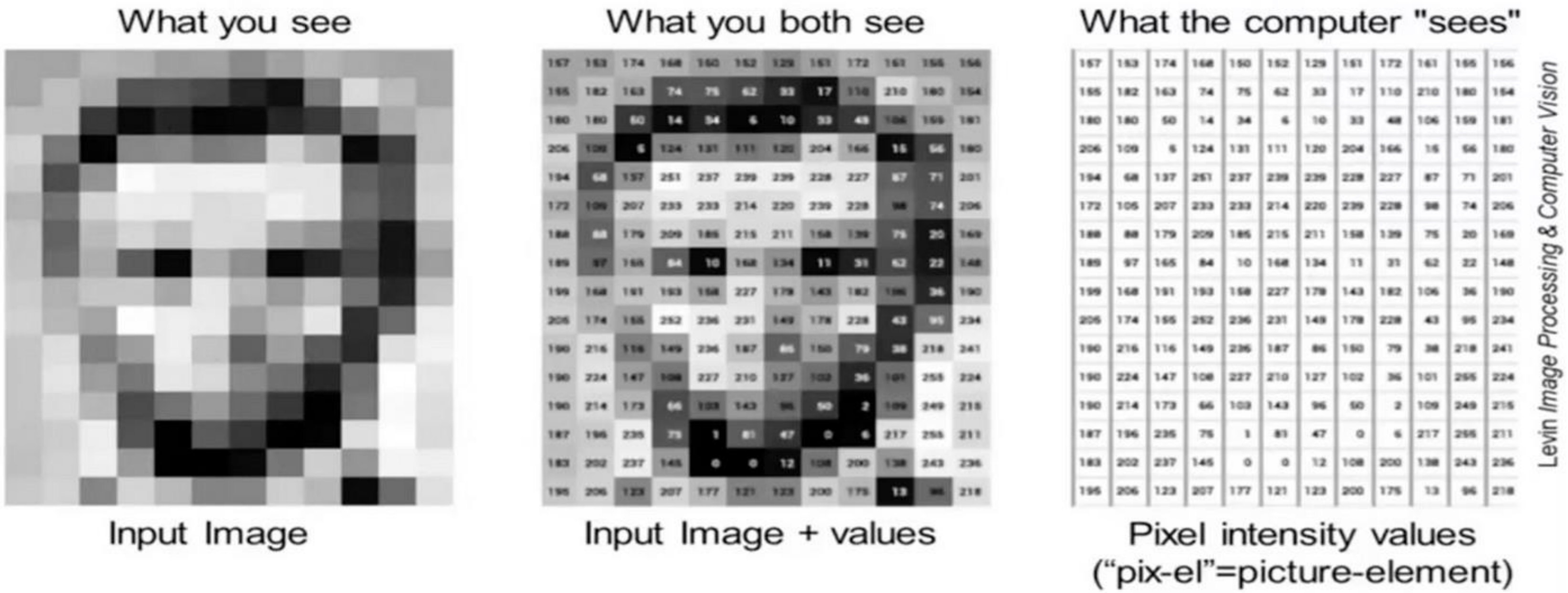
Convolutional neural network (CNN)

HESAM HOSSEINI

SUMMER 2024

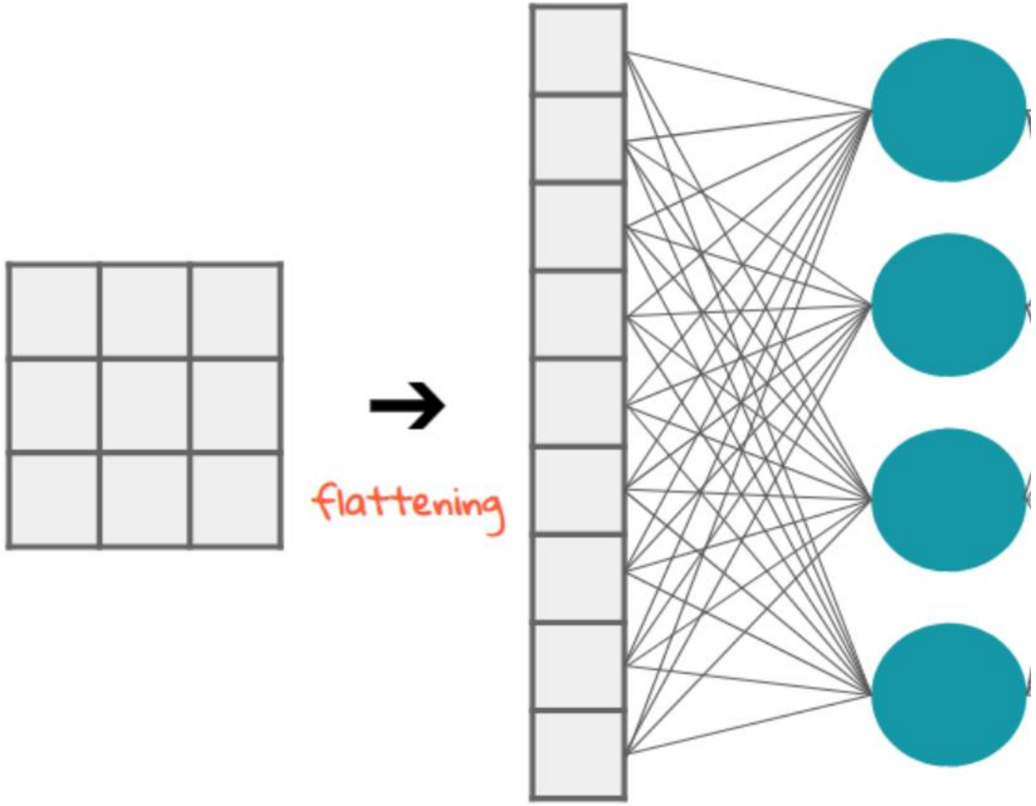
How computer sees images



An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

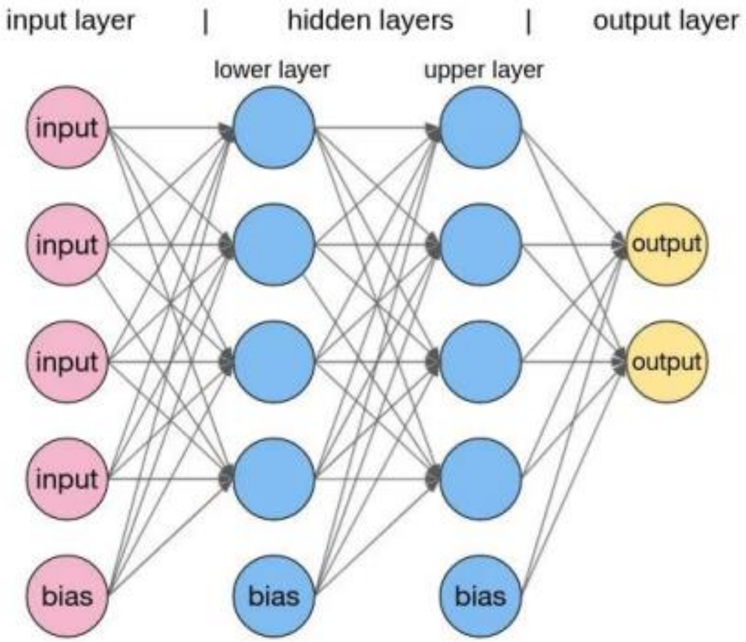
How computer sees color

How can we feed images into neural network



A Problem

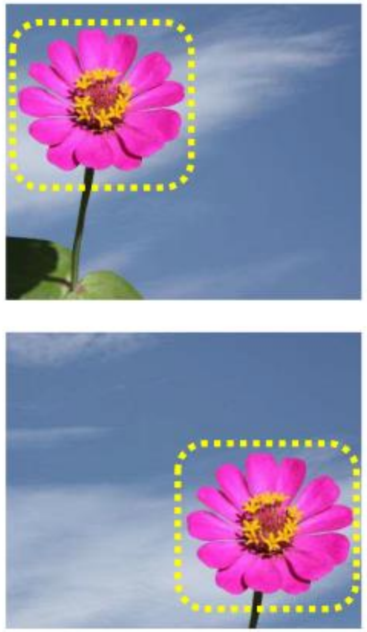
- Can a MLP identify two same types of flowers in these images?



A Problem

We need a network that will activate, regardless of the exact location of the desired object?

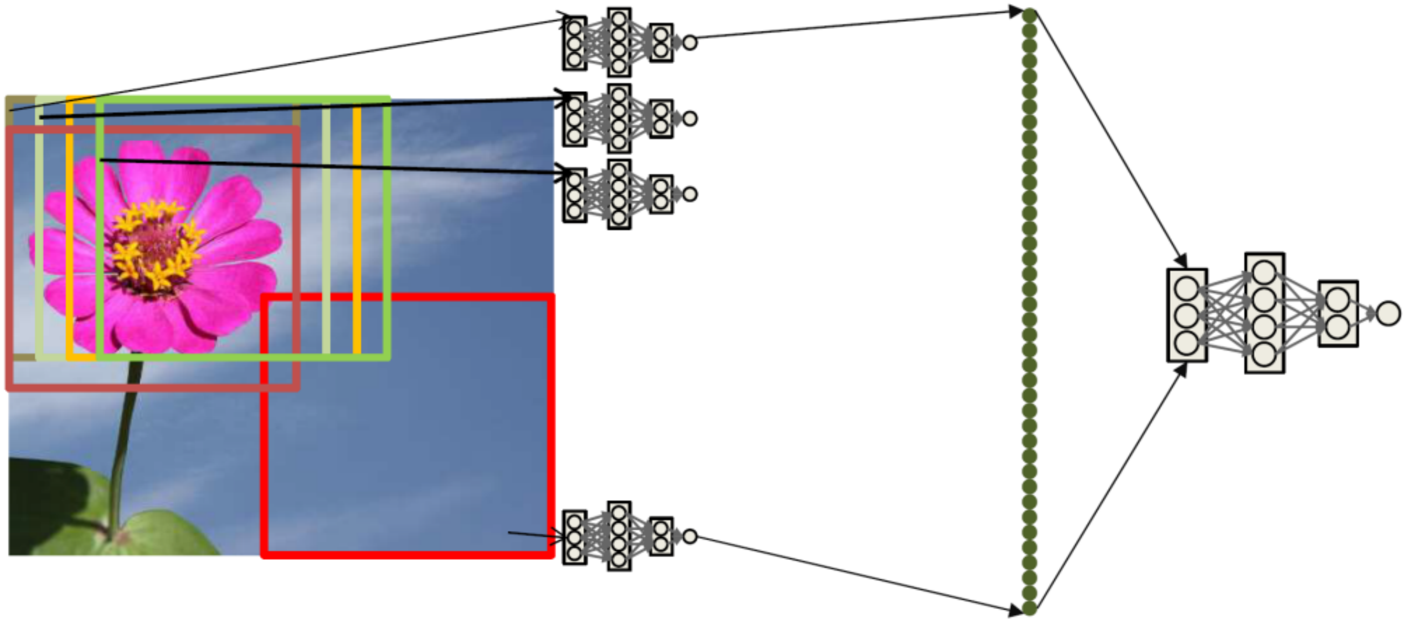
We need shift invariance.



Solution - Scanner

The entire operation can be viewed as a single giant network

- Composed of many "subnets" (one per window)
- With one key feature: all subnets are identical
- These are shared parameter networks



Limitations of FNN

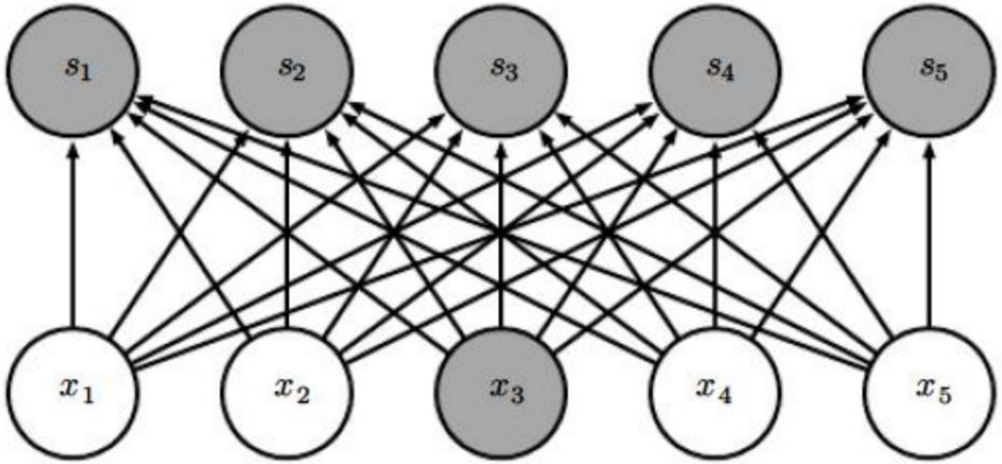
- For complex problems, we need multiple hidden layers in our FNN
  - Compunds the problem of having many weights
- Having too many weights
  - Makes learning more difficult as dimension of search space is increased
  - Makes training more time/resource consuming
  - Increases the likelihood of overfitting
- Problem is further compunded for color images
  - Each pixel in color image represented by 3 values (RGB color mode)
  - Since each pixel represented by 3 values, we say *channel* size is 3
  - Image represented by 64 x 64 x 3 = 12,288 values (rows x columns x channels)
  - Number of weights is now 12,288 x 500 = 6,144,000

