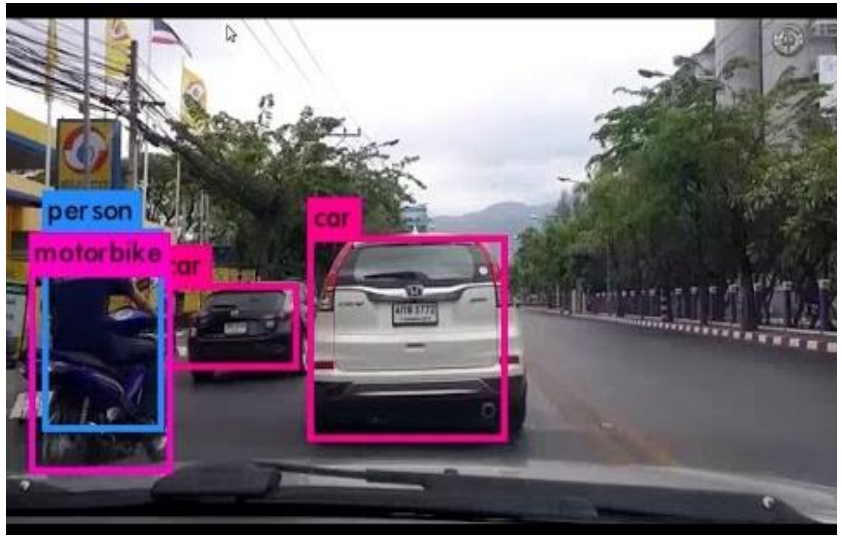
Convolutional Neural Networks

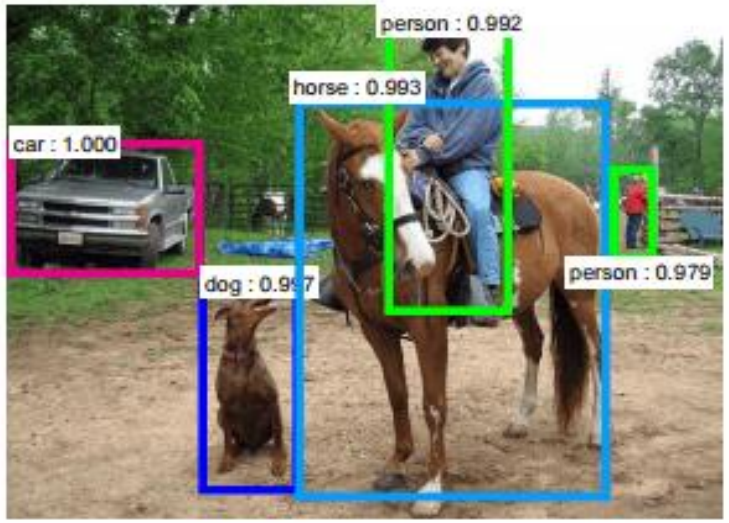
Face Recognition

Style Transferring

Image quality enhancement
Beautification



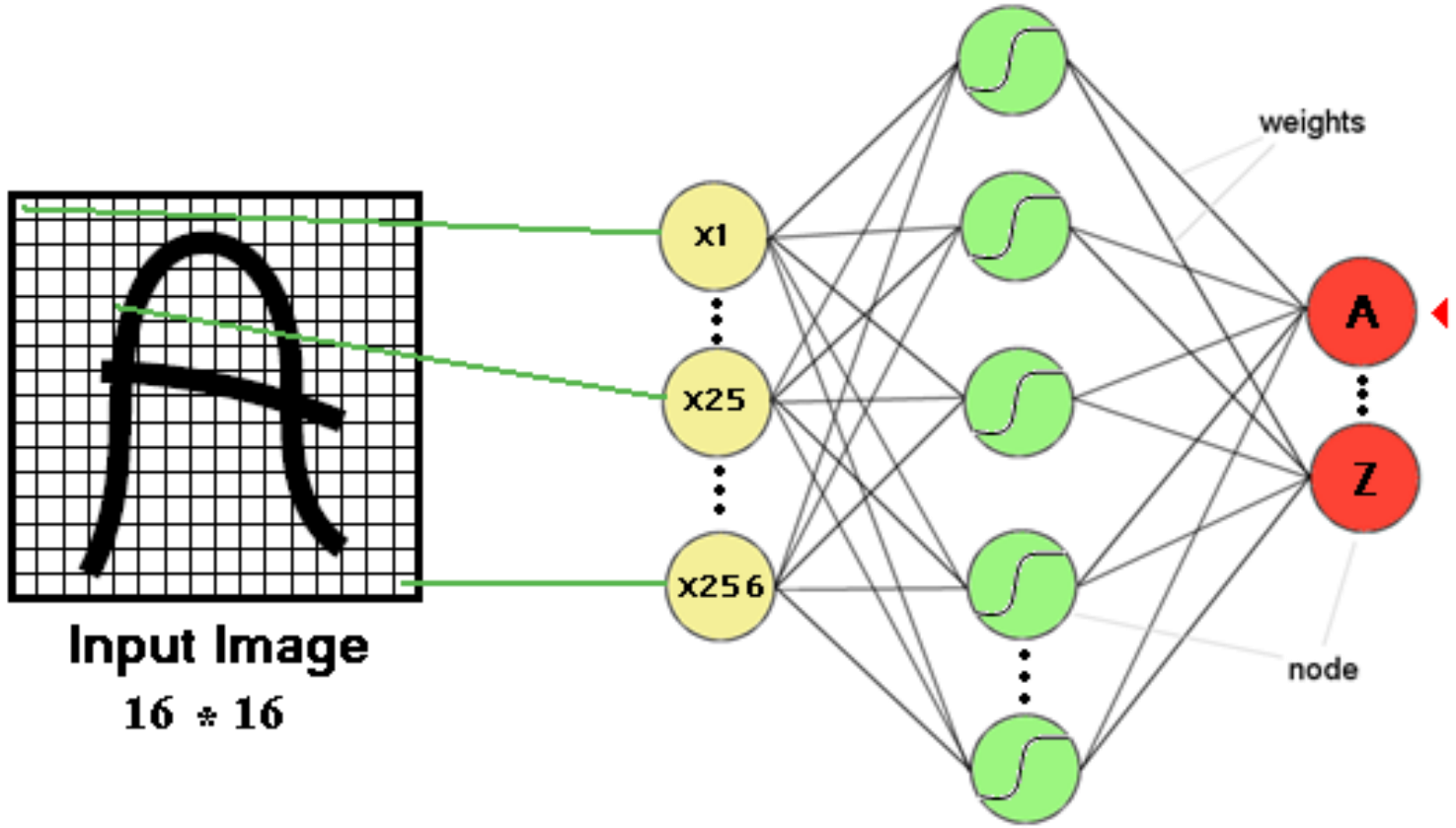
Object detection (Self driving car)



