Optical Character Recognition

Project Report Submitted by :

Aagneyan Namboodiri (CUTM00527)

Akhil P V (CUTM00706)

Rithin Varghese (CUTM00867)

Suraj Kumar M (CUTM00529)

Under the Guidance of

Mr. Ashish Bansal (Mentor)

In fulfillment of the requirements

For the award of the certificate in

ARTIFICIAL INTELLIGENCE/MACHINE LEARNING

Table of Contents

Introduction

As in today's world though all the information is represented electronically, but the information represented on the paper has its own relevance. However, this information is not available for the visually impaired people. To help them in getting this vital information we propose a system in which the document is read electronically and will be converted into speech. This system will help the visually impaired to be aware of the information presented on the document through the help of speech. Pre-processing of the captured image or text will be done, all these features will be provided by Optical Character Recognition (OCR) module. Once the image is being preprocessed Optical Character Recognition will read and recognize the text visible in the image. However the final outcome of the system will be an audible speech which will read out the text recognized by the OCR module, For delivering this voice output Text-To-Speech (TTS) module is being used. This application will read anything that is present in English language and also any numerical integer value. Every special character like comma, exclamation, full stop, question mark will be taken into consideration and right pause will be taken whenever any of these is encountered. Thus using OCR the system will recognize and convert the image into text format and further Text-To-Speech will convert the recognized text into voice which will help the visually impaired to read the document and keep them updated.

1. Background

1.1 Aim

- To build an AI based OCR system that converts images of typed, handwritten, or printed text into machine encoded text and produces an output.

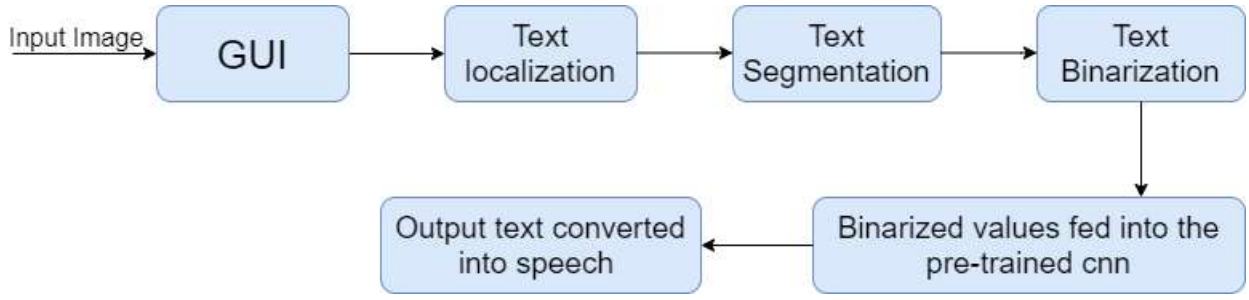
- To convert the given output into a speech format.

- The purpose of delivering the output in form of voice/speech is to serve the information that is present on the document to the visually impaired.

1.2 Technologies

Python (3.7.3)
Convolutional Neural Networks(CNN)
Keras
Tensorflow
tKinter

1.4 Software Architecture



2. System

2.1 Requirements

2.1.1 Functional requirements

The intended function of the software is to take an input image and identify the text in it. Later the text is converted into audio which can be played by the user.

2.1.2 Software Requirements

1. Windows 7 or higher
2. Operating Environments- Pycharm, Jupyter, Google Colab
3. Programming Language-Python

2.1.3 Hardware Requirements

1. A scanner for taking in the image
2. audio output interface
3. i3 Processor Based Computer or higher
4. Memory: 4 GB RAM
5. Hard Drive: 50 GB
6. Monitor
7. Mouse

2.2 Design and Architecture

The Software and Hardware architecture of the proposed model is as follows:

