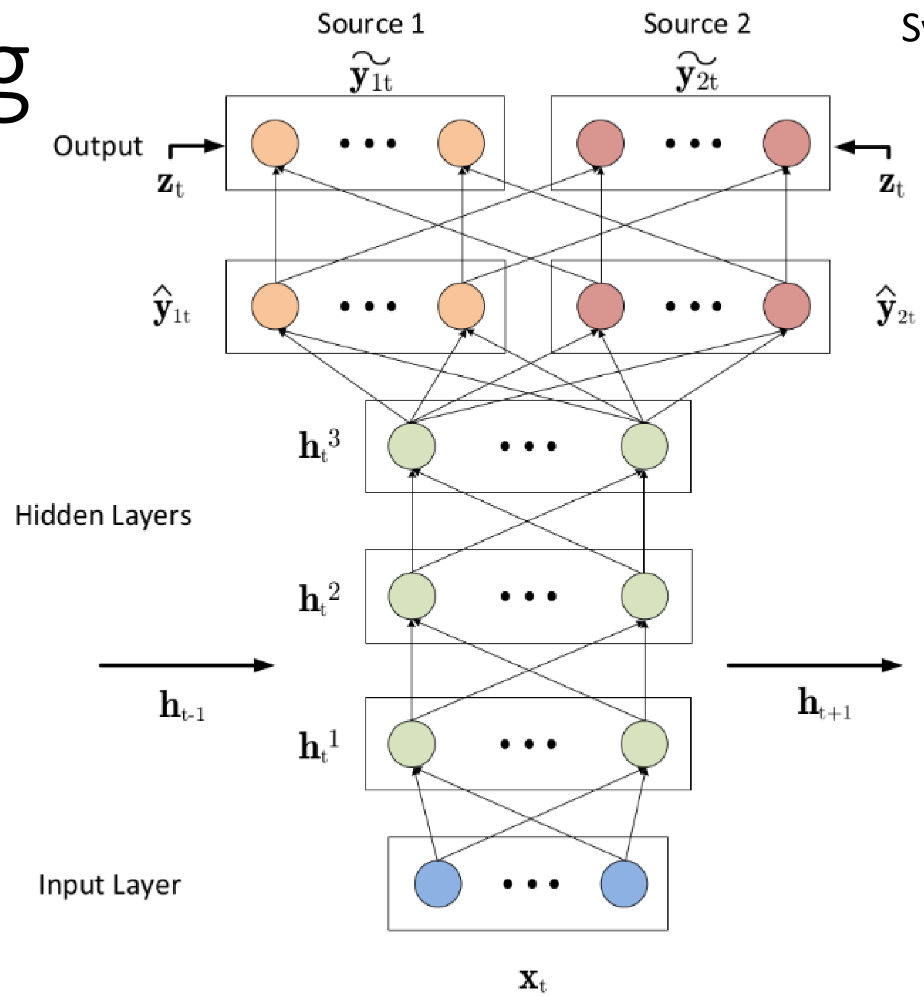
CS440/ECE448 Lecture 16: Deep Learning

Mark Hasegawa-Johnson, 3/2018

Including Slides by

Svetlana Lazebnik, 10/2016



Deep Learning

- Differentiable Perceptron/One-Layer Neural Net
- Two-Layer Neural Net
- Convolutional Neural Net
- Singing-Voice Separation Using Deep Recurrent Network
- Semantic Image Inpainting with Deep Generative Models

Differentiable Perceptron

Suppose we have n training vectors, $\vec{x}_1$ through $\vec{x}_n$. Each one has an associated label $y_i \in \{-1,1\}$. Then we replace the true error,

$$E = \frac{1}{4}\sum_{i=1}^{n}(y_i - \mathrm{sgn}(\vec{w}^T\vec{x}_i))^2$$

with a differentiable error

$$E = \frac{1}{4}\sum_{i=1}^{n}(y_i - \tanh(\vec{w}^T\vec{x}_i))^2$$

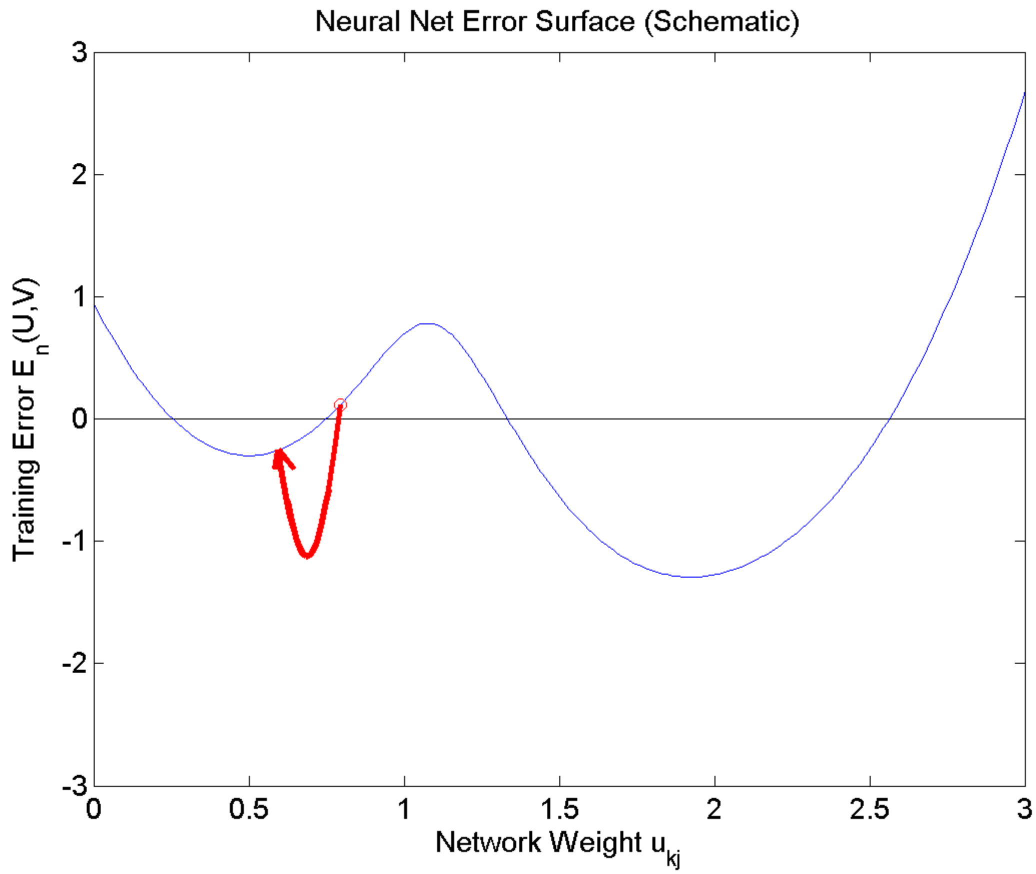
We calculate the form of the derivative in advance, so that we can plug in the particular values of $\vec{x}_i$, y_i and $\vec{w}$ at each training epoch:

$$\frac{\partial E}{\partial w_k} = \frac{1}{4}\sum_{i=1}^{n} 2(\tanh(\vec{w}^T\vec{x}_i) - y_i)\left(\frac{\partial \tanh(\vec{w}^T\vec{x}_i)}{\partial \vec{w}^T\vec{x}_i}\right)\left(\frac{\partial \vec{w}^T\vec{x}_i}{\partial w_k}\right)$$

Differentiable Perceptron

And then the weights get updated according to

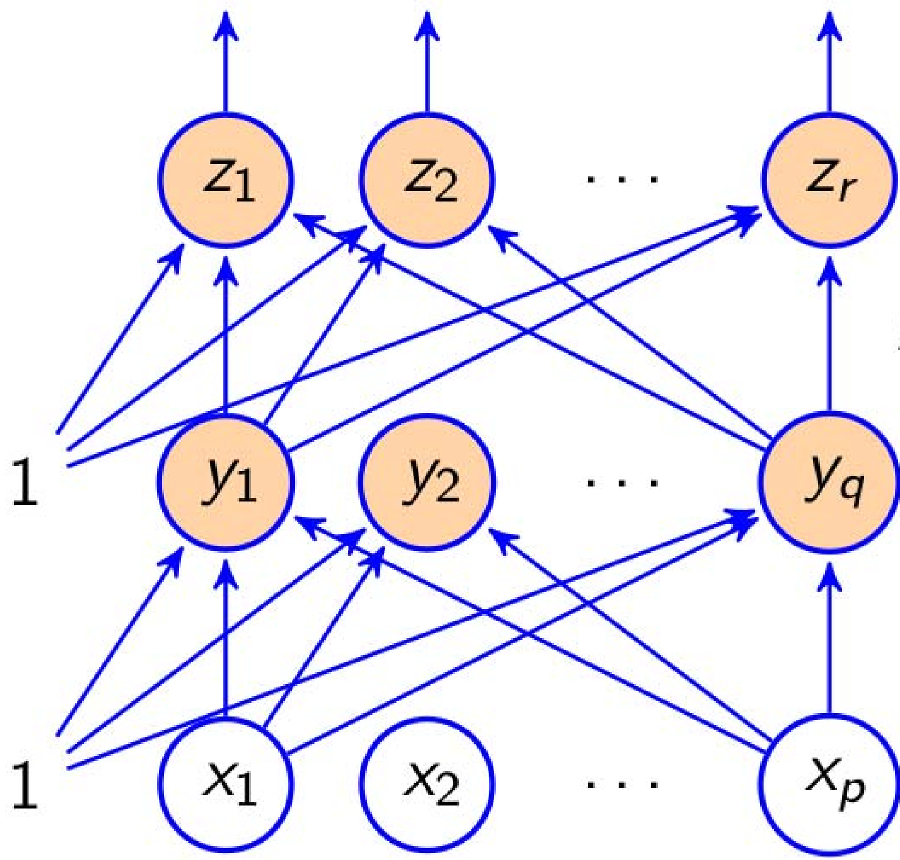
$$w_k = w_k - \eta \frac{\partial E}{\partial w_k}$$



Deep Learning

- Differentiable Perceptron/One-Layer Neural Net
- Two-Layer Neural Net
- Convolutional Neural Net
- Singing-Voice Separation Using Deep Recurrent Network
- Semantic Image Inpainting with Deep Generative Models

