

# **DIGI WRITING**

Submitted in partial fulfillment of the  
requirements for the award of.  
Bachelor of Engineering degree in Computer Science and Engineering

By

**E LOCHAN ( Reg.No - 39110572)**  
**PATTIGULLA SAI PRADEEP ( Reg.No – 39110757)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

## **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A” by NAAC|12B Status  
BY UGC|APPROVED BY AICTE  
JEPPIAAR NAGAR, RAJIV GANDHISALAI,  
CHENNAI - 600119**

**APRIL - 2023**



**SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the Bonafide work of **E LOCHAN (39110572)** and **PATTIGULLA SAI PRADEEP(39110757)** who carried out the Project entitled "**LANGUAGE RECOGNITION FROM HANDWRITING BASED ON MACHINE LEARNING & DEEP LEARNING**" under my supervision from January 2023 to April 2023.

**Internal Guide  
Dr. P. ASHA, M.E., Ph.D**

**Head of the Department  
Dr. L. LAKSHMANAN, M.E., Ph.D.**

**Submitted for Viva voce Examination held on 20.04.2023**

---



**Internal Examiner**

**External Examiner**

## DECLARATION

I, **E LOCHAN** Reg.no-(39110757) and **PATTIGULLA SAI PRADEEP** Reg.no-(39110757) hereby declare that the Project Report entitled "**DIGI WRITING**" done by me under the guidance of **Dr. P. ASHA, M.E.,Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

*E Lochan*

**DATE:20.04.2023**

*pattigulla sai pradeep*

**PLACE: Chennai**

**SIGNATURE OF THE CANDIDATE**

## ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the Board **of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr.L.Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.P.ASHA M.E.,Ph.D.**,for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

## **ABSTRACT**

In order to develop the understandings of the machinery mind, there has been a lot of upfold in the development of machine learning. As humans, we learn how to do a task by learning it, and optimize the tasks by learning from the mistakes in the process. Just like the brain's neurons automatically trigger and quickly perform learnt tasks, machines can comprehend to the situations of strengthening developed neurons. Deep learning is just as interesting as the human brain's concept. Usage of different types of architectures for such neural networks collide for different types of problems, like image and sound classification, object recognition, image segmentation, object detection, etc. Following these layers of different commemorations and accuracy that Artificial Intelligence provides, one can come up with answers to many unresolved issues, problems, and tasks, since machine is now with the capability of approaching the task as if it is of a human's approach, but with the ability to drive through the task with the machine's work function. The fact that Machine Learning and Deep Learning had taken the industry up by storm is the very reason for its unending usages in this field. The very fabrics of precision and balance that a task completed under the machine's supervision is where we can entrust tasks that can bring a change for many lives with challenges, ailments and difficulties. The challenges that humans label as an impairment or a dysfunctional occurrence is where a machine could potentially step as a helping hand, or in our case, a hearing aid.

<b>Chapter No.</b>	<b>TITLE</b>	<b>Page No.</b>
	<b>ABSTRACT</b>	<b>V</b>
	<b>LIST OF FIGURES</b>	<b>VIII</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 <b>OBJECTIVE</b>	<b>1</b>
	1.2 <b>OUTLINE OF THE PROJECT</b>	
<b>2</b>	<b>AIM AND SCOPE OF THE PROJECT</b>	<b>5</b>
	2.1 AIM OF THE PROJECT	<b>5</b>
	2.2 <b>SCOPE OF THE PROJECT</b>	<b>5</b>
<b>3</b>	<b>LITERATURE SURVEY</b>	<b>6</b>
	3.1 INFRENCES FROM LITERATURE SURVEY	<b>6</b>
<b>4</b>	<b>PROPOSED METHODOLOGY</b>	<b>8</b>
	4.1 SELECTED METHODOLOGY FOR THE MODEL	<b>8</b>
	4.2 EXISTING SYSTEM DISADVANTAGES	<b>9</b>
	4.3 ARCHITECTURE OF THE MODEL	<b>10</b>
	4.4 AVAILING THE INITIAL PROJECT REQUIREMENTS	<b>10</b>
	4.5UNDERSTANDING OF MODELS	<b>23</b>
	4.6 IMPLEMENTATION OF MODULES	<b>24</b>
	4.7 POINTERS BEFORE DEVELOPING THE PROGRAM.	<b>25</b>

<b>5</b>	<b>PROJECT DESIGN AND IMPLEMENTATION</b>	<b>33</b>
	5.1 DEVELOPMENT AND DEPLOYEMENT	33
<b>6</b>	<b>RESULTS AND CONCLUSION</b>	<b>36</b>
	6.1 RESULTS OBTAINED	36
<b>7</b>	<b>CONCLUSION AND FUTURE ENHANCEMENTS</b>	<b>42</b>
	7.1 CONCLUSION	42
	7.2 FUTURE ENHANCEMENTS	42
	<b>REFERENCES</b>	<b>43</b>
	<b>APPENDIX</b>	
	A. SOURCE CODE	44
	B. SCREENSHOTS	53
	C. PUBLICATION	54

<b>Fig No.</b>	<b>LIST OF FIGURES</b>	<b>Page No.</b>
<b>4.2</b>	IMPLEMENTATION LAYERS IN TRAINING PROCESS.	<b>25</b>
<b>4.3</b>	ARCHITECTURE DIAGRAM	<b>37</b>
<b>5.1</b>	ALL THE MODULES USED IN THE CODE.	<b>33</b>
<b>6.1</b>	CURSIVE CLASSIFICATION	<b>36</b>
<b>6.2</b>	GRAPH DEPICTING THE ACCURACY PERCENTAGE	<b>37</b>
<b>6.3</b>	The VSC CONTAINING THE CODE	<b>38</b>
<b>6.4</b>	THE VSC CODE STATED TO THE DIRECTORY	<b>38</b>
<b>6.5</b>	RUNNING THE MAIN.PY FILE	<b>39</b>
<b>6.6</b>	RUNNING THE MAIN.PY WITH OTHER INPUT	<b>39</b>
<b>6.7</b>	BEFORE OPTIMIZATION	<b>39</b>
<b>6.8</b>	WRITING THAT IS REMOVED FROM CLASSIFICATION	<b>40</b>



# **CHAPTER 1**

## **INTRODUCTION**

Handwritten recognition is one of the important issues in pattern recognition applications. This technology is very handy to large scale companies to digitalize documents. One of the challenges visually impaired people have to overcome is read and comprehend day to day basis, apart from hearing ques other format of communication poses higher difficulty level. By using deep learning and Artificial Intelligence which will help in solving and can detect many patterns and approach it is of as human's approach. The aim of the project is to develop an end-to-end user interface enabled application for users which automatically recognize a written text of any font and style and convert it into standard digital format and will present you with an audio

### **1.1 OBJECTIVE**

The main objective of this project is to create a platform to digitalize the English language from a handwritten format.

### **1.2 OUTLINE OF THE PROJECT**

This search led us to learning and use the knowledge of Deep Learning, ML, and UI/UX, to develop an interface that allows you to bring forth a piece of information (currently in English), either written or typed, to the program as an input, and in return, the machine gives out a voice note of the very information that has been entered, along with numerous help for the set of information provided. Since Machine Learning has its niche in a spectrum of domains and researches, many approaches have been partially or totally established. Bayesian Network, Clustering, Decision Tree Learning and many more have collectively made Deep Learning only a part of the approaches. The following review mainly focuses on the collated usage of deep learning, the basics of forming this proposed application in said and different fields. Additionally, it presents several figures portraying the viable usage of this project across many nuances. The paper that helped the most to bring this

Impaired People” developed by Andrei Burta, Roland Szabo, and Aurel Gontean. The paper was issued in accordance of developing a speech application that could scan the given picture’s input and speak out the name/description of that product.

The project basically ran under the guidance of Convolutional NNs and Recurrent NNs. So, we learnt the utmost capabilities of CNN and RNN following the understandings of integrating multi-substantial layers to the model, which was later integrated to a rich and intuitive website UI to be used by people for test.

OCR stands for Optical Character Recognition, which is a technology that enables the computer to recognize and read text from an image or a document. OCR systems use complex algorithms and techniques to convert images of written or printed text into machine-readable text. OCR has many applications, such as digitizing paper documents, automated data entry, and text-to-speech conversion.

The OCR technology has come a long way since its inception in the early 20th century. Initially, OCR systems were designed to read printed text only, but now they can recognize handwritten text as well. OCR systems work by analyzing an image of text and recognizing the shapes and patterns of individual characters.

OCR systems use a combination of image processing techniques such as noise reduction, binarization, and segmentation to improve the accuracy of character recognition. Once the text has been recognized, it can be edited, searched, and indexed, just like any other digital text.

OCR technology has numerous applications across various industries. For example, in the banking industry, OCR is used for automated data entry of checks and invoices. In the healthcare industry, OCR is used to digitize medical records and to automate data entry for patient information.

OCR technology is also used in the publishing industry to convert printed books into digital format. The digitized books can then be searched, indexed, and accessed online.

OCR technology has also revolutionized the accessibility of printed materials for people with disabilities. OCR technology can convert printed text into speech, enabling

visually impaired individuals to access written content. This technology has been widely used in e-books and other digital publications.

In conclusion, OCR technology has come a long way since its inception and has become an essential tool for digitizing printed materials. OCR has numerous applications across various industries, and it has revolutionized the accessibility of printed materials for people with disabilities. As OCR technology continues to evolve, we can expect to see even more applications in the future.

The learning model needed an apt User Interface for the users to feel at ease while using the model. When creating a perfect UI for your website, there are many factors to consider. The first is to ensure that the layout is easy to navigate and that all important information is easily accessible with the proper information structure. The design should also be visually appealing to enable the visitors to use the model correctly.

It is also essential to ensure that the website is responsive and works well on all devices in a proper layout. As we knew that a nominal number of users might face visual impairments or meager understanding of the language barrier, we had to make a clear website that used enriched colors and better guidance system for the users to be well-guided in getting their predicted manuscripts right. The website development needed core understanding of HTML, CSS and Flask. Hence, we took the course of “Learn Flask for Beginners” by freeCodeCamp YouTube channel, and “Mastering HTML and CSS” course developed by Howard Hamilton and Sun Developers via Udemy. The courses with a combined screentime of 25 hours took us about 15-20 days of our learning and concentration. Post-learning the courses, we had started developing the apt intuitive website for the recognition model.

they must access the model from the website. To create an easy and accessible website, focusing on the UI’s ease of access was crucial. The UI were to be designed to quickly understand what they need to do to achieve their goal. The UX should be designed so that users can easily navigate through the model of the website.

The website had the following elements that we wished to add in – Home page explaining the websites' purpose, an About Us page introducing ourselves to the users, a Support page for the users to add their valuable feedbacks, and an "IMPLEMENT SCRIPT" button which leads the user to the file upload page. The websites' color scheme was designed to be orange and black since besides the color of red, orange and black are perhaps the more cautious and eye-catching color, and at the same time, soothing and appealing for the user to work with the website. The buttons are kept as a top bar at a rather large size to the right side of the website for the user to access the pages easily. The file upload and camera modes are also kept easily accessible and baselined with the design for the people of older generation to not be startled by the basic usage of a website

## **CHAPTER 2**

### **AIM AND SCOPE OF THE PROJECT**

#### **2.1 AIM**

The aim of the project is to develop an end-to-end user interface enabled application for users to avail the process of converting the given handwritten manuscript (input) into an automated voice note.

#### **2.2 SCOPE**

The scope of the project is to provide the people with partial or permanent visionary deficits to be able to understand/learn the information that is given to them in a written format, or even in typed format. This will help the said set of people able to understand the information from the manuscript in an instant without any external resource/support needed.

## **CHAPTER 3**

### **LITERATURE SURVEY**

#### **3.1 INFRENCES FROM LITERATURE SURVEY**

Qinge Xiao; Congbo Li; Ying Tang; Xingzheng Chen (2020) contributed to the IEEE Transactions on Automation Science and Engineering on developing an Energy Efficiency modelling that could configure a variably dependent cohesive model development using Machine Learning.

Tahir, Ghalib Ahmed; Loo, Chu Kiong (2020) contributed for an open-ended Continual Learning (CL) model that is deemed on Food Recognition by the means of Extreme Learning Machines using the incrementations of class via IEEE Access.

L. Liu, P. Wang., and J. Lin (2021) contributed for describing the detection of intrusion caused by imbalanced Network Traffic purely dependent on ML and DL via IEEE Access.

