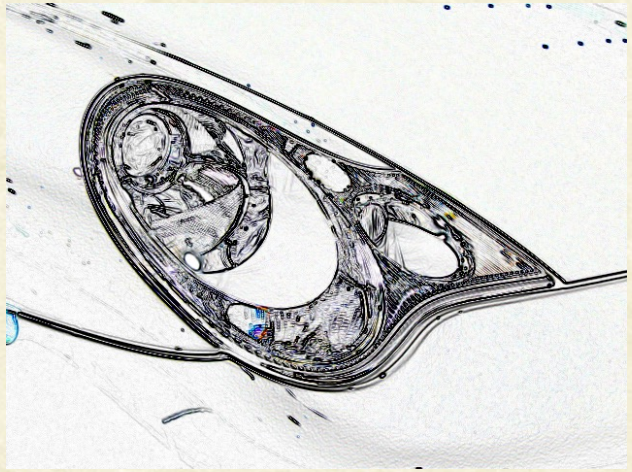
CS7.505: Computer Vision

Spring 2022: Feature Learning: Deep NNs

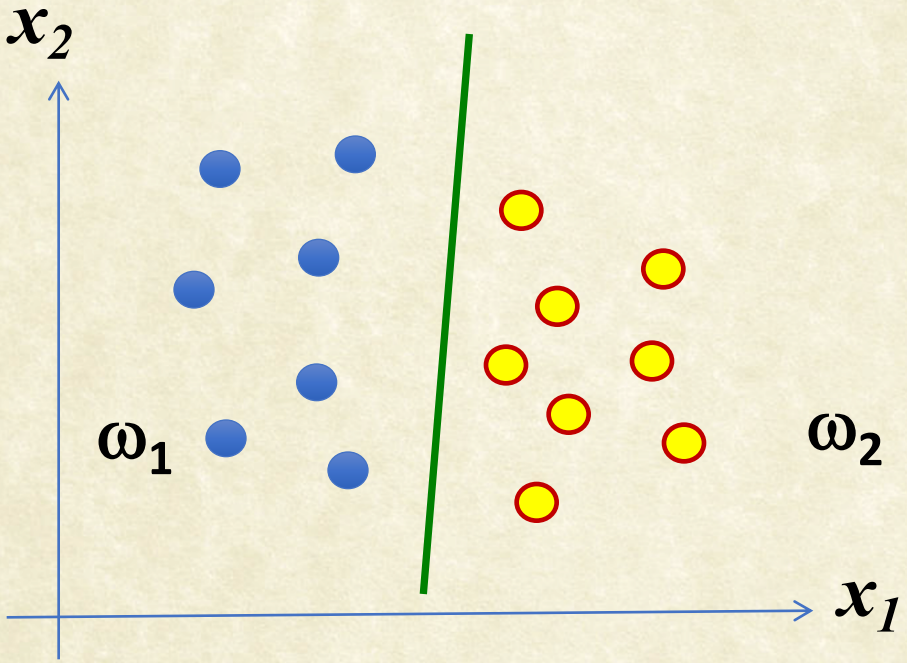




Anoop M. Namboodiri

Biometrics and Secure ID Lab, CVIT,

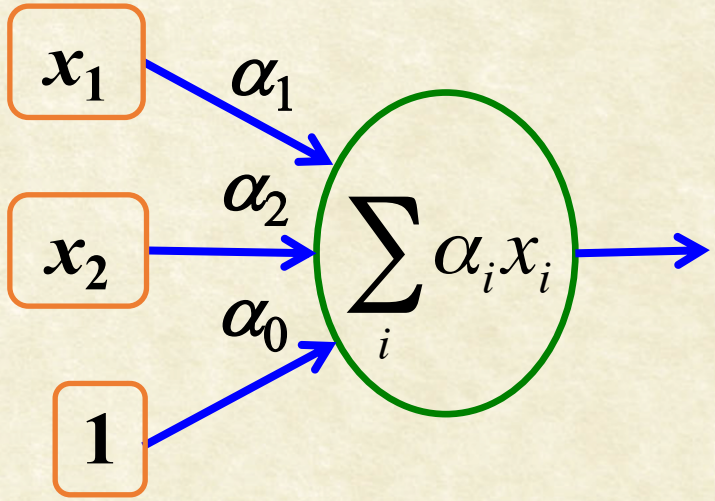
IIIT Hyderabad

Linear Classifier



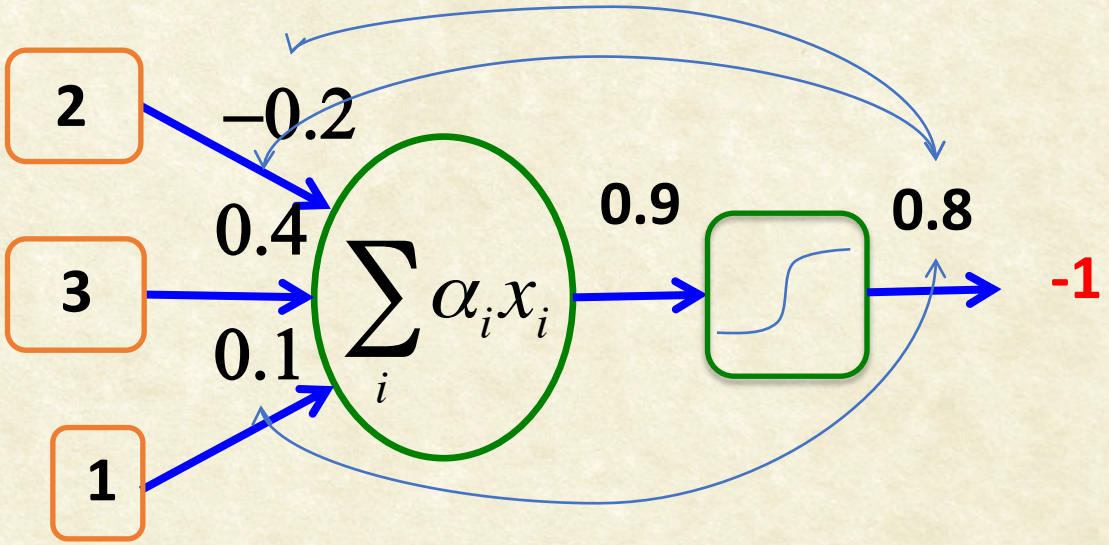
$$\alpha_1 x_1 + \alpha_2 x_2 + \alpha_0 = 0$$

Perceptron



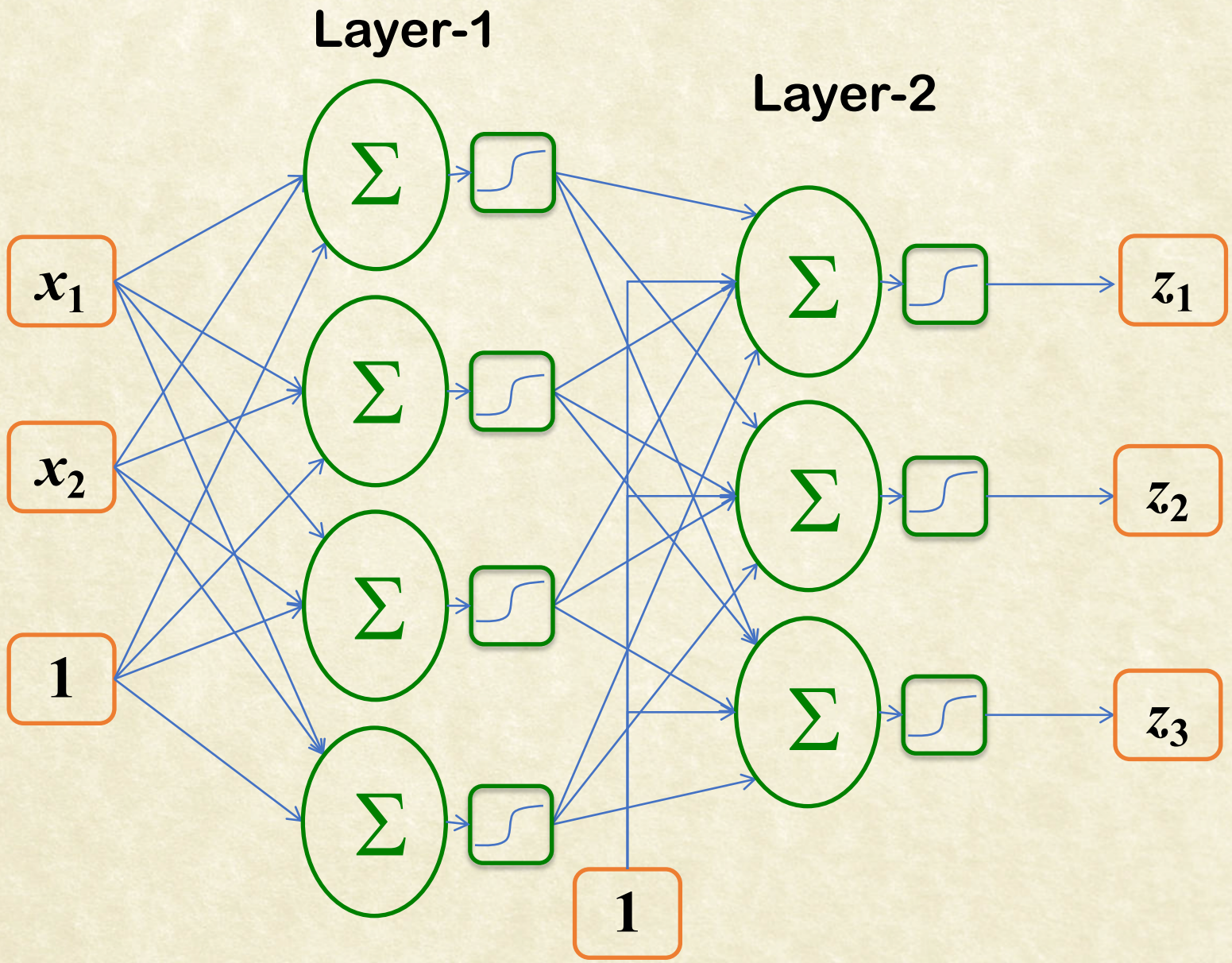
- A linear boundary separates two classes.
- Learning the classifier means learning the weights, α_i.

Perceptron Learning

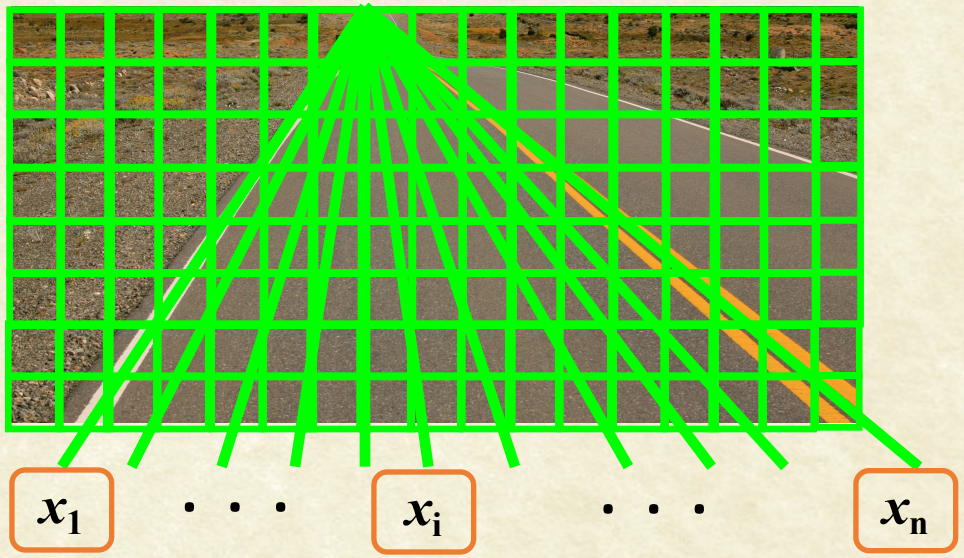


- Randomly Initialize the weights
- For each sample:
  - Feed a sample and find the output (forward pass)
  - Find the difference between actual and desired outputs (cost function)
  - Find the effect of each weight on the cost (derivative)
  - Update the weights with a learning rate (GD)

