Text-To-Speech Device for Visually Impaired People

A project report submitted in partial fulfillment of the requirements of the degree of

Bachelor of Engineering

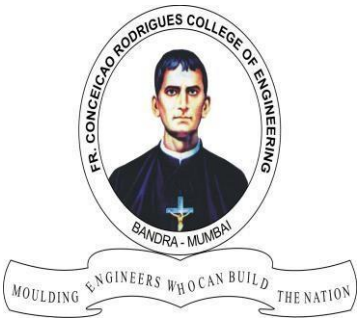
by

Faizan Jetpurwala (8273)
Affan Khan (8280)

Under the guidance of

Prof. Heena Pendhari

DEPARTMENT OF ELECTRONICS ENGINEERING



Fr. Conceicao Rodrigues College of Engineering Bandra (W), Mumbai - 400050
University of Mumbai

November 24, 2020

CERTIFICATE

This is to certify that the project entitled **"Text-To-Speech Device for Visually Impaired People"** is a bonafide work of **Faizan Jetpurwala (8273) Affan Khan (8280)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering (Electronics)**.

(Prof. Heena Pendhari)
Supervisor/Guide

`

(Dr. Sapna Prabhu)
Head of Department

(Dr. (Mrs.) Srija Unnikrishnan)
Principal

PROJECT REPORT APPROVAL

This project report entitled "**Text-To-Speech Device for Visually Impaired People"** by ***Faizan Jetpurwala (8273) Affan Khan (8280)*** is approved for the degree of Bachelor of Engineering.

Examiners

1._____

2._____

Date:

Place:

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Faizan Jetpurwala (8273):_____

Affan Khan (8280):_____

Date:

ABSTRACT

People who suffer from low vision, sight and visual impairment are not able to see words and letters in ordinary newsprint, books and magazines clearly. This can make the reading process difficult which can disturb the learning process and impair the person's intelligence development. Therefore, a device is needed to help them read.

So we want to develop one such device that can scan and read any kind of text by changing it to voice message. The purpose of this device is to process the input Image, pdf, Documents, Textbooks, and News Papers as input into a voice as output.

The device we have proposed aims to help people with visual impairment. In this project, we developed a device that converts an image's text to speech. The basic framework is a system that captures an image, extracts only the region of interest (i.e. region of the image that contains text) and converts that text to speech.

The captured image undergoes a series of image pre-processing steps to locate only that part of the image that contains the text and removes the background. Two tools are used convert the new image to speech. The audio output is heard through the speakers or earphones.

ACKNOWLEDGEMENTS

We have great pleasure in presenting the report on **"Text-To-Speech Device for Visually Impaired "**. We take this opportunity to express our sincere thanks towards the guide Prof. Heena Pendhari, C.R.C.E, Bandra (W), Mumbai, for providing the technical guidelines, and the suggestions regarding the line of this work. We enjoyed discussing our projects progress with sir.

We thank Dr. Sapna Prabhu, Head of Department (Electronics Engineering), our principal and the management staff of C.R.C.E., Mumbai for encouragement and providing necessary infrastructure for pursuing the project.

We also thank all the non-teaching staff for their valuable support, help us complete our project.

Faizan Jetpurwala (8273)

Affan Khan (8280)

Date:

CONTENTS

LIST OF FIGURES AND TABLES

Chapter 1

Introduction

Reading is very important and one aspect in our day today lives. Almost 314 million visually impaired people are there all around the universe , 45 Million are blind and new cases being added each year as per researches done. Emerging technologies and recent developments in computerized vision, digit cameras, PDA and portable computers make it flexible to assist these individuals by developing image-based products that combine computer vision technology with other existing commercial technology such as optical character recognition (OCR) platforms.

Printed text is one of the forms of reports, receipts, bank statements, restaurant menus, classroom handouts, product packages, medicine bottles, banners on road etc. There are many favoring systems available currently but they have little issues in reducing the flexibility for the visually impaired persons.

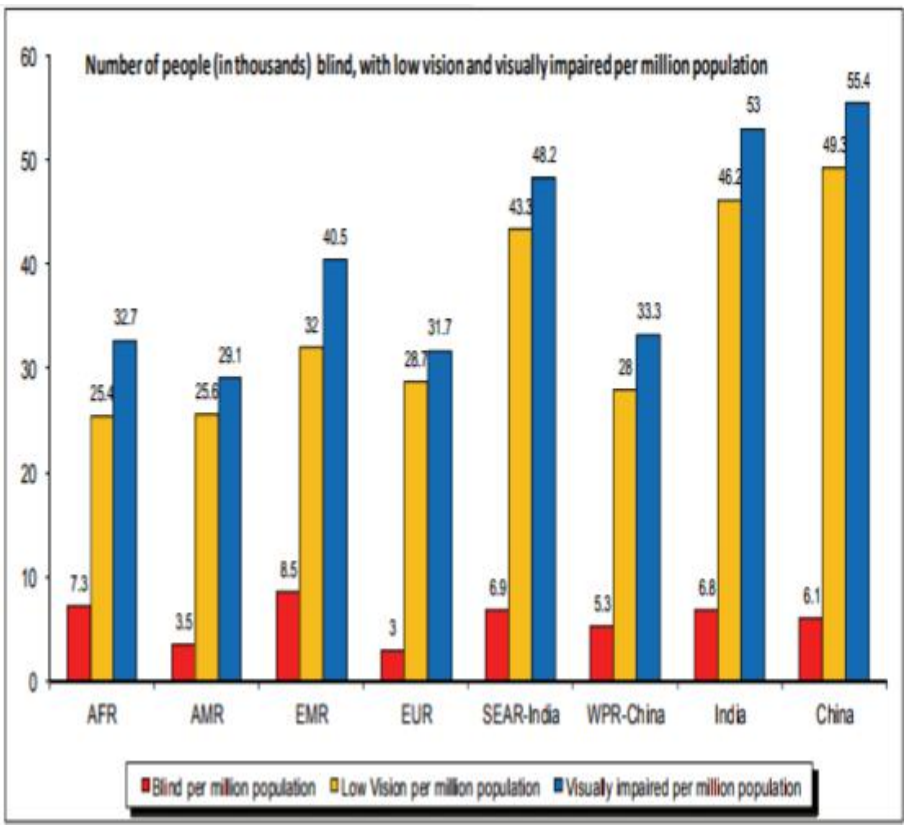
For example, portable bar code readers designed to favor the blind people recognize dissimilar products, it permits the users who are unsighted to ingress. Information about these products through speech and Braille. But a big limitation is that it is very tough for unsighted people to find the location of the bar code and to exact point the bar code reader at the bar code.

There are systems like K Reader Mobile it runs on a cell phone and permit the user to check mail, receipts, fliers, and many other things. Although, the document to be peruse must be nearly kept undisturbed, placed on a coherent, dark surface (i.e., a no confusion background), and contain mostly passage Furthermore, K Reader Mobile accurately sees black print on a white background but has issues in recognizing colored text or text on a colored background. It cannot read text with complex backgrounds. The main goal is to progress such a system that will peruse the texts from composite backgrounds successfully.

Based on the survey by World Health organization in 2010, total population in India is 1181.4 million out of which people who suffer with blindness, low vision and visual impairment are 152.238 Million. Figure shows the number of people who are blind, with low vision and visually impaired (in thousands) per million population.

According to Dr.Bjorn, impaired vision can have negative effects on learning and social interaction. It can affect the natural development of intelligence and academic ability, social, and profession. People who are visually impaired cannot be recovered with the help of glasses. This causes people with low vision, they cannot even see the normal printed paper.

They can only see if the sizes of the characters or letters are big enough. This condition impacted the length of the reading process and made the eyes tired. To help improve the quality of life for people with low vision a tool to read the article is needed. The rate of vision impairment can vary in each individual with low vision. Therefore a device developed in this work utilized other sensory function in receiving information from a text. The device is specifically designed for the people with low vision. So, that they can easily use this device without having to ask for help from others and they can utilize this device for academic and intelligence ability.



