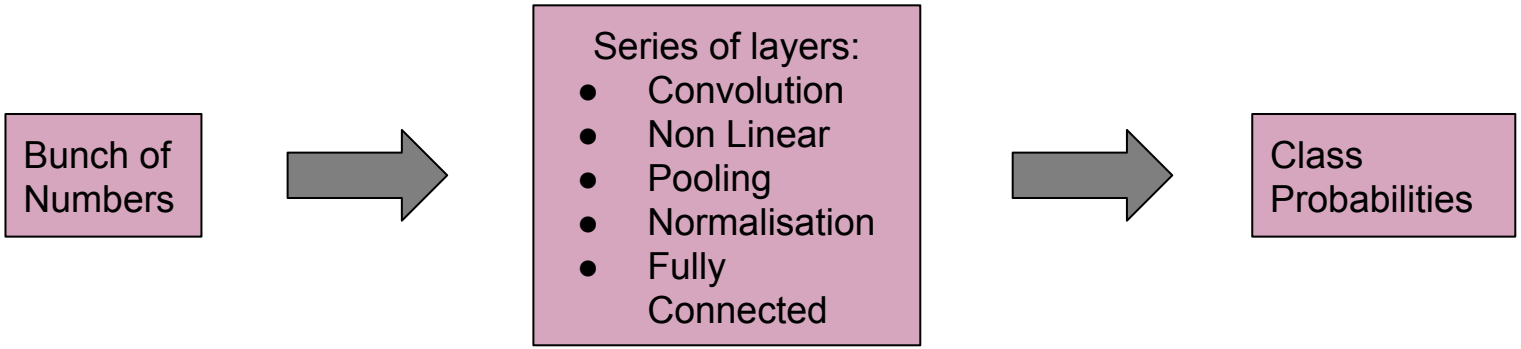
Architecture

Basics of CNN

CNNs for Classification Task



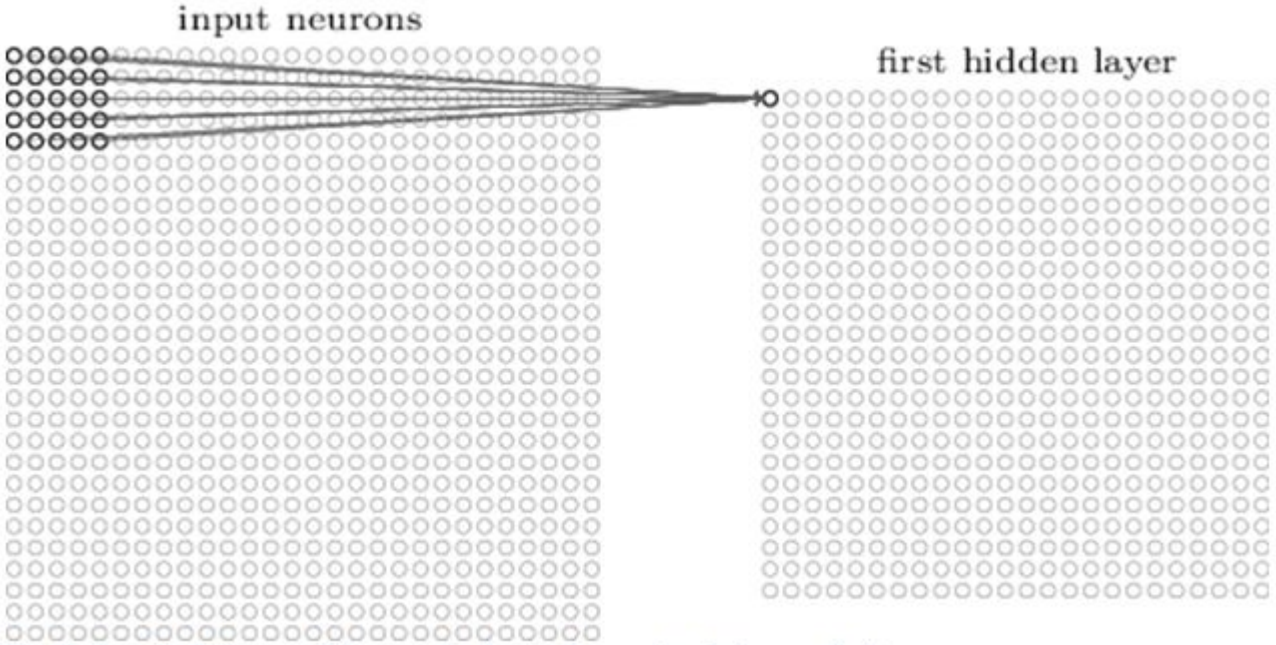
Biological Connection

CNNs are inspired from visual cortex in brain. The cortex has small regions of cells that are sensitive to specific characteristics of the visual field.

CNNs adopt a similar idea, of having a system with multiple components, each component having some specific task.

