CHAPTER 8

Prepared by: Afreen Banu & Ashwin R G

Introduction to Convolutional Neural Networks

Imagine there's an image of a bird, and you want to identify whether it's really a bird or some other object. The first thing you do is feed the pixels of the image in the form of arrays to the input layer of the neural network (multi-layer networks used to classify things). The hidden layers carry out feature extraction by performing different calculations and manipulations. There are multiple hidden layers like the convolution layer, the ReLU layer, and pooling layer, that perform feature extraction from the image. Finally, there's a fully connected layer that identifies the object in the image.

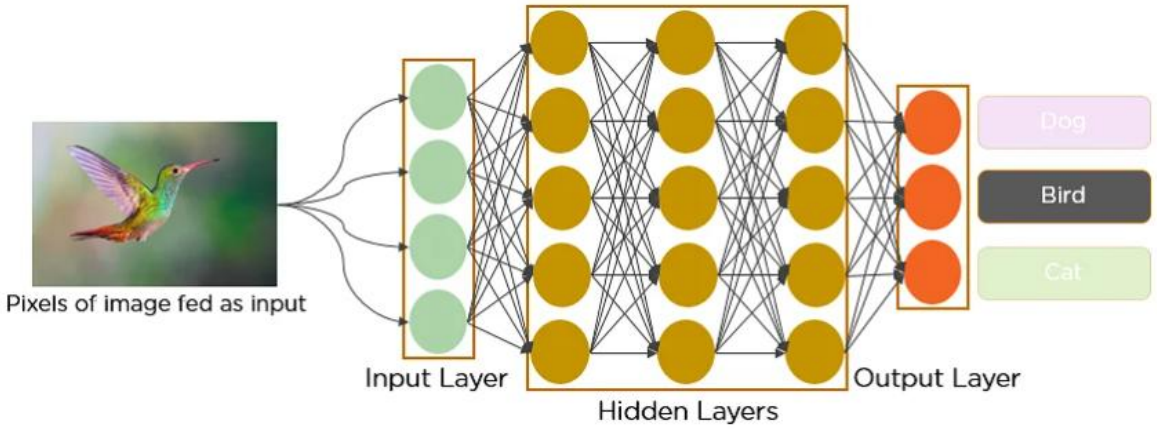
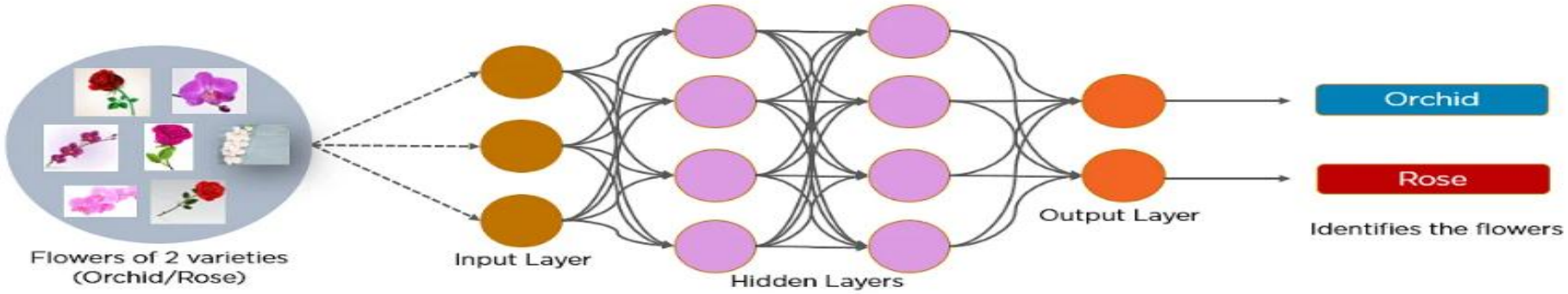


Fig: Convolutional Neural Network to identify the image of a bird

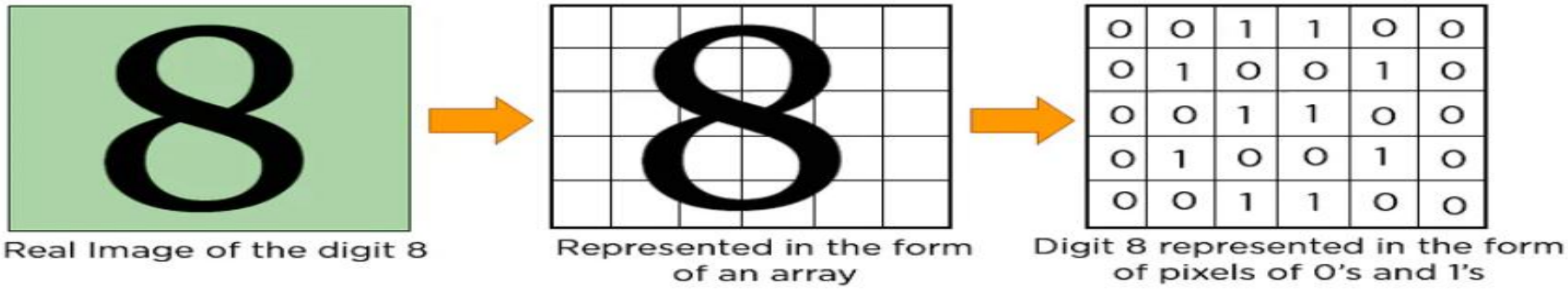
What is CNN?

A convolutional neural network is a feed-forward neural network that is generally used to analyze visual images by processing data with grid-like topology. It's also known as a ConvNet. A convolutional neural network is used to detect and classify objects in an image.

Below is a neural network that identifies two types of flowers: Orchid and Rose.



In CNN, every image is represented in the form of an array of pixel values.



Layers in a Convolutional Neural Network

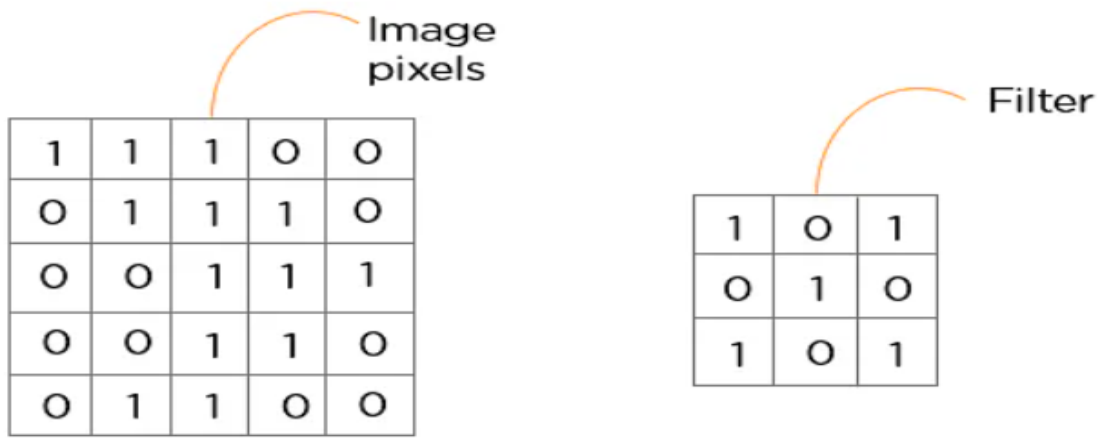
A convolution neural network has multiple hidden layers that help in extracting information from an image. The four important layers in CNN are:

1. Convolution layer
2. ReLU layer
3. Pooling layer
4. Fully connected layer

Convolution Layer

This is the first step in the process of extracting valuable features from an image. A convolution layer has several filters that perform the convolution operation. Every image is considered as a matrix of pixel values.