(Fig 1.1: A Survey on the no of visually impaired people)

Chapter 2

Literature Review

- Wei Ping, Kainan Peng, Andrew Gibiansky, Ajay Kannan, and Sharan Narang- DEEP VOICE 3-SCALING TEXT-TO-SPEECH WITH CONVOLUTIONAL SEQUENCE LEARNING [1]: Majorly based on neural speech synthesis and attention based sequence to-sequence learning. Recent research provide an improvement in Deep Voice 1, Deep Voice 2, Tacotron, and VoiceLoop. The work in Deep Voice 1 and Deep Voice 2 were to retain the traditional structure of TTS pipelines, duration and frequency prediction, and waveform synthesis. Compared to Deep Voice 1 and Deep Voice 2, Deep Voice 3 provides a more compact and robust architecture which helps in accelerating the synthesis process drastically. Deep Voice 3 is similar to Tacotron. Deep Voice 3 avoids Recurrent Neural Networks (RNNs) to speed up training. Automatic speech recognition (ASR) datasets are often much larger than traditional TTS corpora but tend to be less clean, as they typically involve multiple microphones and background noise. Prior work has applied TTS methods to ASR datasets, Deep Voice 3 is, to the best of our knowledge, the first TTS system to scale to thousands of speakers with a single model. Sequence-to-sequence models encode a variable-length, which are then processed by a decoder to produce a target sequence. Recent improvements in attention mechanisms relevant to Deep Voice 3 include enforced-monotonic attention, fully-attentional non-recurrent architectures and convolutional sequence to-sequence models. Deep Voice 3 demonstrates the utility of monotonic attention during training in TTS, a new domain where monotonicity is expected. From our research we can conclude that Deep Voice 3 model is highly meticulous, we can use several aspects of it in our Project.

- T.Rubesh Kumar, C.Purnima "Assistive System for Product Label Detection with Voice Output For Blind Users" International Journal of Research in Engineering & Advanced Technology 2014 [2]: T. Rubesh Kumar etal proposedreading is obviously essential in today's society. Printed text is everywhere in the form of reports, receipts, bank statements. There are already a few systems that have some promise for portable use, but they cannot handle product labeling. But a big limitation is that it is very hard for blind users to find the position of the bar code and to correctly point the bar code reader at the bar code. T.Rubesh Kumar, C.Purnima have proposed a camera-based assistive text reading framework to help blind persons read text labels and product packaging from hand-held objects in their daily lives. Main contributions embodied in this prototype system are: 1) A novel motion-based algorithm to solve the aiming problem for blind users by their simply shaking the object of interest for a brief period; 2) A novel algorithm of automatic text localization to extract text regions from complex background and multiple text patterns; and 3) A portable camera-based assistive framework to aid blind persons reading text from hand-held objects.

- Mallapa D.Gurav, Shruti S. Salimath, Shruti B. Hatti, Vijayalaxmi I. Byakod , “B-LIGHT [3]: A Reading aid for the Blind People using OCR and OpenCV” International Journal of Scientific Research Engineering & Technology (IJSRET 2017 Mallapa D.Guravetal proposed thatthis project presents a smart device that assists the visually impaired which effectively and efficiently reads paper-printed text. The proposed project uses the methodology of a camera based assistive device that can be used by people to read Text document. The framework is on implementing image capturing technique in an embedded system based on Raspberry Pi board. The proposed fully integrated system has a camera as an input device to feed the printed text document for digitization and the scanned document is processed by a software module the OCR (optical character recognition engine).Optical character recognition (OCR) is the identification of printed Characters using photoelectric devices and computer software. It coverts images of typed or printed text into machine encoded text from scanned document or from subtitle text superimposed on an image. In this research these images are converted into audio output. OCR is used in machine process such as cognitive computing, machine translation, text to speech, key data and text mining. The recognition process is done using OCR the character code in text files are processed using Raspberry Pi device on which it recognizes character using tesseract algorithm and python programming and audio output is listened.

- A. SUBBIAH, T. ARIVUKKARASU, M. S. SARAVANAN, V. balaji “Camera based label reader for blind people” Int..Chem. Sci.: 14(S3), 2016, 840-844 ISSN 0972-768X [4]: Here we are using AdaBoost Algorithm for treating the visual information and converting into audio speech. The proposed system helps visually impaired, illiterate, or have a learning disability to read product the project aims to implement a reading aid that is small, lightweight, efficient, cost effective and of course user-pleasant the Raspberry Pi- based system can be equipped with a high- resolution webcam the microcontroller-built system is easier to use when compared to the mobile one. However, the accuracy of the mobile in the conversion efforts is better, mainly due to the high-resolution camera built in the device. Developing technology and in future expansions of this project, the R-Pi based system can be provided with a good and high- resolution camera contrasted with the one used in this project, and we anticipate, this will improve its inevitability. We predict more work will be produced in this critical area of assistive technology, and project that future transportable gadgets will have easy to use and built in mechanism as reading assistances for the blind, similar, to the mobile- based solution presented here. Users should capture image and then system read out the text from image. It will be more applicable for persons those are going through visual surgery. It can be suitable for road side text recognition so that visually impaired person can travel alone.

- K Nirmala Kumari, Meghana Reddy Image “Text to Speech Conversion Using OCR Technique in Raspberry Pi” International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (An ISO 3297: 2007 Certified Organization) Vol. 5, Issue 5, May 2016 [5]: A performance that is high enough and a readability tolerance of less than 2%, with the average time processing. This portable device does not require internet connection. This portable device

does not require internet connection. This paper presented their project for text detection and converts into audio format. Testing of device was done on raspberry pi platform. The R-pi is initially connected to the internet through VLAN. The software is installed using command lines. The first setup is to download the installation script, second command is to convert it to executable form and the last command starts the script which does the rest of the installation work.The paper says they proposed a device to help people with visual impairment. In this project, we developed a device that converts an image's text to audio format. This implementation required hardware. The basic framework is this implemented system that captures an image, extracts only the region of interest (i.e. region of the image that contains text) and converts that text to audio. It is developed using a Raspberry Pi and a Raspberry Pi camera.

- Nagaraja L, Nagarjun R S, Nishanth M Anand "Vision based Text Recognition using Raspberry Pi") National Conference on Power Systems & amp; Industrial Automation (NCPSIA) 2015 [6]: Nagaraja L proposed that the method is a camera based assistive text reading to help blind person in reading the text present on the text labels, printed notes and products. The proposed project involves Text Extraction from image and converting the Text to Speech converter, a process which makes blind persons to read the text. This is carried out by using Raspberry pi, where portability is the main aim which is achieved by providing a battery backup and can be implemented as a future technology. The portability allows the user to carry the device anywhere and can use any time. . To extract the text from image we use optical character recognition technique (OCR). A Text-To-Speech (TTS) synthesizer is a computer-based system that should be able to read any text aloud, whether it was directly introduced in the computer by an operator or scanned and submitted to an Optical Character Recognition (OCR) system.

Chapter 3

Project Description

3.1 Image to text (Part 1):

Digital image processing is the use of computer algorithms to perform image processing on digital images. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing.

It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems.

