Deep Learning - Vision 1
Introduction to CNNs

Sumandeep Banerjee

Module 3: Deep Learning

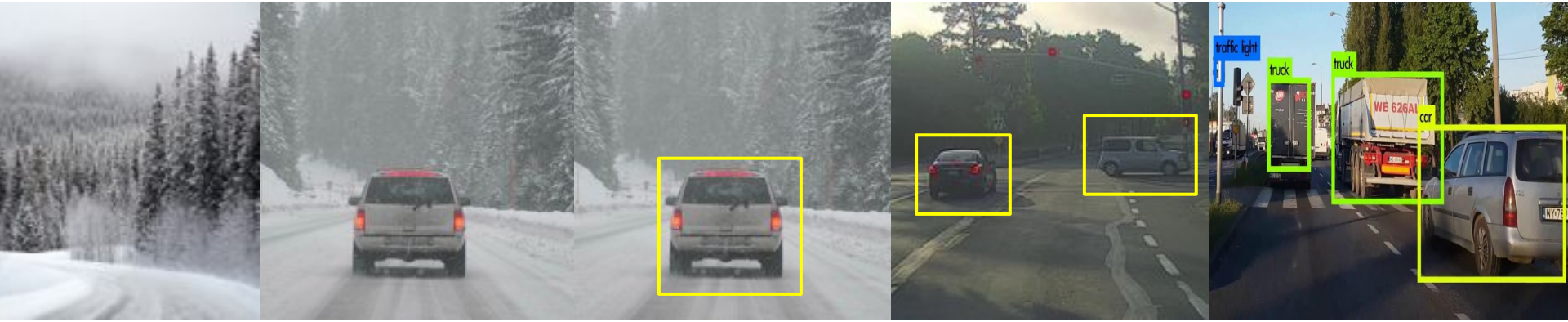
https://shala2020.github.io/

Learning Objectives

- Applications of Computer Vision
- Motivation from Digital Image Processing Filters
- Implementation of Convolution
- Building a Convolutional Neural Network
- Suitability of Convolutions to Computer Vision
- Case Study - Image Classification using CNNs
- Advanced CNN Architectures
- Practical Training Techniques

Applications of Computer Vision

Classification vs Localization vs Detection



Car or No Car ?

Image Classification

Where is the car ?

Object Localization

Where are the cars ?

Object Detection

What all is there ?

Multi-Class Detection

Special Case: Recognition

Single Object

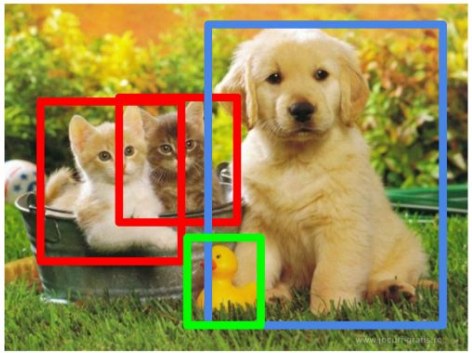
Multiple Objects

Detection vs Segmentation

Object Detection

Object Segmentation

Rectangular Regions



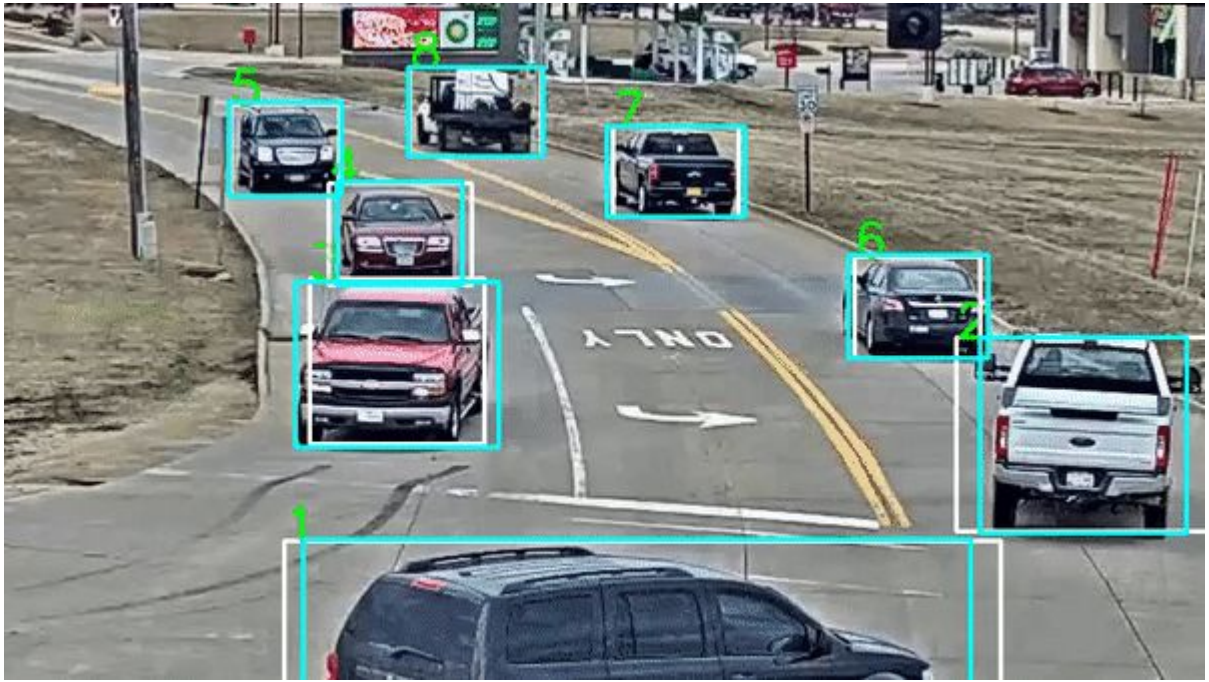
Pixel Level Classification

CAT, DOG, DUCK

CAT, DOG, DUCK

Multiple objects

Object Tracking



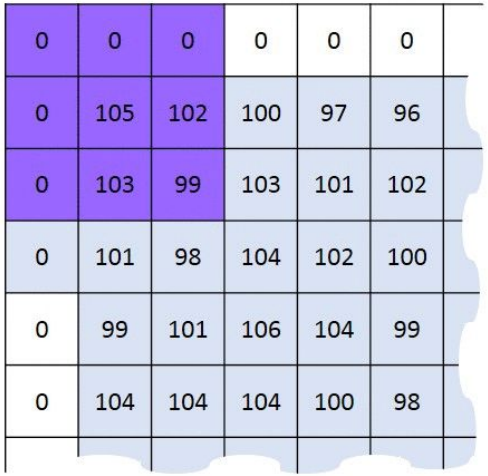
- Where did it go in the next frame ?
- Prediction of changing positions, mapping the correct object identity

Computer Vision using Deep Learning - Challenges

- Classification - Dog or Not Dog ?
- Input Image
  - Size - 1024 X 1024 pixels
  - Colors - 3 channels (RGB)
- Input Layer Dimension
  - 3M X 1 vector
- First Layer Units
  - 1000 - reasonable considering complexity
- Dimension of Weight Matrix
  - 3M X 1000 matrix
  - 3 Billion elements - not sparse !!!
  - 32 bit float => 12GB MEM just the first layer
  - Not feasible
- Solution - Convolution

Motivation from Digital Image Processing Filters

What is Convolution Operation ?



0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

320					

Output Matrix

$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 = 320$$

Edge Detection - Sobel

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

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

Noise Removal - Gaussian LPF

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

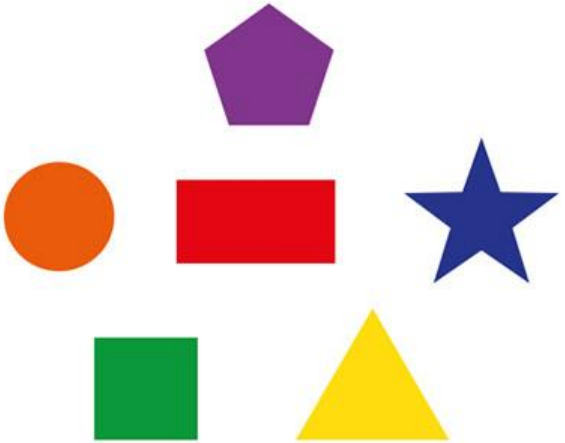
Sharp Feature Detection - Laplacian HPF

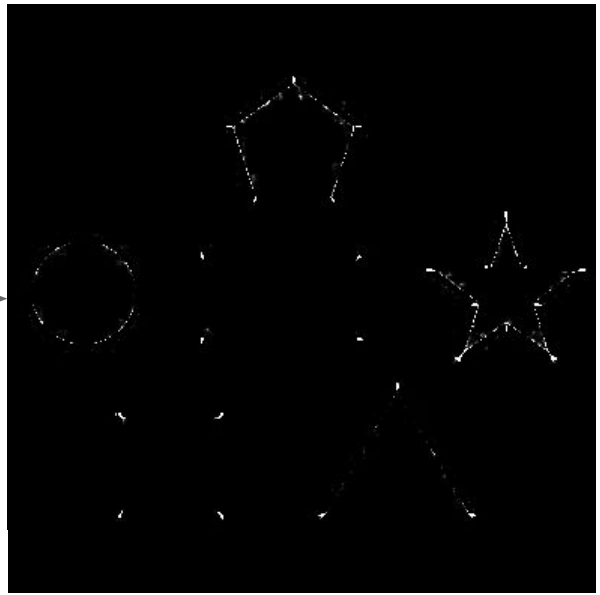
−1	−1	−1
−1	8	−1
−1	−1	−1

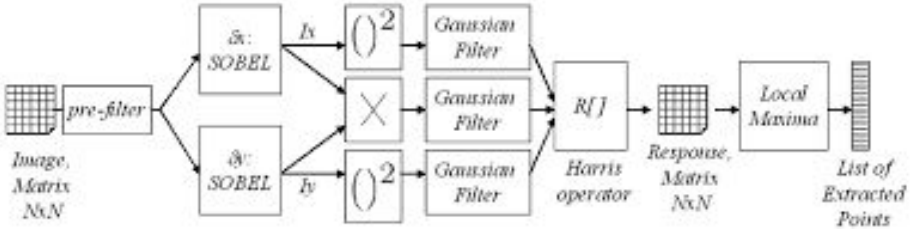
Two Stage Edge Detection - Laplacian of Gaussian

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \oplus \begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix}$$

Harris Corner Detection (6 stage filter)



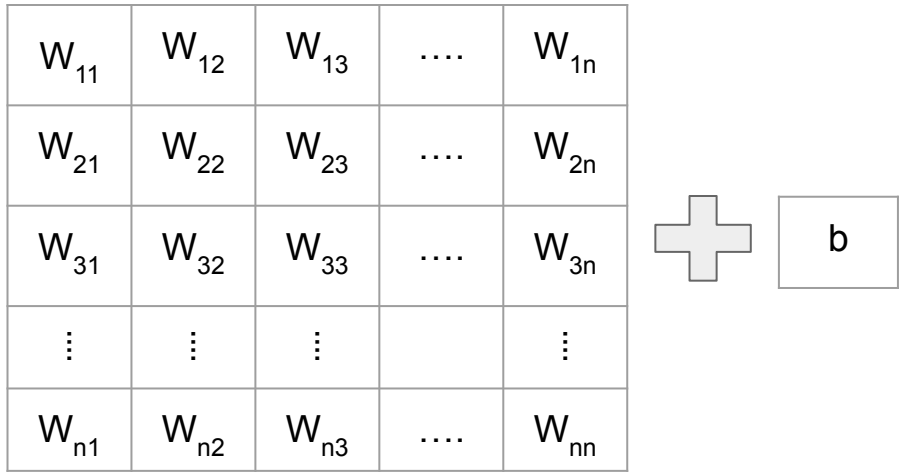




Hand Picked vs Data Driven Filter Design

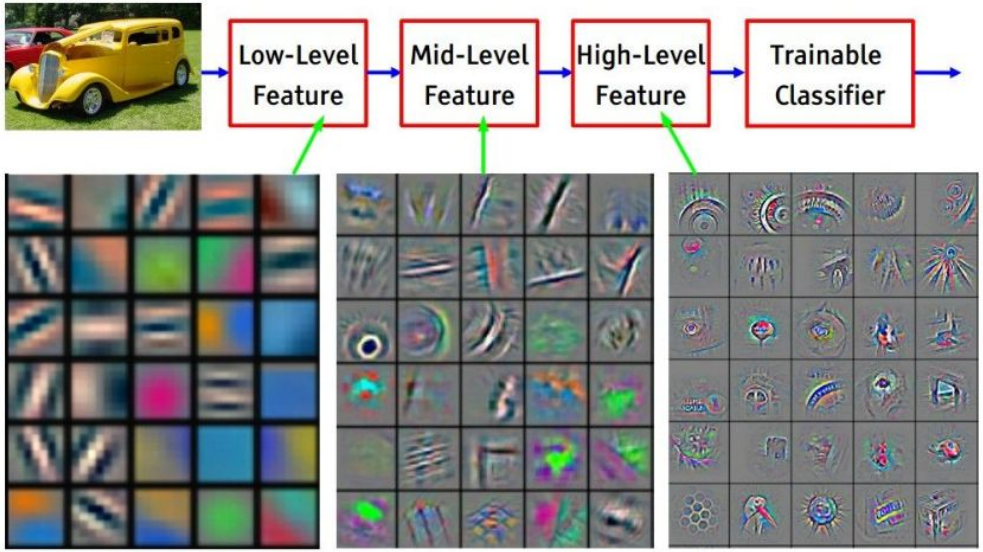
- What features are be computed ?
- What should be the best filter weights ?