Classification

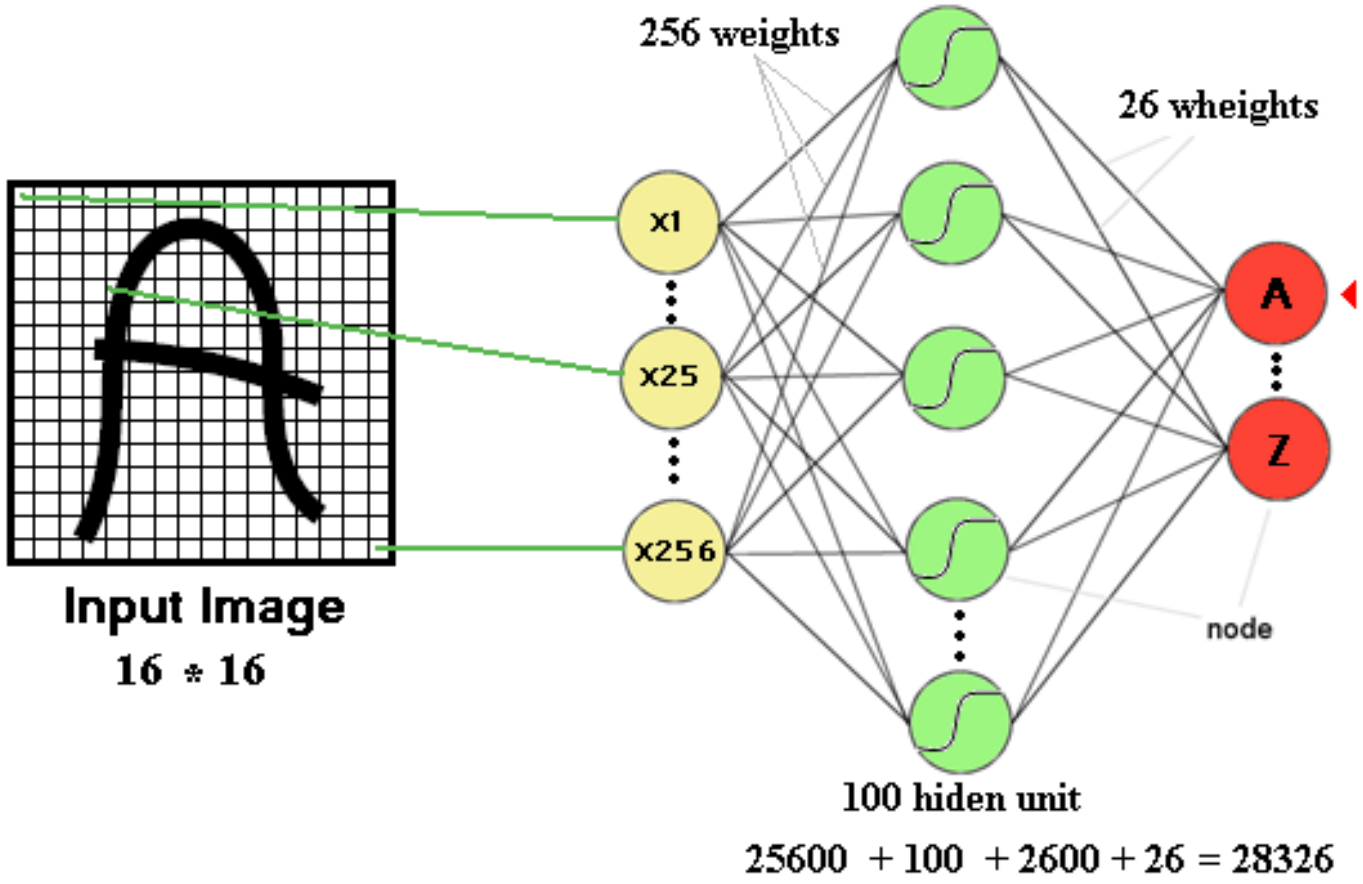
Gesture Recognition

Multi-layer perceptron and image processing



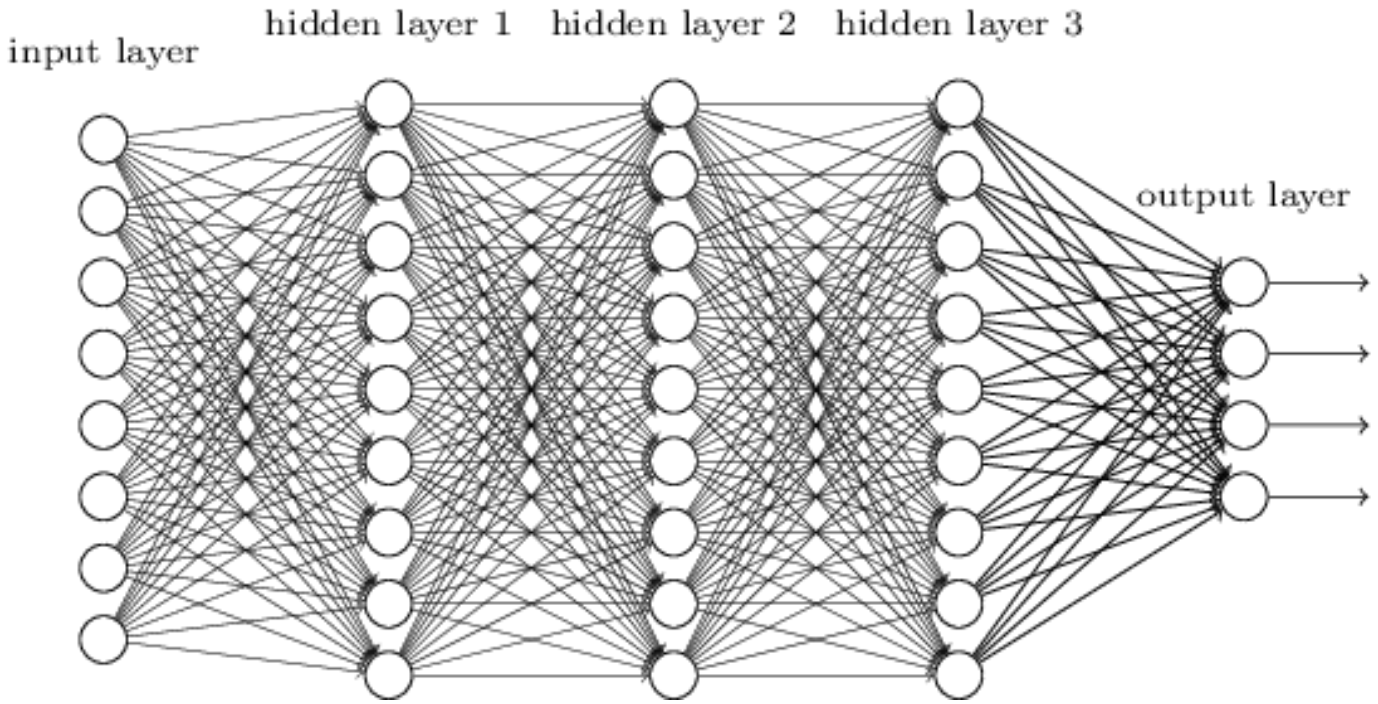
Even a simple 16x16 image

- **256x100 + 100 bias + 100x26 output neurons + 26 bias = 28236**
- the number of trainable parameters becomes extremely large



ANN: Too many parameters

- **We know it is good to learn a small model.**
- **From this fully connected model, do really need all the edges?**
- **Can some of these be shared?**



Can we do with less information?