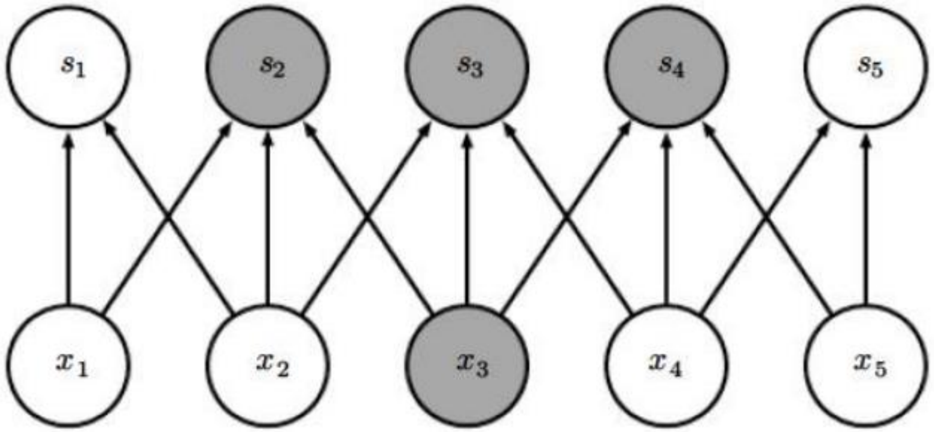
Limitations of FNN

- Clear that FNN cannot scale to larger images (Too many weights)
- Another problem with FNN
  - 2D image represented as 1D vector in input layer
  - Any spatial relationship in the data is ignored

From FCN to CNN

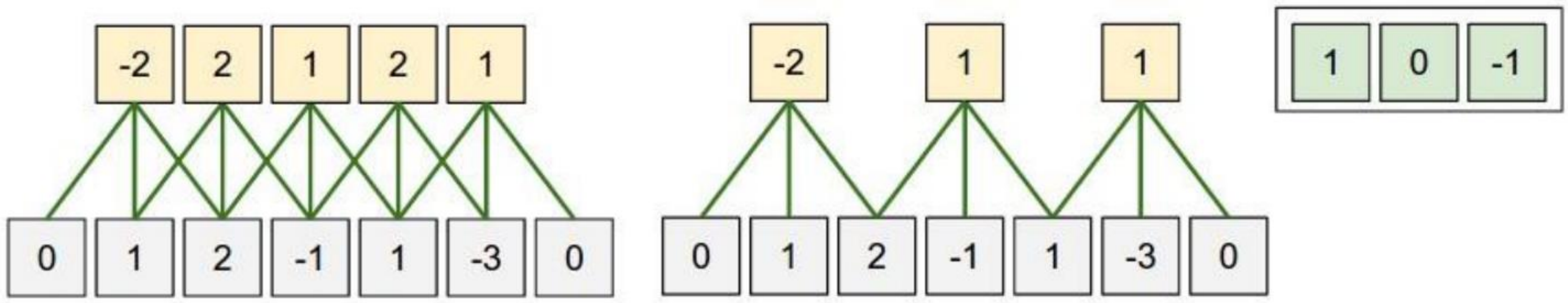
- Sparse Connection/interaction





How to reduce number of parameters?

Parameter Sharing (plus Stride):



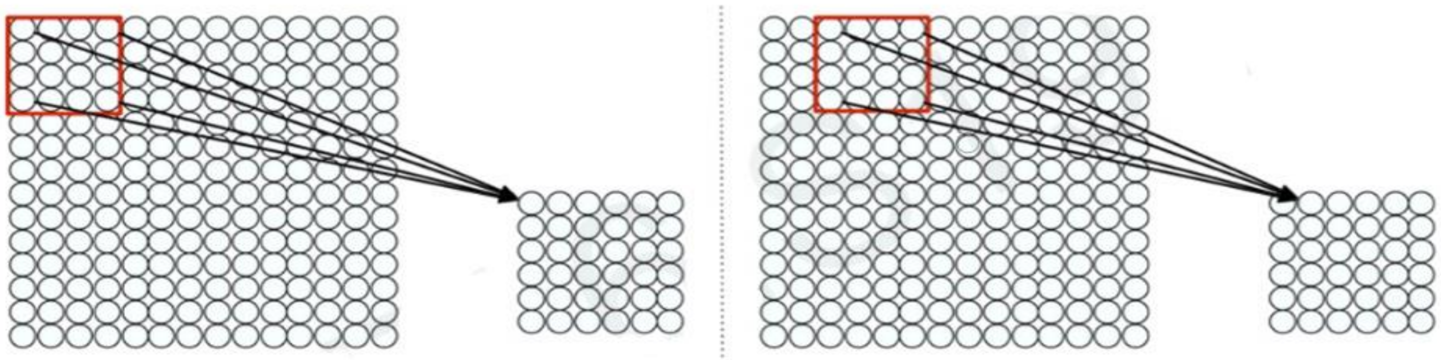
Convolution/Correlation

- Linear Convolution (Linear Shift Invariant Systems)

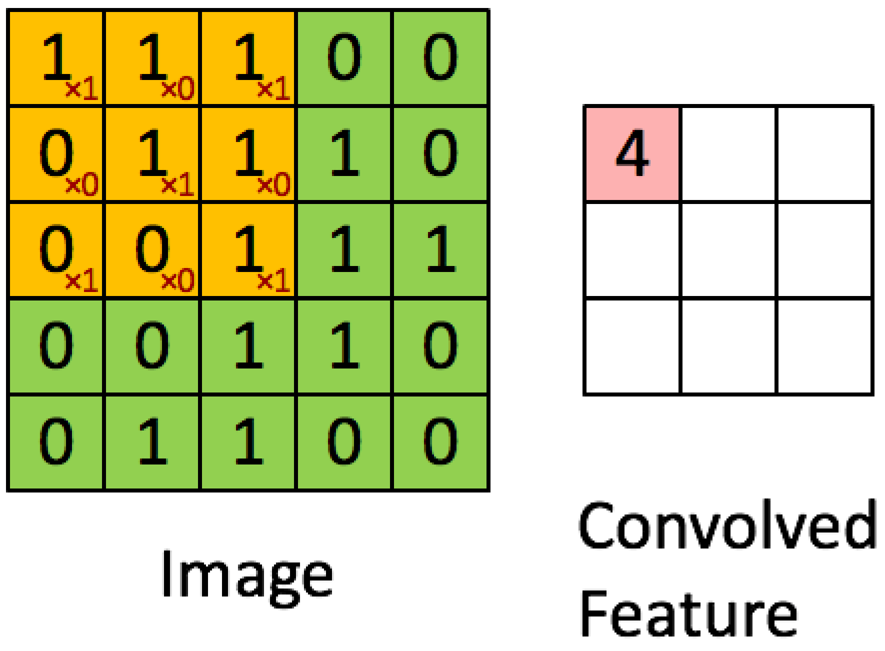
$$y[n] = \sum_{m} x[m]\, h[n-m] = \sum_{k} h[k]\, x[n-k]$$

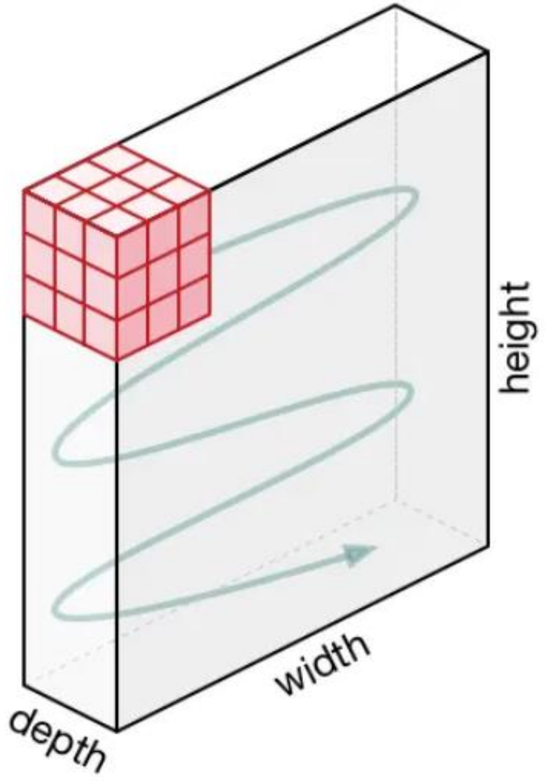
- Correlation:

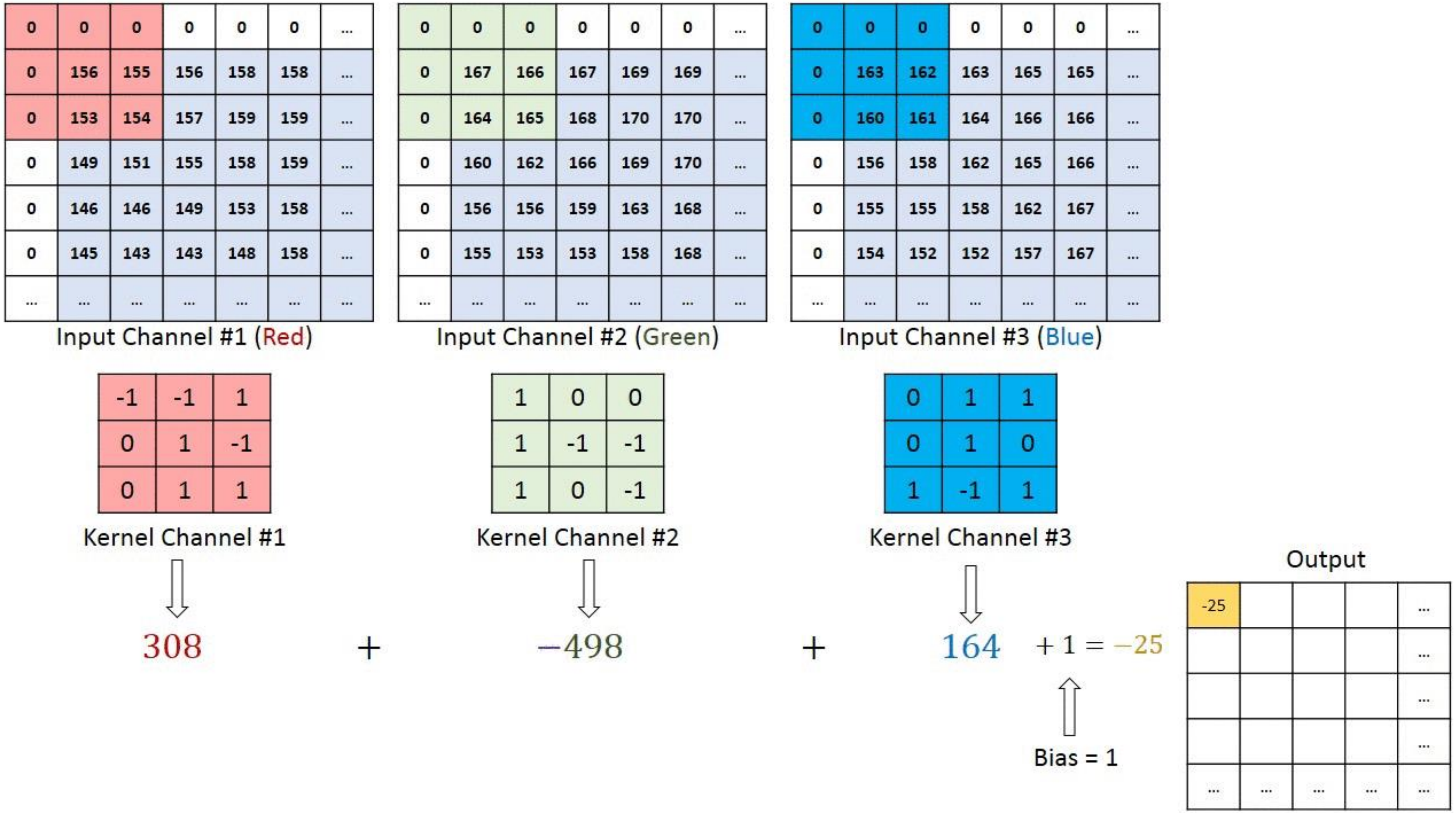
$$y[n] = \sum_{m} x[m]\, h[n+m] = \sum_{k} h[k]\, x[n+k]$$



