Deep learning

Johan Suykens

KU Leuven, ESAT-STADIUS
Kasteelpark Arenberg 10
B-3001 Leuven (Heverlee), Belgium
Email: johan.suykens@esat.kuleuven.be
http://www.esat.kuleuven.be/stadius

Lecture 10

Overview

- Motivations
- Autoencoders and sparsity
- Feature selection
- Pre-training and fine-tuning
- Stacked autoencoders
- Greedy layer-wise training
- Fully versus locally connected networks
- Convolution and pooling

Introduction

Lecture material mainly based on:

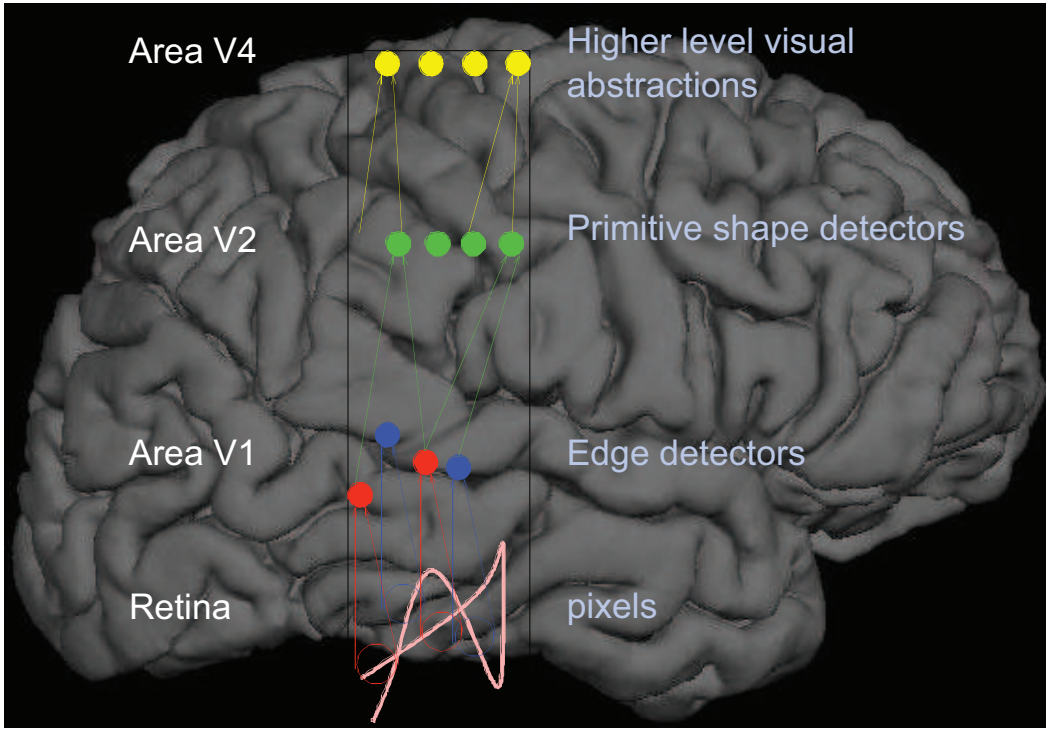
UFLDL Tutorial (Ng et al.)
http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial

Learning deep architectures for AI
Y Bengio, Foundations and trends in Machine Learning, 2009

Bengio & LeCun, Tutorial at ICML2009

Motivations deep learning [Bengio & LeCun, ICML2009]

Deep Architecture in the Brain

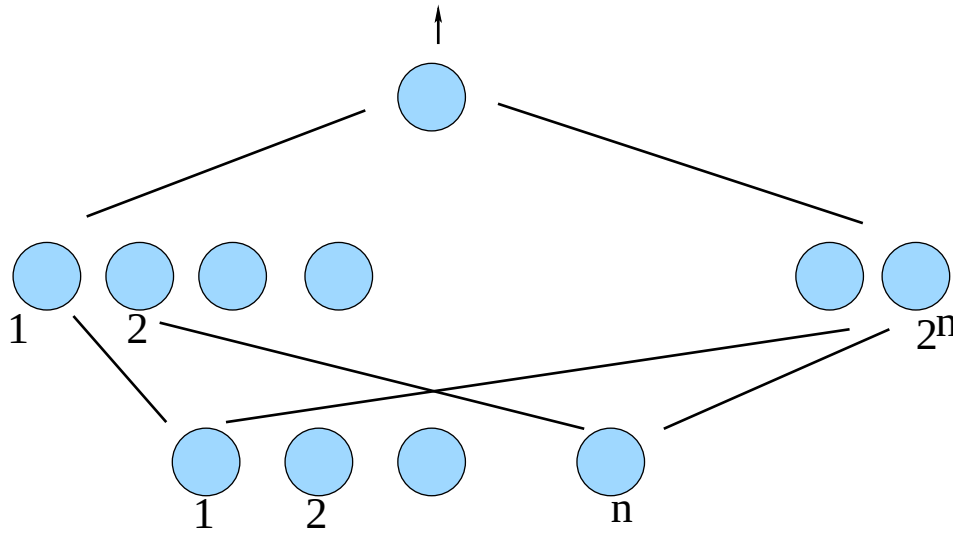


Motivations for deep learning

Neural networks with one hidden layer are universal approximators, but ...

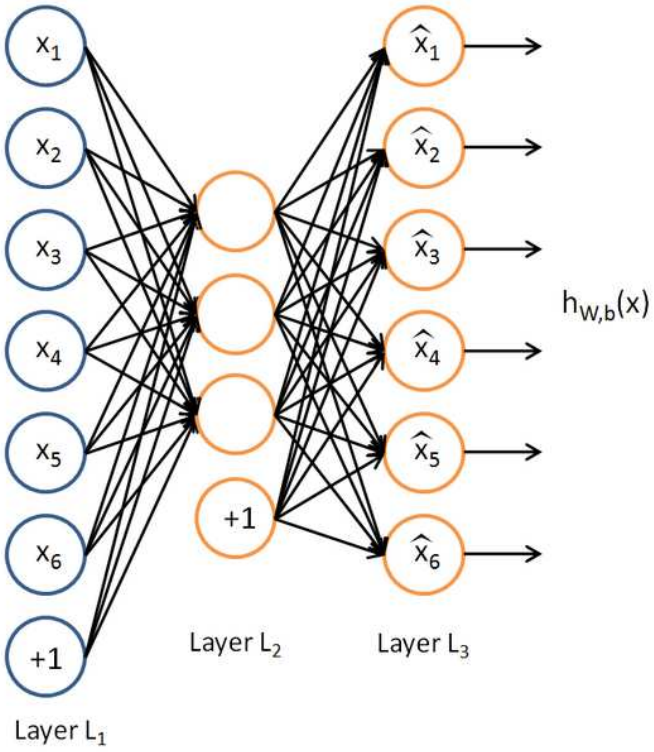
Theorem [Hastad et al 1986, 1991, Bengio et al 2007]
Functions representable compactly with k layers may require **exponential size** with $k-1$ layers.

