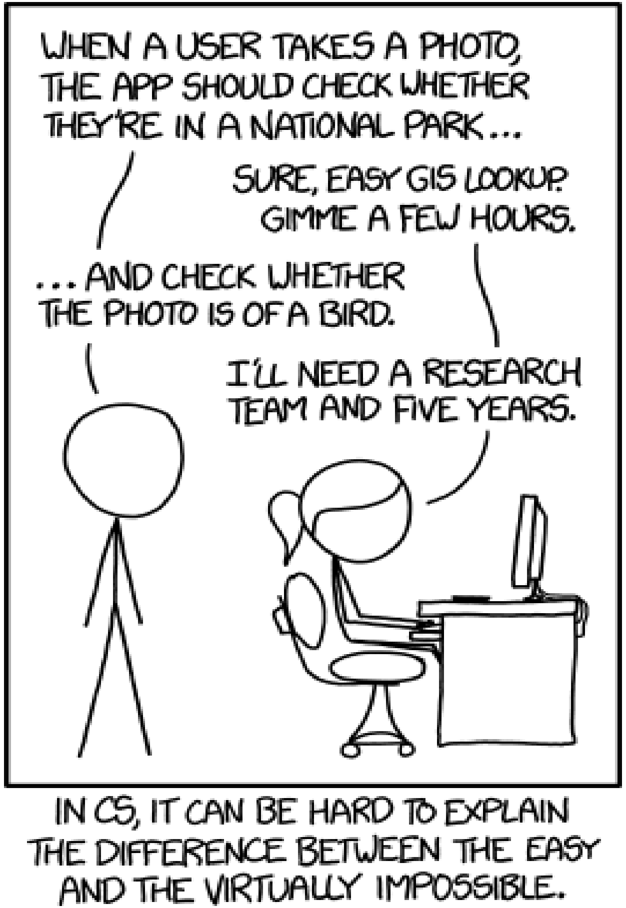


Courtesy of xkcd.com. Used with permission.

Image Classification via Deep Learning

Ryan Alexander, Ishwarya Ananthabhotla, Julian Brown,
Henry Nassif, Nan Ma, Ali Soylemezoglu

Overview

- What is Deep Learning?
- Image Processing
- CNN Architecture
- Training Process
- Image Classification Results
- Limitations

Overview

- What is Deep Learning?
- Image Processing
- CNN Architecture
- Training Process
- Image Classification Results
- Limitations

Deep Learning Refers to...

Machine Learning algorithms designed to

extract **high-level abstractions** from data

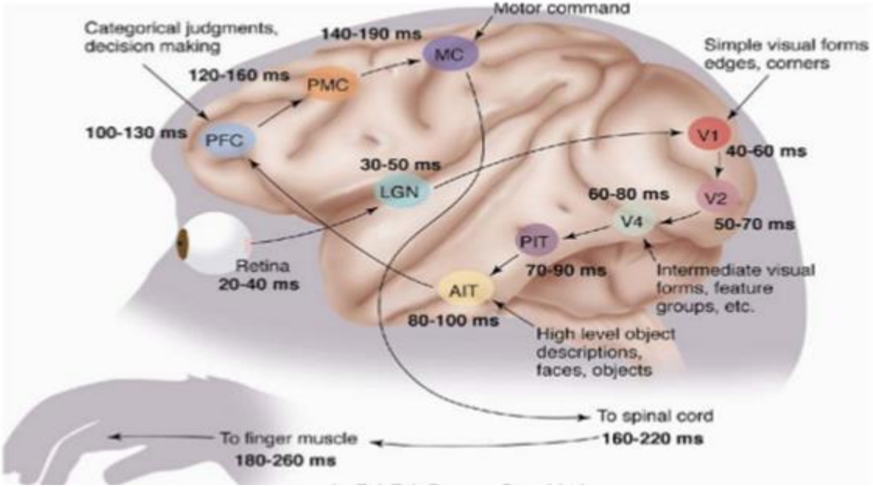
via **multi-layered processing** architectures

using **nonlinear transformations** at each layer

Human Visual System

- Distributed Hierarchical processing in the primate cerebral cortex (1991)

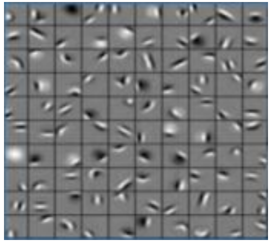
- The ventral (recognition) pathway in the visual cortex
  - Retina → LGN → V1 → V2 → V4 → PIT → AIT (80-100ms)



[picture from Simon Thorpe]

How To Classify a Face?

- Identify where the face region is
  - Foreground Extraction
  - Edge Detection
- Classify features of the face
  - Identify and describe eyes, nose, mouth areas
- Look at face as a collection of those features



Common Architectures

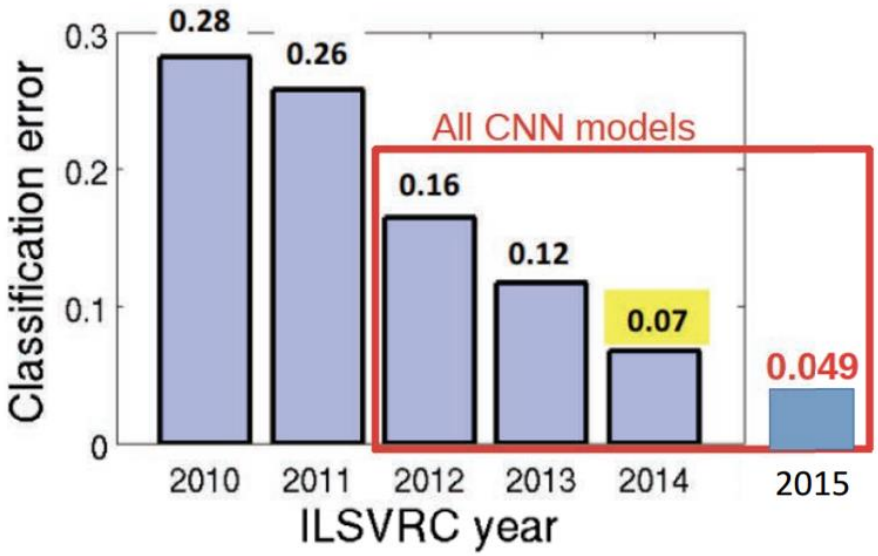
- Deep Convolutional Neural Networks (CNNs)
- Deep Belief Networks (DBNs)
- Recurrent Neural Network

Common Architectures

- Deep Convolutional Neural Networks (CNNs)
- Deep Belief Networks (DBNs)
- Recurrent Neural Network

ImageNet Competition Through Time

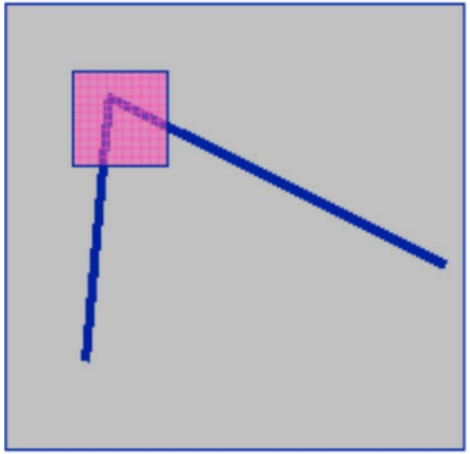
Top-5 error rate

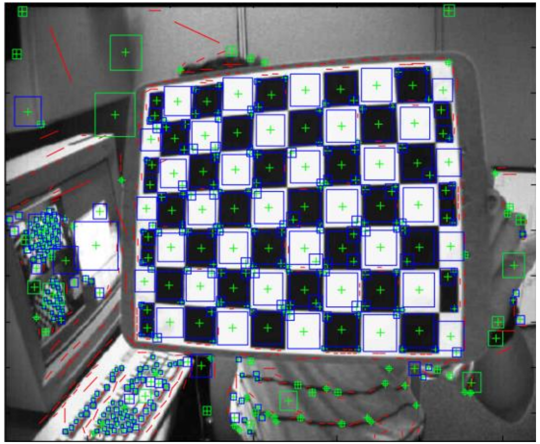


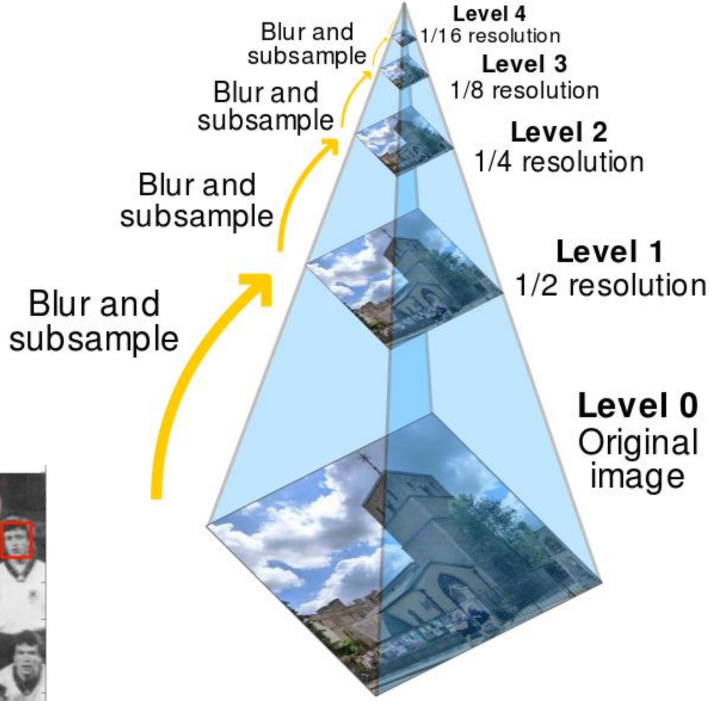
Overview

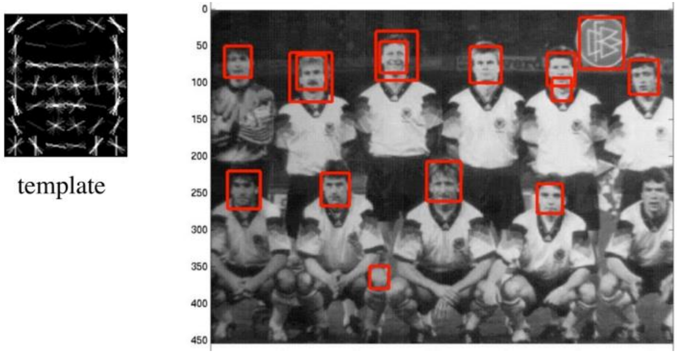
- What is Deep Learning?
- Image Processing
- CNN Architecture
- Training Process
- Image Classification Results
- Limitations

Classic Classification -- Feature Engineering

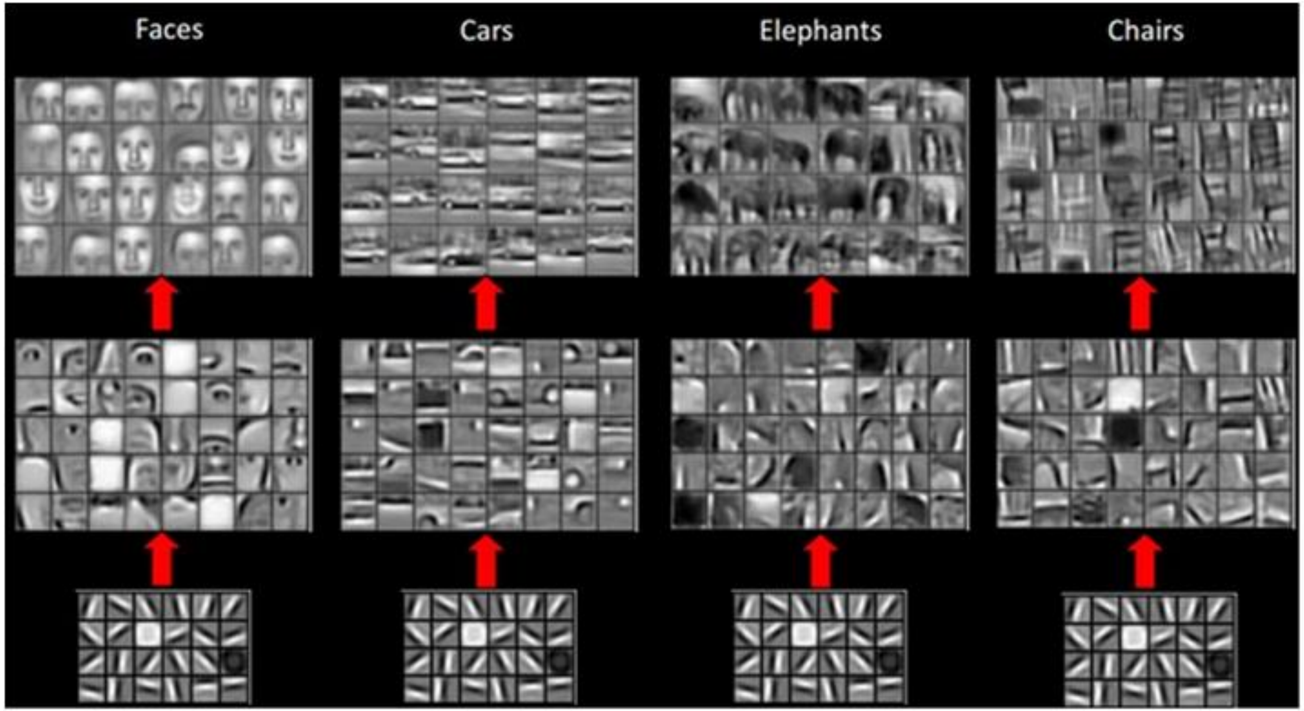








What if the techniques could be "learned"?



Step 1: Convolution - Definition

Informal Definition: Procedure where two sources of information are intertwined.

Formal Definition :

Discrete :

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

Continuous :

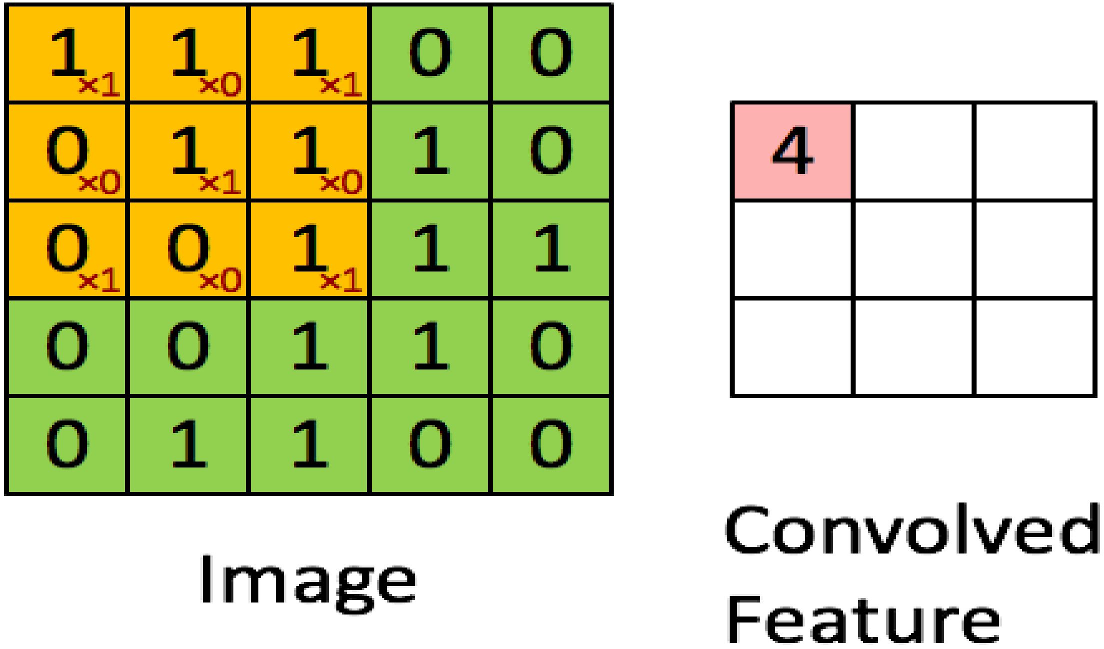
$$f(x,y) * g(x,y) = \int_{\tau_1=-\infty}^{\infty} \int_{\tau_2=-\infty}^{\infty} f(\tau_1,\tau_2) \cdot g(x-\tau_1, y-\tau_2)\, d\tau_1\, d\tau_2$$

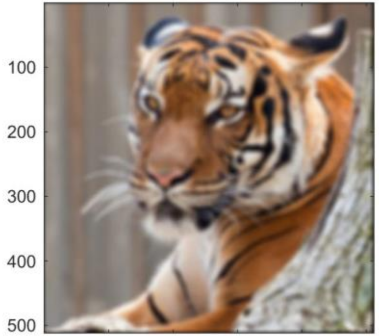
Convolution - Example

Assume the following kernel/filter :