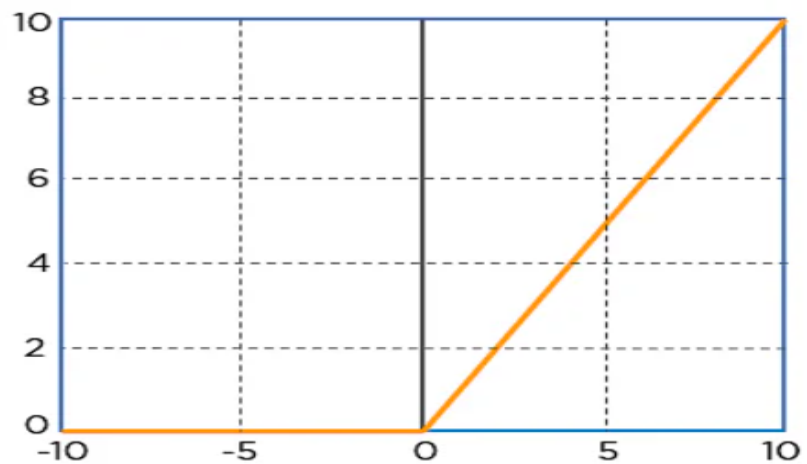
Consider the following 5x5 image whose pixel values are either 0 or 1. There's also a filter matrix with a dimension of 3x3. Slide the filter matrix over the image and compute the dot product to get the convolved feature matrix.



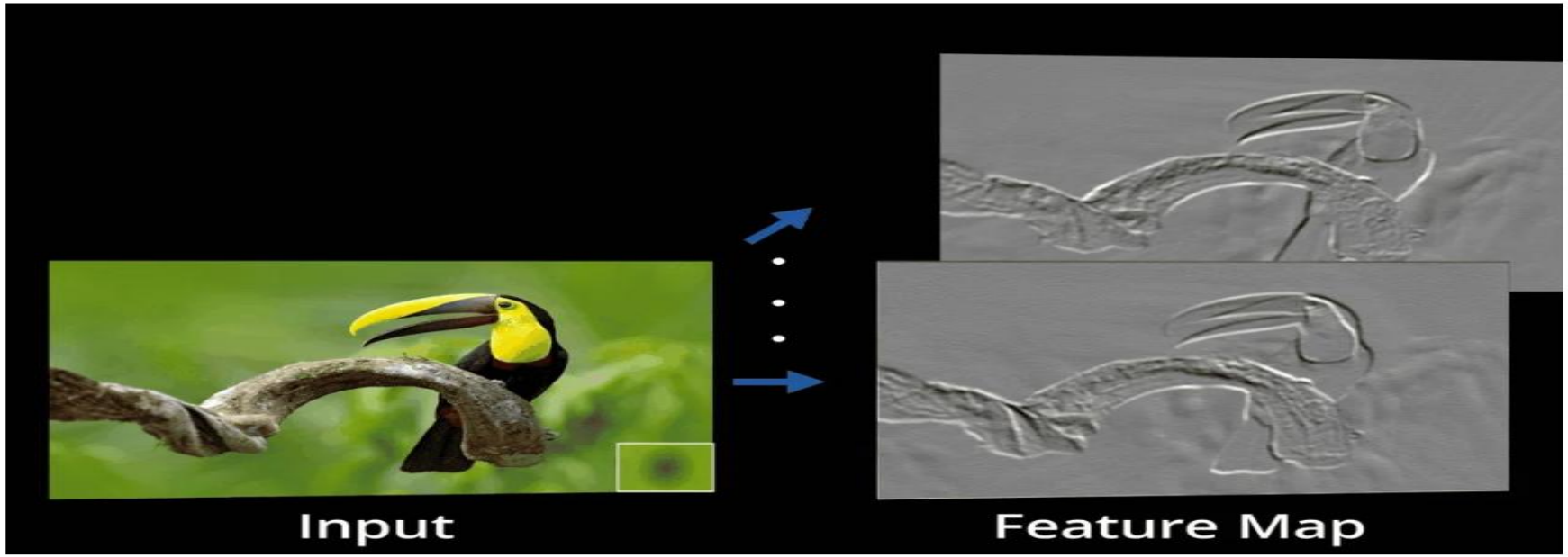
ReLU layer

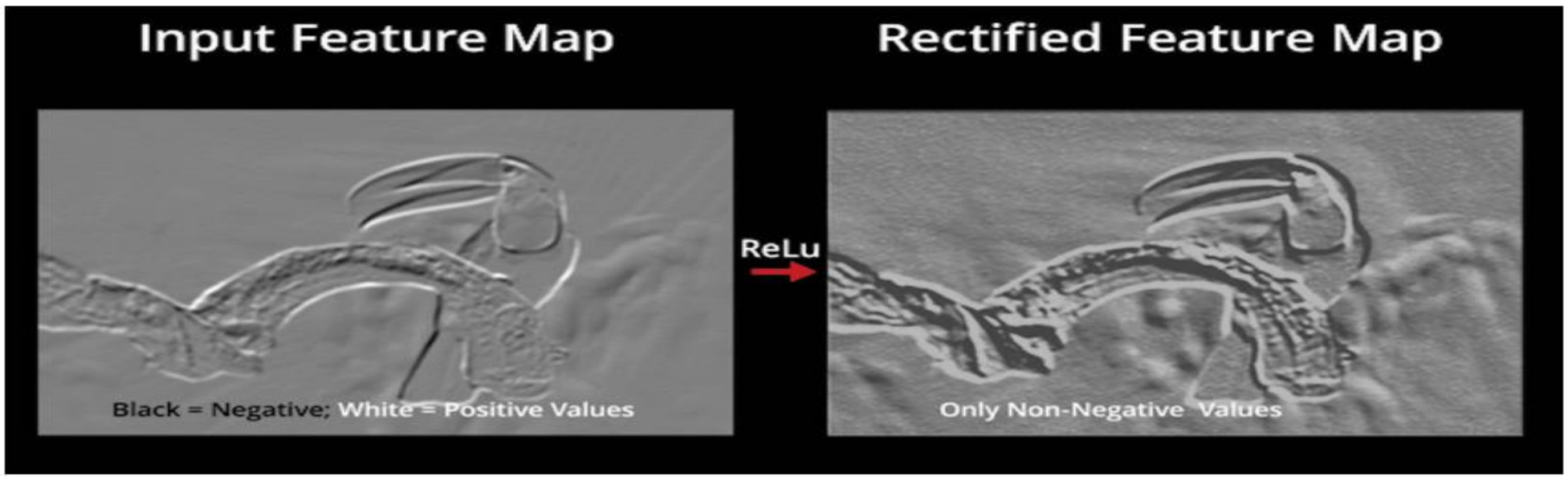
ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is to move them to a ReLU layer.