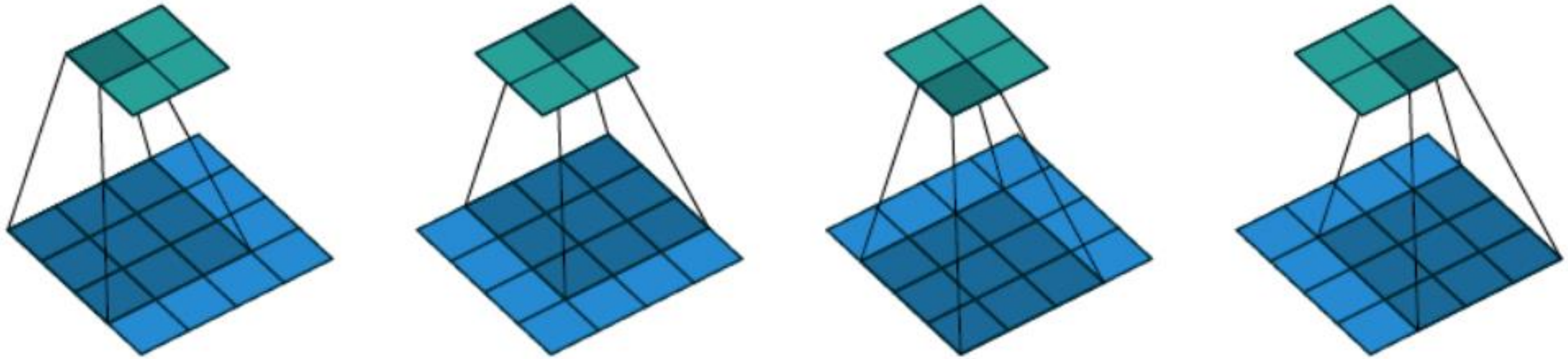
Convolution



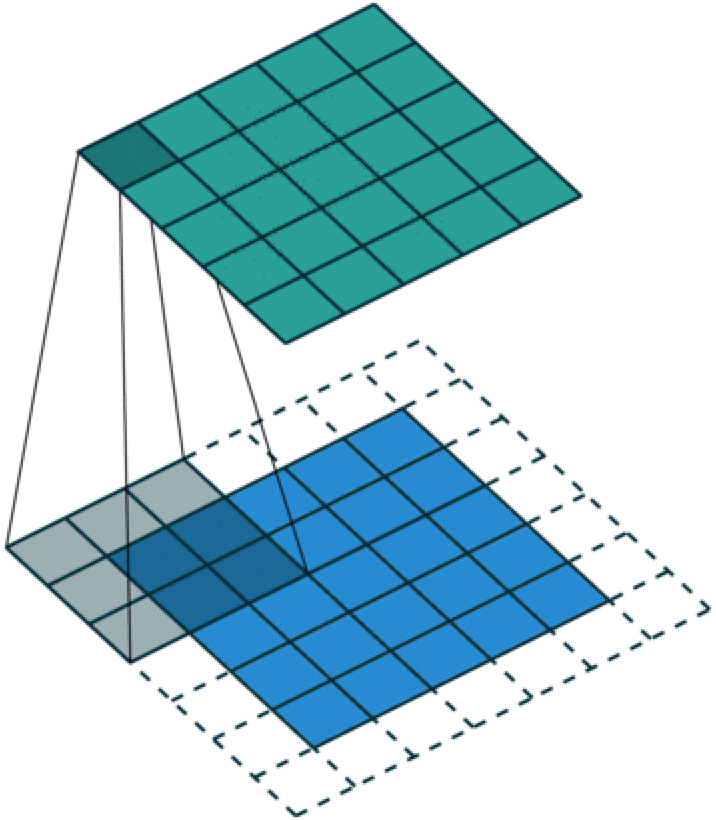



Input Channel #1 (Red)
Input Channel #2 (Green)
Input Channel #3 (Blue)
Kernel Channel #1
Kernel Channel #2
Kernel Channel #3
Output
-25
308 + −498 + 164 + 1 = −25
Bias = 1