Use training to learn the filter coefficients at each stage



What does CNN Layers Do ?

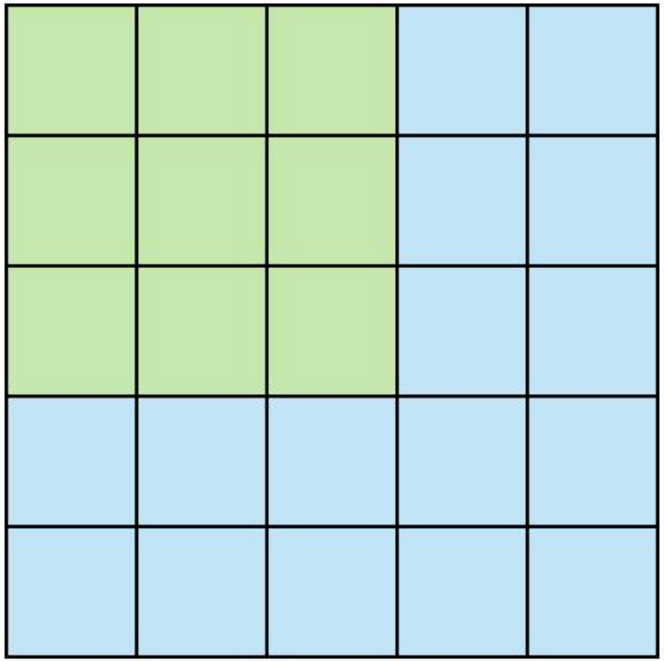
- Early Layers
  - Detects edges, corners, blobs etc
- Middle Layers
  - Detect parts of object
- Later Layers
  - Detects near complete sub-objects
- Last Layer
  - Final classification output



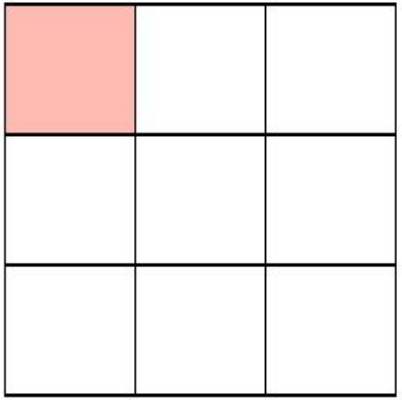
Implementation of Convolution

Valid Convolution

Image: n X n
Filter: f X f



Stride 1

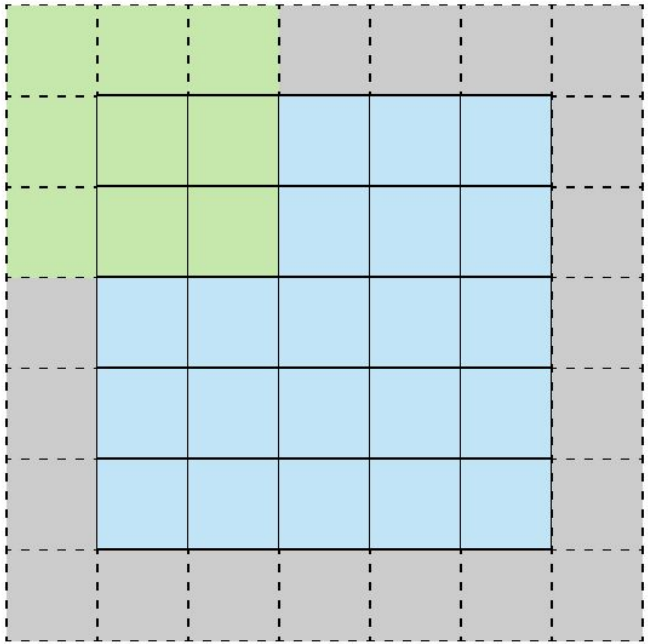


Output: (n-f+1)X(n-f+1)

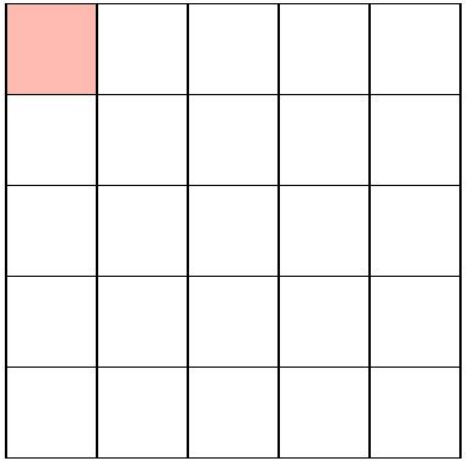
Feature Map

Same Convolution - Padding

Image: n X n
Filter: f X f
Padding: p



Stride 1 with Padding



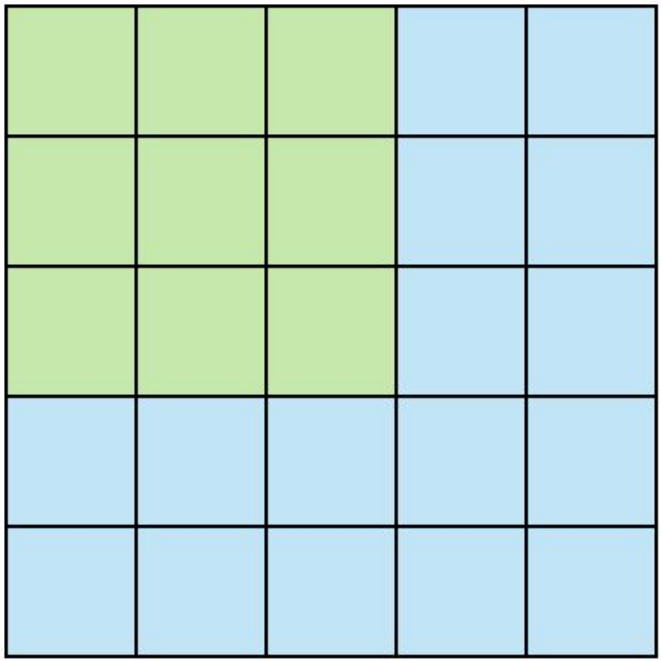
Feature Map

Output: (n+2p-f+1)X(n+2p-f+1)

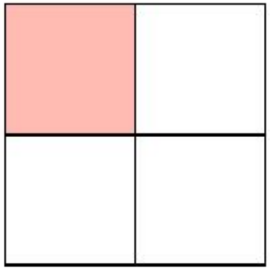
To maintain size: p = (f-1)/2

Strides

Image: n X n
Filter: f X f
Padding: p
Stride: s



Stride 2

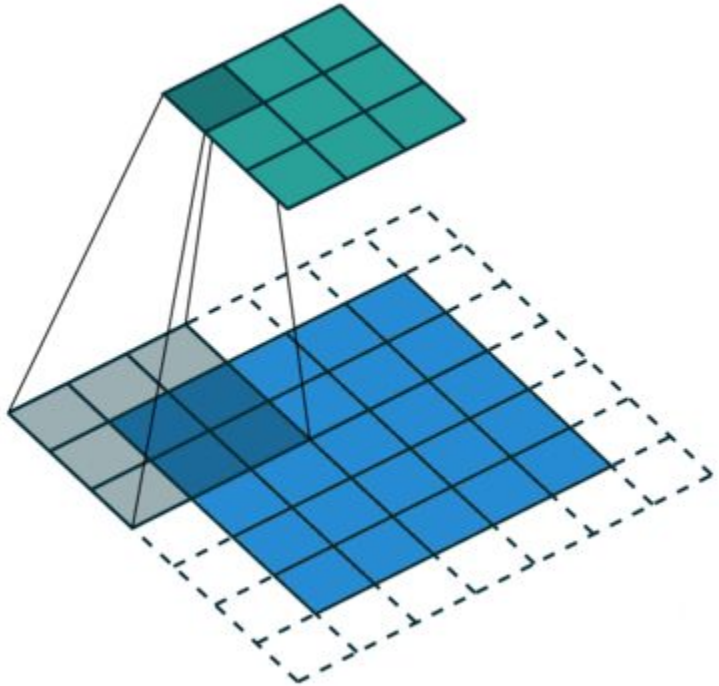


Feature Map

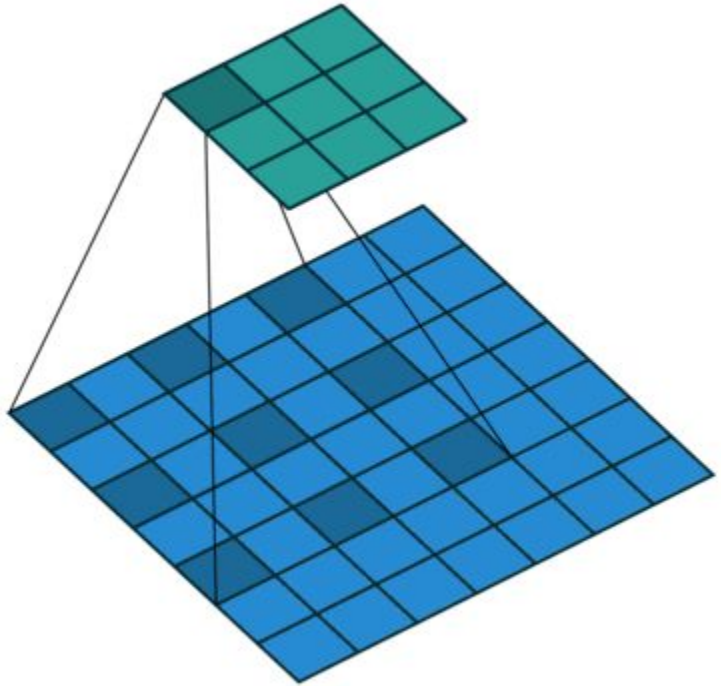
Output: ((n + 2p - f) / s + 1)X((n + 2p - f) / s + 1)

Avoid fractions, use floor function if cannot

Dilated Convolution

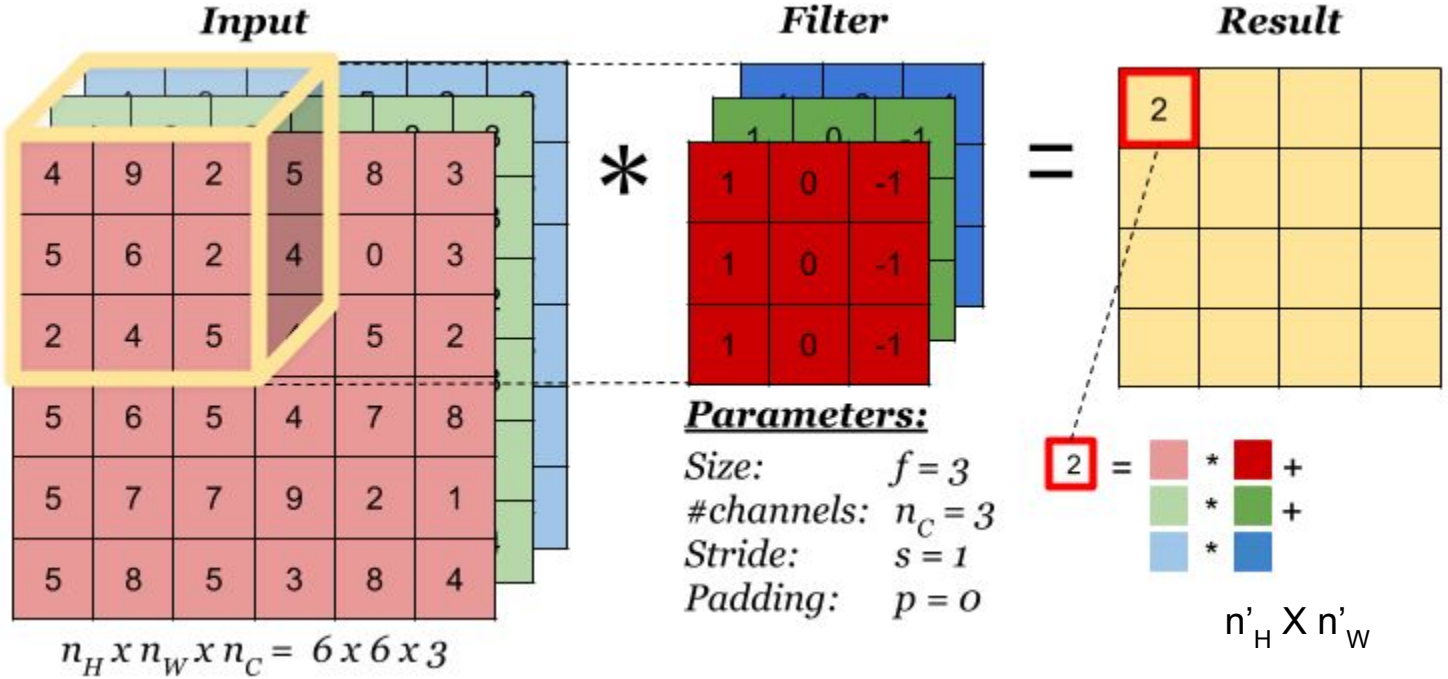


Standard Convolution (l=1)