- **How much information we can throw away and still recognize the object?**

10%

20%

50%

75%

- **Subsampling pixels will not change the object**

bird

Subsampling

bird

We can subsample the pixels to make image smaller

fewer parameters to characterize the image


TIPO EXCELSIUS

CNNs Vs. ANNs

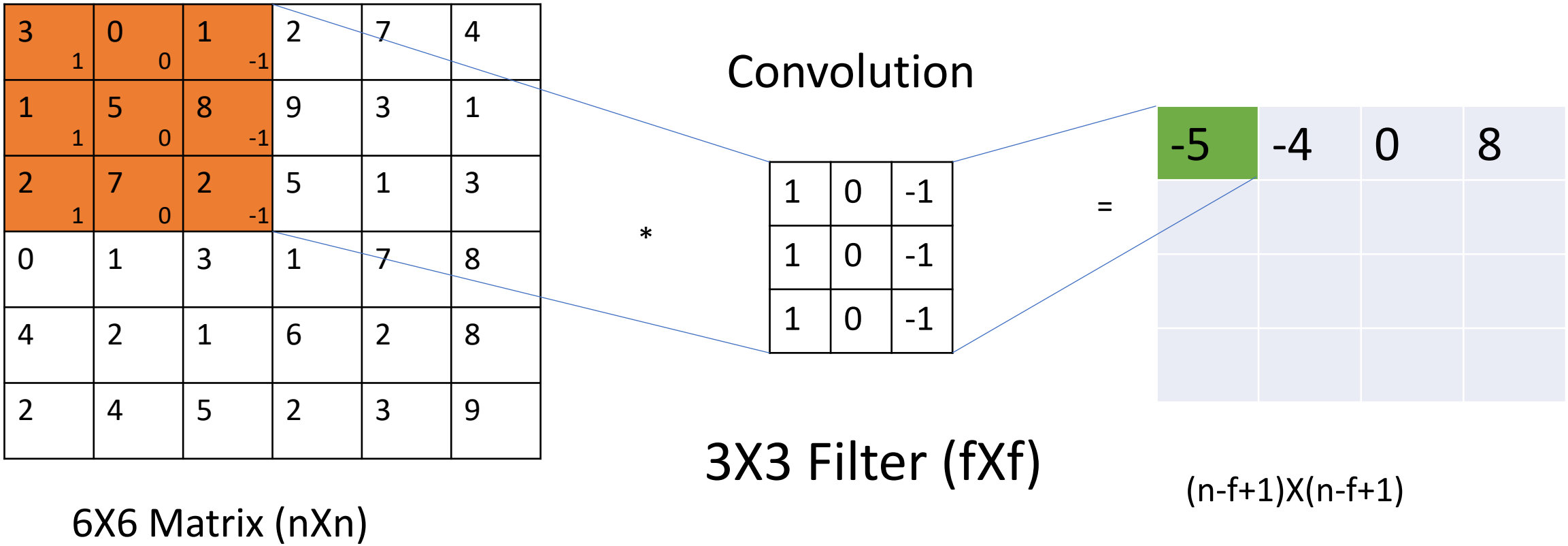
- ANNs suffer from curse of dimensionality when it comes to high resolution images
- We use filters (receptive fields) to exploit spatial locality by enforcing a local connectivity pattern between neurons of adjacent layers

- Parameter Sharing
- Sparsity of connection

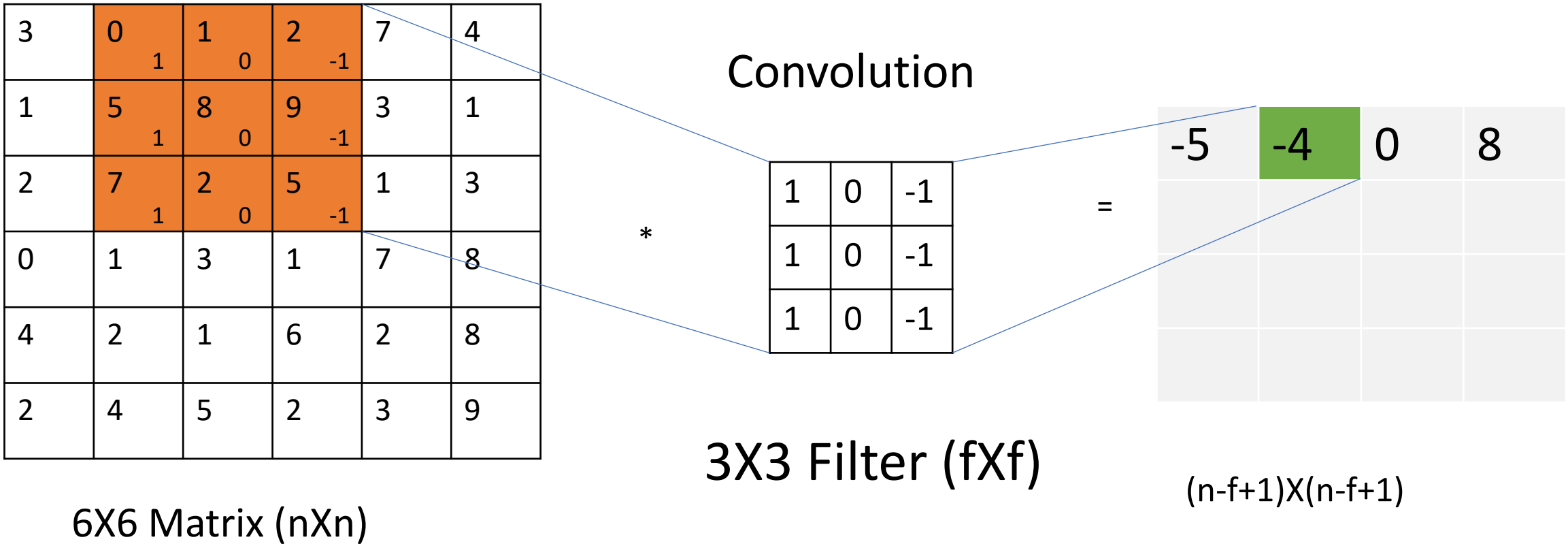
Convolution

- Convolution is a pointwise multiplication of two functions to produce a third function.
- Primary purpose of convolution in CNN is to extract features from the input image.
- Matrix formed by sliding the filter over the image and computing the dot product is called the ‘Convolved Feature’ or ‘Activation Map’ or the ‘Feature Map‘.

