Chapter 3

3. Overview

This section will outline the approach, tools technologies used to navigate this project to completion.

3.1 Deep Learning

Concept of deep learning takes motivation from the human brain; deep learning tries to mimic the way the human brain learns and acts on the environment. Although it is yet to showcase human brain level capability across areas, it is already beating humans at particular tasks. As this technology progresses, we will see many changes in the way we live that are never thought of before. Below figure 6 shows the relationship between machine learning, deep learning and artificial intelligence.

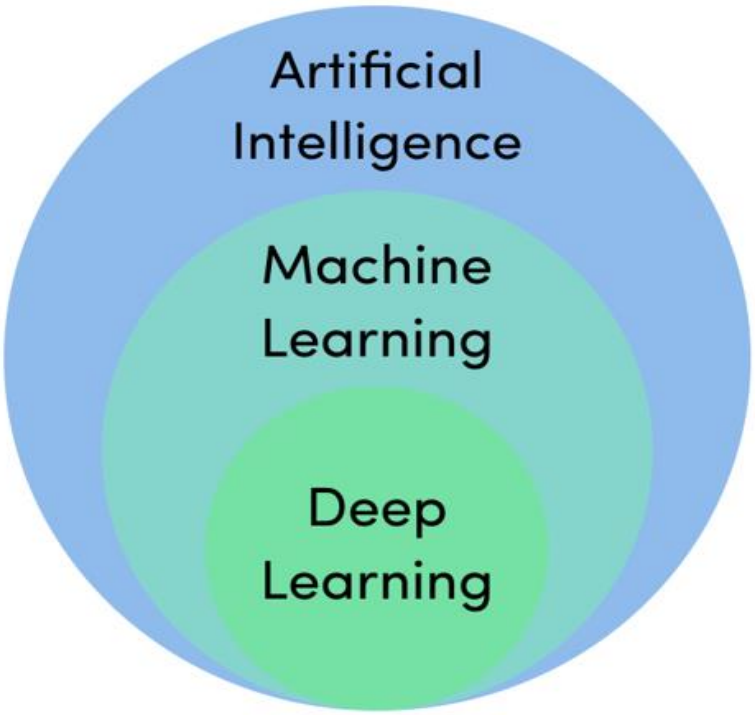

Figure 6 Representation of deep Learning w.r.t AI

A neuron is a basic building block of a neural network, during model training weight is associated with each neuron in the network. Loss is calculated at the last layer, and it is back propagated to adjust weights of the network. This improves accuracy of the model. Deep neural networks are very good at learning patterns compared to traditional machine learning algorithms.

3.2 CNN (Convolutional Neural Network)

Convolutional Neural Networks are at the heart of deep learning. CNNs are instrumental in quickly bridging the gap between humans and machine capabilities, the day is not very far away when machines will outperform humans in almost all areas. CNNs usually takes an Image as input and layer by layer it tries to identify the pattern in the image. We don't have to hand engineer many parameters; it automatically adjusts them. Any Image is broken into pieces as the image goes deeper into the neural network many convolution and pooling operations are performed.

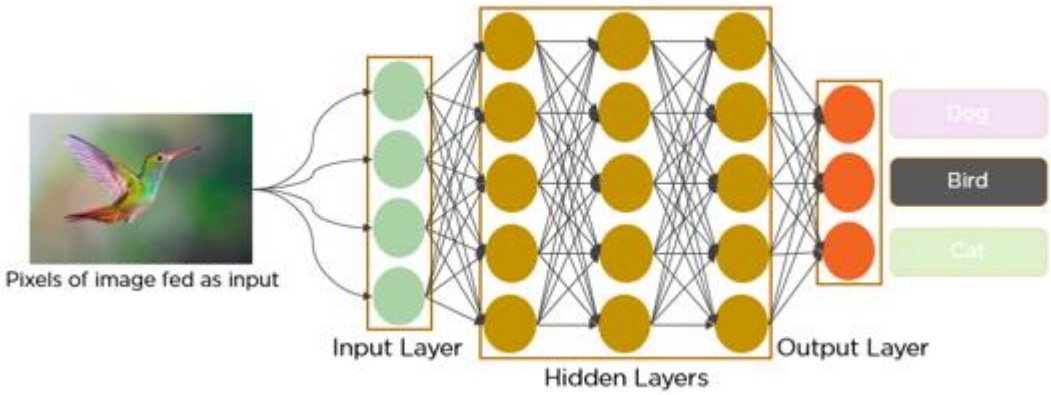

Figure 7 CNN Architecture

Above figure 7 represents CNN architecture. Images are basically a multidimensional matrix, so CNN takes advantage of this idea and by performing many matrix operations It produces optimum results.

