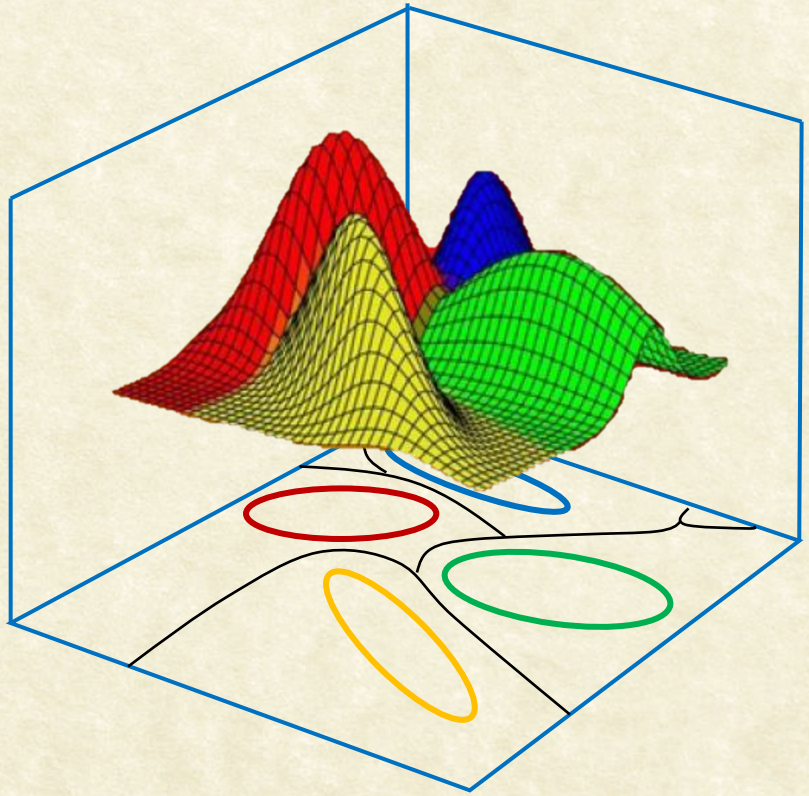
CS7.403: Statistical Methods in AI

Monsoon 2022:

Convolutional Neural Network



Anoop M. Namboodiri

Biometrics and Secure ID Lab, CVIT,

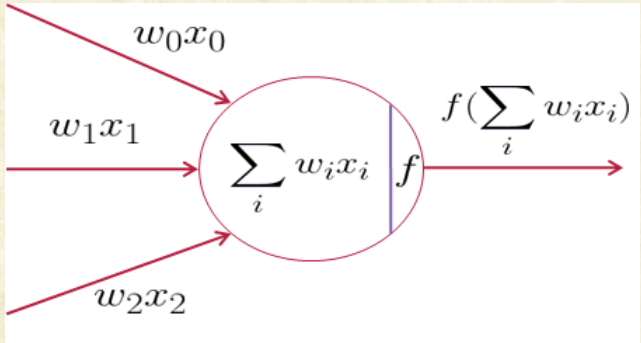
IIIT Hyderabad

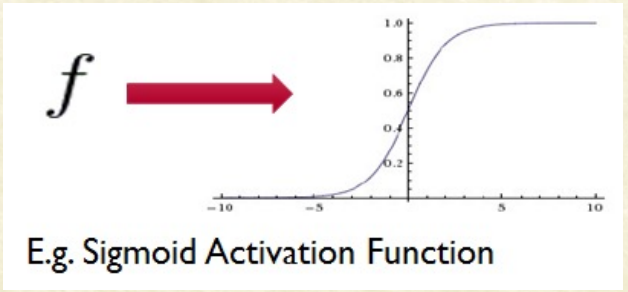
Multilayer Perceptron

A Recap

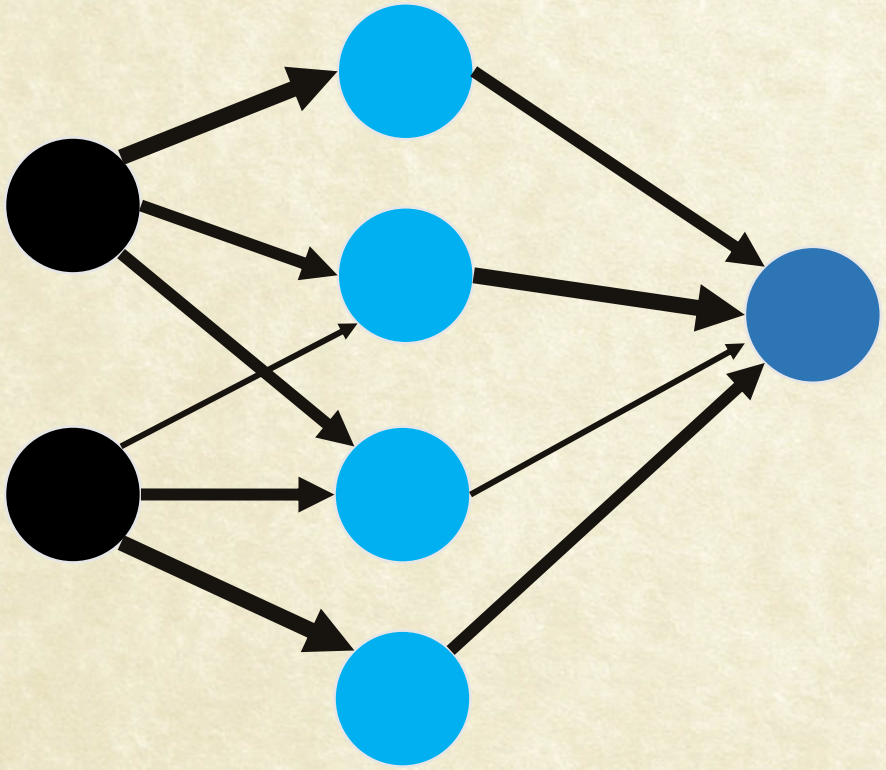
Neural Network

The Multilayer Perceptron



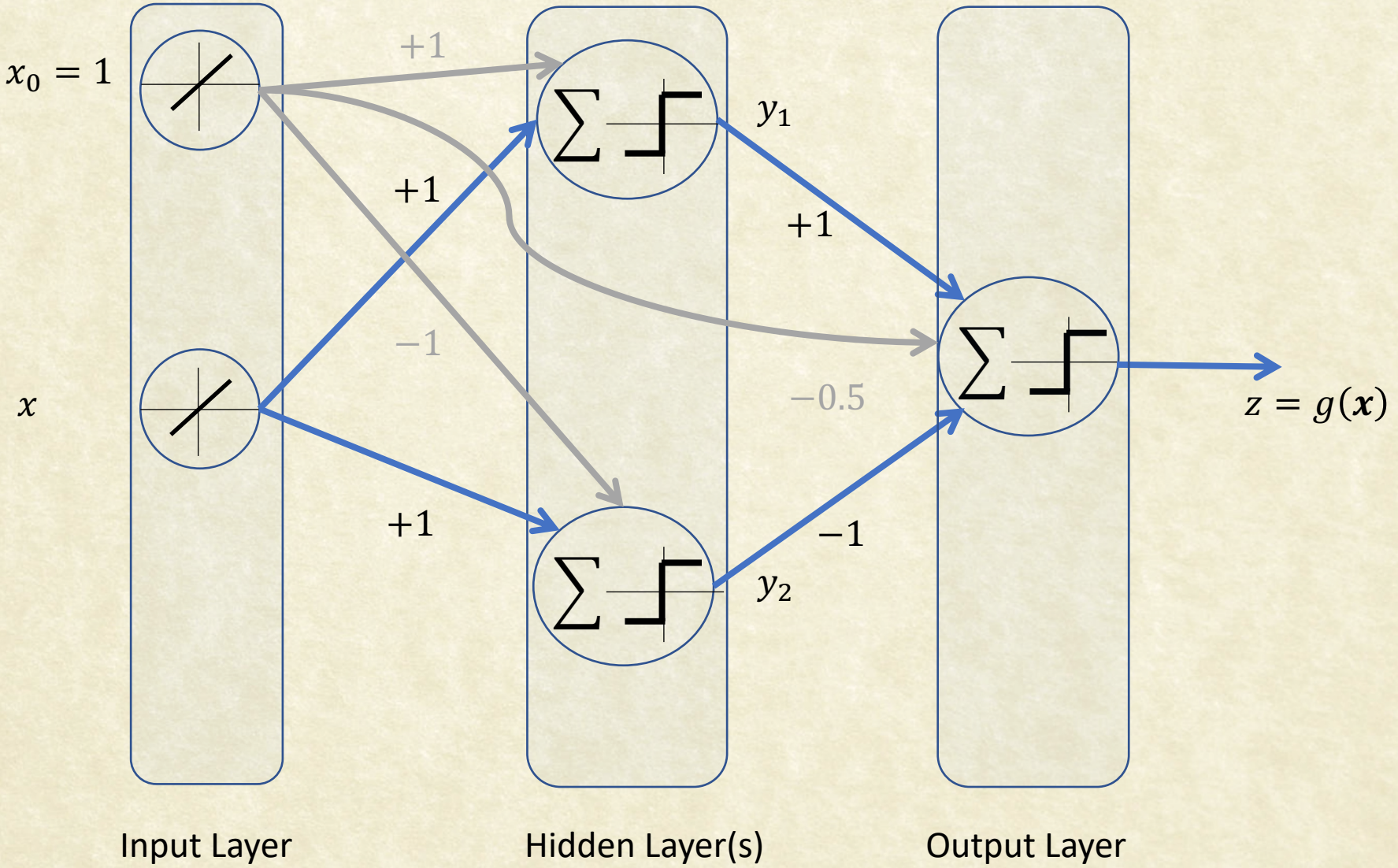


Input Layer *Hidden layer* *Output Layer*

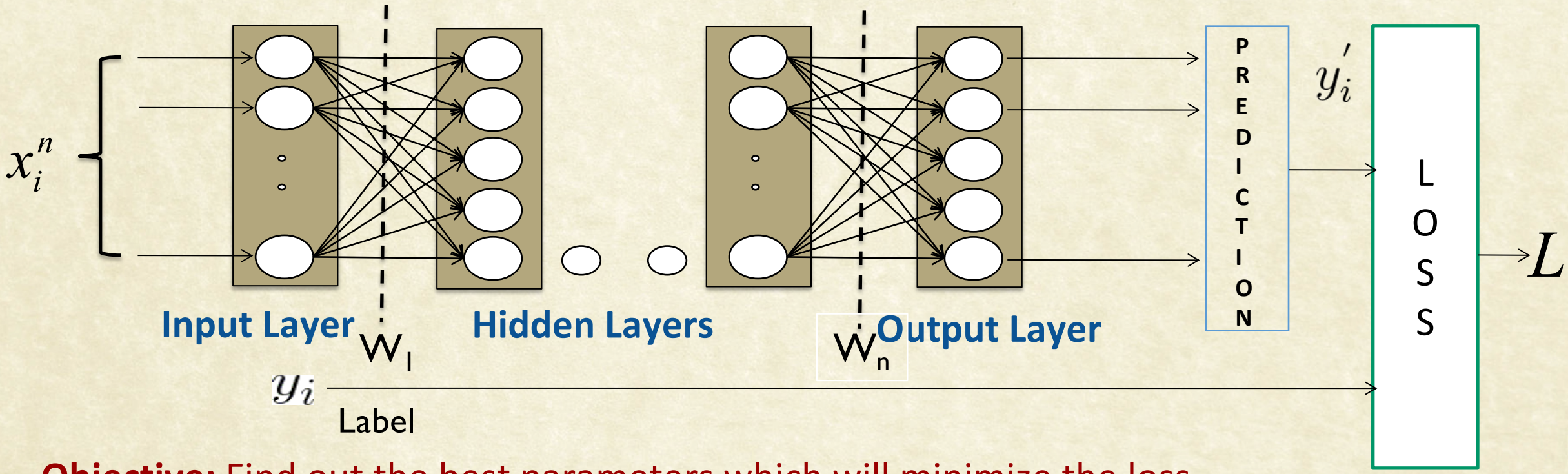


- Biologically inspired networks.
- Complex function approximation through composition of functions.
- Can learn arbitrary Nonlinear decision boundary