Multi-layer Perceptron

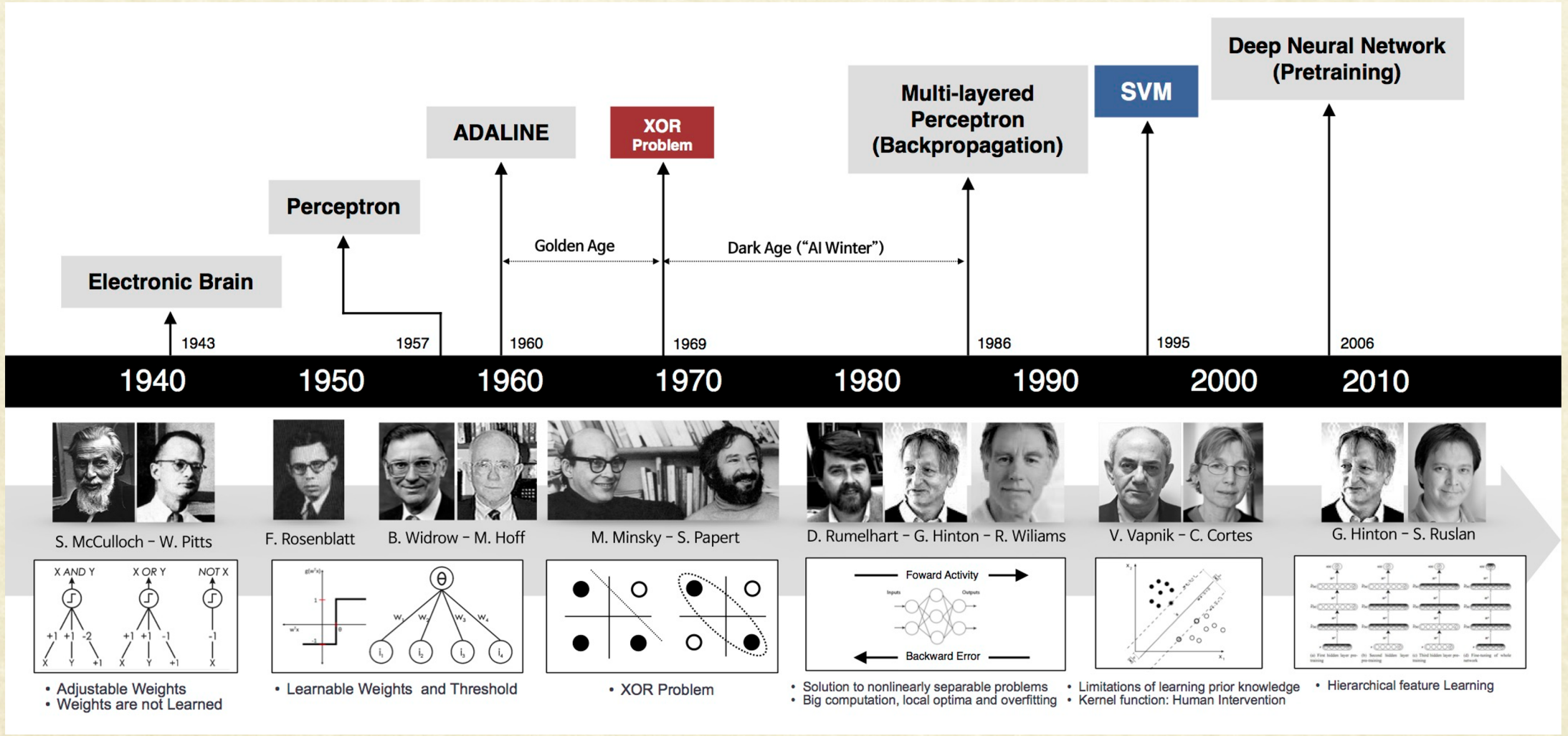


MLP in Computer Vision



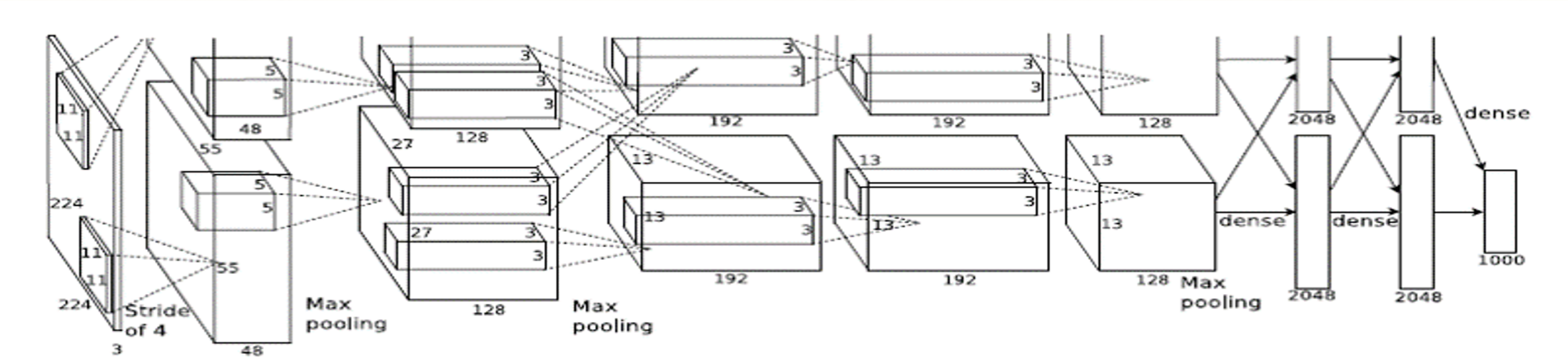
- 30x32 “Input Retina”
- 5 hidden units
- 30 output units
  - Sharp Left to Sharp Right

History of Deep Learning



ImageNet ILSVRC

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

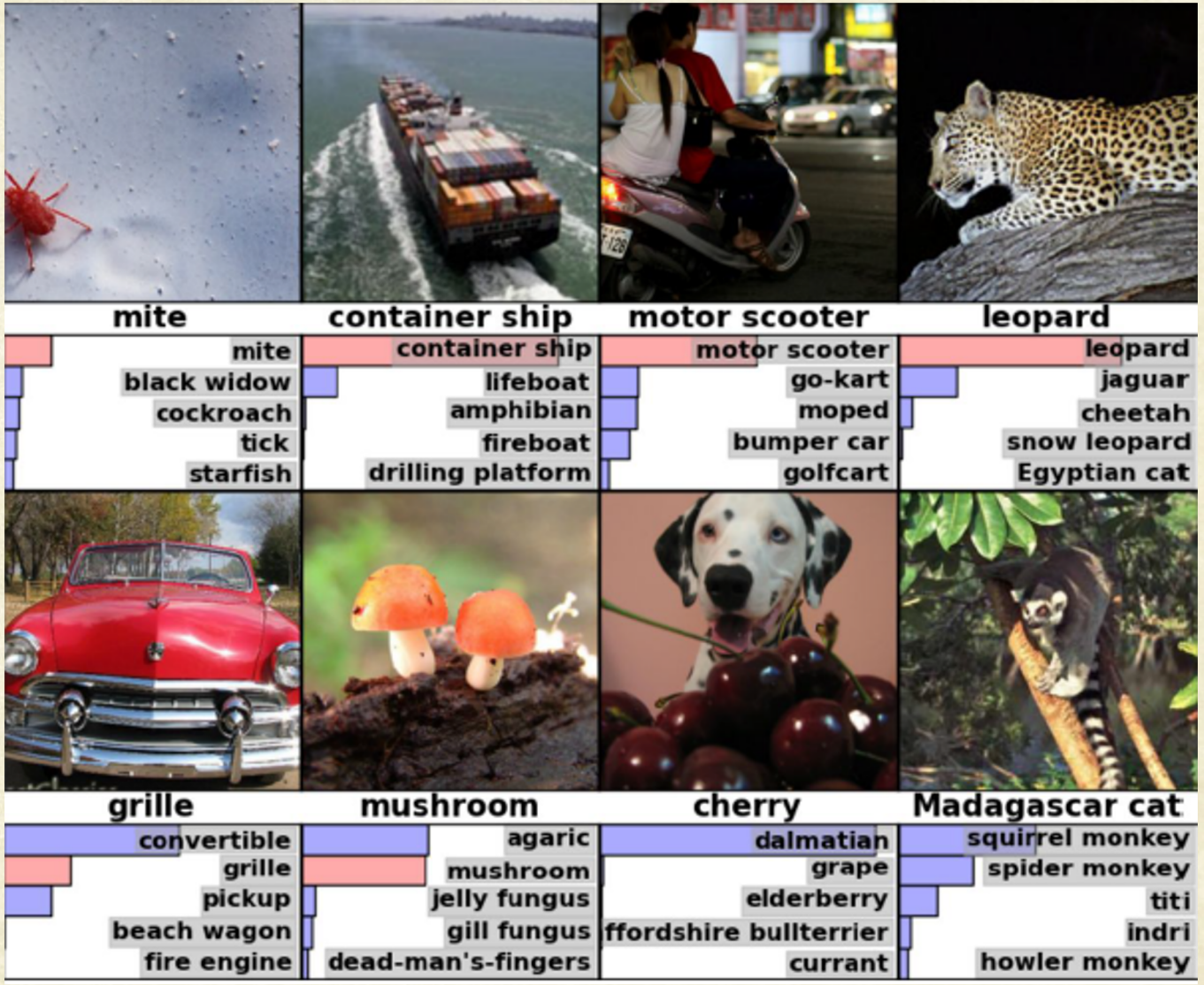
Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

ImageNet Classification Task:

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

ImageNet ILSVRC

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



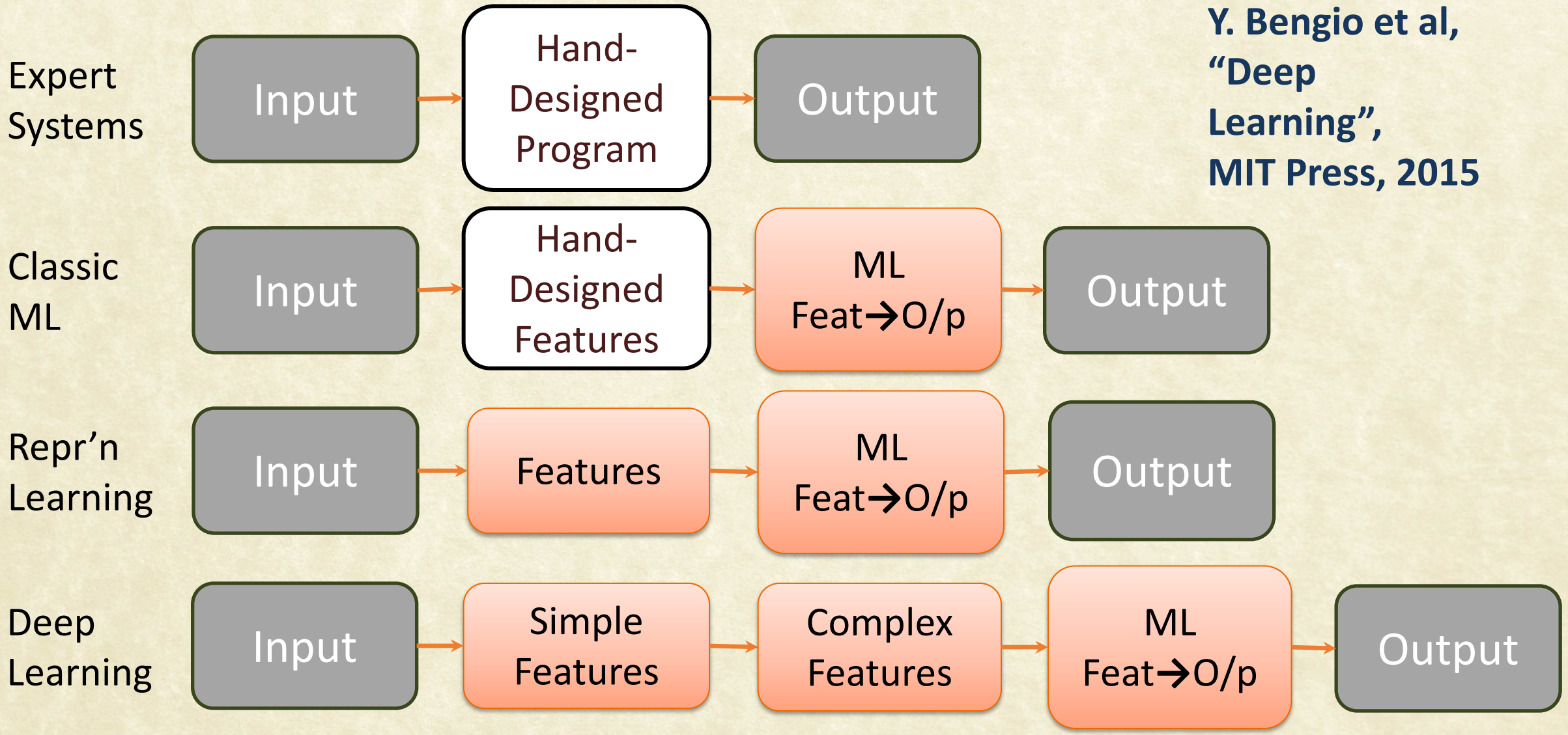
Success of “Deep Learning”: ImageNet Challenge

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 %

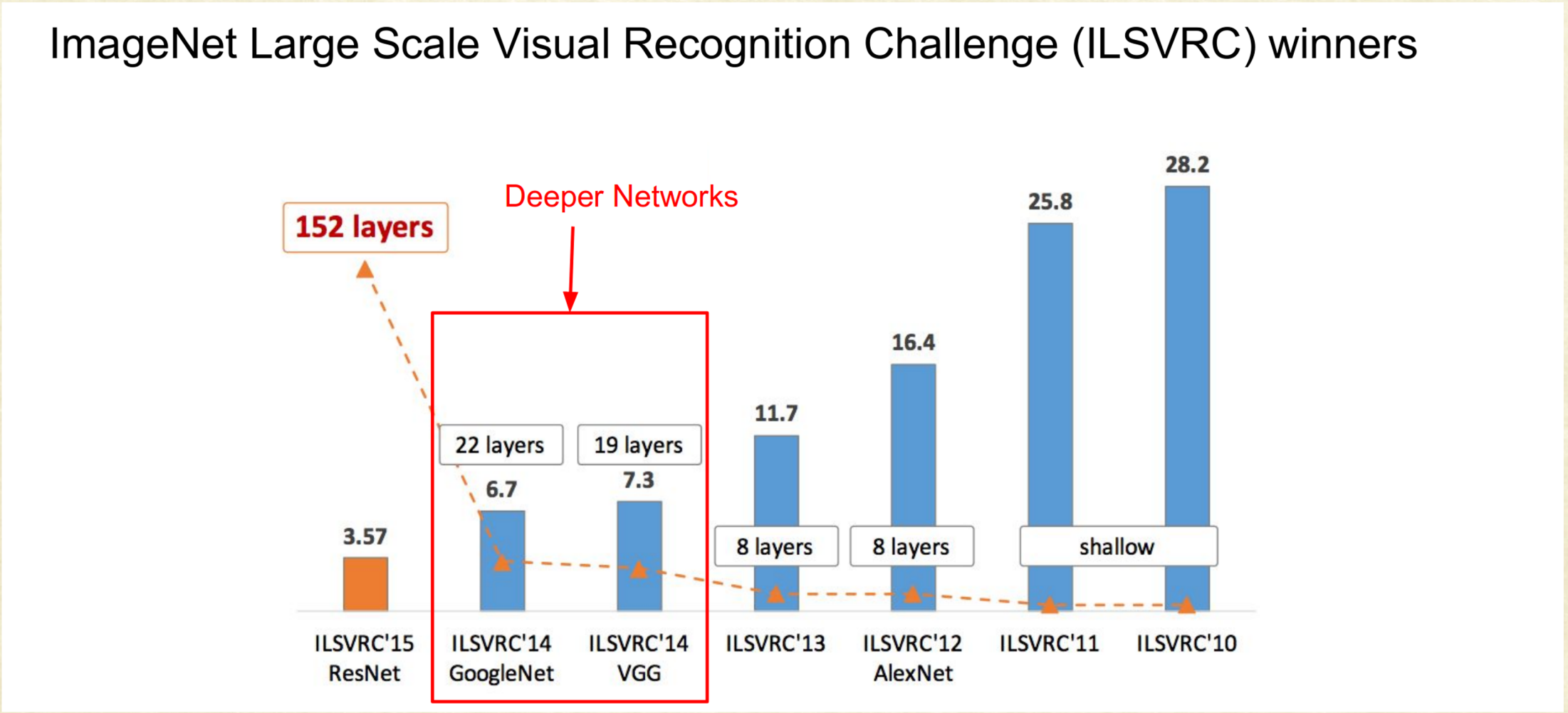
Mostly Deeper Networks
Smaller Convolutions
Many Specific Enhancements