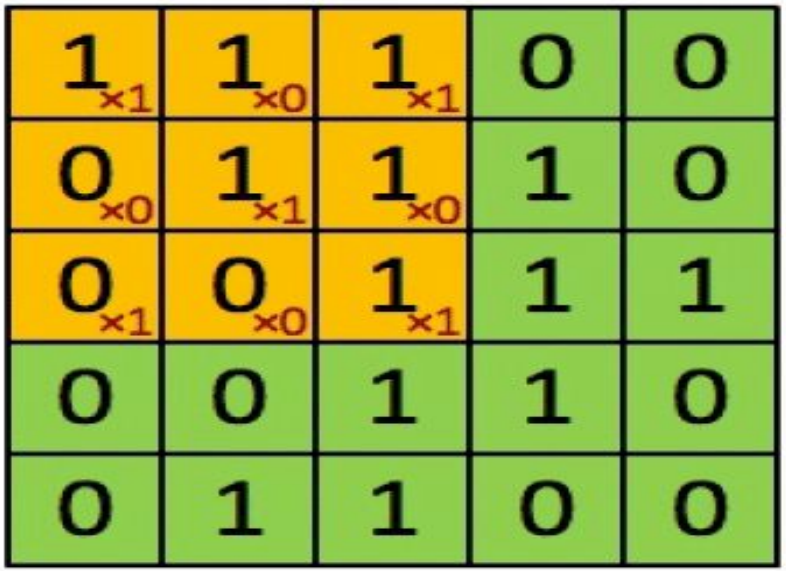
Convolution Layer

A number of ***filters*** or ***kernels*** which are responsible for ***convolution*** operation.

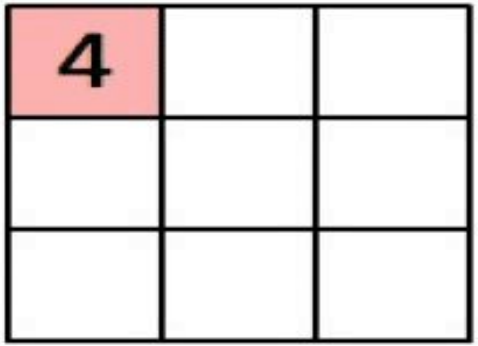


Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

What is Convolution Operation?



Image



Convolved Feature

And what is a Kernel?

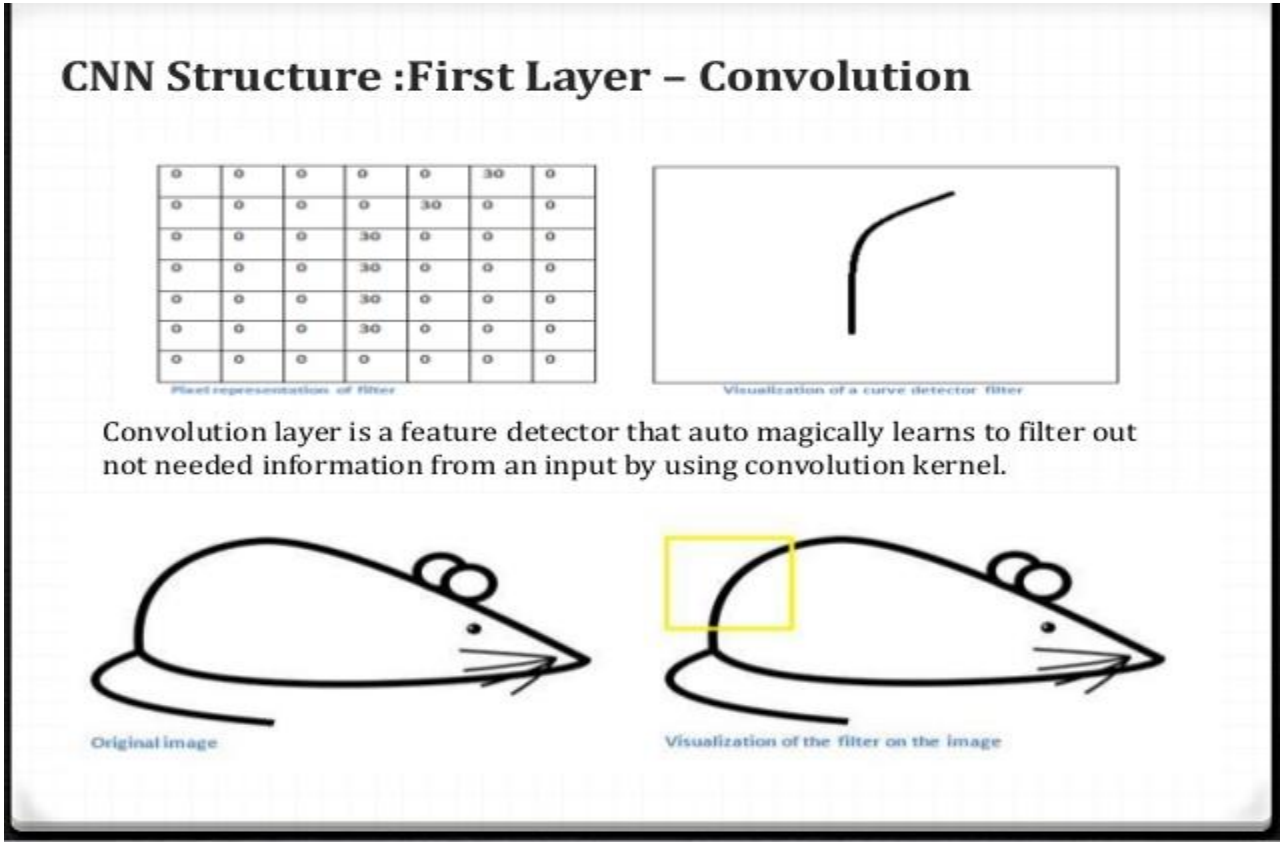
Just another bunch of numbers.