Neuron, Perceptron and MLP



Loss or Objective Function

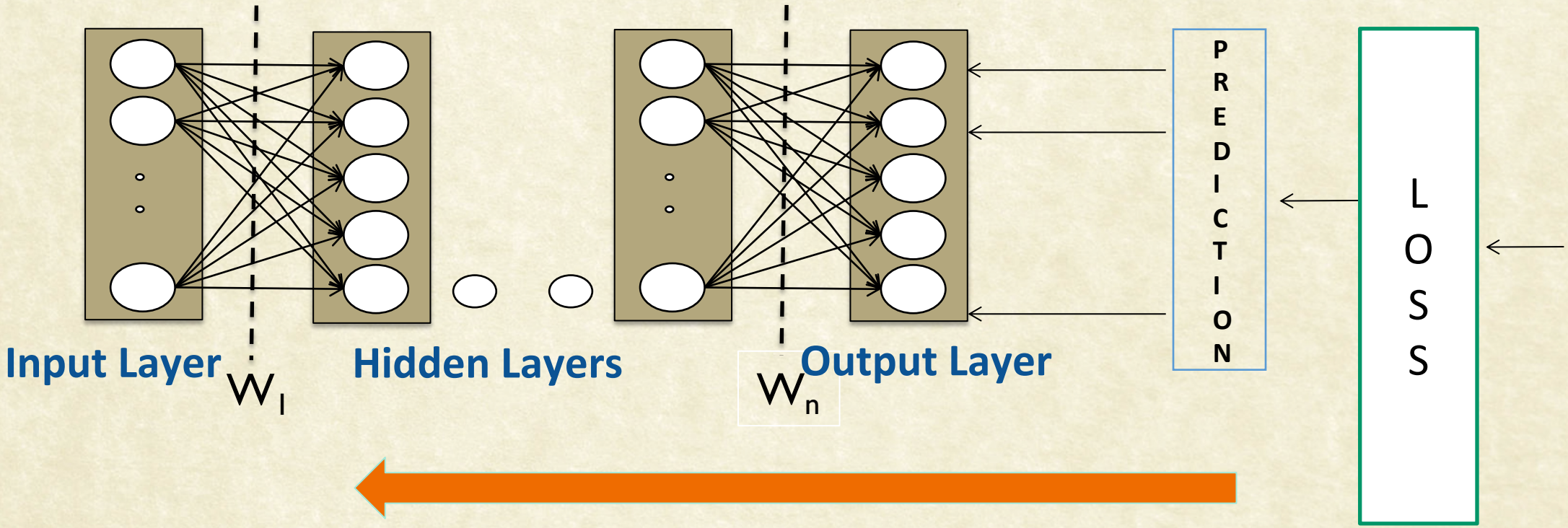


Objective: Find out the best parameters which will minimize the loss.

$$W^* = arg\min_W \sum_{i=1}^{N} L(y_i', y_i; W)$$ → **Weight Vector**

$$L = \frac{1}{2}||y_i' - y_i||_2^2$$ → **E.g. Squared Loss**

Back Propagation



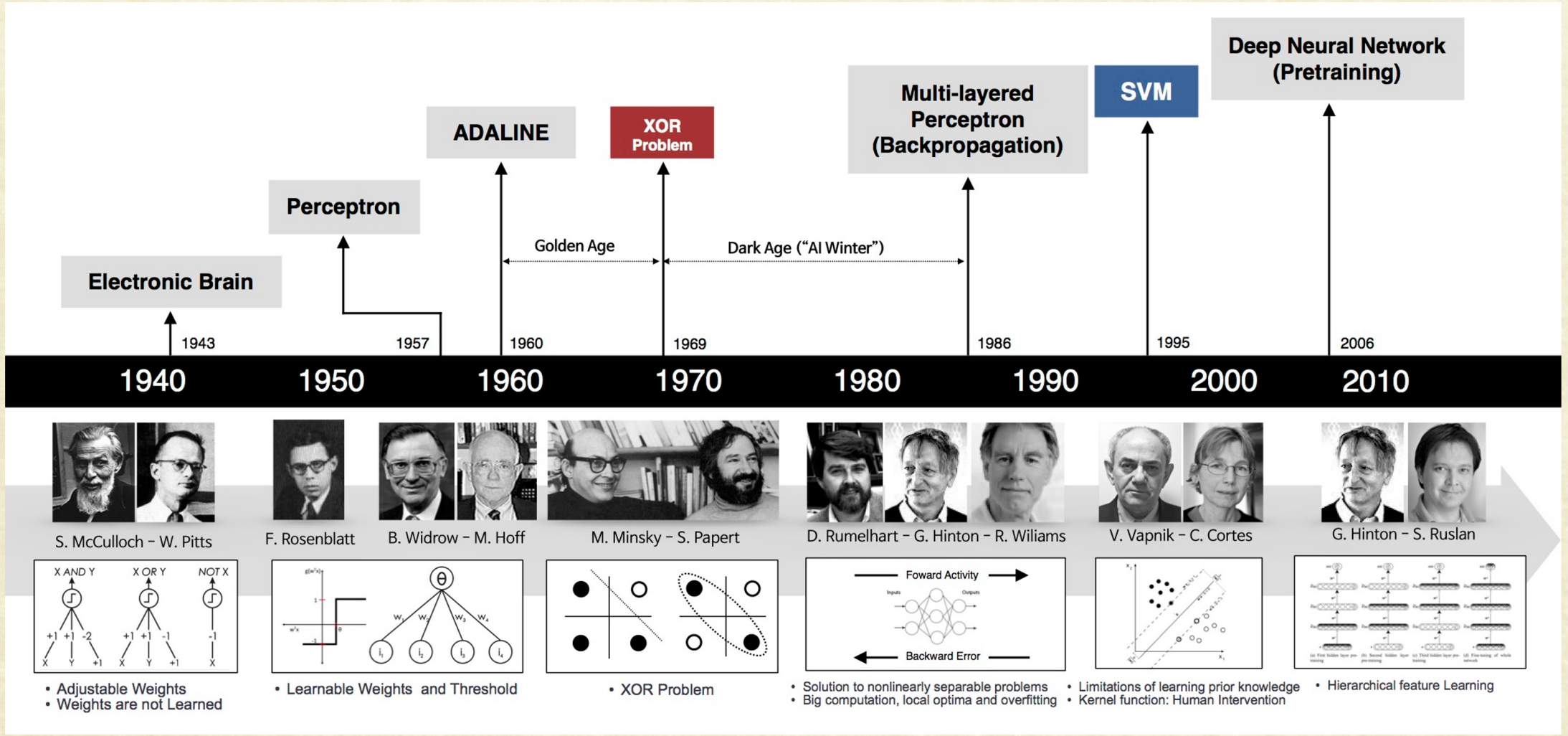
Solution: Iteratively update W along the direction where loss decreases

Each layer's weights are updated based on the derivative of its output w.r.t. inputs and weights

Deep Learning

An Introduction

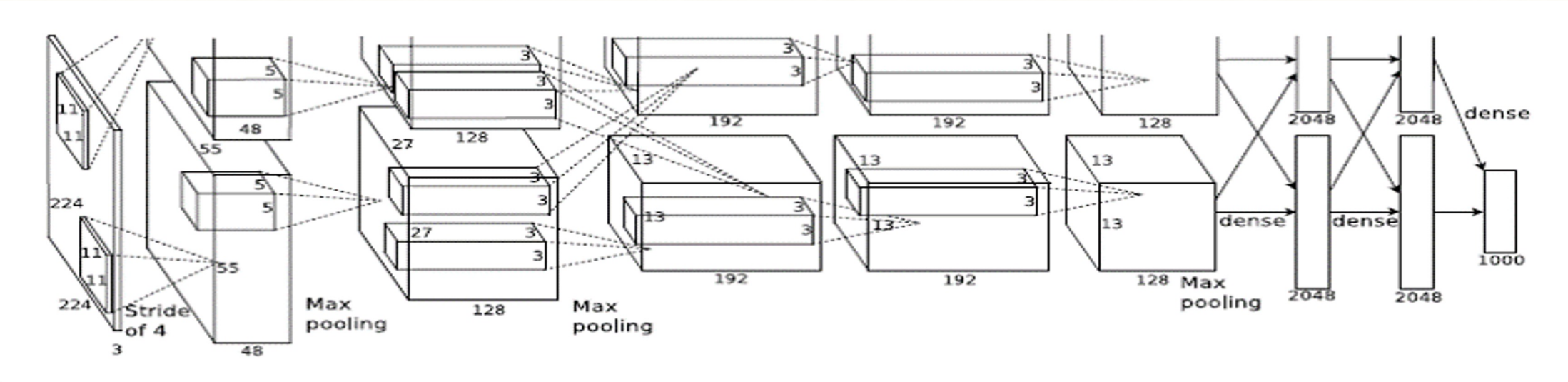
History of Deep Learning



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

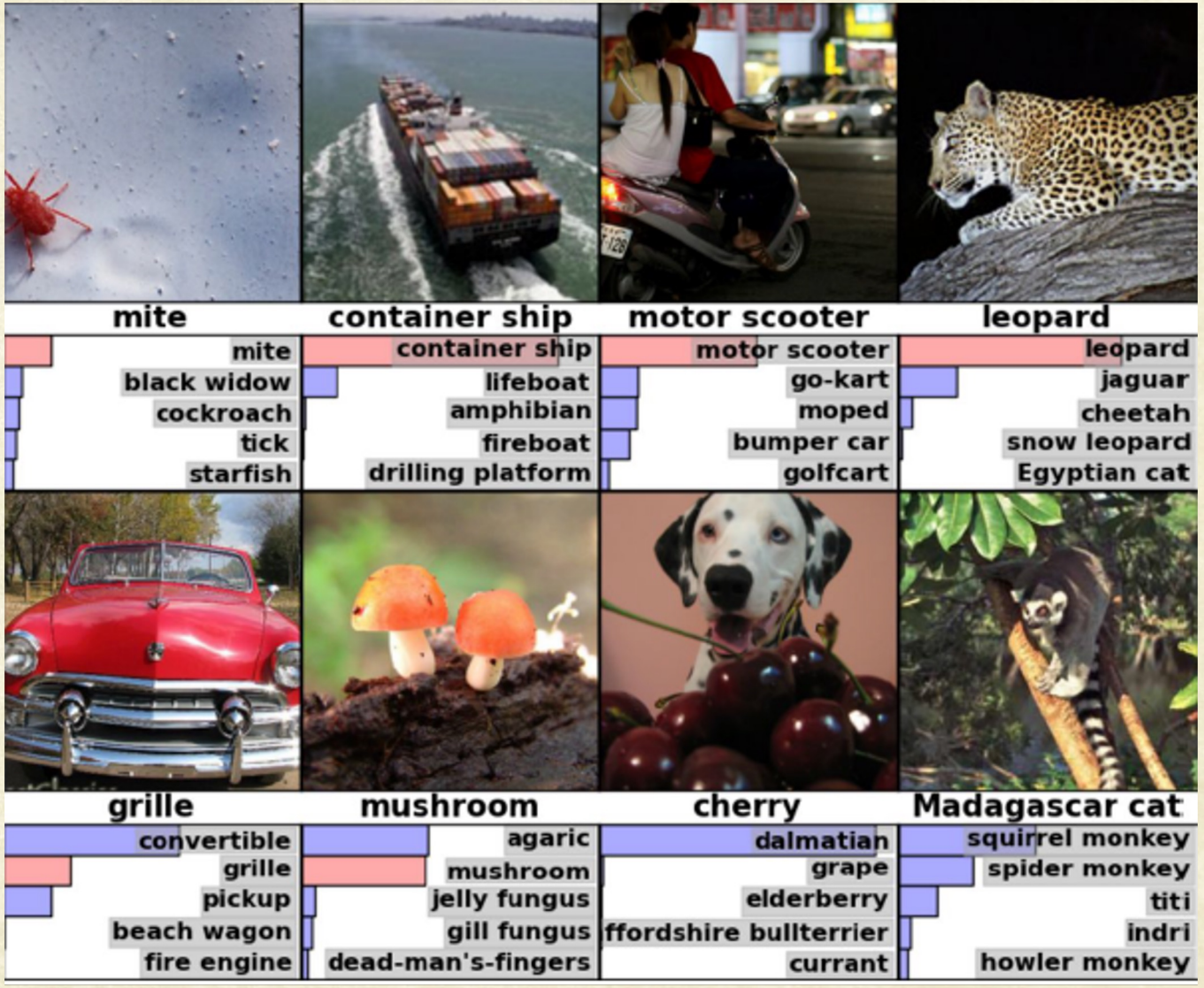
Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

ImageNet Classification Task:

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

ImageNet ILSVRC

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



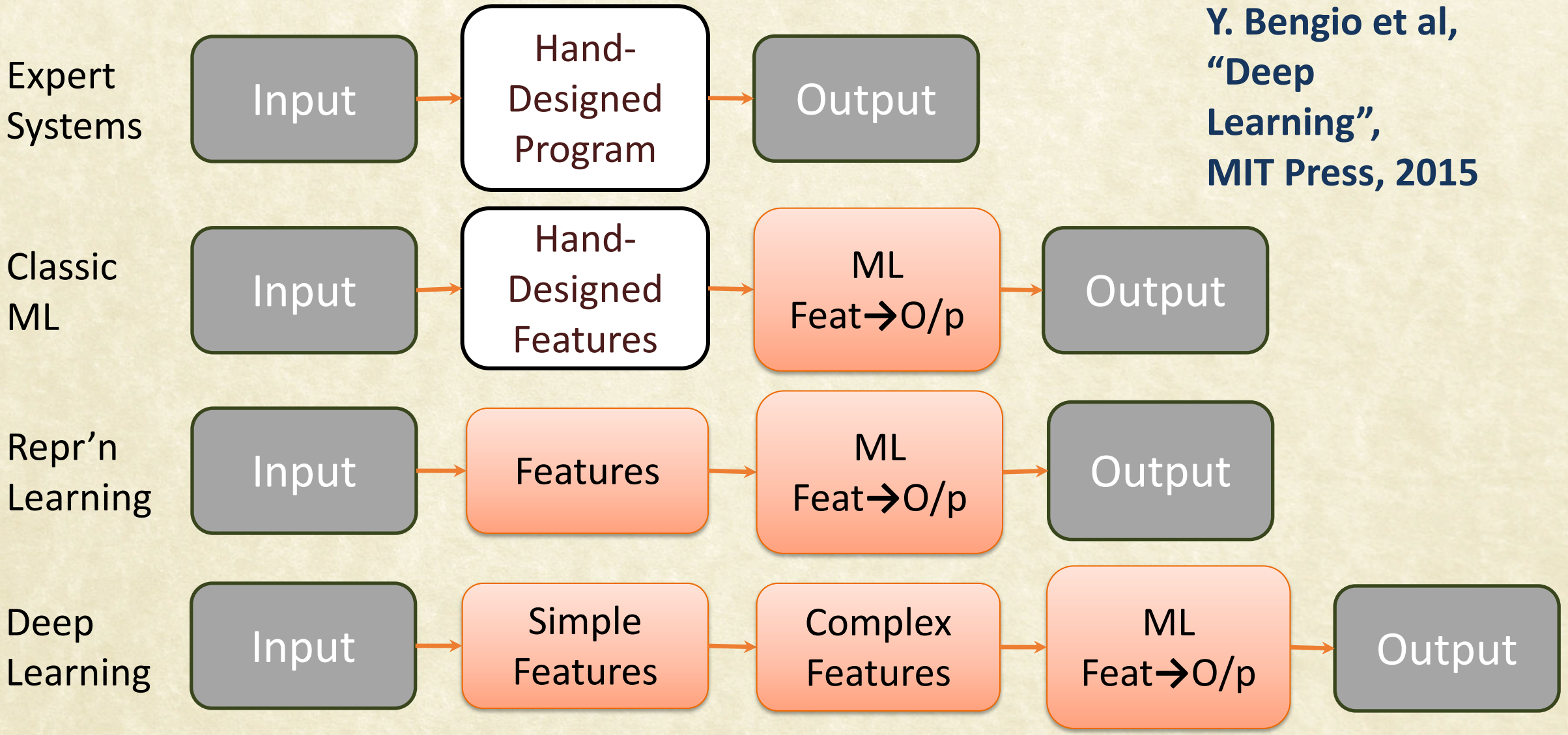
Success of “Deep Learning”: ImageNet Challenge

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 %

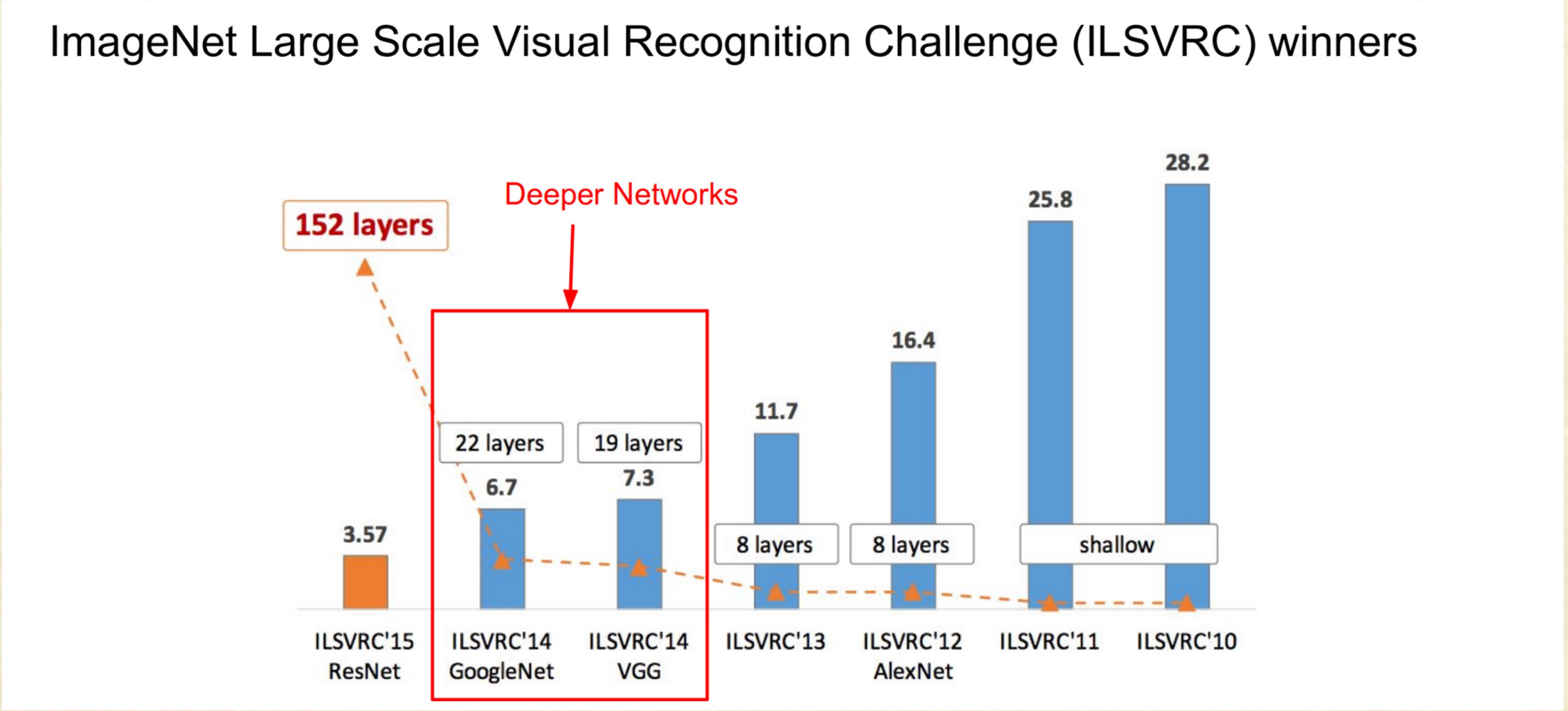
Mostly Deeper Networks
Smaller Convolutions
Many Specific Enhancements

Evolution of Learning



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

Getting Deeper



The Convolutional Layer

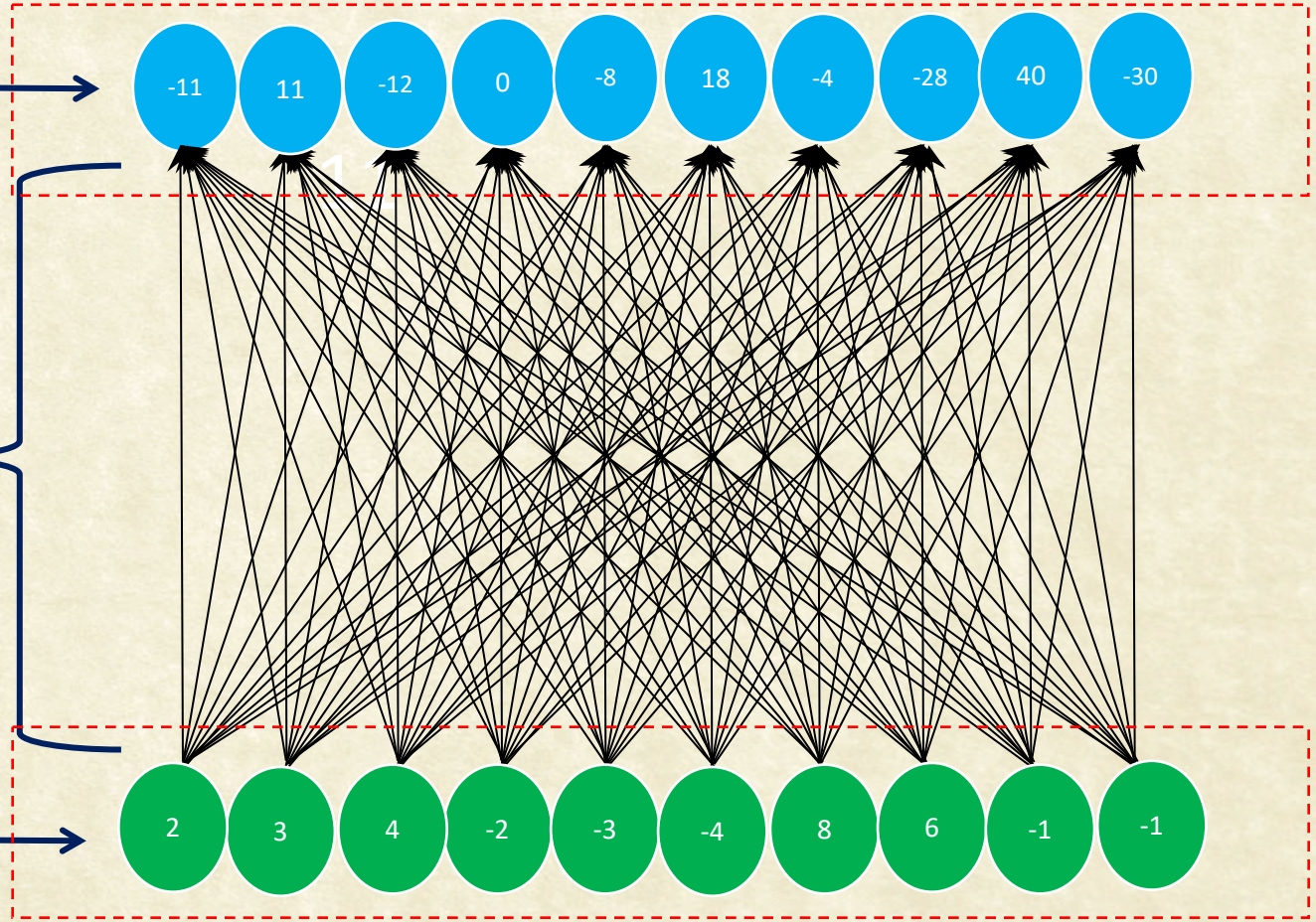
Utilizing Locality

Dense connections

$$Y = W^T X$$

$$W = \begin{bmatrix} -1 & 1 & -2 & 2 & -1 & 1 & 2 & -2 & 3 & -3 \\ -2 & 2 & -1 & 1 & -3 & 3 & -2 & 2 & -1 & 1 \\ -3 & 3 & -1 & 1 & -1 & 1 & -2 & -2 & 2 & 2 \\ -1 & 1 & -2 & 2 & -1 & 1 & 2 & -2 & 3 & -3 \\ -2 & 2 & -1 & 1 & -3 & 3 & -2 & 2 & -1 & 1 \\ -3 & 3 & -1 & 1 & -1 & 1 & -2 & -2 & 2 & 2 \\ -1 & 1 & -1 & 1 & -2 & 2 & -2 & -2 & 3 & -3 \\ -1 & 1 & -1 & -1 & 1 & 1 & +2 & -2 & 3 & -1 \\ -2 & 2 & -1 & 1 & -3 & 3 & -2 & 2 & -1 & 1 \\ -1 & 1 & -1 & -1 & 1 & 1 & +2 & -2 & 3 & -1 \end{bmatrix}$$

$$X$$

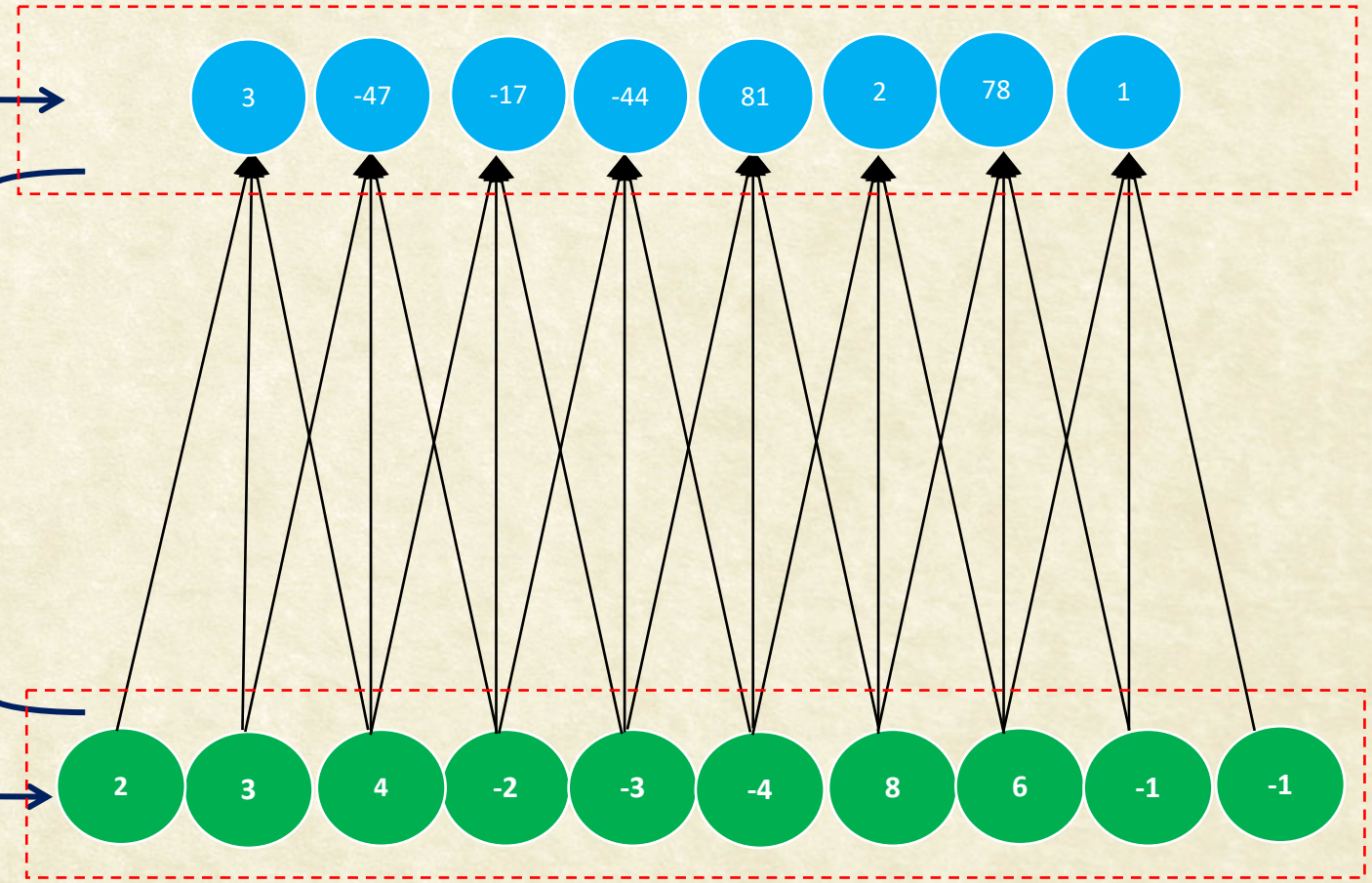


What if connections are only local?

