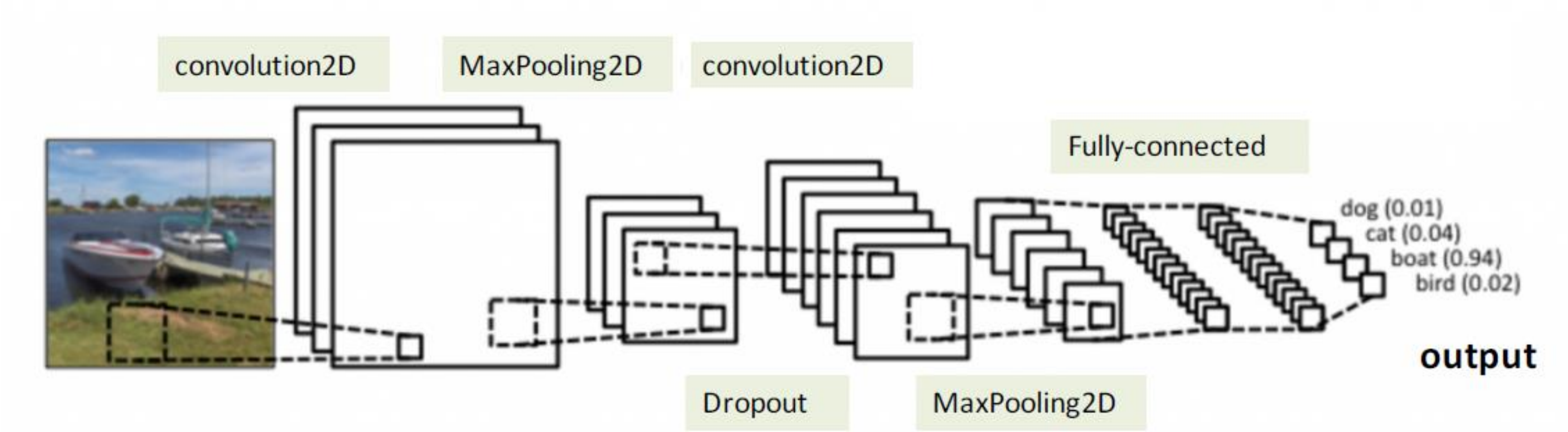
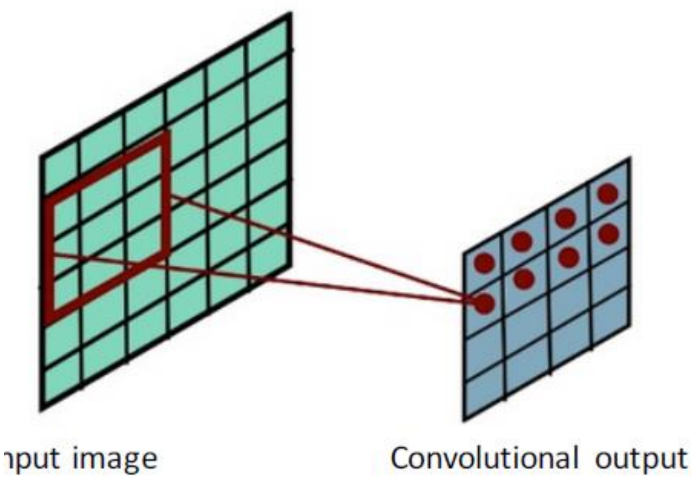
Outline For Today

- Introduction to Deep Learning
  - Learning Features for Predictive Modeling
- Deep Convolutional Networks
  - Very popular in Computer Vision
- Tips for Training Deep Networks
- Brief Overview of other Deep Networks

Deep CNN on CiFAR10

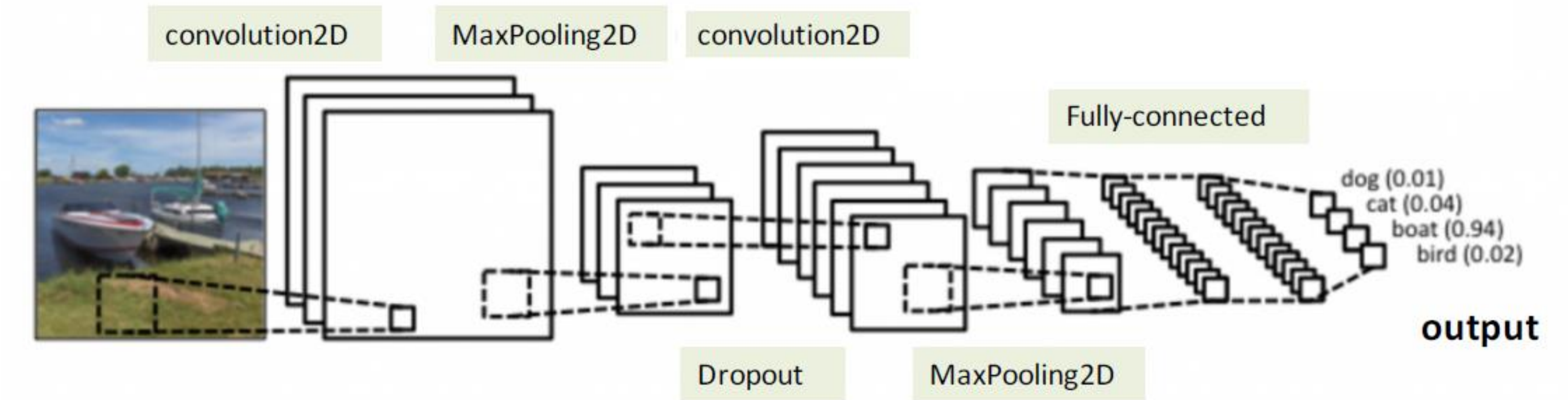


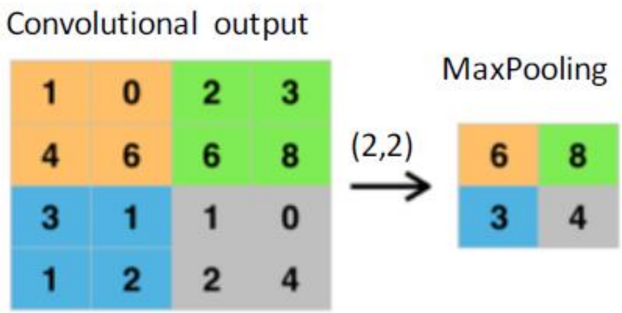


Convolutional Layer: filters work on every part of the image, therefore, they are searching for the same feature everywhere in the image.

http://www.slideshare.net/perone/deep-learning-convolutional-neural-networks

Deep CNN on CiFAR10

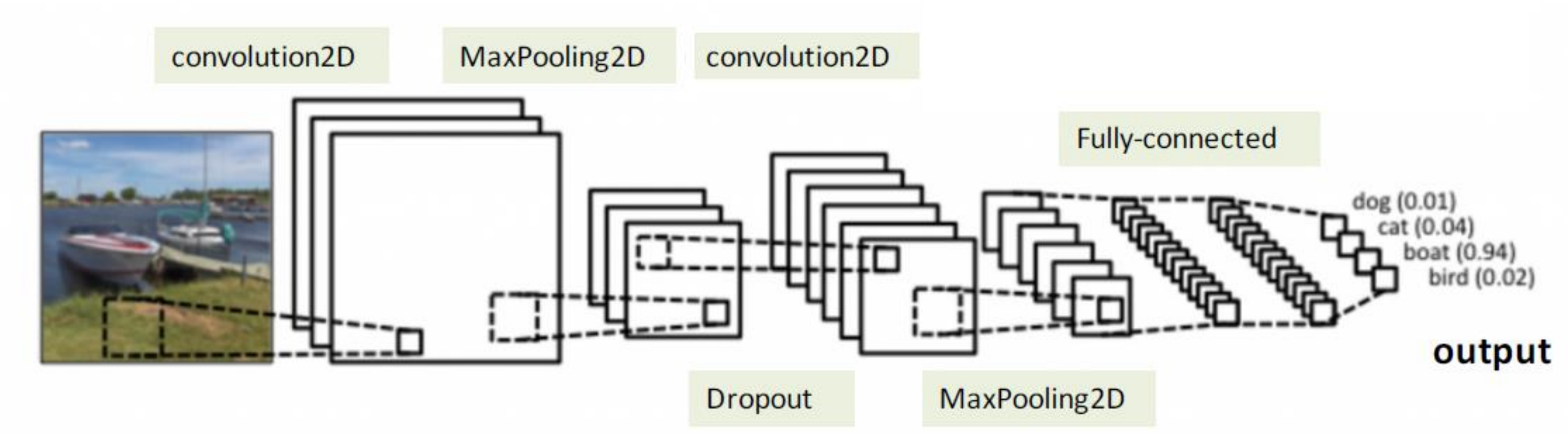


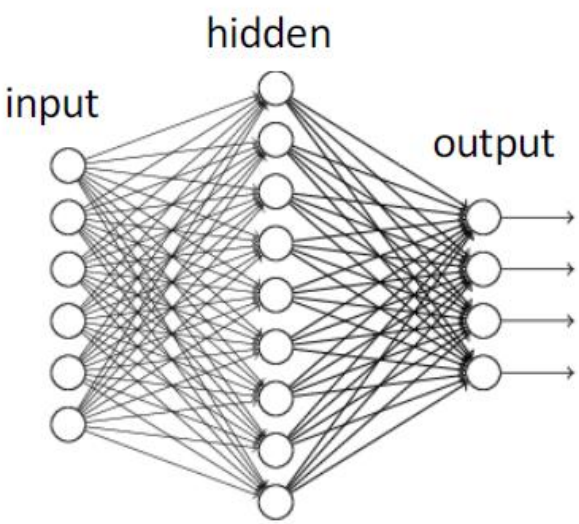


MaxPooling: usually present after the convolutional layer. It provides a down-sampling of the convolutional output

