

Unit-3

Convolutional Neural Networks

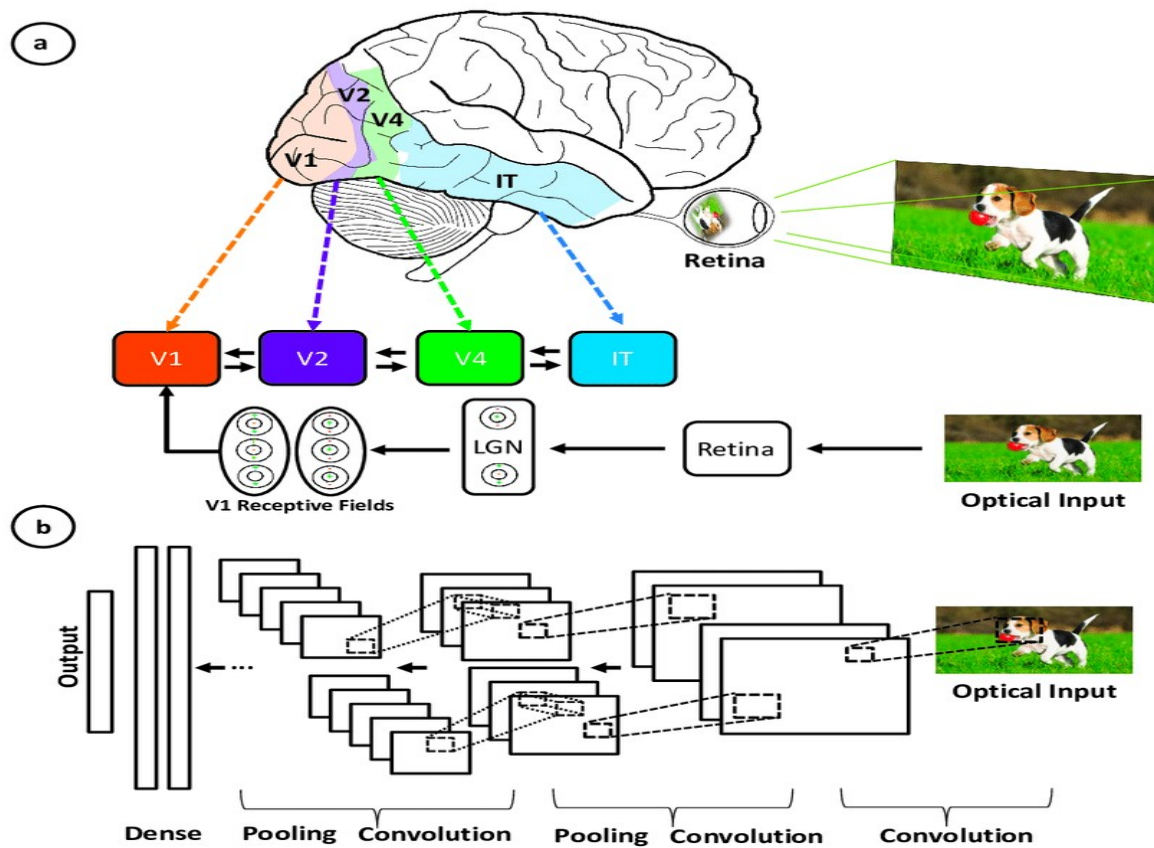
Introduction to CNN

- A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision.
- Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.
- When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text.
- Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.

- The term ‘Convolution’ in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other.
- In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image.

The importance of CNNs

- CNNs are distinguished from classic machine learning algorithms such as SVMs and decision trees by their ability to autonomously extract features at a large scale, bypassing the need for manual feature engineering and thereby enhancing efficiency.
- The convolutional layers grant CNNs their translation-invariant characteristics, empowering them to identify and extract patterns and features from data irrespective of variations in position, orientation, scale, or translation.
- A variety of pre-trained CNN architectures, including VGG-16, ResNet50, Inceptionv3, and EfficientNet, have demonstrated top-tier performance. These models can be adapted to new tasks with relatively little data through a process known as fine-tuning.
- Beyond image classification tasks, CNNs are versatile and can be applied to a range of other domains, such as natural language processing, time series analysis, and speech recognition.



- Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset. For example visual datasets like images or videos where data patterns play an extensive role.
- Before we go to the working of Convolutional neural networks (CNN), let's cover the basics, such as what an image is and how it is represented. An RGB image is nothing but a matrix of pixel values having three planes whereas a grayscale image is the same but it has a single plane.

- **Basic Structure of an Image**

1. Pixels: An image is composed of individual units called pixels. Each pixel represents a specific color or intensity.

2. Channels: Images can have one or more channels:

1. Grayscale Image: A single-channel image where each pixel value represents the intensity of light (from black to white).

2. Color Image: A three-channel image, usually representing the RGB (Red, Green, Blue) color model. Each pixel has three values corresponding to the intensity of red, green, and blue.

3. Dimensions: An image can be represented as a 2D or 3D array (tensor):

1. 2D Array: For grayscale images, the array is two-dimensional, with height and width.

2. 3D Array: For color images, the array is three-dimensional, with height, width, and depth (channels).

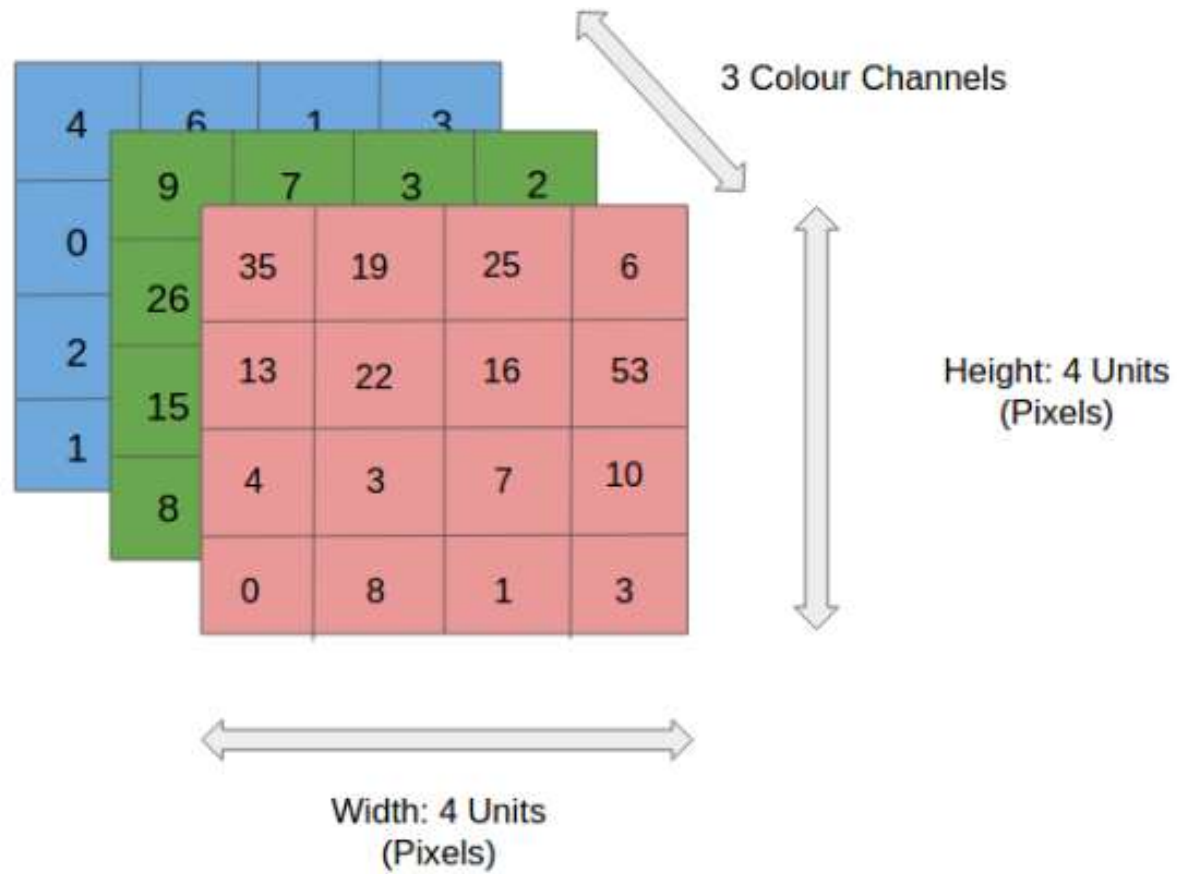
What I see



What a computer sees

08	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	91	08
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	48	04	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	53	88	30	03	49	13	36	65
52	70	95	23	04	60	11	42	69	24	68	56	01	32	56	71	37	02	36	91
22	31	16	71	51	67	63	89	41	92	36	54	22	40	40	28	66	33	13	80
24	47	32	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
32	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	78	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
88	36	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	38	25	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	34	62	99	69	82	67	59	85	74	04	36	16
20	73	35	29	78	31	90	01	74	31	49	71	48	86	81	16	23	57	05	54
01	70	54	71	83	51	54	69	16	92	33	48	61	43	52	01	89	19	67	48

- An RGB image is typically represented as a 3-dimensional array or matrix. If the image is of size $H \times W$ (Height \times Width), then the RGB image will be represented as a $H \times W \times 3$ array. The third dimension corresponds to the three color channels.
- Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300.
- In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values.
- The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at each point.





