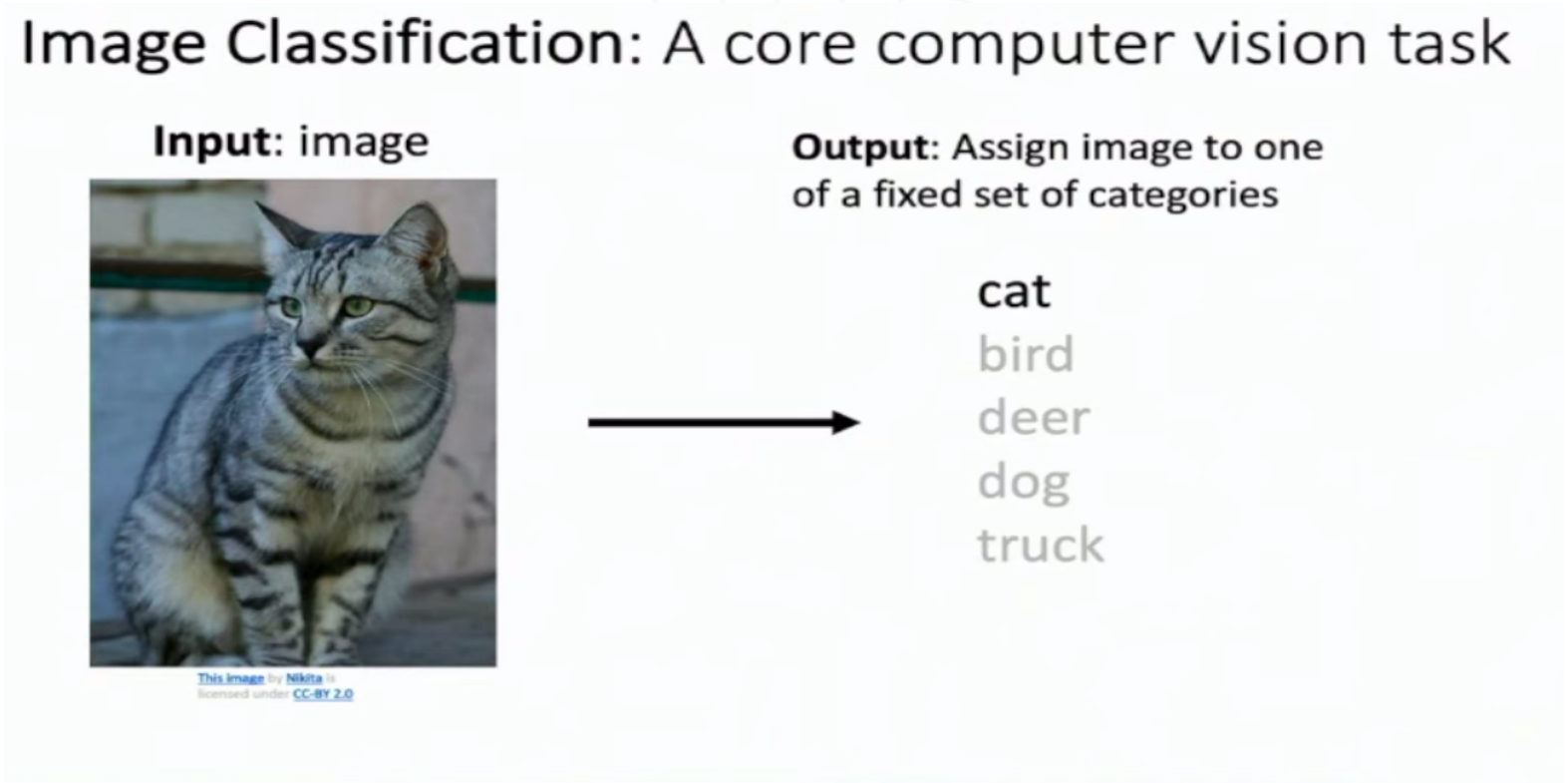
# Convolutional Neural Networks

**(3-0-0-3)**

## Introduction to CNN

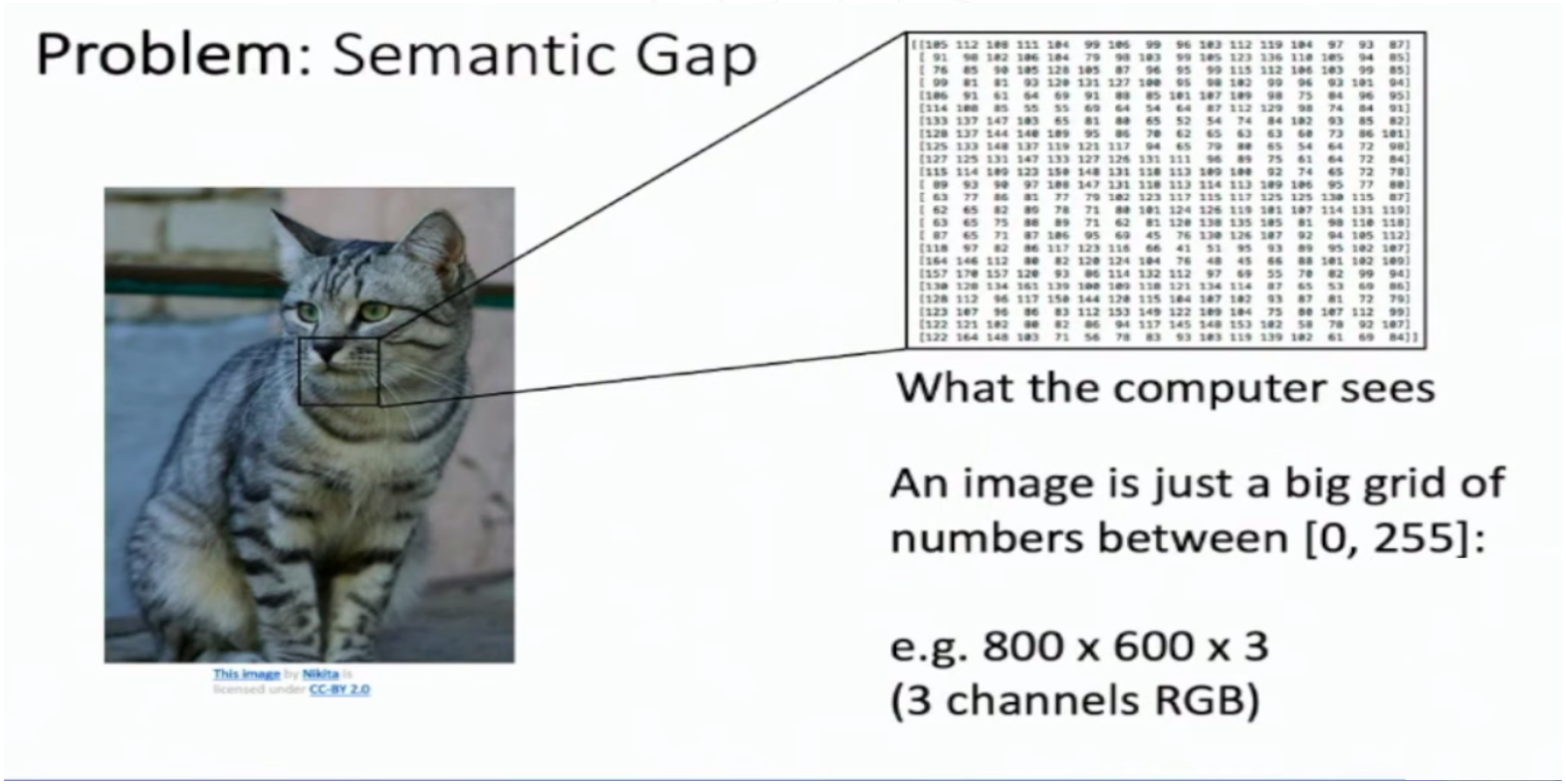
January 26, 2023

# Image classification



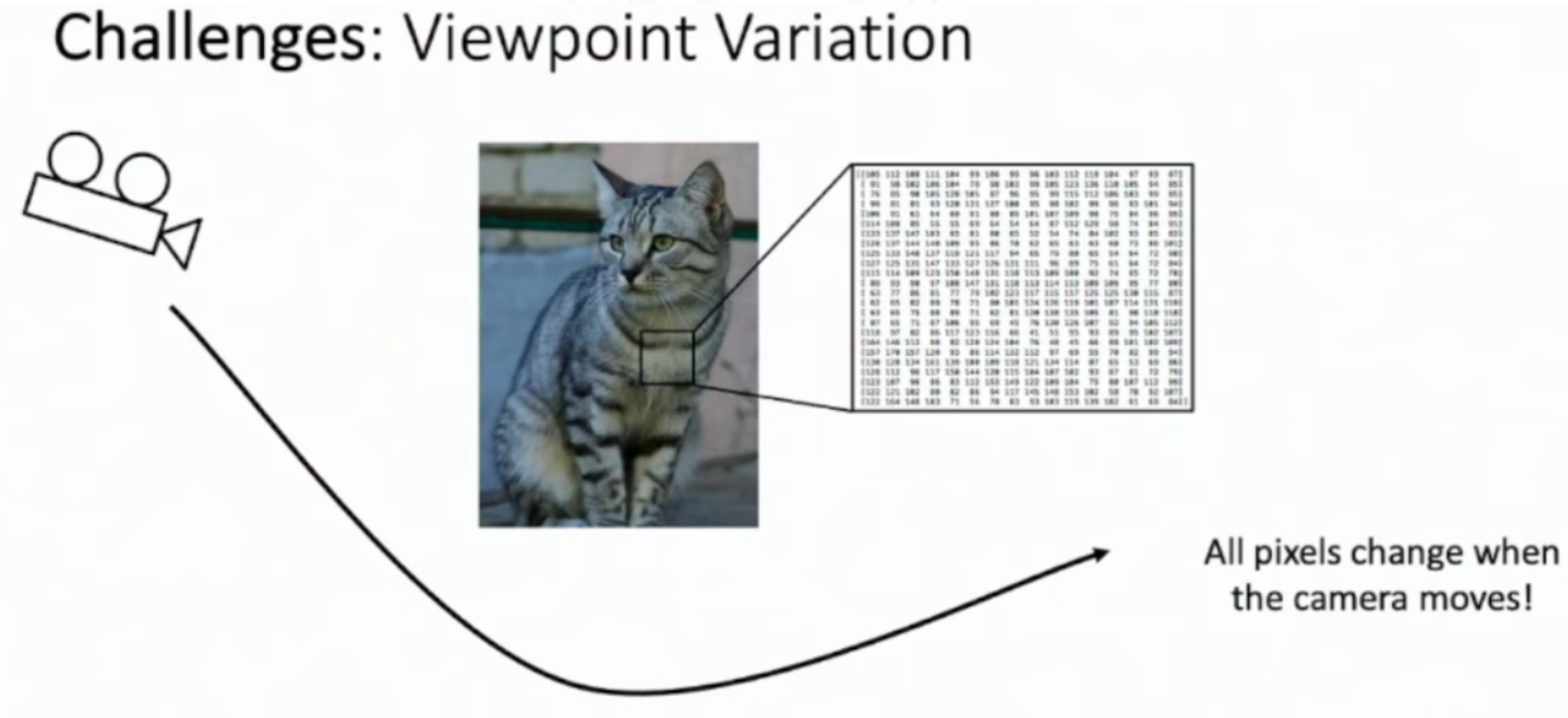

For human, its a trivial task!

# Image classification




Computer sees only numbers between 0 to 255 for gray scale and 0 to 255 with three different channels for RGB images

# Image classification

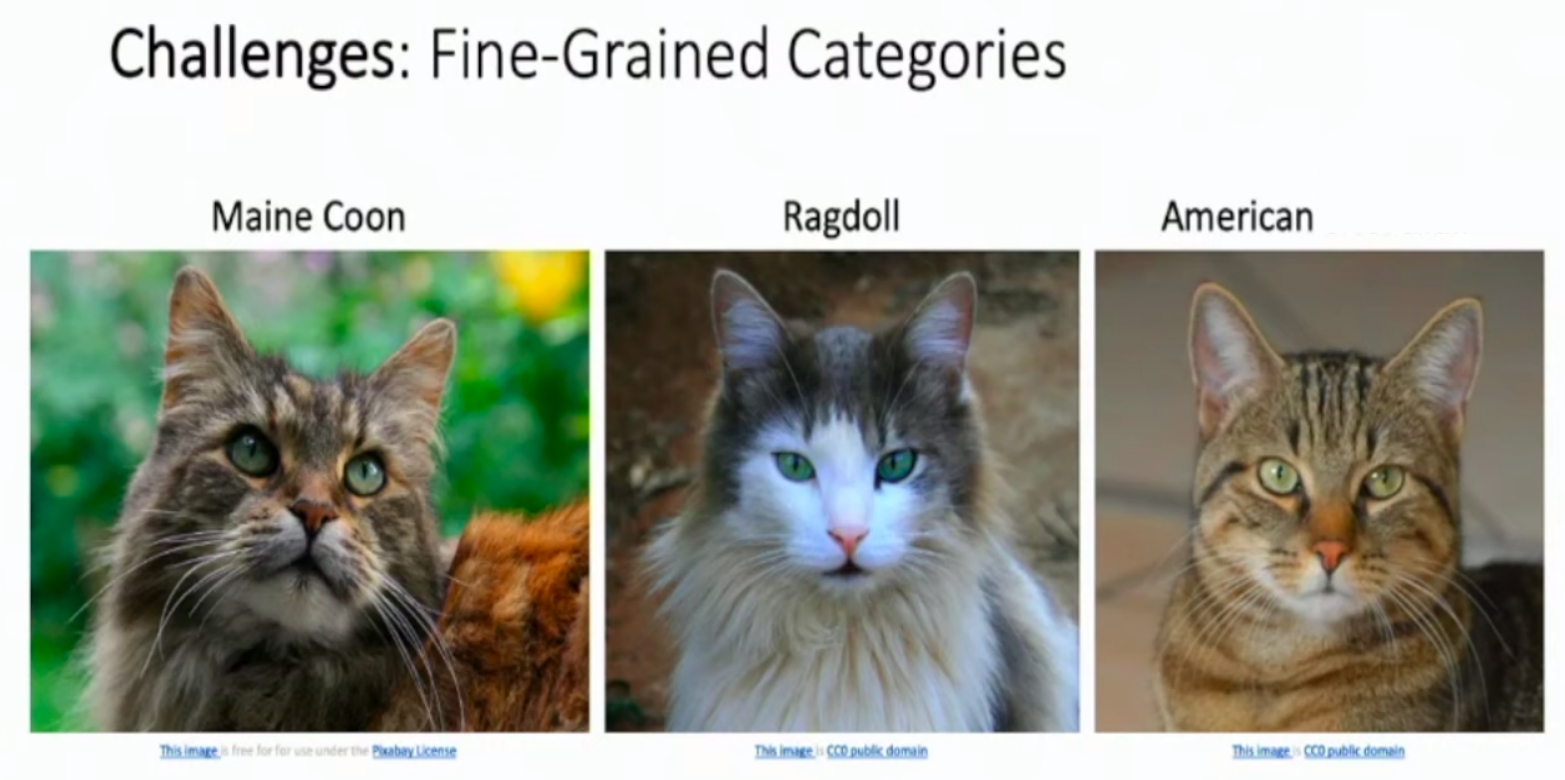



Deep learning course, Justin Johnson, Univeristy of Michigan

# Image classification

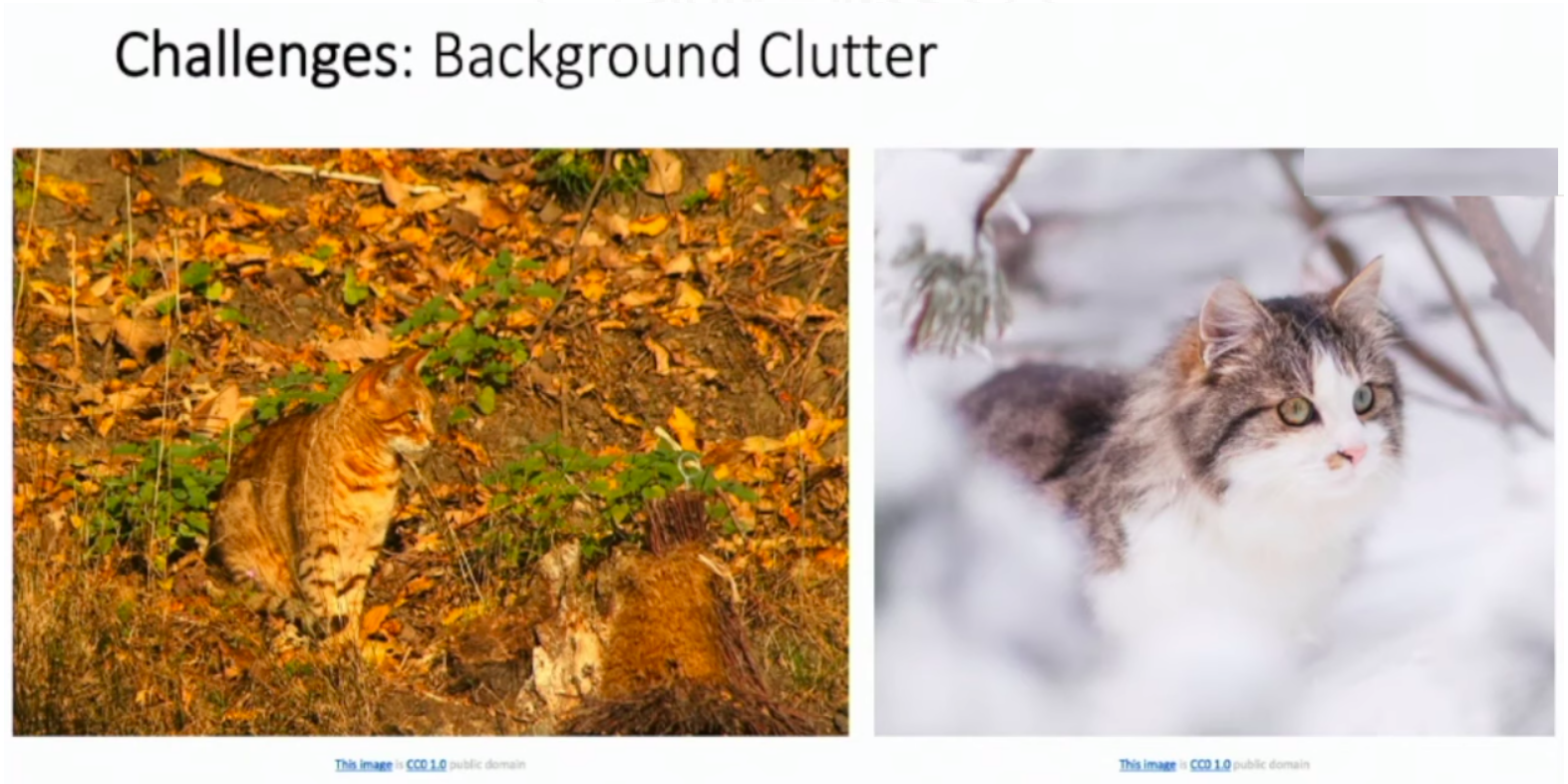

Deep learning course, Justin Johnson, Univeristy of Michigan

# Image classification



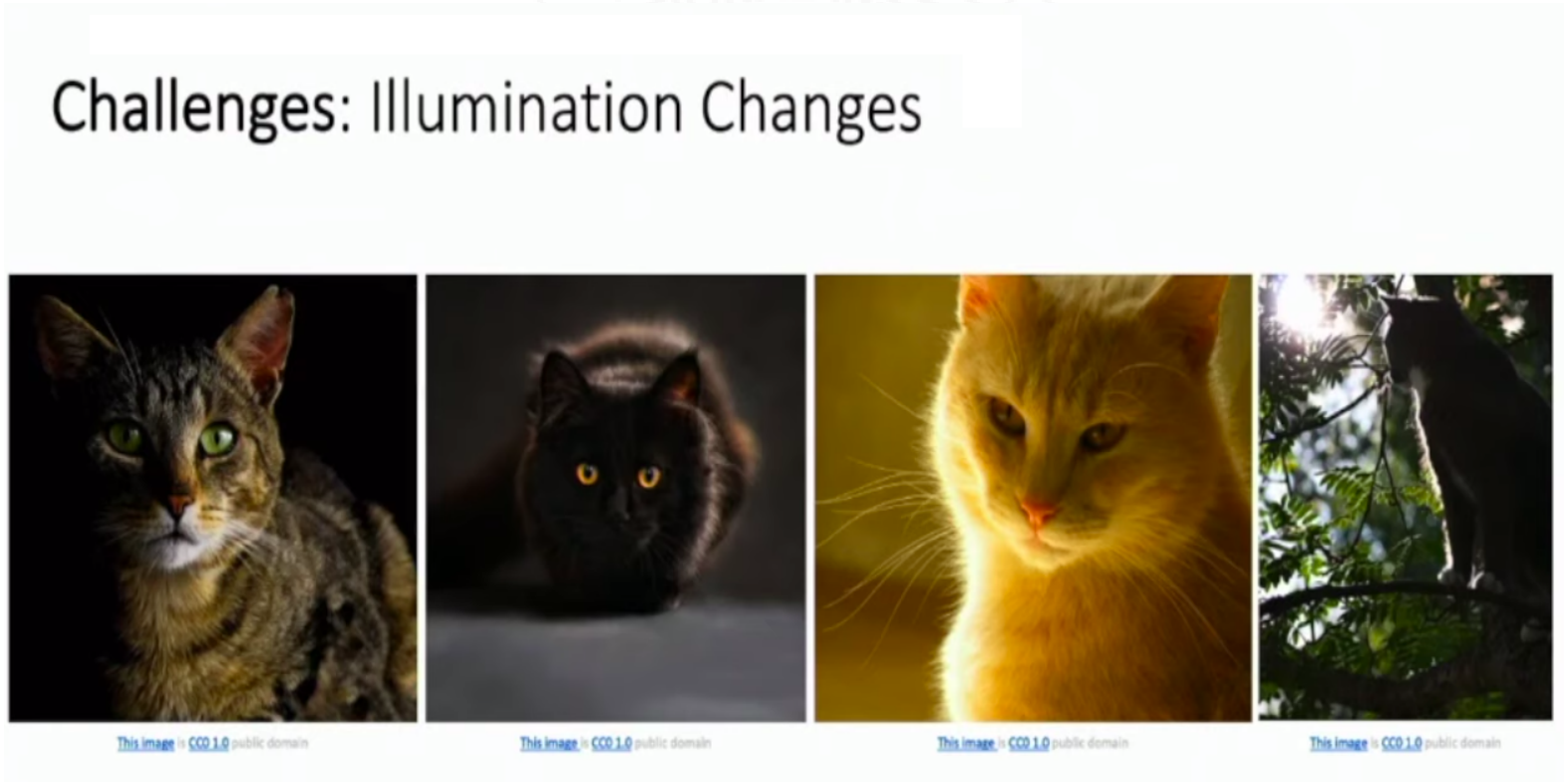

Deep learning course, Justin Johnson, Univeristy of Michigan

# Image classification




Deep learning course, Justin Johnson, Univeristy of Michigan

# Image classification

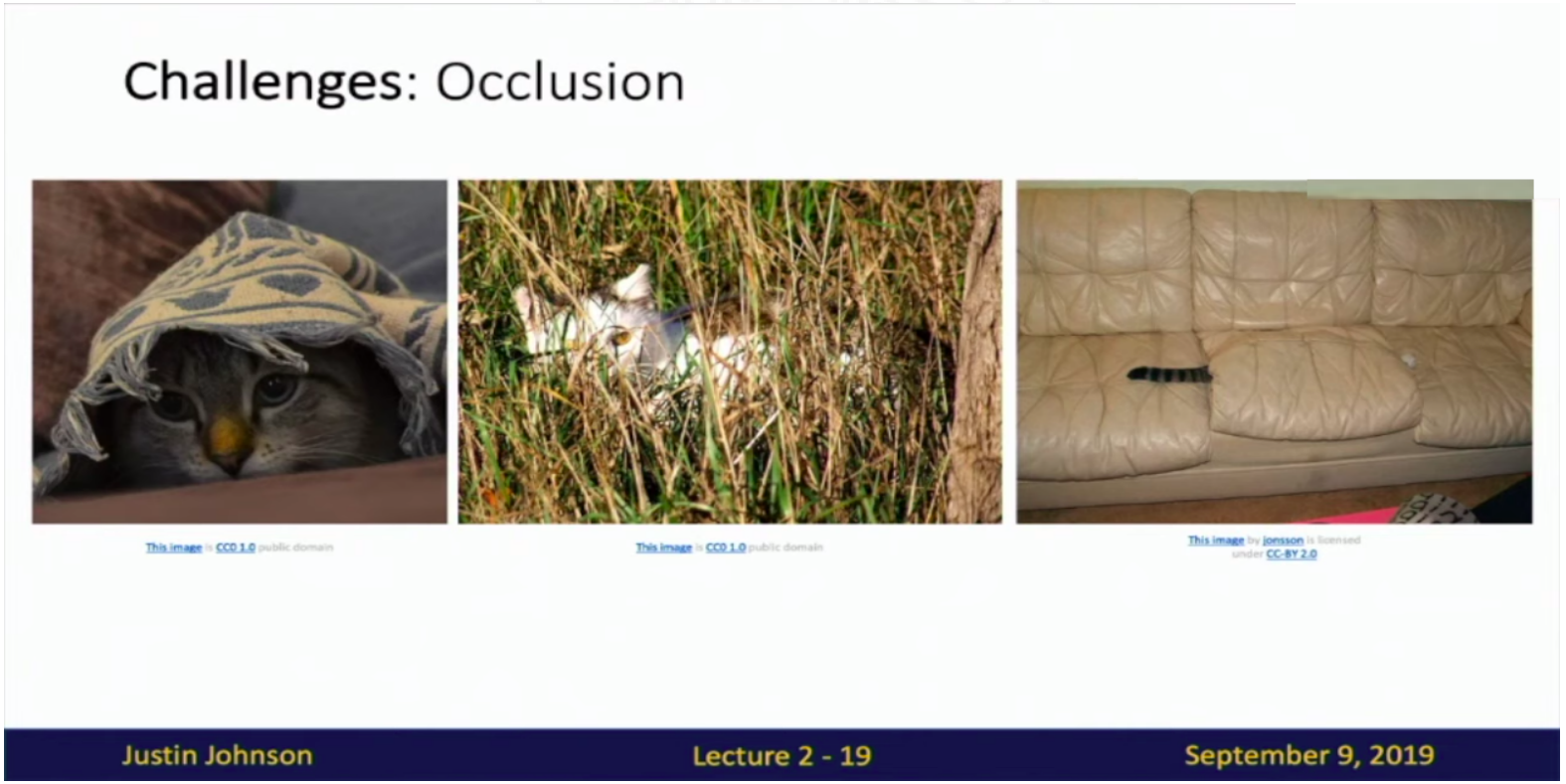



Deep learning course, Justin Johnson, Univeristy of Michigan

# Image classification


Deep learning course, Justin Johnson, Univeristy of Michigan

# Image classification




Deep learning course, Justin Johnson, Univeristy of Michigan

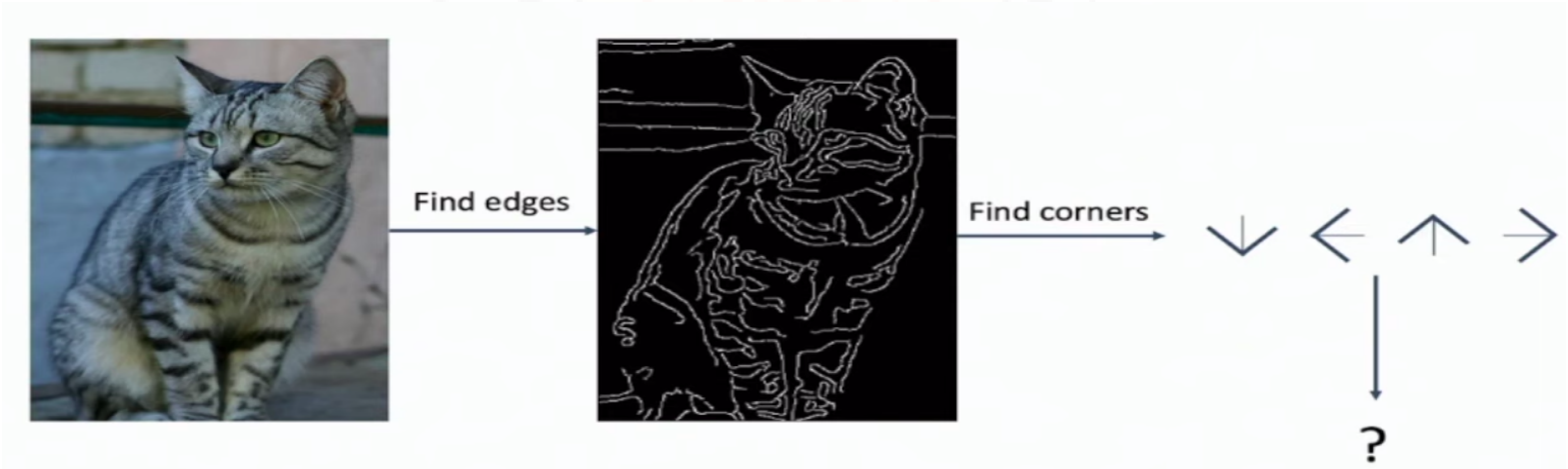
# Image classification

- Image classification is powerful building block in many computer vision problems
- One cannot hard-code the algorithm with pixles by making indefinite number of rules

Deep learning course, Justin Johnson, Univeristy of Michigan

# Image classification

- Image classification is powerful building block in many computer vision problems
- One cannot hard-code the algorithm with pixles by making indefinite number of rules
- One can try edge detection and hard code the algorithm based on edges (like ears, nose etc)




Deep learning course, Justin Johnson, Univeristy of Michigan

# Image classification

- Image classification is powerful building block in many computer vision problems
- One cannot hard-code the algorithm with pixles by making indefinite number of rules
- One can try edge detection and hard code the algorithm based on edges (like ears, nose etc)
- Edge detector may fail based on orientation!!