http://www.slideshare.net/perone/deep-learning-convolutional-neural-networks

Deep CNN on CiFAR10



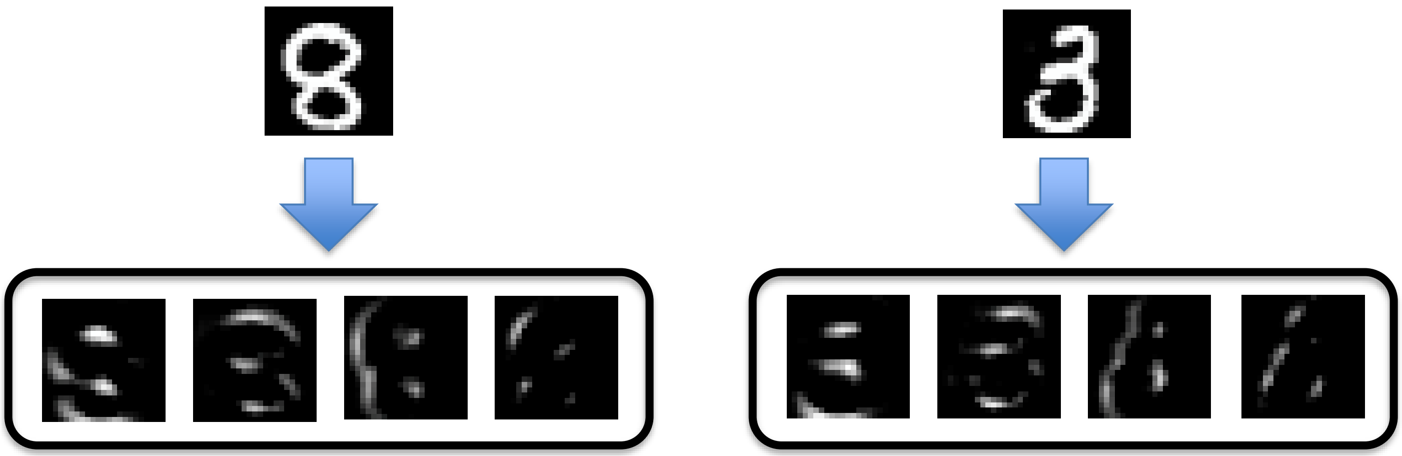


Fully-connected layer (dense): each node is fully connected to all input nodes, each node computes weighted sum of all input nodes. It has one-dimensional structure. It helps to classify input pattern with high-level features extracted by previous layers.

http://www.slideshare.net/datasciencekorea/5-20141107deeplearning

Convolutions

- Images typically have invariant patterns
  - E.g., directional gradients are translational invariant:



- Apply convolution to local sliding windows

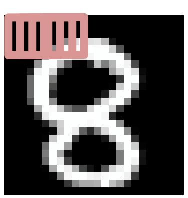
http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

Convolutional Filters

- Applies to an image patch x
  - Converts local window into single value
  - Slide across image

$$x \otimes W = \sum_{ij} W_{ij} x_{ij}$$

Local Image Patch

 $\otimes W =$ 

Left-to-Right
Edge Detector

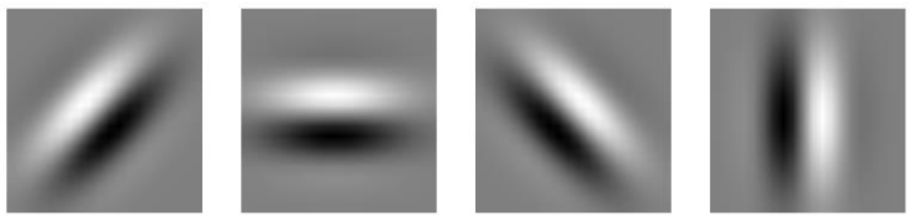
-1	0	+1
-1	0	+1
-1	0	+1

W

Gabor Filters

- Most common low-level convolutions for computer vision

Example W:



- Grey = 0
- Light = positive
- Dark = negative

-1	0	+1
-1	0	+1
-1	0	+1

W

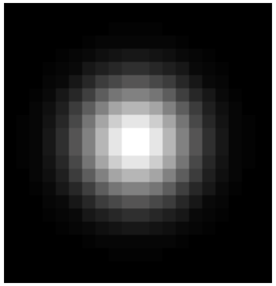
http://en.wikipedia.org/wiki/Gabor_filter

Gaussian Blur Filters

- Weights decay according to Gaussian Distribution
  - Variance term controls radius

Example W:

Apply per RGB Channel

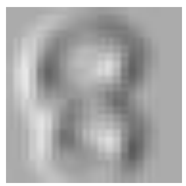
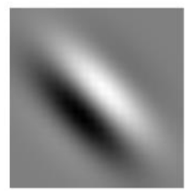


- Black = 0
- White = Positive

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

Deep Convolutional Networks

- Learn layers of convolutional filters W
  - Apply convolution to outputs of previous layer

E.g.:  Ä 

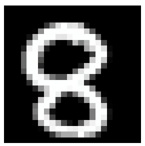
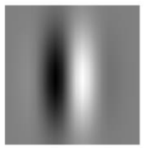
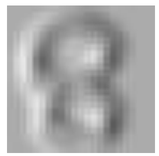
Stride in Convolution

- Adjacent Sliding Window Convolution
  - Yields output of same dimensions as input
- Good to compress into fewer pixels
  - Skip a few pixels for each convolution
- "Stride"
  - How far away next convolution is
  - No Down Sampling: Stride = 1
  - Down Sampling 2x: Stride = 2

Also Max Pooling

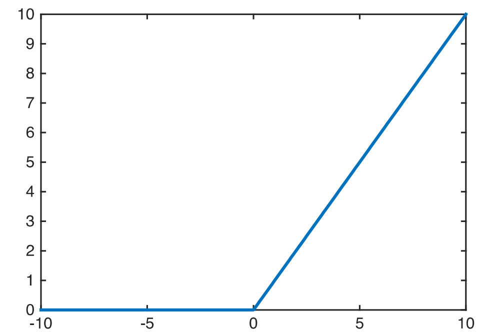
ReLU Activation Function

- Current Convolutional Layer consists of:

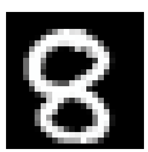
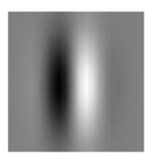
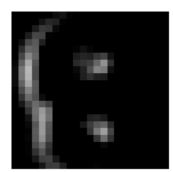
 Ä  = 

Convolution

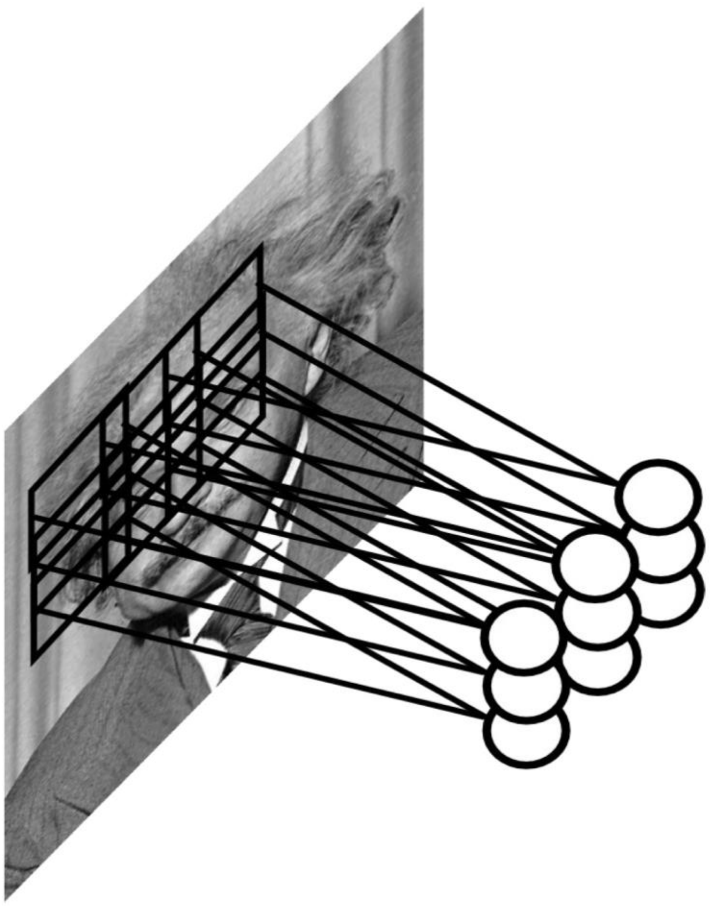
Rectilinear Transform
- Simplifies Backprop
- Chain rule super easy
- Also easier to train



