# TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER 1: INTRODUCTION

In corporations, institutes and offices, overwhelming volume of paper-based data challenges their ability to manage documents and records. Most fields have become computerized because computers can work faster and more efficiently than human beings. Paperwork is reduced day by day and computer system took place of it.

All the early paperwork has now become paperless work. To convert any handwritten, printed or historical documents (e.g. Book) into text document, either we have to type the whole document or scan the document. If the document size is very large, it becomes cumbersome and time consuming to type the whole document and there may be the possibilities of typing errors. Another option is scanning of document. Whenever document size is very large instead of typing, scanning of document is preferable. Forms can be scanned through scanner. Scanning is merely an image capture of the original document, so it can't be edited or searched through in any way and it also occupies more space because it is stored as an image. To edit or search from the document, document must be converted to doc (text) format.

Optical Character Recognition (OCR) is a process of converting scanned document into text document so it can be easily edited if needed and becomes searchable. Recognition engine of the OCR system interpret the scanned images and turn images of handwritten or printed characters into ASCII data (Machine readable characters). It occupies very less space than the image of the document. The document which is converted to text document using OCR occupies 2.35KB while the same document converted to an image occupies 5.38MB. So whenever document size is very large instead of scanning, OCR is preferable.

## 1.1 ARTIFICIAL NEURAL NETWORK (ANN)

Artificial neural networks (ANNs), a form of connectionism, are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the analytic results to identify cats in other images. They have found most use in applications difficult to express in a traditional computer algorithm using rule-based programming.

The long course of evolution has given the human brain many desirable characteristics not present in Von Neumann architecture include.

- Massive parallelism
- Distributed representation and computation
- Learning ability
- Generalization ability
- Adaptivity
- Inherent contextual information processing
- Fault tolerance, and

- Low energy consumption.

It is hoped that devices based on biological neural network will possess some of these desirable characteristics.

Inspired by biological neural networks, ANNs are massively parallel computing systems consisting of an extremely large number of simple processors with many interconnections. ANN model aim to use some "organizational" principles believed to be used in the humans.

### 1.1.1 BIOLOGICAL NEURAL NETWORK

A neuron is a special biological cell that processes information. It is composed of a cell body, or *soma*, and to type of out-reaching tree-like branches: the *axon* and the *dendrites*. The cell body has a nucleus that contains information about hereditary traits and a plasma that holds the molecular equipment for producing material needed by the neuron. A neuron receives signals (impulses) from other neurons through its dendrites (receivers) and transmits signals generated by its cell body along the axon (transmitter), which eventually branches into strands are the *synapses*. A synapse is an elementary structure and functional unit between two neurons. When the impulse reaches the synapse's terminal, certain chemicals called neurotransmitters are released. The neurotransmitters diffuse across the synaptic gap, to enhance or inhibit, depending on the type of the synapse, the receptor neuron's own tendency to emit electrical impulses. The synapse's effectiveness can be adjusted by the signal; passing through it so that the synapses can learn from the activities in which they participate. The dependence on the history acts as a memory, which is possibly responsible for the human memory.

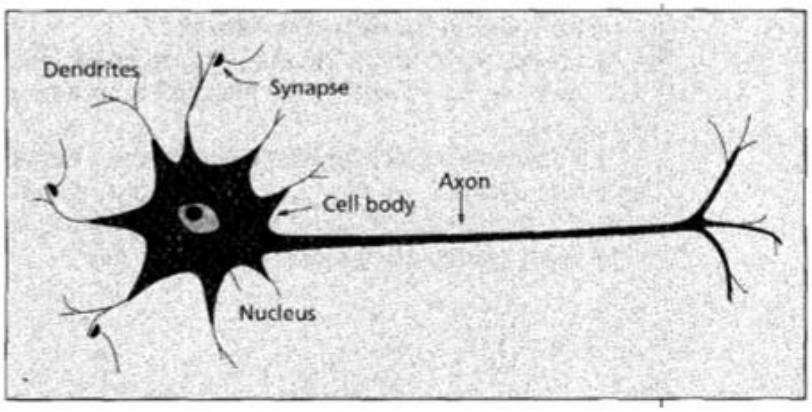



**Figure 1.1: A sketch of biological neuron**

Neuron communicate through a very short train of pulses, typically milliseconds in duration. The message is modulated on the pulse-transmission frequency. This frequency can vary from a few a few to several hundred hertz, which is a million times slower than the fastest switching speed in electronics circuits. However, complex perceptual decisions such as face recognition are typically made by humans by a few hundred milliseconds. These decisions are made by the network of neurons whose operational speed is a few milliseconds. This implies that a computation cannot take more than about 100 serial stages. In other words the brain runs parallel programs that are about 100 steps long for such perceptual tasks. This is known as *hundred step rule*. The same timing considerations show that the amount of information sent from one neuron to another must be very small. This implies that the critical information is not transmitted

directly, but captured and distributed in the interconnections- hence the name *connectionist* model is used to describe ANNs.

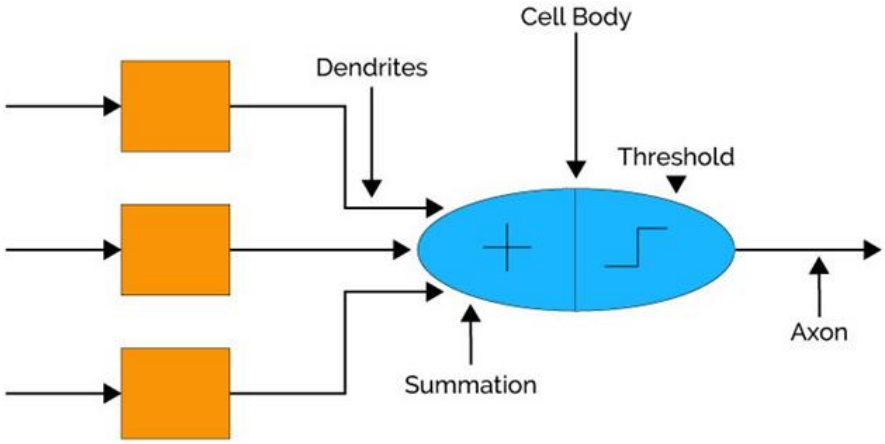



**Figure 1.2: Analogy of ANN with biological neurons**

### 1.1.2 NETWORK ARCHITECTURE

ANN can be viewed as weighted directed graphs in which artificial neurons are nodes and the directed edges (with weights) are connections between neurons outputs and neuron inputs.

Based on the connection pattern (architecture), ANNs can be grouped into two categories:

- *Feed-forward* networks, in which graphs have no loops, and
- *Recurrent (or feedback)* networks, in loops occur because of feedback connections.

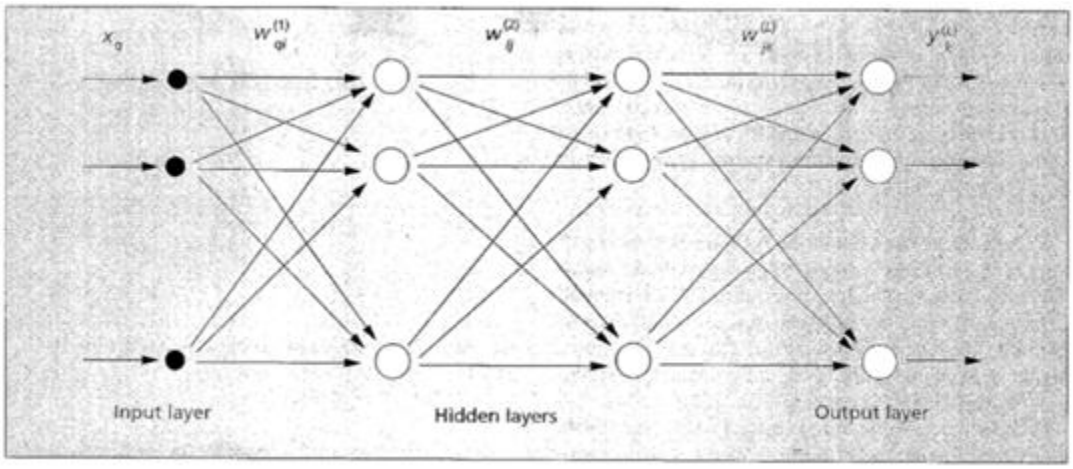



**Figure 1.3: A typical three layer feed-forward network architecture.**

In the most common family of feed-forward networks, called multilayer perception, neurons are organized into layers that have unidirectional connections between them. Figure below shows different networks of each category.

Different connectivities yield different neural network behaviors. Generally speaking, feed-forward networks are *static*, that is, they produce only one set of output values rather than a sequence of values from a given inputs. Feed-forward networks are memory-less in the sense

that their response to an input is independent of the previous network state. Recurrent, or feedback, networks, on the other hand, are dynamic systems. When a new input pattern is presented, the neuron outputs are computed. Because of the feedback paths, the inputs to each neurons are then modified, which leads the network enter the new state.

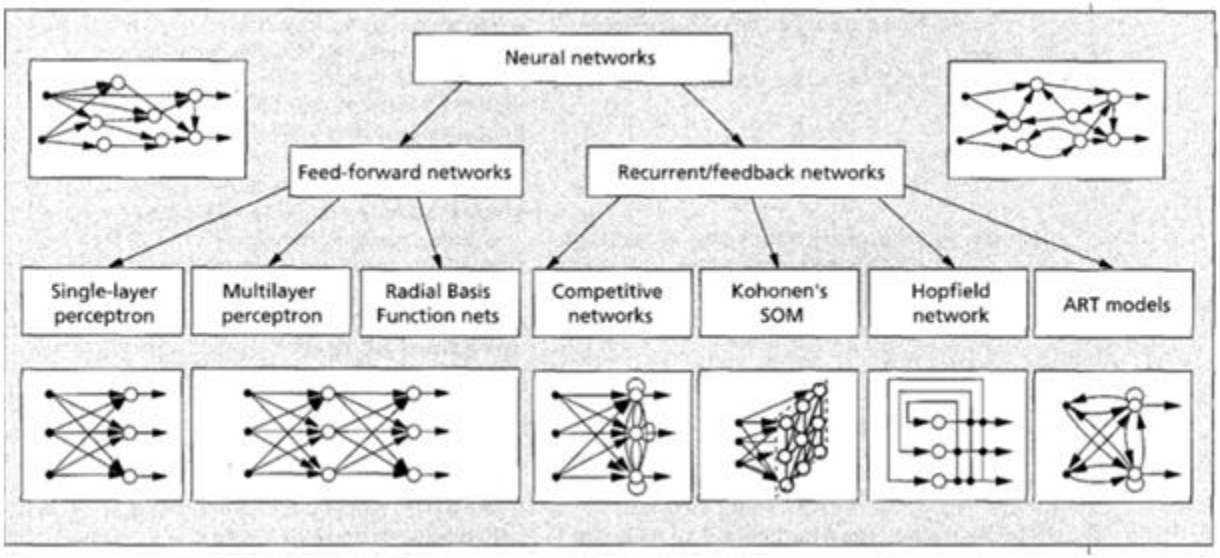



**Figure 1.4: A taxonomy of feed-forward and recurrent/feedback network architectures.**

Different network architectures require appropriate learning algorithms.