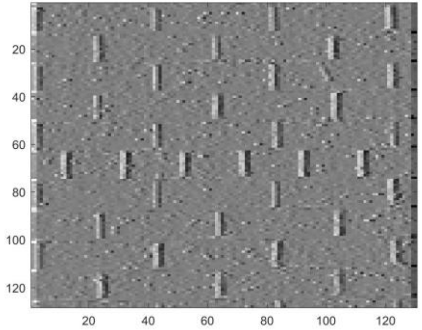
1	0	1
0	1	0
1	0	1

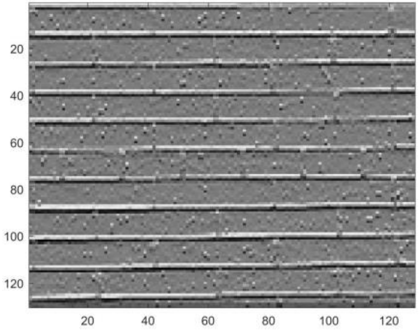
Convolution



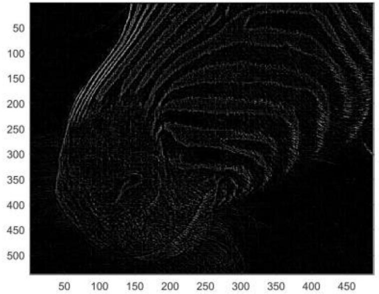


.0113	.0838	.0113
.0838	.6193	.0838
.0113	.0838	.0113





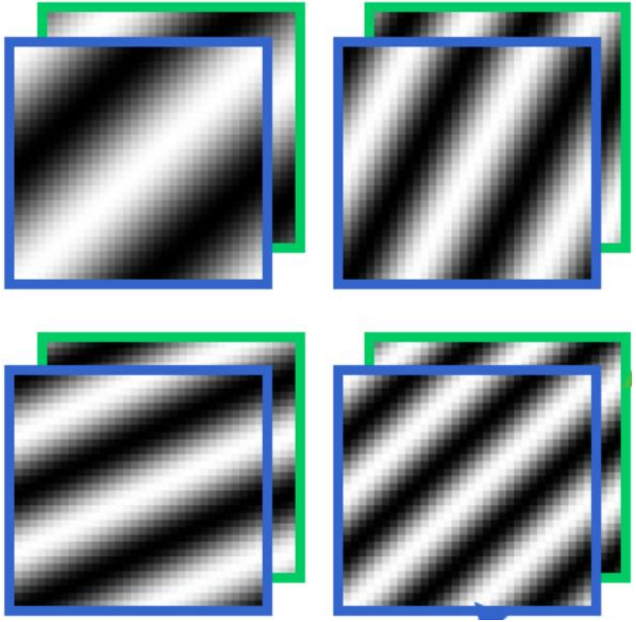
1	2	1
0	0	0
-1	-2	-1



1	1	1
1	-8	1
1	1	1

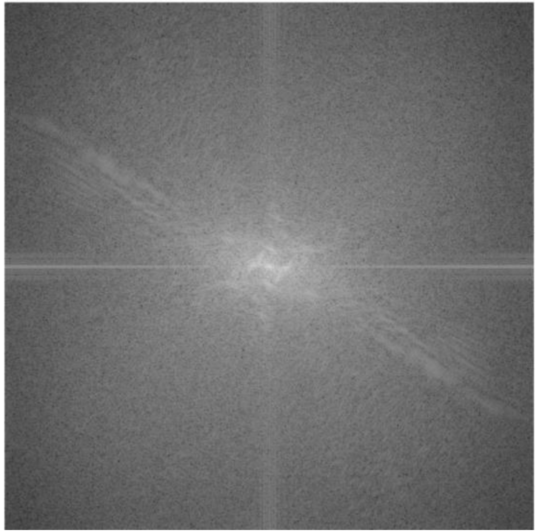
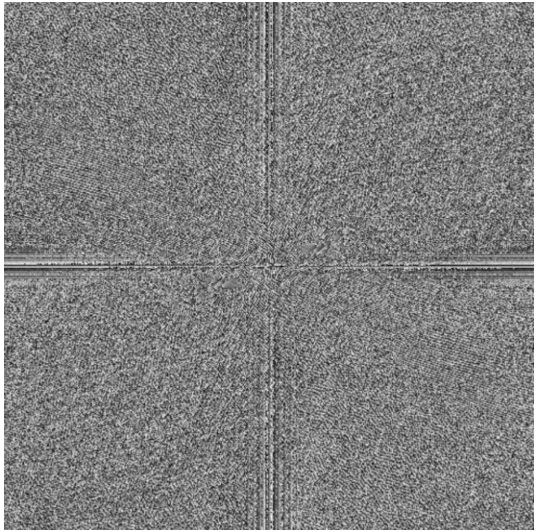
More Information? Fourier Transform!

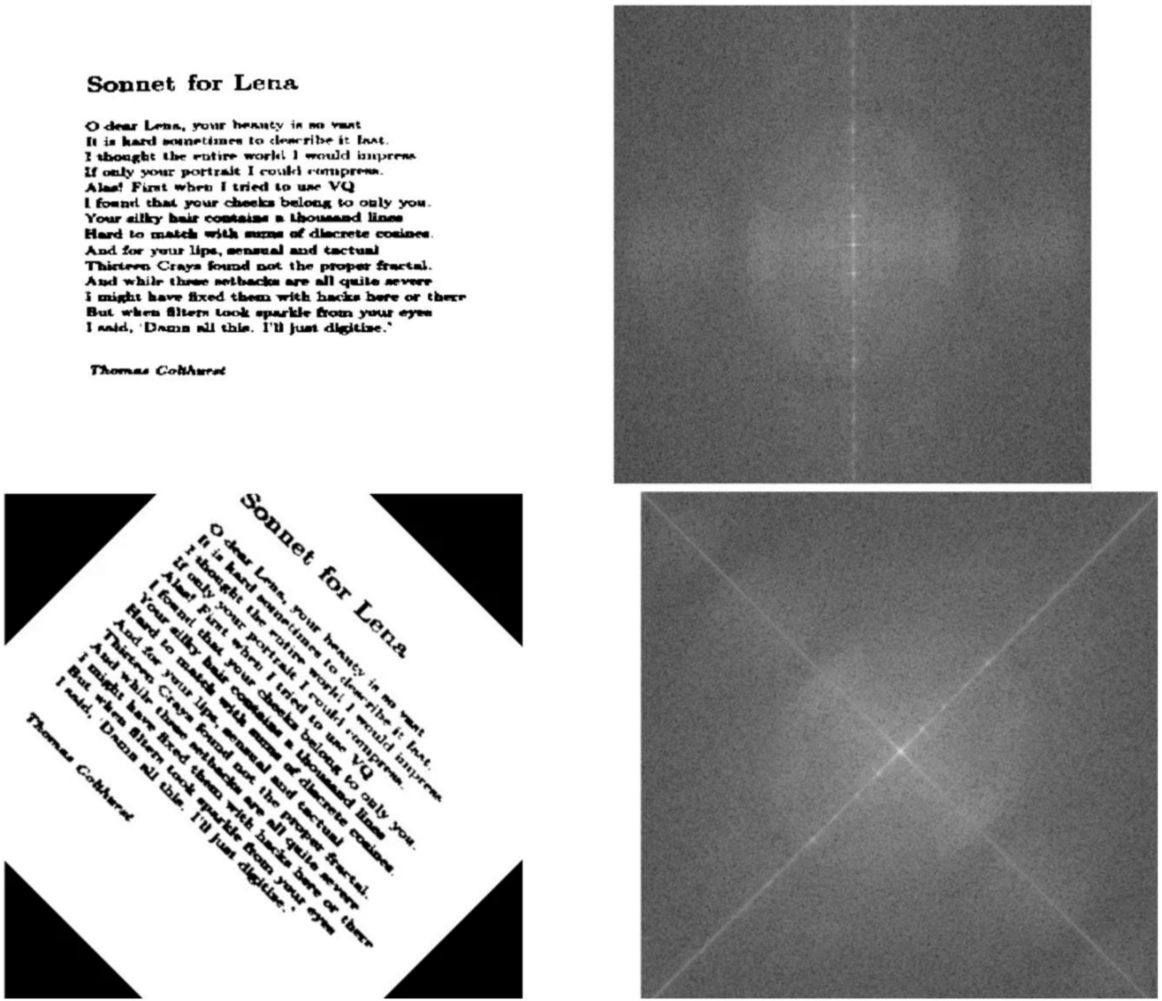
Sum of a set of sinusoidal gratings differing in spatial frequency, orientation, amplitude, phase



Fourier Transform

- Fourier Transform image itself is weird to visualize -- Phase and Magnitude!
- Magnitude -- orientation information at all spatial scales
- Phase -- contour information

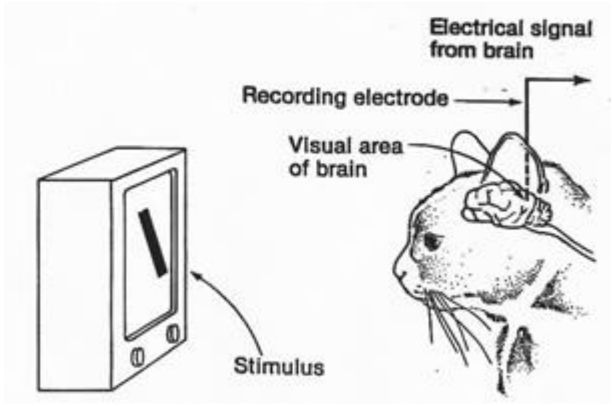

Sonnet for Lena
O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactual
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'
Thomas Colthurst

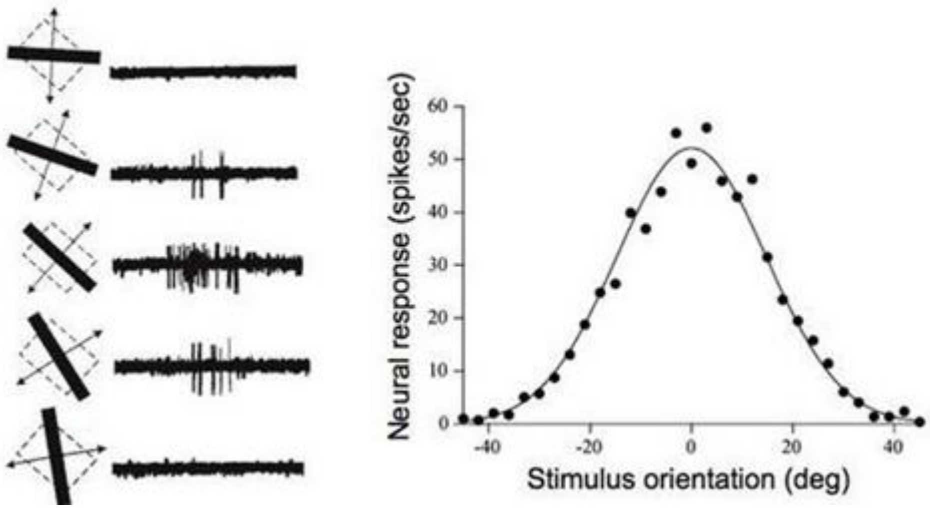
Overview

- What is Deep Learning?
- Image Processing
- CNN Architecture
- Training Process
- Image Classification Results
- Limitations

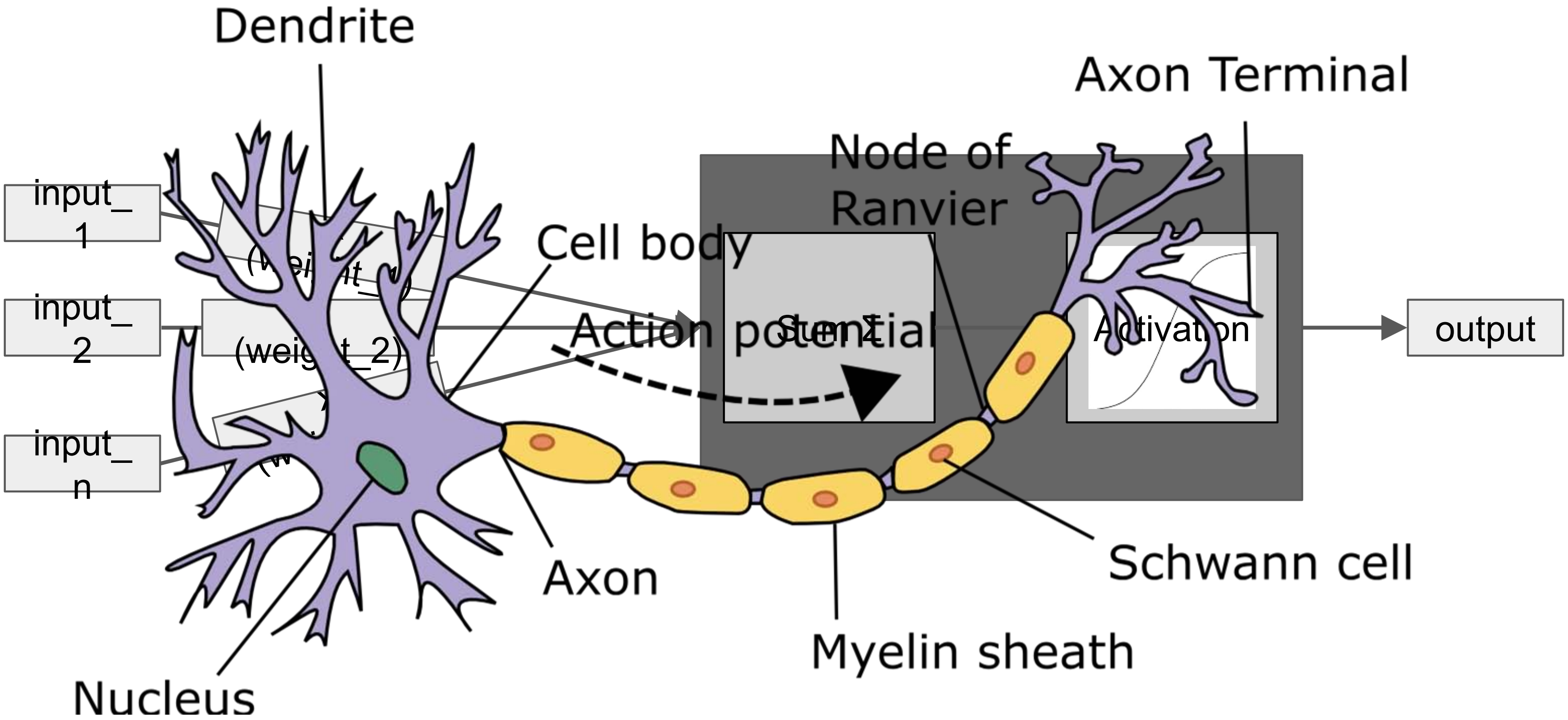
Why Neural Net

Hubel & Wiesel (1959, 1962)