=

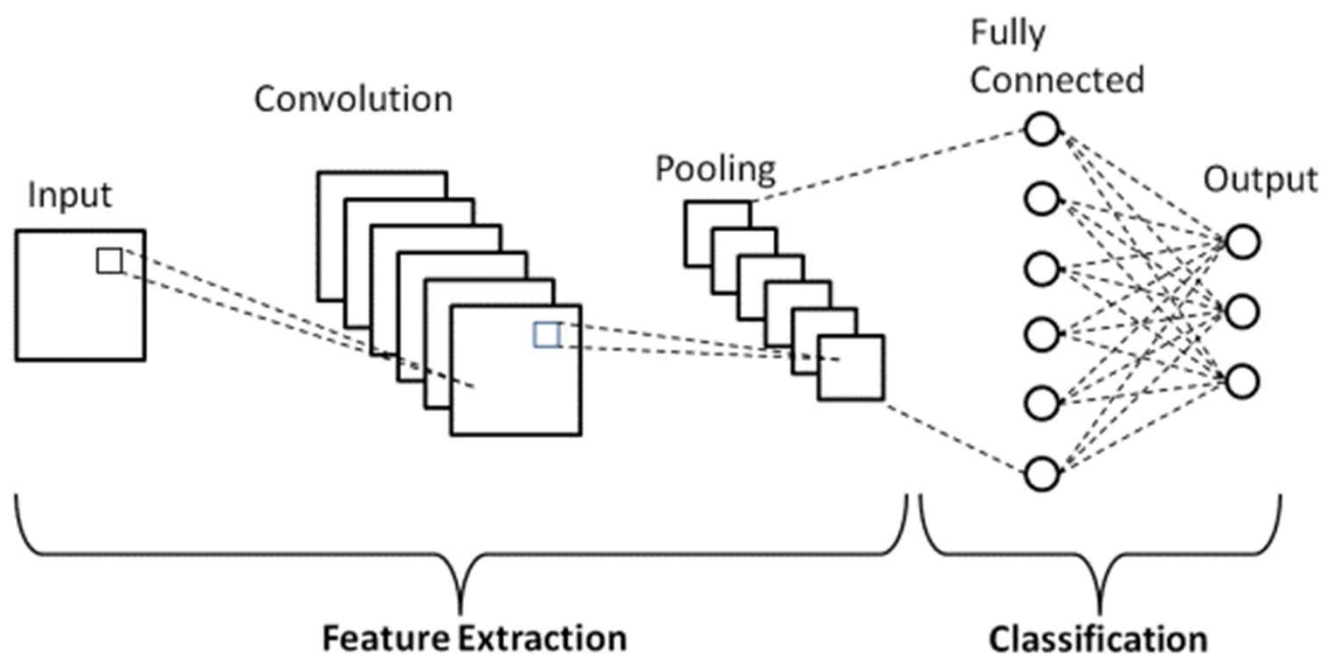


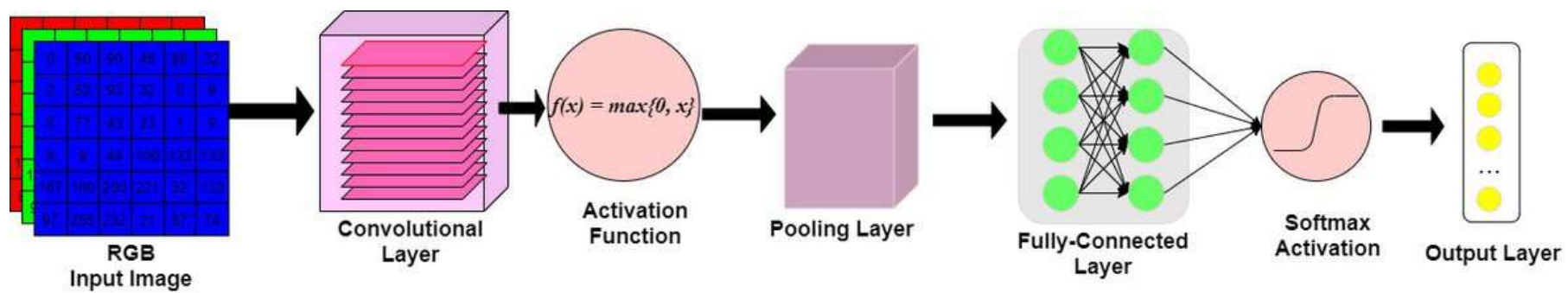
3 Channels



Basics of CNN Architecture

- Convolutional Neural Networks (CNNs) are deep learning models that extract features from images using convolutional layers, followed by pooling and fully connected layers for tasks like image classification. They excel in capturing spatial hierarchies and patterns, making them ideal for analyzing visual data.
- There are two main parts to a CNN architecture
- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction.
- The network of feature extraction consists of many pairs of convolutional or pooling layers.
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.
- This CNN model of feature extraction aims to reduce the number of features present in a dataset. It creates new features which summarizes the existing features contained in an original set of features. There are many CNN layers as shown in the basic CNN architecture with diagram.





Convolutional Layers

- Let's say you're looking at a photograph, but instead of seeing the photograph as a whole, you start by inspecting the photograph from the top left corner and begin moving to the right until you've reached the end of the photograph.
- You move down the line, and begin scanning left to right again. Every time you shift to a new portion of the photo, you take in new information about the image's contents. This is sort of how convolution works.
- Convolutional layers are the building blocks of CNNs. These layers are made of many filters, which are defined by their width, height, and depth.
- Unlike the dense layers of regular neural networks, Convolutional layers are constructed out of neurons in *3-Dimensions*. Because of this characteristic, Convolutional Neural Networks are a sensible solution for image classification.
- The convolutional layer includes **input data**, **a filter**, and **a feature map**.

- **Filters**

- Convolutional Layers are composed of weighted matrices called Filters, sometimes referred to as kernels.
- Filters slide across the image from left-to-right, taking as input only a subarea of the image (the receptive field). We will see how this local connectivity between activated nodes and weights will optimize neural network's performance as a whole.
- Filters have a width and height. Many powerful CNN's will have filters that range in size: 3 x 3, 5 x 5, in some cases 11 x 11. These dimensions determine the size of the receptive field of vision.
- Filters in a convolutional layer are like small windows or patterns that look at a portion of an image to detect specific features. Imagine you have a photo and you want to find all the edges or lines in that photo. A filter helps you do that by sliding over the image and highlighting where these edges or lines are.

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

*

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

=

6		

$7 \times 1 + 4 \times 1 + 3 \times 1 +$
 $2 \times 0 + 5 \times 0 + 3 \times 0 +$
 $3 \times -1 + 3 \times -1 + 2 \times -1$
 $= 6$

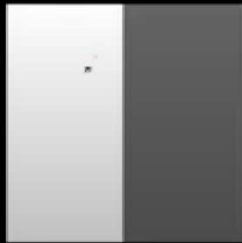
- The weights in a filter (or kernel) of a convolutional layer are not manually decided but are learned automatically during the training process of a neural network

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

- To illustrate how it works, let's assume we have a color image as input. This image is made up of a matrix of pixels in 3D, representing the three dimensions of the image: height, width, and depth.
- The filter — which is also referred to as kernel — is a two-dimensional array of weights, and is typically a **3×3** matrix.
- It is applied to a specific area of the image, and a **dot product** is computed between the input pixels and the weights in the filter.
- Subsequently, the filter shifts by a stride, and this whole process is repeated until the kernel slides through the entire image, resulting in an output array.
- The resulting output array is also known as a feature map, activation map, or convolved feature.