## 1.2 CONVOLUTIONAL NEURAL NETWORK (CNN)

A convolutional neural network (CNN or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. We can take the input image, define a weight matrix and the input is convolved to extract specific features from the image without losing the information about its spatial arrangement.

Another great benefit this approach has is that it reduces the number of parameters from the image. A convolved image has lesser pixels as compared to the original image. This dramatically reduces the number of parameters we need to train for the network.

We need four basic components to define a basic convolutional network.

- The Convolutional Layer
- The Pooling Layer [optional]
- The Dropout Layer [optional]
- The Fully Connected Layer

### Convolution Layer

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns

filters that activate when it detects some specific type of feature at some spatial position in the input.

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

The weight matrix behaves like a filter in an image extracting particular information from the original image matrix. A weight combination might be extracting edges, while another one might a particular color, while another one might just blur the unwanted noise.

The weights are learnt such that the loss function is minimized similar to an MLP. Therefore weights are learnt to extract features from the original image which help the network in correct prediction. When we have multiple convolutional layers, the initial layer extract more generic features, while as the network gets deeper, the features extracted by the weight matrices are more and more complex and more suited to the problem at hand.

**Pooling Layer**

Sometimes when the images are too large, we would need to reduce the number of trainable parameters. It is then desired to periodically introduce pooling layers between subsequent convolution layers. Pooling is done for the sole purpose of reducing the spatial size of the image. Pooling is done independently on each depth dimension; therefore the depth of the image remains unchanged. The most common form of pooling layer generally applied is the max pooling.

In figure 5, max pooling has been applied on the convoluted image. The max pooled image still retains the information that it's a car on a street. If you look carefully, the dimensions if the image have been halved. This helps to reduce the parameters to a great extent.

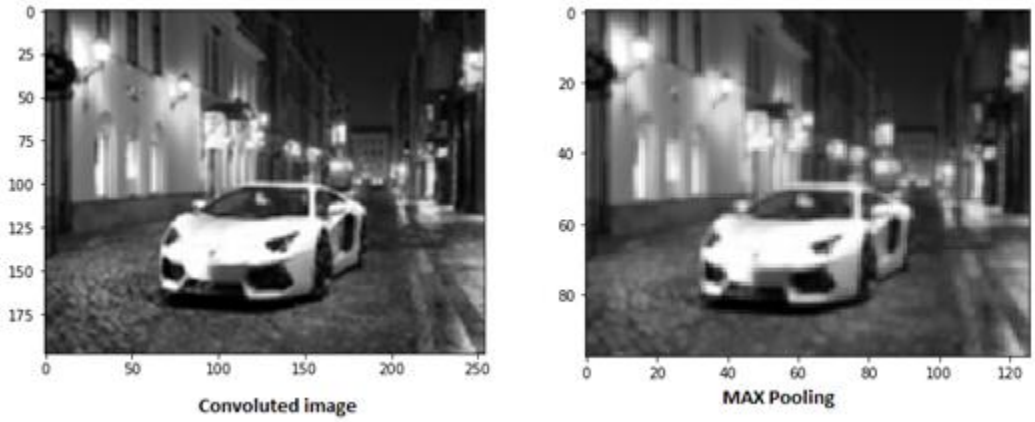



**Figure 1.5: Pooling**

Similarly other forms of pooling can also be applied like average pooling or the L2 norm pooling.

**Dropout Layer**

Simply put, dropout refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random. Ignoring means these units are not considered during a particular forward or backward pass.

More technically, At each training stage, individual nodes are either dropped out of the net with probability 1-p or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed.

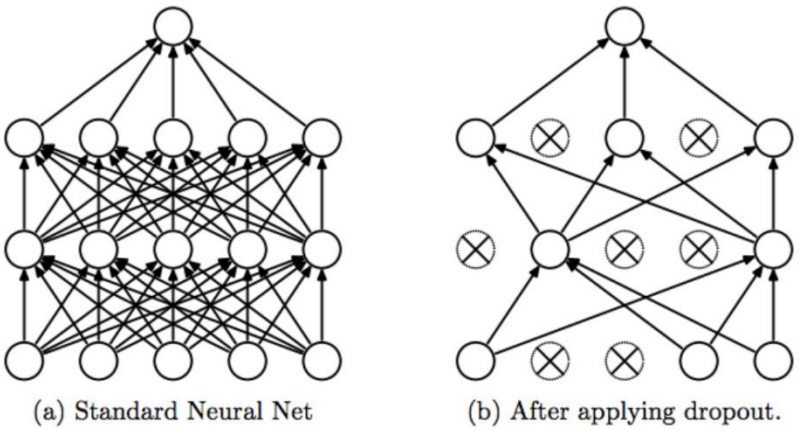



**Figure 1.6: Dropout**

**Fully Connected Layer**

After multiple layers of convolution and padding, the output in the form of a class. The convolution and pooling layers would only be able to extract features and reduce the number of parameters from the original images. However, to generate the final output we need to apply a fully connected layer to generate an output equal to the number of classes we need. It becomes tough to reach that number with just the convolution layers. Convolution layers generate 3D activation maps while we just need the output as whether or not an image belongs to a particular class. The output layer has a loss function like categorical cross-entropy, to compute the error in prediction. Once the forward pass is complete the backpropagation begins to update the weight and biases for error and loss reduction.

### 1.2.1 OUTLOOK OF THE ENTIRE NETWORK

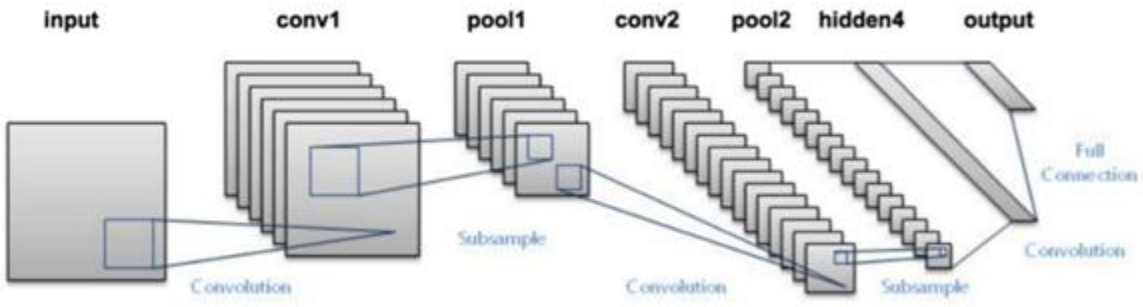



**Figure 1.7: Outlook of CNN**

CNN is composed of various convolutional and pooling layers. Let's see how the network looks like.

- We pass an input image to the first convolutional layer. The convoluted output is obtained as an activation map. The filters applied in the convolution layer extract relevant features from the input image to pass further.
- Each filter shall give a different feature to aid the correct class prediction. In case we need to retain the size of the image, we use same padding(zero padding), other wise valid padding is used since it helps to reduce the number of features.
- Pooling layers are then added to further reduce the number of parameters
- Several convolution and pooling layers are added before the prediction is made. Convolutional layer help in extracting features. As we go deeper in the network more specific features are extracted as compared to a shallow network where the features extracted are more generic.
- The output layer in a CNN as mentioned previously is a fully connected layer, where the input from the other layers is flattened and sent so as the transform the output into the number of classes as desired by the network.
- The output is then generated through the output layer and is compared to the output layer for error generation. A loss function is defined in the fully connected output layer to compute the mean square loss. The gradient of error is then calculated.
- The error is then backpropagated to update the filter(weights) and bias values.
- One training cycle is completed in a single forward and backward pass.

### 1.2.2 THE CONCEPT OF STRIDE AND PADDING

Normally the filter or the weight matrix was moving across the entire image, moving one pixel at a time. We can define it like a hyper-parameter, as to how we would want the weight matrix to move across the image. If the weight matrix moves 1 pixel at a time, we call it as a stride of 1.

