USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science

B.S. Computer Science

FINAL YEAR PROJECT REPORT

Batch-2019

IMAGE DETECTION AND LANGUAGE TRANSLATION

By

Kanwar Azlan	19B-134-CS
Naufil Nadeem	18B-031-CS
Syed Zia Abbas	19B-102-CS
Hammad Akbar	19B-096-CS

Supervised by

Sir Hafeez Adam

ST-13, Block 7, Gulshan-e-Iqbal, Abul Hasan Isphahani Road, Opposite Safari Park, P.O. Box 75300, Karachi, Pakistan. Phone: 34978274-5; 34994305; 34982476; http://www.uit.edu

FYP Report Review

Content	Completeness checklist	Format
Title Page	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Include Project Title, Students Detail, and Supervisor name	As specified in sample report book
Declaration page	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Declaration of project containing Paragraph & Students Detail	As specified in sample report book **Title:** Font Times New Roman 18 **Paragraph:** Font Times New Roman 12, Line spacing 1.5, Align Justify
Acknowledgment	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Thanking those who have supported during FYP, directly and indirectly	
Abstract	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Student will have to express the problem statement, few statements on solution, and few statements on future work	
Table of Content	☐ Implemented ☐ Partially Implemented ☐ Not Implemented	System Generated (MS word/Latex)
List of Tables	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Contain List of Table with its caption used in the report	MS Word Generated Table number: Times New Roman 12 Bold Table caption: Times New Roman 12
List of Figures	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Contain List of Figure with its caption used in the report	MS Word Generated Figure number: Times New Roman 12 Bold Figure caption: Times New Roman 12

Chapter 1 (Introduction)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Explain the brief introduction of the project Describe project with the help of any abstract diagram Brief structure of report, briefly describe what is covered in upcoming chapters	**Heading 1:** Times New Roman 16 Bold **Heading 2:** Times New Roman 14 Bold **Heading 3:** Times New Roman 12 Bold **Paragraph:** Times New Roman 12, Line Spacing 1.5, Align Justify **Follow Guidelines of heading, figures, tables, references, etc., 14.4 -14.10** Use consistent naming convention throughout the report e.g. if usecase name is "Post to Blockchain", then use same name in UML diagrams, interfaces, reports, code, and testcases to show relationship.
Chapter 2 (Background and Literature Review)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Contain the background knowledge: what is so far done related to project in chronological order And related work: **similar applications** (their source, main features, deployment strategies, cost, etc.), **algorithms** (their source, significance, implementation details), preferred from research papers, and books.	
Chapter 3 (Aim and Statement of Problem)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Mention problems Regarding different stakeholders, domain and technology with separate headings and subheadings	
Chapter 4 (Hardware, Software Analysis & Req)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Includes Factfinding techniques used Hardware Analysis with comparison Software Analysis with comparison Requirements according to the project	
Chapter 5 (Software Design and Modeling)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented	

	Includes Architecture used Structural design diagrams Behavioral design diagrams Prototypes	
Chapter 6 (Algorithm analysis and complexity)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Includes Comparison with similar algorithm Show its best, average, and worst values in context of time and space complexity	
Chapter 7 (Implementation)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Refactored code with pretty printing, of technical code Operational, component, and deployment diagram Application flow with state transition diagram	
Chapter 8 (Testing)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Testcases of whitebox testig Testcases of blackbox testing System testing Experiments/Simulation and their results	
Chapter 9 (Discussion) optional	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Experimentation took place and their results	
Chapter 10 (Conclusion)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented A brief Conclusion of the software	
Chapter 9 (Future Work)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Include the Future Enhancement of the project (if any)	

Chapter 10 (Reference)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Include References/Bibliography according to IEEE Format	
Chapter 11 (Plagiarism Report)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Include Turnitin Generated Plagiarism/Similarity Report	
Chapter 12 (Achievements)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented Includes participations in different competitions, conferences, incubation activities, and exhibitions	
Chapter 13 (Annexure)	☐ Implemented ☐ Partially Implemented ☐ Not Implemented In **acknowledgement chapter**, you may include official letters from organizations. In **introduction chapter** 2-3 pages about organization for which you are developing the project In **background and literature review chapter**, research paper that is basis of your project, details of similar projects, any UML diagrams from other sources that has strong relationship with your project In **hardware, software analysis and requirement,** you may add hardware pacification, use case narrations, or detailed requirement specification document. In **software design and modeling chapter**, you may add detailed design documents other than most significant. In **software algorithm and complexity**, you may attach actual algorithm or its research paper.	