- Main modeling concepts!
  – Combine them to create convolutional layer

max{  Ä  ,0} = 

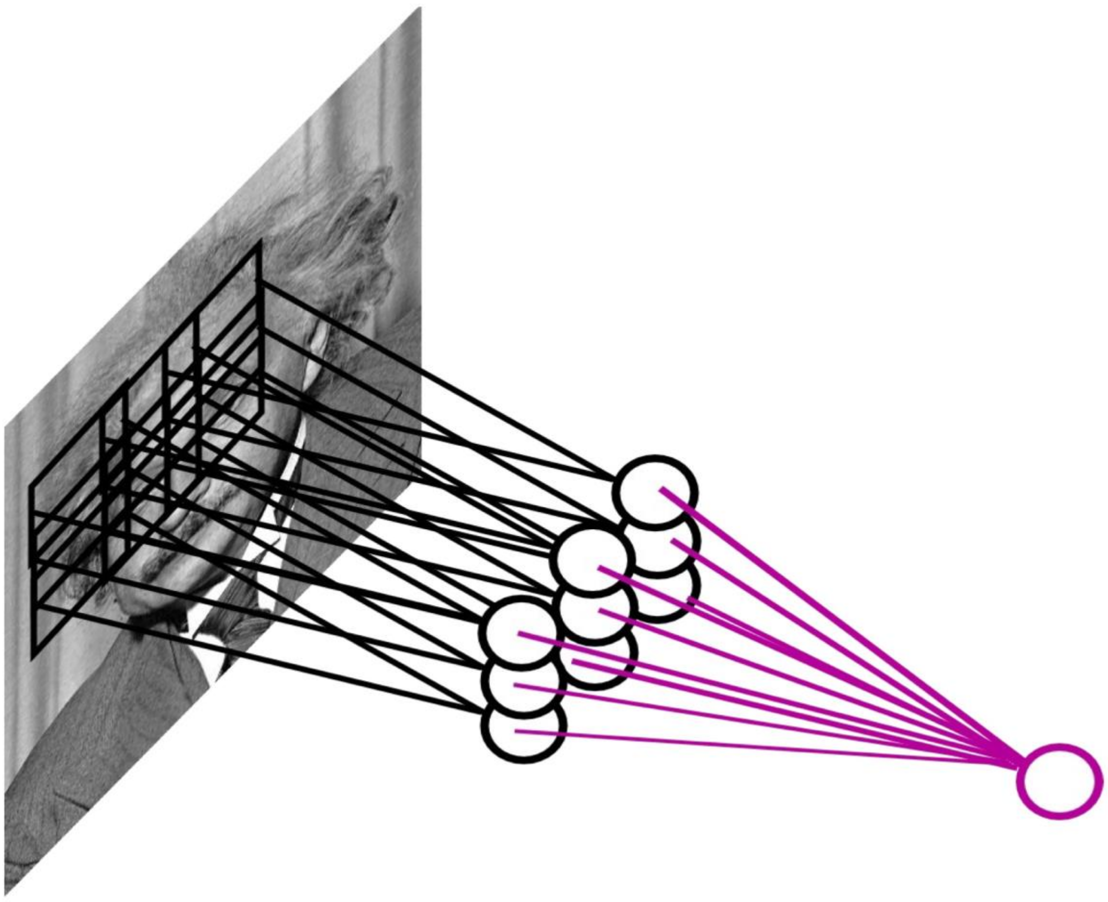
Max Pooling



- Assume Convolution Layer is eye detector
- How to make detector more robust to the exact location of the eye?

http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/tutorial_p2_nnets_ranzato_short.pdf

Max Pooling

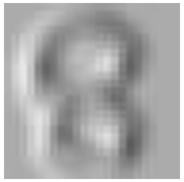


http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/tutorial_p2_nnets_ranzato_short.pdf

Local Contrast Normalization

- Standardize output of convolutional layer using mean & variability estimated from neighboring outputs

- Simple Example:

f: 

$$f_{ij} = \frac{f_{ij} - m_{ij}}{s_{ij}}$$

$$m_{ij} = \text{mean}\left\{ f_{i'j'} \,\middle|\, (i', j') \text{ close to } (i, j) \right\}$$

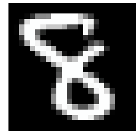
$$s_{ij}^2 = \text{mean}\left\{ \left(f_{i'j'} - m_{i'j'}\right)^2 \,\middle|\, (i', j') \text{ close to } (i, j) \right\}$$

Biologically Inspired!

- Other examples in references below:

http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/tutorial_p2_nnets_ranzato_short.pdf
http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

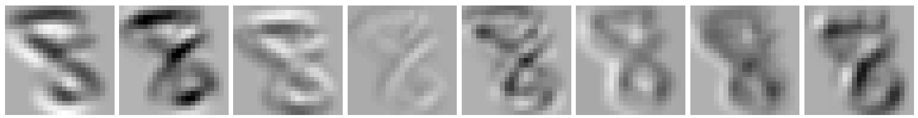
input (24x24x1)
max activation: 1, min: 0



Input

conv (23x23x8)
filter size 6x6x1, stride 1
max activation: 3.73813, min: -8.09174

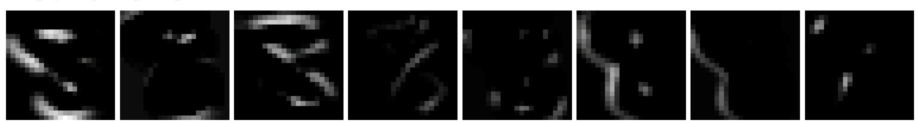
Activations:



8 Convolutional Filters in 1st Layer

relu (23x23x8)
max activation: 3.73813, min: 0
max gradient: 0.00316, min: -0.00215

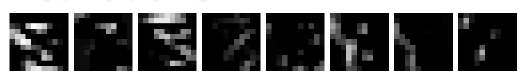
Activations:



Rectilinear Transform

pool (11x11x8)
pooling size 2x2, stride 2
max activation: 3.29955, min: 0

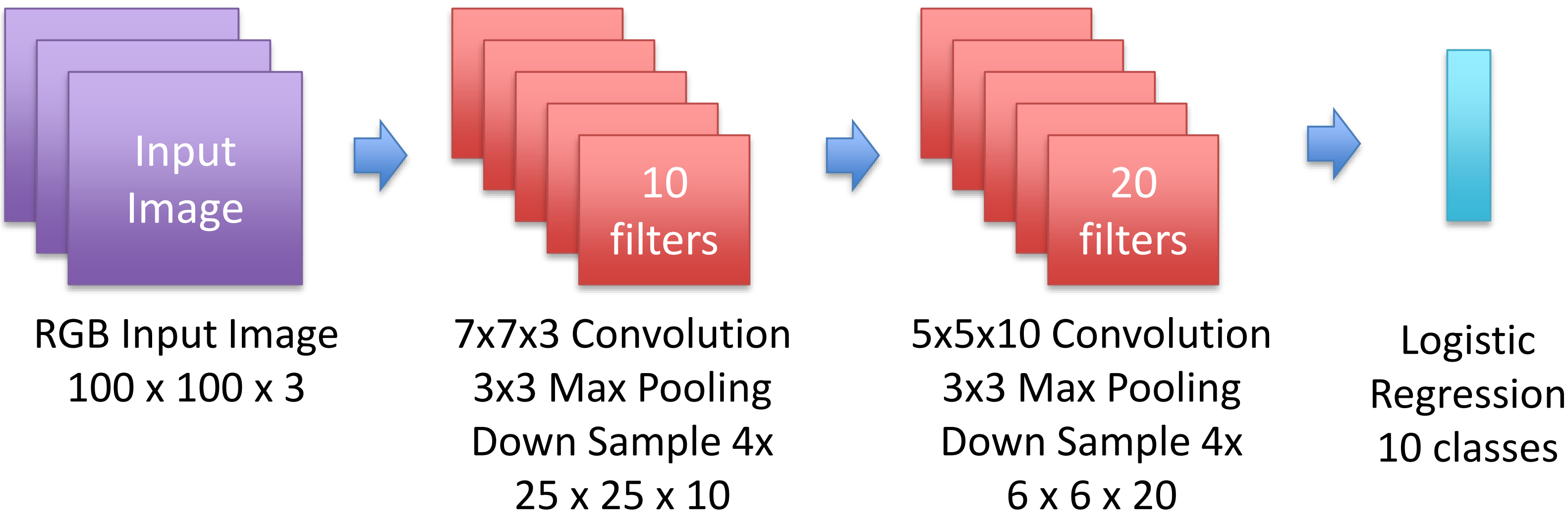
Activations:



Max Pooling

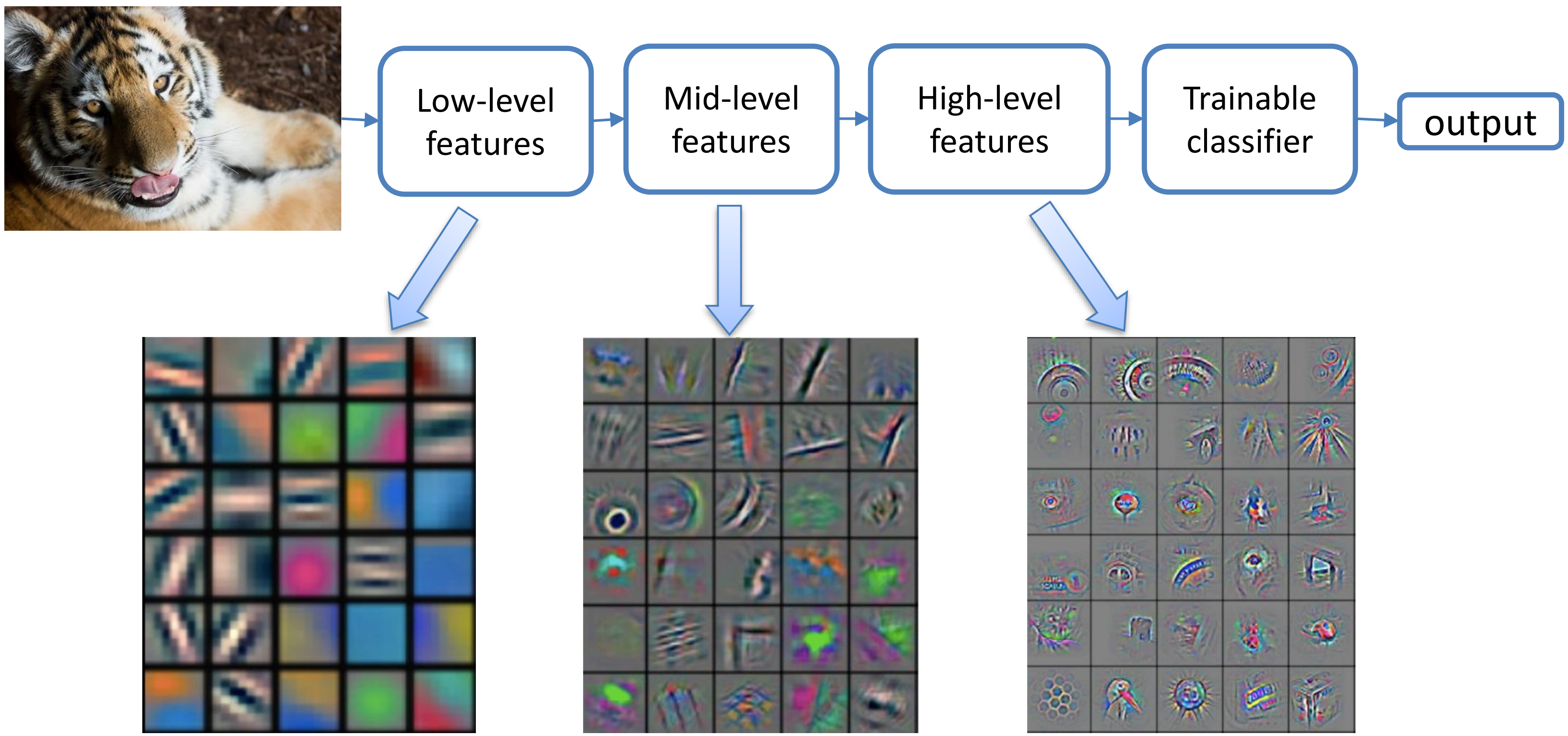
http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

Deep Convolutional Networks



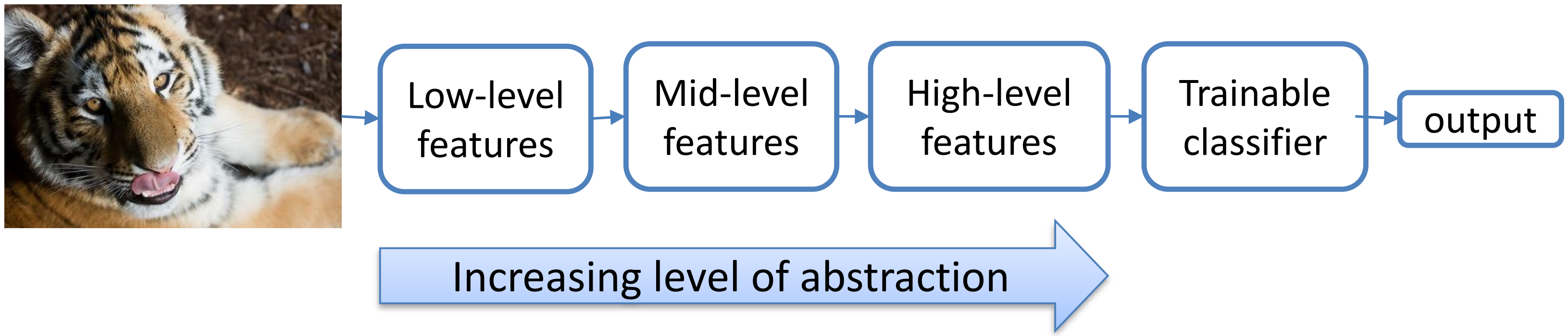
- Stack multiple layers together
- Multiclass logistic regression at top
- Train using gradient descent

Multiple-layer Structure



Feature visualization of convolutional net trained on ImageNet (Zeiler and Fergus, 2013)

Learning Hierarchical Representations



- Hierarchy of representations with increasing level of abstraction.
  - Each stage is a kind of trainable nonlinear feature transformation
- Image recognition
  - Pixel → edge → motif → part → object
- Text
  - Character → word → word group → clause → sentence → story