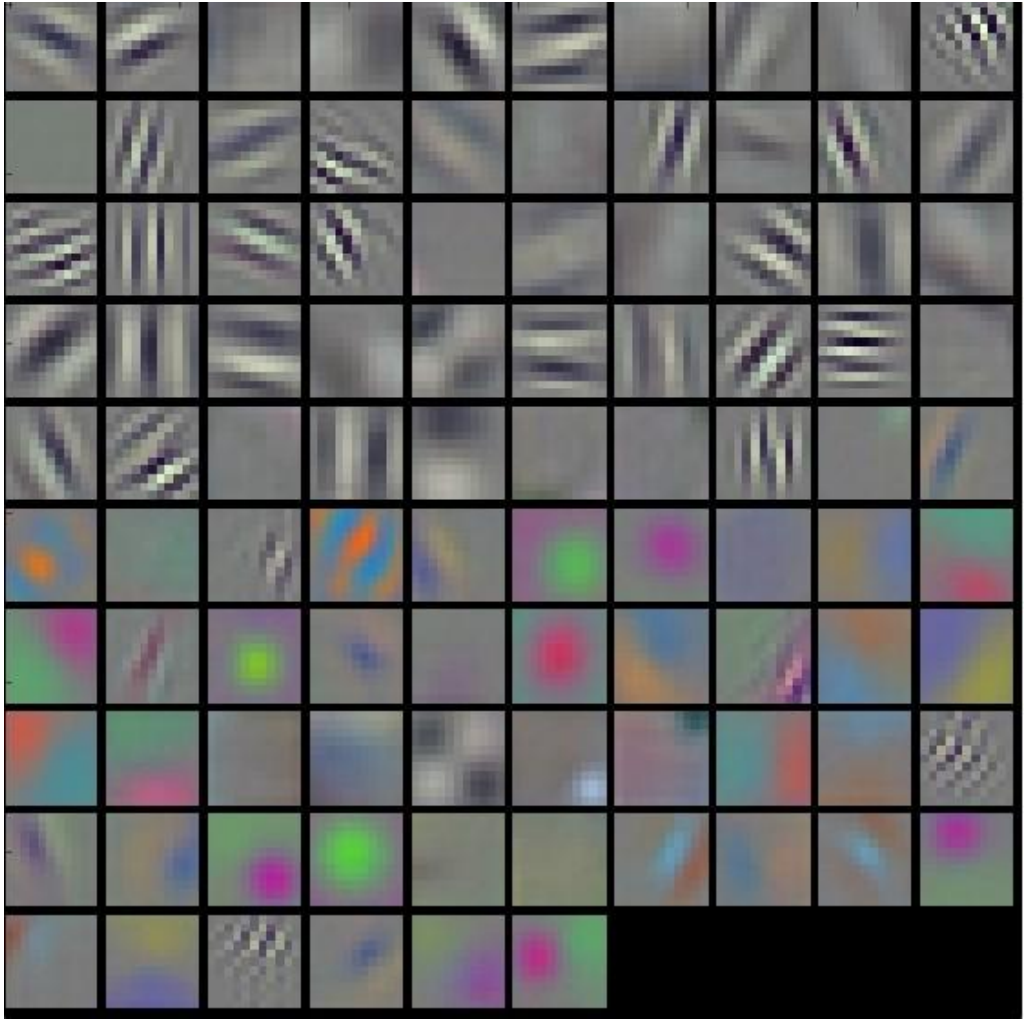
And why is it special?

Because...

Because..
KERNELS get trained to act as feature identifiers.
They are similar to those group of cells in visual cortex which perform a specific task.
They extract specific features from the image and build complex features from simpler ones.

What does that even mean?





Activation Layers

All a Neural Network does is a series of linear algebra computations.

Performing only linear transformations is **NOT COOL**.

Activation functions help neural networks to make sense out of complex, non-linear mappings between input and target variable.