# Image classification

- Image classification is powerful building block in many computer vision problems
- One cannot hard-code the algorithm with pixles by making indefinite number of rules
- One can try edge detection and hard code the algorithm based on edges (like ears, nose etc)
- Edge detector may fail based on orientation!!
- For different problem like identifying car, this algorithm you have developed may not work!

# Image classification

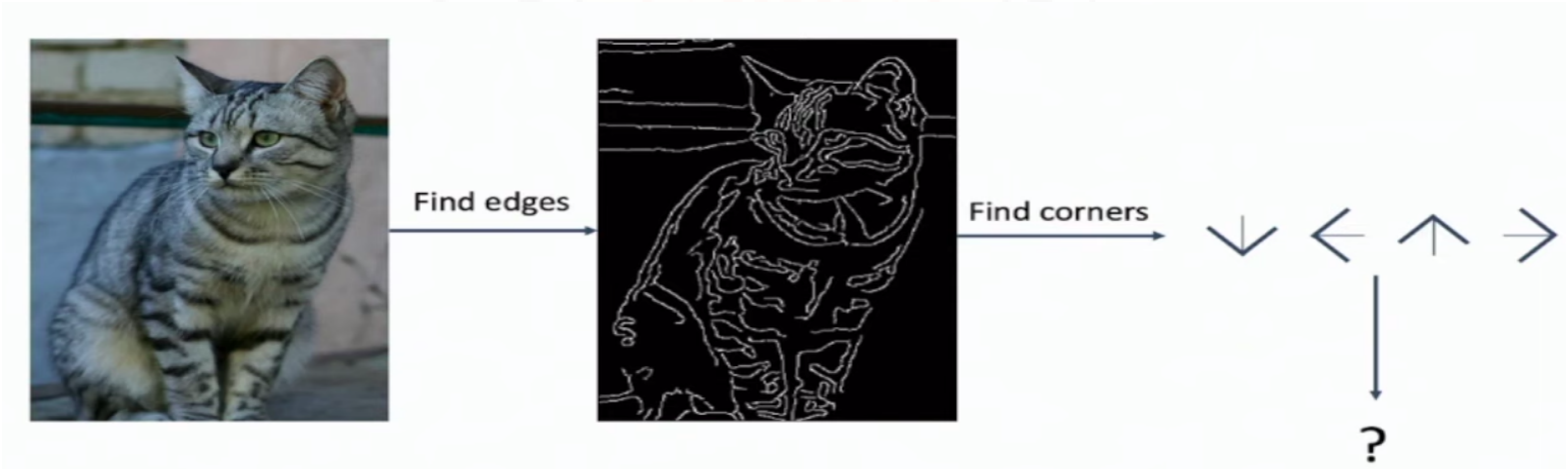
- Image classification is powerful building block in many computer vision problems
- One cannot hard-code the algorithm with pixles by making indefinite number of rules
- One can try edge detection and hard code the algorithm based on edges (like ears, nose etc)
- Edge detector may fail based on orientation!!
- For different problem like identifying car, this algorithm you have developed may not work!

So we want some generalised approach where we dont require human knowledge!!

Deep learning course, Justin Johnson, Univeristy of Michigan

# Image classification

- Image classification is powerful building block in many computer vision problems
- One cannot hard-code the algorithm with pixles by making indefinite number of rules
- One can try edge detection and hard code the algorithm based on edges (like ears, nose etc)




Deep learning course, Justin Johnson, Univeristy of Michigan

# Convolutional Neural Networks

- CNN are specilaised architectures which works well with visual data (i.e) images and videos
- CNN is largely responsible for revolutionalizing 'deep learning' by settiing new benchmarks for many image processing tasks!
- CNN uses many of the working principles of animal visual system [Read more here]
- Rapid inventions in computer vision
  - Image classification
  - Object detection
  - Neural style transfer

# Convolutional Neural Networks

- In case of images, the input to the neural network becomes very large
  Consider an RYB image with array size of 64 x 64

$$64 \text{ x } 64 \text{ x } 3 \Rightarrow 12288$$
$$1000 \text{ x } 1000 \text{ x } 3 \Rightarrow 3 \text{ x } 10^6$$

  The input to the layer (X) is 3 million vlaues

# Convolutional Neural Networks

- In case of images, the input to the neural network becomes very large
  Consider an RYB image with array size of 64 x 64

$$64 \text{ x } 64 \text{ x } 3 \Rightarrow 12288$$
$$1000 \text{ x } 1000 \text{ x } 3 \Rightarrow 3 \text{ x } 10^6$$

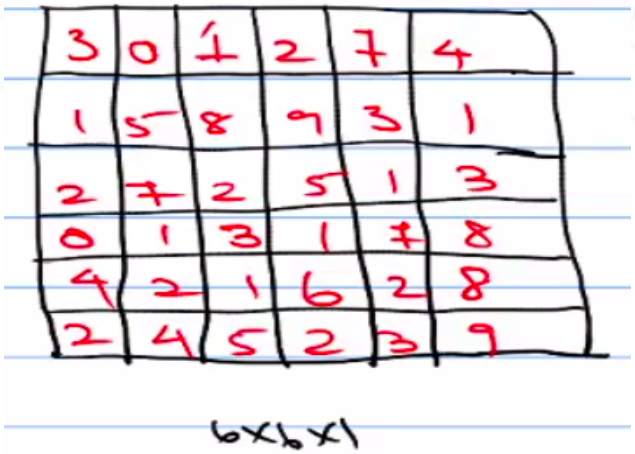
  The input to the layer (X) is 3 million vlaues
- If we have first hidden layer of 1000 units, then the order of trainable parameters

$$W^{[1]} \Rightarrow 1000 \; x \; 3 \; million$$

  This matrix is very large, NN may overfit! The computational requirement is not feasible!

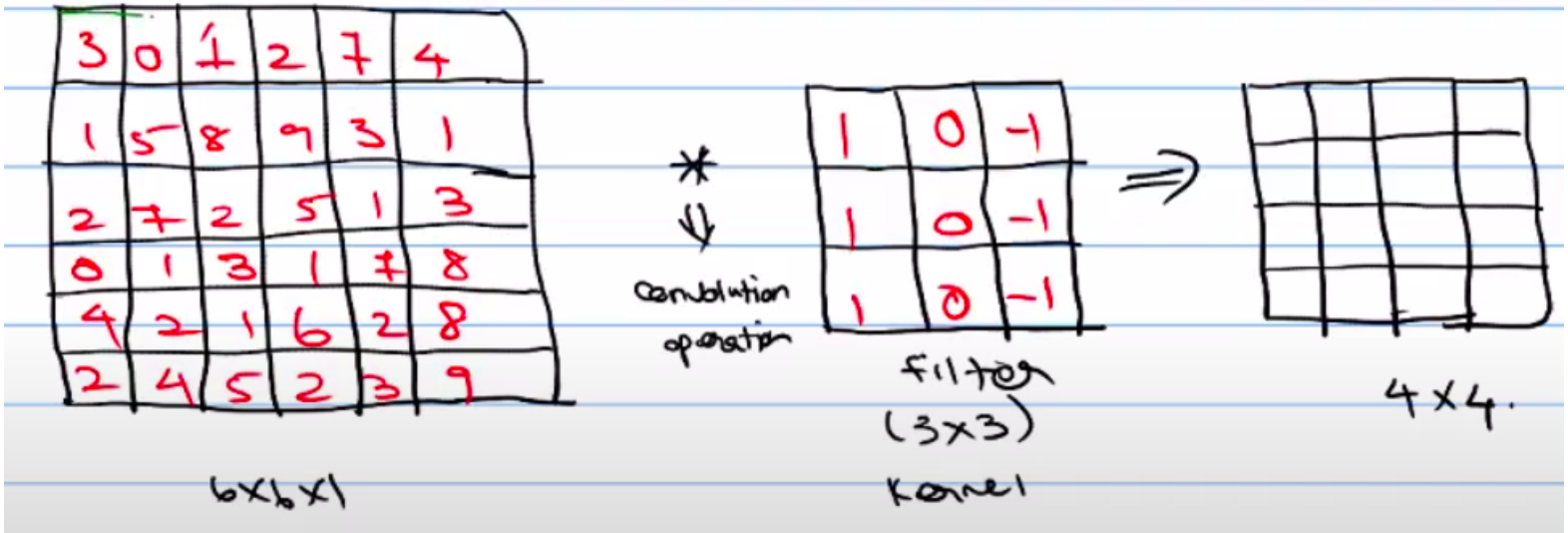
# Convolutional using Edge detection

- Similar to cameras where we add filters to have some special effects, here we use array of no's as filters to extract special effects from the image!
- Consider an gray scale image with array size 6 x 6



# Convolutional using Edge detection

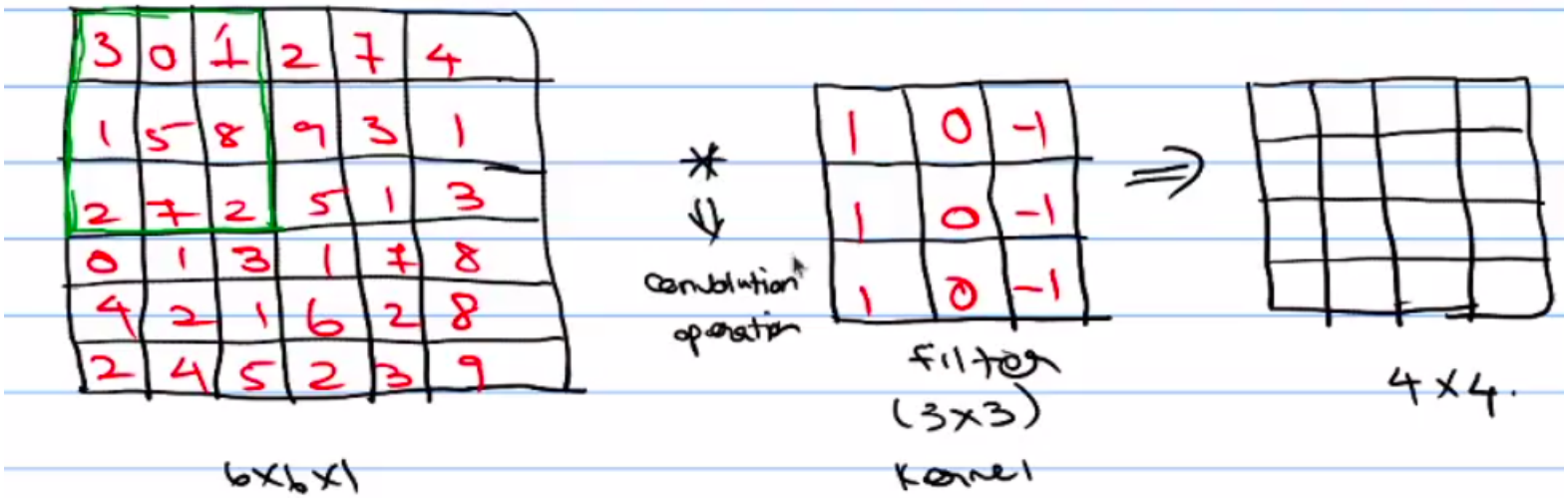
- Similar to cameras where we add filters to have some special effects, here we use array of no's as filters to extract special effects from the image!
- Consider an gray scale image with array size 6 x 6




- Filter is also called as "Kernel"

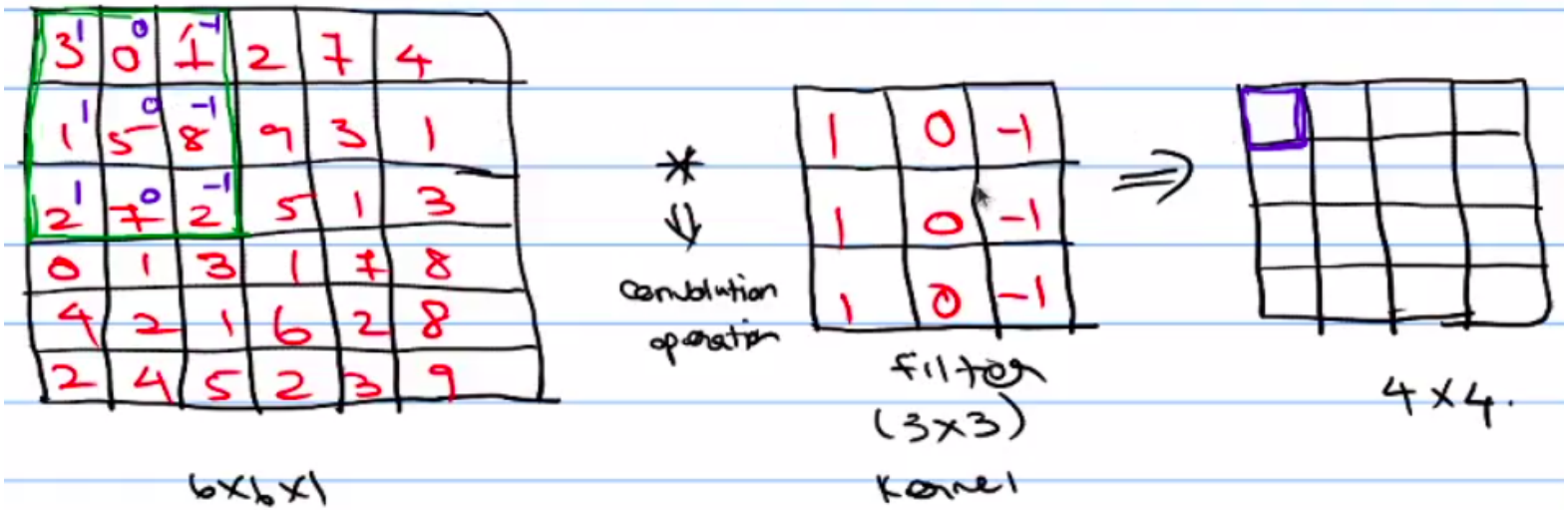
# Convolutional using Edge detection

- Similar to cameras where we add filters to have some special effects, here we use array of no's as filters to extract special effects from the image!
- Consider an gray scale image with array size 6 x 6



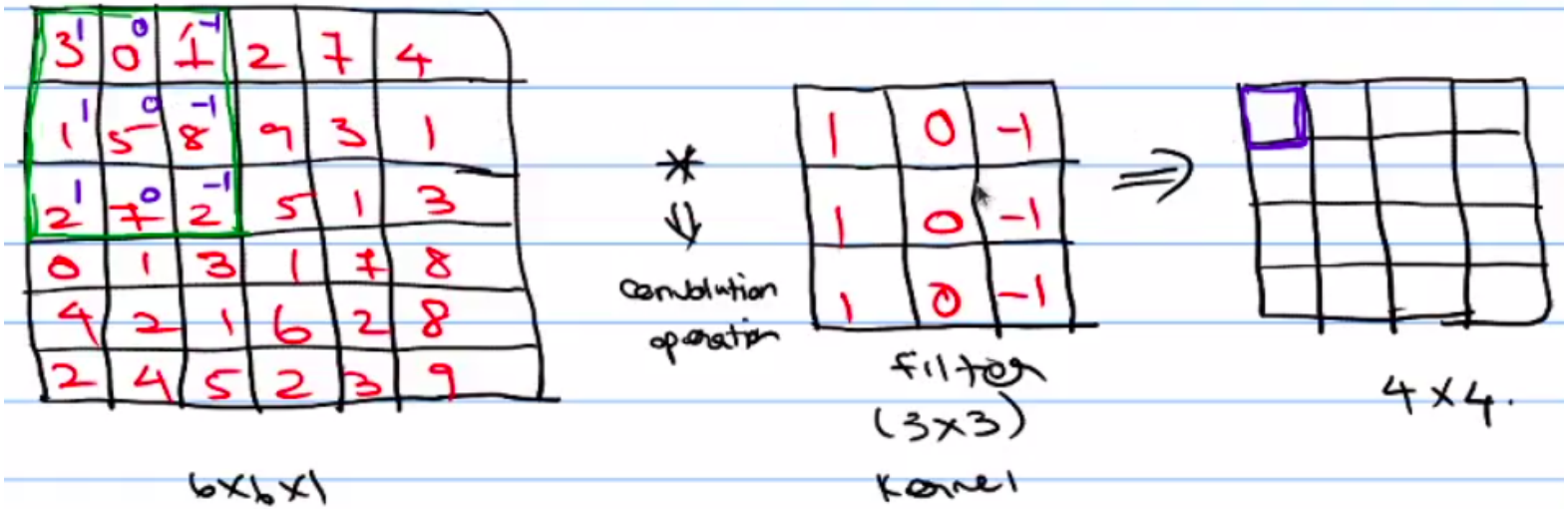
# Convolutional using Edge detection

- Similar to cameras where we add filters to have some special effects, here we use array of no's as filters to extract special effects from the image!
- Consider an gray scale image with array size 6 x 6



# Convolutional using Edge detection

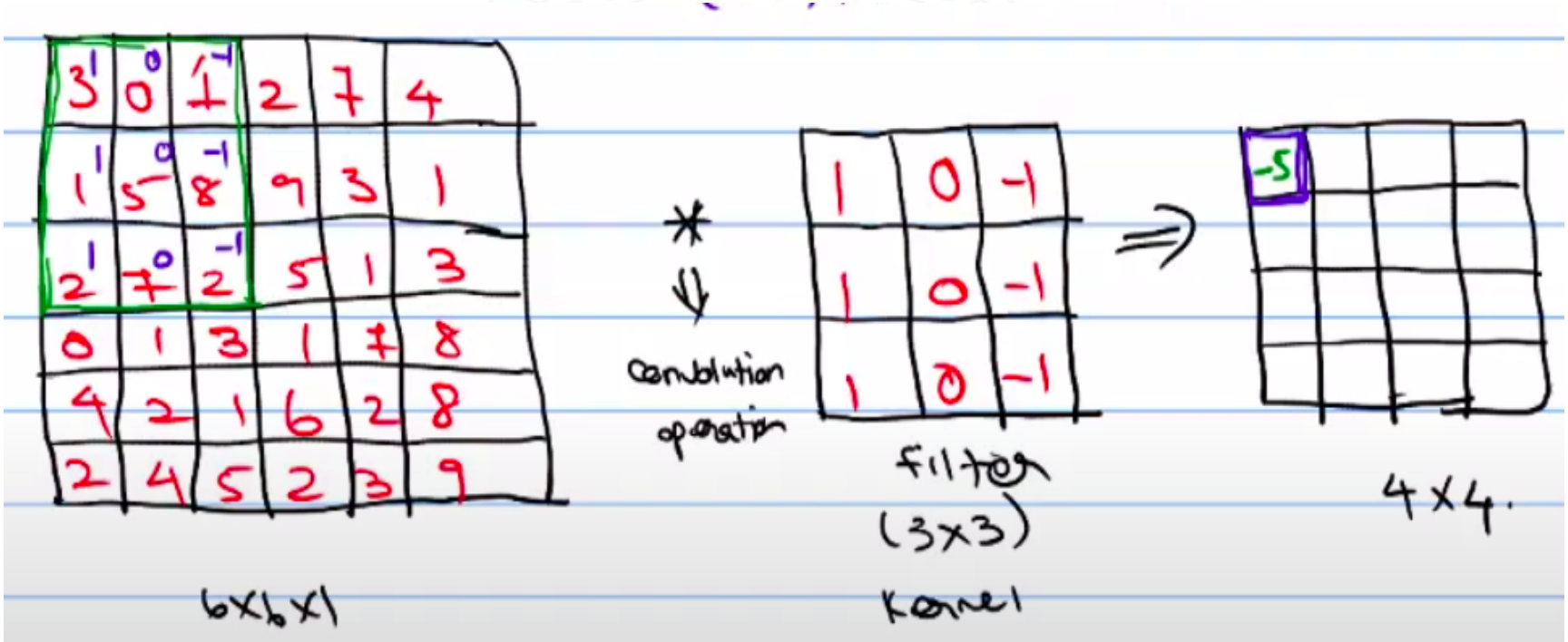
- Similar to cameras where we add filters to have some special effects, here we use array of no's as filters to extract special effects from the image!
- Consider an gray scale image with array size 6 x 6



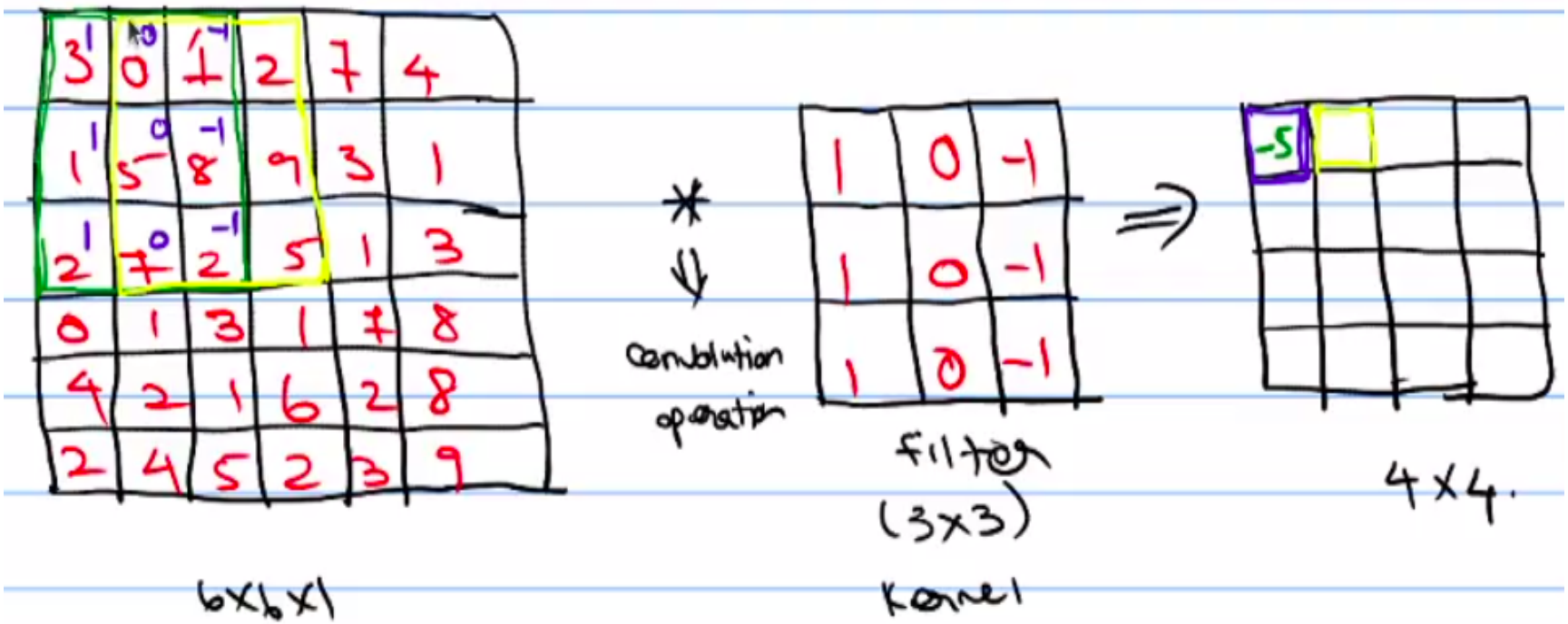

Position 1:

=(3*1)+(0*0)+(1*-1)+(1*1)+(5*0)+(8*-1)+(2*1)+(7*0)+(2*-1)=-5

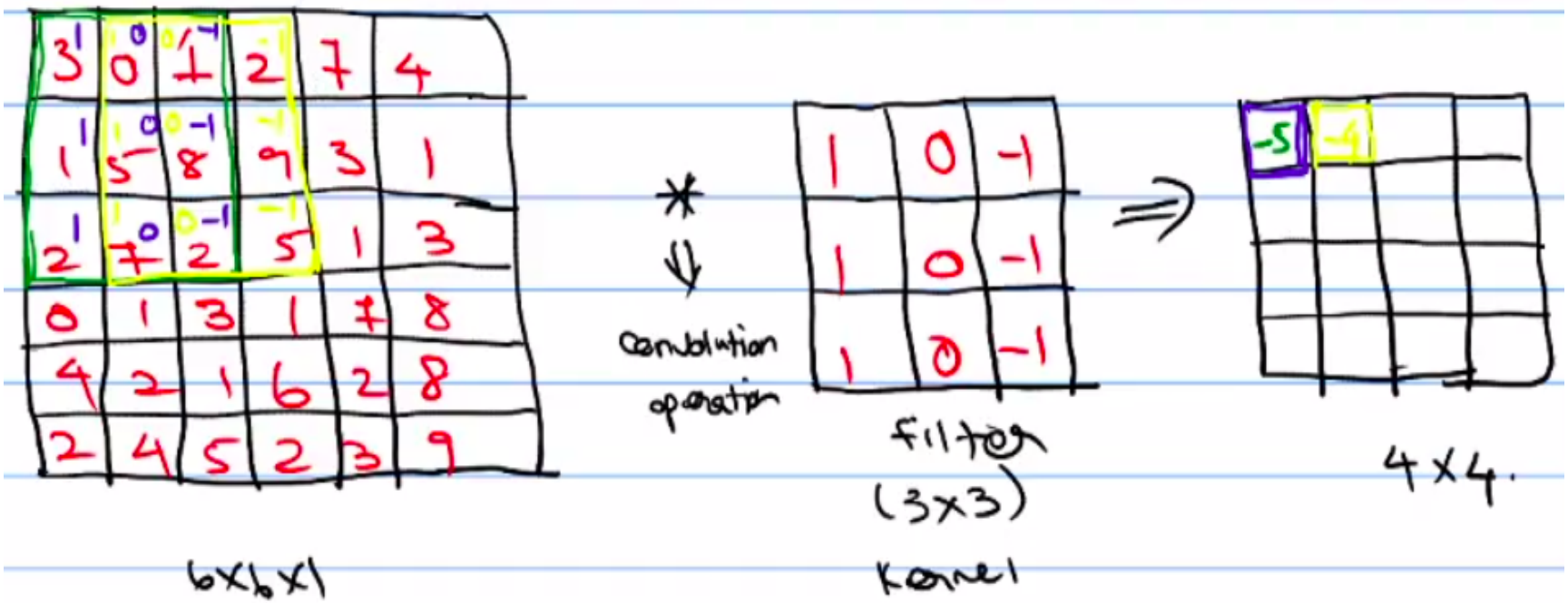
# Convolutional using Edge detection



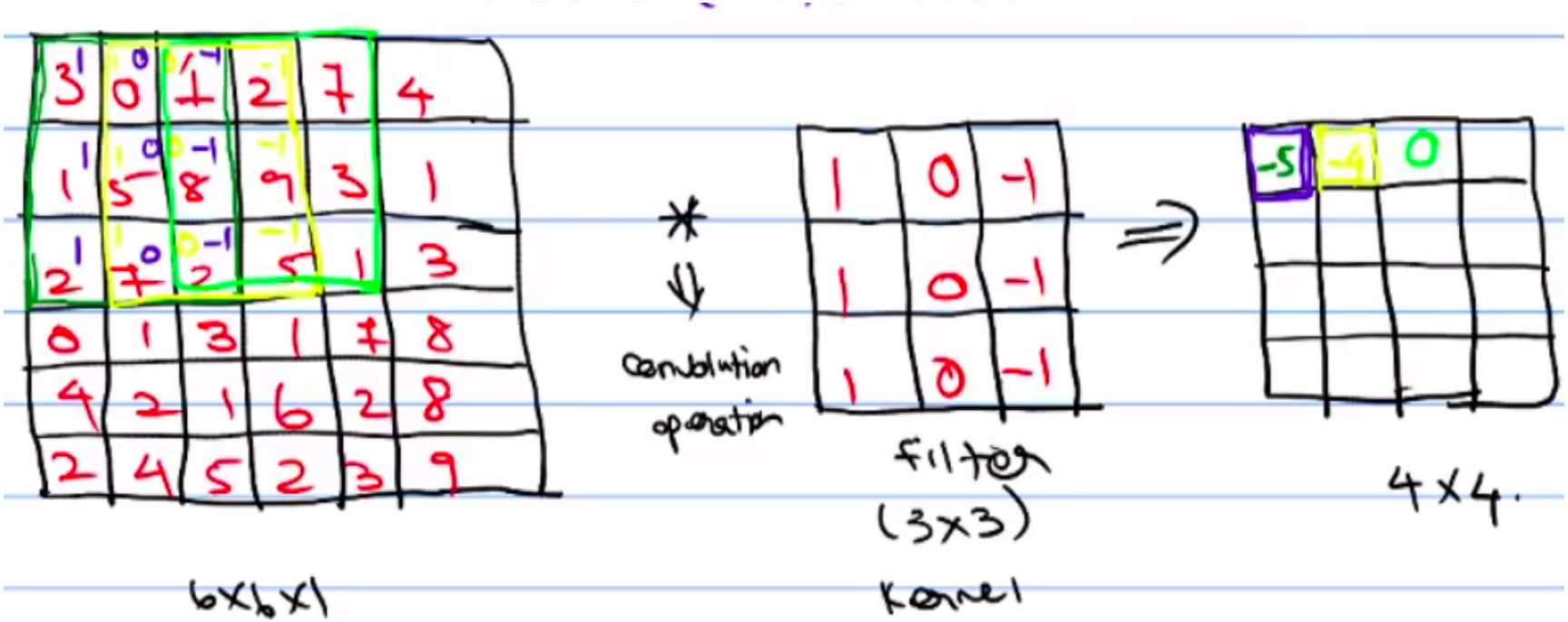
# Convolutional using Edge detection



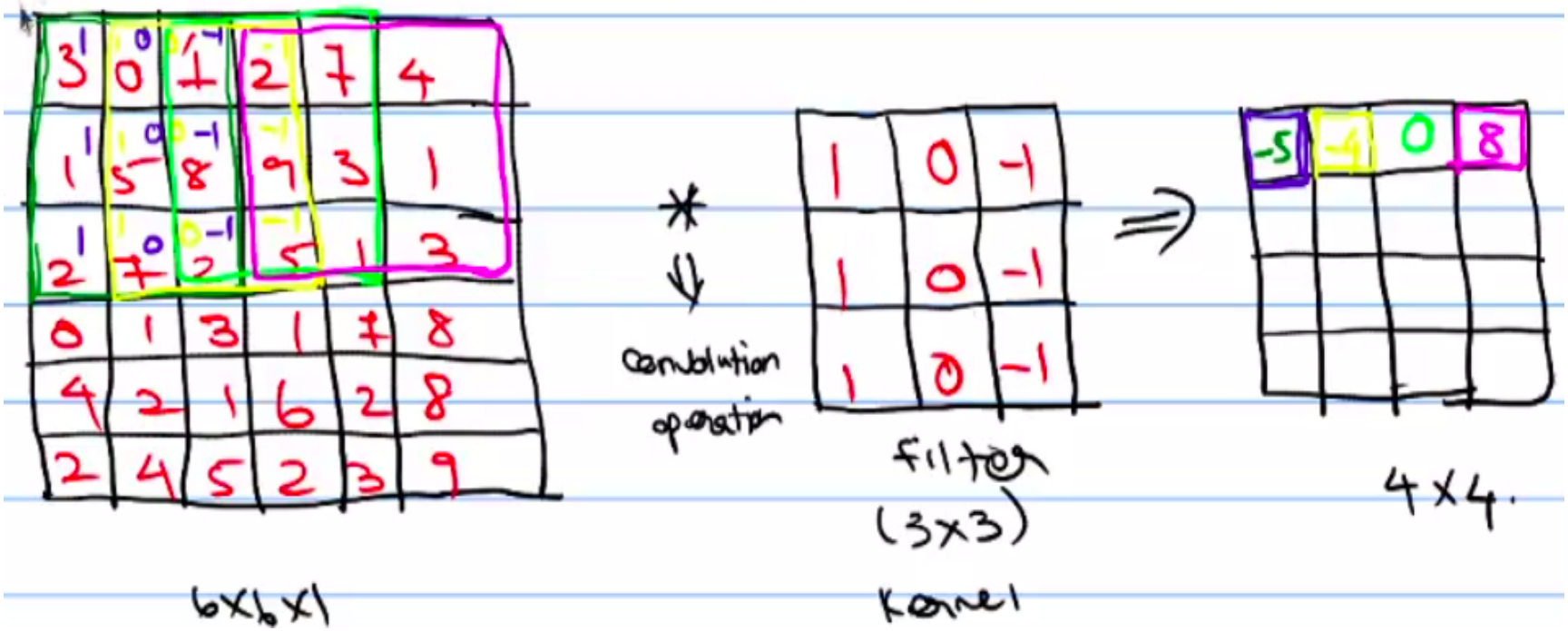
# Convolutional using Edge detection



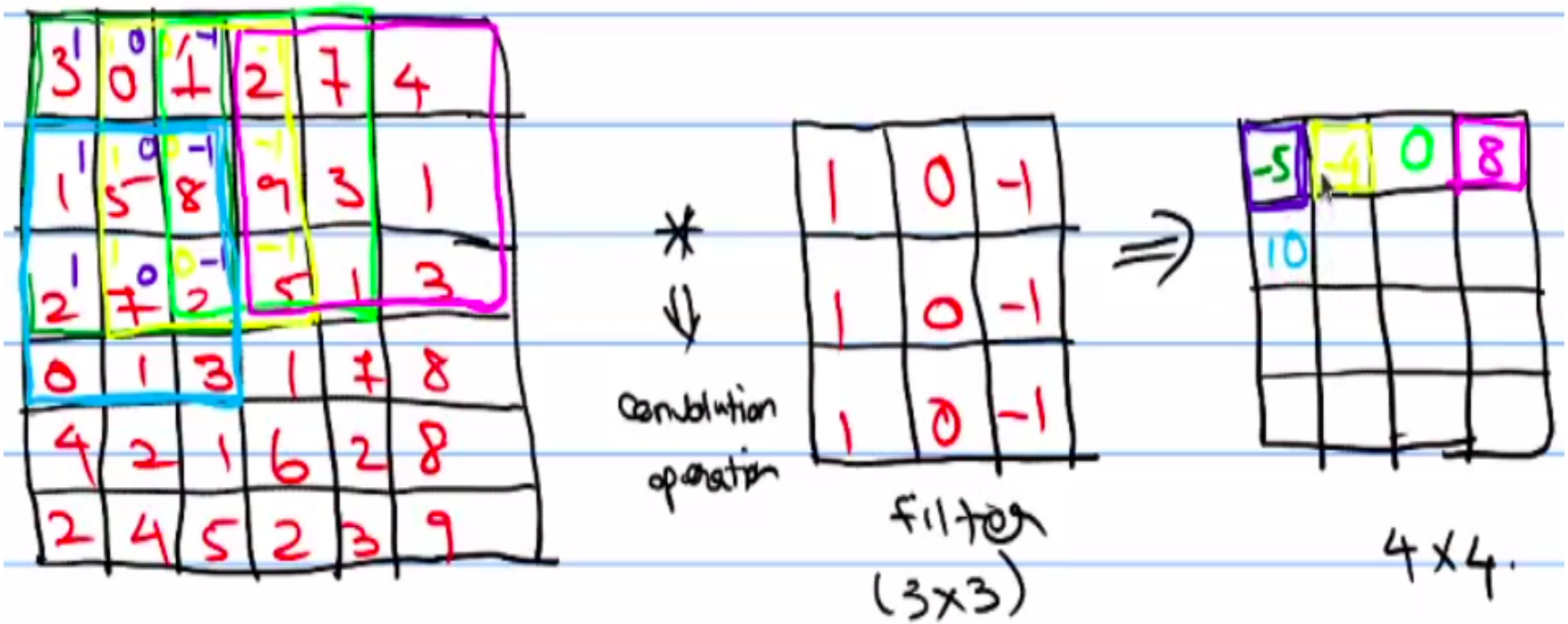
# Convolutional using Edge detection



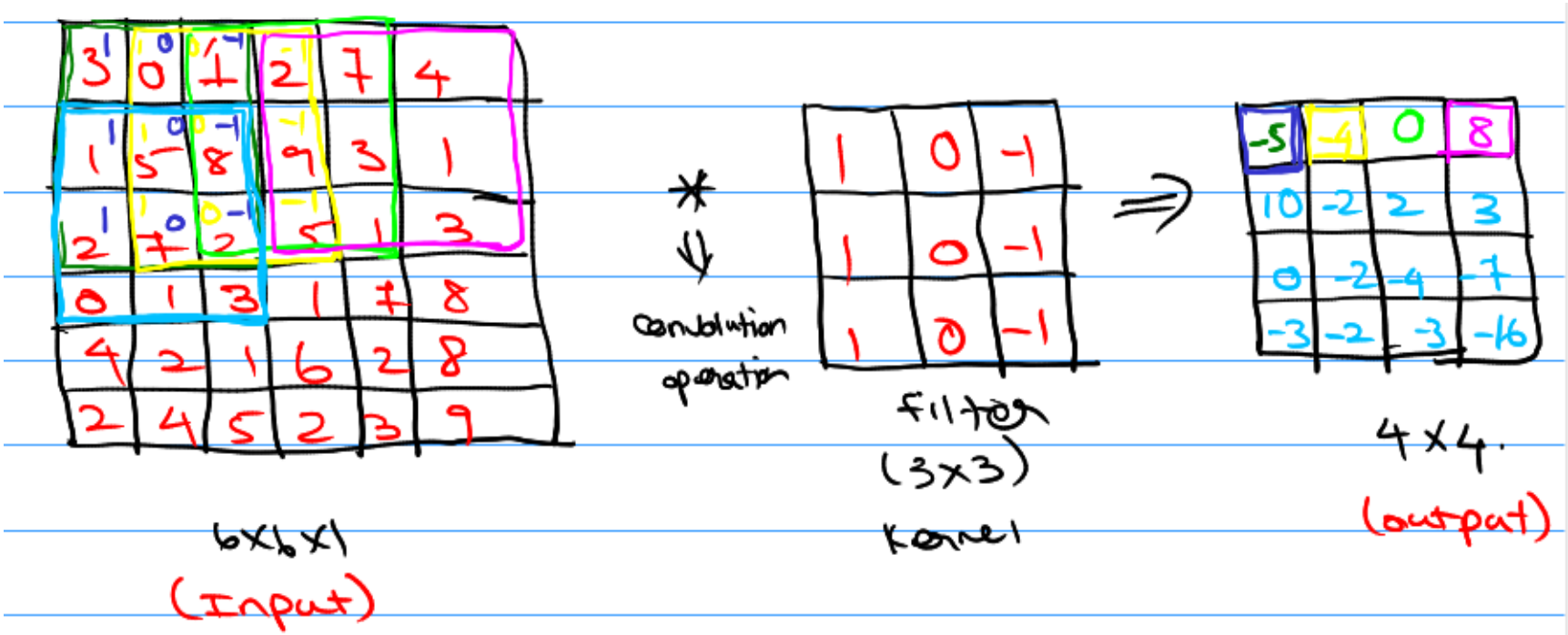
# Convolutional using Edge detection



# Convolutional using Edge detection



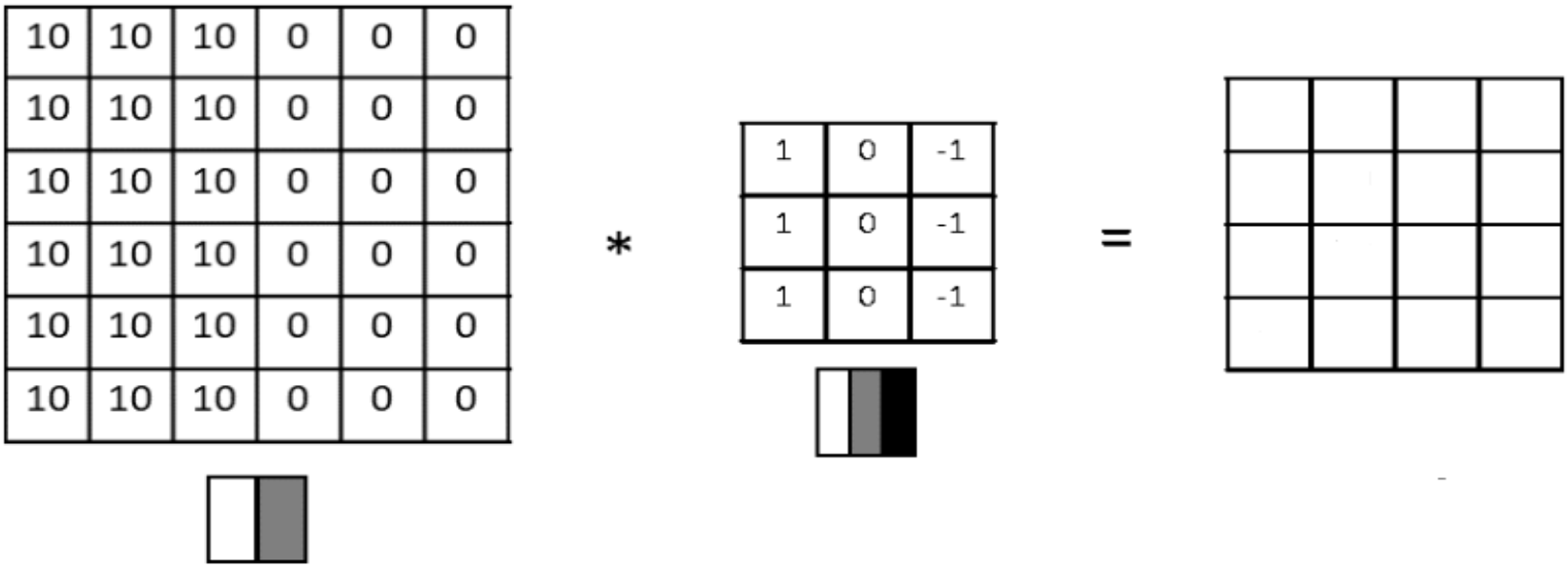
# Convolutional using Edge detection




- This type of filter detects “Vertical edges” and hence called as **Vertical edge detection filter**
- The generalised way of calculating output size

$$(n \times n) * (f \times f) \Rightarrow (n - f + 1) * (n - f + 1)$$

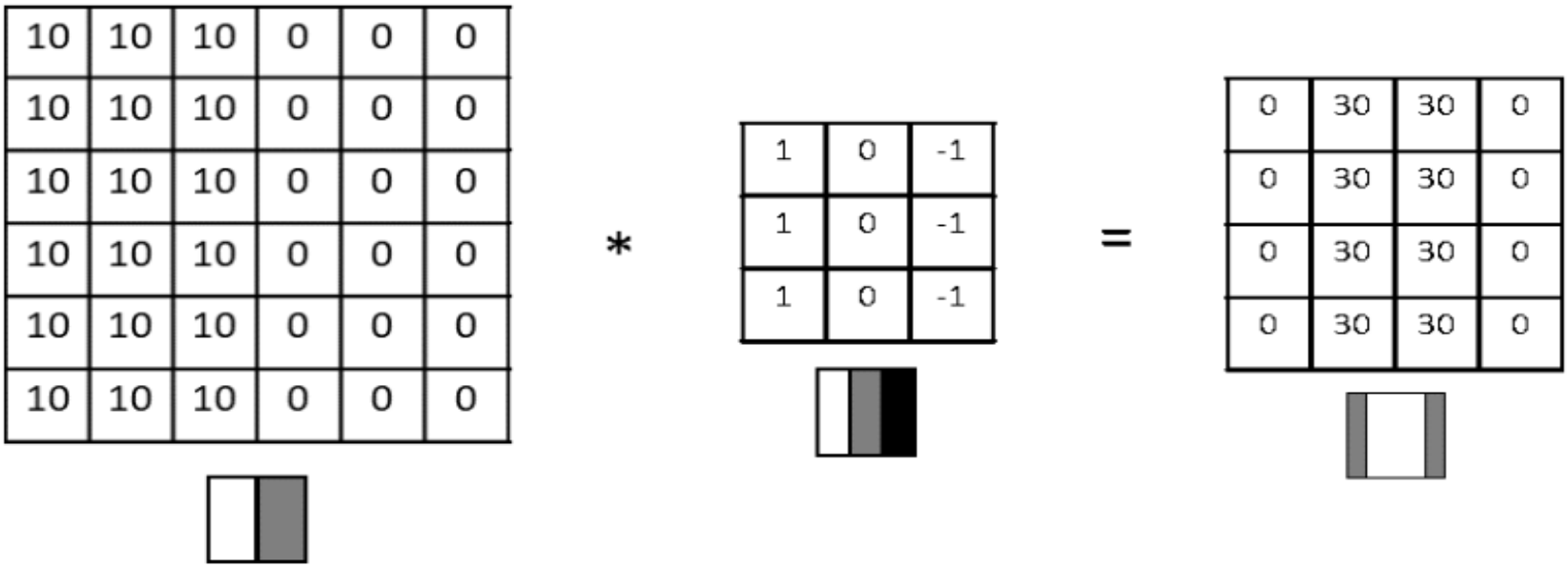
# Vertical Edge detection




*An example of a vertical edge detection. Here we have light to dark transition.*

- Vertical edge detection filter is used

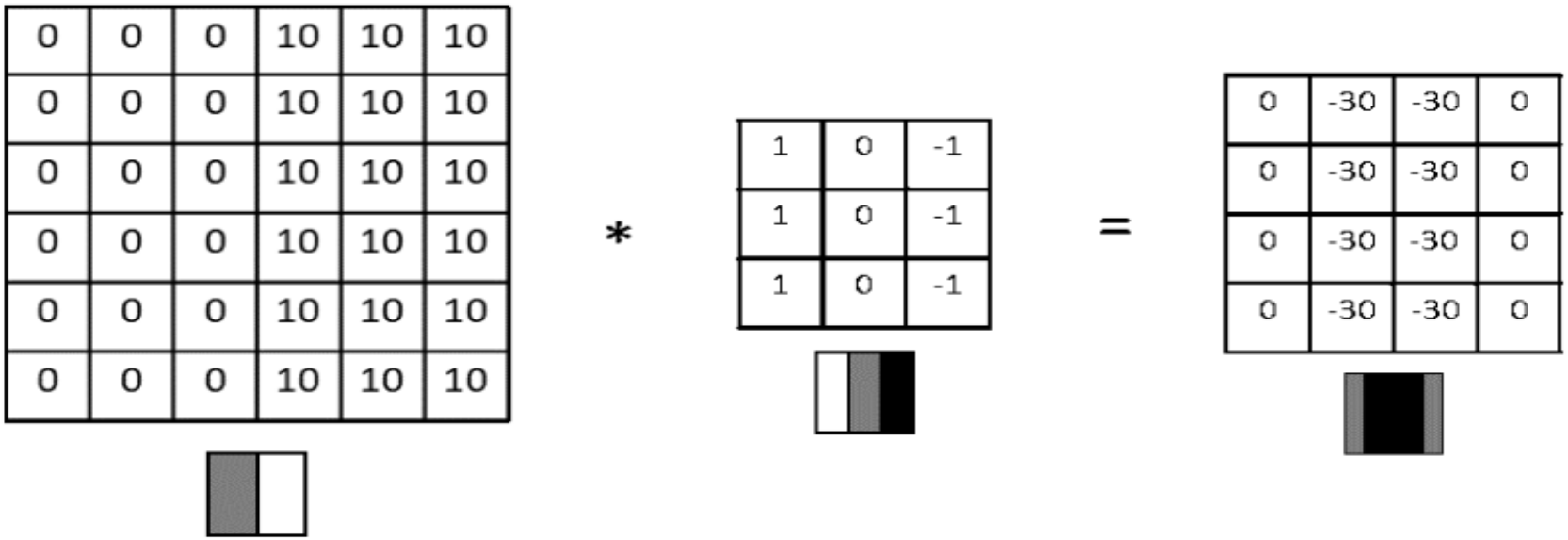
# Vertical Edge detection




*An example of a vertical edge detection. Here we have light to dark transition.*

- White in the output indicates transition from bright to dark

# Vertical Edge detection




*An example of vertical edge detection of dark to light transition*

- Black in the output indicates transition from dark to bright

# Horizontal Edge detection

| | | | | | |
|---|---|---|---|---|---|
| 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 |
| 10 | 10 | 10 | 10 | 10 | 10 |
| 10 | 10 | 10 | 10 | 10 | 10 |
| 10 | 10 | 10 | 10 | 10 | 10 |
| 10 | 10 | 10 | 10 | 10 | 10 |

8 X 6

*

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

3 X 3

=

6 X 4

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 735 | 735 | 735 | 735 |
| 735 | 735 | 735 | 735 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

- White in the output indicates transition from bright to dark

# Edge detection

convolutional operation

Input Image

| | | | | | |
|---|---|---|---|---|---|
| 3 | 1 | 6 | 2 | 8 | 0 |
| 4 | 8 | 5 | 9 | 1 | 5 |
| 6 | 8 | 0 | 0 | 1 | 9 |
| 3 | 6 | 9 | $^{1}4$ | $^{0}2$ | $^{-1}1$ |
| 5 | 2 | 5 | $^{1}0$ | $^{0}7$ | $^{-1}1$ |
| 8 | 3 | 0 | $^{1}5$ | $^{0}3$ | $^{-1}1$ |

6x6

*

filter

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

3x3

=

Output

| | | | |
|---|---|---|---|
| 2 | 4 | 1 | -3 |
| -1 | 9 | 10 | -2 |
| 0 | 12 | 4 | -7 |
| 2 | 2 | 2 | 6 |

4x4

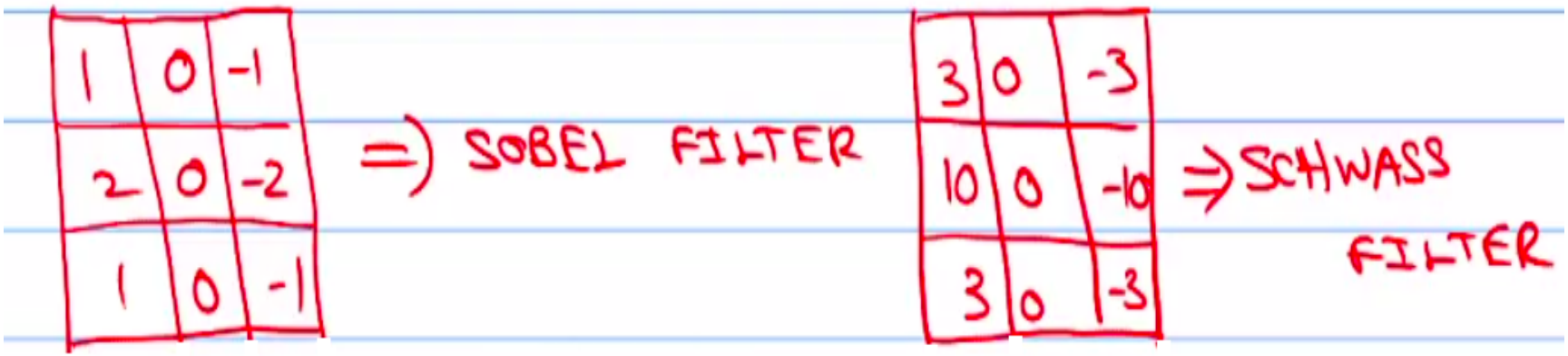
= 1*4 + 1*0 + 1*5 + 0*2 + 0*7 + 0*3 + (-1)*1 + (-1)*1 + (-1)*1 = 6

Visual for input matrix

Current visual for output matrix

Convolutional Operation

# Other filters in Image processing




- These filter give weightage more to central values

# Other filters in Image processing

- Inorder to identify edges in the complex image, rather than using pre-defined filters, one can define them as parameters in NN and obtain them using backpropagation


- Rather than learning only horizontal and vertical edges, this filter may findout angles $45^o, 37^o$, *etc*..