Evolution of Learning



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

Getting Deeper



Convolutional Neural Networks

Course on CNN in Computer Vision at Stanford

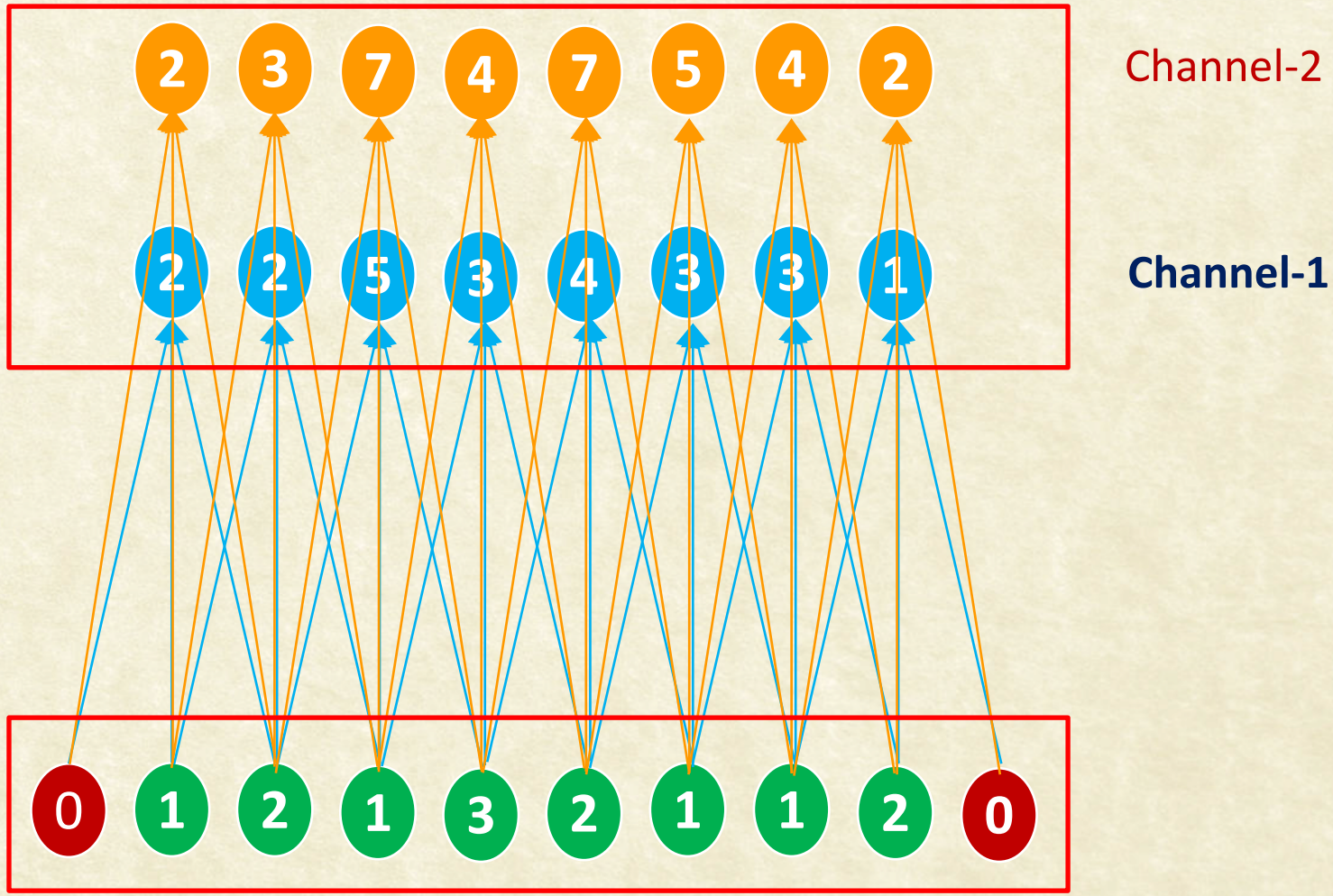
Fei-Fei Li, Andrej Karpathy and Justin Johnson

Convolution layer: in 1D

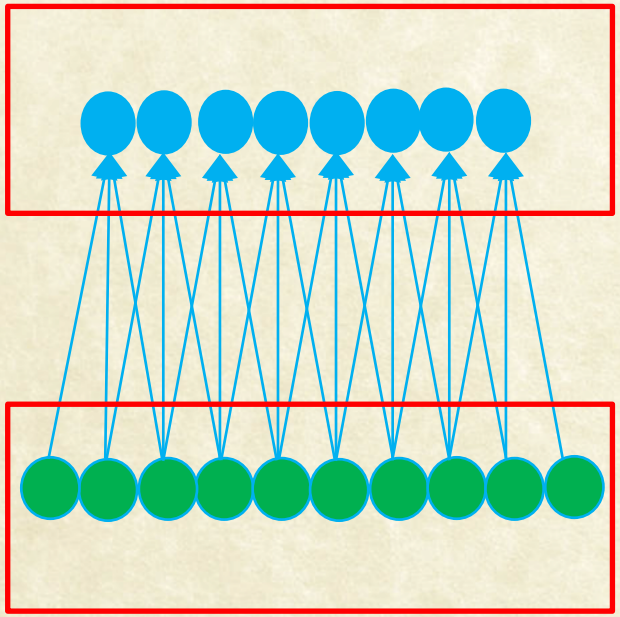
Filter-2 2 0 1

Filter-1 1 0 1

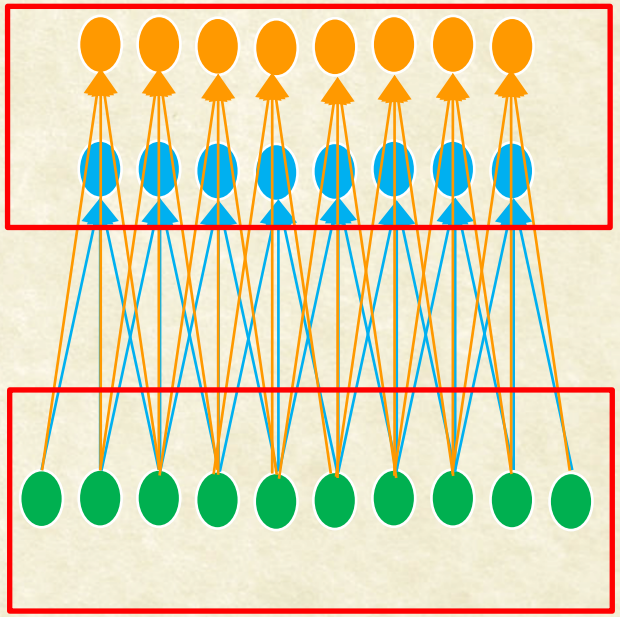
Two such filters/weights (2 X 3 = 6 !!)



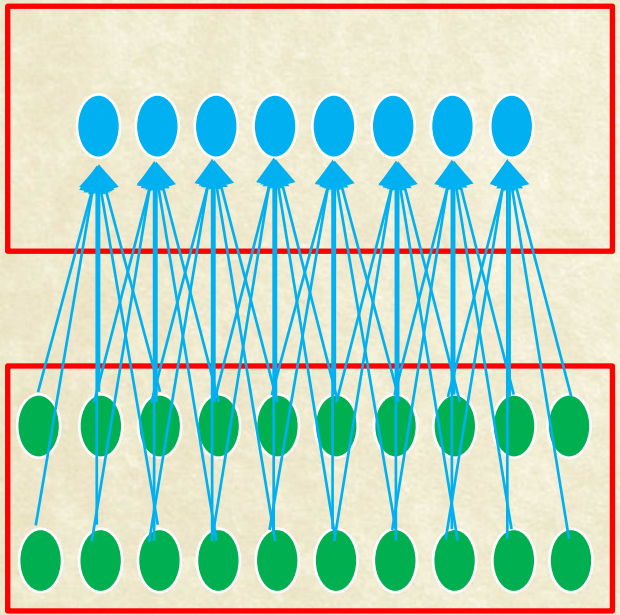
Convolution layer: Different Possibilities



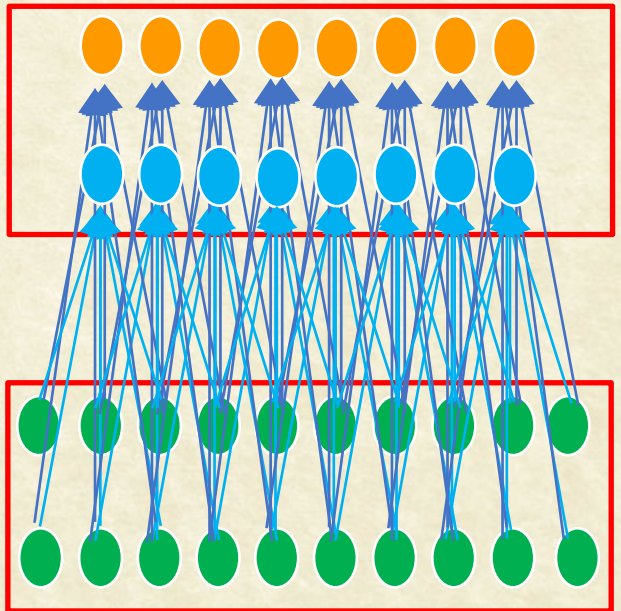
Channels:
- I/P =1
- O/P=1
- #Parameters = 3



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

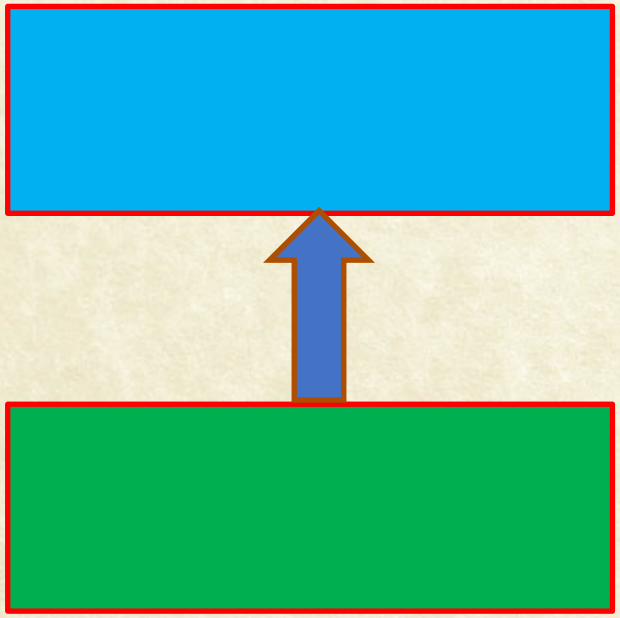


Channels:
- I/P =2
- O/P=1
- #Parameters = 6

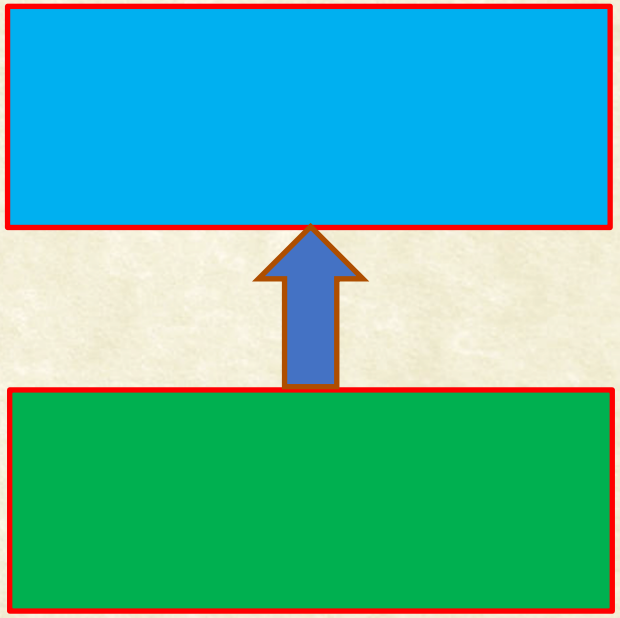


Channels:
- I/P =2
- O/P=2
- #Parameters = 12

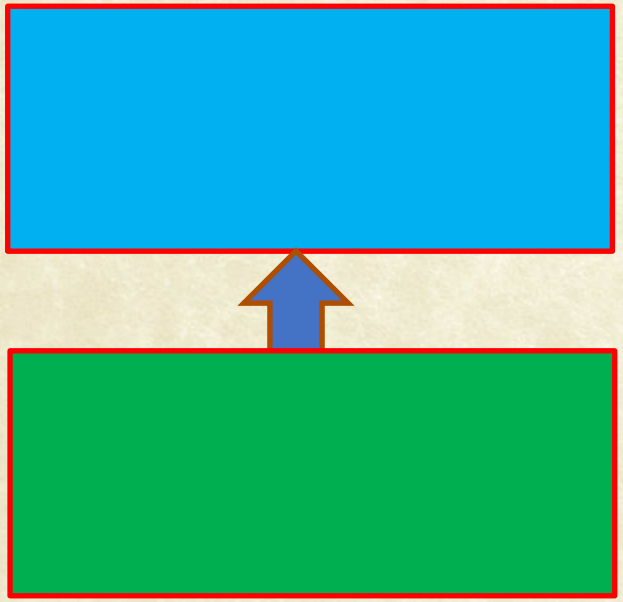
Convolution Layer: Simplified Picture



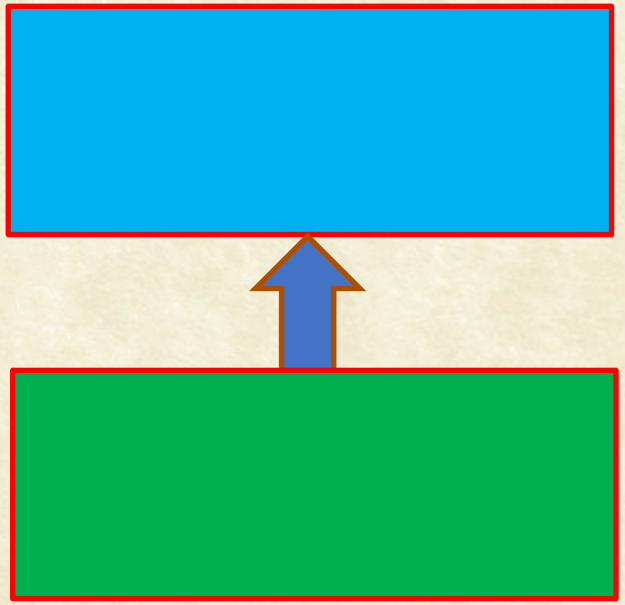
Channels:
- I/P =1
- O/P=1
- #Parameters = 3



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 in 2D

These are the network parameters to be learned.