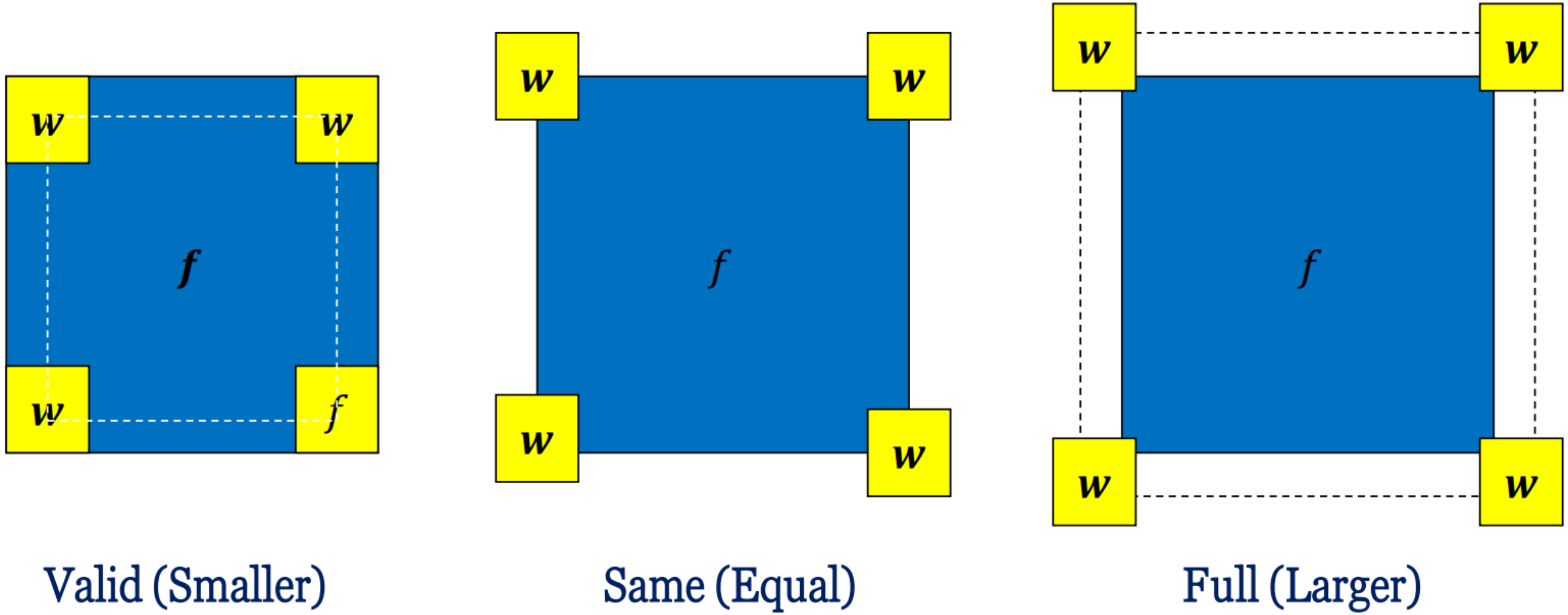
Convolution shrinks



padding

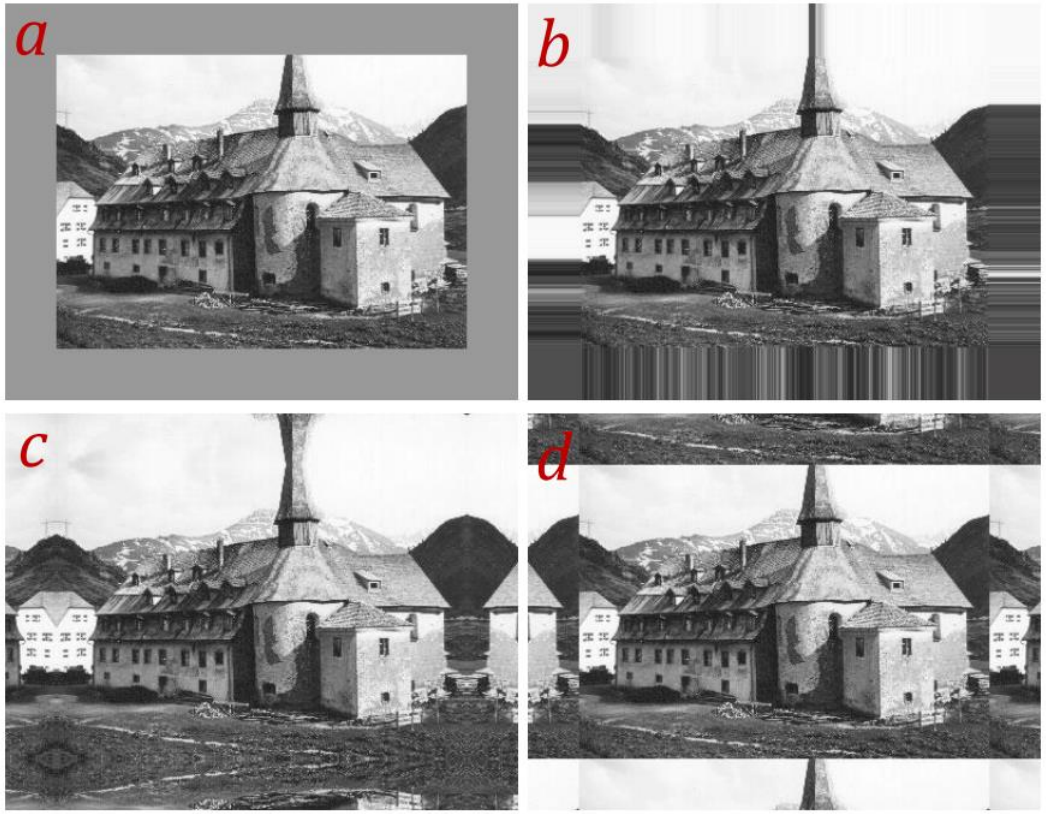


Padding

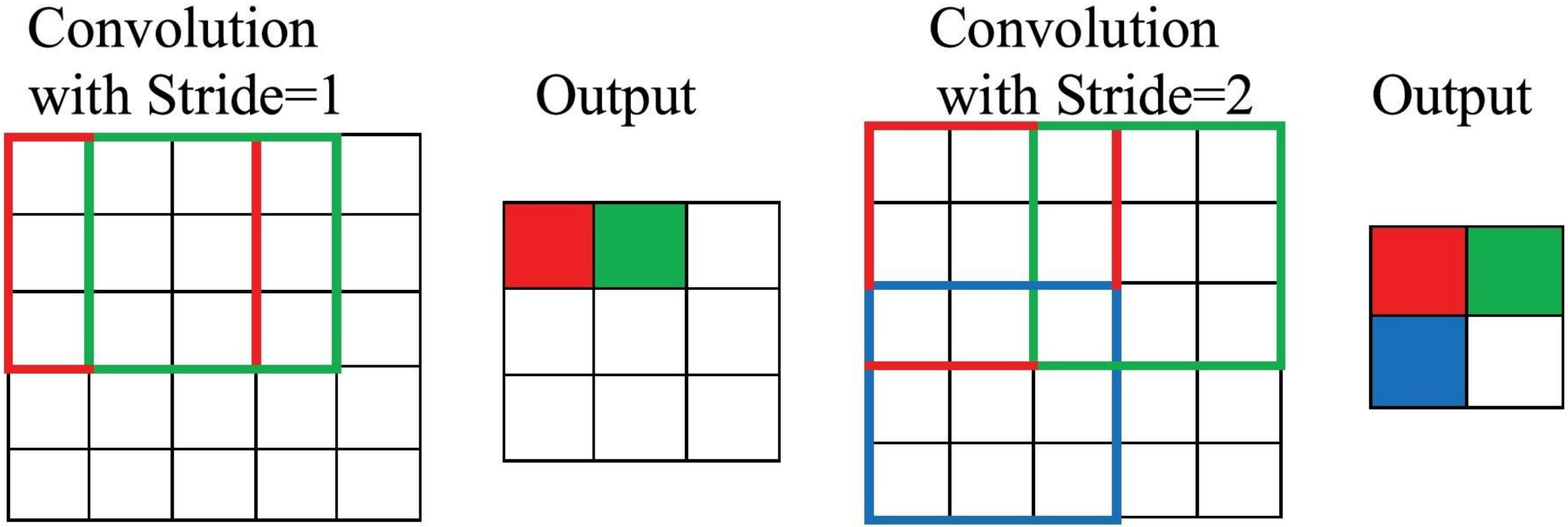


Padding

a) Constant
b) Replicate (nearest)
c) Symmetric (mirror)
d) Circular



Stride


Convolution
with Stride=1
Output
Convolution
with Stride=2
Output

Stride and padding

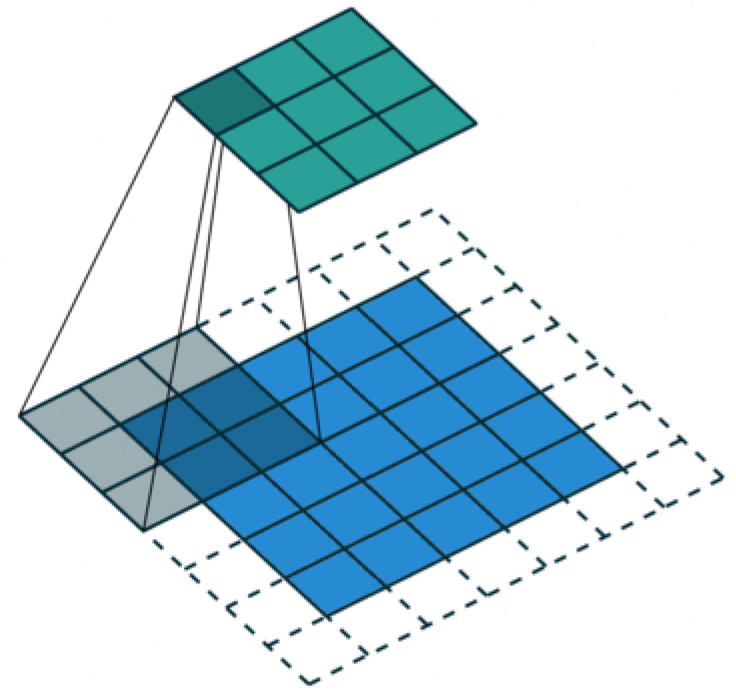
$$M = \frac{N + 2P - f}{stride} + 1$$

$Input = N \times N$

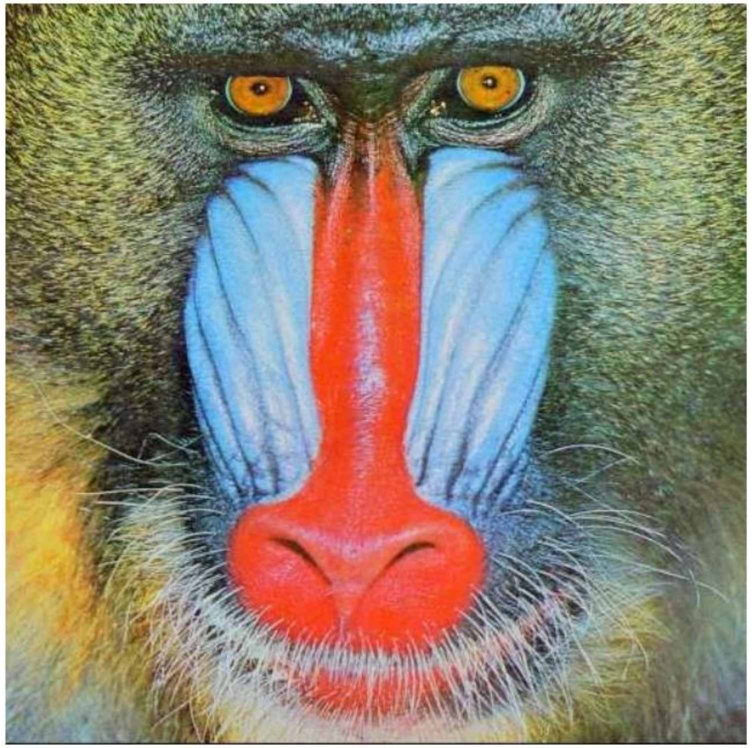
$Padded\ Input = (N + 2P) \times (N + 2P)$

$Filter = f \times f$

$Output = M \times M$

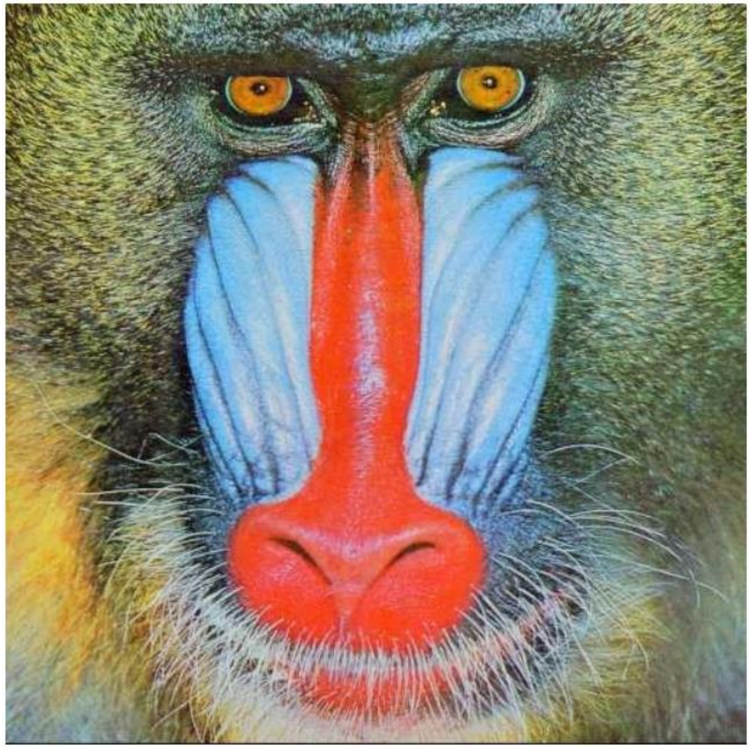


Convolution Effect!

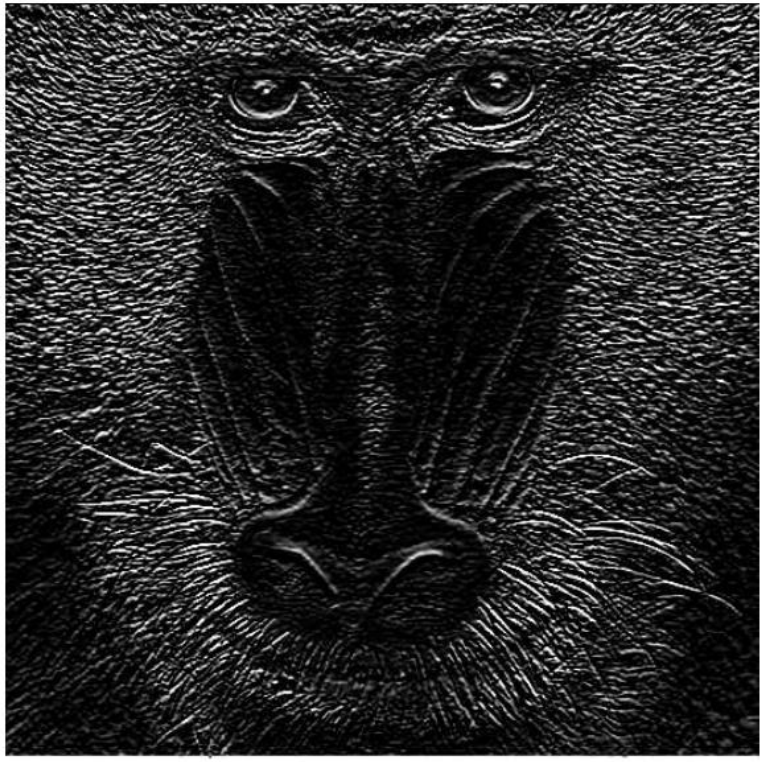


$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

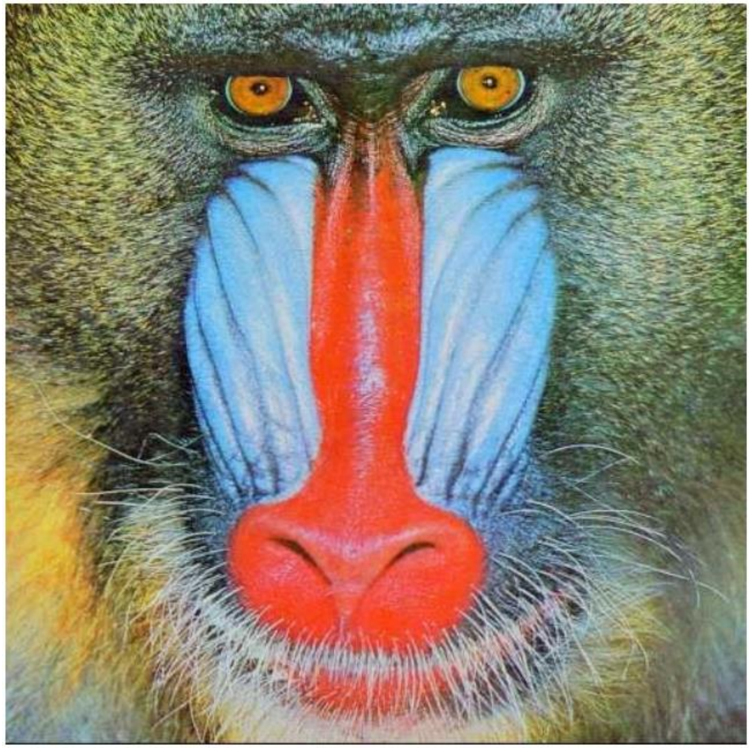
Convolution Effect!



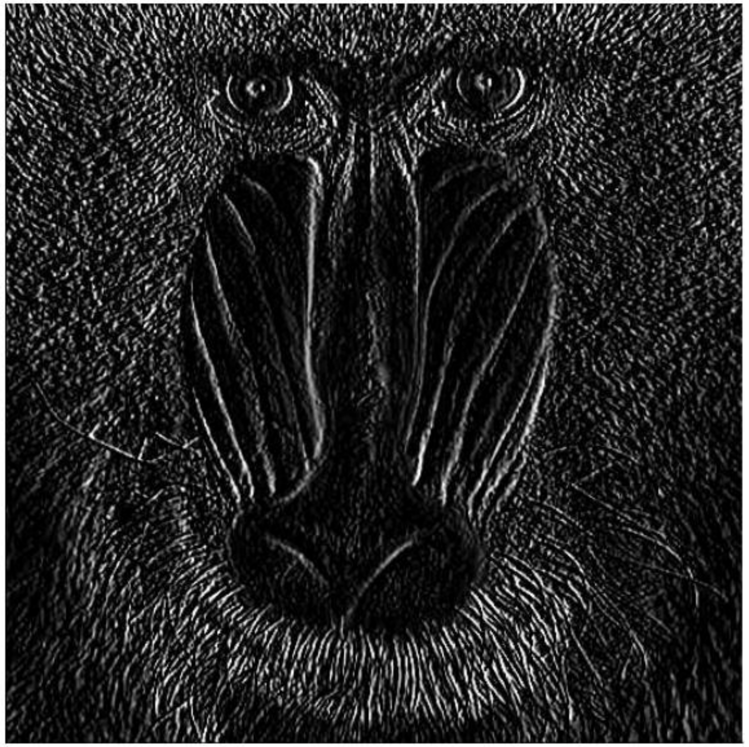
$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$



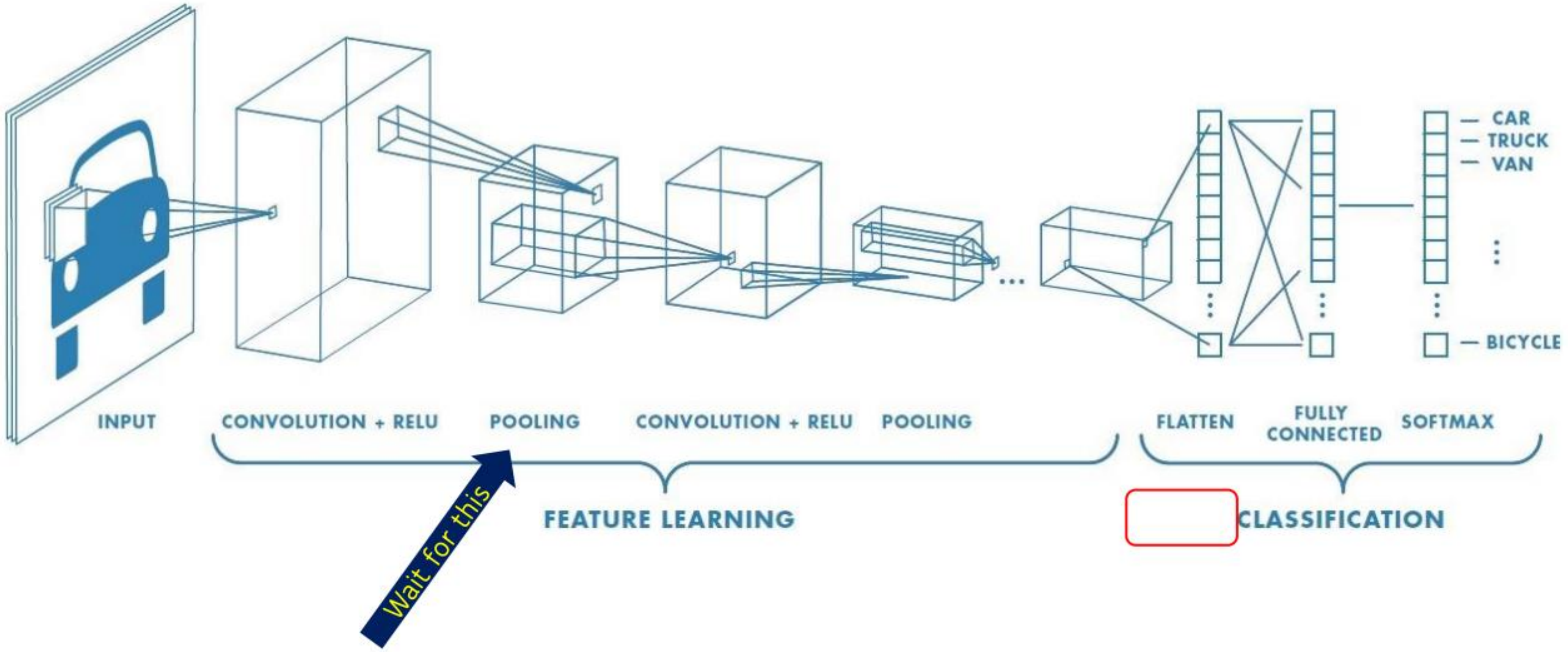
Convolution Effect!