A thesis on enabling an Ensemble Hierarchical EL (Extreme machine learning) for dereverberation of different speech patterns was conducted by Tassadaq Hussain,

Sabato Marco Siniscalchi, Hsiao-Lan Sharon Wang, Yu Tsao, Salerno Valerio Mario, and Wen-Hung Liao via IEEE 2020

Dali Zhu; Long Yang; Zhanxun Li; and Hualin Zeng contributed to IEEE Symposium on Computers and Communications (ISCC Rennes, 2020) with a marvellous project on Remote Speech Extraction from a given Speckle Image using the functions of the modulated CNN (convolutional neural network).

Lavanya Bhaskar and Ranjith R attended the 2020 11th International Conference on Computing, Communication and Networking Technologies to demonstrate the Robust Text Extraction in Images for Personal Event Planner.

Fuwei Cui; Qian Cui; and Yongduan Song (2020) made a convoluted survey on Learning-Based approaches for computation and execution of Human-Machine Dialog Systems via IEEE.

Hao Tang; Hong Liu; Wei Xiao; and Nicu Sebe (2020) developed a thesis to demonstrate the intuitive connections of Machine Learning and Deep Learning's coding NNs for Image Recognition with Limited Data via IEEE.

Guoqiang Zhong; Kang Zhang; and Hongxu; Yuchen Zheng; and Junyu Dong (2019) discussed on the vast sectors of Marginal Deep Architecture, which further followed up on the learning modules and stacking features to develop Deep Learning Models via IEEE

Lazhar Khelifi and Max Mignotte (2020) came up with the analysis of different variants of Remote Sensing Images congregated with its Change Detection in reasoning to Deep Learning via IEEE

Asghar Ali Chandio, M. D. Asikuzzaman and Mark R. Pickering (2020) had developed a Cursive-writing recognition from the inputs of Natural Scenary by the usage multilevel CNNs and Neural Network Fusion via IEEE

Yi-Chao Wu, Fei Yin, Zhuo Chen, and Cheng-Lin Liu had developed a model on Handwritten Text Recognition that could recognize the handwritten Chinese manuscripts using Separable Multi-Dimensional RNNs via IEEE

Yucheng Zhou and Zhixian Gao (2019) propelled the idea of executing the Medical Motion Image Recognition with the combined help of Convolutional Neural Network and IoTs (internet of things) via IEEE

Yulia S. Chernyshova, Alexander V. Sheshkus and Vladimir V. Arlazarov came up with the development of a Two-Step CNN Framework that serves the purpose of Text Line Recognition from the images captured via cameras in different places via IEEE.

## **CHAPTER 4**

### **PROPOSED METHODOLOGY**

#### **4.1. METHODOLOGY**

So far, we have seen the needs of the project and the final product the algorithm aims for. Typically, any project's life cycle includes four stages:

Initiation - Where initial research is done and scope is identified. In our case, we have scouted our initial scope to be developing an accurate model and an intuitive UI for the model in the form of a website.

Planning - Where you create a plan, determine end results, and finalize the project timeline. The timeline of learning the given modules and languages, and starting to gather the given datasets, followed by developing the model and website takes an achievement time of 4–6 months depending on external and internal factors.

Execution - Where you break down the project into smaller milestones, start the work, collaborate, and reach the end goals of the project. The team of two here had assigned the tasks of developing the initial model and training the model as the first set of tasks for each member and developing the UI (website) and designing the logo and website



thoroughly for the user's ease as the second and final set of individual tasks. The final step of integrating the model to the website is as mentioned, an integrated teamwork.

Closure – The result is evaluated and checked for accessibility and quality. By the end of execution, minor debugging and updates with the model accuracy and website's responsiveness are necessary before deploying the product for the users.

Not just is this project for a sustainable end-product, but also a susceptible end-product for the users as mentioned before. Thus, the project needs the utmost care with every aspect of elements chosen – starting from the layers of neural networks chosen, accuracy of the learning model, accessible UI for the model website with the right elements and a proper integration of the model with the website so that in the website does not lag or execute a deadlock.

## 4.2 EXISTING SYSTEM DISADVANTAGES

Modern technology used for automatic document processing can tackle many types of damage that can happen to a hard copy of a document. That may be a few drops of mayonnaise from a morning sandwich, a signature written in the wrong part of the doc, or a smudged piece of printed text. In this short article, you'll see six common problems of image-to-text extraction that Alpha moon's AI OCR handles easily.

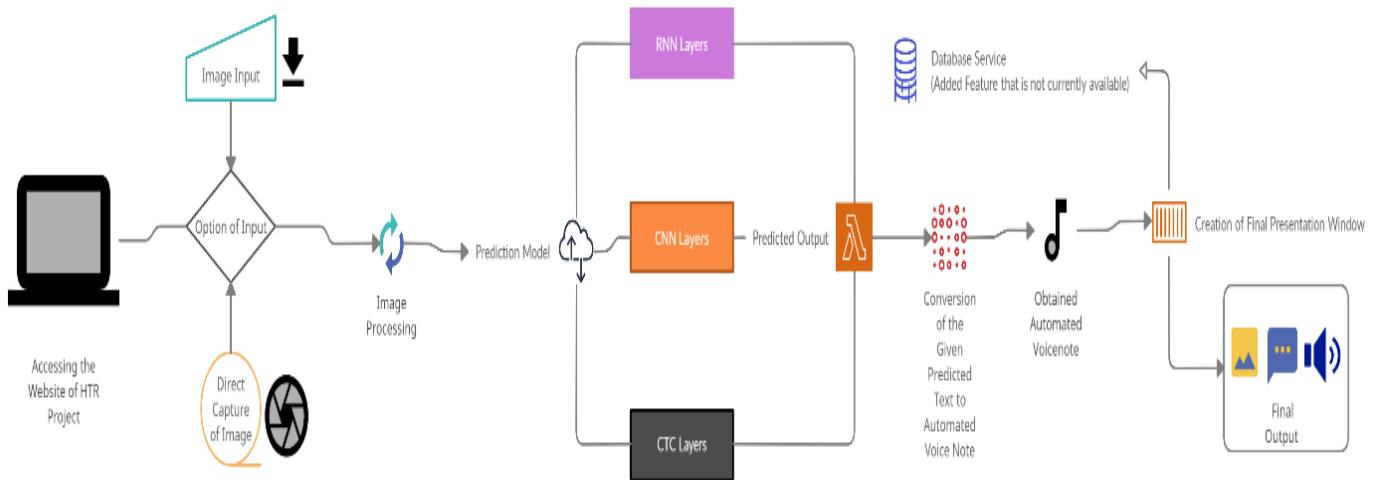
Even though companies lean towards process automation and digitization of documents, hard copies of documents are nonetheless far from becoming a thing of the past. In some cases, hard copies have far more importance than digitized copies. So is the situation in the legal sector where physical documents are the only valid evidence usable in court proceedings, whereas the goal of digitization is archiving.

The legal sector is not alone. Hard copies of documents are a common practice in finance, real estate, and medical sectors. A staggering 90% of professionals in the latter sector admit they're still heavily reliant on paper and manual processes.

Since physical documentation isn't going anywhere, it's best to automate as much of its processing as possible. Now, we'll cover the following issues with printed documents and shed some light on how Alpha moon's engine performs when encountering these obstacles:

- stains on documents
- multilingual signs in text
- blurred text
- "text on text" problem
- small characters

### 4.3 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM



**Fig.4.1. The flowchart of Language Recognition Project**

### 4.4. AVAILING THE INITIAL PROJECT REQUIREMENTS

The project mainly required the support of some key programming languages, the necessary package files, trainable datasets, and other expertise in employing the right and efficient APIs and Databases. With that in mind, we integrated the availability of the following tools:

#### 4.4.1. USAGE OF PYTHON, FLASK, AND WEB APPLICATIONS

In case of the back-end side of things, the correct flowing of the project requires the following set of installations and implementations – Ver 3.8 of Python, Ver 1.3

of TensorFlow, and any or latest versions of NumPy and OpenCV. These tools are used accordingly in a secure open-source IDE (such as Visual Studio Code in our case) to maintain an ideal and superlative process in the project. Some of the common Python packages are also taken into consideration since the project requires a few sets of APIs for conversions. The Python packages that we used in this project are – Flask, RoughLine, and Transfer. They have the purposes of integrating to the developed website, maintaining a small data sheet for the derived predictions, and congregation of the predictions to the database that will be integrated in future. (Since some of the high-level databases require financial installments)

Flask is a lightweight and open-source web application framework that allows developers to quickly build web applications in Python. Flask is known for its simplicity, flexibility, and easy-to-learn interface, making it an ideal choice for beginners as well as experienced developers.

Flask was created by Armin Ronacher in 2010, and since then, it has gained a lot of popularity due to its minimalist approach to web application development. Flask follows the Model-View-Controller (MVC) architecture, which helps to keep the code organized and maintainable.

One of the key features of Flask is its extensibility. Flask provides a robust extension system that allows developers to add additional functionality to their web applications quickly. There are a wide range of Flask extensions available, including libraries for database integration, authentication, and user management.

Flask also supports a wide range of templating engines, which allow developers to create dynamic HTML pages easily. Flask supports popular templating engines such as Jinja2, Mako, and Cheetah.

Flask is a great choice for developing small to medium-sized web applications. Flask is easy to learn and has a low learning curve, which makes it an ideal choice for developers who are new to web development. Flask provides a lot of flexibility and customization options, which means that developers can build applications that are tailored to their specific needs.

One of the key advantages of Flask is its simplicity. Flask does not come with a lot of built-in functionality, which means that developers can choose which libraries and tools they want to use. This makes Flask a lightweight and efficient framework that is easy to use and maintain.

Flask also provides excellent documentation and an active community. The Flask community is very supportive, and there are many resources available, including tutorials, documentation, and examples. This makes it easy for developers to learn Flask and to get help when they need it.

In conclusion, Flask is a lightweight and flexible web application framework that is easy to learn and use. Flask is ideal for small to medium-sized web applications and provides a lot of customization options. Flask is also extensible and has a wide range of extensions available, making it an ideal choice for developers who need to add additional functionality to their web applications. Flask's simplicity and excellent documentation make it an excellent choice for beginners as well as experienced developers.

In order to develop an intuitive user interface i.e., a website, that collects the data from user and gives the output in form of a voice note, we've employed the services of implementing framework using Bootstrap, and making the interactive UI using HTML & CSS, with the functionality procedures of Js. So far, we've thought of either making the model with just a one-time use and destroy interface that uses the input only once to replicate the output rather than saving the whole process. If the latter is in the need of implementation, we employ either AWS or Firebase for the creation of a nominal database. The website has deemed to contain all the logical and technical information regarding the project, such as About Us, Review Link, Project Explanation, and the actual application. This was fabricated from the beginning and was implemented with the usage of the above-mentioned web applications and Flask.

#### **4.4.2. USAGE OF CERTAIN APIS IN THE PROJECT**

Now, apart from all the Python and Web App implementations, certain APIs played a key role in developing the nuances of the project in a formidable way, such as GoogleTTS, which was the interface that delivered the automated voicenote i.e., conversion of derived text from the model to speech to the user. This was one of the easier options to use for the text-to-speech module, since developing another model or program to perform that task might cause a clash of memory input and complexity with the ongoing text-prediction model. Hence, GTTS was the right option to choose. Apart from this, we implemented an easy core socket for the future developer to add a database to the project with ease and develop a directory without hassles. We have left it as an option of access between the modern cloud-based servers such as AWS or Firebase as we mentioned earlier.

There are several APIs (Application Programming Interfaces) that can be used in OCR (Optical Character Recognition) applications. These APIs offer pre-trained machine learning models and other tools that can help developers build accurate OCR systems quickly and easily. Here are some popular OCR APIs:

1. Google Cloud Vision API: This API provides OCR capabilities, along with other image analysis tools such as face detection, label detection, and object recognition. The OCR feature can detect and extract text from images in over 50 languages and can be used to recognize handwritten text as well as printed text.
2. Microsoft Azure Computer Vision API: This API provides OCR capabilities along with other image analysis tools such as object detection, color analysis, and image tagging. The OCR feature can detect and extract text from images in over 25 languages and can be used to recognize handwritten text as well as printed text.
3. Tesseract OCR: This is an open-source OCR engine developed by Google. Tesseract can be integrated into various programming languages such as Python, Java, and C++. Tesseract can recognize over 100 languages and can be used to recognize handwritten text as well as printed text.

4. Amazon Textract: This API provides OCR capabilities along with other document analysis tools such as table extraction and form recognition. The OCR feature can detect and extract text from images in over 60 languages and can be used to recognize handwritten text as well as printed text.
5. OCR.Space API: This is a cloud-based OCR service that can recognize over 100 languages. OCR.Space API provides a simple REST API interface, making it easy to integrate with various programming languages.