Example: One hidden layer and n inputs may require 2^n hidden units



Autoencoders and sparsity (1)

- Unlabeled training examples $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots\}$, where $x^{(i)} \in \mathbb{R}^n$
- Autoencoder: unsupervised learning algorithm applying backpropagation, setting target values equal to inputs: $y^{(i)} = x^{(i)}$.



[UFLDL Tutorial]

Autoencoders and sparsity (2)

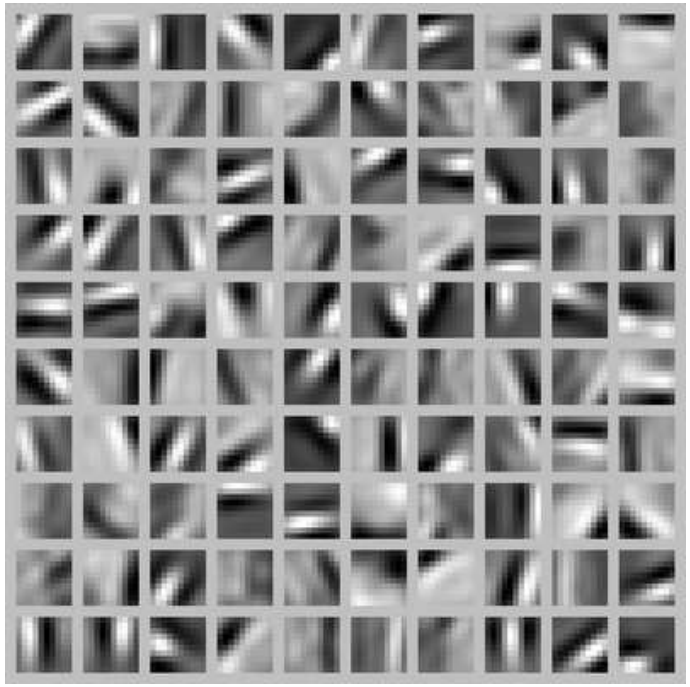
- This looks like learning the identity map (which seems trivial at first sight), but putting additional constraints can reveal interesting structure about the data. This is achieved by limiting the number of hidden units (i.e. by imposing a sparsity constraint).

- One wants to minimize:

$$\|\text{input} - \text{decoder}(\text{code})\|^2 + \text{sparsity}(\text{code})$$

- **Example:** inputs x are the pixel intensity values from a 10×10 image (100 pixels) with e.g. 50 hidden units and $y \in \mathbb{R}^{100}$. The network is forced then to learn a **compressed representation** of the input (similar to a PCA reconstruction problem).

Visualizing a trained autoencoder



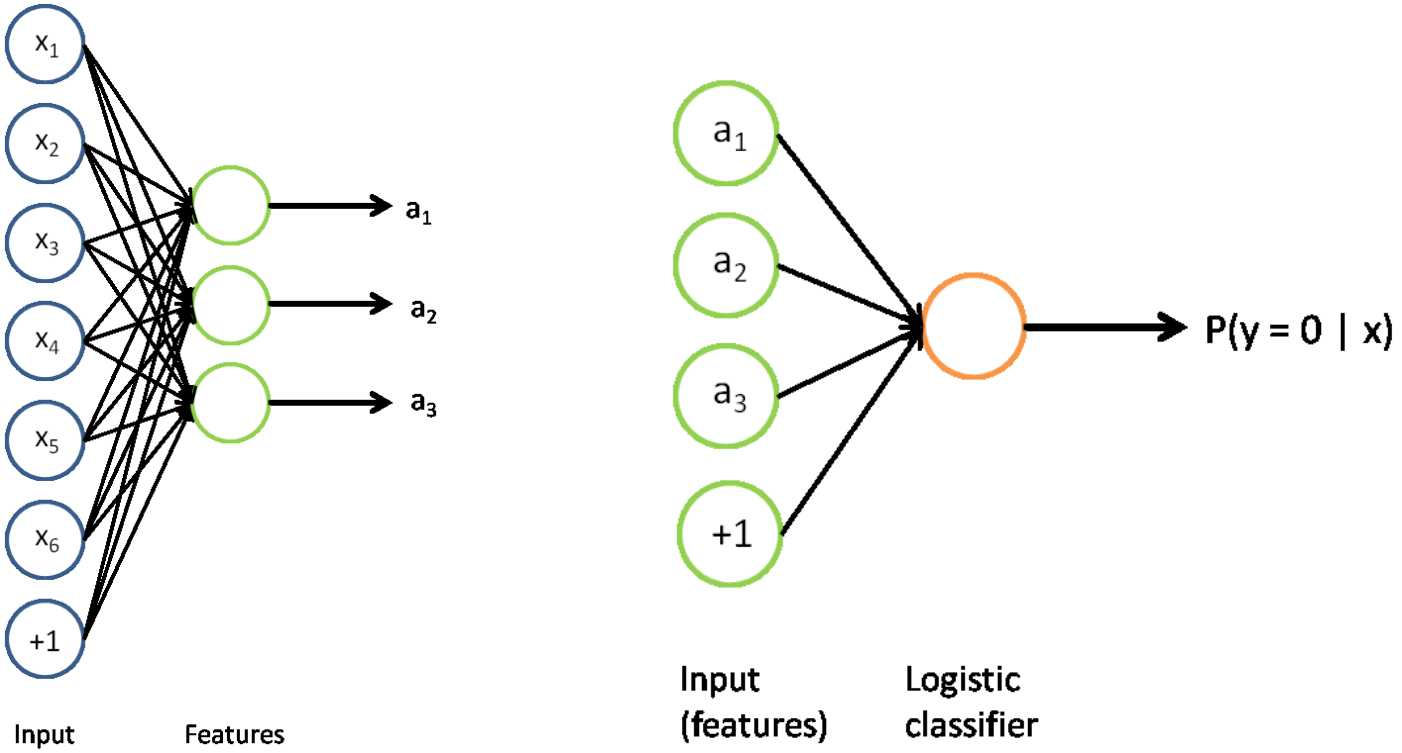
[UFLDL Tutorial]

- Each square in the figure shows the (norm bounded) input image x that maximally activates one of 100 hidden units.
- Learned features were obtained by training on whitened natural images.
- Different hidden units have learned to detect edges at different positions and orientations in the image.

