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

- 1D Convolution Layer
- 2D Convolution Layer and Fully connected layer

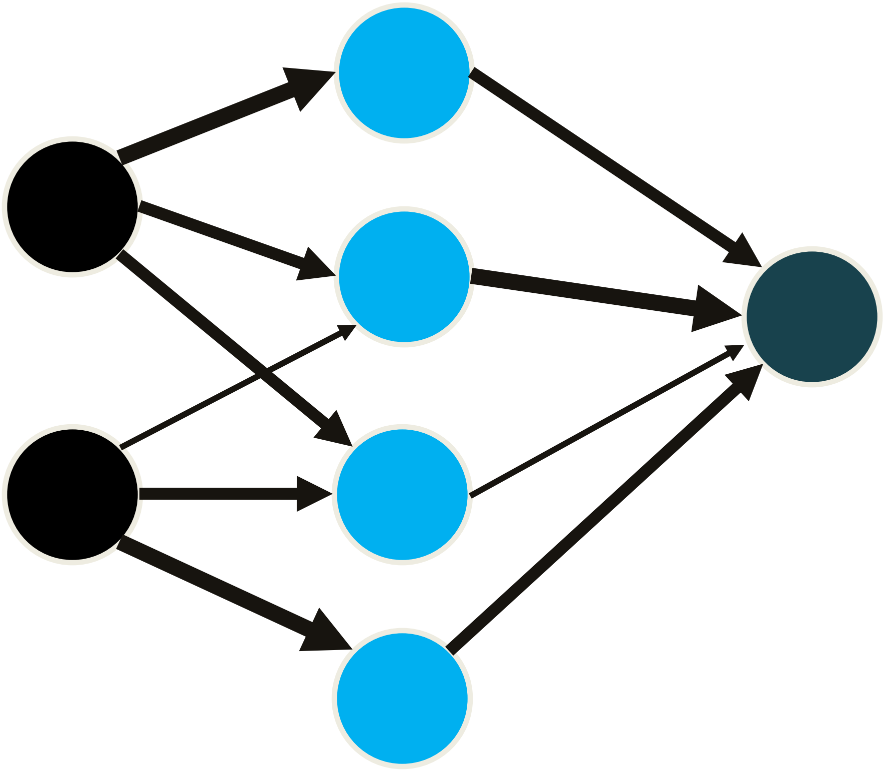
Convolution Layer

Neural Networks (Intro to CNNs)

Neural Networks

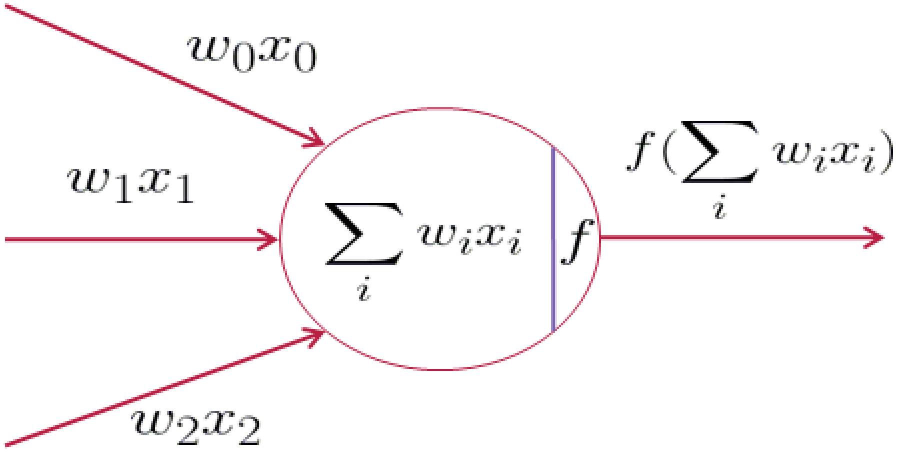
A Simple Neural Network

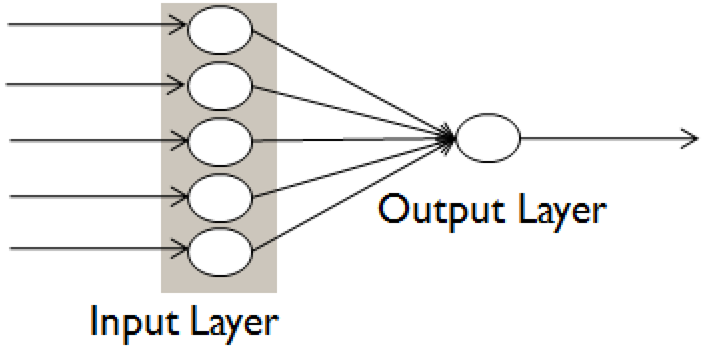
Input Layer *Hidden layer* *Output Layer*



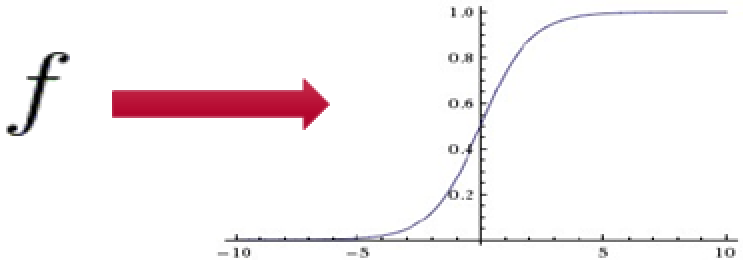
- Biologically inspired networks.
- Complex function approximation through composition of functions.
- Can learn arbitrary Nonlinear decision boundary