$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$



A Typical CNN

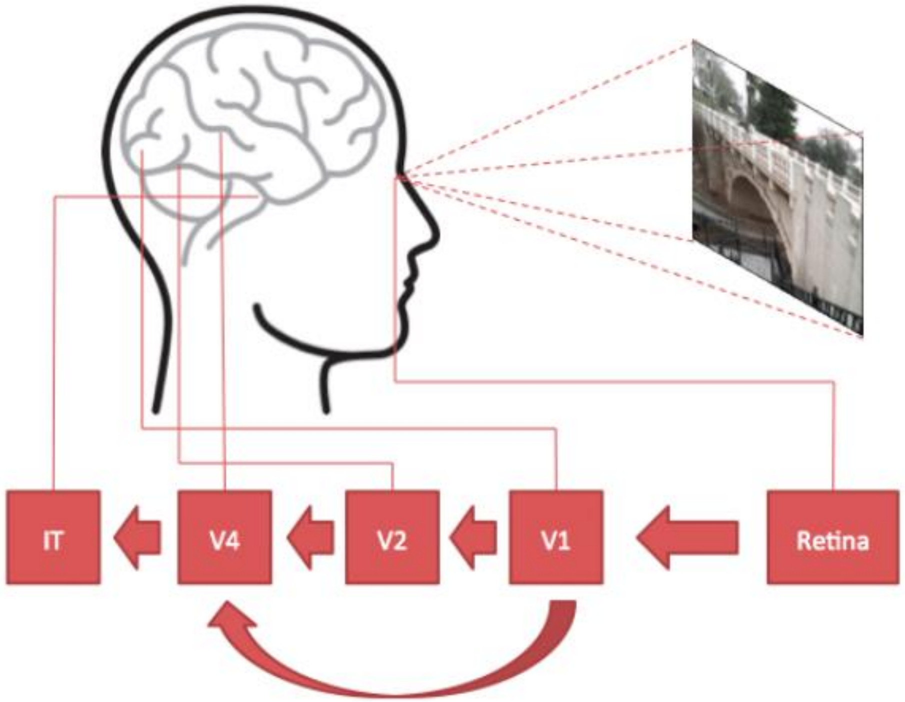


The Neuroscientific Basis

- In 1959 Hubel & Wiesel did an experiment to understand how visual cortex of brain processes visual info
  - Recorded activity of neurons in visual cortex of a cat
  - While moving a bright line in front of the cat
- Some cells fired when bright line is shown at a particular angle/location
  - Called these *simple* cells
- Other cells fired when bright line was shown regardless of angle/location
  - Seemed to detect movement
  - Called these *complex* cells
- Seemed complex cells receive inputs from multiple simple cells
  - Have an hierarchical structurea
- Hubel and Wiesel won Noble prize in 1981

The Neuroscientific Basis

Primary Visual Cortex, Theory: from 1959-1985



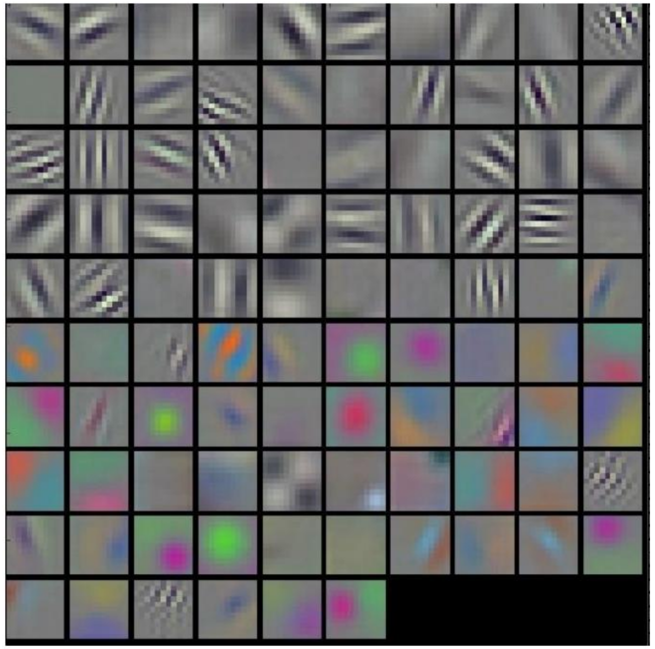
V1: Edge detection, etc.

V2: Extract simple visual properties (orientation, spatial frequency, color, etc)

V4: Detect object feature of intermediate complexity

TI: Object recognition

CNN Visualization



Filter or weights

Baseball—or stripes?
mixed4a, Unit 6

Animal faces—or snouts?
mixed4a, Unit 240

Clouds—or fluffiness?
mixed4a, Unit 453

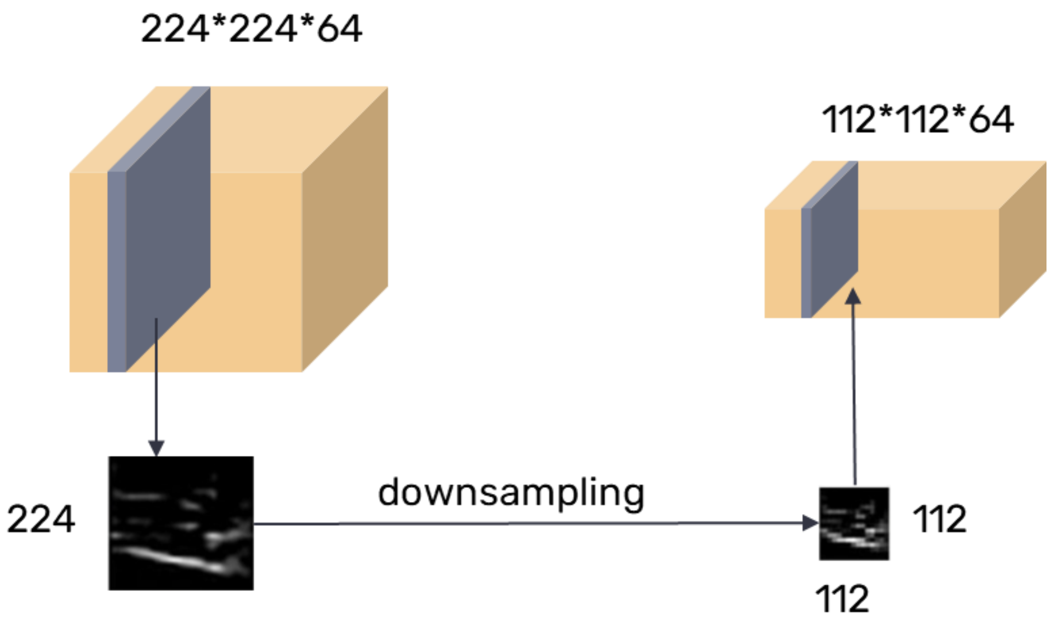
Buildings—or sky?
mixed4a, Unit 492

Pooling

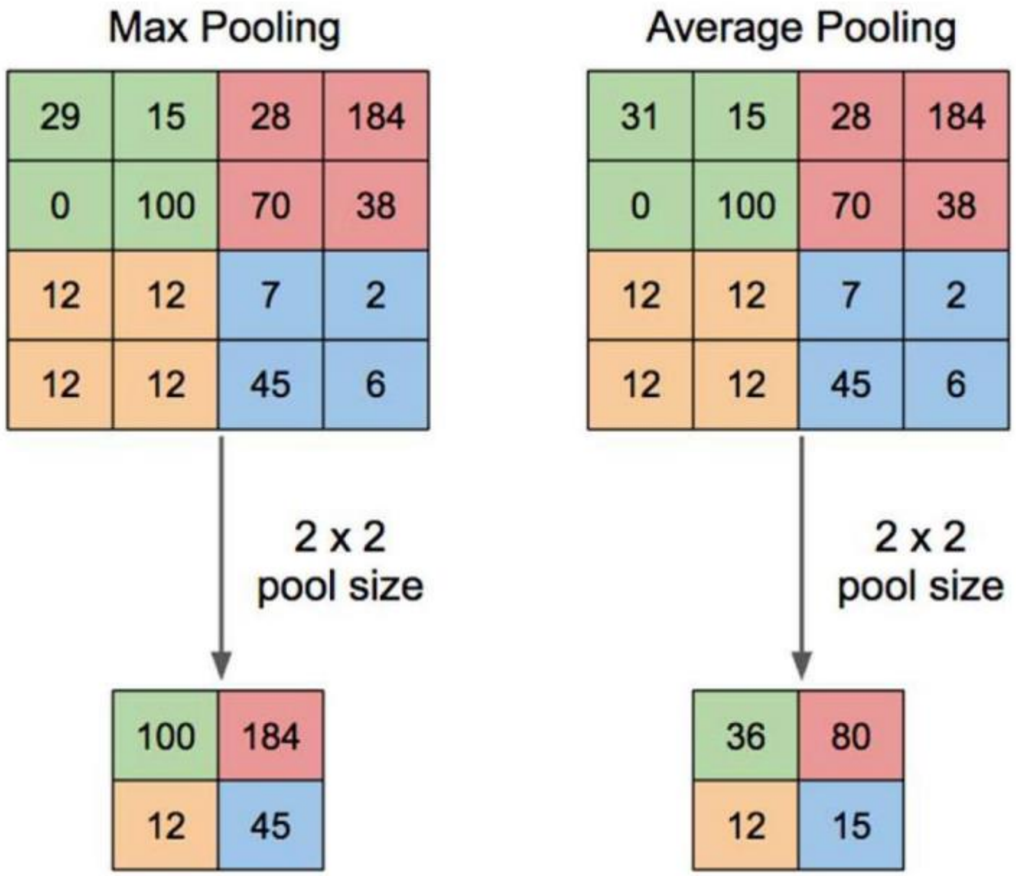
Modify and Subsample output of detector (ReLu):

Using neighbors
- Max/min pooling
- Means pooling
- Median pooling

- Make Invariant to small variation (like shift)
- Reduce number of parameters
- Regularization



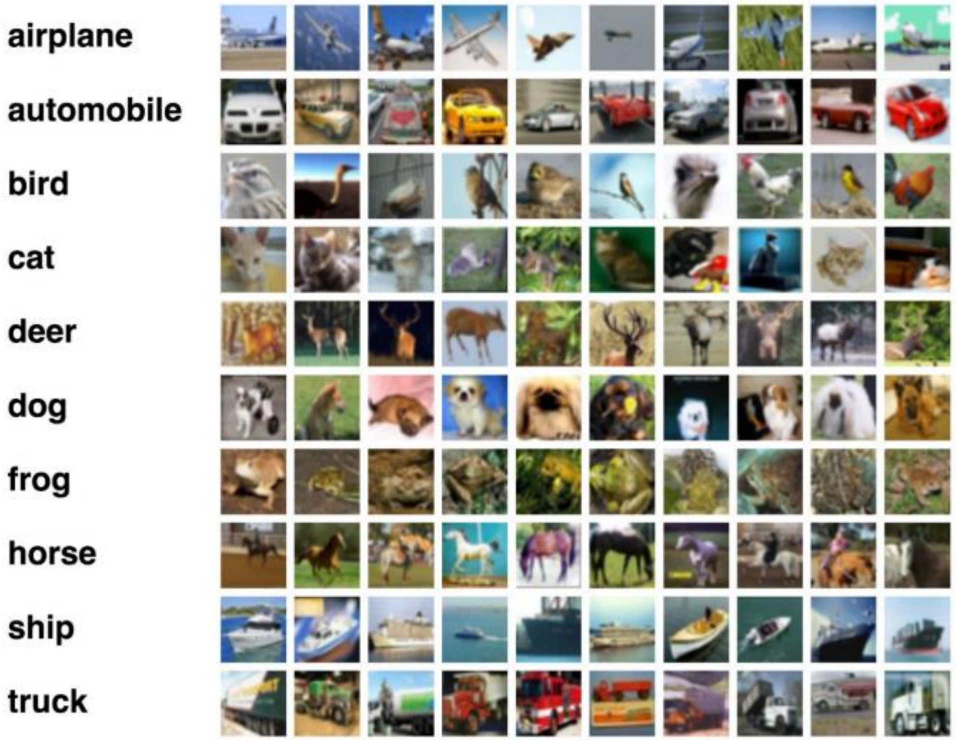
Max Pooling vs Mean (Average) Pooling:



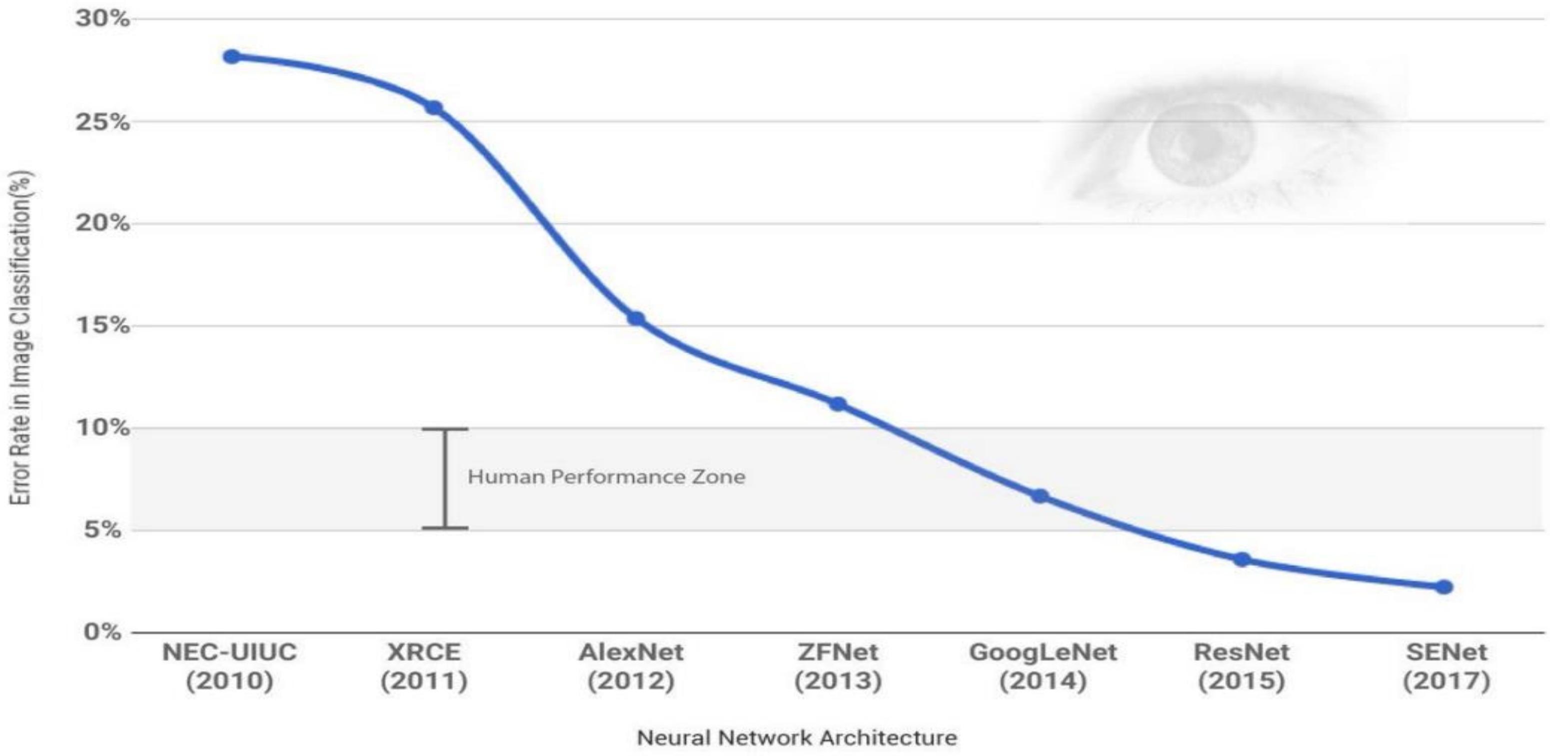
A Challenging Problem: IMAGENET!

Image Net

- 1000 Classes
- Train: 1.2 M
- test: 100 K



performance on Image net



AlexNet(2012)

In Brief:

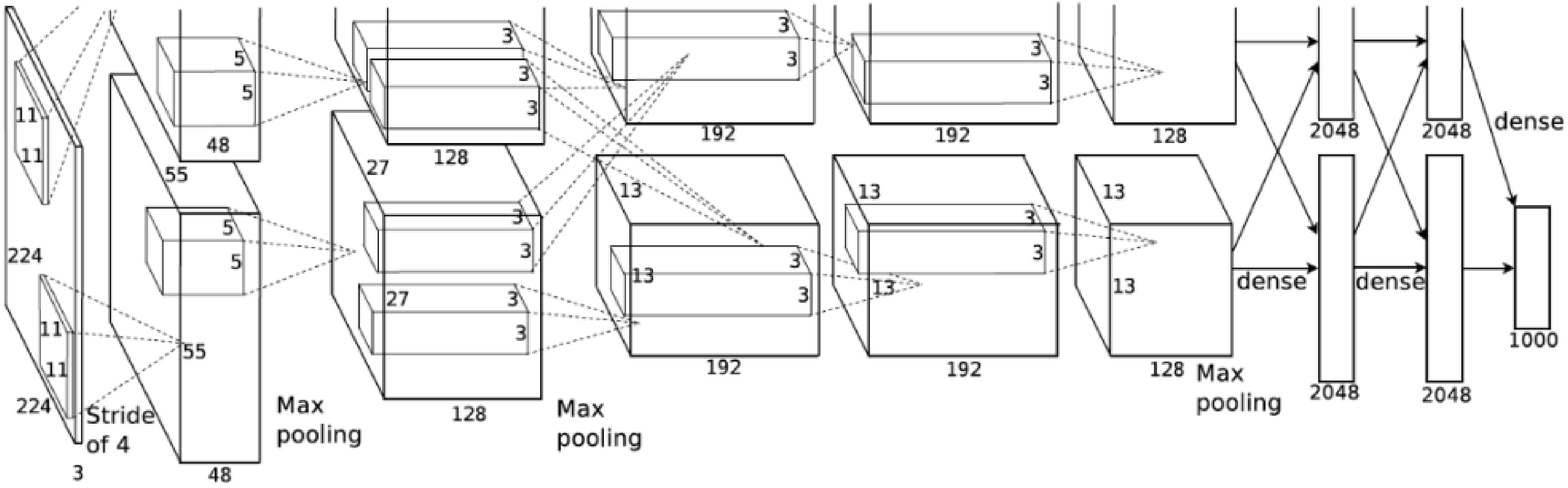
- ReLU (x6 faster than tanh)
- Regularization: 50% Dropout (x2 training time)
- 5 CL (Convolution Layer), 11×11, 5×5, 3×3
- 3 FC (Fully Connected), 4096-4096-1000
- MaxPool Layer: 3×3
- ReLu after all CL and FC
- Image Size: 256x256
- input Size: 227x227x3

Details:

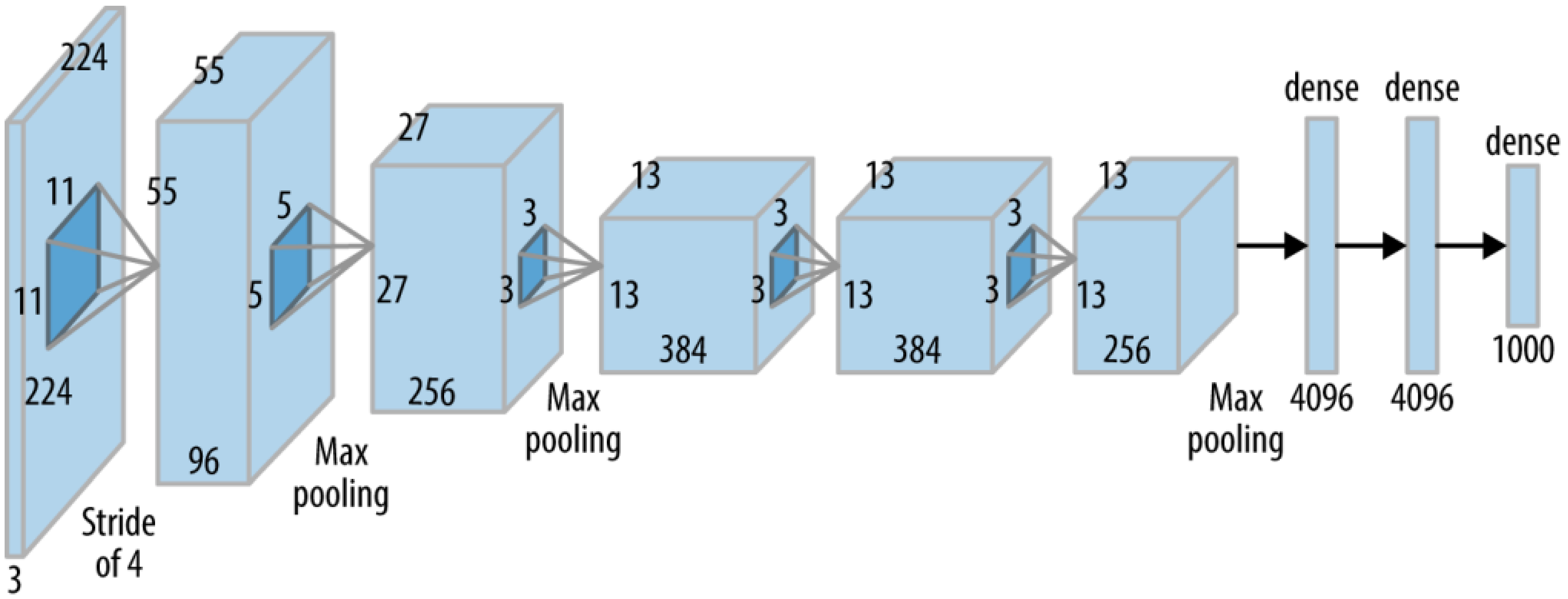
- First use of ReLU
- Normalize ReLu Output
- Several Data Augmentation
- Dropout: 0.5
- Batch Size: 128
- SGD Momentum: 0.9
- Learning rate 10−2
- 60M parameters

AlexNet

Original (paper) Figure:



AlexNet

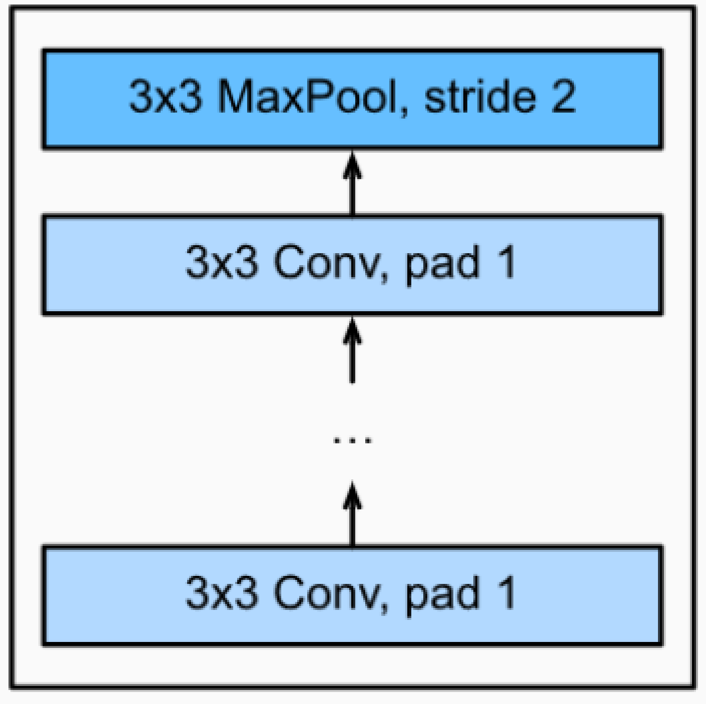

224
55
27
13
13
13
dense
dense
dense
11
55
5
3
3
3
11
5
27
3
13
3
13
3
13
384
384
256
1000
224
256
Max
pooling
4096
4096
Max
pooling
96
Max
pooling
Stride
of 4
3

VGGnet

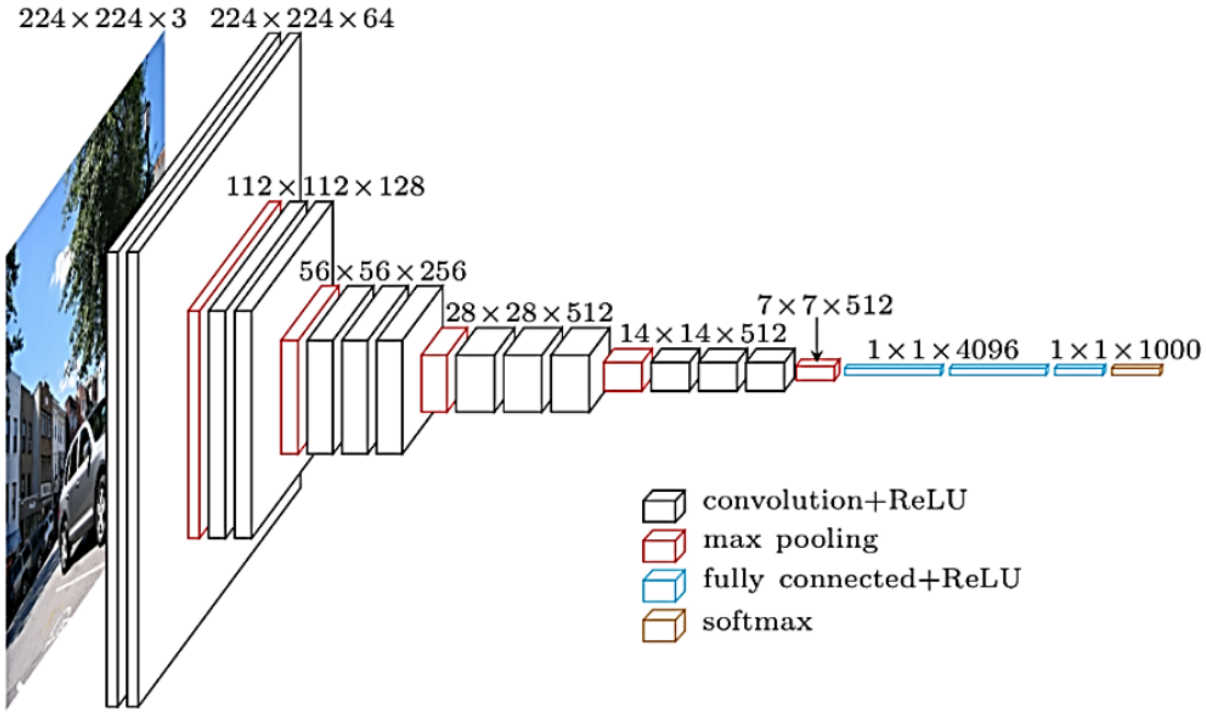
VGG: Visual Geometry Group

Idea: Smaller filter but deeper!