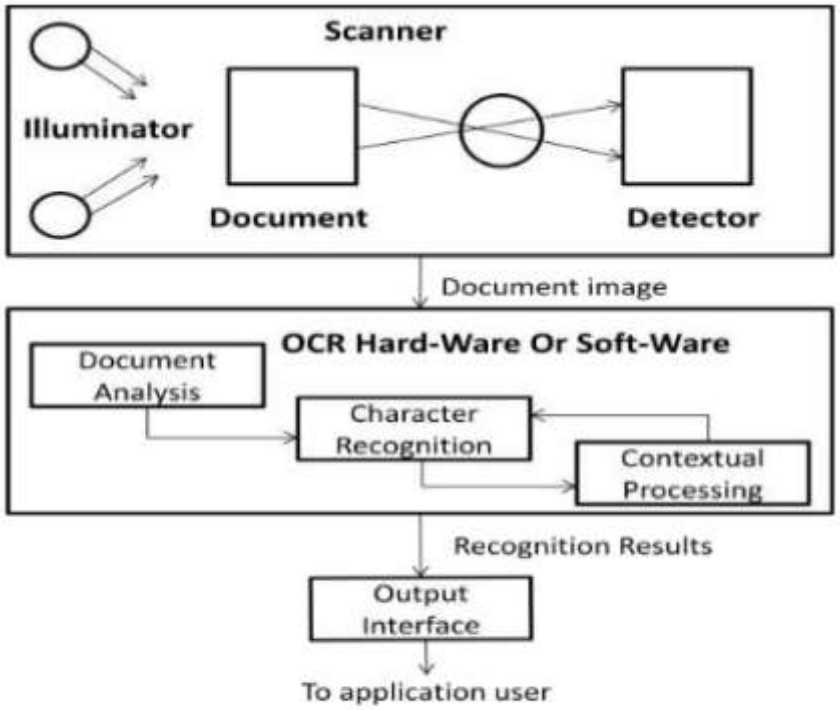


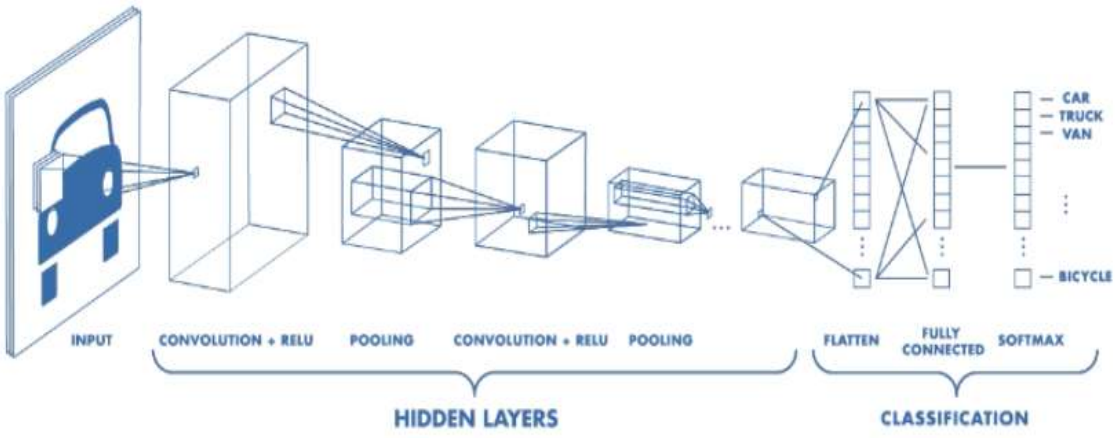
Figure.1: OCR Architecture

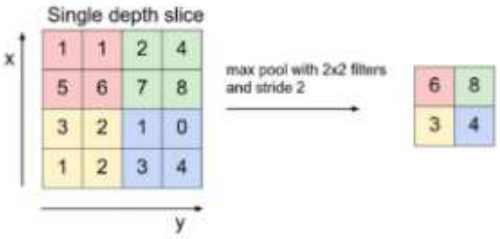
The whole design of the software revolves mainly around convolutional neural networks