Neuron, Perceptron and MLP

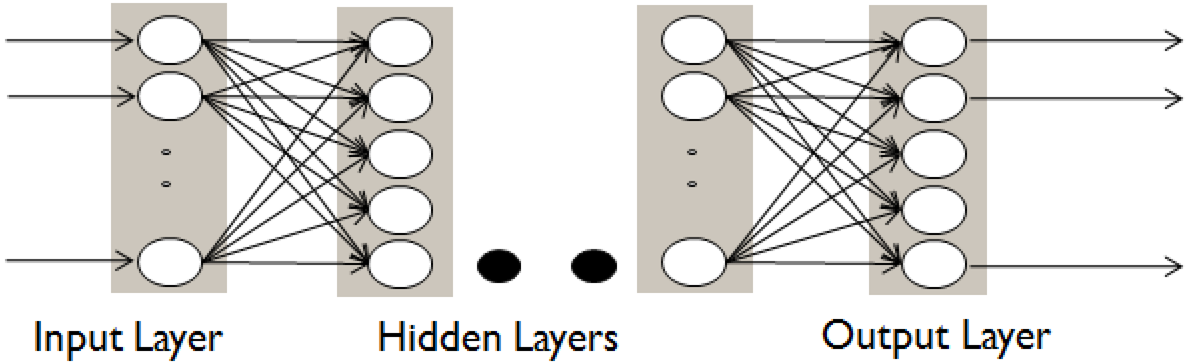




Perceptron



E.g. Sigmoid Activation Function



Multi Layer Perceptron

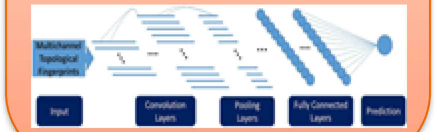
BLANK SLIDE/Discussions

BLANK SLIDE/Discussions

Convolutional Layer

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

- 1D Convolution Layer



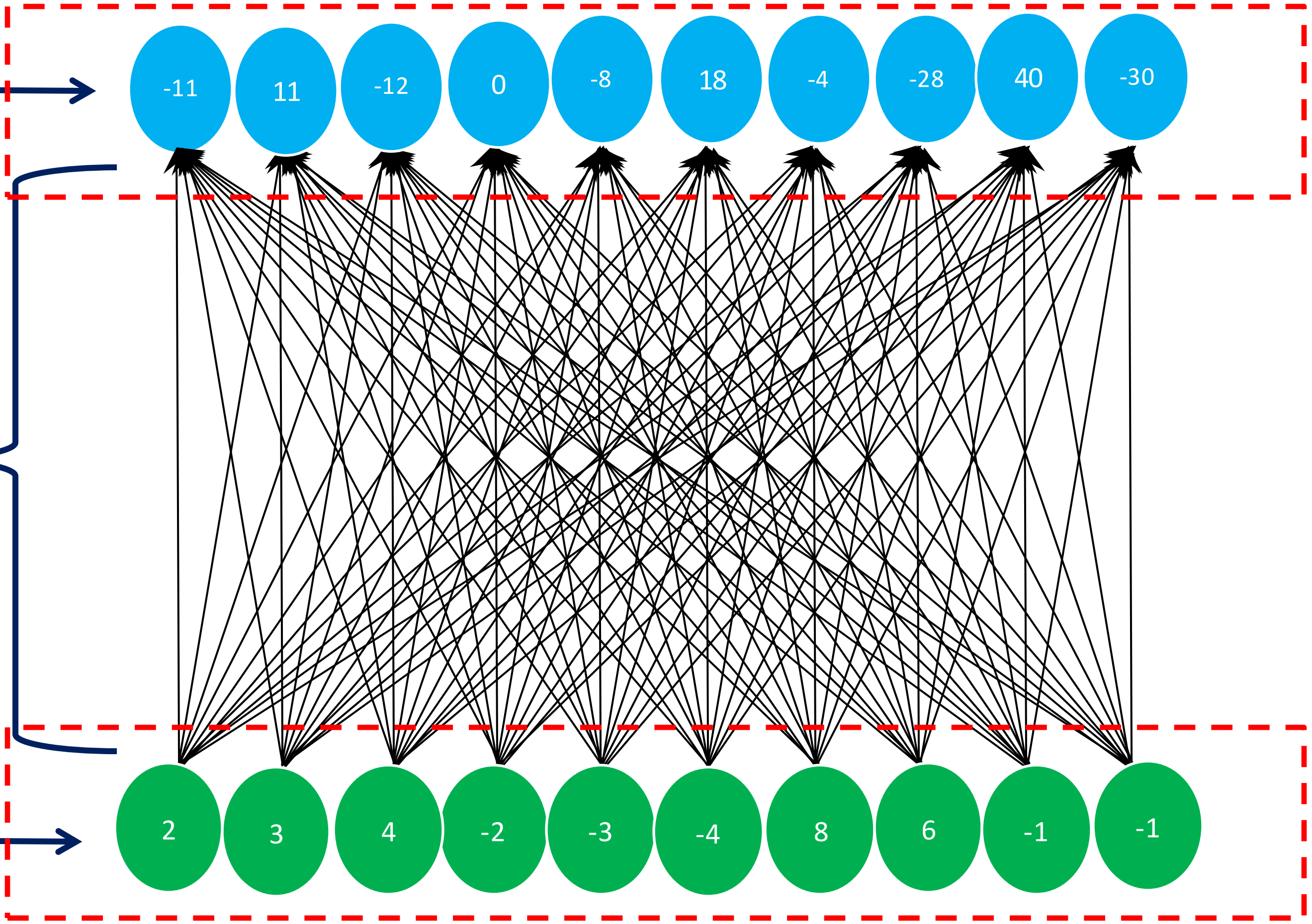
Example: 1D Convolution layer

Dense connections

$$Y = WX$$

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

$$X$$

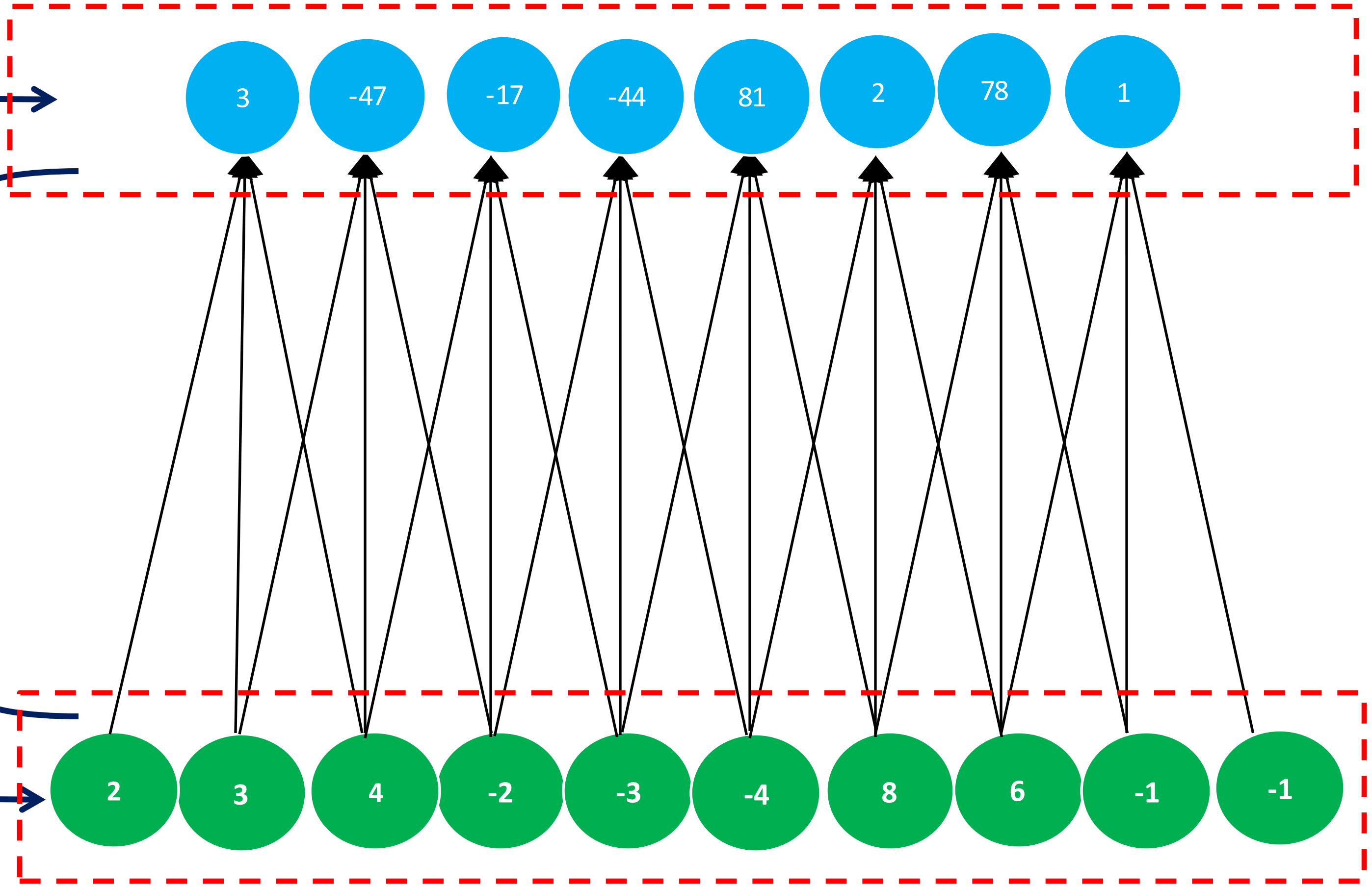


What if connections are only local?