Using selected features within a classifier (1)

Step 1: pre-training phase:

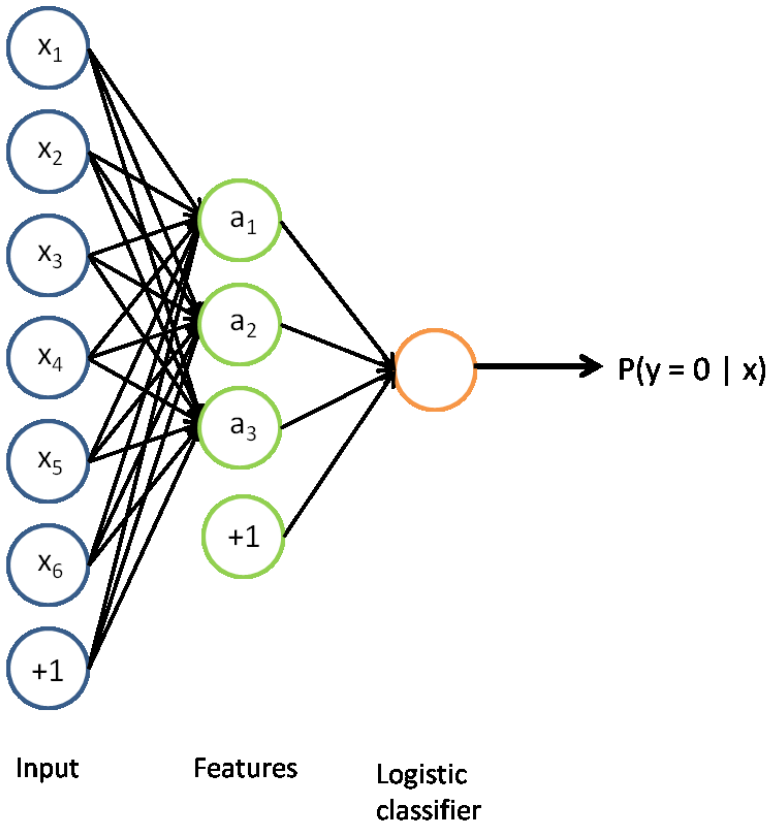
- Train a sparse autoencoder on the unlabeled data.
- Given a new example x, used the hidden layer to extract features a.
- Use the extracted features a as new inputs to train a classifier.



[UFLDL Tutorial]

Using selected features within a classifier (2)

Step 2: fine-tuning phase:
Consider the overall classifier and fine-tune by further supervised training.



[UFLDL Tutorial]

Deep networks versus shallow networks

- **Shallow network:** network conisting of an input, hidden and output layer, where the features are computed using only one layer.

- **Deep network:** multiple hidden layers to determine more complex features of the input.

- Important to use a **nonlinear activation function** (multiple layers of linear functions would compute only a linear function of the input)

Deep networks: advantages

- Compactly represent a significantly larger set of functions
- *There are functions which a k-layer network can represent compactly (with a number of hidden units that is* **polynomial** *in the number of inputs), that a (k-1)-layer network cannot represent, unless it has an exponentially large number of hidden units.*
-
-

Deep networks: advantages

- Compactly represent a significantly larger set of functions

- *There are functions which a k-layer network can represent compactly (with a number of hidden units that is* **polynomial** *in the number of inputs), that a (k-1)-layer network cannot represent, unless it has an exponentially large number of hidden units.*

- One can learn **part-whole decompositions** (feature hierarchies): e.g.
  layer 1: learn to group pixels in an image to detect edges
  layer 2: group together edges to detect longer contours
  (detect simple parts of objects)
  layer 3,4,...: group together contours or detect complex features

- **Cortical** computations in the brain also have multiple layers of processing:
  cortical area V1
  cortical area V2

Deep networks: difficulty of training

- Supervised learning using labeled data by applying gradient descent (e.g. backpropagation) on deep networks usually does not work well. Too **many bad local minima** occur.

- Gradients becoming very small in deep networks for randomly initialized weights. When using backpropagation to compute the derivatives, the gradients that are backpropagated rapidly decrease in magnitude as the depth of the network increases (called **diffusion of gradient problem**).

- **Greedy layer-wise training works better**.

Stacked autoencoders

- Autoencoders can be stacked in a greedy layerwise fashion for pre-training the weights of a deep network.

- A stacked autoencoder is a neural network consisting of multiple layers of sparse autoencoders in which the outputs of each layer are wired to the inputs of the successive layer.

- n layers where $W^{(k,1)}, W^{(k,2)}, b^{(k,1)}, b^{(k,2)}$ denote the parameters $W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}$ for the k-th autoencoder:

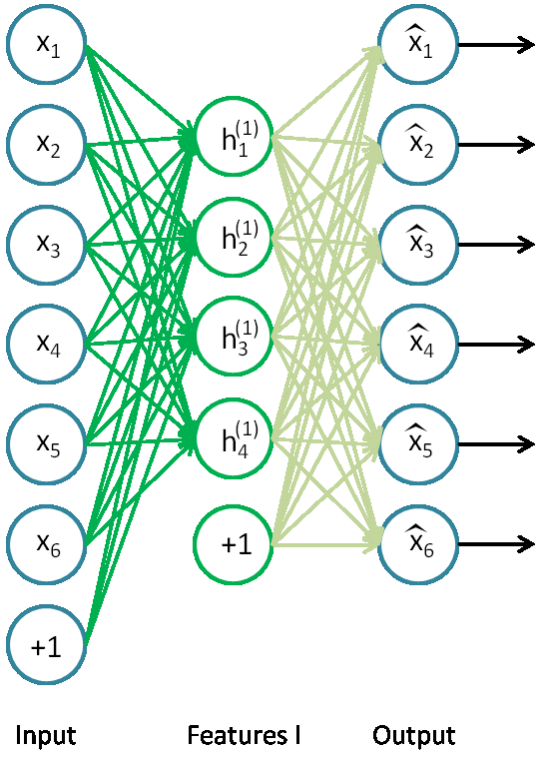
$$\begin{aligned} a^{(l)} &= f(z^{(l)}) \\ z^{(l+1)} &= W^{(l,1)} a^{(l)} + b^{(l,1)} \\ &\dots \\ a^{(n+l)} &= f(z^{(n+l)}) \\ z^{(n+l+1)} &= W^{(n+l,2)} a^{(n+l)} + b^{(n+l,2)} \end{aligned}$$

  Activation $a^{(n)}$ of the deepest layer gives an input representation in terms of higher-order features, and can be used for pre-training a classifier.