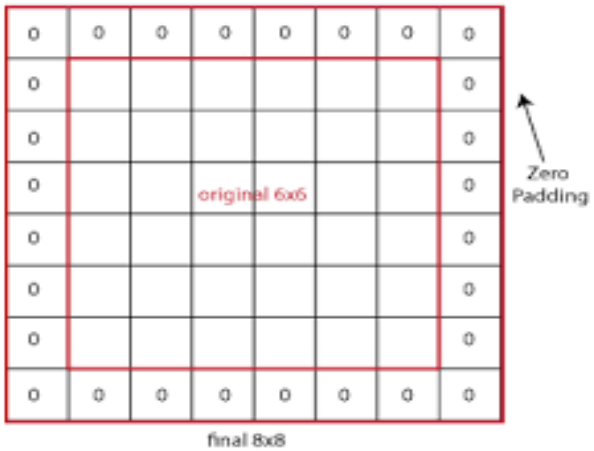
## Padding

- When we apply convolution operator everytime, image shrinks!
- Pixels ar the corner of the matrix is used only in one of the matrix computation
- Pixels at the center is used in more matrix computation.
- Pixles a the corner are least used

To overcome this problem, we use “**padding**”

# Padding

**Padding is the process of adding border with "0" to the given image**



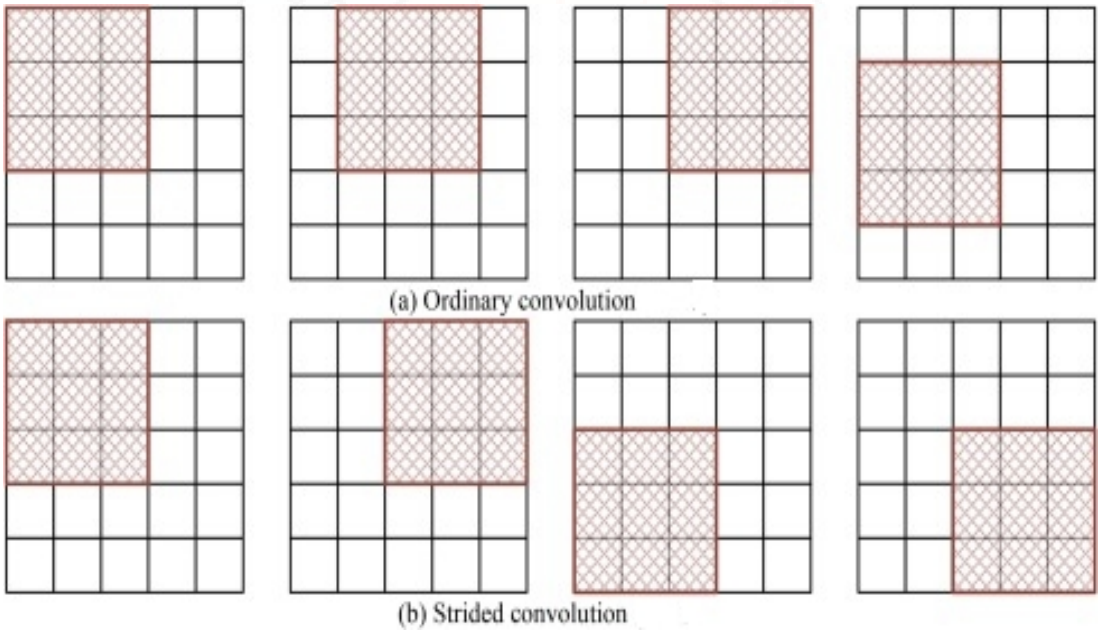

Original image $\Rightarrow$ 6 x 6
With padded image $\Rightarrow$ 8 x 8
Filter $\Rightarrow$ 3 x 3

Output without padding $\Rightarrow$ 4 x 4
Output with padding $\Rightarrow$ 6 x 6

- The general expression for output size with padding is $(n\ x\ n) * (f\ x\ f) \Rightarrow (n+2p-f+1) * (n+2p-f+1)$
- In padding, sometimes instead of zero, one use the value of edge pixels

# Strided Convolution

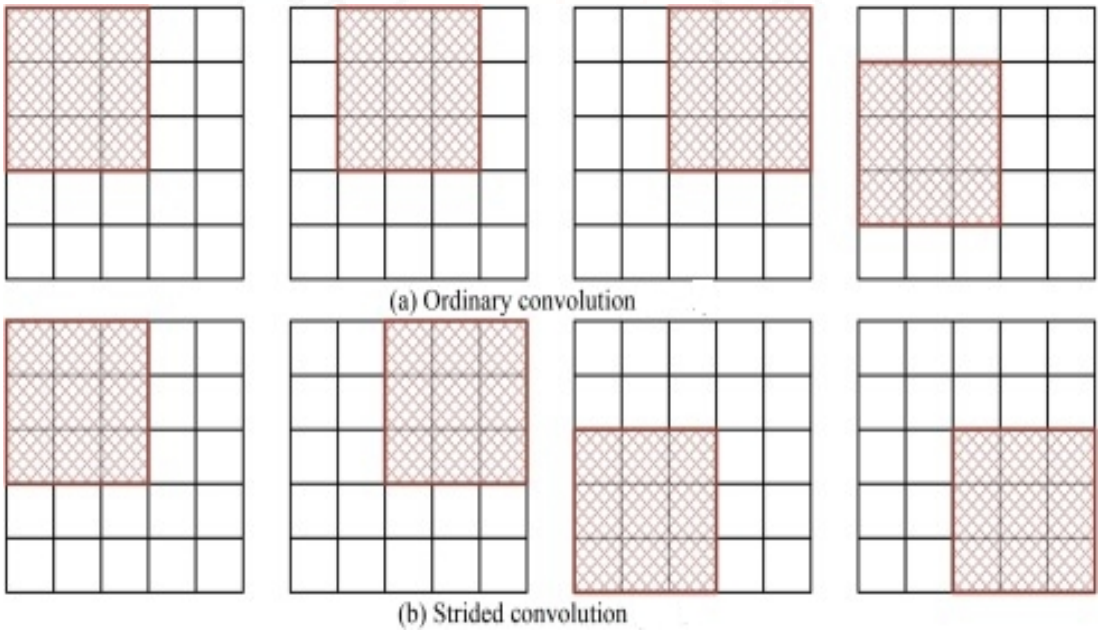
**Rather than moving to the next cell continuously, we jump based on stride number**




- Row-1 $\Rightarrow$ Ordinary convolution (Stride = 1)
- Row-2 $\Rightarrow$ Strided convolution (Stride=2)
- Stride (jump) is on both horizontal and vertical side

# Strided Convolution

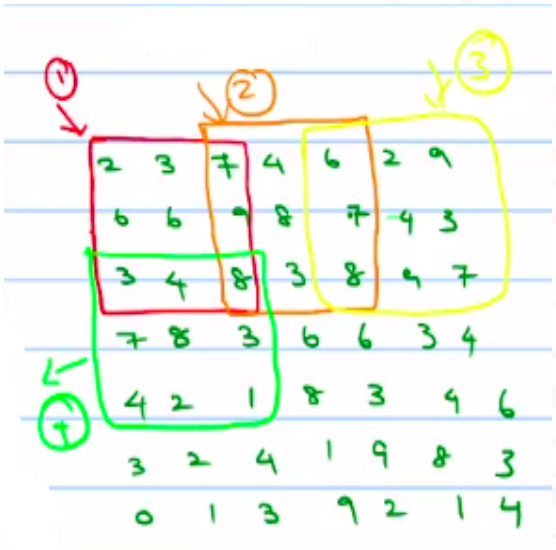
**Rather than moving to the next cell continuously, we jump based on stride number**



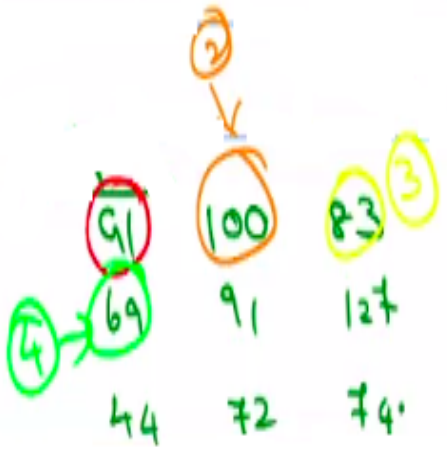

- Row-1 $\Rightarrow$ Ordinary convolution (Stride = 1)
- Row-2 $\Rightarrow$ Strided convolution (Stride=2)
- Stride (jump) is on both horizontal and vertical side

# Strided Convolution

**Rather than moving to the next cell continuously, we jump based on stride number**



**Output:**



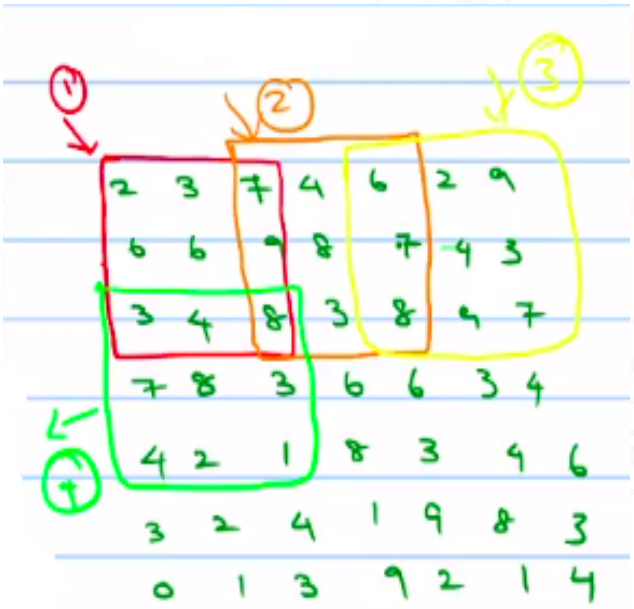
- The general expression for output size with padding is

  $(n \times n) * (f \times f) \Rightarrow (\frac{n+2p-f}{S} + 1) * (\frac{n+2p-f}{S} + 1)$

  where 'p' is the padding and 'S' is the stride

# Strided Convolution

**Rather than moving to the next cell continuously, we jump based on stride number**



**Filter:**

$$\begin{matrix} 3 & 4 & 4 \\ 1 & 0 & 2 \\ -1 & 0 & 3 \end{matrix}$$

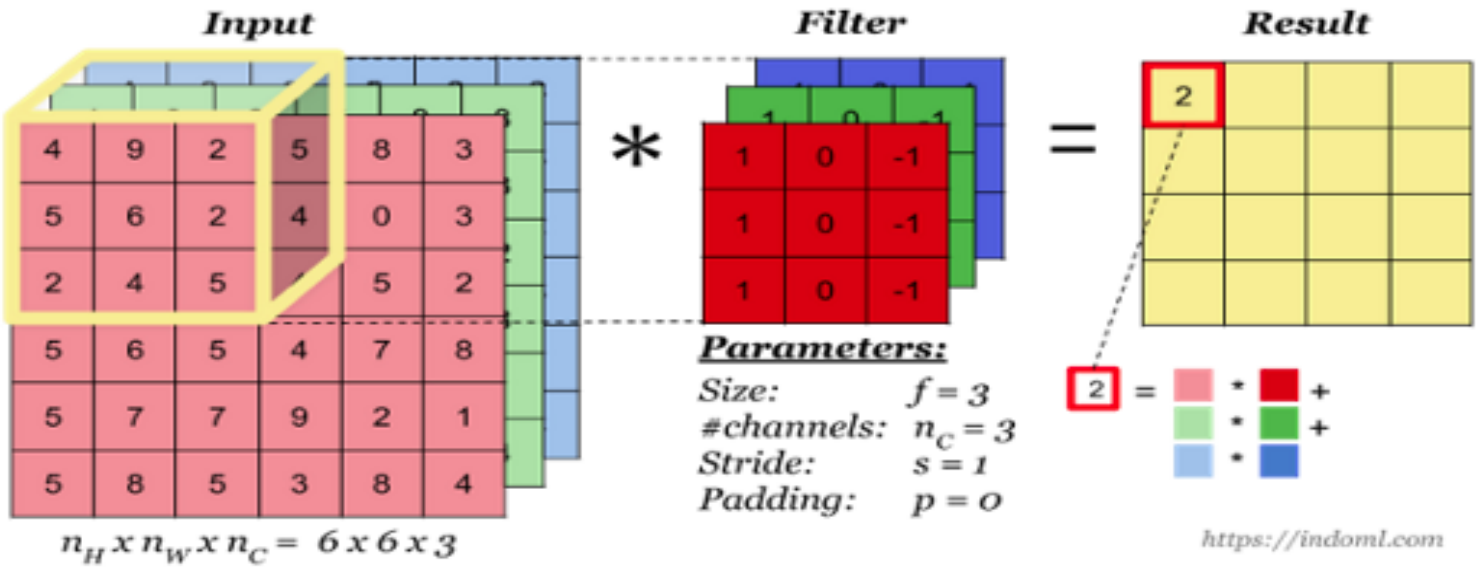
**Output:**

# Strided convolution

- The rate at which we are scanning the image is called stride (rate at which we move the filter)
- When we use higher stride, if the length is not sufficient for convolution, we will pad at the edges
- If one is interested in very fine features, then stride should be small
- If one is interested in macro features then stride should be large

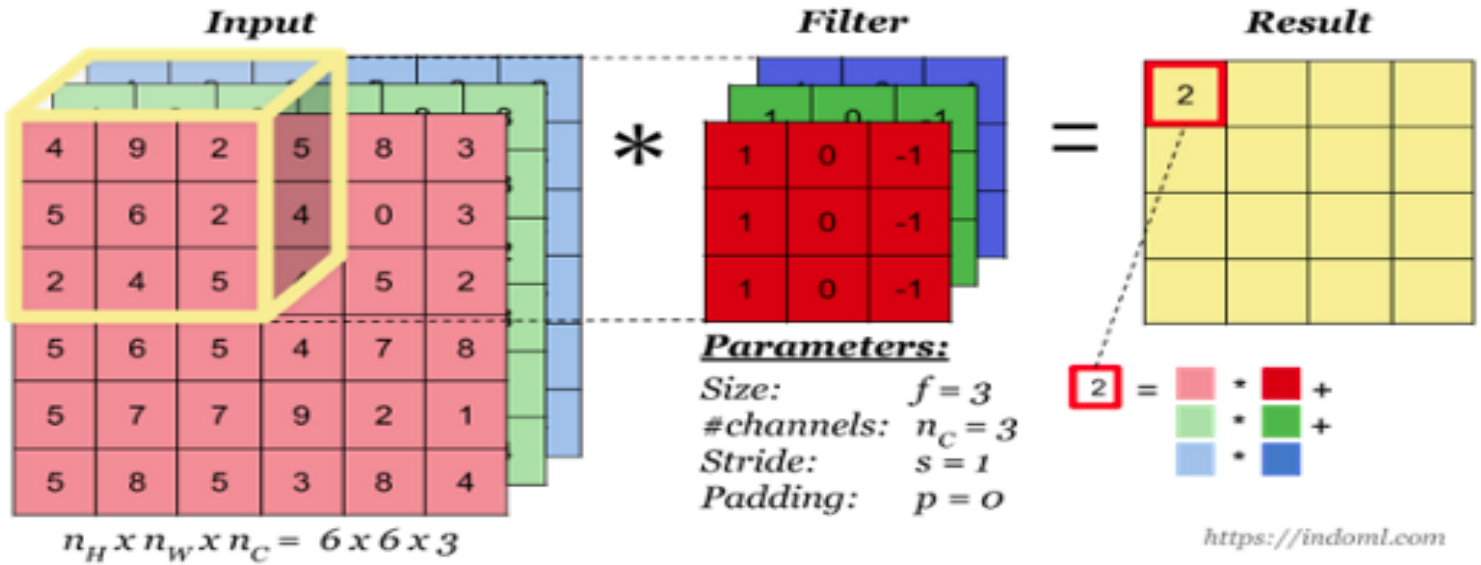
# Convolution over volume

- Till now, we have discussed convolution over 2D images (Gray scale). Now, we will discuss about covolution over volume (more than 2D) for RGB images




Convolution over volume: element-wise product in each channel then adding them together.
Source:https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

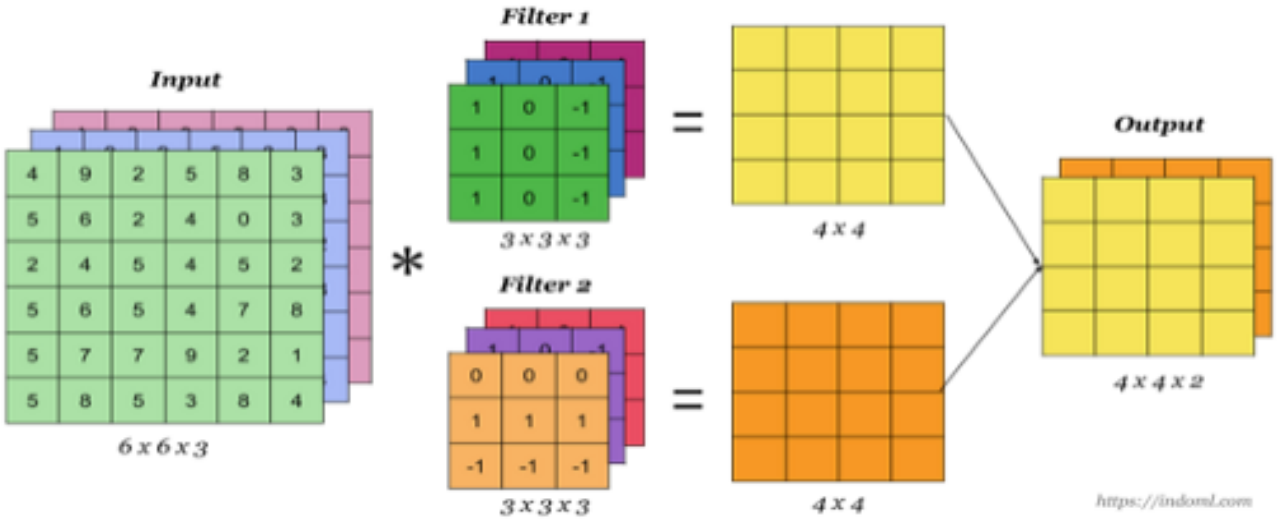
# Convolution over volume




Convolution over volume: element-wise product in each channel then adding them together.
Source:https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

- Each of 27 number in each filter channel is mulitiplied with 27 number in images and add everything to compute the values
- Number of channels in the image should match with number of channels in the filter

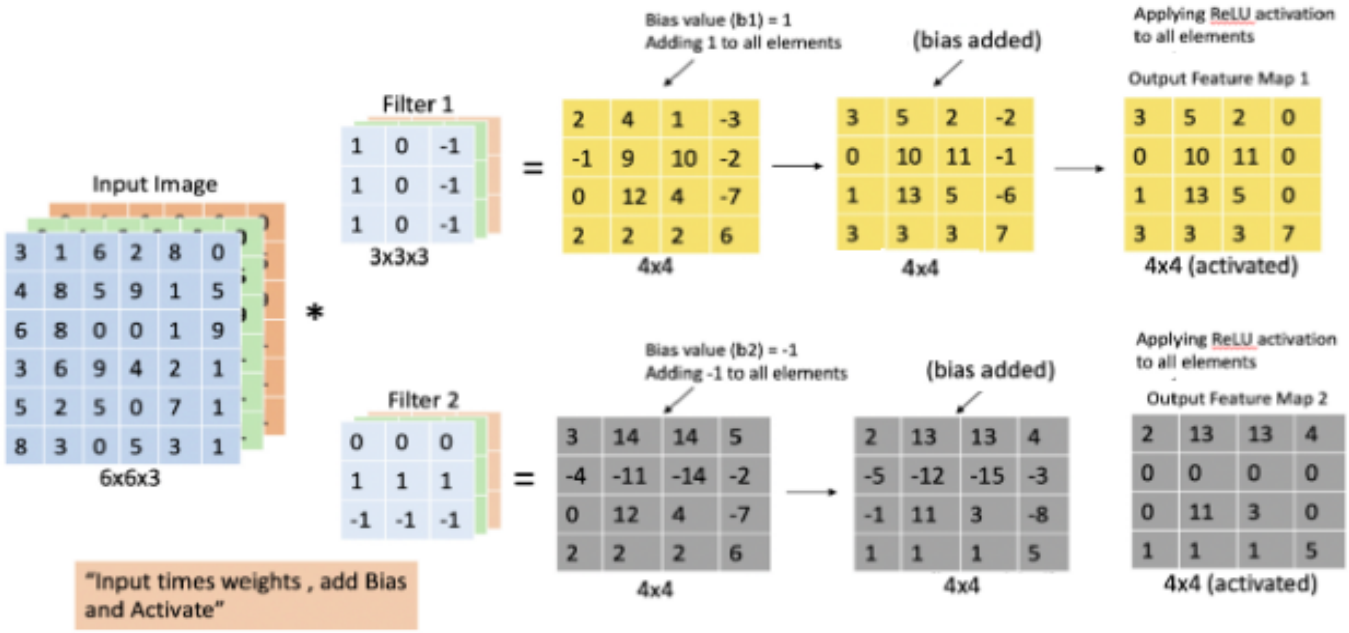
# Convolution with multiple filters

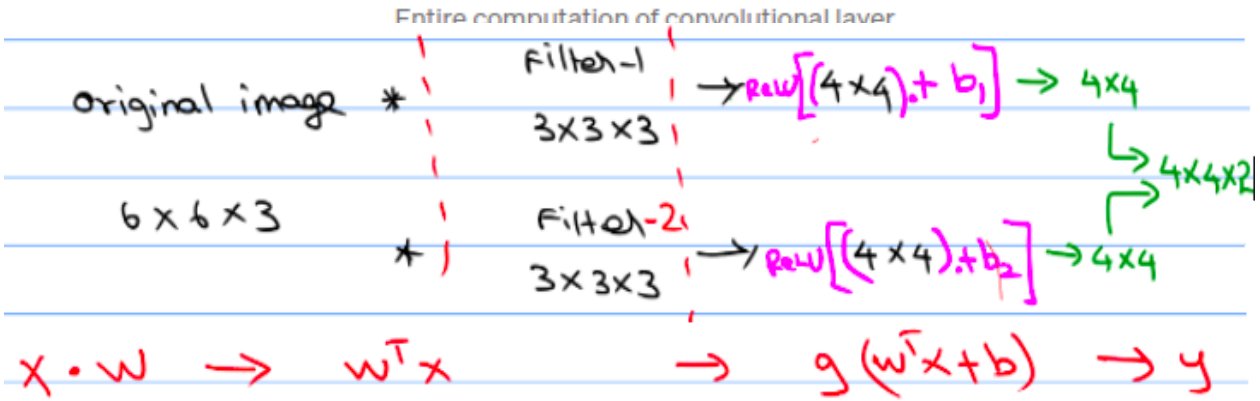



Convolution using 2 filters. The output is two 4×4 feature maps. One from each filter Source: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

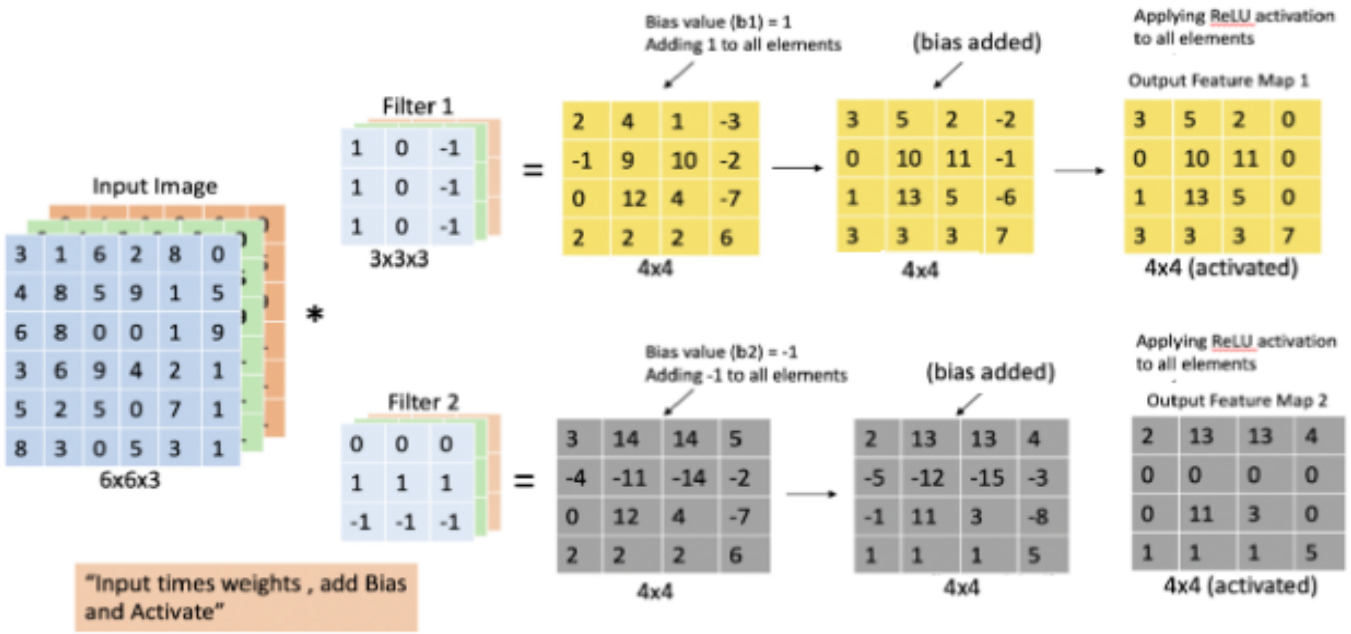
- While using multiple filters, output is stacked!
  Image $\Rightarrow$ n x n x $n_c$
  Filter $\Rightarrow$ f x f x $n_c$
  If there are $n'_c$ number of filters, then
  Output $\Rightarrow$ (n-f+1) x (n-f+1) x $n'_c$

# Single Convolution layer



# Single Convolution layer

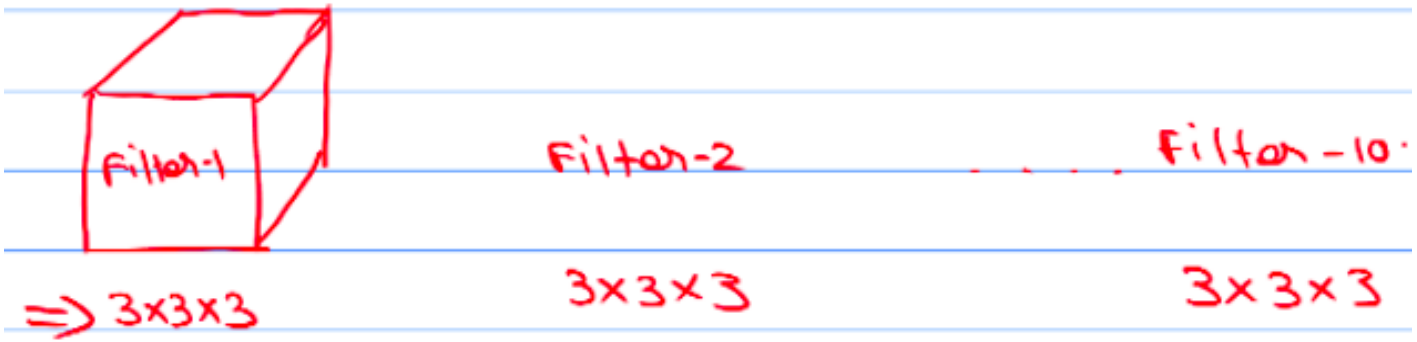



Entire computation of convolutional layer

- This process is very similar to forward propagation
- Filters are acting as weights and it is multiplied with input X
- To this product, add bias and then apply activation
- The output from activation is called as "**feature map**"

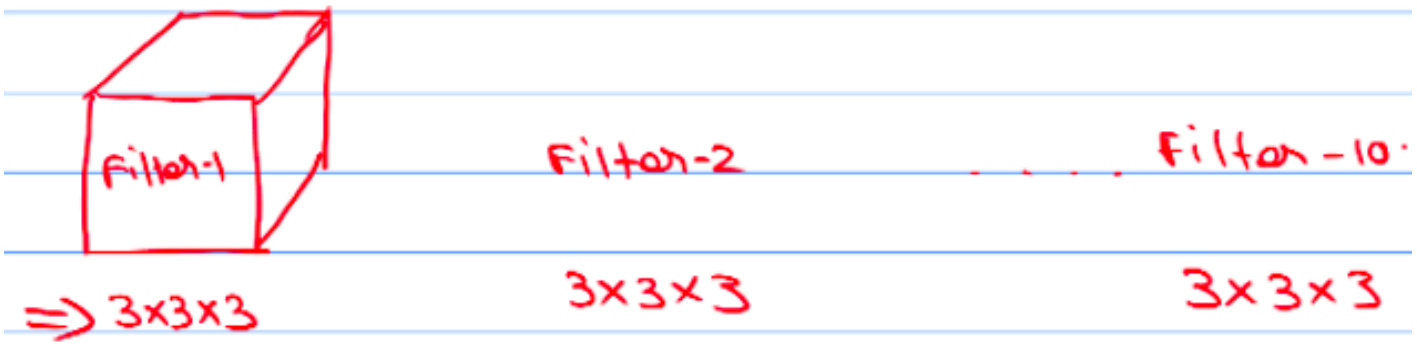
# Number of parameters in one layer

- Consider we have 3 x 3 RYB filters which are 10 in numbers.
- Trainable parameters are W and b. **Calculate total number of parameters**.



# Number of parameters in one layer

- Consider we have 3 x 3 RYB filters which are 10 in numbers.
- Trainable parameters are W and b. **Calculate total number of parameters**.




  - For each filter, there will be 27 weights and one bias totalling to 28 parameters
  - For 10 filters, total parameters = 28 x 10 = **280 parameters**
- Irrespective of image size, the number of parameters remain same.