Convolution Operation in CNN

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

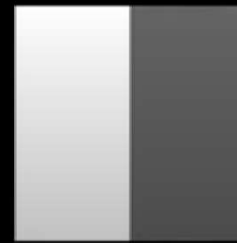


Convolution Operation in CNN

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

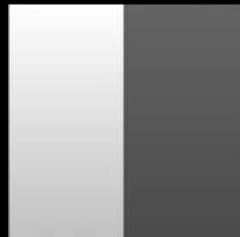
*

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



Convolution Operation in CNN

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



*

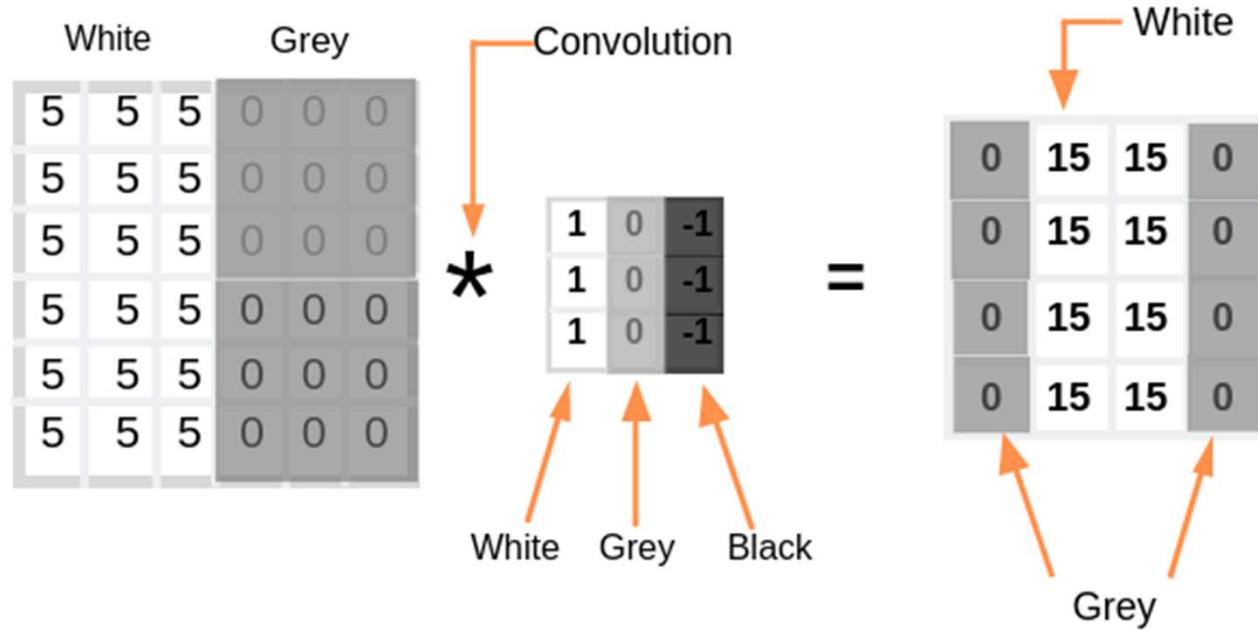
1	0	-1
1	0	-1
1	0	-1

=

0	3	3	0
0	3	3	0
0	3	3	0
0	3	3	0





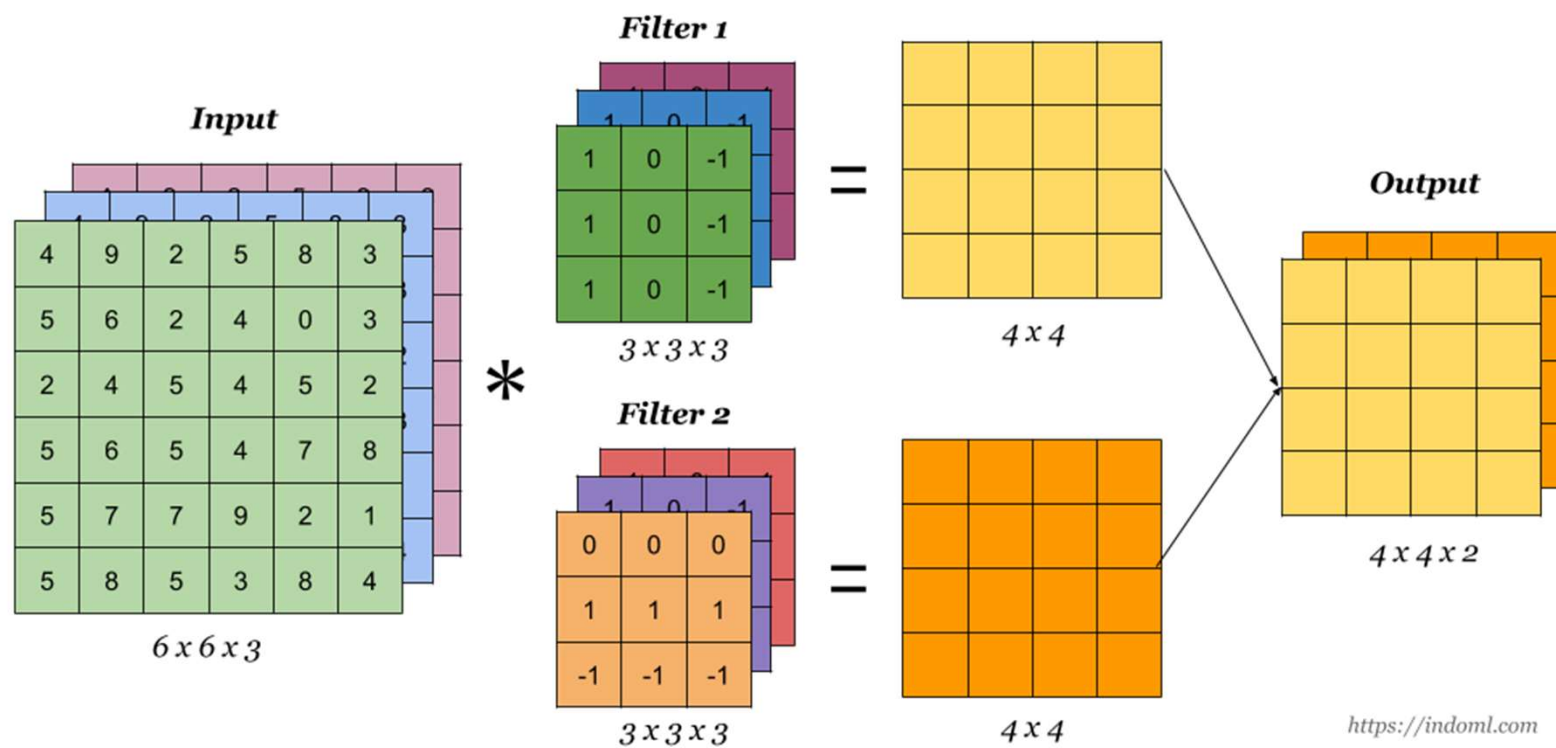


1	0	-1
1	0	-1
1	0	-1

Vertical

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

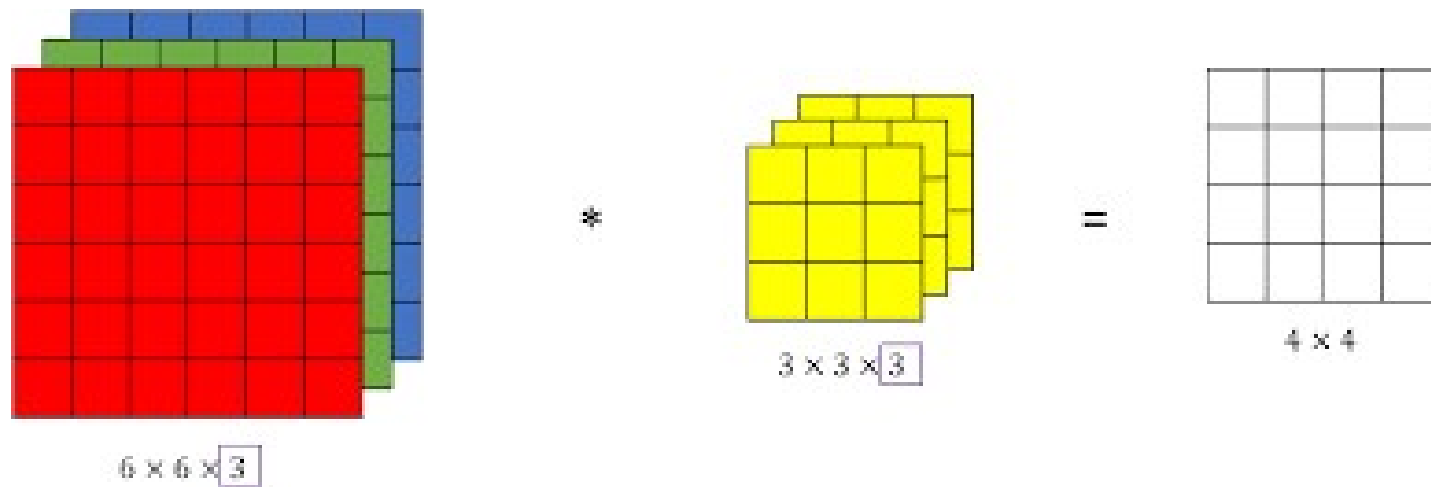
Horizontal



<https://indoml.com>

$$n \times n \times 1 \quad f \times f \times c \quad (n-f+1) \times (n-f+1) \times c$$

Convolutional operation for colour images



$n \times n \times 3$

$f \times f \times 3$

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

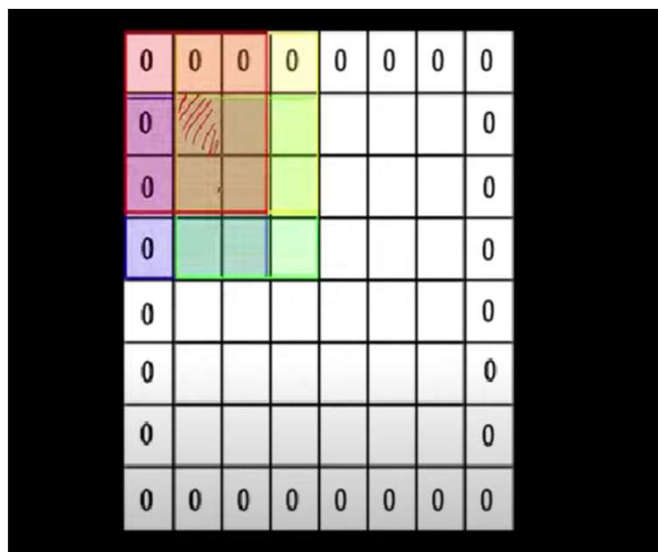
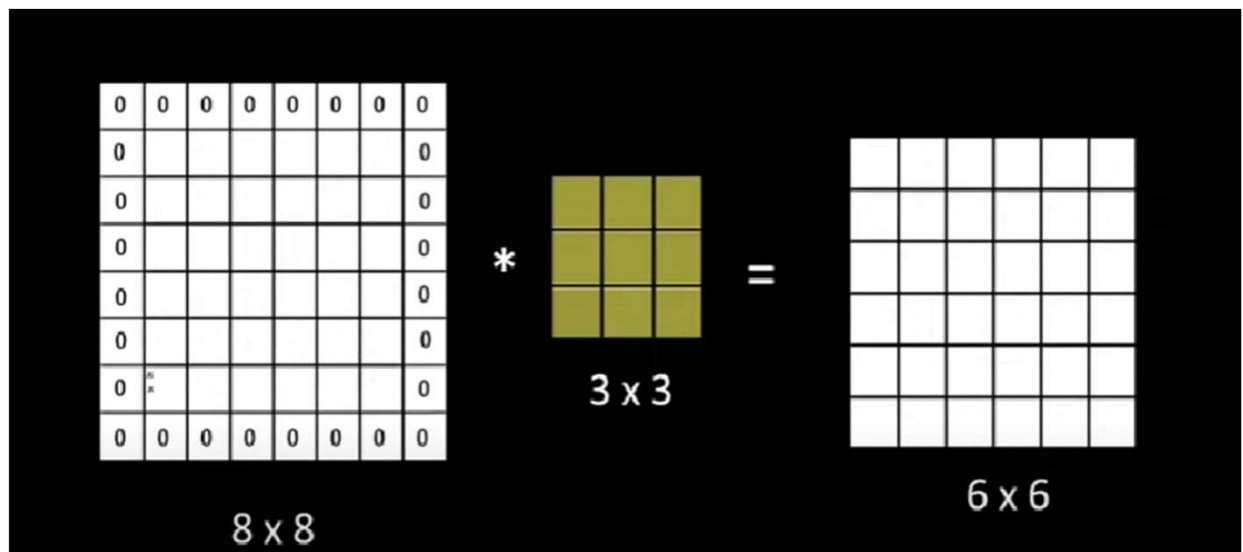
Padding

- Padding is a process of adding zeros to the input matrix symmetrically
- a zero-padding scheme will 'pad' the edges of the output volume with zeros to preserve spatial information of the image
- with an input image size of 5 x 5 and a filter size of 3 x 3. We can calculate the size of the resulting image with the following formula:

$$(n - f + 1) * (n - f + 1)$$

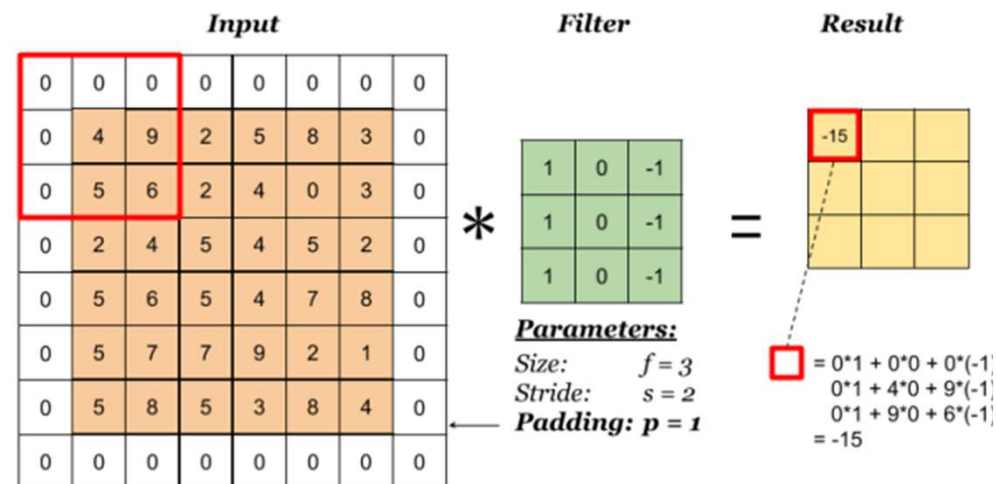
- The size of the resulting feature map would have smaller dimensions than the original input image.
- This could be detrimental to the model's predictions, as our features will continue to pass through many more convolutional layers. How do we ensure that we do not miss out on any vital information?
- When we incorporate *stride* and *padding* into the process, we ensure that our input and output volumes remain the same size - thus maintaining the spatial arrangement of visual features.

- 1.Preserve Spatial Dimensions:** Without padding, each convolution operation reduces the spatial dimensions (height and width) of the input image. This can lead to a significant reduction in size after multiple layers, which might not be desirable for certain applications where maintaining the spatial dimensions is crucial.
- 2.Edge Information:** Pixels at the edges and corners of an image often contain important information. Without padding, the filters would only pass over these pixels fewer times compared to the central pixels. This means the edge information would be less influential in the output feature map, potentially leading to loss of valuable details.
- 3.Control Output Size:** Padding helps control the output size of the convolutional layers. By appropriately padding the input, we can ensure the output feature map has the desired dimensions, which can be crucial for designing certain types of network architectures.



- Let say 'p' is the padding
- Initially(without padding)
- $(N \times N) * (F \times F) = (N-F+1) \times (N-F+1) \text{---(1)}$
- After applying padding
- If we apply filter $F \times F$ in $(N+2p) \times (N+2p)$ input matrix with padding, then we will get output matrix dimension $(N+2p-F+1) \times (N+2p-F+1)$. As we know that after applying padding we will get the same dimension as original input dimension $(N \times N)$.