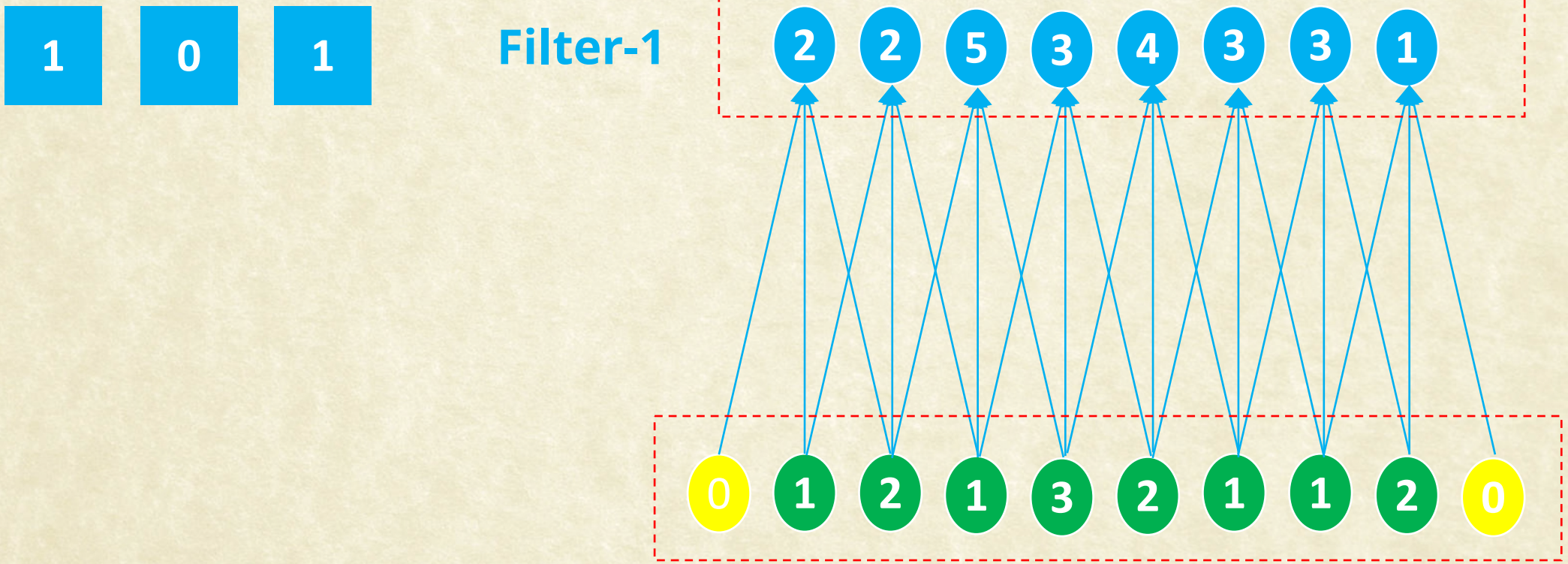
$$Y = W^T X$$

$$W = \begin{bmatrix} 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -4 & -9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 4 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & -9 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & -8 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & -4 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & -6 & -7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6 \end{bmatrix}$$

$$X$$



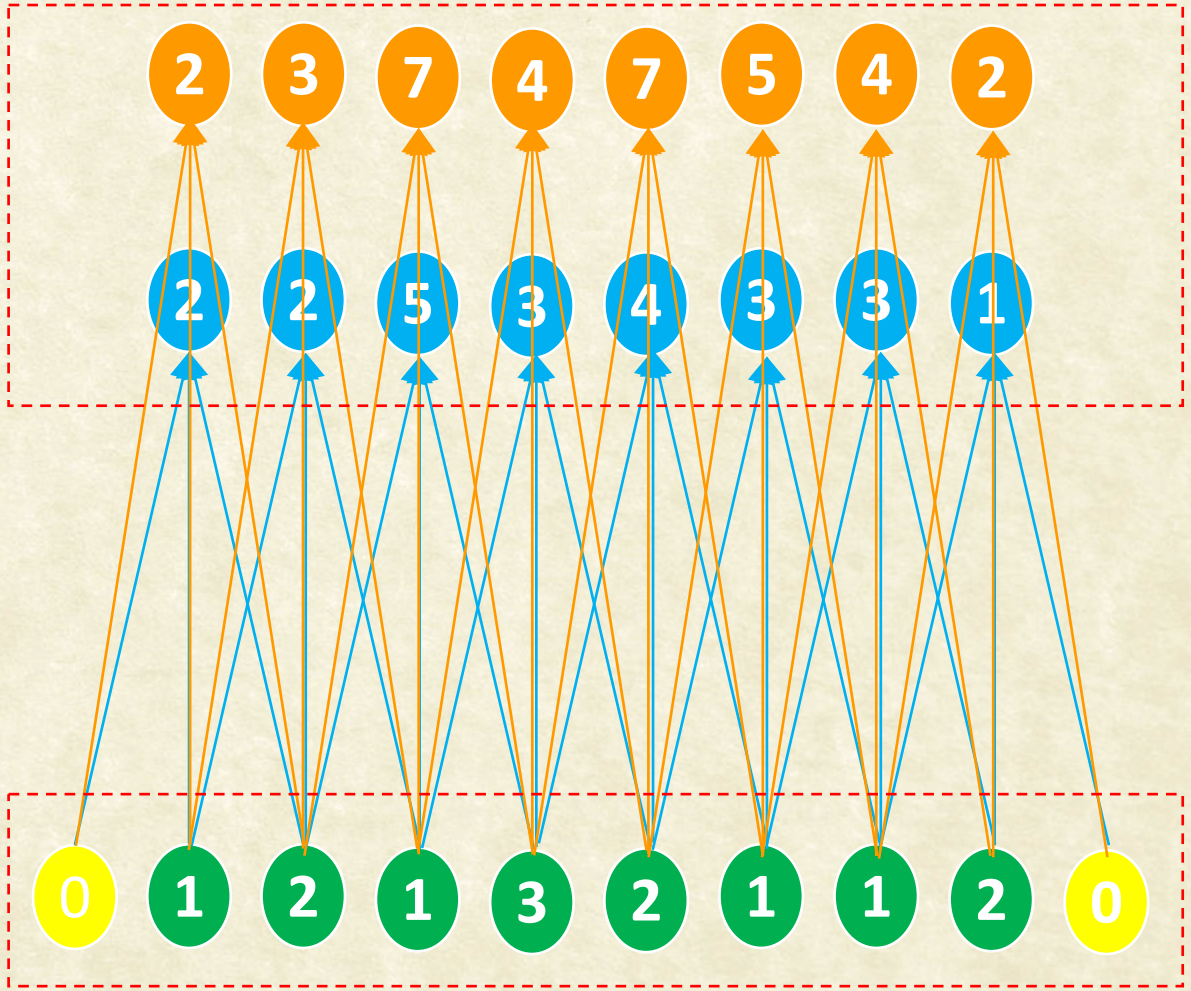
What if weights are same/shared?



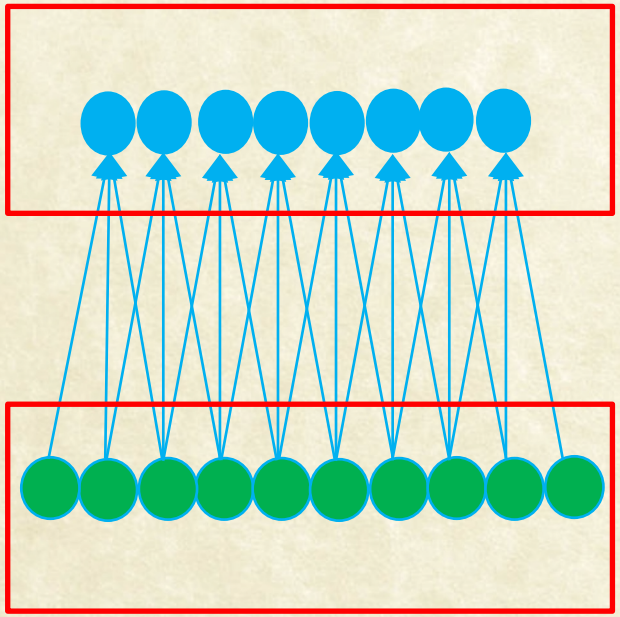
Two such filters/weights

2	0	1

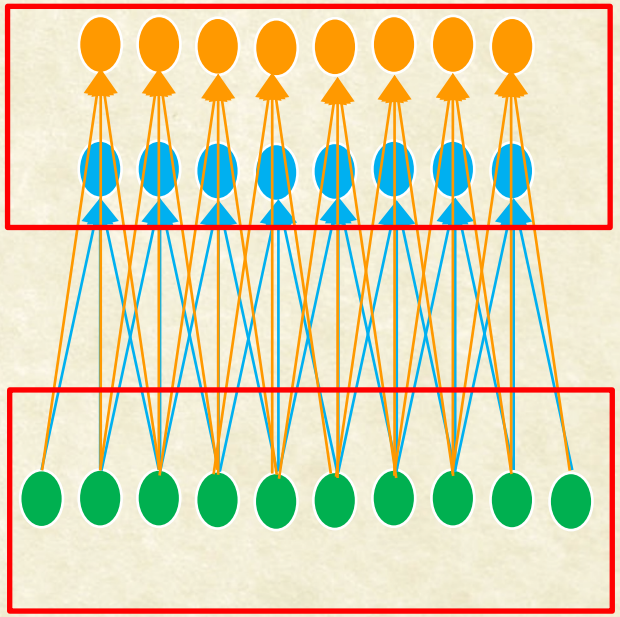
1	0	1



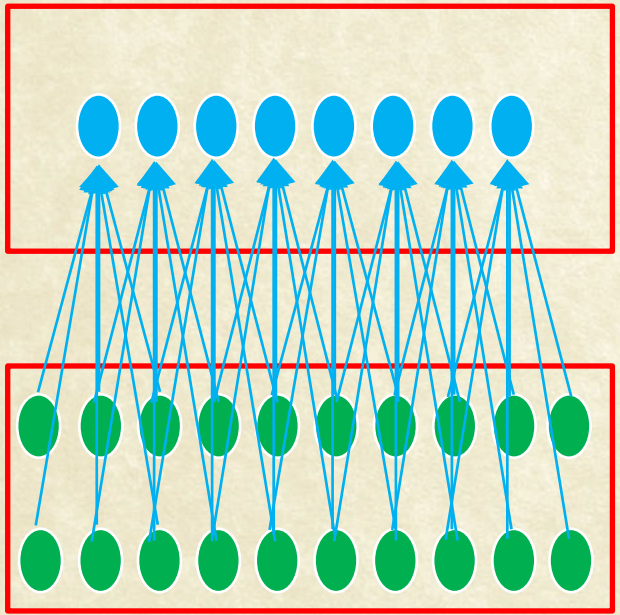
Convolution layer: Different Possibilities



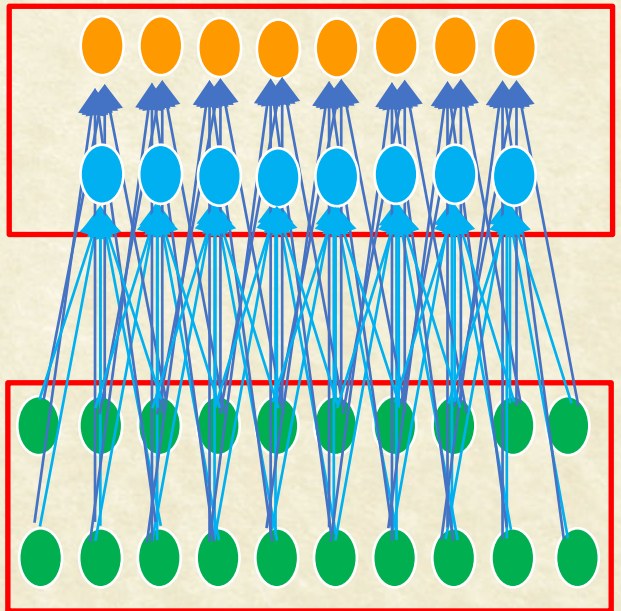
Channels:
- I/P =1
- O/P=1
- #Parameters = 3



