Focus for this lecture

Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

- Deep Learning Architectures
- CNN
- CNN in Images, Speech and Text

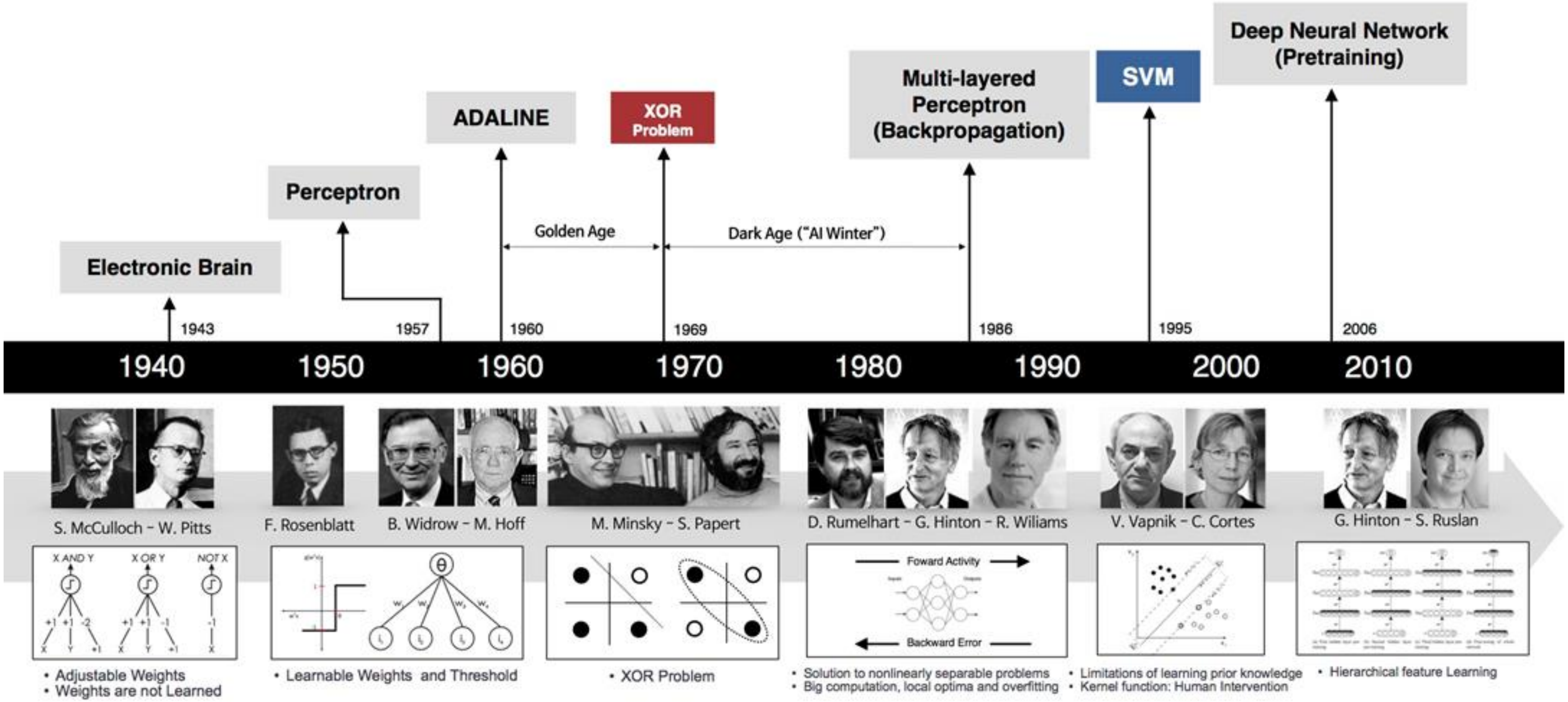
CNN Architecture

Convolution Layer to CNNs and DL

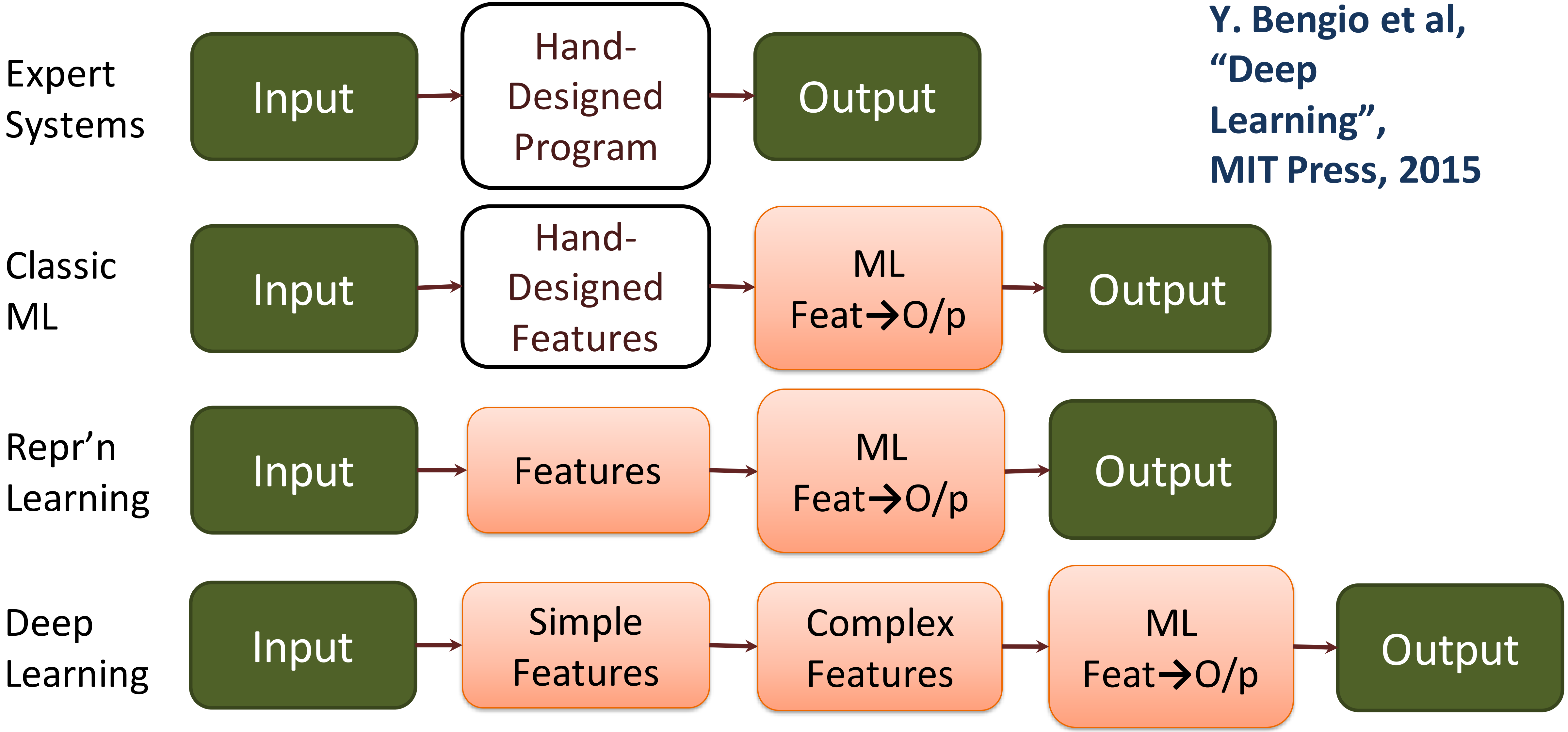
Agenda

- Intro to Deep Learning
- Revisit:
  - 1D Convolution
  - 2D Convolution
  - Terminologies and Utilities
- Convolutional Layer to CNNs
  - Typical architectures
  - Why simple depth is not enough
- Applications in different Modalities (Next Lecture)

History of Deep Learning



Evolution of Learning



Y. Bengio et al, "Deep Learning", MIT Press, 2015

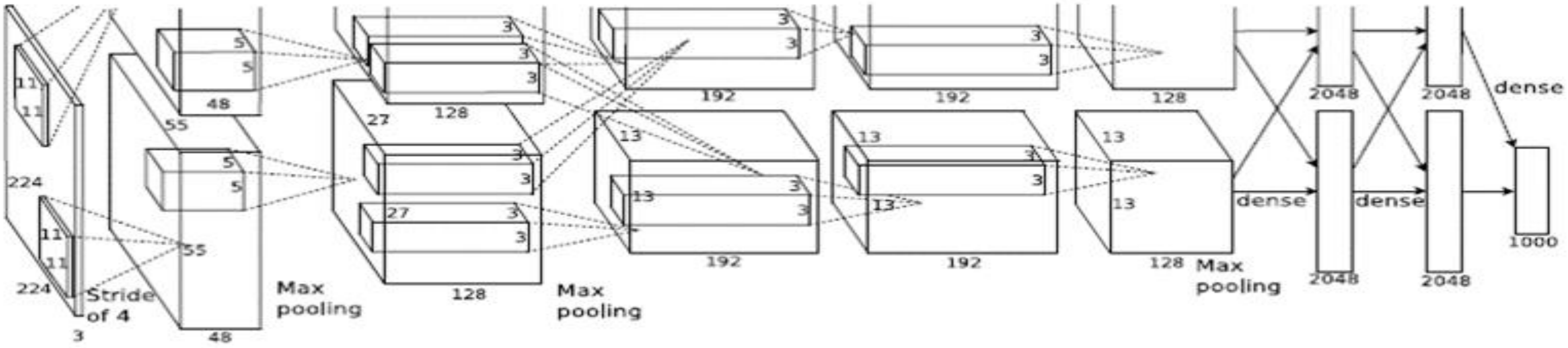
Case Study

ImageNet ILSVRC

ImageNet ILSVRC

IMAGENET

AlexNet (NIPS 2012)



ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

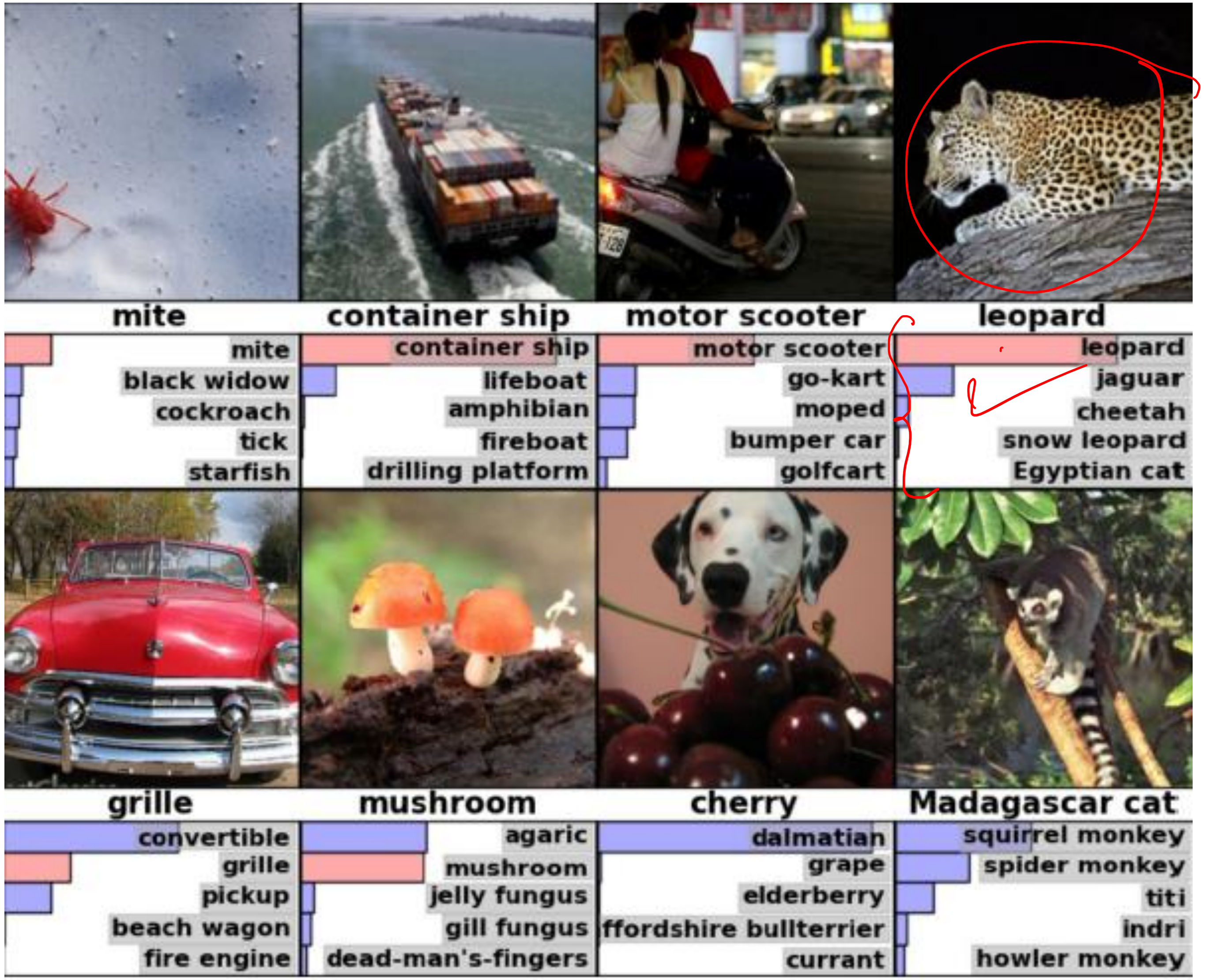
Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