## Number of parameters in $L^{th}$ layer

If layer L is a convolutional layer,

$f^{[L]} \Rightarrow$ *Filter size*
$p^{[L]} \Rightarrow$ *Padding*
$s^{[L]} \Rightarrow$ *Stride*
$n_c^{[L]} \Rightarrow$ *Number of Filter*

$Input \Rightarrow n_H^{[L-1]} \; x \; n_W^{[L-1]} \; x \; n_c^{[L-1]}$ $\quad$ $Output \Rightarrow n_H^{[L]} \; x \; n_W^{[L]} \; x \; n_c^{[L]}$

## Number of parameters in $L^{th}$ layer

If layer L is a convolutional layer,

$f^{[L]} \Rightarrow$ *Filter size*
$p^{[L]} \Rightarrow$ *Padding*
$s^{[L]} \Rightarrow$ *Stride*
$n_c^{[L]} \Rightarrow$ *Number of Filter*

$Input \Rightarrow n_H^{[L-1]} \; x \; n_W^{[L-1]} \; x \; n_c^{[L-1]}$ $\quad$ $Output \Rightarrow n_H^{[L]} \; x \; n_W^{[L]} \; x \; n_c^{[L]}$

$n_H^{[L]} \Rightarrow \left(\frac{n^{[L-1]}+2P^{[L]}-f^{[L]}}{S^{[L]}}+1\right)$ $\quad$ $n_W^{[L]} \Rightarrow \left(\frac{n^{[L-1]}+2P^{[L]}-f^{[L]}}{S^{[L]}}+1\right)$

# Number of parameters in $L^{th}$ layer

$Input \Rightarrow n_H^{[L-1]} \; x \; n_W^{[L-1]} \; x \; n_c^{[L-1]}$ $\quad$ $Output \Rightarrow n_H^{[L]} \; x \; n_W^{[L]} \; x \; n_c^{[L]}$

$n_H^{[L]} \Rightarrow \left(\frac{n^{[L-1]}+2P^{[L]}-f^{[L]}}{S^{[L]}}+1\right)$ $\quad$ $n_W^{[L]} \Rightarrow \left(\frac{n^{[L-1]}+2P^{[L]}-f^{[L]}}{S^{[L]}}+1\right)$

## Number of parameters in $L^{th}$ layer

$Input \Rightarrow n_H^{[L-1]} \; x \; n_W^{[L-1]} \; x \; n_c^{[L-1]}$ $\quad$ $Output \Rightarrow n_H^{[L]} \; x \; n_W^{[L]} \; x \; n_c^{[L]}$

$n_H^{[L]} \Rightarrow \left(\frac{n^{[L-1]}+2P^{[L]}-f^{[L]}}{S^{[L]}}+1\right)$ $\quad$ $n_W^{[L]} \Rightarrow \left(\frac{n^{[L-1]}+2P^{[L]}-f^{[L]}}{S^{[L]}}+1\right)$

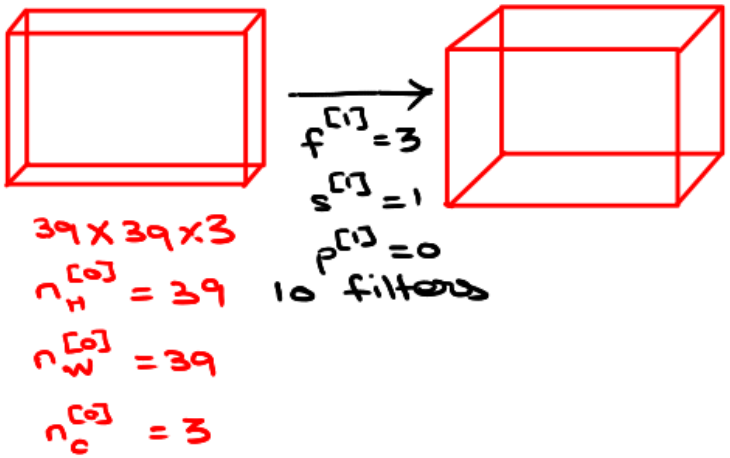
If there are $n_c$ channels, then the filter size becomes

$Filter\ size \Rightarrow f^{[L]} \; x \; f^{[L]} \; n_c^{[L-1]}$

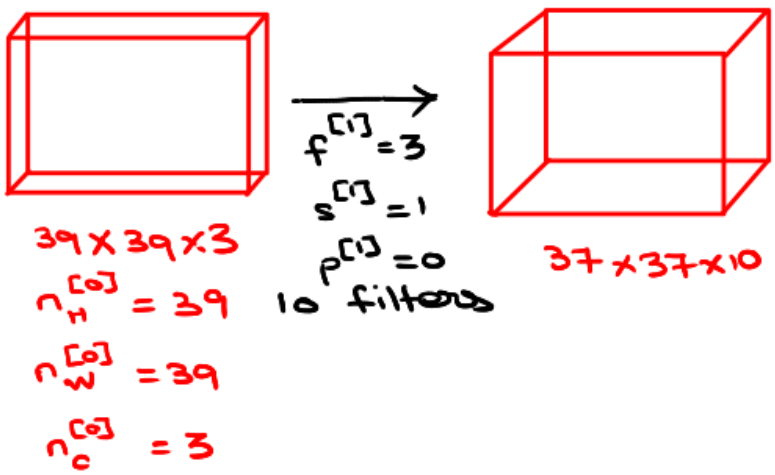
The filter channel must match with input channel
If there are $n_c'$ filters in the L layer

$Weights \Rightarrow f^{[L]} \; x \; f^{[L]} \; x \; n_c^{[L-1]} \; x \; n_c^{[L]'}$
$Bias \Rightarrow n_c^{[L]}$
$Vectorised\ Output\ A^{[L]} \Rightarrow m \; x \; n_H^{[L]} \; x \; n_W^{[L]} \; x \; n_c^{[L]}$

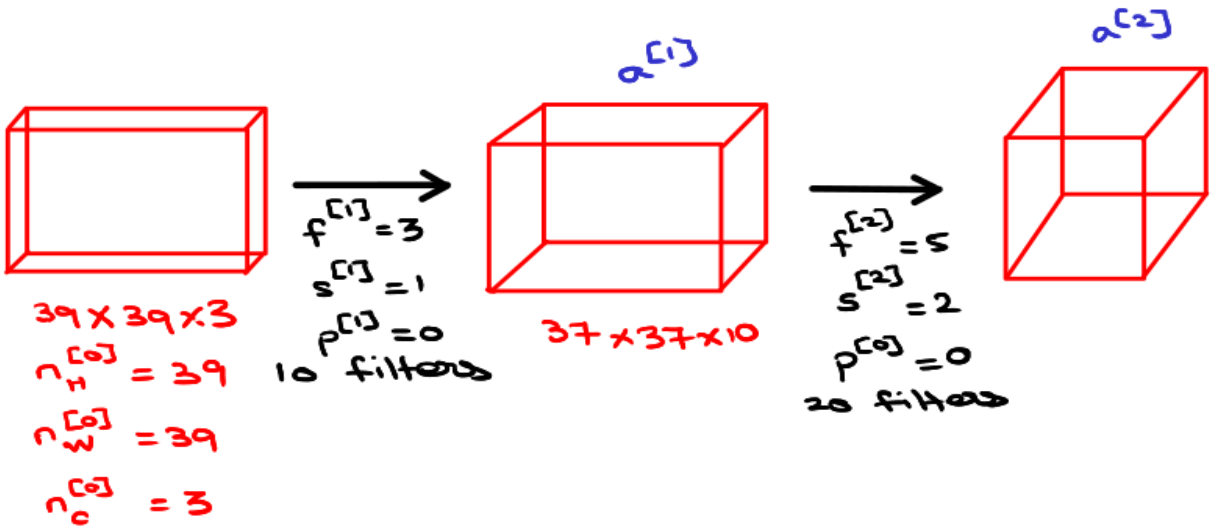
# Convolutional Neural Network



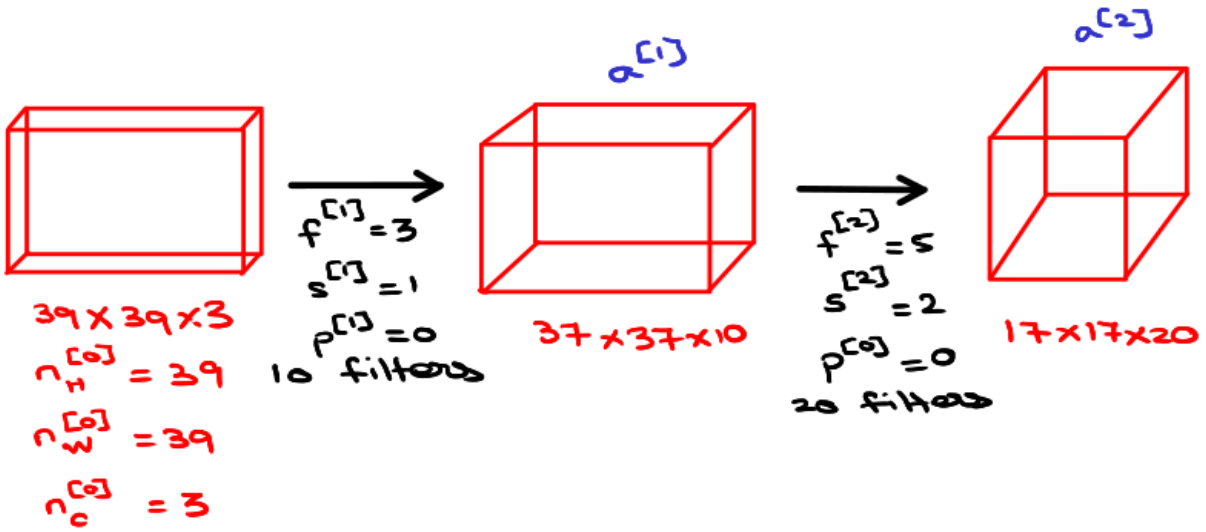
# Convolutional Neural Network



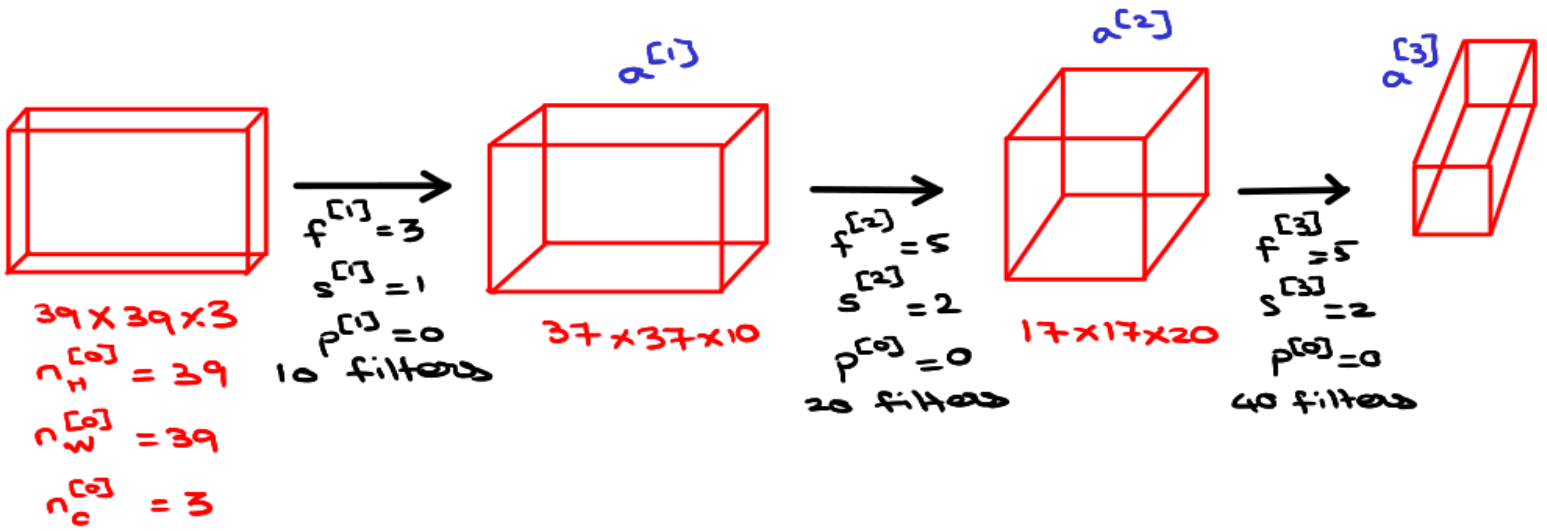
# Convolutional Neural Network



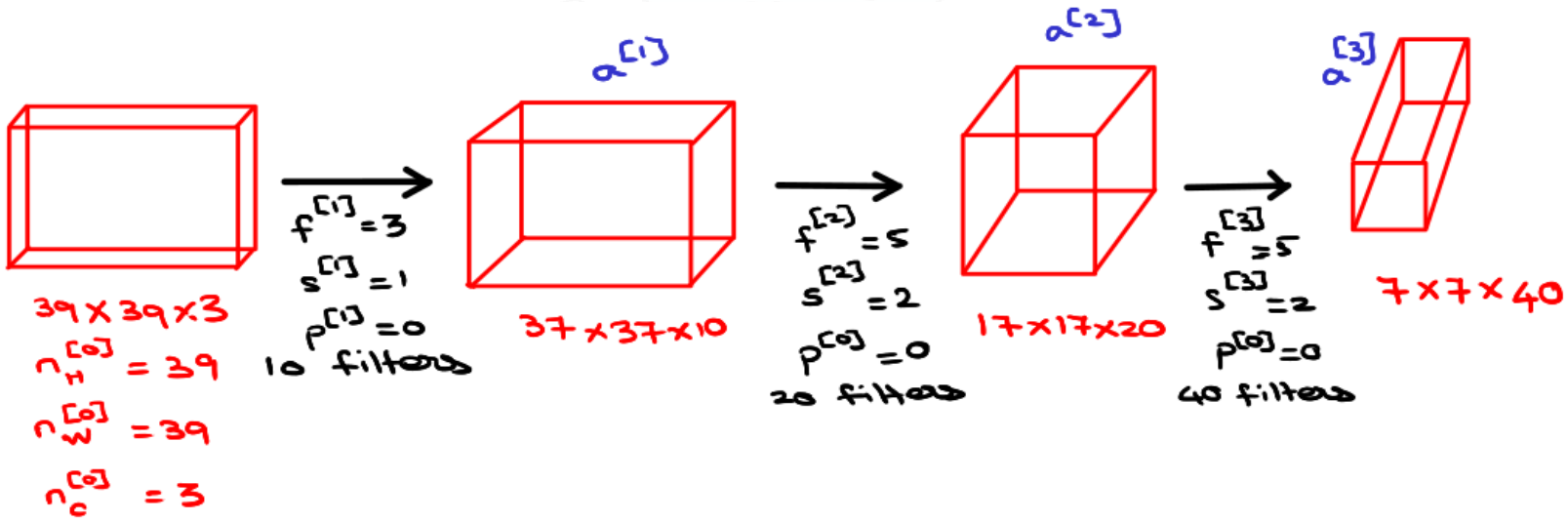
# Convolutional Neural Network



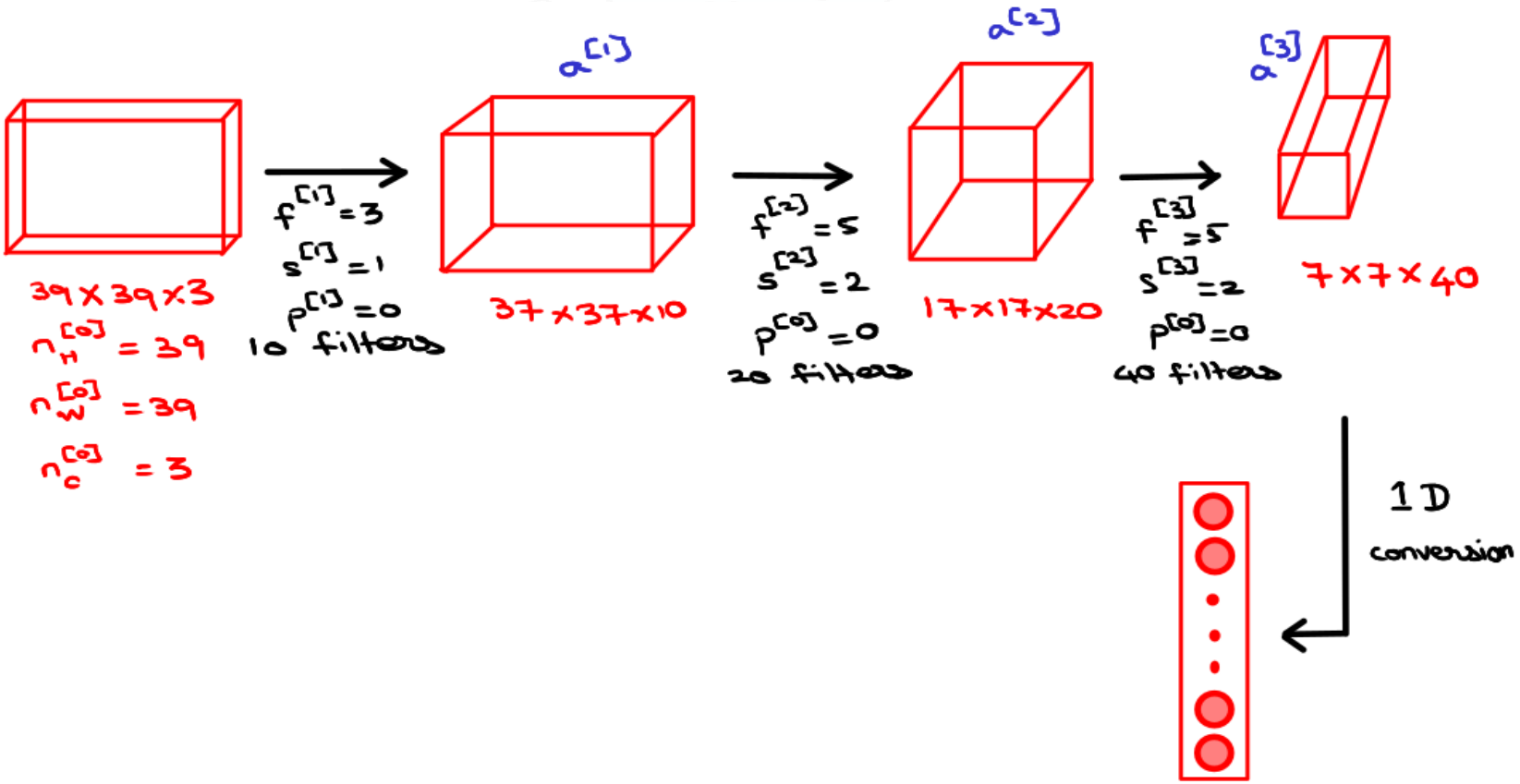
# Convolutional Neural Network



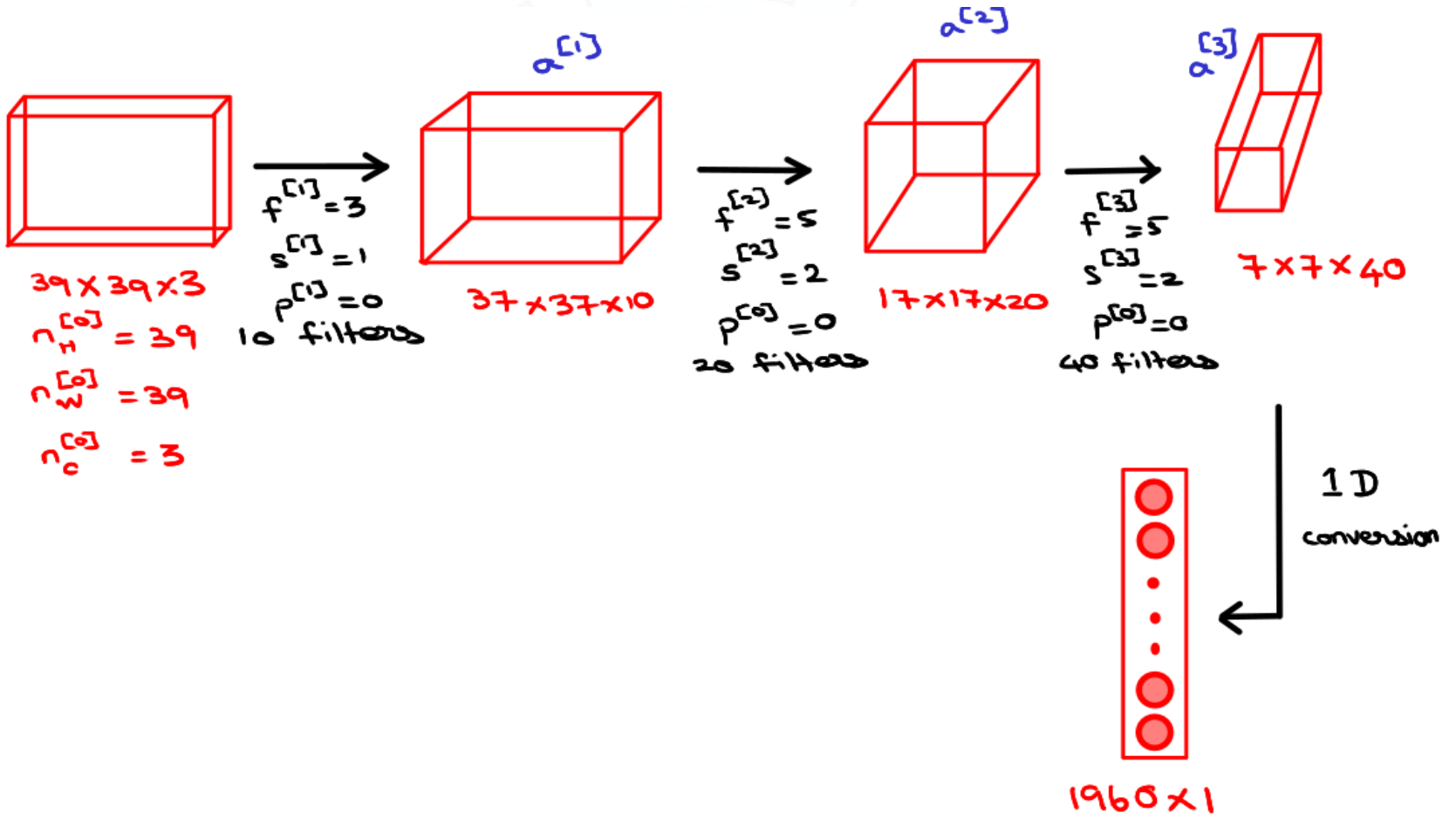
# Convolutional Neural Network



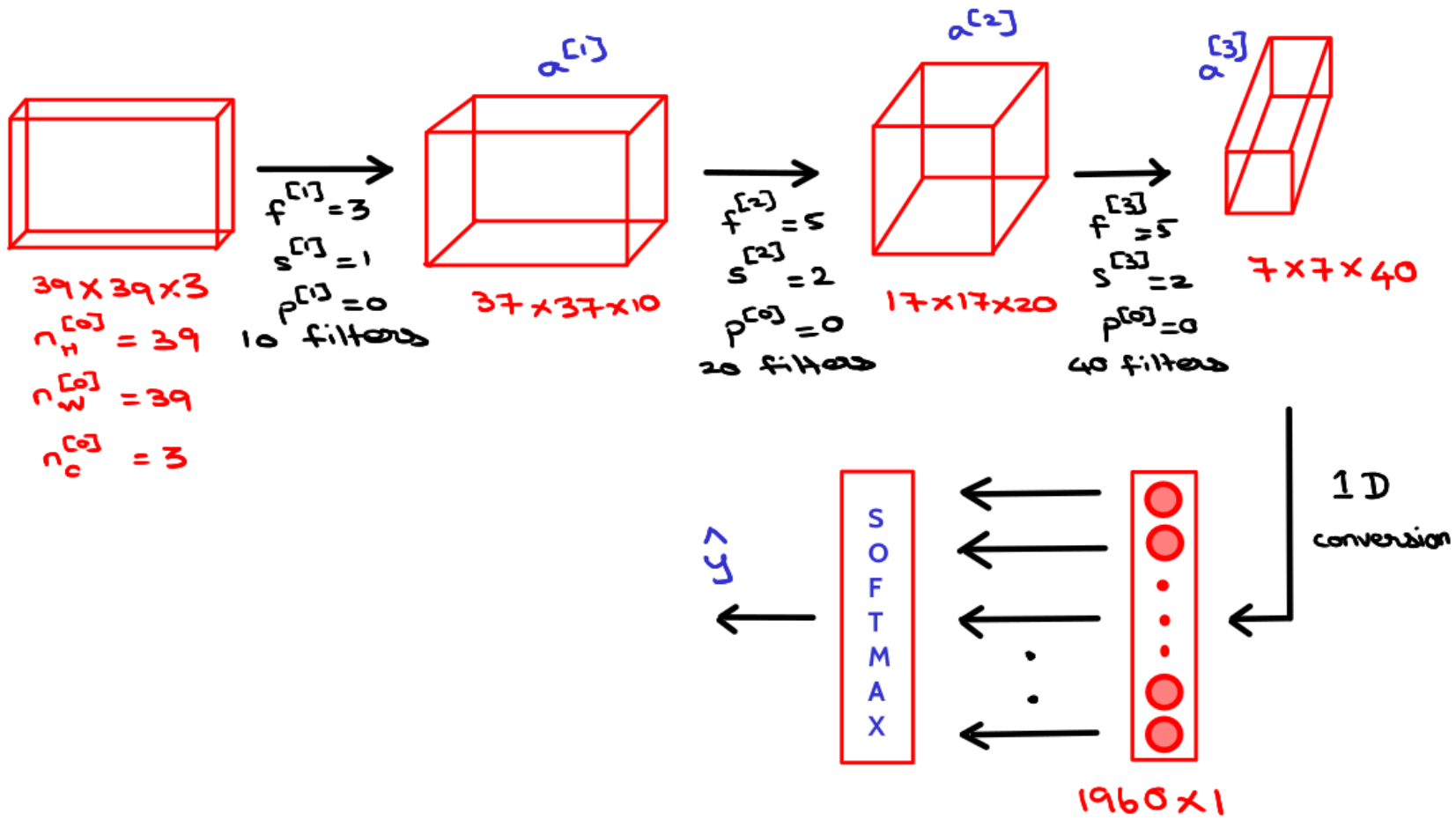
# Convolutional Neural Network



# Convolutional Neural Network



# Convolutional Neural Network



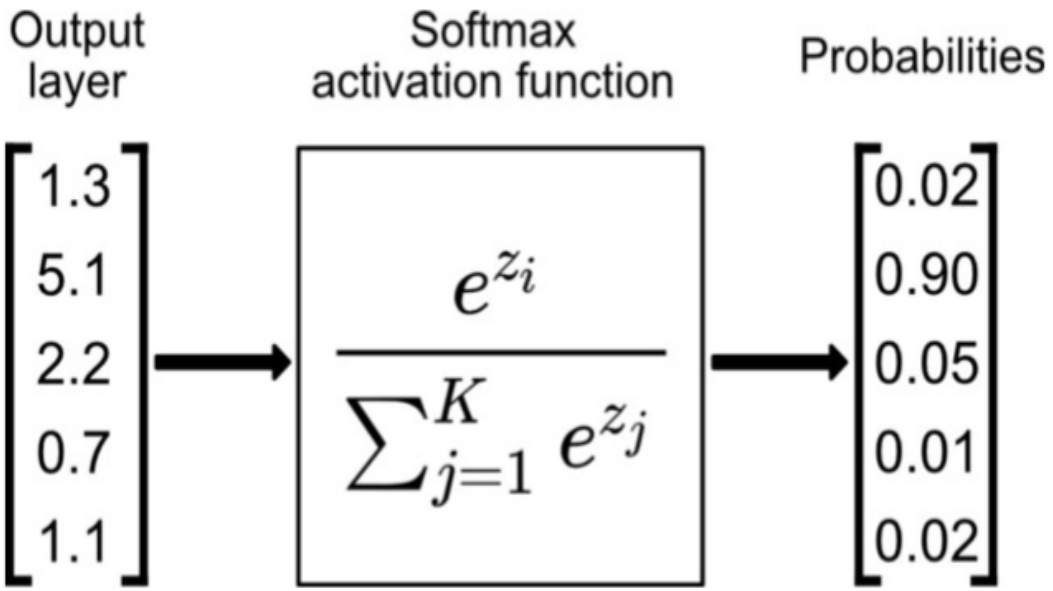

- Image size is going down (39-37-17-7)
- Number of filters (3-10-20-40)

# Convolutional Neural Network: Softmax layer

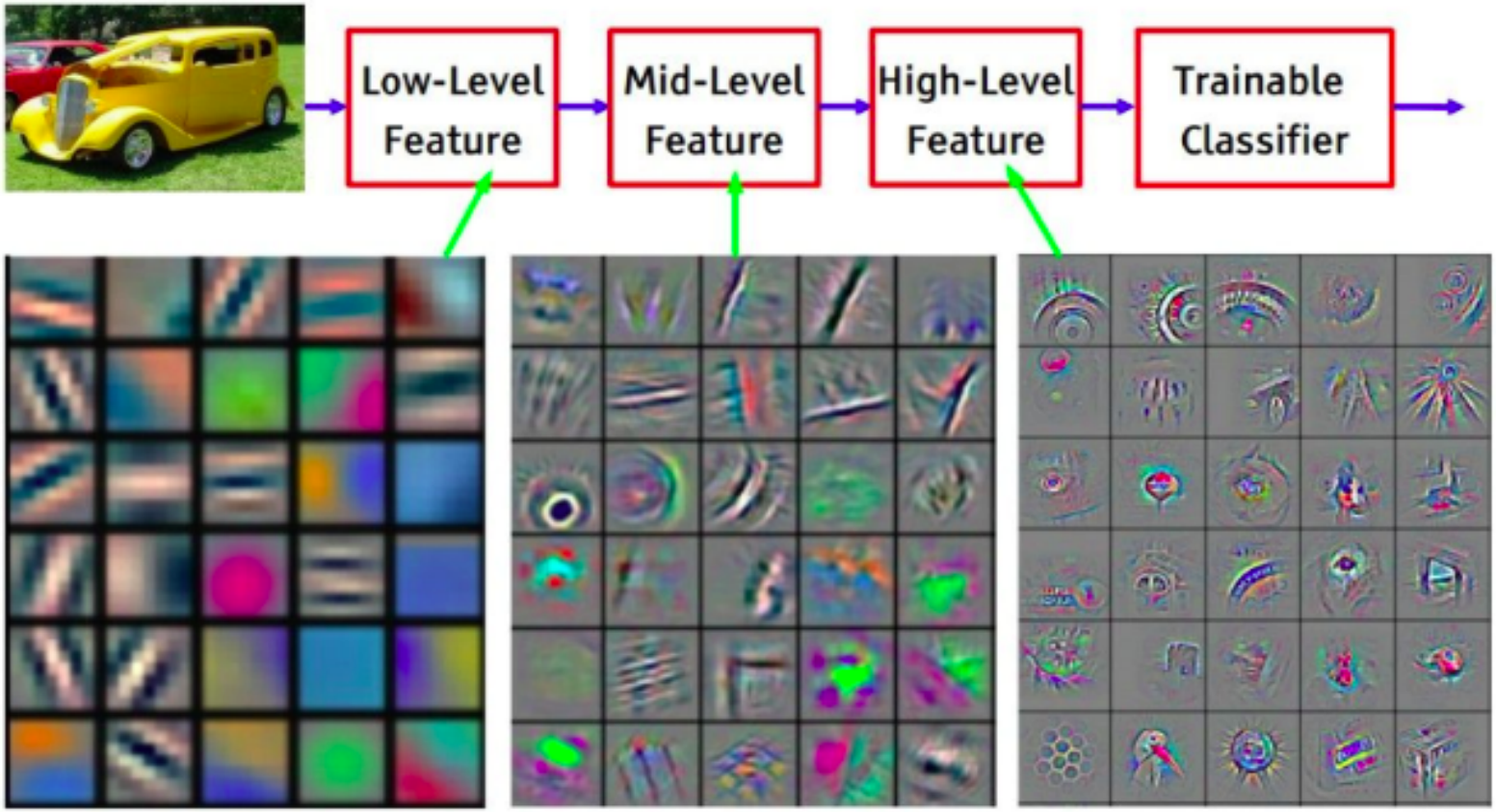
- Used for multi-class classification
- Gives the output probabilities of particular class and adds up to 1
- If there are 'c' classes, then there will be c values
- Array as an input and array as an output