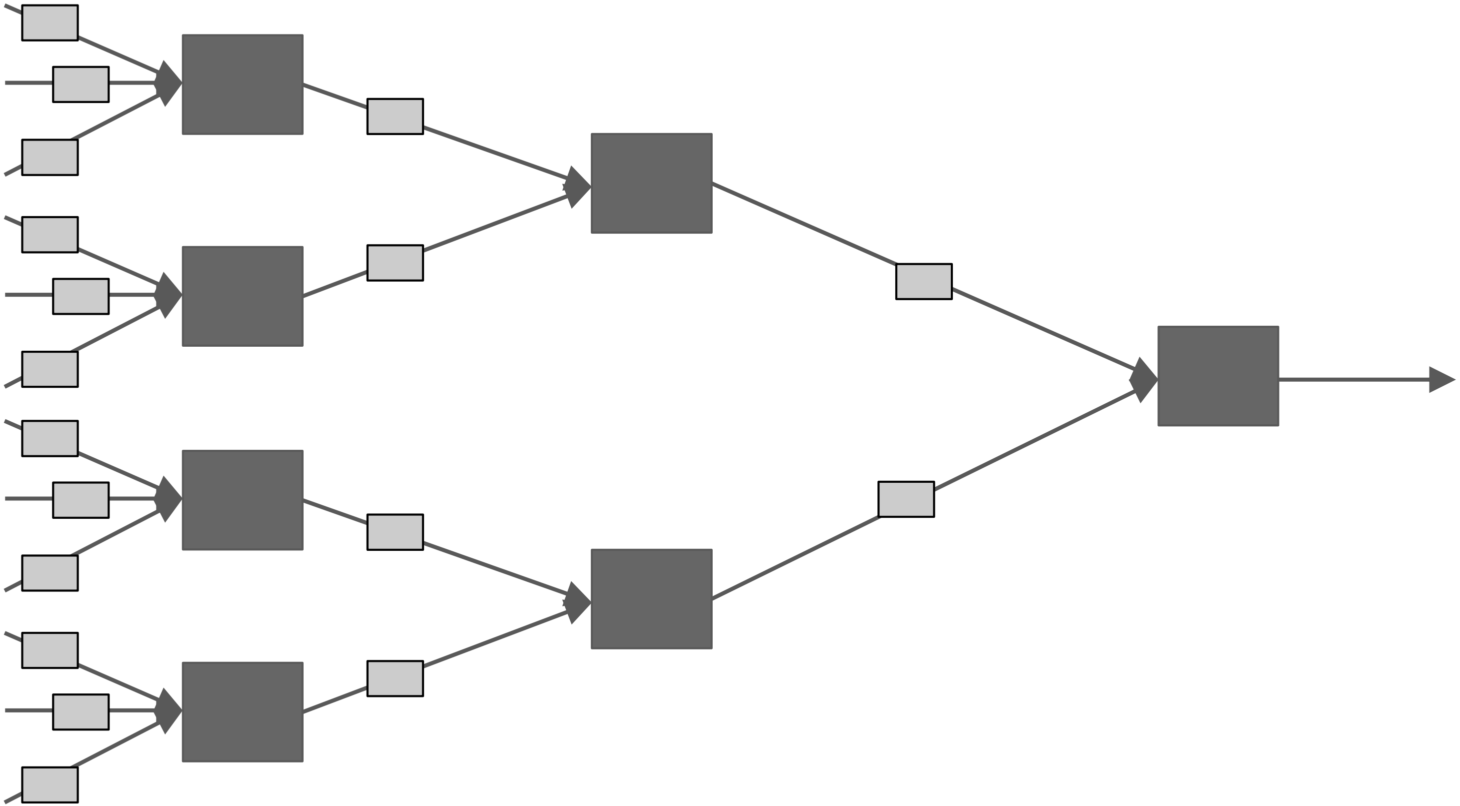




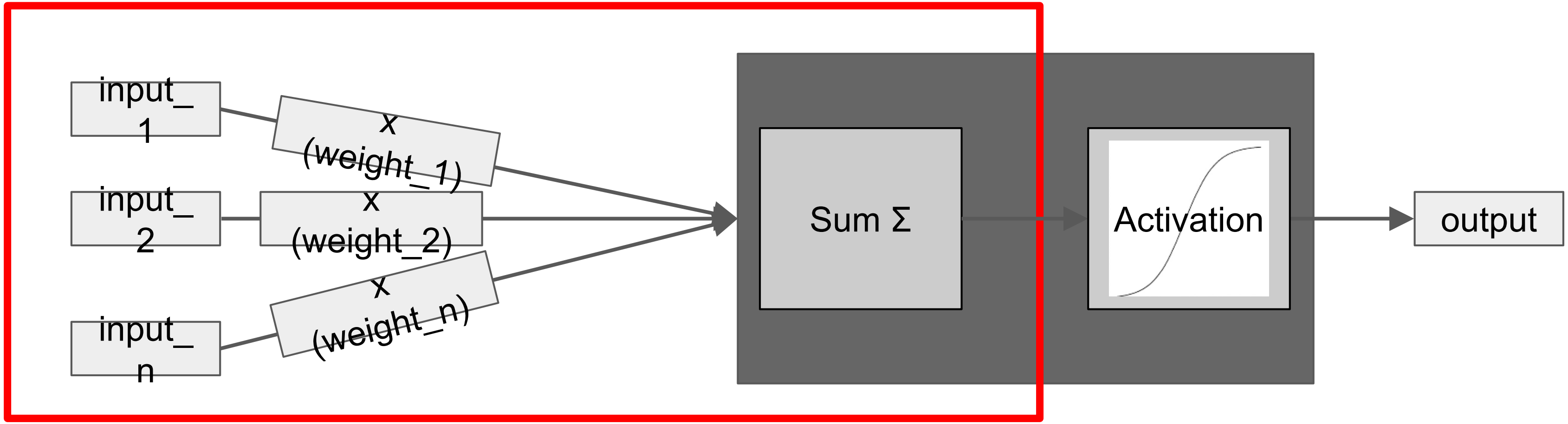
The Structure of a Neuron



Combining Neurons into Nets

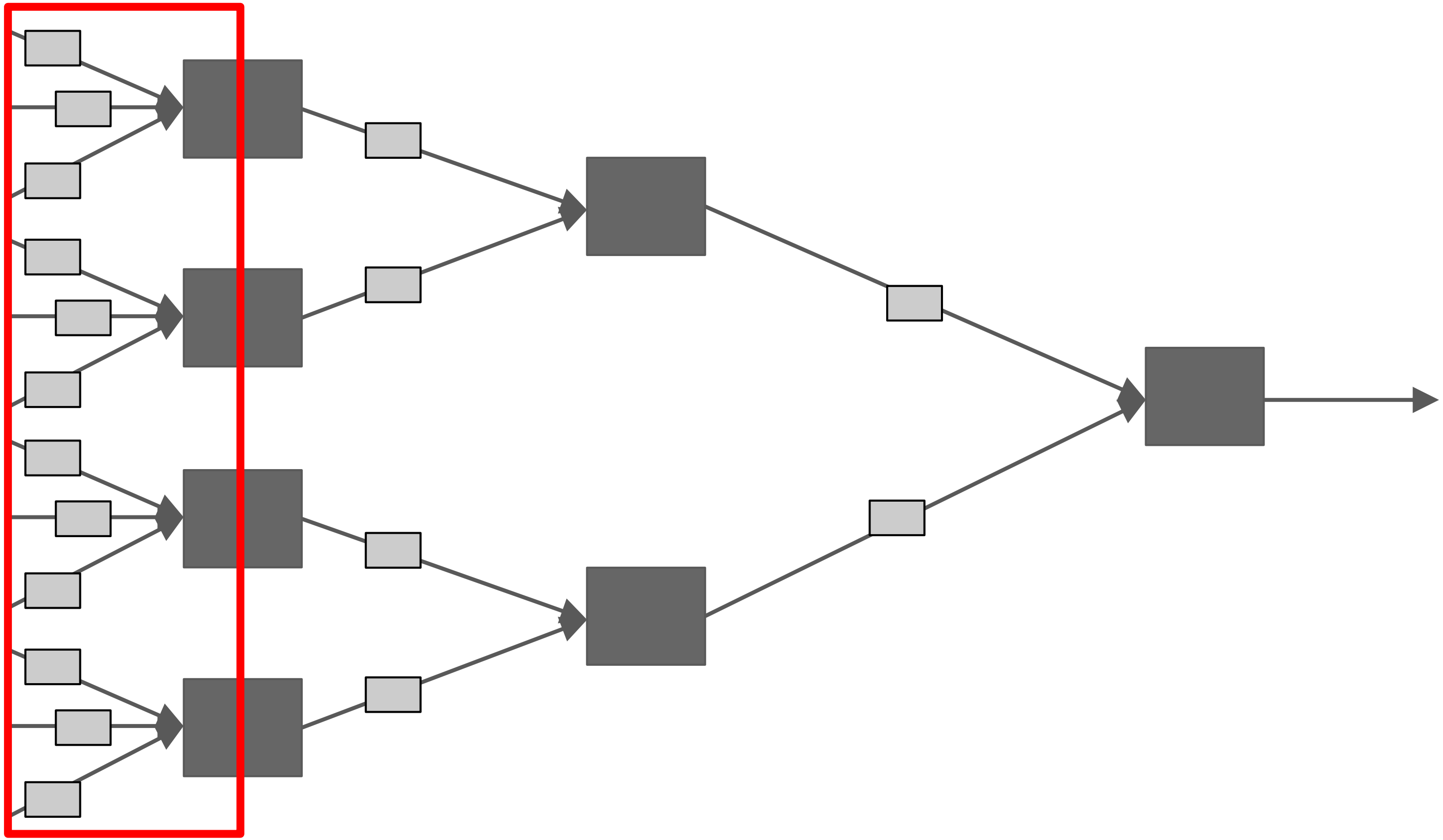


Convolution Step

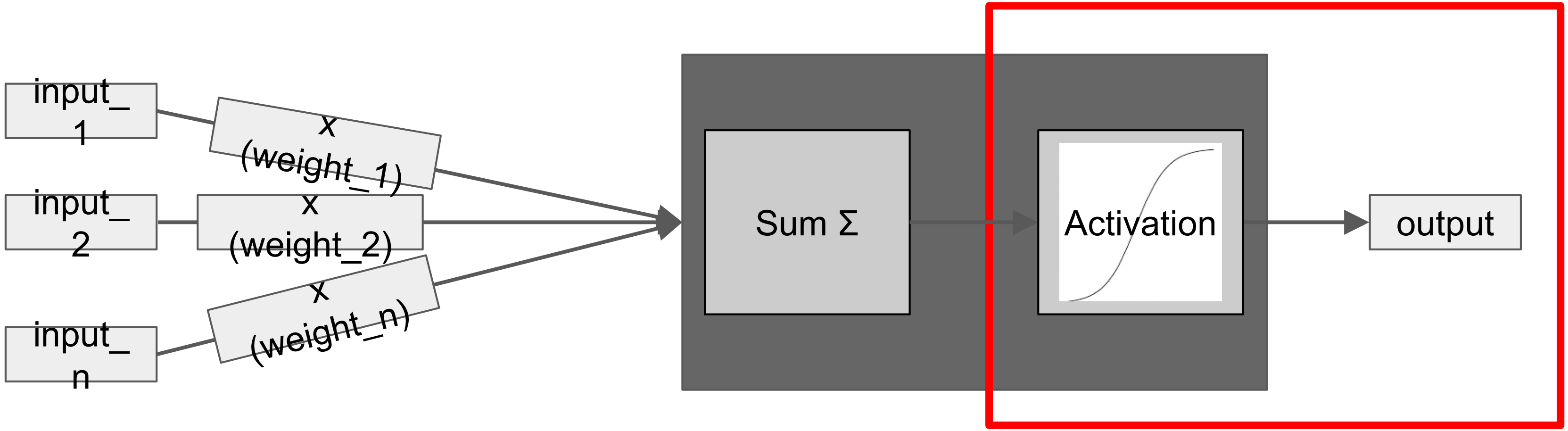


Convolution Step
(dot product between filter and input)

Convolutional Layer

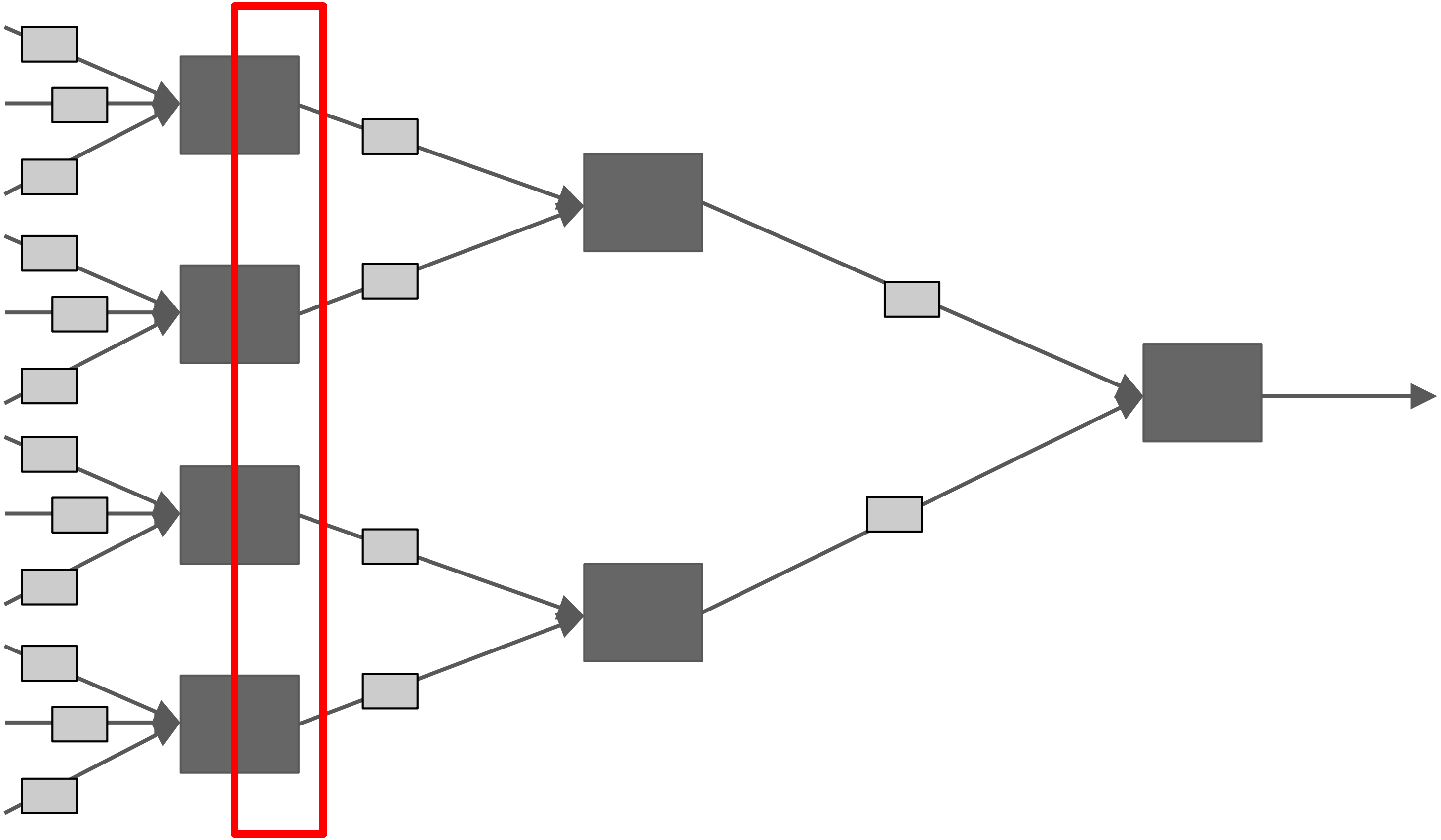


Activation Step

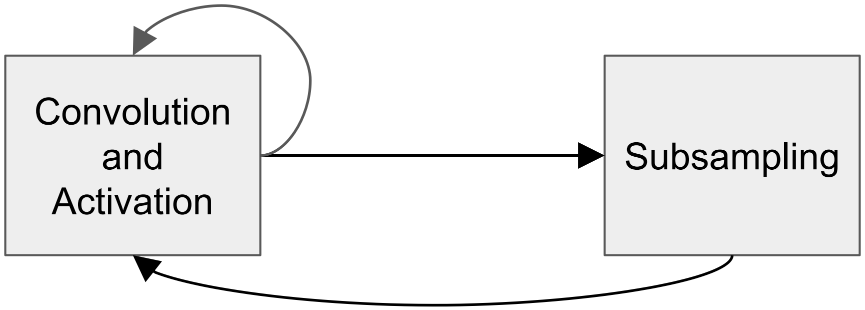


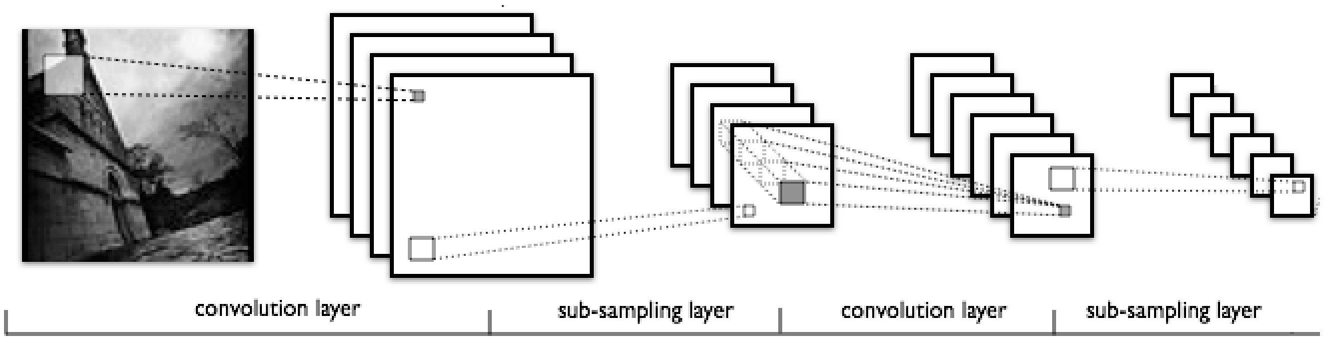
Activation Step

Activation Layer



CNN overview

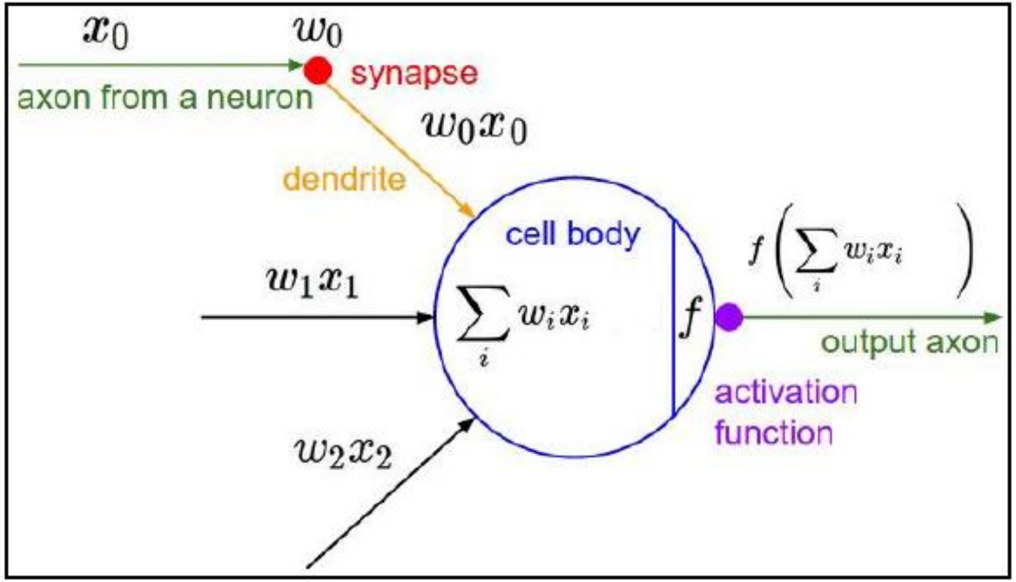




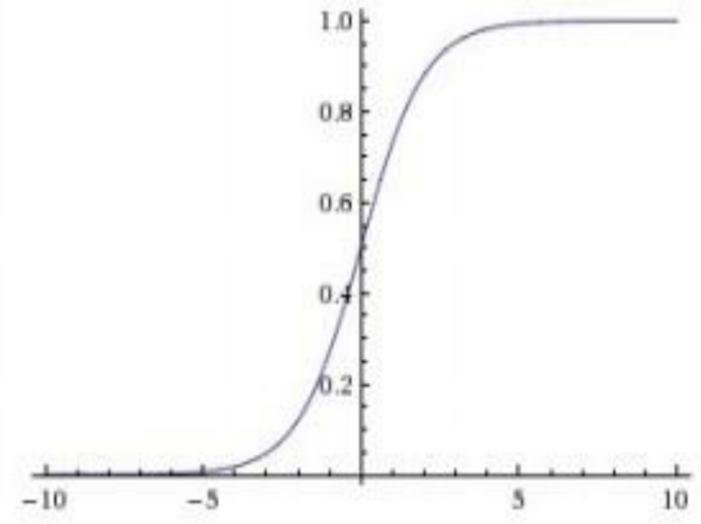
Activation Step

Each neuron adds up its inputs, and then feeds the sum into a function -- the activation function -- to determine the neuron's output.