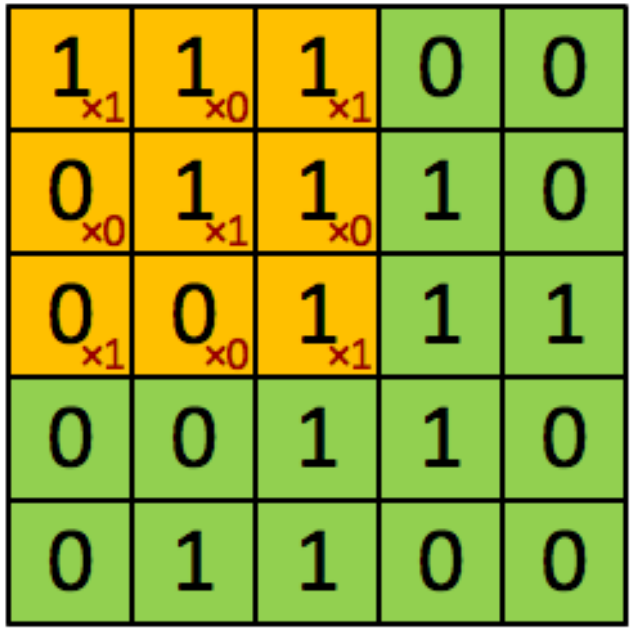
Convolution Example



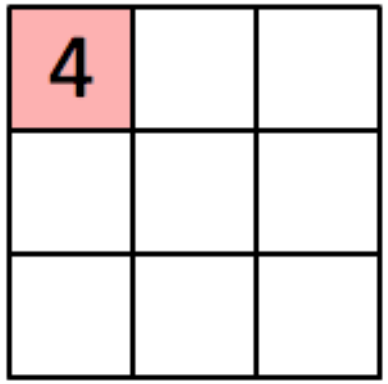
Convolution Example



Convolution Example



Image



Convolved Feature

Detecting Vertical edges

6*6 = 36

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

4*4=16

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Filter Weights

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

Horizontal Filter

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

Sobel Filter

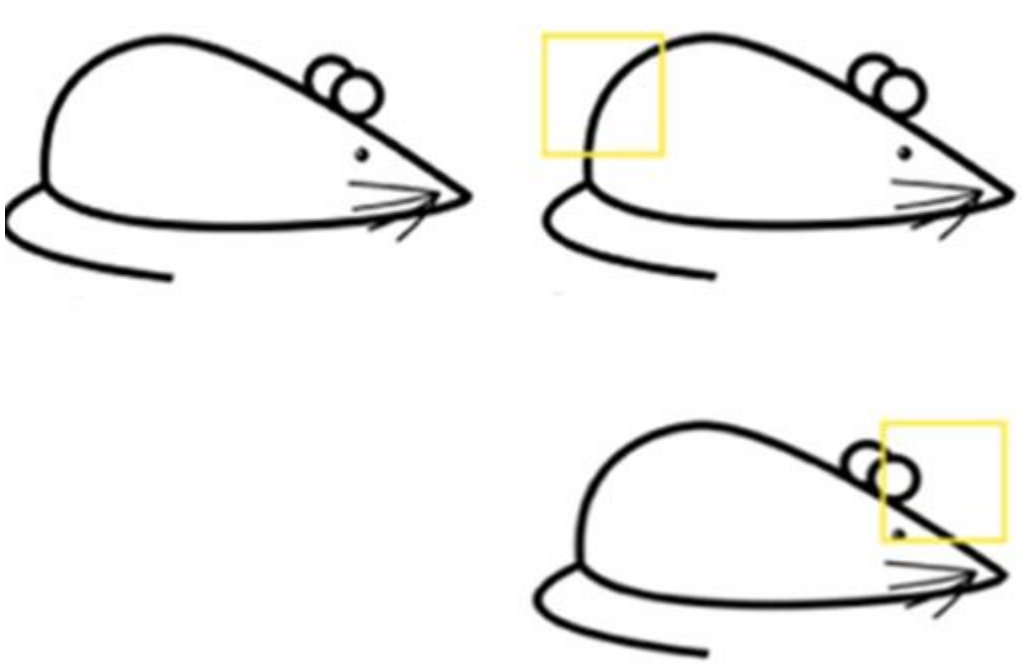
3	0	-3
10	0	-10
3	0	-3

Schorr Filter

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Convolutional Neural Networks automatically estimates the weights of the filter

More Intuition



0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

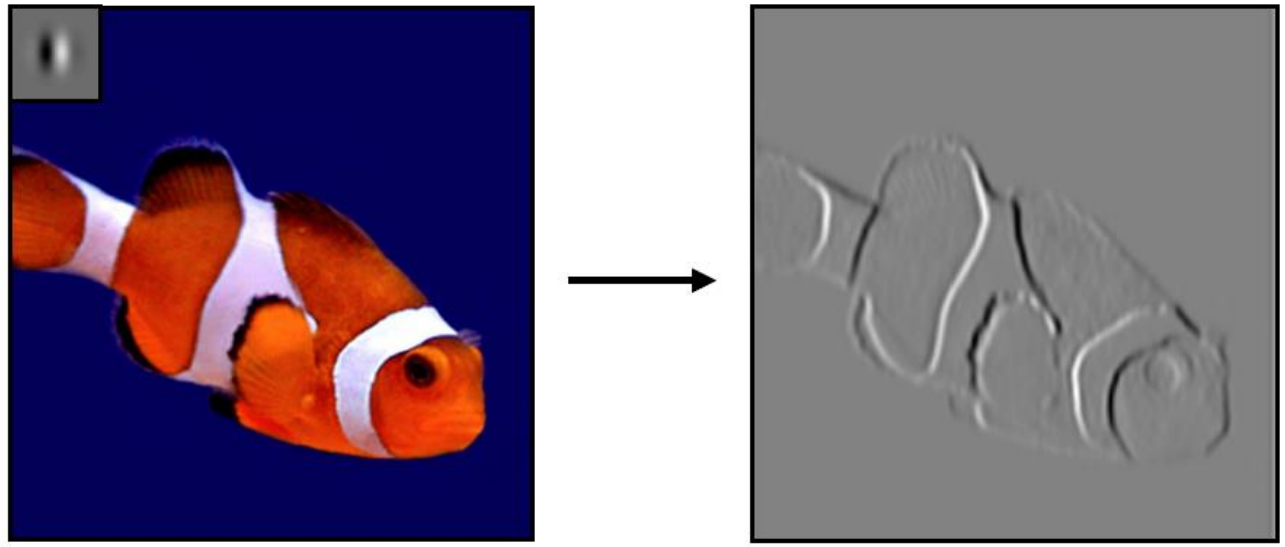
Pixel representation of filter

Visualization of a curve detector filter

Interpretation

Convolution is just another way of computing W^TX

In CNN, **input** is **image**, **kernel** is **convolution filter** to be learned, **response** is the **feature map**