ImageNet Classification Task:

Previous Best : ~25% (CVPR-2011)
AlexNet : ~15 % (NIPS-2012)

ImageNet ILSVRC

- 1000 object classes
- Images:
    - 1.2M train
    - 100k test



Success of "Deep Learning": ImageNet Challenge

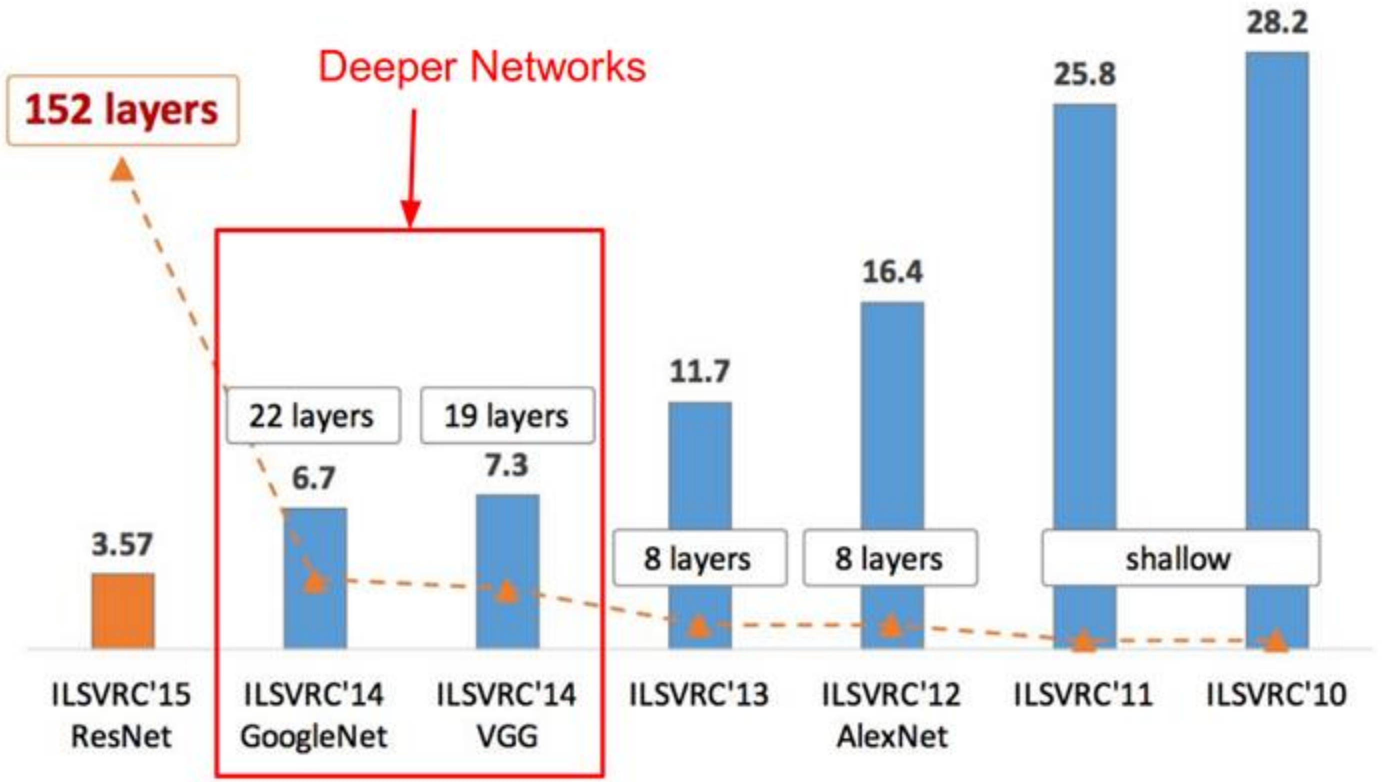
Top-5 Error on Imagenet Classification Challenge (1000 classes)

Method	Top-Error Rate
SIFT+FV [CVPR 2011]	~25.7%
AlexNet [NIPS 2012]	~15%
OverFeat[ICLR 2014]	~ 13%
ZeilerNet [ImageNet 2013]	~11%
Oxford-VGG [ICLR 2015]	~7%
GoogLeNet [CVPR 2015]	~6%, ~4.5%
ResNet [CVPR16]	~3.5%
Human Performance	3 to 5 %

Mostly Deeper Networks
Smaller Convolutions
Many Specific Enhancements

Getting Deeper

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

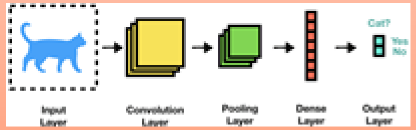


Blank Slide

Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

✓ Deep Learning Architectures

- CNN

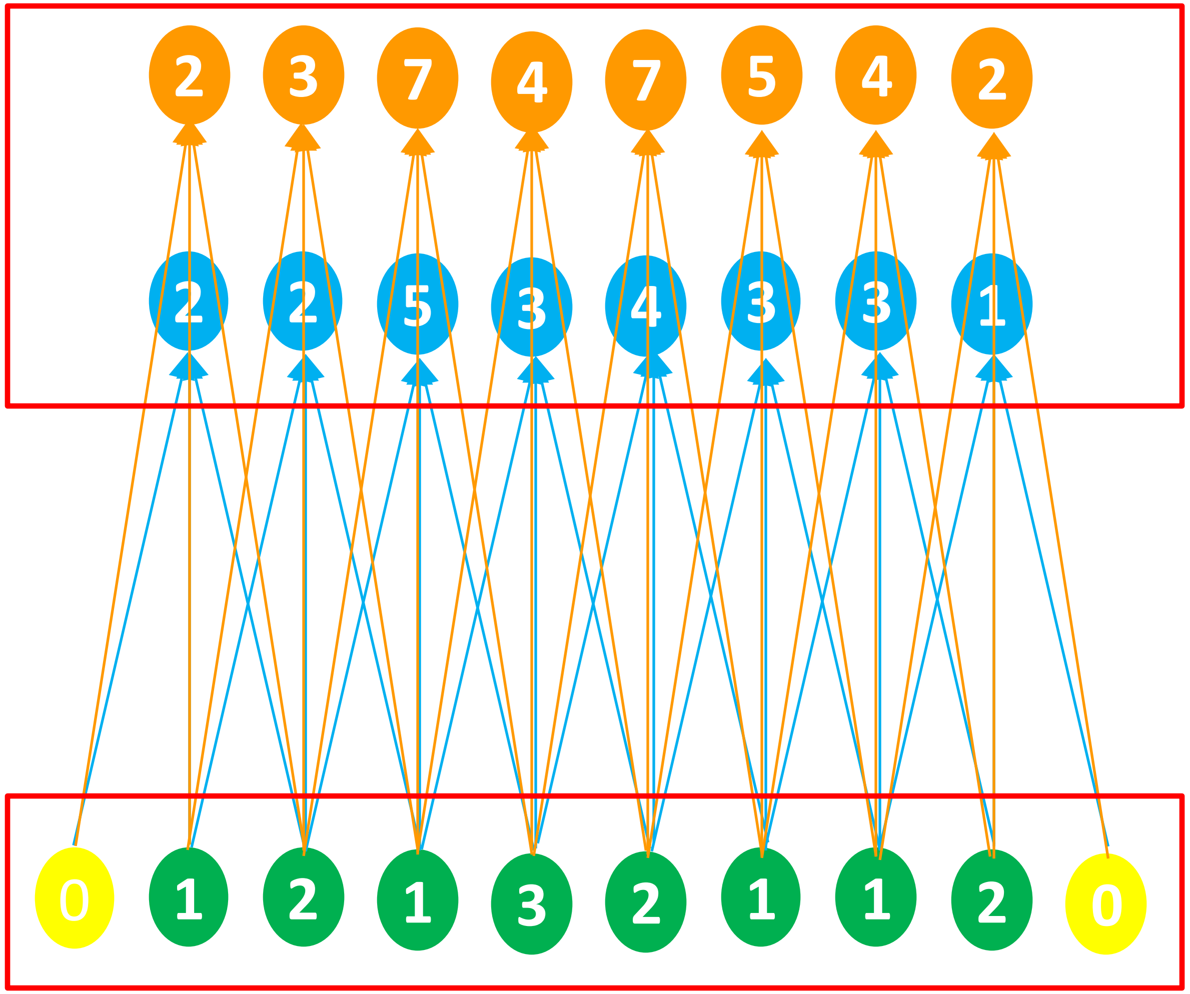


Recap: Convolutions Layer in 1D and 2D

Convolution Layer and Feature Enrichment

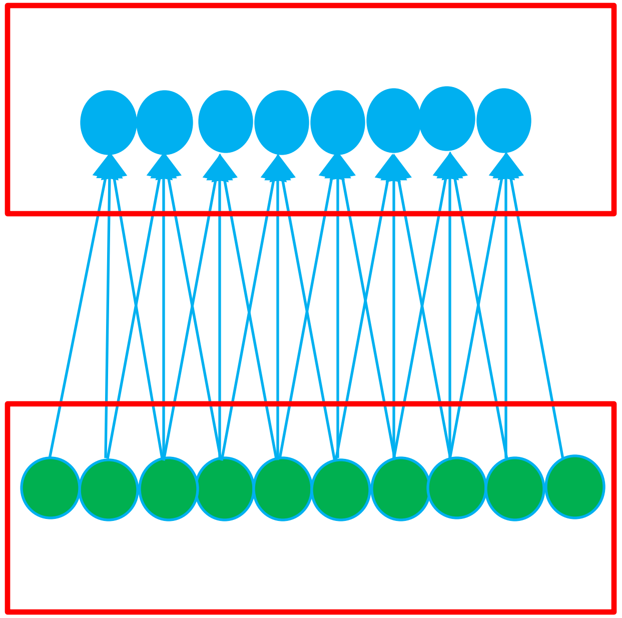
Revisit: Convolution layer

2	0	1	**Filter-2**
1	0	1	**Filter-1**

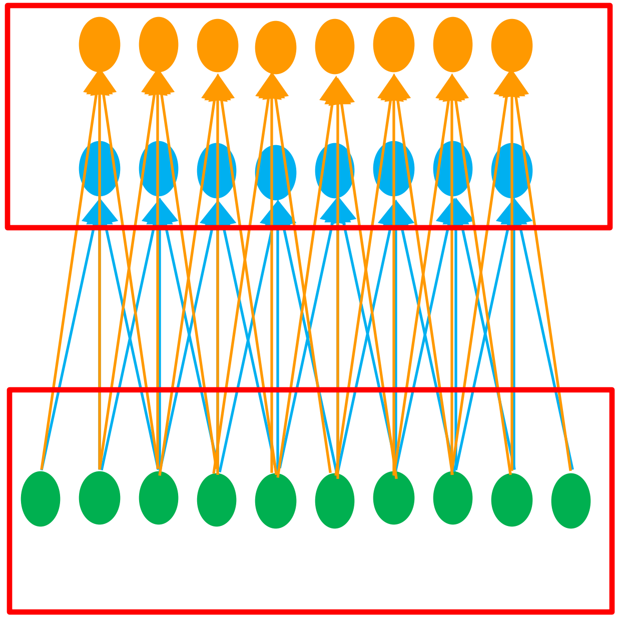


Two such filters/weights (2 X 3 = 6 !!)