Eg : Sigmoid, tanh, ReLu



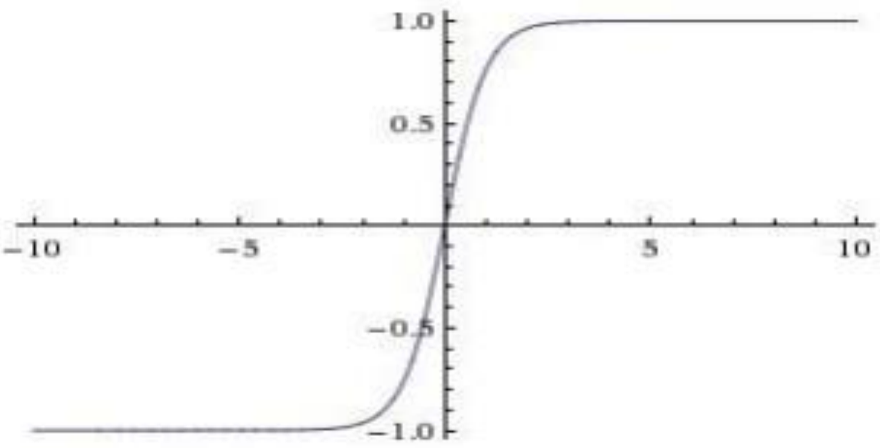
Activation functions - sigmoid



sigmoid activation function

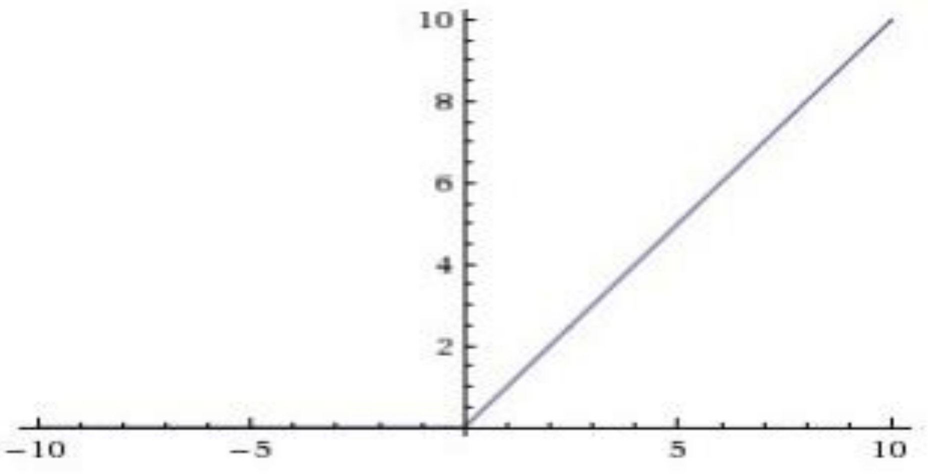
$$\frac{1}{1+e^{-x}}$$

Activation function - tanh



tanh(x)

Activation function - ReLu



ReLU

f(x) = max(0,x)

Non-linearity Constraint

Activation function is to introduce ***non-linearity into the network***

Without a *nonlinear* activation function in the network, NN, no matter how many layers it has, will behave like a linear system and we will not be able to mimic a ‘complicated’ function

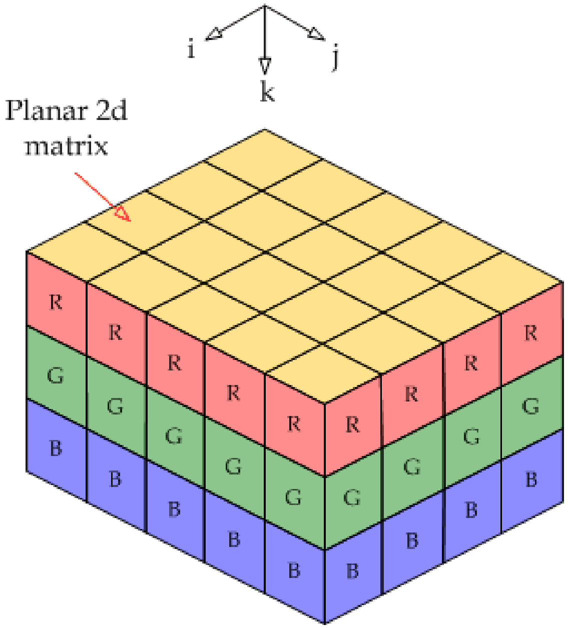
A neural network may very well contain neurons with linear activation functions, such as in the output layer, but these require the company of neurons with a nonlinear activation function in other parts of the network.

Convolution Step

An RGB image is represented by a 3 dimensional matrix

- The first channel holds the ‘R’ value of each pixel
- The second channel holds the ‘G’ value of each pixel
- The third channel holds the ‘B’ value of each pixel

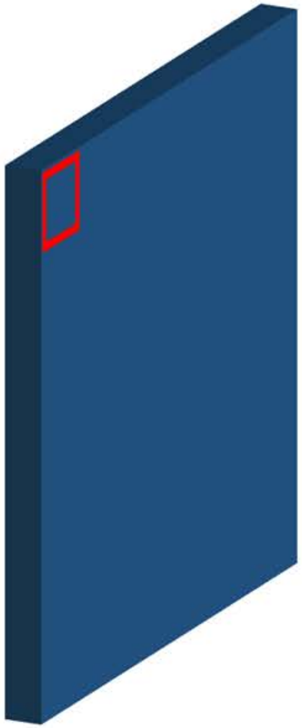
Eg: A 32x32 image is represented by a 32x32x3 matrix



Graphical presentation of RGB 3d matrix

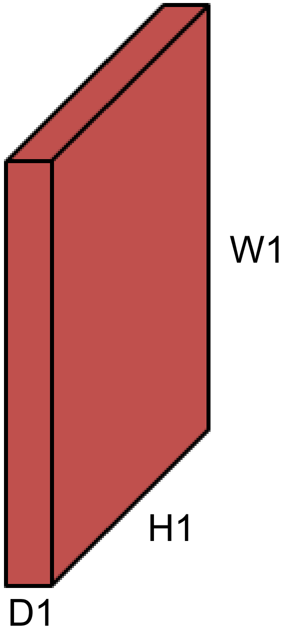
Filter 5x5x3

32x32x3

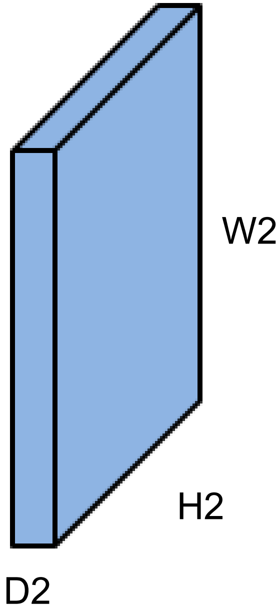


32x32x3

Input Volume vs Output Volume for convolution



Input

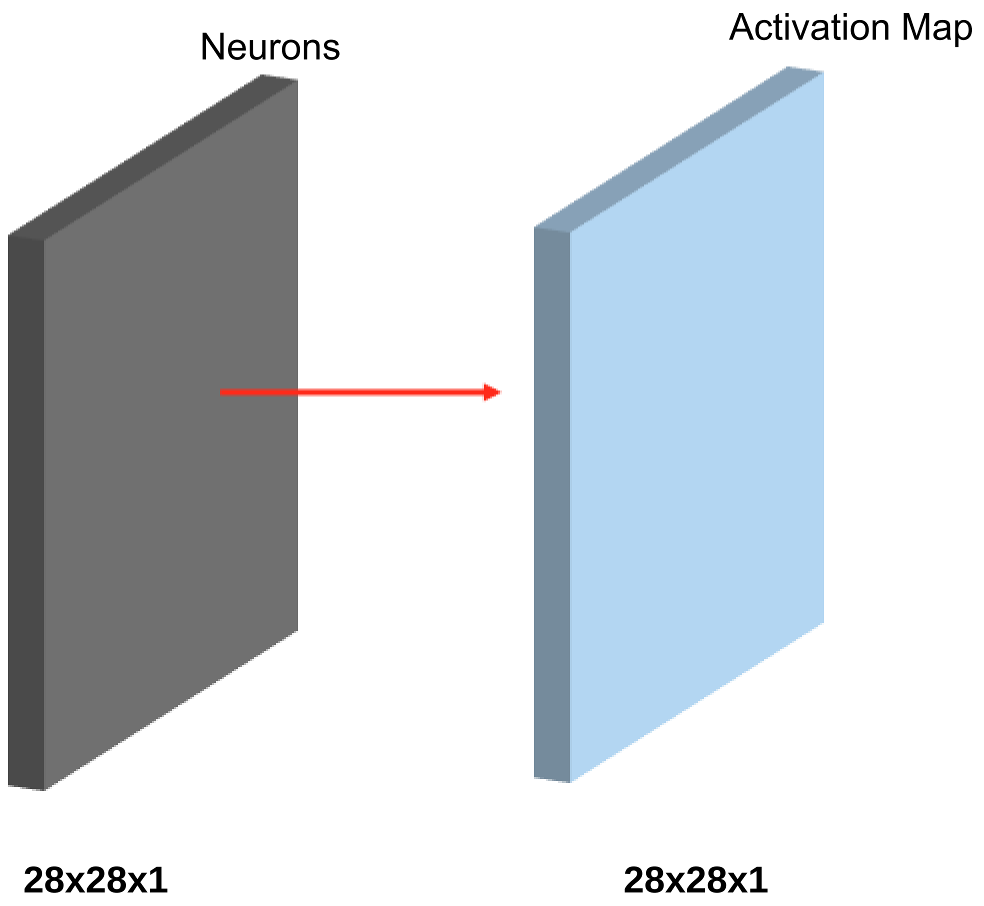


Output

W2 = W1 - (*filter width*) + 1

H2 = H1 - (*filter height*) + 1

D2 = *1 (D1 = filter depth)*


Neurons
Activation Map
28x28x1
28x28x1

32x32x3

(28x28x1) * 5

Parameters

Input volume: 32x32x3

Filter size : 5x5x3

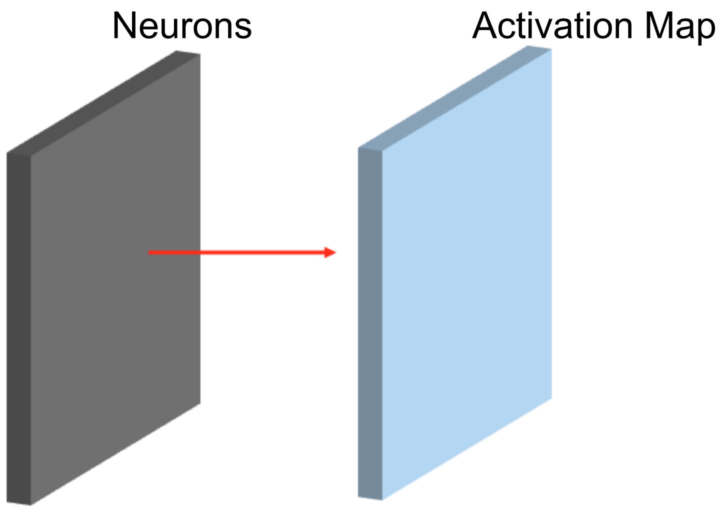
Size of 1 activation map: 28*28*1

Depth of first layer: 5

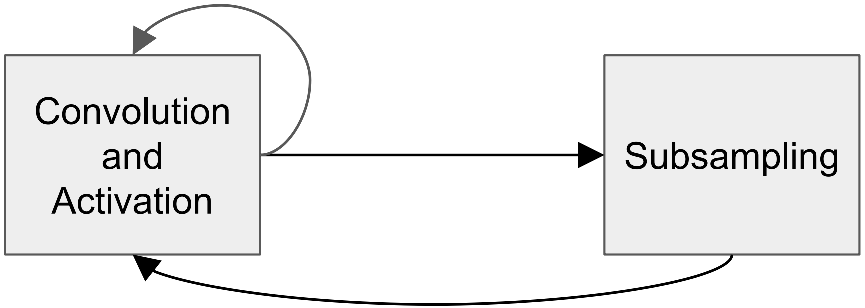
Total Number of neurons: 28*28*5 = 3920

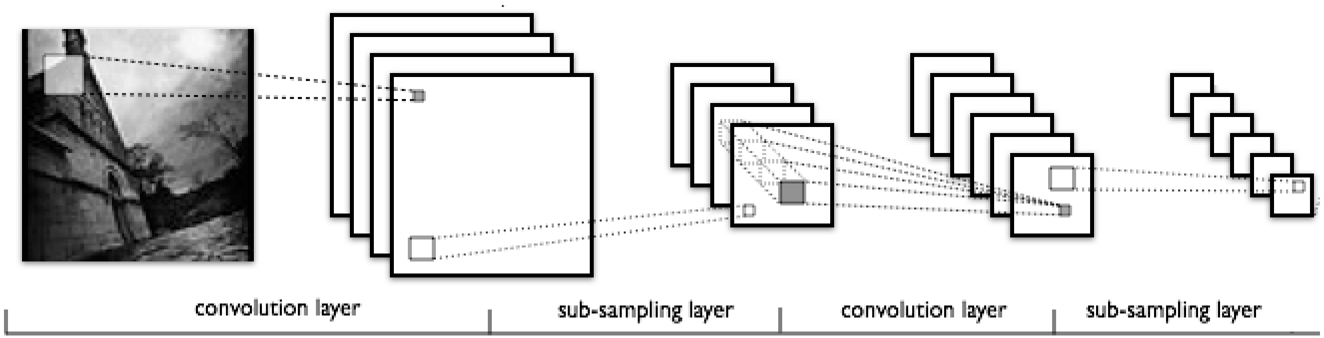
Weights per neuron: 5*5*3 = 75

Total Number of parameters: 75*3920 = 294 000



CNN overview





Subsampling

Objectives:

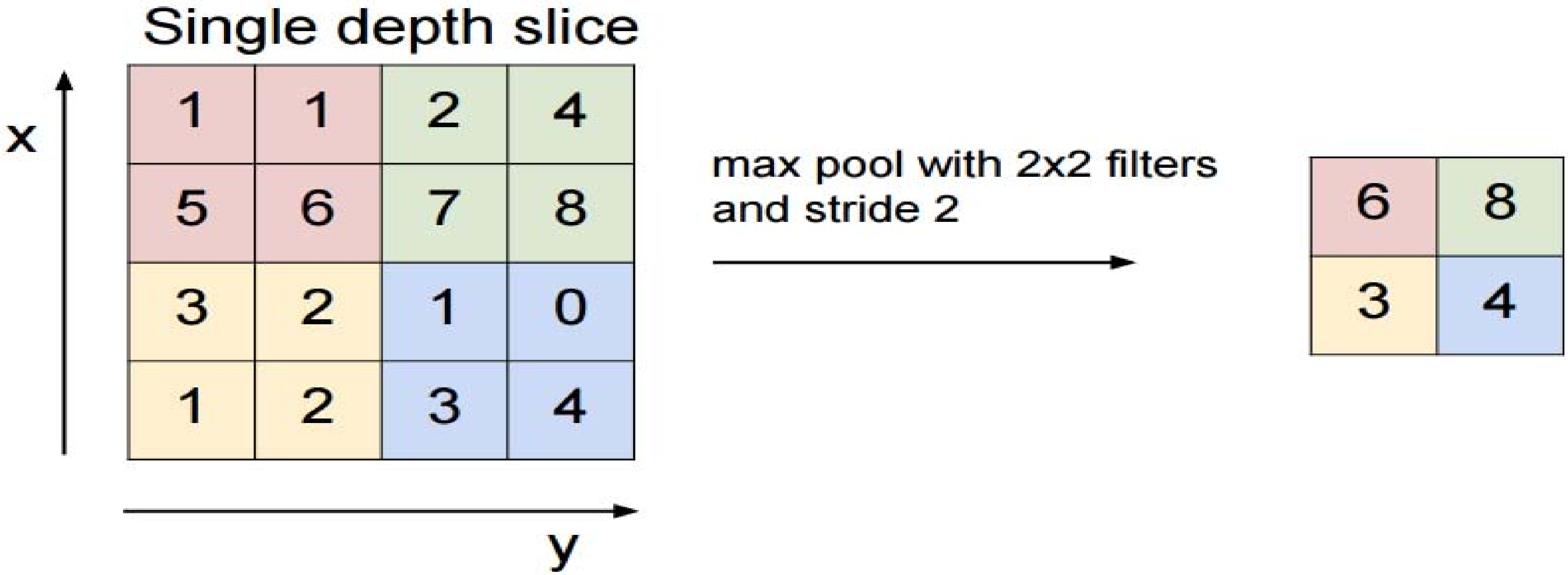
Reduce the size of input/feature space

Keep output of the most responsive neuron of the given interest region.