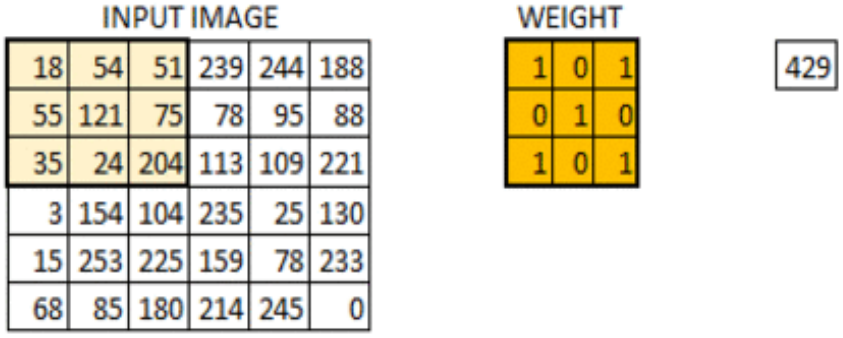



**Figure 1.8: Convolution of input image with weight**

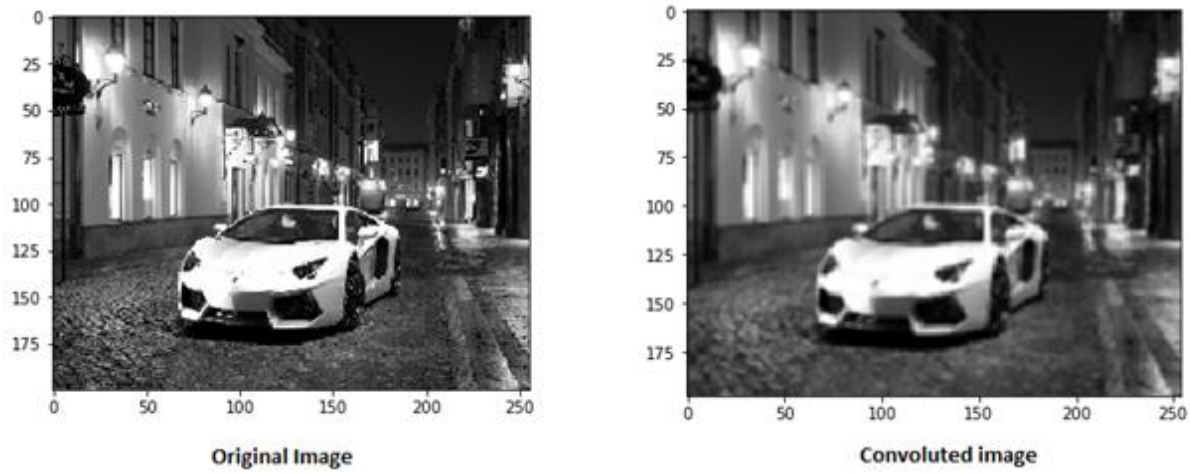



**Figure 1.9: Convoluted Image**

The size of image keeps on reducing as we increase the stride value. Padding the input image with zeros across it solves this problem for us. We can also add more than one layer of zeros around the image in case of higher stride values.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 18 | 54 | 51 | 239 | 244 | 188 | 0 |
| 0 | 55 | 121 | 75 | 78 | 95 | 88 | 0 |
| 0 | 35 | 24 | 204 | 113 | 109 | 221 | 0 |
| 0 | 3 | 154 | 104 | 235 | 25 | 130 | 0 |
| 0 | 15 | 253 | 225 | 159 | 78 | 233 | 0 |
| 0 | 68 | 85 | 180 | 214 | 245 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 1.10: Image with 0 padding**

We can see how the initial shape of the image is retained after we padded the image with a zero. This is known as same padding since the output image has the same size as the input (here 6*6)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 18 | 54 | 51 | 239 | 244 | 188 | 0 |
| 0 | 55 | 121 | 75 | 78 | 95 | 88 | 0 |
| 0 | 35 | 24 | 204 | 113 | 109 | 221 | 0 |
| 0 | 3 | 154 | 104 | 235 | 25 | 130 | 0 |
| 0 | 15 | 253 | 225 | 159 | 78 | 233 | 0 |
| 0 | 68 | 85 | 180 | 214 | 245 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

WEIGHT

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

139

**Figure 1.11: Convolution of padded image with weight**

### 1.2.3 MULTIPLE FILTERS AND THE ACTIVATION MAP

The depth dimension of the weight should be same as the depth dimension of the input image. The weight extends to the entire depth of the input image. Therefore, convolution with a single weight matrix would result into a convolved output with a single depth dimension. In most cases instead of a single filter(weight matrix), we have multiple filters of the same dimensions applied together.

The output from the each filter is stacked together forming the depth dimension of the convolved image. Suppose we have an input image of size 32*32*3. And we apply 10 filters of size 5*5*3 with valid padding. The output would have the dimensions as 28*28*10.

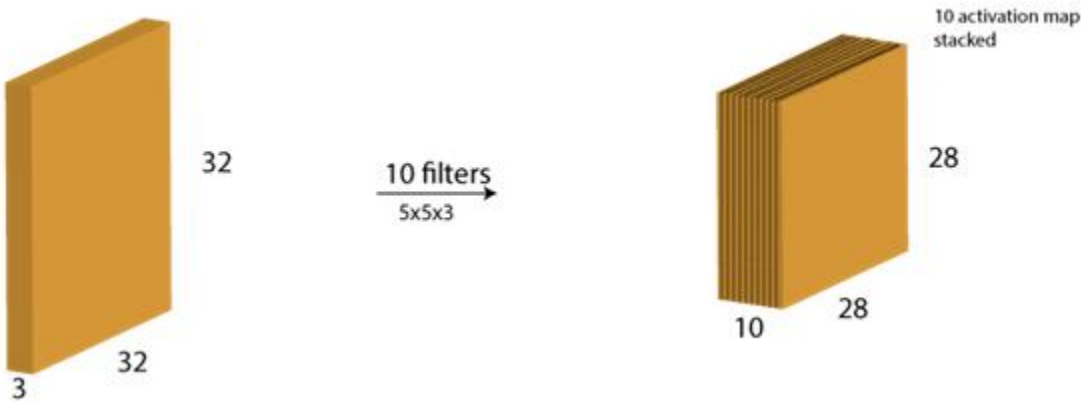



**Figure 1.12: Activation map**

This activation map is the output of the convolution layer.

## 1.3 APPLICATIONS

Followings are some of the areas, where ANN is being used. It suggests that ANN has an interdisciplinary approach in its development and applications.

**Speech Recognition**

Speech occupies a prominent role in human-human interaction. Therefore, it is natural for people to expect speech interfaces with computers. In the present era, for communication with machines, humans still need sophisticated languages which are difficult to learn and use. To ease this communication barrier, a simple solution could be, communication in a spoken language that is possible for the machine to understand.

Great progress has been made in this field, however, still such kinds of systems are facing the problem of limited vocabulary or grammar along with the issue of retraining of the system for different speakers in different conditions. ANN is playing a major role in this area. Following ANNs have been used for speech recognition −

- Multilayer networks
- Multilayer networks with recurrent connections
- Kohonen self-organizing feature map

The most useful network for this is Kohonen Self-Organizing feature map, which has its input as short segments of the speech waveform. It will map the same kind of phonemes as the output array, called feature extraction technique. After extracting the features, with the help of some acoustic models as back-end processing, it will recognize the utterance.

**Character Recognition**

It is an interesting problem which falls under the general area of Pattern Recognition. Many neural networks have been developed for automatic recognition of handwritten characters, either letters or digits. Following are some ANNs which have been used for character recognition −

- Multilayer neural networks such as Backpropagation neural networks.
- Neocognitron

Though back-propagation neural networks have several hidden layers, the pattern of connection from one layer to the next is localized. Similarly, neocognitron also has several hidden layers and its training is done layer by layer for such kind of applications.

**Signature Verification Application**

Signatures are one of the most useful ways to authorize and authenticate a person in legal transactions. Signature verification technique is a non-vision based technique.

For this application, the first approach is to extract the feature or rather the geometrical feature set representing the signature. With these feature sets, we have to train the neural networks using an efficient neural network algorithm. This trained neural network will classify the signature as being genuine or forged under the verification stage.

**Human Face Recognition**

It is one of the biometric methods to identify the given face. It is a typical task because of the characterization of "non-face" images. However, if a neural network is well trained, then it can be divided into two classes namely images having faces and images that do not have faces.

First, all the input images must be preprocessed. Then, the dimensionality of that image must be reduced. And, at last it must be classified using neural network training algorithm. Following neural networks are used for training purposes with preprocessed image −

- Fully-connected multilayer feed-forward neural network trained with the help of back-propagation algorithm.
- For dimensionality reduction, Principal Component Analysis (PCA) is used.

## 1.4 OBJECTIVES

- To create an OCR program that inputs an image containing handwritten text in real time, and recognizes the characters using CNN Onand write those characters in a text file.

# CHAPTER 2: DESCRIPTION

The project is divided into five steps as shown in figure.


**Figure 2.1: Block diagram of OCR**

## 2.1 READ IMAGE

An image is given as input to the system. The image can be in .jpg or .png format. The system reads the image in grayscale using the 'imread' function of opencv. This image is passed on to the next stage for pre-processing.

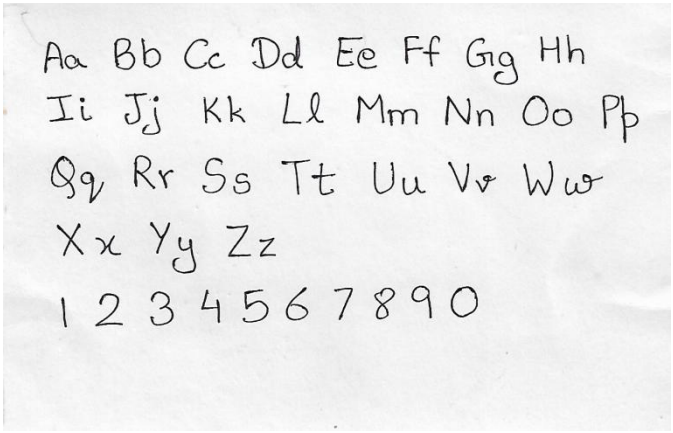



**Figure 2.2: Original Image**

## 2.2 PREPROCESSING

Preprocessing is done in three steps namely, noise removal, thresholding and rotation.

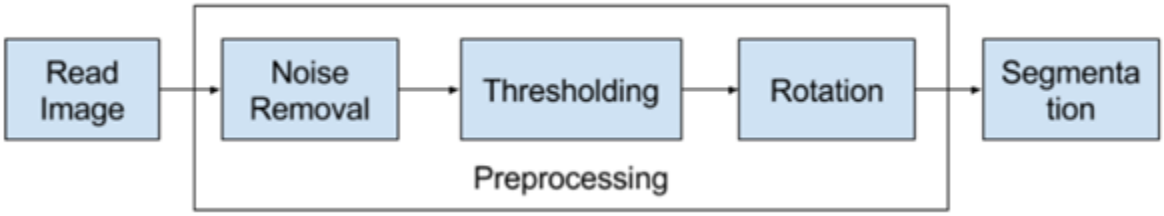



**Figure 2.3: Steps in preprocessing**

### 2.2.1 NOISE REMOVAL

The original image may contain some noise. We have considered the noise to be Gaussian Noise because this is the most common form of noise in nature. To remove such type of noise, there is a function in opencv which is called 'gaussianBlur'. It tries to smoothen the image.


**Figure 2.4: Gaussian Blur. Right: Origianl Image. Left: Output after Gaussian Blur**

In this approach a Gaussian Kernel is used. User should specify the width and height of the kernel which should be positive and odd. The user should specify the standard deviation in the X and Y directions, sigmaX and sigmaY respectively. Gaussian filtering is highly effective in removing Gaussian noise from the image.

### 2.2.2 THRESHOLDING

Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images.

The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity $I_{i,j}$ is less than some fixed constant T (that is, $I_{i,j} < T$), or a white pixel if the image intensity is greater than that constant.