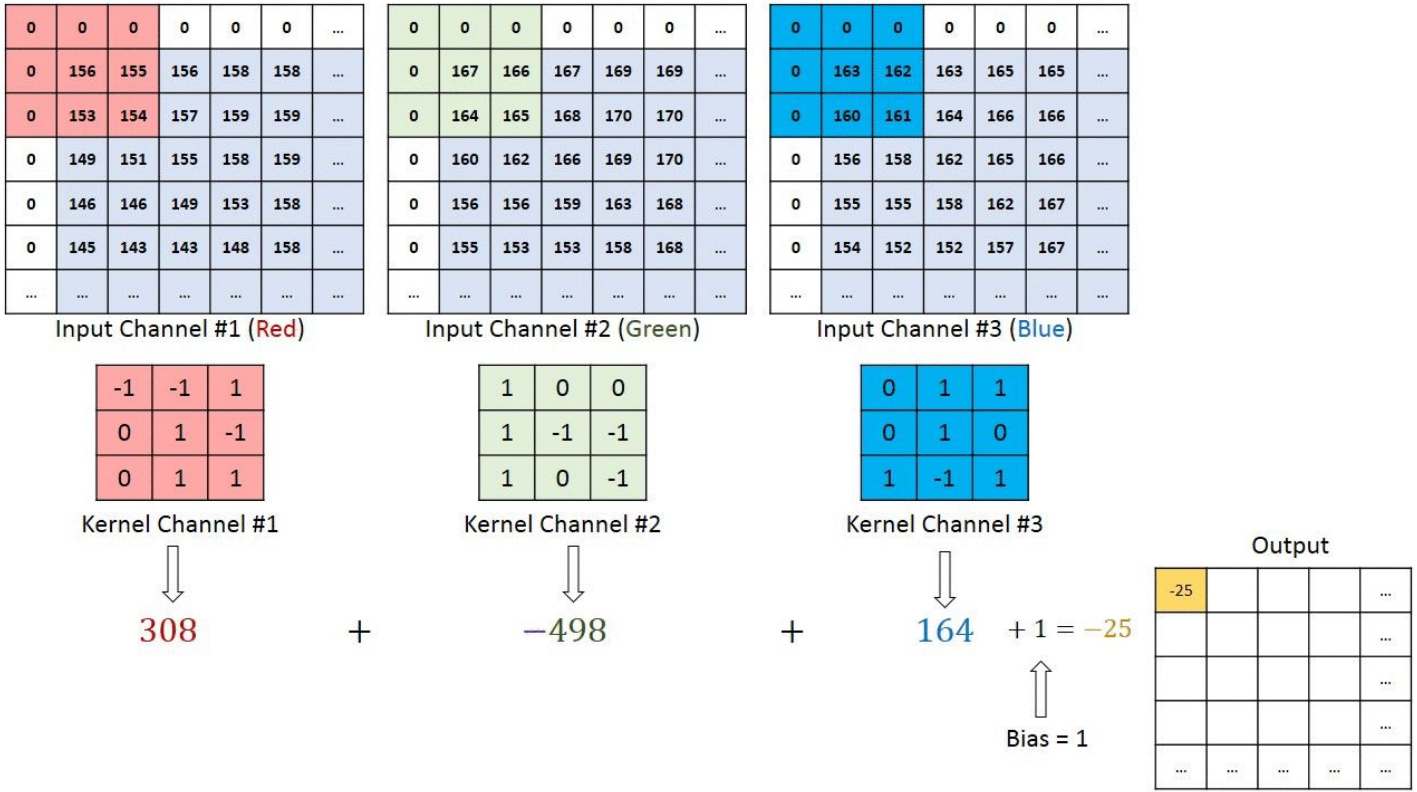


Dilated Convolution (l=2)

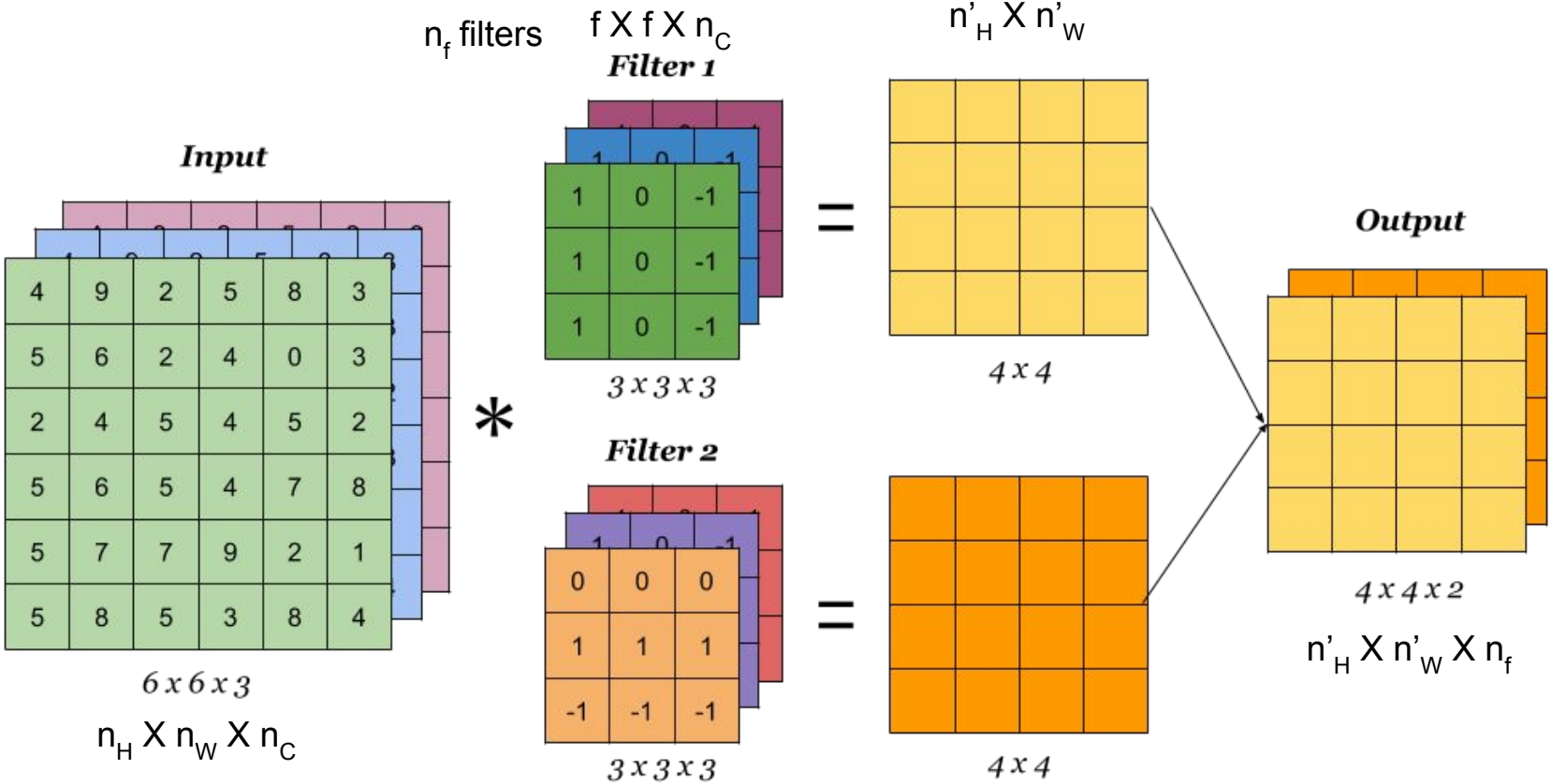
Multi-Channel Convolution



Multi-Channel Convolution

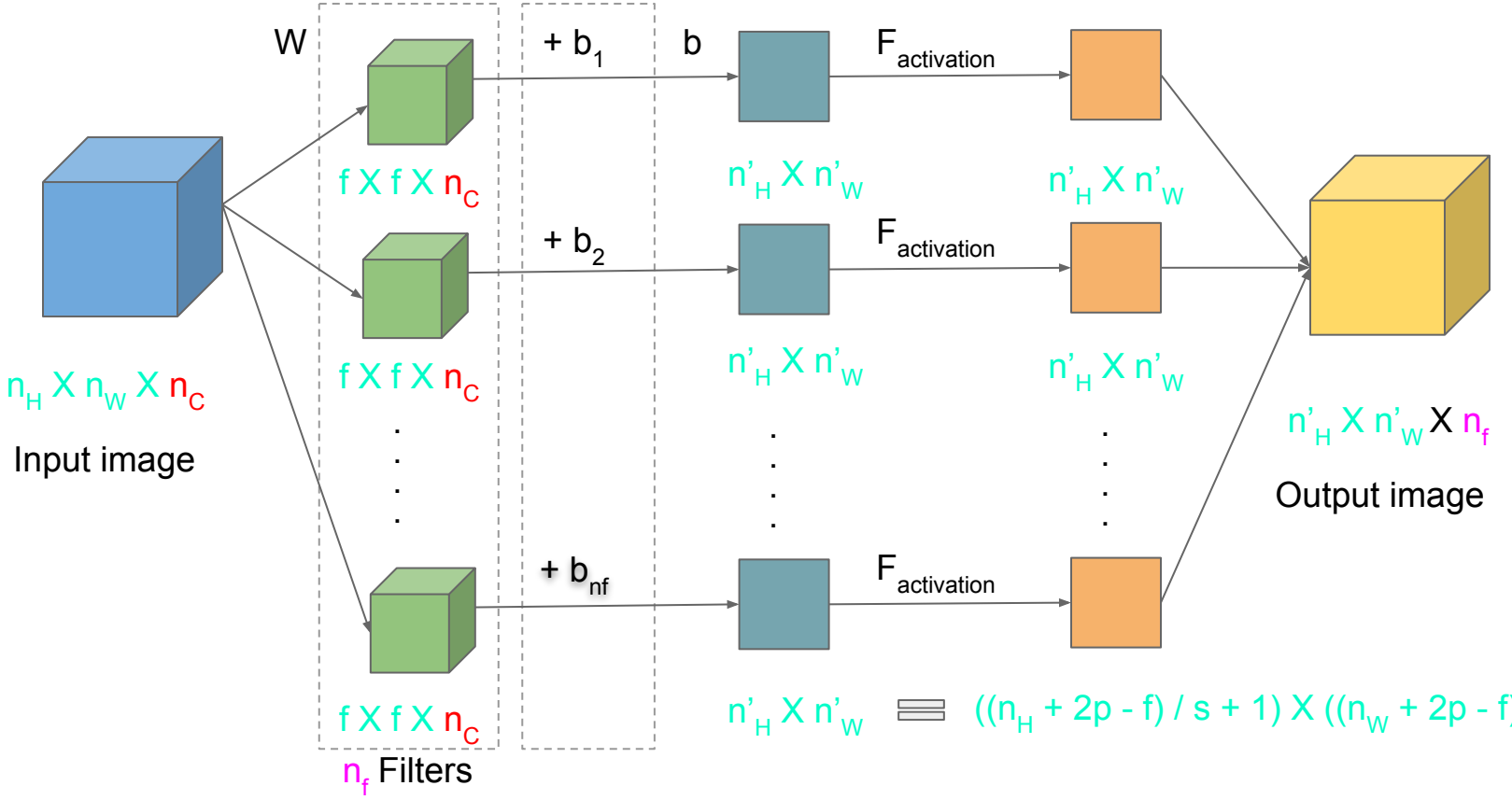


Multi-Filter Convolution



Building a Convolutional Neural Network

Single Convolutional Layer

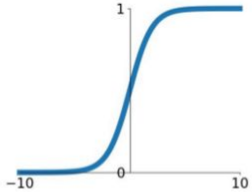


Activation Functions

Adds non-linearity to the network

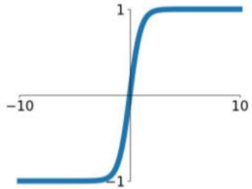
Sigmoid

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



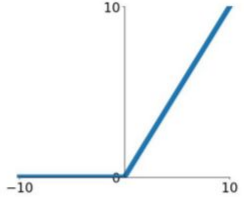
tanh

$$\tanh(x)$$



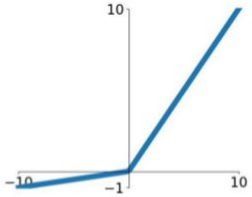
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