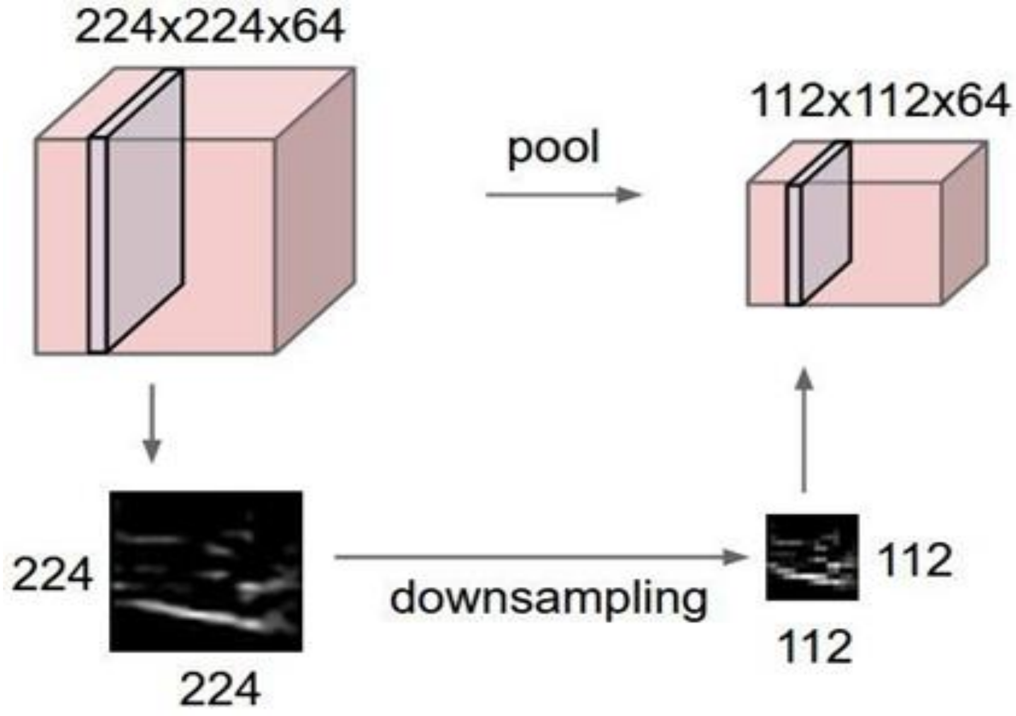
Common Methods:

- Max Pooling
- Average Pooling

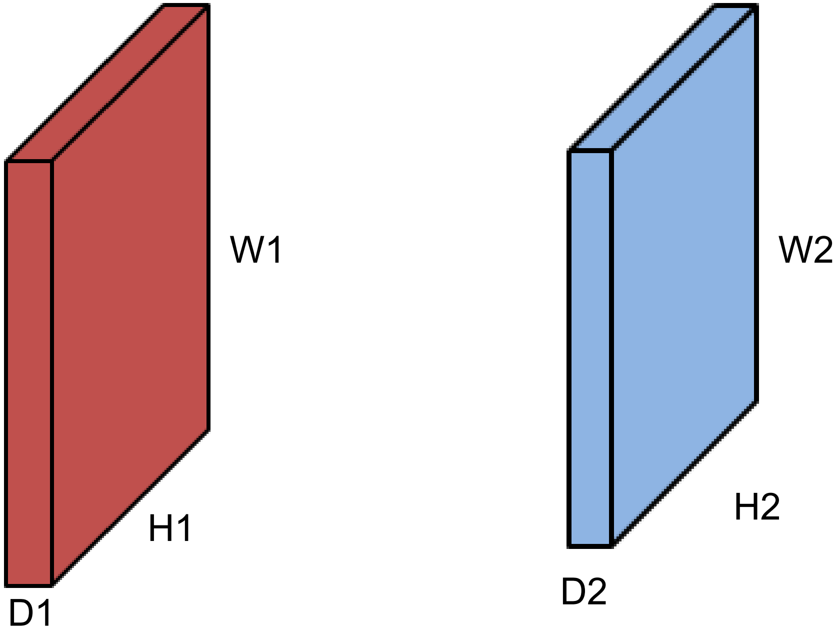
This involves splitting up the matrix of filter outputs into small non-overlapping grids and taking the maximum/average


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Max Pooling



Input Volume vs Output Volume for Max Pooling



Input

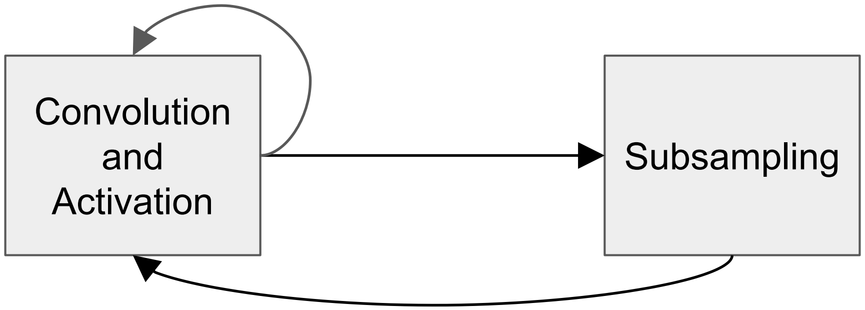
Output

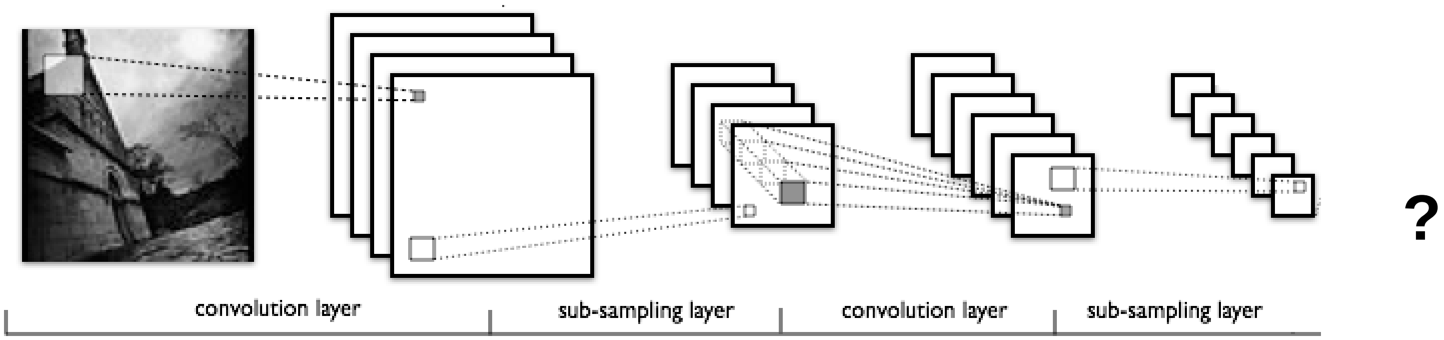
W2 = W1 - (pool width) + 1

H2 = H1 - (pool height) + 1

D2 = D1

CNN overview

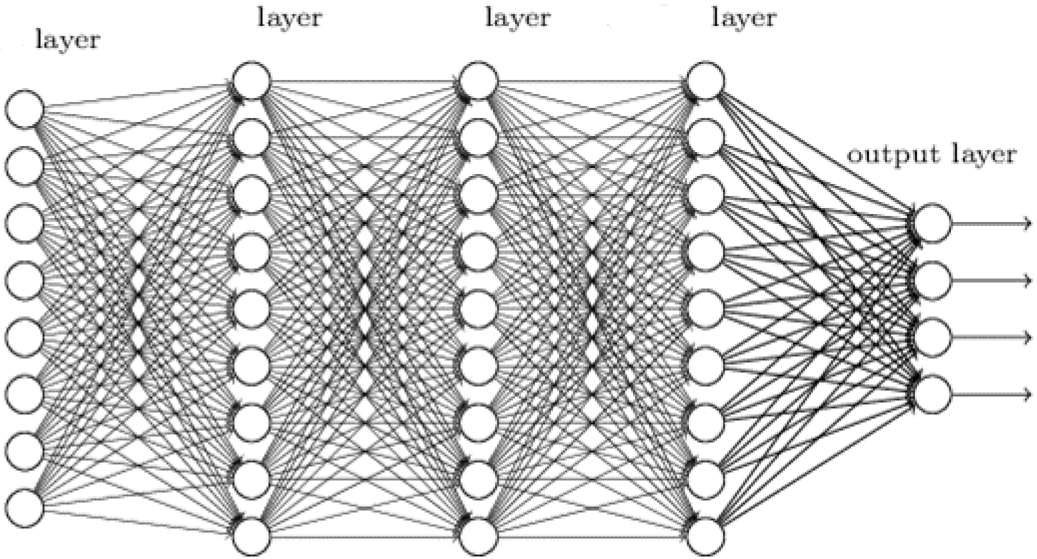




Fully Connected Layer

Neurons in fully connected layers have full connections to all activations in the previous layer

...



Softmax

Typically, output layer has one neuron corresponding to each label/class

The **softmax** function, or **normalized exponential**, "squashes" multi-dimensional vector of arbitrary real values to a multi-dimensional vector of values in the range (0, 1) that add up to 1.

$$P(y = j|\mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

Overview

- What is Deep Learning?
- Image Processing
- CNN Architecture
- Training Process
- Image Classification Results
- Limitations

Train the Network (setup the problem)

- The training is in fact to find a set of weights (for the filters) that minimize the cost functions, C(w,b).

- Normally, gradient descent algorithm is used to find the optimal

- Therefore, we need to find ∂C/∂wljk and ∂C/∂blj,and we update the weights and bias by:

$$w \rightarrow w - \eta \frac{\partial C}{\partial w}$$

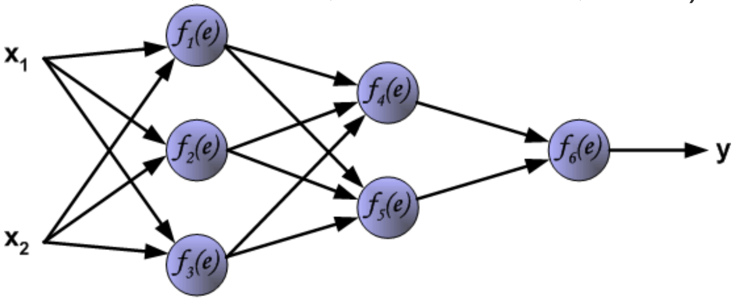
$$b \rightarrow b - \eta \frac{\partial C}{\partial b}$$

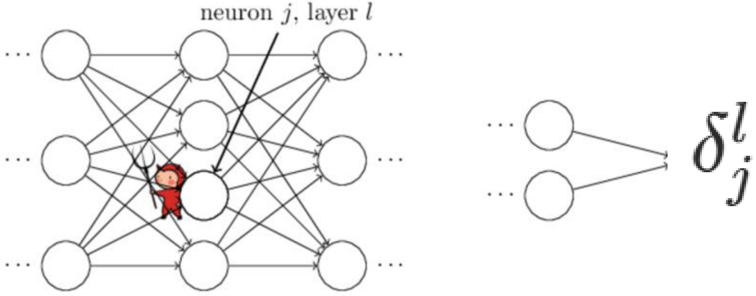
Train the Network (compute the gradient)

- Traditionally, for **one** training data, If using conventional method (central difference) and we have a **million** weights, the cost function, **C(w,b),** will need to be calculated a **million** times !!

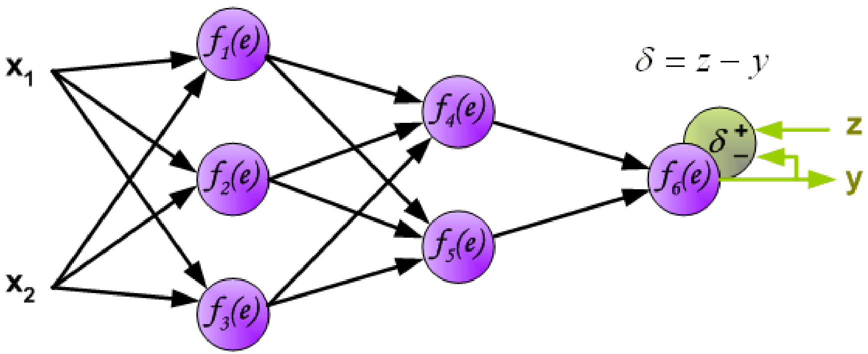
$$\frac{\partial C}{\partial w_j} \approx \frac{C(w + \epsilon e_j) - C(w)}{\epsilon}$$

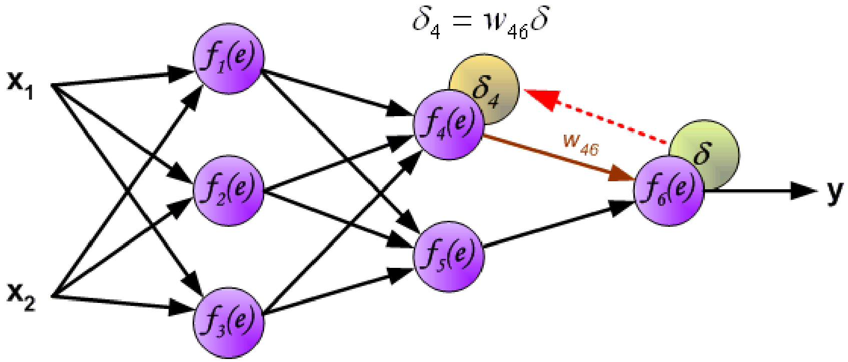
- How can we just calculate **C(w,b)** once? -- (Backpropagation Algorithm, Rumelhart, Hinton, and Williams, 1986).



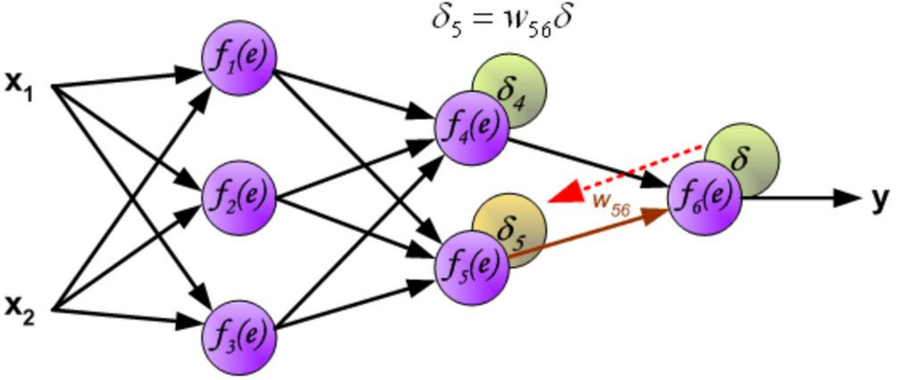


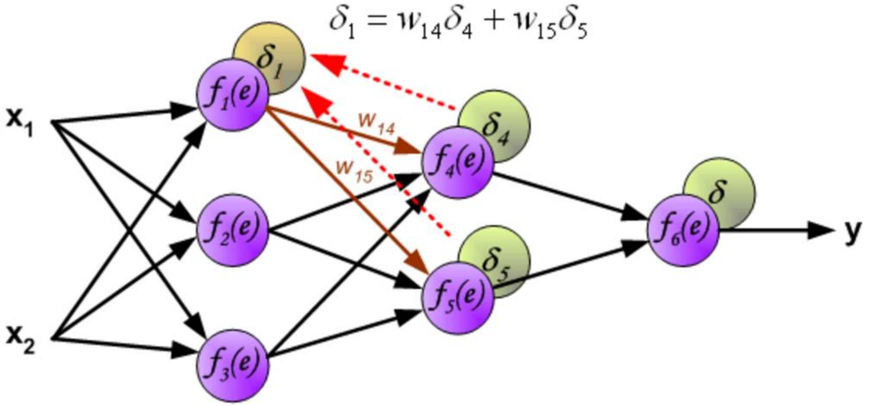
Backward Propagation of Errors





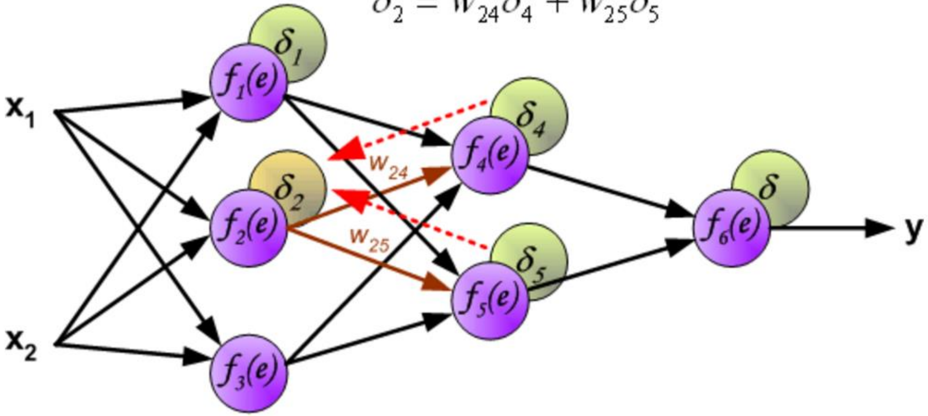
Backward Propagation of Errors



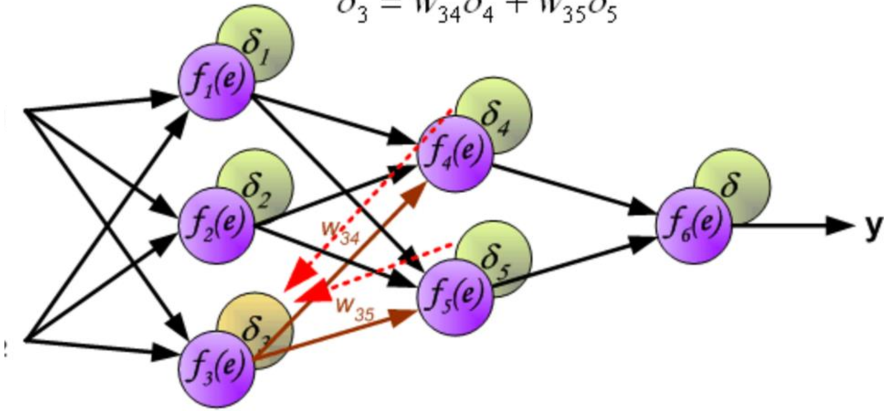


Backward Propagation of Errors

$$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$$



$$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$$



Backward Propagation of Errors (put it together)