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

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

Filter 1

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

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution in 2D

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

Filter 1

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

Dot product →

3 -1

6 x 6 image

Convolution in 2D

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

Filter 1

If stride=2

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

3 -3

Convolution in 2D

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

Filter 1

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

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Convolution in 2D

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

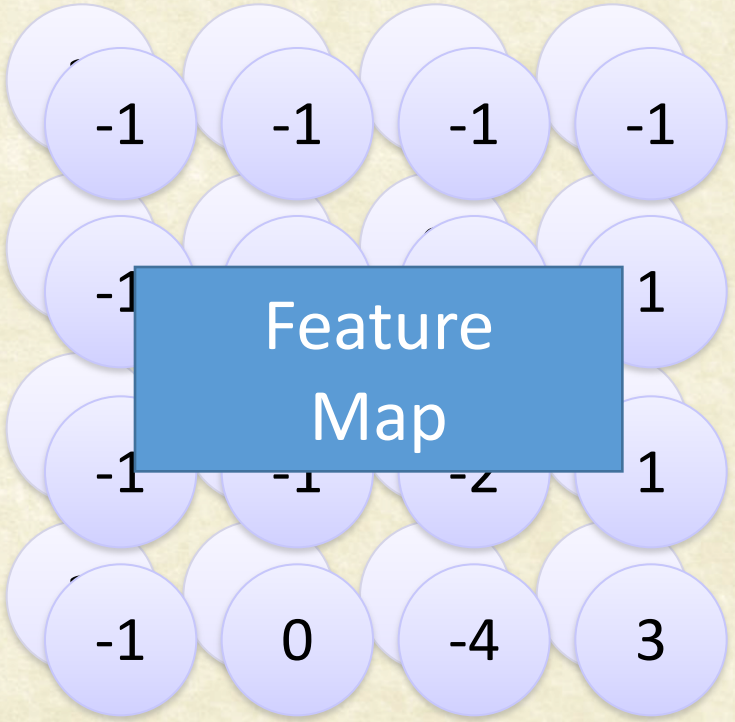
Filter 2

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

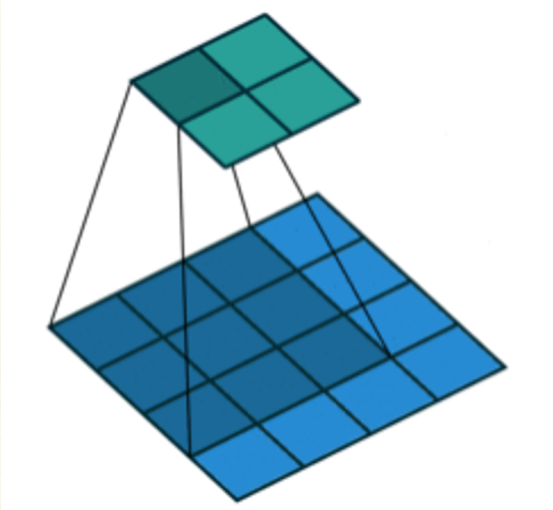
Repeat this for each filter



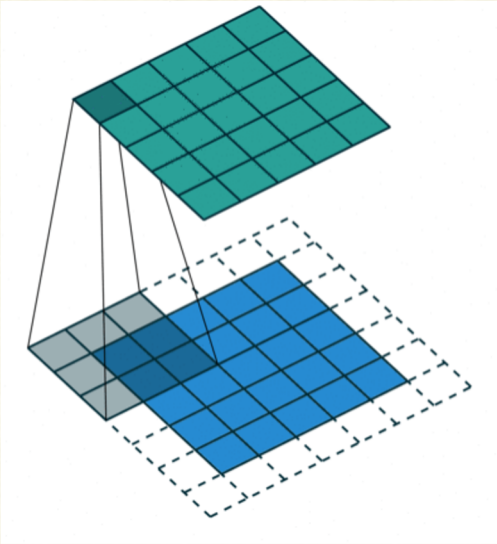
Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Window Size and Padding

- Window size
- Padding
- Stride



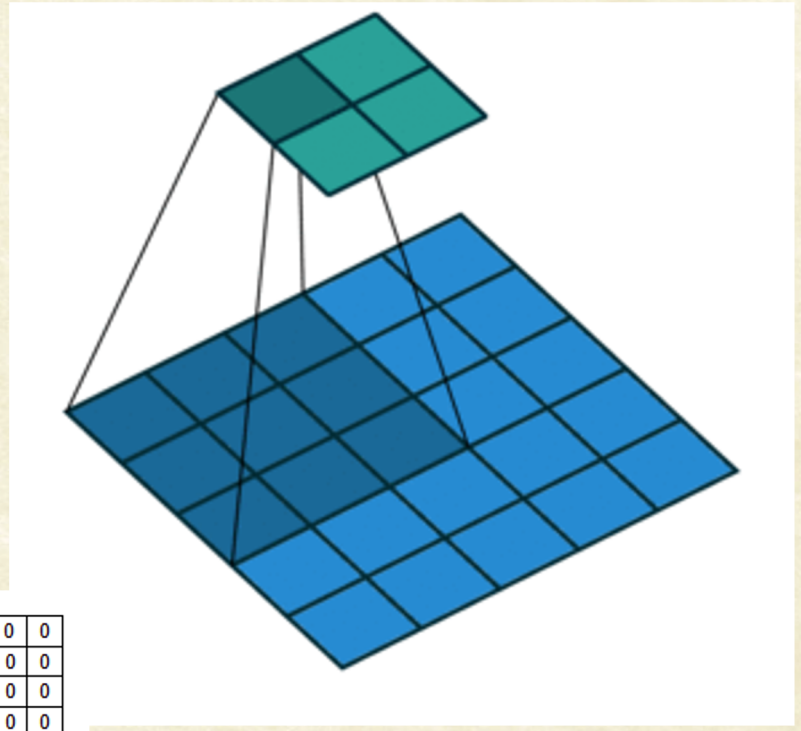
I/p size: 4x4
Window size: 3x3
Padding: 0
Stride: 1
O/p size: 2x2

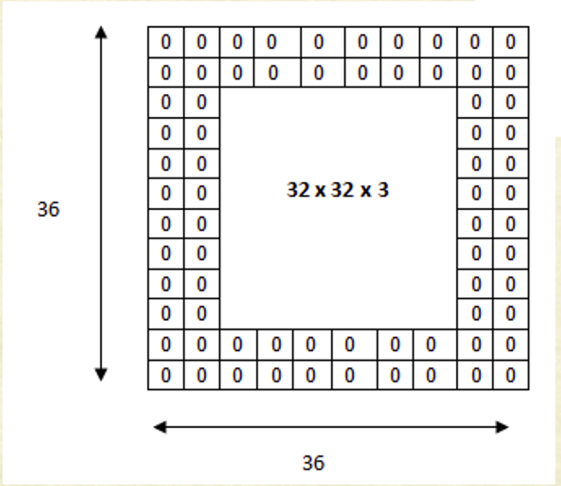


I/p size: 5x5
Window size: 3x3
Padding: 1
Stride: 1
O/p size: 5x5

Stride

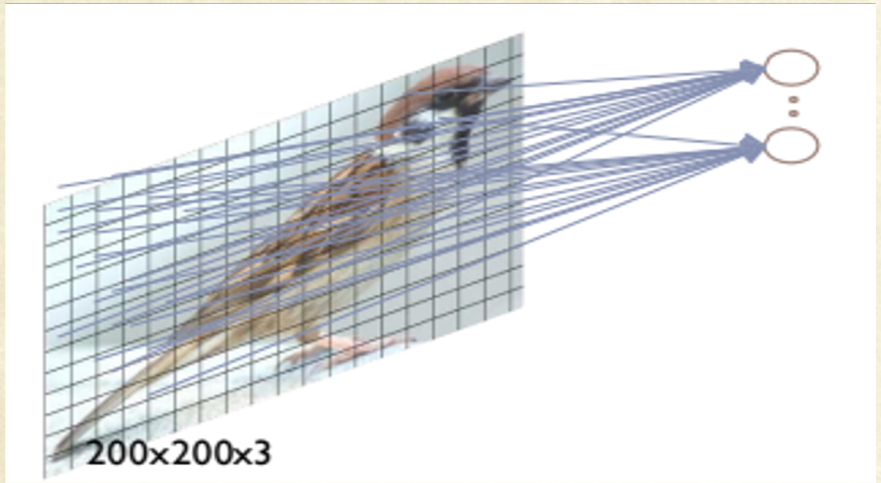
- Strides reduces dimension
- What is o/p size in terms of i/p size, window size, stride and padding?





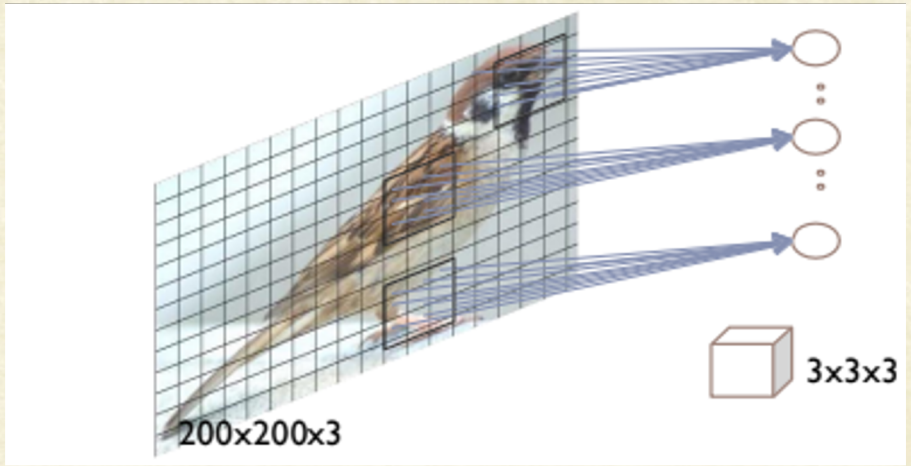
Convolution layer

- 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!

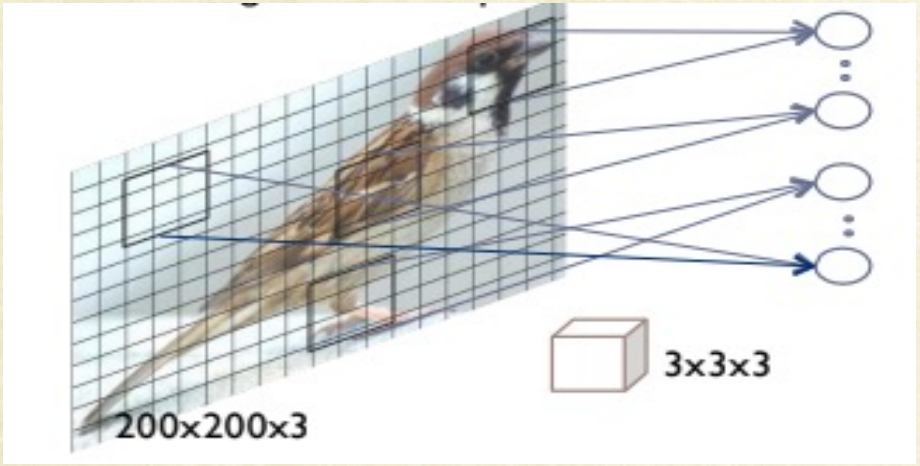
Parameter Calculations



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

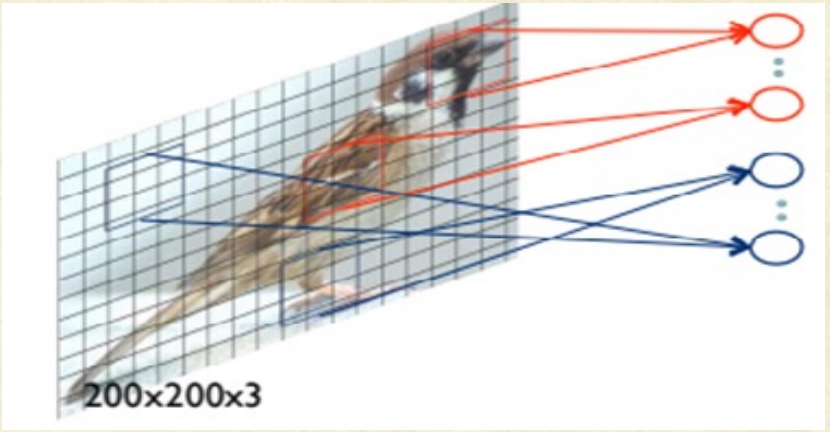
Convolution Layer

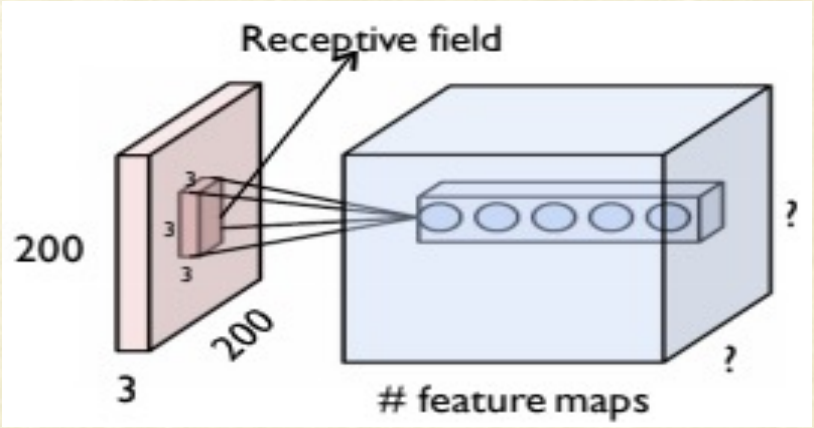
- **Convolutional layer with a single feature map**



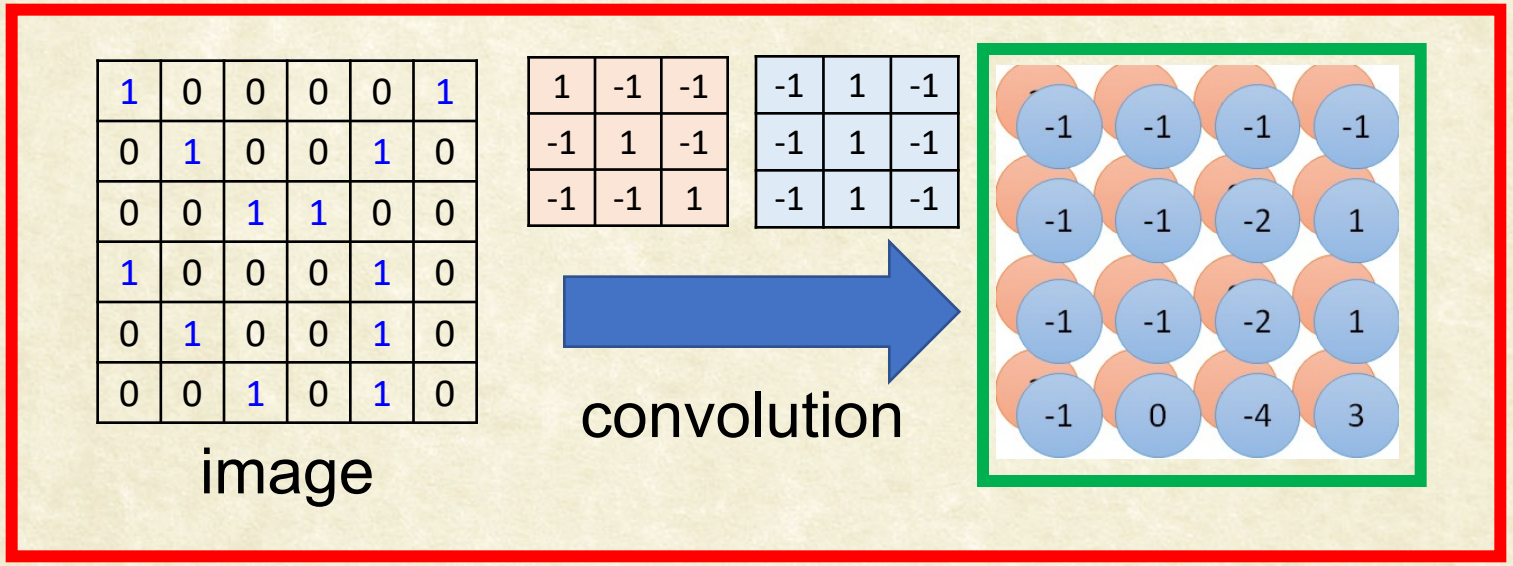
- #Hidden Units: 120,000
- #Params: 27 x #Feature Maps
- Sharing parameters
- Exploits the stationarity property and preserves locality of pixel dependencies