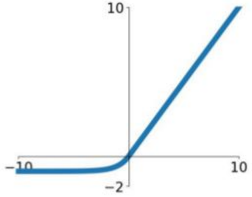


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

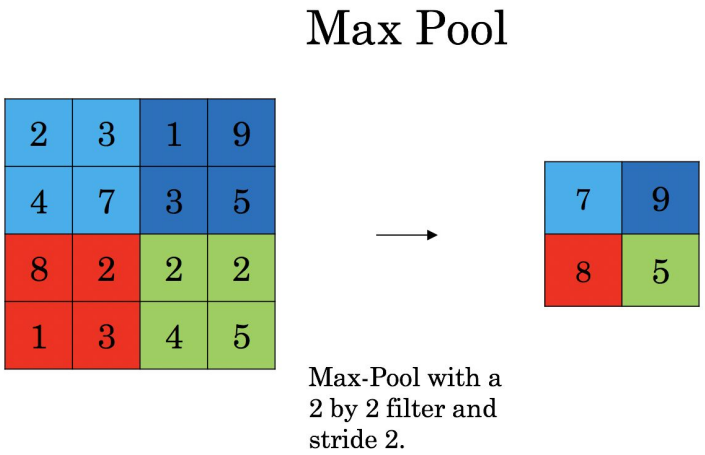
ELU

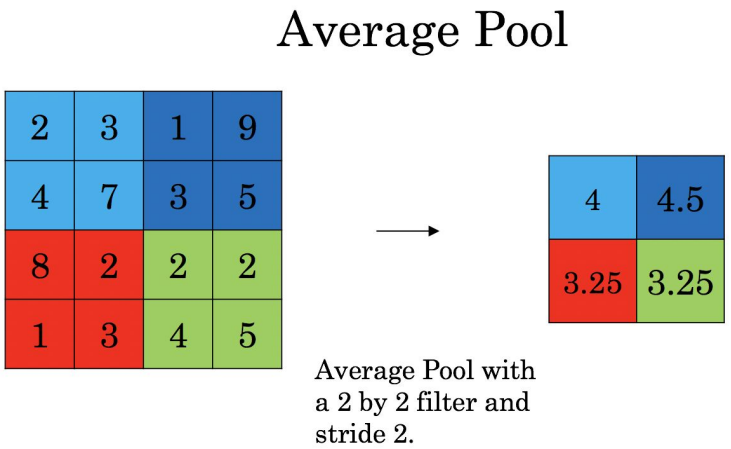
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Pooling Layers

Enlarges the receptive field of subsequent layers





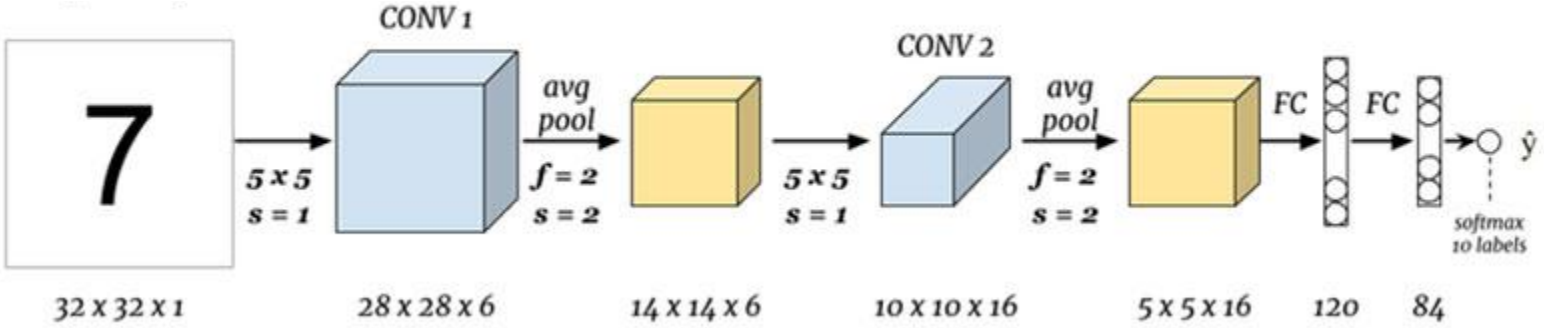
Select the most important feature
(almost exclusively used nowadays)

Select the average of incoming features
(earlier research)

Apply on each channel independently

LeNet-5 Model

n_H, n_W goes down, n_c goes up along the depth. Why?

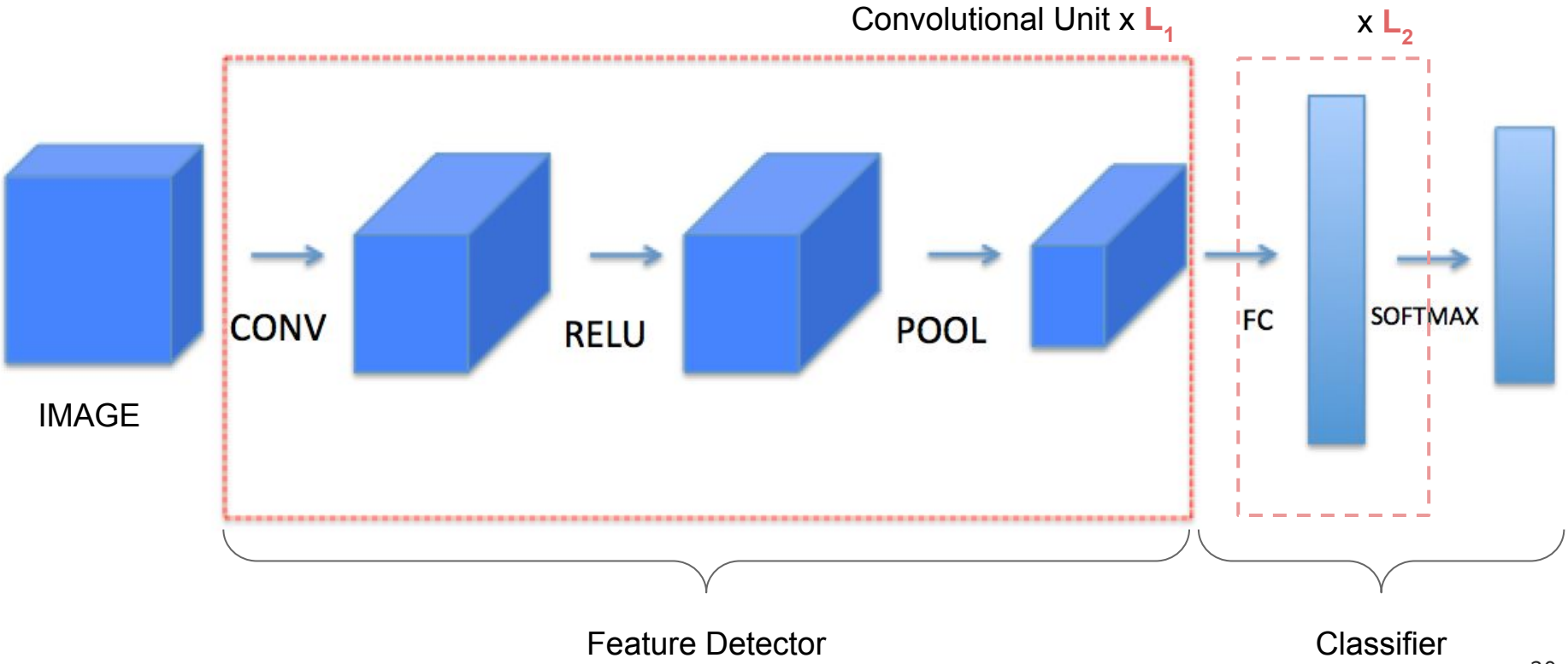


tanh activation for all layers
softmax activation for last (output) layer

~ 61K parameters

[LeCun et al., 1998, Gradient-based learning applied to document recognition]

A Complete Convolutional Neural Network



Hyperparameters of a Classic CNN

- Input image size (n_H, n_W, n_C)
- Number of Convolutional Units (L_1)
  - Size of filters in each layer (f)
  - Padding (p)
  - Stride (s)
  - Number of filters in each layer (n_f)
  - Size of pooling filter(f_p)
  - Stride of pooling filter (s_p)
- Number of fully connected layers (L_2)
  - Number of nodes in each FC layer (n_u)

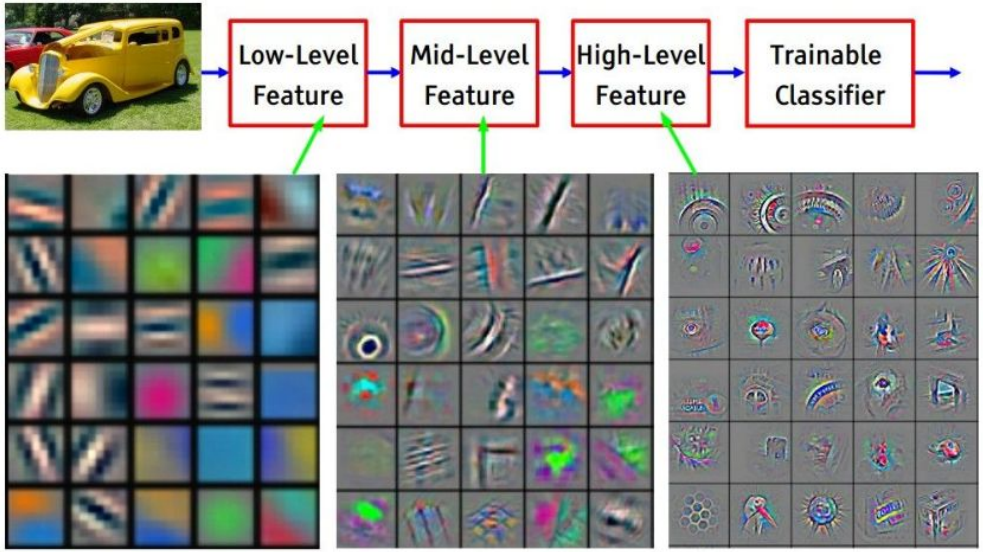
Suitability of Convolutions to Computer Vision

Why Convolutions Over Dense Neural Networks ?

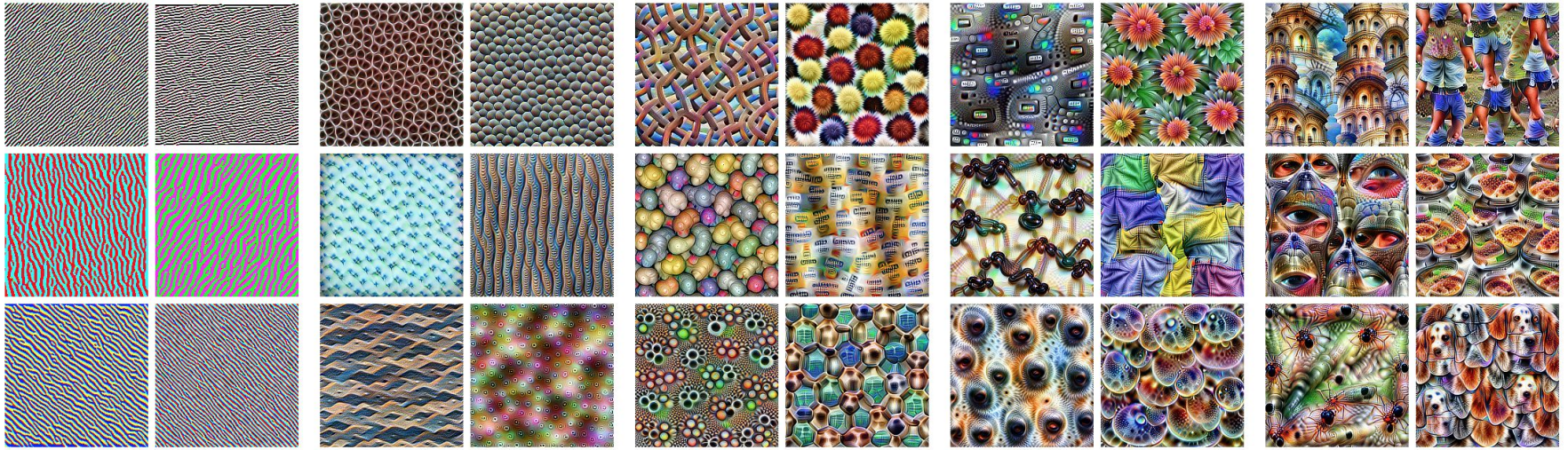
- Parameter Sharing => **Less Computations**
  - Each filter detects a different feature
  - Same feature detector useful in one part of image most likely useful in another part as well
- Sparsity of Connections => **Robustness to Localization**
  - Each output pixel is only connected to a limited number of input values
  - Relevant content may be contained only in a part of an image and not span the whole image

What does CNN Layers Do ?

- Early Layers
  - Detects edges, corners, blobs etc
- Middle Layers
  - Detect parts of object
- Later Layers
  - Detects near complete sub-objects
- Last Layer
  - Final classification output



Feature Visualization of CNN Layers



Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a) **Parts** (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

Feature visualization allows us to see how GoogLeNet [1], trained on the ImageNet [2] dataset, builds up its understanding of images over many layers. Visualizations of all channels are available in the appendix.