Stacked autoencoders: greedy layer-wise training (1)

Example: train a stacked autoencoder with 2 hidden layers for classification

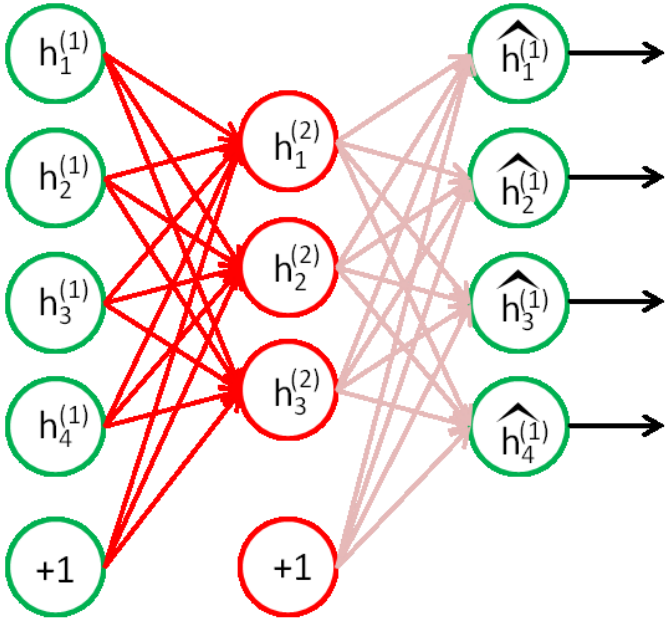
Step 1: train a sparse autoencoder on the inputs $x^{(i)}$ to learn primary features $h^{(1)(i)}$ on the input.



[UFLDL Tutorial]

Stacked autoencoders: greedy layer-wise training (2)

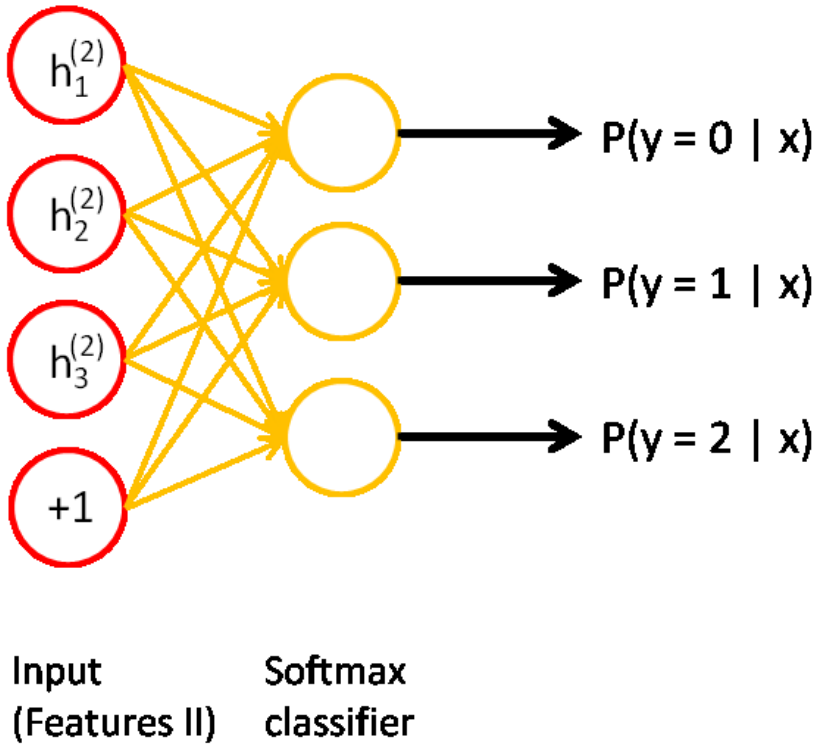
Step 2: use the primary features as the input to a next sparse autoencoder to learn secondary features $h^{(2)(i)}$ on these primary features.



[UFLDL Tutorial]

Stacked autoencoders: greedy layer-wise training (3)

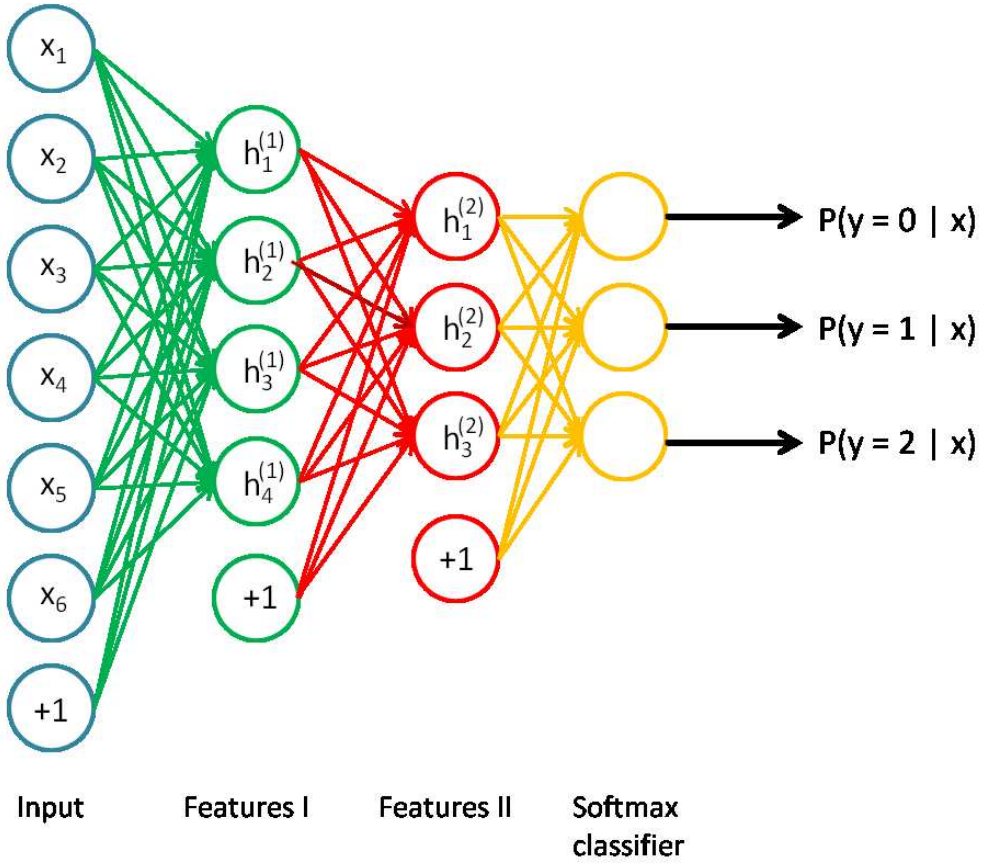
Step 3: take the secondary features as input to a classifier



[UFLDL Tutorial]

Stacked autoencoders: greedy layer-wise training (4)

Step 4: combine all layers together to form a stacked autoencoder with 2 hidden layers and a classifier layer, and apply fine-tuning



[UFLDL Tutorial]

Limitations of fully connected networks

- 28×28 images (in the MNIST dataset):
  Computationally feasible to learn features on the entire image.
  Use of fully connected network is possible.
-
-

Limitations of fully connected networks

- 28×28 images (in the MNIST dataset):
  Computationally feasible to learn features on the entire image.
  Use of fully connected network is possible.