$$Y = WX$$

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

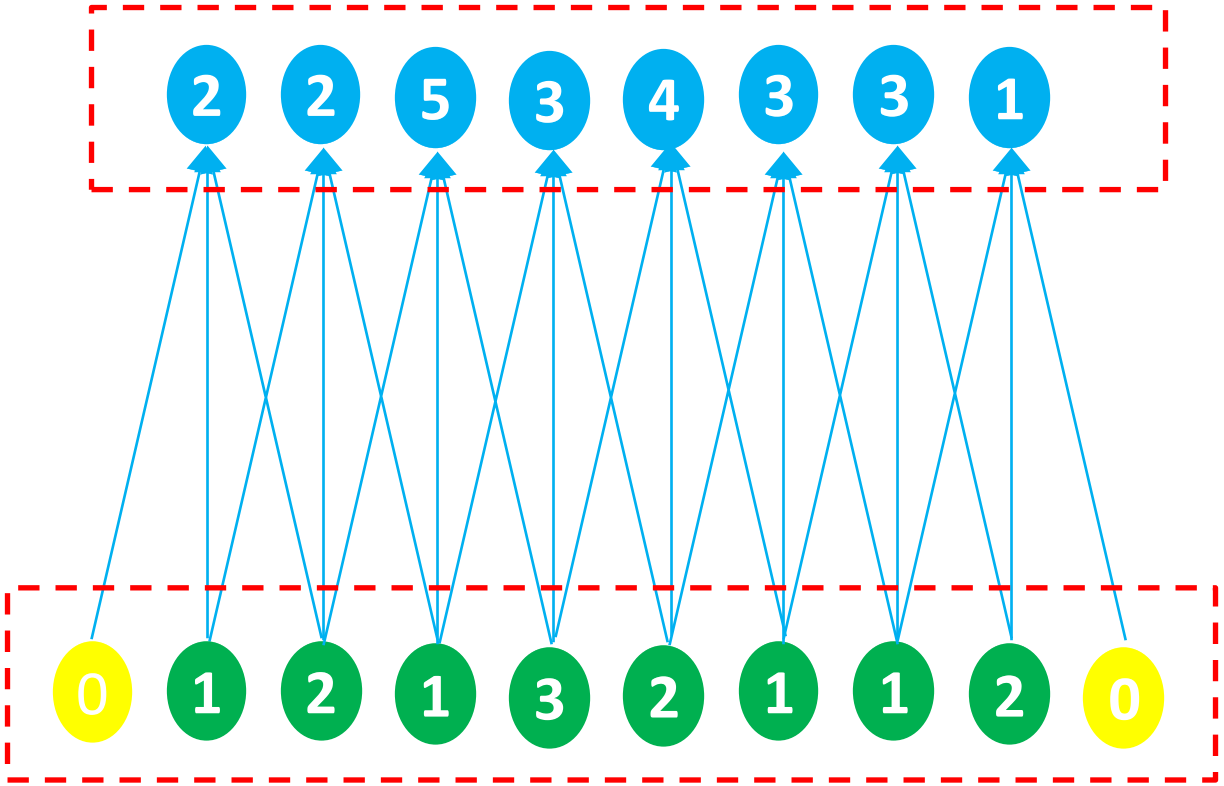
$$X$$



What if weights are same/shared?

Filter-1

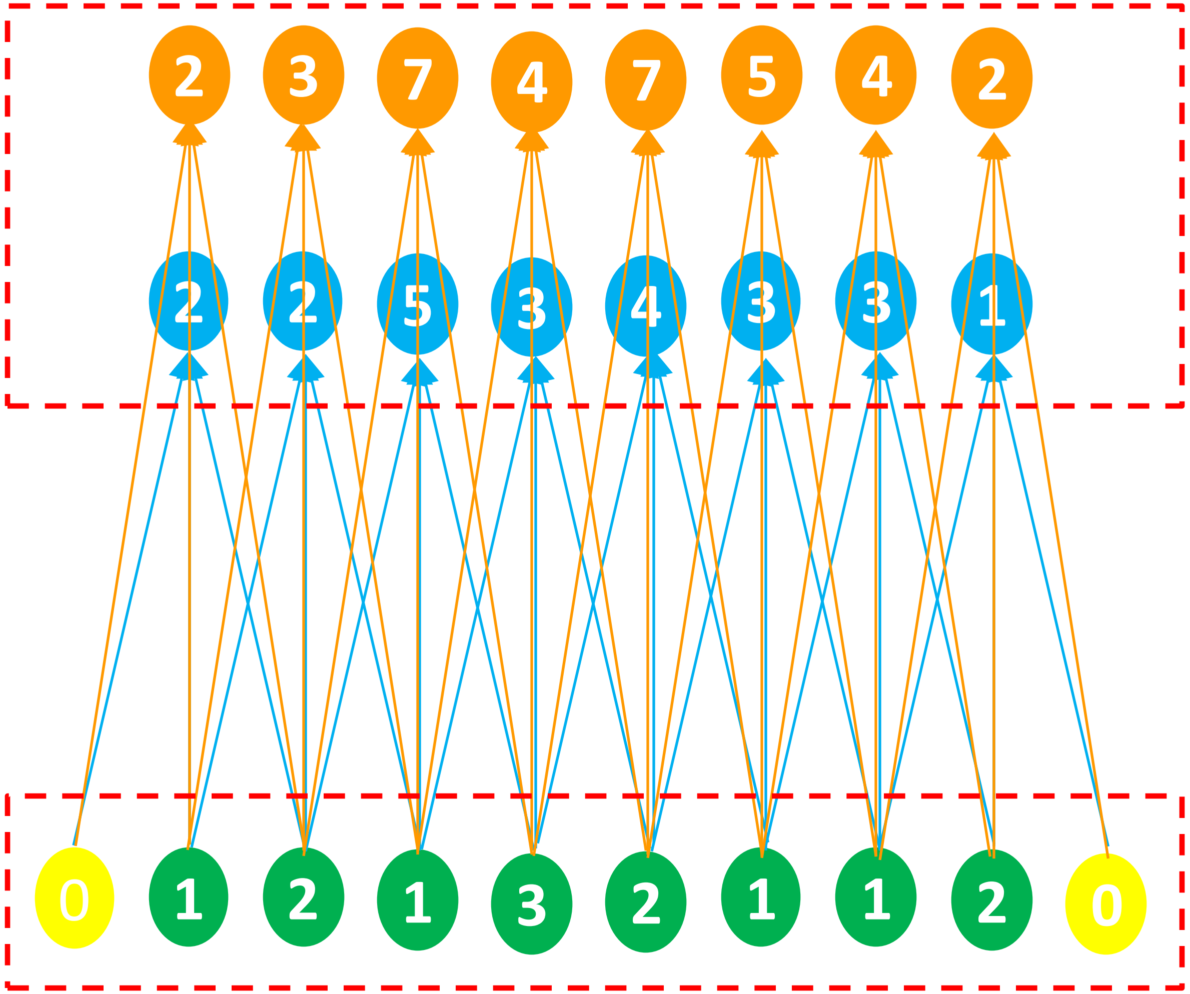
1	0	1



Weights that eventually we learn with Backpropagation (filter size = # Parameters = # weights = 3)