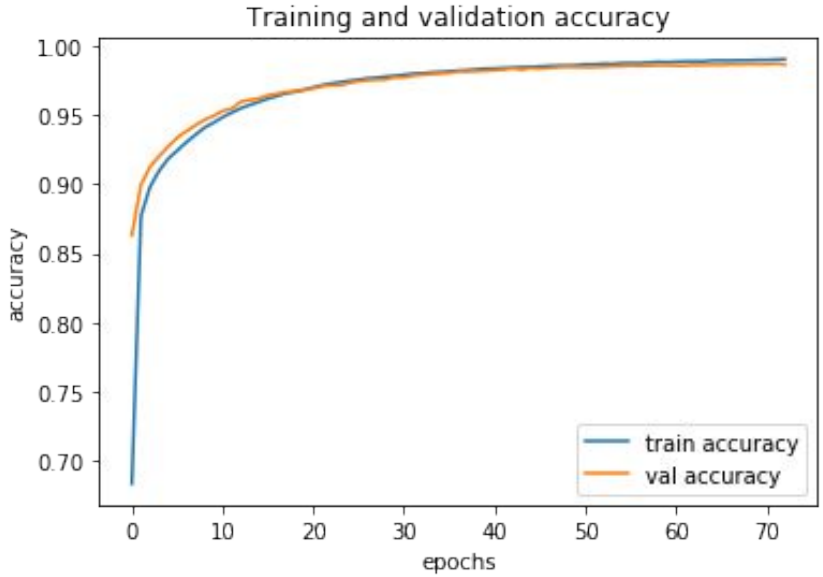
https://distill.pub/2017/feature-visualization/

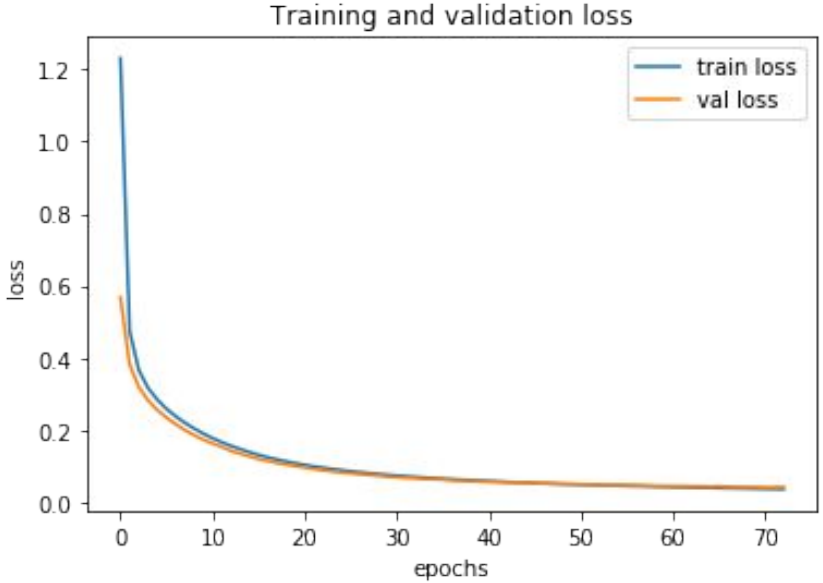
Case Study - Image Classification using CNNs

MNIST

- MNIST
  - handwritten number classification set (0-9)
  - 60K train set, 10K test set - 28X28 grayscale

MNIST Dataset Classification using LeNet5 CNN

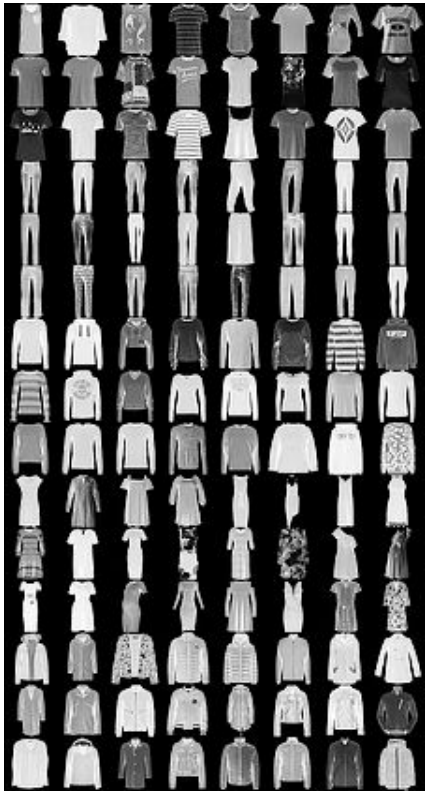




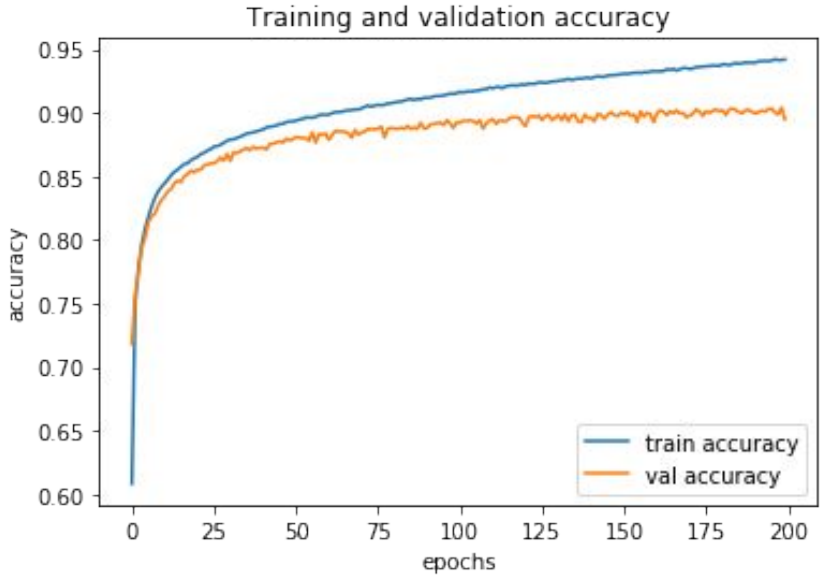
- Training accuracy reached **99%**. Validation accuracy reached **98.5%**
- Model is a **good fit**
  - Very good training and validation accuracy
  - Validation loss tracks training loss

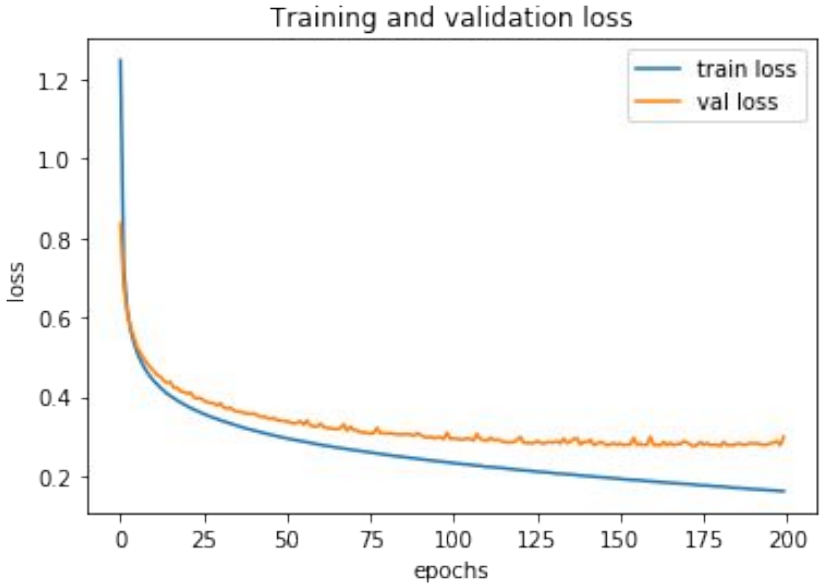
Fashion-MNIST

- Fashion-MNIST
  - image object classification set - **t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot**
  - 60K train set, 10K test set - 28X28 grayscale



Fashion MNIST Dataset Classification using LeNet5 CNN

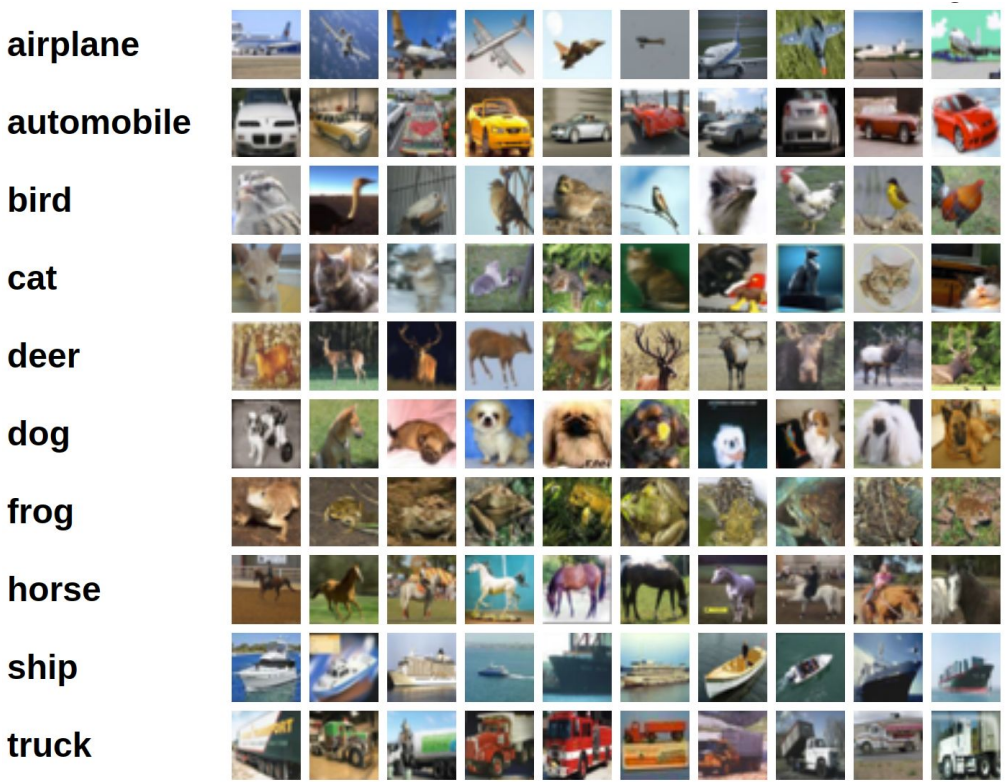




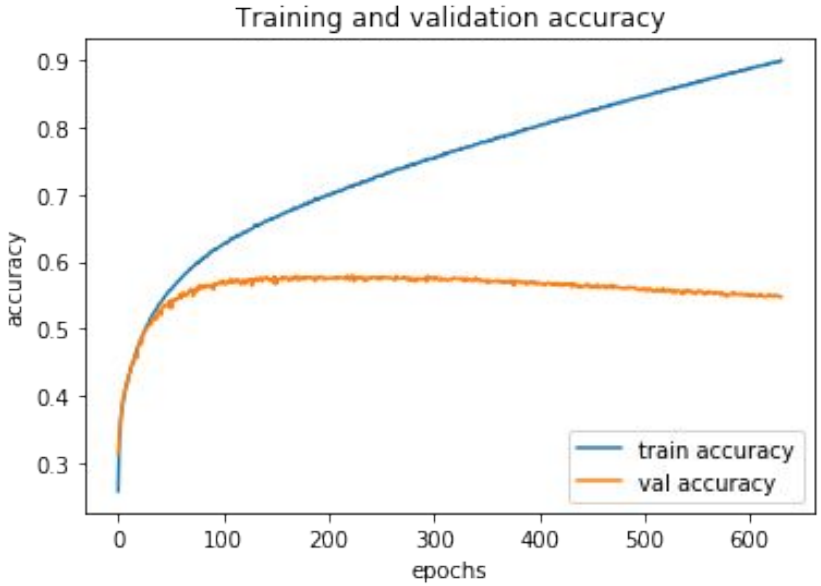
- Training accuracy reached **95%.** Validation accuracy reached **90%**
- Model is **average with a tendency of slight overfitting**
  - Good training accuracy and acceptable validation accuracy
  - Validation loss diverges from training loss after initial few epochs

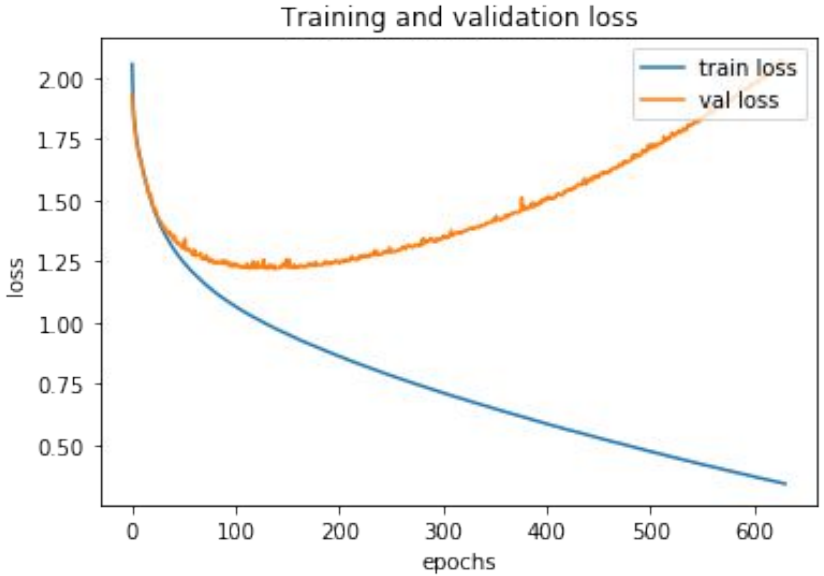
CIFAR10

- CIFAR10
  - image object classification set - **airplane, car, bird, cat, deer, dog, frog, horse, ship, truck**
  - 50K train set, 10K test set - 32X32 rgb color