- **Convolutional layer with multiple feature maps**

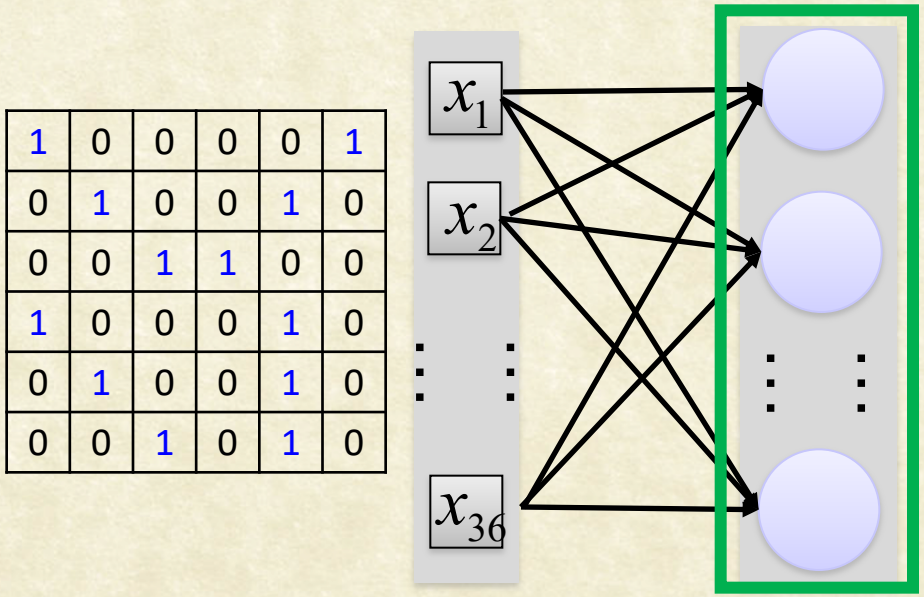




Convolution vs Fully Connected

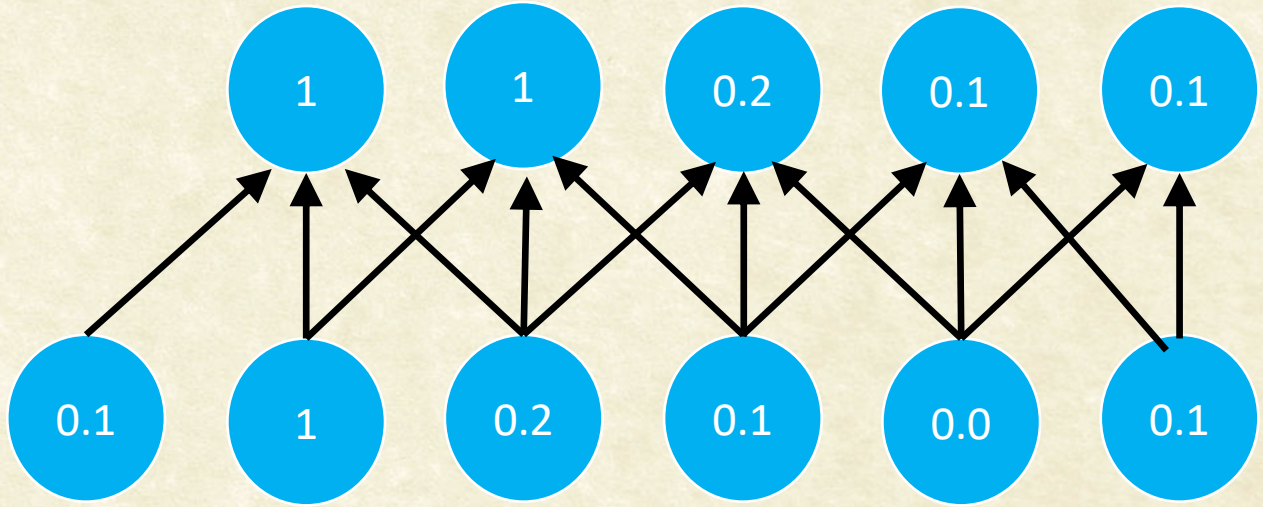


Fully-connected

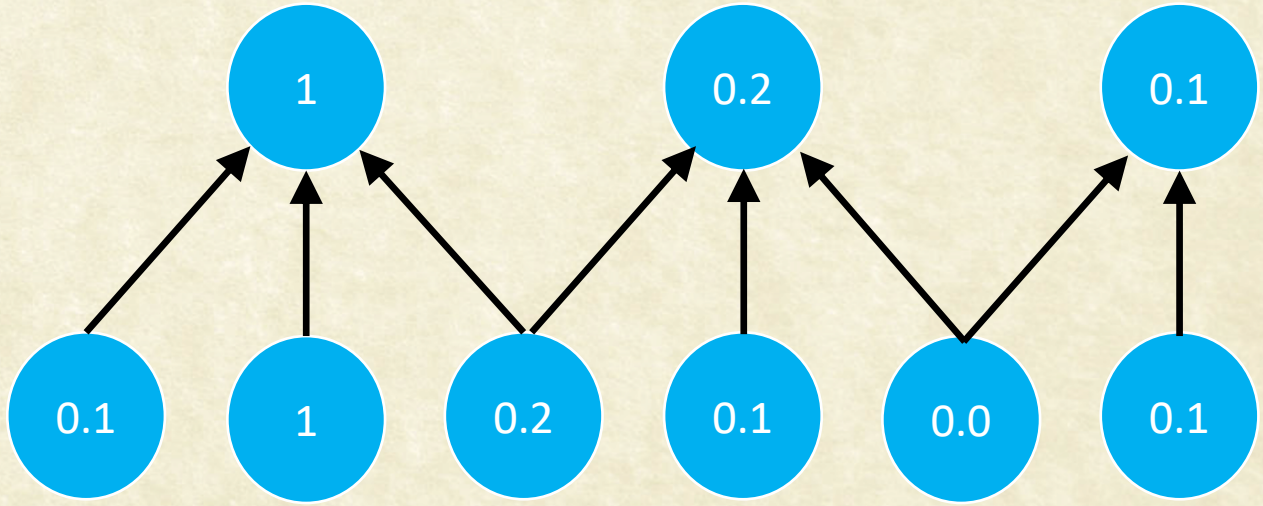


Max Pooling

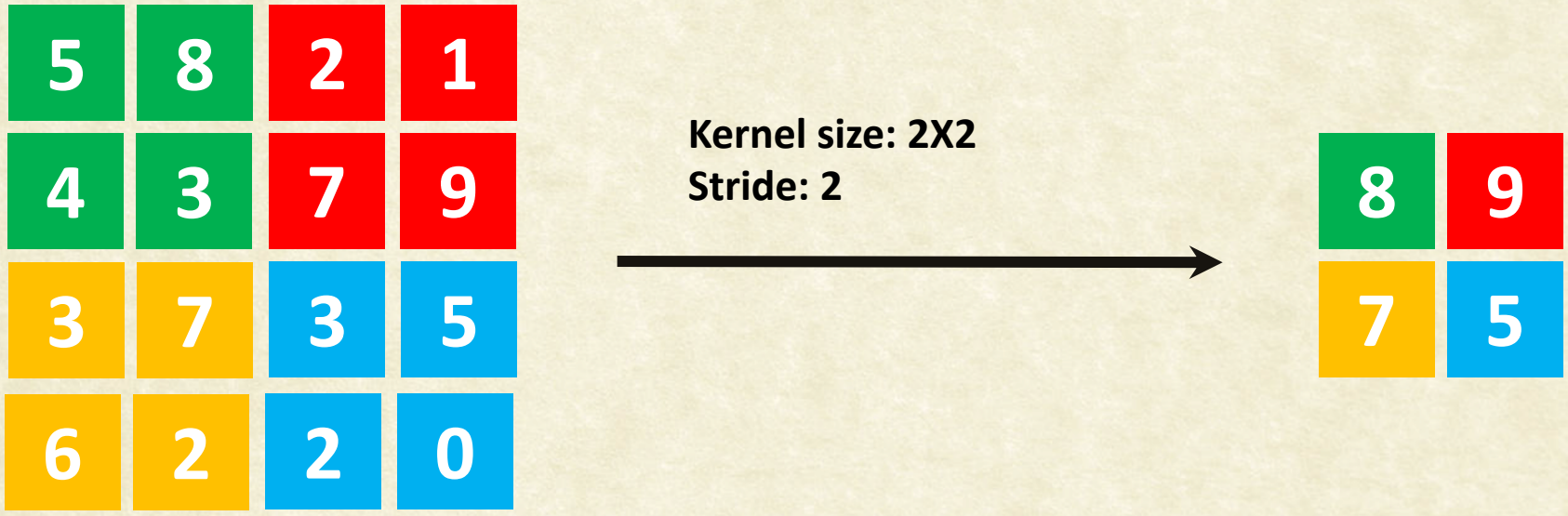
- Window size : 3
- Stride : 1



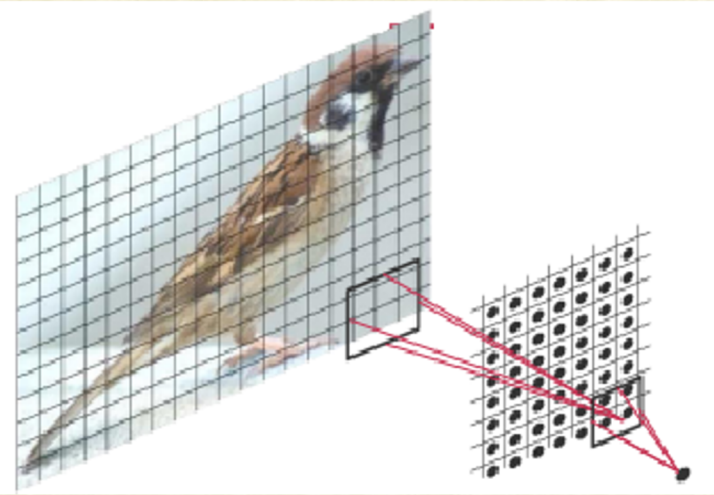
- Stride: 2



Max pooling in 2-D



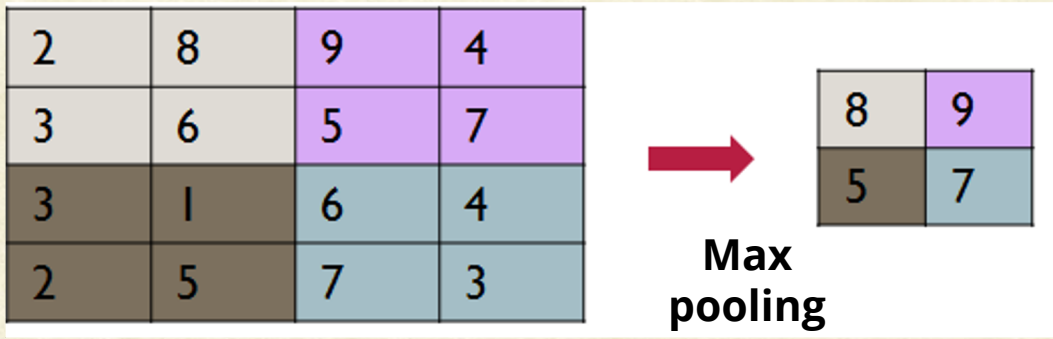
Pooling Layer



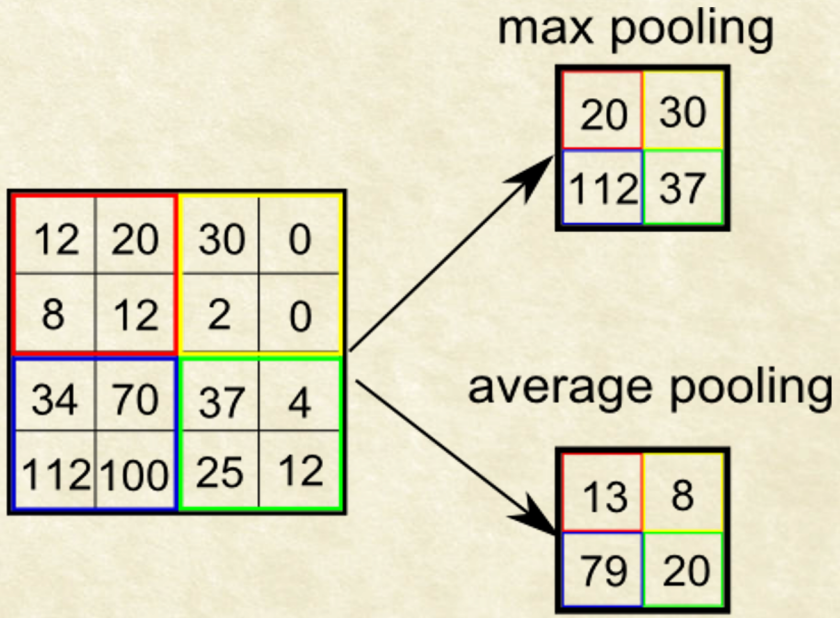
Pool Size: 2x2

Stride: 2

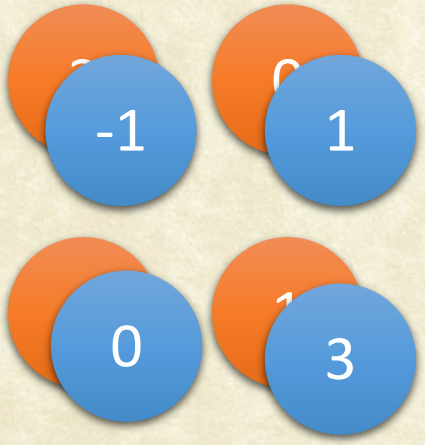
Type: Max



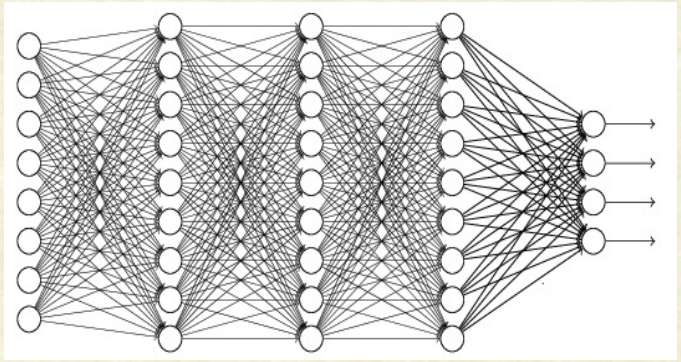
- Role of an aggregator.
- Invariance to image transformation and increases compactness to representation.
- Pooling types: Max, Average, L2 etc.



Flattening



Flattened



Fully Connected
Feedforward network

Terminologies

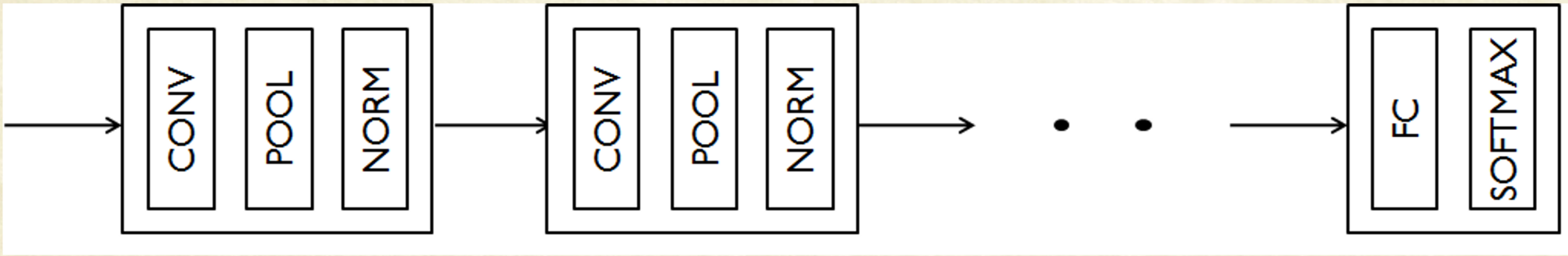
- # Input Channels
- # Output channels

- 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



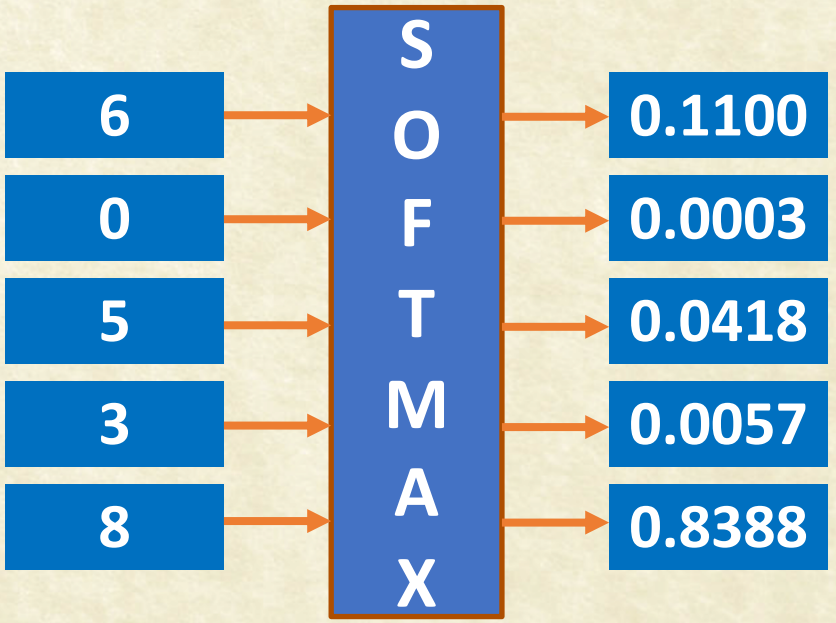
- Other layers
  - Pooling
  - Normalization
  - Fully connected
  - etc.