Two-Layer Neural Network



$$\vec{z} = h(\vec{x}, U, V)$$

$$z_\ell = g(b_\ell) \qquad \vec{z} = g(\vec{b})$$

$$b_\ell = v_{k0} + \sum_{k=1}^{q} v_{\ell k} y_k \qquad \vec{b} = V\vec{y}$$

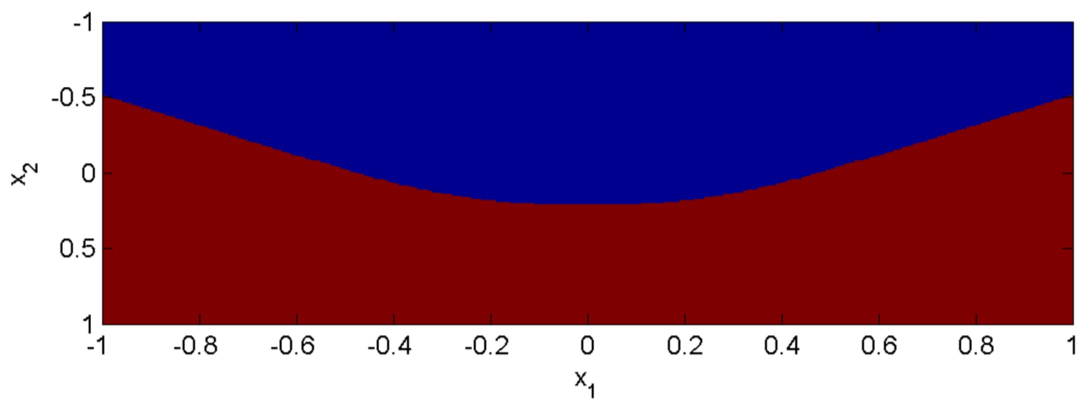
$$y_k = f(a_k) \qquad \vec{y} = f(\vec{a})$$

$$a_k = u_{k0} + \sum_{j=1}^{p} u_{kj} x_j \qquad \vec{a} = U\vec{x}$$

$\vec{x}$ is the input vector

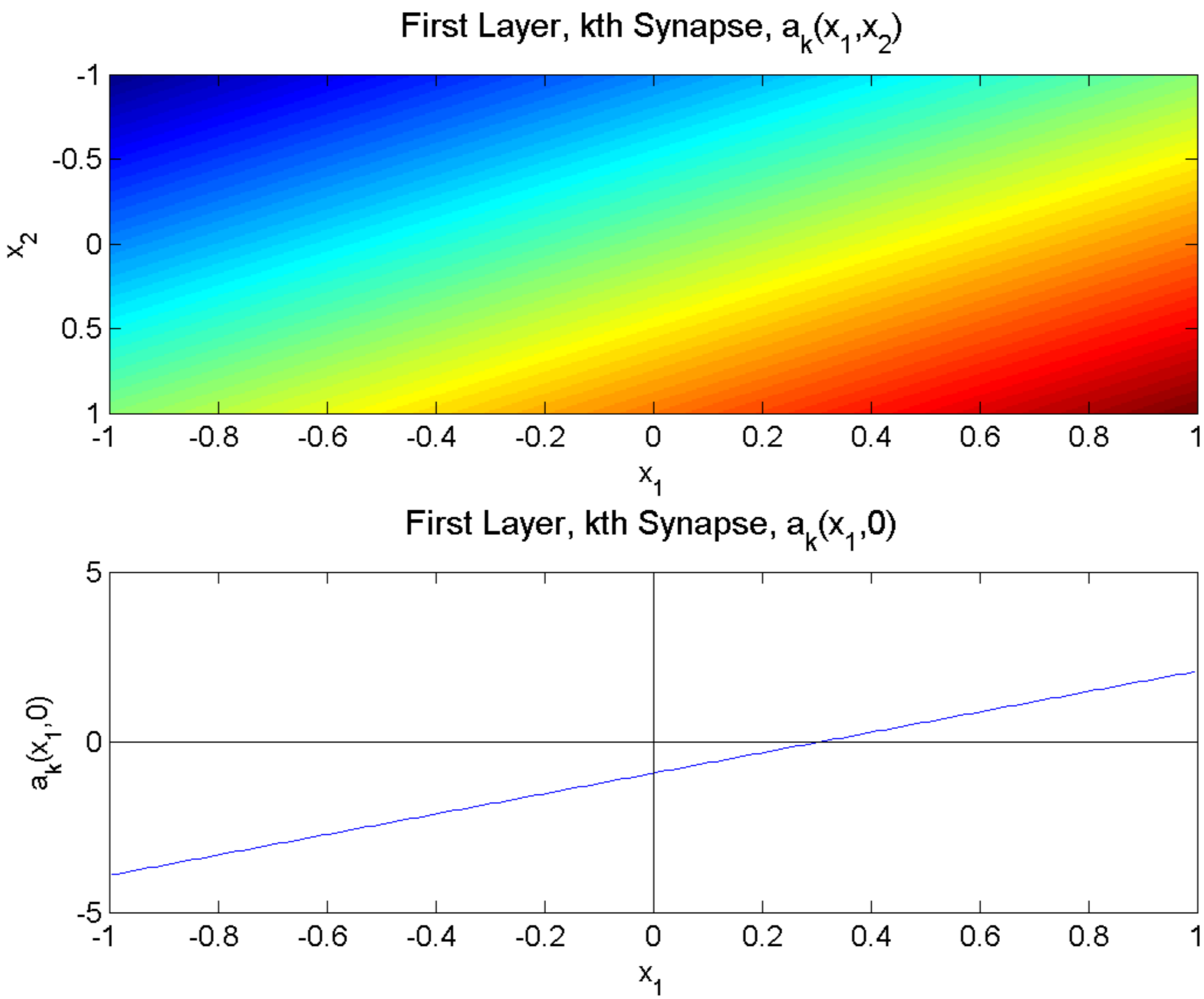
Two-Layer Neural Network

- WHY: Barron (1993) proved that a two-layer neural network is able to learn ANY function z=f(x), in the limit as the number of hidden nodes goes to infinity.
- EXAMPLE: suppose we want to learn the following classification boundary, meaning that objects with negative x_2 are always in "RED" class. Objects with positive values of x_2 might be RED or BLUE class, depending on x_1:

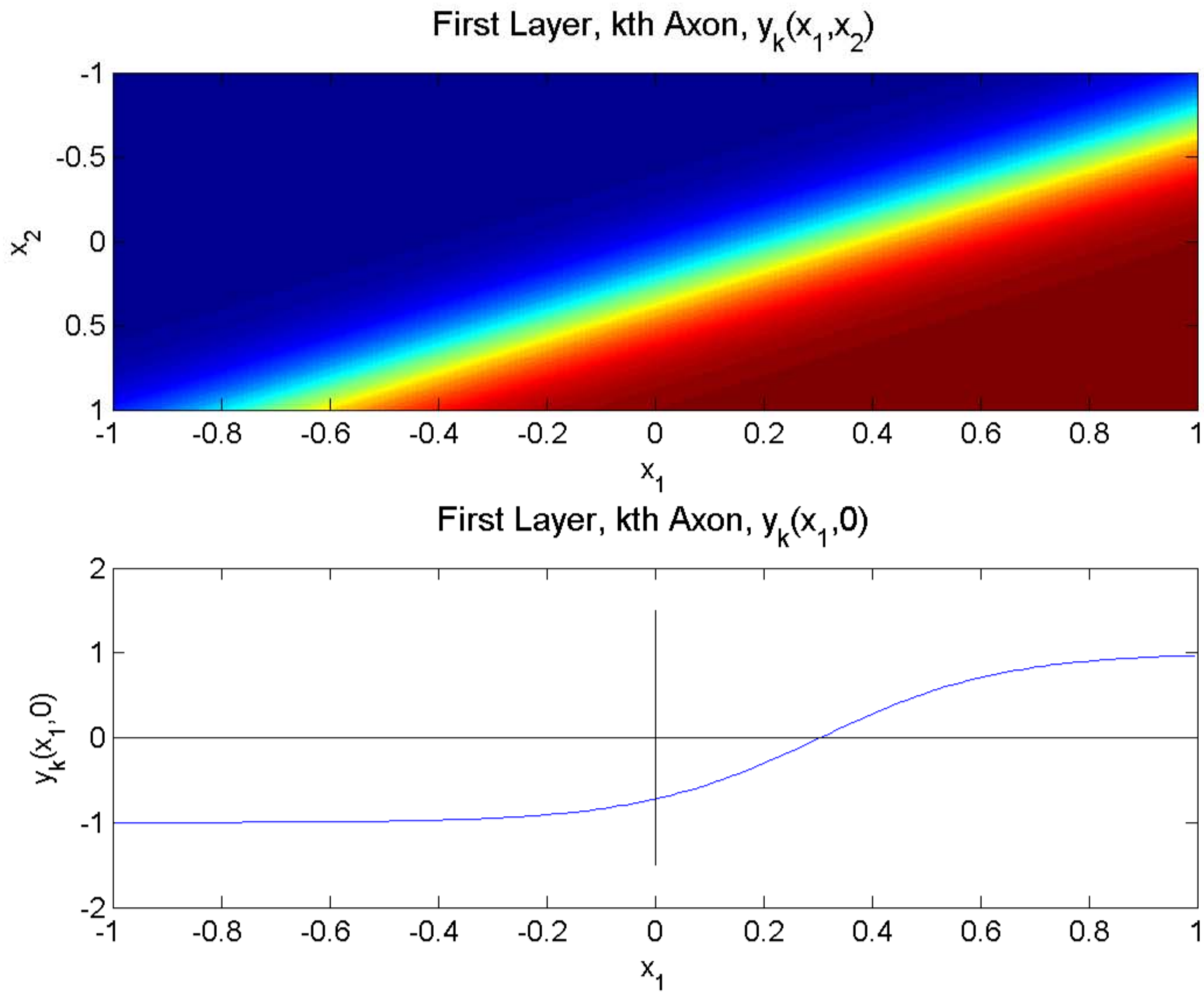


We can learn that using a two-layer neural net with just TWO hidden nodes. The next 4 slides will show you how.

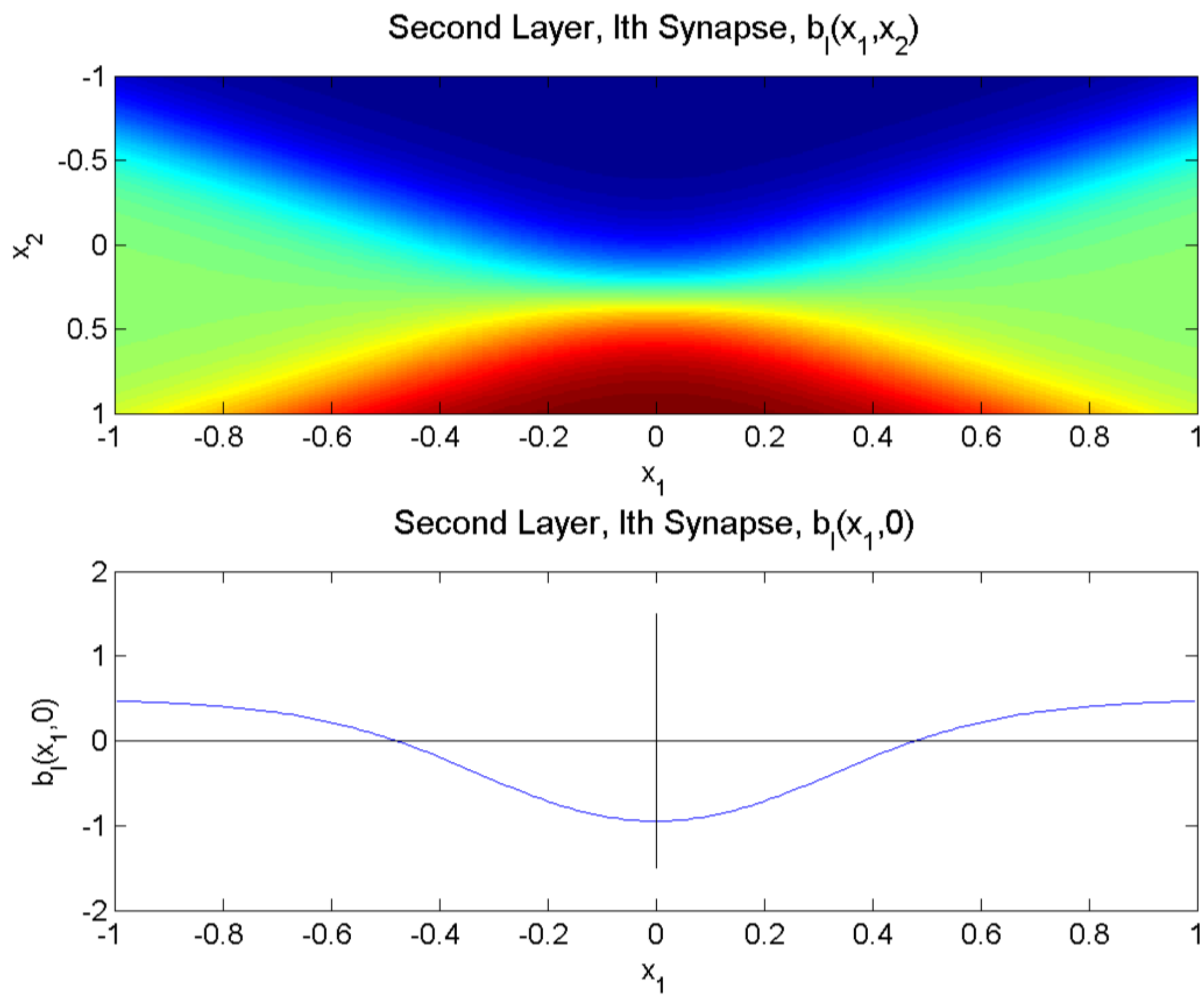
Synapse, First Layer: $a_k = u_{k0} + \sum_{j=1}^{2} u_{kj} x_j$