The architecture of a CNN includes:

- Convolution
- Pooling
- Classification

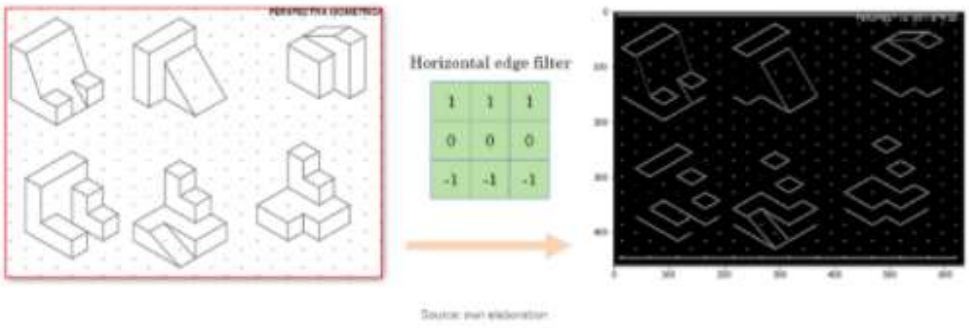
Convolution and Pooling:





Convolution

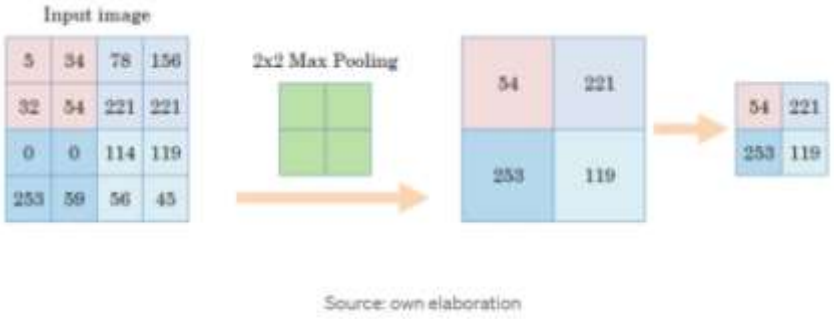
- A filter is applied to the image to get a resulting image.
- The image can be represented as a matrix of pixels, with each pixel representing a color intensity.
- We build another image by applying the filter to our input image. Note that depending on the filter we apply (its shape and values), we will get a different image.



Pooling

- Take groups of pixels (for example, groups of 2x2 pixels) and perform an aggregation over them. One of the possible aggregations that can be chosen for the process is known as Max Pooling.

- By doing so, the image size is reduced to its half: by retaining the maximum only the maximum of the 2x2 pixels, now the image is half bigger.



Source: own elaboration

After this stage, the image is flattened out into a 1D array and passed through various layers of the cnn. The error obtained at the output layer is rectified using back propagation.

2.3 Implementation

The image is converted into a numpy array and passed through the CNN.

The exact structure of the neural network used to train the model is as follows where each *conv_2d* layer represents a convolution layer and *max_pooling2d* represents a pooling layer

```
In [13]: act_model.summary()
```