	In **achievement chapter**, you have to mention correspondence (letters, emails etc.), copy of certificate, pictures of participation specially at time of award ceremony.	
Chapter 14 (General guidelines)	Read carefully all guidelines specially **14.4 -14.10**	

Declaration

I affirm that this project report is a product of my own original work, with proper acknowledgment given to all citations and quotations used. Furthermore, I confirm that this report has not been submitted for any other degree or award at USMAN INSTITUTE OF TECHNOLOGY or any other institution simultaneously or in the past.

Kanwar Azlan (19B-134-CS):

Signature: ________________________________

Naufil Nadeem (18B-031-CS):

Signature: ________________________________

Syed Zia Abbas (19B-102-CS):

Signature: ________________________________

Hammad Akber (19B-096-CS):

Signature: ________________________________

Date: 31/Jul/2023

Acknowledgments

Final year project is the most significant part of every institute as well as students who willing take this opportunity to enhance their professional skills. This Report is also a way of presenting the efforts of students in the regard of FYP and can take a better understanding of the actual hard work. From the Head of Usman Institute to all the teachers as well as HOD's and supervisor's, they really give their precious time in order to make the FYP successful, So, from my whole FYP Group, I hereby thanks to all the supporting Team that make this happen.

Abstract

Nowadays, the digitization of documents and the adoption of paperless workplaces have become commonplace in commercial and professional endeavors. Having a simple and efficient method to create, store, and safeguard important documents is a wise choice. Document scanning emerges as a practical solution to replace the conventional manual approach to document management, offering numerous advantages such as optimized office space utilization, easy information storage and sharing, enhanced data security, and reliable data recovery.

In this project, our aim is to provide an enhanced and user-friendly application that keeps pace with the rapidly evolving world. We have chosen to develop an Android application due to the widespread familiarity and hands-on experience of users with this platform. The project idea revolves around leveraging the capabilities of two languages in tandem.

The primary purpose of this application is to enable users to enter text in their source language and have it translated into their desired target language. To achieve this, we employ image detection techniques. Users start by scanning an image containing text, which is then converted into editable text using Optical Character Recognition (OCR) technology. However, extracting text from natural scene photos presents unique challenges due to complex backgrounds, varying text patterns in terms of font, color, scale, intensity, and orientation. Despite these difficulties, our application employs advanced text identification and extraction methods to accurately compute the regions of the images containing text characters or strings captured through the camera.

The final step involves utilizing Natural Language Processing (NLP) techniques to translate the extracted text into the desired target language. By combining OCR and NLP, our application offers a seamless and efficient solution for translating text from images, empowering users with swift and accurate language translation capabilities.

Table of contents

List of Tables

List of figures

1. Introduction

Text extraction from images is a highly intricate task within digital image processing. Detecting and recognizing text from an image involves complex processes. Computer software can potentially achieve text extraction from images using sophisticated algorithms. However, such solutions may not always be applicable in the current environment.
Currently available language translators, including voice-based and keyboard-based options, may not be user-friendly. This work aims to establish a dynamic connection between text and interactive visualization imagery, showcasing its potential.
Using the Android device's camera, this project demonstrates the feasibility of text extraction from images, employing an algorithm implemented in Java. With the vast number of mobile users worldwide, the convenience of capturing images for text extraction is evident.
The primary objective of this project is to implement text extraction from images and subsequently translate the extracted text. The information extracted from camera-captured images in natural scenes holds promise in various image-based applications, such as assistive navigation, auxiliary reading, image retrieval, and scene understanding.
Extracting text from images in natural scenes poses significant challenges due to complex backgrounds and the considerable variation in text patterns, including font, color, scale, intensity, and orientation. Therefore, a crucial step in extracting text from camera-captured images involves text detection and extraction, which calculates the regions containing text characters or strings.
Upon capturing the image, various processes are initiated to detect and extract the text, which is then translated. This project aims to address the complexities of text extraction from images and explore its potential applications in diverse areas, contributing to the advancement of digital image processing technology.
The main purpose to serve the people with an enhanced and useful application that brings them much faster as the world is growing. To capture the idea of dealing with the android application is chosen due to the vast and hands on experience of the users to the android app. As far as project idea is concern, it gives us the full exposure to use of two languages in parallel.

The project is basically allowing the user to enter the text in the source language and it will be translated into the target language. This work is done by the image detection. First the user will scan the image containing some text. Then it will be converted into the readable text by the help of OCR (Optical Character Recognition). Then it will be translated into the target language by using a technique called NLP (Natural Language Processing).