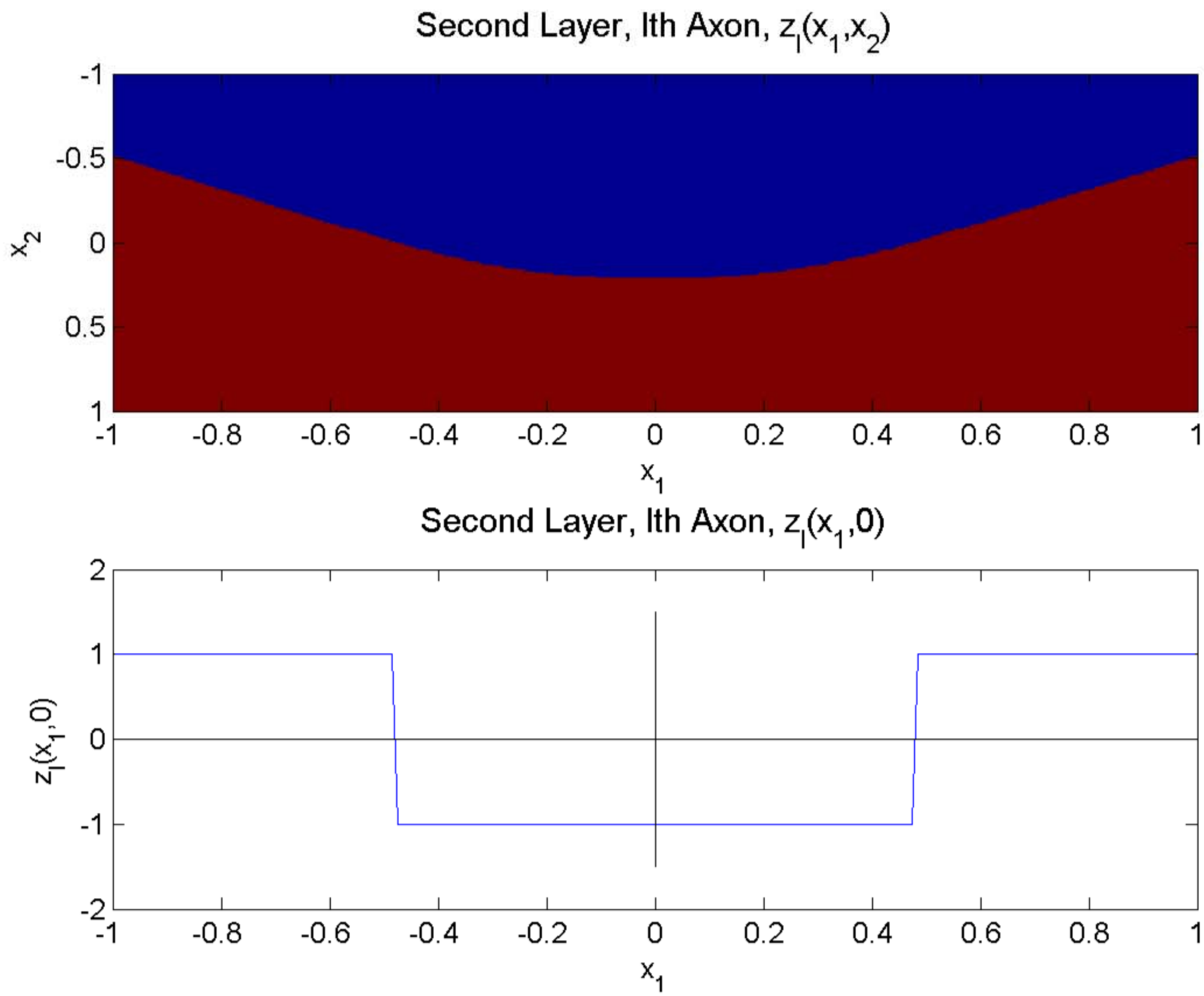
Axon, First Layer: $y_k = \tanh(a_k)$



Synapse, Second Layer: $b_\ell = v_{\ell 0} + \sum_{k=1}^{2} v_{\ell k} y_k$



Axon, Second Layer: $z_\ell = \mathrm{sign}(b_\ell)$



Training the Network

To train it, we just need to find the derivative of the error with respect to each network weight. If the error is

$$E = \frac{1}{2}\sum_{i=1}^{n}\left(z_{li} - \tanh\left(\sum_{k=0}^{q} v_{lk}\tanh\left(\sum_{j=0}^{p} u_{kj}x_{ji}\right)\right)\right)^2$$

Then its derivative is just

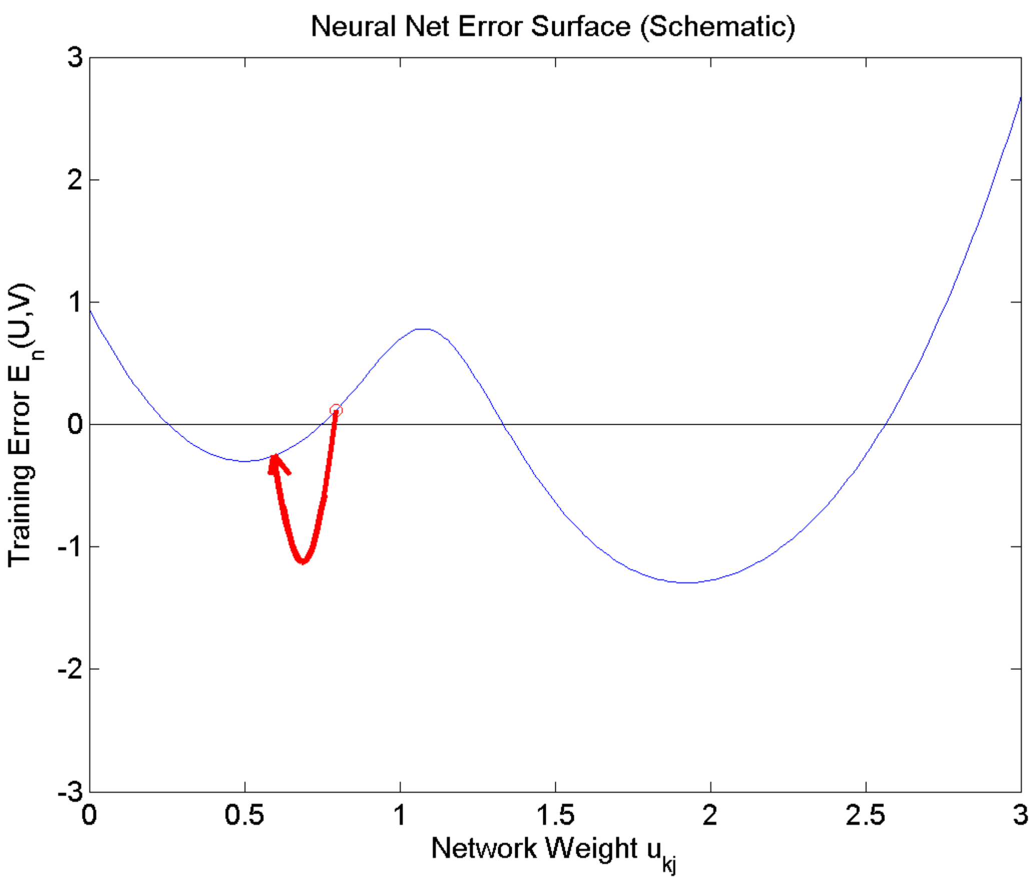
$$\frac{\partial E}{\partial u_{kj}} = \sum_{i=1}^{n}(\tanh(\blacksquare) - z_i)\left(\frac{\partial \tanh(\blacksquare)}{\delta \blacksquare}\right)\left(\sum_{k=0}^{q} v_{lk}\frac{\partial \tanh(\cdot)}{\delta\ \cdot}\right)x_{ji}$$

…where the values of x_{ji}, $\cdot$, and $\blacksquare$ need to be calculated and plugged in for each training token.

Two-layer Neural Net

And then we just train the network weights according to

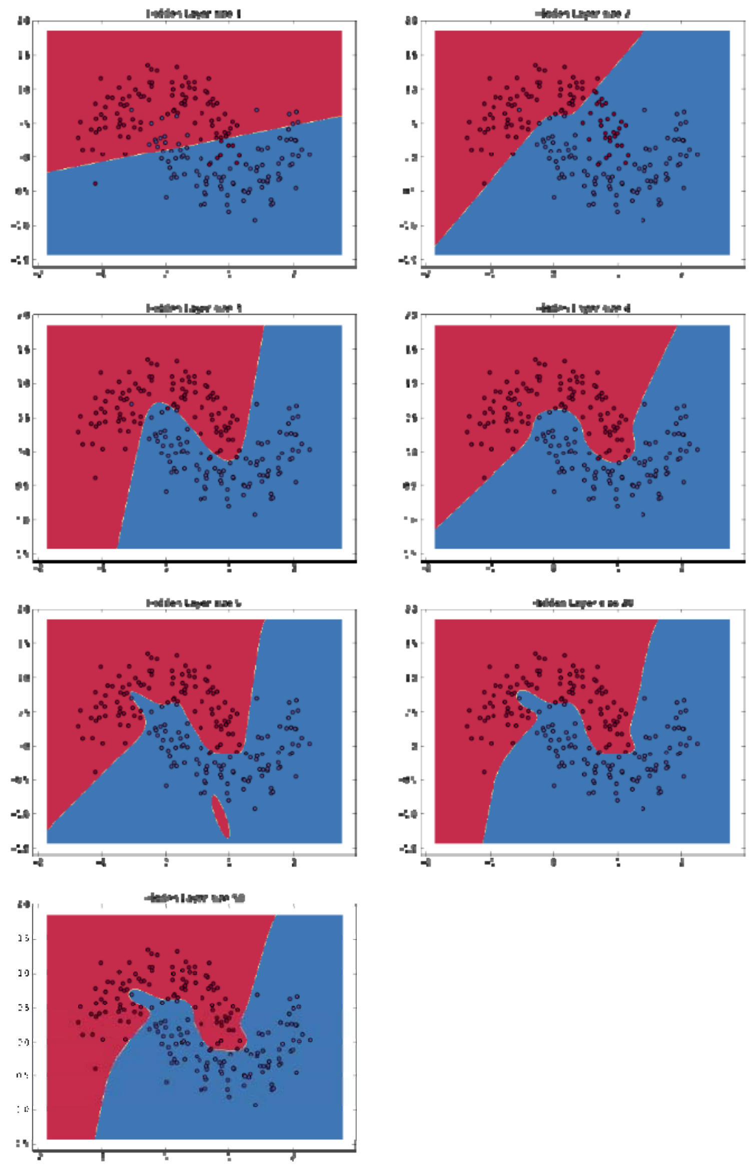
$$u_{kj} = u_{kj} - \eta \frac{\partial E}{\partial u_{kj}}$$