Two such filters/weights

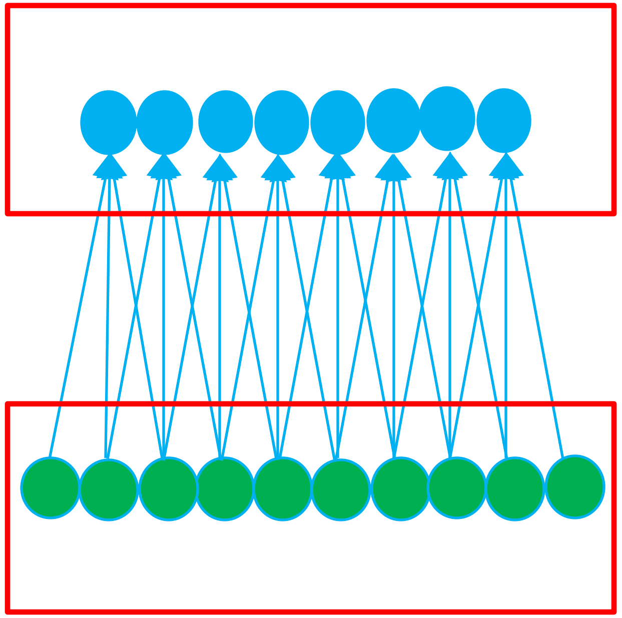
Filter-2	2	0	1
Filter-1	1	0	1



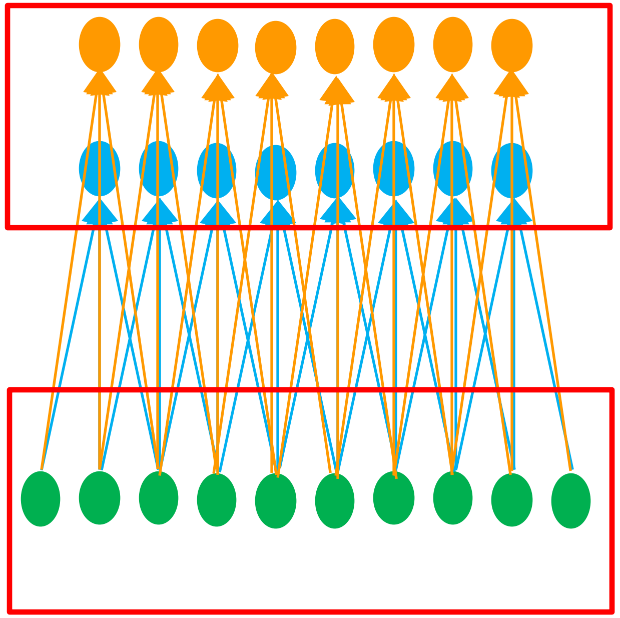
Pad Extra Samples so that output size does not reduce

Weights that eventually we learn with Backpropagation (filter size = # Parameters = # weights = 3)

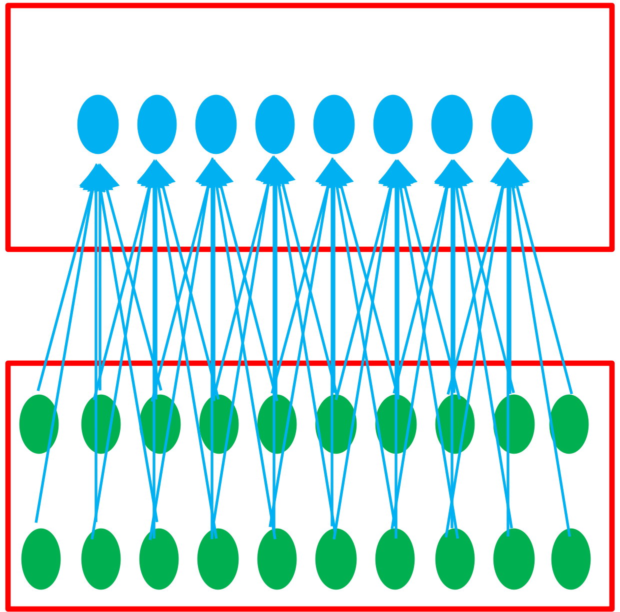
Convolution layer: Different Possibilities



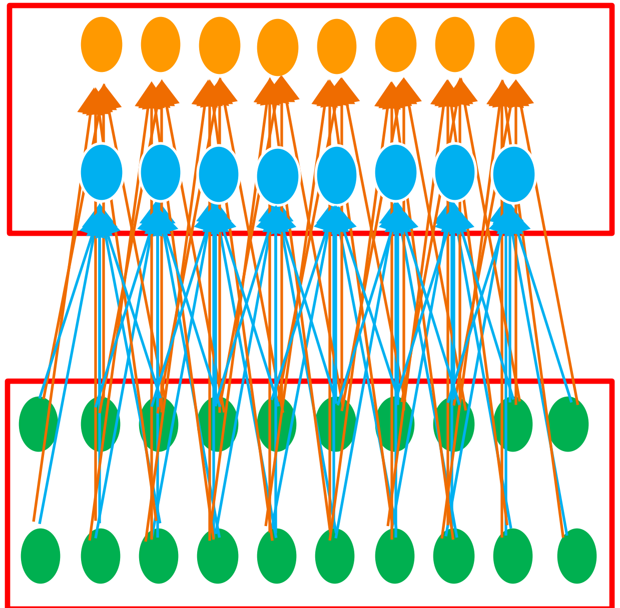
Channels:
- I/P =1
- O/P=1
- #Parameters = 3



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

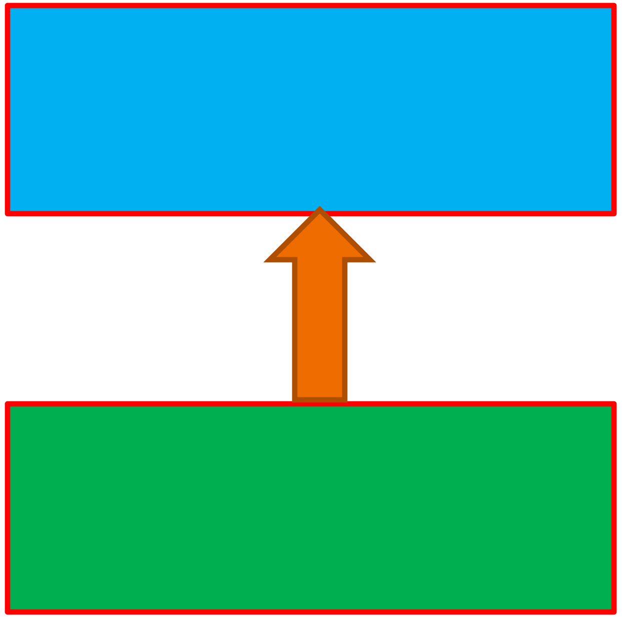


Channels:
- I/P =2
- O/P=1
- #Parameters = 6

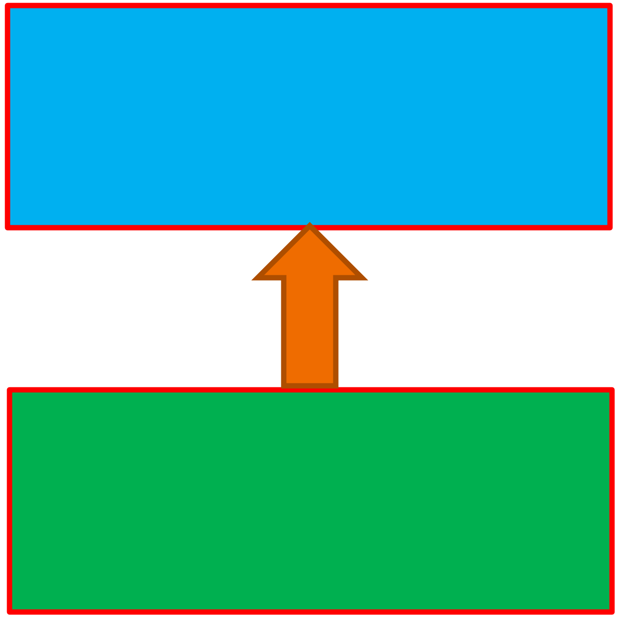


Channels:
- I/P =2
- O/P=2
- #Parameters = 12

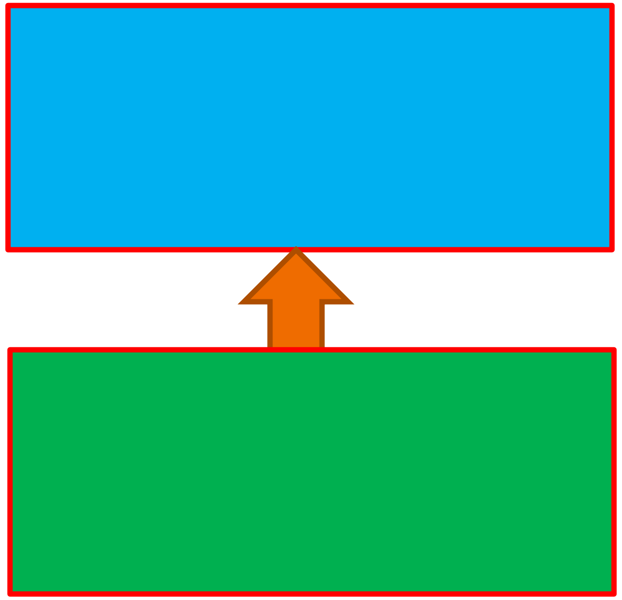
Convolution layer: Different Possibilities