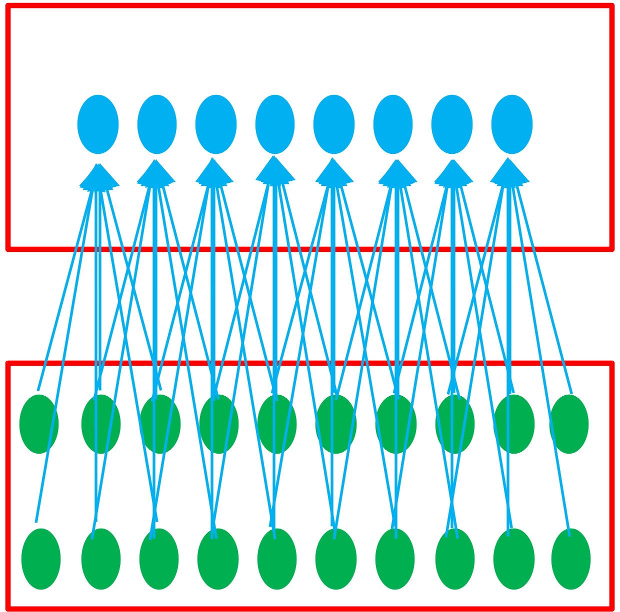
Convolution layer: Different Possibilities



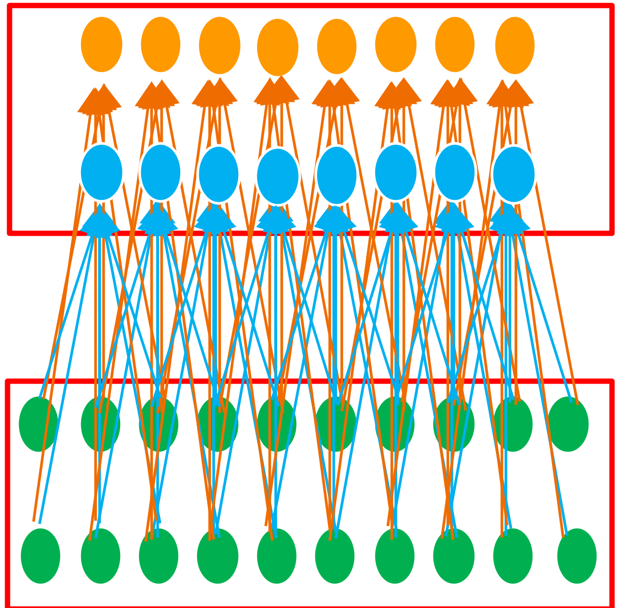
Channels:
- I/P =1
- O/P=1
- #Parameters = 3



Channels:
- I/P =1
- O/P=2
- #Parameters = 6

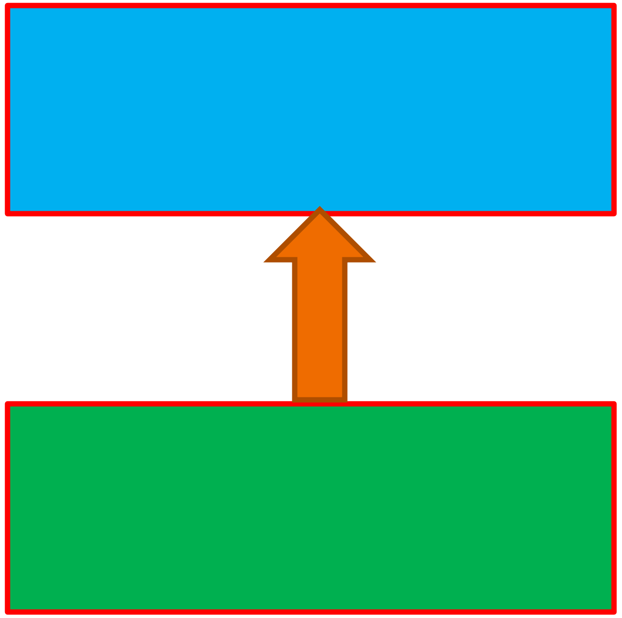


Channels:
- I/P =2
- O/P=1
- #Parameters = 6

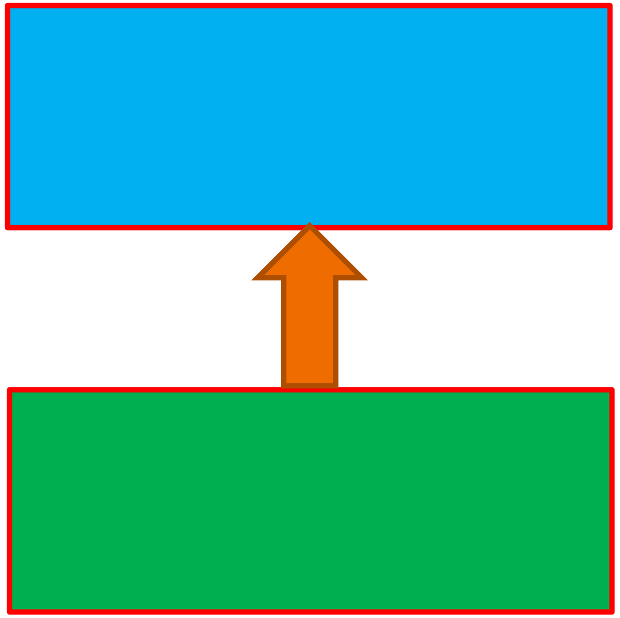


Channels:
- I/P =2
- O/P=2
- #Parameters = 12

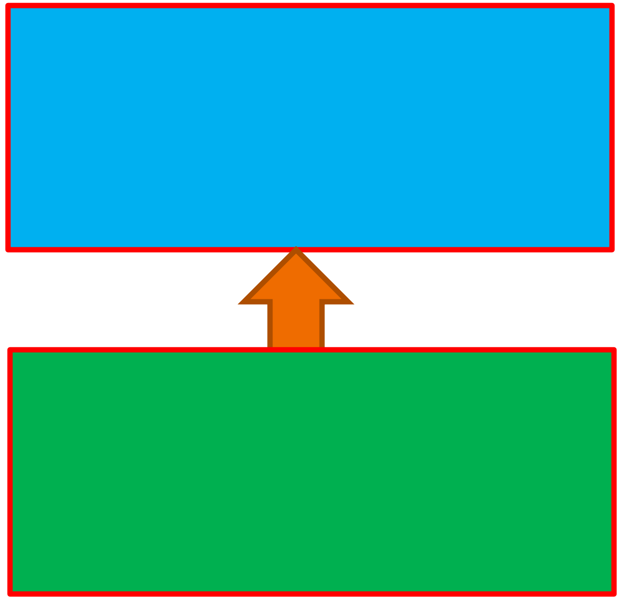
Convolution layer: Different Possibilities



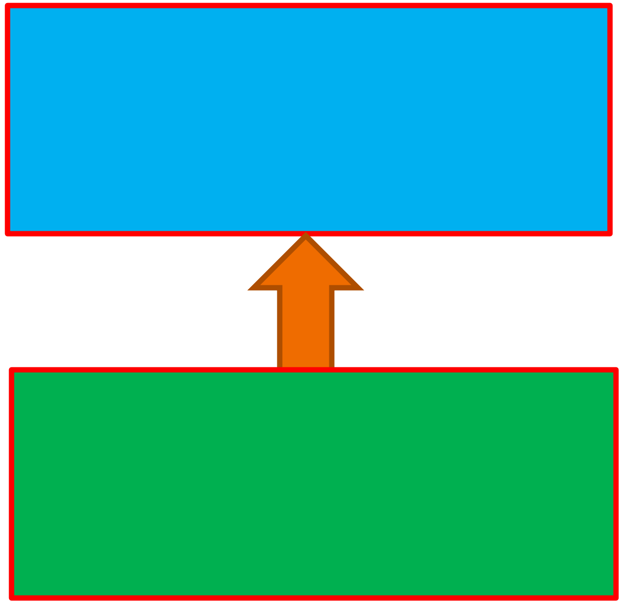
Channels:
- I/P =1
- O/P=1
- #Parameters = 3



Channels:
- I/P =1
- O/P=2
- #Parameters = 6

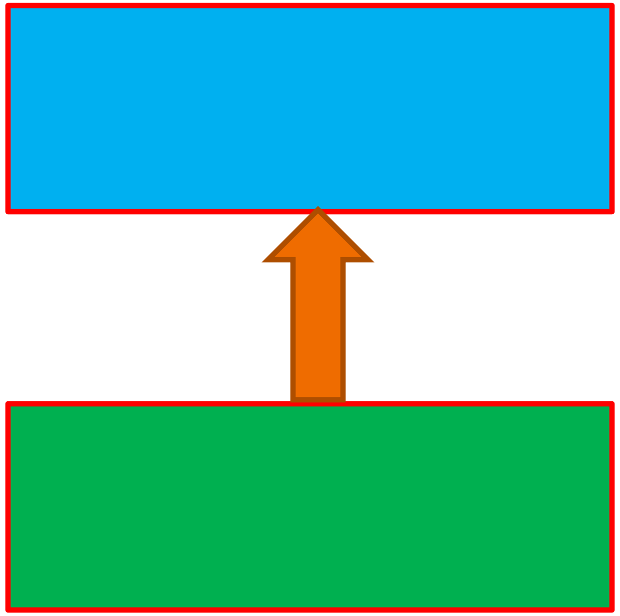


Channels:
- I/P =2
- O/P=1
- #Parameters = 6



Channels:
- I/P =2
- O/P=2
- #Parameters = 12

We Know now ..



Channels:

- I/P =m
- O/P=n
- Filter size: k
- #Parameters = m*n*k

Key Words

- # Input Channels
- # Output channels
- Feature Maps/Channels
- Filters/Weights
- Filter Size/Window Size
- Stride
- Padding

What happens when you convolve ?

Convolution Example (Recap)

-1	1	-1
-1	1	-1
-1	1	-1

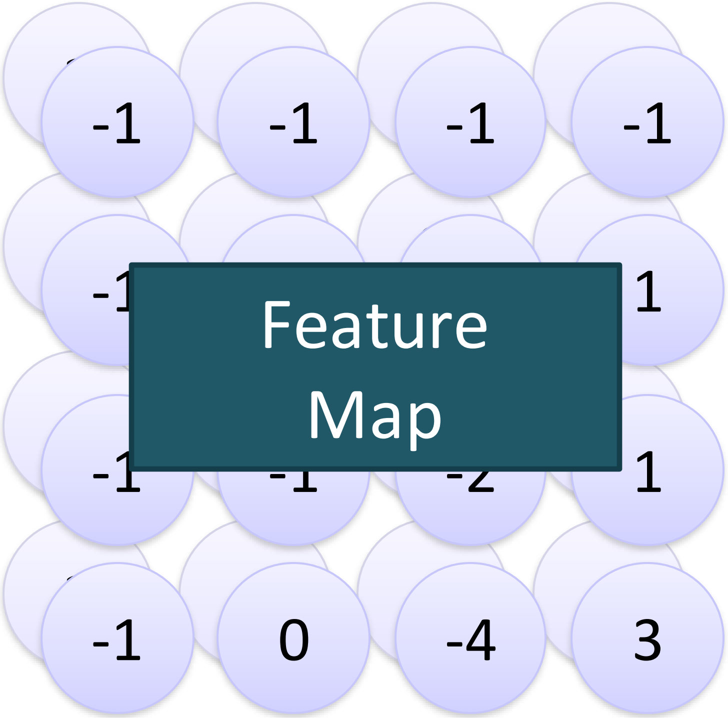
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

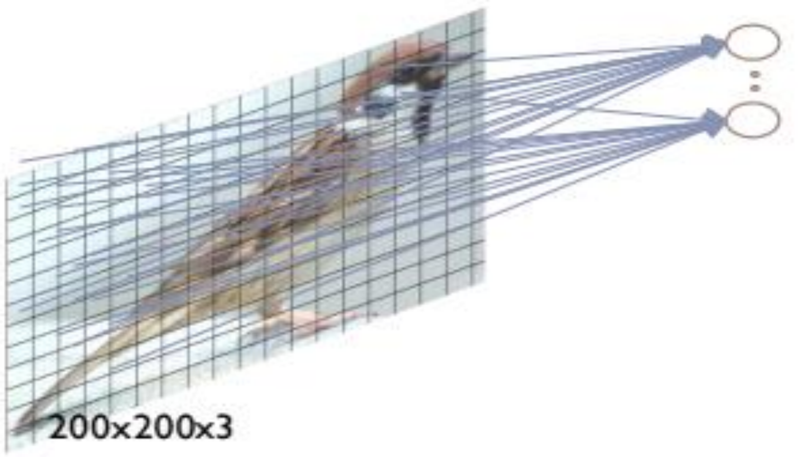
Repeat this for each filter



Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Convolution layer

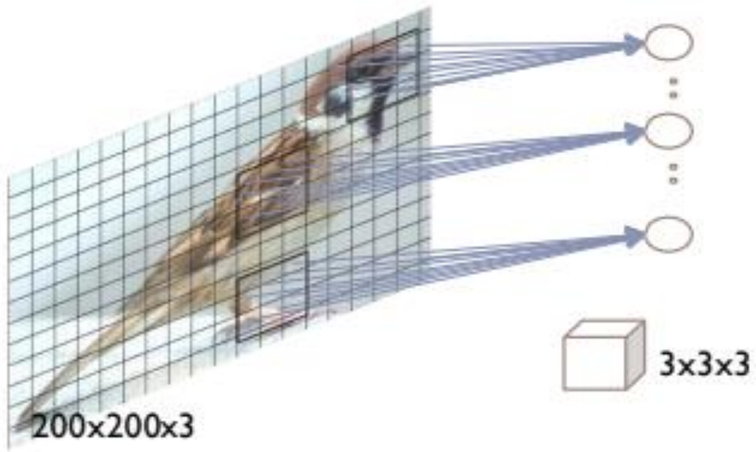
- Fully connected layer



- Image of size 200 X 200 and 3 colours (RGB)
- #Hidden Units: 120,000 (= 200X200X3)
- #Params: 14.4 billion (= 120K X 120K)
- Need huge training data to prevent over-fitting!