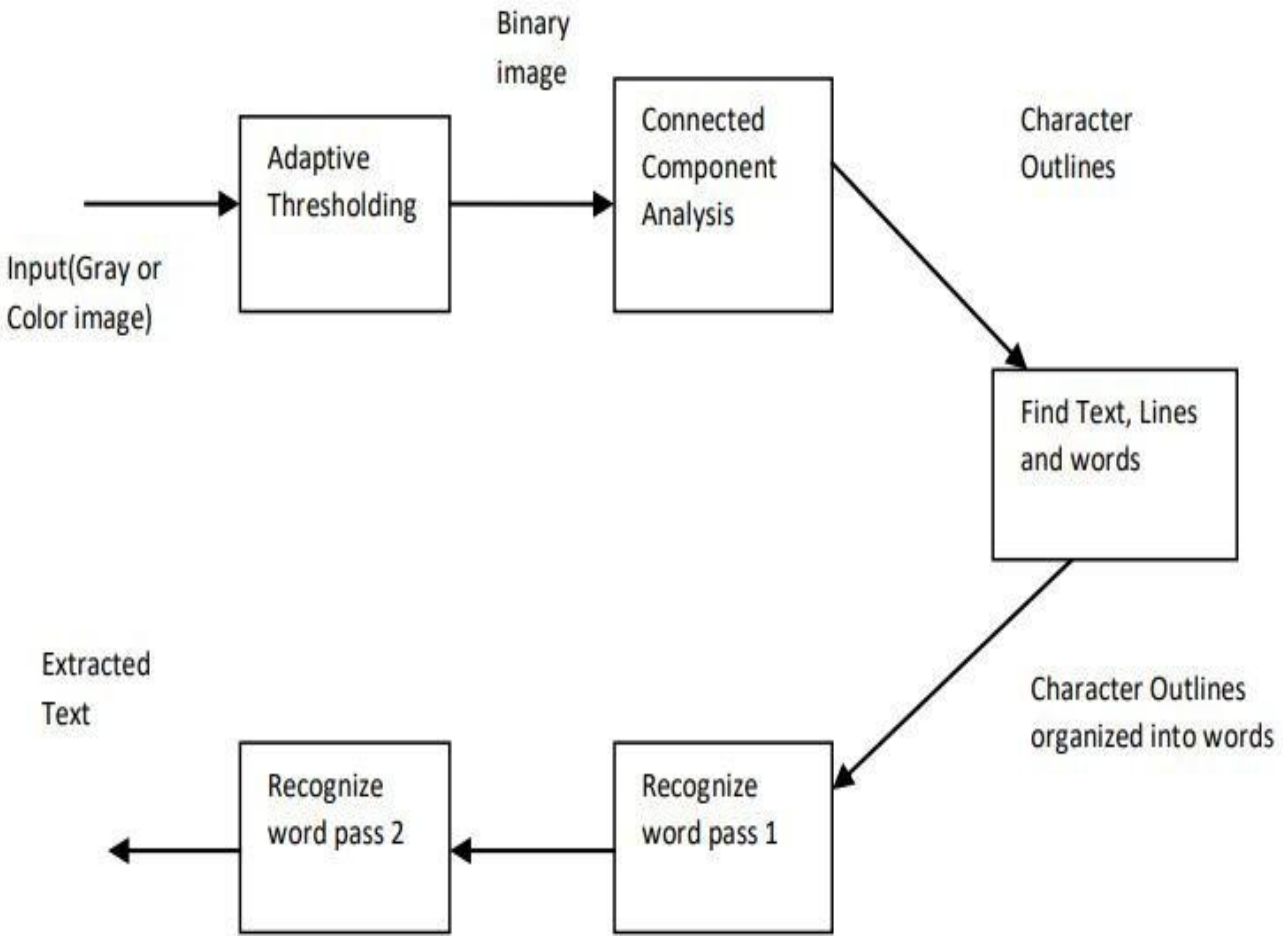
It is among rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too. Image processing is done on digital images by using computer algorithms.

So in the first part of our project we are performing image processing using OCR with help of pytessaract. We have used the following algorithm as shown in fig 3.1.1.

Optical Character Recognition (OCR) is the conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a photo from a scene (billboards in a landscape photo) or from a text superimposed on an image (subtitles on a television broadcast).

By passing the image through multiple stages we have obtained text from the image which will be used further in the project to convert text to speech. Detailed working is explained in the implementation part.

3.1.1 Block Diagram – Part 1:



(Fig 3.1.1 Block Diagram of the System)

3.2 Text to speech (Part 2):

Text-to-speech (TTS) systems convert written language into human speech. TTS systems are used in a variety of applications, such as human-technology interfaces, accessibility for the visuallyimpaired, media and entertainment. Traditional TTS systems are based on complex multi-stage hand-engineered pipelines.

Typically, these systems first transform text into a compact audio representation, and then convert this representation into audio using an audio waveform synthesis method called a vocoder.

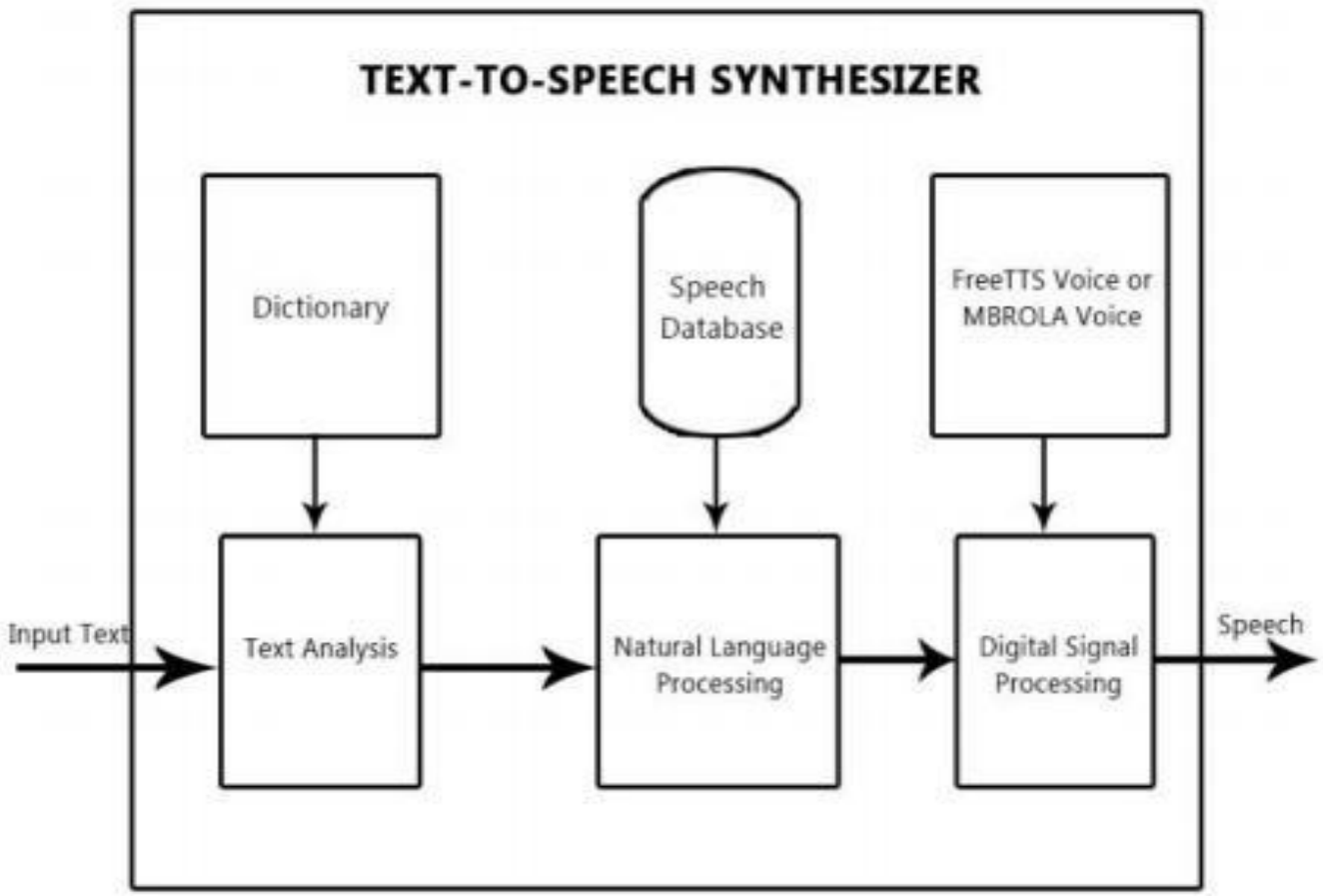
Recent work on neural TTS has demonstrated impressive results, yielding pipelines with simpler features, fewer components, and higher quality synthesized speech. There is not yet a consensus on the optimal neural network architecture for TTS. However, sequence-to-sequence models have shown promising results.