3.3 Types of Pooling Layers

There are 3 types of pooling layers used in CNN.

1. Max Pooling Layer
2. Min Pooling Layer
3. Average Pooling Layer

3.3.1. Max Pooling Layer

In case of max pooling maximum pixel value is picked up from a group of pixels. It chooses the brightest pixel; Max pooling layer is suitable when the background is dark. It highlights the image against the dark background by choosing the only pixels that can be clearly seen.

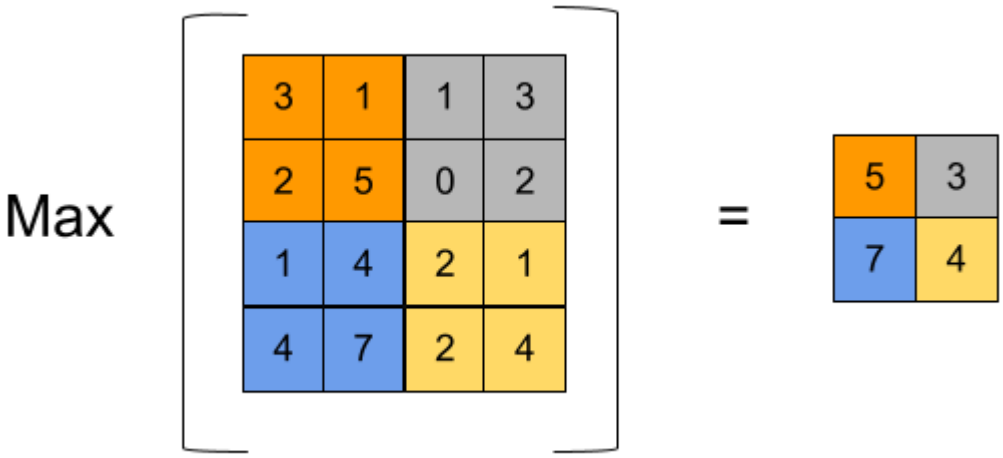

Figure 8 Representation of max pool operation

As we can see from figure 8 a 2 * 2 feature map is taken, and maximum value is taken from the available values.

3.3.2. Average Pooling Layer

As the name suggests It averages the values of pixels in a particular stride. So, the whole Image is down sampled to average pixel values. Average pooling layer helps to extract smooth features of an image.

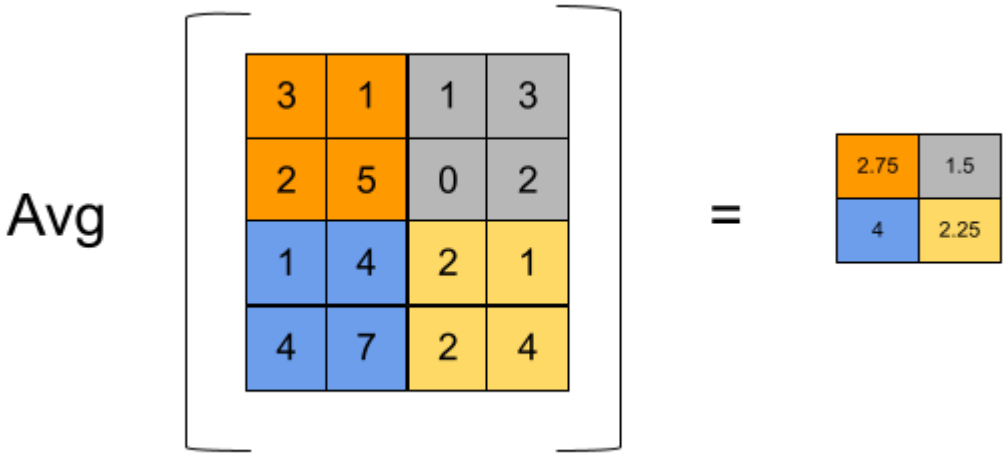

Figure 9 Representation of Average pool operation

As visible from the figure 9, the average pooling layer gives the average of the area covered by the feature map. Another use case of pooling layers is to reduce the dimensions of an image and save on the computation cost and reduce the number of trainable parameters. It also gives a summary of features in a feature map.