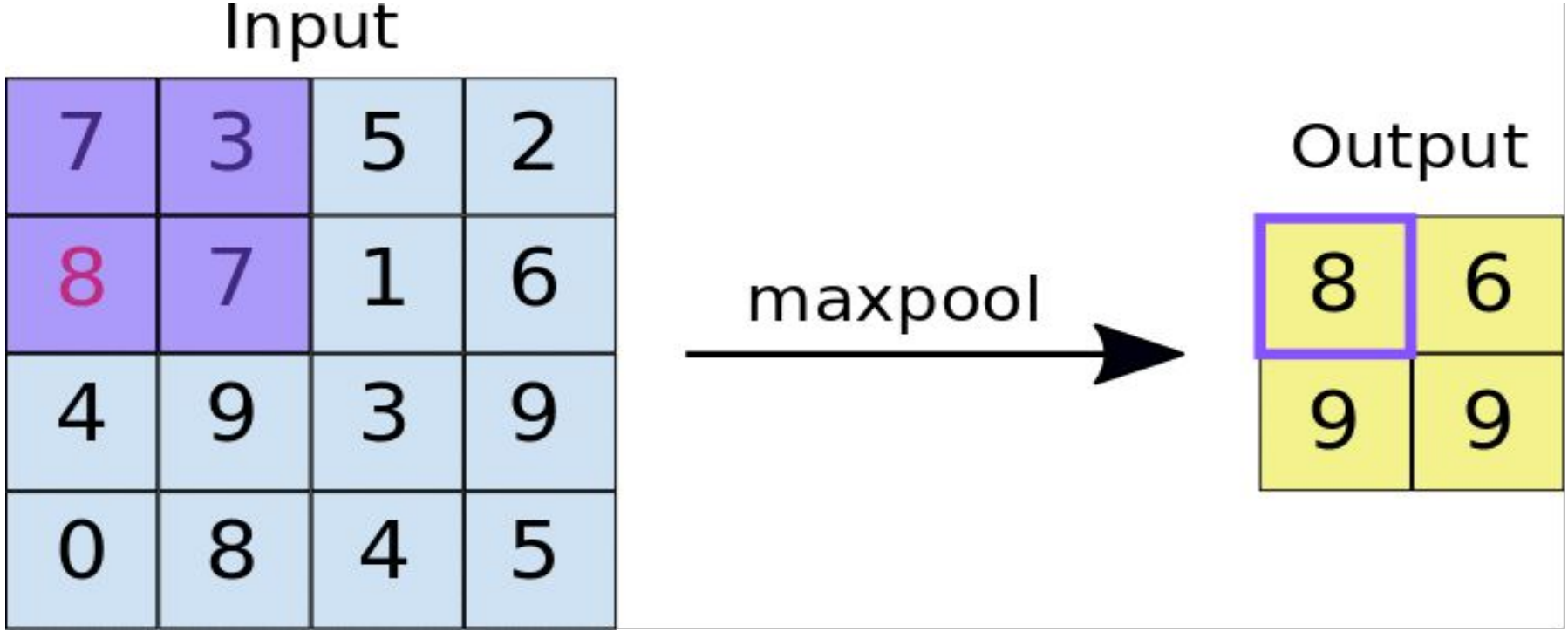
Note: *Activation functions should be differentiable for backpropagation to work.*

Fully Connected Layers

These takes in the feature maps extracted by the convolution layers and correlate those features to a particular output class.

Yes, that’s pretty much all about them.

Pooling Layers



Normalisation Layers

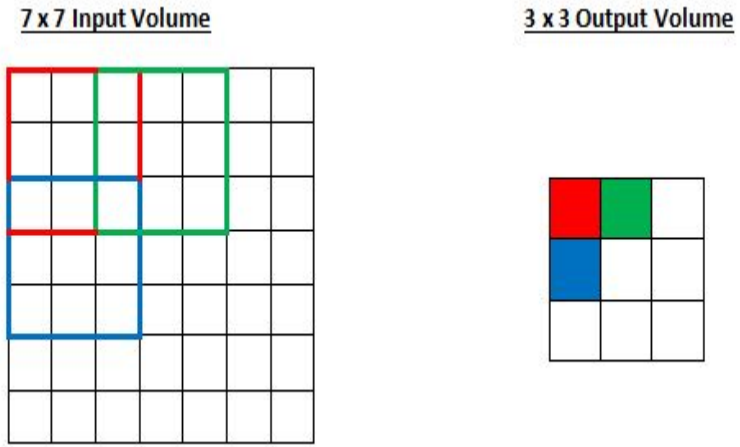
Normalisation scales and shifts the numbers coming out of a layer.

Reduces the disparity in the values in a neighbouring region

An important term- *stride*

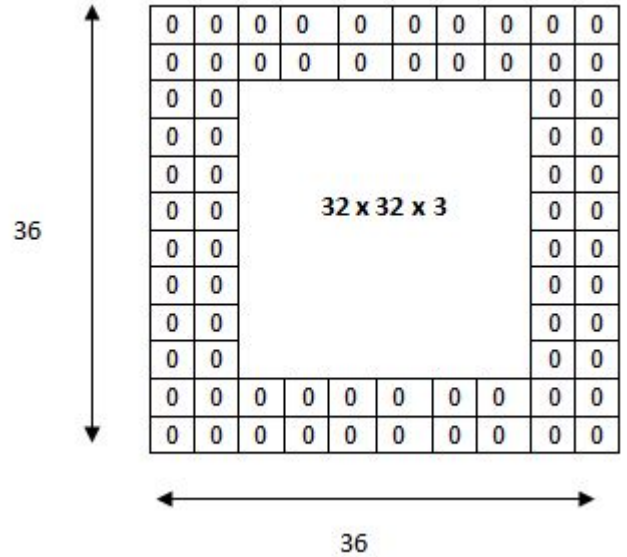
Stride is the number of pixels kernels slides.