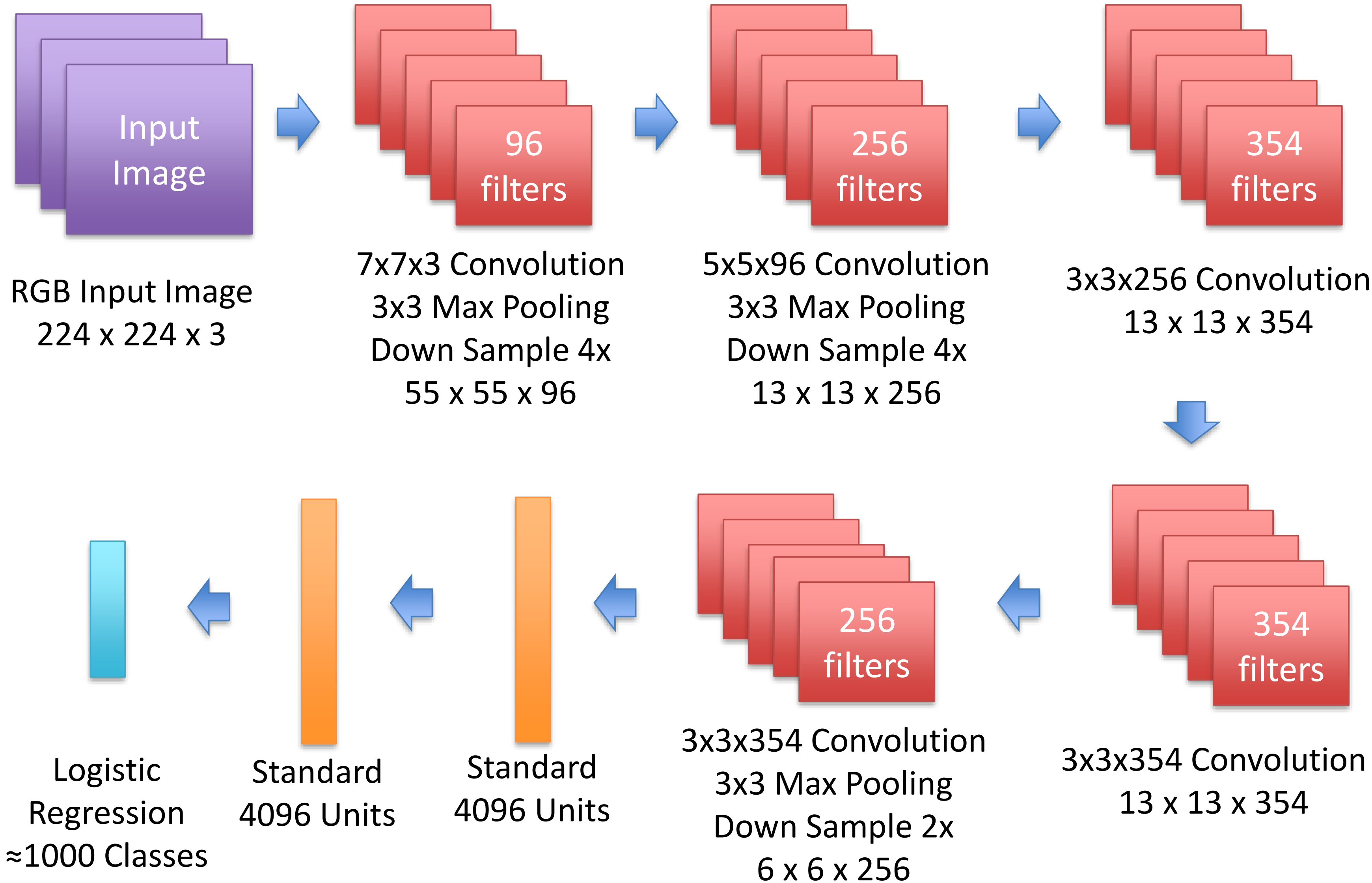
Online Demo

http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

ImageNET

- Object recognition competition (2012)
  - 1.5 Million Labeled Training Examples
  - ≈1000 classes
- 7 Hidden Layers
  - 5 Convolutional
  - 2 Regular
- Trained using stochastic gradient descent
  - And a lot of tricks
- Won the 2012 ImageNET competition

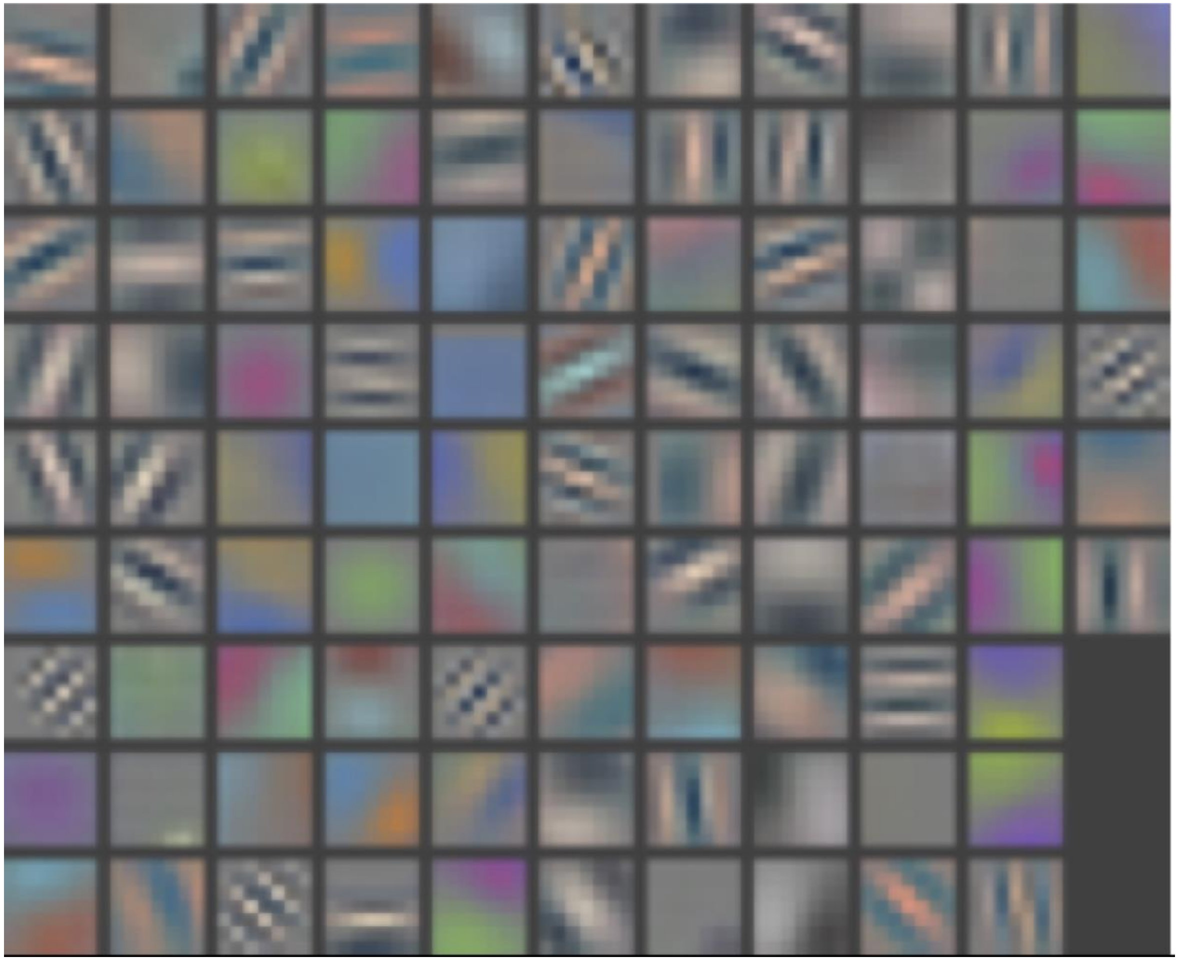
http://www.image-net.org/



http://www.image-net.org/

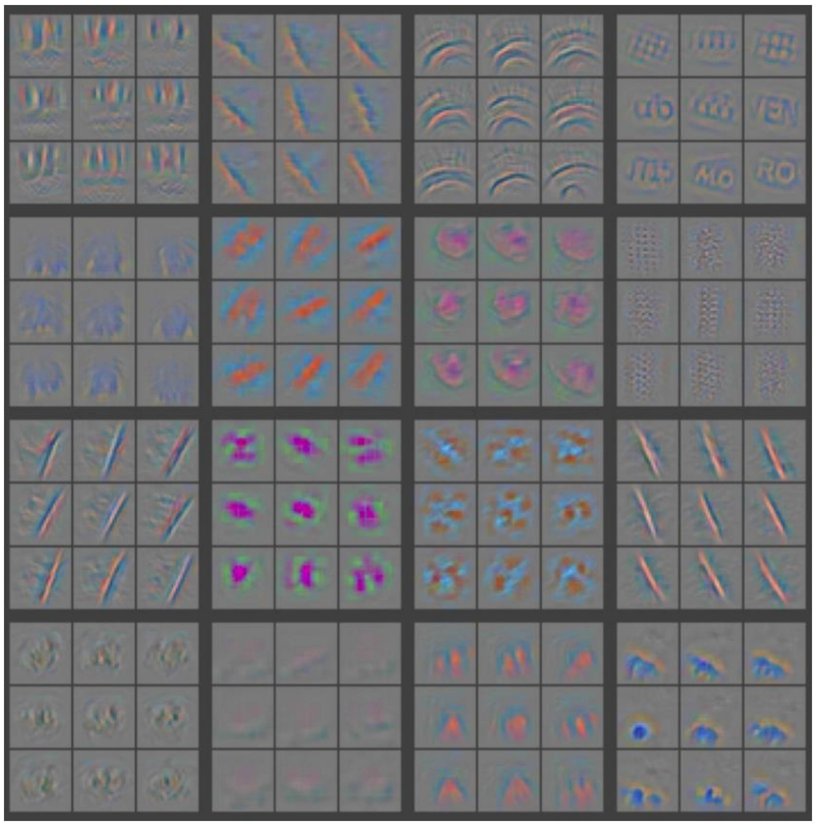
http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf
http://ftp.cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf

Visualizing CNN (Layer 1)