We have used Otsu Thresholding which is an algorithm by Otsu. In global thresholding, there is an arbitrary value for threshold. But this value may not work for every input image. Otsu binarization takes a bimodal image (bimodal image is an image whose histogram has two peaks) and takes a value that lies approximately in the middle of the two peaks. In simple words, it automatically calculates a threshold value from image histogram for a bimodal image.

Then the image is inverted. Means the black pixels are converted to white and white are converted to black.

### 2.2.3 ROTATION

The image may have a tilt which needs to be corrected. For that, we first detect text pixels, and then create a minimum area rectangle enclosing the pixels. We find the tilt angle of the rectangle and then perform necessary rotation to correct the image.

‘findNonZero’, ‘minAreaRect’ and ‘getRotationMatrix2D’ functions were used for this purpose.

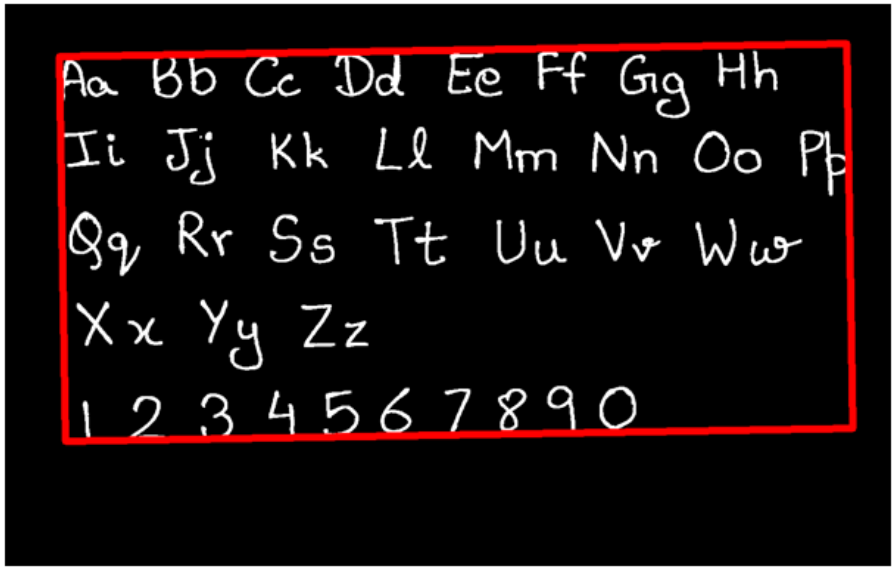



**Figure 2.5: Detecting tilt**

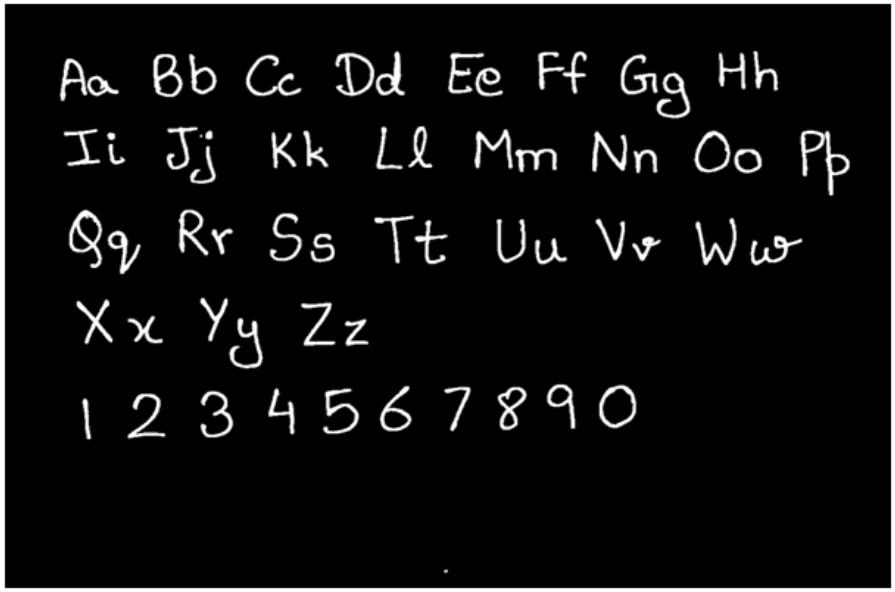



**Figure 2.6: After tilt correction**

## 2.3 SEGMENTATION

Segmentation is also done in three steps. First lines are segmented, then words are segmented line by line. After that characters are segmented in each word.

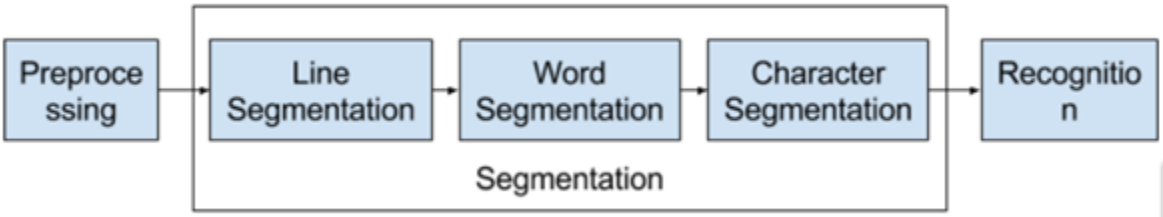



**Figure 2.7: Steps in segmentation**

### 2.3.1 LINE SEGMENTATION

The whole 2D image is converted to a 1D matrix by taking horizontal projection of each row. The projection contains average of each row. Each element contains value between 0 and 255. Value 0 means space and other value means text. Spaces are present as a continuous streak of 0s. This information is used to find out a threshold height lines in the image.

The threshold is used to find out if the continuous streak of 0s is due to separation in consecutive lines or due to any noise. Then y coordinates of each line is found out. Lines are drawn in the image. This image is saved for the user for debugging and visualization purposes.

Number of lines are also counted and returned to the user.

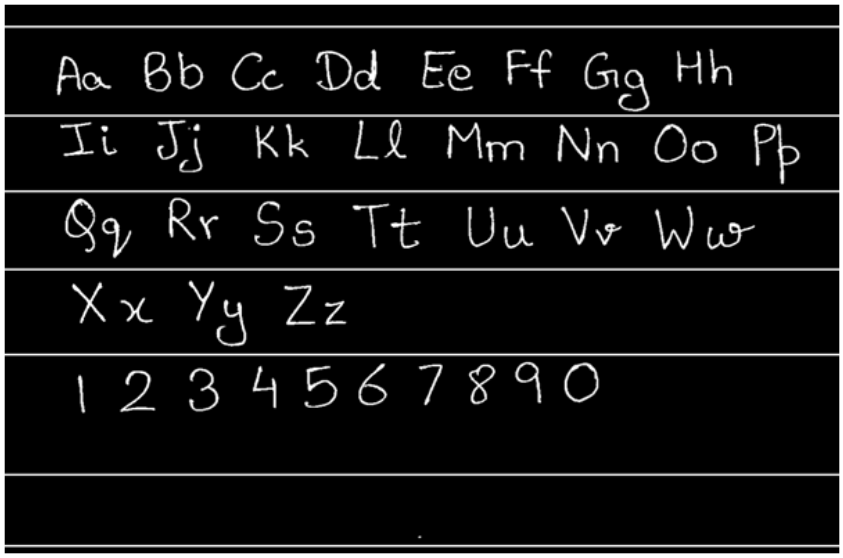



**Figure 2.8: Line segmentation**

### 2.3.2 WORD SEGMENTATION

Word segmentation is similar to line segmentation. The program takes one line at a time. In each line, The 2D image is converted to 1D matrix by taking vertical projection of the image. This 1D matrix is processed and an image is returned to the user. Number of words in each line is also counted and returned to the user.


**Figure 2.9: Word segmentation**

### 2.3.3 CHARACTER SEGMENTATION

Character segmentation has two parts. First is letter segmentation part and the second is correcting the bounding boxes of some letters.

In letter segmentation, we find the contour. A contour is defined as an outline representing or bounding the shape or form of something. We find the area of each contour which is then used to find out a threshold area. Every contour is then compared with the threshold. If contour area is less than the threshold area, that means it is a noise and is thus ignored. Other contours are taken and rectangles are drawn around them.

Bounding boxes of some characters need to be corrected. These letters are i, j, .(full stop) and ,(comma). All of these characters have small parts. First we detect if the part lies in the lower half of the line. This may be a full stop or comma. These characters are 0 padded on top. Else if the dot lies in the upper half, this may be a part of 'i', or 'j'. Its second part is found by comparing the x coordinate value. And both of them are merged.

Bounding boxes are drawn on the image and returned to the user for debugging and visualization. Now, these rectangles are reshaped to images of 28*28 matrices.

## 2.4 RECOGNIZE

In this project, we have used a Convolutional Neural Network (CNN) for recognition of characters. CNNs have repetitive blocks of neurons that are applied across space (for images) or time (for audio signals etc). For images, these blocks of neurons can be interpreted as 2D convolutional kernels, repeatedly applied over each patch of the image. For speech, they can be seen as the 1D convolutional kernels applied across time-windows. At training time, the weights for these repeated blocks are 'shared', i.e. the weight gradients learned over various image patches are averaged.

We have trained our CNN for 28*28 sized images with an accuracy of 87.18%. The neural network consists of 7 layers in this sequence: Two convolutional layers, one pooling layer, two dropout layers and two fully connected layers.

The first convolutional layer, or the input layer takes an image of 28*28 pixels. The output is then passed onto another convolutional layer, followed by a pooling layer to reduce the number of generated outputs. Some neurons are then dropped in the dropout layer. The output generated till now is a 2D matrix. This matrix is flattened and passed onto the fully connected layer. The output is again fed to a dropout and then a fully connected layer. The fully connected layer is the output layer. It gives a dictionary of character and its confidence score as output.

EMNIST (Extended Modified National Institute of Standards and Technology) dataset was used to train the neural network. The EMNIST dataset is a set of handwritten letters and digits derived from the NIST Special Database 19 and converted to a 28x28 pixel image format. The dataset consists of 814,255 characters.

## 2.5 GENERATE OUTPUT

The output of each character in a word are entered in a text file. When a word is over, a blank space is entered and when a line is finished, a new line character '\n' is entered in the document.

Finally this output text file is opened and the program terminates.