Padding

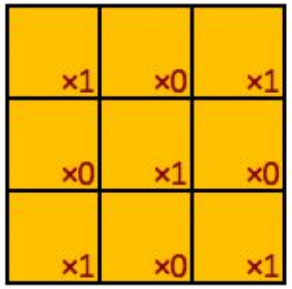
Padding is used to preserve the original dimensions of the input

Zeros are added to outside of the input

Number of zero layers depend upon the size of the kernel

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

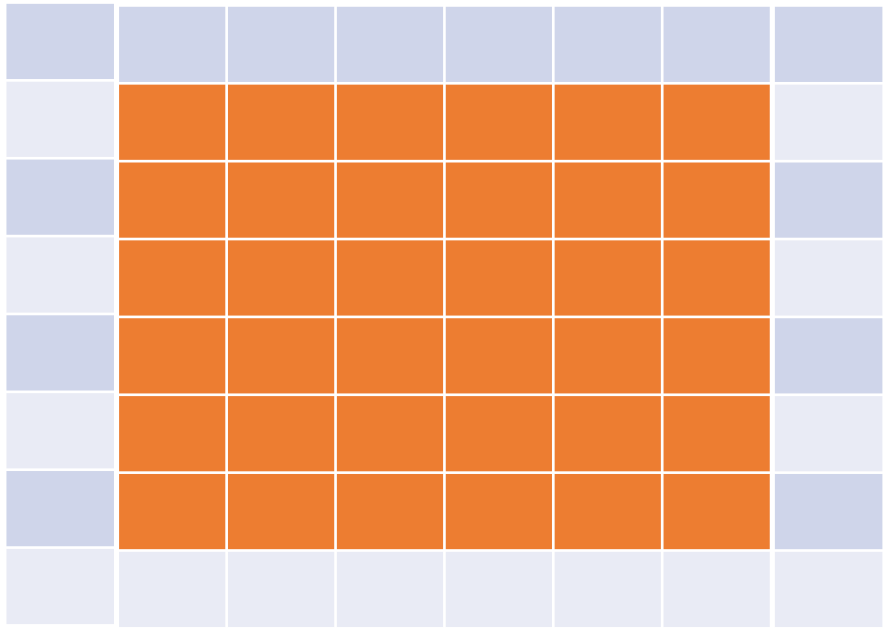
5X5 (with padding)



2	2	3	1	1
1	4	3	4	1
2	2	4	3	3
1	2	3	4	1
1	2	3	1	1

5X5

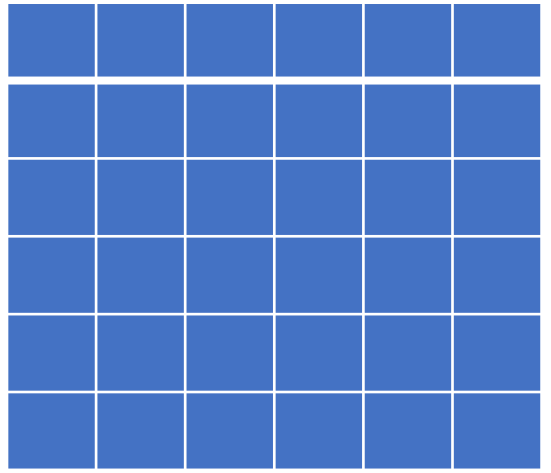
Padding



*

f 3X3

=



nXn 6X6 to 8X8 Padding=1

(n-f+1)X(n-f+1) to (n+2p-f+1)X(n+2p-f+1)

Stride

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

6X6 Matrix

Stride=s (3 Here)