When selecting an OCR API, developers should consider factors such as accuracy, speed, language support, and cost. Some APIs may offer more features than others, so developers should choose an API that meets their specific needs. It's also important to consider the API's documentation and support resources to ensure that developers can get help when they need it.

In conclusion, there are several OCR APIs available that can help developers build accurate OCR systems quickly and easily. These APIs offer pre-trained machine learning models and other tools that can recognize handwritten text as well as printed text. Developers should consider factors such as accuracy, speed, language support, and cost when selecting an OCR API.

#### ***4.4.3. GATHERING OF THE DATASET FROM IAM AND CLASSIFICATION FOR LINE AND WORD FORMAT***

The dataset that was used to train the model with is from the IAM Dataset Collection. The IAM Handwriting Database is a publicly accessible and freely available handwriting dataset that is used for non-commercial research purposes. Upon using data from the IAM Handwriting Database, a registration is required in order to be aware of who is using their data. Since we are publishing scientific work based on the IAM Handwriting Database, we were requested to include IAMs reference to the paper.

The IAM Handwriting Database is hierarchically structured into forms that are of different speech sets: “words” database for the training of model with singular

words, “lines” database to train model with a couple of words (1-3 words in a line), and “sentences” database that could train the model for a set of sentences (1-2 lines). The datasets that we used in our project were lines and sentences databases. The model had showed steady epochs with the given database. IAM also provided certain broken images (that we implemented to filter separately in our model) to find the necessary utterances that could help the model identify outliers, or in our case, disoriented or meaningless words.

#### **4.5. UNDERSTANDING OF THE MODEL’S CORE REQUIREMENTS/MANAGEMENT PLAN**

To manifest the application that we dreamt of seeing in action, we decided to implement the base of the application on the cores of the following model – Convolutional Neural Network, that has always been known as the apt Neural Network model to handle and process audio, image, and video clips, Long Short-Term Memory (LSTM) implementation of Recurrent Neural Network (RNN) has proven applicable to compile and transfer information in a larger surface of data layer with better data training, and the Connectionist Temporal Classification (CTC) links with the RNN to compute the loss value. CTC’s interference comes up as the task of mining the given array of dataset and finding the apt data node for a certain tested data and give the final text output.

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are two of the most popular types of neural networks used in deep learning. While CNNs are often used for image and video recognition tasks, RNNs are used for processing sequential data such as text and speech.

##### **Convolutional Neural Networks (CNNs)**

CNNs are a type of feedforward neural network that is designed to process images and videos. They are widely used in computer vision applications such as object detection, image classification, and image segmentation.

CNNs use a technique called convolution, which involves applying a set of filters to the input image. Each filter detects a specific feature of the image, such



as edges or corners. The output of each filter is then combined to produce a feature map, which represents the most important features of the image.

CNNs typically consist of several convolutional layers, each followed by a pooling layer. The pooling layer reduces the dimensionality of the feature map by subsampling, which reduces the amount of computation required in subsequent layers.

One of the key advantages of CNNs is their ability to learn features automatically. This makes them well-suited for tasks such as image classification, where the features that are important for distinguishing between different classes may not be known in advance.

### Recurrent Neural Networks (RNNs)

RNNs are a type of neural network that is designed to process sequential data such as text and speech. They are widely used in natural language processing (NLP) tasks such as language translation, speech recognition, and sentiment analysis.

Unlike feedforward neural networks, which process input data in a single pass, RNNs process sequential data by maintaining a hidden state that is updated at each time step. The hidden state acts as a kind of memory that allows the network to capture long-term dependencies between different elements of the sequence.

RNNs are trained using a technique called backpropagation through time (BPTT), which is a variant of the standard backpropagation algorithm. BPTT involves computing the gradients of the loss function with respect to the parameters of the network at each time step, and then propagating these gradients backwards through time to update the parameters.

One of the key advantages of RNNs is their ability to model sequential data of variable length. This makes them well-suited for tasks such as speech recognition, where the length of the input sequence can vary depending on the length of the spoken utterance.

One of the challenges of RNNs is the vanishing gradient problem, which occurs when the gradients of the loss function with respect to the parameters of the network become very small as they are propagated backwards through time. This can make it difficult for the network to learn long-term dependencies between different elements of the sequence. To address this problem, several variants of RNNs have been developed, including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks.

In conclusion, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are two of the most important types of neural networks used in deep learning. While CNNs are typically used for image and video recognition tasks, RNNs are used for processing sequential data such as text and speech. Both types of networks have their own strengths and weaknesses, and the choice of which type to use depends on the specific requirements of the task at hand.

The Connectionist Temporal Classification (CTC) is a technique used in deep learning that is particularly useful for speech and handwriting recognition tasks. The CTC algorithm allows a neural network to learn how to map variable-length input sequences to variable-length output sequences, without the need for explicit alignment between input and output sequences.

In traditional sequence-to-sequence models, the length of the input sequence is fixed and the length of the output sequence is known in advance. In contrast, CTC allows the length of both the input and output sequences to vary, which is important in tasks such as speech recognition where the length of the spoken utterance may vary from one example to another.

The CTC algorithm involves training a neural network to predict a sequence of labels that correspond to the input sequence. The output of the neural network is a probability distribution over a set of possible labels at each time step. The CTC algorithm then uses a dynamic programming approach to convert this sequence of probabilities into a final output sequence by collapsing repeated labels and removing blank labels.

The CTC algorithm allows the neural network to learn how to align the input and output sequences implicitly, without the need for a separate alignment step. This is important in tasks such as speech recognition, where the alignment between the spoken words and the corresponding text may not be known in advance.

The CTC algorithm has been used in a wide range of applications, including speech recognition, handwriting recognition, and music transcription. It has been shown to achieve state-of-the-art performance in many of these tasks, especially when combined with other deep learning techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

One of the key advantages of the CTC algorithm is its ability to handle variable-length input and output sequences. This makes it well-suited for tasks such as speech recognition, where the length of the spoken utterance may vary from one example to another.

In conclusion, the Connectionist Temporal Classification (CTC) is a powerful technique used in deep learning for variable-length sequence-to-sequence mapping. The CTC algorithm allows a neural network to learn how to map variable-length input sequences to variable-length output sequences without the need for explicit alignment between input and output sequences. The CTC algorithm has been successfully used in a wide range of applications, including speech recognition, handwriting recognition, and music transcription. Its ability to handle variable-length input and output sequences makes it particularly useful for tasks where the length of the sequence may vary from one example to another.

To see every neural networks' purpose in detail, we have to understand the neural network in detail. In the case of a Convolutional Neural Network, they are considered the most volumetric Neural Network for the amount of data to be streamed from it. CNN is a type of neural network that would only require a specific convolutional layer, but can have other types of nonlinear, pooling, and fully connected layers as well, which would create a deeper convolution to the pre-existing convolutional neural network. Depending on our application, CNN

can be beneficial, since to handle multiple words in a sentence, we require a lot of accuracy to be built up. However, it brings additional parameters for training. As we know, in the CNN, convolutional filters are trained via backpropagation method. The shapes of the filter structure depend on the given sentence, or sometimes the input. For instance, let us review the application, one filter can perform letter extraction, whereas another can carry out curve extraction. However, we do not fully control these filters in CNN, and their values are determined through learning.

Numerous works have been published on the LSTM type of an RNN, many of them reporting on the results achieved across a wide variety of application domains where data is sequential. The impact of the LSTM network has been notable in language modeling, speech-to-text transcription, machine translation, and other applications. All major open-source machine learning frameworks offer efficient, production-ready implementations of a number of RNN and LSTM network architectures. The advantage of this lengthier LSTM path is that it affords an opportunity to build a correct choice of selection that can prove beneficial during all phases of the process of incorporating an open-source module to suit the needs of their research effort or a long data collecting application, preparing the dataset, data troubleshooting, and tuning.

LSTM, or Long Short-Term Memory, is a type of neural network architecture that is commonly used for tasks related to sequential data, such as speech recognition, natural language processing, and time series prediction. LSTM networks are particularly effective at handling long-term dependencies between inputs, which can be a challenge for other types of neural networks.

At its core, an LSTM network consists of a series of memory cells that are connected by gates. These gates allow the network to selectively add, remove, or update information in the memory cells. The three types of gates used in an LSTM network are the input gate, the forget gate, and the output gate.

The input gate determines how much new information should be added to the memory cells based on the current input. The forget gate decides how much information should be discarded from the memory cells based on the previous

output and the current input. Finally, the output gate controls how much information should be output from the memory cells to the next layer of the network.

Each of these gates is implemented as a sigmoid activation function that outputs values between 0 and 1. These values are then multiplied with the input or output to control the flow of information through the network. The memory cells themselves are implemented as simple recurrent neural network (RNN) units that maintain a value over time.

Training an LSTM network involves updating the weights and biases of the network to minimize a loss function. The most common approach is to use backpropagation through time (BPTT), which involves unrolling the network over a sequence of inputs and then applying the backpropagation algorithm to update the weights and biases.

One of the key advantages of LSTM networks is their ability to handle vanishing gradients, which can occur when gradients become too small to be useful for updating the weights and biases. This is a common problem with traditional RNNs, which can struggle to maintain information over long sequences. LSTMs avoid this problem by using the gates to control the flow of information and ensure that useful gradients are propagated through the network.

Another advantage of LSTM networks is their ability to handle variable-length input sequences. This makes them particularly useful for tasks such as natural language processing, where the length of the input text can vary greatly. LSTMs can process each word in the input sequence independently, while still maintaining a memory of previous words.

Overall, LSTM networks have proven to be a powerful tool for handling sequential data, with applications in a wide range of fields. While they can be more complex to train than traditional neural networks, their ability to handle long-term dependencies and variable-length sequences make them a valuable addition to any machine learning toolkit.

Labelling unsegmented sequence data is rather a tedious problem in a sequence learning model. It is particularly common in tasks such as handwriting recognition, speech recognition, gesture recognition and any other learning task where noisy, real-valued input streams are annotated with strings of discrete labels, such as letters or words, such as our project. This is where Connectionist Temporal Classification comes to play. A CTC network has a SoftMax output layer with one more unit than there are labels in  $L$ , the labels being every predicted word's accuracy. The activations of the first  $L$  units are interpreted as the probabilities of observing the corresponding labels at particular times. The activation of the extra unit is the probability of observing a 'blank', or no label, defining the letter/word has no value.

OCR (Optical Character Recognition) is a technology that enables digital devices to recognize printed or handwritten text from images or documents. Both Android and iOS platforms offer OCR capabilities through various third-party apps and native features. However, there are some differences between OCR in Android and iOS.

### 1. Platform-specific APIs

Android and iOS platforms offer different APIs (Application Programming Interfaces) for OCR. Android provides the Google Mobile Vision API, while iOS provides the Vision framework. These APIs offer developers with pre-built OCR features to integrate into their apps.

Google Mobile Vision API uses Google's machine learning technology to recognize text from images, while the Vision framework in iOS uses Apple's Core ML technology. The two APIs use different algorithms, and this may result in different accuracy levels when recognizing text.

### 2. Performance

The performance of OCR on Android and iOS platforms varies depending on the device's hardware and software specifications. Android devices are

available in different hardware configurations, and this may result in different OCR performance levels.

In contrast, iOS devices are limited to Apple's hardware specifications. As a result, OCR performance on iOS devices may be more consistent than on Android devices.

### 3. Integration with platform-specific services

Android and iOS platforms provide different services that can be integrated with OCR technology. For example, Android devices can integrate with Google Translate, which allows users to translate recognized text into different languages. iOS devices, on the other hand, can integrate with Siri, which enables users to interact with recognized text using voice commands.

### 4. User interface

OCR apps on Android and iOS platforms may have different user interfaces, depending on the developer's preferences. However, both platforms offer similar user interface elements, such as camera access, text recognition, and editing capabilities.

### 5. Availability of OCR apps

There are several OCR apps available on both Android and iOS platforms. Some popular OCR apps on Android include Google Lens, Text Scanner OCR, and Adobe Scan. On iOS, popular OCR apps include Microsoft Office Lens, Scanner Pro, and OCR Scanner.

OCR on Android and iOS platforms has some similarities and differences. Both platforms provide OCR APIs for developers, but they use different machine learning algorithms. OCR performance on Android may vary due to the wide range of hardware configurations available, while iOS devices offer more consistent performance. Additionally, both platforms offer integration with different services and apps. Ultimately, the choice of platform for OCR may depend on the user's specific needs and preferences.