# CHAPTER 3: COMPARISON BETWEEN K-NN, ANN AND CNN

The K-nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). It has been used in statistical estimation and pattern recognition in the beginning of 1970's as a non-parametric technique.

K-nn needs to be carefully tuned, the choice of K and the metric (distance) to be used are critical. It is also very sensitive to bad features (attributes) so feature selection is also important. K-nn is also sensitive to outliers and removing them before using k-nn tends to improve results. Still k-nn is very low on accuracy (as compared to other deep learning techniques such as ANN, CNN) and might have been used historically, but now it is not the state of the art algorithm for classification.

Artificial neural networks are one of the main tools that is being used after k-nn in machine learning. As the "neural" part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

Advantages of ANN over k-nn are-

1. Rich feature space (Some people have tried to overcome this by kernelizing k-nn, but k-nn can perform poorly in high dimensional space. It tends to work better in low-dimensional spaces, so there are no guarantees that kernelizing k-nn will give any improvements).
2. Faster deployment. While ANN takes significant training time, the computational requirements for classifying the instances is quite minimal.

Artificial neural networks even have a number of problems that make them less ideal for certain cases. For example, imagine a case where we wanted to classify images of handwritten digits. An image is just a 2D array of pixel intensity values, so a small 28×28 pixel image has a total of 784 pixels.

If we wanted to classify this using an ANN, we would flatten the 2D array into a 1D vector of size 784 and treat each pixel as a separate input neuron. So for a small image, we already have almost 1000 parameters to learn, and that is just for a single layer. To make things worse, images also contain three separate colour channels, so we actually have 3x as many input neurons per image. As input images approach more realistic sizes, the parameter count becomes very large.

ANNs also assume feature independence. This assumption is generally not tenable in most real world data and can hurt the performance of the model. In the context of images, this assumption states that the input features (i.e. the individual pixels) are unrelated to each other. However, this is not the case with the vast majority of images because pixels that are close each other likely belong to the same object/visual structure and should be treated similarly. There is nothing in the architecture of an ANN that allows that proximity to be learned, or at least accounted for.

These issues (along with many others) mean image classification problems are typically handled with Convolutional Neural Networks (CNN).

While still feedforward in nature, convolutional neural networks use a different network architecture to ANNs, and use kernel convolution as the primary operation. There are two additional layer types that need to be considered with a CNN: convolution and pooling.

**Convolution layers**

Convolution layers convolve a kernel over some input to produce a feature map (sometimes called an activation map):

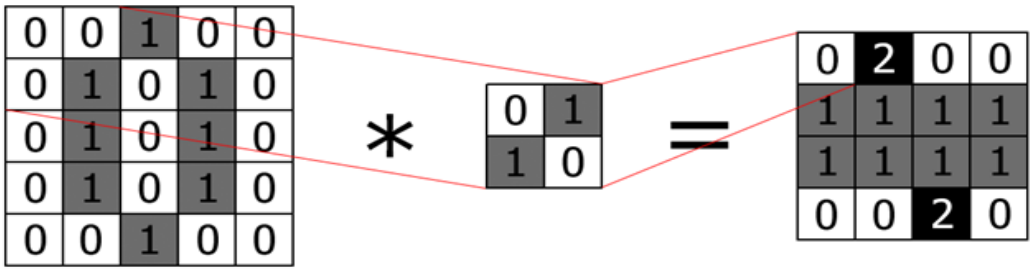



**Figure 3.1: Convolution Layer**

Imagine we were classifying grayscale handwritten digits of size 5×5. The input would be a 2D array containing the pixel intensity values. We then specify a kernel of a particular size (2×2 in this case) and convolve it over the input to produce a 4×4 feature map as the output. This output feature map is then passed (as input) into the next layer. Convolution will reduce the size of the input unless we pad the input with 0 values (which we didn't do in the example above).

A CNN, like other architectures previously discussed, is learning a set of weights. The weights in this case are the parameter values within the kernels. In essence, the network is learning the best kernel for separating the classes. There are a number of hyper parameters for a convolution layer:

1. Number of kernels, which produce the same number of feature maps. In the example above we only have 1 kernel, which produces 1 feature map.
2. Kernel size
3. Stride, which is the number of spatial steps to move the kernel during convolution. Stride 1 will move the kernel 1 pixel at a time. Stride 2 will skip every other pixel.
4. Padding, which is typically used to control the size of the feature map.

It is extremely important to note that the inputs and kernels are typically 3-dimensional. If the above input was a colour image, its size would be 5x5x3. The kernel always has the same depth as the input it is convolving over, so in the example the kernel would be 2x2x3. In other words, the kernel learns 3 2×2 spatial filters, one for each input channel. Each spatial filter contains a different set of weights. Finally, convolving a 3D kernel over a 3D input volume will produce a single 2D feature map.
To drive the point home:

- Input: 5x5x3 colour images
- 10 3×3 kernels (given the 3 channels of the input, each kernel is actually 3x3x3)
- Convolution produces the same number of feature maps as there are kernels, so we get 10 different 4×4 feature maps
- This 4x4x10 volume is then passed onto the next layer (if we stacked 2 convolution layers together, this volume is treated in the same way as the original 5x5x3 image, only with a smaller spatial extent and more depth channels)

ANN treats each image pixel as a separate neuron with its own parameter to be learned. Convolution layers allow the entire image to share the same kernel parameters, which drastically reduces the parameter count. Furthermore, the inputs are not treated independently (like in an ANN) because a kernel is looking at a number of surrounding pixels simultaneously. This allows the network to learn spatial structure. Of course, the input doesn't have to be an image. If the input was time series data, for example, the kernel would be learning temporal structure instead.

**Pooling layers**

A pooling layer down samples any volume that it receives. If we took the convolution layer in the figure above and passed it directly into a pooling layer, then we get the following:

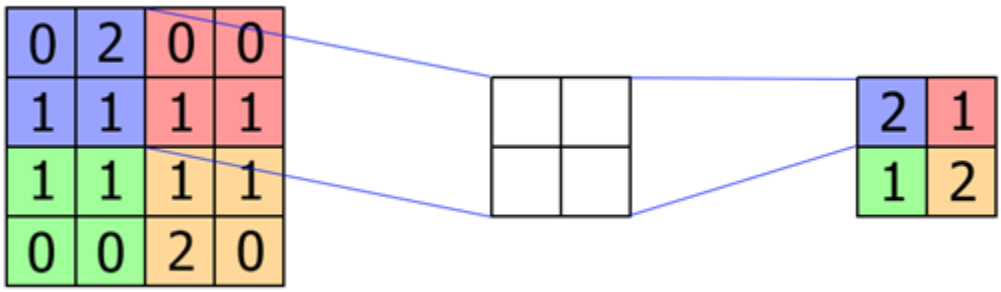



**Figure 3.2: Pooling Layer**

We choose a (parameter-less) kernel of a particular size and a stride (2×2 kernel with stride 2 in this example). Max-pooling is the most common form of pooling in CNNs. As we stride the kernel over the input, we simply take the maximum value within the kernel and set it as the output value. In the figure above you can see we've reduced the input size by half, which reduces the memory footprint of the rest of the network.

**Note**: Pooling is done separately for each depth slice of the (typically) 3D input volume.

**Fully connected layers**

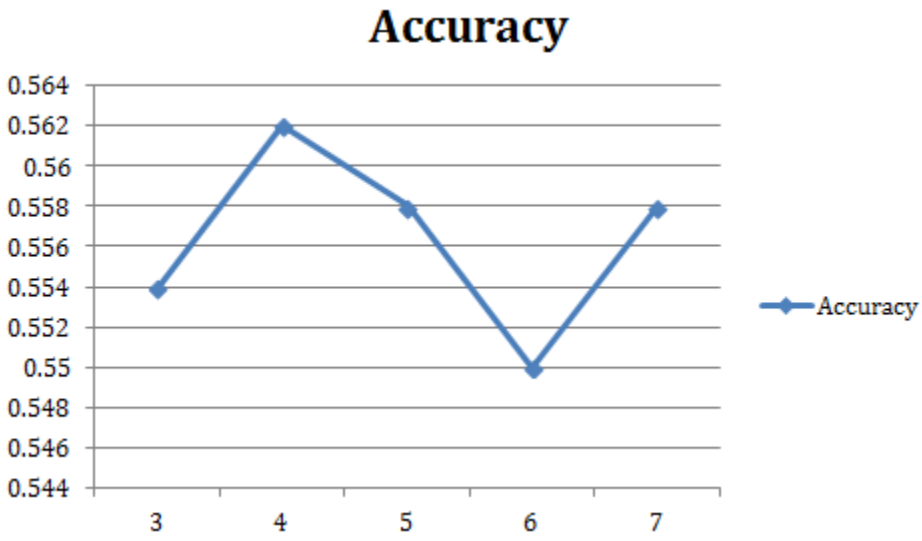
Finally, fully connected layers are typically added to CNN architectures. This is the same fully connected layer type that can be found in traditional ANNs. It flattens the entire input volume into a single vector. Each element is treated as a separate input neuron. If we stack multiple layers together then we have the ANN architecture. Fully connected layers are typically placed at the end of a network and connect to the classifier layer for prediction.