We are using Deep Voice 3, a fully-convolutional attention-based neural textto-speech (TTS) system. Deep Voice 3 matches state-of-the-art neural speech synthesis systems in naturalness while training an order of magnitude faster. We scale Deep Voice 3 to dataset sizes unprecedented for TTS, training on more than eight hundred hours of audio from over two thousand speakers. In addition, we identify common error modes of attention-based speech synthesis networks, demonstrate how to mitigate them, and compare several different waveform synthesis methods.

So in our part 2 of the project that is converting text to speech we are using deep voice 3 a model which is pre-trained using a very large dataset which helps us in converting our text to speech. To get a brief idea of the algorithm used see fig 3.2.2.

We give text obtained in part 1 as the input to the model, the model then compares the input with the dataset used to train it and then gives a suitable output speech.

3.2.2 Block Diagram – Part 2:



(Fig 3.2.2 Block Diagram of the System)

Chapter 4

Implementation

4.1. Converting Image to Text.

4.1.1. Opening of the image in Python using Pillow:

Firstly the image containing the text is opened (Which is read only function) using the Pillow library which is the friendly PIL fork by Alex Clark and Contributors .

The Python Imaging Library adds image processing capabilities to your Python interpreter.

This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

The image we used for testing was this:

(Fig 4.1.1.1 : Original Image)

4.1.2. : Converting the colorful image into greyscale using pytesseract:

After getting the image into Python, the image is converted to a greyscale image using pytesseract so that the text can be easily identified using OCR.

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images.

Python-tesseract is a wrapper for Google’s Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

The image then converted to greyscale image is saved for further processing.

The greyscale image is as follow:

(Fig 4.1.2.1: Greyscale Image)

4.1.3. : Converting the greyscale image to Black and white image:

Further the greyscale image is converted to a black and white image using pytessaract so that the text is more visible per pixel for OCR to identify the text.

Here using a suitable threshold value we compare every pixel of the image with the threshold value, and depending on the comparison every pixel in the image is given a black or white value.

This helps the pytessaract to identify the text more efficiently as it compares every pixel to a predefined value thereby giving output image containing only two values.

The output after converting into Black and white image:

(Fig 4.1.3.1 : Black and white image)

4.1.4. : Using Image.LANCOS or other filters to increase the resolution of image:

A new **PIL.Image.LANCZOS** constant was added instead of **ANTIALIAS**.

When **ANTIALIAS** was initially added, it was the only high-quality filter based on convolutions. It's name was supposed to reflect this. Starting from Pillow 2.7.0 all resize method are based on convolutions. All of them are antialias from now on. And the real name of the **ANTIALIAS** filter is Lanczos filter.

Further we are using filter to increase the resolution, we have used **Image.LANCOS** other filters can also be used.

The output after filtering :

(Fig 4.1.4.1: Filtered Image)

4.1.5. : Extracting the text from image using OCR and pytessaract:

After filtering, the image is converted to text by using OCR and pytessaract.

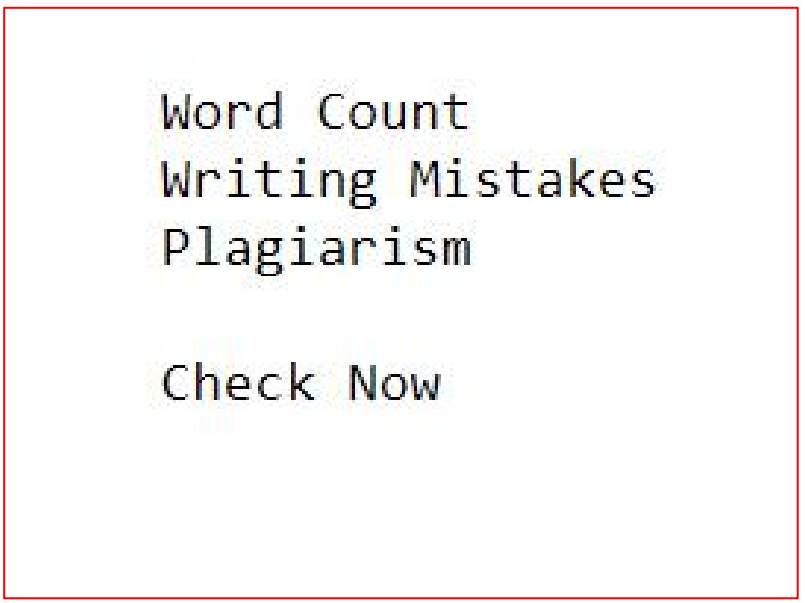
The OCR detects the outlines of the text and converts those outlines to text.

OCR = Optical Character Recognition. In other words, OCR systems transform a two-dimensional image of text, that could contain machine printed or handwritten text from its image representation into machine-readable text. OCR as a process generally consists of several sub-processes to perform as accurately as possible.

Tesseract - an open-source OCR engine that has gained popularity among OCR developers. Even though it can be painful to implement and modify sometimes, there weren't too many free and powerful OCR alternatives on the market for the longest time. Tesseract began as a Ph.D. research project in HP Labs, Bristol. It gained popularity and was developed by HP between 1984 and 1994. In 2005 HP released Tesseract as an open-source software. Since 2006 it is developed by Google**.**

'pytesseract.image_to_string' by using this function the Tessaract converts the image to Text using OCR.

The final output of the text :



(Fig 4.1.5.1: Output of image to text)

4.2. Converting Text to Speech.

- Currently we are performing text to speech conversion by using a pre- trained model named ‘Deep Voice 3’ and predicting the outcome, also checking the accuracy of the model.
- First hyper parameter from json files are extracted along with url location of weights.
- After extracting the information required by the model from json files, The model is then supplied weights by downloading the pre trained weights at the url.
- Model then converts the text given by the user into associated waveform and vocals extraced using dataset to form speech.

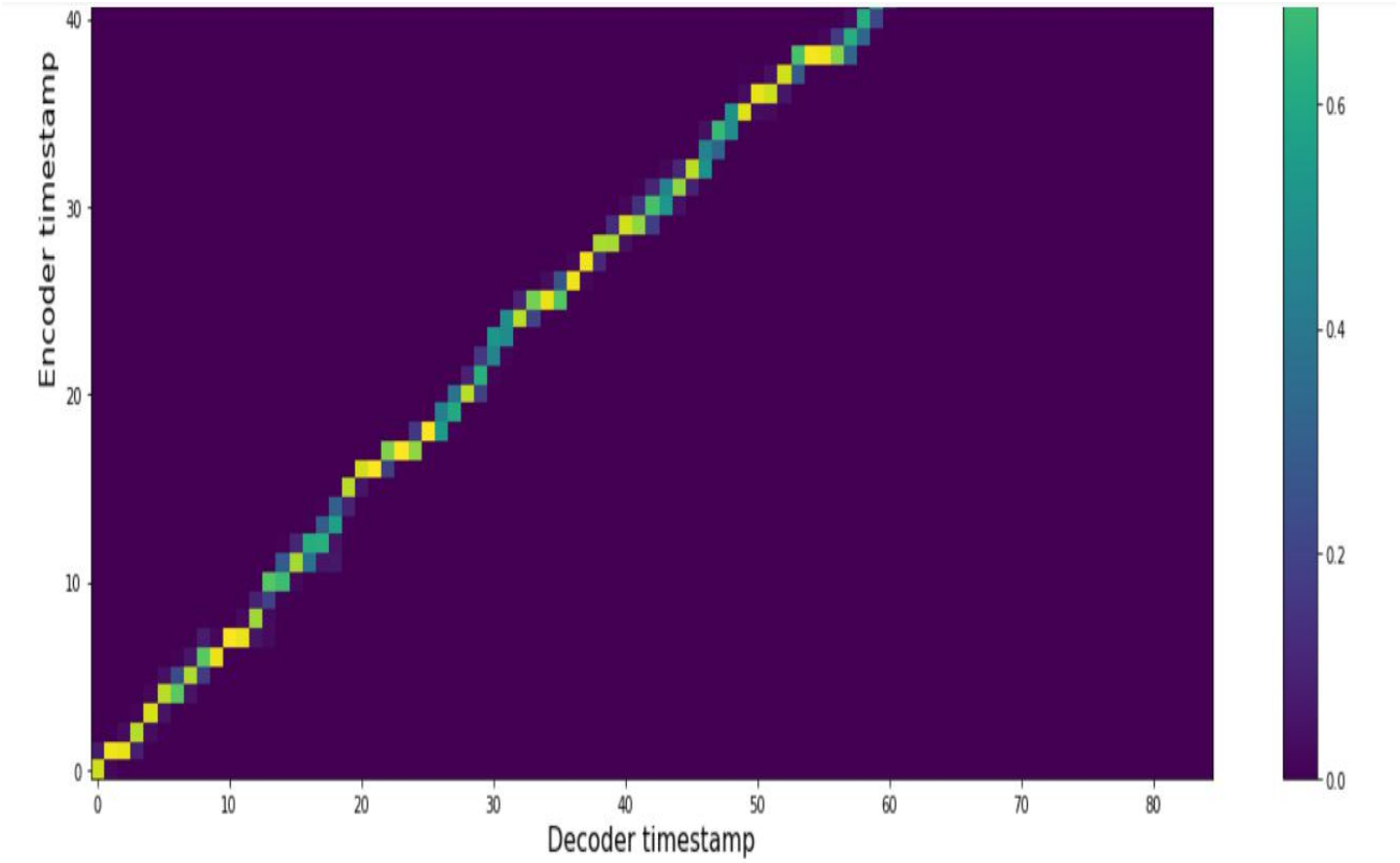
Output for text to speech:

1.wav

2.wav

3.wav

- Particular word is mapped to its decibel value creating a chain of decibel and notes corresponding of it are retrieved from the model and this is how text is converted to speech.



(Fig 4.2.1: Graph of encoder vs Decoder timestamp)

- The Deep Voice 3 Model consists of three components:

1) Encoder: A fully-convolutional encoder, which converts textual features to an internal learned representation.

- The encoder network begins with an embedding layer, which converts characters or phonemes into trainable vector representations, *he*. These embeddings *he* are first projected via a fully-connected layer from the embedding dimension to a target dimensionality.

- Then, they are processed through a series of convolution blocks to extract time-dependent text information. Lastly, they are projected back to the embedding dimension to create the attention key vectors *hk*.

- The attention value vectors are computed from attention key vectors and text embeddings, $hv = \sqrt{0.5}(hk + he)$, to jointly consider the local information in *he* and the long-term context information in *hk*.

- The key vectors *hk* are used by each attention block to compute attention weights, whereas the final context vector is computed as a weighted average over the value vectors *hv*.

2) **Decoder: A fully-convolutional causal decoder, which decodes the learned representation with a multi-hop convolutional attention mechanism into a low-dimensional audio representation (mel-scale spectrograms) in an autoregressive manner.**

- The decoder generates audio in an autoregressive manner by predicting a group of *r* future audio frames conditioned on the past audio frames. Since the decoder is autoregressive, it must use causal convolution blocks.

- It choose mel-band log-magnitude spectrogram as the compact low-dimensional audio frame representation. It is empirically observed that decoding multiple frames together (i.e. having r > 1) yields better audio quality.

- The decoder network starts with multiple fully-connected layers with rectified linear unit (ReLU) nonlinearities to preprocess input mel-spectrograms (denoted as “PreNet”). Then, it is followed by a series of causal convolution and attention blocks.

- These convolution blocks generate the queries used to attend over the encoder’s hidden states . Lastly, a fully-connected layer output the next group of *r* audio frames and also a binary “final frame” prediction (indicating whether the last frame of the utterance has been synthesized).

- Dropout is applied before each fully-connected layer prior to the attention blocks, except for the first one. An L1 loss is computed using the output mel-spectrograms and a binary cross-entropy loss is computed using the final-frame prediction.

3) Converter: A fully-convolutional post-processing network, which predicts final vocoder parameters (depending on the vocoder choice) from the decoder hidden states. Unlike the decoder, the converter is non-causal and can thus depend on future context information.

- The converter network takes as inputs the activations from the last hidden layer of the decoder, applies several non-causal convolution blocks, and then predicts parameters for downstream vocoders. Unlike the decoder, the converter is non-causal and non-autoregressive, so it can use future context from the decoder to predict its outputs.

4.3. Interfacing of both the parts.

- So previously, we had divided the project into two parts i.e Part 1 (converting image to text) and part 2 (converting text to speech), and both the parts were being implemented separately.

- As our project consist of converting a text image into audio form, we needed to interface both the parts together i.e. the text extracted from the image in the first part needed to be passed on to the second part which will use the text to convert into audio form.

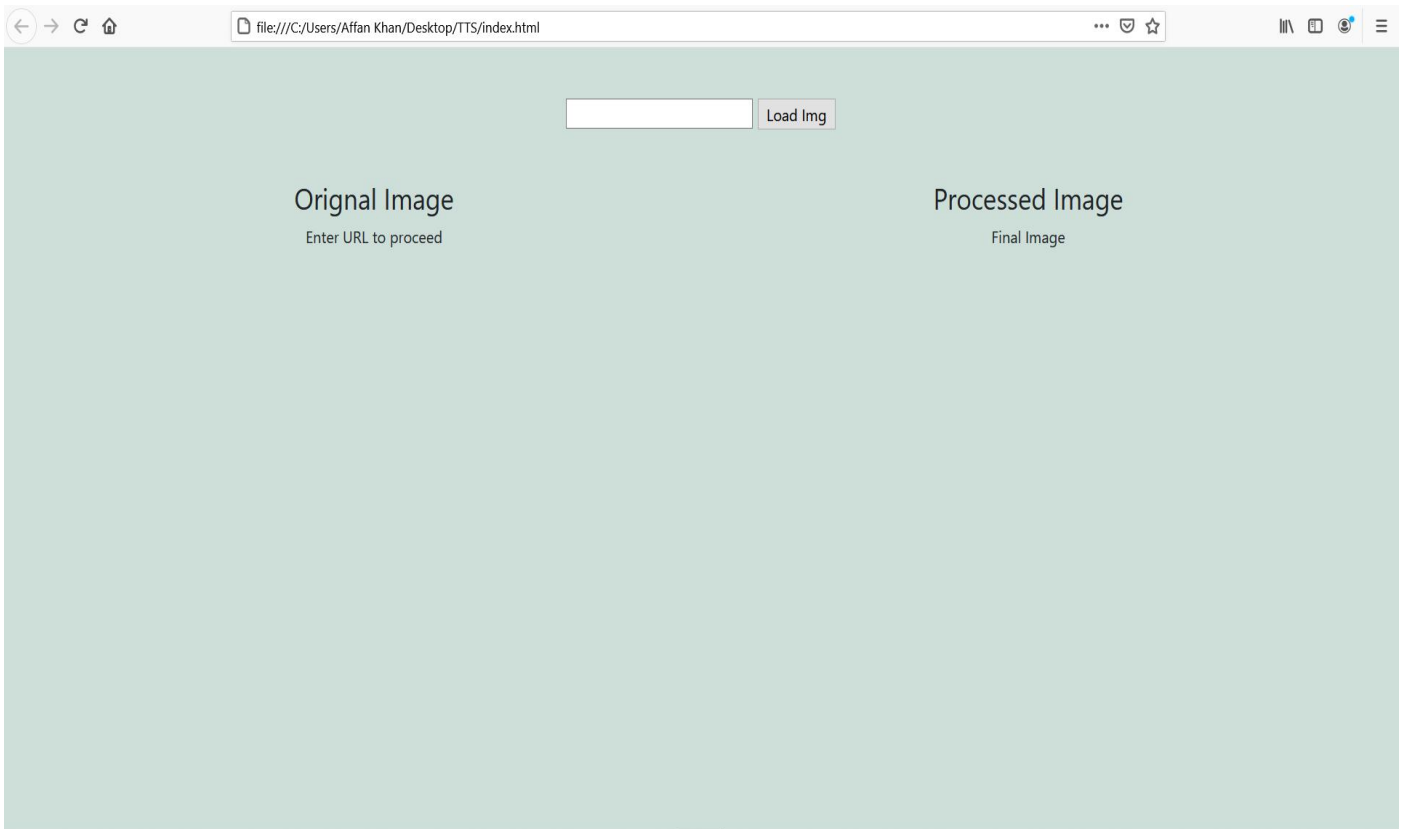
- So basically to do so first we take an image which goes through part 1 of the project i.e. conversion of an image to text, after part 1 is completed and we have successfully extracted the required text from the image, the text is then sent to the next part of the project.