Example from a blog

Here's an example of training a two-layer network in python. It has a good display of the different decision boundaries you get with different #s of hidden nodes.

https://medium.com/ml-algorithms/neural-networks-for-decision-boundary-in-python-b243440fb7d1

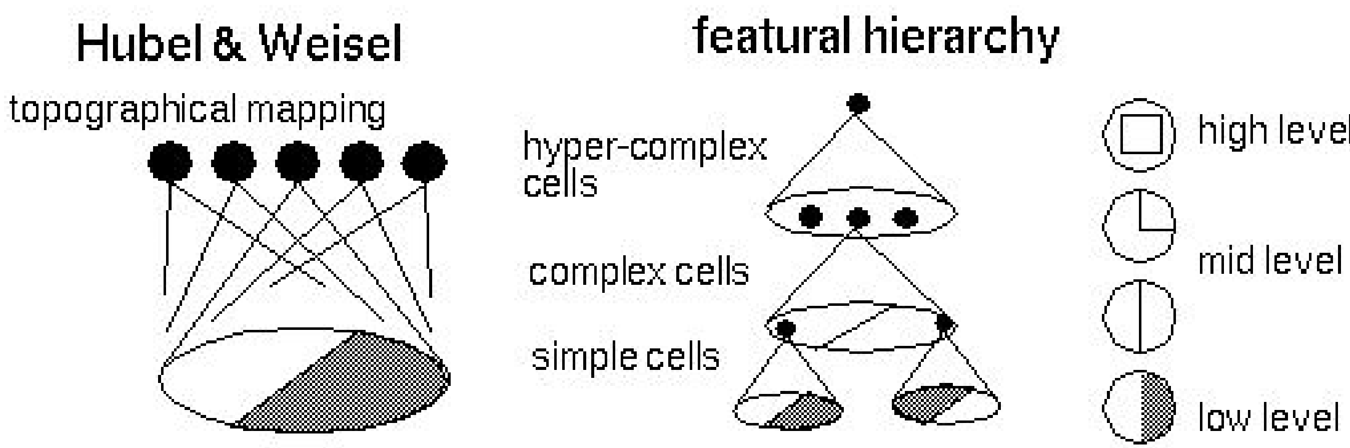


Deep Learning

- Differentiable Perceptron/One-Layer Neural Net
- Two-Layer Neural Net
- Convolutional Neural Net
- Singing-Voice Separation Using Deep Recurrent Network
- Semantic Image Inpainting with Deep Generative Models

Biological inspiration

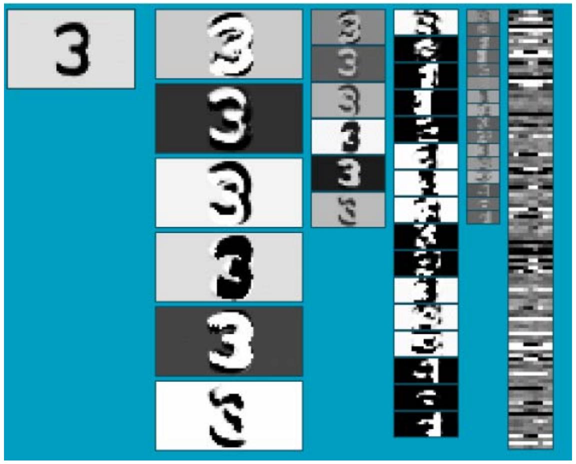
- D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981)
  - Visual cortex consists of a hierarchy of *simple*, *complex*, and *hyper-complex* cells

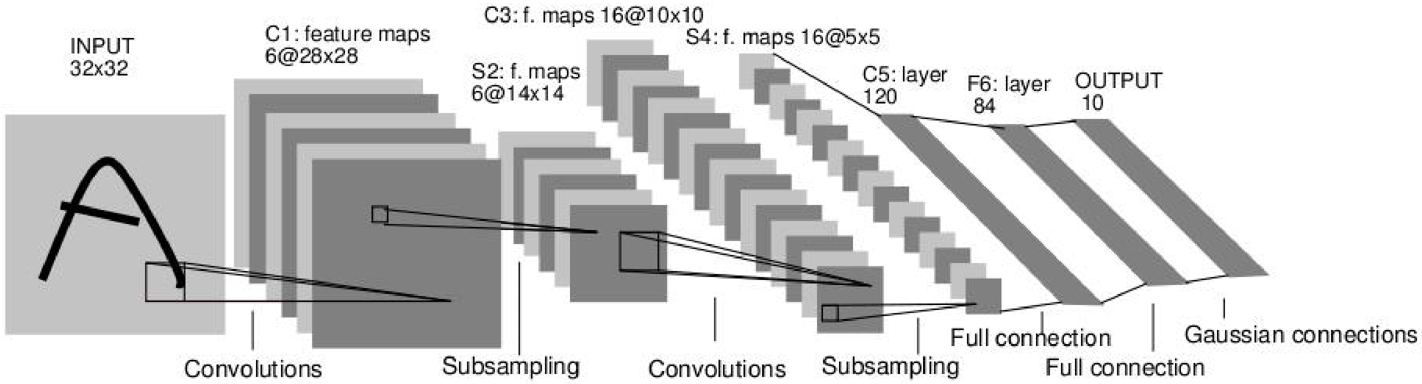


Source

Convolutional Neural Networks

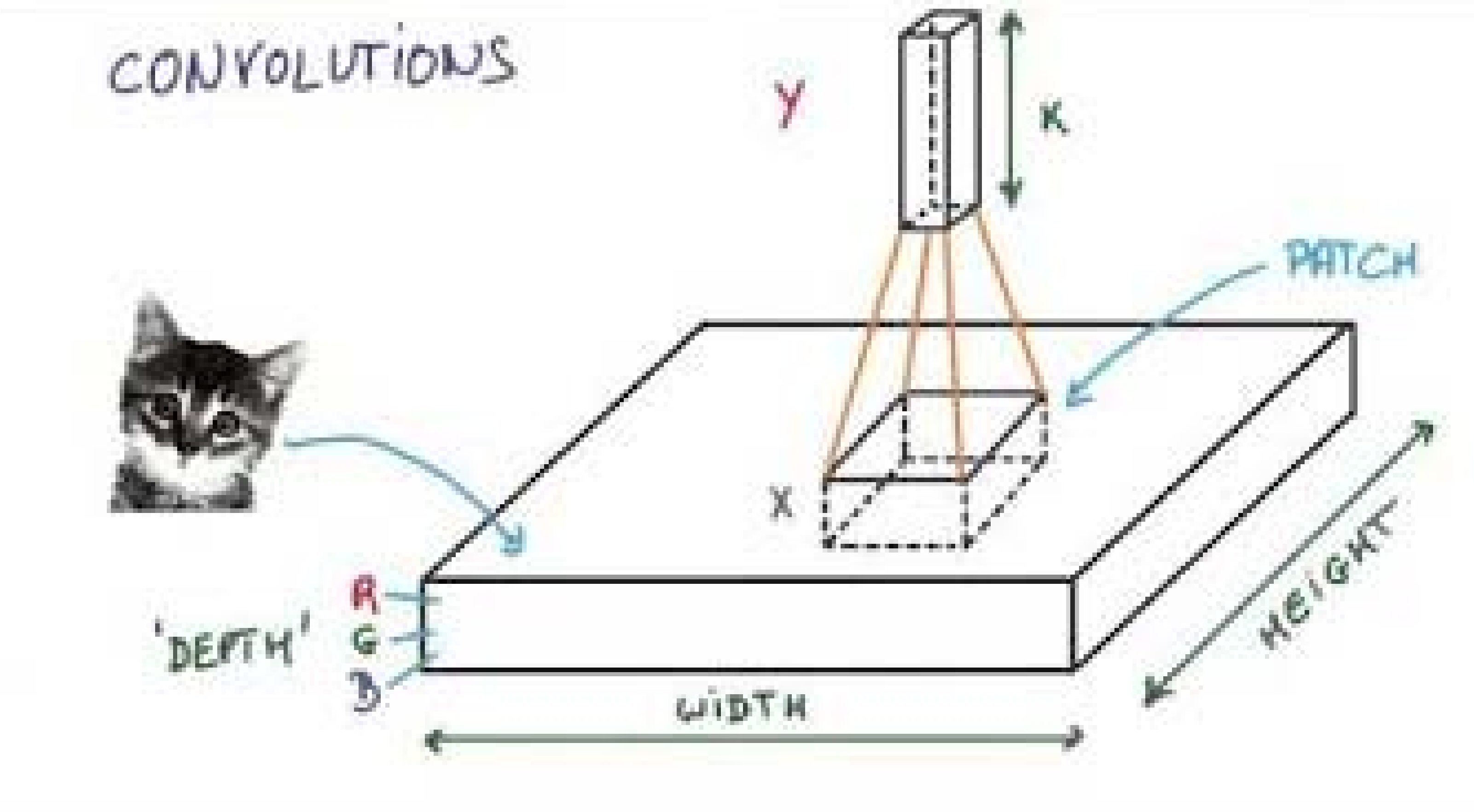
- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant features
- Classification layer at the end





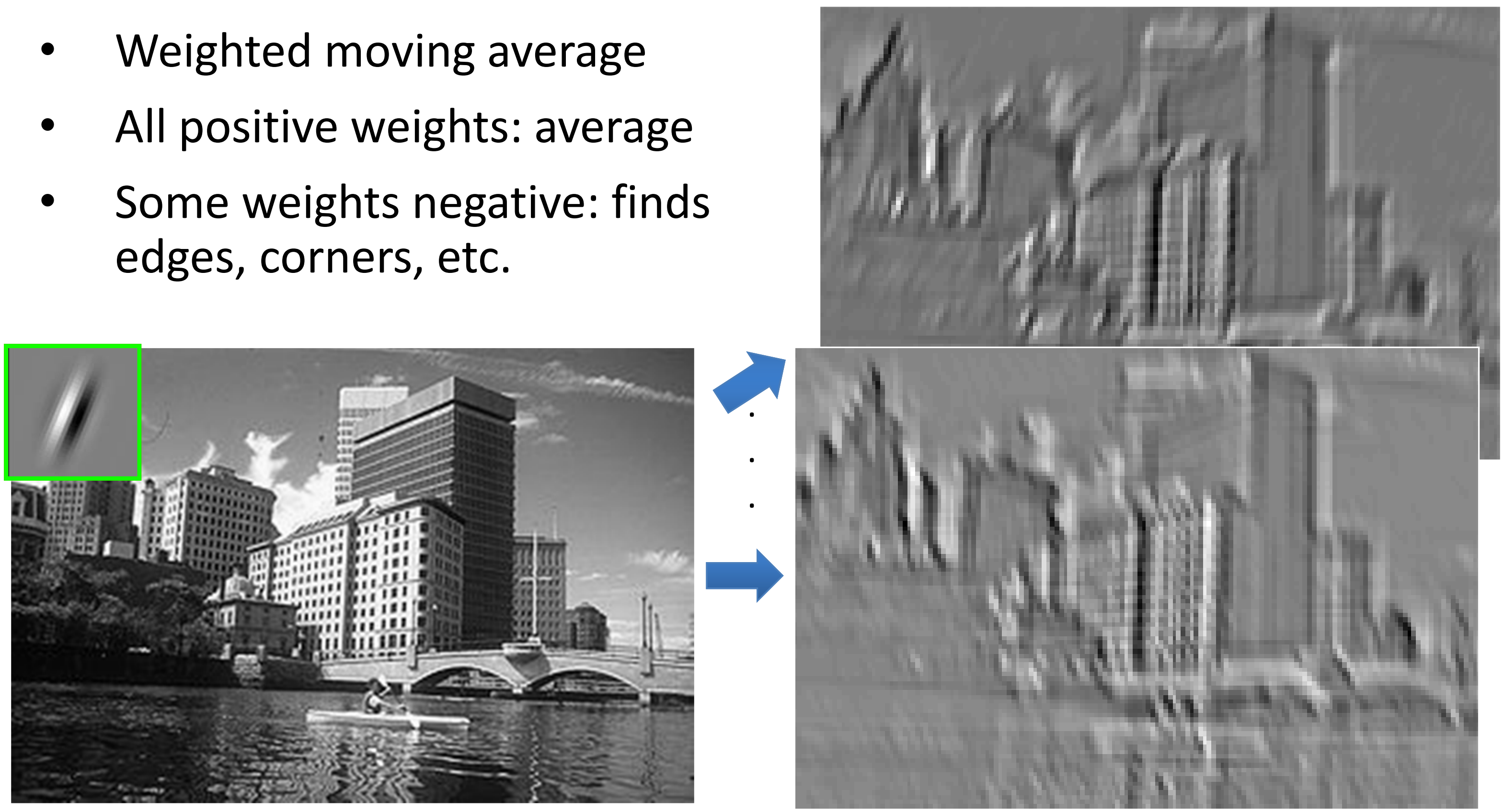
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, [Gradient-based learning applied to document recognition](), Proc. IEEE 86(11): 2278–2324, 1998.