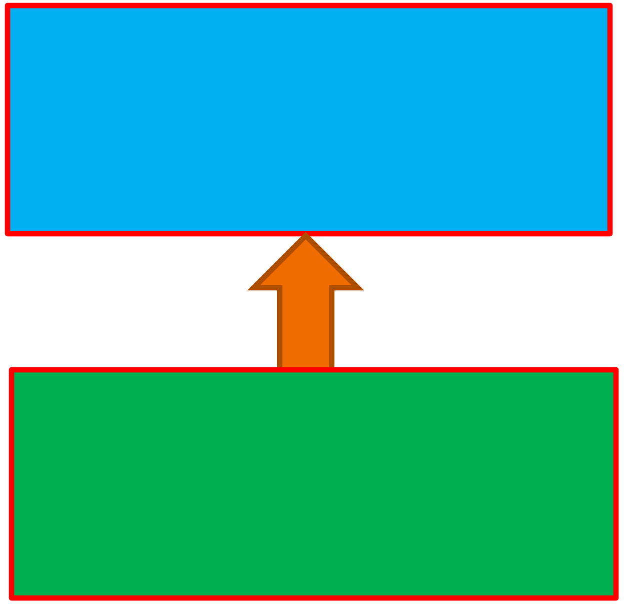
Channels:
- I/P =1
- O/P=1
- #Parameters = 3



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

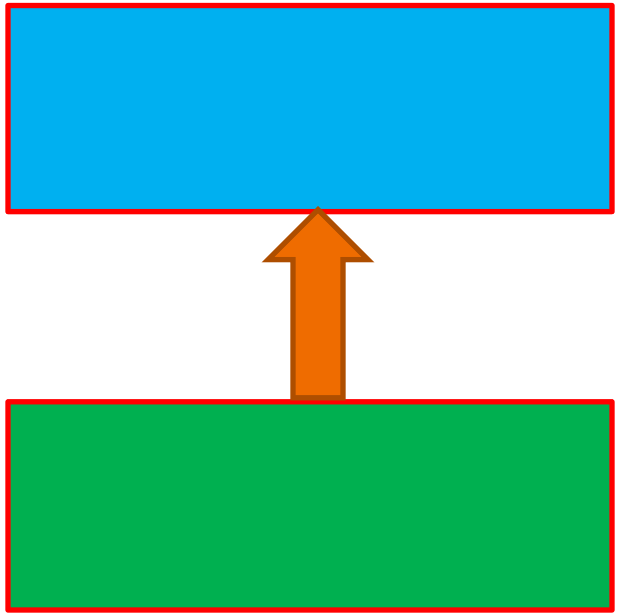


Channels:
- I/P =2
- O/P=1
- #Parameters = 6



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

We know by now ..



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

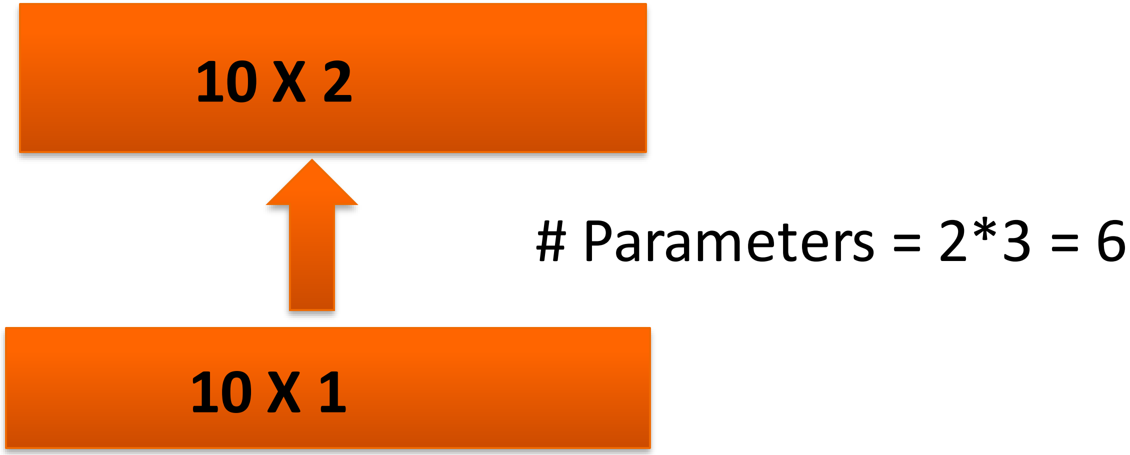
Key Words

- # Input Channels
- # Output channels
- # Weights/Parameters

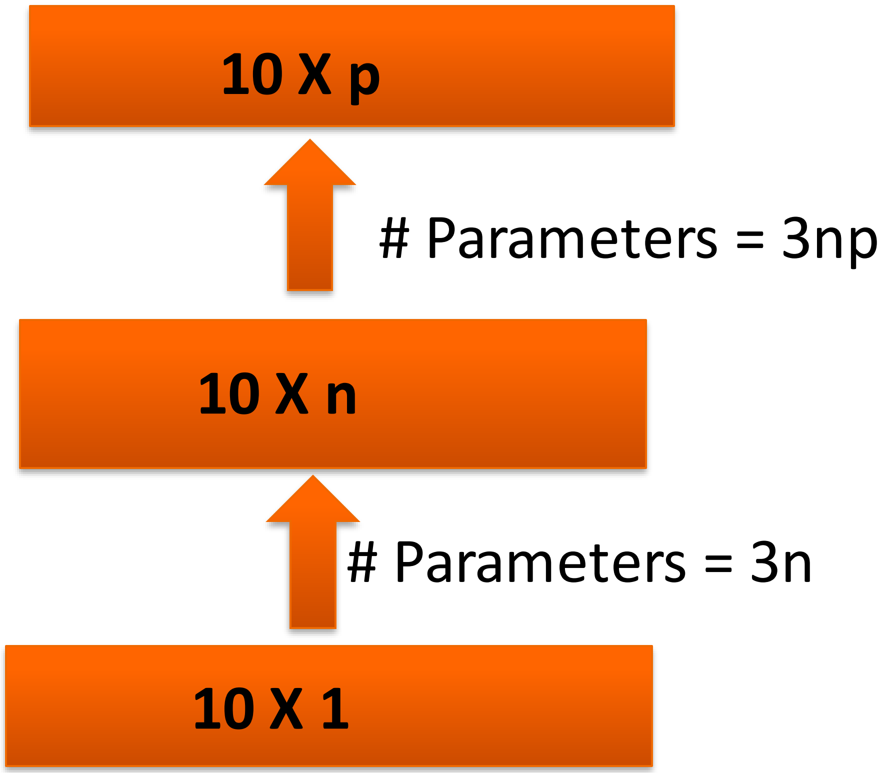
Key Words

- Feature Maps/Channels
  - A representation of the data
- Filters/Weights
  - Learnable parameters (problem specific)
- Filter Size/Window Size
  - We can change. Not much to play with.
- Stride (wait)
  - Skip/reduction in size
- Padding (seen ?)
  - Extra elements so that no reduction