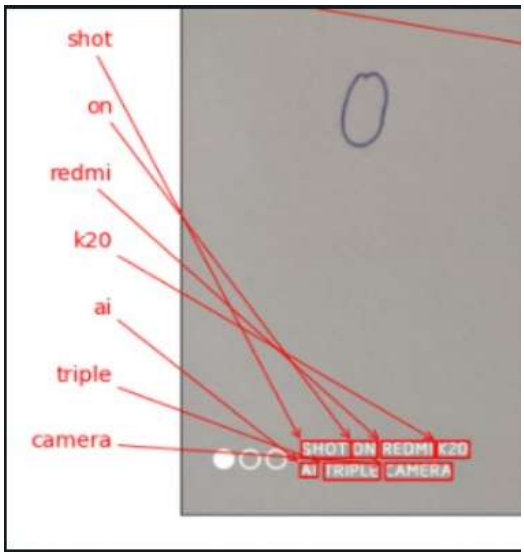
```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 32, 128, 1)]      0
_________________________________________________________________
conv2d (Conv2D)              (None, 32, 128, 64)       640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 64, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 64, 128)       73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 32, 128)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 32, 256)        295168
_________________________________________________________________
dropout (Dropout)            (None, 8, 32, 256)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 32, 256)        590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 32, 256)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 32, 512)        1180160
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 32, 512)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 4, 32, 512)        2048
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 32, 512)        2359808
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 32, 512)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 4, 32, 512)        2048
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 32, 512)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 1, 31, 512)        1049088
_________________________________________________________________
lambda (Lambda)              (None, 31, 512)           0
_________________________________________________________________
bidirectional (Bidirectional (None, 31, 512)           1574912
_________________________________________________________________
bidirectional_1 (Bidirection (None, 31, 512)           1574912
_________________________________________________________________
dense (Dense)                (None, 31, 79)            40527
=================================================================
Total params: 8,743,247
Trainable params: 8,741,199
Non-trainable params: 2,048
_________________________________________________________________
```

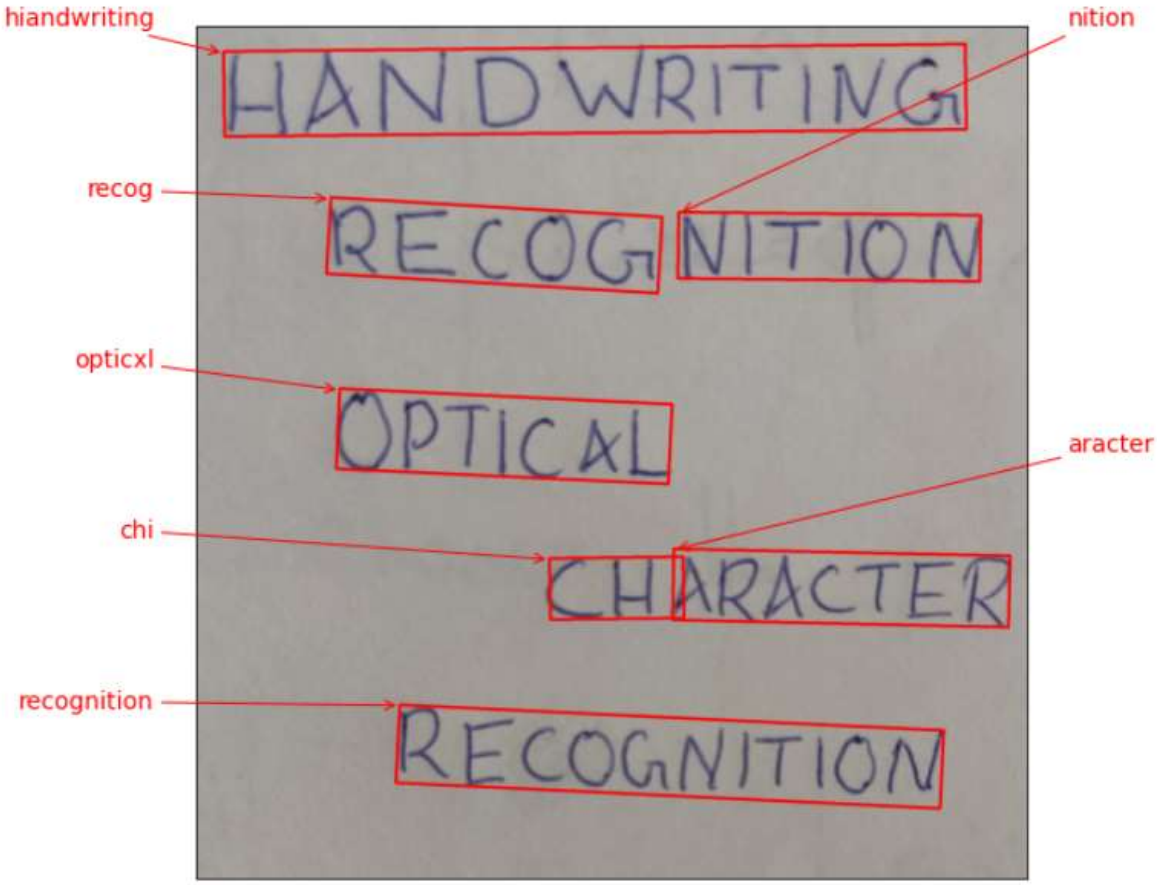
Batch normalisation *(batch_normalisation)* is performed inorder to normalise the data that is passed through the neuron. The normalisation ensures that the inputs have a mean of 0 and a standard deviation of 1, meaning that the input distribution to every neuron will be the same. This helps to achieve a better accuracy score and helps the model to train better.

