Columnar Architecture of V1
RIGHT
LEFT
RIGHT
LEFT

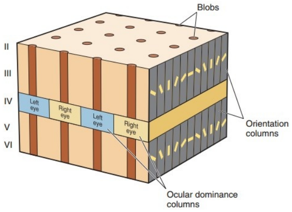
Retinotopic Map

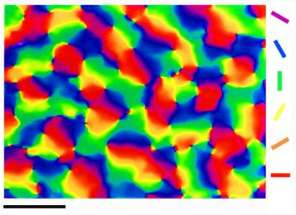


Matteo Carandini (2012), Scholarpedia, 7(7):12105

Nearby points in visual field project to nearby neurons in V1

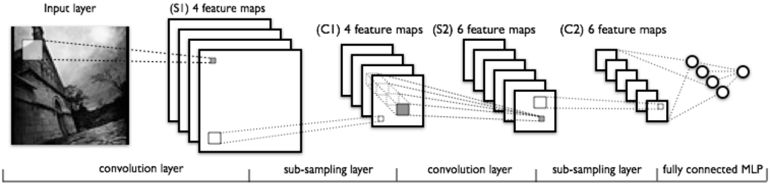
Columnar Organization of V1





Right: Blasdel & Salama (1986)

The receptive fields are tiled to cover the entire visual field

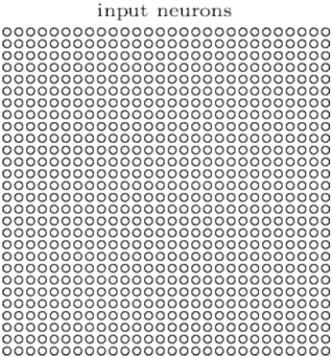


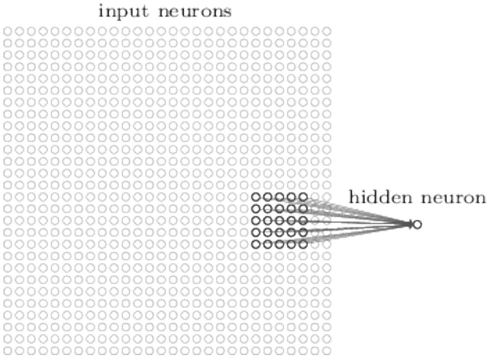
The visual cortex and neural networks solve the same task: use **retinotopy**, **local receptive fields** and **hierarchy** to constrain fully connected neural networks.

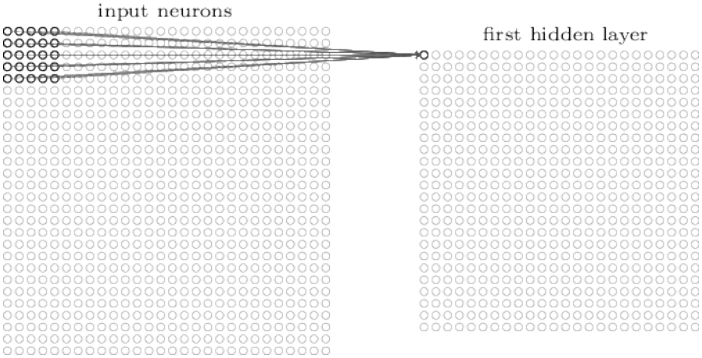
Two new type of layers:

- Convolutions
- Sub-sampling layers (pooling)

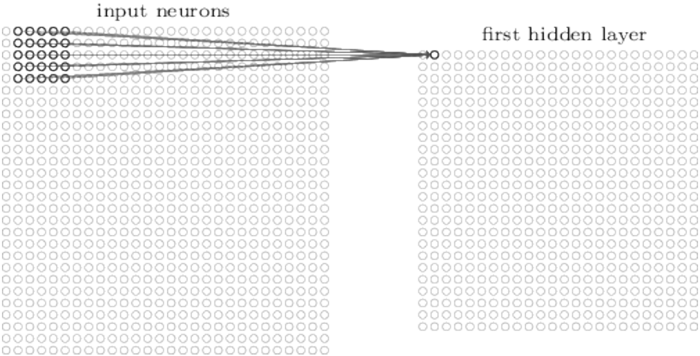
Convolution Layer




input neurons
hidden neuron

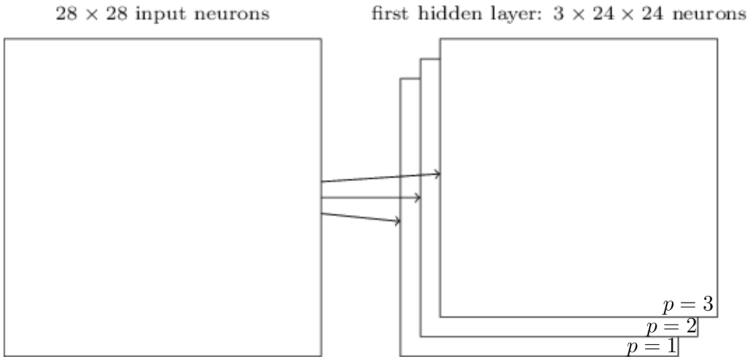


$$\text{output hidden unit (0,0)} = \sigma\left(b_0 + \sum_{l=0}^{5}\sum_{m=0}^{5} w_{lm}\, input_{0+l,0+m}\right).$$