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

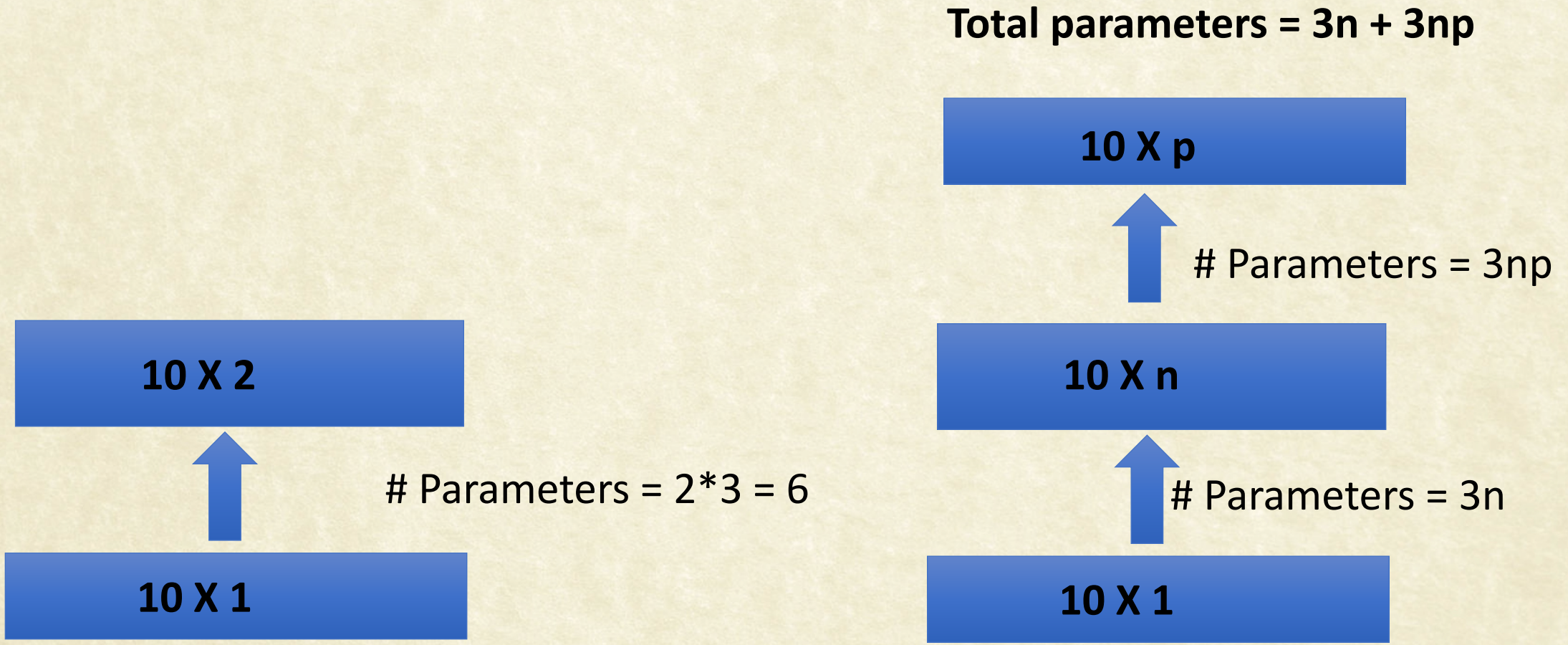


Channels:
- I/P =2
- O/P=1
- #Parameters = 6



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

Pictorially



Convolutional in 2D

Locality is not one-dimensional

Convolution

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

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

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

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

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

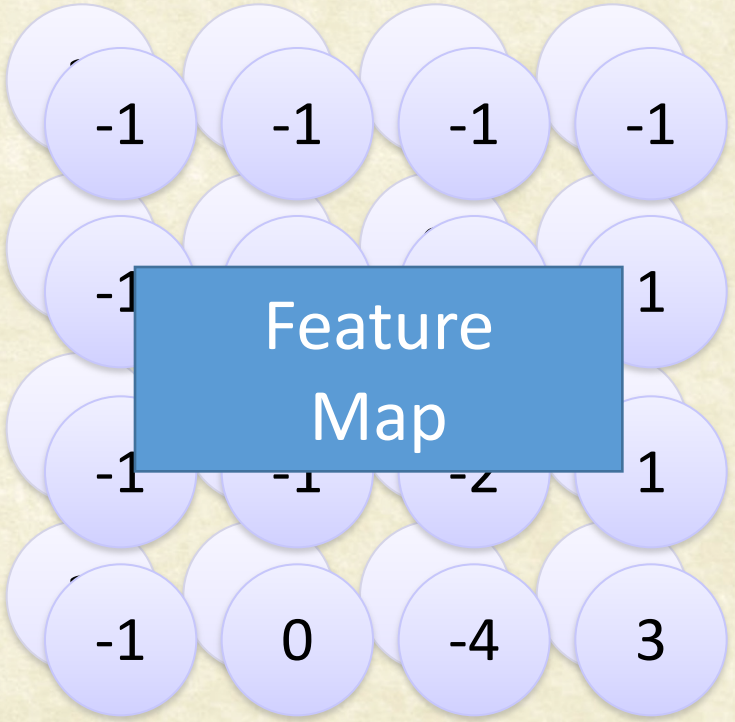
Filter 2

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

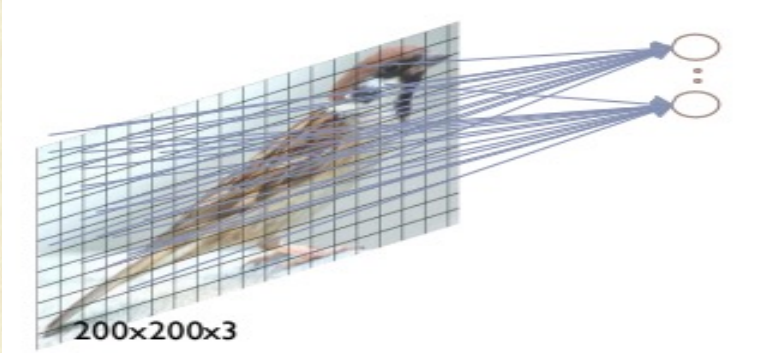
Repeat this for each filter



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

Convolution layer

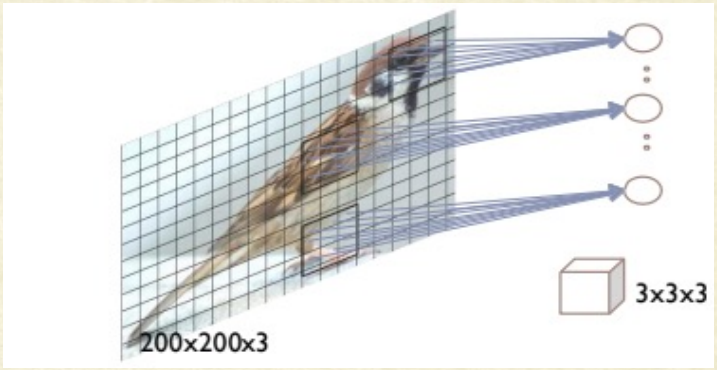
- Fully connected layer



- 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!

Parameter Calculations

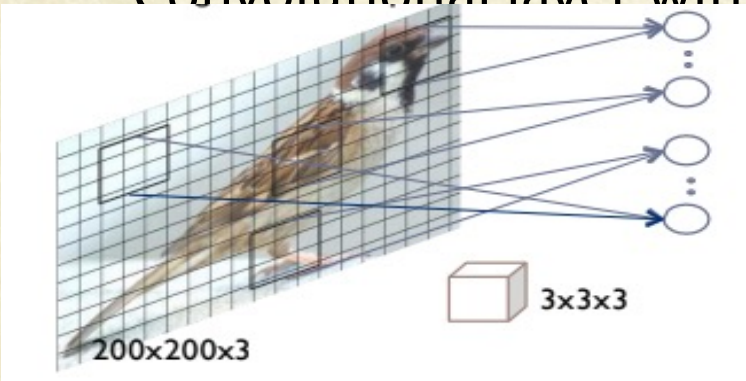
- Locally connected layer



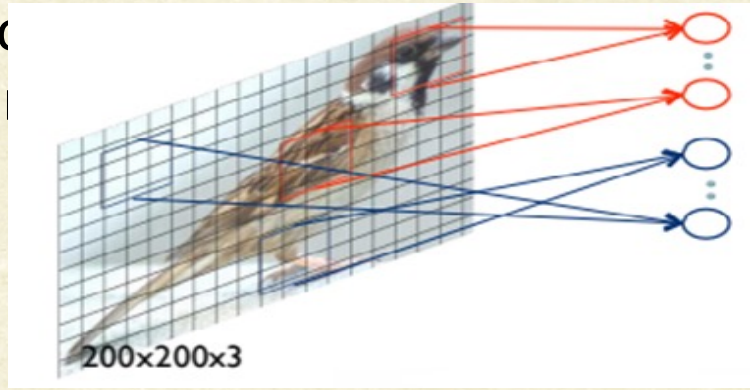
- #Hidden Units: 120,000
- #Params: 3.2 Million (= 120K X 27)
- Useful when the image is highly registered