- 96×96 images: about 10^4 inputs.
  Assuming learning 100 features, then about 10^6 weights to learn!
  Therefore there is a need for **locally connected networks**.

- **Early visual system in the brain**:
  neurons in the visual cortex have localized receptive fields (they respond only to stimuli in a certain location).

Feature extraction using convolution and pooling (1)

- Natural images have the property of being **stationary** (i.e. statistics of one part of the image are the same as any other part).

  This suggests that the features that we learn at one part of the image can also be applied to other parts of the image, and we can use the same features at all locations.

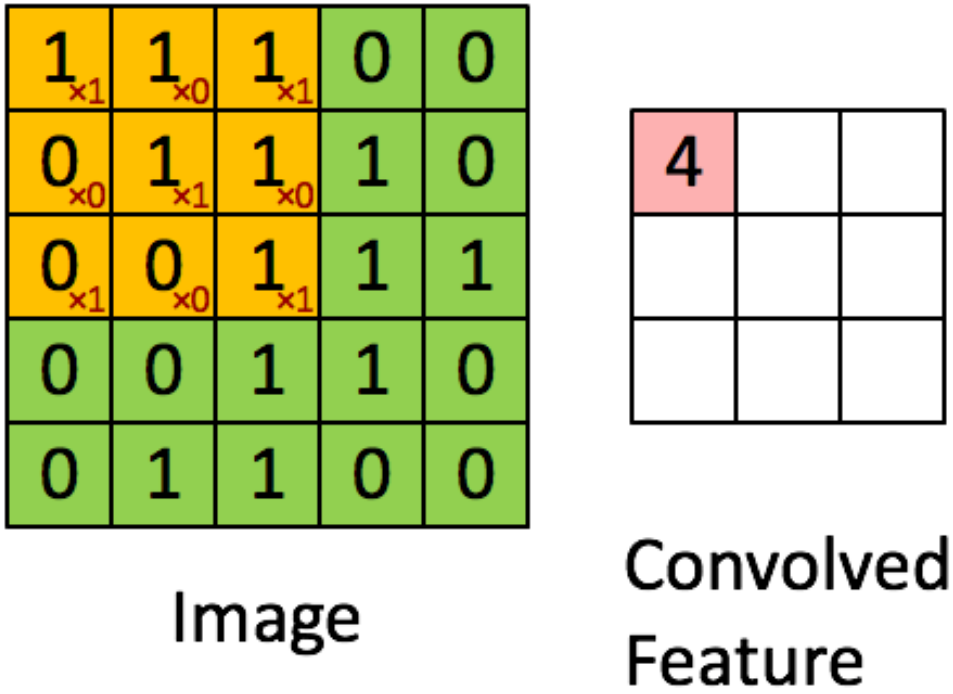
- **Example**: having learned features over small 8×8 patches sampled randomly from the larger 96×96 image, apply then the learned 8×8 feature detector anywhere in the image.

- Take the learned 8×8 features and **convolve** them with the larger image.

- **Pooled** convolved features (by max pooling or mean pooling operation over a region of the image) to limit the number of features.

Feature extraction using convolution and pooling (2)

- $r \times c$ image x_{large}
- Train sparse autoencoder on small $a \times b$ patches x_{small}, learning k features
- $k \times (r - a + 1) \times (c - b + 1)$ **convolved features**
- **Example:** image with $r = c = 5$, patch with $a = b = 3$



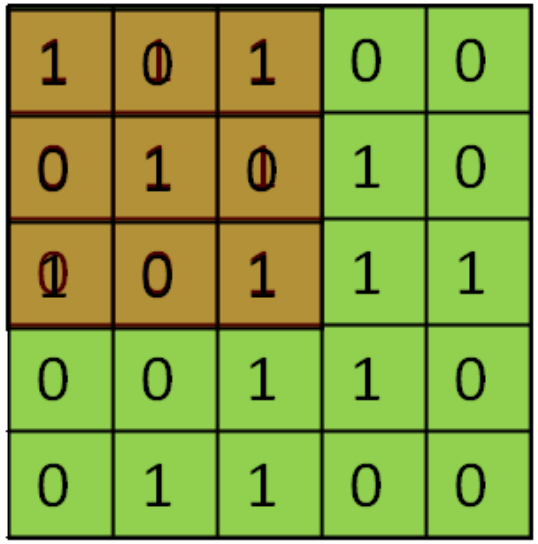
[UFLDL Tutorial]

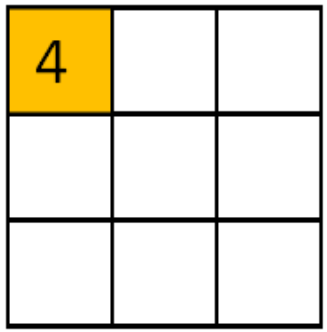
Feature extraction using convolution and pooling (3)

- **Example:**
  96×96 image
  Learned 400 features over 8×8 inputs
  Each convolution results in output size $(96 - 8 + 1).(96 - 8 + 1) = 7921$
  Features per example: 400 . 7921 = 3,168,400

- Apply **pooling** to limit the number of features:
  Compute the mean (or max) value of a particular feature over a region of the image. Such summary statistics are lower dimensional.