Some of the key points of improvement agreed to be implemented as a future update were the following:

- The development of an Android and iOS application for the following model could bring a lot of ease for anyone wishing to use this model from a convenient place anytime. This would need the deployment of the model to a user-friendly application via ADS (Android Development Studio) or Swift for Android and iOS application respectively.
- The development of the model in different languages used in our country commonly such as Hindi, Tamil, Telugu, Marathi, Urdu, etc. for the users belonging to certain regions of India be able to convert manuscripts of certain language to the automated voice notes. So far, the model can assess data and labels of different languages perfectly, in accordance to a completely prepared and well-defined dataset and labels of that language to be used for training the model.

Assessing the neural network with adding a few more CNNs and RNNs for more accuracy. According to our experimentation, the minimal number of layers needed for the model to have a threshold accuracy of identifying a few to couple of lines to be predicted is about 5 for Convolutional NNs and 2 for Recurrent NNs. Any more RNNs added to the model results in a good accuracy of predicting the correct phrase from the given inputs of a CNN, which in turn may also require a good improved input from CNN's end that needs more significant number of CNN layers. A predictive and accurate scoring from CTC can also be prepared for an accurate loss error rate of the predicted words

Together, these outputs of the collective neural networks define the probabilities of all possible ways of aligning all possible sequences of the words with the input sentence. The total probability of any one labelled prediction can then be found by adding the probabilities of the sentence's different alignments in the neural network summation.

TensorFlow is an open-source software library for machine learning and artificial intelligence applications. It was developed by Google Brain Team and was released in November 2015. TensorFlow is written in C++, and it provides



a user-friendly interface in Python, which makes it easy to use even for beginners.

The main idea behind TensorFlow is to make the implementation of machine learning algorithms more accessible to developers and researchers. It provides a flexible, high-performance platform that can run on a variety of hardware, including GPUs and TPUs (Tensor Processing Units), which are specialized hardware accelerators for machine learning tasks.

TensorFlow is used in a wide range of applications, including image and speech recognition, natural language processing, robotics, and even games. It provides a range of pre-built models and tools that can be used to build complex machine learning algorithms quickly and efficiently.

One of the key features of TensorFlow is its ability to build and train deep neural networks. Deep learning is a subset of machine learning that involves training neural networks with multiple layers. TensorFlow provides a range of built-in functions that can be used to create and train deep neural networks. It also supports a variety of optimization algorithms that can be used to improve the performance of these networks.

Another important feature of TensorFlow is its ability to handle large datasets. Machine learning algorithms require large amounts of data to be trained effectively. TensorFlow provides a range of tools for loading and preprocessing large datasets, including tools for data augmentation, image cropping, and resizing.

TensorFlow also provides a range of visualization tools that can be used to monitor the performance of machine learning models during training. These tools can be used to identify issues with the model and to fine-tune the parameters to improve its performance.

In conclusion, TensorFlow is a powerful and flexible machine learning platform that has revolutionized the field of artificial intelligence. It provides a range of tools and functions that can be used to build complex machine learning models

quickly and efficiently. TensorFlow is easy to use, even for beginners, and is widely used in a variety of applications, from image and speech recognition to robotics and games. With its continued development and expansion, TensorFlow is likely to remain a key player in the world of machine learning and artificial intelligence for years to come.

The final NN model had consisted of 5 Convolution NN and 2 Recurrent NN layers, which outputs a character-probability matrix. This matrix is either used as a throughput to the model improvement techniques, that were displayed to improve the recognition accuracy of the shown developed model.

For improving the recognition accuracy of the model, we followed some of the common regimens – we enabled Data augmentation to be used as to expand the size of the dataset by enabling non-sequential input transformation of the images via Regeneration and Duplication.

By increasing the size of the input of the NN layer, if the layer is large enough, we could enable the text-lines completely. We confined the output words to be a proper word from the dictionary is permitted, and if the dictionary doesn't seem to find the predicted word from the input, then a text correction is done

by searching for the most similar one, and the cursive writing style was removed from the input images, or classified as broken images to conclude the input invalid for conversion

#### **4.6. IMPLEMENTATION OF THE PYTHON MODULES**

As the first block, a Neural Network is built that is completely trained on images of words and sentences from a designated dataset (IAM). Keeping the input layer moderately small for the word/line images, every other layer is compelled with the initial layers, making it feasible for the NN training on the CPU. These are the initial steps to build a base (i.e., a Neural Network) for the conversion that we're aiming in this project. The main layers that are needed in this case are as follows: Convolutional NN layer for depicting the word-line image, Recurrent NN for the feature scaling, and a final layer of CTC (Connectionist

Temporal Classification) for length and relevancy domain. The depiction of the Neural Network is also feasible in a more formal way as a functional equation, mapping a precise image  $M$  (considering a matrix) with a length between 0 and  $L$ . The necessity of classifying the words on a character level is plausible for the improvement of the model's overall purpose of increasing its accuracy on the "words and sentences" scheme of things. Thus,

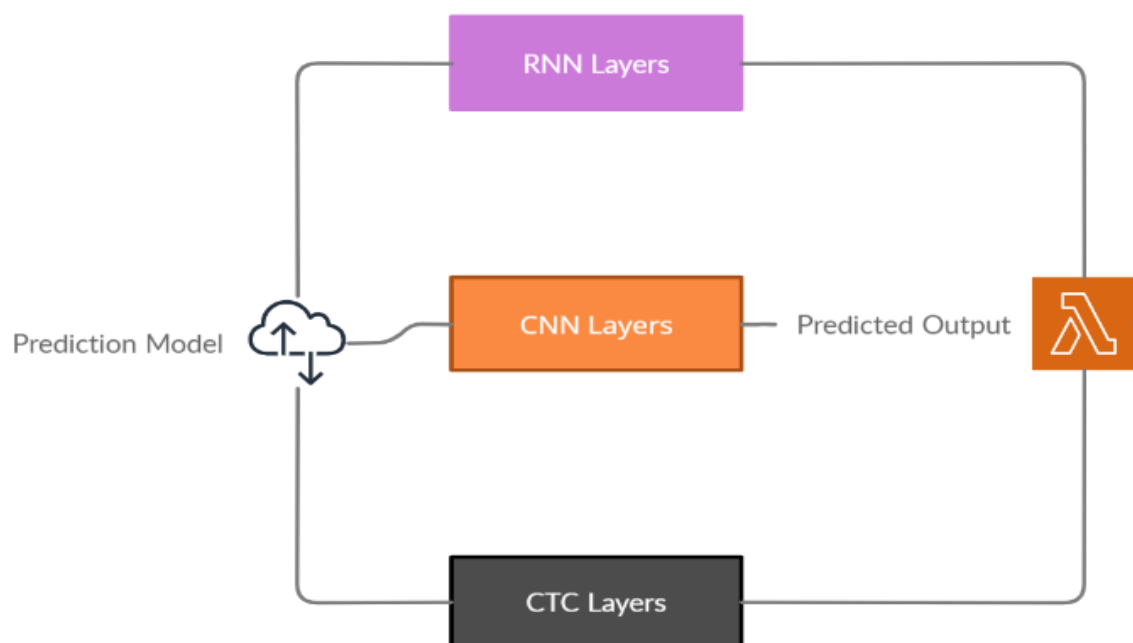
$$\text{NN: } \underset{W \times H}{M} \rightarrow (\underset{0 \leq n \leq L}{C_1, C_2, \dots, C_n})$$

With the implementation of the Python modules in this project, we use about 5 modules that are assigned to the various tasks of the prediction model and text-to-speech conversion. Firstly, we have the "sample pre-processor" module that pre-processes the dataset of images for the model. Which is followed by "data loader" module that reads the obtained samples and loads them in batches to create a sort of iterator-interface to read the whole set of samples, and the actual "model" that predicts the word/sentence from the given input using the samples' pattern. The "T2SConvert" module converts the predicted sentence from the input to an automated voice output for the user as the final product. As a main method to make a collective

working of all the modules mentioned above, we use a HTRWork.py module that integrates the model's functions, and links with the flask python module to be integrated with the website/UI.

In the case of CNN, the CNN layers fix the output rate length at a sequence of 32. There are 256 features for every entry, which are processed furthermore by the secondary layers of RNN. Even amidst all this, some cases are definitive where high correlation is seen amongst a few sets of handwriting from the input, such as the characters "e" and "l" have a lot in common when it comes in the cursive handwriting, or in the case of duplicate or misspelt characters (classic case of comparing "ll" to "tt"). In the case of the writing of the word "little", it can

be seen that most of the time, the predictions of the images are done in contrast to the placement of that letter in the word, like the letter "i" in comparison to the letter "L" or "t". Since the CTC operation is a position-free entity and seldom goes out of proportion for a handwritten word, the characters "l", "i", "t", "e" and the void label of the text's classification is a minimalistic task to decode, hence we concatenate the most similar pattern of the words in the sentence to that output line. This creates a best-case scenario for that sentence, thus eliminating all repeated words, letters, and blanks, and with that, we conclude "l---ii--t-t--l---e" → "l---i--t-t--l---e" → "little".



**Fig.4.2.IMPLEMENTATION LAYERS IN TRAINING PROCESS.**

#### **4.7. POINTERS BEFORE DEVELOPING THE PROGRAM.**

With the implementation of the said neural networks and the python modules, it is time to get forward with developing the learning model, and finally, the Language Recognition program. Before we move forward with the development part, please make sure to keep the following pointers in mind:

- Nowhere must the files that are about to be used and executed in this program leave or be accessed from a different directory altogether. Please try to maintain all the files regarding the project to be maintained in one directory for the ease of access.
- Please do not run the model in cmd (Command Prompt) for its smooth running. You may find the cmd to execute to model just as fine as any other IDEs, however the main drive will hold unnecessary cache that will be associated with the main directory in future use and cause unwanted disputes of running the file in any other platform.
- Please do not refrain from using different dataset (either from third party websites, or your own created ones) on this model, since you might want to develop the same model with any different language, such as your mother tongue. This will be of great use to the world regardless, so please access the “regain” notepad file from the data file for more information on that.
- Do not shift any file from the given folder to any other folders and subfolders in the main directory as it could cause confusions while executing the end product.

#### **4.8 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT**

So far, we have seen the needs of the project and the final product the algorithm aims for. Typically, any project’s life cycle includes four stages:

Initiation - Where initial research is done and scope is identified. In our case, we have scouted our initial scope to be developing an accurate model and an intuitive UI for the model in the form of a website.

Planning - Where you create a plan, determine end results, and finalize the project timeline. The timeline of learning the given modules and languages, and starting to gather the given datasets, followed by developing the model and

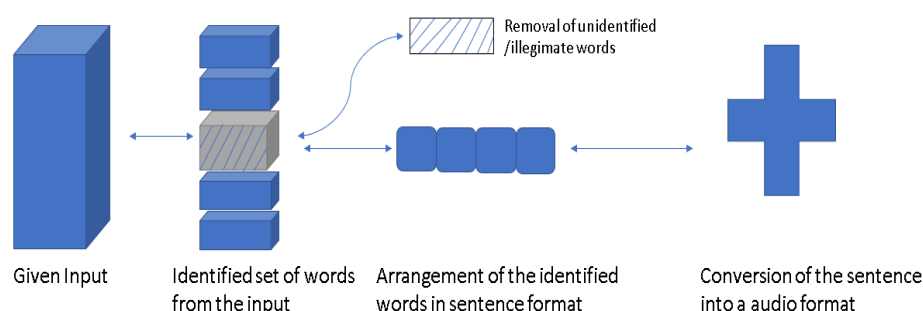
website takes an achievement time of 4-6 months depending on external and internal factors.

Execution - Where you break down the project into smaller milestones, start the work, collaborate, and reach the end goals of the project. The team of two here had assigned the tasks of developing the initial model and training the model as the first set of tasks for each member, and developing the UI (website) and designing the logo and website thoroughly for the user's ease as the second and final set of individual tasks. The final step of integrating the model to the website is as mentioned, an integrated teamwork.

Closure – The result is evaluated and checked for accessibility and quality. By the end of execution, minor debugging and updates with the model accuracy and website's responsiveness are necessary before deploying the product for the users.

Not just is this project for a sustainable end-product, but also a susceptible end-product for the users as mentioned before. Thus, the project needs the utmost care with every aspect of elements chosen – starting from the layers of neural networks chosen, accuracy of the learning model, accessible UI for the model website with the right elements and a proper integration of the model with the website so that the website doesn't lag or execute a deadlock.

The project mainly required the support of some key programming languages, the necessary package files, trainable datasets, and other expertise in employing the right and efficient API and Databases. With that in mind, we integrated the availability of the following tools:



### ***Fig.4.3. Architecture diagram***

- Usage of Python, Flask, and Web Applications
- Usage of Certain APIs in the Project
- Gathering of the Dataset from IAM and Classification for Line and Word Format

The IAM Handwriting Dataset is one of the most popular and widely used datasets in the field of handwriting recognition. It was created to enable researchers to develop and evaluate algorithms for handwriting recognition and related tasks, such as text line segmentation and word spotting.