$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

# Convolutional Neural Network: Softmax layer



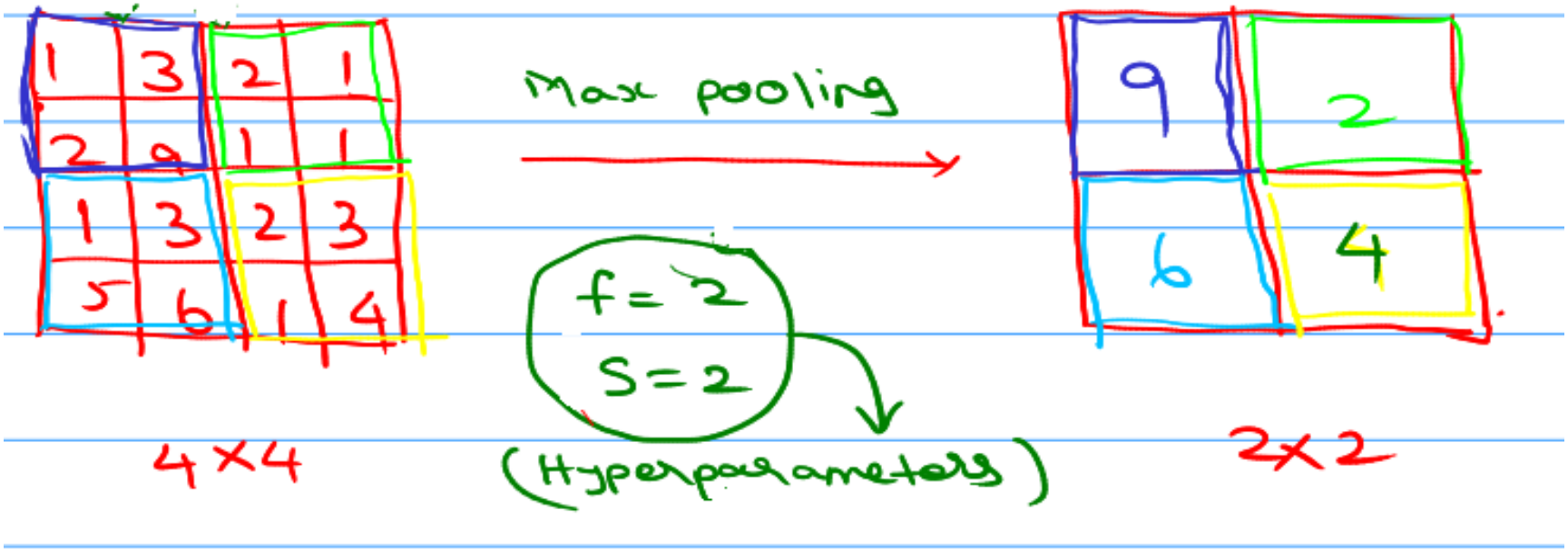
# CNN: Features at different levels




Different features recognised at different layers

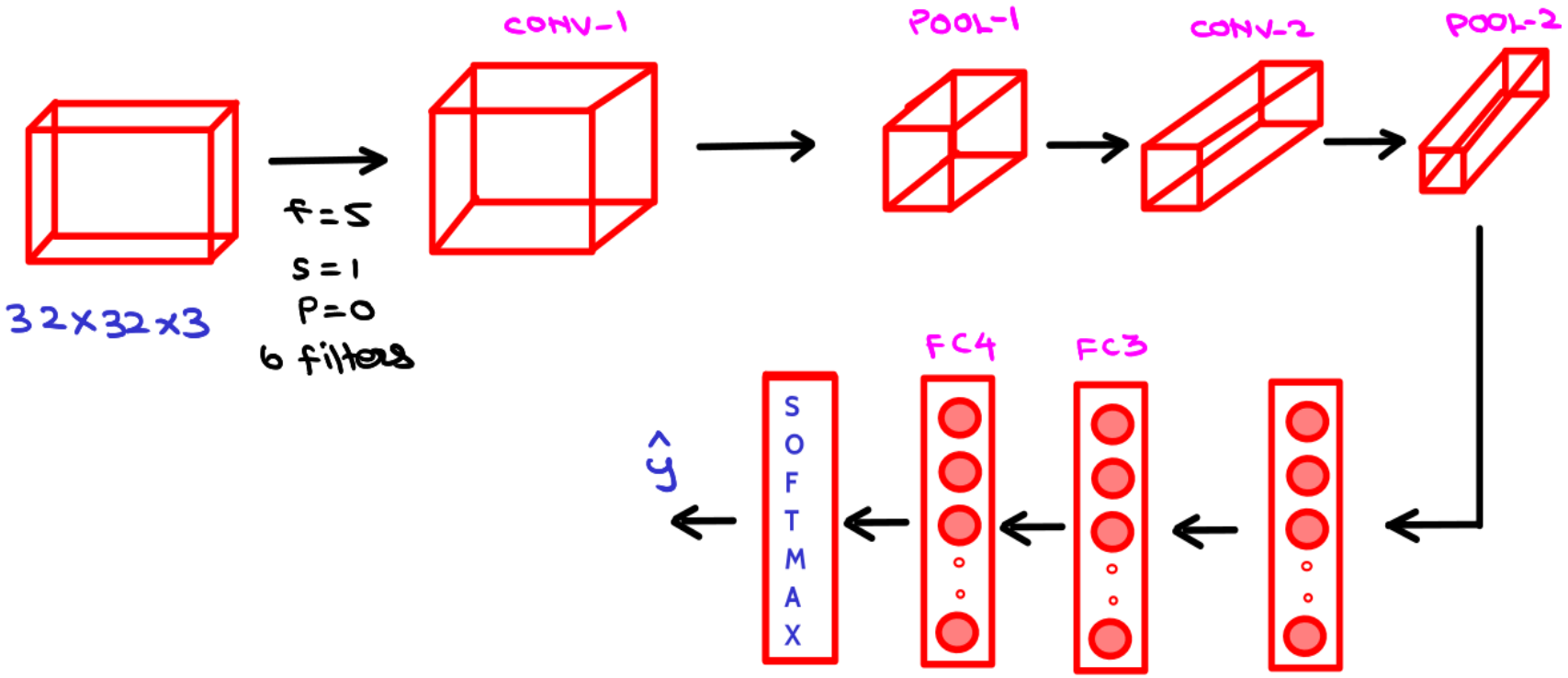
# Convolutional Neural Network: Pooling layer

- Pooling layer is used for speedup the computation
- The value of 'f' and 's' are defined by the user
- No learnable parameters in the pooling layer