CNN (Convolutional Neural Networks)



What is a convolution?

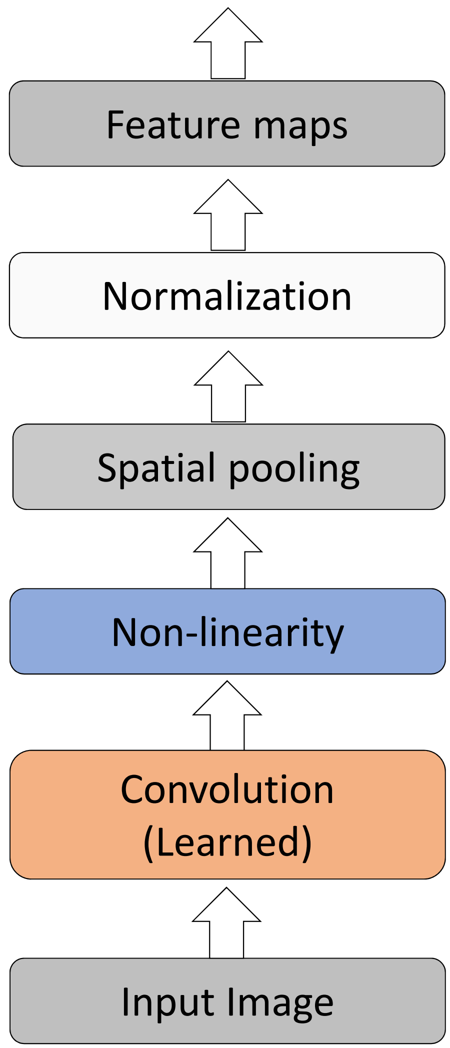
- Weighted moving average
- All positive weights: average
- Some weights negative: finds edges, corners, etc.



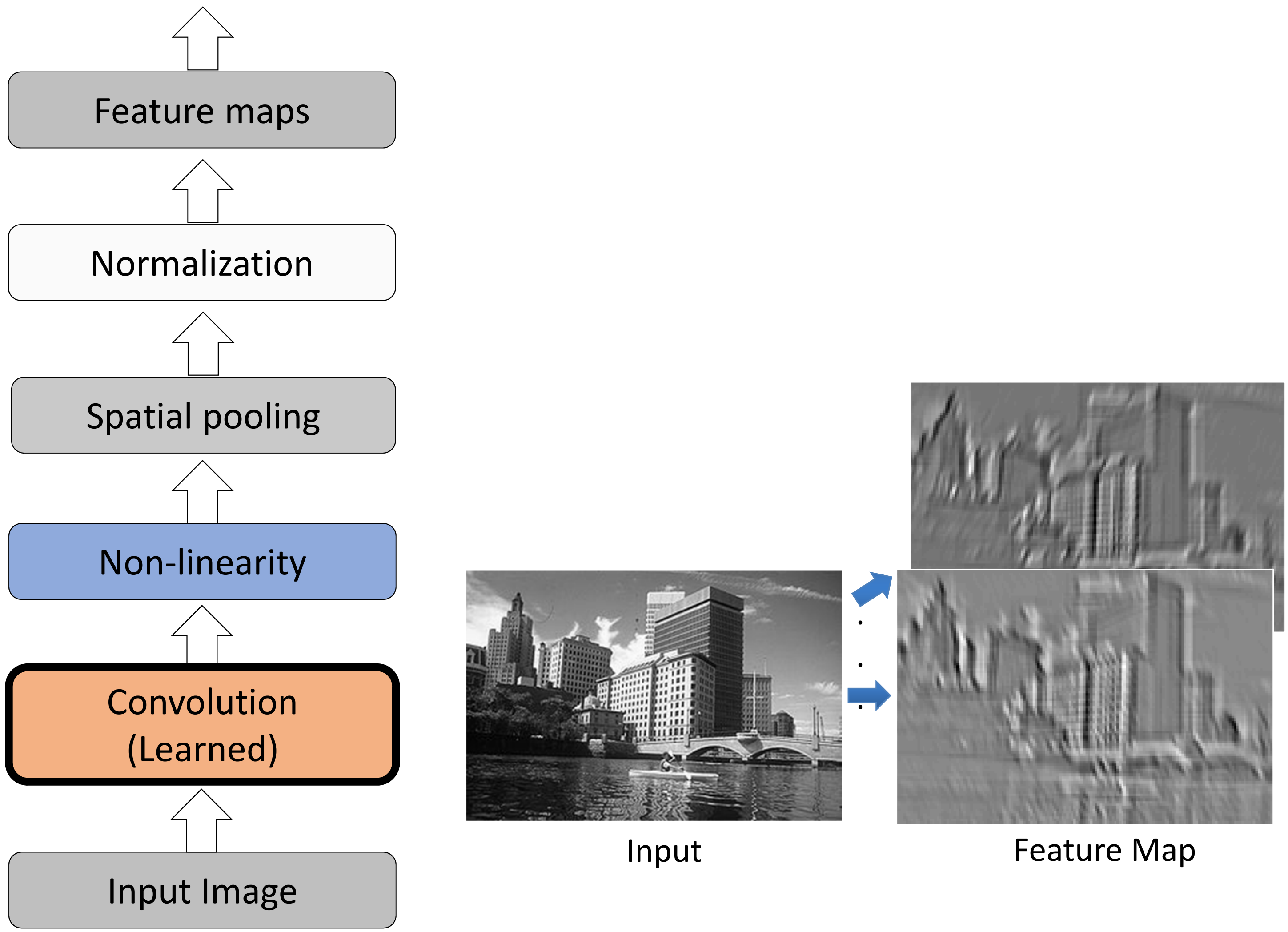
Input

Feature Map

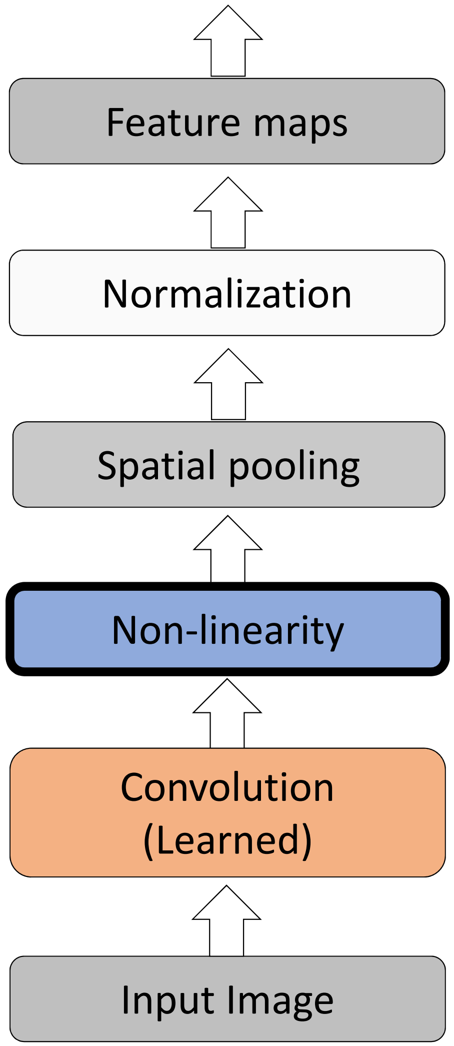
Convolutional Neural Networks

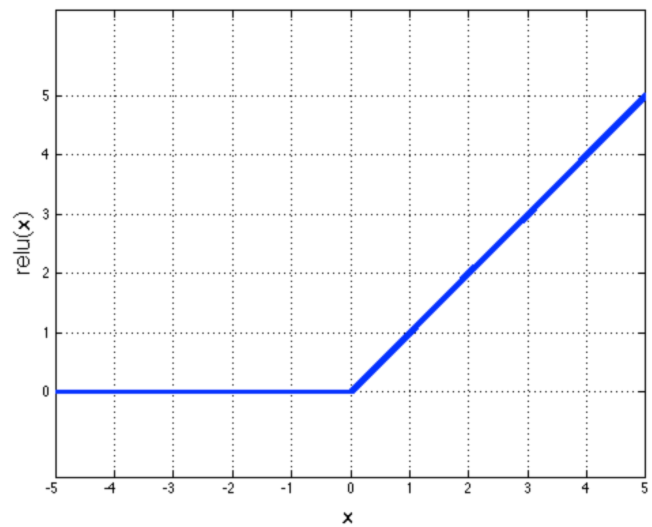


Convolutional Neural Networks



Convolutional Neural Networks





Convolutional Neural Networks

Feature maps

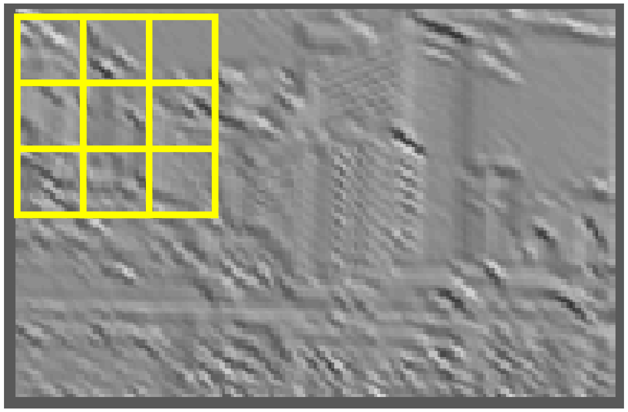
Normalization

Spatial pooling

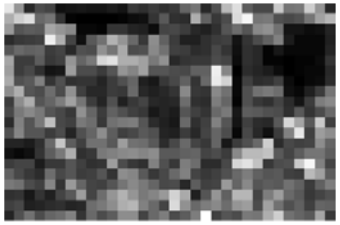
Non-linearity

Convolution
(Learned)