The dataset contains a large collection of handwritten text images, which were captured from forms, letters, and other types of documents. The images were collected using a flatbed scanner and a digital camera, and they were then processed to remove noise and artifacts. The dataset also includes the corresponding ground truth transcriptions, which were generated by human annotators.

The IAM Handwriting Dataset is a highly versatile dataset, as it contains a wide variety of handwriting styles, including cursive, printed, and mixed styles. It also includes text in multiple languages, such as English, German, and French. The dataset consists of over 1,000 pages of handwritten text, which makes it one of the largest and most comprehensive datasets of its kind.

One of the key strengths of the IAM Handwriting Dataset is that it contains text in a variety of contexts and formats. For example, it includes unconstrained handwritten text, such as letters and notes, as well as structured handwritten text, such as forms and tables. This makes it well-suited for developing and testing algorithms for a wide range of applications, such as handwriting recognition for postal automation, document analysis, and historical document preservation.

The IAM Handwriting Dataset has been used in numerous research studies and competitions in the field of handwriting recognition. It has also been used as a

benchmark dataset for evaluating the performance of various handwriting recognition algorithms. Researchers have achieved high levels of accuracy on this dataset, which demonstrates its effectiveness as a tool for developing and evaluating handwriting recognition systems.

In conclusion, the IAM Handwriting Dataset is a valuable resource for researchers and practitioners in the field of handwriting recognition. Its large size, diverse content, and high quality ground truth transcriptions make it an ideal dataset for developing and testing algorithms for a wide range of applications.

The algorithms used in the project are solely based on the models of CNN, RNN, CTC and LSTM. The flow goes in the manner of deriving the data from the dataset that we use in the program, the IAM Dataset. Following that, we use the pre-processing methods to batch, loss-take and load the data into the model. The model also has a character error check module which will continually improve the model's accuracy as the error is received via the model running, and keeps the model in its sanity if the model has predicted the input just about right.

OpenCV (Open Source Computer Vision Library) is a popular open-source computer vision and machine learning software library. It is written in C++, and provides a wide range of functions for processing and analyzing images and videos in real-time. The library is cross-platform and runs on Windows, macOS, Linux, Android, and iOS.

OpenCV was first developed by Intel in 1999 and was later maintained by Willow Garage and now supported by the OpenCV Foundation. The library has been used in a wide range of applications including robotics, surveillance, medical imaging, augmented reality, and more.

The library provides over 2500 optimized algorithms that can be used for various computer vision tasks such as object detection, facial recognition, image



segmentation, camera calibration, and more. The algorithms are designed to be fast and efficient, making it suitable for real-time applications.

One of the core strengths of OpenCV is its support for multiple programming languages, including C++, Python, Java, and MATLAB. This allows developers to use the language they are most comfortable with and integrate OpenCV into their existing workflows. Additionally, OpenCV has a large and active community of users and developers who contribute to the development and maintenance of the library.

OpenCV provides a variety of image processing functions, including color conversion, filtering, geometric transformations, and more. It also provides advanced functions such as edge detection, feature detection, and machine learning-based object detection. These functions can be combined to create complex image processing pipelines that can analyze and understand images and videos.

OpenCV also provides support for working with video streams, including real-time video capture and video playback. This allows developers to create applications that can process and analyze video streams in real-time. The library also provides support for working with multiple cameras and video formats.

Another strength of OpenCV is its support for computer vision hardware, such as depth cameras, stereo cameras, and GPU acceleration. This allows developers to take advantage of specialized hardware to improve the performance of their computer vision applications.

In conclusion, OpenCV is a powerful and widely used library for computer vision and image processing. Its extensive set of functions, support for multiple

OpenCV (Open Source Computer Vision Library) is a powerful and popular open-source library for computer vision and machine learning applications. It provides a wide range of functions and tools for image and video processing, including image recognition, segmentation, feature detection, object tracking, and optical character recognition (OCR).

OCR is a technique for recognizing and extracting text from images or scanned documents. OpenCV can be used for OCR in various ways, including:

1. **Preprocessing:** OpenCV provides various image preprocessing techniques such as thresholding, blurring, and morphological operations. These techniques can help to improve the quality of the input image and make it easier to recognize text.
2. **Feature Extraction:** OpenCV can be used to extract features such as edges, lines, and contours from images. These features can be used to detect text regions and separate them from the background.
3. **Text Detection:** OpenCV provides algorithms for text detection, such as the Stroke Width Transform (SWT) and the Maximally Stable Extremal Regions (MSER) algorithm. These algorithms can be used to locate and extract text regions from images.
4. **Text Recognition:** Once the text regions are extracted, OpenCV can be used to recognize the text. OpenCV does not provide a built-in OCR engine, but it can be integrated with other OCR engines such as Tesseract OCR or Google OCR.

OpenCV can also be used for real-time OCR applications, such as reading text from a camera feed or scanning documents. Real-time OCR requires fast and efficient algorithms, and OpenCV provides optimized implementations for many image processing tasks.

In conclusion, OpenCV is a valuable tool for OCR applications as it provides various functions and tools for image processing and feature extraction. It can be used for preprocessing, feature extraction, text detection, and text recognition. OpenCV can also be used for real-time OCR applications. By combining OpenCV with OCR engines, developers can build powerful OCR applications with high accuracy and speed.

programming languages, and active community of developers make it a valuable tool for researchers, developers, and hobbyists working in the field of computer vision

- Word validations occur predominantly as the pre-processor will get the image from OpenCV and infer the model's run for the specified. This will also check the error ratio and rate for the further simplification and improvement of the model in the future. An inferring function is run alongside the cv2 module to convert the given image in a recognizable manner for the machine (grey-scale) and give the recognized output with the probability percentage of the given phrase's success in the model.

With all the modules mentioned above executed, the Language Recognition project must be developed perfectly. The Flask should be integrated correctly with the model and website developed as well. However, there are the final set of integrations, file inclusions and OS allowances that must be kept in mind and executed before running the project in order to connect the project's loose ends:

- The *model.py* and *htrwork.py* must definitely be imported into the flask's python application (*app.py*) in order to integrate the model and website and encapsulate these two individual entities as a whole and working project.
- All the python applications are made sure to be kept in the main file of the directory and not in any subdirectories to avoid any functionality errors in the website or models' working.
- The files of inputs (image) and outputs (prediction voice note) are made to be saved at the "files" sub-directory of the project. But as mentioned before, anybody willing to develop a database for the project in the future can be enabled to integrate the same using the code of input and database management found in *preprocessor.py* and *model.py* properly.
- The camera input is currently integrated using the OpenCV module and the model is efficient to predict the text from the image in a webcam, however with less accuracy due to the picture with pixelization and less resolution obtained from a usual webcam. Consider using the webcam in a well-lit environment with the manuscript shown well. As an Android/Apple application, the project may thrive well with volatile and

immediate camera-based inputs.

If the prediction is found to have any sort of demeaned or inaccurate prediction, please make sure to enable the default word decoder to be “WordBeamSearch” for more accuracy, or try developing the model’s accuracy by training the model with new dataset either user-made or third party accessed

## CHAPTER 5

### PROJECT DESIGN AND IMPLEMENTATION

#### 5.1 Development and Deployment Setup

In recent times, many new inventions have been made in order to help people with a certain set of disabilities/disorders. We see the development of speech-to-neural transmitter (earpiece) for the deaf, VR-enabled course looker (yet to be developed) for the physically disabled, and so on. One of the main problems that the blind people have to tackle in their daily life is to convert any piece of information that is non-Braille in nature. Since everyone isn't very keen in the aspect of sharing their piece of information in the usable method for the disabled, we seek to find a method that disassembles the correct way of bringing about a valid piece of technology to help this part of community.

HTRWork.py	Executes all the relevant .py programs together and acts as a link to flask Ui module
SamplePreprocessor.py	The data is prepared from the help of neural network. Which is accessed by dataset
DataLoader.py	Reads all the pre-processed sample given and segregates most common data
defT2SConvert()	The text module is converted to give as an output in the model.py to an automated speech

**TABLE 5.1 All the modules used in the code.**

This search led us to learning and use the knowledge of Deep Learning, ML, and UI/UX, to develop an interface that allows you to bring forth a piece of information (currently in English), either written or typed, to the program as an input, and in return, the machine gives out a voice note of the very information that has been entered, along with numerous help for the set of information provided. Since Machine Learning has its niche in a spectrum of domains and researches, many approaches have been partially or totally established. Bayesian Network, Clustering, Decision Tree Learning and many more have collectively made Deep Learning only a part of the approaches.

The following review mainly focuses on the collated usage of deep learning, the basics of forming this proposed application in said and different fields. Additionally, it presents several figures portraying the viable usage of this project across many nuances. The paper that helped the most to bring this project to its fullest potential was the “Object Recognition Development for Android Mobile Devices with Text-to-Speech Function Created for Visually Impaired People” developed by Andrei Burta, Roland Szabo, and Aurel Gontean. The paper was issued in accordance of developing a speech application that could scan the given picture’s input and speak out the name/description of that product.

This project was implemented in order to help the old-aged people with identifying some of the essential products while purchasing them in markets. The project basically ran under the guidance of Convolutional NNs and Recurrent NNs. So, we learnt the utmost capabilities of CNN and RNN following the understandings of integrating multi-substantial layers to the model, which was later integrated to a rich and intuitive website UI to be used by people for test.

OCR (Optical Character Recognition) can be implemented via machine learning by training models to recognize text patterns from images or documents. Machine learning algorithms enable computers to learn from data and make predictions or decisions based on that learning. OCR is an application of machine learning that involves training models to recognize characters and words from images or scanned documents.

The implementation of OCR via machine learning involves several steps:

1. Data collection and preprocessing

The first step is to collect a dataset of images or scanned documents that contain the characters or words to be recognized. The data needs to be preprocessed to remove any noise, skew, or distortion from the images.

2. Training the model

The next step is to train a machine learning model using the preprocessed data. The model can be trained using supervised or unsupervised learning methods. In supervised learning, the model is trained using labeled data, where each image or document is labeled with the corresponding text. In unsupervised learning, the model is trained on unlabeled data, where the model learns to identify patterns and similarities in the data.

### 3. Feature extraction and selection

After training the model, the next step is to extract and select relevant features from the images or documents. The feature extraction process involves identifying important parts of the image or document that are relevant for character or word recognition. The selected features are used to create a feature vector that represents the image or document.

### 4. Classification and prediction

Once the feature vector is created, the model can be used to classify or predict the characters or words in the image or document. The model makes predictions by comparing the feature vector with the ones it learned during the training phase. The output of the model is the recognized text from the image or document.

### 5. Evaluation and optimization

The final step is to evaluate the performance of the OCR system and optimize the model to improve its accuracy. The OCR system can be evaluated using metrics such as precision, recall, and F1-score. The model can be optimized by adjusting hyperparameters, selecting different algorithms, or using more data.

In conclusion, implementing OCR via machine learning involves collecting and preprocessing data, training a model, extracting and selecting relevant features, classifying and predicting the text, and evaluating and optimizing the system.

## CHAPTER 6

### RESULTS AND DISCUSSION

#### 6.1. RESULTS OBTAINED

The project had been implemented and developed perfectly, which resulted in the execution of the application successfully. Following the implementation of the python modules and the learning model, the website is then integrated with flask and the project is run, as we have seen.

The “Home” screen of the website as we have seen earlier, gets navigated to the “Implement Script” page, we find the options for giving the input via the file upload scheme. Upon selecting the input and uploading, the website then uploads that to the model.

The model predicts the text from the image and later loads to a new page where the following output comes up (cmd output given for now). Here we show the model’s input and output format followed by the website’s input and output format:

By increasing the size of the input of the NN layer, if the layer is large enough, we could enable the text-lines completely. We confined the output words to be a proper word from the dictionary is permitted, and if the dictionary doesn’t seem to find the predicted word from the input, then a text correction is done by searching for the most similar one, and the cursive writing style was removed from the input images, or classified as broken images to conclude the input invalid for conversion.

A photograph of a piece of paper with cursive handwriting. The text is written in a fluid, connected script. The words are difficult to decipher due to the cursive style, but appear to be "He fore friend of the family, like the".

**Fig 6.1. Cursive Writing that is removed from classification**



Looking at the initial stages of the model's training aspects, the training data – testing data was split in the ratio of 4:1 (i.e., 80% - 20%). The model was trained by the mean of the loss values of the batch elements.

With regards to analyzing the preloaded dataset (IAM Sentences version and some self-made dataset), the images of various pre-written sentences is linked and trained along with their actual phrase in the ASCII (or word) format.