- Locally connected layer

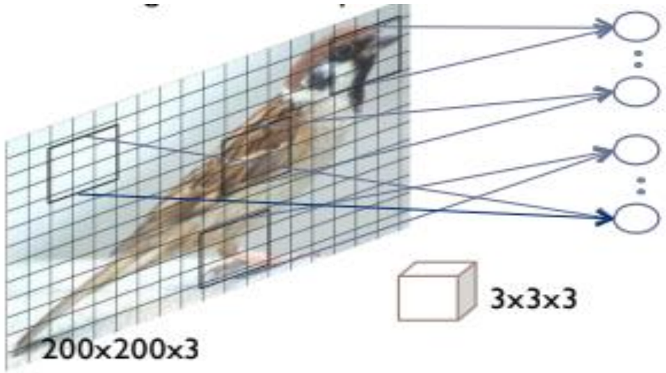
Parameter Calculations



- #Hidden Units: 120,000
- #Params: 3.2 Million (= 120K X 27)
- Useful when the image is highly registered

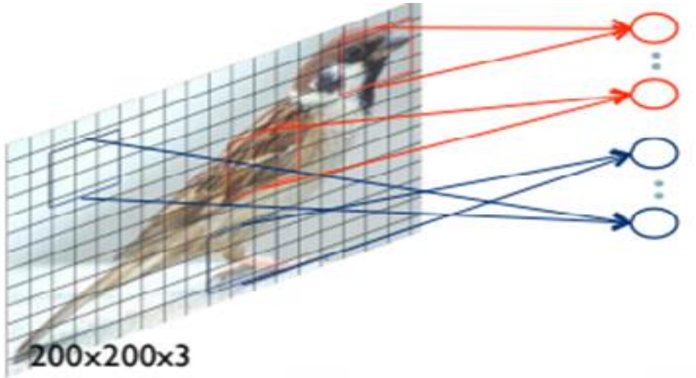
Convolution layer

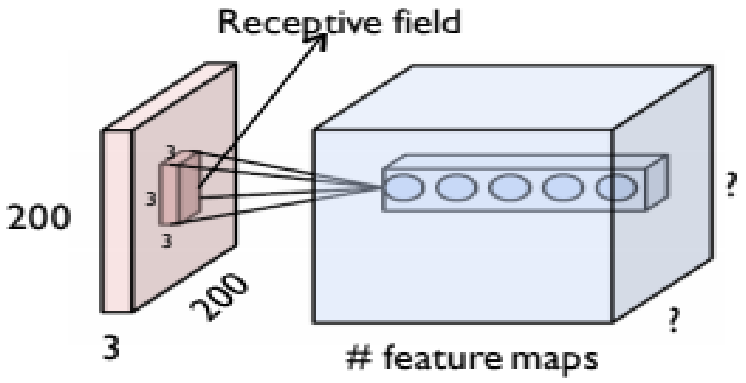
- **Convolutional layer with a single feature map**



- #Hidden Units: 120,000
- #Params: 27 x #Feature Maps
- Sharing parameters
- Exploits the stationarity property and preserves locality of pixel dependencies

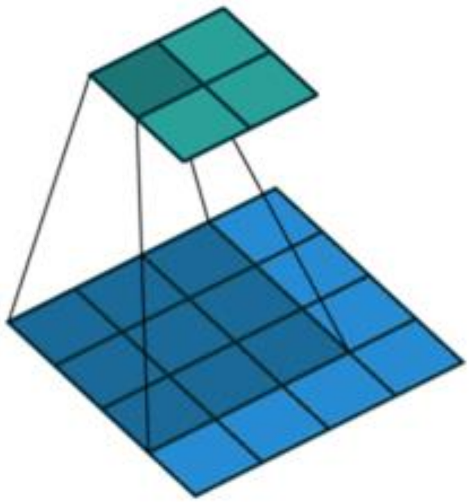
- **Convolutional layer with multiple feature maps**



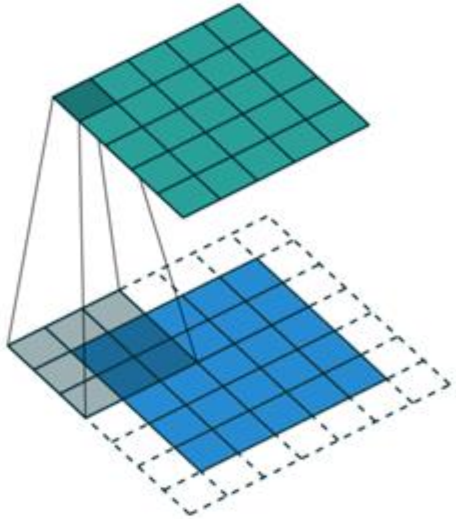


Revisit: Convolution layer

- Window size
- Stride
- Padding
- Pool



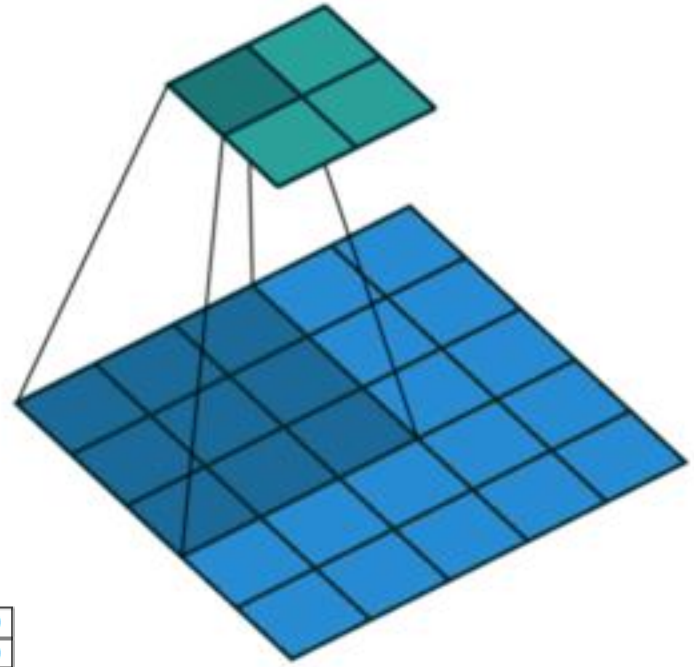
Window size: 3x3
Stride: 1
Padding: 0

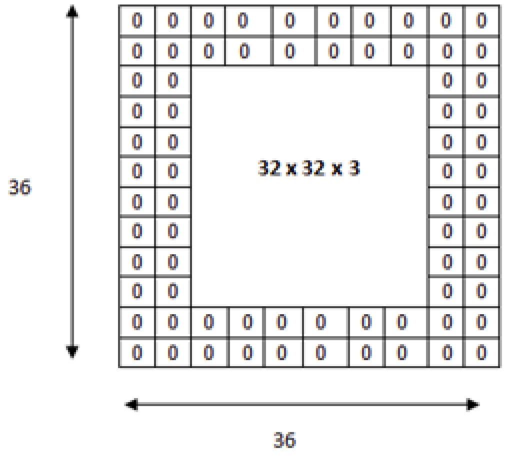


Window size: 3x3
Stride: 1
Padding: 1

CNNs

- Strides reduces dimension

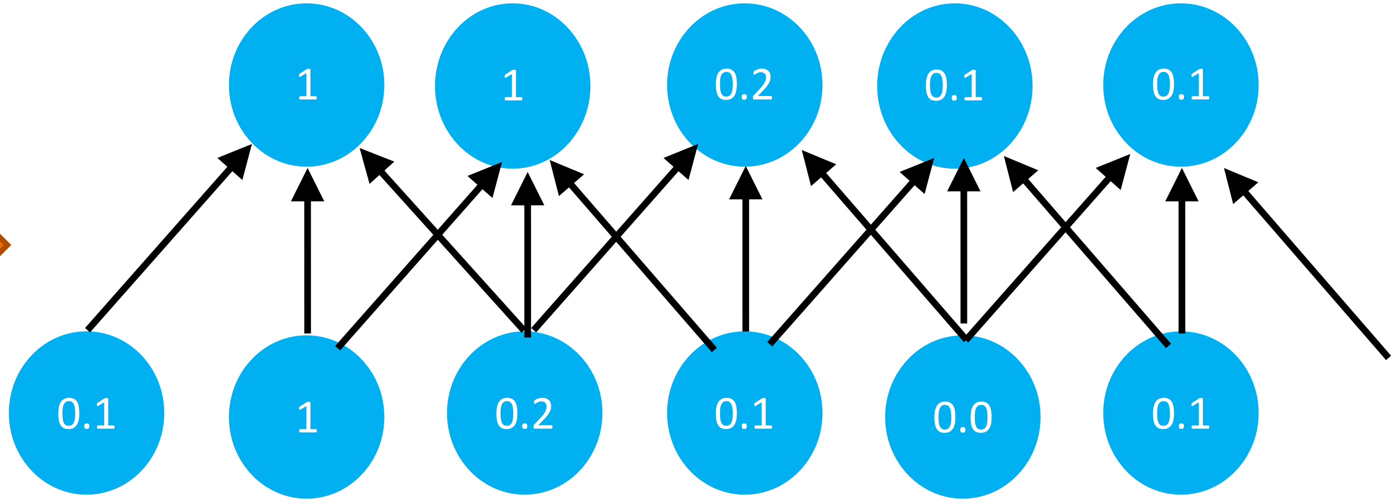


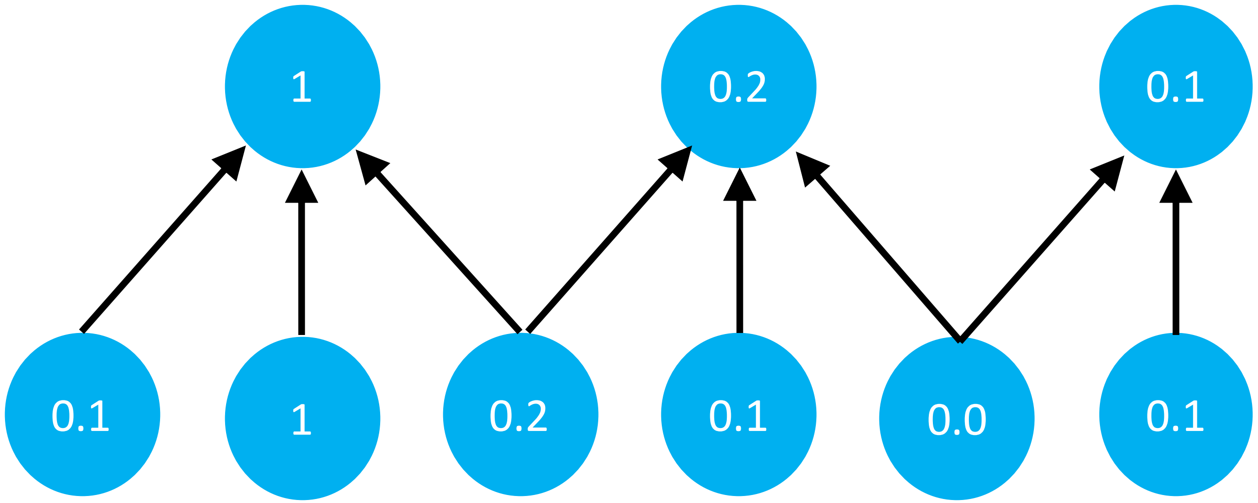


$$O = \frac{(W - K + 2P)}{S} + 1$$

Max Pool and Stride

- Window Size = 3
- Stride =1





- Window Size = 3
- Stride =2

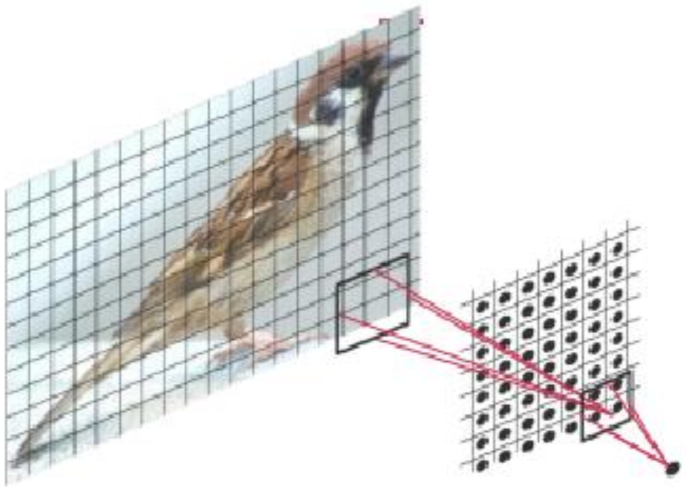
Max pooling in 2-D

5	8	2	1
4	3	7	9
3	7	3	5
6	2	2	0

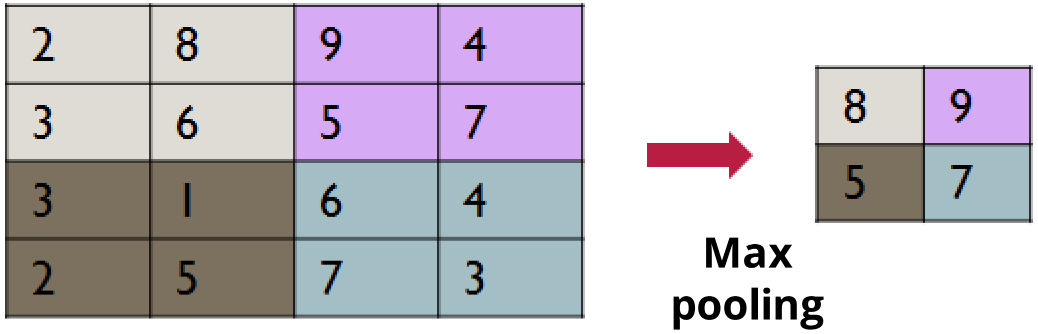
Kernel size: 2X2
Stride: 2

8	9
7	5

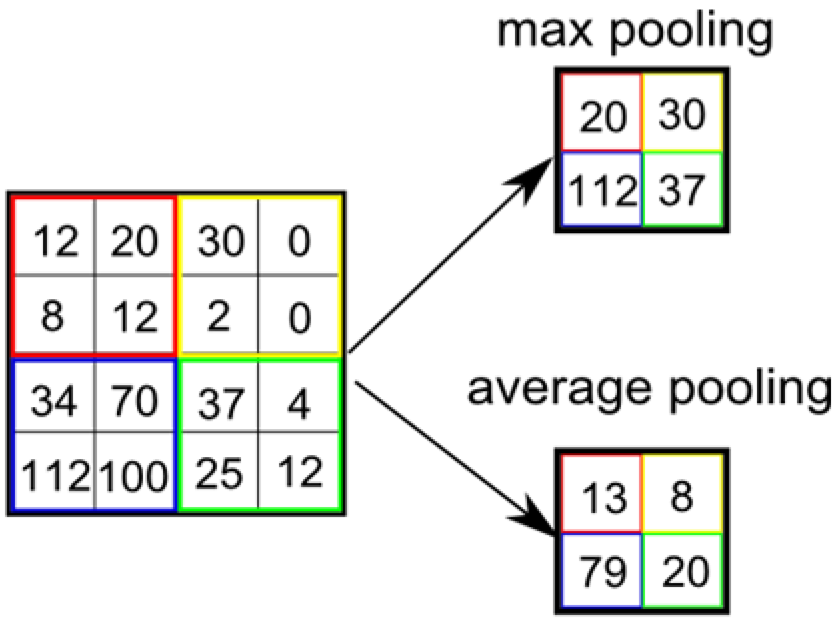
Pooling Layer



Pool Size: 2x2
Stride: 2
Type: Max



- Role of an aggregator.
- Invariance to image transformation and increases compactness to representation.
- Pooling types: Max, Average, L2 etc.