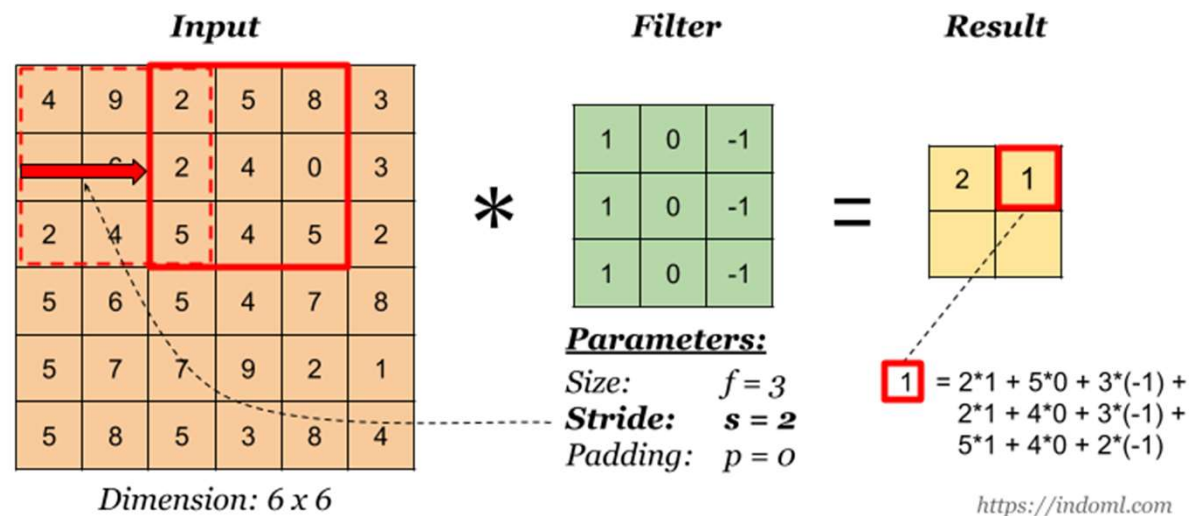
- $(N+2p-F+1) \times (N+2p-F+1)$ equivalent to $N \times N$
- $N+2p-F+1 = N$ ---(2)
- $p = (F-1)/2$ ---(3) The equation (3) clearly shows that Padding depends on the dimension of filter.



- There are three different kinds of padding:
- **Valid padding:** Also known as *no padding*. In this specific case, the last convolution is dropped if the dimensions do not align.
- **Same padding:** This padding ensures that the output layer has the exact same size as the input layer.
- **Full padding:** This kind of padding increases the size of the output by adding zeros to the borders of the input matrix.

Stride

- the distance the filter moves at a time. A filter with a stride of 1 will move over the input image, 1 pixel at a time.
- Stride governs how many cells the filter is moved in the input to calculate the next cell in the result.



Stride = 2

6x7

1	6	9	10	2	8	5
2	5	1	8	4	2	4
3	7	4	9	10	3	7
9	8	3	6	7	9	3
8	0	9	4	7	2	1
9	10	12	6	9	8	0

$$\left\lfloor \frac{n-f}{s} + 1 \right\rfloor$$

stride
floor

$$\lfloor 2.7 \rfloor = 2$$

$n_1 \times n_2$
fxf

$$\left\lfloor \frac{n_1-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_2-f}{s} + 1 \right\rfloor$$

$$\left\lfloor \frac{6-3}{2} + 1 \right\rfloor \times \left\lfloor \frac{7-3}{2} + 1 \right\rfloor = \lfloor 2.5 \rfloor \times \lfloor 3 \rfloor = 2 \times 3$$

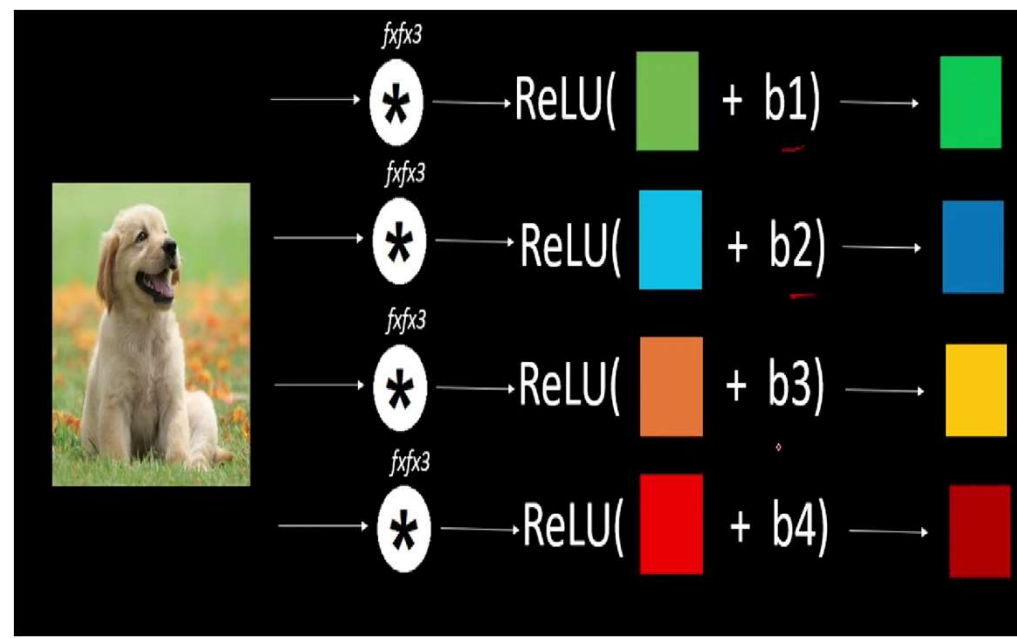
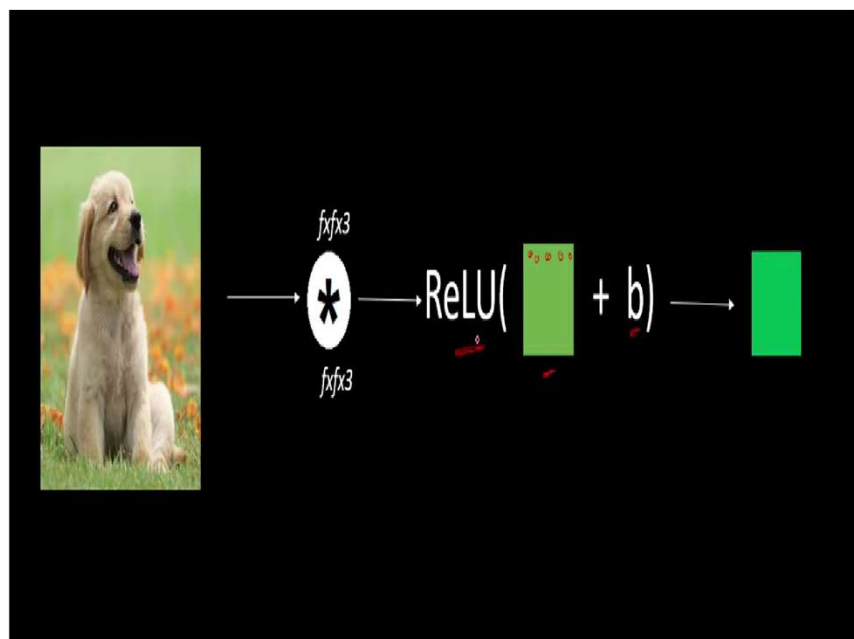
With both stride and padding

$$n^{[l]} = \left\lfloor \frac{n^{[l-1]} + 2p^{[l-1]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

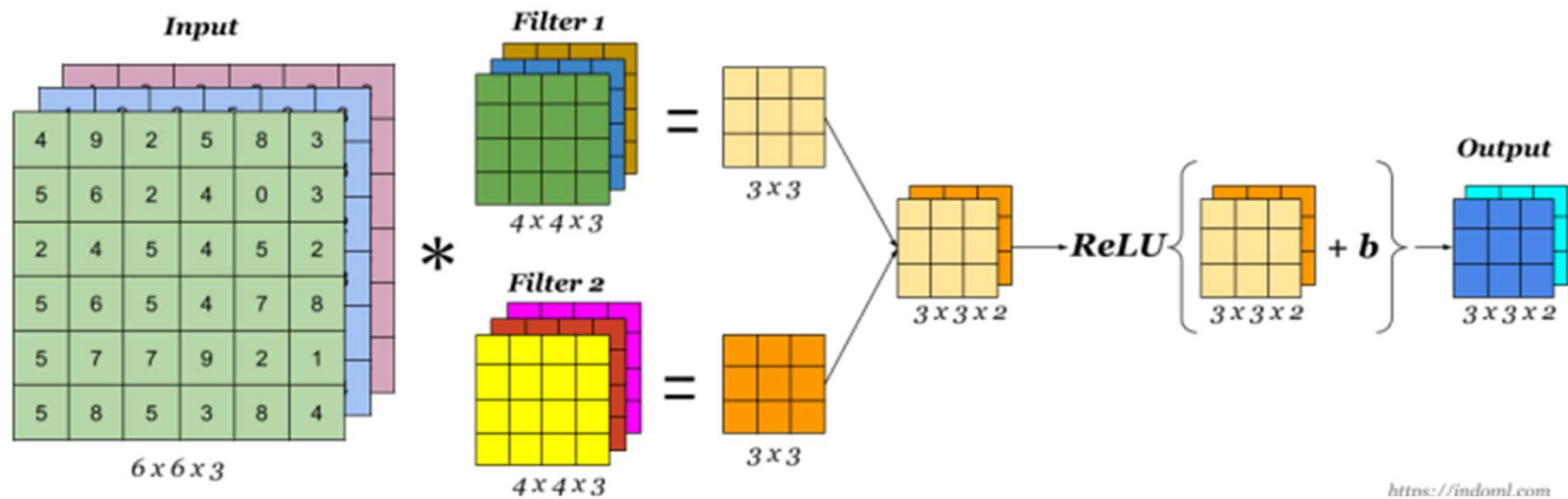
- It is important to note that the weights in the filter remain fixed as it moves across the image. The weight values are adjusted during the training process due to backpropagation and gradient descent.
- Besides the weights in the filter, we have other three important parameters that need to be set before the training begins:
- **Number of Filters:** This parameter is responsible for defining the depth of the output. If we have three distinct filters, we have three different feature maps, creating a depth of three.
- **Stride:** This is the distance, or number of pixels, that the filter moves over the input matrix.
- **Zero-padding:** This parameter is usually used when the filters do not fit the input image. This sets all elements outside the input matrix to zero, producing a larger or equally sized output.

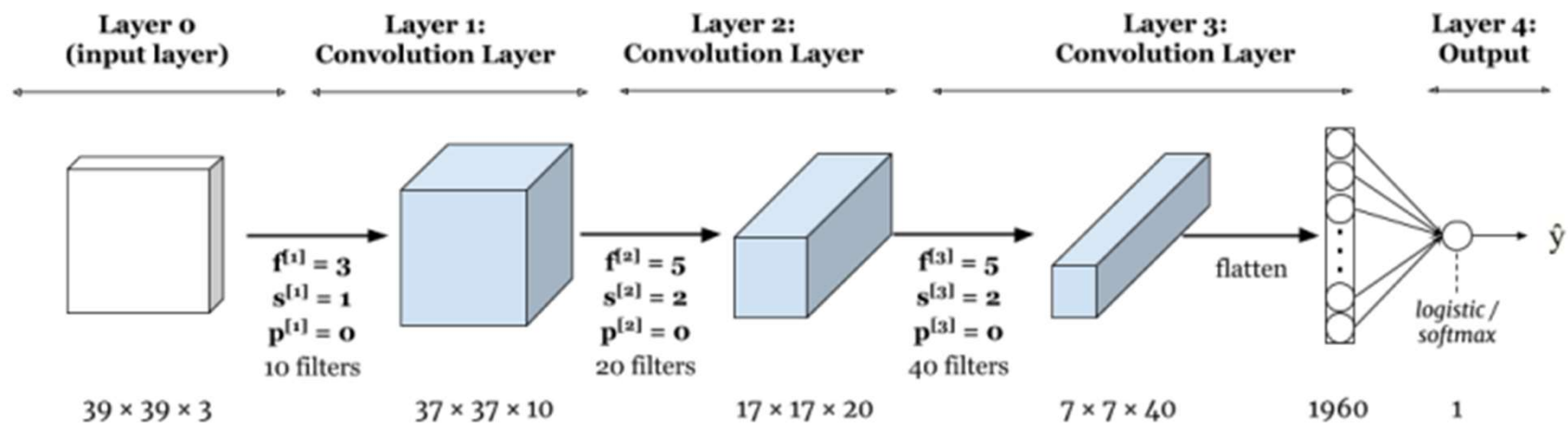
- After each convolution operation, we have the application of a ***Rectified Linear Unit (ReLU)*** function, which transforms the feature map and introduces nonlinearity.
- As mentioned earlier, the initial convolutional layer can be followed by additional convolutional layers.
- The subsequent convolutional layers can see the pixels within the receptive fields of the prior layers, which helps to extract and interpret additional patterns.