- **Proof**: http://neuralnetworksanddeeplearning.com/chap2.html

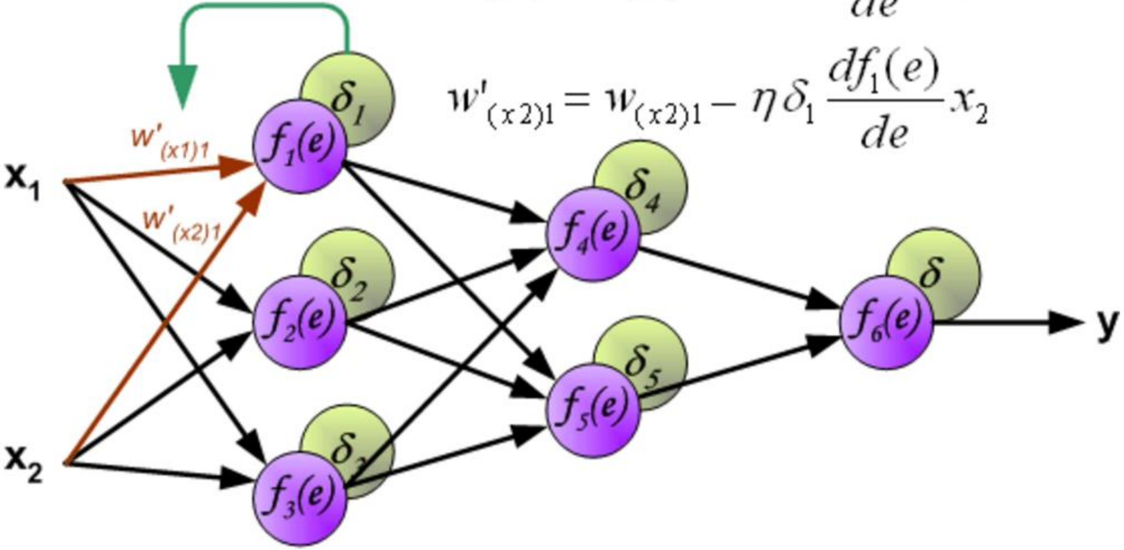
$$\frac{\partial C}{\partial w} = \begin{array}{c} \delta \times derivate\ of\ activation\ function \\ \times\ output\ from\ the\ neuron\ in\ the\ previous\ layer \end{array}$$

Backward Propagation of Errors (put it together)

- Tutorial: http://neuralnetworksanddeeplearning.com/chap2.html

$$w'_{(x1)1} = w_{(x1)1} - \eta \delta_1 \frac{df_1(e)}{de} x_1$$

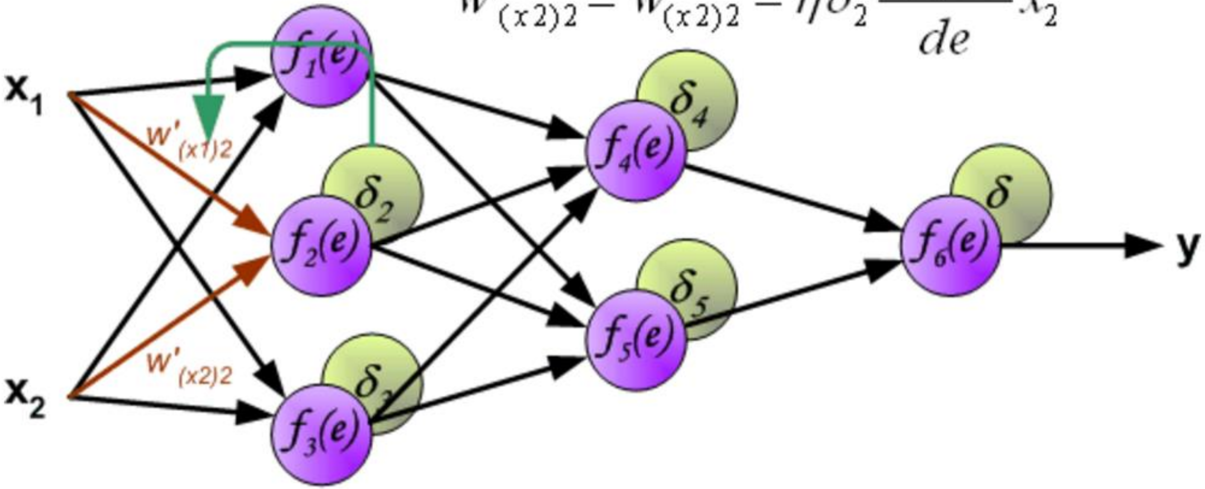
$$w'_{(x2)1} = w_{(x2)1} - \eta \delta_1 \frac{df_1(e)}{de} x_2$$



Backward Propagation of Errors (put it together)

$$w'_{(x1)2} = w_{(x1)2} - \eta \delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} - \eta \delta_2 \frac{df_2(e)}{de} x_2$$

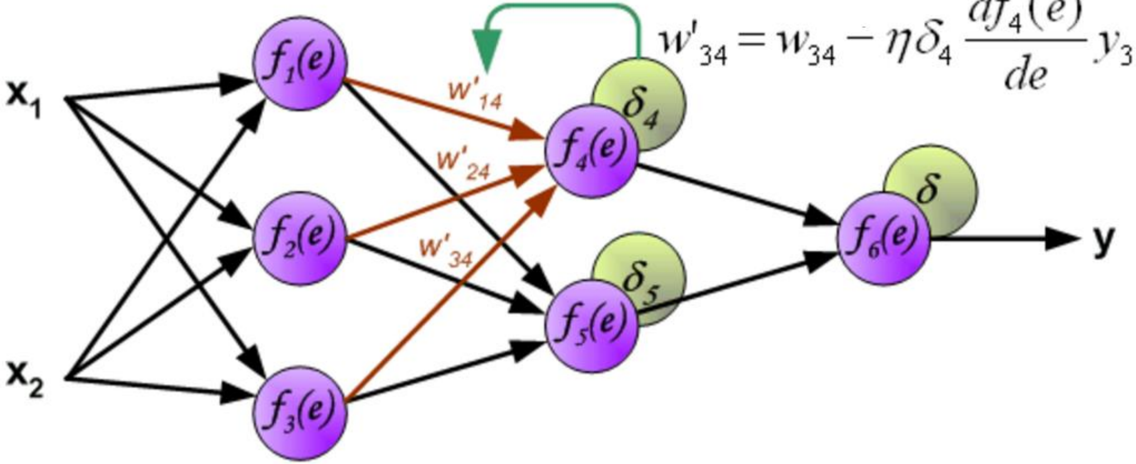


Backward Propagation of Errors (put it together)

$$w'_{14} = w_{14} - \eta \delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} - \eta \delta_4 \frac{df_4(e)}{de} y_2$$

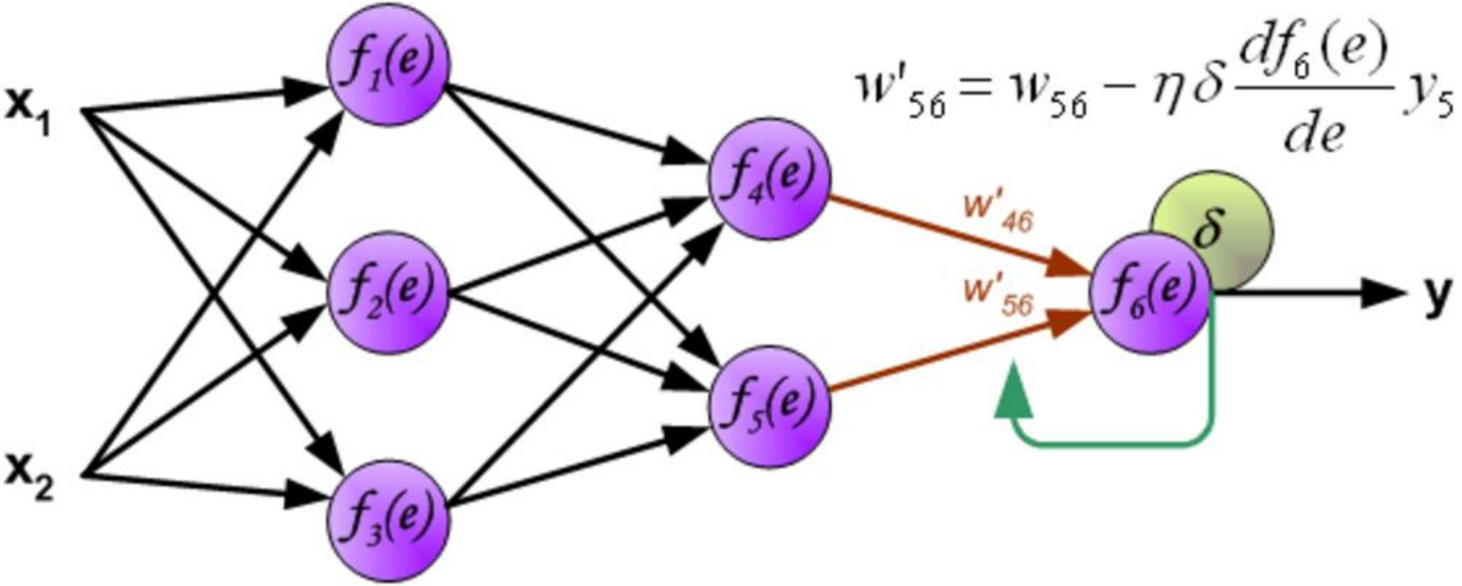
$$w'_{34} = w_{34} - \eta \delta_4 \frac{df_4(e)}{de} y_3$$



Backward Propagation of Errors (put it together)

$$w'_{46} = w_{46} - \eta\,\delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} - \eta\,\delta \frac{df_6(e)}{de} y_5$$

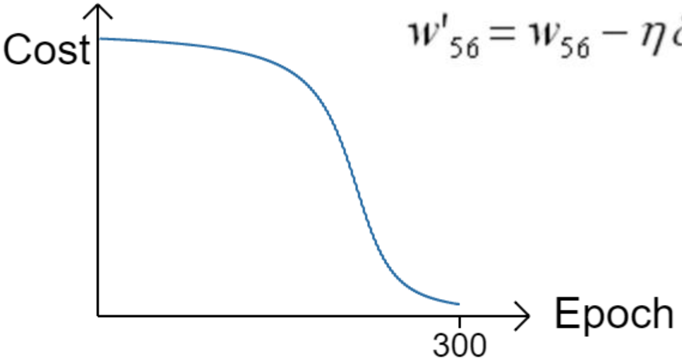


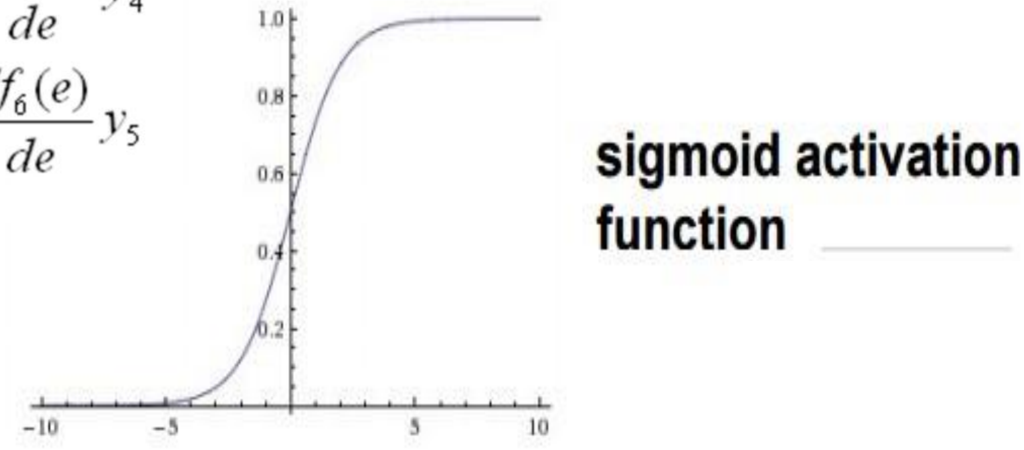
Train the network (Initializing Weights)

Initialization is need for the gradient descent algorithm and it is critical for the learning performance:

$$w'_{46} = w_{46} - \eta\delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} - \eta\delta \frac{df_6(e)}{de} y_5$$



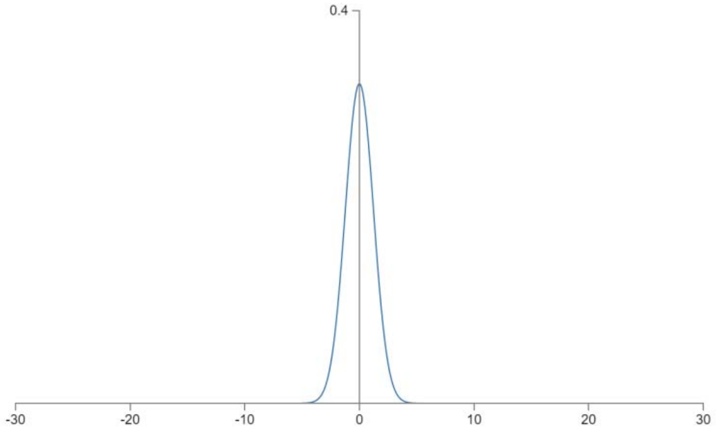


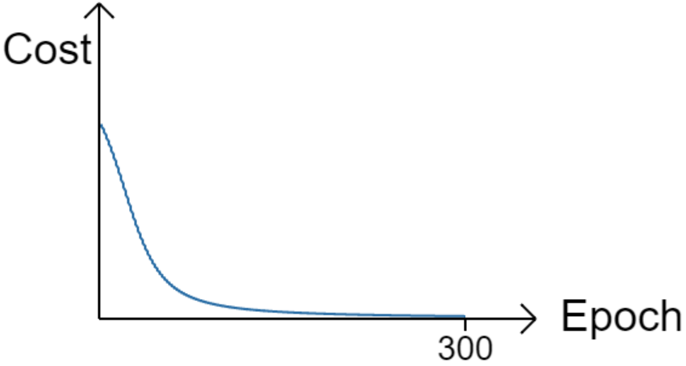
Initial Weights

We want to stay away from the saturation area.

Suppose there is n weights coming in one Neuron

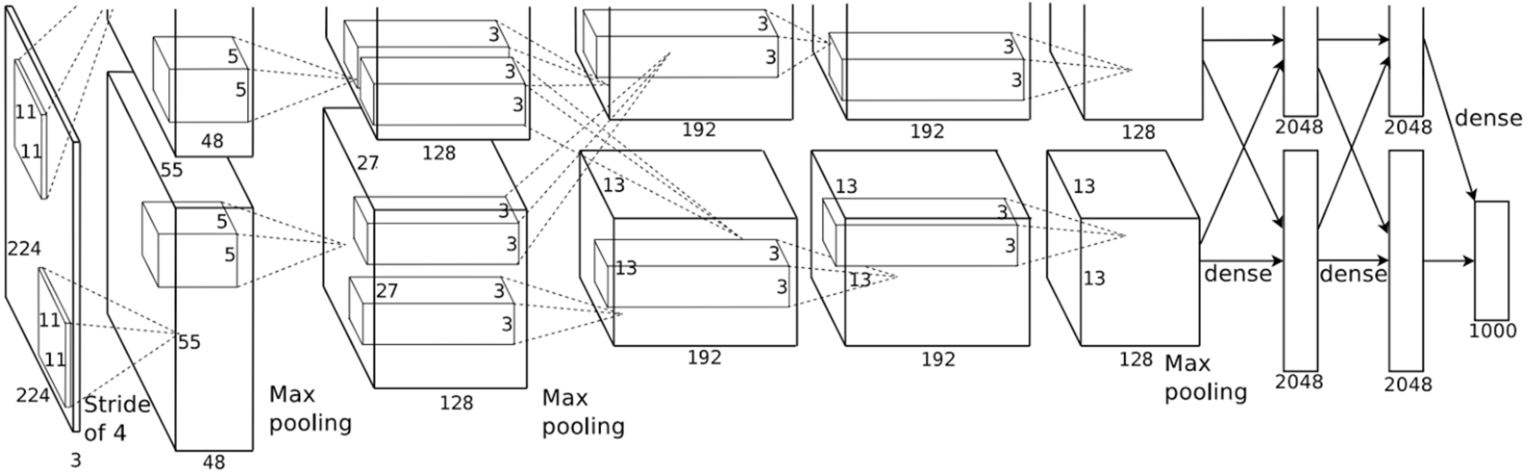
Best strategy is: Normal(0, $1/\sqrt{n_{\text{in}}}$)





Example architecture

Alex Net, 61 millions weights

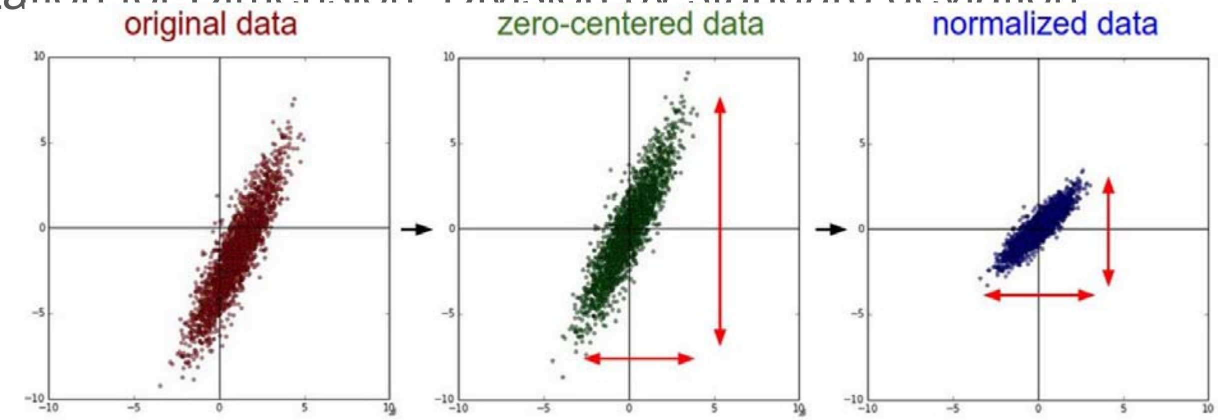


Preprocessing Tricks and Tips

Suppose we have dataset X = [N X D], where N is number of data points, and D is their dimensionality