2. Background and Literature Review

BACKGROUND:

Text extraction from image is one of the complicated areas in digital image processing. It is a complex process to detect and recognize the text from image. It's possible of computer software can provide extracted text from image using most complicated algorithm. So it can't be use anywhere in this existing environment. Here different types of language translators are available such as voice based translator, keyboard based translator etc. But those translators are not easy to use. The purpose of this work is to demonstrate that a tight dynamical connection may be made between text and interactive visualization imagery. The Android device camera can prove this type of extraction and also the algorithm will easily implement using java language. Millions of mobile users in this world and they always have mobile in their hand, so simply they can capture the image to extract the text. The purpose of this project is to implement text extraction from the image and translating the text. Captured text information from camera in natural scene images can serve as indicative marks in many image based applications such as assistive navigation, auxiliary reading, image retrieval, scene understanding, etc.

Extracting text from natural scene images is a more challenging problem as compared to scanned document because of complex backgrounds and also large variations of text patterns such as font, color, scale, intensity and orientation. Therefore, to extract text from camera captured images, text detection & extraction is an important and essential step which computes the sub regions of the images containing text characters or strings. Once the image is captured from camera, the image went through various processes whose task is to detect the text within the image and extract those texts then translates that text.

There was a lot of people who doesn't know how to read English and Urdu so this application translates the text from English to Urdu or Urdu to English (Dual Zone) and also have a facility of voice command

As the problem states that there are most of the people which doesn't understand and read the English and there are most of the people who doesn't know how to read Urdu but understands Urdu so this app will be beneficial for those. This app solves a problem of three types of people:

1.) Once a person doesn't Understand the English so this app converts document through image processing in Urdu
2.) Once a person doesn't understand or read Urdu so this app converts Urdu document through image processing in English.
3.) Once a person doesn't know how to read Urdu/English but he/she Understand English /Urdu so the app has also ability to read the document

In today's business and work landscape, the trend towards paperless offices and digitizing documents has become increasingly common. Embracing an effortless approach to creating, storing, and safeguarding essential documents is crucial. Document scanning emerges as an excellent solution, offering numerous advantages over traditional manual methods.

By opting for document scanning, businesses can free up valuable office space while ensuring efficient information storage and sharing. Additionally, it enhances data security and facilitates smoother data recovery processes. In this paper, we propose a software interface for document scanning, aiming to streamline the transition to a paperless and digitally-driven environment.

The main purpose to serve the people with an enhanced and useful application that brings them much faster as the world is growing. To capture the idea of dealing with the android application is chosen due to the vast and hands on experience of the users to the android app. As far as project idea is concern, it gives us the full exposure to use of two languages in parallel.

LITERATURE REVIEW:

Now a day it’s a vast experience to handle a software where you can extract some information related to an image. To be more precise it’s a powerful work experience to build something like OCR that actually make your work easier. OCR is basically an Optical Character Recognition Tool that helps to extract a text from image. Scan your desired image and extract the text with the help of OCR.

It works really simple in a way that it separates the background and foreground of the image and extract the pixels out of the image, through which the text can be extracted. Now Comes the most important and vital part of our project. The Text translation is the legit work to be done. It basically involves AI and ML that include some transformers and a technique called NLP that creates the opportunity of the language translation or machine translation.

By processing natural language to translate it into another natural language, machine translation (MT) plays a significant role in helping linguists, sociologists, computer scientists, and other professionals. And with the massive interchange of information between various regions with various regional languages over the past few of years, this demand has increased enormously. There are many difficulties with machine translation, some of which include:

Two given languages may have entirely distinct structures because of the following reasons:
a) Not every word in one language has an equivalent word in another language; and
b) Words can convey multiple meanings.

Due to these difficulties and numerous more, MT has been a popular subject of study for more than 50 years. Numerous approaches have been put forth in the past with the intent of either enhancing the calibre of the translations produced by them or investigating the resilience of these systems by evaluating how well they work across a wide range of languages.

We discuss statistical methods (in particular word- and phrase-based methods) and neural methods in our survey of the literature since they have produced cutting-edge findings in a variety of important languages.

The process of transforming the original language's (the source language, or SL) structure into another language's (the target language, or TL) is known as translation. This procedure fits the definition of intra-lingual translation, which is the understanding of verbal cues using some other languages (Munday, 2016: 8). Larson (1998:3) offers a different definition, claiming that translation is the process of transferring meaning from texts written in the source language into texts written in the receptor language.

To complete the procedure, the semantic structure must be changed from the form of the ST to the form of the TT. Only the forms change; the meaning remains unchanged. According to Catford (1978: 20), translation is the process of changing text in one language (SL) by.