ReLU performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map. Below is the graph of a ReLU function:



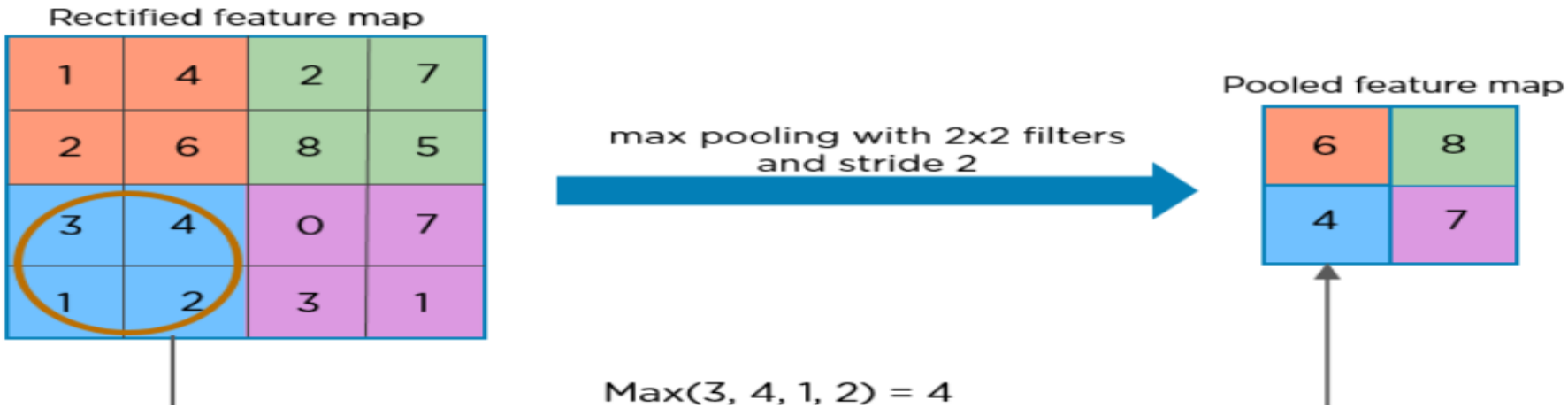
The original image is scanned with multiple convolutions and ReLU layers for locating the features.



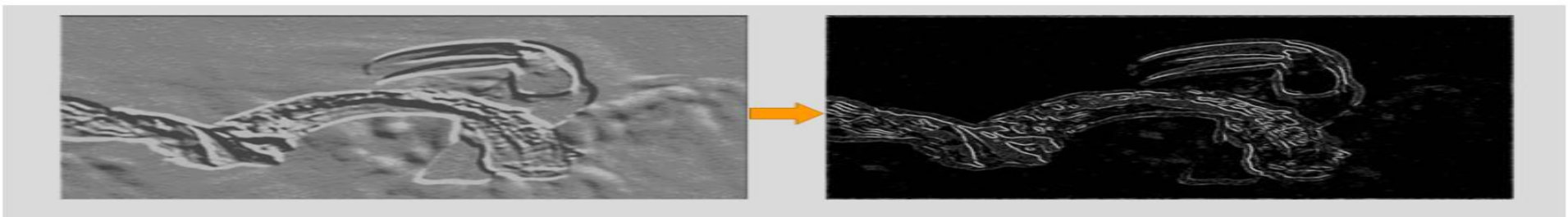


Pooling Layer

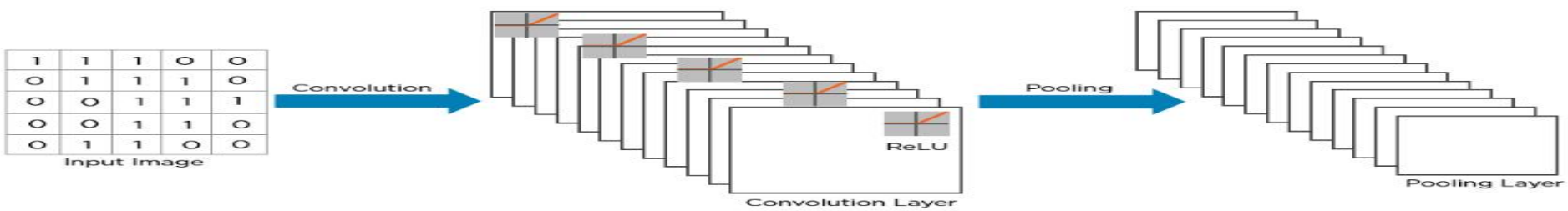
Pooling is a down-sampling operation that reduces the dimensionality of the feature map. The rectified feature map now goes through a pooling layer to generate a pooled feature map.



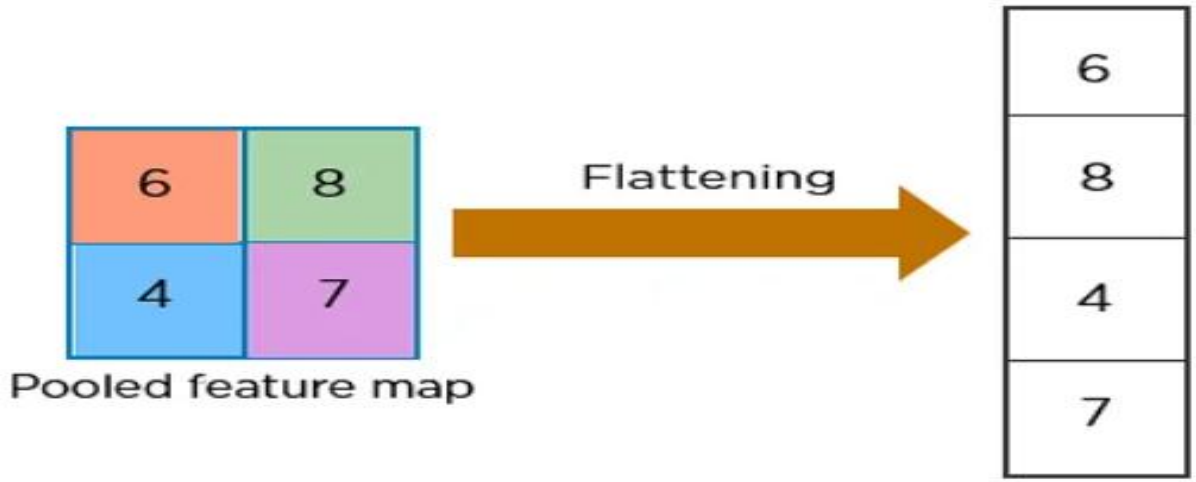
The pooling layer uses various filters to identify different parts of the image like edges, corners, body, feathers, eyes, and beak.



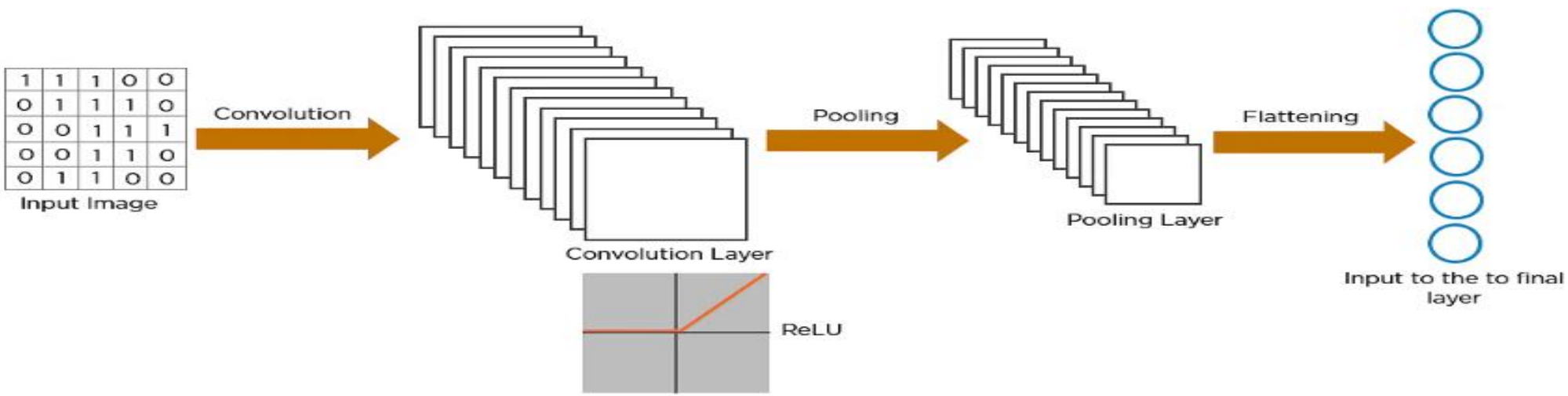
Here's how the structure of the convolution neural network looks so far:

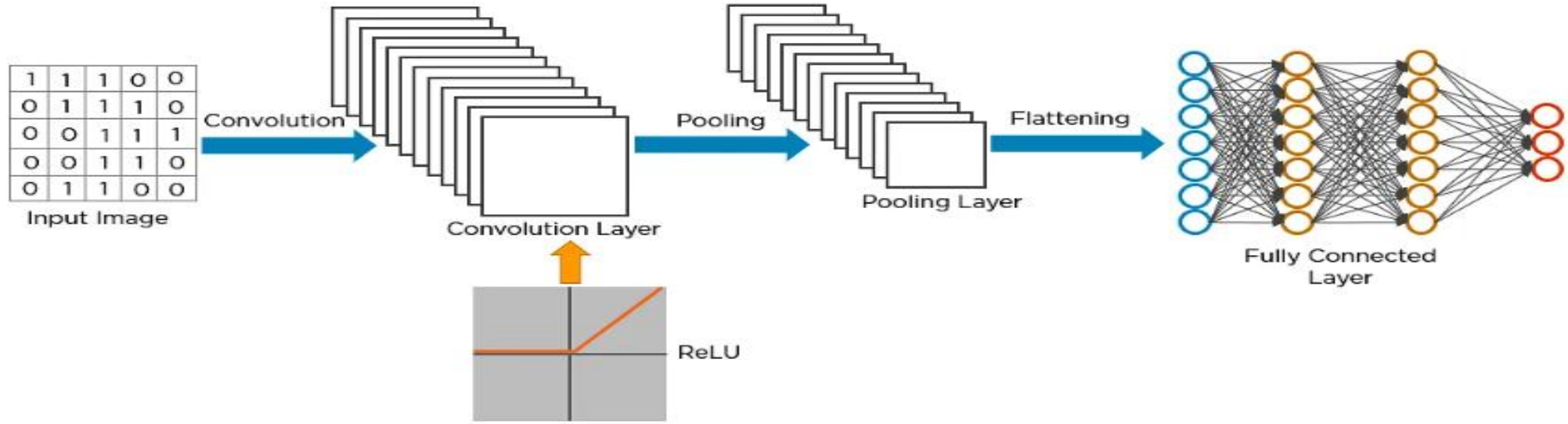


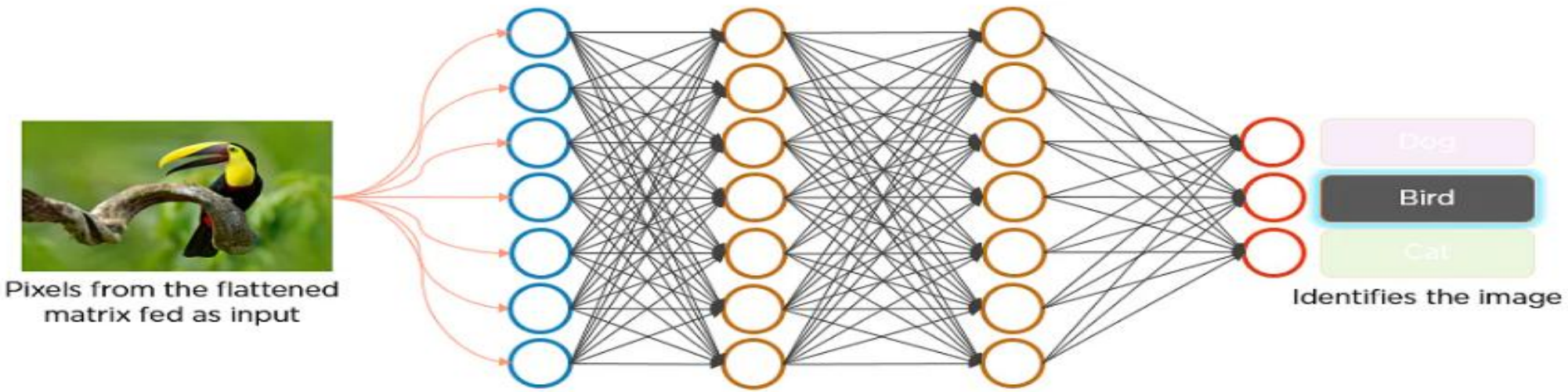
The next step in the process is called flattening. Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector.



The flattened matrix is fed as input to the fully connected layer to classify the image.

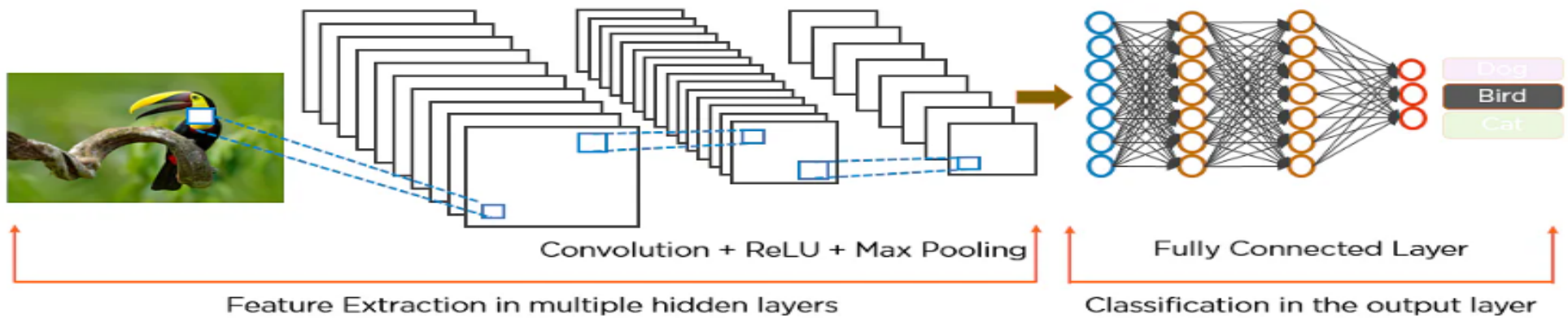



1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
Input Image
Convolution
Convolution Layer
Pooling
Pooling Layer
Flattening
Fully Connected Layer
ReLU


Pixels from the flattened matrix fed as input
Dog
Bird
Cat
Identifies the image

Here's how exactly CNN recognizes a bird:

- The pixels from the image are fed to the convolutional layer that performs the convolution operation
- It results in a convolved map
- The convolved map is applied to a ReLU function to generate a rectified feature map
- The image is processed with multiple convolutions and ReLU layers for locating the features
- Different pooling layers with various filters are used to identify specific parts of the image
- The pooled feature map is flattened and fed to a fully connected layer to get the final output



Let’s understand first the basic terminology used in Computer Vision

1. **Color Space:** It is a way to represent color channels in the image. Basically three types of color spaces are there: 1. RGB/BGR 2. HSV 3. CMYK (we will discuss in detail later)
2. **Hue:** A hue is, to what degree, a color is different from Green, Blue or Red.
3. **Saturation:** Image is saturated, if no white light is present in the color of the image.
4. **Aspect Ratio:** Aspect ratio of an image is the ratio of width to height of an object in an image.
5. **Contrast:** A contrast is the difference between lightest and darkest region of the image.
6. **Edge:** Edges are boundaries of objects present in an image.
7. **Tone:** Tone describes lightness or darkness of the image.
8. **Morphological Operation:** In image processing morphological operation deals with changes in structure of the object present in the image.
9. **Noise:** In image processing, noise means any undesirable change in pixel value(pixel density)
10. **Pixel:** The smallest square unit of an image that gives visual information of that part.
11. **Sharpening:** Sharpening is an enhancement technique that focus on details of the image like edge highlight.