1. Mean Image Subtraction: Subtraction of the mean across each individual feature in dataset

2. Normalization for Dimension: Division by standard deviation



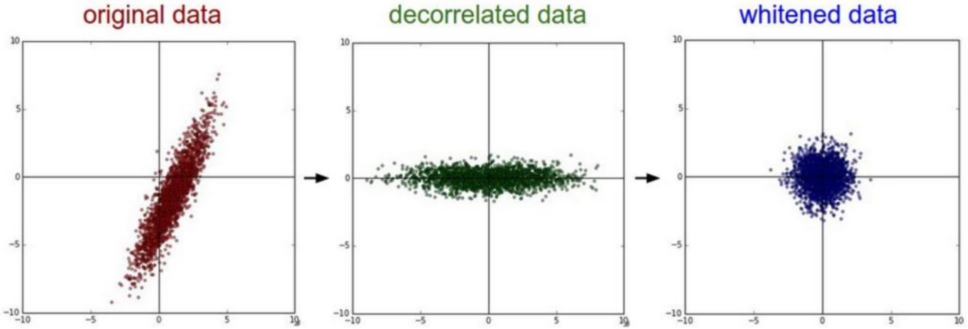
Preprocessing Tricks and Tips

3. Principle Component Analysis (PCA) for dimensionality reduction
   - Generate covariance matrix across the data
   - SVD factorization
   - Decorrelation, rotation into Eigenbasis
   - Choose a top-k eigenvalues: X' = [NxK]
4. Whitening
   - Divide by eigenvalues (square roots of singular v
   - Result: Zero mean, Identity Covariance

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)})(x^{(i)})^T.$$

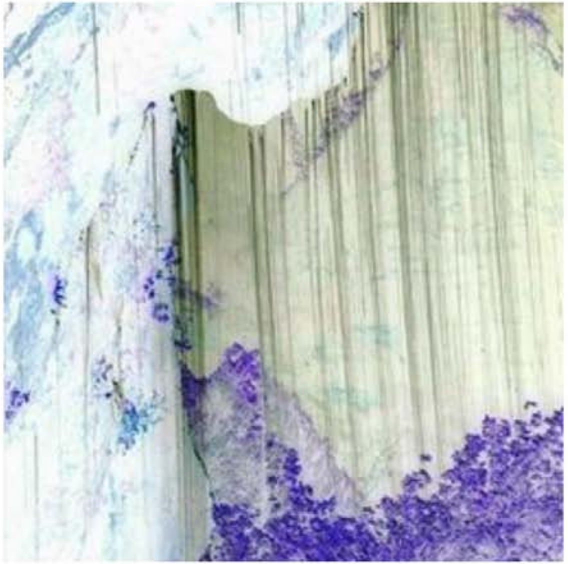
$$U = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_n \\ | & | & & | \end{bmatrix}$$

$$x_{\mathrm{rot}} = U^T x = \begin{bmatrix} u_1^T x \\ u_2^T x \end{bmatrix}$$



Data Augmentation

1. Rotations
2. Reflections
3. Scaling
4. Cropping
5. Color space remapping
6. *Randomization!*



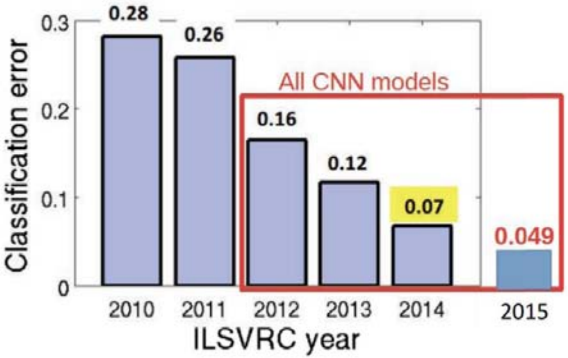
Overview

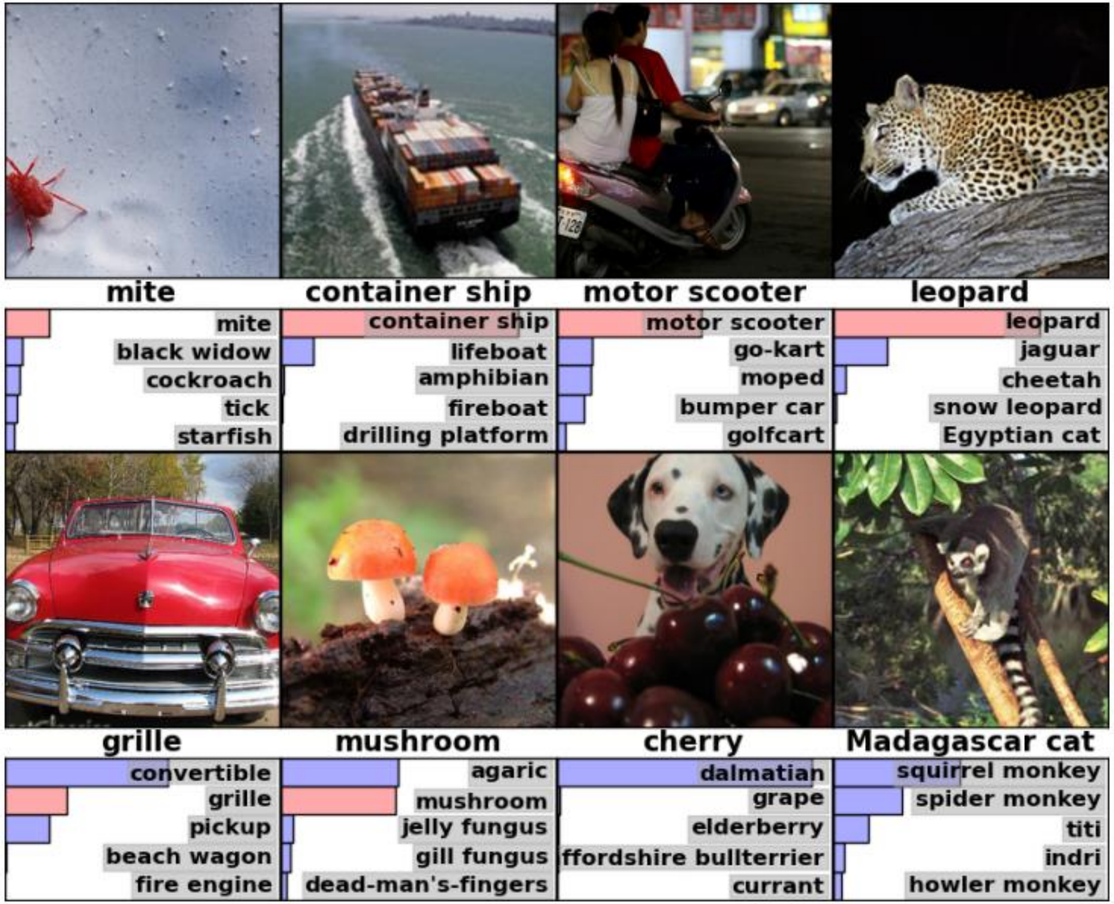
- What is Deep Learning?
- Image Processing
- CNN Architecture
- Training Process
- Image Classification Results
- Limitations

Revisiting the ImageNet Competition (ILSVRC 2010)

Model	**Top-1 error rate**	**Top-5 error rate**
Sparse coding	0.47	0.28
SIFT + FVs	0.46	0.26
CNNs	0.37	0.17

Top-5 error rate

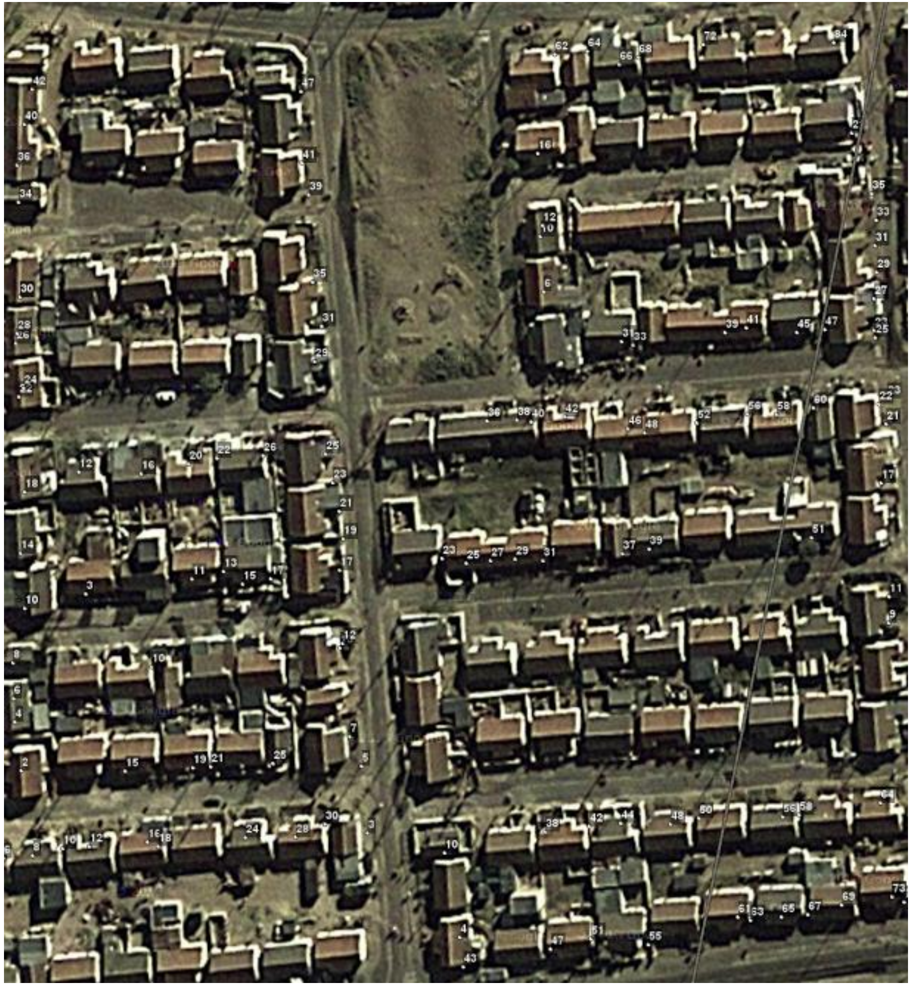




Krizhevsky, Alex et al. Imagenet classification with deep convolutional neural nets. NIPS 2012

Google Street View House Numbers

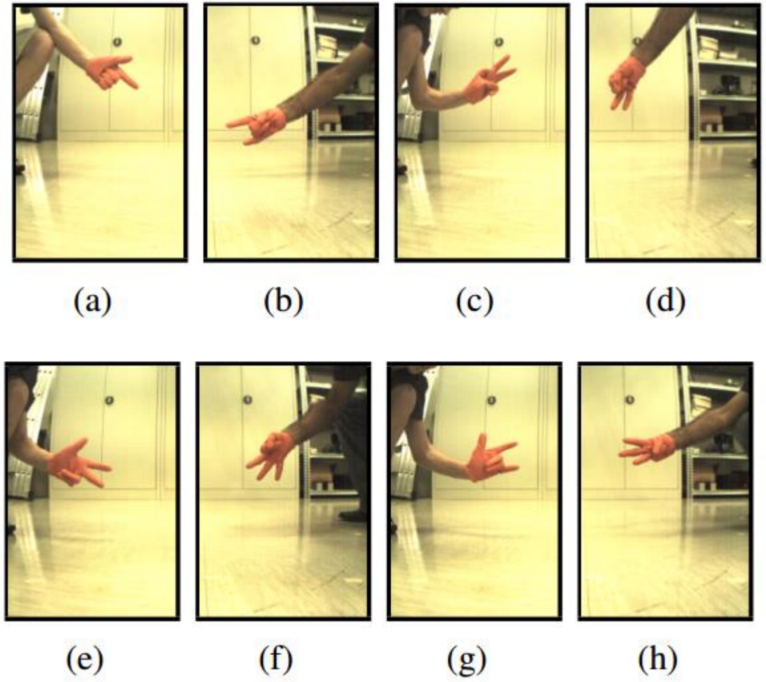
*"Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks" by Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet



Courtesy of Goodfellow, Ian et al. Used with permission.

“Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks” by Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz,   Sacha Arnoud, Vinay Shet

Recognizing Hand Gestures-HCI application

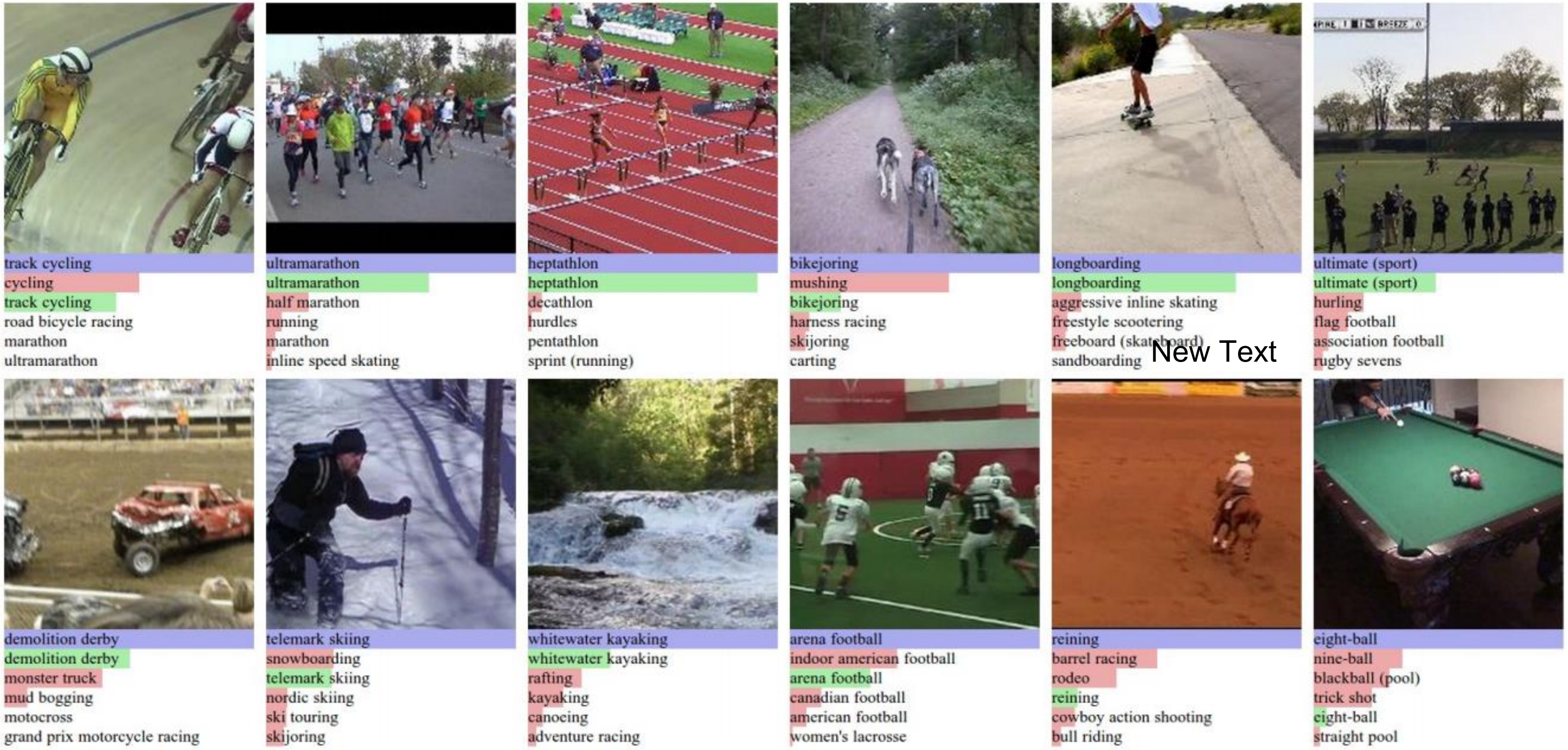


N. Jawad, D. Frederick, A. Gianni, C. Dan and M. Ueli, “Max-pooling convolutional neural networks for vision-based hand gesture recognition”, IEEE International Conference on Signal and Image Processing Applications, 2011.

Extended Image Classification: Video Classification

Extend image classification by adding temporal component to classify videos

- Note that this adds additional complexity, but the underlying system is the same: Convolutional Neural Nets



Karpathy, Andrej, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. "Large-Scale Video Classification with Convolutional Neural Networks." *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014)

Overview

- What is Deep Learning?
- Image Processing
- CNN Architecture
- Training Process
- Image Classification Results
- Limitations

Even the Best have Issues

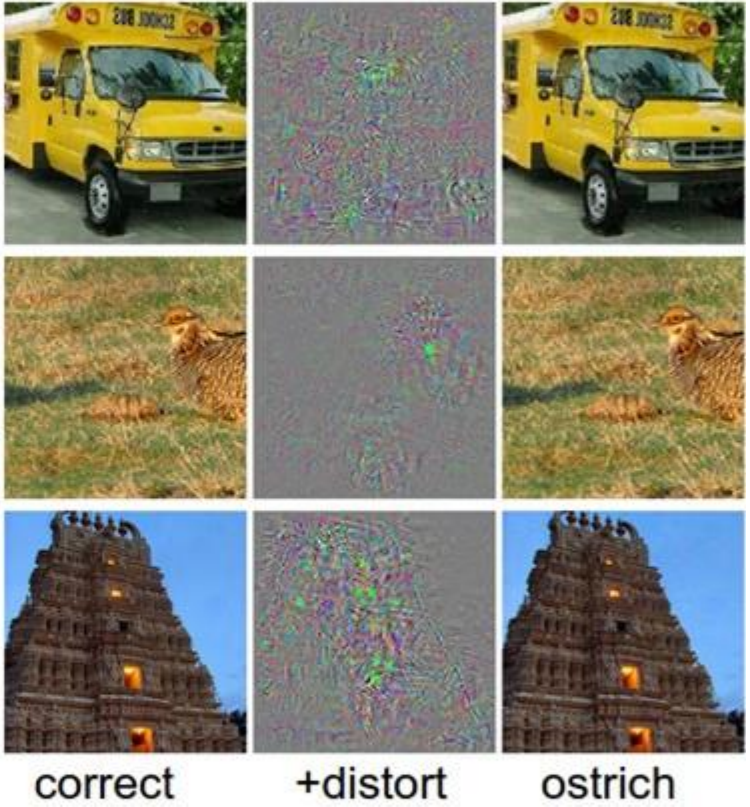
Microsoft won the most recent ImageNet competition and currently holds the state-of-the-art implementation

They can recognize 1000 categories of images, extremely reliably.