$\text{Floor}(\frac{n+2p-f}{s}+1)\ X\text{Floor}(\frac{n+2p-f}{s}+1)$

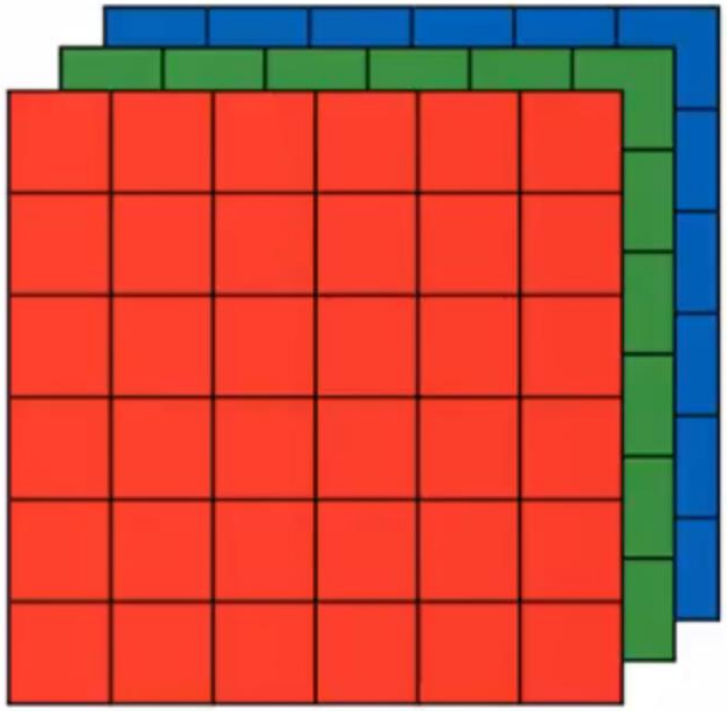
1	0	-1
1	0	-1
1	0	-1

3X3 Filter

-5	8
-3	-16

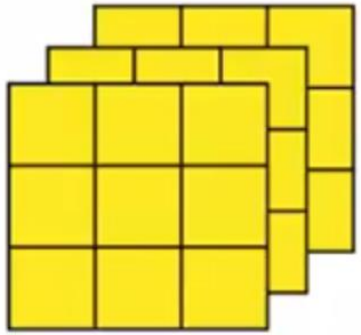
2X2

Channels



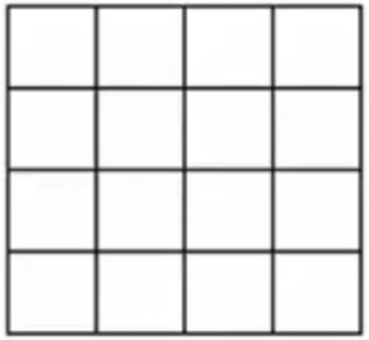
6X6 Matrix

*



3X3 Filter

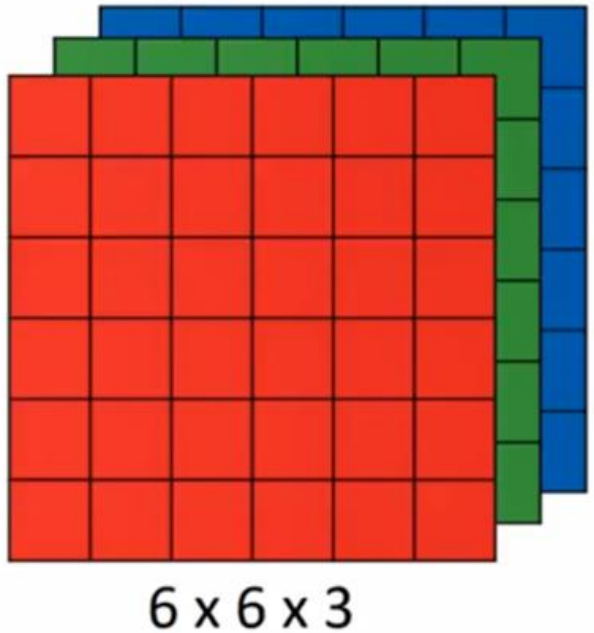
=



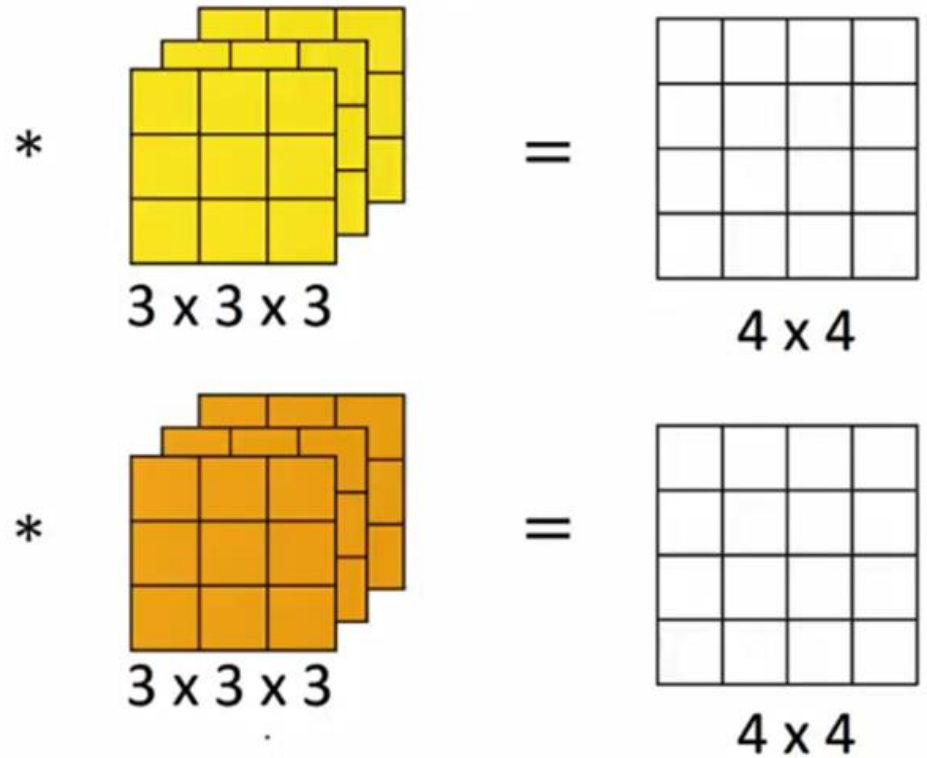
4 x 4

Padding =0 Stride=1

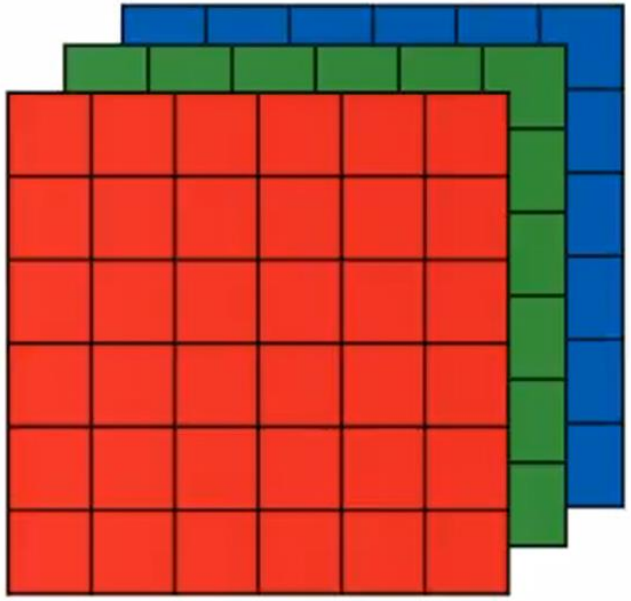
$nXnXn_c$



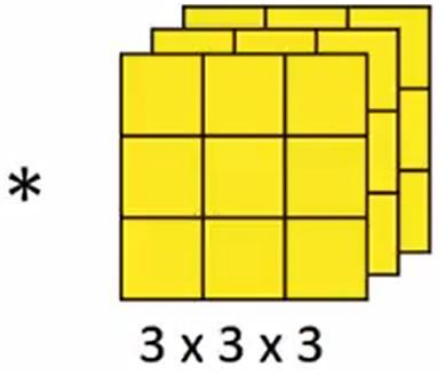
$fXfXn_c$

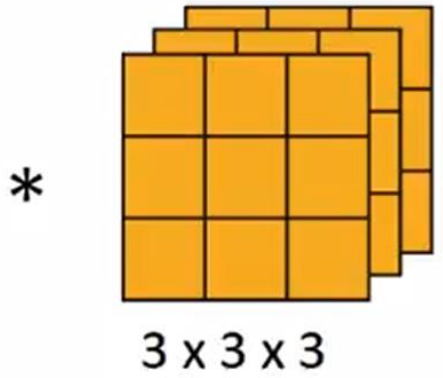


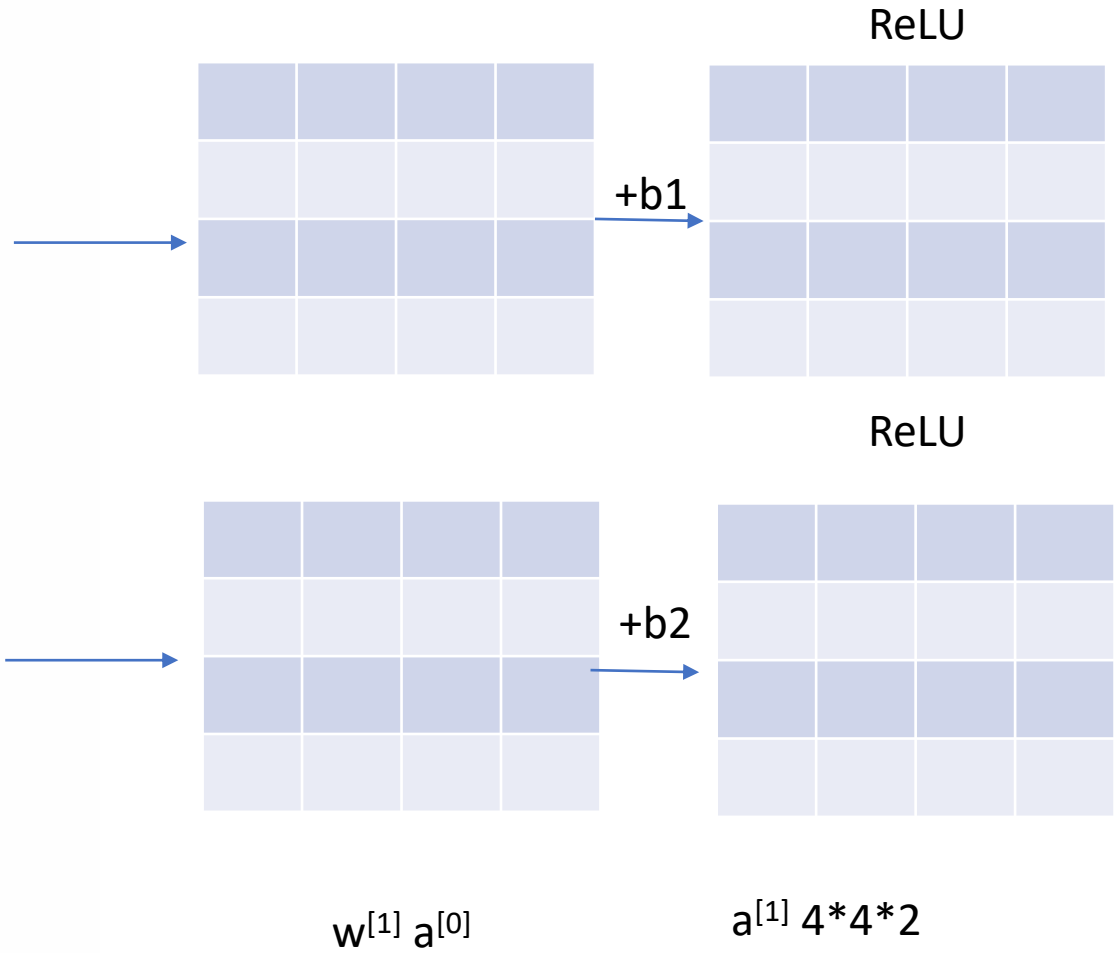
$n-f+1Xn-f+1Xn_{c'}$ c'=no of filters



$a^{[0]}$







$w^{[1]}$ $a^{[0]}$

$a^{[1]}$ 4*4*2