How to install OpenCV in python?

You just need to run the following command in the Jupyter notebook for OpenCV installation.

```
1  !pip install opencv-python
```

installation of openCV hosted with ❤ by GitHub view raw

View this gist on GitHub

Output:

```
Collecting opencv-python
  Downloading opencv_python-4.5.3.56-cp38-cp38-win_amd64.whl (34.9 MB)
Requirement already satisfied: numpy>=1.17.3 in c:\users\arpita\anaconda3\lib\site-packages (from opencv-python) (1.19.5)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.5.3.56
```

1. Read an image using OpenCV

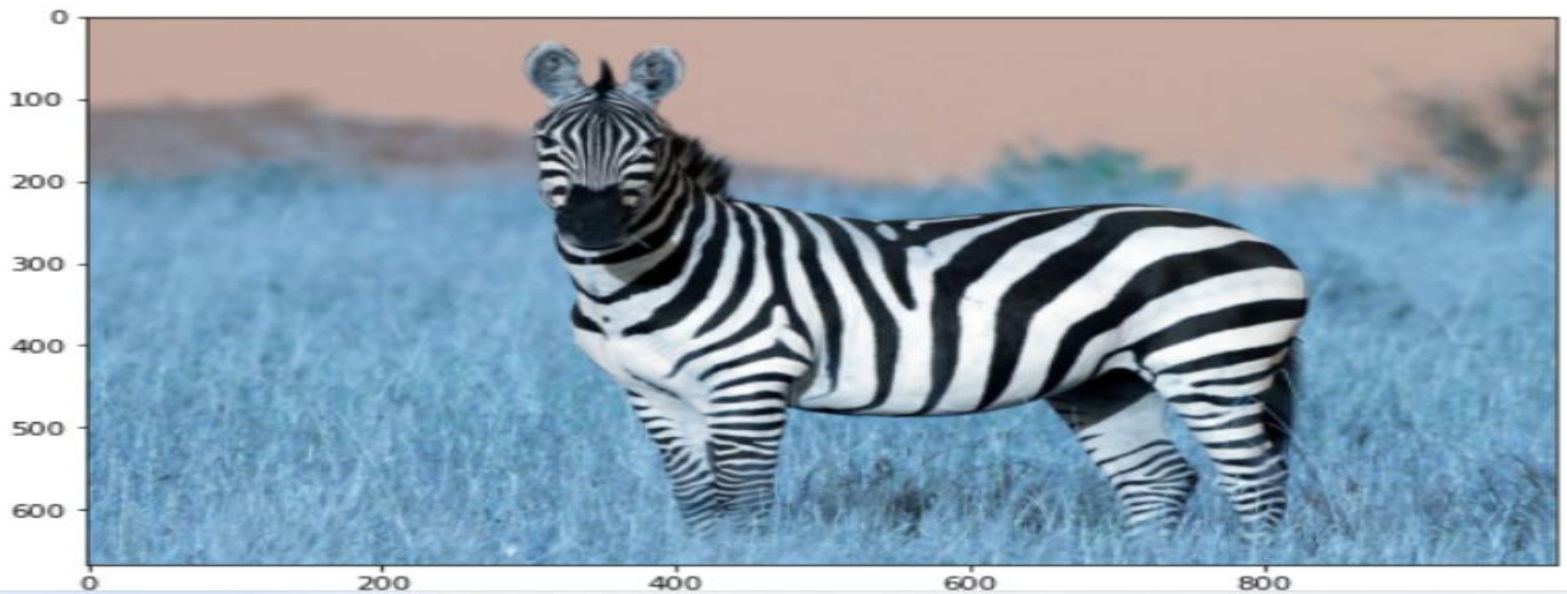
```
1  img = cv2.imread('Zebra.jpg')
2  plt.figure(figsize = (10,6))
3  plt.imshow(img)
```

Image reading.py hosted with ❤ by GitHub view raw

View this gist on GitHub

Output:

```
Out[3]: <matplotlib.image.AxesImage at 0x1f9faba89d0>
```



2. Display the image you just read

To display an image on a temporary basis, we need to use the cv2.imshow() function. This function helps to display the image in a separate window.

> Note:
>
> cv2.waitkey() is keyboard binding function. It waits for the user for any key is to be pressed for particular milliseconds. In case '0' is passed as an argument of waitkey(), it means it will wait indefinitely times unless any key is pressed.

DestroyAllWindows() simply destroys all windows if anything is created already.

```
1  cv2.imshow('zebra',img)
2  cv2.waitKey(0)
3  cv2.destroyAllWindows()
```

Display the image hosted with ❤ by GitHub view raw

View this gist on GitHub

Output:

Display the image

Change colored image to grayscale

```
1  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
2  cv2.imshow('zebra',gray)
3  cv2.waitKey(0)
4  cv2.destroyAllWindows()
```

image_conversion.py hosted with ❤ by GitHub view raw

View this gist on GitHub

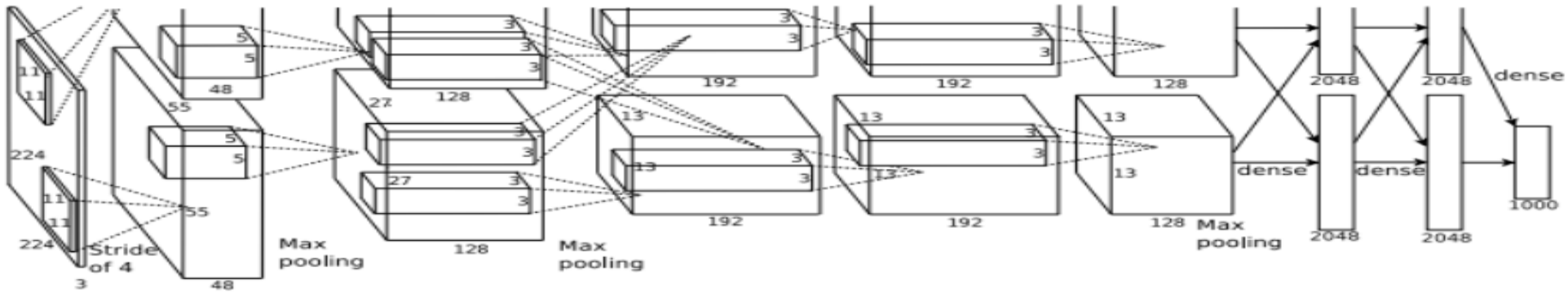
Output:

CNN Architectures

- **A Convolutional Neural Network** (**CNN**, or **ConvNet**) are a special kind of multi-layer neural networks, designed to recognize visual patterns directly from pixel images with minimal preprocessing..