3.3.3. Min Pooling Layer

When we want to highlight an object in an image against the light background then the min pooling layer is used in a convolutional neural network.
It selects the features that are not very prominent and tries to highlight them for the neural network to learn.

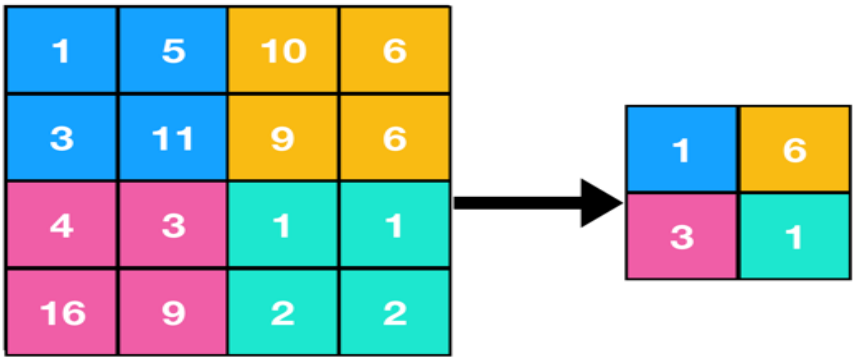

Figure 10 Representation of min pool operation

Here In the figure 10, we can see that 2 * 2 size feature map is taken and min value is picked up from the available values.

3.4 Types of Activation Functions used in this Project

1. Sigmoid
2. TanH
3. ReLU
4. LeakyReLU

Sigmoid

Sigmoid transforms the value of weights between 0 and 1. It is a non-linear activation function.

$$f(x) = \frac{1}{1+e^{-x}} \quad (5)$$

Equation 5 generates the output between 0 and 1.

TanH

TanH activation function is somewhat similar to sigmoid activation function, only difference is that it is symmetric around zero. So, its range is between -1 and 1.
It means that input to the next layer can be of opposite sign.

$$tanh(x) = 2 * sigmoid(2 * x) - 1 \quad (6)$$

Equation 6 generates the output between -1 and 1.

ReLU

ReLU is also a type of non-linear activation function. It is pronounced as Rectified Linear Unit. Main feature of ReLU is that it does not activate all the neurons, it activates only those neurons where value is positive and deactivates the neurons where the value is less than zero.

$$f(x) = max(0, x) \quad (7)$$

Equation 7 we can understand that ReLU discards negative values.

LeakyReLU

As the name suggests, it is nothing but an improved version of Rectified Linear Unit. In ReLU what happens is if the value is 0 or less than 0 i.e., < 0 then the neuron is deactivated so due to this there can be many dead neurons in a particular region to address this issue LeakyReLU uses extremely small linear component of a given variable.

$$f(x) = 0.01x, x < 0 \; or \; x, x >= 0 \quad (8)$$

As can be seen from equation 8 a small slope is introduced for values less than 0 so it avoids discarding all the negative values in this way.

3.5 Types of Layers Used in Model

1. Conv2d
2. ConvTranspose2d
3. BatchNorm2d
4. Flatten

Conv2d

This layer is a part of PyTorch deep learning library. It applies two-dimensional convolution over the input image, where all the features of the image are given like size, width, length, channels. A convolution operation is carried out on image represented as 2D matrix

These are the parameters used while using conv2d.

- Kernel Size
- Stride
- Padding
- Dilation
- Bias

ConvTranspose2d

In Pytorch this module is a gradient of conv2d. ConvTranspose2d is also known as deconvolution.

These are the parameters used with convTranspose2d

- Kernal_size
- Stride
- Padding
- Output padding

BatchNorm2d

BarchNorm2d is used to apply batch normalization over an input which is of 4 dimensions. The standard-deviation and mean are calculated for each dimension over the mini-batches and gamma and beta are learnable parameters of size C.

Parameters used in BatchNorm2d-

- Num_features

Flatten

Flatten layer is used to convert a 2D matrix to a one-dimensional vector. Its output is fed to a fully connected layer.