The phrase "textual material" emphasises that only portions of the original sign language text are translated in favour of their translations into the target language. For instance, if we translate "How old are you?" from the Indonesian "Berapa umurmu?" into English, the grammar and lexis of the source language (Indonesia) are replaced by those of the target language (English), which are equivalent. 12 According to Newmark (1988: 21), there are two methods for translating.

[1] According to this paper (N.Mangrulkar, 2021),

[2] In this research (Misra, Swain, & Mantri, 2012),

[3] Regarding the transformers (Liu, 2021),

[4] In this Article (D.Bahdanau, 2016),

[5] The researchers conclude (h.Sutskever, 2014),

[6] According to this conference proceeding (B.Shi, 2015),

[7] In the addition of Transfomers (Zhang, 2017),

[8] The Article states (Baniata, 2021),

[9] Researchers describes as (Sefara, 2021),

3. Aim and Statement of Problem

As the problem states that, there are most of the people, which does not understand and read the English and there are most of the people who does not know how to read Urdu but understands Urdu so this app will be beneficial for those. This app solves a problem of three types of people:

1) Once a person does not understand the English so this app converts document through image processing in Urdu.
2) Once a person does not understand or read Urdu so this app converts Urdu document through image processing in English.
3) Once a person does not know how to read Urdu/English but he/she Understand English /Urdu so the app has also ability to read the document.

1. Bridge the Linguistic Barriers to the Countries

It is helpful to have an interpreter on hand when conversing with somebody from a different nation whose language you do not understand. Language translators have special training to speak and communicate across different languages effortlessly. If they have the necessary qualifications, you can be confident that your communications will be correctly understood and won't suffer from any unintentional miscommunications. Professional translators have become an essential component of contemporary enterprises because they enable people and organisations to reach new markets by enabling them to communicate with a specific target audience in their native tongue.

2. Improved in the Communication

With the aid of language translation services, parties can interact and exchange ideas without linguistic barriers. You can simply convey difficult subjects to someone else in their own language through translation. Professional translators are able to maintain a very high level of precision in their job, rapidly and accurately communicating ideas without changing their intended meaning or overlooking minute details.

4. Hardware, Software analysis and requirements

4.1 Software Analysis:

Front End:

1- **Flutter (Framework):**
We are using Flutter as our front-end tool. Basically all other frameworks are available for the mobile application development like Django, flask etc. But Flutter is widely used now a days and is created by Google. A Platform for the Front end tool that helps us to take better approach in the mobile application development. Dart is the language used by flutter framework. Flutter apps are written in the Dart language and make use of many of the language's more advanced features. While writing and debugging an application, Flutter runs in the Dart virtual machine, which features a just-in-time execution engine.

Back End:

1- **OCR (Optical Character Recognition):**

Optical character recognition (OCR) Text recognition, also known as OCR (Optical Character Recognition), involves the extraction and conversion of data from various sources such as scanned documents, camera images, and image-only PDFs. The OCR program identifies individual letters within the image and assembles them into words, further organizing the words into sentences. This process enables users to access and edit the original content, eliminating the necessity for manual data entry. OCR technology plays a significant role in transforming images into machine-readable and editable text, enhancing data accessibility and efficiency in various applications.

NLP (Natural Language Processing):

Natural Language Processing (NLP) is an interdisciplinary field that integrates computer science, information engineering, and artificial intelligence (AI). Human communication relies on words, but computers process information using numerical language. NLP bridges this gap by enabling computers to understand, interpret, and generate human language, facilitating seamless interactions between humans and machines.

2- **Python (Programming Language):**
Python is widely used and most effective high-level programming language. We Train our model with the data of source language and target language using python.

4.2 Requirements:

- As a Front-end developer, I want flutter framework to be used in our project so that it will help us in the easy execution of the mobile application.
- As a Back end developer, I want Python to be chosen as our programming language so that we have a great scope and it provides many libraries built in that will help us throughout the process.
- As a programmer or student, I want to first train or test our program so train a model that consists of desired data and expected to give desired output can do it.

- As a student, I want to learn NLP and its execution so that it will help us in the domain of language translation and it will be helpful in order to train our model.
- As a data analyst, I want data collection of the source and the target languages so that we can have a better approach to how we execute our model.