$w^{[1]}$ 2 filters means two units here

Pooling

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

Max Pooling : One example of pooling layer

8	9
9	4

f=2

s=2 4X4 converted to 2X2

Function of Pooling is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network.

Pooling

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

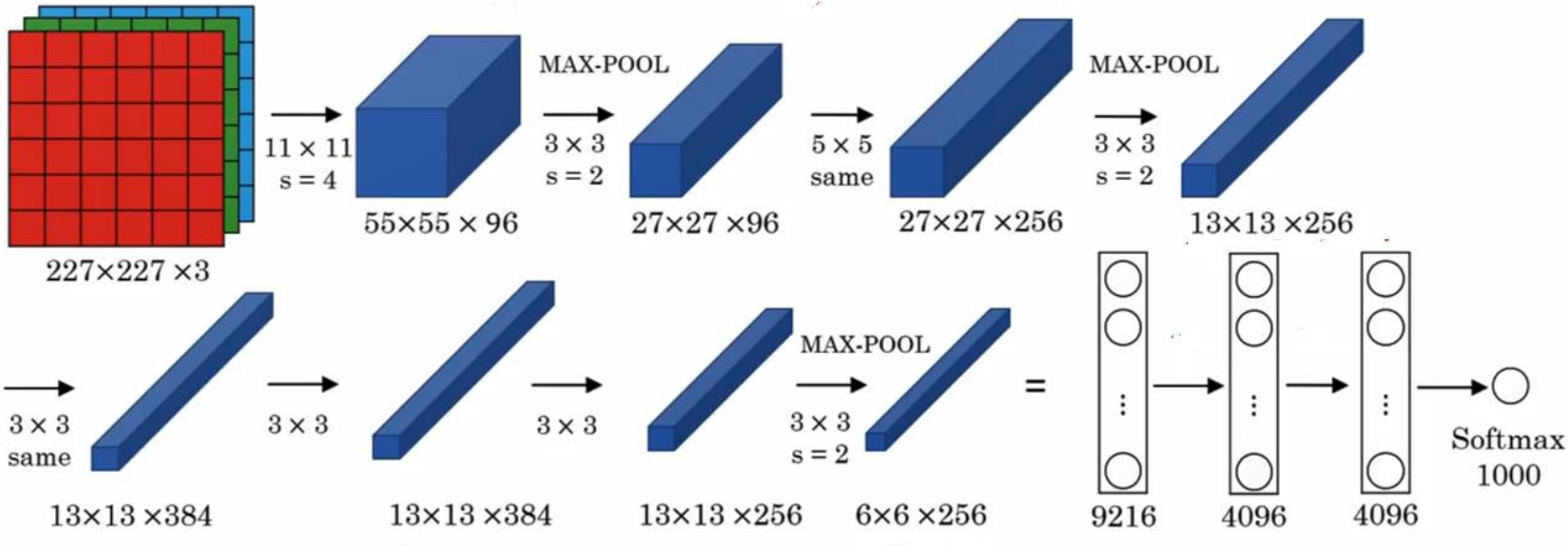
3.5	6.25
4.5	2.5

Average Pooling : Another example of pooling layer

f=2

s=2 4X4 converted to 2X2

AlexNet


227×227 ×3
11 × 11
s = 4
55×55 × 96
MAX-POOL
3 × 3
s = 2
27×27 ×96
5 × 5
same
27×27 ×256
MAX-POOL
3 × 3
s = 2
13×13 ×256
3 × 3
same
13×13 ×384
3 × 3
13×13 ×384
3 × 3
13×13 ×256
MAX-POOL
3 × 3
s = 2
6×6 ×256
=
9216
4096
4096
Softmax
1000