LSTM is used to store the data over extended periods of time and help to backtrack the error and reduce the amount of noise generated during the process.

Bidirectional LSTMs *(bidirectional)* train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.

Samples


shot
on
redmi
k20
ai
triple
camera
SHOT ON REDMI K20
AI TRIPLE CAMERA


hiandwriting
nition
recog
opticxl
aracter
chi
recognition
HANDWRITING
RECOGNITION
OPTICAL
CHARACTER
RECOGNITION

2.4 Testing

Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement

2.4.1 Test Plan Objectives

To ensure that our system is error free we have to execute the program with the aim of finding errors.

1. The system should meet the user's requirements

2. Ideally the testing should be done by a independent assessor.

2.4.2 Data Entry

The dataset is taken from the following link:

https://fki.tic.heia-fr.ch/databases/download-the-iam-handwriting-database

The dataset consists of 1,15,320 images containing isolated and labelled handwritten uppercase and lowercase letters contributed by 657 writers all having a different handwriting style.

The words have been extracted from pages of scanned text using an automatic segmentation scheme and were verified manually.

All word images are provided as PNG files and the corresponding label files, including segmentation information and variety of estimated parameters are included in the image files as meta-information in XML format which is described in the file word.txt

2.4.4 Test Strategy

1. Optimal instead of exhaustive testing is what is suggested.

2. Usually errors follow the Pareto (80-20) rule of 80% error coming from 20% of programme components

3. It is imperative that we cover the basic tests before going to complex ones.

2.4.5 System Test

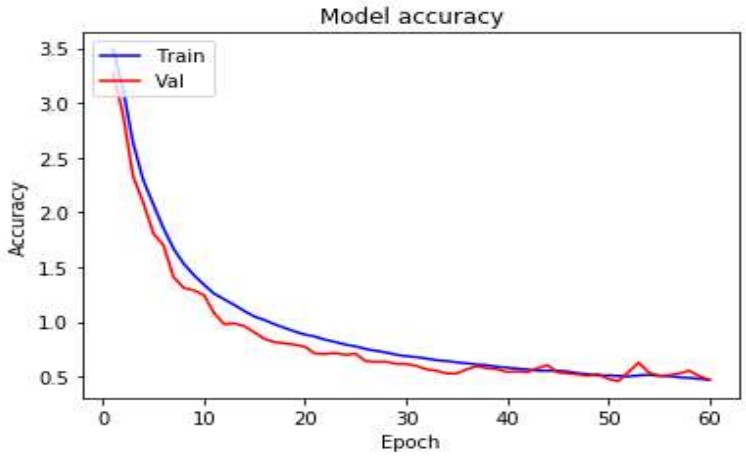
System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results.

Our system works on windows 7 and respective higher versions.