Pictorially Summarizing



Total parameters = 3n + 3np



Filter size = 3

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

- ✓ 1D Convolution Layer
- 2D Convolution Layer and Fully connected layer

Convolution layer in 2D (popular)

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

6 x 6 image

$\boxed{3}$	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
$\boxed{3}$	-2	-2	-1

Convolution

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

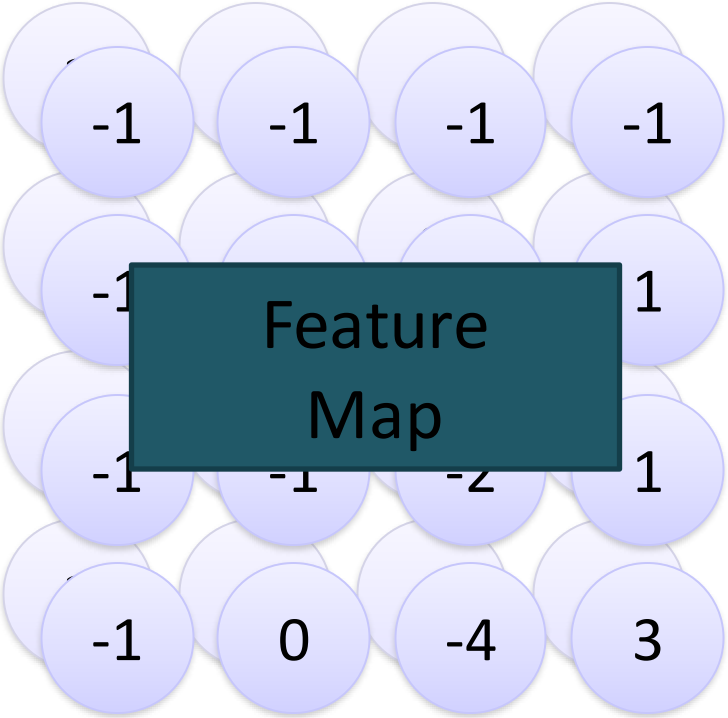
Filter 2

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

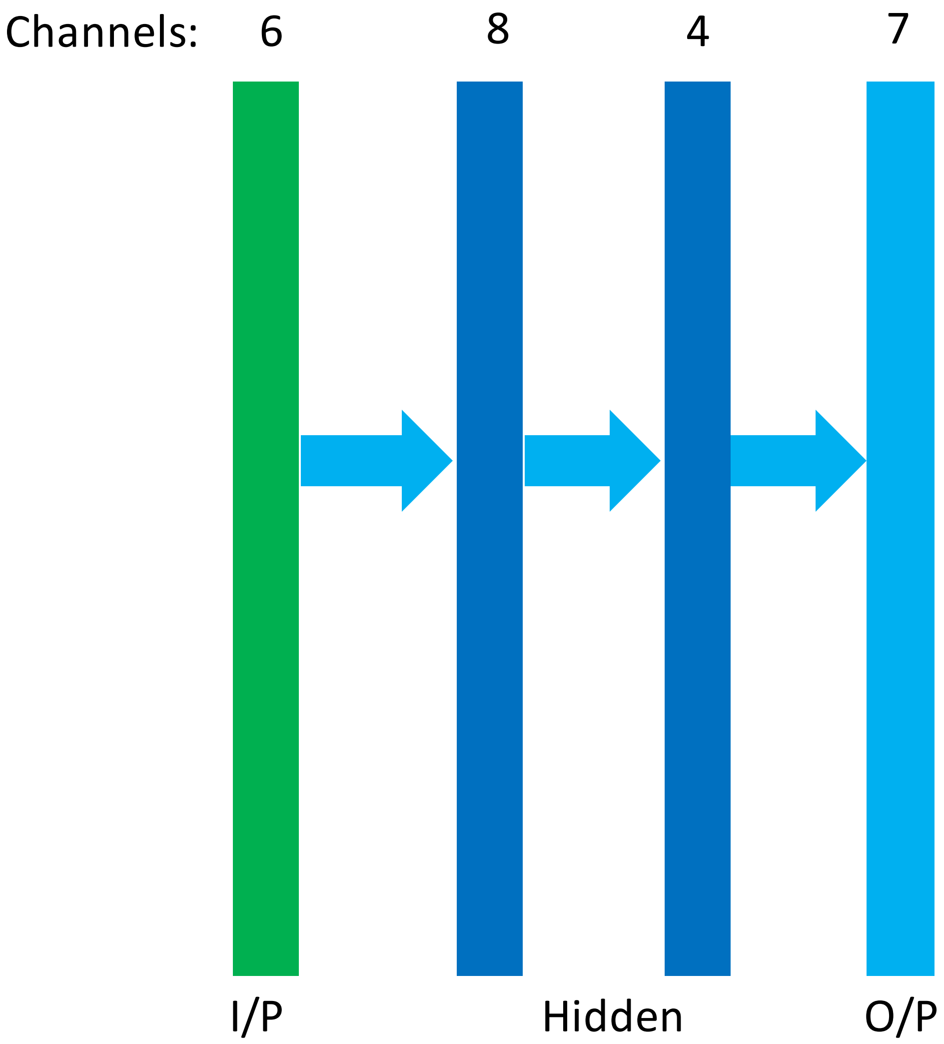
6 x 6 image

Repeat this for each filter

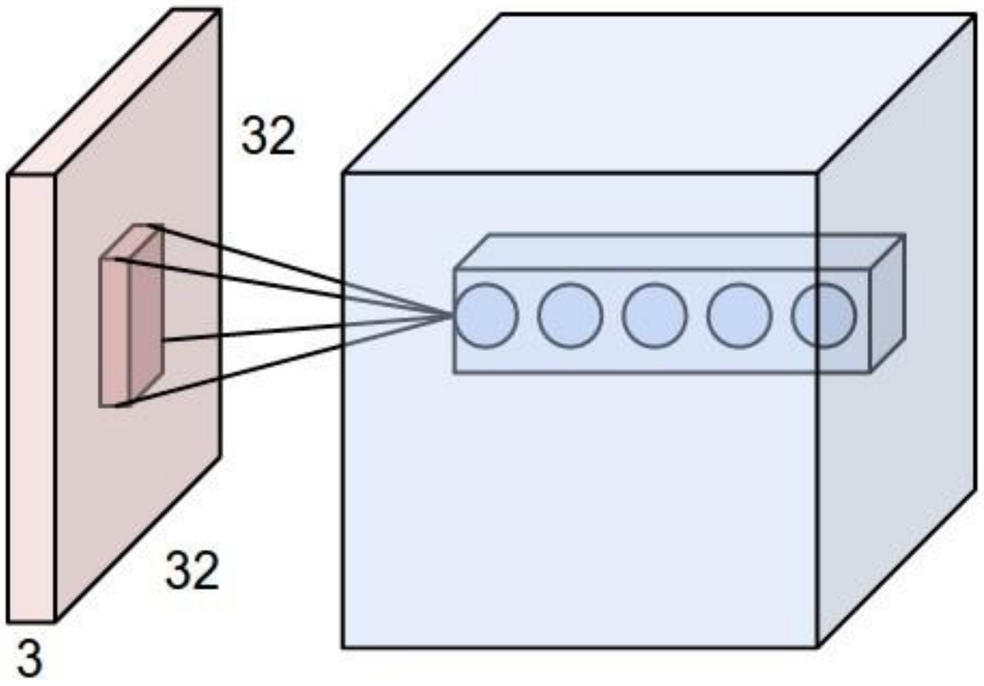


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Layer wise abstraction

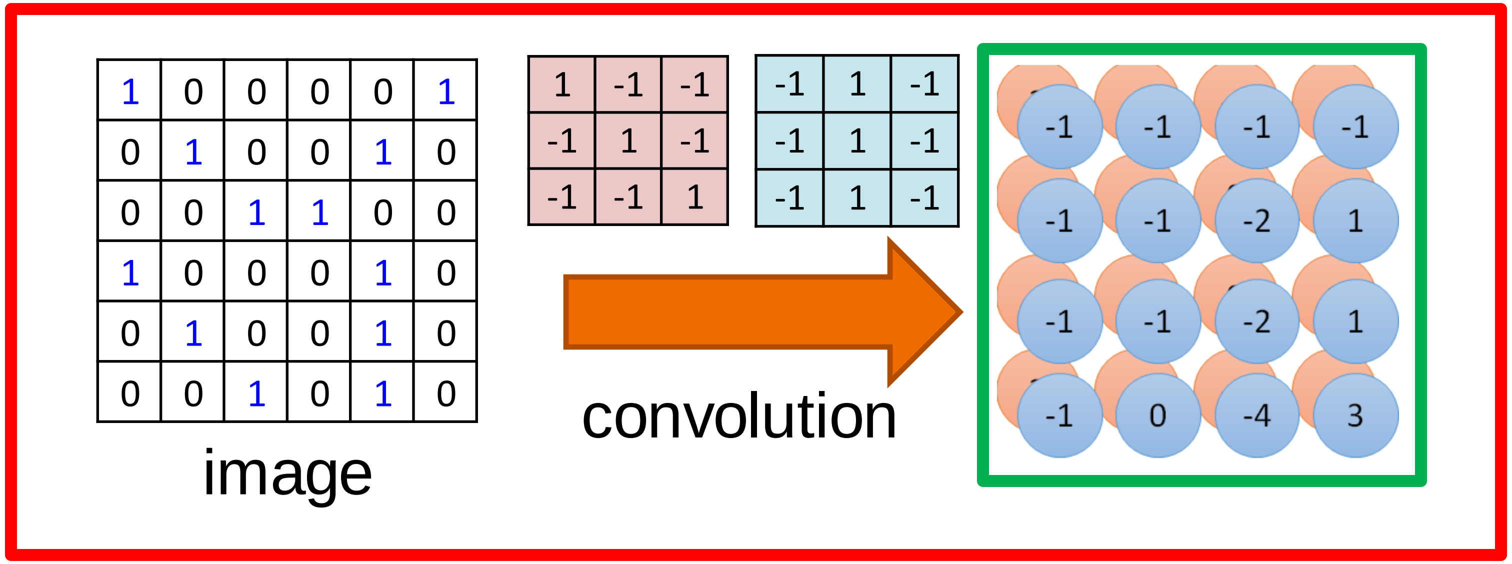


1-D Convolution