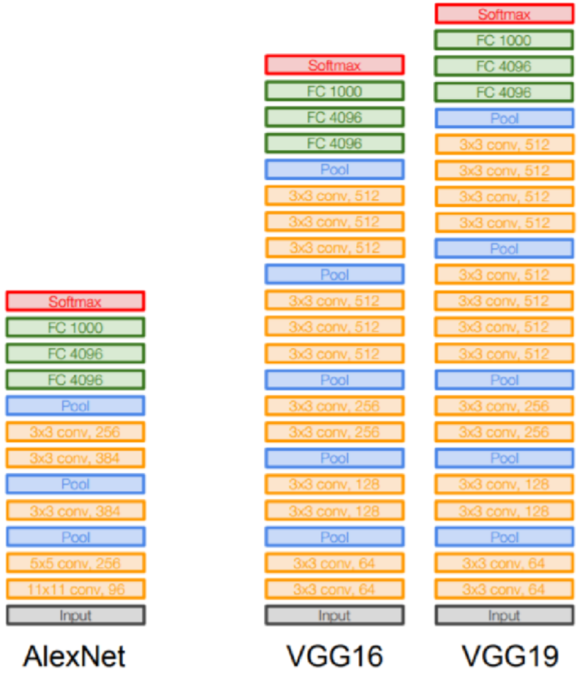
- 8 Layers in Alexnet → (16-19) ConvLayers
- ConvLayer (16/19): Only 3×3, Stride: 1, Padding: 1
  - More Non-linearity
- MaxPooling (5): 2×2, Stride: 2
- 138M Parameters



VGGnet



224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax



Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 256
3x3 conv, 384
Pool
3x3 conv, 384
Pool
5x5 conv, 256
11x11 conv, 96
Input
AlexNet
Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input
VGG16
Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input
VGG19

GoogLeNet

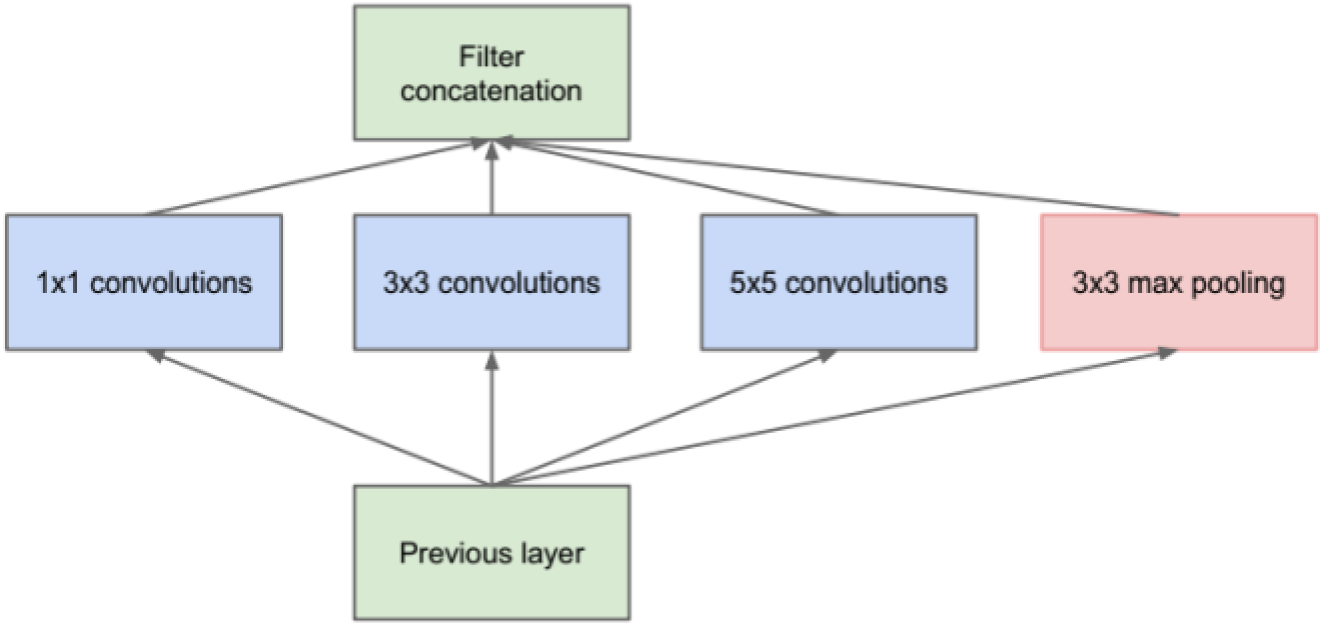
Inception Module

Idea and Motivation:

- Multiresolution Analysis (avoid cascade only convolution)
- Global Average Pooling before FC (dimension reduction)
- Bottleneck Layer (one dimension reduction)
- Auxiliary Classifier (shallow/weak output)

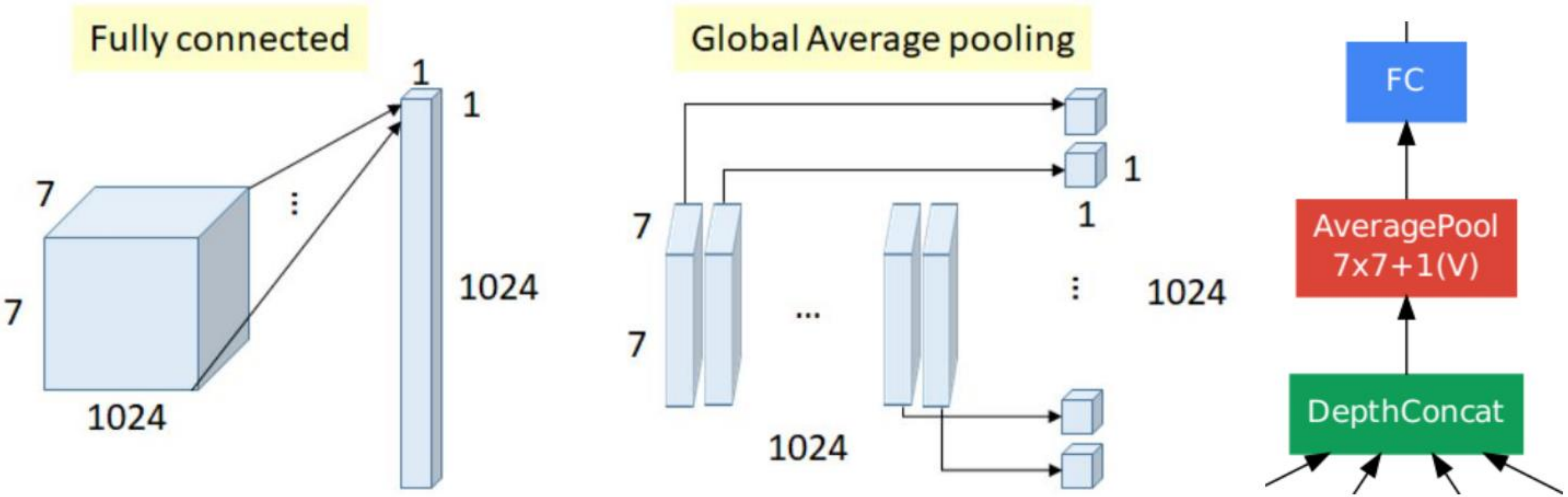
GoogLeNet

Idea#1: Why Resolution reduction in successive Layer?



GoogLeNet

Idea#2: Global Average Pooling Before FC

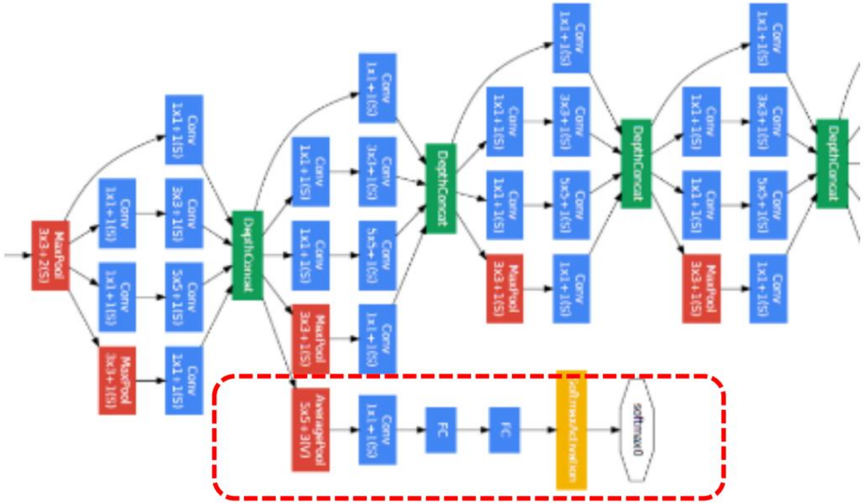


GoogLeNet

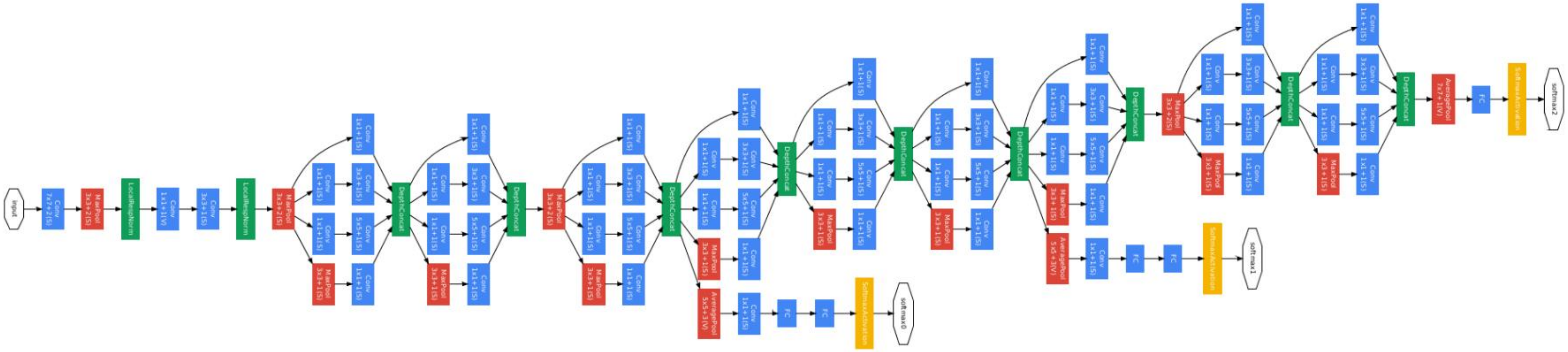
Idea #3: Auxiliary Classifier

Too Deep (22 Layers) → Gradient flow problem!

Consider Shallow Outputs!



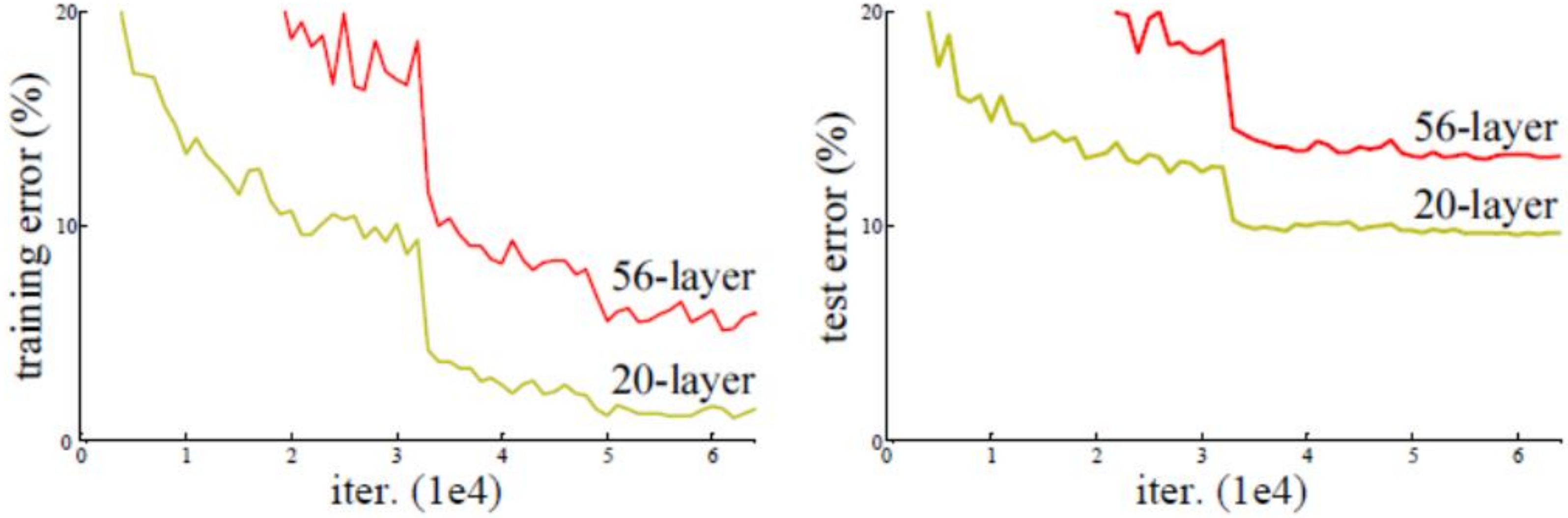
GoogLeNet



ResNet - Deep Residual Learning for Image Recognition (Microsoft)