In the middle of the training process, some broken images are also added to identify the potential outliers in the input of such were present. During the Training Phase, the model is set to find a stable accuracy rate with every count batch it analyses, and the completion of a standard 25 epochs (batch analysis) of the whole training dataset, the accuracy increases up to about 78%.

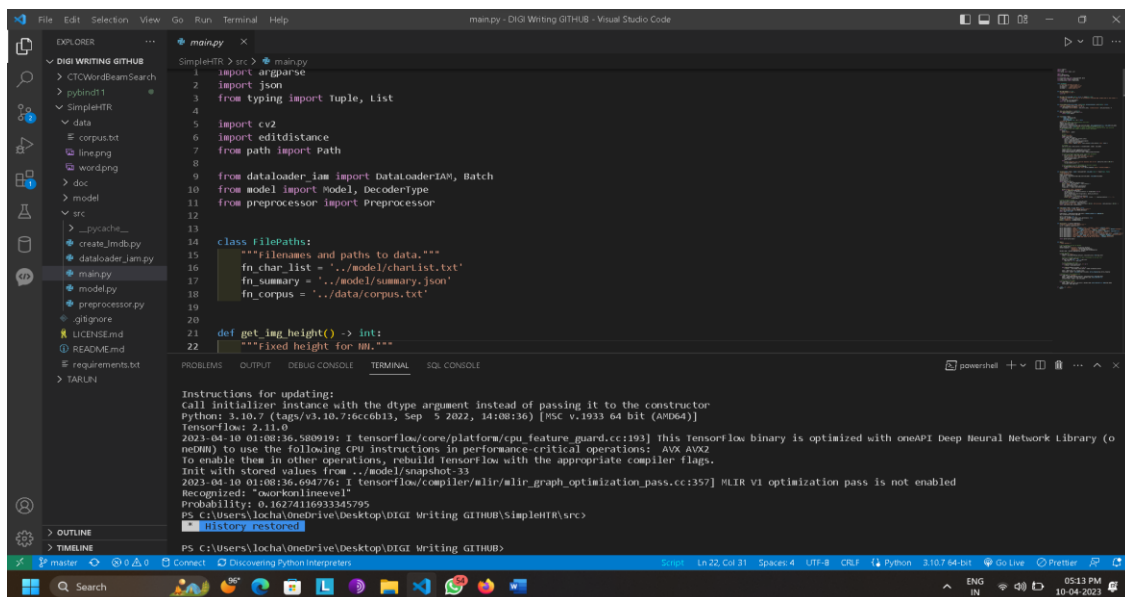
Words	Accuracy
Word	0.90
School	0.82
Battery	0.80
Today	0.85
Going	0.75
Final	0.81
Everybody	0.70
Using	0.83
Urgent	0.89
Waiting	0.72

**TABLE 6.2. TABLE DEPICTING THE ACCURACY PERCENTAGE OF THE MODEL'S TRAINING-TESTING SEQUENCE**

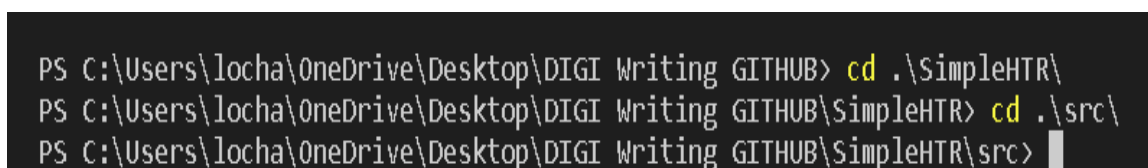
The missed-out features of user account formation and database facilities could possibly be the major improvements to be implemented in the future works. As said earlier, this project is capable of being trained with different language's datasets that could enable the model to predict different languages' texts and convert them to the voice of that language, hence there is not much feature editing needed.

This paper has been possible by learning the works of the output of the project developed by Andrei Burta, Roland Szabo, and Aurel Gontean. In their paper discussed before, their sole purpose was to enact the task of captioning the shown product to the camera to its nomenclature (name) for the user to know.

We had taken the initiative to have changed the approach of reading a product, to reading a script. In the end, the aim from both the ends were to serve the people with either visual impairment or lack of literal knowledge would be benefited by this project



**Fig 6.3. The Visual Studio Code IDE (VSC) containing the project code**



**Fig 6.4. The Visual Studio Code IDE (VSC) terminal stated to the main directory**

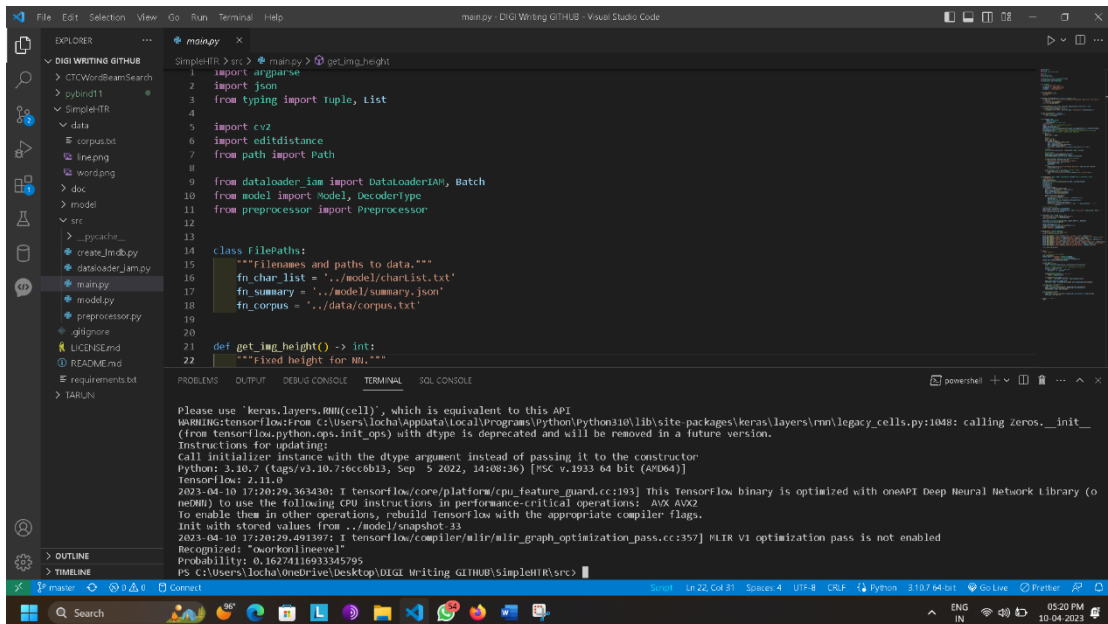
```

19
20
21 def get_img_height() -> int:
22     """Fixed height for MNIST"""

Please use 'keras.layers.RNN(cell)', which is equivalent to this API
WARNING:tensorflow:From C:\Users\locha\AppData\Local\Programs\Python\Python310\lib\site-packages\keras\layers\rnn\legacy_cells.py:1048: calling Zeros.__init__
(from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Python: 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)]
TensorFlow: 2.11.0
2023-04-10 17:17:35.639355: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (o
neDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Init with stored values from ../model/snapshot-33
2023-04-10 17:17:35.760156: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:357] MLIR V1 optimization pass is not enabled
Recognized: "word"
Probability: 0.9513834118843079
PS C:\Users\locha\OneDrive\Desktop\DIGL Writing GITHUB\SimpleHTR\src>

```

**Fig 6.5. Running main.py open the project website**



**Fig 6.6. Running main.py with another input open the project website**



```
if args.mode == 'train':

    loader = DataLoaderIAM(args.data_dir, args.batch_size, fast=args.fast)

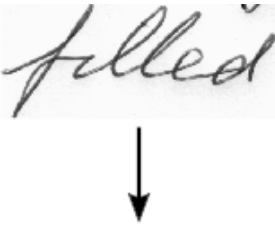
    # when in line mode, take care to have a whitespace in the char list

    char_list = loader.char_list

    if args.line_mode and ' ' not in char_list:

        char_list = [' '] + char_list
```

So as we can see the above code is where the training inputs and the whitespace is taken care of and the that whitespace should be taken care of again while giving the output



A handwritten word "filled" in cursive script is shown at the top. A black arrow points downwards from the word to a table below.

Best path decoding	"fuleid" ❌
Vanilla beam search	"fuleid" ❌
Word beam search	"filled" ✅

**Fig6.10 DECODER COMPARING**

## **CHAPTER 7**

### **CONCLUSION**

#### **7.1 Conclusion**

We discussed about a Neural Network that was able to recognize text in images, whether handwritten or typed, and converted the text format to a voice note. A Python (.py) implementation using TensorFlow (TF) is provided, along with their roles in the project. Some important parts of the model were introduced separately and inspected for their functions. As different entities, they worked out to be a consistent application with the help of different modules.

The final NN model had consisted of 5 Convolutional NN and 2 Recurrent NN layers, which outputs a character-probability matrix. This matrix is either used as a throughput to the model improvement techniques, that were displayed to improve the recognition accuracy of the shown developed model.

Overall, the project was executed successfully with the algorithm of development performed rightly, and the end model of the project was entertained well with the beta users, guides, and the internal examiners.

#### **7.2 FUTURE ENHANCEMENTS**

For improving the recognition accuracy of the model, we followed some of the common regimens – we enabled Data augmentation to be used as to expand the size of the dataset by enabling non-sequential input transformation of the images via Regeneration and Duplication.

## REFERENCES

- [1] Aurélien Géron (2019). Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Second Edition. *Pages 324-481*
- [2] Hisham, A., Harin, S. (2017). Deep Learning – the new kid in Artificial Intelligence. *Pages 97-210*
- [3] Jeffrey Zeldman (2009). Designing with Web Standards (3rd Edition). *Pages 129-305*
- [4] Robin Nixon (2014). Learning PHP, MySQL, JavaScript, CSS & HTML5: A Step-by-Step Guide to Creating Dynamic Websites. *Full Book*
- [5] Dali Zhu, Long Yang, Zhanxun Liyz, Hualin Zeng, Chinese Academy of Sciences, Beijing, China. Remote Speech Extraction from Speckle Image by Convolutional Neural Network
- [6] K. Sasindu Anuradha. and Samantha Thelijagoda. (2020). Sri Lanka Institute of Information Technology, Sri Lanka. Machine translation system to convert Sinhala and English Braille documents into voice.
- [7] K. Sakthidasan, K.Kavitha, S. Haritha Priya. Content Based Image Retrieval Process for Speech Annotated Digital Images.
- [8] Albawi, Saad; Mohammed, Tareq Abed; Al-Zawi, Saad (2017). [IEEE 2017 International Conference on Engineering and Technology (ICET) - Antalya, Turkey (2017.8.21-2017.8.23)] 2017 International Conference on Engineering and Technology (ICET) – “Understanding of a convolutional neural network.”, ppt.1–6.
- [9] Bourbakis, Nikolaos (2008). [IEEE 2008 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI) - Dayton, OH, USA (2008.11.3-2008.11.5)] 2008 20th IEEE International Conference on Tools with Artificial Intelligence - Automatic Image-to-Text-to-Voice Conversion for Interactively Locating Objects in Home Environments.