Convolution example



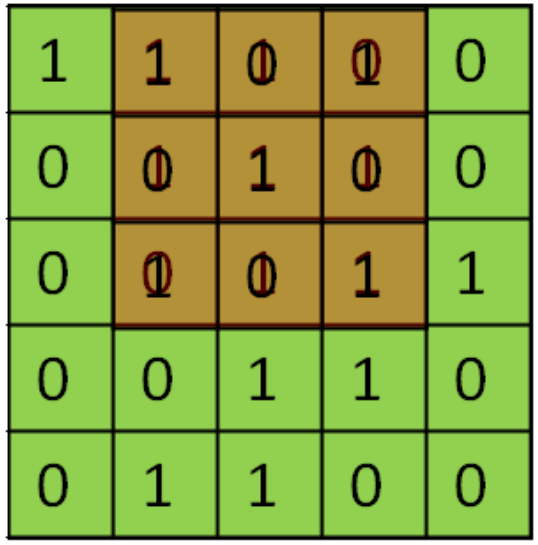


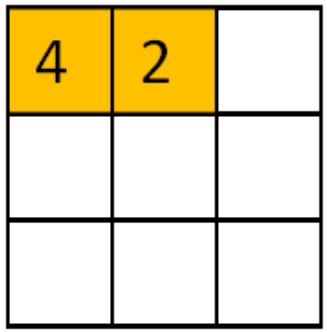
Image

Convolved feature

[UFLDL Tutorial]

Convolution example



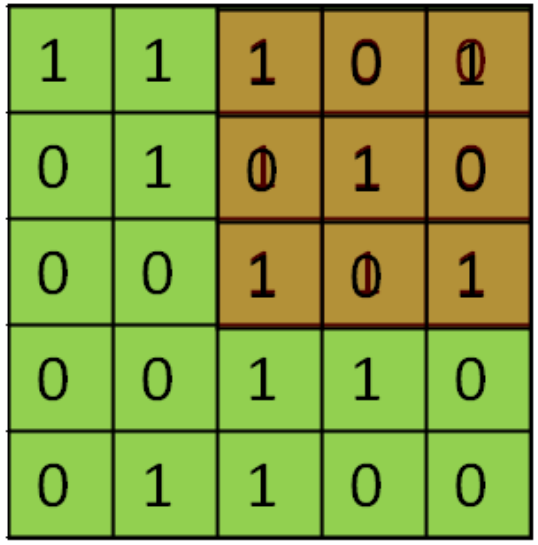


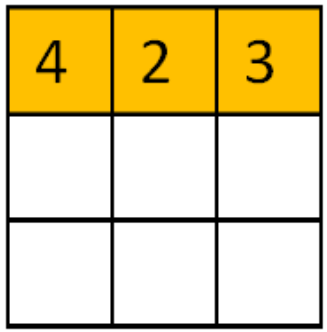
Image

Convolved feature

[UFLDL Tutorial]

Convolution example



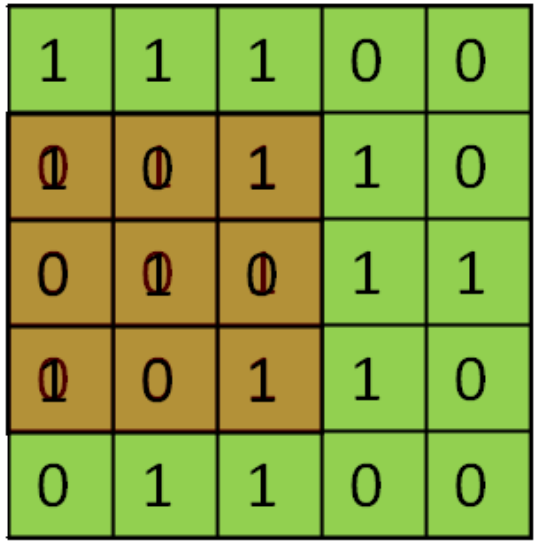


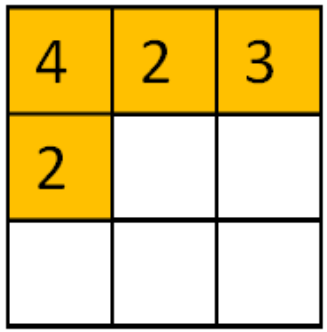
Image

Convolved feature

[UFLDL Tutorial]

Convolution example



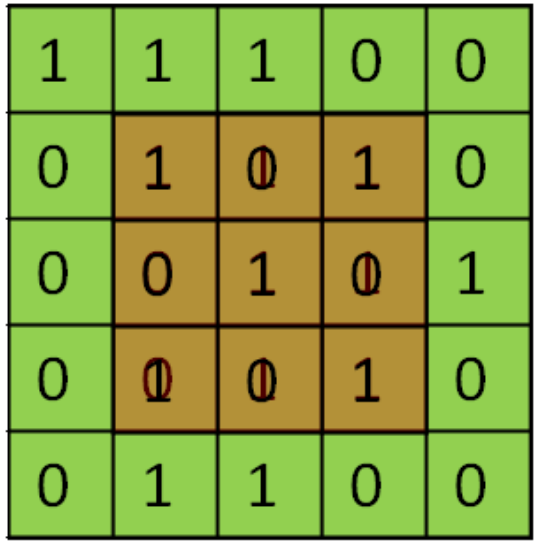


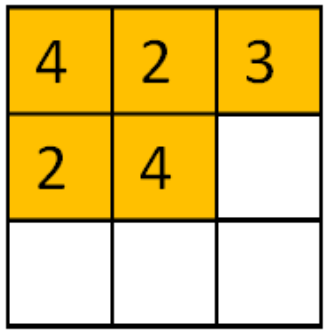
Image

Convolved feature

[UFLDL Tutorial]

Convolution example



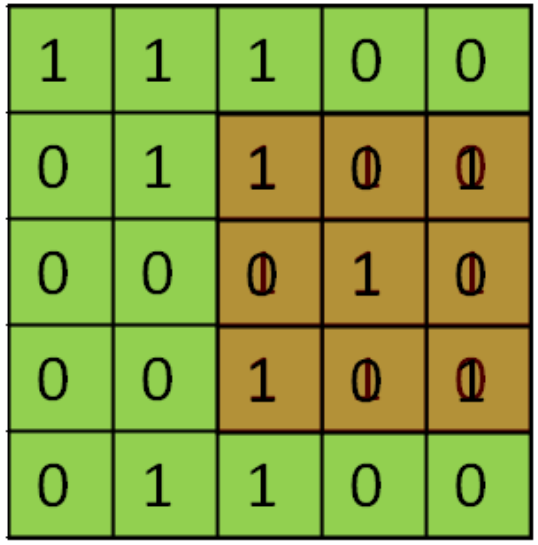


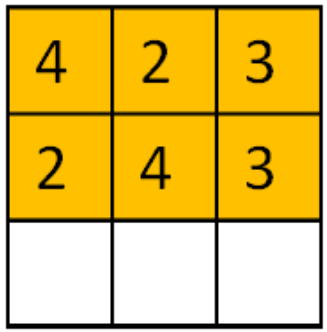
Image

Convolved feature

[UFLDL Tutorial]