CIFAR-10 Dataset Classification using LeNet5 CNN



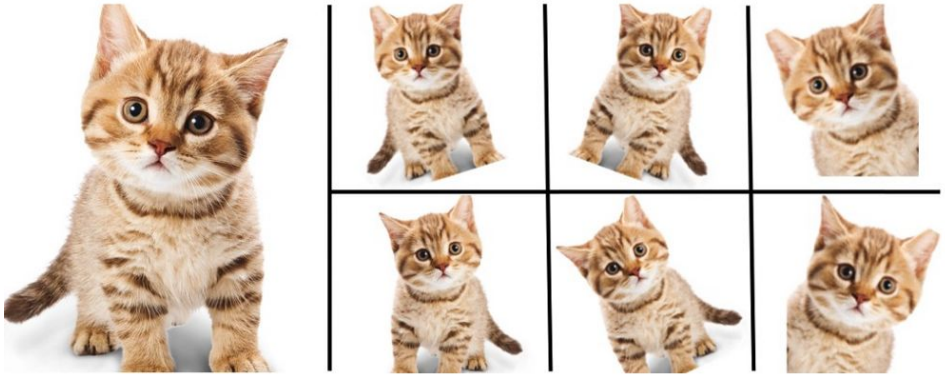


- Training accuracy reached **90%.** Validation accuracy reached **55%**
- Model is **poor with extreme overfitting**
  - Good training accuracy and very bad validation accuracy
  - Validation loss hugely diverges from training loss after a number of epochs

Practical Training Technique I - Data Augmentation

- Overfitting
  - Insufficient data
    - Trains well
  - Training data does not contain all potential variations
    - Hence poor validation/test results
- Gathering more data is expensive and time consuming
  - Labeling lots of new data is extremely tedious task
- Solution: Simulate variations
  - Modify training images and add to the training

Data Augmentation



- Mirroring
  - Horizontal flip
  - Vertical flip (rare)
- Zoom +/-
- Translation
- Rotation
- Shearing
- Distortions
  - Local Warping
- Brightness
- Color Shifting
  - Change tint

Advanced CNN Architectures

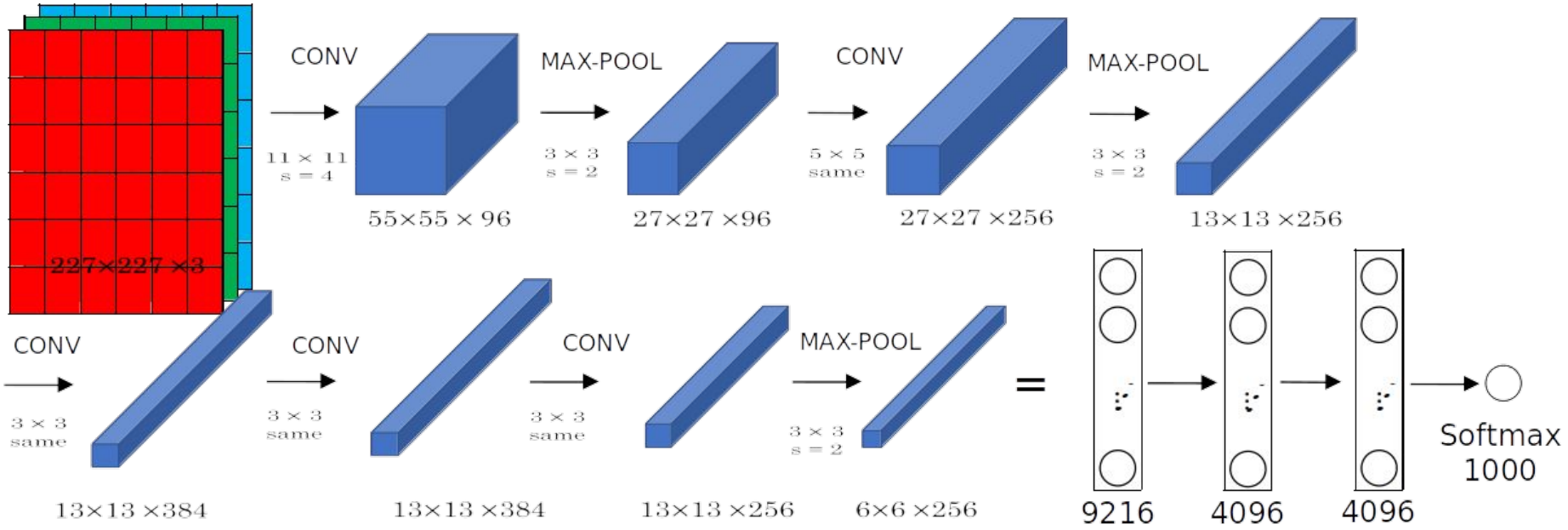
Progression of CNN Architectures

Story of Image Net Challenge Winners since 2012

- 2012 - AlexNet, University of Toronto - **83.6%**
  - 5 convolutional with/without max pooling layers, 3 fully connected layers with dropout, first use of ReLU activation
- 2013 - VGG-16/19, University of Oxford - **92.7%**
  - 16 -19 layers with only small 3X3 receptive fields
- 2014 - GoogleLeNet, Google - **93.3%**
  - 22 layers utilizing Network-in-Network inception module
- 2016 - ResNet, Microsoft - **96.4%**
  - 152 layers having residual blocks with identity skip connections

PS: Human Top 5 Accuracy ~ **95%**

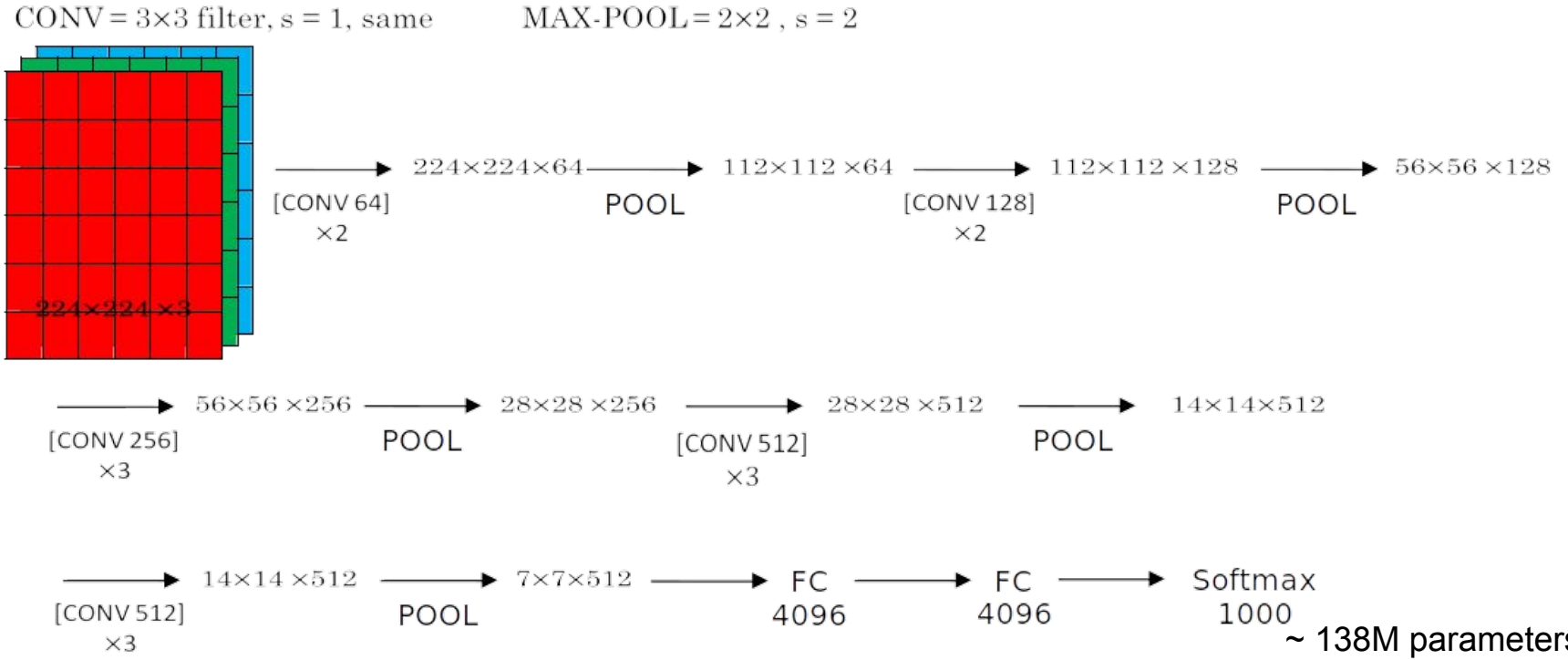
AlexNet Architecture (9 Layers)



~ 60M parameters

[Krizhevsky et al., 2012, ImageNet classification with deep convolutional neural networks]

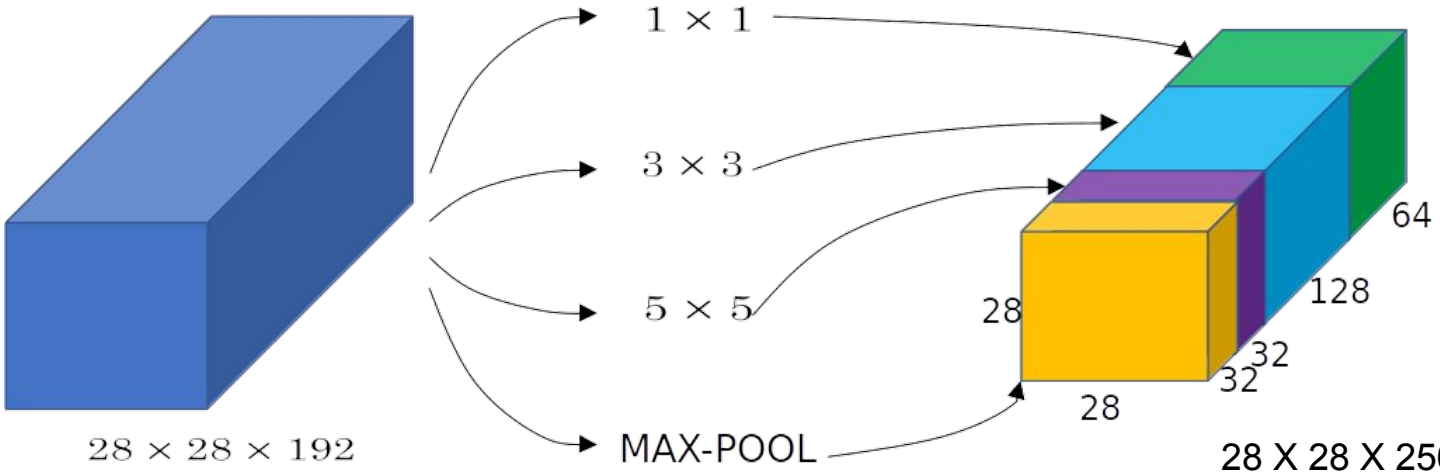
VGG-16



~ 138M parameters

[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

Filter Concatenation aka Network-in-Network

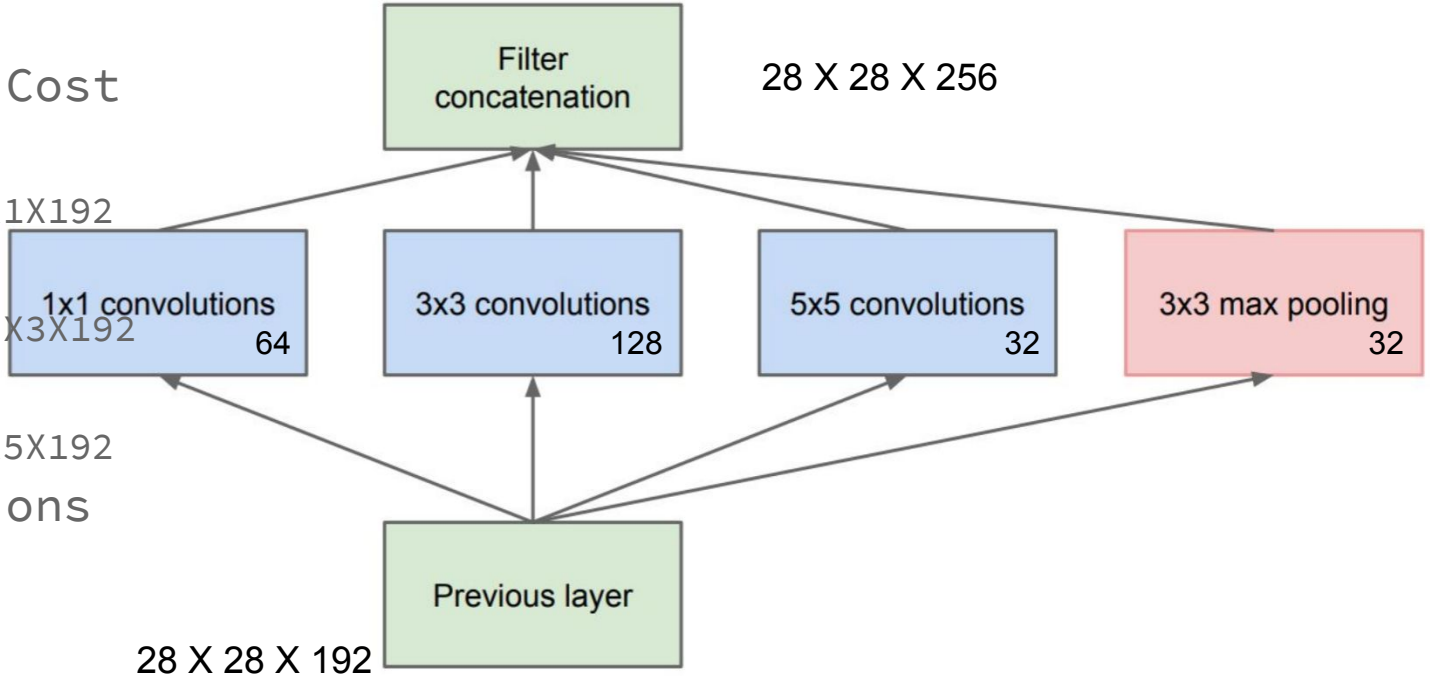


- 1X1 Conv: Effectively a fully connected layer within sub-data (weighted sum across channels with activation)
  - Non-trivially alter n_C just as Pooling layers shrink n_H, n_W