# CHAPTER 4: OBSERVATIONS

## 4.1 TRAINING:

While training the **k-nn**, we got the following accuracy for different values of 'K':

k = 3: accuracy = 0.554
k = 4: accuracy = 0.562
k = 5: accuracy = 0.558
k = 6: accuracy = 0.55
k = 7: accuracy = 0.558
Max accuracy: 0.562

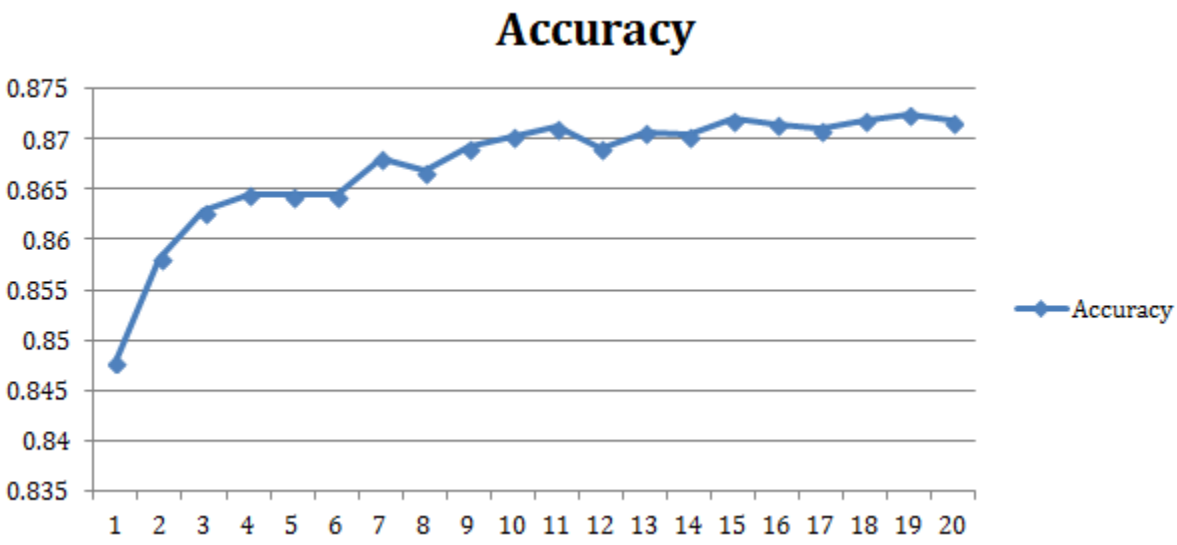



**Graph 4.1: Accuracy vs k (k-nn)**

While training the **CNN**, we got the following accuracy in different epochs:

Epoch 1/20: accuracy = 0.8478
Epoch 2/20: accuracy = 0.8581
Epoch 3/20: accuracy = 0.8628
Epoch 4/20: accuracy = 0.8645
Epoch 5/20: accuracy = 0.8644
Epoch 6/20: accuracy = 0.8644
Epoch 7/20: accuracy = 0.8681
Epoch 8/20: accuracy = 0.8668
Epoch 9/20: accuracy = 0.8692
Epoch 10/20: accuracy = 0.8703
Epoch 11/20: accuracy = 0.8712
Epoch 12/20: accuracy = 0.8691
Epoch 13/20: accuracy = 0.8707

Epoch 14/20: accuracy = 0.8704
Epoch 15/20: accuracy = 0.8720
Epoch 16/20: accuracy = 0.8715
Epoch 17/20: accuracy = 0.8710
Epoch 18/20: accuracy = 0.8719
Epoch 19/20: accuracy = 0.8724
Epoch 20/20: accuracy = 0.8718
Test accuracy: 0.8718