Famous CNN Models

- LeNet - 1990
- AlexNet - 2012
- ZFNet - 2013
- GoogLeNet - 2014
- VGGNet - 2014
- ResNet - 2015
- Inception v3 - 2016
- MobileNet – 2017
- Squeezenet - 2017

Open Source Implementations

-Developers has spent months in developing some good open source networks

-Many of them have access to high-speed GPUs and have the capacity to do lot of experimentation for months

-It is always good to start with a known open source implementation

-Search <name of the network> Github

-you will get few links pointing you to Github profiles of individuals who have put these network models along with the learned weights online

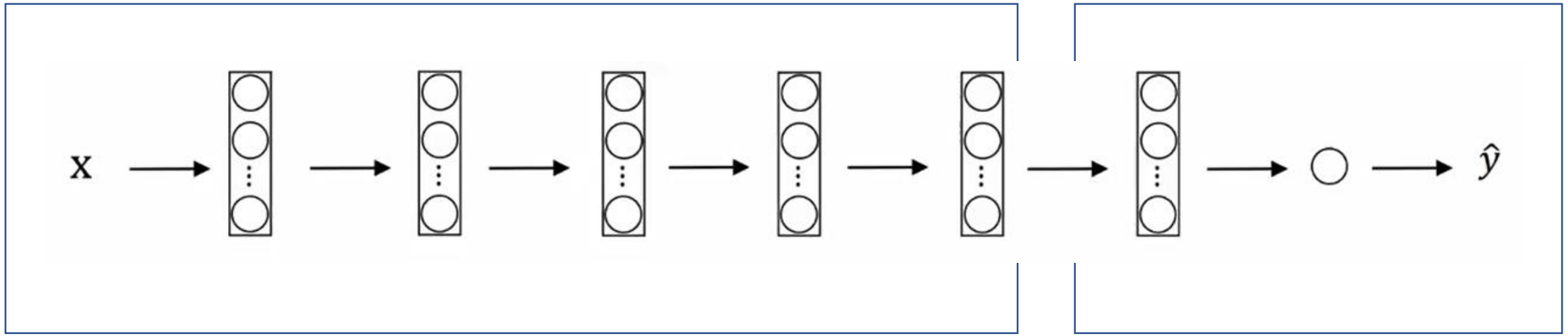
-Download or clone these network depending upon whether you are working on the your machine or remotely on a notebook

-Use this trained model fully or partially depending upon your requirements

Transfer Learning

- Sharing the Knowledge gained solving one problem and applying to a different but related problem
- Transfer Learning is next popular driver of deep learning after supervised learning
- The feature spaces of the source and target domain are different, e.g. the documents are written in two different languages.
- The marginal probability distributions of source and target domain are different, e.g. the documents discuss different topics.
- Simulation training is becoming a hot area within the sphere of Deep Learning. Few labs have also started using AR/VR Technologies to be integrated for making advance learning models for some of the critical problems area

Transfer Learning – Small Data



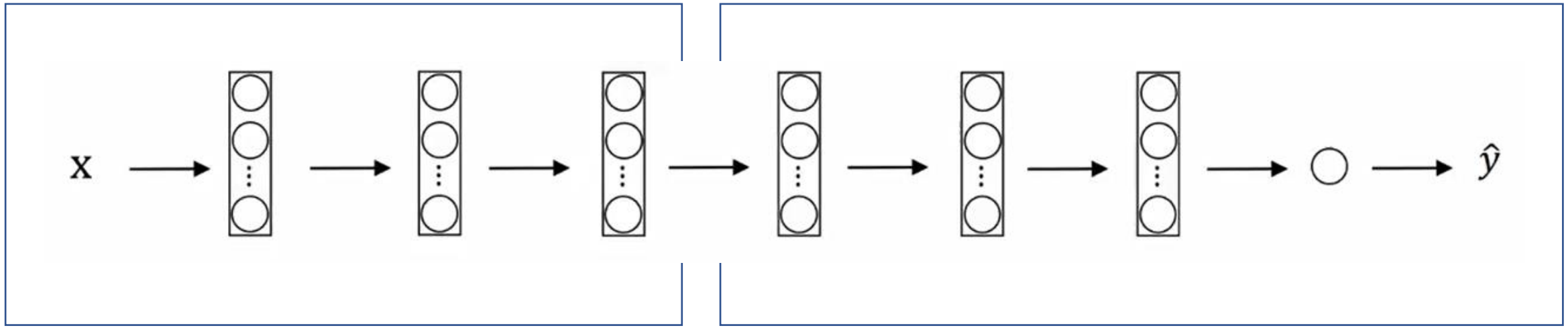
*Finding a Bengal Cat or British Cat where you have small dataset of these categories of cats

*You can download a cat classifier and train last few layers on your data specifically

*In Popular Deep Learning platforms, now you have good support for transfer learning

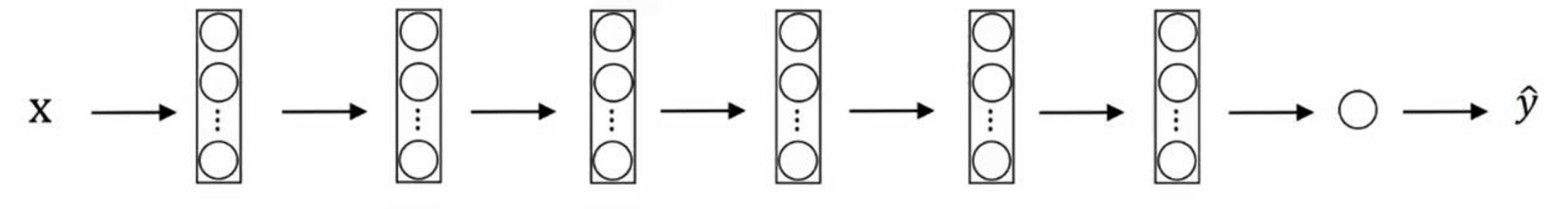
* Functions like Trainable_Parameters and Freezing specific layers are available

Transfer Learning-Mid Size Data



In this category we use initial set of layers from the open source model and use the trained weight values. For the remaining layers there are two ways to handle: a) either we can train the layers from start or b) we can start the weight optimization from the existing set of weights that we have received from the open source model

Transfer Learning- Enough Data



In scenarios, where we have enough data to train our new model, open source model may still be useful. We can use the pre-trained weights of the model from the same domain and consider that as a starting point for our model. It may be much easier and faster to adapt these model for new set of data that we want to train upon.