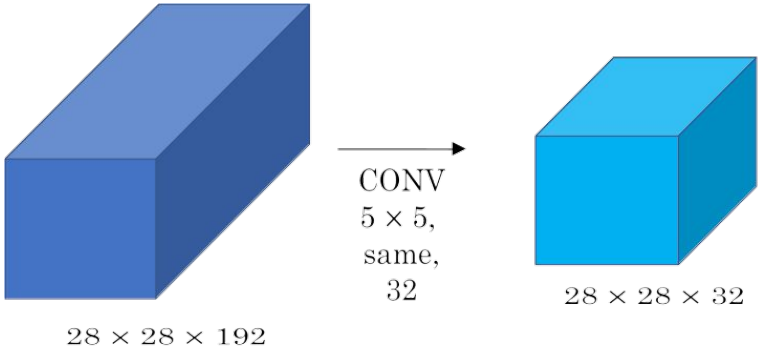
[Lin et al., 2013. Network in Network]

Inception Module

- Computation Cost
- [1X1 Conv]
  - 28X28X64X1X1X192
- [3X3 Conv]
  - 28X28X128X3X3X192
- [5X5 Conv]
  - 28X28X32X5X5X192
- 303M operations



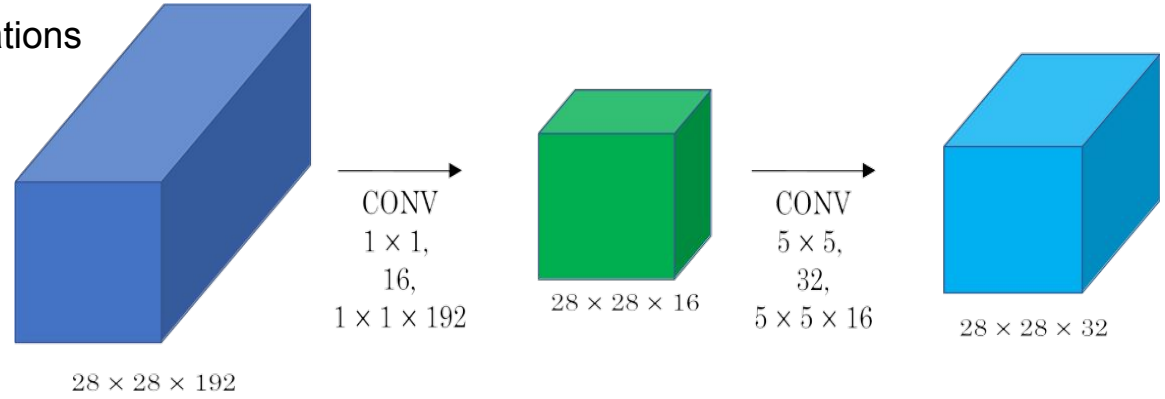
High Computational Cost - Soln. is Bottleneck Layer



Channel lowering within reason doesn't show any noticeable drop in accuracy

28X28X32 X 5X5X192 ~ 120M operations

10X Speed-up

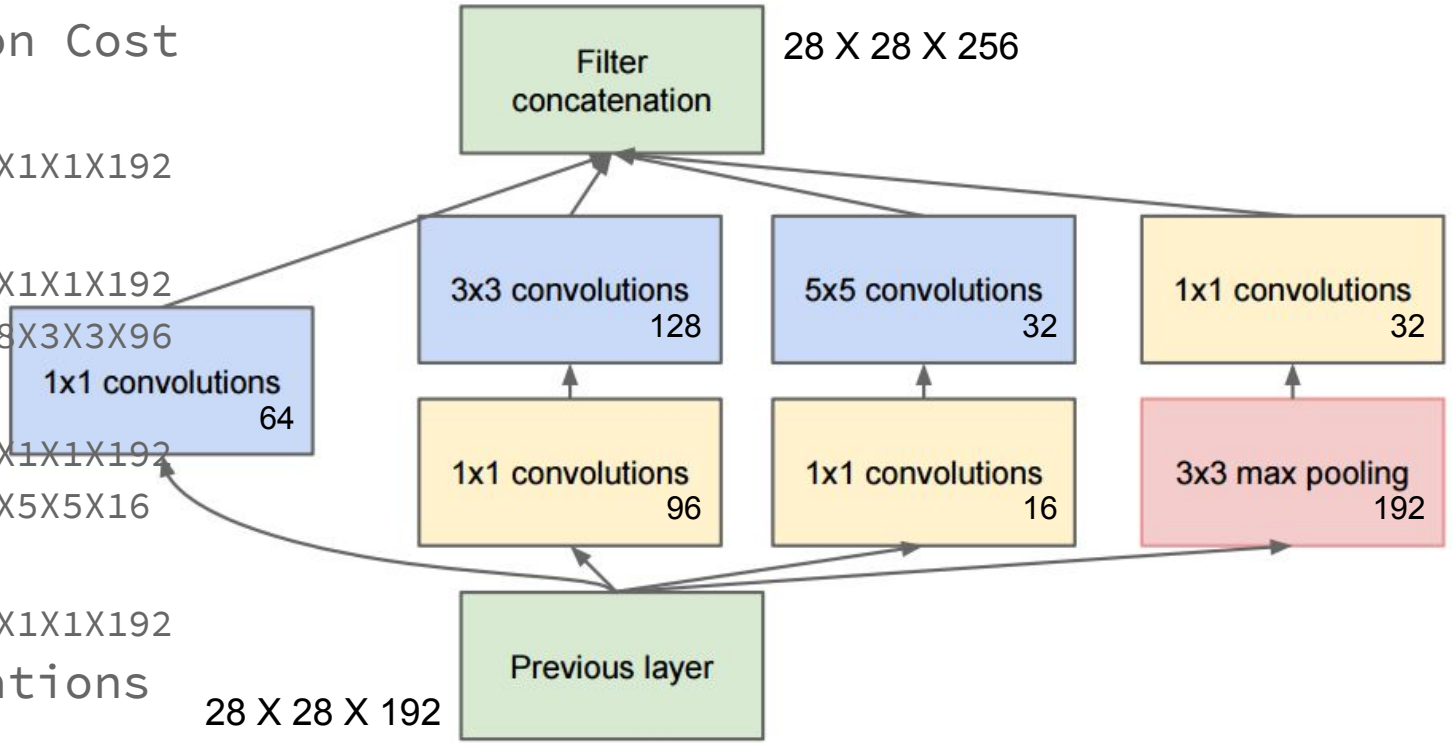


Total ~ 12.4M Only → 28X28X16 X 192 ~ 2.4M

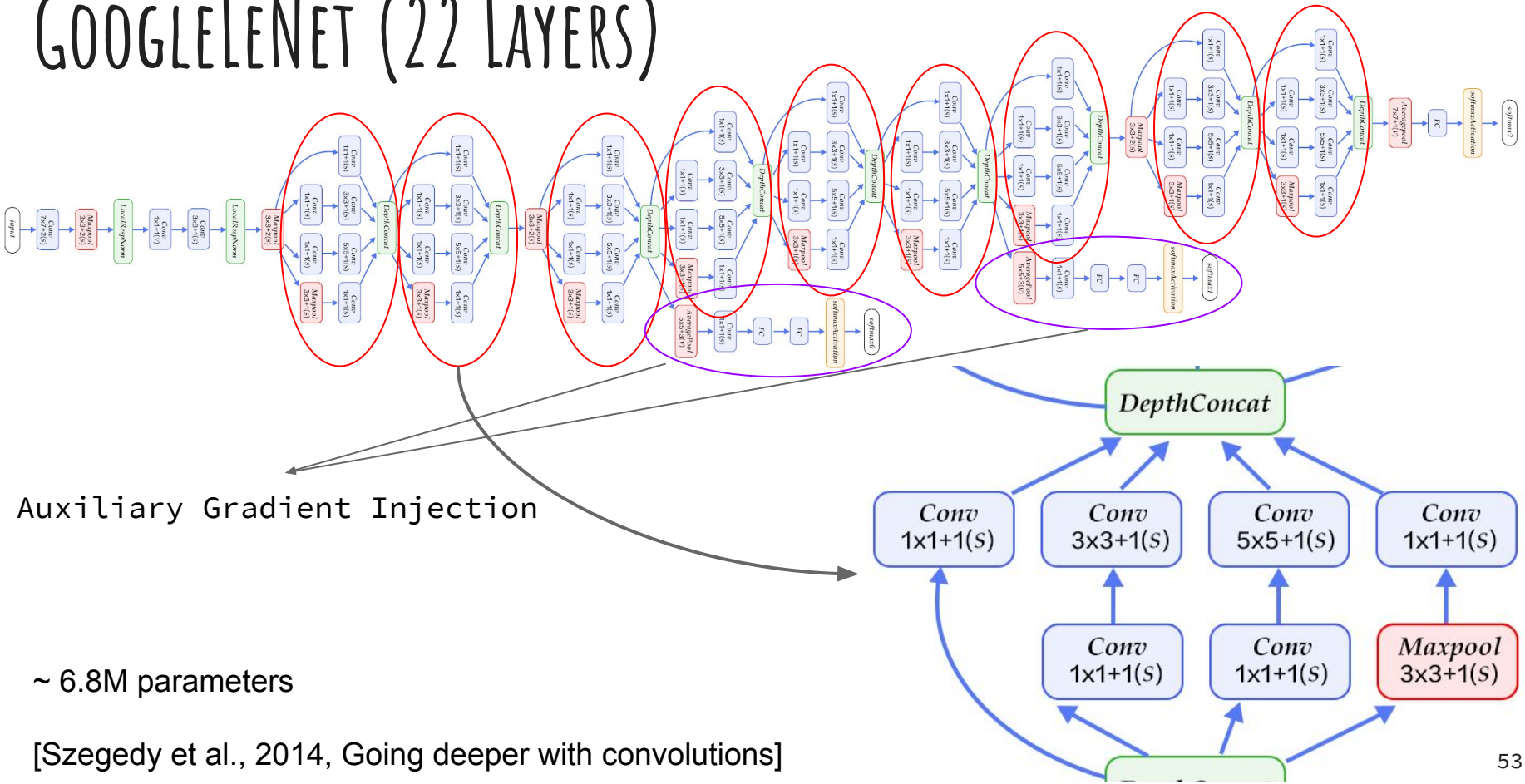
28X28X32 X 5X5X16 ~ 10M

Inception Module - Optimized

- Computation Cost
- [1X1 Conv]
  - 28X28X64X1X1X192
- [3X3 Conv]
  - 28X28X96X1X1X192
  - 28X28X128X3X3X96
- [5X5 Conv]
  - 28X28X16X1X1X192
  - 28X28X32X5X5X16
- [1X1 Conv]
  - 28X28X32X1X1X192
- 128M operations



GoogLeNet (22 Layers)