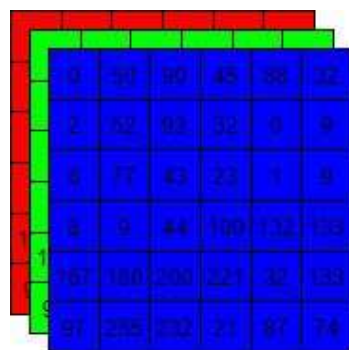
- The output volume of the Conv. layer is fed to an elementwise activation function, commonly a Rectified-Linear Unit (ReLU).
- The ReLU layer will determine whether an input node will 'fire' given the input data.
- This 'firing' signals whether the convolution layer's filters have detected a visual feature.
- A ReLU function will apply a $\max(0, x)$ function, thresholding at 0.
- The dimensions of the volume are left unchanged.



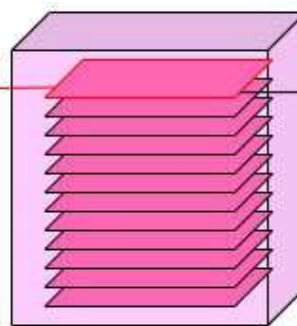
A Convolution Layer



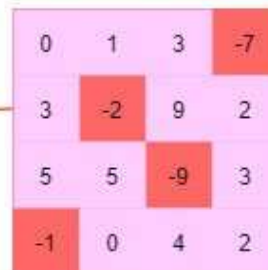




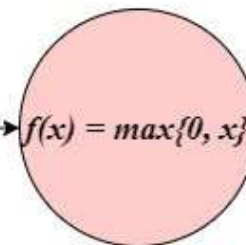
RGB
Input Image



Convolutional
Layer



Feature Map



Activation
Function

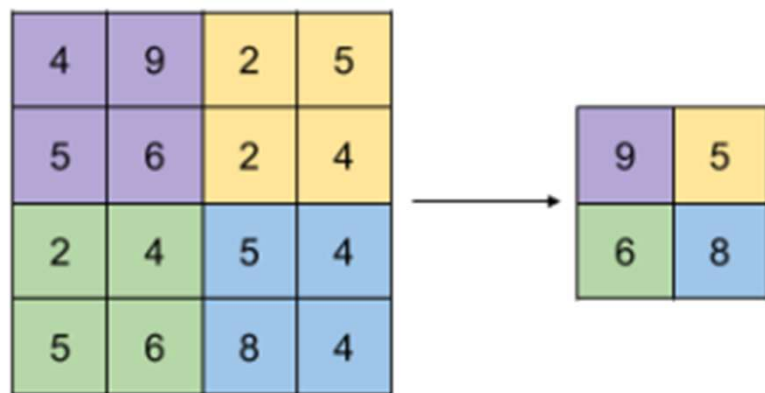


Rectified Feature
Map

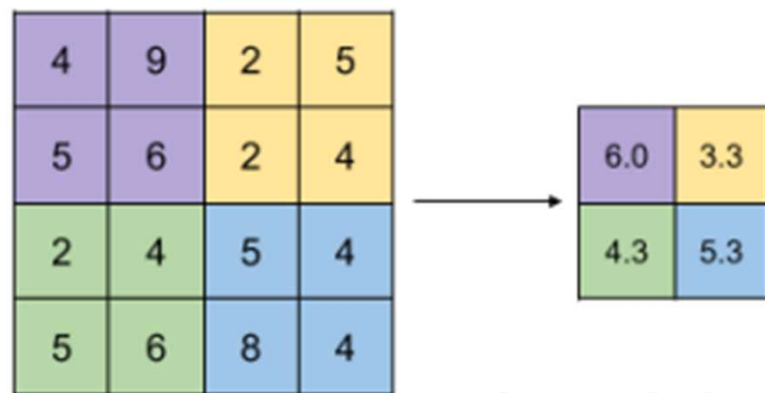
Pooling Layer

- The pooling layer is responsible for reducing the dimensionality of the input. It also slides a filter across the entire input — without any weights — to populate the output array. We have two main types of pooling:
 - • **Max Pooling:** As the filter slides through the input, it selects the pixel with the highest value for the output array.
 - • **Average Pooling:** The value selected for the output is obtained by computing the average within the receptive field.

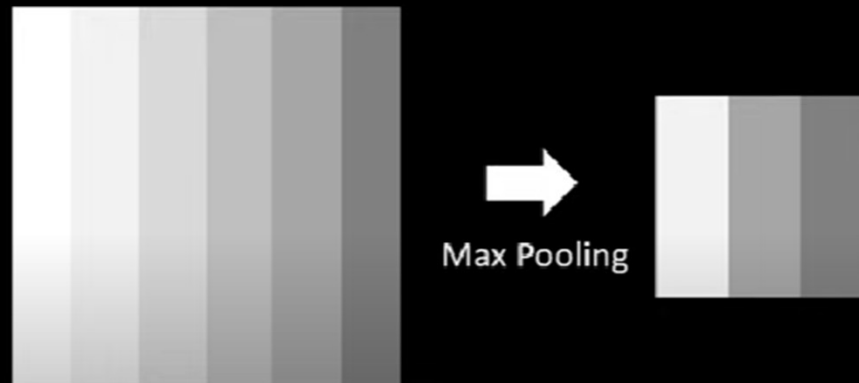
Max Pooling



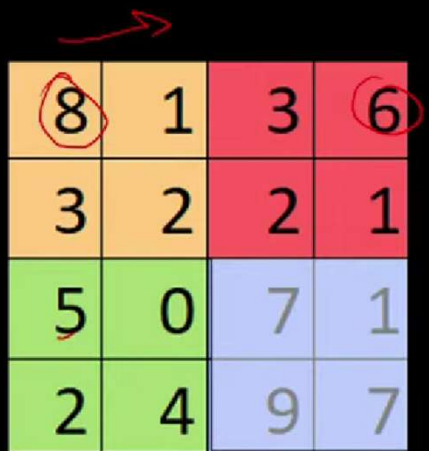
Avg Pooling



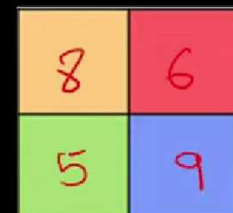
Max Pooling Layer in CNN



- Interesting properties of pooling layer:
- Stride usually depends on filter size (if filter is 2×2 , than stride can be 2)
- it has hyper-parameters:
 - **size** (f)
 - **stride** (s)
 - **type** (max or avg)

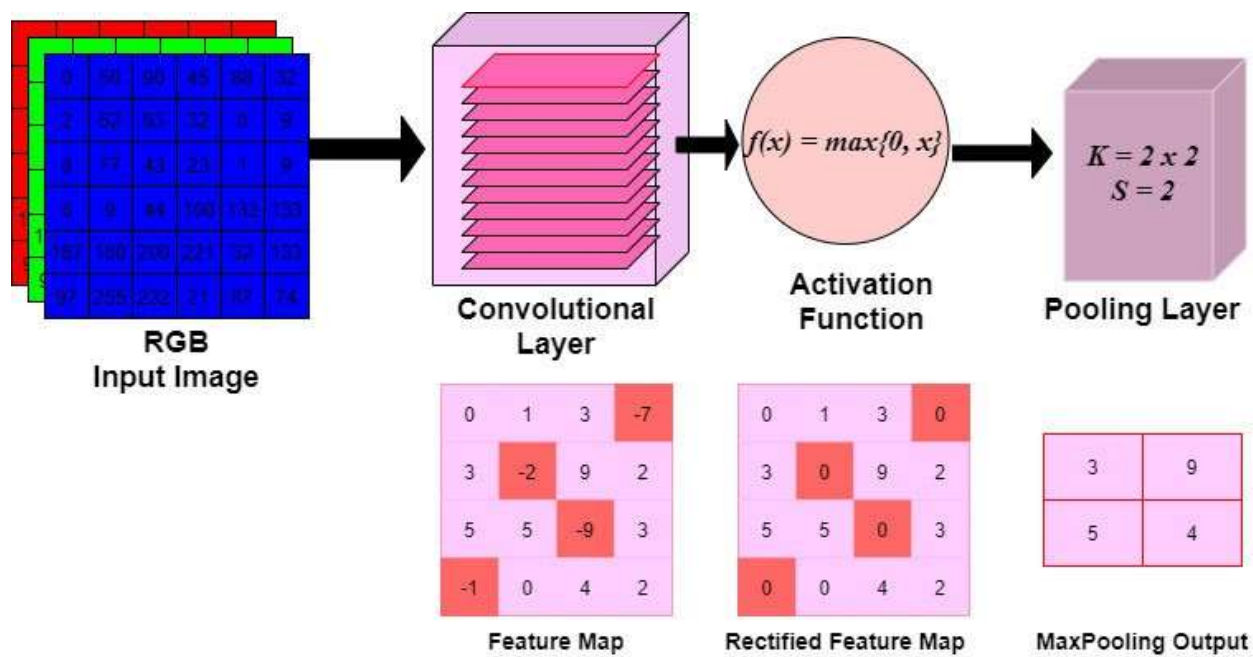


8	1	3	6
3	2	2	1
5	0	7	1
2	4	9	7



8	6
5	9

Filter of size 2x2
Stride = 2



Why Max Pooling?