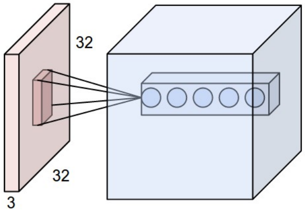
$$\text{output hidden unit (1,0)} = \sigma\left(b_1 + \sum_{l=0}^{5}\sum_{m=0}^{5} w_{lm} input_{1+l,1+m}\right).$$

Convolution Layer



output hidden unit (i, j) for feature $p = \sigma\left(b_{i,j,p} + \sum_{l=0}^{5}\sum_{m=0}^{5} w_{lm}^{p} input_{i+l,j+m}\right)$.

Multiple features

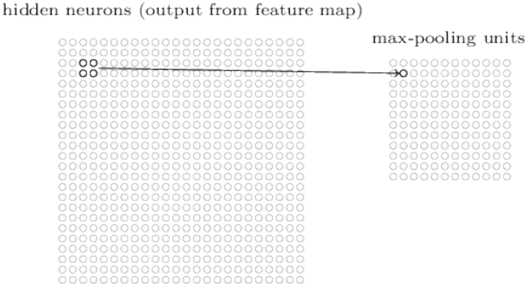


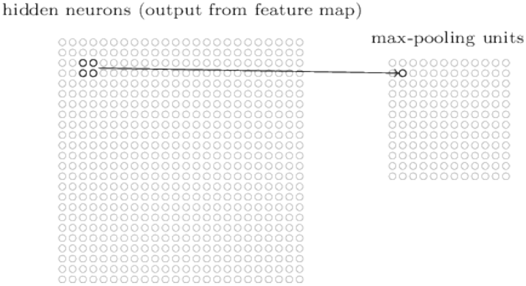
Stanford CS231n class slides

Convolutional layer parameters:

- Depth: Number of filters we would like to us
- Stride: The number of pixels by which we slide the filter
- Padding: Coping with boundaries by adding zeros around the input

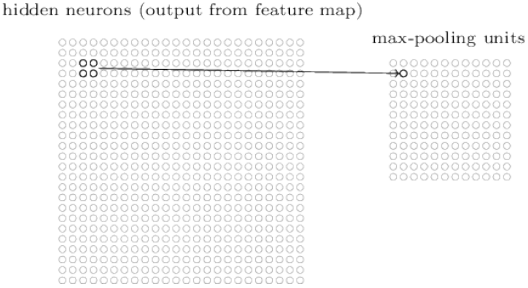
Pooling





max-pooling unit $(i, j) = \max\left(input_{i+0,j+0}, input_{i+0,j+1}, input_{i+1,j+0}, input_{i+1,j+1}\right)$

Pooling



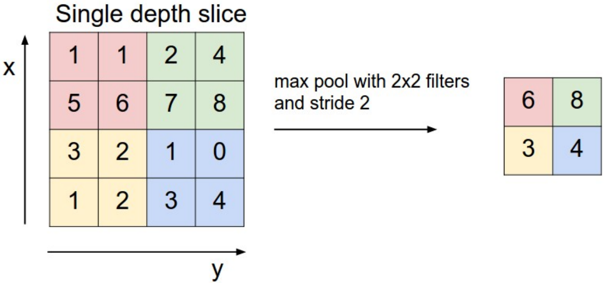
$$\text{max-pooling unit } (i, j) = \max\left(input_{i+0,j+0}, input_{i+0,j+1}, input_{i+1,j+0}, input_{i+1,j+1}\right)$$

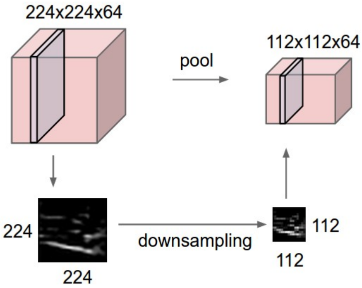
$$\text{mean-pooling unit } (i, j) = \frac{1}{4}\left(input_{i+0,j+0} + input_{i+0,j+1} + input_{i+1,j+0} + input_{i+1,j+1}\right)$$

The precise location of a feature is not critical.