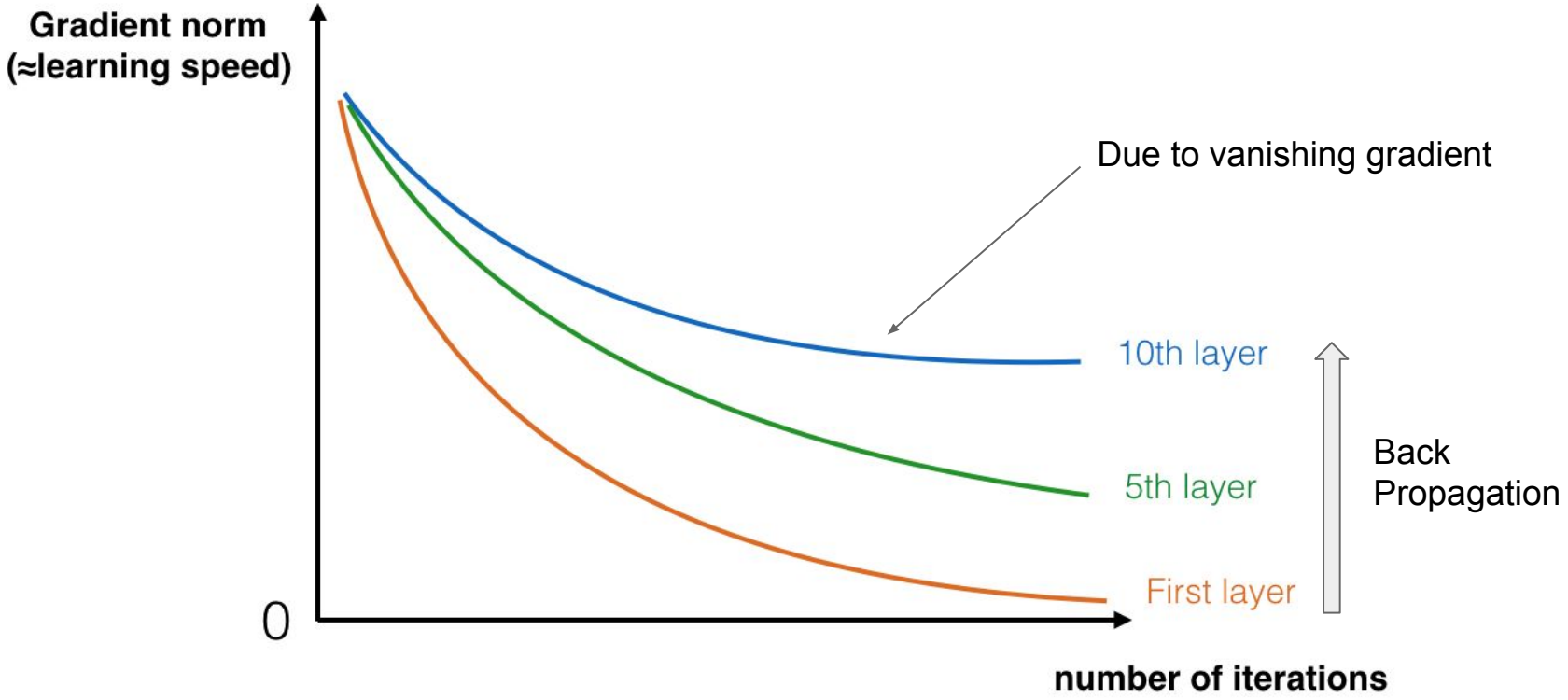


Auxiliary Gradient Injection

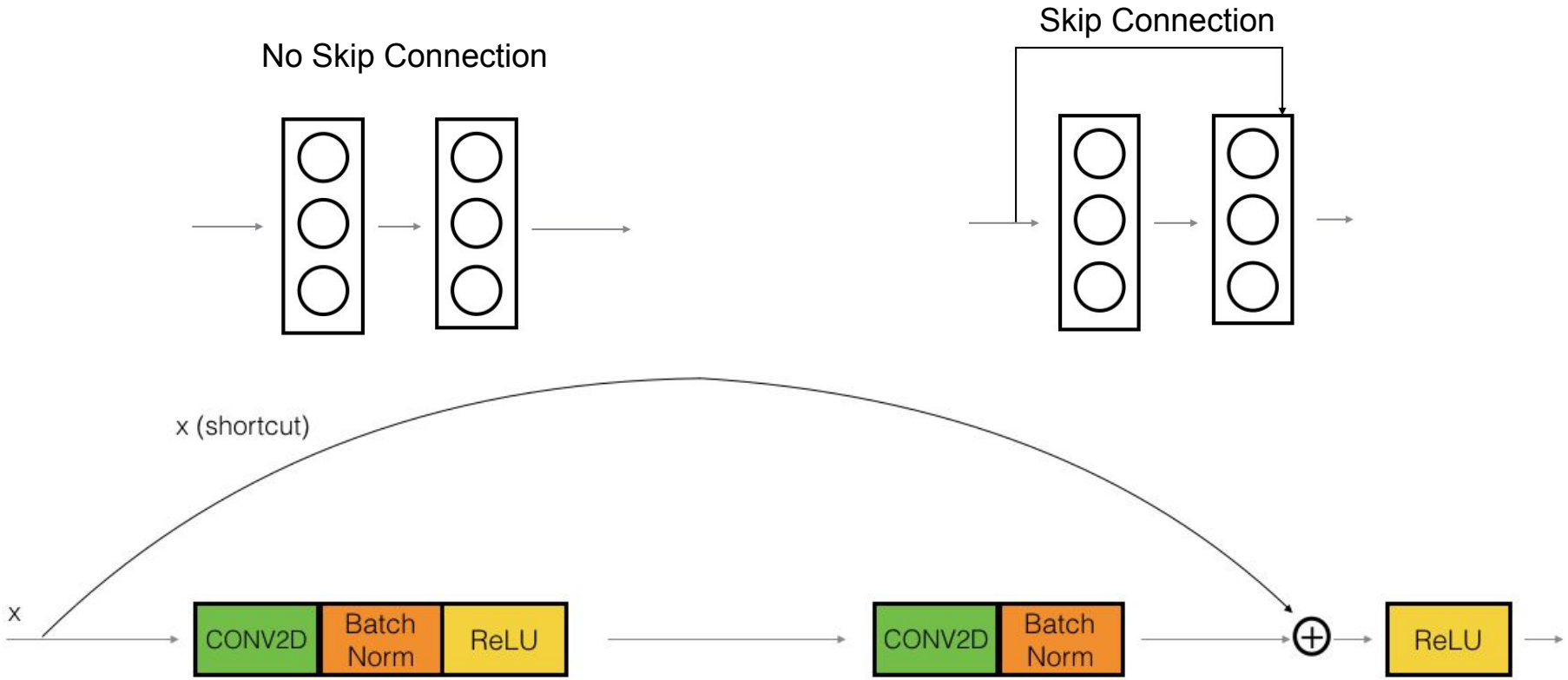
~ 6.8M parameters

[Szegedy et al., 2014, Going deeper with convolutions]

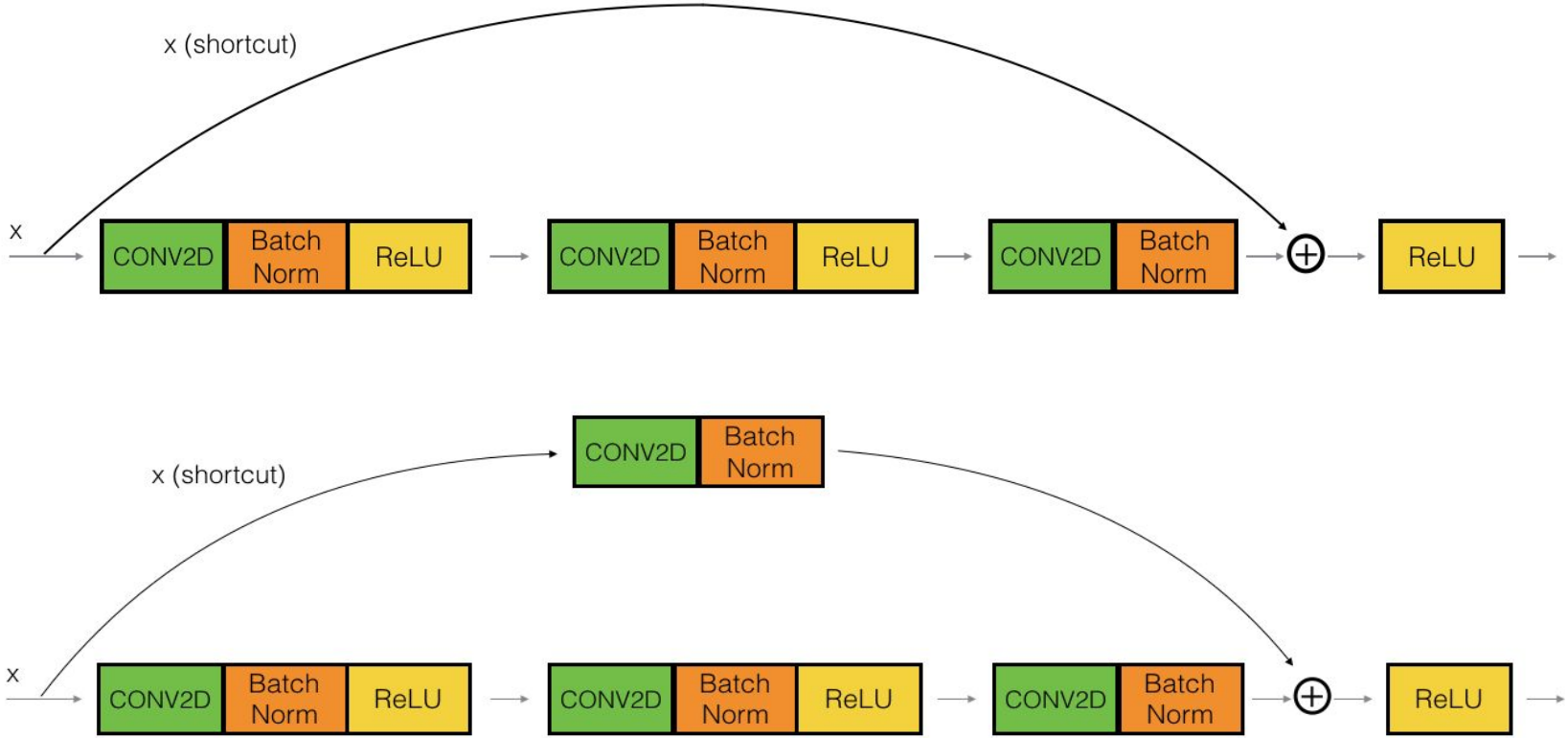
Problem with very deep neural networks



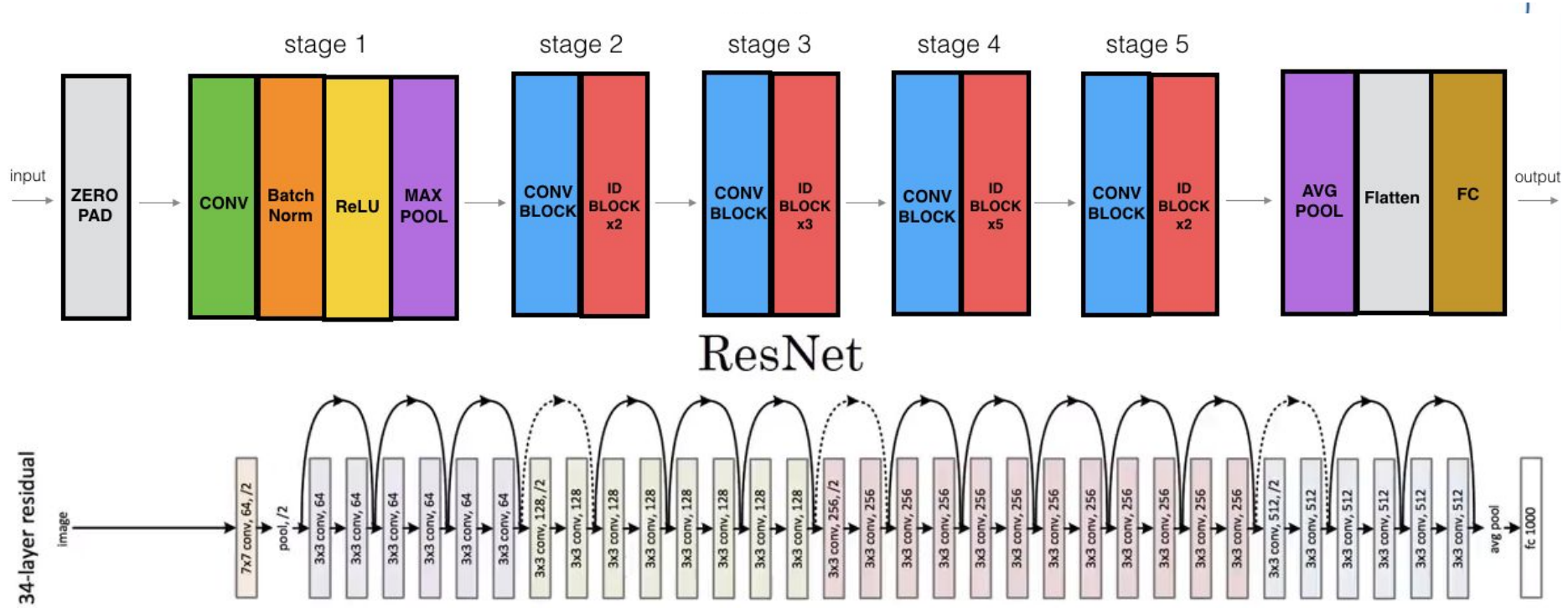
ResNet Identity Block



More Types of Resnet Identity Blocks



ResNet 50 Model



[He et. al., 2015, Deep Residual Learning for Image Recognition]

Practical Training Technique II - Transfer Learning

- Use open source model implementation
  - Architecture of networks often published by authors
  - Can use the whole model or a part of it
- Use pre-trained model weights
  - Use them as it is or as a starting point instead of random initialization then fine tune
- Utilizes prior efforts on time consuming steps
  - Creating extensive training data
  - Training using expensive computation resources such as multiple GPUs
  - Hyperparameter tuning

Transfer Learning

- Transfer learning works well in Computer Vision
- Early/mid layers of CNN perform generic image processing and feature extraction
  - Basic constituent building blocks such as edges, corners, blobs, tend to apply across multiple applications
  - ‘Freeze’ these layers, train rest of the network
- Train a small alternative network with the output of ‘frozen’ layers for your own needs
  - Usually Fully Connected
  - May also be Convolutional
- How many layers we ‘unfreeze’ ?
  - Similarities / differences with new application
  - Amount of new data available

Thank You