- So the extracted text is received at the receiver end of part 2 i.e. conversion of text to speech, the machine learning model uses the text as input to give out the required output audio.

- So to finally complete our project we have successfully interfaced both the parts together and we are getting the desire output.

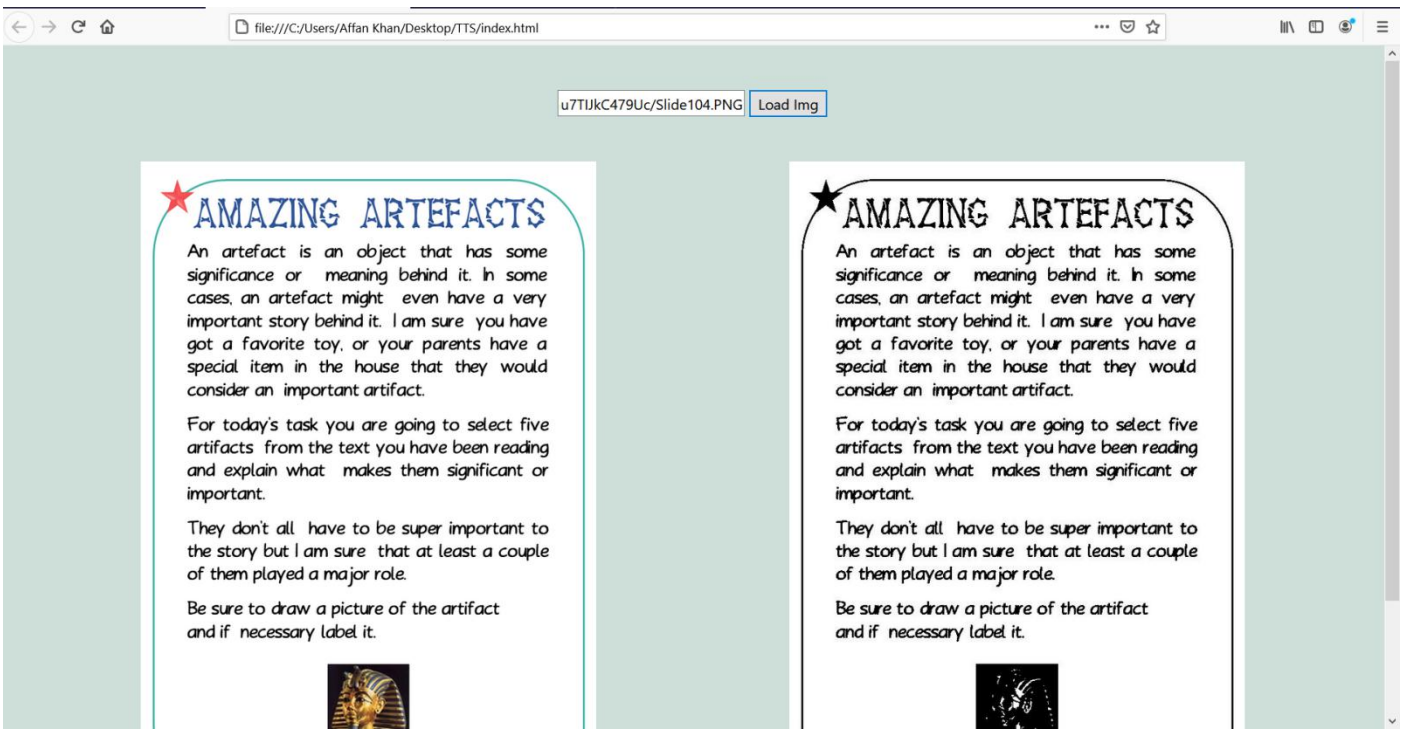
4.4. Web page and API.

- We have created a website using HTML which acts as the front-end for the project.
- While API (Application programming interface) acts as the back-end.
- The API is served using Flask. Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.
- On the webpage we have a bar in which we post any image's address, using that address the image is received from internet and forwarded to the API in back-end.



(Fig 4.4.1: Initial webpage to upload the image)

- The API then send the image to the main source code where the image is processed, after the image is converted to binarized image the API then sends binarized image as response and shows it in website.



(Fig 4.4.2: Webpage after receiving binarized image)

- At the back-end in the source code the binarized image is converted in text and that text is then converted to speech.

- After we click the load image button on the web page the whole process is done automatically without any interference and just by one click on the web page we get the audio of the image.

4.5. Limitations:

- Our project works pretty well in most of the conditions, but there are few conditions which provide hindrance to the process. Below are the conditions which not necessarily provides a failure in the working of the project but gives unwanted results.

4.5.1 Limitation in converting image to text:

- For converting our image into text we are using pytesseract (Python-tesseract is an optical character recognition (OCR) tool for python). The image goes through various image processing steps and then finally we are able to extract the text from the image using pytessaract.

- We are able to convert almost most of the pictures into text format, but there are few conditions which provides certain unwanted results as shown:

1.Images with dull background colors:

- As in the process of image to text conversion we pass the image through various steps that is first converting the image into a greyscale image, then converting the greyscale image into a binarized image (Here using a suitable threshold value we compare every pixel of the image with the threshold value, and depending on the comparison every pixel in the image is given a black or white value.).

- In most of the cases the images are converted to a well binarized image and are converted into a suitable text, but there are some cases leading to unfavourable results

- Here we will take an image having dull background to see what really happens to the image processing system.

(Fig 4.5.1.1: Image with dull background)

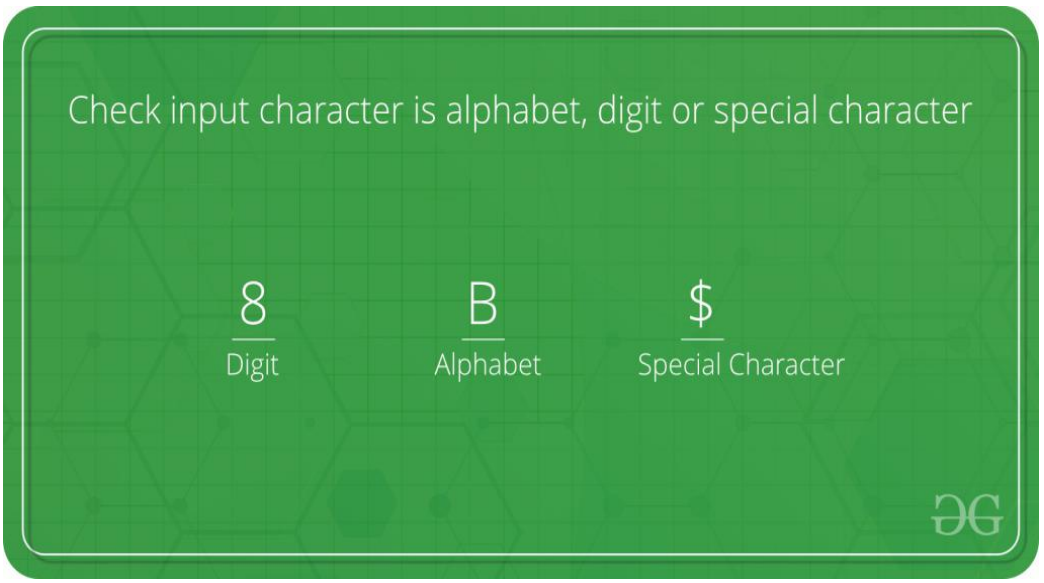
- So here we have a picture with grey background, after applying the image processing on the image we get the result as shown in fig 4.5.1.2, as we can see the image is distorted with missing parts of the text.