## **APPENDIX**

### **A. SOURCE CODE(Main.py)**

```
import argparse

import json

from typing import Tuple, List


import cv2

import editdistance

from path import Path


from dataloader_iam import DataLoaderIAM, Batch

from model import Model, DecoderType

from preprocessor import Preprocessor


class FilePaths:

    """Filenames and paths to data."""

    fn_char_list = '../model/charList.txt'

    fn_summary = '../model/summary.json'

    fn_corpus = '../data/corpus.txt'


def get_img_height() -> int:
```

```
"""Fixed height for NN."""
```

```
return 32
```

```
def get_img_size(line_mode: bool = False) -> Tuple[int, int]:
```

```
    """Height is fixed for NN, width is set according to training mode (single words  
    or text lines)."""
```

```
    if line_mode:
```

```
        return 256, get_img_height()
```

```
    return 128, get_img_height()
```

```
def write_summary(char_error_rates: List[float], word_accuracies: List[float]) ->  
    None:
```

```
    """Writes training summary file for NN."""
```

```
    with open(FilePaths.fn_summary, 'w') as f:
```

```
        json.dump({'charErrorRates': char_error_rates, 'wordAccuracies':  
word_accuracies}, f)
```

```
def char_list_from_file() -> List[str]:
```

```
    with open(FilePaths.fn_char_list) as f:
```

```
        return list(f.read())
```

```
def train(model: Model,
```

```
        loader: DataLoaderIAM,
```

```

    line_mode: bool,

    early_stopping: int = 25) -> None:

    """Trains NN."""

    epoch = 0 # number of training epochs since start

    summary_char_error_rates = []

    summary_word_accuracies = []

    preprocessor = Preprocessor(get_img_size(line_mode), data_augmentation=True,
                                line_mode=line_mode)

    best_char_error_rate = float('inf') # best validation character error rate

    no_improvement_since = 0 # number of epochs no improvement of character
    error rate occurred

    # stop training after this number of epochs without improvement

    while True:

        epoch += 1

        print('Epoch:', epoch)

        # train

        print('Train NN')

        loader.train_set()

        while loader.has_next():

            iter_info = loader.get_iterator_info()

            batch = loader.get_next()

            batch = preprocessor.process_batch(batch)

            loss = model.train_batch(batch)

            print(f'Epoch: {epoch} Batch: {iter_info[0]}/{iter_info[1]} Loss: {loss}')

```

```

# validate

char_error_rate, word_accuracy = validate(model, loader, line_mode)


# write summary

summary_char_error_rates.append(char_error_rate)

summary_word_accuracies.append(word_accuracy)

write_summary(summary_char_error_rates, summary_word_accuracies)


# if best validation accuracy so far, save model parameters

if char_error_rate < best_char_error_rate:

    print('Character error rate improved, save model')

    best_char_error_rate = char_error_rate

    no_improvement_since = 0

    model.save()

else:

    print(f'Character error rate not improved, best so far: {char_error_rate * 100.0}%')

    no_improvement_since += 1


# stop training if no more improvement in the last x epochs

if no_improvement_since >= early_stopping:

    print(f'No more improvement since {early_stopping} epochs. Training stopped.')

    break

```

```

def validate(model: Model, loader: DataLoaderIAM, line_mode: bool) ->
    Tuple[float, float]:

    """Validates NN."""

    print('Validate NN')

    loader.validation_set()

    preprocessor = Preprocessor(get_img_size(line_mode), line_mode=line_mode)

    num_char_err = 0

    num_char_total = 0

    num_word_ok = 0

    num_word_total = 0

    while loader.has_next():

        iter_info = loader.get_iterator_info()

        print(f'Batch: {iter_info[0]} / {iter_info[1]}')

        batch = loader.get_next()

        batch = preprocessor.process_batch(batch)

        recognized, _ = model.infer_batch(batch)

        print('Ground truth -> Recognized')

        for i in range(len(recognized)):

            num_word_ok += 1 if batch.gt_texts[i] == recognized[i] else 0

            num_word_total += 1

            dist = editdistance.eval(recognized[i], batch.gt_texts[i])

            num_char_err += dist

            num_char_total += len(batch.gt_texts[i])

        print('[OK]' if dist == 0 else '[ERR:%d]' % dist, '' + batch.gt_texts[i] + '',
            '->',

```

```

        ''' + recognized[i] + ''')

# print validation result

char_error_rate = num_char_err / num_char_total

word_accuracy = num_word_ok / num_word_total

print(f'Character error rate: {char_error_rate * 100.0}%. Word accuracy:
      {word_accuracy * 100.0}%.')

return char_error_rate, word_accuracy


def infer(model: Model, fn_img: Path) -> None:

    """Recognizes text in image provided by file path."""

    img = cv2.imread(fn_img, cv2.IMREAD_GRAYSCALE)

    assert img is not None

    preprocessor = Preprocessor(get_img_size(), dynamic_width=True, padding=16)

    img = preprocessor.process_img(img)

    batch = Batch([img], None, 1)

    recognized, probability = model.infer_batch(batch, True)

    print(f'Recognized: "{recognized[0]}"')

    print(f'Probability: {probability[0]}')


def parse_args() -> argparse.Namespace:

    """Parses arguments from the command line."""

```

```

parser = argparse.ArgumentParser()

parser.add_argument('--mode', choices=['train', 'validate', 'infer'],
                    default='infer')

parser.add_argument('--decoder', choices=['bestpath', 'beamsearch',
                    'wordbeamsearch'], default='bestpath')

parser.add_argument('--batch_size', help='Batch size.', type=int, default=100)

parser.add_argument('--data_dir', help='Directory containing IAM dataset.',
                    type=Path, required=False)

parser.add_argument('--fast', help='Load samples from LMDB.',
                    action='store_true')

parser.add_argument('--line_mode', help='Train to read text lines instead of
                    single words.', action='store_true')

parser.add_argument('--img_file', help='Image used for inference.', type=Path,
                    default='../data/word.png')

parser.add_argument('--early_stopping', help='Early stopping epochs.', type=int,
                    default=25)

parser.add_argument('--dump', help='Dump output of NN to CSV file(s).',
                    action='store_true')

return parser.parse_args()

```

```

def main():

    """Main function."""

    # parse arguments and set CTC decoder

    args = parse_args()

    decoder_mapping = {'bestpath': DecoderType.BestPath,

```

```

        'beamsearch': DecoderType.BeamSearch,

        'wordbeamsearch': DecoderType.WordBeamSearch}

decoder_type = decoder_mapping[args.decoder]

# train the model

if args.mode == 'train':

    loader = DataLoaderIAM(args.data_dir, args.batch_size, fast=args.fast)

    # when in line mode, take care to have a whitespace in the char list

    char_list = loader.char_list

    if args.line_mode and ' ' not in char_list:

        char_list = [' '] + char_list

    # save characters and words

    with open(FilePaths.fn_char_list, 'w') as f:

        f.write(''.join(char_list))

    with open(FilePaths.fn_corpus, 'w') as f:

        f.write(' '.join(loader.train_words + loader.validation_words))

    model = Model(char_list, decoder_type)

    train(model, loader, line_mode=args.line_mode,
          early_stopping=args.early_stopping)

# evaluate it on the validation set

elif args.mode == 'validate':

```



```

loader = DataLoaderIAM(args.data_dir, args.batch_size, fast=args.fast)

model = Model(char_list_from_file(), decoder_type, must_restore=True)

validate(model, loader, args.line_mode)


# infer text on test image

elif args.mode == 'infer':

    model = Model(char_list_from_file(), decoder_type, must_restore=True,
dump=args.dump)

    infer(model, args.img_file)


if __name__ == '__main__':

    main()

```

## B.SCREENSHOTS

```
Call initializer instance with the dtype argument instead of passing it to the constructor
Python: 3.8.10 (tags/v3.8.10:3d8993a, May 3 2021, 11:48:03) [MSC v.1928 64 bit (AMD64)]
tensorflow: 2.5.0
2022-04-07 12:59:57.513619: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Init with stored values from ../model/snapshot-13
*
*
*
*
*
Welcome to the Language Recognition Project. Your image's audio result will arrive shortly!

L
D
A
D
I
V
S
T
T

Recognized: "or work on line level"
Probability: 66.7436752319336
Do you want me to repeat the phrase? Press Y/N
no
Thank you for using our project! Come back soon!
```

```
19
20
21 def get_img_height() -> int:
22     """Fixed height for MNIST"""

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE powershell +v [ ] ... ^ X

Please use 'keras.layers.RNN(cell)', which is equivalent to this API
WARNING:tensorflow:From C:\Users\locha\AppData\Local\Programs\Python\Python310\lib\site-packages\keras\layers\rnn\legacy_cells.py:1048: calling Zeros.__init__
(from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Python: 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)]
tensorflow: 2.11.0
2023-04-10 17:17:35.639355: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (o
neDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Init with stored values from ../model/snapshot-33
2023-04-10 17:17:35.760156: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:357] MLIR V1 optimization pass is not enabled
Recognized: "word"
Probability: 0.9513834118843079
PS C:\Users\locha\OneDrive\Desktop\DIGI Writing GITHUB\SimpleHTR\src> |
```

# C.RESEARCH PAPER

## DIGI WRITING

Enugula Lochan  
Department of Computer Science and  
Engineering,  
Sathyabama Institute of Science and  
Technology,  
Chennai, India  
lochan2e@gmail.com

Pattigulla Sai Pradeep  
Department of Computer Science and Engineering,  
Sathyabama Institute of Science and Technology,  
Chennai, India  
gouthamsaipradeep@gmail.com

Dr. P Asha  
Department of Computer Science and Engineering,  
Sathyabama Institute of Science and Technology,  
Chennai, India  
Asha\_cse@sathyabama.ac.in

**Abstract** — There has been a lot of progress in the development of machine learning for us to develop the understandings of the machinery mind. As humans, we learn how to do a task by doing it, and we optimize the tasks by learning from our mistakes. Machines can comprehend situations of strengthening developed neurons in the same way that neurons in the brain automatically trigger and quickly perform learned tasks. Deep learning is as intriguing as the concept of the human brain. The use of various types of architectures for such neural networks collides for various types of problems, such as image and sound classification, object recognition, image segmentation, object detection, and so on. Following these layers of different commemorations and accuracy that Artificial Intelligence provides, many unsolved issues can be solved here by one machine, problems, and tasks, because the machine is now capable of approaching the task as if it's thought process is as same as that of a human brain with a precise accuracy. As we all know that in today's world it's all about machine learning and deep learning in the market which is the reason for their applications in this field is never ending. Changes can be done by taking tasks to lives as a challenges, ailments, and difficulties to the very fabrics that will be having very balanced result of the task under the supervision of the trained machine.

**Keywords** – *Neural Networks, Image and Sound Classification, Object Recognition, Image Segmentation, Object Detection*

### 1. INTRODUCTION

Machine learning has always been aggressive in the digital market and the approach towards the task with a logic. Here in machine learning we use algorithms and statistical models and help of improve the efficiency of the machine for better results and improve lives. The significant impact is in healthcare, Transportation, Finance, agriculture, Retail, Manufacturing, Energy last but not least Natural Language Processing (NLP), which is the main component of our project where it helps us in converting the data given by the user and improve it by interactive with it as a speech recognition as well as text-to speech and finally language recognition. Here text-processing involves cleaning and preparing the data from the user and remove all the punctuation and stop words which are common words that will not add any meaning to the sentence and then part tagging were it identifies parts of speech and use to understand the model input individually and then syntax parsing were grammatical structure of the text is corrected and determining the sentiment by sentiment analysis and after that the nouns and entities will be located throughout the input given by the user and then role labeling where it identifies the role of the words in the sentence and describe it and then text summarization where the input will shrink but the meaning to it will be labeled to it and then the last part where with the help of machine algorithms the machine will generate similar text as the user has requested

### 2. LITERATURE SURVEY

Optical Character Recognition (ORC) is a technology used to identify written text characters and printed in images even scanned which will later be converted into machine readable language. This is very advanced technology which has been used widely in numerous fields.

K.R Gopinath and N.Venkateswaran [1] classification, feature extraction, segmentation, preprocessing, and the related survey is been provided and also the author discuss the challenges and future scope related to the optical character recognition

Hualin Zeng Zhanxun [2] Using the function of modulated CNN (convolutional neural network) remote speech extraction from the input speckle image which was contributed to IEEE Symposium on computers and Communications (ISCC Rennes 2020)

Yuncheng Zhou (2019) [3] With the combined help of Convolutional Neural Network and Internet of things the author executed the idea of medical motion image recognition

A.Antonacopoulos and D. Karatzas (2010) [4] The author will talk about all the retrieval techniques which also includes image document, text recognition and also the information retrieval

Yulia S. Chernyshova, Alexander V. Sheshkus and Vladimir V. Arlazarov With the help of different camera via IEEE the two step text line Recognition framework is achieved which serves the purpose

Tahir, Ghalib Ahmed; Loo, Chu Kiong (2020) contributed for an open-ended Continual Learning (CL) model that is regarded on food recognition through extreme learning Machines using the increasing of class via IEEE Access.

### 3. METHODOLOGY

Here in this machine, there are various parameters that will be taken under consideration while built it and the working machine model to analyze these various inputs in the form of image, audios, or videos. With the help of Udemy courses such as "MACHINE LEARNING A-Z: HANDS OF PYTHON AND R DATA SCIENCE" we will be learning every module of the machine learning that could ever be developed and with few IEEE research paper we can easily use it for reference to build a high accuracy model.

- There will be a wide range of module with-in it such as machine translation, which is commonly known as language-united translations. Machine translation algorithms defined a major number of spikes in a variety of applications where spelling and grammatical errors were has to be rectified thoroughly. Use of a different language or script as a complete which is entirely up to the user/developer tasked with developing the language processing model.

#### a. The Aim behind this Project

Lately, many new inventions have been introduced for people with certain disabilities/disorders. We are witnessing the development of a neural (ear) speech transmitter for deaf observers with VR support for people with physical disabilities and who are suffering on a day-to-day basis. Blind people face so many problems daily and one of the main is the conversion of information that is not in Braille. Because not everyone is interested in sharing their part Information in a method that is useful for people with disabilities, we will be looking towards a method that hides the right way to introduce something to help all the disability people and do something good for this part of the community as to make them feel same as everybody. Here deep learning, Machine learning as well as UI/UX and all the uses of it to develop an interface that makes it possible for people to chance to communicate with other people just as a normal human being is doing in his day to day live where the device will be giving the output of a voice as information.