Convolution example



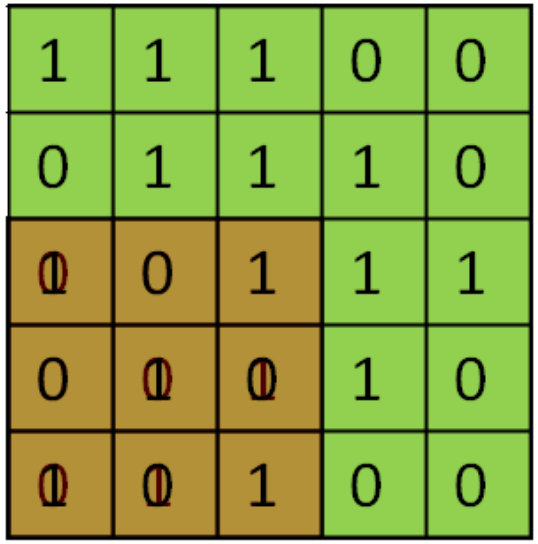


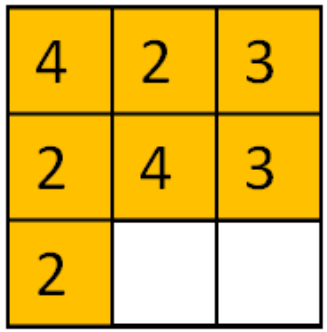
Image

Convolved feature

[UFLDL Tutorial]

Convolution example



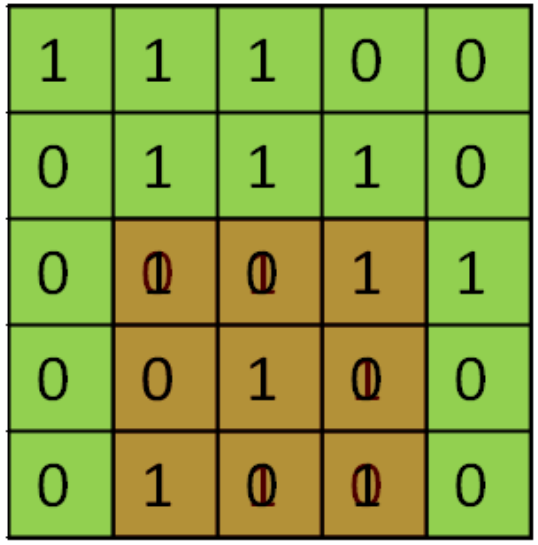


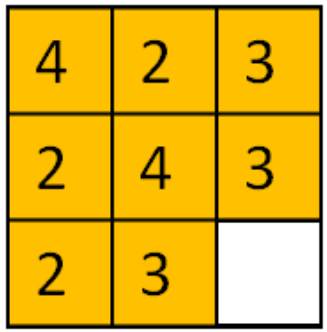
Image

Convolved feature

[UFLDL Tutorial]

Convolution example



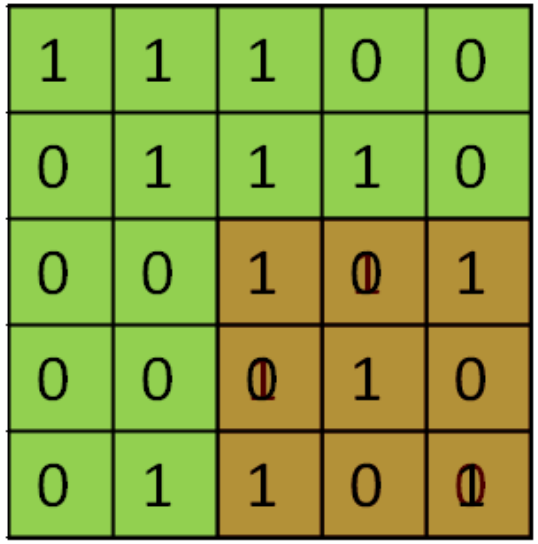


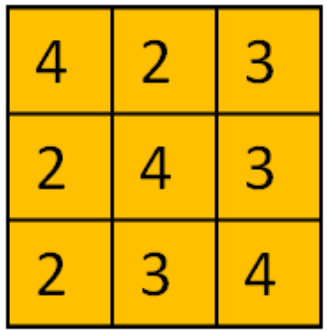
Image

Convolved feature

[UFLDL Tutorial]

Convolution example



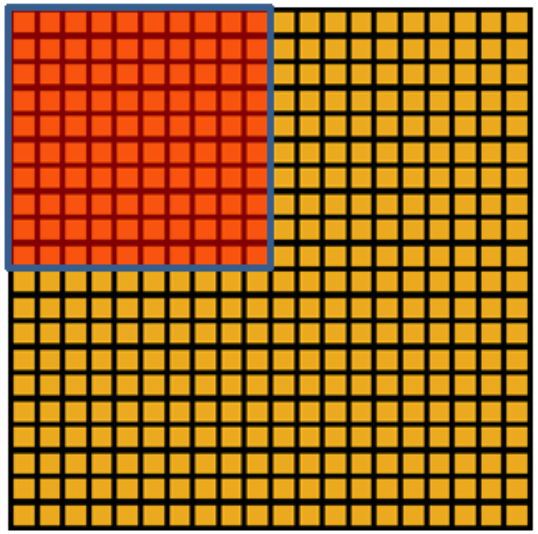


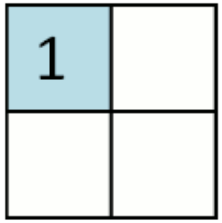
Image

Convolved feature

[UFLDL Tutorial]

Pooling example



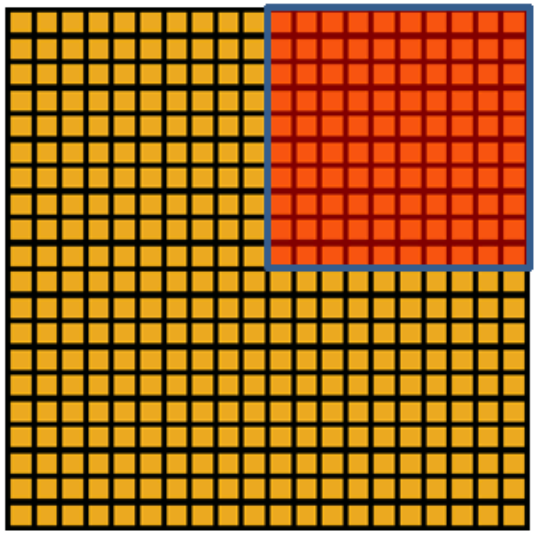


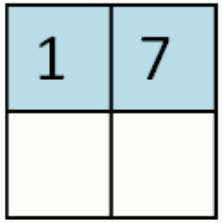
Convolved feature

Pooled feature

[UFLDL Tutorial]