1. Reduce image size, thus reduce computational cost

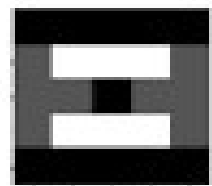
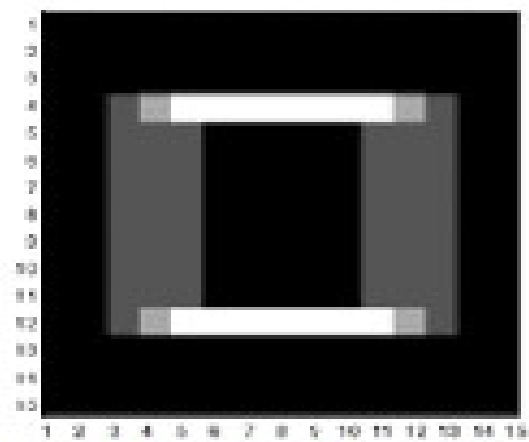
2. Enhances Features

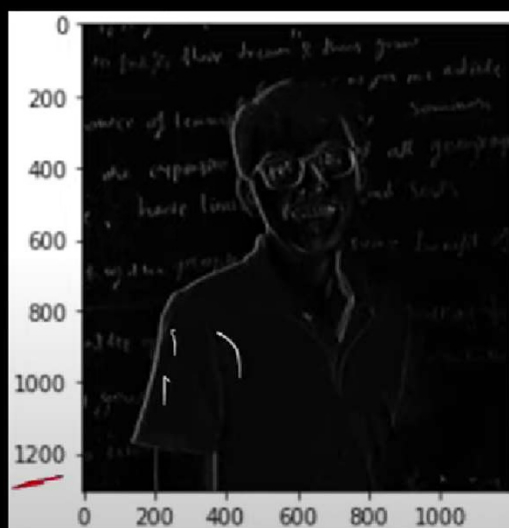
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5

Input

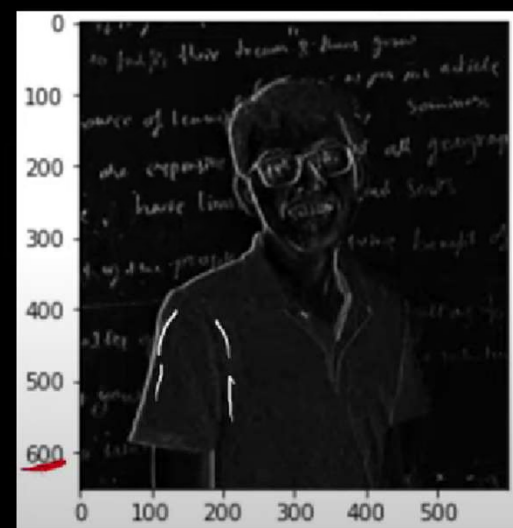
0.1	0.3	0.5
0.1	0.3	0.5
0.1	0.3	0.5

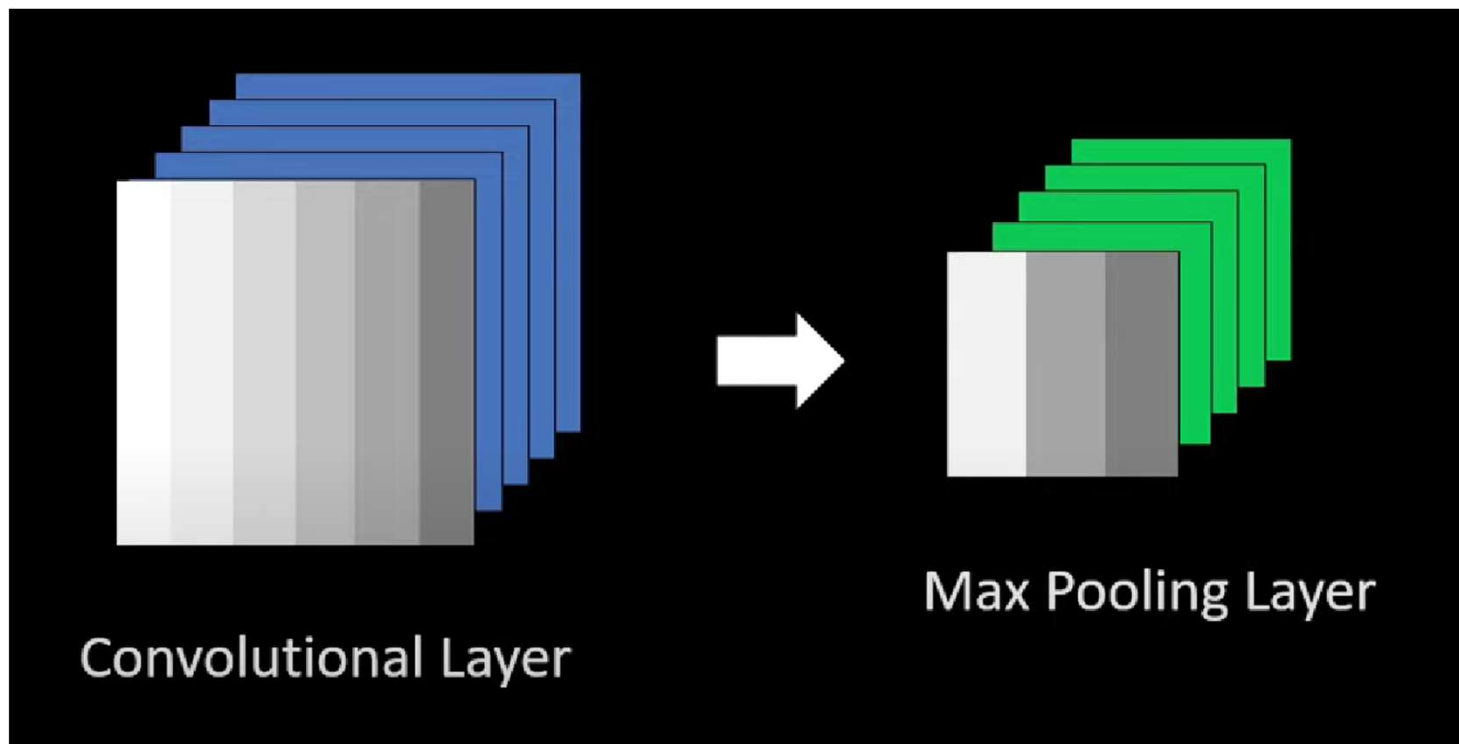
Output





Max Pooling

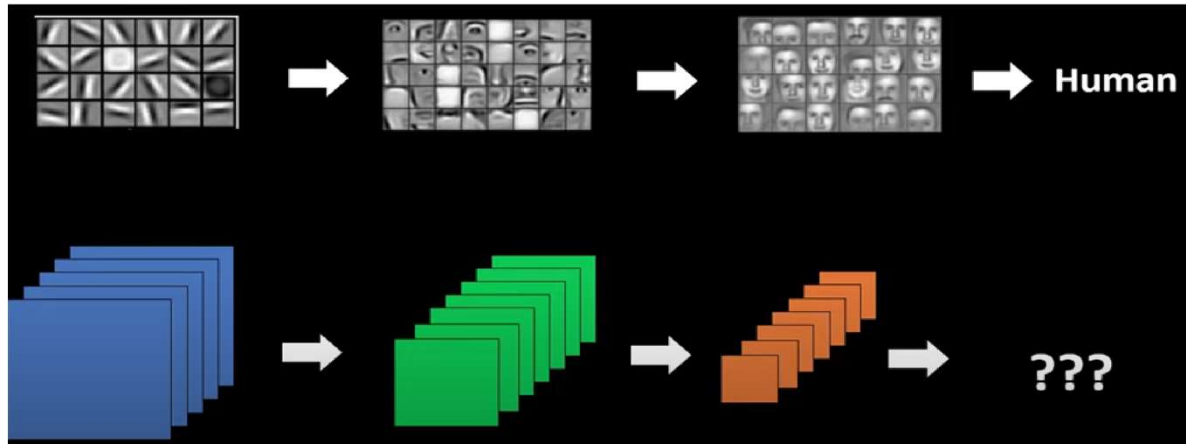


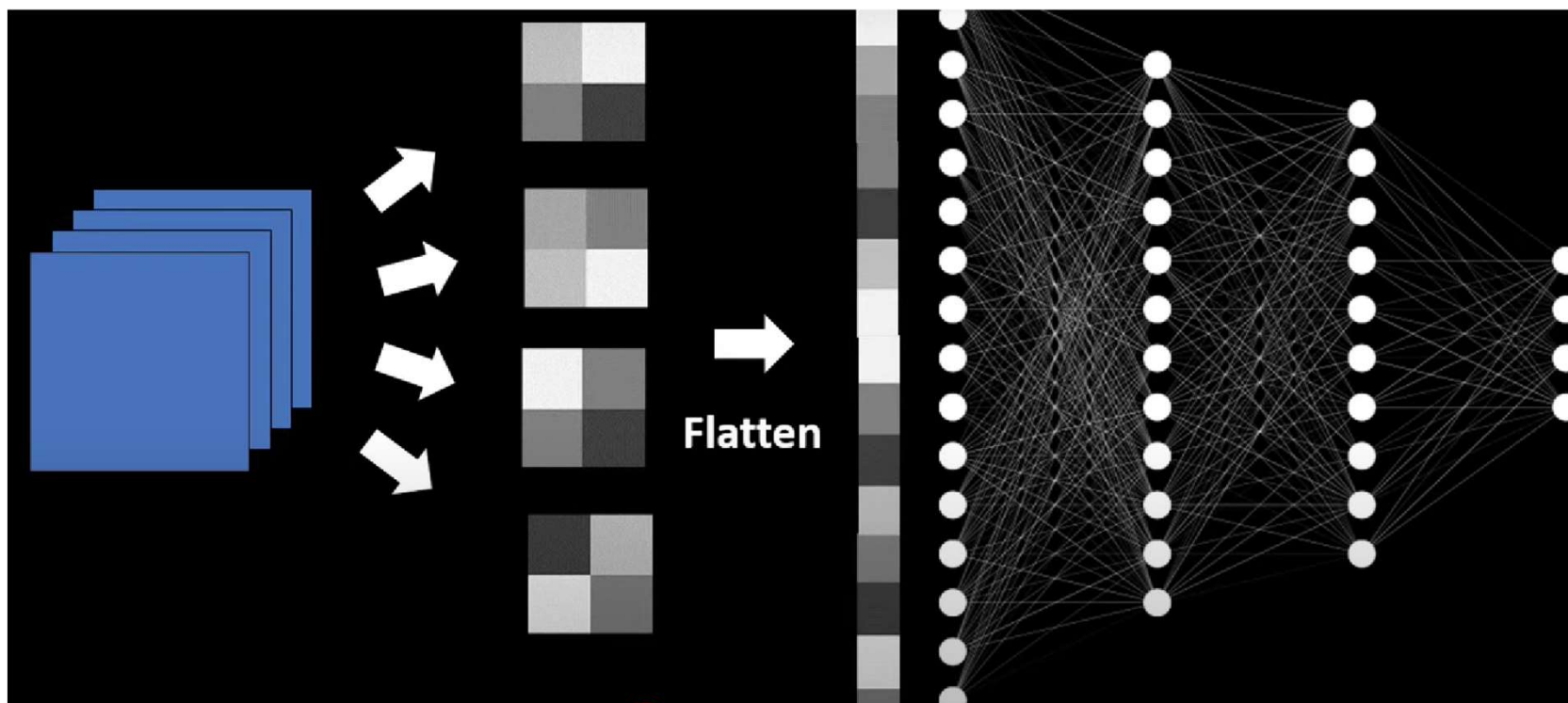


- Number of images generated as output at max pooling layer are same as number of inputs from convolutional layer
- After each convolutional layer operation, max pooling is not required.
- No parameters are must, so its kind of transformation operation