AlexNet (2012)

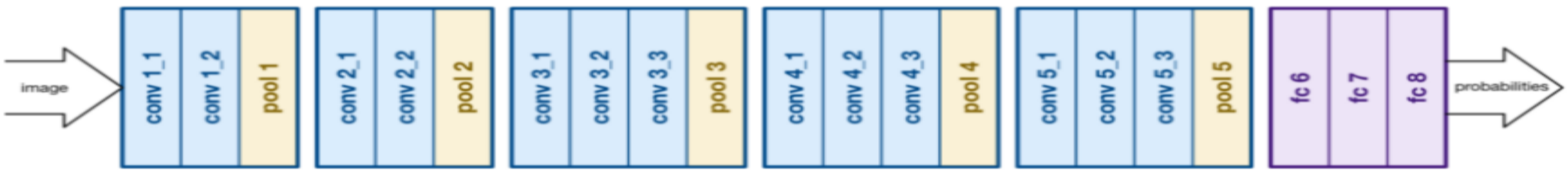
In 2012, AlexNet significantly outperformed all the prior competitors and won the challenge by reducing the top-5 error from 26% to 15.3%. The second place top-5 error rate, which was not a CNN variation, was around 26.2%.



The network had a very similar architecture as LeNet by Yann LeCun et al but was deeper, with more filters per layer, and with stacked convolutional layers. It consisted 11x11, 5x5,3x3, convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum. It attached ReLU activations after every convolutional and fully-connected layer. AlexNet was trained for 6 days simultaneously on two Nvidia Geforce GTX 580 GPUs which is the reason for why their network is split into two pipelines. AlexNet was designed by the SuperVision group, consisting of Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever.

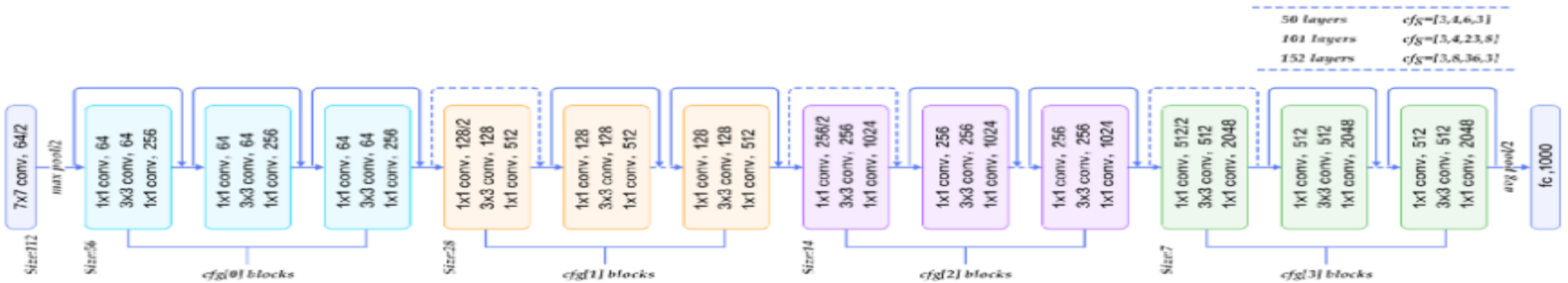
VGGNet (2014)

The runner-up at the ILSVRC 2014 competition is dubbed VGGNet by the community and was developed by Simonyan and Zisserman. VGGNet consists of 16 convolutional layers and is very appealing because of its very uniform architecture. Similar to AlexNet, only 3x3 convolutions, but lots of filters. Trained on 4 GPUs for 2–3 weeks. It is currently the most preferred choice in the community for extracting features from images. The weight configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor. However, VGGNet consists of 138 million parameters, which can be a bit challenging to handle.



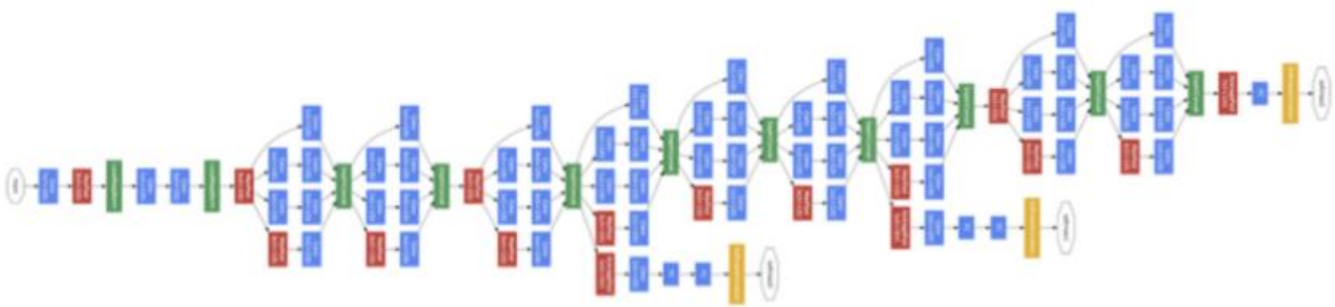
ResNet(2015)

At last, at the ILSVRC 2015, the so-called Residual Neural Network (ResNet) by Kaiming He et al introduced anovel architecture with “skip connections” and features heavy batch normalization. Such skip connections are also known as gated units or gated recurrent units and have a strong similarity to recent successful elements applied in RNNs. Thanks to this technique they were able to train a NN with 152 layers while still having lower complexity than VGGNet. It achieves a top-5 error rate of 3.57% which beats human-level performance on this dataset.



GoogLeNet/Inception(2014)

The winner of the ILSVRC 2014 competition was GoogLeNet(a.k.a. Inception V1) from Google. It achieved a top-5 error rate of 6.67%! This was very close to human level performance which the organisers of the challenge were now forced to evaluate. As it turns out, this was actually rather hard to do and required some human training in order to beat GoogLeNets accuracy. After a few days of training, the human expert (Andrej Karpathy) was able to achieve a top-5 error rate of 5.1%(single model) and 3.6%(ensemble). The network used a CNN inspired by LeNet but implemented a novel element which is dubbed an inception module. It used batch normalization, image distortions and RMSprop. This module is based on several very small convolutions in order to drastically reduce the number of parameters. Their architecture consisted of a 22 layer deep CNN but reduced the number of parameters from 60 million (AlexNet) to 4 million.



Object Detection

Object detection can be defined as a computer vision technique which aims to identify and locate objects on an image or a video. Computers might be able to process information way faster than humans, however, it is still difficult for computers to detect various objects on an image or video. The reason for this is that the computer interprets the majority of the outputs in the binary language only.

Difference between Object Detection and Image Classification

As long as there is only one object, image classification techniques can be used.

But if we have multiple objects, that's when the concept of Object Detection comes into play. By building rectangular boxes around the object of interest we can help the machine recognize the object each box contains. We can also indicate the exact location of the objects using this method. It is possible for a single picture to contain many objects, so multiple bounding boxes may be shown.

Object detection applications are limitless, but they generally identify and detect the real-objects such as human beings, buildings, cars and many more. Additionally, a machine needs a lot of labeled data of different kinds of objects for it to recognize those objects in the future. This means the ML model being trained on that labeled dataset will have a better chance to make accurate predictions.

Object Detection Models

Now that we are clear with the definition of Object Detection, let's have a look at some popular Object Detection models.

R-CNN, Faster R-CNN, Mask R-CNN

The most popular object detection models belong to the family of regional based CNN models. This model has revolutionized the way the world of Object Detection used to work. In the past few years, they've not only become more accurate but more efficient too.

SSD and YOLO

There are a plethora of models belonging to the single shot detector family which were published in 2016. Although SSDs are faster than CNN models, their accuracy rate is much lower than that of the CNNs.

YOLO or you only look once, is quite different from region-based algorithms. Just like SDDs, yolo is faster than R-CNNs but lags behind because of low accuracy. For mobile or embedded devices, SDDs are the perfect choice.