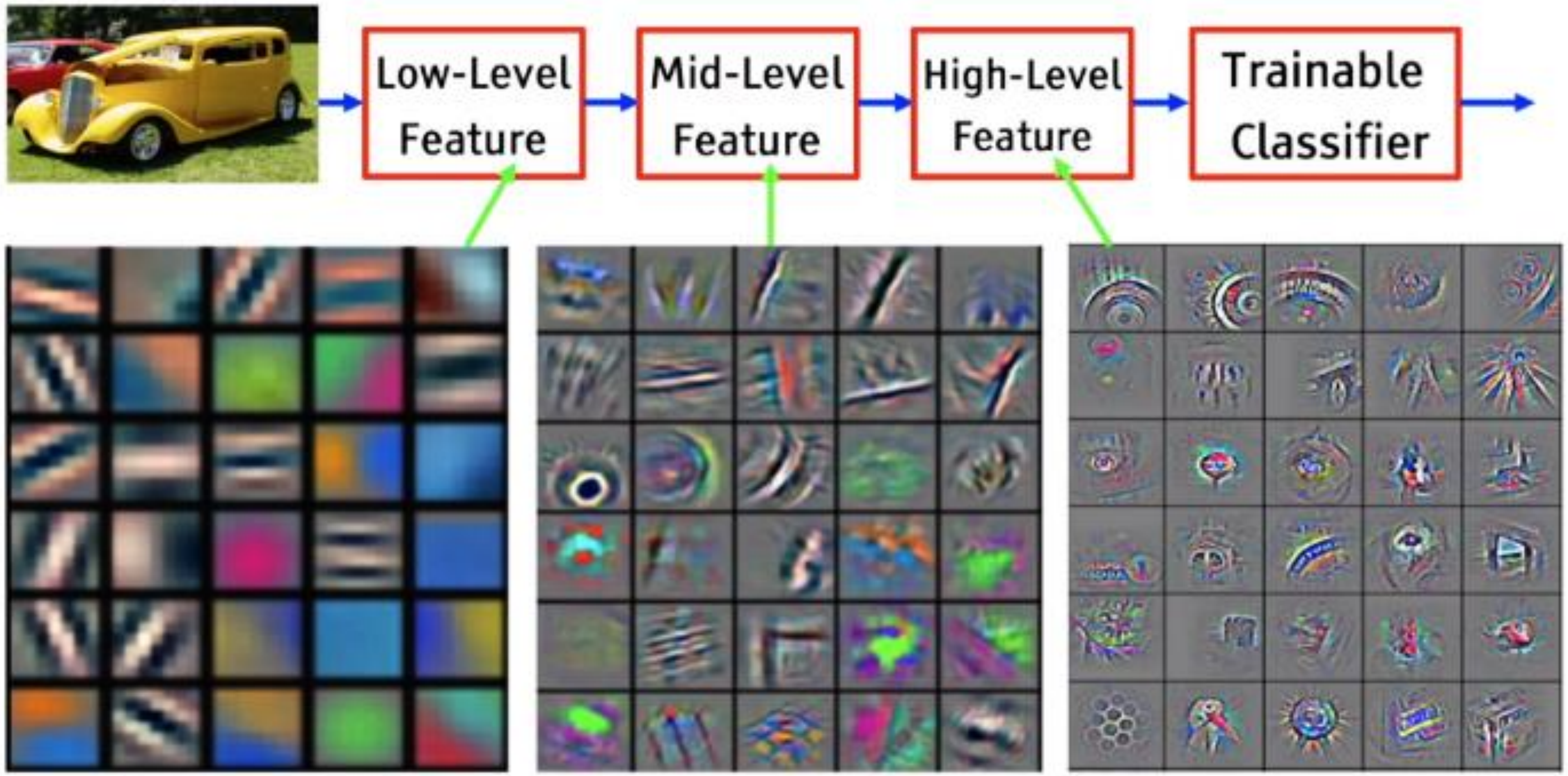
Blank Slide

Couple of things to appreciate

- Convolution layer is much more “compact” compared to fully connected layers (FC) used in MLPs.
- Convolution layer has “multiple filters” and they act as “feature Detectors” or “Feature Extractors” for the raw data.
- This feature learning removes the need of “hand crafting” features. Also we can learn/use the features that works for the problem.

Deep Learnt Features

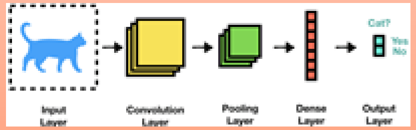
- It's deep if it has more than one stage of non-linear feature transformation.



Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

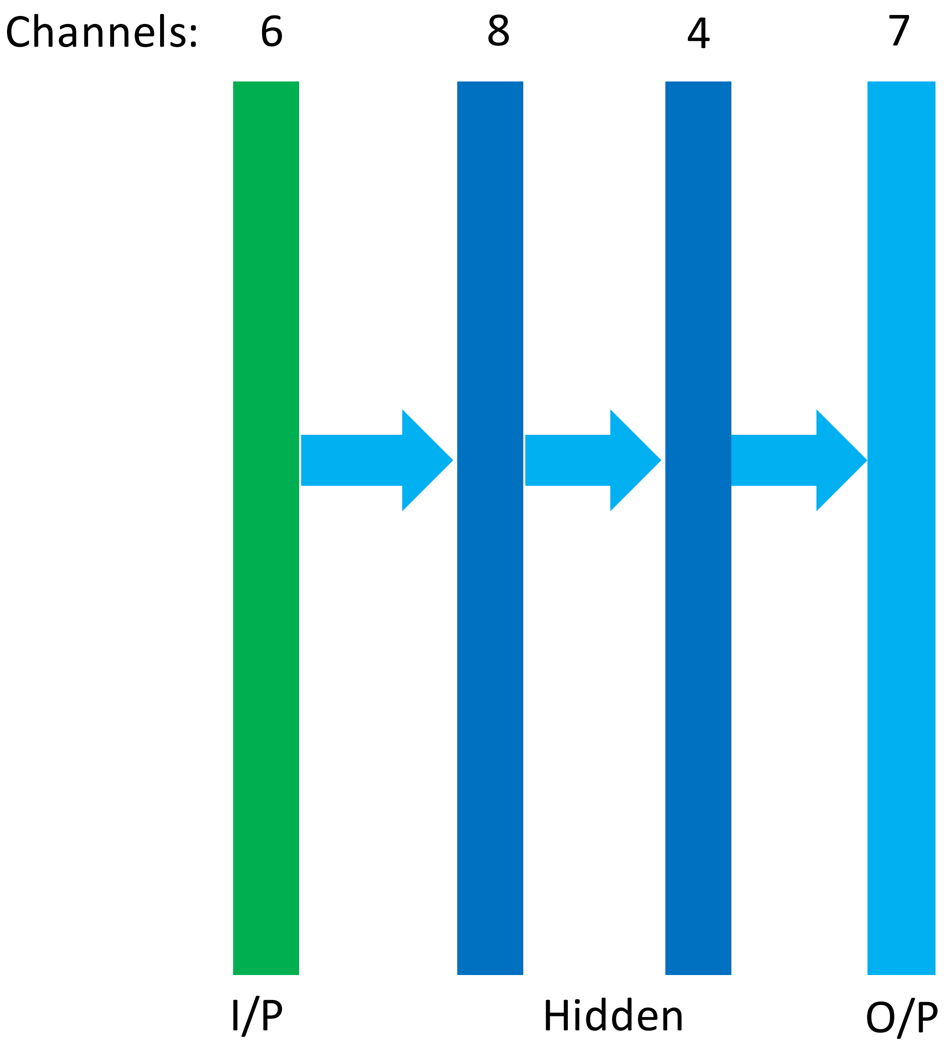
✓ Deep Learning Architectures

- CNN

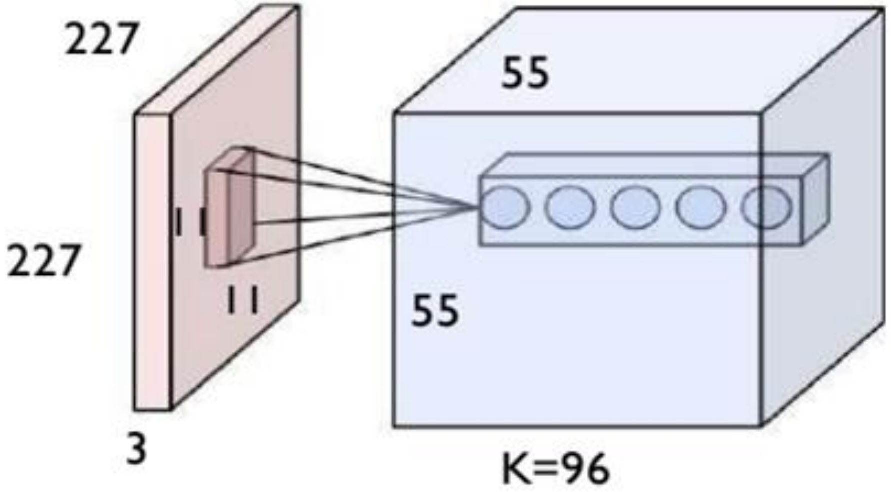


Architectures from Blocks

Layer wise abstraction

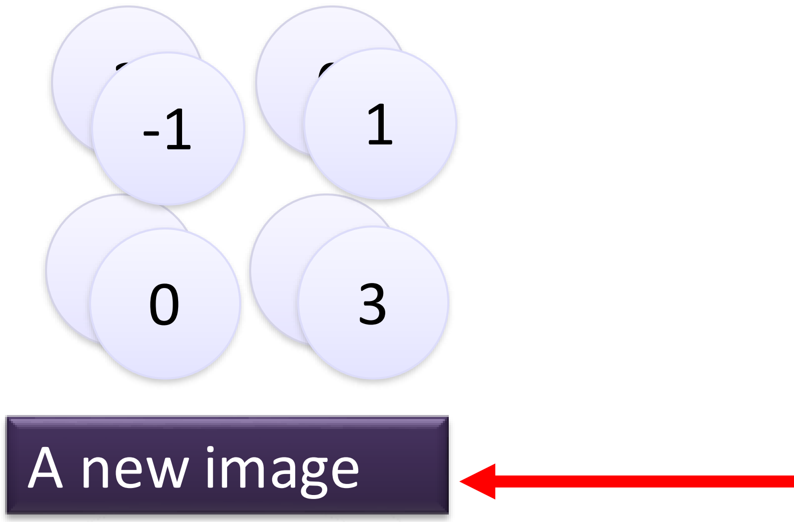


1-D Convolution



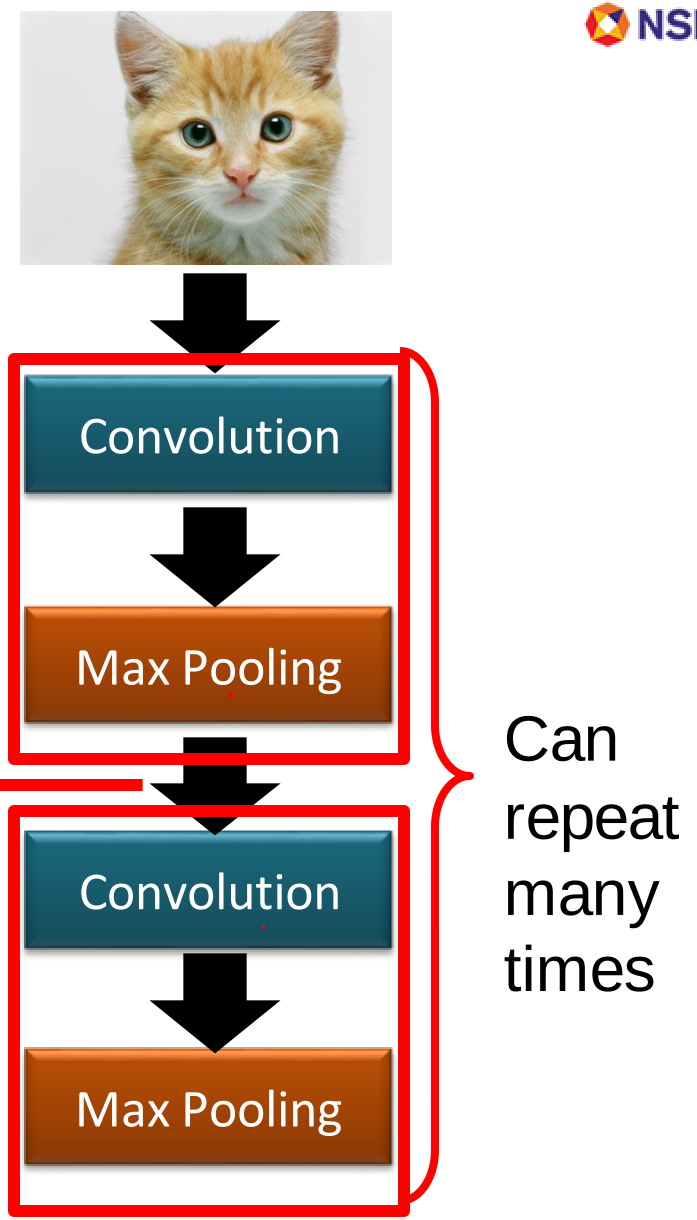
2-D Convolution

The whole CNN

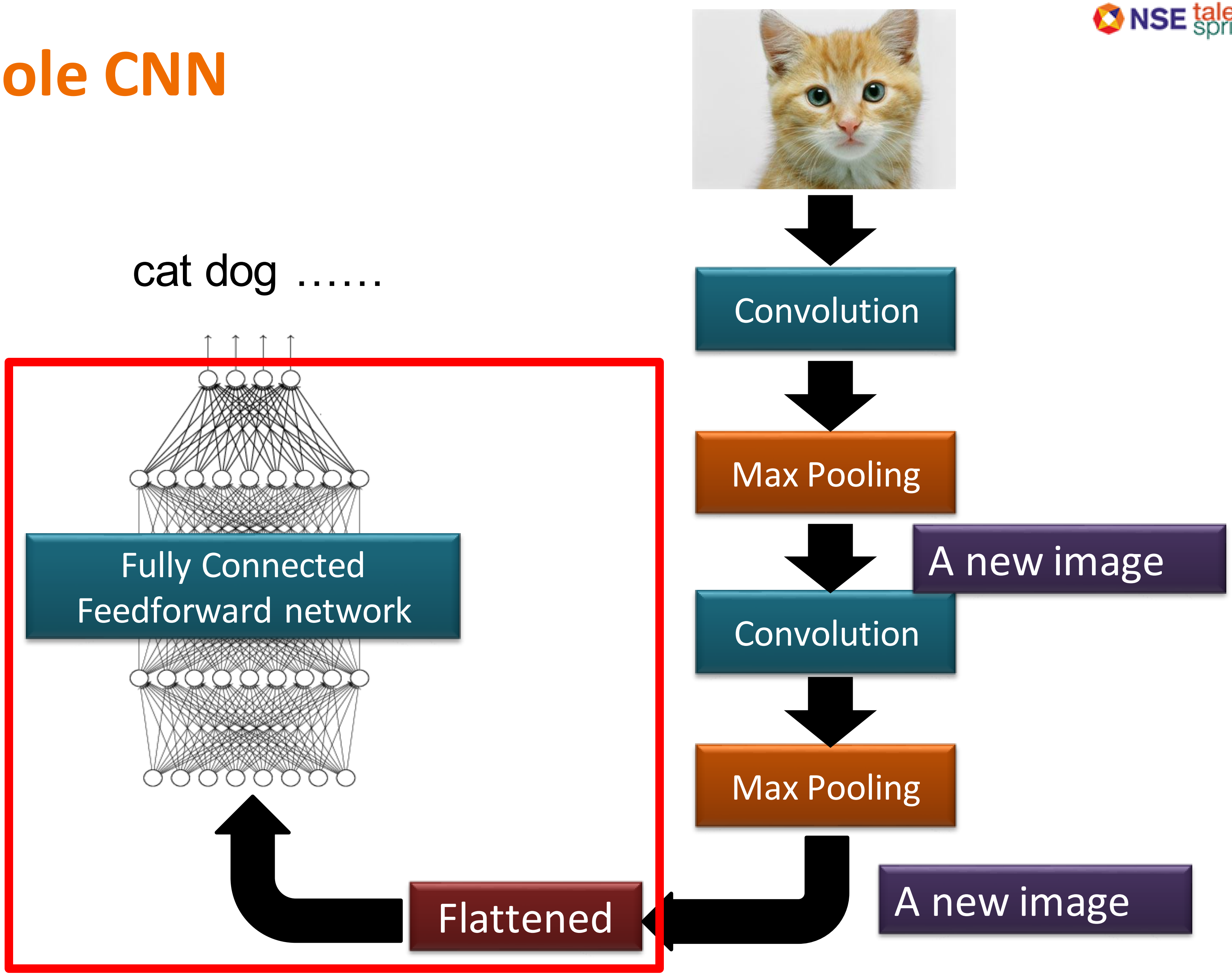


Smaller than the original image

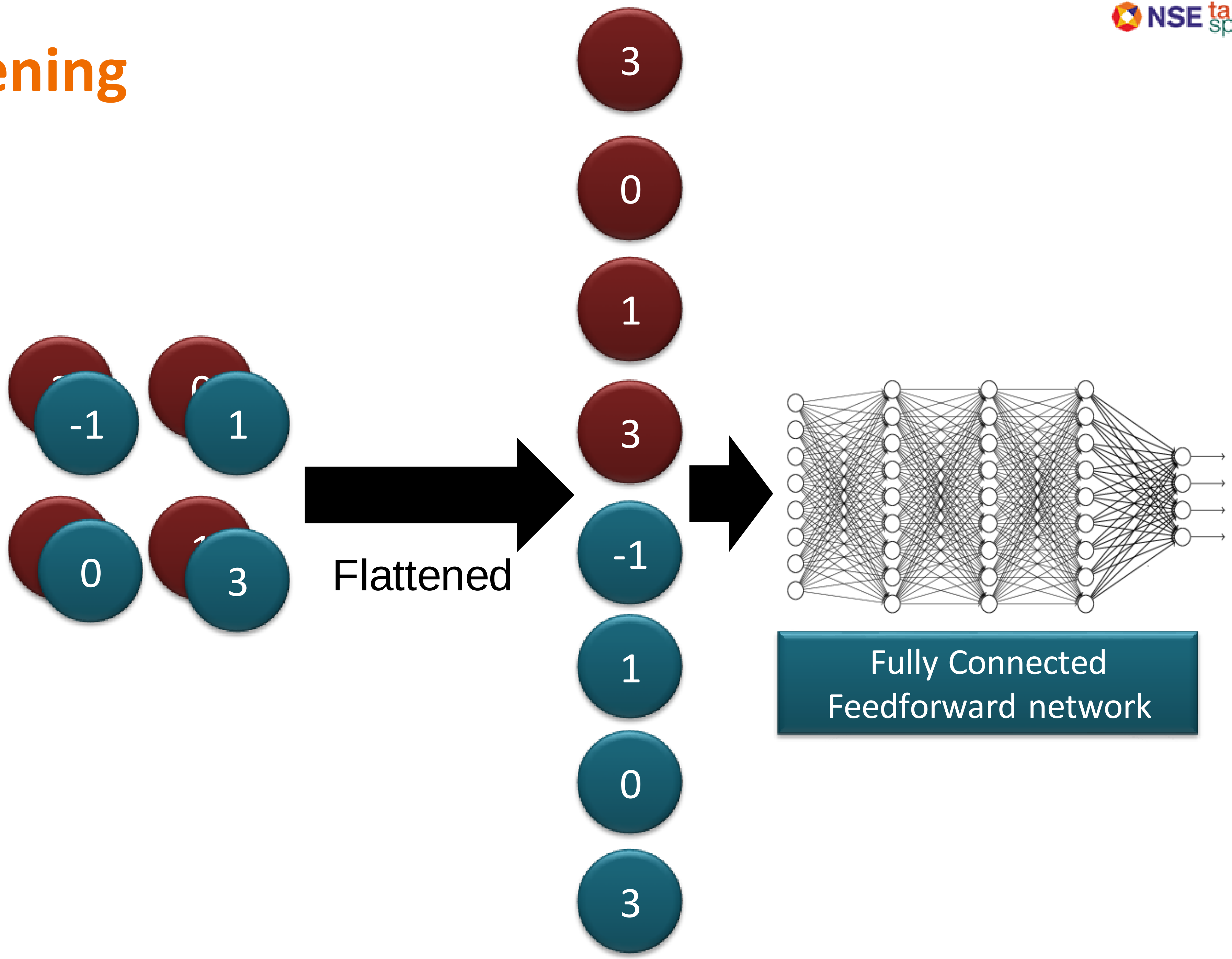
The number of channels is the number of filters



The whole CNN



Flattening



-1
1
0
3
Flattened
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

Terminologies

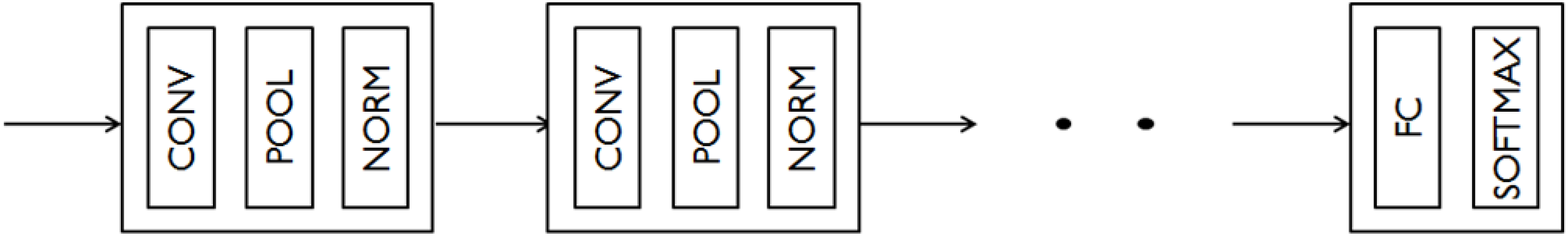
- # Input Channels
- # Output channels

- Feature Maps/Channels
- Filters/Weights
- Filter Size/Window Size

- Stride
- Pooling (Max/Average)
- Fully Connected Layer
- Soft-Max
- Normalization
- Flattening
- Convolution Layer

Typical Architecture

- A typical deep convolutional network



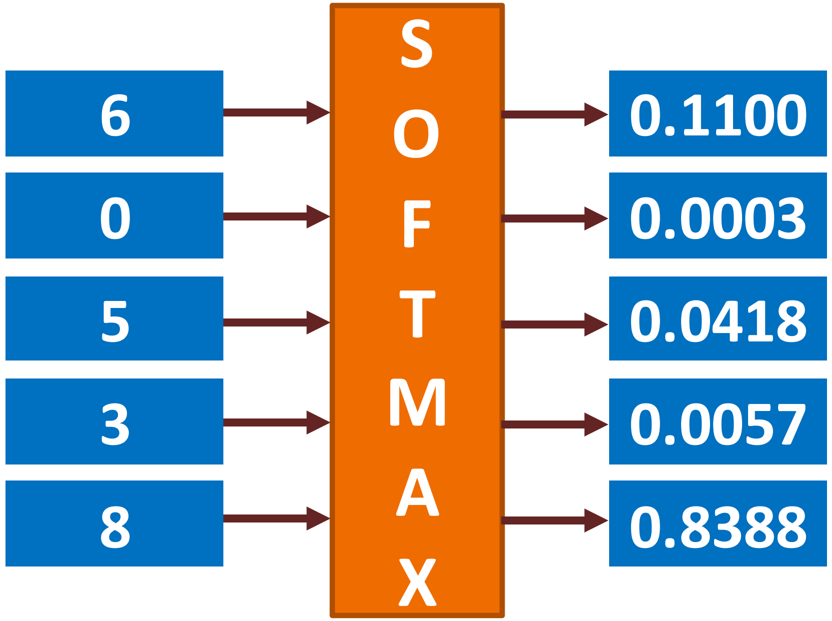
- Other layers
  - Pooling
  - Normalization
  - Fully connected
  - etc.