It is normally set to a value such that output dimension is an integer.



Another important term- *padding*

Padding is addition of extra rows/columns image or feature map.



Dimensions of Input and Output of a layer

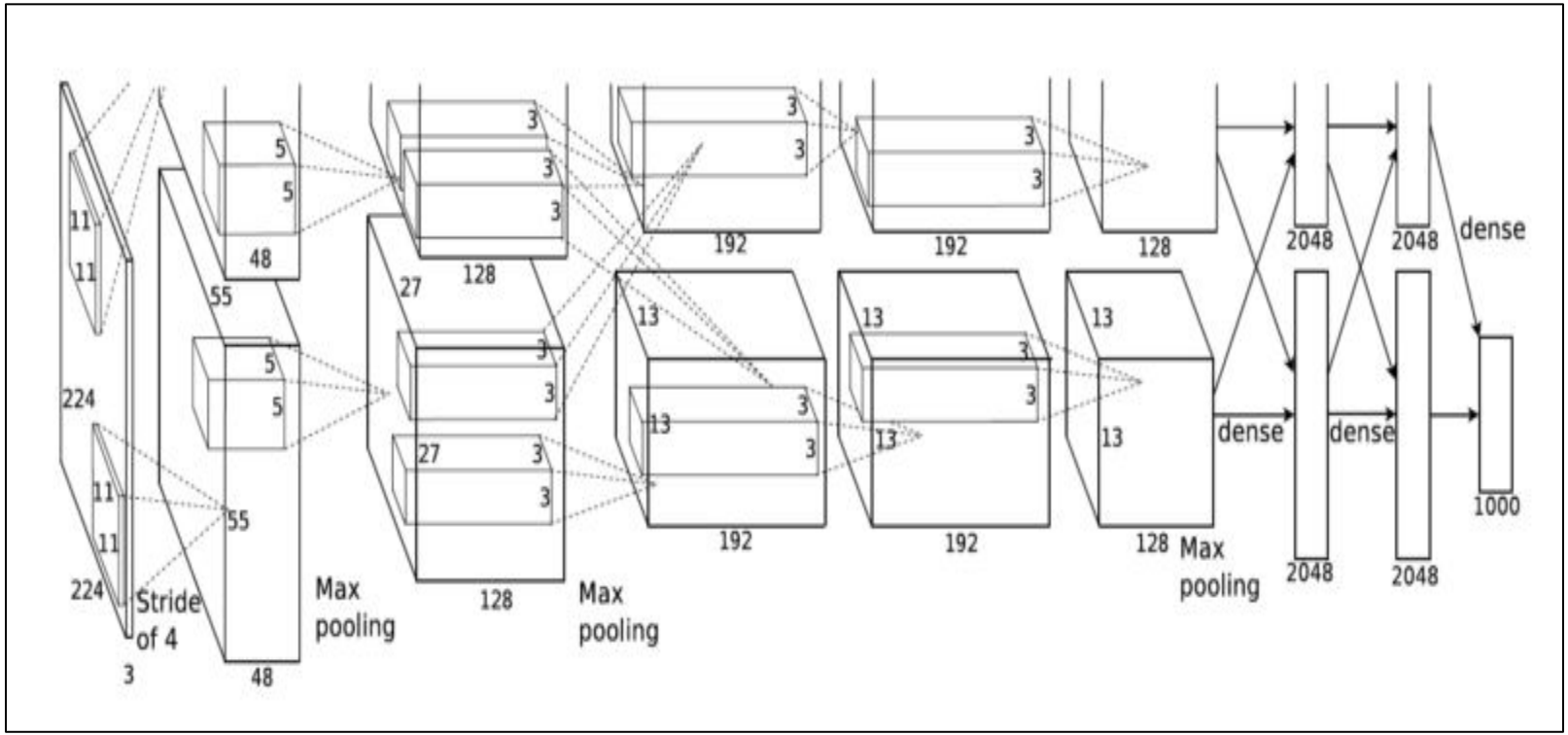
$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