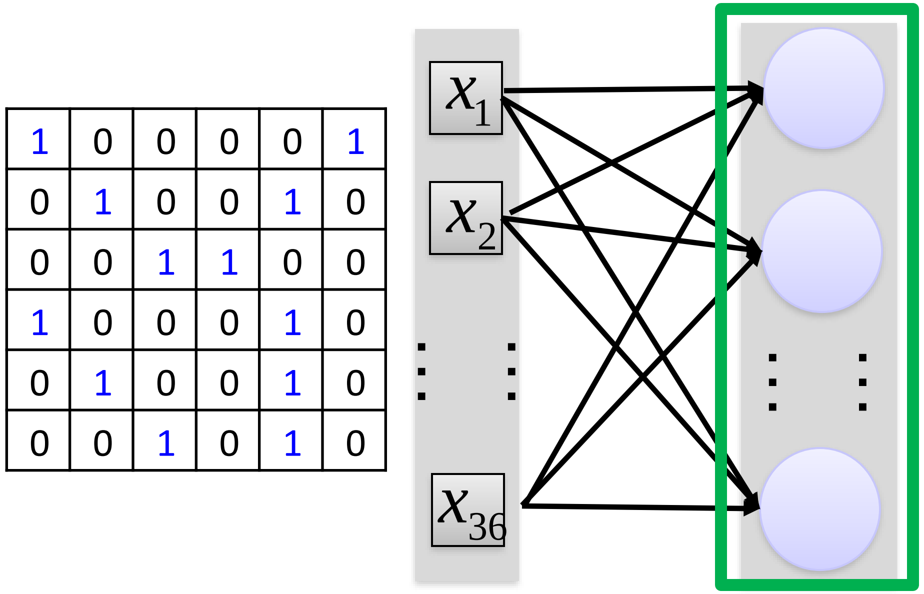


2-D Convolution

Convolution vs Fully Connected



Fully-connected



Fully connected

- Multi layer perceptron
- Role of a classifier
- Generally used in final layers to classify the object represented in terms of discriminative parts and higher semantic entities.
- SoftMax
  - Normalizes the output.
  - K is total number of classes

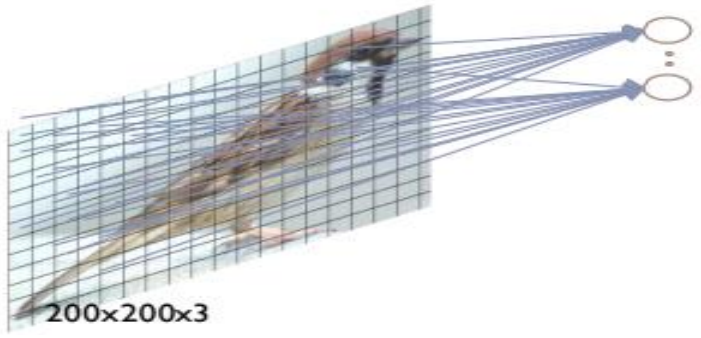
$$z_n = \frac{e^{x_n}}{\sum_{i=1}^{K} e^{x_i}}$$

Convolution Layer

- Maps a representation to another representation
  - Eg. A colour image to an “edge” image
  - Eg. A image to “heat maps” of parts of interest
    - Eg. “Hate words” in a sentence
    - Eg. “Location of eyes” in an image
- Role of Pattern Detector
- Sequence of convolutional layers extract higher and higher levels of patterns.