Convolution layer

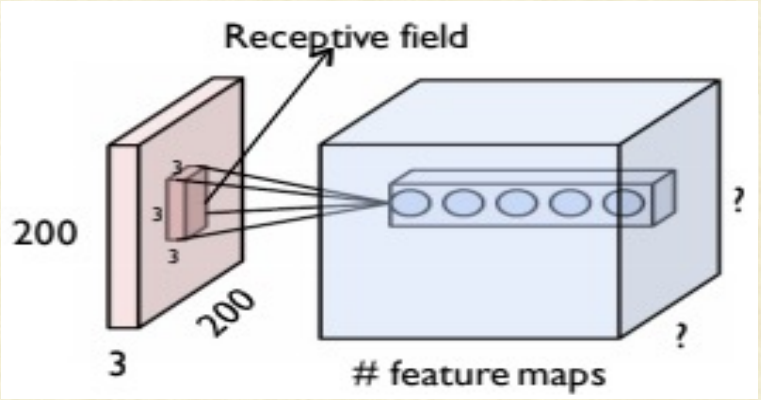
- Convolutional layer with



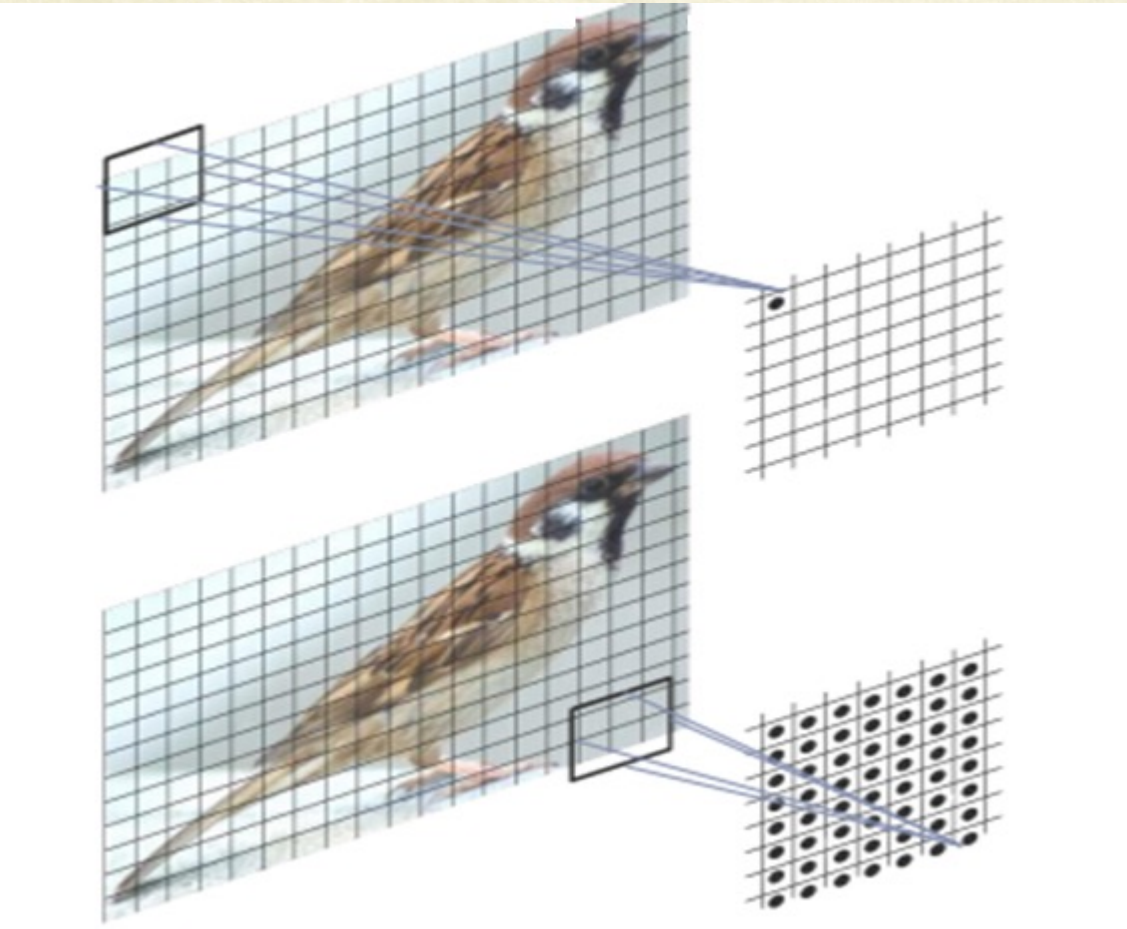
- Convo
featu



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



Convolution layer



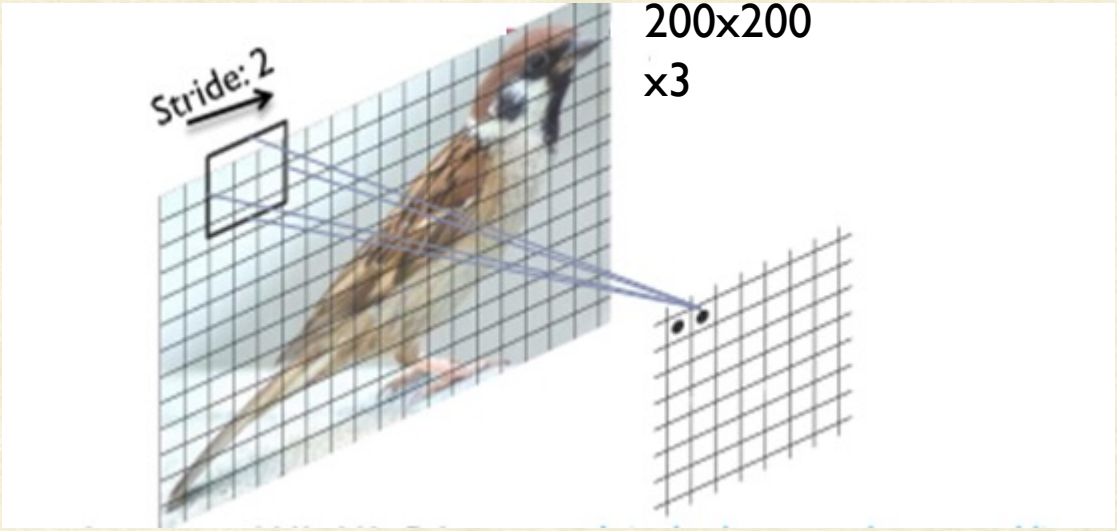
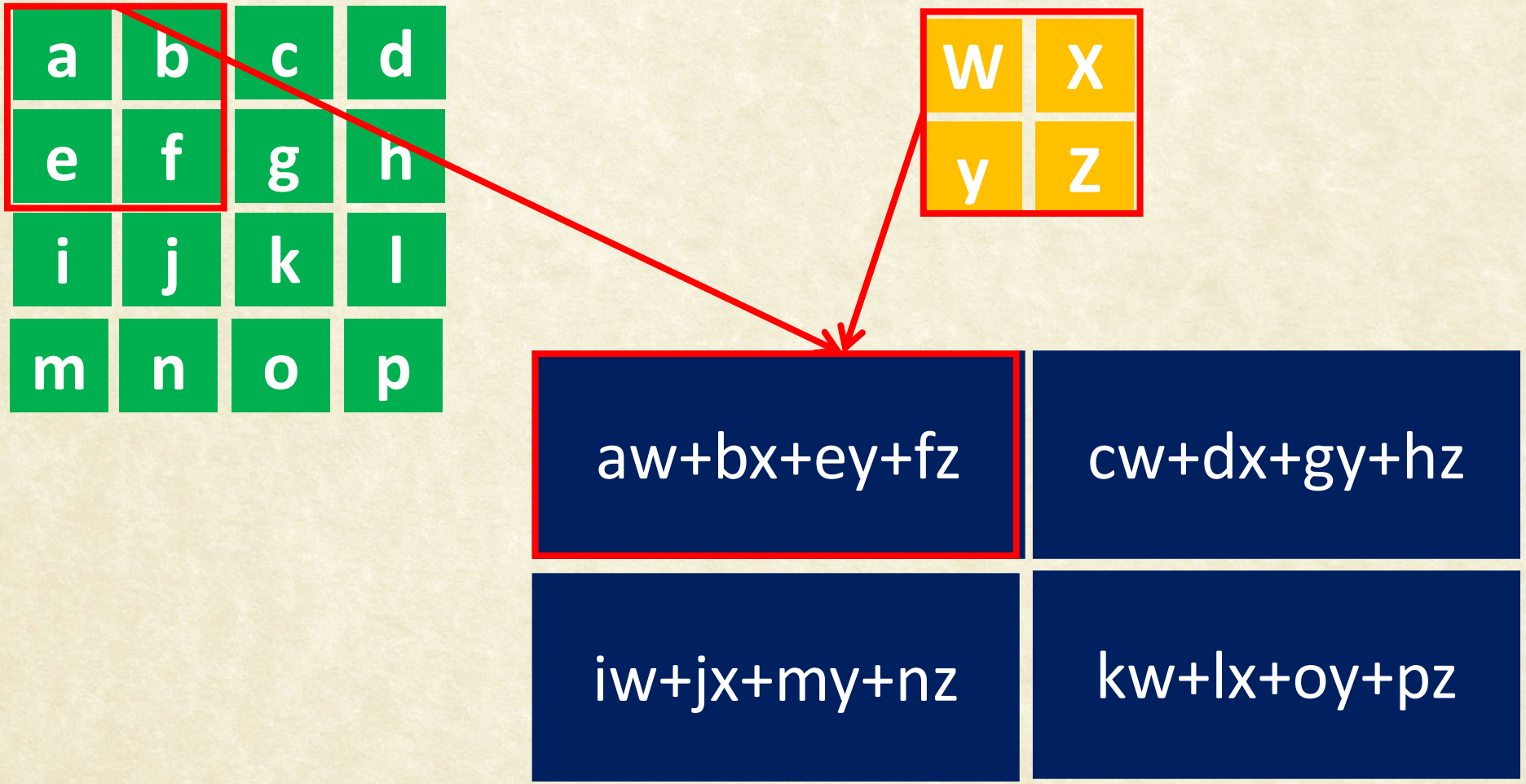


Image size: W1xH1xD1
Receptive field size: FxF
#Feature maps: K

W2=(W1-F)/S+1
H2=(H1-F)/S+1
D2=K

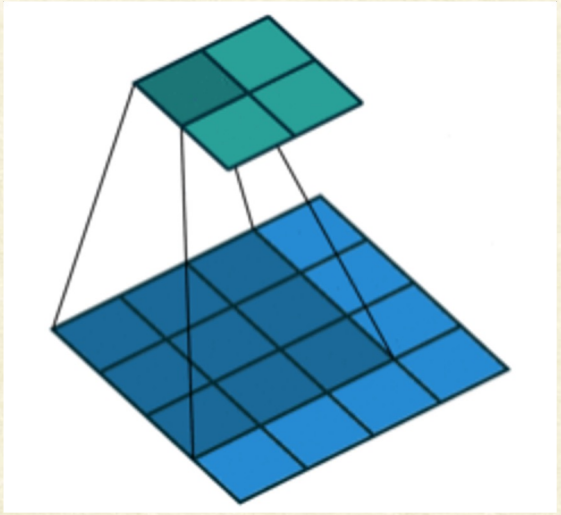
It is also better to do zero padding to preserve input size spatially.

Convolution in 2-D

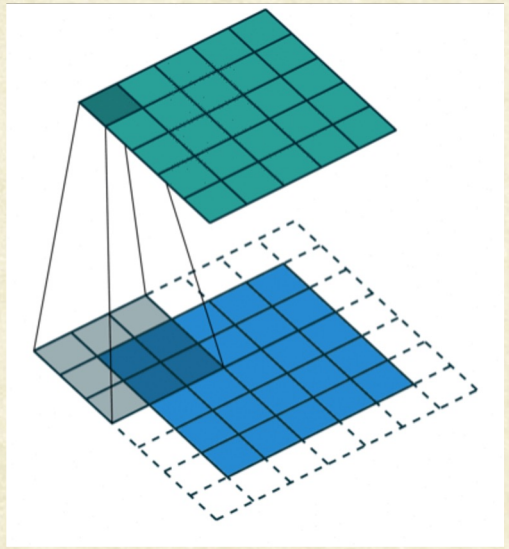


CNNs

- Window size
- Stride
- Padding



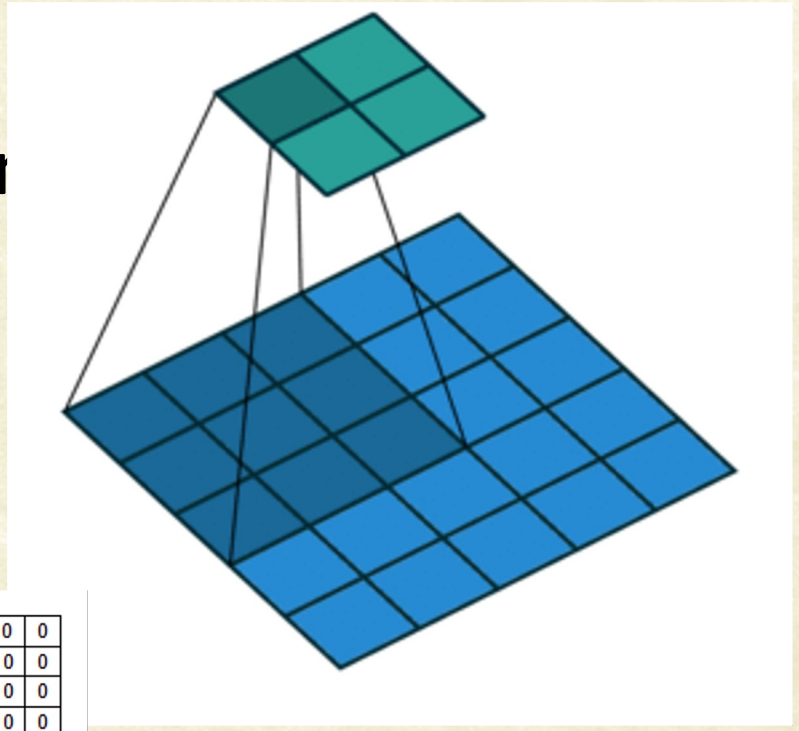
Window size: 3x3
Stride: 1
Padding: 0



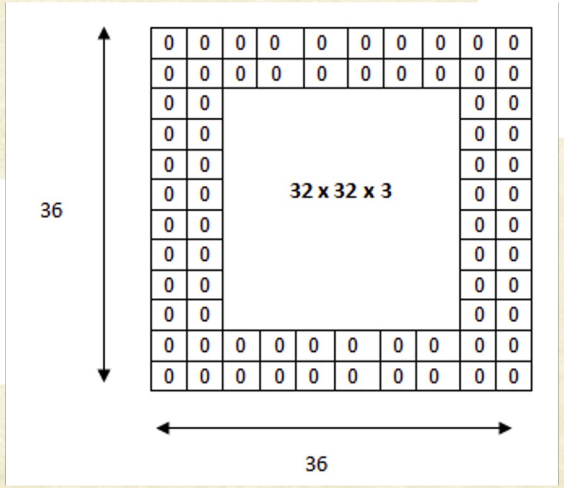
Window size: 3x3
Stride: 1
Padding: 1

CNNs

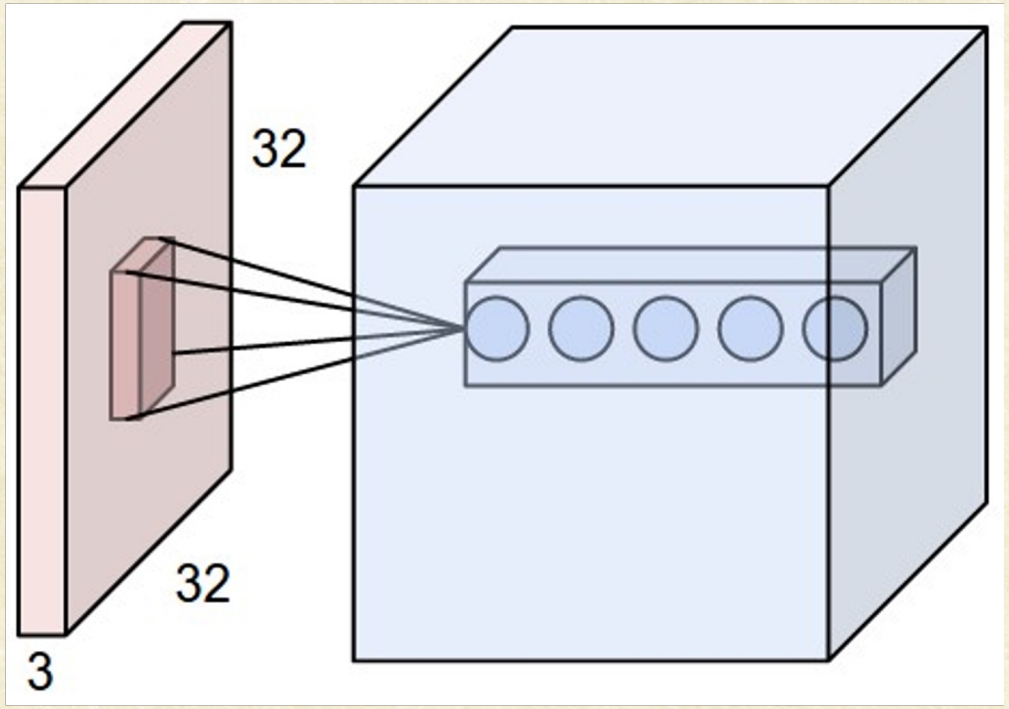
- Larger stride reduces dimensio



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

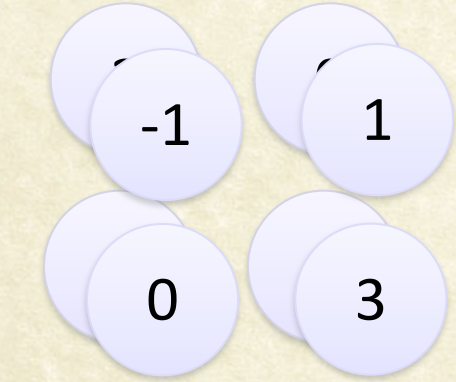


Input, Output Channels : Multiple Filters



DEMO: http://cs231n.github.io/convolutional-networks/

The whole CNN



A new image

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Smaller than the original image