HTRWork.py	Executes all the revelent .py programs together and acts as a link to flask Ui module
SamplePreprocessor.py	The data is prepared from the help of neural network. Which is accessed by dataset
DataLoader.py	Reads all the pre-processed sample given and segregates most common data
deft2SConvert()	The text module is converted to give as an output in the model.py to an automated speech

#### b. Project and work related about it

"ANDROID MOBILE OBJECT RECOGNITION WITH TEXT TO SPEECH FOR THE VISUALLY IMPARED" is a article published by Andrei burt. While this article will help us in development in speech applications that will scan the input which is basically a a specific image and will give a audio output of the given input. The aim of this paper and Digi writing is the same where the main productive outcome is identifying most items from the shopping mall/Markets .Convolutional NN (CNN) and RNN are the basic essential element to develop this machine after understanding it's working modules and integrating multiple layers to the model which will help in usage of the machine and will be very effective and also help full for the owner to test the model and get any future updates

#### c. Methods and the UI/UX

Successful model can be achieved with the help of - Convolutional Neural Network (Which are the code model of Machine learning), which has always been called the neural network model, the suitable for management and audio, image and video clip processing, an implementation of Recurrent in Long-Term Memory (LSTM)The neural network (RNN) has been shown to be able to assemble and transmit information over a larger data layer range with better data training, and the connectionist temporal classification (CTC) is combined with RNN to calculate the loss values. CTC interference is shown as table scan activityRecord and find the appropriate data node for the specific data to be tested and return the final text result. CNN is considered an important implementation of most pattern recognition, up to Speech recognition for image processing. CNNs have the additional benefit of handling the number of

parameters of artificial neural networks (ANNs). Classic ANN output conversion port error was Therefore, advances in the use and dissemination of models via CNNs have been greatly appreciated. Abstracting functionality was another commendable tool from CNN, allowing for deep layers in passing input. Let's say for image classification, the first group of layers detects image edges, then the second layer analyzes easy shapes, then n-case layers for n high-level text features, like in our case i misinterpreted texts.

#### d. Implementation of the Model

As the principal block, there is a organization named Brain is constructed that is totally prepared on pictures where it consist of words and sentences and the dataset is taken form the database called IAM. Saving by this information decently little for the word/line pictures, here after the NN will be preparing the model with the underlying layers and making sure to attain the accuracy These are the underlying moves toward fabricate a base (Brain Organization) with the help of the layers we can change to attain the desired output.

We utilize a Brain Organization for fostering a model for our errand. The fundamental necessary layers for this situation are as per the following: Convolutional NN layer for portraying the word-line picture, Intermittent NN for the element scaling, and a last layer of CTC (Connectionist Transient Characterization) for length and significance space.

The portrayal of the Brain Organization is additionally double in a more proper manner as a utilitarian condition, planning an exact picture M (considering a network) with a length among 0 and L. The need of ordering the words on a person level is conceivable to improve the model's general reason for expanding its precision on the "words and sentences" plan of things. Consequently,

#### e. Tools/Data Required

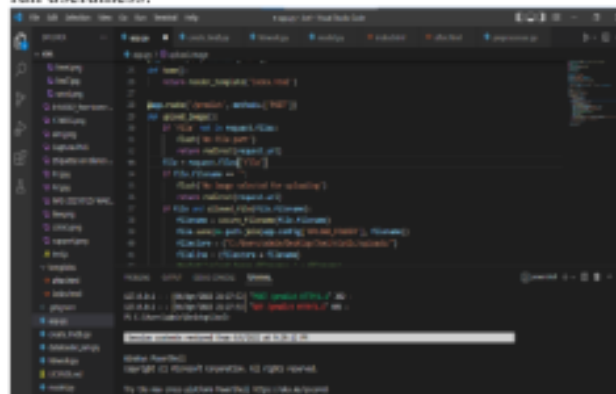
- "Information" is plural, not solitary The right streaming of the venture requires the accompanying arrangement of establishments and executions of the latest version at the time Python, with the latest version of TensorFlow, and the same latest version at the time of NumPy and OpenCV.
- IDE like PyCharm can be used for this situation as it a solid open-source IDE for all the process for this project and PyCharm is a python exclusive platform
- To foster the connection point site that gathers the information from client and gives the result in type of a voice note, we've utilized the administrations of carrying out system utilizing Bootstrap, and making the intelligent UI utilizing HTML and CSS, with the usefulness strategies of Js.
- Up to this point, we've considered either making the model with only a one-time use or obliterate point of interaction that utilizes the information just a single time to recreate the result instead of saving the entire cycle. If the last option is in the need, of execution, we utilize either AWS or Firebase for the production of an ostensible data set.

#### f. Implementation

With the execution of the Python modules in this venture, we use around 5 modules that are doled out to the different errands of the forecast model and text-to-discourse change. Here we have "EXAMPLE PRE-PROCESSES" module which will pre-process the input with the help of the input algorithm . Which is trailed by "information loader" module that peruses the got tests and loads them in groups to make a kind of iterator-connection point to peruse the entire arrangement of tests, and the real "model" that predicts the word/sentence from the given information utilizing the examples'

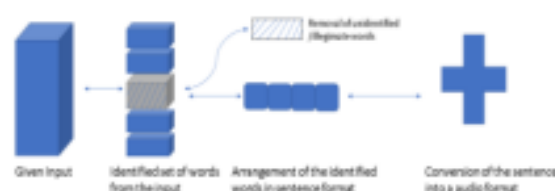


example. The "T2SConvert" module changes the anticipated sentences from the contribution over completely to a robotized voice yield for the client as the eventual outcome. This large number of modules are typified into a "primary" module for the consistent working of all modules as single program. The handled result line is then sent into the following capability that changes over the given result message state into a computerized voice notes for the client to pay attention to and assemble the data. This finishes up the flowchart of the undertaking and characterizes its full usefulness.



- CNN, RNN, CTC and LSTM are used in calculations utilized in the task are exclusively founded on the models of. The stream goes in the way of getting the information from the dataset called IAM will be used in the model/machine. Following that, we utilize the pre-handling techniques to group, misfortune take and burden the information into the model. The model likewise has a person mistake check module which will constantly work on the model's precision as the blunder is gotten throughout the function of the model which will help the module mental soundness in the event that the model has anticipated the information pretty much right.
- Word approvals happen transcendentally as the pre-processor Open CV will present us with the input picture deduce the rune of the model for the predefined. As the model will likewise actually look at the blunder proportion and rate for the further disentanglement and improvement of the model from now on. A surmising capability is run close by the cv2 module to change over the given picture in an unmistakable way for the machine (dark scale) and give the perceived result with the likelihood level of the given expression's outcome in the model.
- The handled result line is then sent into the following capability that changes over the given result message express into a mechanized voice note for the client to pay attention to and accumulate the data. This closes the flowchart of the task and characterizes its full usefulness.

#### g. Working



- As talked about before, the utilization of CNN and RNN (in type of LSTM) is the essential important job of this undertaking. To assemble the precision of the model and to forestall overfitting, we have utilized the elements of Component Scaling, Ensembling, k-cresce Cross Approval and Regeneralization.



- When every we get a dark or low scale picture. It is normally either a greatest dim scale picture or a moderate Black/white, which will have a size ("128x32") as of this type. The model will find couple of pictures form the IAM database and will take the result with is most precise and has with high accuracy, thus the pictures are resized which happens without mutilation. Now the picture will subsequently replicate to an objective picture of a similar determined size as long as the picture has been scaled only with the references between of (128 width) or" (32 level)". At last, the standardization of the dim upsides of the picture, which is to be performed, to improve on the Brain Organization's exhibition.
- Here CNN, these CNN modules will be fixing the result at the grouping of 32 with the help of rate length. On a total there are 256 elements for each of which they are being handled moreover with the help of these layers from the RNN we can get the output. There are so many complications while making this model as for a example the letter "e" and "l" share a lot of CNN and RNN square entities when it comes identification of the input. To overcome this problem, we will be using CTC which will leave extent for manually written words like letter E and I.

#### h. Analysis of the model

Taking a gander with an underlying phase that the module has preparation viewpoints, the preparation information - testing information was parted in the proportion of 4:1 (i.e., 80% - 20%). The model was prepared through the misfortune upsides of the cluster components. Concerning examining form the dataset we used as mentioned before it is taken from the dataset called IAM, from this dataset the, pictures from different composed sentence will be connected as well as prepared alongside from their genuine expression taken from the ASCII or from the words that are design. In the preparation cycle, a few broken pictures are likewise added to distinguish the possible exceptions that has been contributed which

are available. In the process of the Preparation Stage, a steady and precision rate will get with each count bunch it investigations, and the fulfillment of a standard 25 ages (group examination) of the entire preparation dataset from the IAM, exactness expands.

#### 4. RESULTS AND DISCUSSION

The project had been implemented and developed perfectly, which resulted in the execution of the application successfully. Following the implementation of the python modules and the learning model, the website is then integrated with flask and the project is run, as we have seen. The model predicts the text from the image and later loads to a new page where the following output comes up. Here we show the model's input and output format. Assessing the neural network with adding a few more CNNs and RNNs for more accuracy. According to our experimentation, the minimal number of layers needed for the model to have a threshold accuracy of identifying a few to couple of lines to be predicted is about 5 for Convolved NNs and 2 for Recurrent NNs. Any more RNNs added to the model results in a good accuracy of predicting the correct phrase from the given inputs of a CNN, which in turn may also require a good, improved input from CNN's end that needs more significant number of CNN layers. A predictive and accurate scoring from CTC can also be prepared for an accurate loss error rate of the predicted words.

Words	Accuracy
Word	0.90
School	0.82
Battery	0.80
Today	0.85
Going	0.75
Final	0.81
Everybody	0.70
Using	0.83
Urgent	0.89
Waiting	0.72



#### 5. CONCLUSION

With the help of RNN, CNN we have built a model for taking a input from the user as a image or as a text and give the output as Standard English and attached with a audio file. Python with the help of TensorFlow made the task easy and with optimal layers and all the underlying layers. As these layers will help us with similar search of the letter for example the search algorithm for 'e' and 'i' is same which is not optimized at the highest.

#### 6. Future Works

In future we will be working on the User interface of the website so that everyone can use and understand it better and will be able

to use it to the maximize extent and will be reviewing more data to get high accuracy and deal with blur and similar search which leads to poor accuracy. The use of model.py and trywork will be use used to full extent and all the python applications are made sure to kept in the main file. We will also try to make this application offline which is quite challenging and would be helpful the user even in non-service area.

#### 7.REFERENCES

- [1] Qinge Xiao; Congbo Li; Ying Tang; Xingzheng Chen Automation Science engineering on how energy efficiency modulating developing using machine learning .
- [2] Survey of Post-OCR Processing Approaches:Thi-Tuyet-Hai Nguyen,AJatowt on optical character recognition on printed documents
- [3] Tahir, Ghalib Ahmed; Loo, Chu Kiong for the open-ended continual learning (CL) model that is regarded on extreme Learning Machines using the increasing of class.
- [4] Yulia S. Chernyshova, publisher on IEE with and idea of a Two-Step CNN Framework that helps as Text Line Recognition from the images captured
- [5] Yucheng Zhou and Zhixian the idea to carry out fully the Medical Motion Image Recognition with the joined help of convolutional Neural Network and IOTs (internet of things) via IEEE.

Pr2

ORIGINALITY REPORT

3%

SIMILARITY INDEX

2%

INTERNET SOURCES

2%

PUBLICATIONS

1%

STUDENT PAPERS

PRIMARY SOURCES

1

T. Daniya, S. Vigneshwari. "Rice leaf Disease Recognition based on deep learning using Zipier S-Method", 2021 Innovations in Power and Advanced Computing Technologies (i-PACT), 2021

Publication

1%

2

[www.philstat.org.ph](http://www.philstat.org.ph)

Internet Source

1%

3

[hrcak.srce.hr](http://hrcak.srce.hr)

Internet Source

<1%

4

Yucheng Zhou, Zhixian Gao. "Intelligent Recognition of Medical Motion Image Combining Convolutional Neural Network With Internet of Things", IEEE Access, 2019

Publication

<1%

5

Vishnu Preetham Revelli, Gauri Sharma, S. Kiruthika devi. "Automate extraction of braille text to speech from an image", Advances in Engineering Software, 2022

Publication

<1%