Softmax

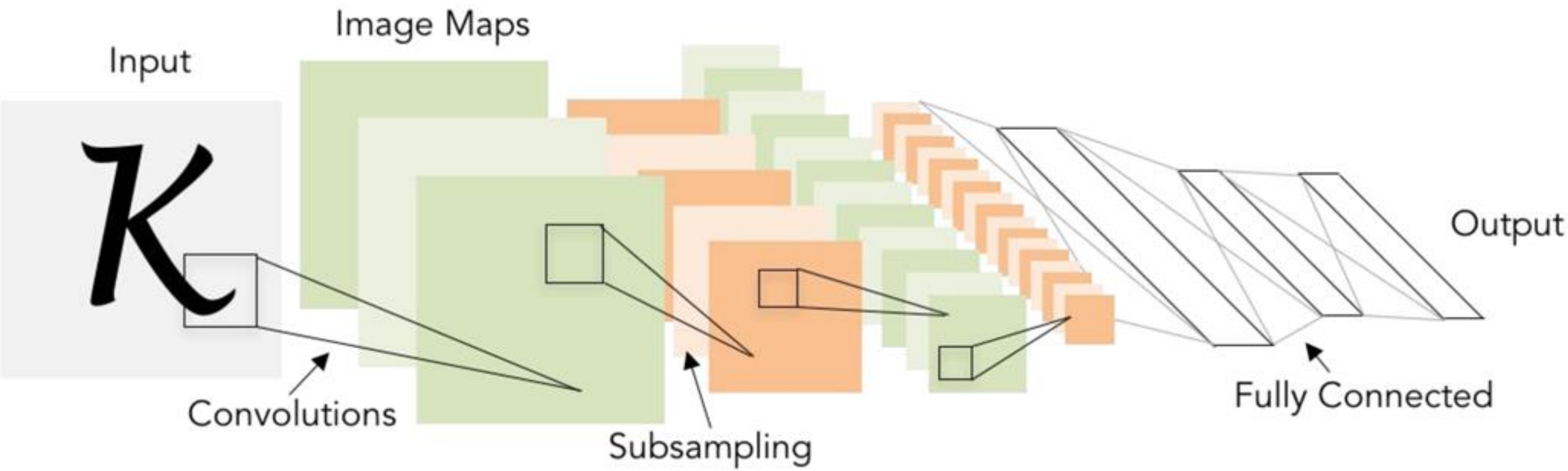
```
Out[12]: array([ 6.,  0.,  5.,  3.,  8.])

In [8]:  exp = (np.e)**(x)
         exp
executed in 6ms, finished 01:47:23 2018-08-21

Out[8]:  array([  4.03428793e+02,   1.00000000e+00,   1.48413159e+02,
                   2.00855369e+01,   2.98095799e+03])

In [9]:  sigma_e = np.sum(exp)
         sigma_e
executed in 9ms, finished 01:47:25 2018-08-21

Out[9]:  3553.8854765602264

In [11]: z = exp/sigma_e
         z
executed in 8ms, finished 01:47:34 2018-08-21

Out[11]: array([  1.13517669e-01,   2.81382168e-04,   4.17608165e-02,
                   5.65171192e-03,   8.38788421e-01])
```

- Normalizes the output.
- K is total number of classes

$$z_n = \frac{e^{x_n}}{\sum_{i=1}^{K} e^{x_i}}$$

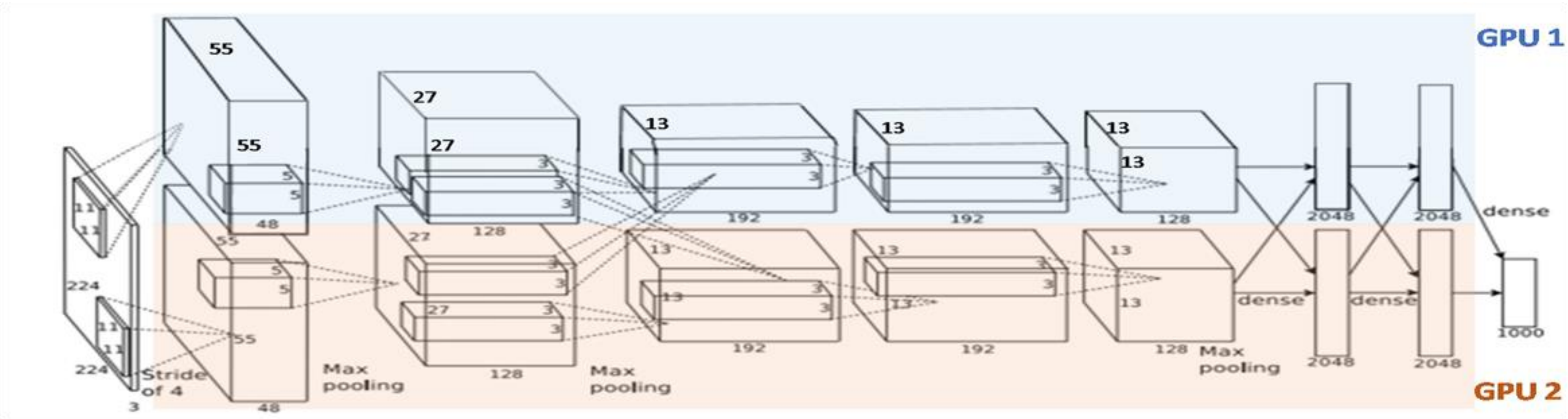


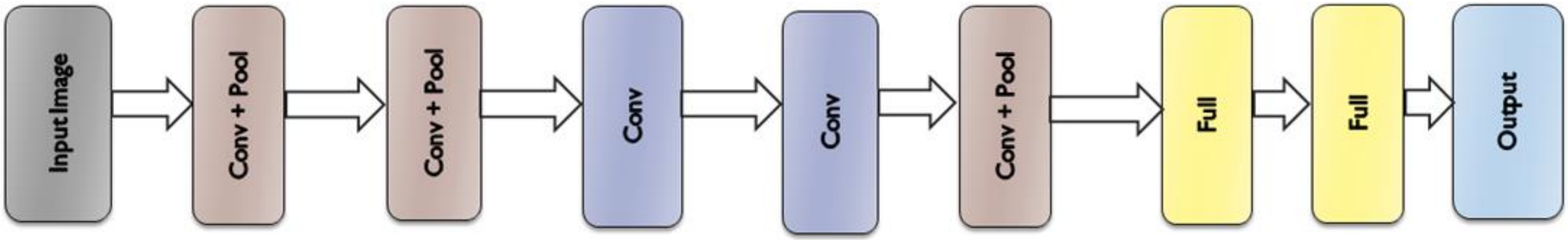
LeNet



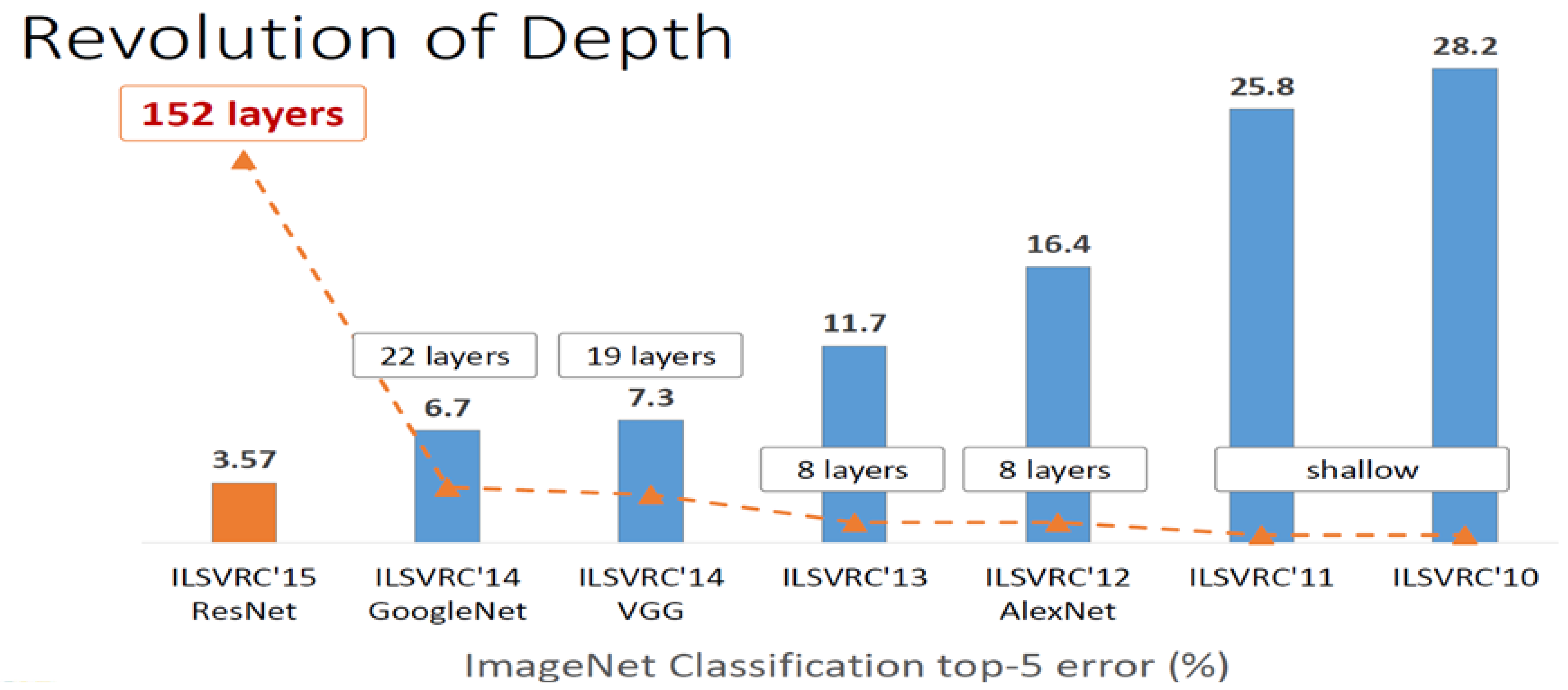
Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

AlexNet Architecture





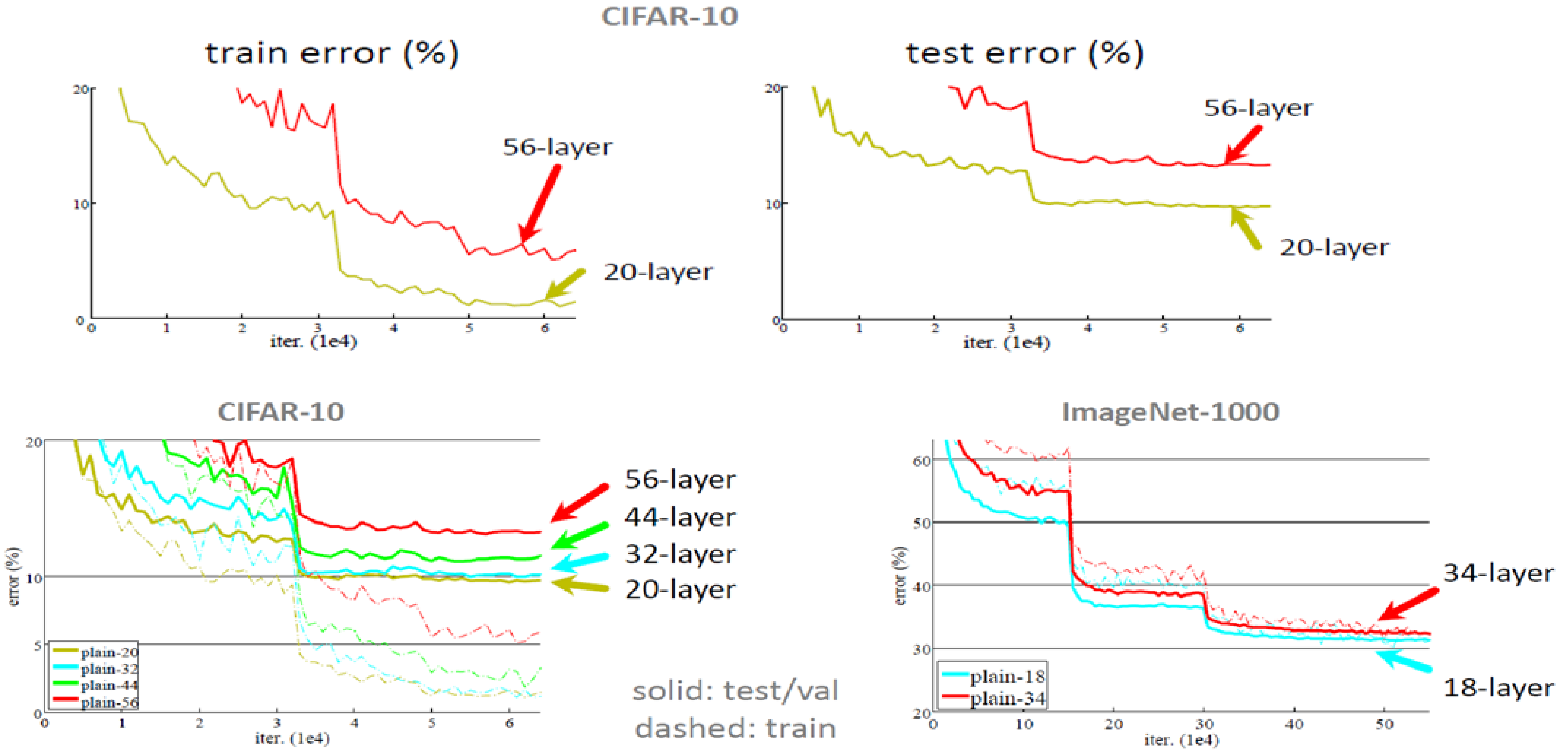
Residual Net [CVPR2016]