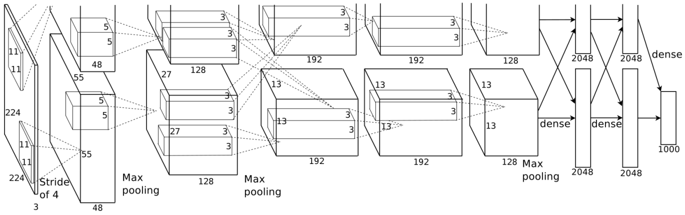
Pooling discards positional information to reduce dimensionality of the layer (down-sampling)

Example: max pooling



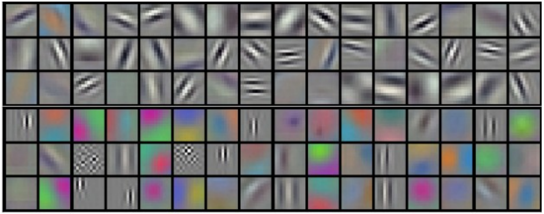


Stanford CS231n class slides

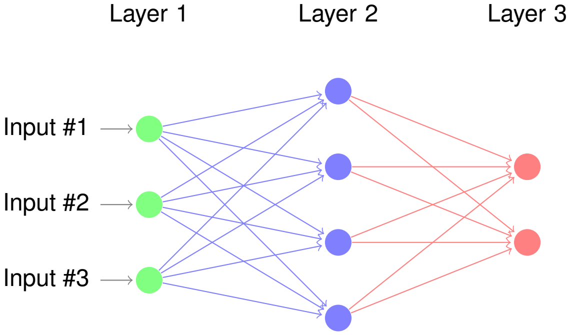


Krizhevsky, Sutskever, and Hinton, *Advances in neural information processing systems*, 2012

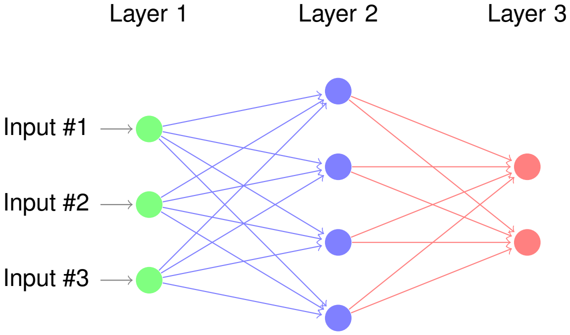
Features obtained at the first convolutional layer



w_{ij}^{L}: means weight j of unit i in layer $L = 1, 2, 3, ...$



w_{ij}^{L}: means weight j of unit i in layer $L = 1, 2, 3, \ldots$.



$$\frac{\partial C_{\text{MSE}}}{\partial w_{jk}^{2}} = 2 \sum_{\text{training set}} \sum_{i} (y_i - t_i)\sigma'(a_i^3)\frac{\partial a_i^3}{\partial w_{jk}^2}.$$

$$\frac{\partial a_i^3}{\partial w_{jk}^2} = \sum_{j'} w_{ij'}^3 \sigma'(a_{j'}^2)\frac{\partial a_{j'}^2}{\partial w_{jk}^2}$$

$$\frac{\partial a_{j'}^2}{\partial w_{jk}^2} = x_k^1 \quad \text{for } j = j'\text{, otherwise } 0$$

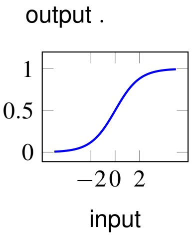
Backpropagation:

$$\frac{\partial C_{\text{MSE}}}{\partial w^2_{jk}} = 2 \sum_{\text{training set}} \sum_i (y_i - t_i)\sigma'(a^3_i)\frac{\partial a^3_i}{\partial w^2_{jk}}.$$

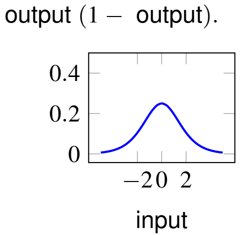
$$\frac{\partial a^3_i}{\partial w^2_{jk}} = \sum_{j'} w^3_{ij'}\sigma'(a^2_{j'})\frac{\partial a^2_{j'}}{\partial w^2_{jk}}$$

$$\frac{\partial a^2_{j'}}{\partial w^2_{jk}} = x^1_k \quad \text{for } j = j', \text{ otherwise } 0$$

Sigmoid unit



Derivative Sigmoid unit



- If an output is close to 0 or 1, the gradient is very small.
- Backpropagating means multiplying numbers typically smaller than 1.