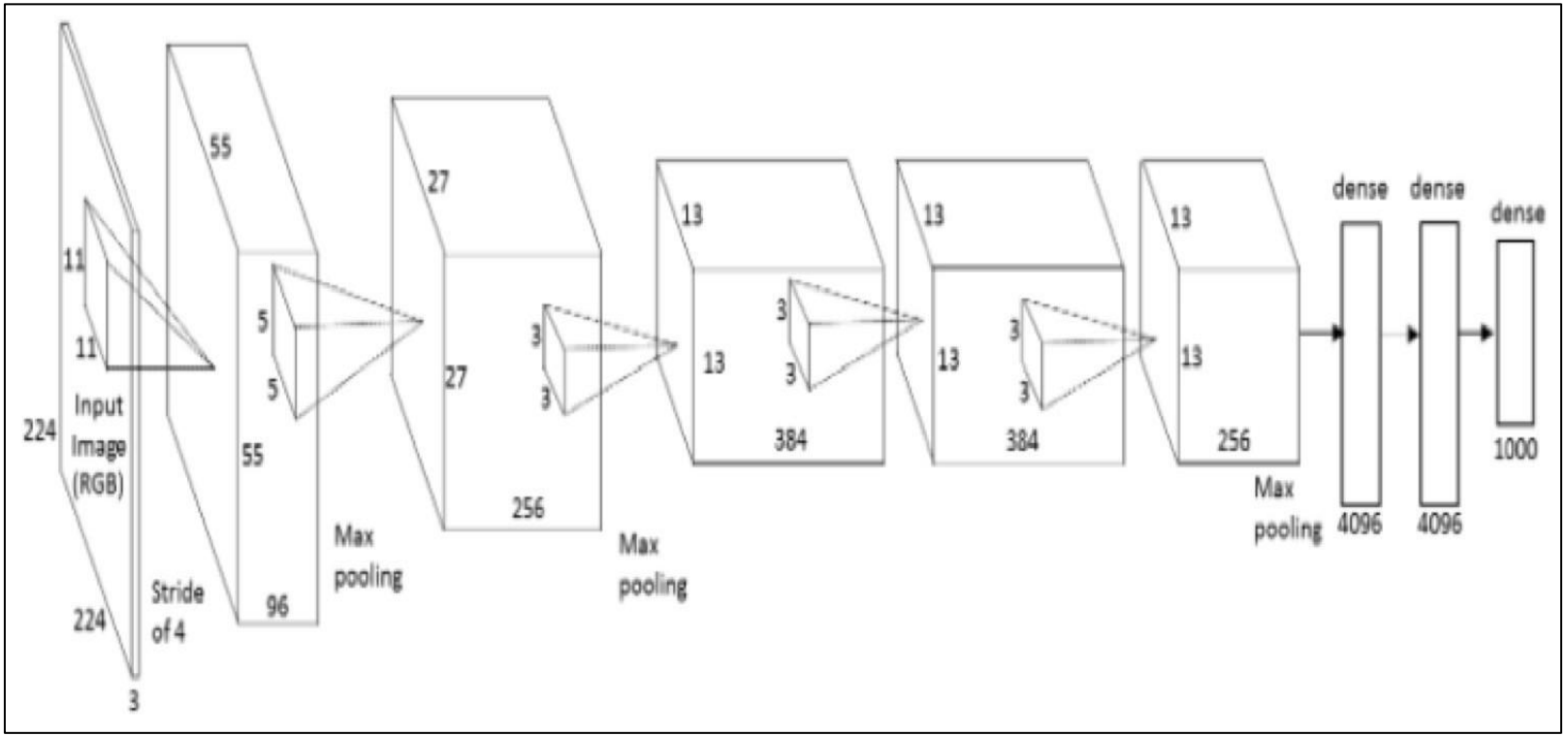
Architecture

AlexNet

The scary looking architecture



Simplified view



Characteristics of the network

- 5 **Convolution** layers followed by 3 **Fully Connected** layers
- Use of **ReLU** activation function
- Use of **Local Response Normalisation**
- Overlapping **Pooling**
- Use of **Dropout** to reduce overfitting

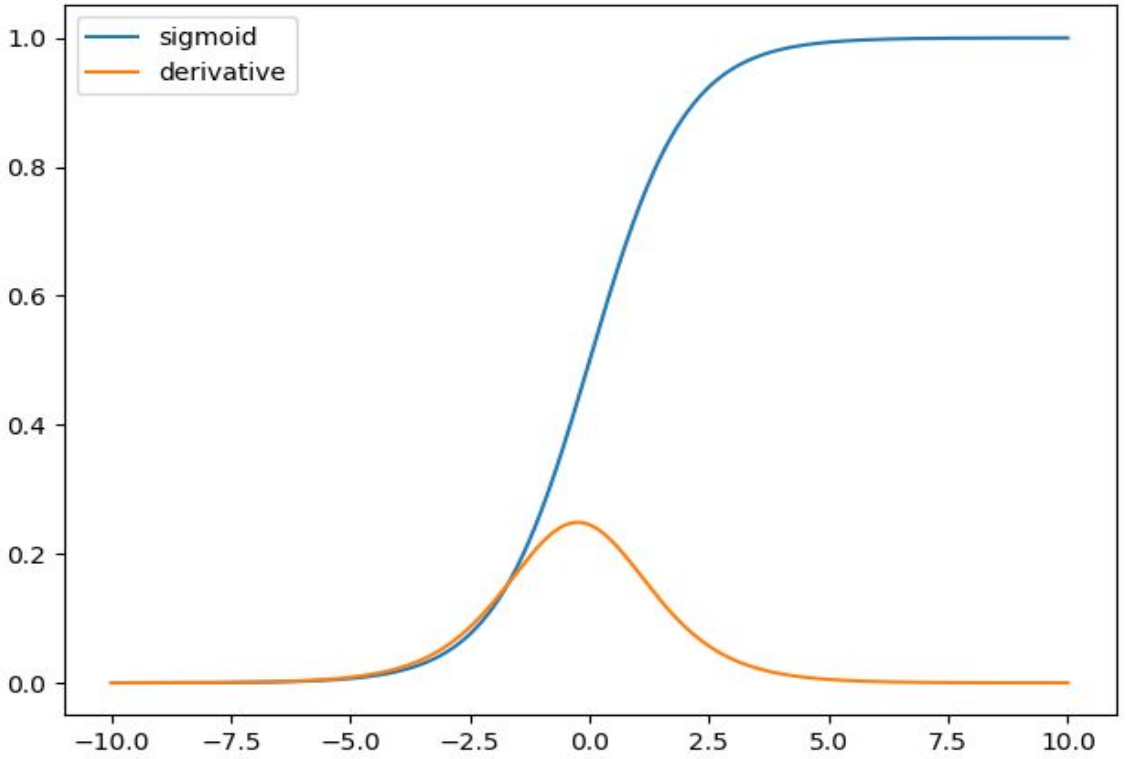
*“A humongous network with **60M** parameters and **650K** neurons”*