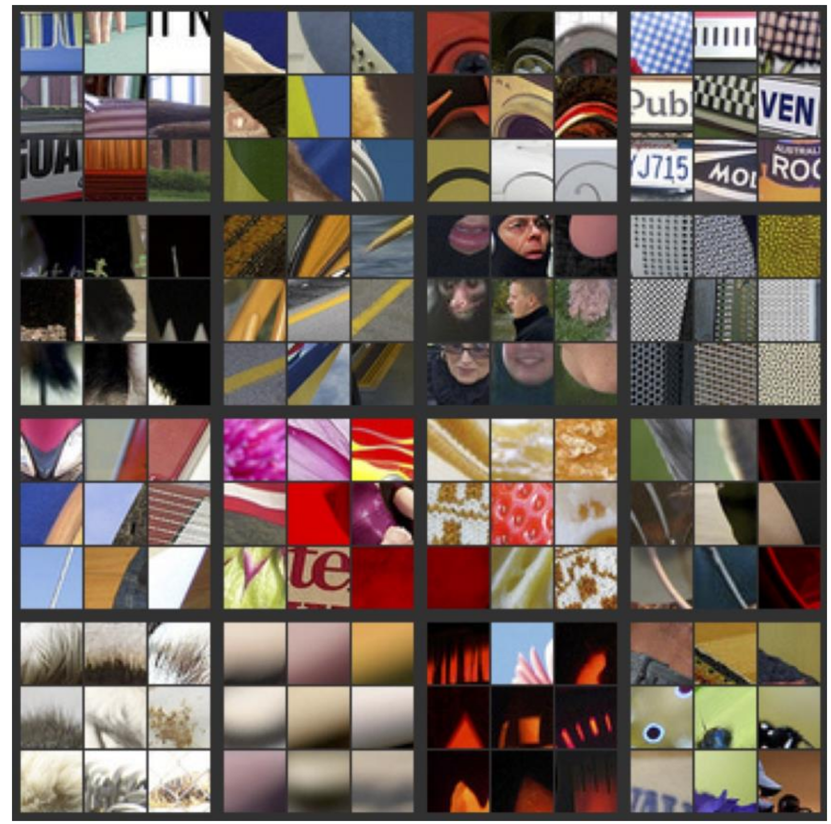


http://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf
http://cs.nyu.edu/~fergus/presentations/nips2013_final.pdf

Visualizing CNN (Layer 2)



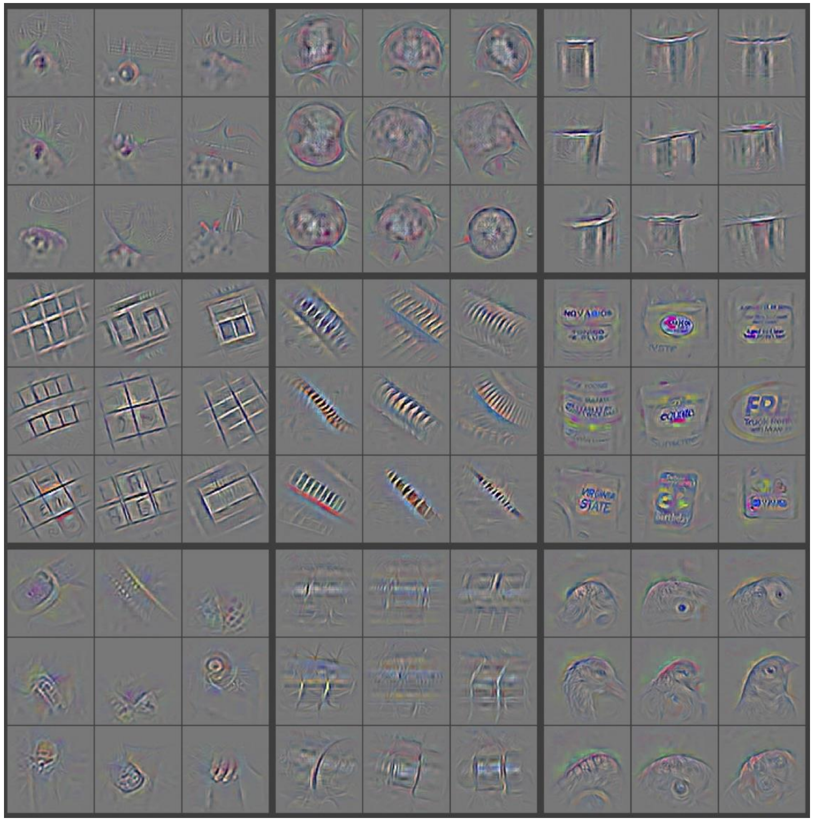
Part that Triggered Filter



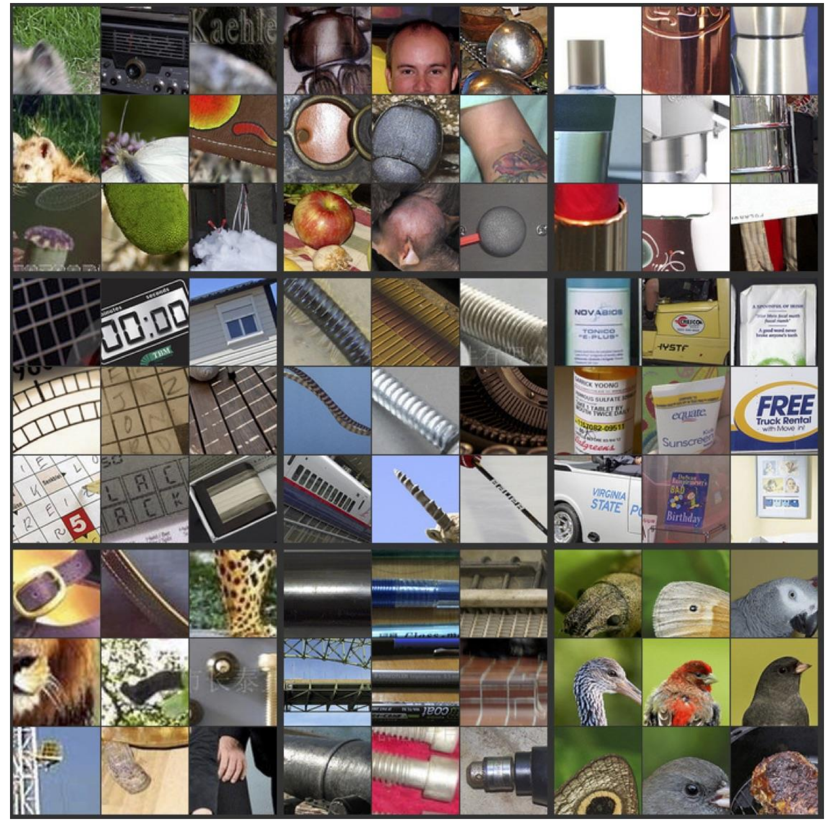
Top Image Patches

http://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf
http://cs.nyu.edu/~fergus/presentations/nips2013_final.pdf

Visualizing CNN (Layer 3)