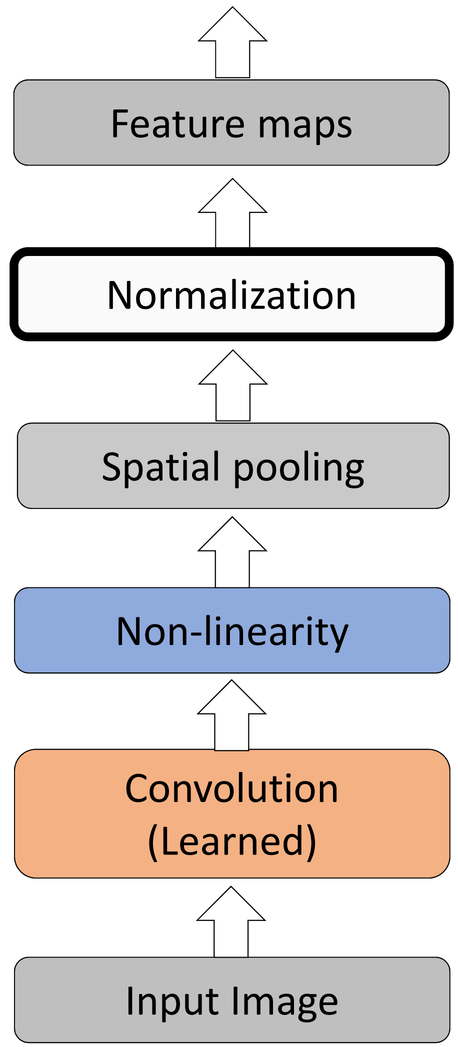
Input Image

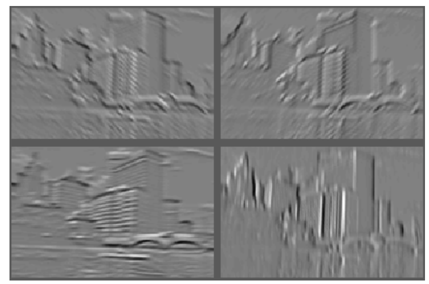


Max

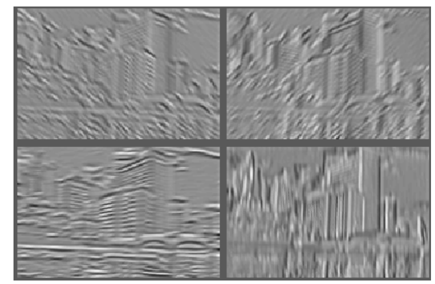


Convolutional Neural Networks



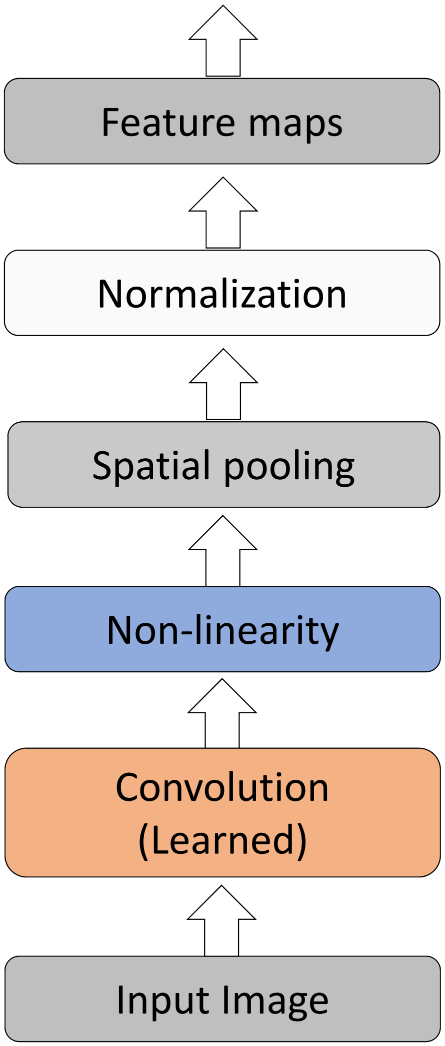


Feature Maps



Feature Maps
After Contrast
Normalization

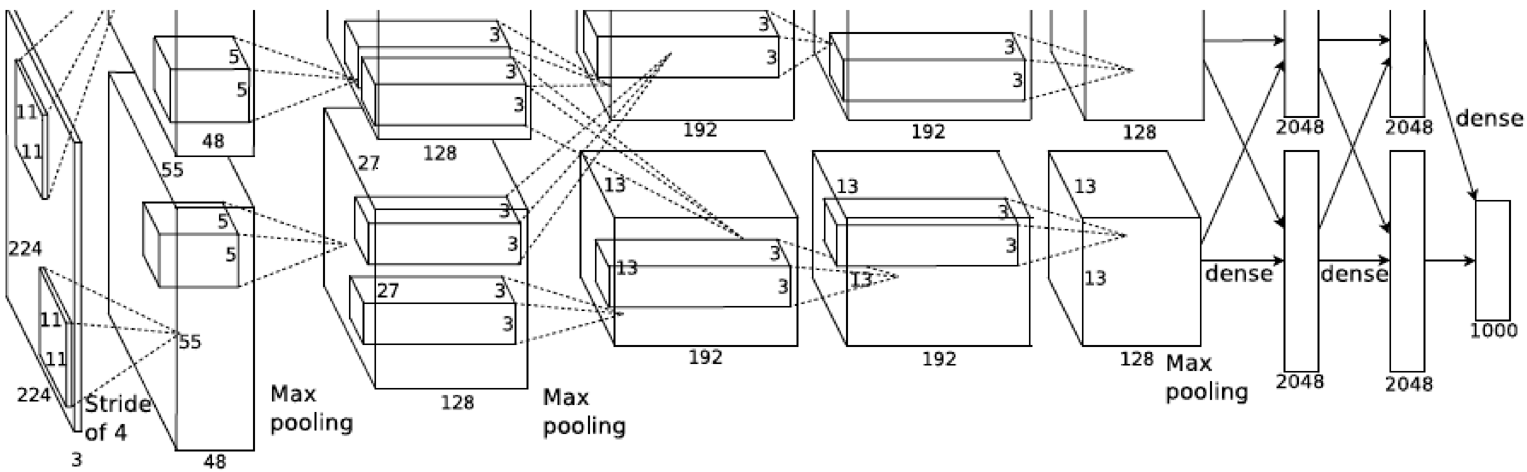
Convolutional Neural Networks



- Convolutional filters are trained in a supervised manner by back-propagating classification error
- Basically, you can think of the top layer as a "linear classifier," and the layer below it learns features. And its features are computed from the outputs of the layer below that, and so on.

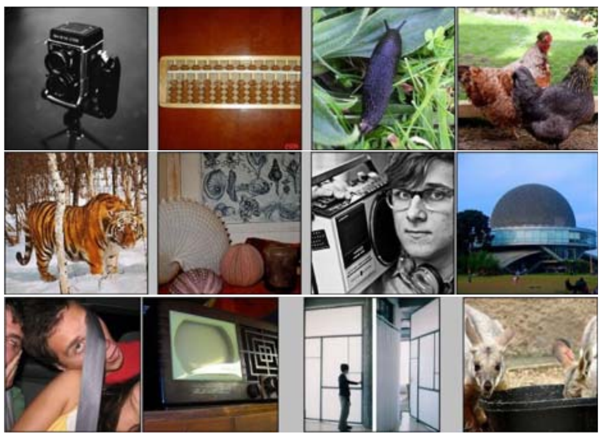
AlexNet

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week



A. Krizhevsky, I. Sutskever, and G. Hinton, [ImageNet Classification with Deep Convolutional Neural Networks](), NIPS 2012

ImageNet Challenge

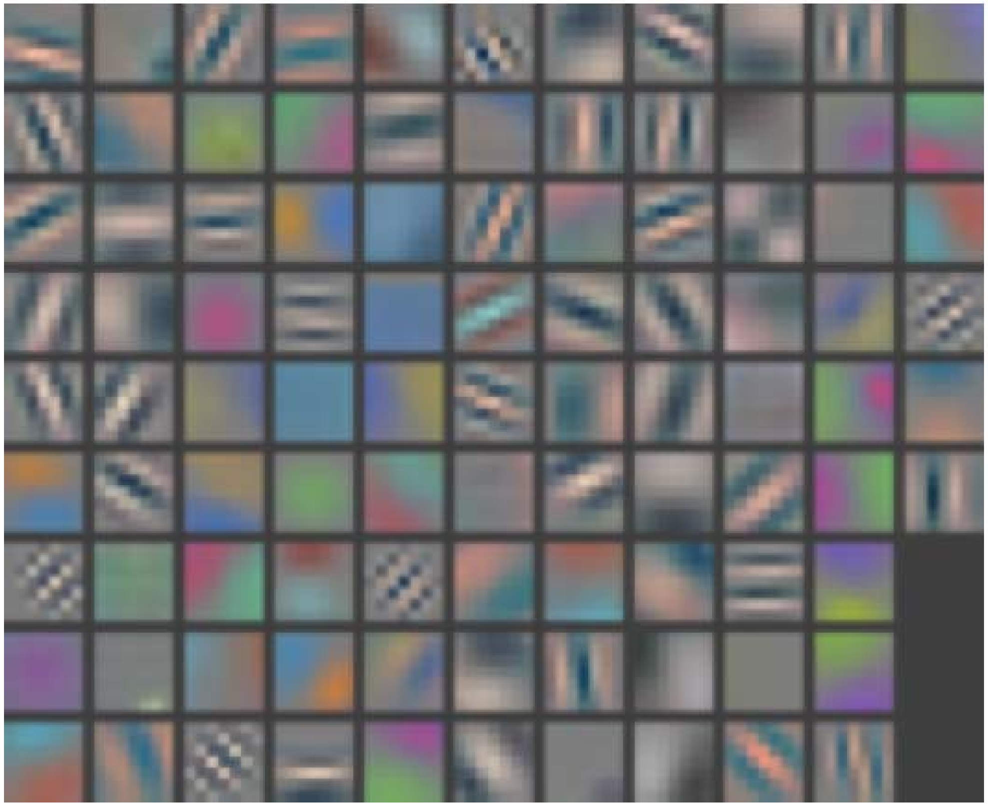


[Deng et al. CVPR 2009]