Pooling example



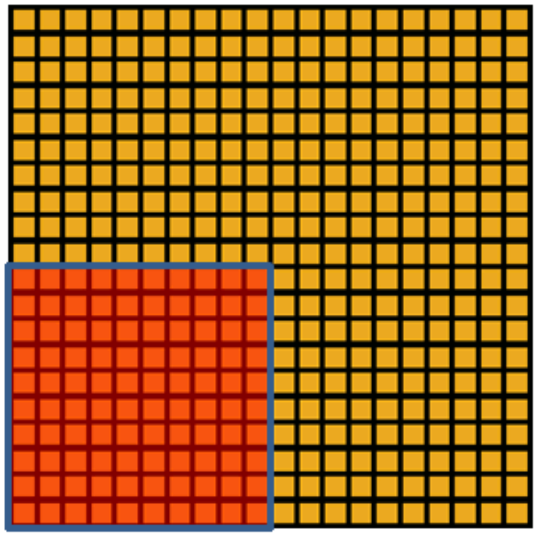


Convolved feature

Pooled feature

[UFLDL Tutorial]

Pooling example



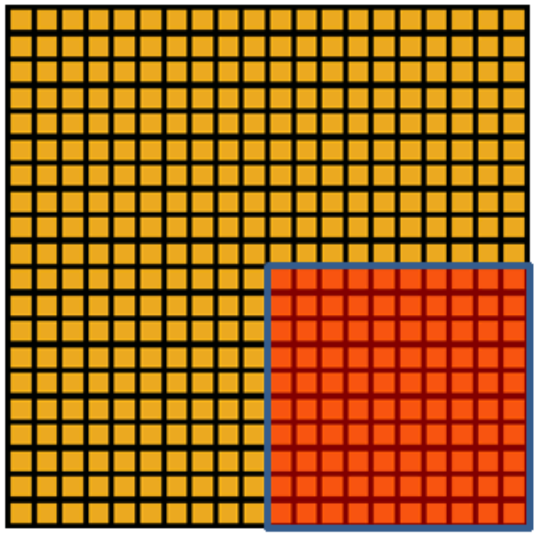
1	7
5	

Convolved feature

Pooled feature

[UFLDL Tutorial]

Pooling example



1	7
5	9

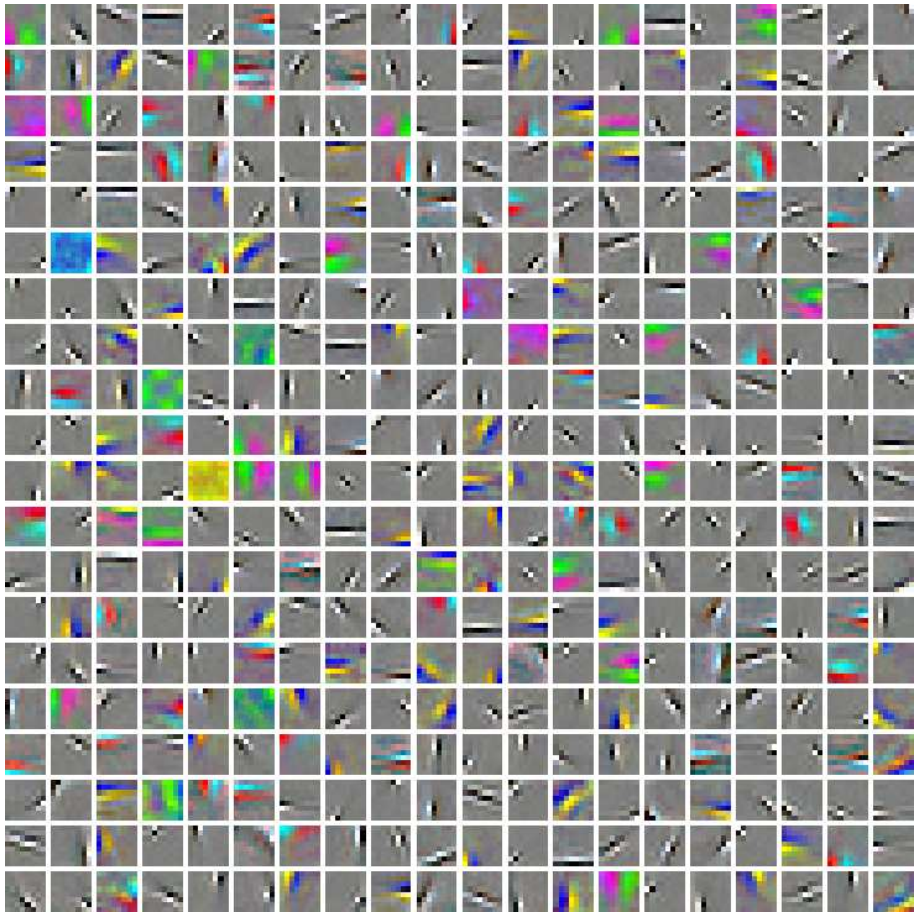
Convolved feature

Pooled feature

[UFLDL Tutorial]

Learned features: example

64×64 images from 4 classes (airplane, car, cat, dog)
Application of convolution and pooling



[UFLDL Tutorial]

youtube.com

Speed Sign Recognition by Convolutional Neural Networks

References

- UFLDL Tutorial (Ng et al.)
  http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial
- Y. Bengio, Learning deep architectures for AI, *Foundations and trends in Machine Learning*, 2(1): 1-127, 2009
- Bengio & LeCun, Tutorial at ICML2009
- G. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks. *Science*, 313(5786): 504-507, 2006.
- N. Jones, Computer science: The learning machines, *Nature*, news feature, 8 Jan 2014
- Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, A. Ng, Building high-level features using large scale unsupervised learning, arxiv.org/abs/1112.6209 (2011)
- A. Krizhevsky, I. Sutskever, G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, In *Advances in Neural Information Processing Systems* 25, available at http://go.nature.com/ibace6
- R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation http://arxiv.org/abs/1311.2524 (2013).
- S. Smale, L. Rosasco, J. Bouvrie, A. Caponnetto, T. Poggio, Mathematics of the Neural Response, *Foundations of Computational Mathematics*, 10(1):67-91, 2010.