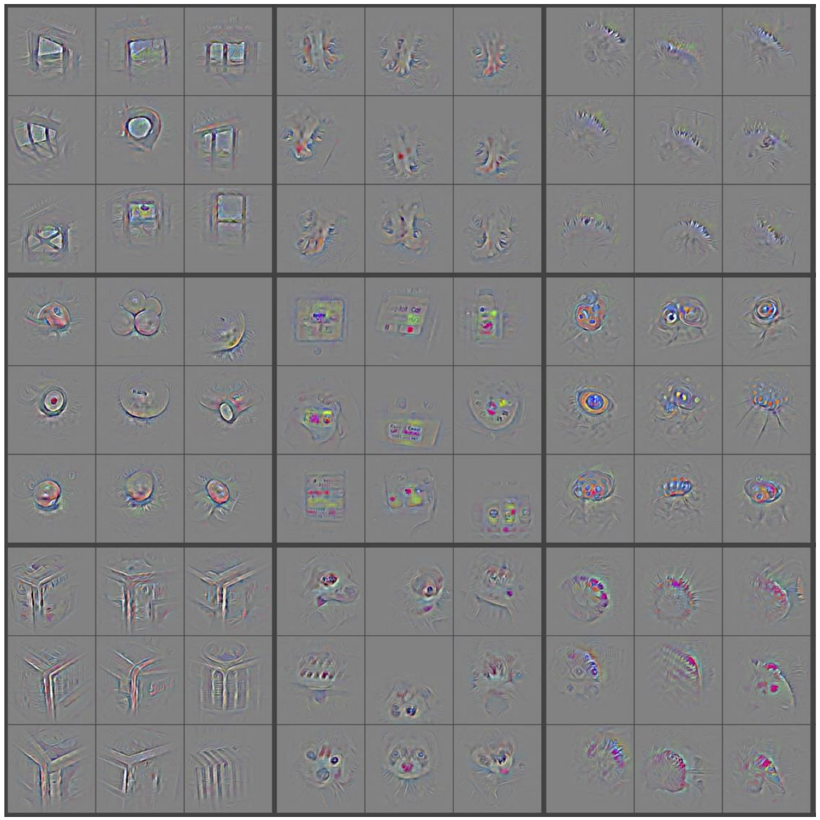
Part that Triggered Filter



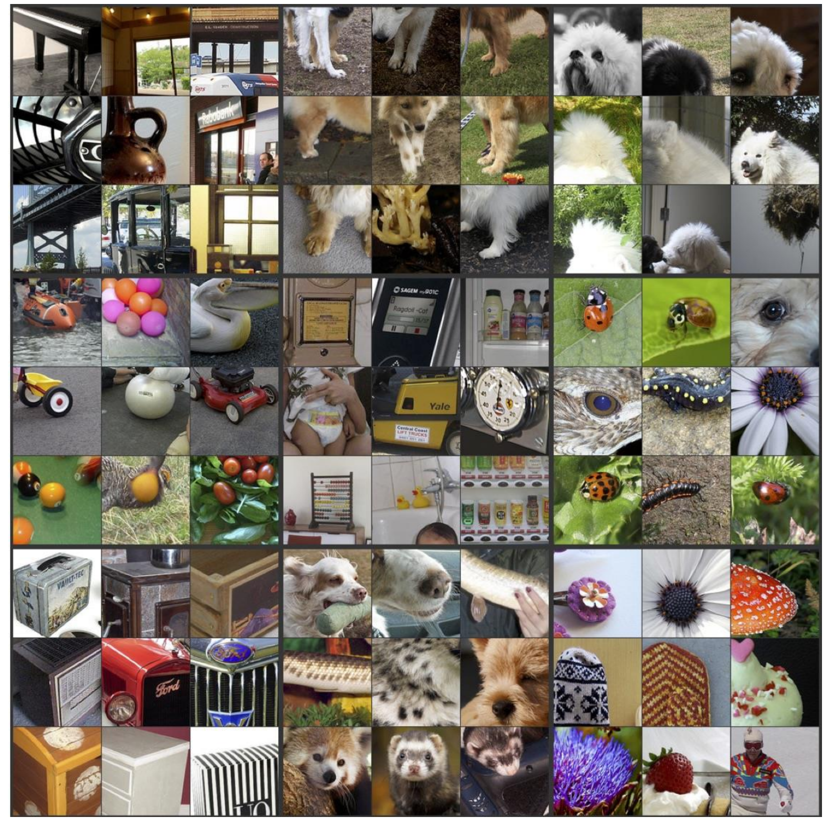
Top Image Patches

http://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf
http://cs.nyu.edu/~fergus/presentations/nips2013_final.pdf

Visualizing CNN (Layer 4)



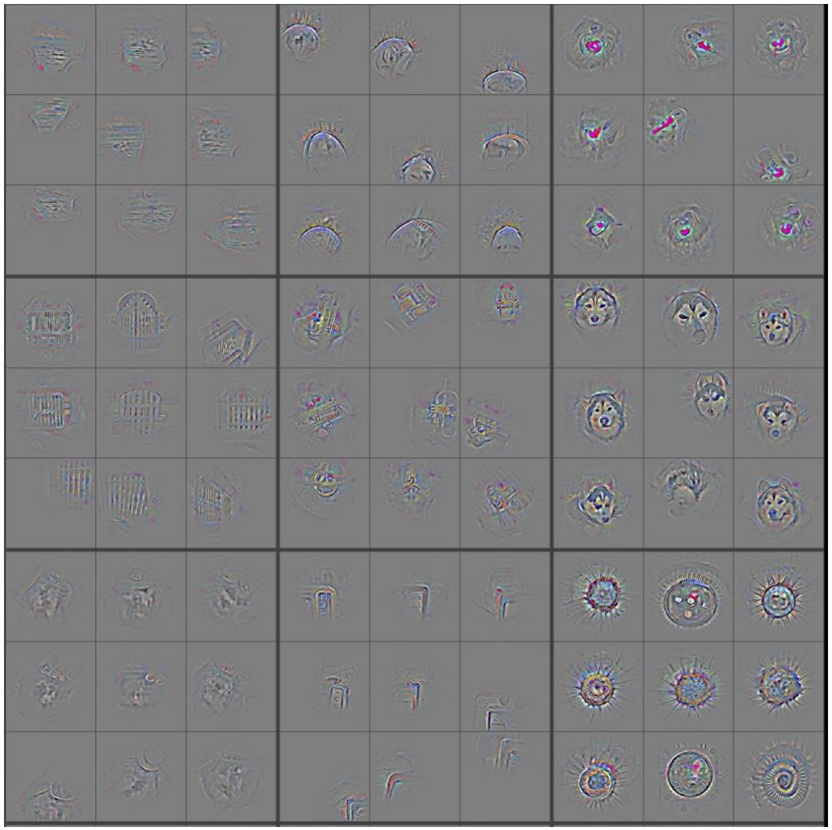
Part that Triggered Filter



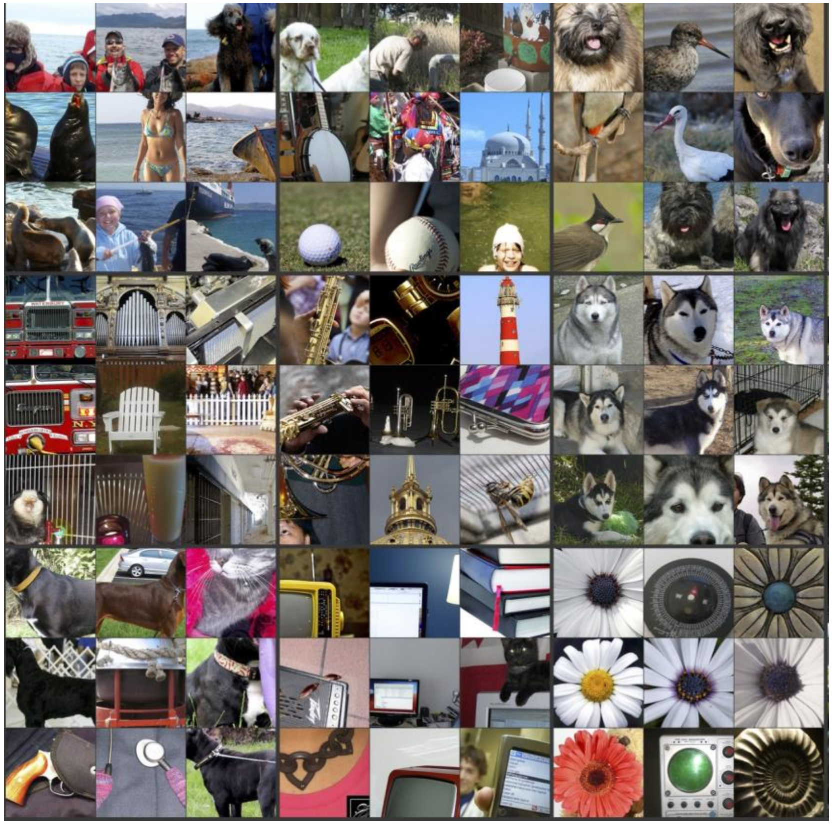
Top Image Patches

http://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf
http://cs.nyu.edu/~fergus/presentations/nips2013_final.pdf

Visualizing CNN (Layer 5)



Part that Triggered Filter



Top Image Patches

http://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf
http://cs.nyu.edu/~fergus/presentations/nips2013_final.pdf

Stochastic Gradient Descent + Tricks!

- Some related to choice of model/architecture
  - Rectilinear over sigmoid transfer functions
  - Local contrast normalization
  - Sparse Connections that enable parallelism
  - Ensemble of Deep Networks
- Rest are optimization techniques
  - Gradient Clamping
  - Mini-batching
  - Momentum
  - Adaptive Learning Rates
  - Random Initialization
  - Dropout