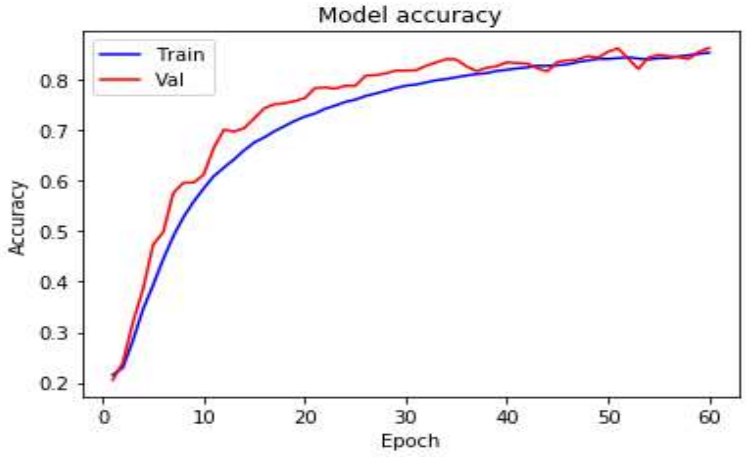
2.4.12 Performance Test

This is graph shows the accuracy and validation accuracy of the model per epoch for the 60 epochs for which the model has been trained.

```
In [63]: plotgraph(epochs, loss, val_loss)
```




```
In [64]: plotgraph(epochs, acc, val_acc)
```

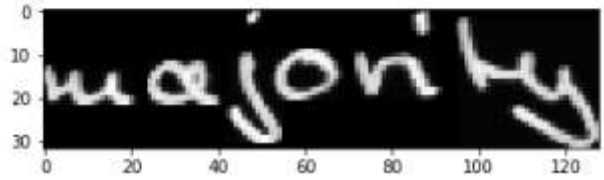


The accuracy and validation accuracy stay close to each other throughout the entire training process with only marginal difference and validation accuracy getting a slightly better score at the end showing that the model is not overfit.

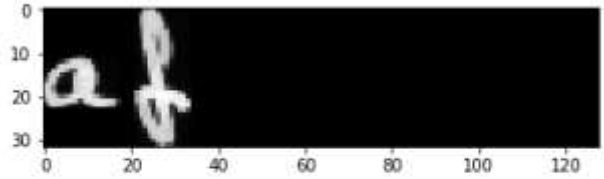
Here are a few samples from the training data showing the accuracy of the model.

```
original_text =   large
predicted text = large
```

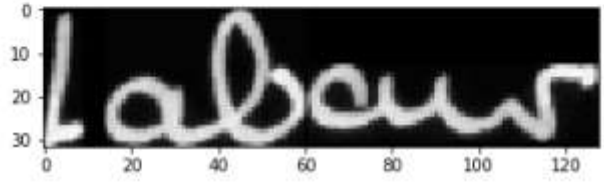
```
original_text =   majority
predicted text = majority
```



```
original_text =   of
predicted text = of
```



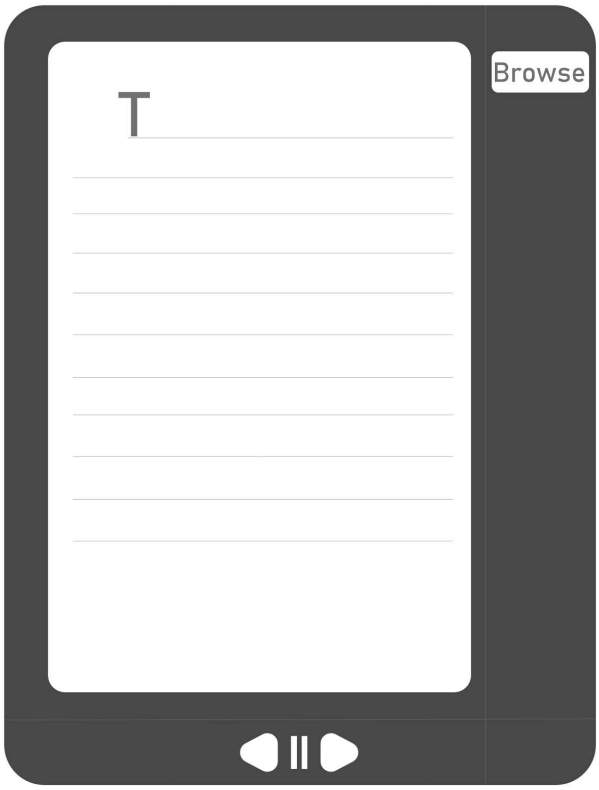
```
original_text =   Labour
predicted text = Labour
```