Softmax

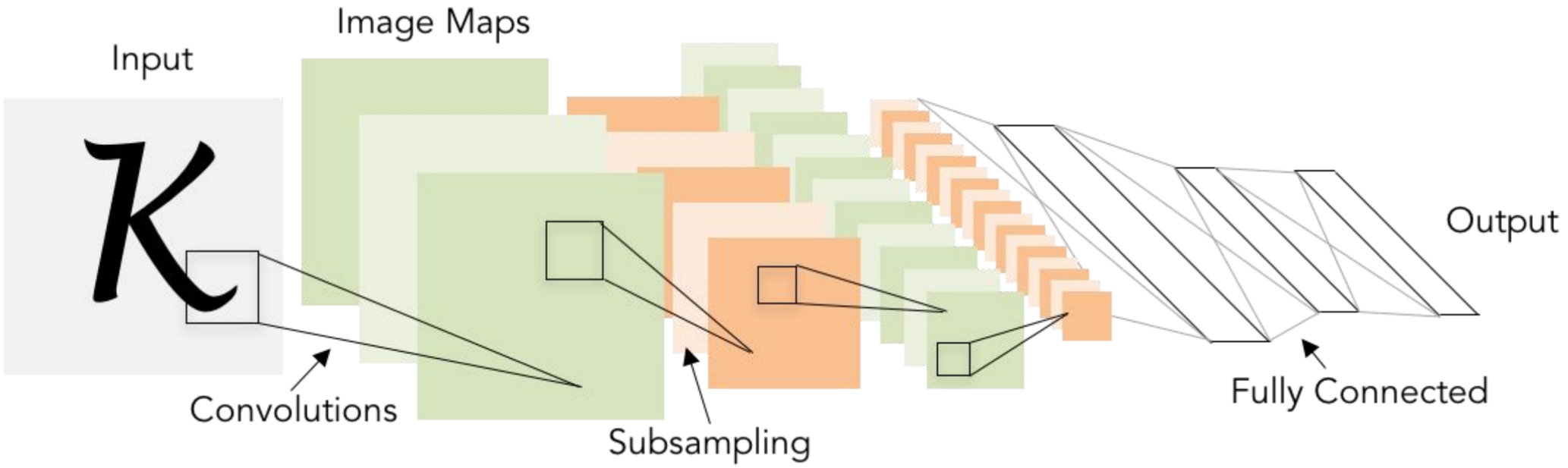
```
Out[12]: array([ 6.,  0.,  5.,  3.,  8.])

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

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

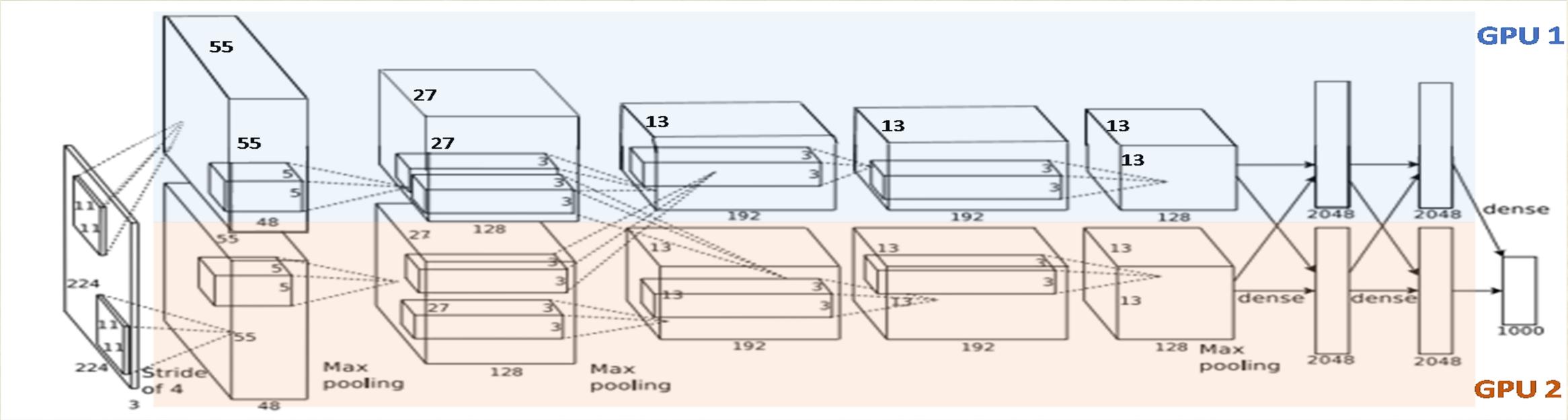


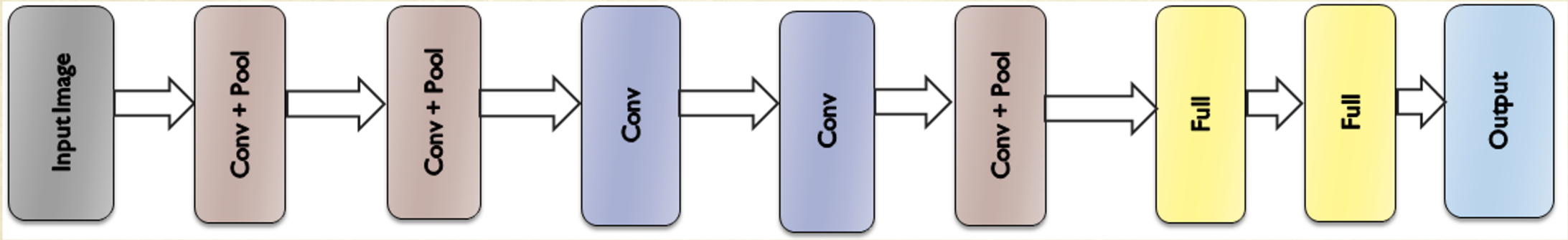
LeNet



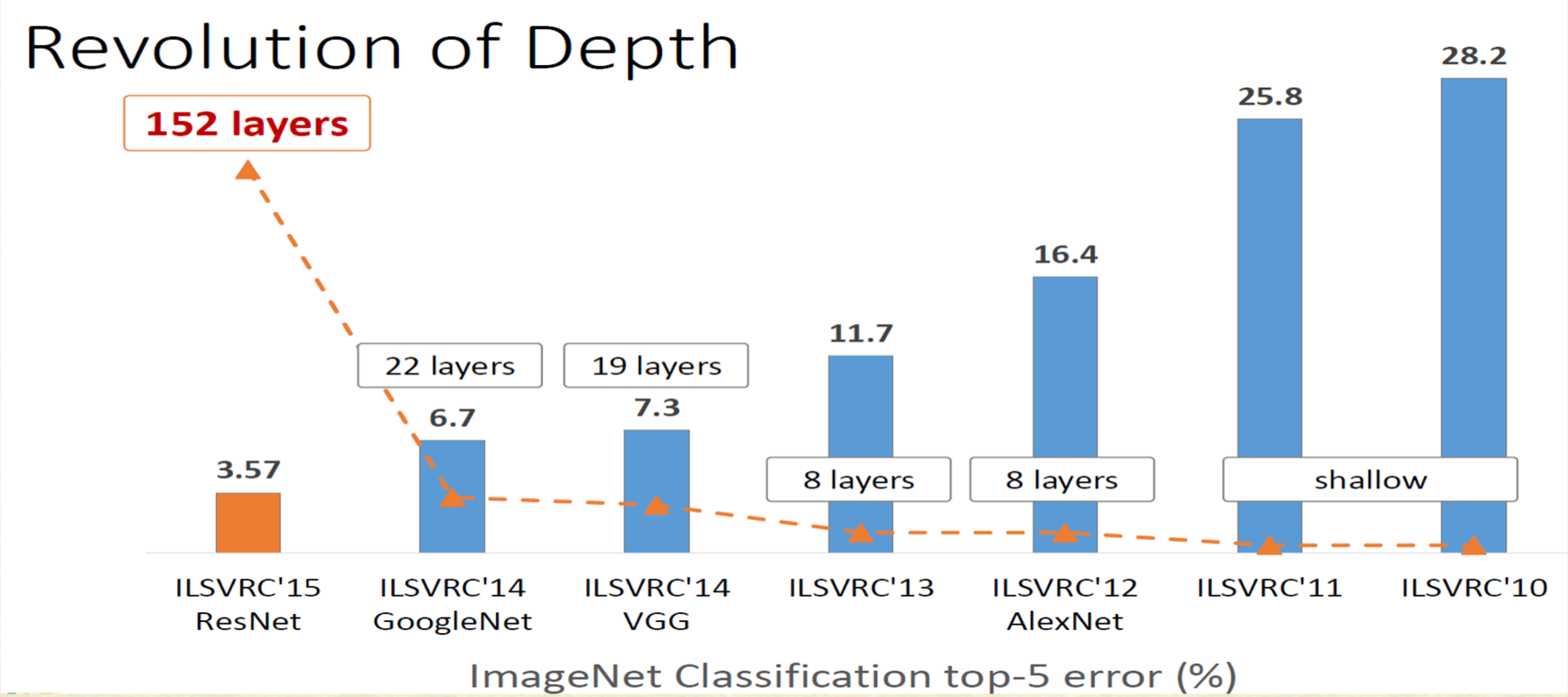
Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

AlexNet Architecture





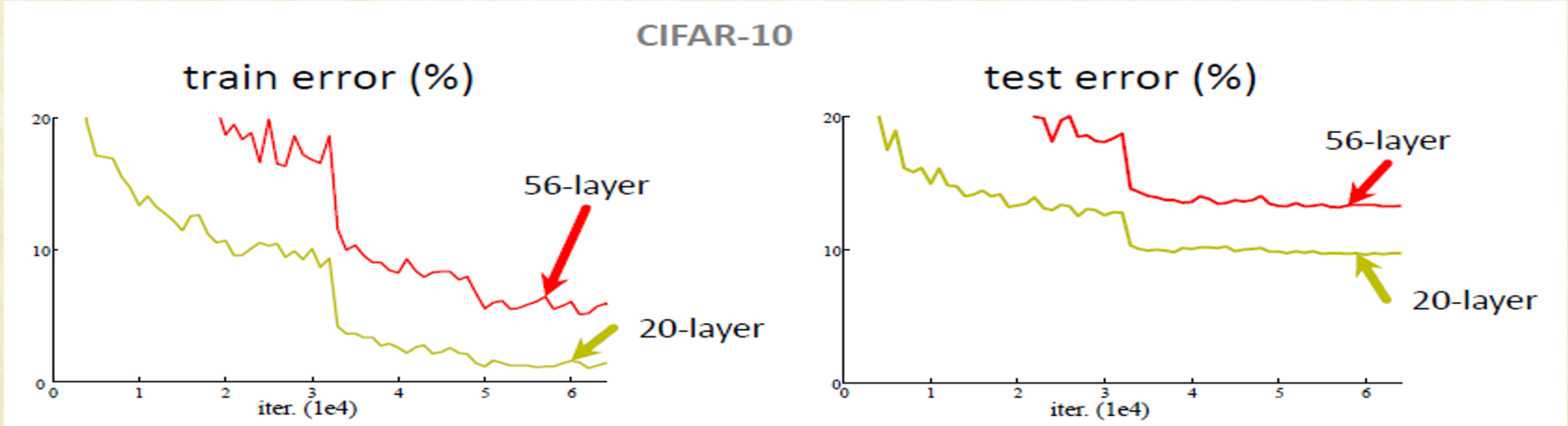
Residual Net [CVPR2016]

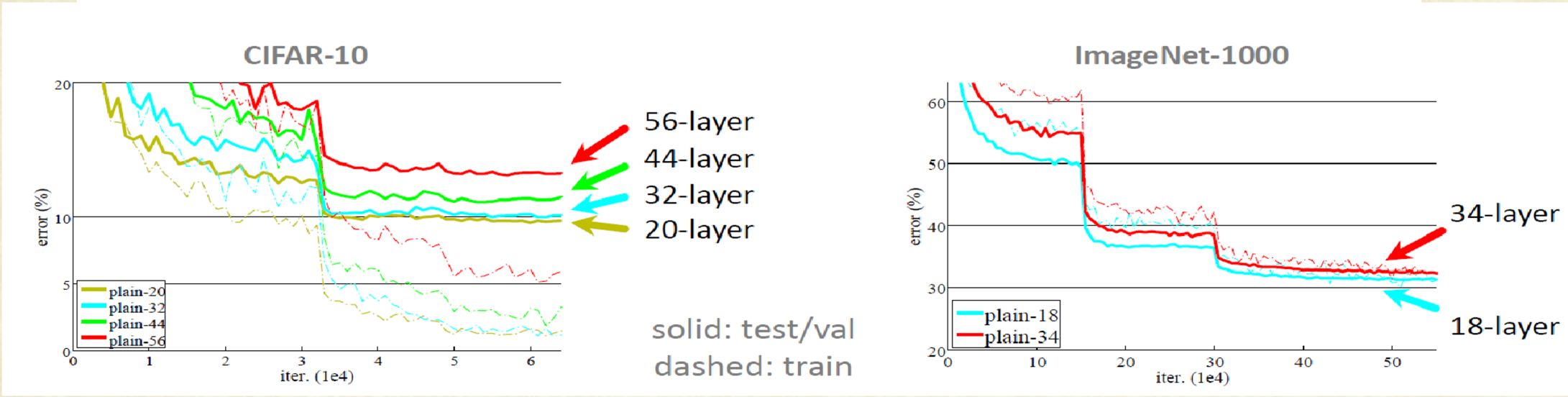


Challenge with Depth

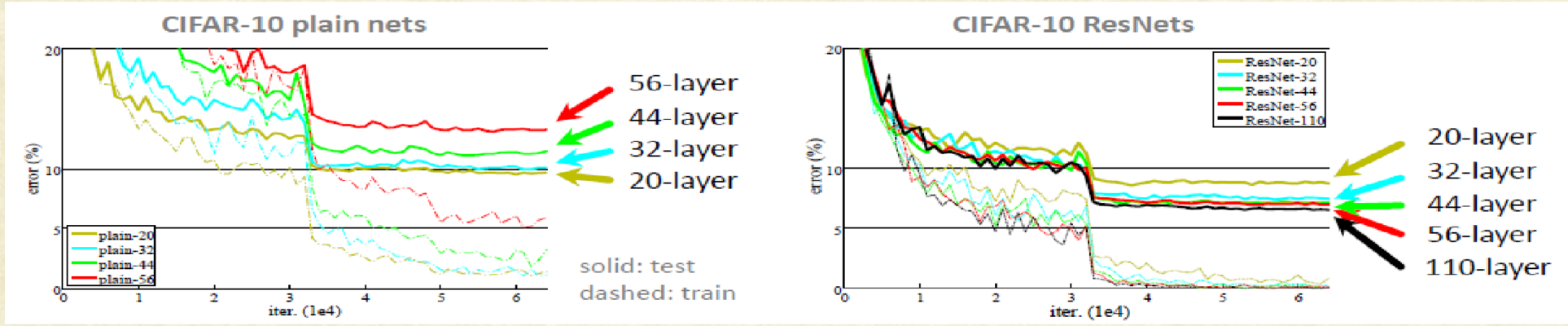
- Vanishing Gradients
  - Error signal don’t reach (enough) the early layers
  - Multiplication of many small numbers (less than one) and become almost zero
- Exploding gradients
  - If gradients are large, product become too big and huge changes in weights

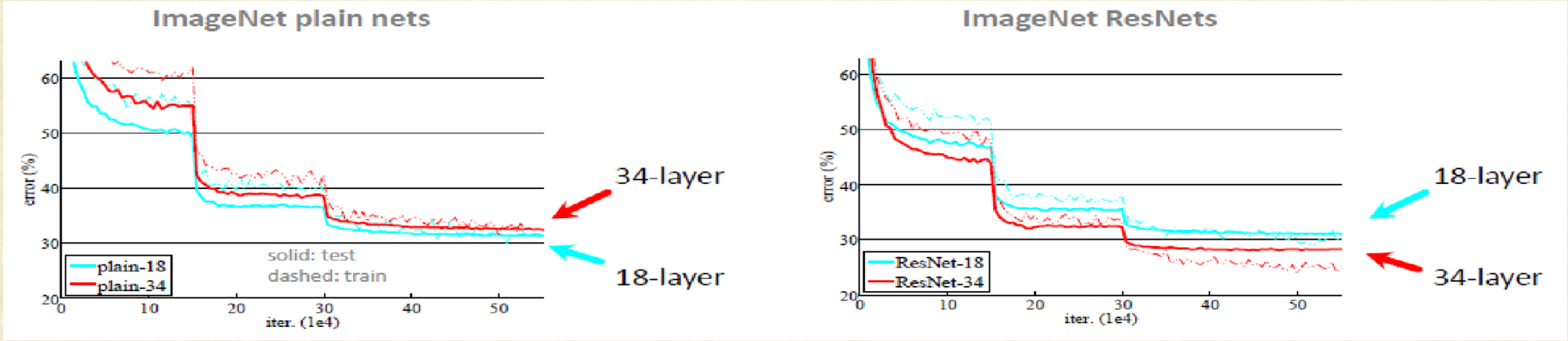
Problems with Simple Deep (cf: Resnet)





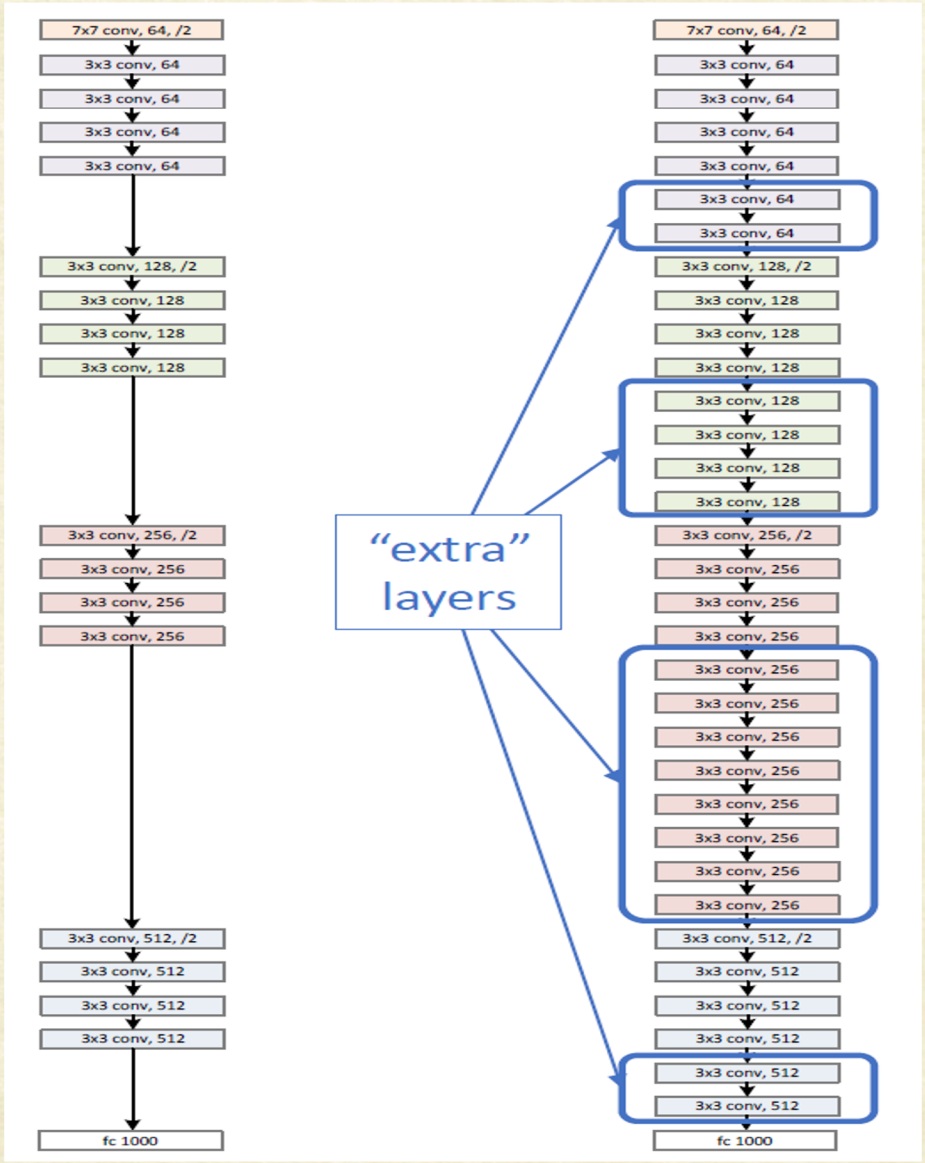
PlainNet Vs ResNet





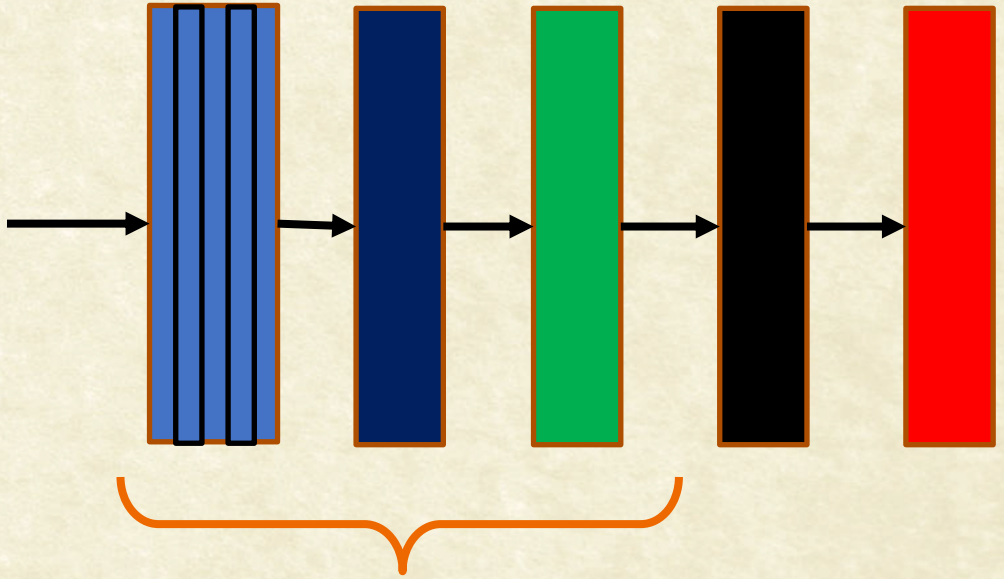
Simple Argument

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



Summary: Layers of a Neural Network

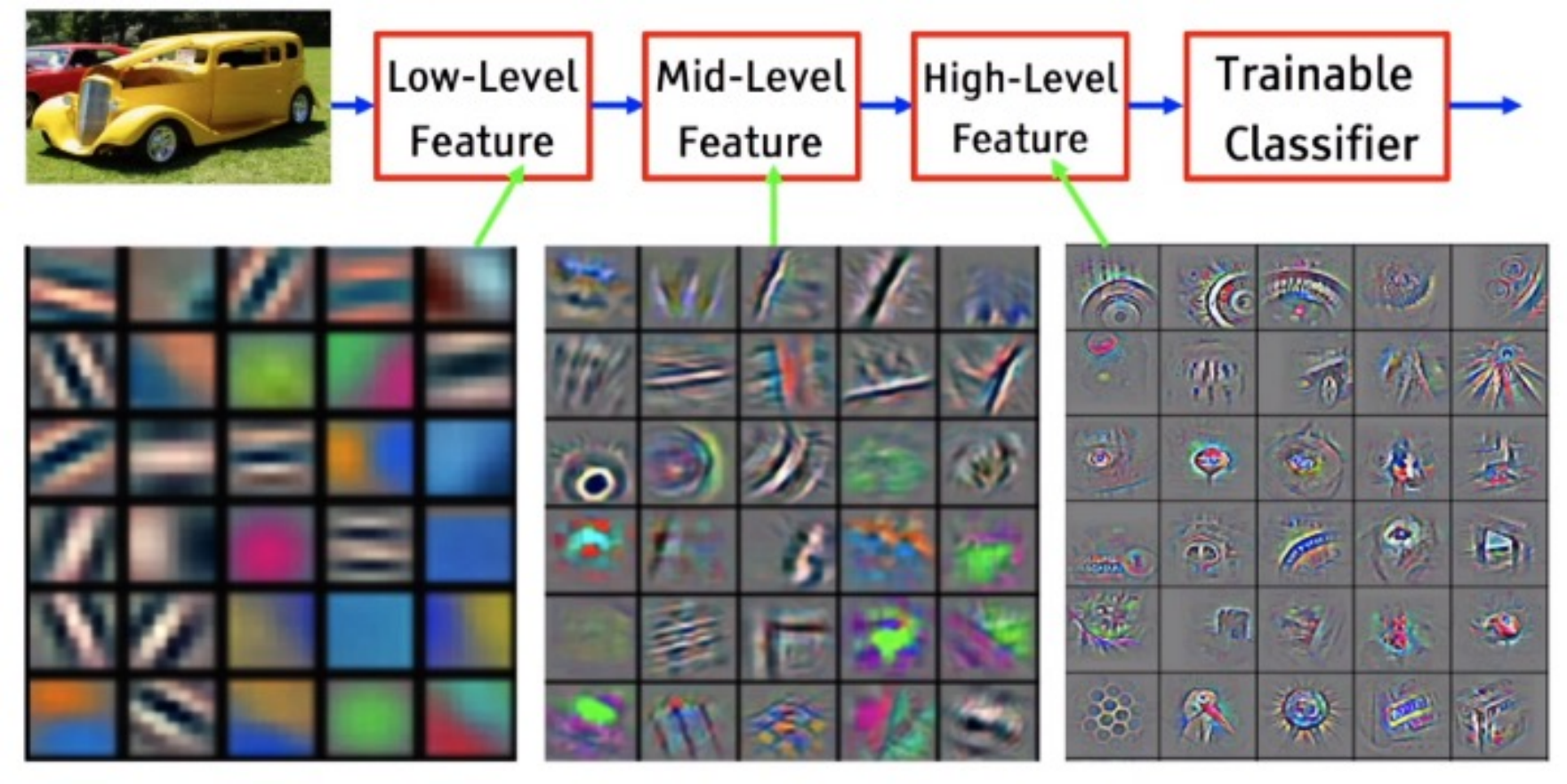