Convolution layer

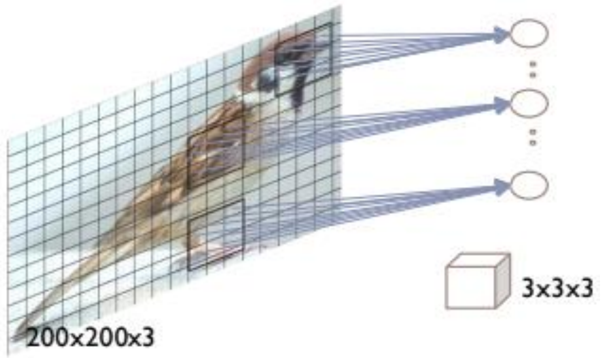
- **Fully connected layer**



- 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

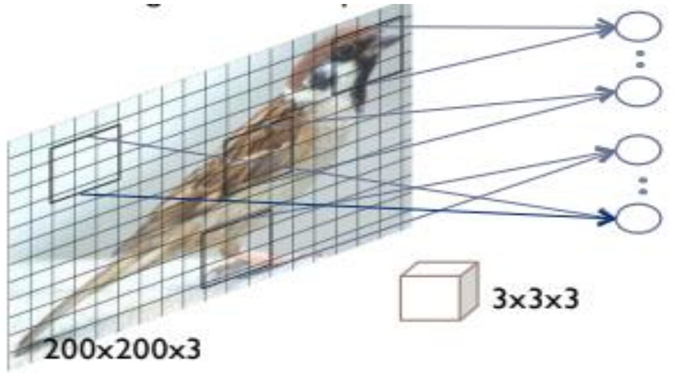
- **Locally connected layer**



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

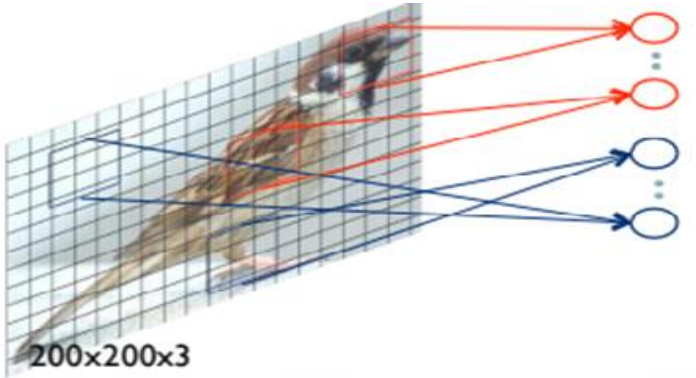
Convolution layer

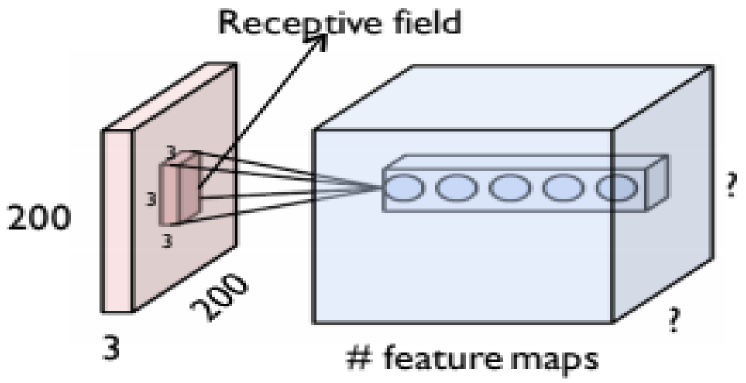
- **Convolutional layer with a single feature map**



- #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**





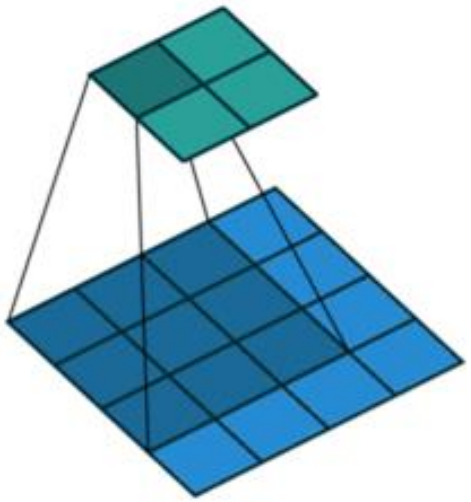
New Terms

- **1-D Convolution**
- **2-D Convolution**
- **Padding**
- **FeatureMap/Channels**
- **FeatureSize/WindowSize**
- **FilterCoeff/Weights**
- Stride and Pool (Seen)
- Parameter Calculations (Seen)

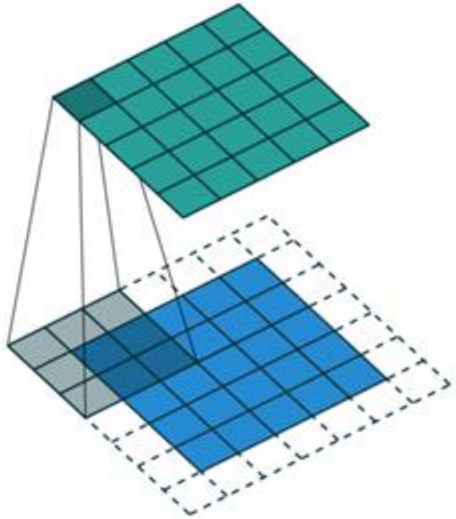
BLANK SLIDE

CNNs

- Window size
- Stride
- Padding

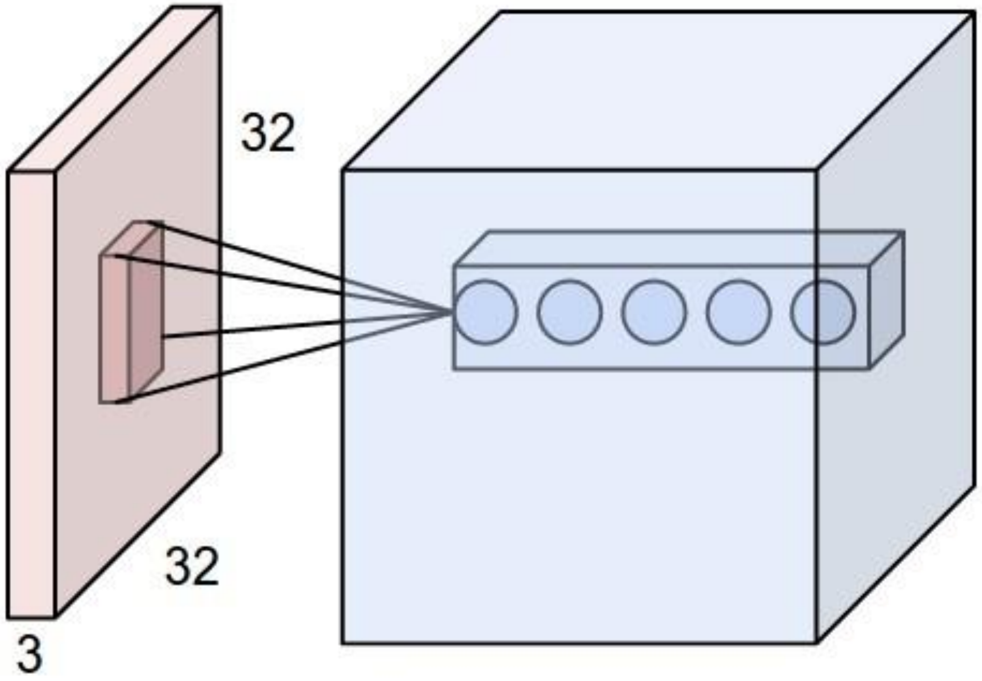


Window size: 3x3
Stride: 1
Padding: 0



Window size: 3x3
Stride: 1
Padding: 1

Input, Output Channels : Multiple Filters



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

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	0	1	1	0
0	0	1	2	2	1	0
0	2	0	2	0	2	0
0	1	2	2	1	0	0
0	2	0	1	1	1	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	1	1	2	1	0
0	2	2	0	1	1	0
0	2	2	2	1	2	0
0	2	0	2	2	0	0
0	1	2	0	1	0	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	2	1	0	2	0
0	2	2	1	1	1	0
0	1	2	0	1	0	0
0	1	1	2	2	0	0
0	2	0	2	1	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

0	-1	-1
1	1	0
0	0	1

w0[:,:,1]

-1	-1	0
1	-1	0
1	-1	-1

w0[:,:,2]

-1	-1	0
-1	0	0
-1	1	1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	-1	0
0	-1	1
0	0	0

w1[:,:,1]

0	-1	0
0	1	0
0	1	0

w1[:,:,2]

-1	-1	0
-1	-1	0
1	1	1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

1	2	4
-2	-8	-4
-4	-4	-1

o[:,:,1]

5	3	1
-1	0	-3
-7	-9	-6

toggle movement

Back to Top

Summary

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

- ✓ 1D Convolution Layer
- ✓ 2D Convolution Layer and Fully connected layer

Thanks!!

Questions?