2.4.12 User Acceptance Test

We did a pilot test with a few users to check whether our system would be acceptable to users. It was found to be very much in tune with the user requirements and the results were nothing short of acceptable.

2.5 Graphical User Interface (GUI) Layout



The GUI consists of a browse option from where the required image can be loaded. The processed output can then be listened to using the built in feature. The converted text is automatically saved into a document locally for future reference.

2.7 Evaluation

We have checked the main functions. The static code was analysed and the entire code evaluated.

2.7.1 Static Code Analysis

For doing the static code analysis,we used

1]Radon package for code complexity

2]Pylint package for checking code duplication

3]Pytest

4]Flake-8

Errors other than syntax errors were cleaned up.

2.7.2 Test of Main Function

With the test of main function we validated the software system against the functional requirements that we had specified. We tested each function of the application, by providing appropriate input and verifying the output which worked just fine.

3. Conclusion

In this project, we were able to implement an Optical Character Recognition software using Convolutional Neural Networks and then use it to output the text in an audio format to help the blind read text from handwritten documents.
The projected accuracy of this model was found out to be about 87% which is relatively good, and any minor errors that may occur during the output can be easily corrected by the person that uses it.

This application scans the text and then converts it into digital text which is recognized by the system and displays the translated text and gives speech output. To understand the dynamics of the project, a basic idea about what is AI and OCR is required. This report explains the entire working of Language Translator, along with minimum requirements needed to implement it. Hence, visually impaired people can easily use this AI based Reading system as a friendly simple application all around the globe.

4. Further development or research

Currently the model detects words individually. In order for the model to produce an output in the audio format, it is necessary for the model to produce words from the handwritten text. A high accuracy for this is questionable considering the spacing of words in handwritten documents and the hardware requirement in order to process the whole dataset in a relatively shorter period of time. The GUI however can be improved and can be made more appealing towards the user that uses it. Hence there is a room for further development considering this aspect of the OCR.

5. References

1. Jisha Gopinath, Aravind S, Pooja Chandran, Saranya S S, "Text to Speech Conversion System using OCR", International Journal of Emerging Technology and Advanced Engineering , Volume 5, Issue 1, January 2015

2. Aaron James S, Sanjana S, Monisha M, "OCR based automatic book reader for the visually impaired using Raspberry PI", Vol. 4, Issue 7, January 2016.

3. Sonia Bhaskar, Nicholas Lavassar, Scott Green, "Implementing Optical Character Recognition on the Android Operating System for Business Cards"

4. Julinda Gllavata, Ralph Ewerth and Bernd Freisleben, "A Robust Algorithm for Text Detection in Images".

5. Chucai Yi, Yingli Tian, “Scene Text Recognition in Mobile Applications by Character Descriptor and Structure Configuration”, IEEE Transactions on Image Processing, Vol. 23 No. 7, July 2014.

6. Christopher G Relf, "Image Acquisition and Processing with LabVIEW", CRC Press, 2004.

7. Ashwani Kumar, Ankush Chourasia, "Blind Navigation System Using Artificial Intelligence", March 2018

8. https://nanonets.com/blog/ocr-with-tesseract/

9. http://en.wikipedia.org/wiki/Optical_character_recognition