- Based on the connection pattern and operations, we can think of a layer in a Neural Network as:
  - Convolutional
    - A Layer can have multiple Channels
  - Non-Linear (often not drawn)
  - Max-Pooling
  - Fully Connected
  - Soft Max



This is often repeated multiple times

Deep Learnt Features

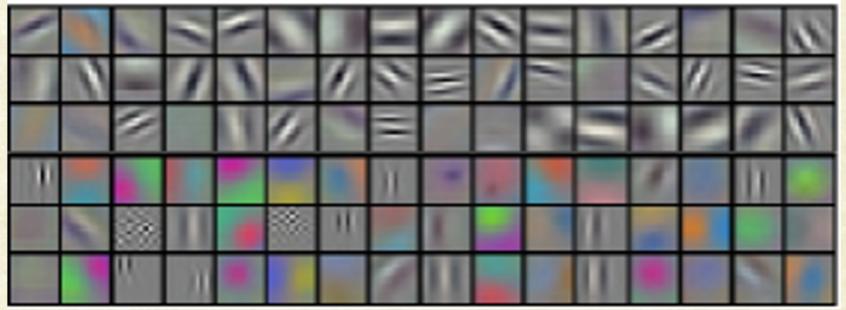
- It's deep if it has more than one stage of non-linear feature transformation.



Visualizing CNNs

- CNNs are cool ☺ but some of the below questions need answers before we move forward :-
- How do I interpret the learned filters?
- What is it that stimulates/excites a neuron?
- How do I decide the architecture or improve existing ones?

Source: Krizhevsky et.al. NIPS'12

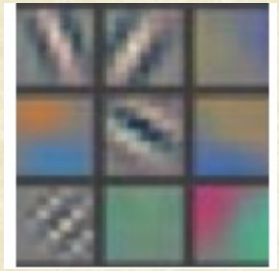


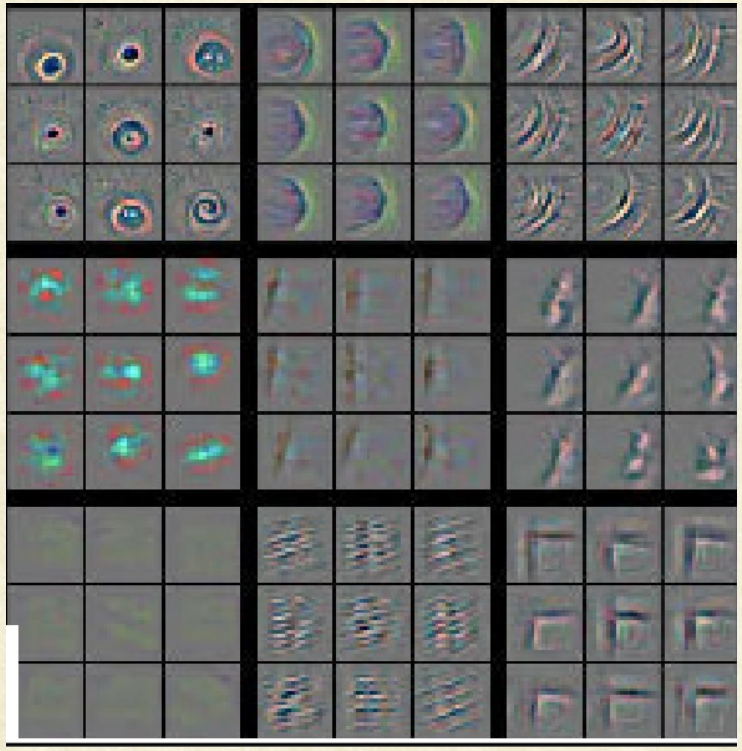
Visualizing the first conv. layer is possible but how about the later layers.

?

Zeiler and Fergus , Visualizing and Understanding Convolutional Networks,. ECCV 2014

Composition of Filters





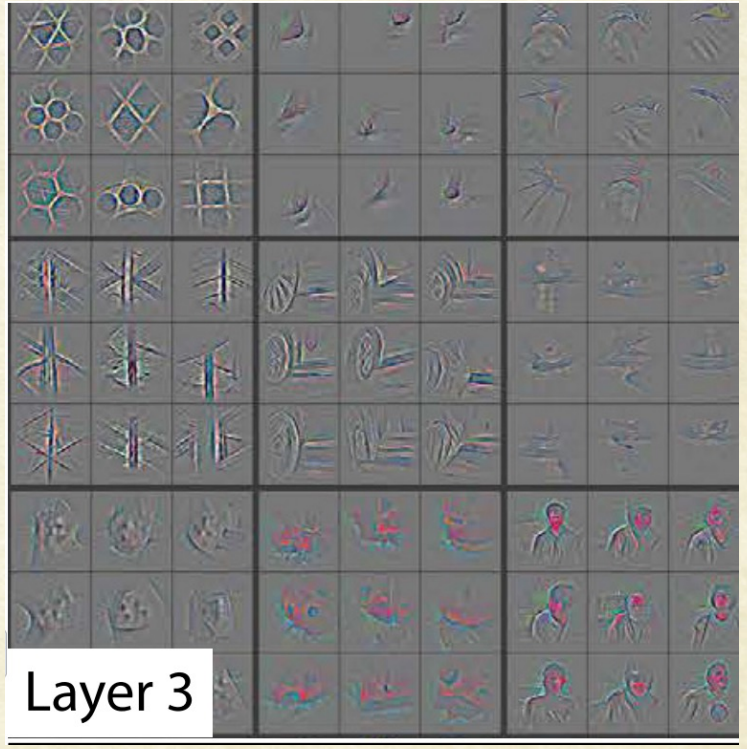
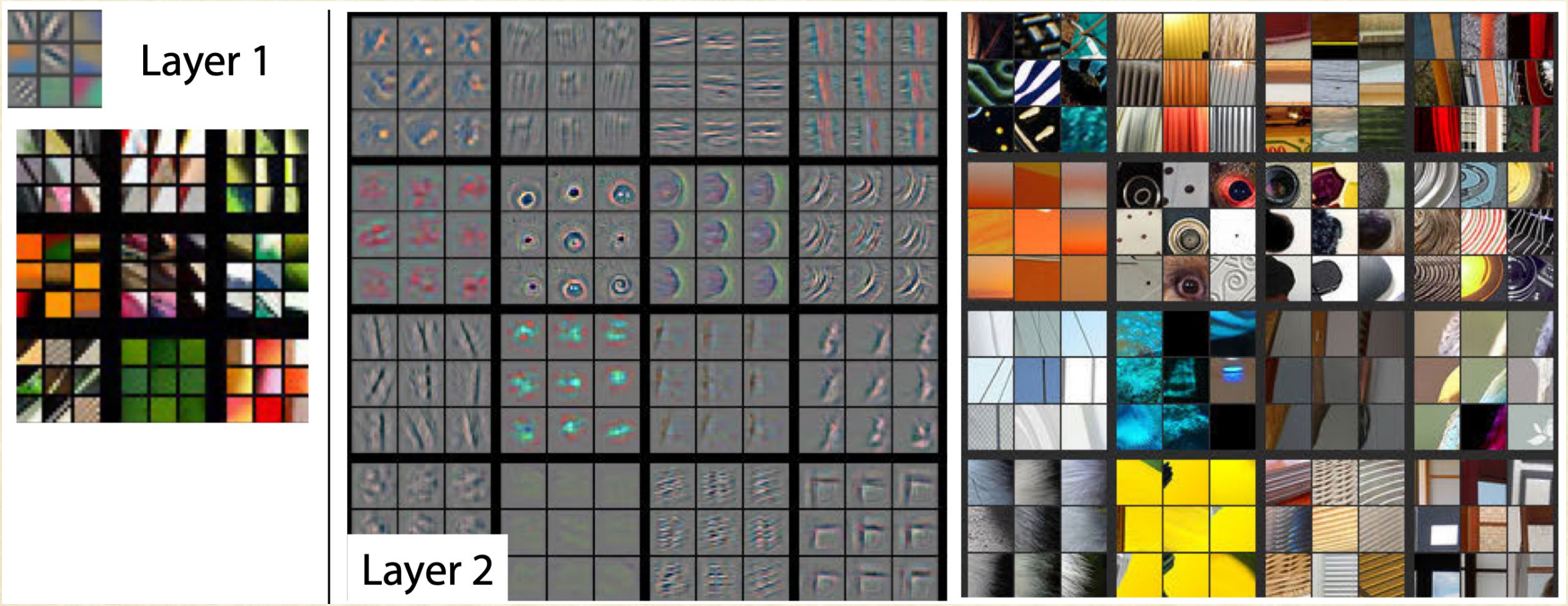


Image source:

Zeiler, Matthew D., and Rob Fergus. 2014. "Visualizing and Understanding Convolutional Networks. *ECCV 2014*

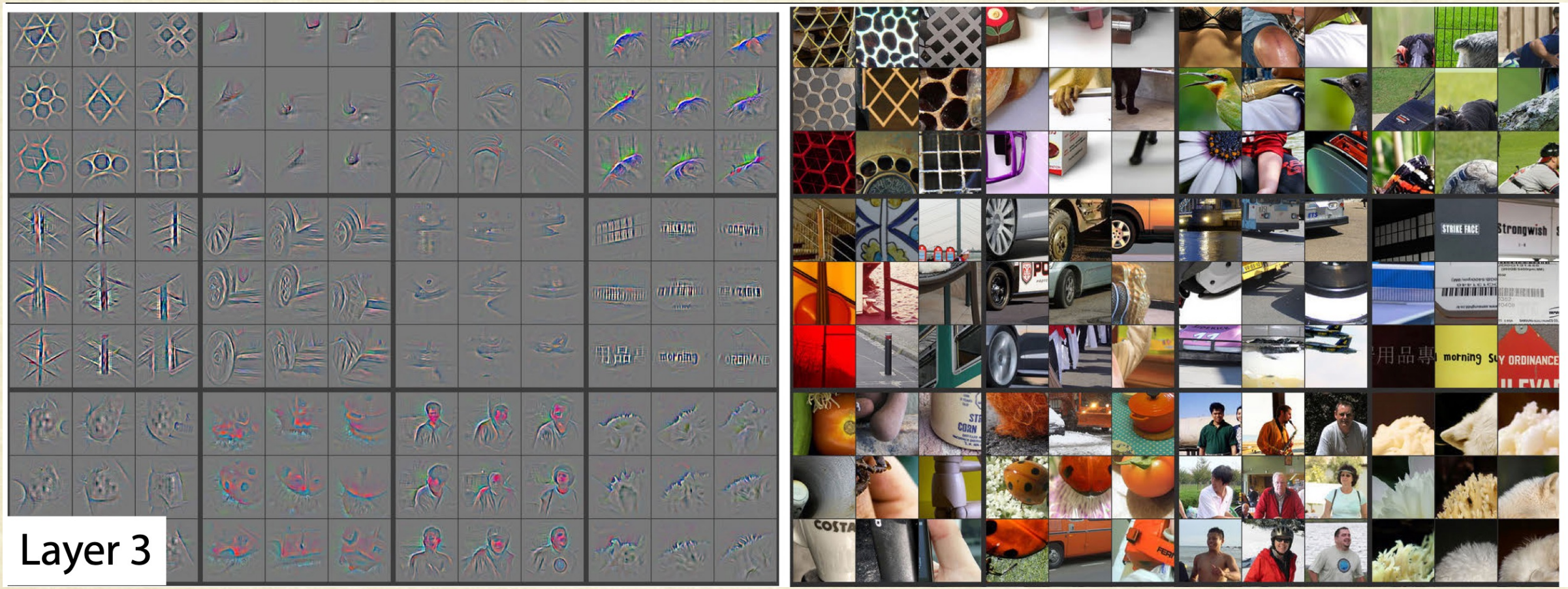