Architecture

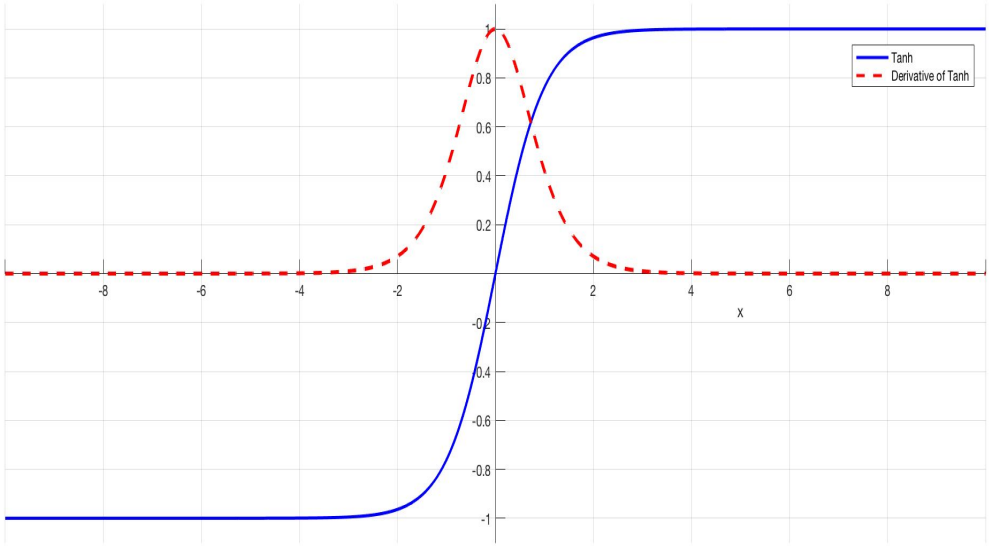
ReLU Nonlinearity

Sigmoid Function

$$f(x) = \frac{1}{1 + e^{-x}}$$



Hyperbolic Tangent Function



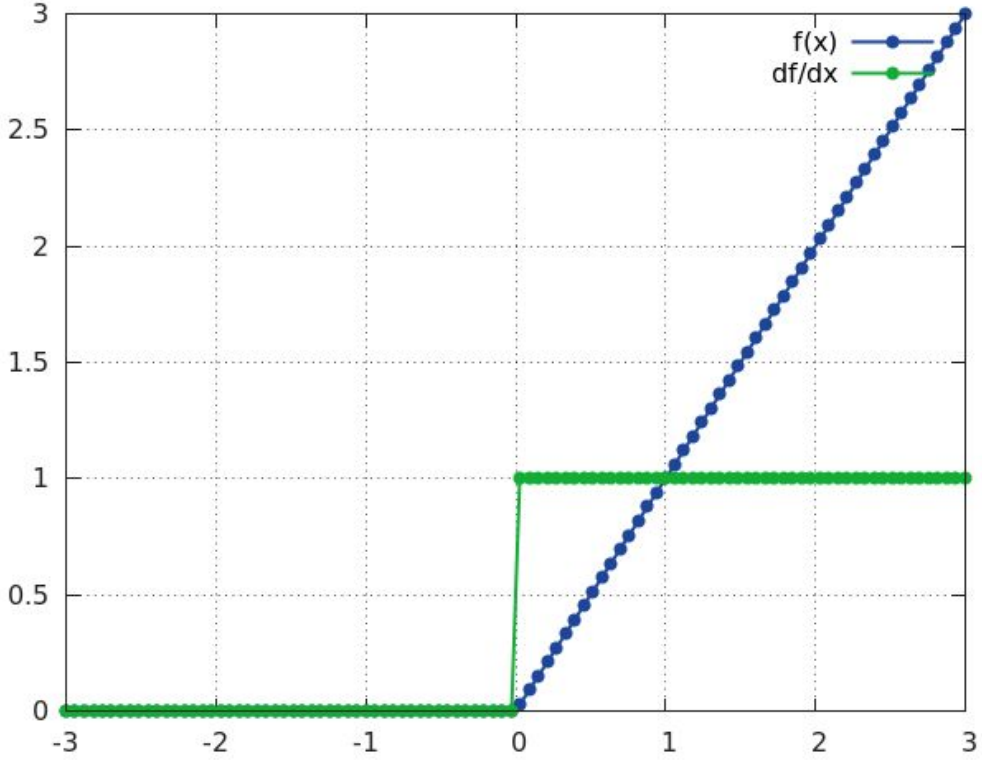
The problems?