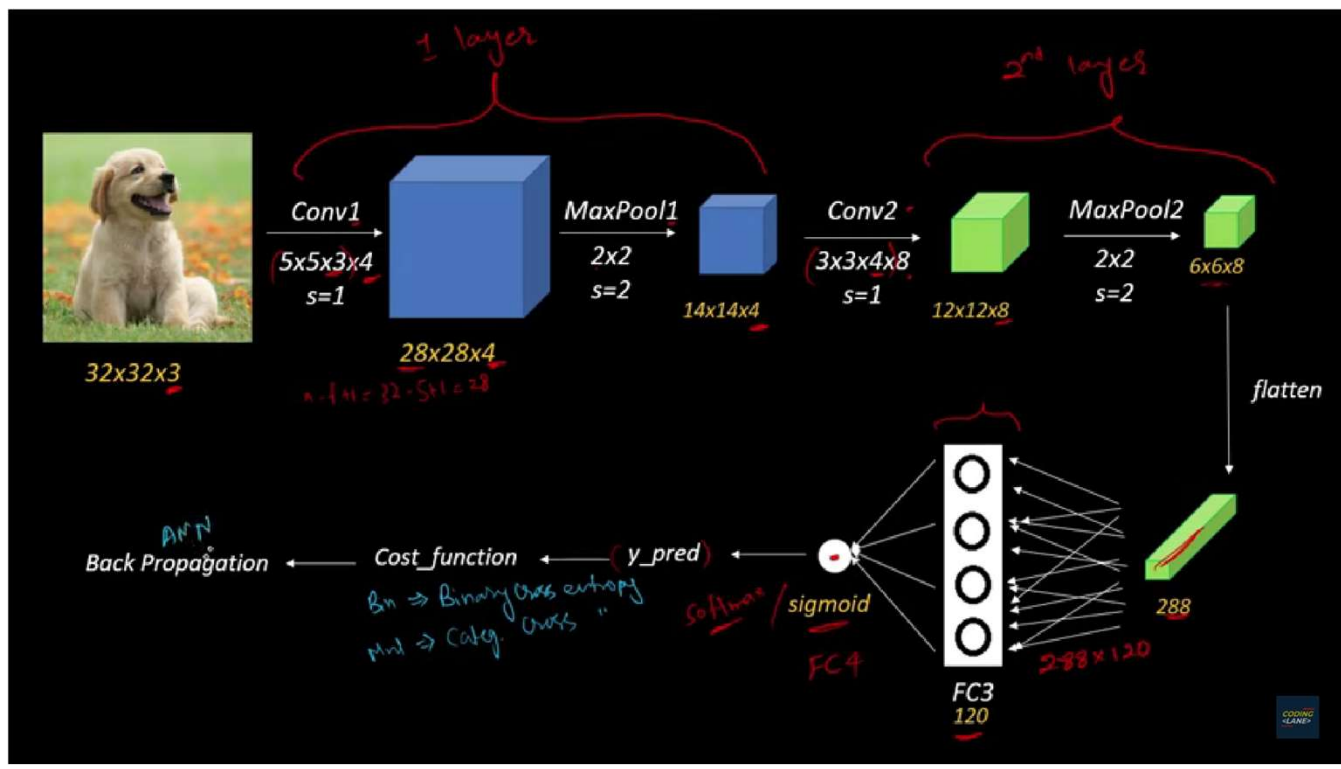
Fully-connected Layer

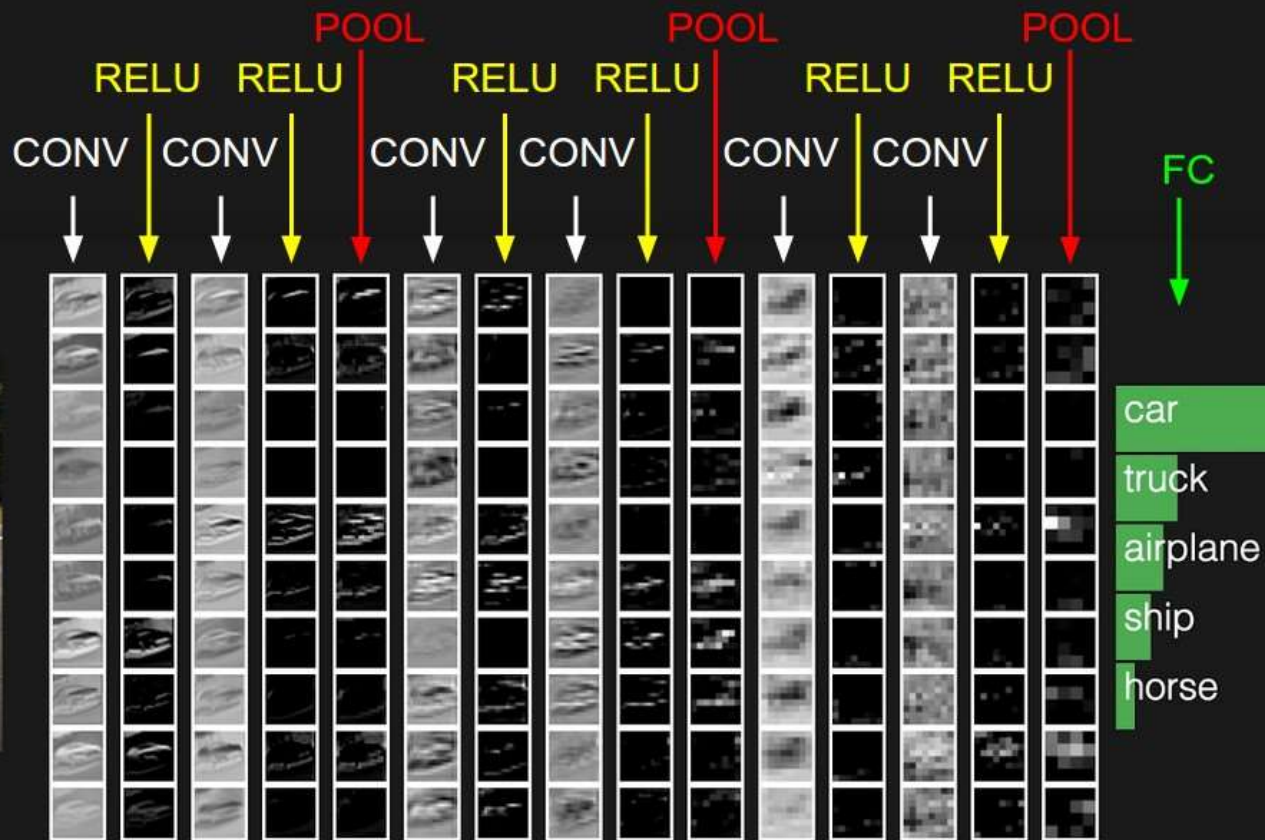
- This is the layer responsible for performing the task classification based on the features extracted during the previous layers. While both convolutional and pooling layers tend to use **ReLU** functions, fully-connected layers use the **Softmax** activation function for classification, producing a probability from 0 to 1.



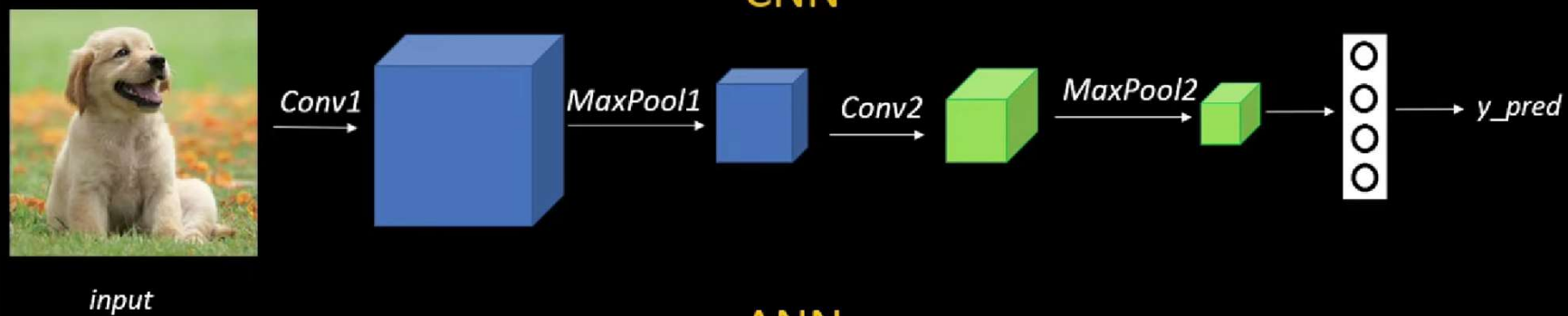


- After extracting features from pooling layer, array will be flattened in 1D and will be given as input to the fully connected layer
- The layer is call fully connected cause nodes of one layers are connected to all the nodes of previous layer as well as next layer.
- No of outputs are same as number of categories for classification.

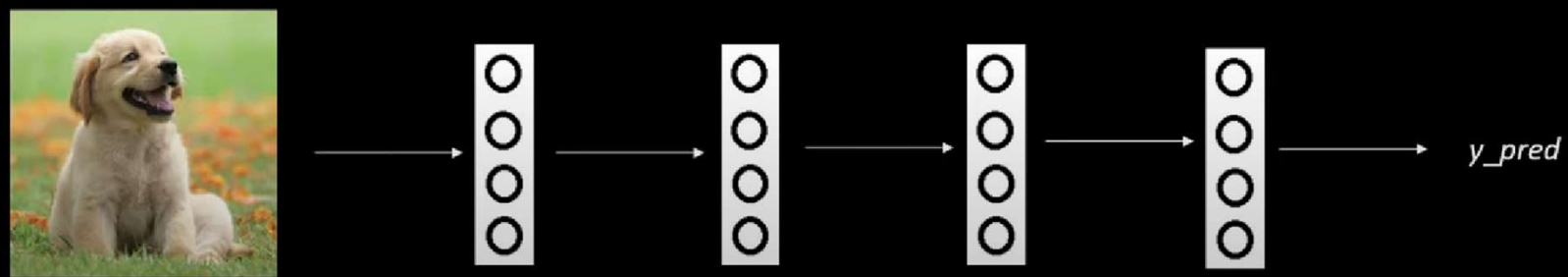




CNN



ANN



Regularization

- Imagine that you have a model that has learned to detect pictures of cats. The model that you have defined has a lot of parameters.
- You will notice that the loss value is minimal, however when you deploy your model to detect a picture of a cat in a different setting compared to your training set you will notice that your model will not be able to recognize the cat very well.
- This happens because your model performed extremely well on the training set. We use the term 'overfitting' to describe models that perform extremely well on the training set but fail to generalize well on a test set
- One way to prevent overfitting is to use regularization

- Regularization is a method that controls the model complexity.
- In this example, the images have certain features that help the model identify it as a cat, like a cat's whisker, ears, eyes, etc.
- Each feature is assigned a certain weight. If there are a lot of features then there will be a large number of weights, which will make the model prone to overfitting.
- So regularization reduces the burden on these weights. The weights then end up having less impact on the loss function which determines the error between the actual label and predicted label.

Parameter Sharing in CNN

- In a neural network, "parameters" usually refer to the weights and biases of the neurons. In a fully connected neural network, every neuron in one layer is connected to every neuron in the next layer, each connection having its own weight. This leads to a large number of parameters, especially for high-dimensional inputs like images.
- Parameter sharing scheme is used in Convolutional Layers to control the number of parameters.
- Parameter sharing forces sets of parameters to be similar as we interpret various models or model components as sharing a unique set of parameters. We only need to store only a subset of memory.
- Suppose two models A and B, perform a classification task on similar input and output distributions.
- In such a case, we'd expect the parameters for both models to be identical to each other as well.
- We could impose a norm penalty on the distance between the weights, but a more popular method is to force the parameters to be equal.

- The idea behind Parameter Sharing is the essence of forcing the parameters to be similar.
- A significant benefit here is that we need to store only a subset of the parameters (e.g., storing only the parameters for model A instead of storing for both A and B), which leads to significant memory savings.
- The most extensive use of parameter sharing is in convolutional neural networks. Natural images have specific statistical properties that are robust to translation.
- For example photo of a cat remains a photo of a cat if it is translated one pixel to the right. Convolution Neural Networks consider this property by sharing parameters across multiple image locations. Thus we can find a cat with the same cat detector in column i or $i+1$ in the image.