These are top activations projected down to pixel space using deconvolution architecture

Visualising CNN: Layers 1 & 2



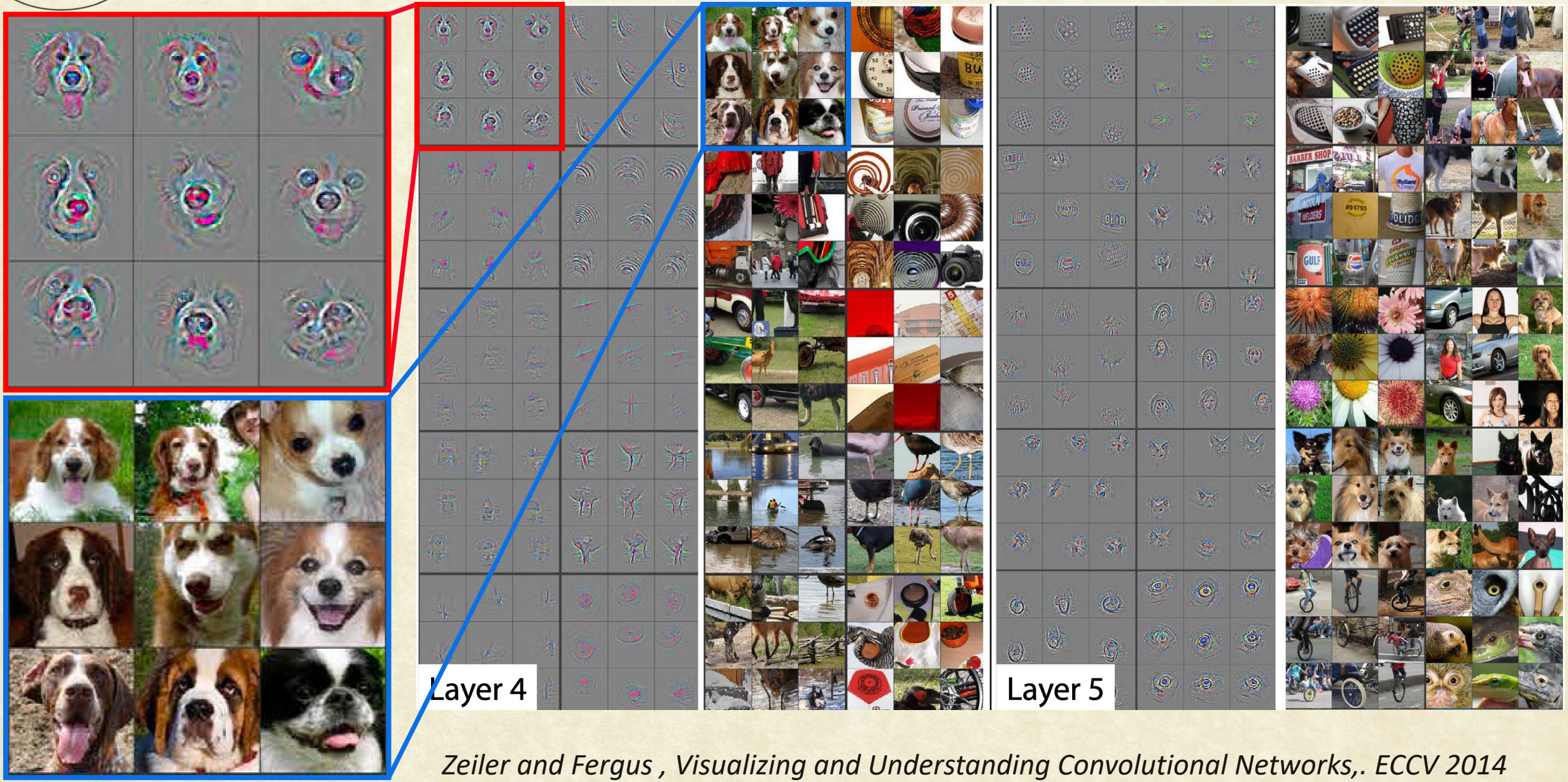
Source: Zeiler and Fergus , Visualizing and Understanding Convolutional Networks,. ECCV 2014

Visualising CNN: Layer 3



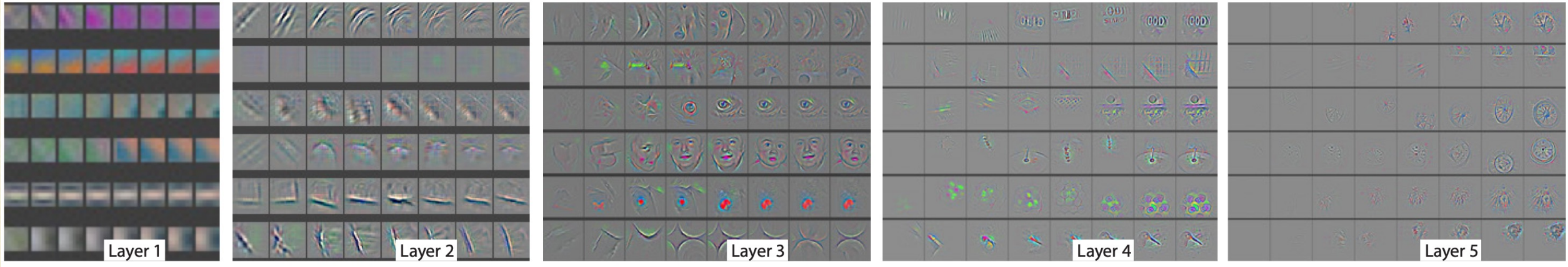
Source: Zeiler and Fergus , Visualizing and Understanding Convolutional Networks,. ECCV 2014

Visualising CNN: Layers 4 & 5



Zeiler and Fergus , Visualizing and Understanding Convolutional Networks,. ECCV 2014

Early Layers Converge Faster



Evolution of a randomly chosen subset of model features through training. Each layer's features are displayed in a different block. Within each block, a randomly chosen subset of features is shown at epochs [1,2,5,10,20,30,40,64].

Summary

- Convolutional Networks can learn useful features from data
  - Can be Learned using Back-Propagation (GD)
  - Avoid hand-crafting of features
  - Automatically identify best features for a task
- Generic guidelines for architecture
  - Conv-Pool-Norm : FC : SoftMax
  - Variants: LeNet, AlexNet, VGG, GoogLeNet, ResNet, etc.
- Learnt features seems to capture Hierarchy of objects
- Several techniques to overcome challenges
  - Overfitting, Vanishing Gradients

Questions?