http://yyue.blogspot.com/2015/01/a-brief-overview-of-deep-learning.html

Learning Rate & Momentum

$$w = w - \eta \partial_w$$

Gradient Descent

- If validation performance plateaus or gets worse
  - Divide learning rate by 2

$$w = w - \eta \partial_w + \gamma \mu_w$$

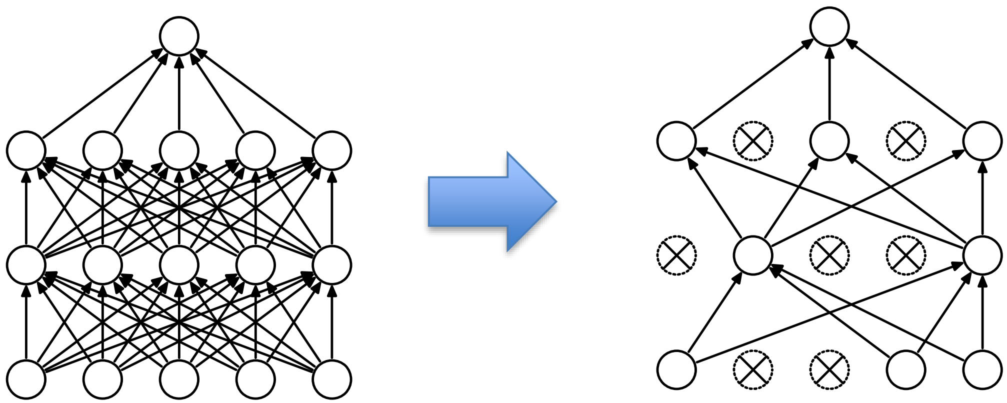
Momentum

- Momentum is a weighted combination of recent gradient updates

http://www.cs.toronto.edu/~fritz/absps/momentum.pdf

Dropout

- Randomly turn off nodes during training

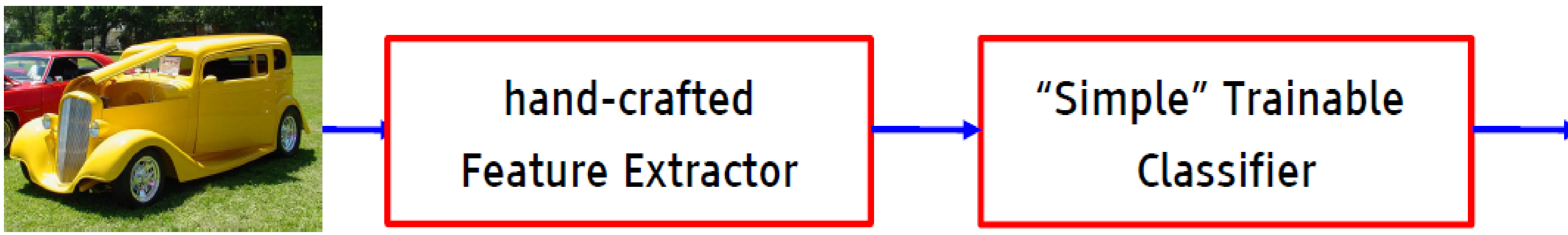


- Choose randomly for each SGD minibatch
  - Decorrelates node in each layer
  - Less overfitting

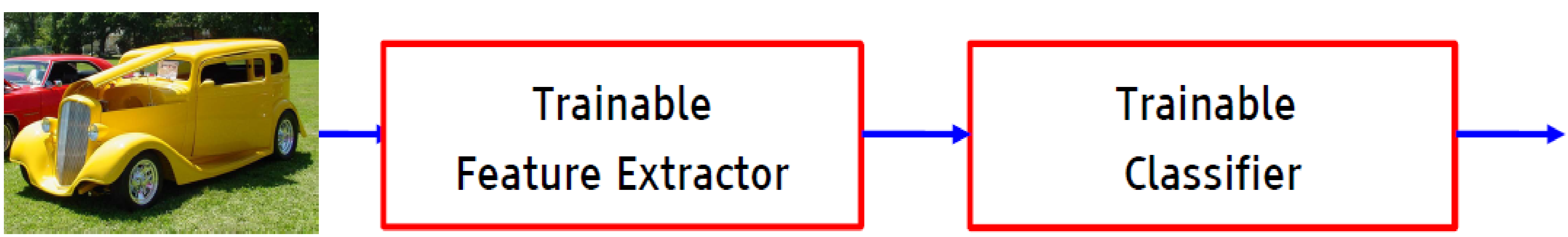
http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

Deep Learning = Learning Representations/Features

- The traditional model of pattern recognition (since the late 50's)
  - Fixed/engineered features (or fixed kernel) + trainable classifier

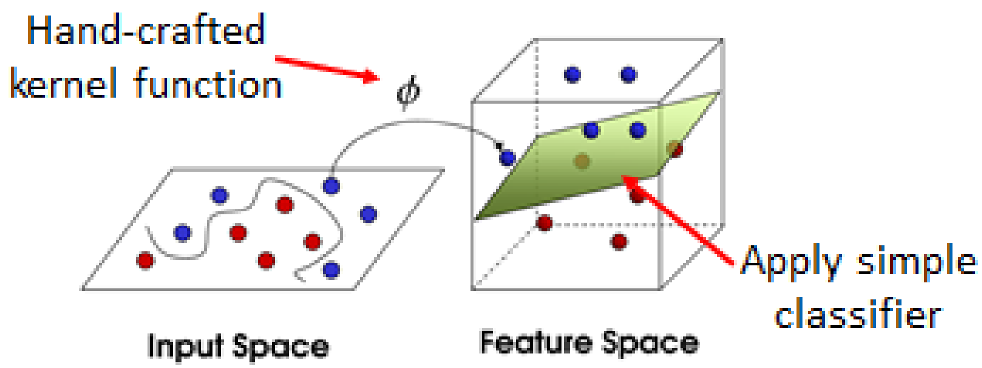


- End-to-end learning / Feature learning / Deep learning
  - Trainable features (or kernel) + trainable classifier



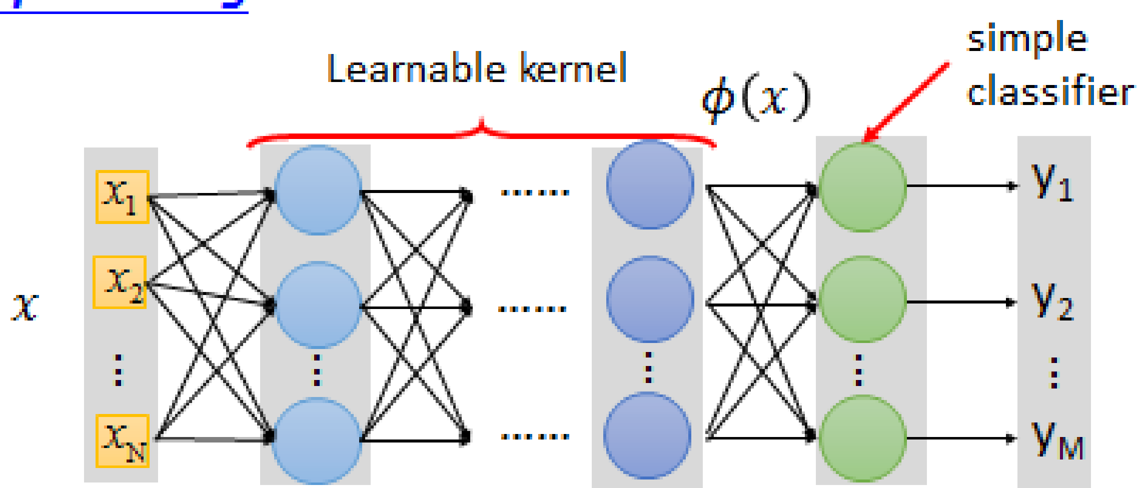
SVM vs. Deep Learning

SVM



Source of image: http://www.gipsa-lab.grenoble-inp.fr/transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

Deep Learning

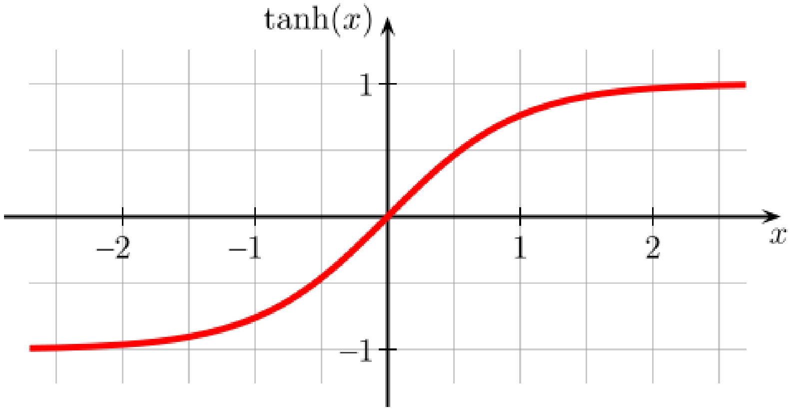


Resources

- http://caffe.berkeleyvision.org/
- http://deeplearning.net/software/theano/
- http://torch.ch/
- https://code.google.com/p/cuda-convnet/
- http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/
- http://deeplearning.net/tutorial/
- http://deeplearning.stanford.edu/tutorial/
- http://nlp.stanford.edu/sentiment/

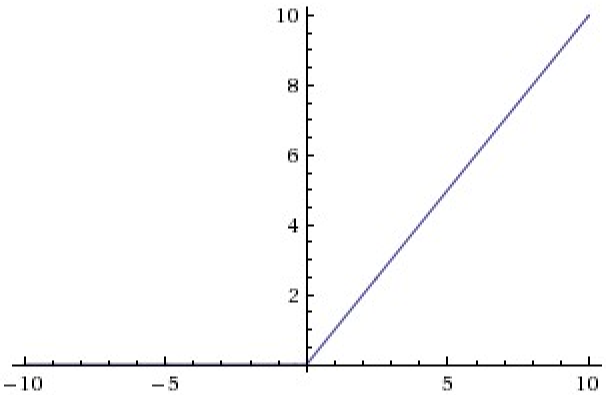
Non-linear stage

Tanh(x)



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

ReLU



$$f(x) = \max(0, x)$$

Pooling

- Common pooling operations:
  - Max pooling
    - Report the maximum output within a rectangular neighborhood.
  - Average pooling
    - Report the average output of a rectangular neighborhood (possibly weighted by the distance from the central pixel).