The number of channels is the number of filters

The whole CNN

Convolution

Max Pooling

A new image

Convolution

Max Pooling

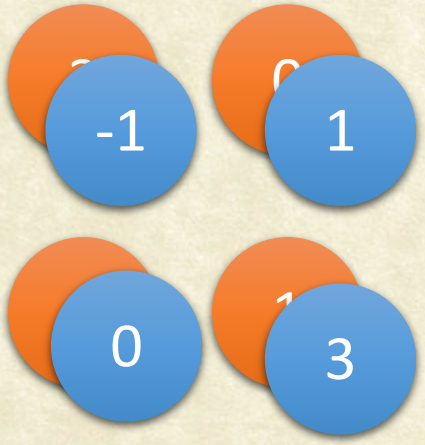
A new image

Flattened

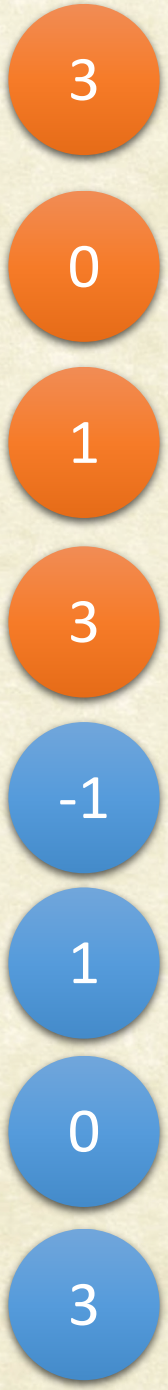
Fully Connected Feedforward network

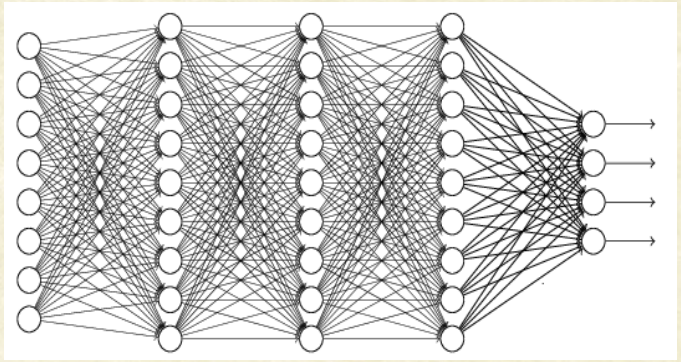
cat dog ……

Flattening



Flattened





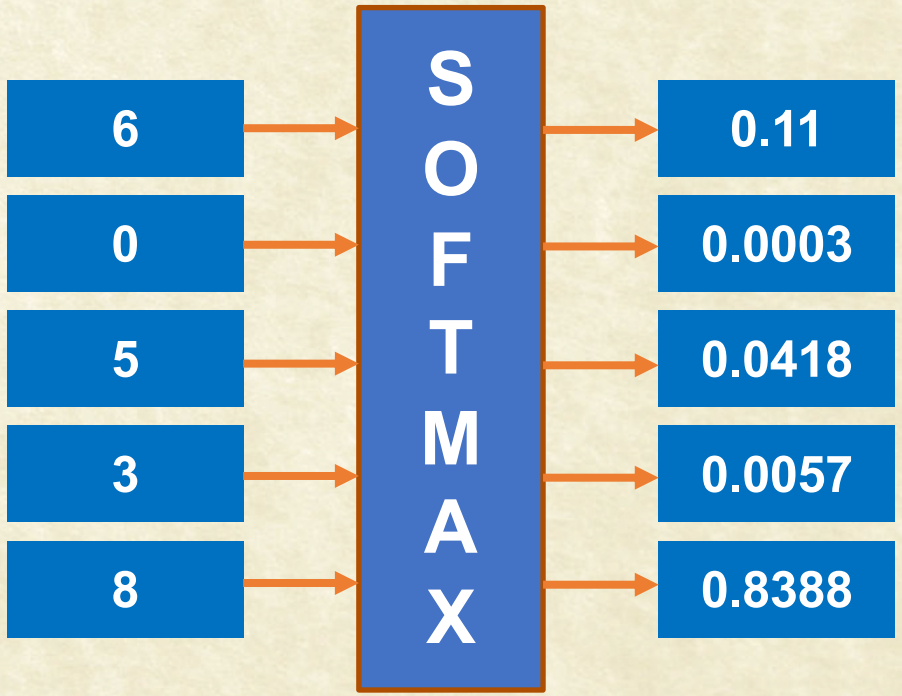
Fully Connected
Feedforward network

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

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



Terminologies

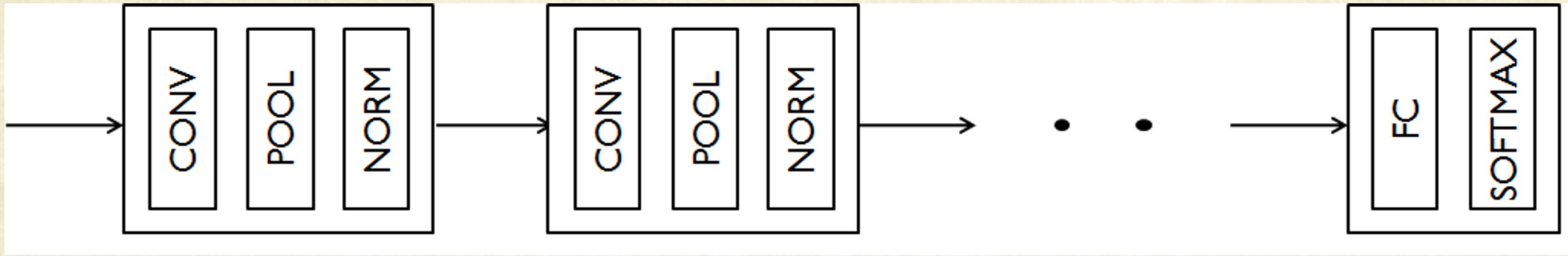
- # Input Channels
- # Output channels

- 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

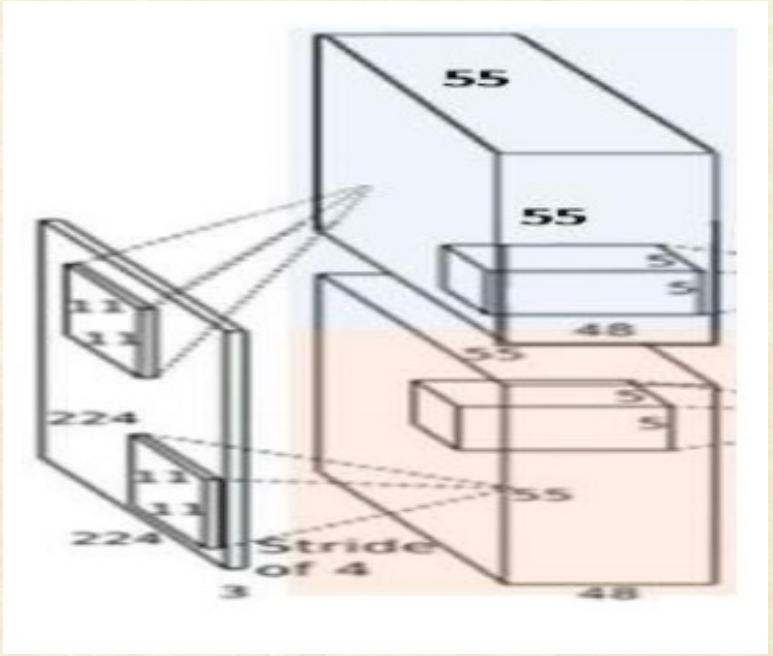


- Other layers
  - Pooling
  - Normalization
  - Fully connected
  - etc.

Parameter Calculation

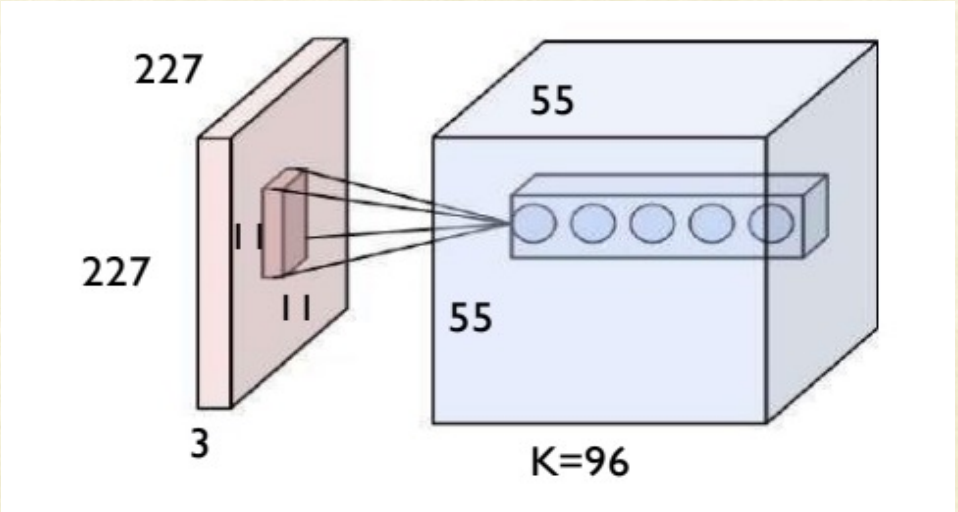
- Filter Size: F
- Input volume streams: D
- **#** filters: K
- **#** parameters in a layer is **(F .F .D) .K**

  Example:
- For layer 1, Input images are 227 x 227 x 3
- F = 11 and K = 96
- Each filter has 11 x 11 x 3 = 363 and 1 (bias) i.e., 364 weights
- # weights = 364 x 96 = **35 K** (approx.)