(Fig 4.5.1.2: Distorted image)

- As we see in fig 4.5.1.2 we get a distorted image which we can't use further for text processing, this is due to binarization of the picture, as we have set a certain threshold value for comparing every pixel of the image, certain pixels doesn't match with the process and instead of getting a proper black and white binarized image we get a distorted image with missing text from it.

2. Image with special characters:

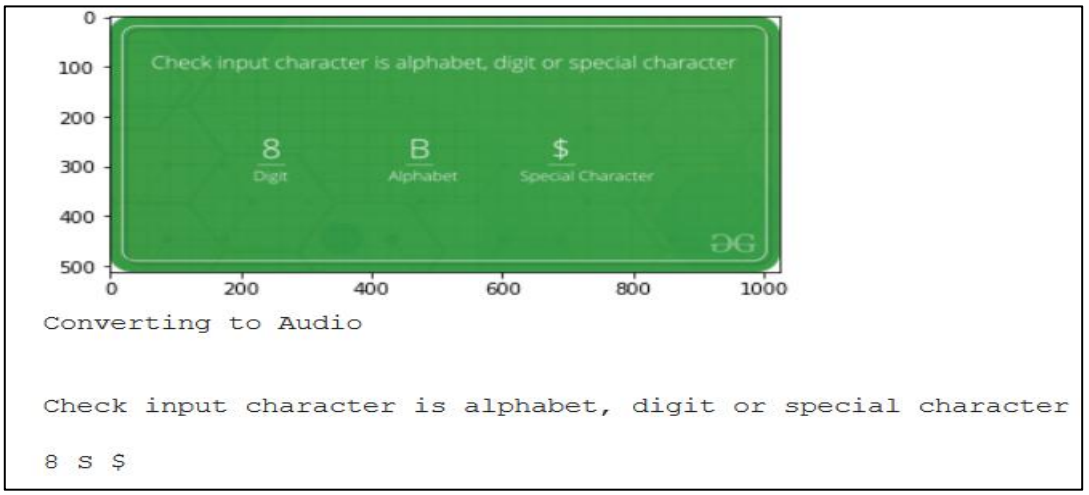
- Here we are looking with images having special characters (like !,@,$,etc)in them, our project is able to convert the image into a proper text but it fails to convert the text characters into speech.
- In some cases there are special characters which are not also identified and converted into text by the image processing and the pytessaract engine.
- Let us see the process by taking an image with special characters in it as show in fig:



(Fig 4.5.1.3: Image with special characters)

- Here in the above image (Fig 4.5.1.3) the project fails to identify and convert the special character 'B', as it is an unprecedented character which is not in the database of the OCR tool, it fails to identify the character and this proofs to be a limitation in the project.

- There are certain special characters which are identified by the OCR tool as seen in the Fig 4.5.1.4, like as we have in the image '$' which is also a special character, yet it is identified by the OCR tool.



(Fig 4.5.1.4: Output text of special character image)

- Even after the special characters are identified or not, in the end they can't be converted to audio as our model deepvoice3 used to convert text to image, converts normal text to audio; in terms of special characters the model is not trained with the suitable dataset to do so.

- Future improvement can be to train the model with special characters so that the model can also identify and convert the special characters into text as well as into audio format.

4.5.2 Limitation in Webpage and API:

- In the webpage we are retrieving images by using the address of the images (the address of the fig 4.5.1.3 is 'https://media.geeksforgeeks.org/wp-content/cdn-uploads/check-if-character-is-alphabet-digit-or-special-character-1024x512.png'), and we are further processing the images by sending the retrieved images to the backend code.

- As we know that images have some specific type of extensions like .jpg, .png, .jpeg, etc. So the webpage retrieves images with suitable extensions as we have seen in the above examples, but problems occur when we have an image having address without any extensions.

- So here I'll be showing the above problem by taking an image that fails to work in the system. We have a image with address 'https://www.wikihow.com/images/thumb/a/a3/Cite-Short-Stories-in-MLA-Step-3.jpg/v4-460px-Cite-Short-Stories-in-MLA-Step-3.webp', as we can see the address ends with an extension that is not used for images.

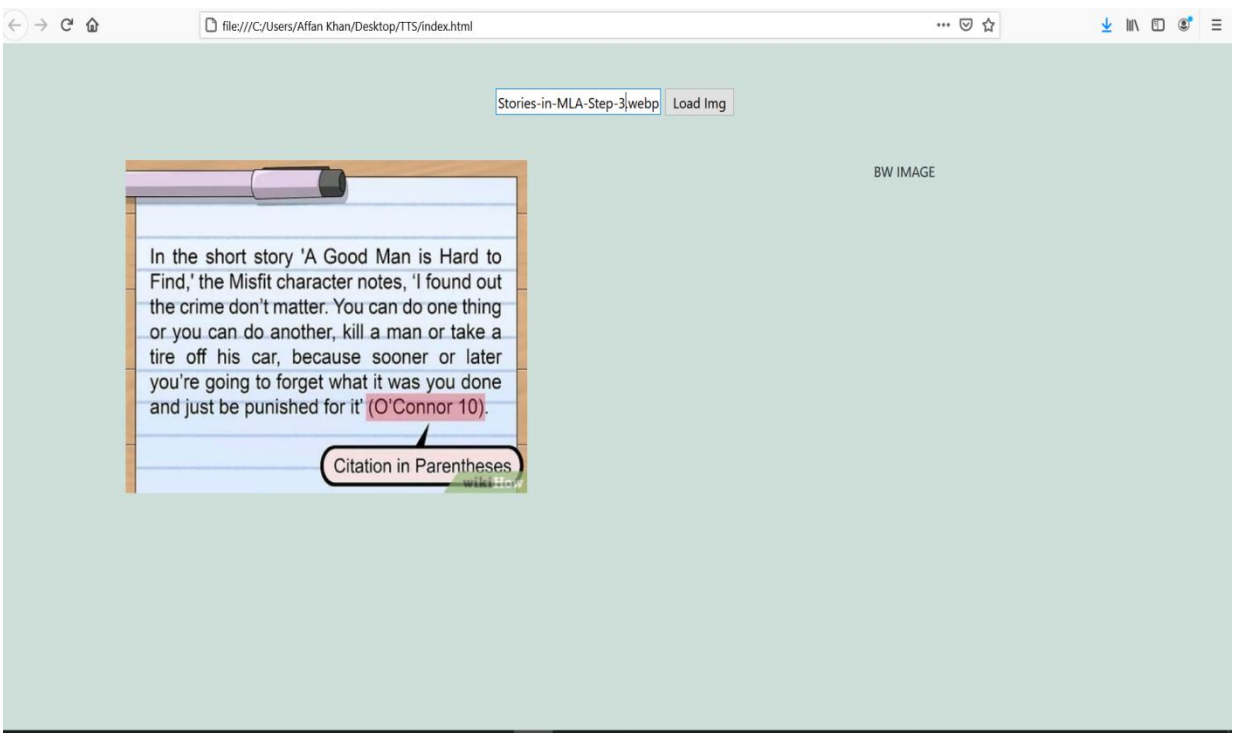


Fig 4.5.2.1: Webpage limitation

- As we can see in fig 4.5.2.1 the webpage cannot parse the image further due to invalid address of the image and therefore the image is not sent to the backend code and is not processed.

Chapter 5
Result and Conclusion

5.1. Conclusion

We have successfully implemented conversion of text image into audio format. We also developed technique for object detection in an image and cropping of textual part of image using OpenCV libraries. Our algorithm successfully processes the image and reads it out clearly with set volume. And also detect the object and crop the text part of image. This is an efficient as well as helpful device for the visually impaired, illiterate, or have a learning disability people. We have applied our algorithm on many images and found that it successfully does its conversion.

The existing systems encounters issues while performing scan on documents with complex backgrounds and the output is expected to have less accuracy. The proposed system ensures to read text present in the image for assisting blind people. Pre- processing part ensures efficient background separation with an improved algorithm.

5.2. Future Scope

The overall performance of this proposed device is quite positive with a few sections to be improved in future. A major improvement would be

1) It will be concentrated on developing an efficient product that can convert the text in the image to speech with high accuracy.