Challenge with Depth

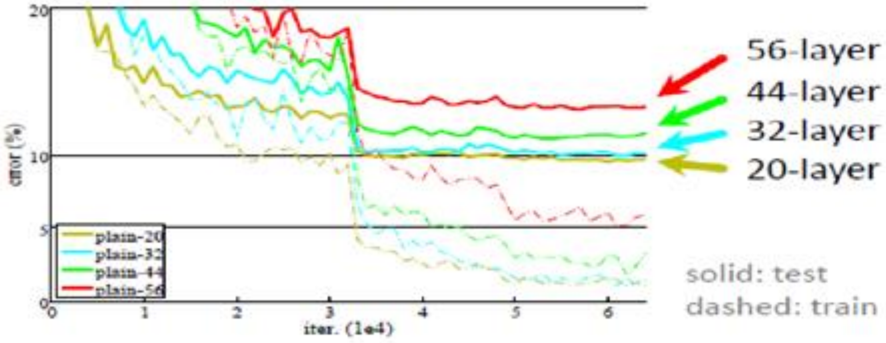
- Vanishing Gradients
  – Error signal don’t reach (enough) the early layers
  – Multiplication of many small numbers (less than one) and become almost zero
- Exploding gradients
  – If gradients are large, product become too big and huge changes in weights

Problems with Simple Deep (cf: Resnet)

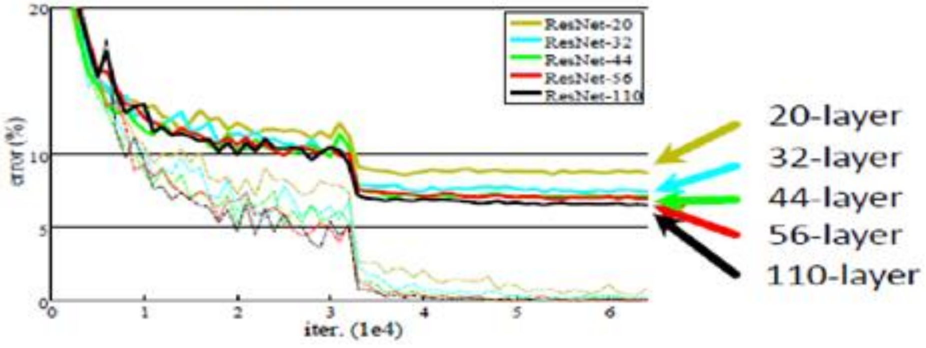


PlainNet Vs ResNet

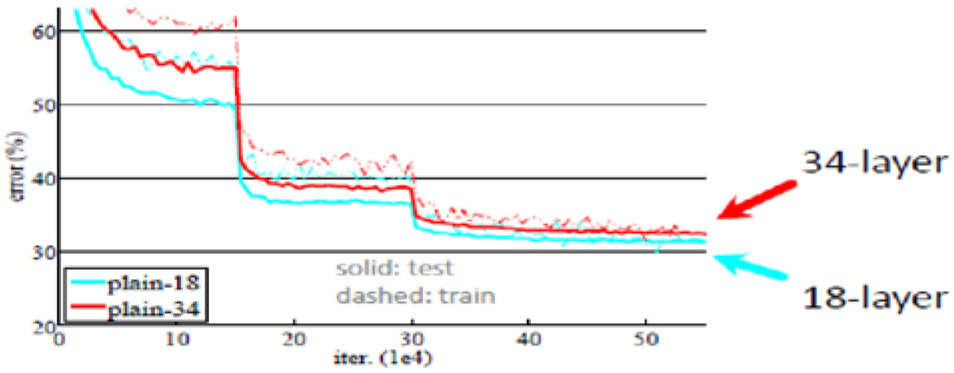
CIFAR-10 plain nets



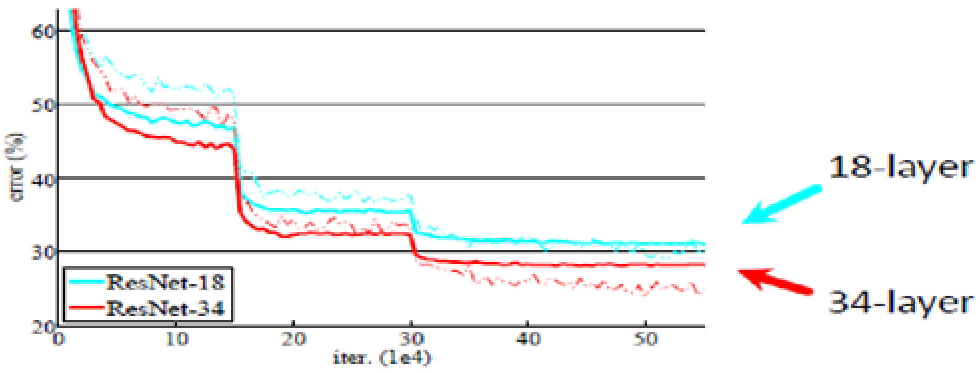
CIFAR-10 ResNets



ImageNet plain nets

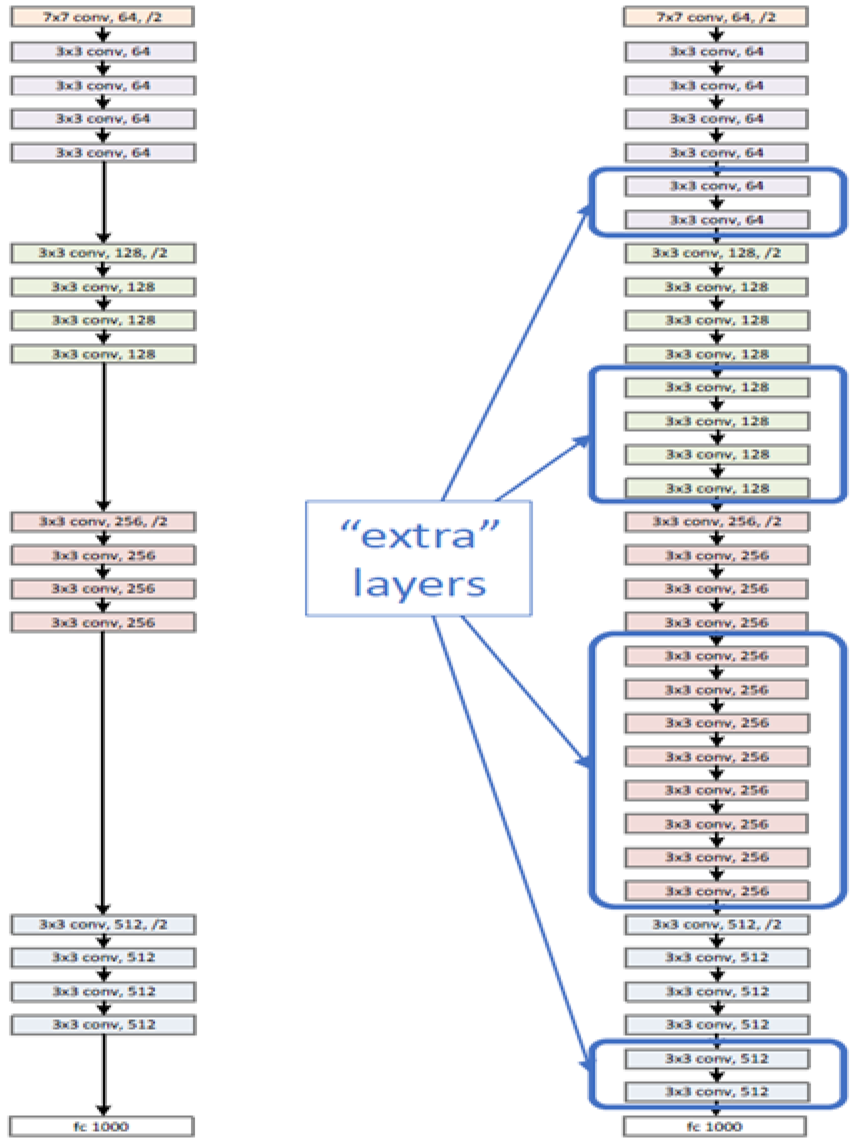


ImageNet ResNets



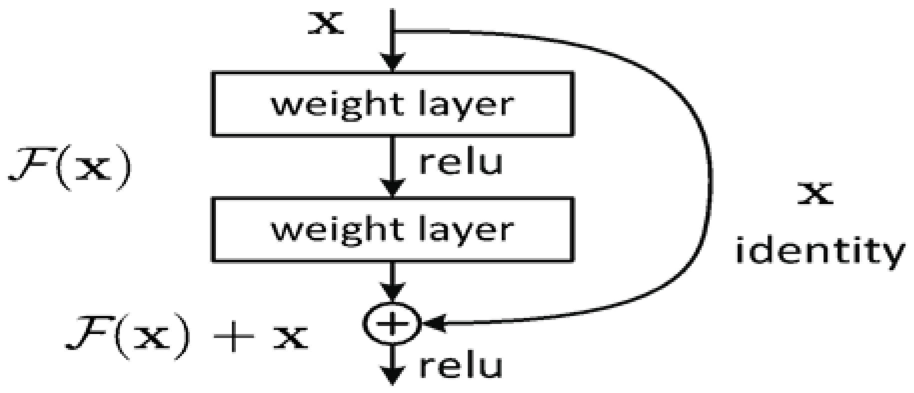
Simple Argument

- Naïve solution
  - If extra layers are an identity mapping, then training errors do not increase



BLANK SLIDE

Residual Learning

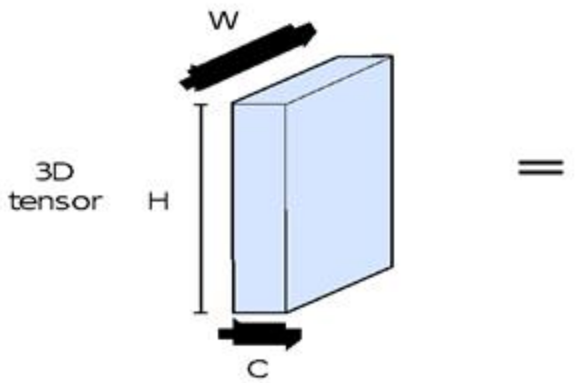


If Identity is optimal, easy to set weights as zero.

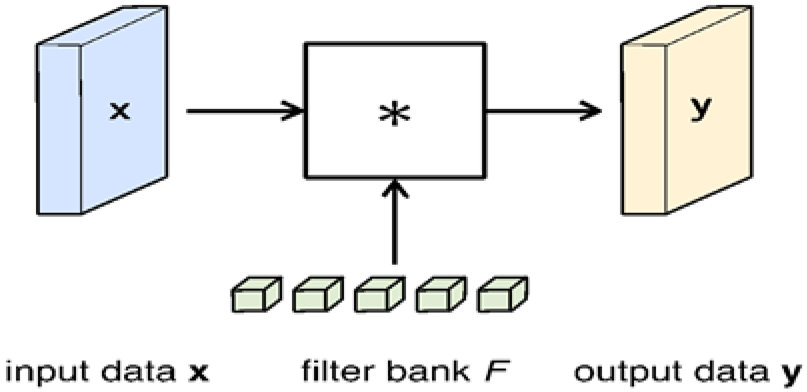
If optimal mapping is close to identity, easier to find small fluctuations.

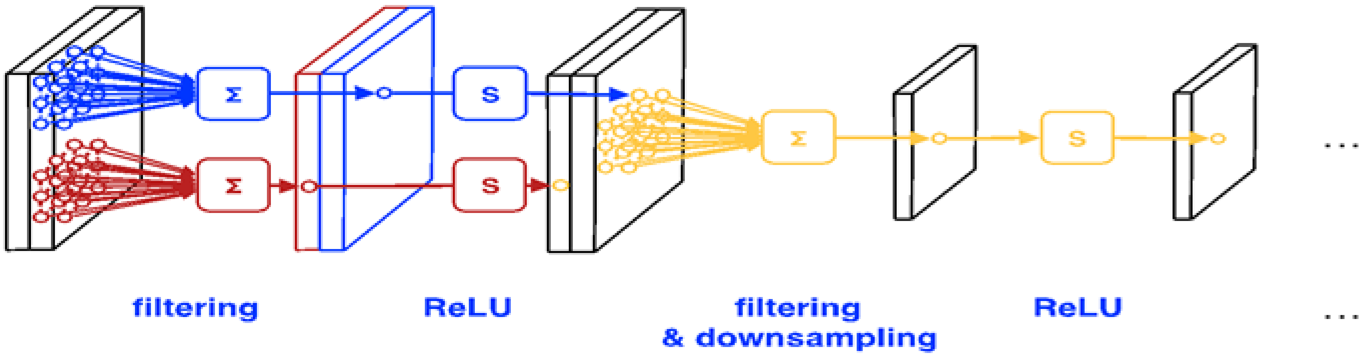
Let $\mathcal{H}(x)$ be the desired underlying mapping. Instead of learning it directly, fit a residual mapping of the form $\mathcal{F}(x) := \mathcal{H}(x) - x$.

CNNs: Summary



$$\boxed{\mathbf{y} = F * \mathbf{x} + b}$$





Thanks!!

Questions?