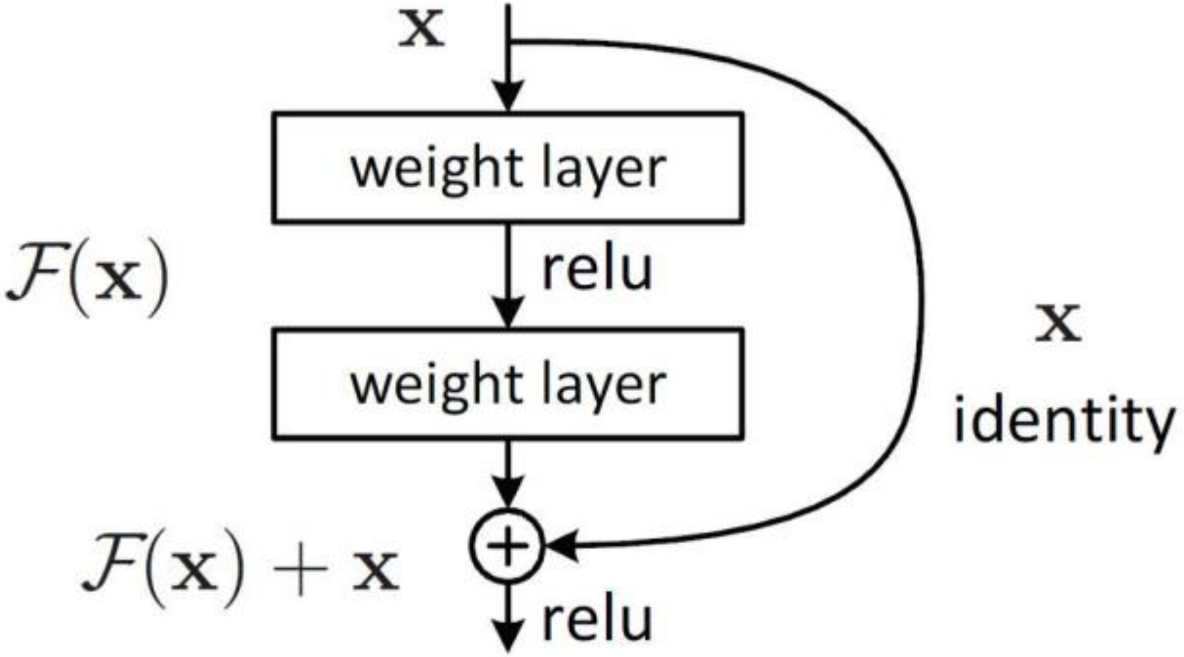
Problems with Deeper Networks:

- Performance degradation, Not due to overfitting
- Gradient vanishing/exploding



ResNet

Idea: New Block (Shortcut Connection/Skip Connections)



ResNet

A Identity map is hard to capture by highly nonlinear map.

In Residual Learning:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, W) + \mathbf{x}$$

$H(x) = F(x) + x$, the $F(x)$ may learn to be zero!

- No Extra Parameter! ☺
- No Computational Complexity! ☺
- Problem of dimension F and X
  - Solution:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, W) + W_s\mathbf{x}$$

ResNet

ResNet Benefits:

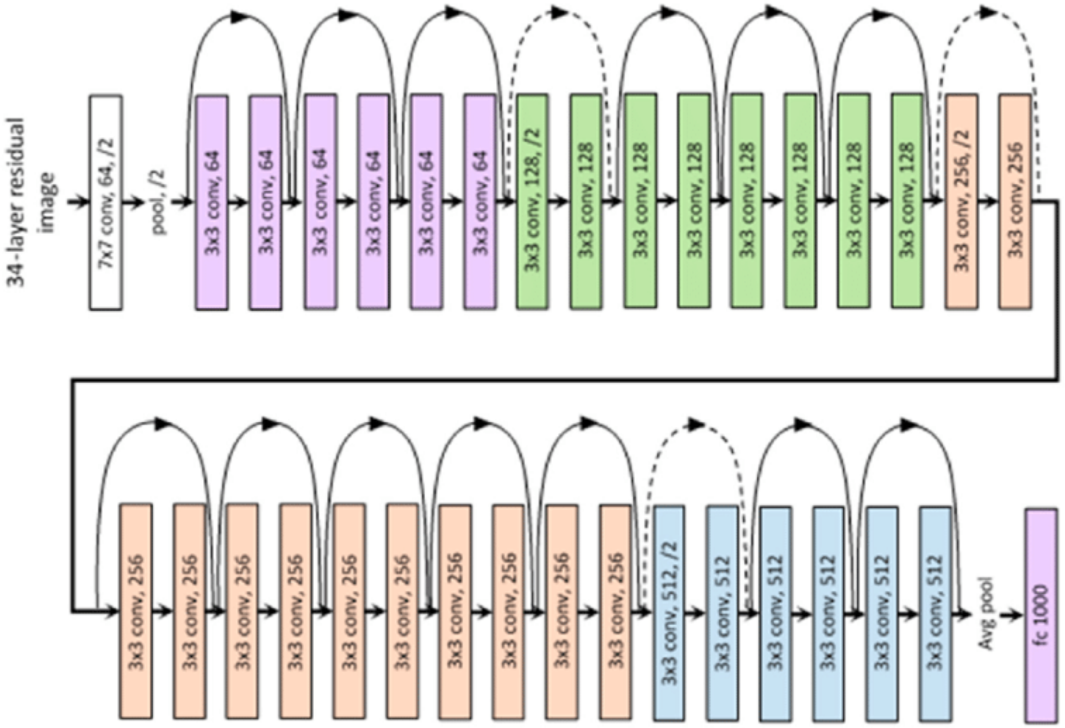
- Vanishing and exploding gradients
- May increase # of layer

Specification:

- Batch Normalization after each ConvLayer
- SGD + momentum (0.9)
- Minibatch Size: 256
- No Dropout

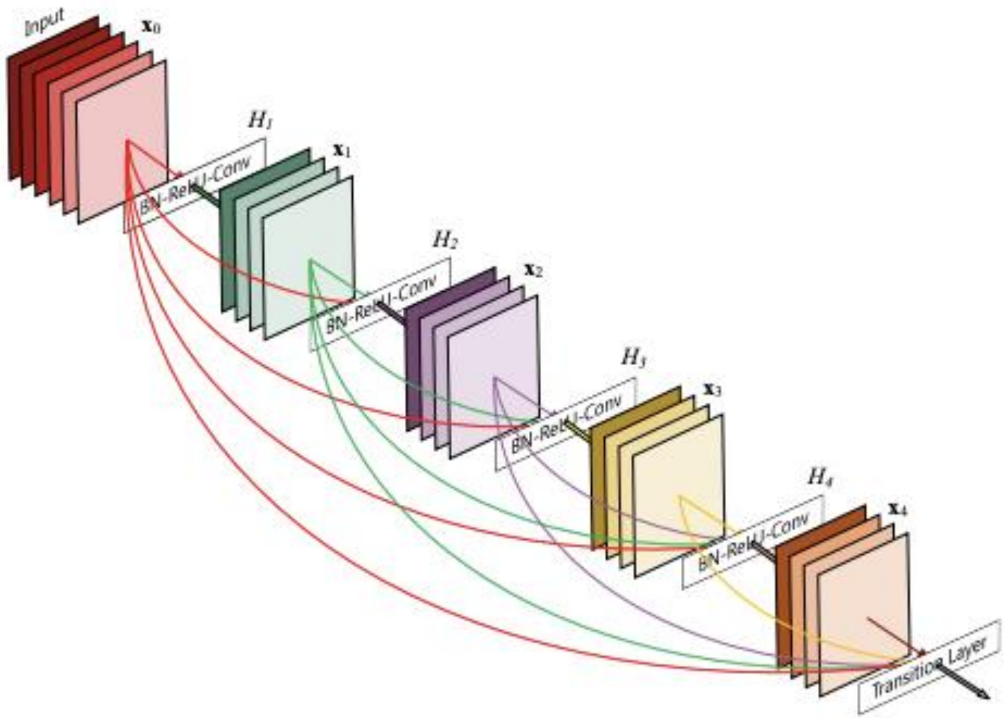
ResNet

(a)

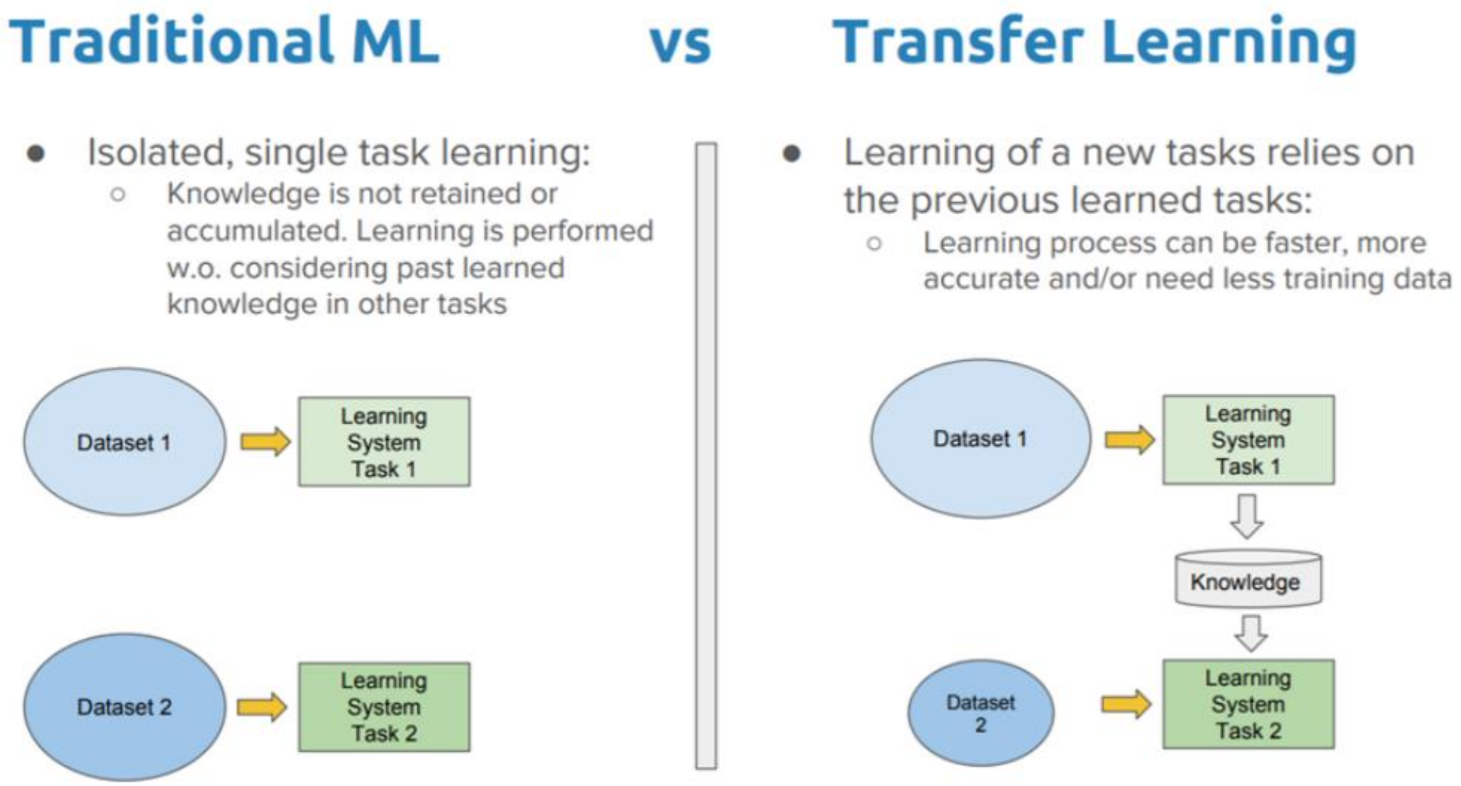

34-layer residual image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Avg pool
fc 1000

(b)

DensNet


Input
$\mathbf{x}_0$
H_1
$\mathbf{x}_1$
BN-ReLU-Conv
H_2
$\mathbf{x}_2$
BN-ReLU-Conv
H_3
$\mathbf{x}_3$
BN-ReLU-Conv
H_4
$\mathbf{x}_4$
BN-ReLU-Conv
Transition Layer

Transfer Learning



Transfer Learning

New dataset is small and similar to original dataset:

- Use final feature layer and train a linear or simple classifier

New dataset is large and similar to the original dataset:

- Fine-tune the full network

New dataset is small but very different from the original dataset:

- Train a SVM classifier from activations (middle layer, not final) in the network

New dataset is large and very different from the original dataset:

- Train from scratch (Initialize with weights from a pre-trained model)

Transfer Learning