Benefits of Object detection to Real-world

Object detection is completely inter-linked with other similar computer vision techniques such as image segmentation and image recognition that assist us to understand and analyze the scenes in videos and images. Nowadays, several real-world use cases are implemented in the market of object detection which make a tremendous impact on different industries.

Here we'll specifically examine how object detection applications have impacted in the following areas.

Self-driving cars

The primary reason behind the success of autonomous vehicles is real-time object detection artificial intelligence based models. These systems allow us to locate, identify and track the objects around them, for the purpose of safety and efficiency.

Video Surveillance

Real-time object detection and tracking the movements of objects allow video surveillance cameras to track the record of scenes of a particular location such as an airport. This state-of-the-art technique accurately recognizes and locates several instances of a given object in the video. In real-time, as the object moves through a given scene or across the particular frame, the system stores the information with real-time tracking feeds.

Crowd Counting

For heavily populated areas such as shopping malls, airports, city squares and theme parks, this application performs unbelievably well. Generally, this object detection application proves to be helpful to large enterprises and municipalities for tracking road traffic, violation of laws and number of vehicles passing in a particular time frame.

Challenges and Solutions of Object detection Modelling

Dual Synchronization

The first challenge for object detection is to classify the image and position of the object, which is known as object localization. In order to address this problem, most developers often use a multi-tasking loss function to penalize both localization and misclassification errors.

- **Solution:** Regional based Convolutional neural networks displays one class of object detection framework that consist of region generation proposals where objects are likely to be located, followed by CNN models processing to classify and rectify the object locations. Fast-R CNN model can improve the initial results with R-CNN. As its name denotes, this Fast R-CNN model provides tremendous speed, but accuracy also improves only because the localization and object classification tasks are optimized using a multi-task loss function.

Real-time detection speed

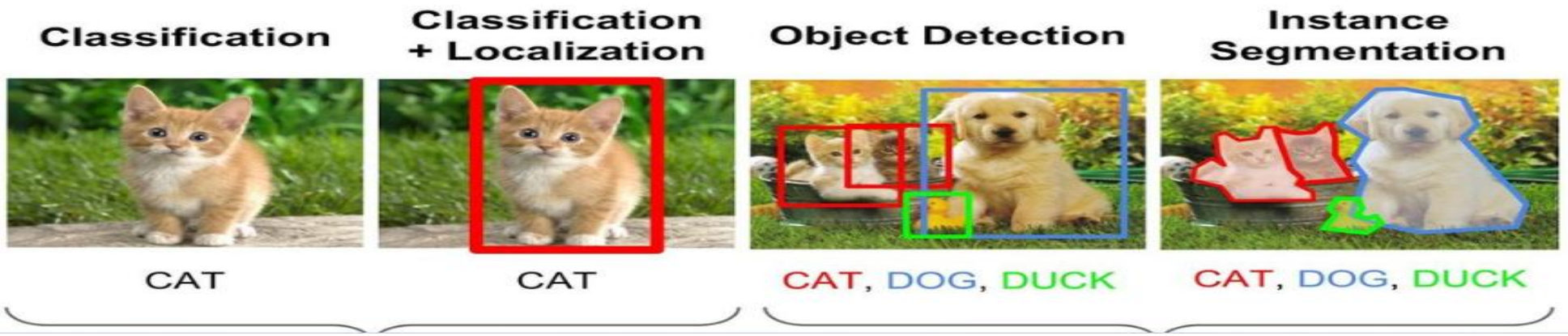
Fast speed of object detection algorithms has always been a major problem to classify and localize the crucial objects accurately at same time to meet the real-time video processing. Over the years, several algorithms improved the test time from 0.02 frames per second to 155 fps.

- **Solution:** Faster R-CNN and Fast R-CNN models aim to speed up the original speed of R-CNN approach. Because R-CNN uses the selective search to produce 2000 candidate regions of interest and passes through each CNN based model individually, that may cause a heavy bottleneck since the model processing gets down. Whereas, Fast R-CNN model transmits the whole image through CNN base once and then matches the ROIs created with selective search to feature map, considering 20-fold reduction in processing time.

Semantic Segmentation

Deep learning has been very successful when working with images as data and is currently at a stage where it works better than humans on multiple use-cases. The most important problems that humans have been interested in solving with computer vision are **image classification, object detection and segmentation** in the increasing order of their difficulty.

In the plain old task of image classification we are just interested in getting the labels of all the objects that are present in an image. In object detection we come further a step and try to know along with what all objects that are present in an image, the location at which the objects are present with the help of bounding boxes. Image segmentation takes it to a new level by trying to find out accurately the exact boundary of the objects in the image.



1. **Semantic segmentation** :- Semantic segmentation is the process of classifying each pixel belonging to a particular label. It doesn't different across different instances of the same object. For example if there are 2 cats in an image, semantic segmentation gives same label to all the pixels of both cats

2. **Instance segmentation** :- Instance segmentation differs from semantic segmentation in the sense that it gives a unique label to every instance of a particular object in the image. As can be seen in the image above all 3 dogs are assigned different colours i.e different labels. With semantic segmentation all of them would have been assigned the same colour.

NLP

Vanishing and exploding gradients in RNNs

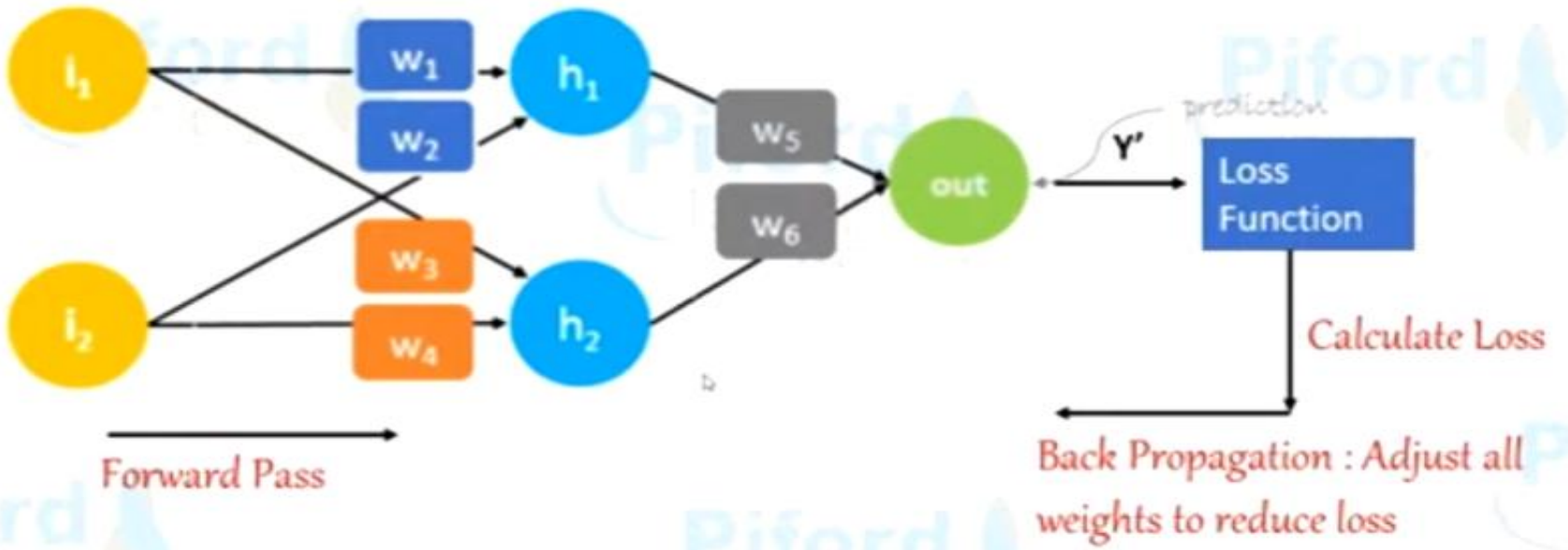
Vanishing –

As the backpropagation algorithm advances downwards(or backward) from the output layer towards the input layer, the gradients often get smaller and smaller and approach zero which eventually leaves the weights of the initial or lower layers nearly unchanged. As a result, the gradient descent never converges to the optimum. This is known as the ***vanishing gradients*** problem.