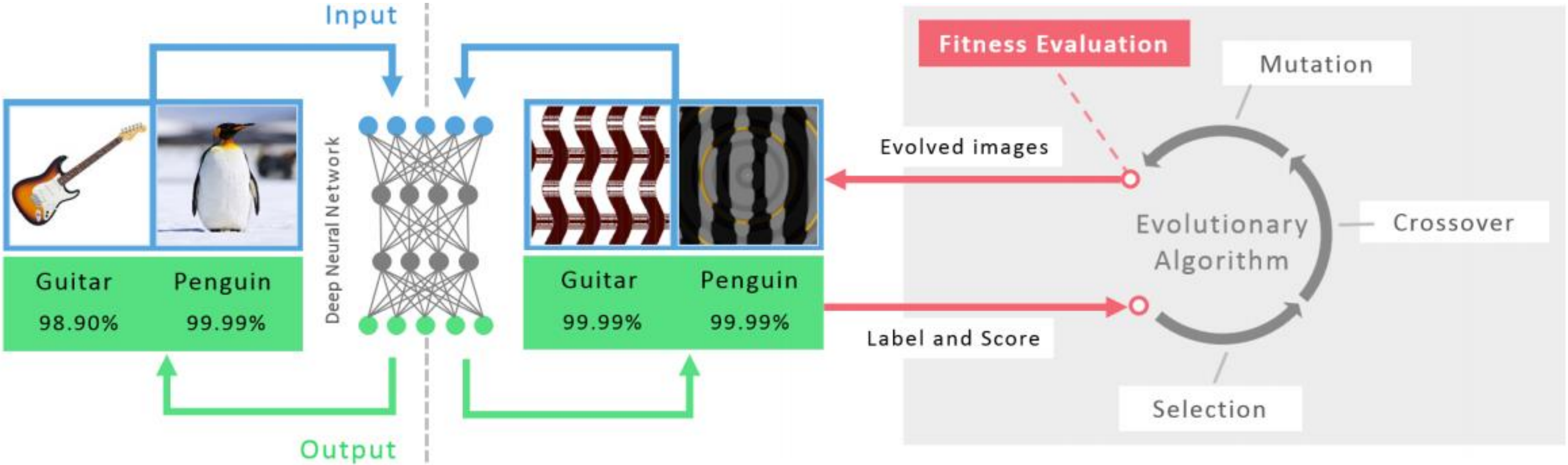
However:

- 1000 categories does not cover as many objects as you might expect.
- Uses 1.28 million images to train
- Takes weeks to train on multiple GPUs, with heavy optimization

http://arxiv.org/abs/1512.03385

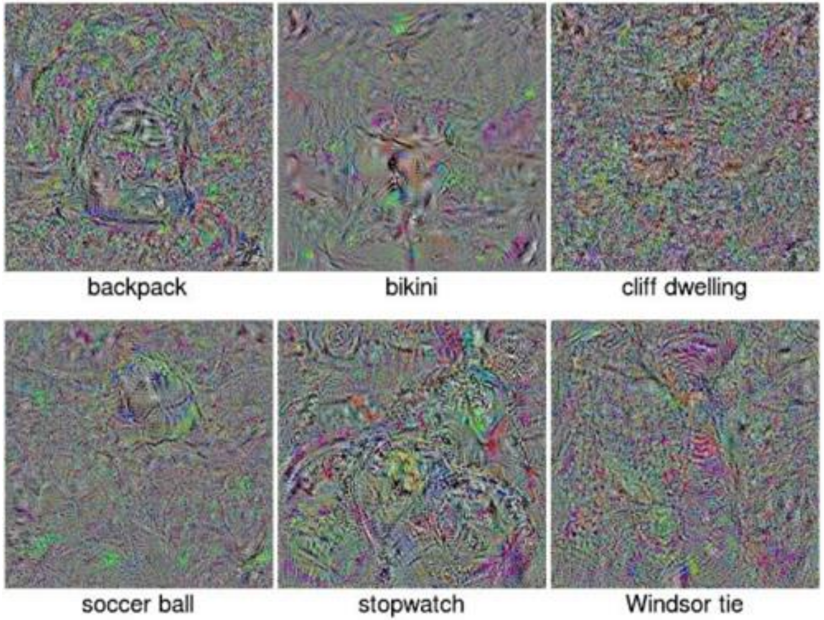


Szegedy et al. Intriguing Properties of Neural Networks. 2014.



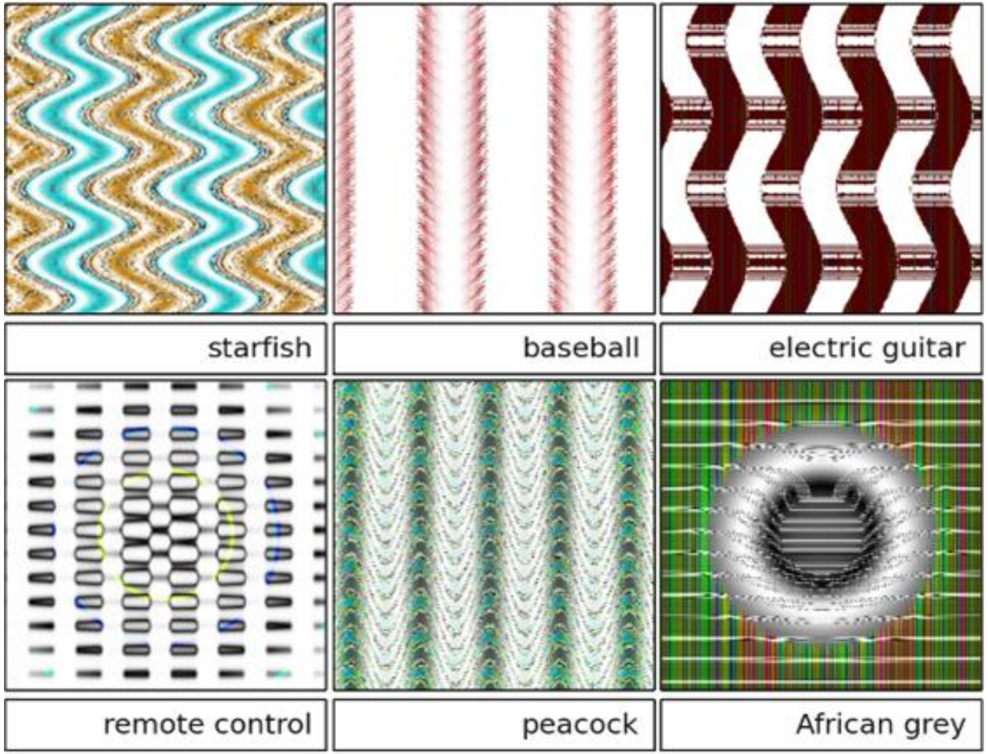
Nguyen A, Yosinski J, Clune J. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In Computer Vision and Pattern Recognition (CVPR ’15), IEEE, 2015.

Gradient Ascent

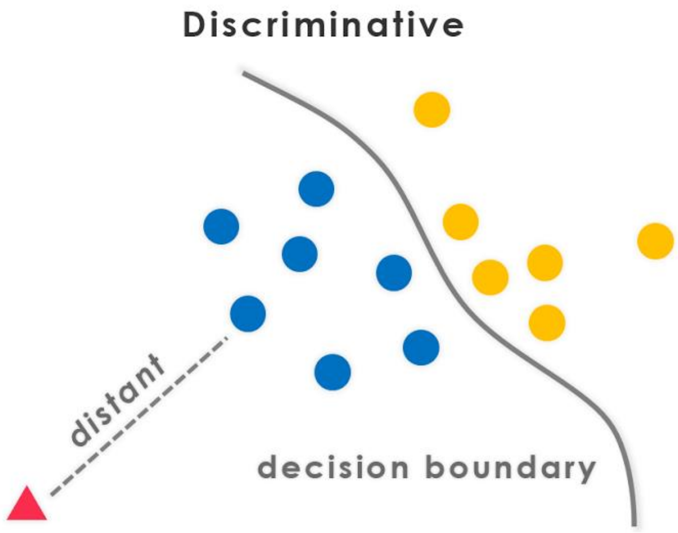


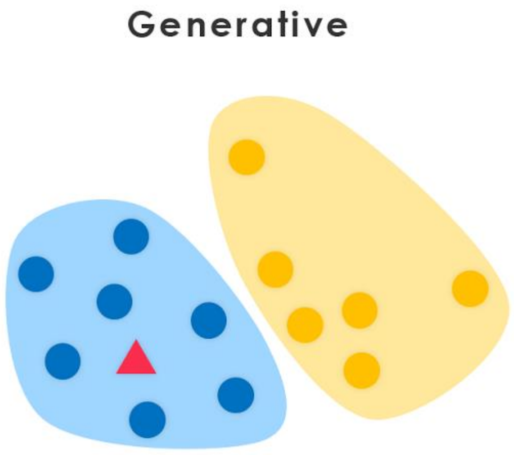
Nguyen A, Yosinski J, Clune J. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In Computer Vision and Pattern Recognition (CVPR ’15), IEEE, 2015.

Indirect Encoding



Nguyen A, Yosinski J, Clune J. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In Computer Vision and Pattern Recognition (CVPR ’15), IEEE, 2015.


Discriminative
distant
decision boundary


Generative


DW C
LD LMT 217400
LT WT 79600
60K


The colours, shapes, smells, patterns and sensations which formed the root of instincts.
Tinbergen succeeded in isolating the traits which triggered certain instincts...
...and then made an interesting discovery.


The instincts had no bounds.
Instead of stopping at a 'sweet spot', the instinctive response would still be produced by unrealistic stimuli.
Once the researchers isolated the instincts' trigger...
...they could create greatly exaggerated dummies which the animals would choose instead of a realistic alternative.
ACME

Takeaways

- Deep Learning is a powerful tool that relies on many iterations of processing
- CNNs outperform all other algorithms for image classification because of the image processing power of convolutional filters
- Backpropagation is used to efficiently train CNNs
- CNNs need tons of data and processing power

Getting Started With Deep Learning

TensorFlow

References

ImageNet Classification with Deep Convolutional Neural Networks. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

Approximation by Superpositions of a Sigmoidal Function. G. Cybenko https://www.dartmouth.edu/~gvc/Cybenko_MCSS.pdf

Backpropagation Tutorial http://neuralnetworksanddeeplearning.com/chap2.html

https://papers.nips.cc/paper/877-the-softmax-nonlinearity-derivation-using-statistical-mechanics-and-useful-properties-as-a-multiterminal-analog-circuit-element.pdf

Nguyen A, Yosinski J, Clune J. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In Computer Vision and Pattern Recognition (CVPR ’15), IEEE, 2015.

“Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks” by Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet

Deep Residual Learning for Image Recognition. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. http://arxiv.org/abs/1512.03385

Appendix

Backward Propagation of Errors

- The gradient of weights and bias can be found by back chaining the auxiliary variable, defined as:

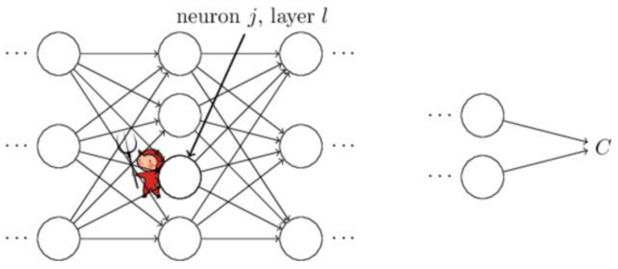
$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l} \qquad z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \qquad a^l = \sigma(z^l)$$

- By chain rule:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L) \qquad \delta^L = \nabla_a C \odot \sigma'(z^L)$$

- The back propagate it (chain rule again):

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$



Backward Propagation of Errors (put it together)

Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

Train the Network (put it together)

1. **Input a set of training examples**

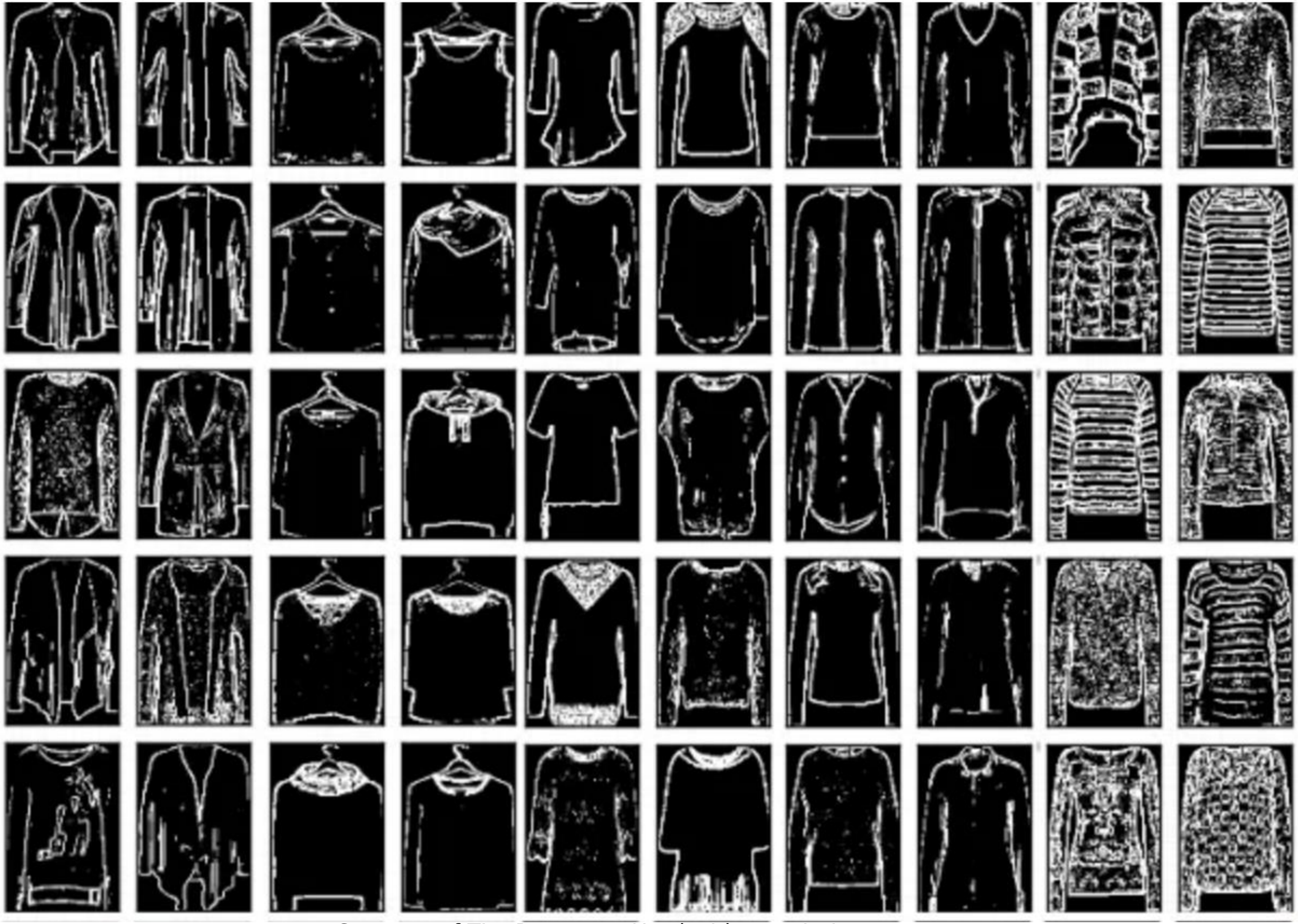
2. **For each training example x:** Set the corresponding input activation $a^{x,1}$, and perform the following steps:

   - **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^{x,l} = w^l a^{x,l-1} + b^l$ and $a^{x,l} = \sigma(z^{x,l})$.

   - **Output error $\delta^{x,L}$:** Compute the vector $\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L})$.

   - **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l})$.

3. **Gradient descent:** For each $l = L, L-1, \ldots, 2$ update the weights according to the rule $w^l \rightarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$, and the biases according to the rule $b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$.



Courtesy of Tim Dettmers. Used with permission.

Convolution: Filters

An output pixel's value is some function of the corresponding input pixel's neighbors

Examples:

- Smooth, sharpen, contrast, shift
- Enhance edges
- Detect particular orientations

App… Convolution

1/9

1	1	1
1	1	1
1	1	1

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	90	0
0	0	0	90	90	90	90	90	90	0
0	0	0	90	90	90	90	90	90	0
0	0	0	90	90	90	90	90	90	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

= 40

Convolution for 2D matrices

Given two three-by-three matrices, one a kernel, and the other an image piece, convolution is the process of multiplying entries and summing

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = (1*i)+(2*h)+(3*g)+(4*f)+(5*e)+(6*d)+(7*c)+(8*b)+(9*a)$$

The output of this operation constitutes the input to a single neuron in the following layer.