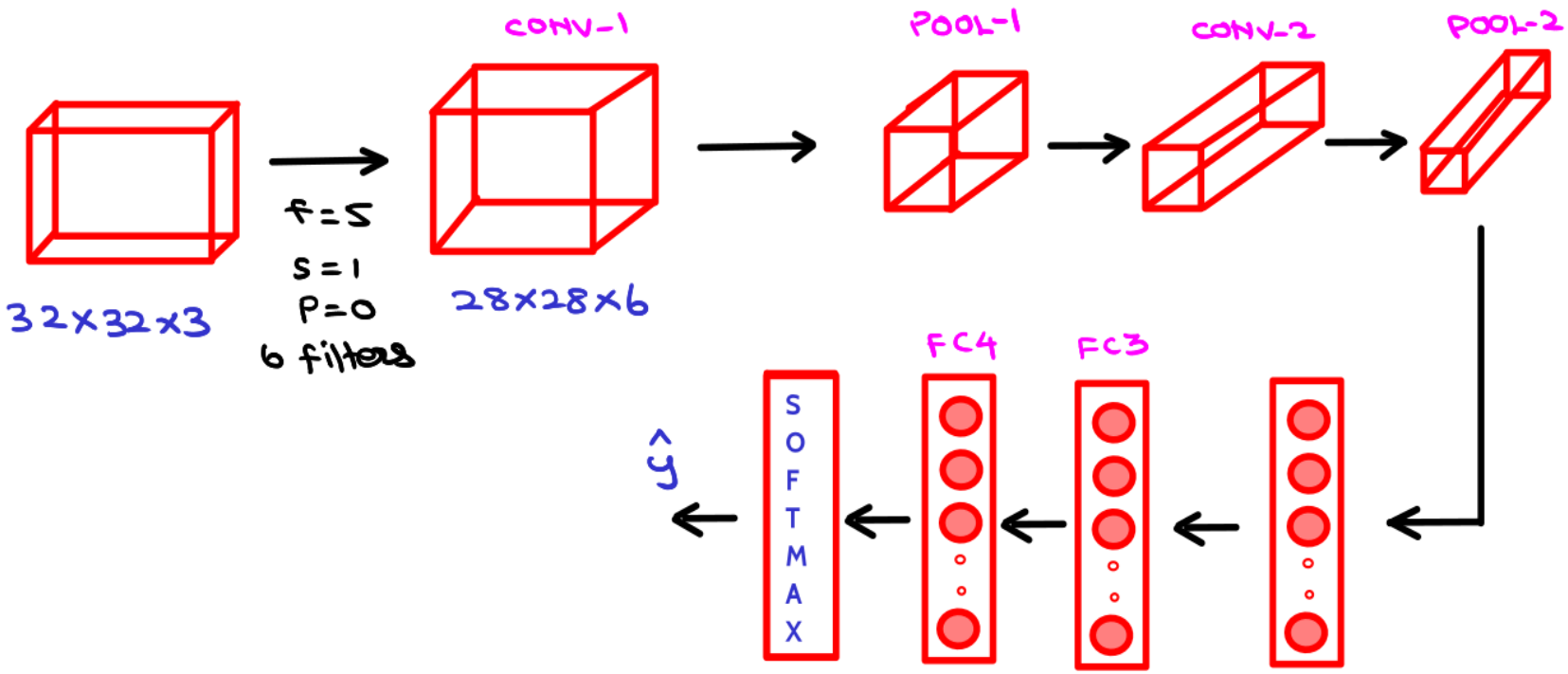

- The generalised output size $\Rightarrow \frac{n+2p-f}{S} + 1$

# Parameters in CNN



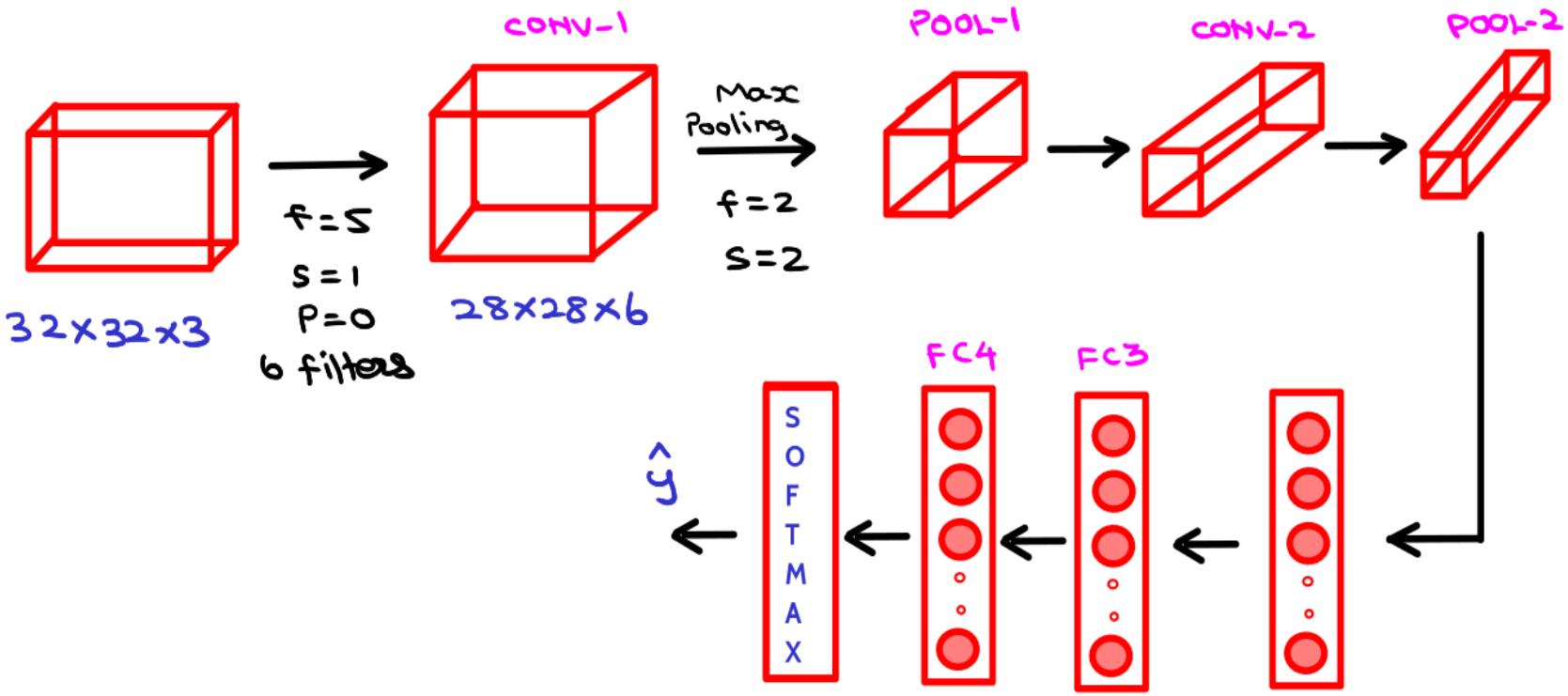

- CONV1-POOL1-CONV2-POOL2-FC3-FC4-Softmax

# Parameters in CNN



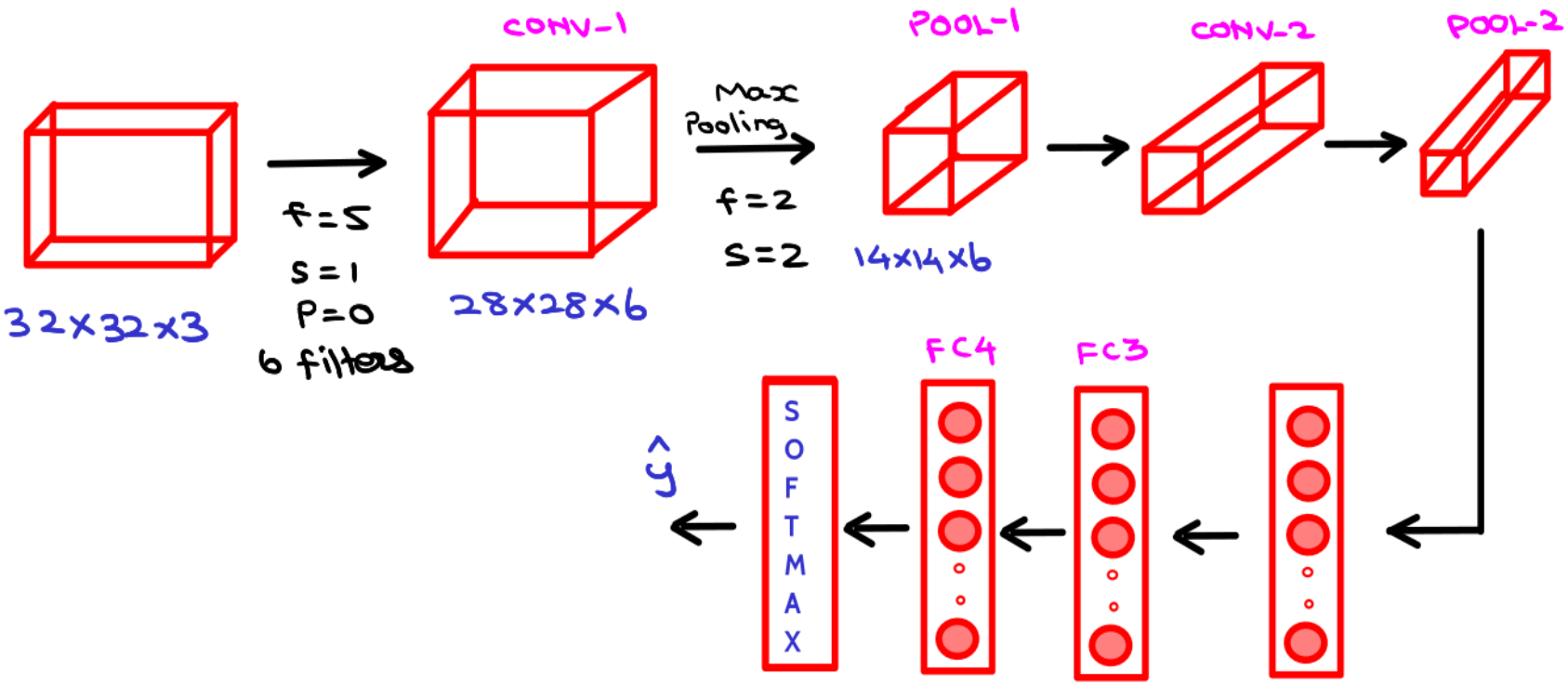

- CONV1-POOL1-CONV2-POOL2-FC3-FC4-Softmax

# Parameters in CNN



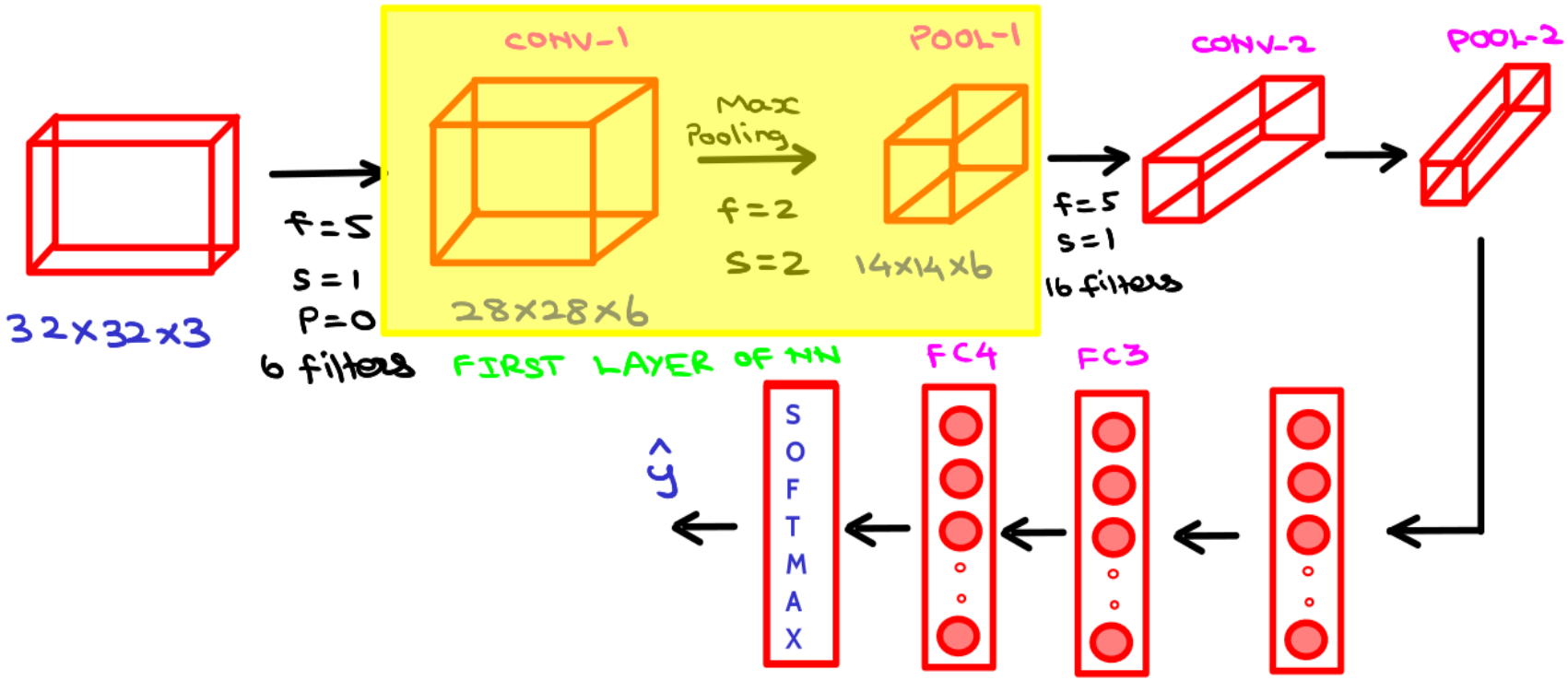

- CONV1-POOL1-CONV2-POOL2-FC3-FC4-Softmax

# Parameters in CNN



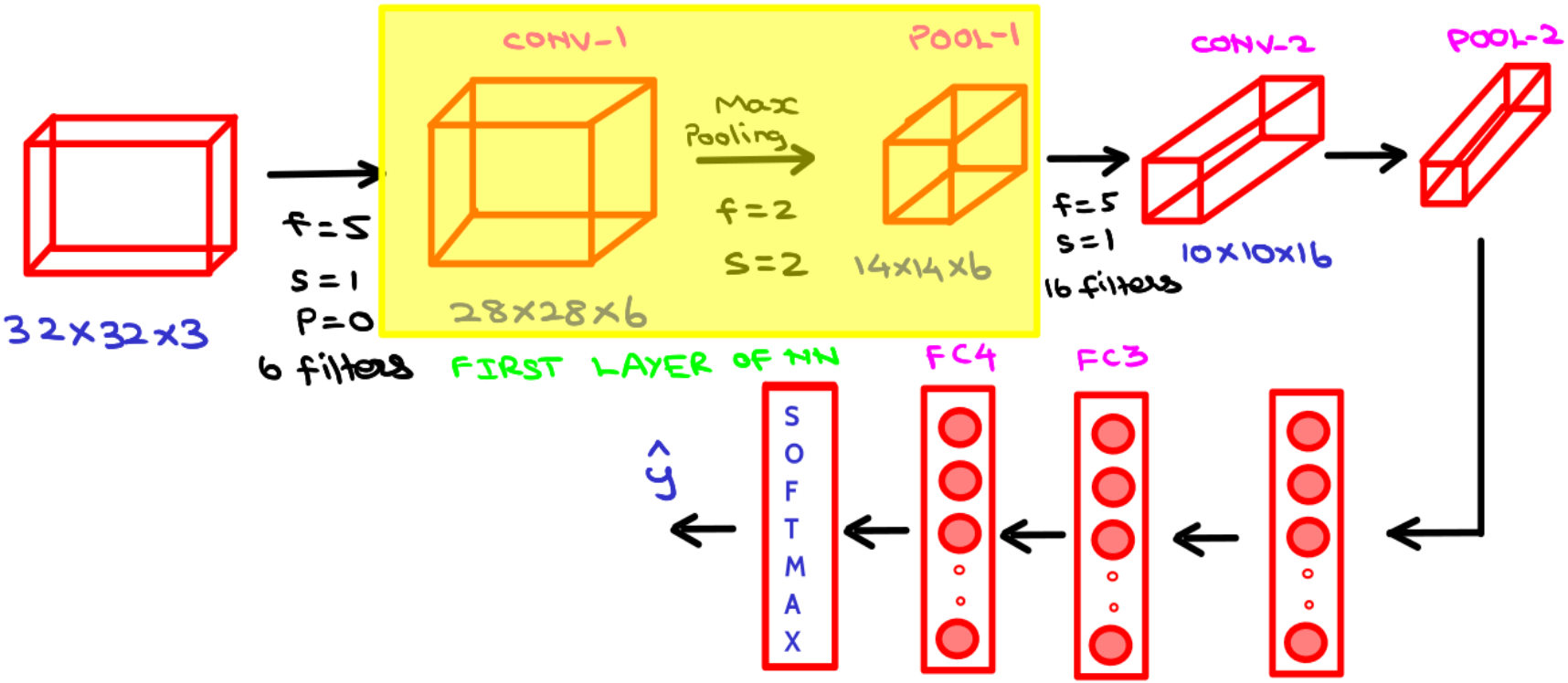

- CONV1-POOL1-CONV2-POOL2-FC3-FC4-Softmax

# Parameters in CNN



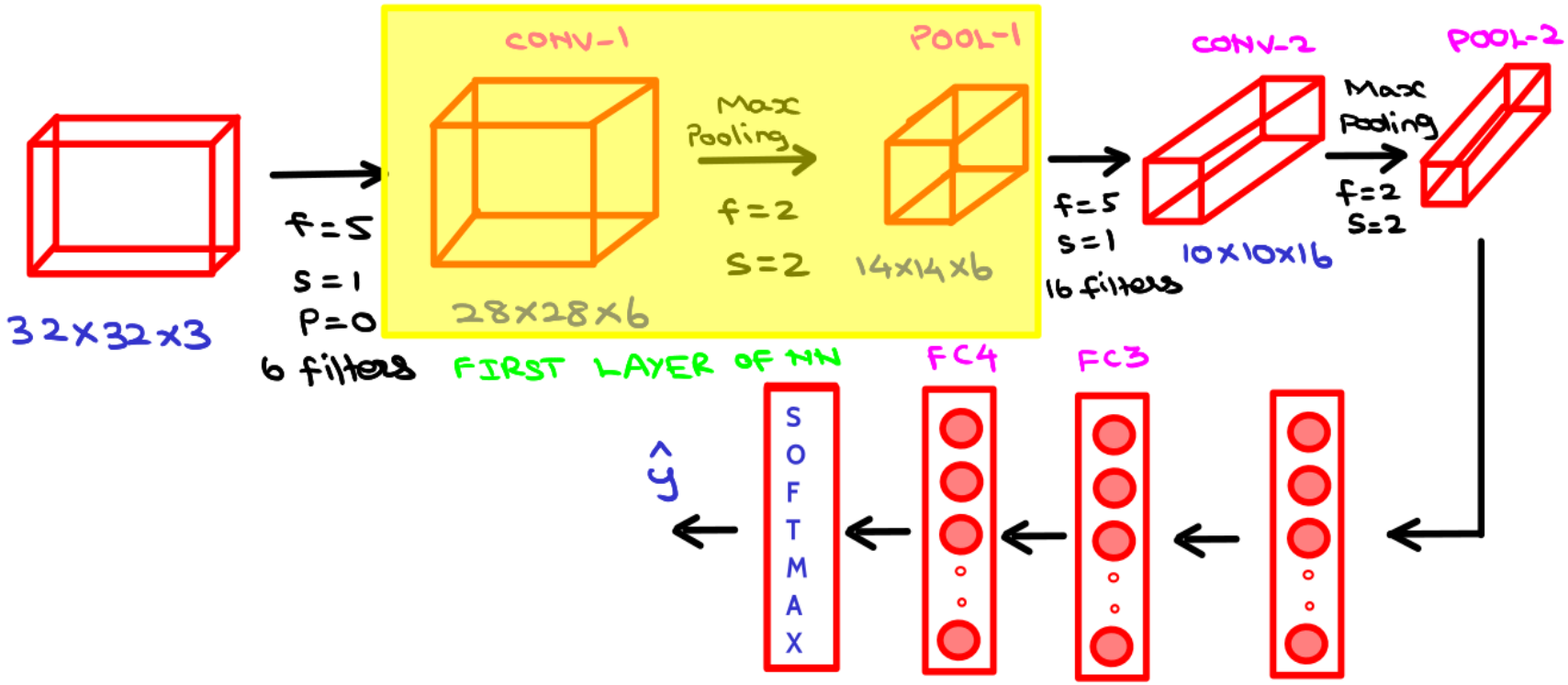

- CONV1-POOL1-CONV2-POOL2-FC3-FC4-Softmax

# Parameters in CNN



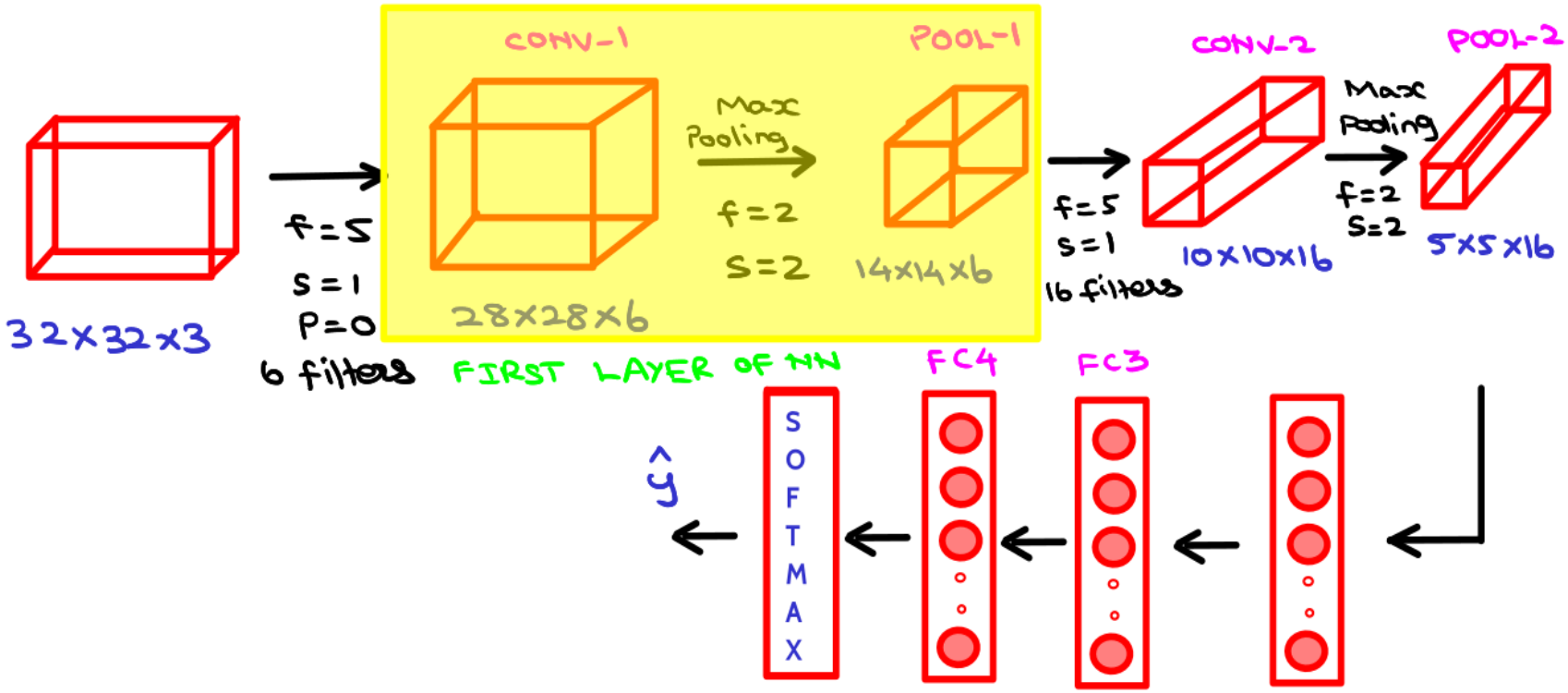

- CONV1-POOL1-CONV2-POOL2-FC3-FC4-Softmax

# Parameters in CNN



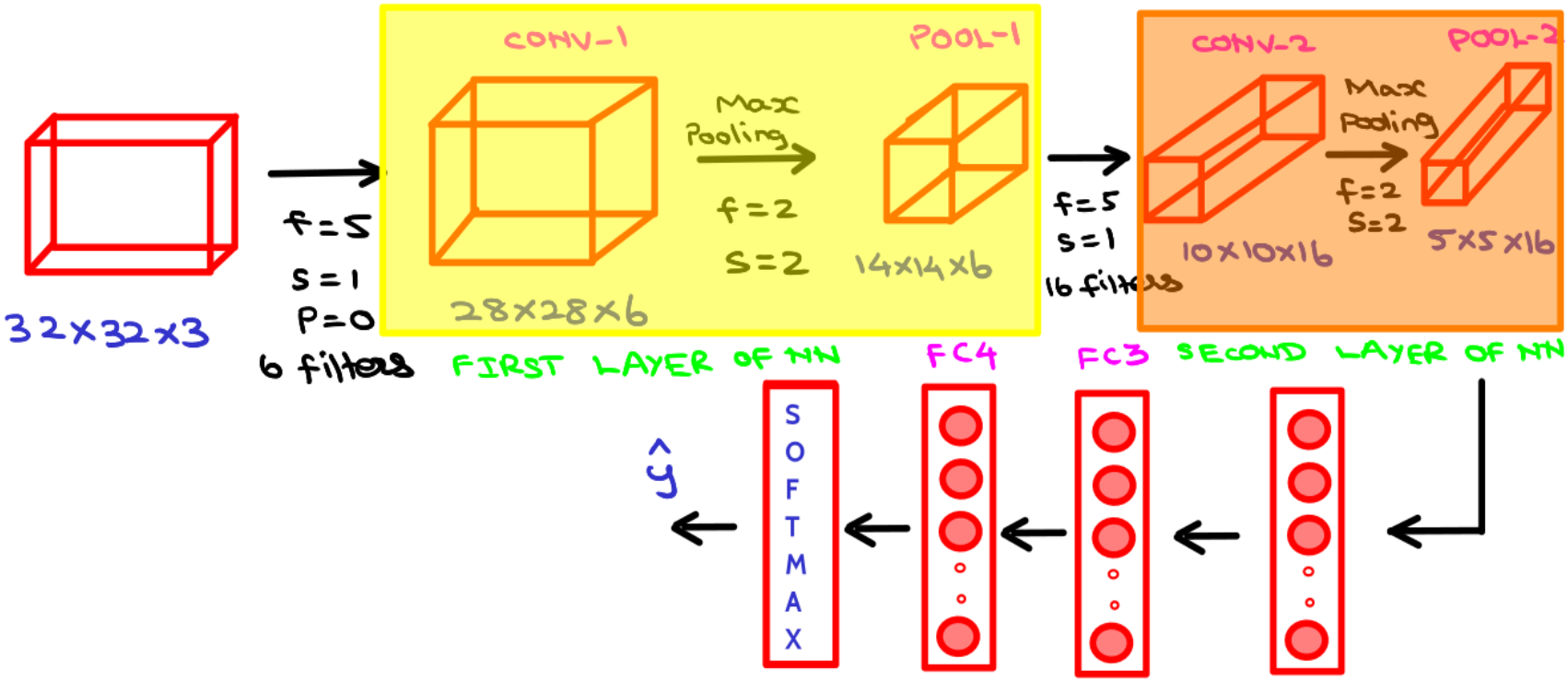

- CONV1-POOL1-CONV2-POOL2-FC3-FC4-Softmax

# Parameters in CNN



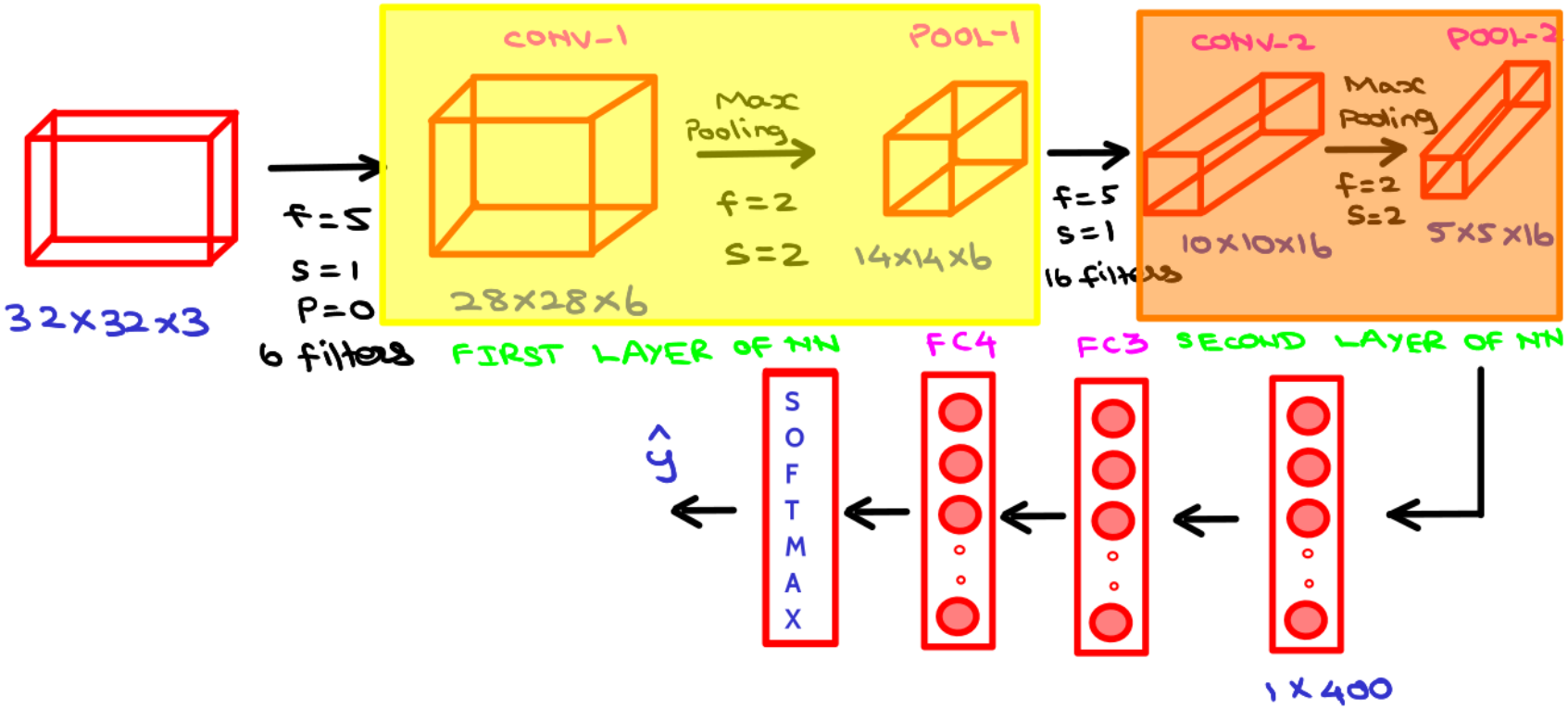

- CONV1-POOL1-CONV2-POOL2-FC3-FC4-Softmax

# Parameters in CNN



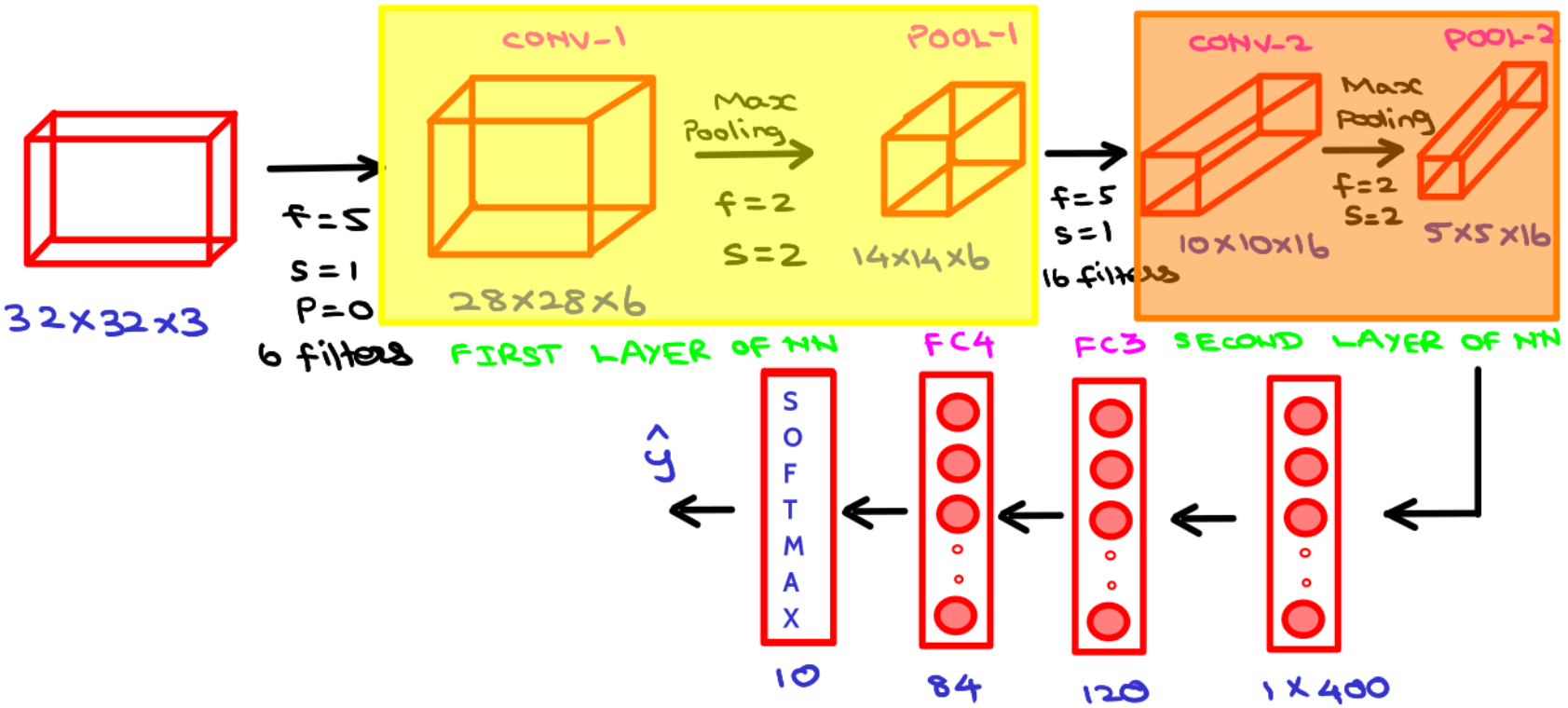

- CONV1-POOL1-CONV2-POOL2-FC3-FC4-Softmax

# Parameters in CNN




- CONV1-POOL1-CONV2-POOL2-FC3-FC4-Softmax

# Parameters in CNN




- CONV1-POOL1-CONV2-POOL2-FC3-FC4-Softmax

# Parameters in CNN

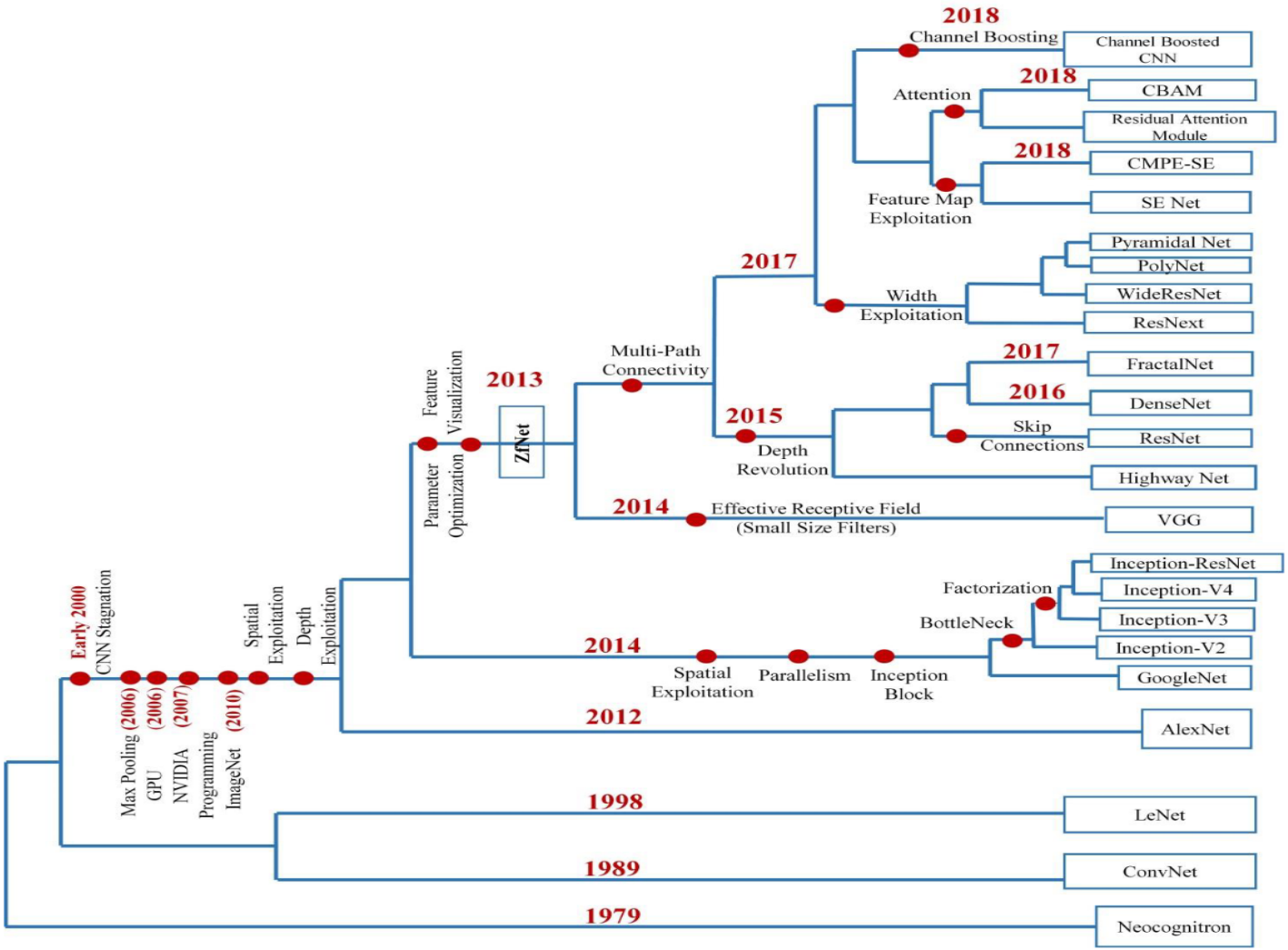
| Layer | Shape | Size | Parameter |
|---|---|---|---|
| Input layer | (32,32,3) | 3072 | 0 |
| CONV-1 (f=5, S=1, 6 filters) | | | |
| POOL-1 | | | |
| CONV-1 (f=5, S=1, 16 filters) | | | |
| POOL-2 | | | |
| FC3 | | | |
| FC4 | | | |
| Softmax | | | |

# Parameters in CNN

| Layer | Shape | Size | Parameter |
|---|---|---|---|
| Input layer | (32,32,3) | 3072 | 0 |
| CONV-1 (f=5, S=1, 6 filters) | (28,28,6) | 4704 | 456 |
| POOL-1 | (14,14,6) | 1176 | 0 |
| CONV-1 (f=5, S=1, 16 filters) | (10,10,16) | 1600 | 2416 |
| POOL-2 | (5,5,16) | 400 | 0 |
| FC3 | (120,1) | 120 | 48120 |
| FC4 | (84,1) | 84 | 10164 |
| Softmax | (10,1) | 10 | 850 |

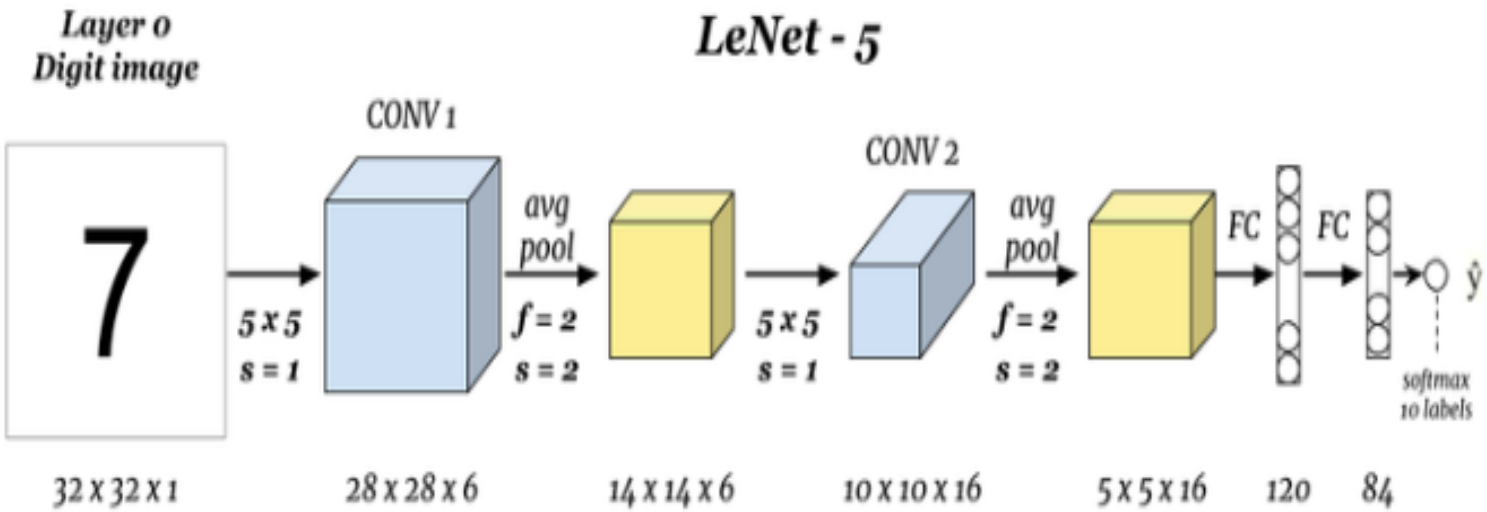
**Total number of learnable parameters: 62006**

# Evolution of image classifiers



# LeNet

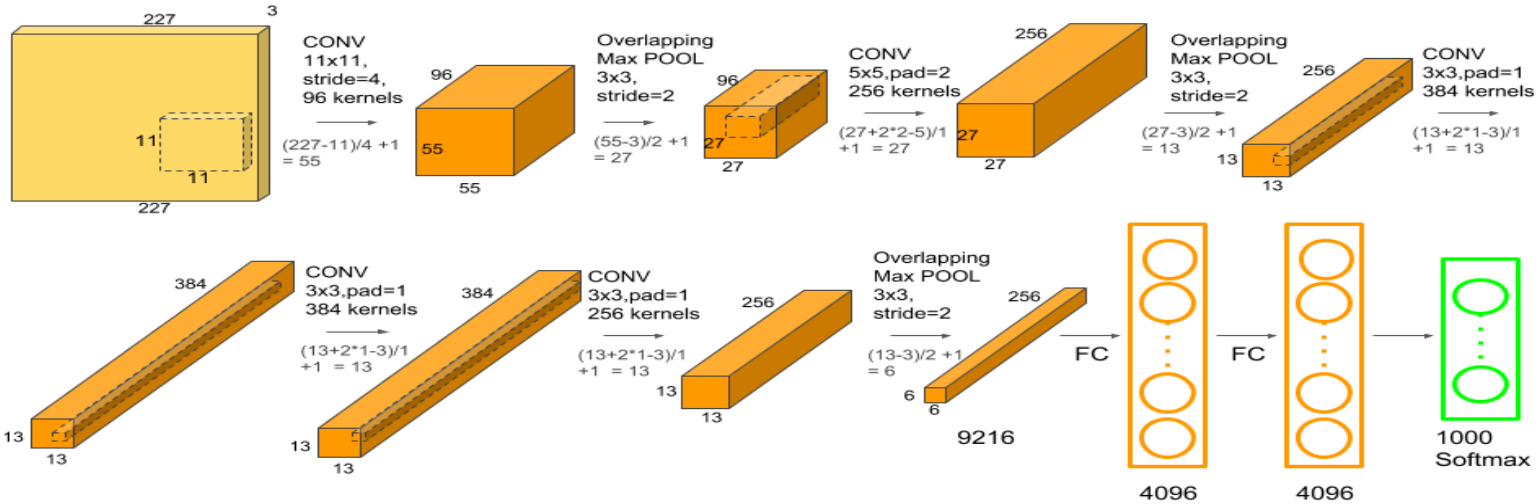
- Developed in 1998 and used for pincode digit recognition in postal service
- Around 60k parameters
- Sigmoid and Tanh were used




Lecun et al, 1998

# AlexNet

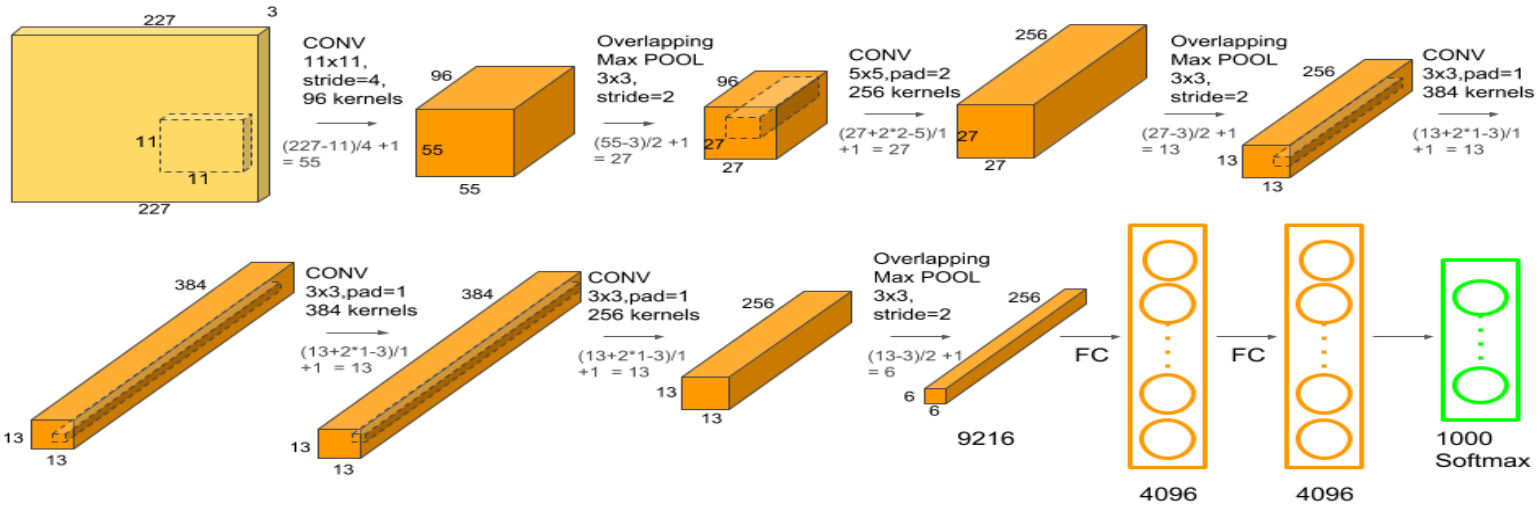
- Developed in 2012, similar to LeNet but with more layers
- 60 Million parameters
- Relu activation and local response normalization (LRN) is used