Exploding –

On the contrary, in some cases, the gradients keep on getting larger and larger as the backpropagation algorithm progresses. This, in turn, causes very large weight updates and causes the gradient descent to diverge. This is known as the ***exploding gradients*** problem.

Vanishing Gradient



Activation function:tanh

Tanh gets the summation of input and then it will transform our output between -1 to 1.

x	f(x)=x	F(x)
-4	f(-4) = -1	-1
-2	f(-2) = -0.7	-0.7
2	f(2) = 0.7	0.7
4	f(4) = 1	1

How we reduce LOSS?

$$w\ new = w\ old\ - n\frac{dL}{dw\ old}$$

Chain rule of Derivatives

$$\frac{dL}{dw1\ old} = \frac{d\ out}{d\ h1} \cdot \frac{d\ h1}{d\ w1}$$

Suppose I want to update the weight of w1 and then the formula for this is :

$$w1\ new = w1\ old\ - n\frac{dL}{dw1\ old}$$

We know our output is out and out is getting impacted by h1 and h1 is dependent on w1. We are following chain rule here:

$$\frac{dL}{dw1\ old} = \frac{d\ out}{d\ h1} \cdot \frac{d\ h1}{d\ w1}$$

0.23 * 0.03 =0.0069

0.23 is the value we got from this derivative

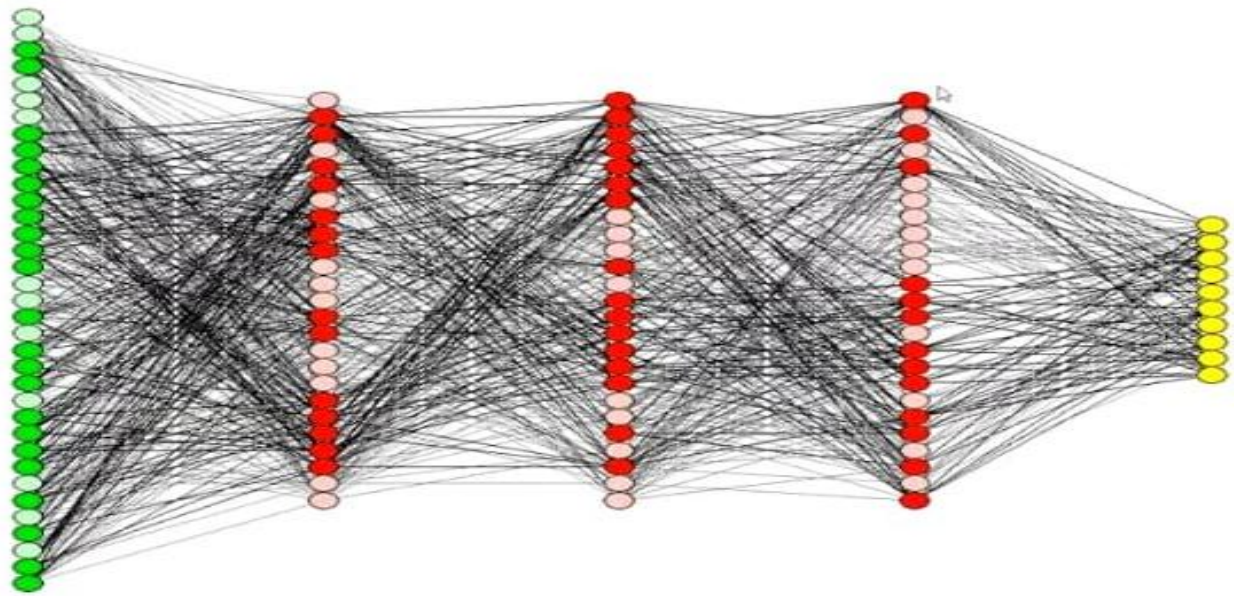
0.03 is the value we got from this derivative

Always remember when we go backwards or we do back propagation this value will get decreased

Suppose 0.25 is the initial weight of w1 and learning rate is 1

$$w\,new = w1\,old - n\frac{dL}{dw1\,old}$$

$$w\,new = 0.25 - 1 * 0.0069 = 0.2431$$



Exploding gradient

Solution for vanishing and exploding:LSTM

- Long Short-Term Memory (LSTM) networks are **an extension of RNN that extend the memory**. LSTM are used as the building blocks for the layers of a RNN. LSTMs assign data “weights” which helps RNNs to either let new information in, forget information or give it importance enough to impact the output.

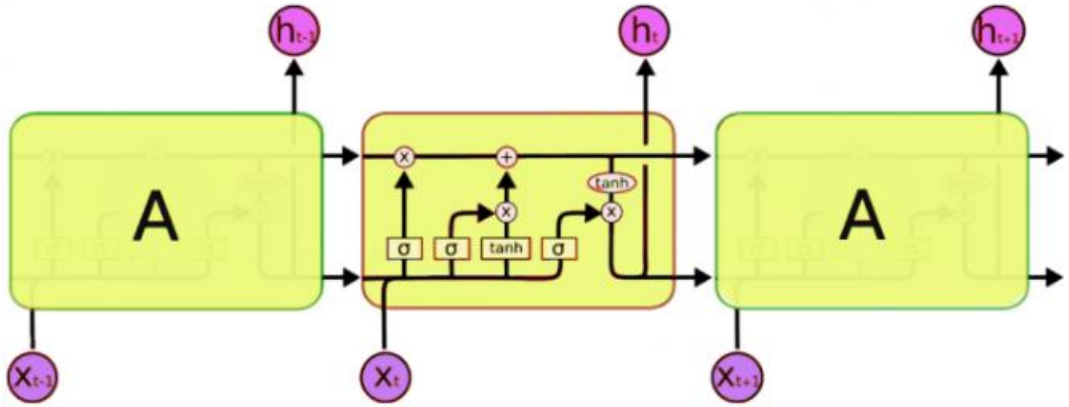
LSTMs on the other hand, make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

LSTMs on the other hand, make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

These dependencies can be generalized to any problem as:

1. The previous cell state *(i.e. the information that was present in the memory after the previous time step)*
2. The previous hidden state *(i.e. this is the same as the output of the previous cell)*
3. The input at the current time step *(i.e. the new information that is being fed in at that moment)*

Architecture of LSTMs



Now, this is nowhere close to the simplified version which we saw before, but let me walk you through it. A typical LSTM network is comprised of different memory blocks called **cells** (the rectangles that we see in the image). There are two states that are being transferred to the next cell; the **cell state** and the **hidden state**. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called **gates.** Each of them is being discussed below.

Forget Gate

Taking the example of a text prediction problem. Let's assume an LSTM is fed in, the following sentence:

Bob is a nice person. Dan on the other hand is evil.

As soon as the first full stop after "*person*" is encountered, the forget gate realizes that there may be a change of context in the next sentence. As a result of this, the *subject* of the sentence is *forgotten* and the place for the subject is vacated. And when we start speaking about "*Dan*" this position of the subject is allocated to "*Dan*". This process of forgetting the subject is brought about by the forget gate.