**Graph 4.2: Accuracy vs Epoch (CNN)**

## 4.2 TEST 1:

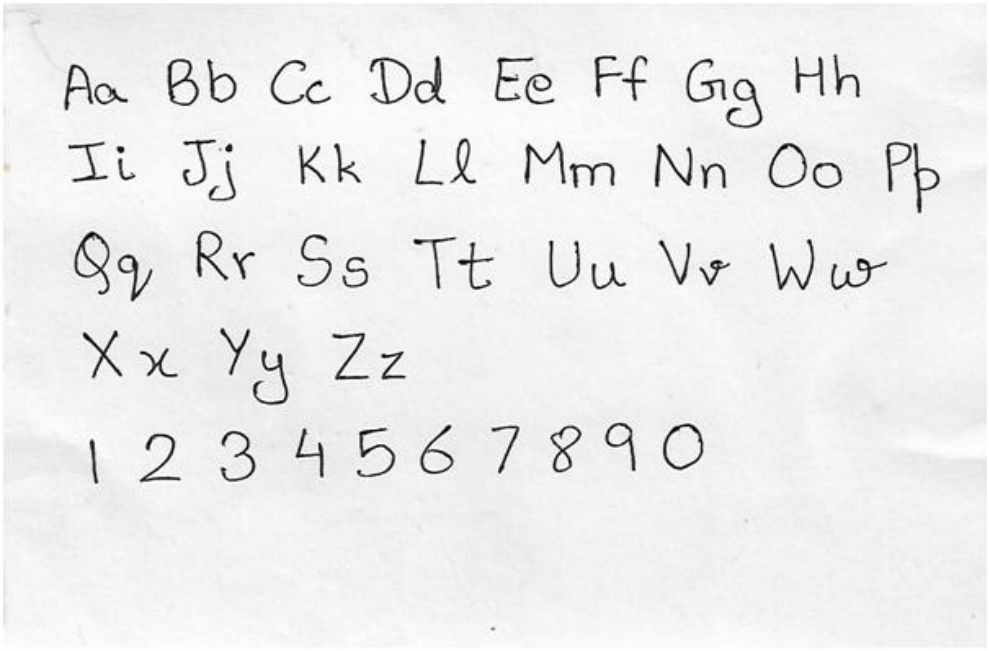



**Figure 4.1: Input Image**

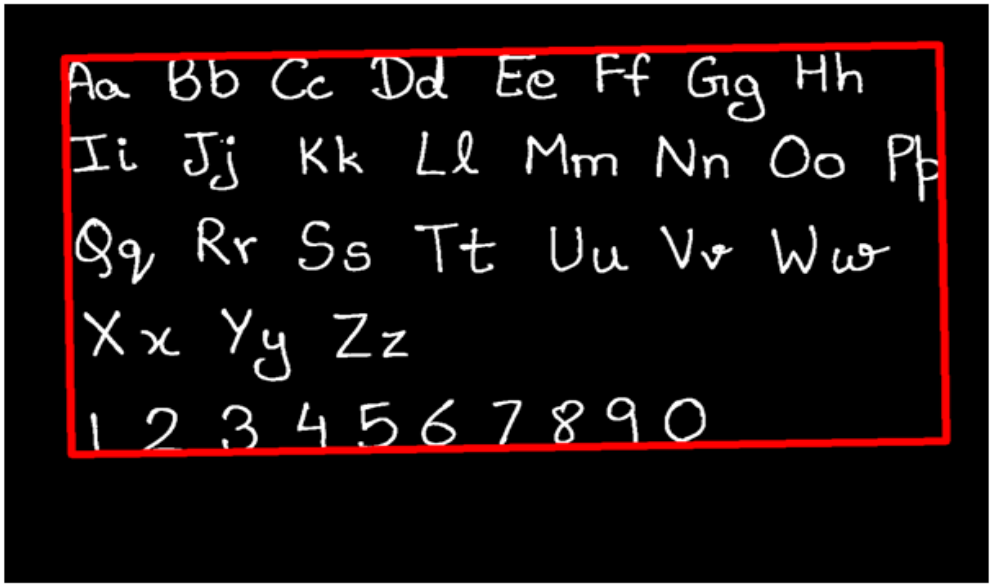



**Figure 4.2: Tilt Detection**

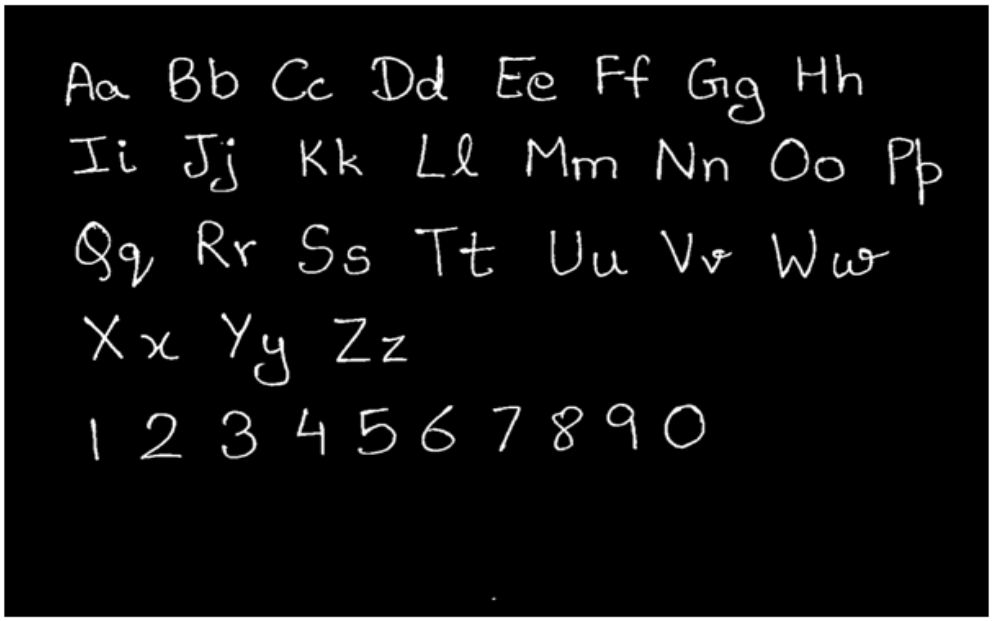



**Figure 4.3: After tilt correction**

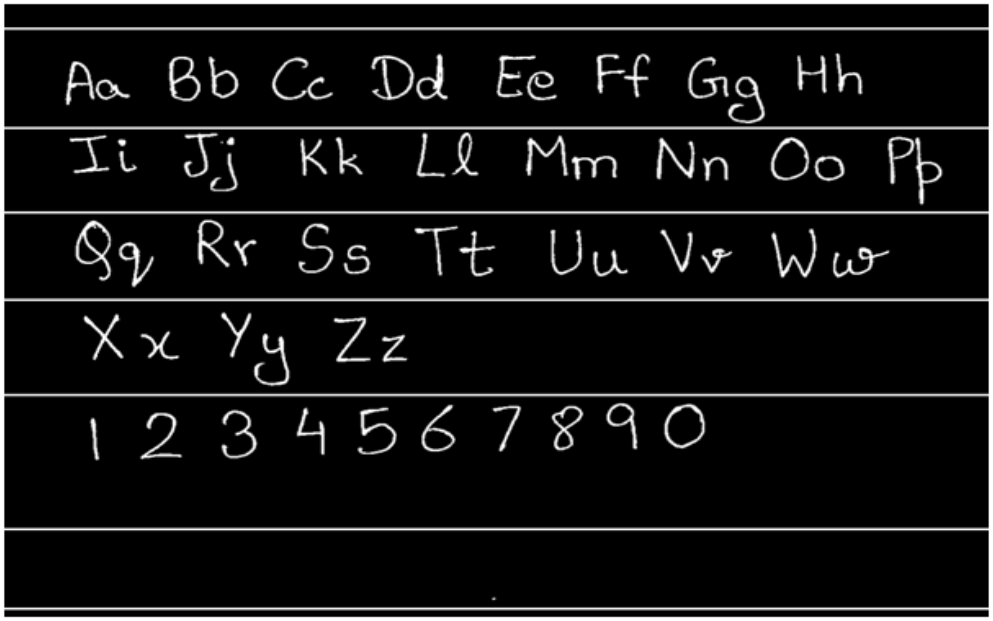



**Figure 4.4: Line Segmentation**

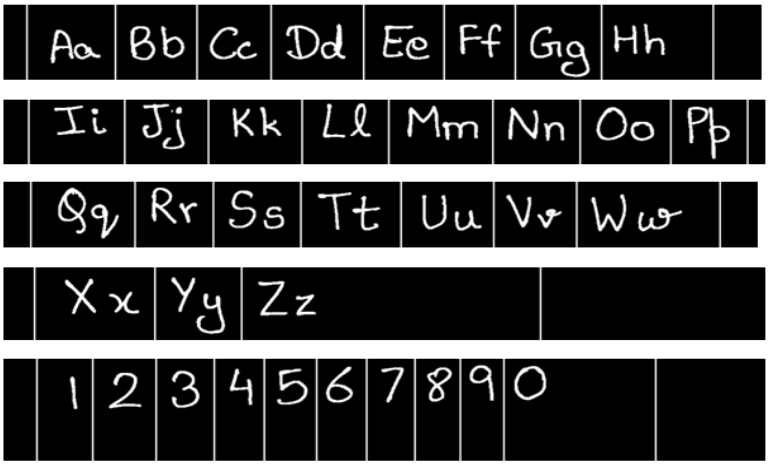


**Figure 4.5: Word Segmentation**

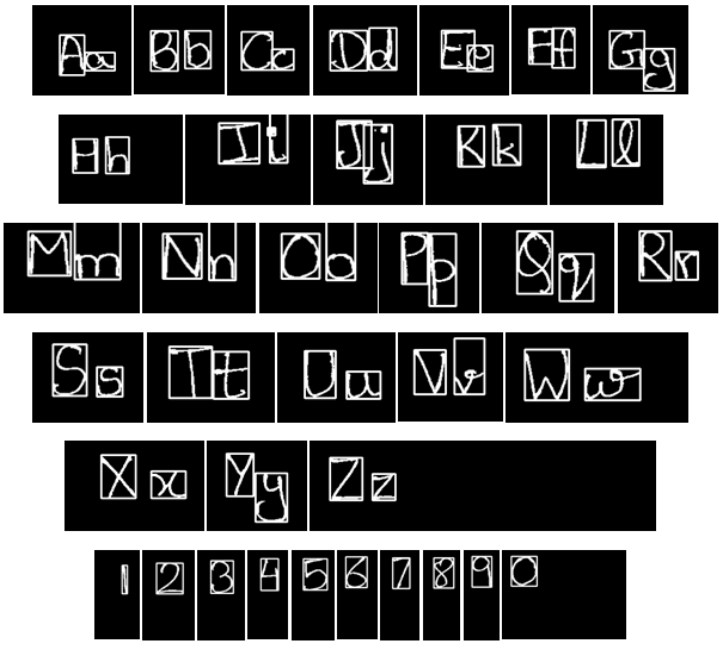


**Figure 4.6: Character Segmentation**

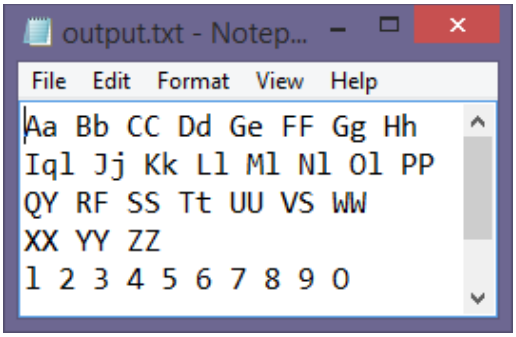



**Figure 4.7: Final Output**

### 4.3 TEST 2:

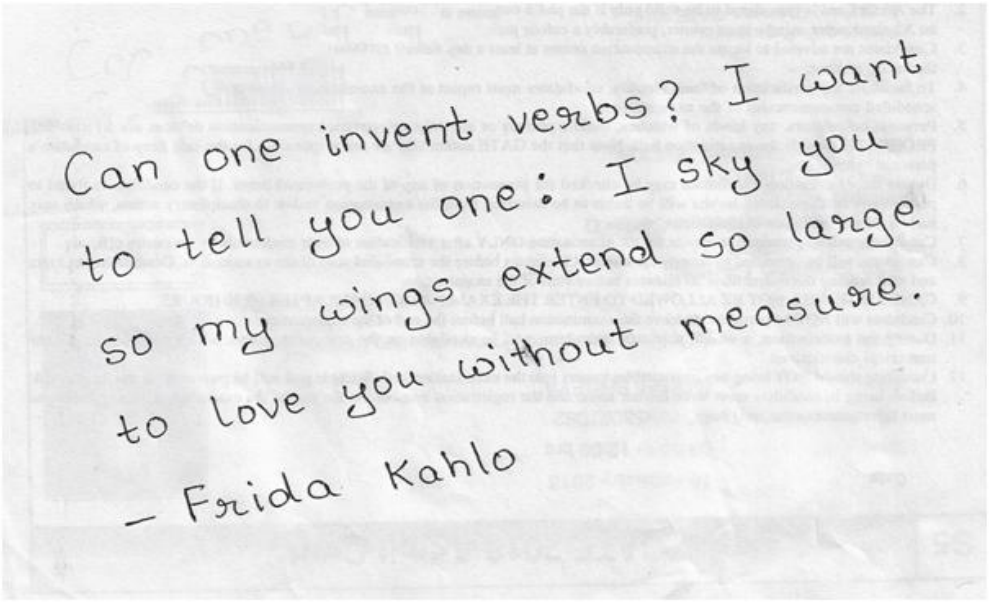



**Figure 4.8: Input Image**

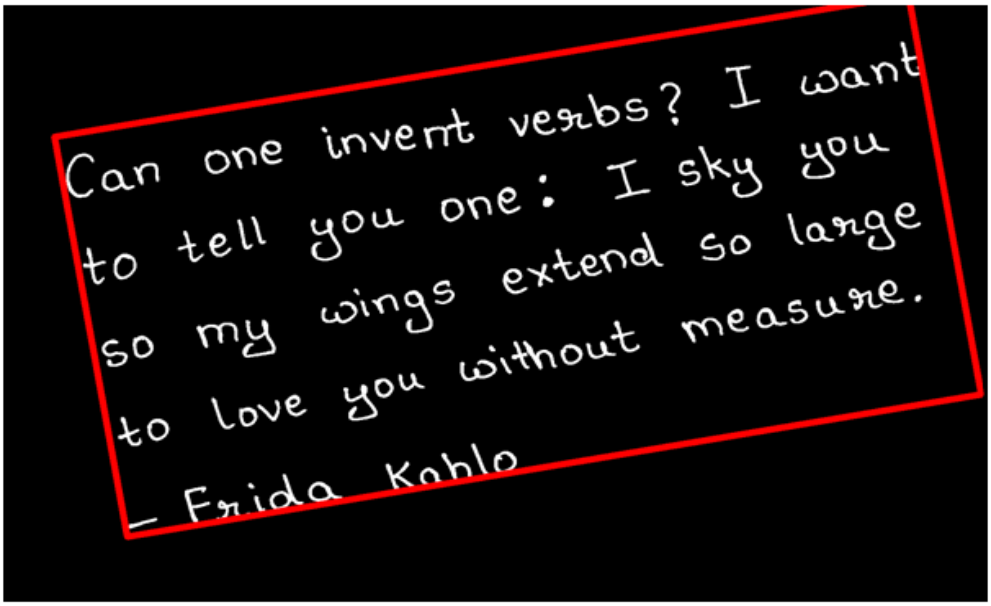



**Figure 4.9: Tilt Detection**

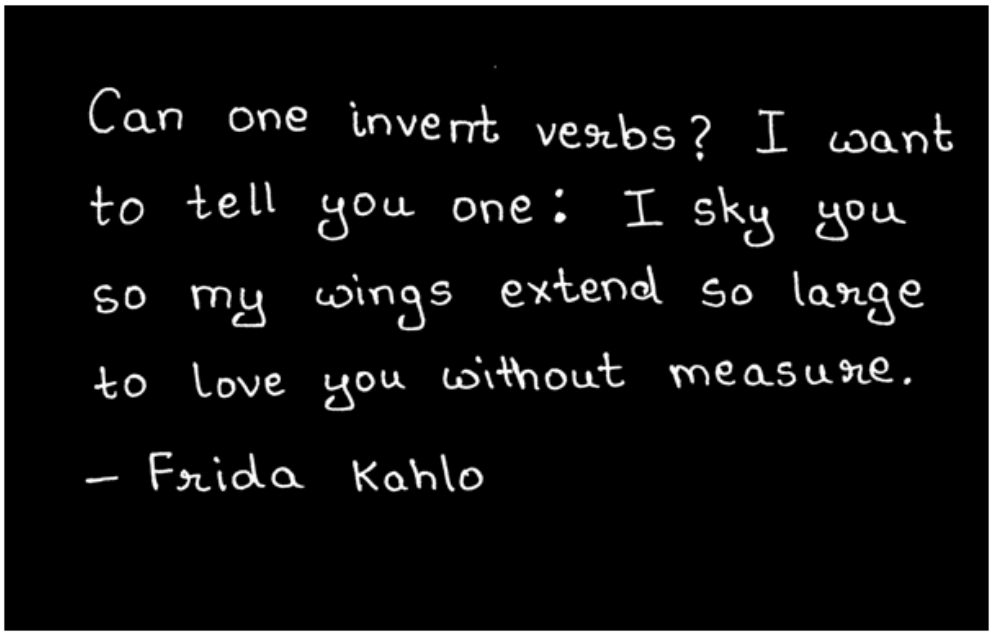



**Figure 4.10: After tilt correction**

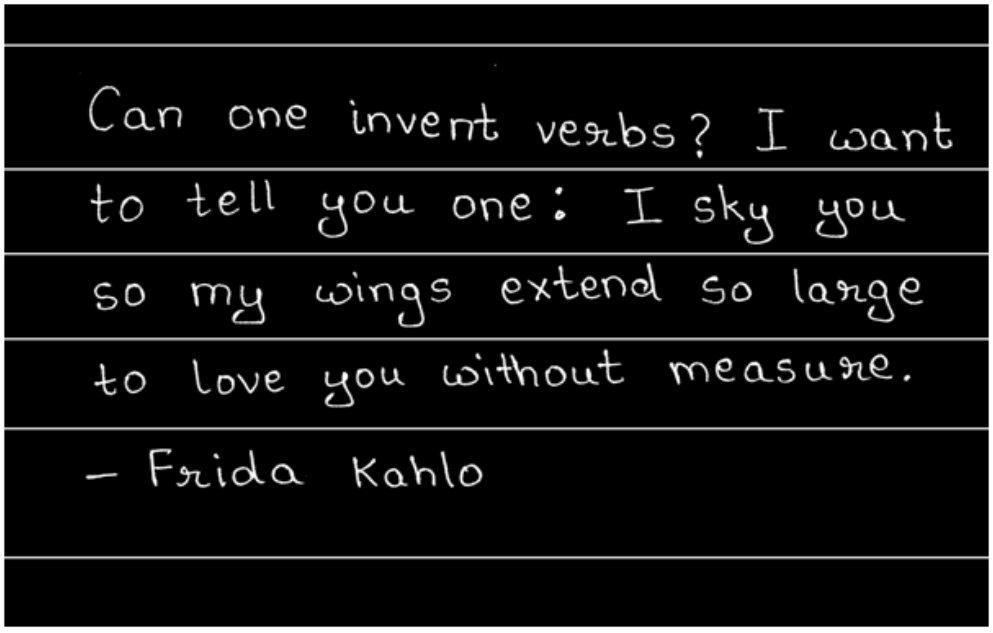



**Figure 4.11: Line Segmentation**

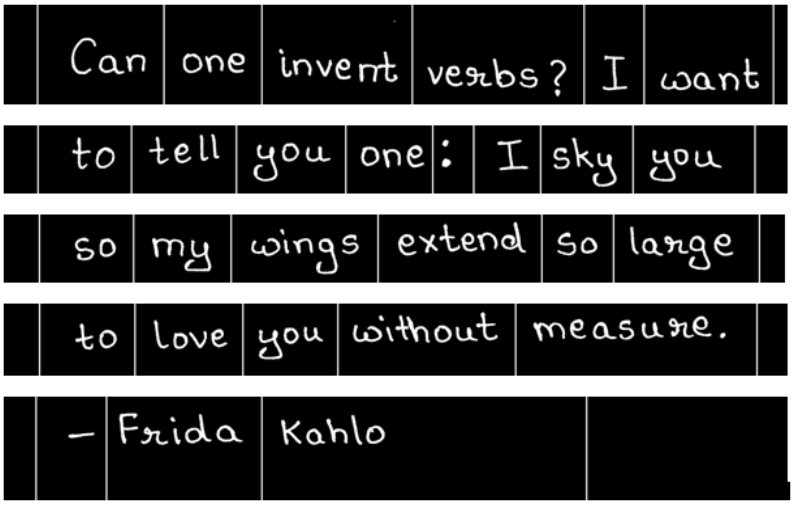



**Figure 4.12: Word Segmentation**

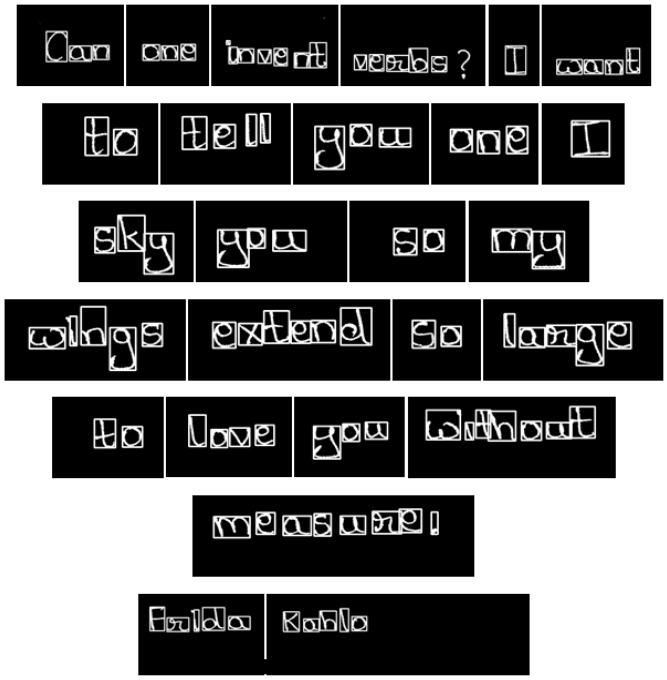



**Figure 4.13: Character Segmentation**

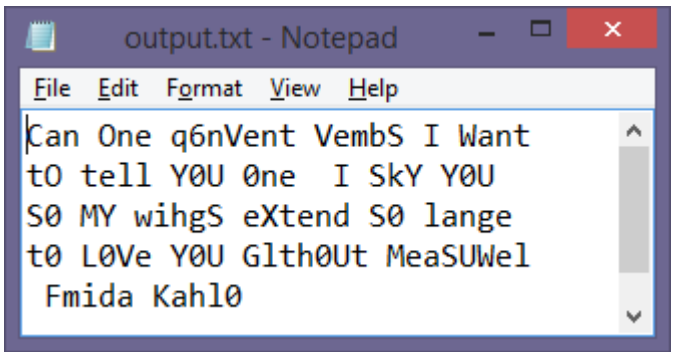



**Figure 4.14: Final Output**

# CHAPTER 5: RESULTS AND CONCLUSIONS

OCR was successfully implemented using CNN algorithm and operated on python with the help of openCV, numpy and Keras libraries.

CNN provided an accuracy of 0.8718 which is far better when compared to that given by k-nn which is 0.562.

## 5.1 Problems faced and their solutions:

- **Problem** - Illumination was not appropriate in order to recognize the images in real-time.
  **Solution** - Special illuminating lamps were used in order to generate sufficient amount of lighting enabling the web-cam to recognize images in real time.

- **Problem** – Training of the network proved to be a difficult task because of the amount of time it was taking with the resources at hand.

## 5.2 Limitations of the OCR implemented:

- Every letter in a specific word needs to be separated to give effective results.
- The text should be darker and the background should be brighter.
- Text needs to be written on absolute white paper to compensate for the low lighting.
- The text lines in the input image should be prominently separated such that it is possible for detection by horizontal projection method.
- This OCR does not recognize special characters because we haven't trained our network for the same.

## 5.3 Future work

- Making the program user friendly by adding User Interface.
- For conflicts between similar looking characters, introducing optimisation by comparing words with English dictionary**.**
- Recognition of special characters
- Recognition using mobile phone
- Improved accuracy by training it further using GPUs.

# REFERENCES


[1] M. Abu Ghosh, M. and Y. Maghari, A. (2017). "*A Comparative Study on Handwriting Digit Recognition Using Neural Networks,*" 2017 International Conference on Promising Electronic Technologies. [online] Available at: https://ieeexplore.ieee.org/abstract/document/8109042/ [Accessed 22 Apr. 2018].