Alex K et al, 2012

# VGG16

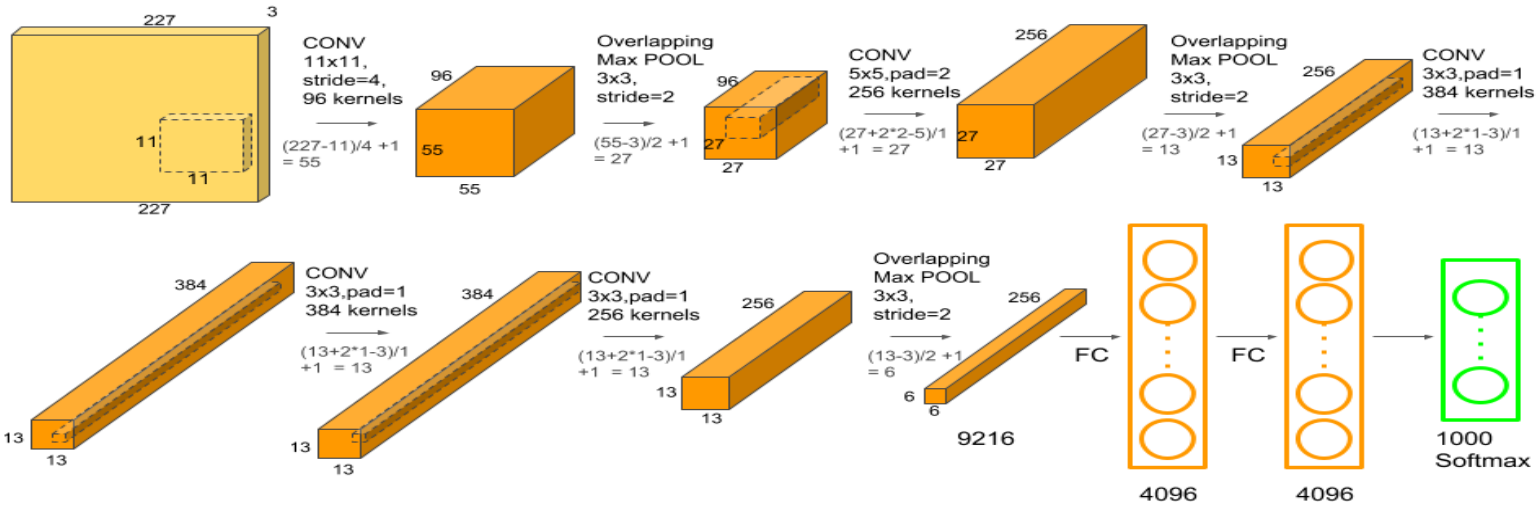
- Developed in 2015, CONVNET with 3 x 3 filters
- 16 layers with 138 Million parameters
- Number of Filters: 64 $\rightarrow$ 128 $\rightarrow$ 256 $\rightarrow$ 512
- Error rate is 17% compared to 8% in AlexNet




Simonyan and zisserman, 2015

# ResNet

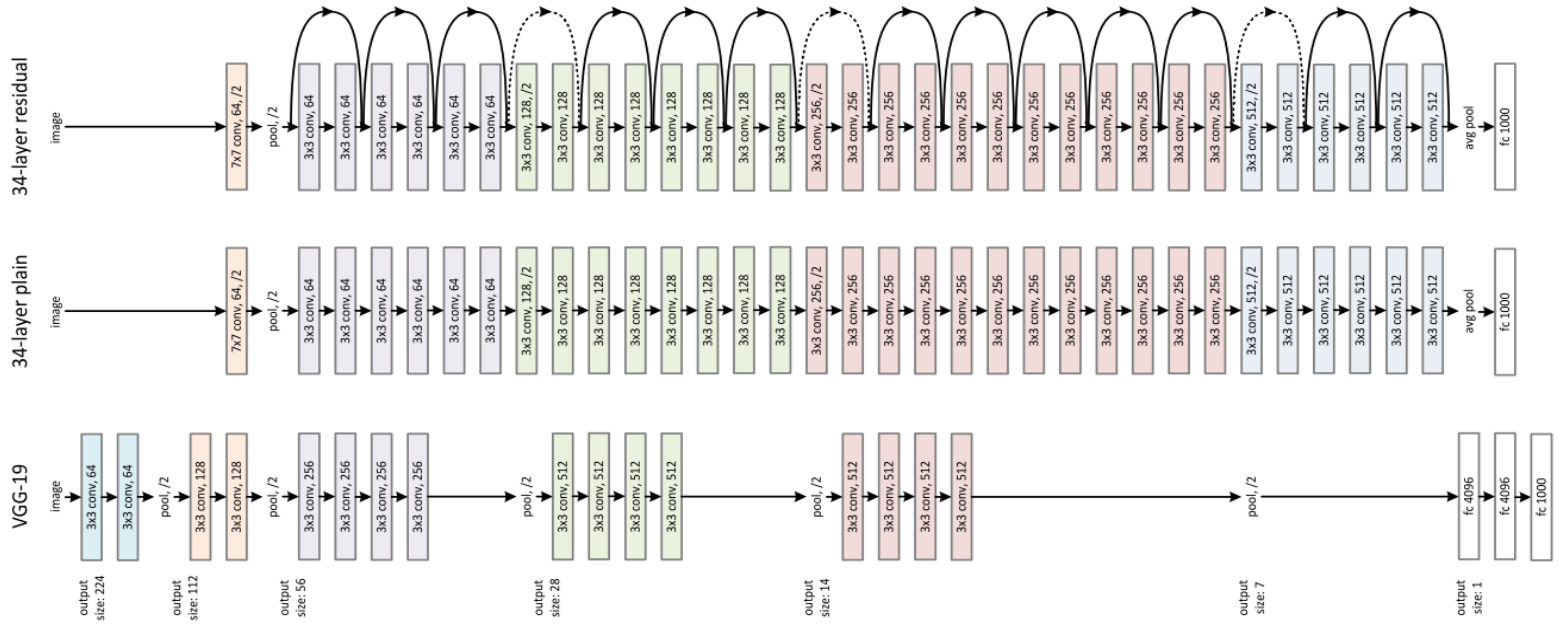
- Developed in 2015, CONVNET with 3 x 3 filters
- 16 layers with 138 Million parameters
- Number of Filters: 64 $\rightarrow$ 128 $\rightarrow$ 256 $\rightarrow$ 512
- Error rate is 17% compared to 8% in AlexNet




Simonyan and zisserman, 2015

# Residual Network

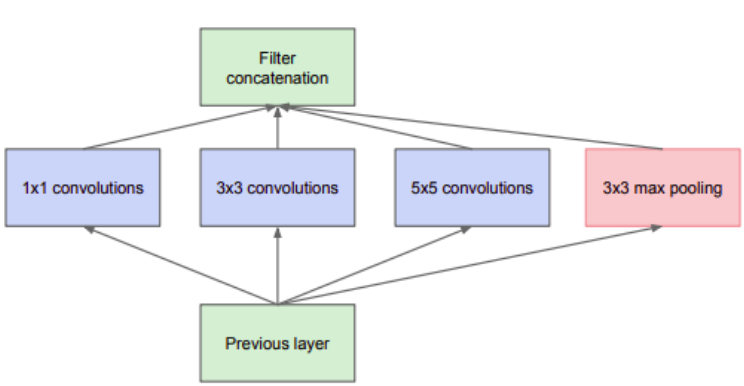
- Very deep NN has problem of vanishing and exploding gradients
- Skip connections are used which helps in training very deep layers



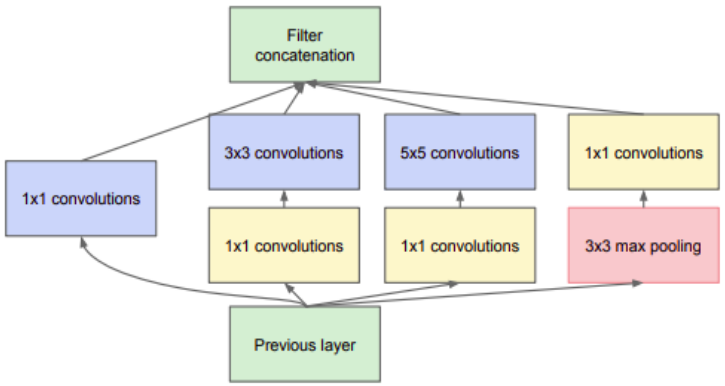

He et al, 2015

# GoogLeNet

- Homage to Yann LeCuns pioneering LeNet-5 network
- 1 x 1 convolution to shink larger volume to smaller volume
- Inception module to use different filter in same layer




(a) Inception module, naïve version

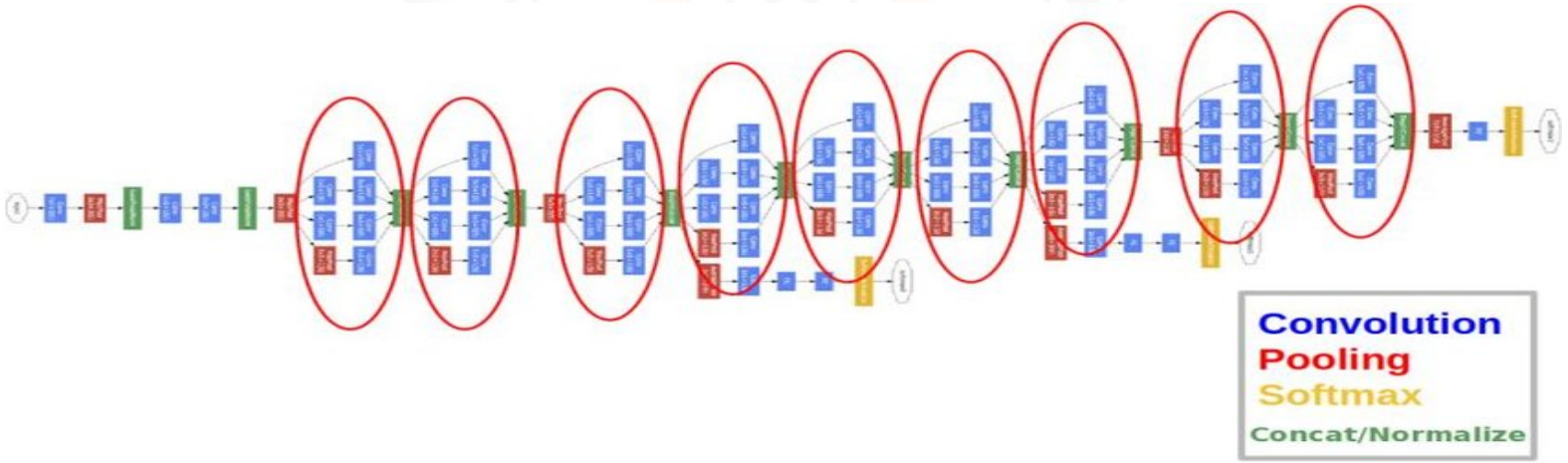



(b) Inception module with dimension reductions

Christian et al, 2015

# GoogLeNet

- 22 layers deep network with inception module
- Quality gain at modest increase in computational requirement

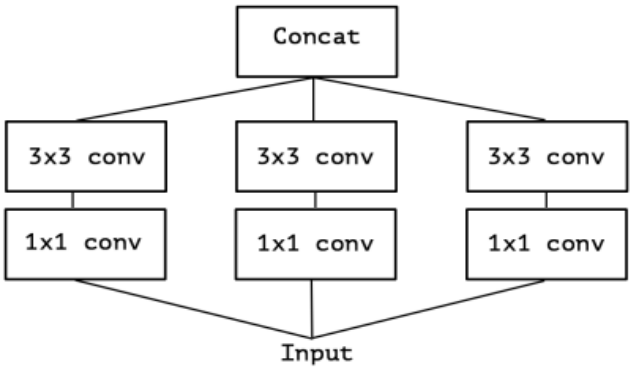



Christian et al, 2015

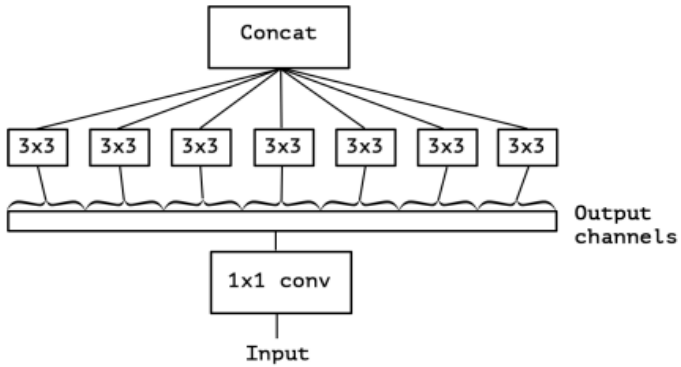
# Xception Network

- An extreme version of inception module
- Deepwise seperable convolutions in NN architecture
- Similar parameters compared to inception V3 with gain in classification performance

A simplified Inception module.




An "extreme" version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.




Choller, 2017

## Transfer learning

- CNN are basically feature extractors which learn to perform image classification
- A model trained on ImageNet Challenge have learnt to extract features from a wide range of images.
- **Can we transfer this knowledge to solve some other problems as well?**

Transfer learning is the practice of reusing the skills learnt from solving one problem to solve a new related problem

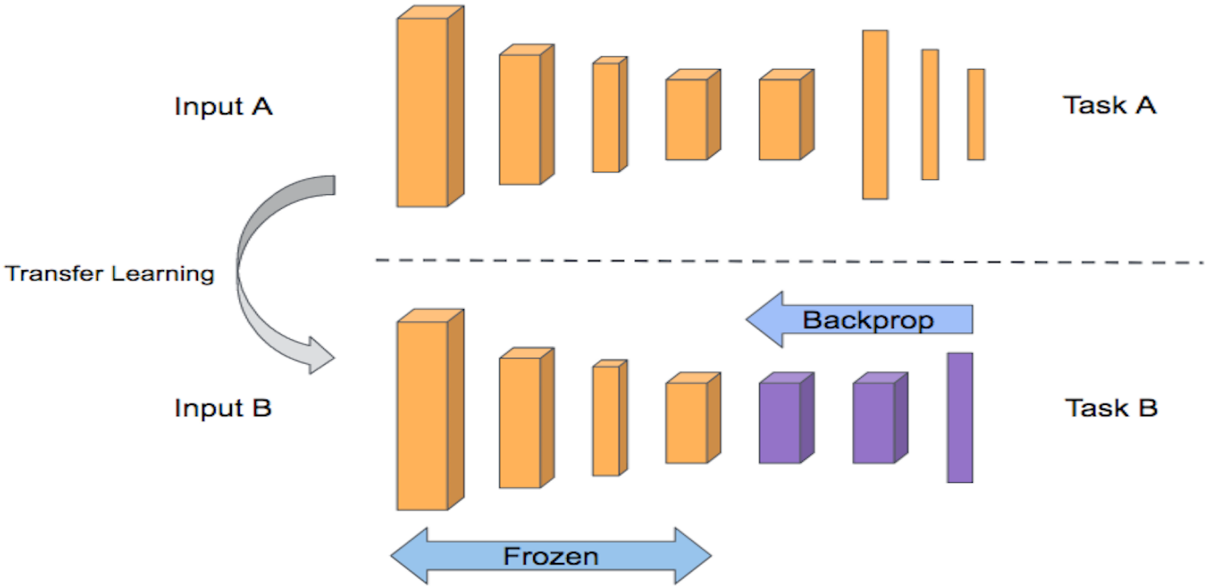
Ex: Bicycle and two-wheeler

## Main reasons for Transfer learning

- Less data for particular problem: Data abundance in one task and data chrunch in other related task
  Eg: Model for driverless car in Indian roads
- No computational power (enough data available for training)
- For most computer vision problems, pre-trained models such as AlexNet, VGGNet, GoogleNet etc works better.
  Eg: Coal mine and elephant feature extraction.
- Last few layers are only used for descrimination and there are lot of learning happened in previous layer

# Ways of Transfer learning

- Freeze weights of initial few layers and train few later layers
- Retain the entire network of all weights
- If we have lot of data, then we use open source architecture and train the whole model



# PREDICTION: IITM'S CRICKET AND CODE CHALLENGE


**Problem Statement:**

- For upcoming IPL T20 matches, participants should submit their code to predict the score at the end of 6 overs in each of the innings.
- Given: Ball-by-ball data from past 1600 innings
- Expected output: For each innings of each match, the score at the end of the 6th over.
- Input: Team batting, team bowling, name of the batsman and bowler during the six overs

# PREDICTION: IITM'S CRICKET AND CODE CHALLENGE


## Data preparation:

- From the 1600 matches, we need to seperate IPL matches with following features as input data
  - Batting team name (CSK-1, KKR-2, SRH-3 etc)
  - Bowling team name (CSK-1, KKR-2, SRH-3 etc)
  - No of wickets (1,2,3...)
  - No of bowlers used during six overs (1,2,3...)
  - Name of the ground (MAC-1, Wankade-2, kotla-3 etc.)
  - Runs scored during first six overs (30,40,65...)
  - Batting first/chasing (0/1)

# PREDICTION: IITM'S CRICKET AND CODE CHALLENGE

## Sample data file:

| Batting team | Bowling team | Ground | Number of wickets | Number of bowlers | Batting first | Runs scored |
|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 2 | 1 | 45 |
| 2 | 1 | 2 | 1 | 3 | 0 | 54 |
| 4 | 5 | 4 | 3 | 2 | 1 | 36 |
| 5 | 4 | 3 | 2 | 4 | 0 | 70 |

# PREDICTION: IITM'S CRICKET AND CODE CHALLENGE

**Models developed:**

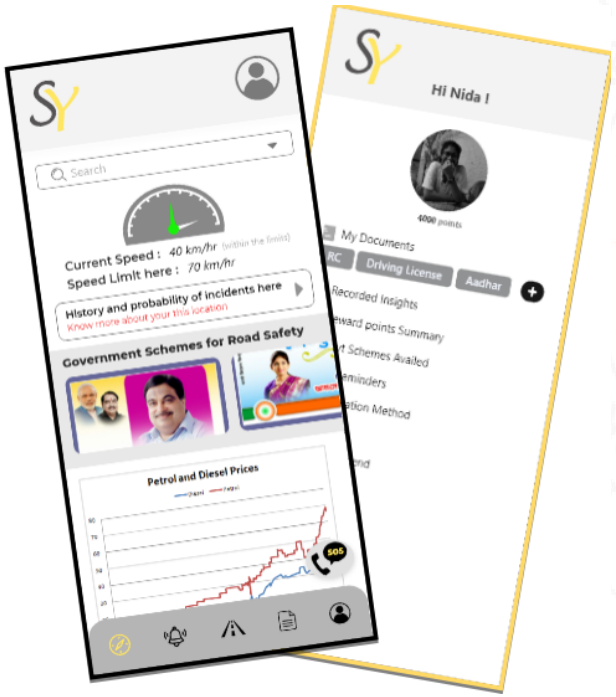
- Team specific model
- Ground specific model
- General model

Based on the cricket knowledge, we selected which model to use before each match.

We are among the top 15 teams in the list of 500 teams

Rather than type of model, features (input) plays an vital role in prediction

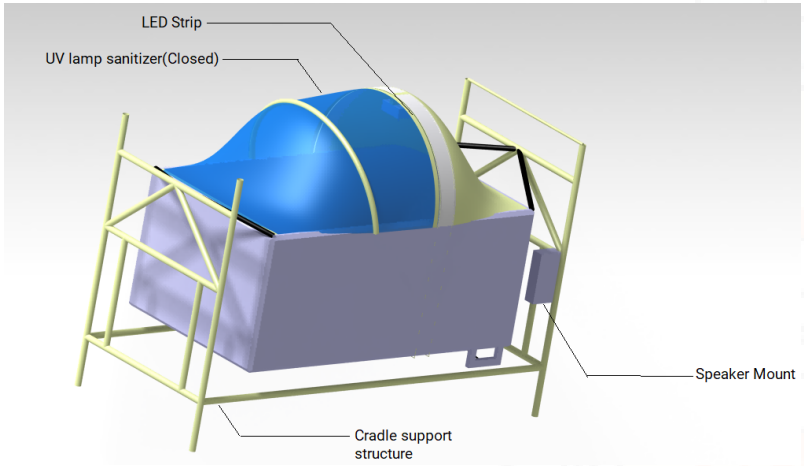
# CLASSIFICATION: ROAD SAFETY IDEATION CHALLENGE

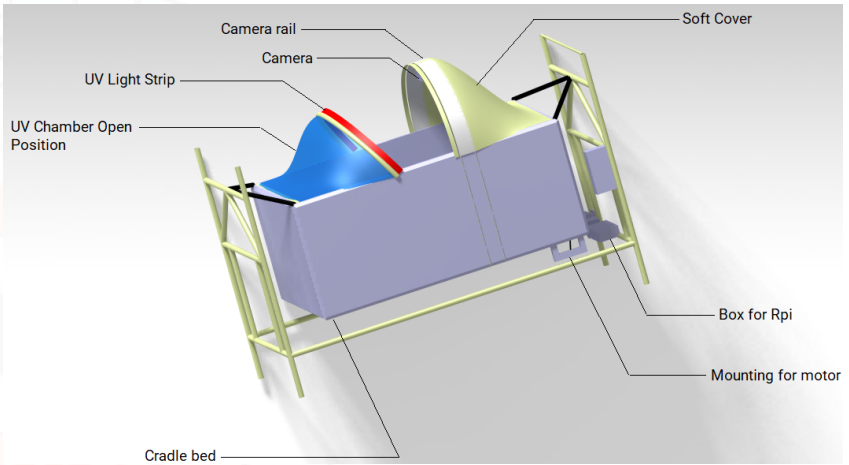



- ML based warning system for potholes
- Advanced warning system about blind spots, speed limits and upcoming traffic lights
- Checking the adherence of rider
- Collection of data for insurance claim

**Second prize has been won by the team for this ideation with cash rewards of 35,000 from GI council**

# PRODUCT DESIGN: SMART INFANT MANAGEMENT SYSTEM




## Features:

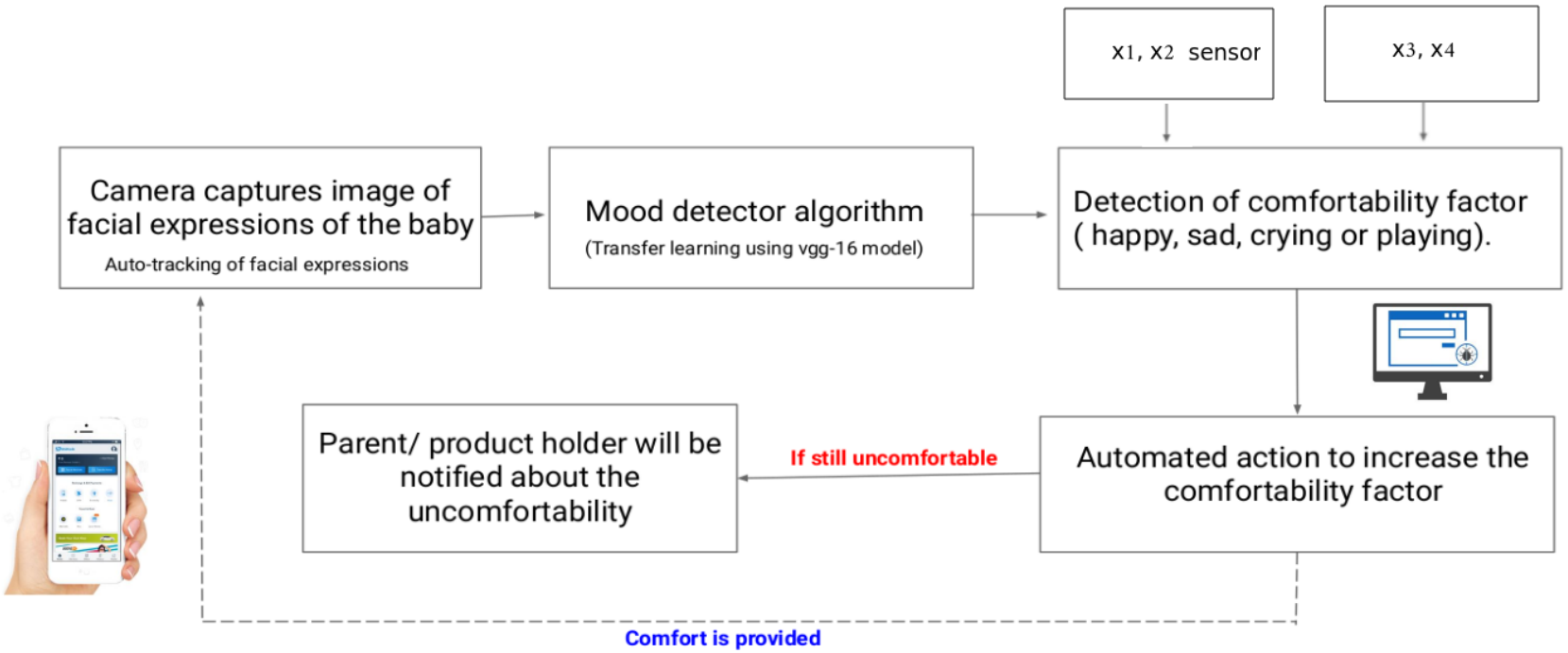
- UV santisation facility for the cardle
- ML based emotion classification
- Web and app control of cardle
- Medical data storage for analytics

## Target users:

- Infant wards
- Dual working parents

# PRODUCT DESIGN: SMART INFANT MANAGEMENT SYSTEM

## Emotion classification overview:



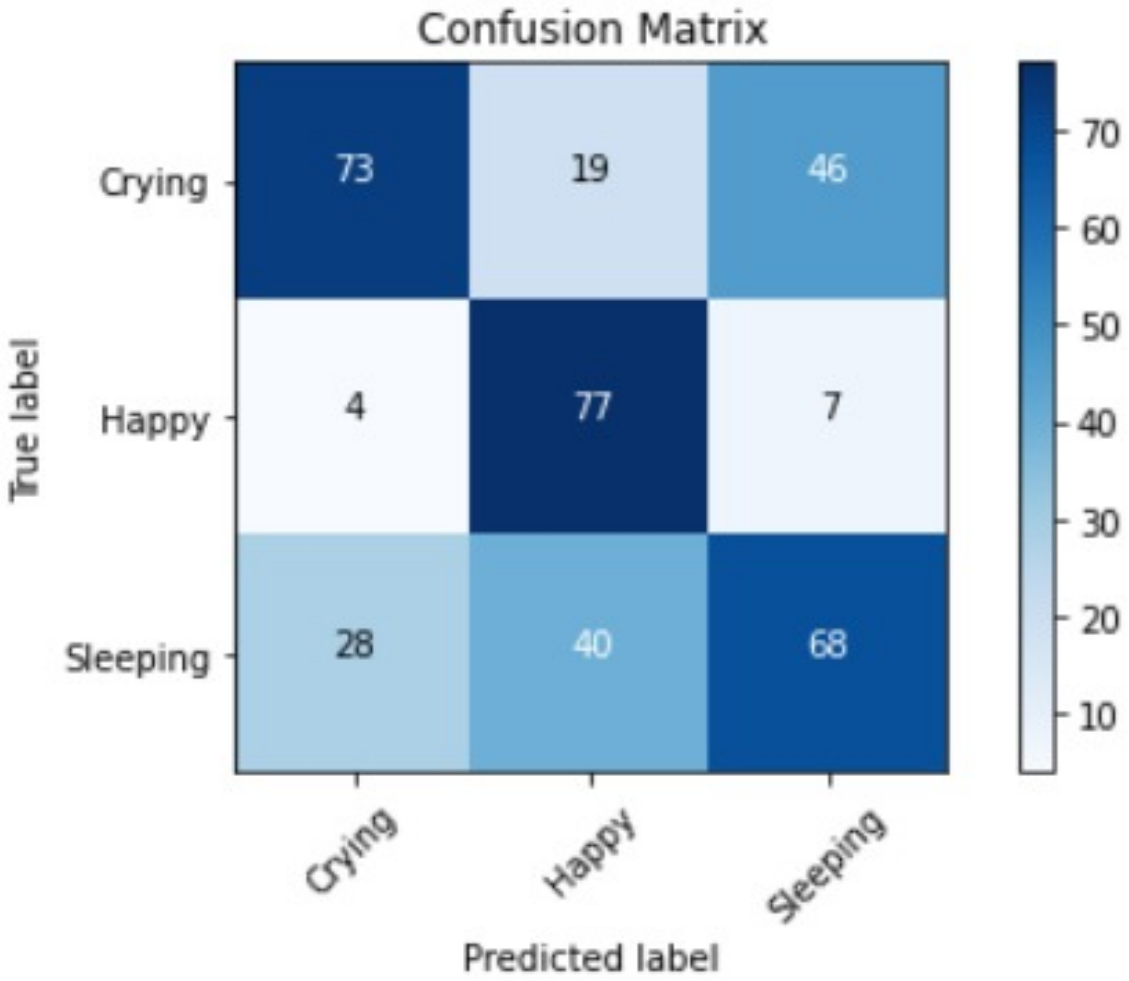
# PRODUCT DESIGN: SMART INFANT MANAGEMENT SYSTEM

## Dataset collection:

- Three categories of emotion
  - Happy
  - Sleeping
  - Crying
- Datasets are collected from google, instagram videos, youtube videos and directly approaching parents
- Dataset used
  - Training set (4.5k images)
  - Validation set (750 images)
  - Test set (362 images)

# PRODUCT DESIGN: SMART INFANT MANAGEMENT SYSTEM

## Confusion matrix for the model X:



# TEAM MEMBER PACKAGES




60+ 46 + 44 + 40 + 20 + 22 +8