2) It can be used in blind schools and colleges. This can also be used as an application of artificial intelligence as it can be used for illiterate people.

3) We can make mobile application of same topic.

Chapter 6

References

[1] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O. Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, John Miller, “Deep Voice 3- Scaling Text-To-Speech With Convolutional Sequence learning” ICLR 2018.

[2] T.Rubesh Kumar, C.Purnima “Assistive System for Product Label Detection with Voice Output For Blind Users” International Journal of Research in Engineering & Advanced Technology 2014.

[3] Mallapa D.Gurav, Shruti S. Salimath, Shruti B. Hatti, Vijayalaxmi I. Byakod , “B-LIGHT: A Reading aid for the Blind People using OCR and OpenCV” International Journal of Scientific Research Engineering & Technology IJSRET-2017.

[4] A. Subbiah, T. Arivukkarasu, M. S. Saravanan, V. Balaji “Camera-based label reader for blind people” Int. Chem. Sci.: 14(S3), 2016, 840-844 ISSN 0972-768X.

[5] K Nirmala Kumari, Meghana Reddy “Image Text to Speech Conversion Using OCR Technique in Raspberry Pi” International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (An ISO 3297: 2007 Certified Organization) Vol. 5, Issue 5, May 2016.

[6] Nagaraja L, Nithin D, Nagarjun R S, Veena S Murthy, Nishanth M Anand “Vision based Text Recognition using Raspberry Pi” National Conference on Power Systems & Industrial Automation (NCPSIA 2015).

Chapter 7

Code of the project

7.1 Code for converting image to text and text to speech:

```
!sudo apt install tesseract-ocr
!pip install pytesseract
!pip install flask-ngrok
!pip install flask_cors
!pip install lws==1.2.6

%tensorflow_version 1.x
import tensorflow
import os
from os.path import exists, join, expanduser

# Clone
name = "deepvoice3_pytorch"
if not exists(name):
  ! git clone https://github.com/r9y9/$name
# Change working directory to the project dir
os.chdir('/content/deepvoice3_pytorch')

!git checkout 7a10ac6763eda92595e257543494b6a95f64229b --quiet


# Install dependencices
!pip install -q -e '.[bin]'
%pylab inline
! pip install -q librosa nltk
import torch
import numpy as np
import librosa
import librosa.display
import IPython
from IPython.display import Audio
# need this for English text processing frontend
import nltk
! python -m nltk.downloader cmudict

import pytesseract
import shutil
```

```
import os
import random
try:
 from PIL import Image
except ImportError:
 import Image
import tensorflow as tf
from flask import Flask, jsonify, make_response, request
from flask import Flask
from flask_cors import CORS, cross_origin
import threading
from skimage import io
from matplotlib.pyplot import imshow ,show
from flask import send_file

def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn

preset = "20180505_deepvoice3_ljspeech.json"
checkpoint_path = "20180505_deepvoice3_checkpoint_step000640000.pth"
if not exists(preset):
  !curl -O -L "https://www.dropbox.com/s/0ck82unm0bo0rxd/20180505_deepvoice3_ljspeech.json"
if not exists(checkpoint_path):
  !curl -O -
L "https://www.dropbox.com/s/5ucl9remrwy5oeg/20180505_deepvoice3_checkpoint_step000640000.pth"


import hparams
import json


# Load parameters from preset
with open(preset) as f:
  hparams.hparams.parse_json(f.read())

# Inject frontend text processor
import synthesis
import train
from deepvoice3_pytorch import frontend
synthesis._frontend = getattr(frontend, "en")
train._frontend =  getattr(frontend, "en")

# alises
fs = hparams.hparams.sample_rate
```

```
hop_length = hparams.hparams.hop_size

def tts(model, text, p=0, speaker_id=None, fast=True, figures=True):
  from synthesis import tts as _tts
  waveform, alignment, spectrogram, mel = _tts(model, text, p, speaker_id, fast)
  return waveform



from train import build_model
from train import restore_parts, load_checkpoint

model = build_model()
model = load_checkpoint(checkpoint_path, model, None, True)



def retriveAudio(img):
  img = img.convert('L')
  img.save('greyscale_noise.jpg')
  threshold=192
  output_image=img

  for x in range(output_image.width):
     for y in range(output_image.height):
        if output_image.getpixel((x,y))< threshold:
           output_image.putpixel( (x,y), 0 )
        else:
           output_image.putpixel( (x,y), 255 )

  img=output_image
  img.save('binarized.jpg')
  text = pytesseract.image_to_string(img)


  extractedInformation = text
  textArr=[]
  count=0
  tempStr=""
  splittedText = extractedInformation.split(' ')
  totalLen = len(splittedText)
  for indx,i in enumerate(splittedText):
    tempStr +=i+" "
    count +=1
    if count>=20 or indx==totalLen-1:
```

```
      textArr.append(tempStr)
      tempStr=""
      count=0
  # Try your favorite senteneces:)
  texts = textArr
  waveform = np.asarray([])
  print('Converting to Audio')
  for idx, text in enumerate(texts):
    print(text, end=' ')
    temp = tts(model, text, figures=False)
    waveform = numpy.append(waveform,temp)
  IPython.display.display(Audio(waveform, rate=fs,autoplay=True))
```

```
from flask import Flask
from flask_ngrok import run_with_ngrok
app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'
run_with_ngrok(app)

@app.route("/")
@cross_origin()
def hello():
    return "I am alive Affan vhv!"
@app.route('/loadAudio', methods=['POST'])
@cross_origin()
def mytts():
    imgSrc = request.get_json()
    image = io.imread(imgSrc['imgSrc'])
    imshow(image)
    show()
    retriveAudio(Image.fromarray(image))
    return send_file("binarized.jpg", mimetype='image/gif')

app.run()
```

7.2 HTML code for webpage:

```
<html>
  <head>
    <title>Text to Speech</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  </head>
  <body style="background-color:#cedfd9">


    <div class="col-xs-1 text-center" >
      <div id="above" class="m-5">
        <input type="text" id="imgSrc">
        <input type="button" id="sendButton" value="Load Img">
      </div>
      <div class="row justify-content-md-center m-5">
        <div id="og_img" class="col-sm-6 ">
          <h3>Orignal Image</h3>
          <p>Enter URL to proceed</p>
        </div>
        <div id="bw_img" class="col-sm-6">
          <h3>Processed Image</h3>
          <p>Final Image </p>
        </div>
      </div>
    </div>
    <script>


      var apiLink = "http://fb15e5984c6c.ngrok.io/";




      var httpLink ;
      function arrayBufferToBase64(buffer) {
      var binary = '';
      var bytes = [].slice.call(new Uint8Array(buffer));
```

```
        bytes.forEach((b) => binary += String.fromCharCode(b));

        return window.btoa(binary);
        };

        document.getElementById("sendButton").addEventListener('click',()=>{
           var imgSrc = document.getElementById("imgSrc").value
           httpLink = imgSrc
           var og_image_container = document.getElementById("og_img");
           og_image_container.innerHTML = "";
           var curr_image = document.createElement("IMG");
           curr_image.setAttribute("src", imgSrc);
           curr_image.setAttribute("width", "500");
           curr_image.setAttribute("alt", "OG IMAGE");
           og_image_container.appendChild(curr_image);
           let data = {"imgSrc": httpLink};
           fetch(apiLink+"loadAudio", {
              method: "POST",
              headers: {'Content-Type': 'application/json'},
              body: JSON.stringify(data)
              }).then(response=> {
                 response.arrayBuffer().then((buffer) => {
                    var base64Flag = '';
                    var imageStr = arrayBufferToBase64(buffer);
                    var curr_image = document.createElement("IMG");
                    curr_image.setAttribute("src", base64Flag + imageStr);
                    curr_image.setAttribute("width", "500");
                    curr_image.setAttribute("alt", "BW IMAGE");
                    var bw_image_container = document.getElementById('bw_img');
                    bw_image_container.innerHTML = "";
                    bw_image_container.appendChild(curr_image);


                    // document.getElementById('img').src = base64Flag + imageStr;
                 })
              })

        })




     </script>
  </body>
</html>
```