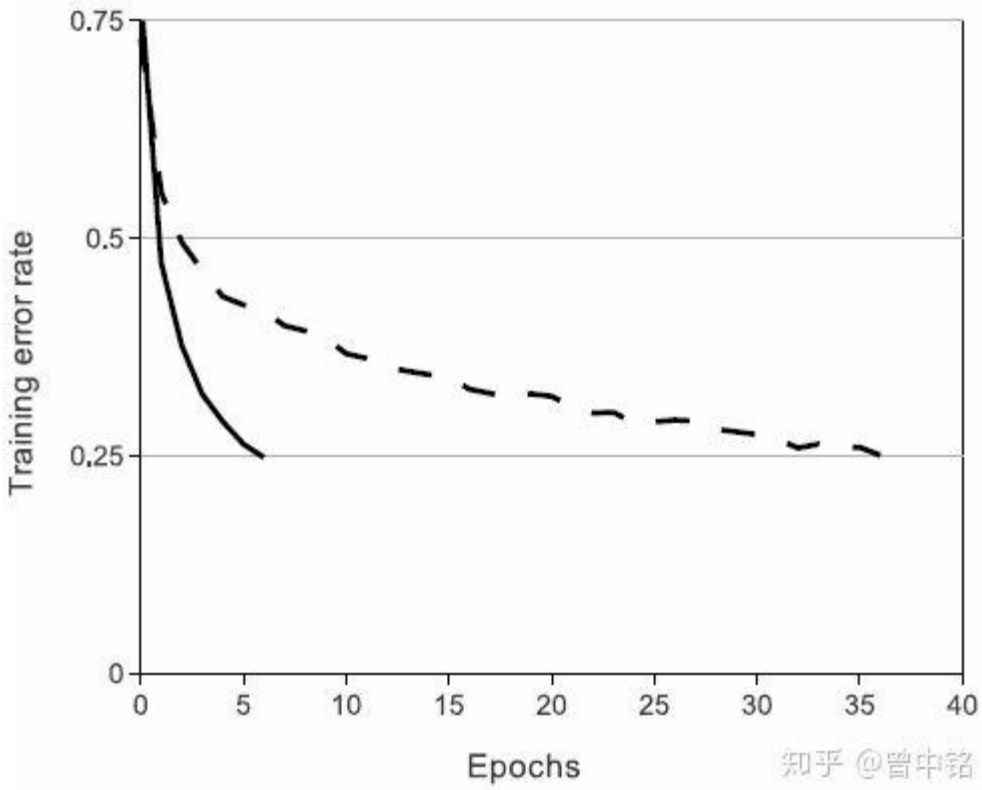
- Heavy Computations
- Vanishing Gradient

The solution?

Rectified Linear Units (ReLU)

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




0.75
0.5
0.25
0
Training error rate
0
5
10
15
20
25
30
35
40
Epochs
知乎 @曾中铭

Architecture

Local Response Normalization

The formula

$$b^i_{x,y} = a^i_{x,y} / (k + \alpha \sum_{j=max(0,i-n/2)}^{j=min(N-1,i+n/2)} {a^j_{x,y}}^2)^\beta$$

where

$b^i_{x,y}$ – regularized output for kernel i at position x, y

$a^i_{x,y}$ – source ouput of kernel i applied at position x, y

N – total number of kernels

n – size of the normalization neigbourhood

$\alpha, \beta, k, (n)$ – hyperparameters

Architecture

Overlapping Pooling

Use of kernel size 3x3 and stride of 2 instead of traditional kernels of size 2x2 and stride 2

Architecture

AlexNet, once again

The layers in AlexNet

- Input dimension: 224x224x3
- Conv1 with 96 kernels of size 11x11x3, having stride = 4. Output dimension: 55x55x96
- LRN1 and Pool1. Output dimension: 27x27x96
- Conv2 with 256 kernels of size 5x5x48. Output dimension: 27x27x256
- LRN1 and Pool2. Output dimension: 13x13x256
- Conv3 with 384 kernels of size 3x3x256. (This one take maps from both GPUs). Output dim: 13x13x384
- Conv4 with 384 kernels of size 3x3x192. Output dim: 13x13x384
- Conv5 with 256 kernels of size 3x3x192. Output dim: 13x13x256
- Pool3. Output dim: 6x6x256
- FC1 and FC2 having 4096 units each
- FC3, the output layer. 1000 units, one for each class.

Let's see the code!