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

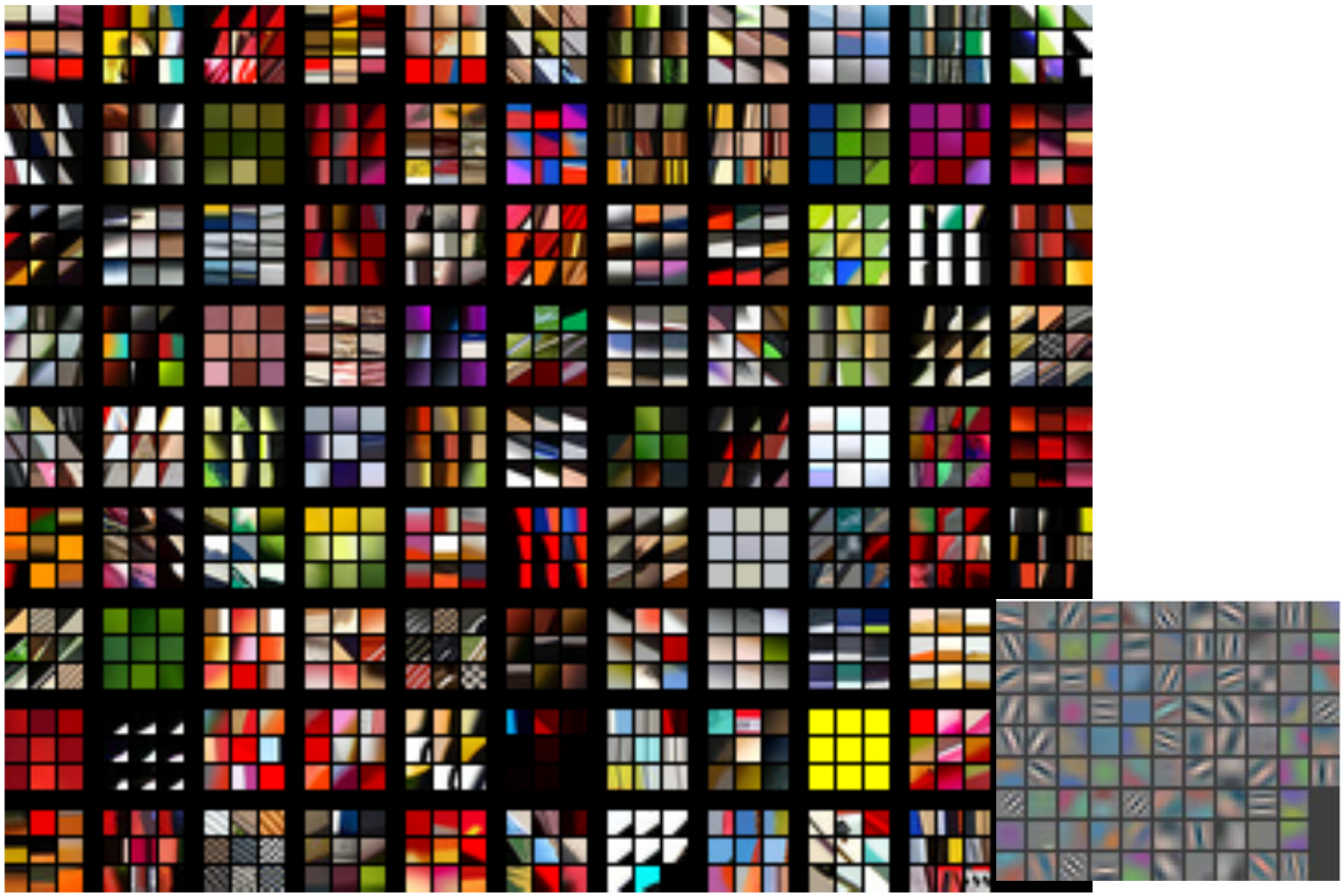
A. Krizhevsky, I. Sutskever, and G. Hinton, [ImageNet Classification with Deep Convolutional Neural Networks](), NIPS 2012

Layer 1 Filters



M. Zeiler and R. Fergus, [Visualizing and Understanding Convolutional Networks](),
arXiv preprint, 2013

Layer 1: Top-9 Patches



Layer 2: Top-9 Patches

- Patches from validation images that give maximal activation of a given feature map

Layer 3: Top-9 Patches


Layer 4: Top-9 Patches

Layer 5: Top-9 Patches

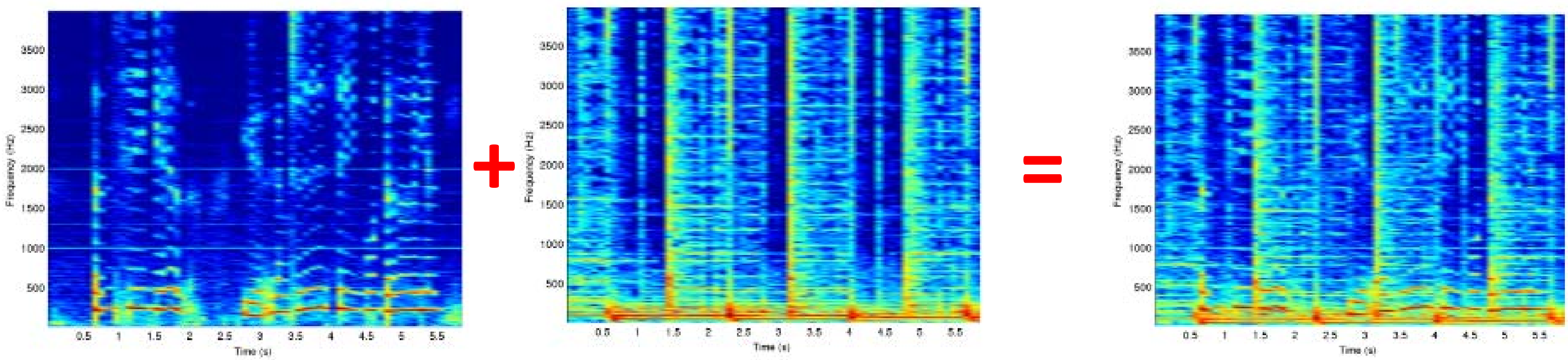
Deep Learning

- Differentiable Perceptron/One-Layer Neural Net
- Two-Layer Neural Net
- Convolutional Neural Net
- Singing-Voice Separation Using Deep Recurrent Network
- Semantic Image Inpainting with Deep Generative Models

Singing-Voice Separation from Monaural Recordings Using Deep Recurrent Neural Networks

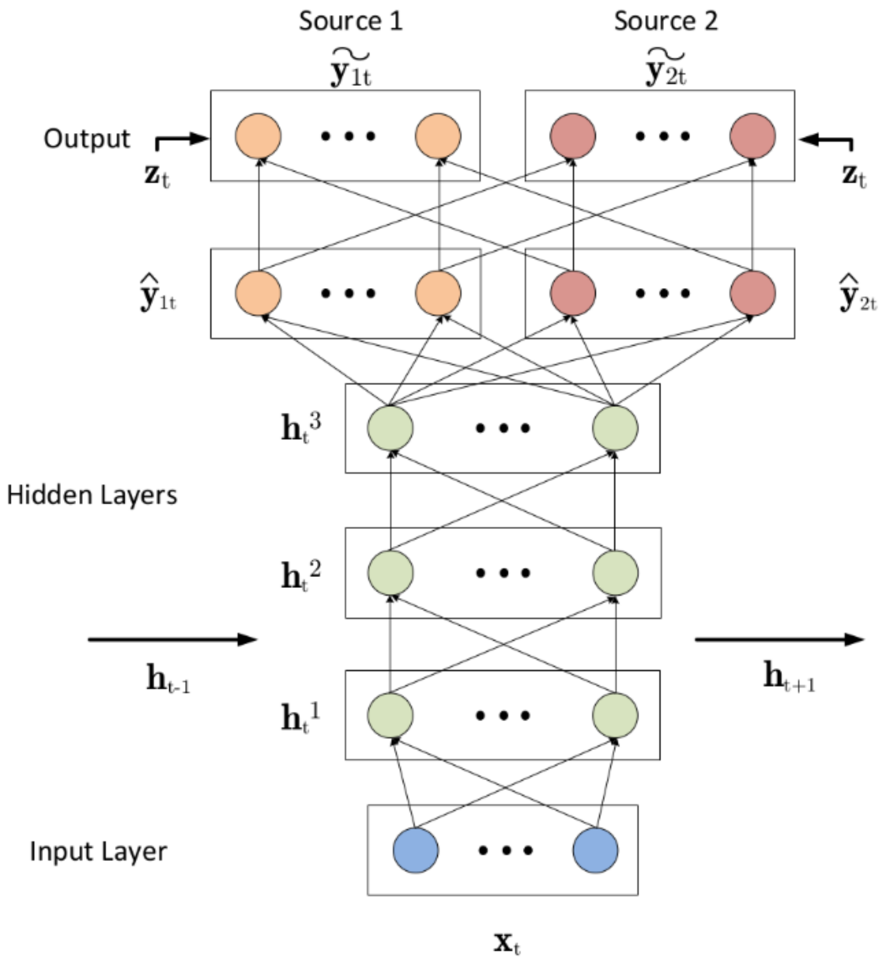
Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson and Paris Smaragdis, ISMIR 2014

The problem:



Singing-Voice Separation

The solution is to train this:



To minimize this:

$$J_{MSE} = ||\hat{\mathbf{y}}_{\mathbf{1}_t} - \mathbf{y}_{\mathbf{1}_t}||_2^2 + ||\hat{\mathbf{y}}_{\mathbf{2}_t} - \mathbf{y}_{\mathbf{2}_t}||_2^2$$

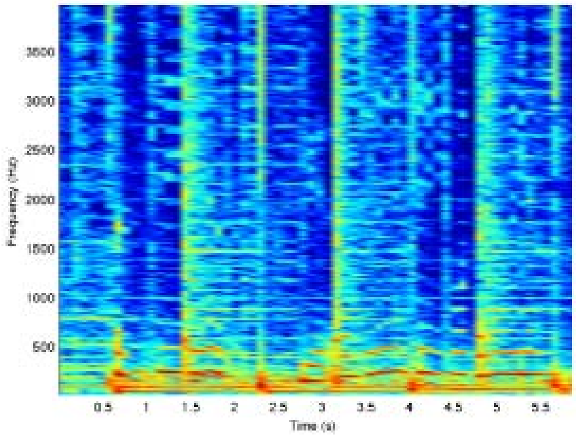
Using these specialized output nodes:

$$\tilde{\mathbf{y}}_{\mathbf{1}_t} = \frac{|\hat{\mathbf{y}}_{\mathbf{1}_t}|}{|\hat{\mathbf{y}}_{\mathbf{1}_t}| + |\hat{\mathbf{y}}_{\mathbf{2}_t}|} \odot \mathbf{z}_t$$

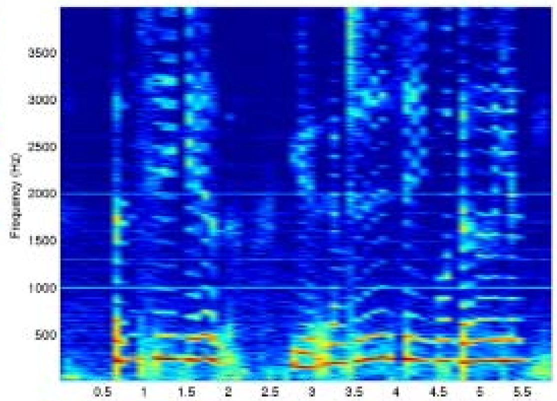
$$\tilde{\mathbf{y}}_{\mathbf{2}_t} = \frac{|\hat{\mathbf{y}}_{\mathbf{2}_t}|}{|\hat{\mathbf{y}}_{\mathbf{1}_t}| + |\hat{\mathbf{y}}_{\mathbf{2}_t}|} \odot \mathbf{z}_t,$$

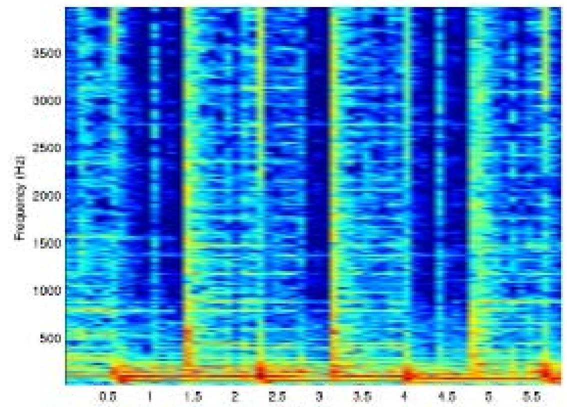
Singing-Voice Separation Results Example

- Input:

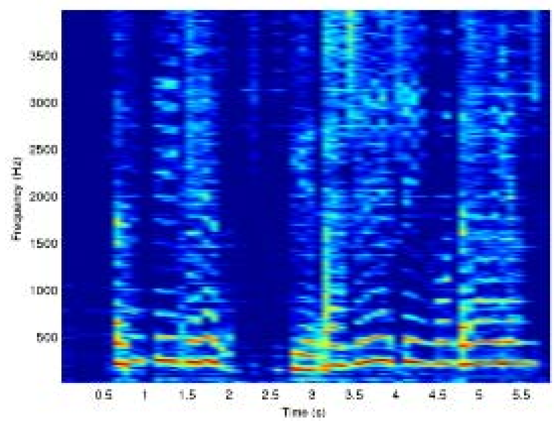


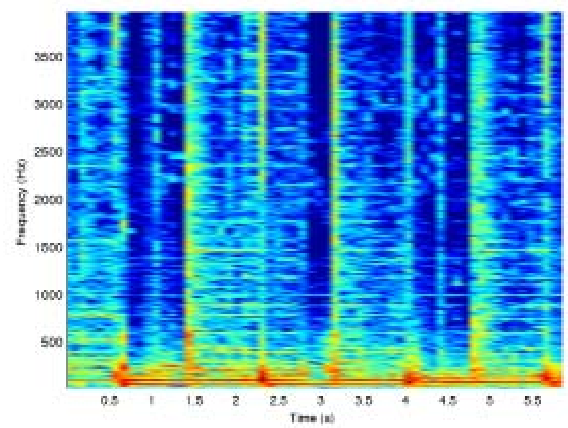
- Goal:





- Actual Network Outputs:





Deep Learning

- Differentiable Perceptron/One-Layer Neural Net
- Two-Layer Neural Net
- Convolutional Neural Net
- Singing-Voice Separation Using Deep Recurrent Network
- Semantic Image Inpainting with Deep Generative Models

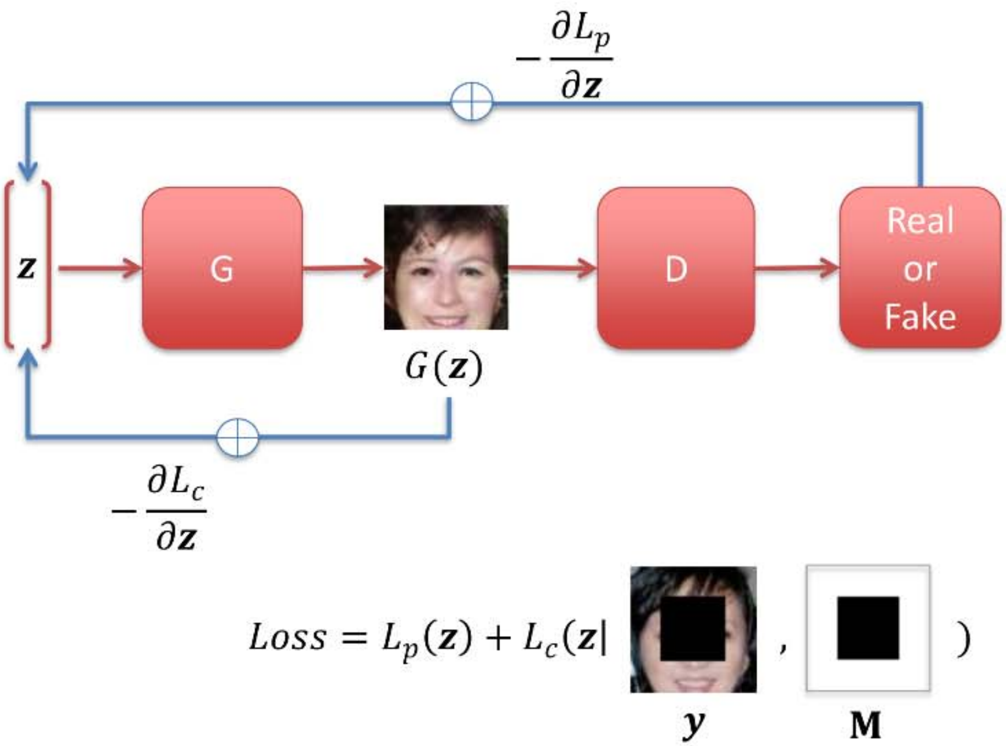
Semantic Image Inpainting with Deep Generative Models

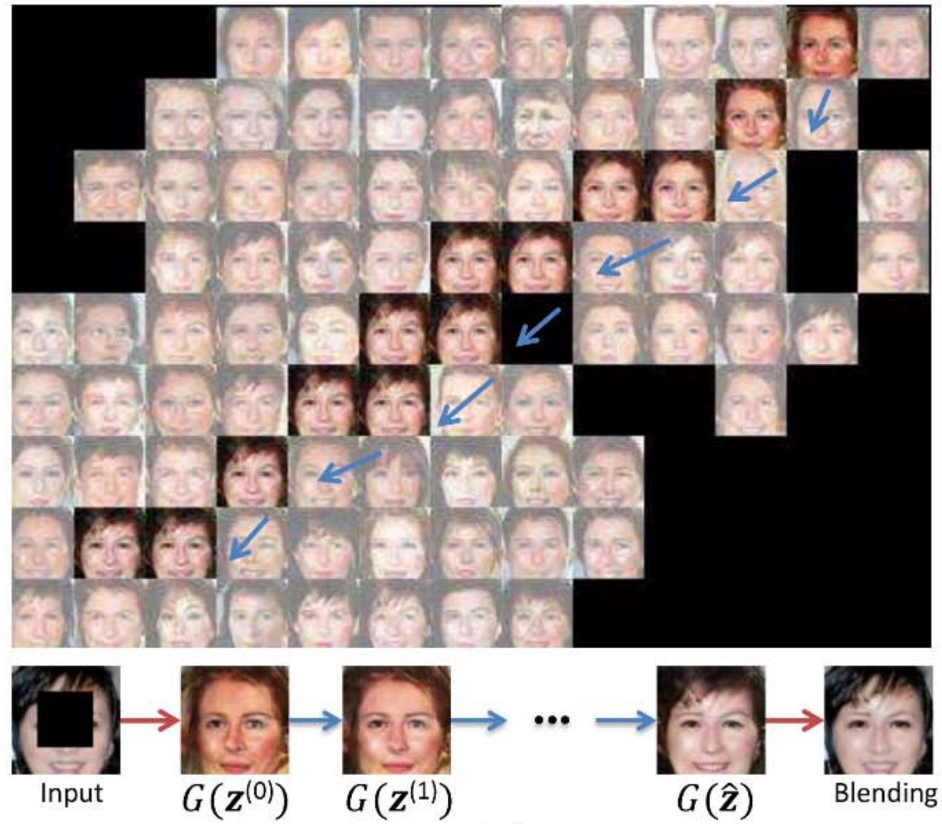
Raymond Yeh, Chen Chen, Teck Yian Lim, Alexander G. Schwing, Mark Hasegawa-Johnson and Minh Do

The problem:

Semantic Image Inpainting

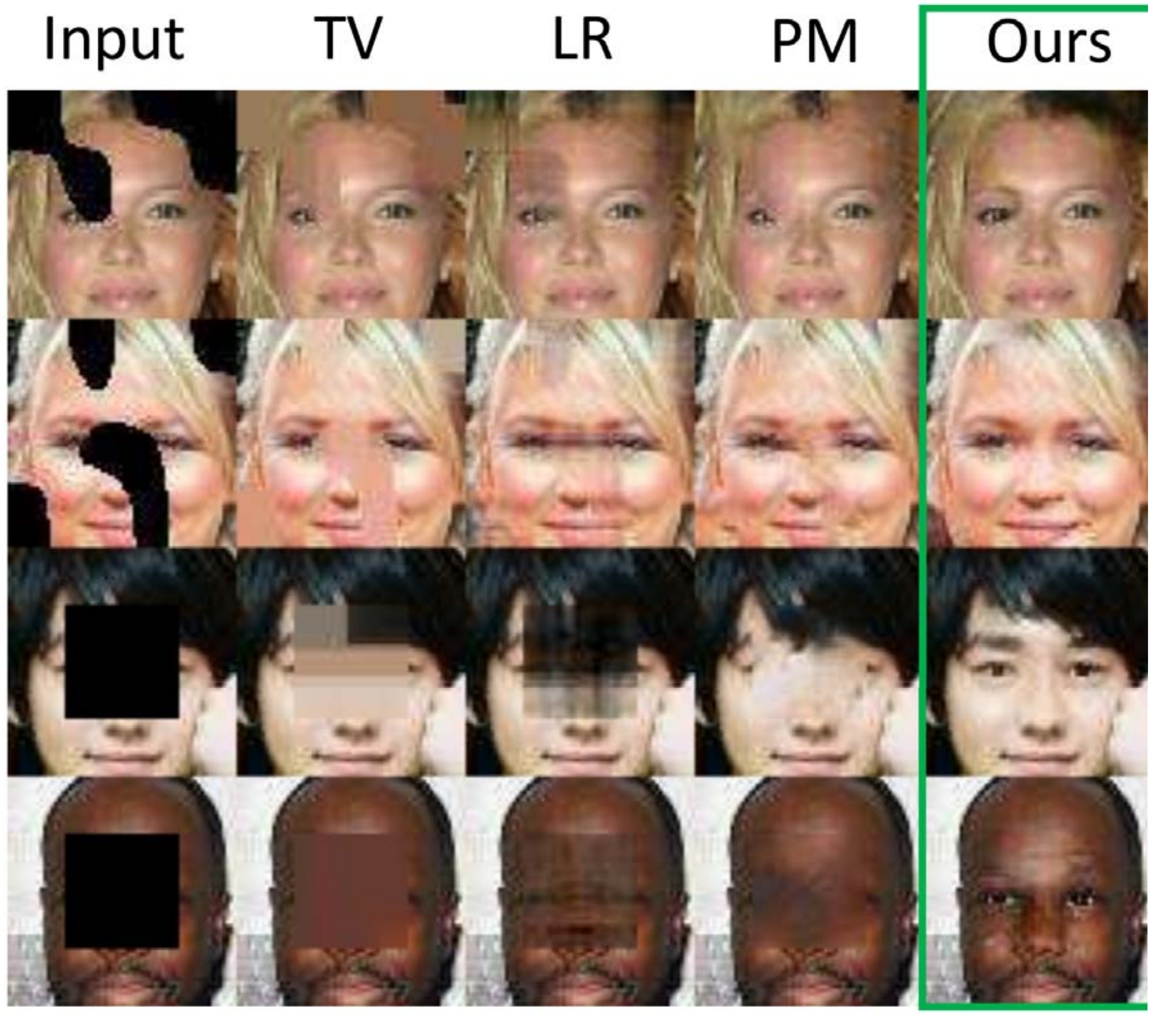
The solution:





Semantic Image Inpainting

The results:



Summary: You now know…

- What does a deep neural net compute?

$$z_{li} = \tanh\left(\sum_{k=0}^{q} v_{lk} \tanh\left(\sum_{j=0}^{p} u_{kj} x_{ji}\right)\right)$$

- How is it trained?

$$\frac{\partial E}{\partial u_{kj}} = \sum_{i=1}^{n} (\tanh(\blacksquare) - z_i) \left(\frac{\partial \tanh(\blacksquare)}{\delta \blacksquare}\right) \left(\sum_{k=0}^{q} v_{lk} \frac{\partial \tanh(\cdot)}{\delta \cdot}\right) x_{ji}$$

- How can it be used? Examples: image classification, singing voice separation, semantic image inpainting.