[2] J. Bai, Z. Chen, B. Feng, and B. Xu, "*Image character recognition using deep convolutional neural network learned from different languages,*" 2014 IEEE International Conference on Image Processing (ICIP), 2014.

[3] Bouchain, D. (2007). "*Character Recognition Using Convolutional Neural Networks.*" [ebook] Available at: https://pdfs.semanticscholar.org/7221/51798c178c71108ec6e64d25b3829e69892b.pdf [Accessed 22 Apr. 2018].

[4] NIST. (2018). The EMNIST Dataset. [online] Available at: https://www.nist.gov/itl/iad/image-group/emnist-dataset [Accessed 22 Apr. 2018].

[5] Binti Abdul Hamid, N. and Nur Binti Amir Sjarif, N. (n.d.). "*Handwritten Recognition Using SVM, k-nn and Neural Network*." [online] Available at: https://arxiv.org/ftp/arxiv/papers/1702/1702.00723.pdf [Accessed 22 Apr. 2018].

[6] Zhang, X., Bengio, Y. and Liu, C. (2017). "*Online and offline handwritten Chinese character recognition: A comprehensive study and new benchmark*." Pattern Recognition, 61, pp.348-360.

[7] Satyanarayana, P., Sujitha, K., Sai Anitha Kiron, V., Ajitha Reddy, P. and Ganesh, M. (2017). "*Assistance Vision for Blind People Using k-nn Algorithm and Raspberry Pi.*" Proceedings of 2nd International Conference on Micro-Electronics, Electromagnetics and Telecommunications, pp.113-122.

[8] En.wikipedia.org. (2018). Convolutional neural network. [online] Available at: https://en.wikipedia.org/wiki/Convolutional_neural_network [Accessed 22 Apr. 2018].

[9] Gupta, D. (2018). Architecture of Convolutional Neural Networks (CNNs) demystified. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/ [Accessed 22 Apr. 2018].

[10] How OCR Works | A Close Look at Optical Character Recognition. (n.d.). Let's Take Things Step by Step, Shall We? | How OCR Works. [online] Available at: http://www.how-ocr-works.com/OCR/OCR.html [Accessed 22 Apr. 2018].

[11] "*Optical character recognition*", En.wikipedia.org, 2018. [Online]. Available:

https://en.wikipedia.org/wiki/Optical_character_recognition#Pre-processing. [Accessed 22 Apr. 2018]

[12] Dutt, A. and Dutt, A. (2017). "*Handwritten Digit Recognition Using Deep Learning.*" International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), [online] 6(7). Available at: http://ijarcet.org/wp-content/uploads/IJARCET-VOL-6-ISSUE-7-990-997.pdf [Accessed 22 Apr. 2018].

[13] Medium. (2018). Learning Less to Learn Better—Dropout in (Deep) Machine learning. [online] Available at: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 [Accessed 23 Apr. 2018].

[14] En.m.wikipedia.org. (n.d.). Artificial neural network. [online] Available at: https://en.m.wikipedia.org/wiki/Artificial_neural_network [Accessed 23 Apr. 2018].

[15] Maladkar, K. (n.d.). 6 Types of Artificial Neural Networks Currently Being Used in ML. [online] Analytics India Magazine. Available at:https://analyticsindiamag.com/6-types-of-artificial-neural-networks-currently-being-used-in-todays-technology/ [Accessed 23 Apr. 2018].

[16] TechCrunch. (n.d.). Neural networks made easy. [online] Available at: https://techcrunch.com/2017/04/13/neural-networks-made-easy/ [Accessed 23 Apr. 2018].