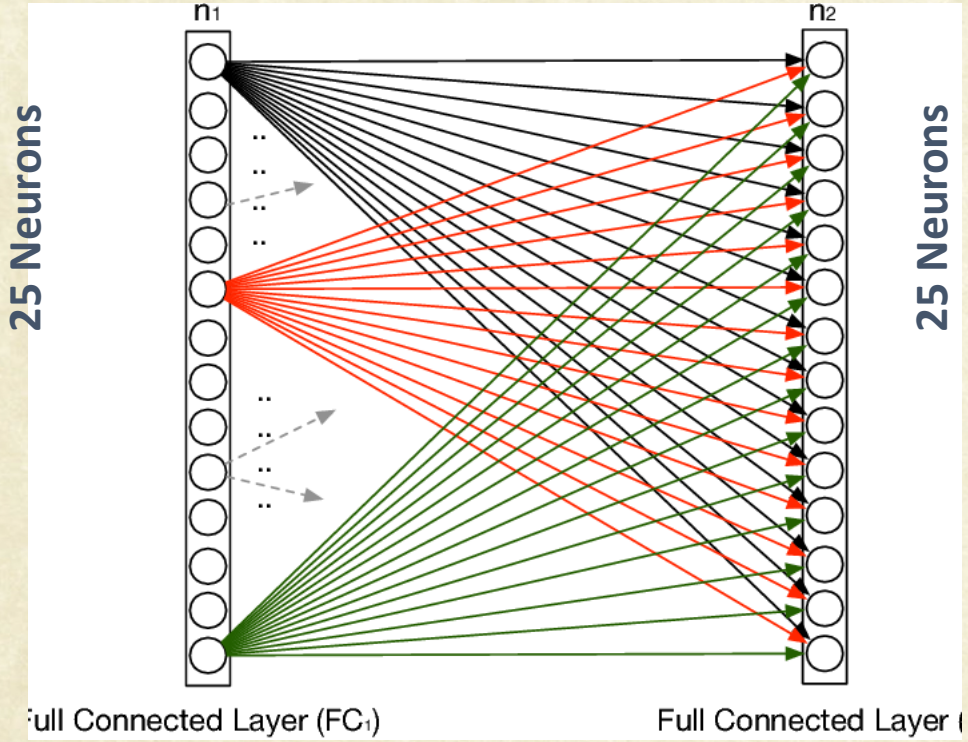


Parameter Calculation

- Stride S
- Zero padding P
- Input Size: W_1 x H_1 x D_1
- Output Size: **W_2 x H_2 x D_2**
- W_2 = [(W_1 – F + 2P) / S] + 1 and D_2 = K
- S = 4, W = 227, F =11, P = 0, D_2 = 96
- W_2 = (227 -11)/4 + 1 = 55
- Output Size: **55 x 55 x 96**

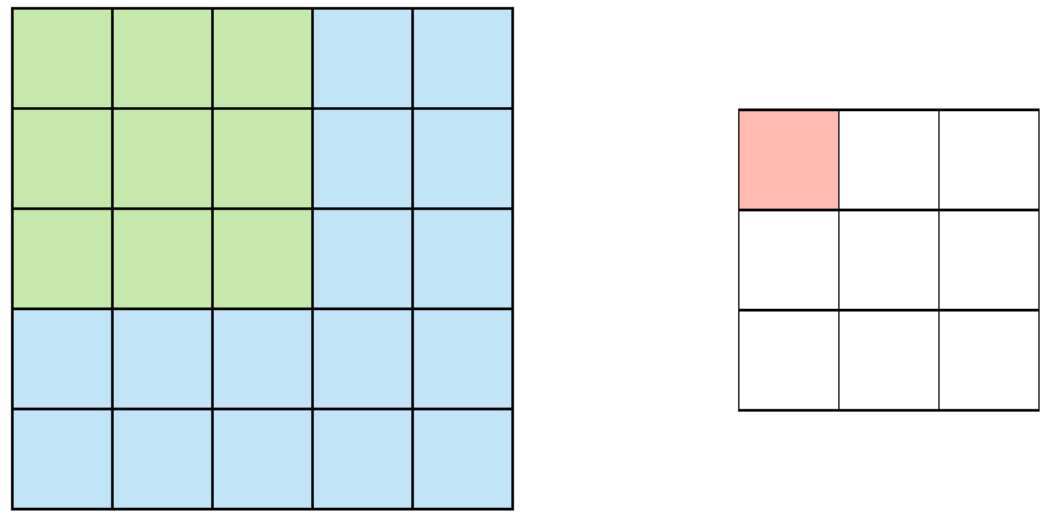


Fully Connected vs. Convolutional Layer



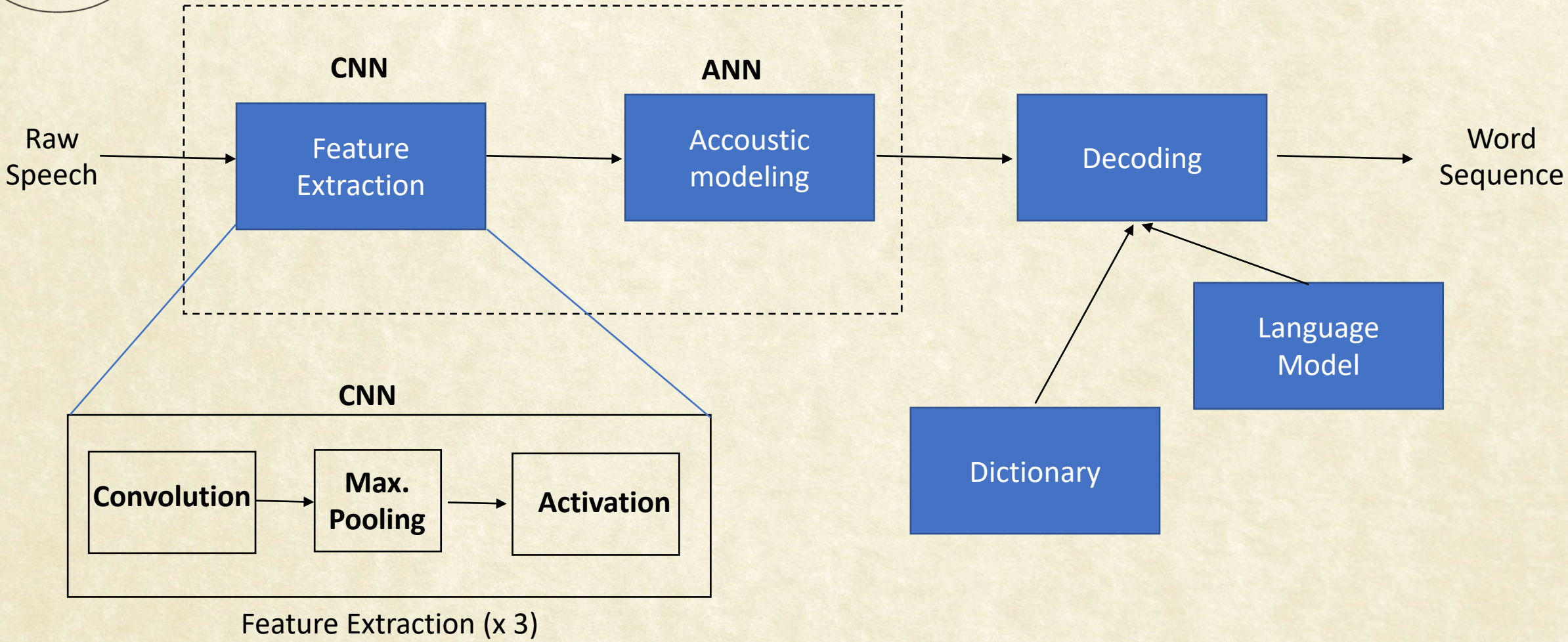
- Consider a 5*5 image (i.e., 25 pixels/values)
- **#Neurons** in L_1, L_2 = n_1, n_2 = 25
- **Weights** = $n_1 * n_2$ = 25 * 25 = 625
- **Operations**:
  - Multiplications(M) = n_2 per each neuron in L_1 = 25
  - Additions (A) = $n_2 - 1$ per neuron in L_1 = 24
  - Total operations = (M+A)* n_1 = (25+24)*25 = 1225

Fully Connected vs. Convolutional Layer



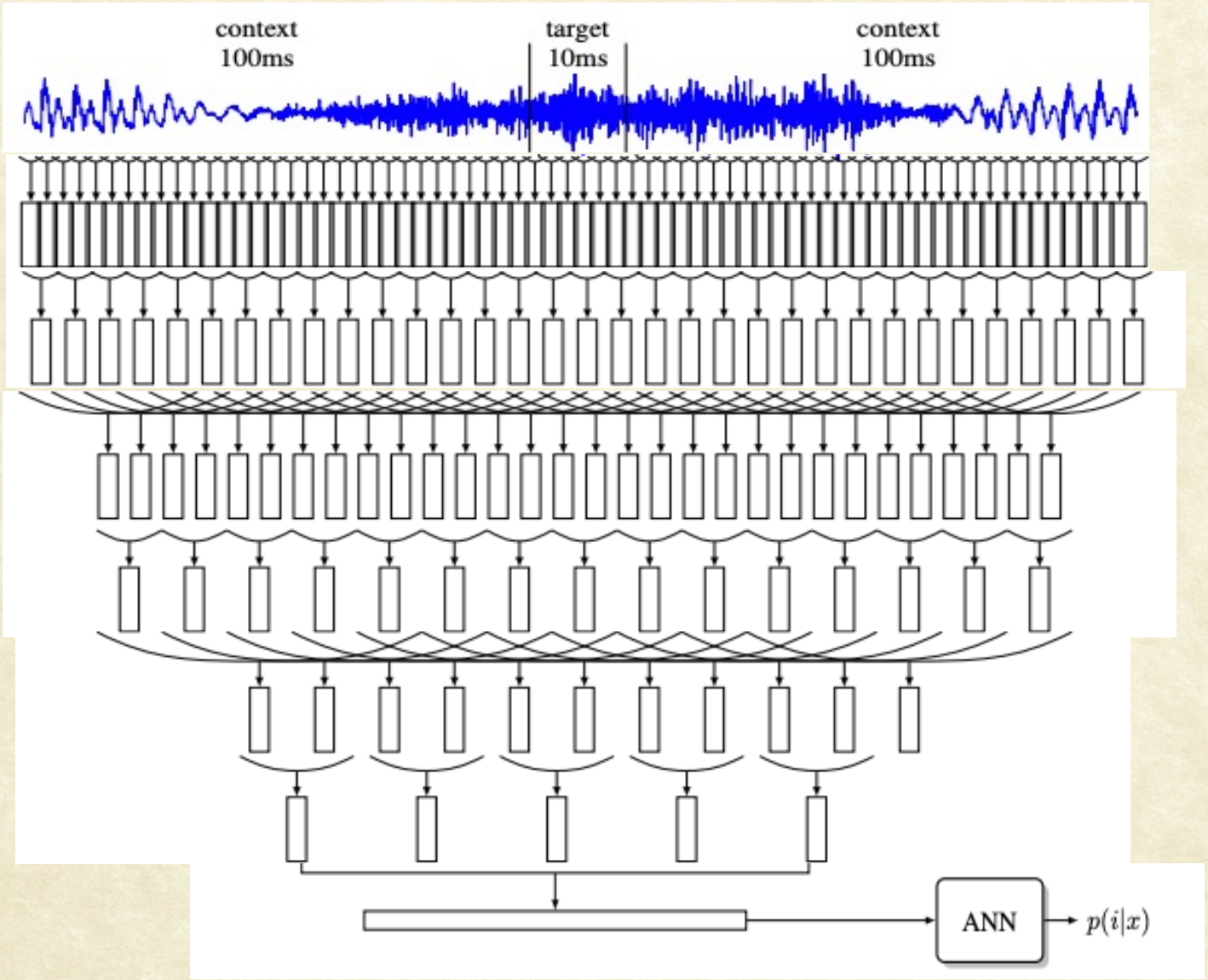
- Consider a 5 * 5 Image
- **#Filters** (n)= 25
- **Filter size** (K)= 3*3 (9 weights)
- **Stride**(S)= 1
- **Total steps** to cover the image (TS) = 9
- **Total Weights** = K * n = 9*25 = 225
- **Operations**
  - Multiplications (M)= 9 per filter at one step
  - Additions (A) = 8 per filter at one step
  - **Total Operations** = (M+A) * TS * n
    = (8+9)*9*25
    = **3825**

Speech Recognition with CNN



Dimitri Palaz, Mathew Magimai-Doss and Ronan Collobert, “Convolutional Neural Networks-based Continuous Speech Recognition using Raw Speech Signal”, ICASSP 2015, pp.4295-4299.

Detailed View



Convolution

Max pooling

Convolution

Max pooling

Convolution

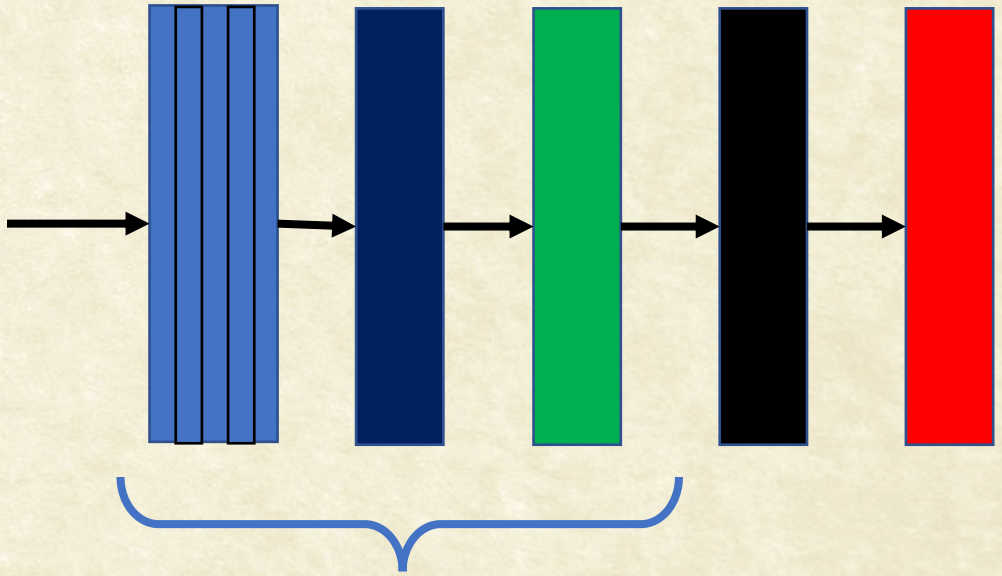
Max pooling

WSJ Word Error Rate:
64 % vs 5.6 % for
MFCC+ANN vs CNN

https://pdfs.semanticscholar.org/d224/6d41fe4ca1cb1f13eaa7bec0b86b95fdd8ac.pdf

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

Questions?