- In a CNN, parameter sharing means that instead of having unique weights for each connection, we use the same set of weights (called a filter or kernel) across different parts of the input. This drastically reduces the number of parameters and allows the network to detect patterns (like edges, textures) regardless of their position in the input image.

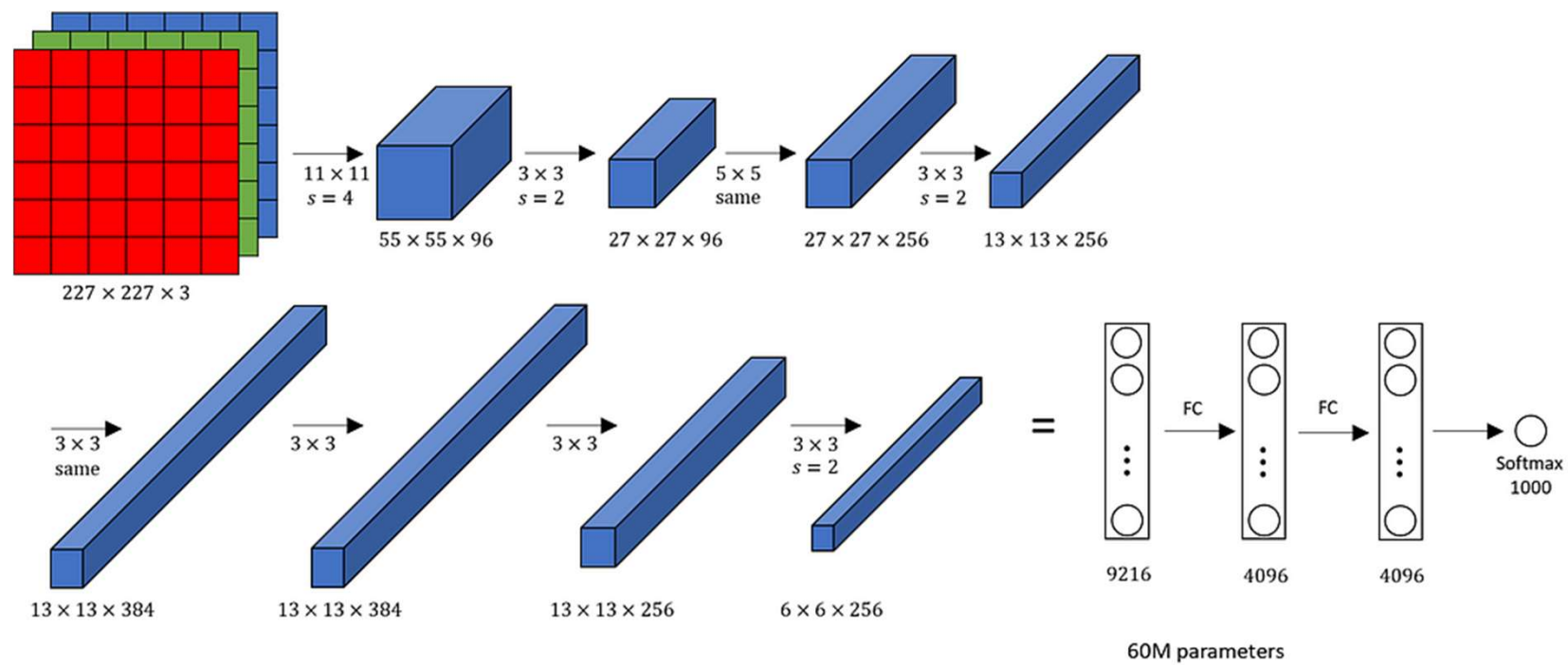
AlexNet

- Numerous variations of CNN have developed over the years resulting in several CNN architectures. The most common amongst them are:
- 1. LeNet-5 (1998)
- **2. AlexNet (2012)**
- 3. ZFNet (2013)
- 4. GoogleNet / Inception(2014)
- 5. VGGNet (2014)
- **6. ResNet (2015)**
- **AlexNet** was primarily designed by Alex Krizhevsky. It was published with Ilya Sutskever and Krizhevsky's doctoral advisor Geoffrey Hinton, and is a Convolutional Neural Network or CNN.
- After competing in ImageNet Large Scale Visual Recognition Challenge, AlexNet shot to fame. It achieved a top-5 error of 15.3%. This was 10.8% lower than that of runner up.

AlexNet Architecture

- AlexNet was the first convolutional network which used GPU to boost performance.
- AlexNet architecture consists of 5 convolutional layers, 3 max-pooling layers, 2 normalization layers, 2 fully connected layers, and 1 SoftMax layer.
- Each convolutional layer consists of convolutional filters and a nonlinear activation function ReLU.
- The pooling layers are used to perform max pooling.
- Input size is fixed due to the presence of fully connected layers.
- The input size is mentioned at most of the places as $224 \times 224 \times 3$ but due to some padding which happens it works out to be $227 \times 227 \times 3$
- AlexNet overall has 60 million parameters.

- Used Normalization layers which are not common anymore
- Batch size of 128
- SGD Momentum as learning algorithm
- Heavy Data Augmentation with things like flipping, jittering, cropping, color normalization, etc.
- Ensemble of models to get the best results.



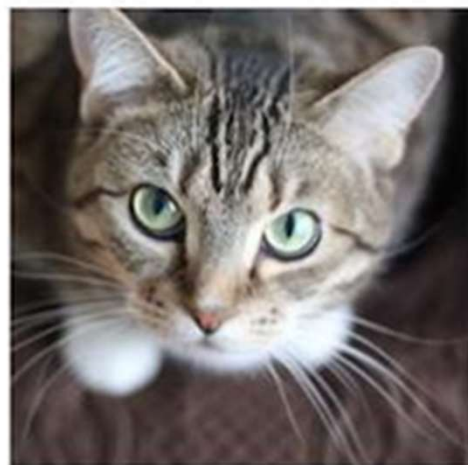
- The Overfitting Problem. AlexNet had 60 million parameters, a major issue in terms of overfitting.
- Two methods to reduce overfitting:
- **Data Augmentation**
- **Dropout**

Data Augmentation

- Overfitting can be avoided by showing Neural Net various iterations of the same image. Additionally, it assists in producing more data and compels the Neural Net to memories the main qualities.
- **Augmentation by Mirroring**
- Consider that our training set contains a picture of a cat. A cat can also be seen as its mirror image. This indicates that by just flipping the image above the vertical axis, we may double the size of the training datasets.
-



Mirror Image



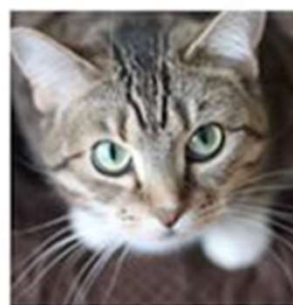
- **Augmentation by Random Cropping of Images**

- Randomly cropping the original image will also produce additional data that is simply the original data shifted.
- For the network's inputs, the creators of AlexNet selected random crops with dimensions of 227 by 227 from within the 256 by 256 image boundary. They multiplied the size of the data by 2048 using this technique.



256

Random Crops



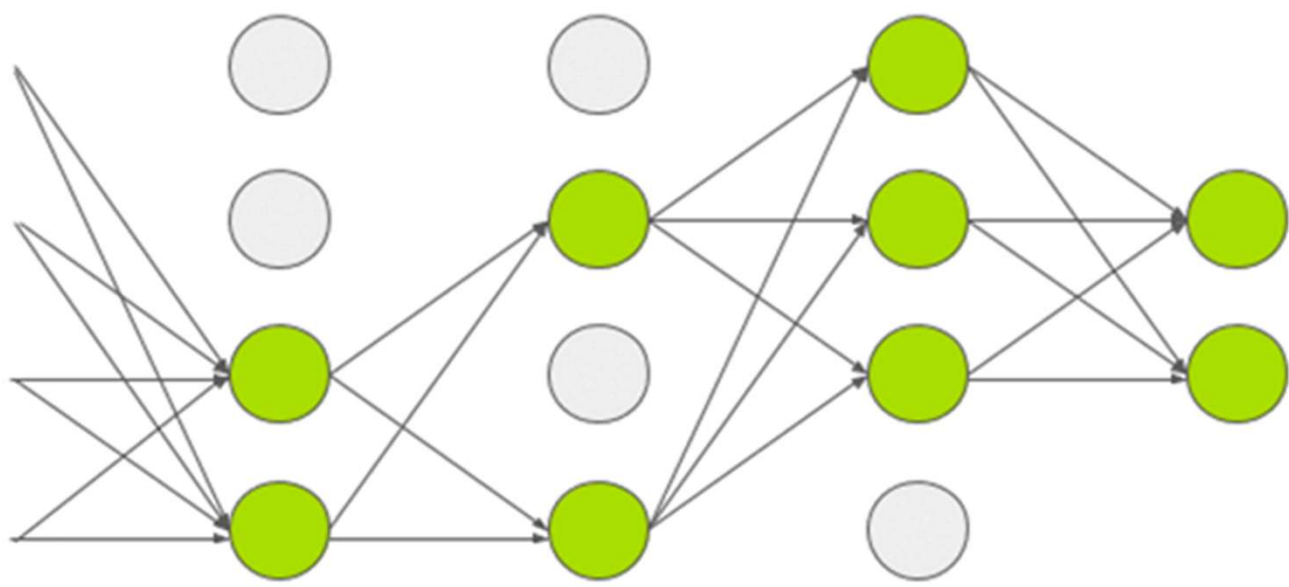
256

227

227

Dropout

- A neuron is removed from the neural network during dropout with a probability of 0.5.
- A neuron that is dropped does not make any contribution to either forward or backward propagation. As seen in the graphic , each input is processed by a separate Neural Network design. The acquired weight parameters are therefore more reliable and less prone to overfitting



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax



mite

container ship

motor scooter

leopard

	mite	container ship	motor scooter	leopard
	black widow	lifeboat	go-kart	jaguar
	cockroach	amphibian	moped	cheetah
	tick	fireboat	bumper car	snow leopard
	starfish	drilling platform	golfcart	Egyptian cat



grille

mushroom

cherry

Madagascar cat

	convertible	agaric	dalmatian	squirrel monkey
	grille	mushroom	grape	spider monkey
	pickup	jelly fungus	elderberry	titi
	beach wagon	gill fungus	ffordshire bullterrier	indri
	fire engine	dead-man's-fingers	currant	howler monkey

Pros of AlexNet

1. AlexNet is considered as the milestone of CNN for image classification.
2. Many methods, such as the conv+pooling design, dropout, GPU, parallel computing, ReLU, are still the industrial standard for computer vision.
3. The unique advantage of AlexNet is the direct image input to the classification model.
4. The convolution layers can automatically extract the edges of the images and fully connected layers learning these features
5. Theoretically the complexity of visual patterns can be effectively extracted by adding more convlayer

Cons of AlexNet

1. AlexNet is NOT deep enough compared to the later model, such as VGGNet, GoogLeNet, and ResNet.
2. The use of large convolution filters (5×5) is not encouraged shortly after that.
3. Use normal distribution to initiate the weights in the neural networks, can not effectively solve the problem of gradient vanishing, replaced by the Xavier method later.
4. The performance is surpassed by more complex models such as GoogLeNet (6.7%), and ResNet (3.6%)

ResNet