4.3 System Diagram

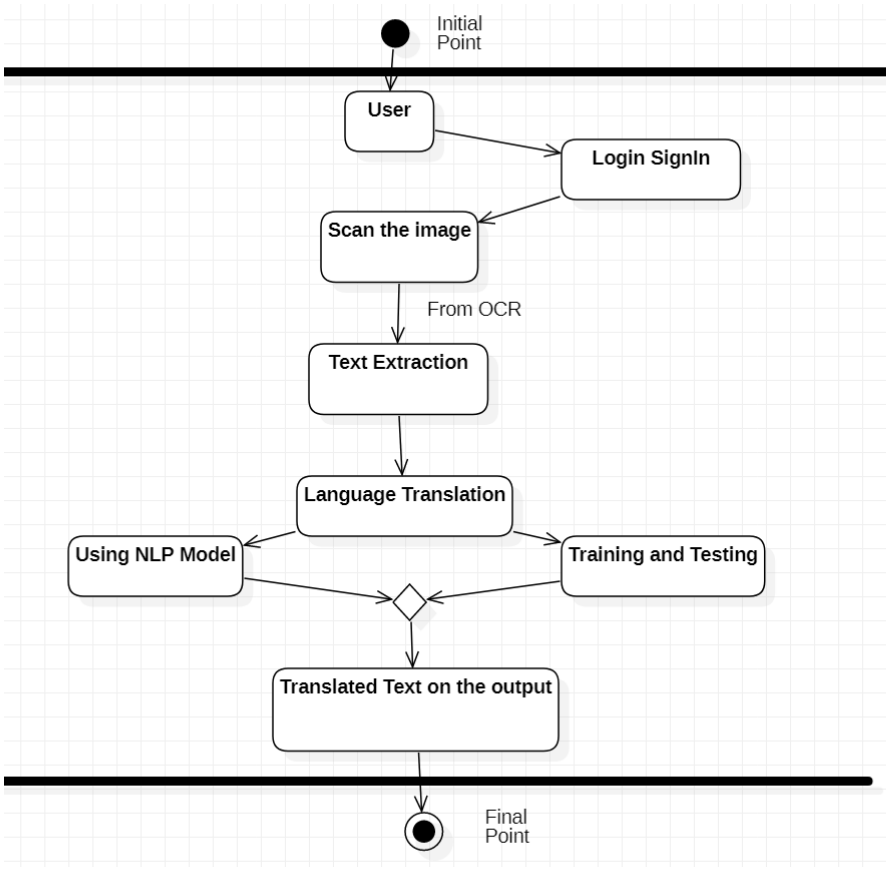


Figure 1: System Flow Diagram

4.4 Activity Diagram

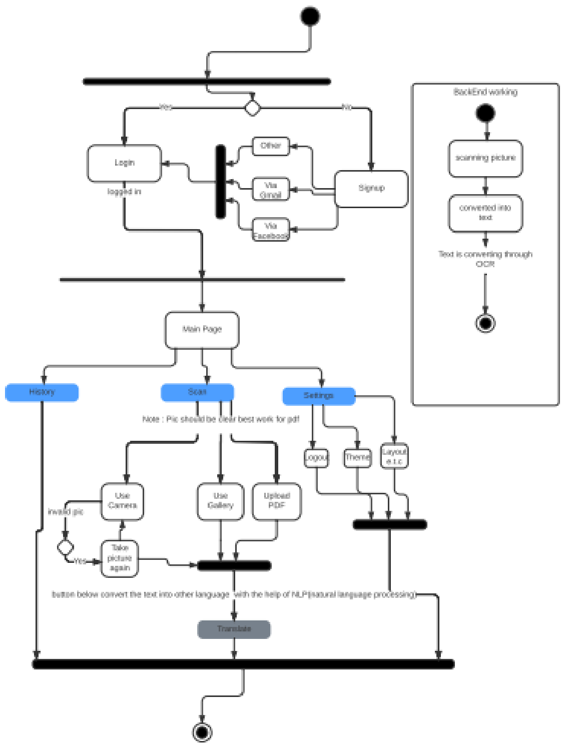


Figure 2: Activity Diagram

This is the android application, which helps the user for language translation. First user will enter the credentials (ID, Password) or can sign up by google, twitter etc. then we have our main user interface (UI) or our main page that gives three options to the user that either user want to scan the image by opening the camera, or user can upload an image or user can upload PDF document. Then firstly, our software extracts the text from the image by the help of OCR. It works at the back end by separating the colors of the image or pixels and background and foreground and then extract the text easily. Now comes the main task of this App, this text is being converted to another language (Target language) by the help of NLP (natural language processing). It works as a model, which is trained by some data input, and then with the help of data it will show us the output in the target language.

4.5 Use Case Diagram

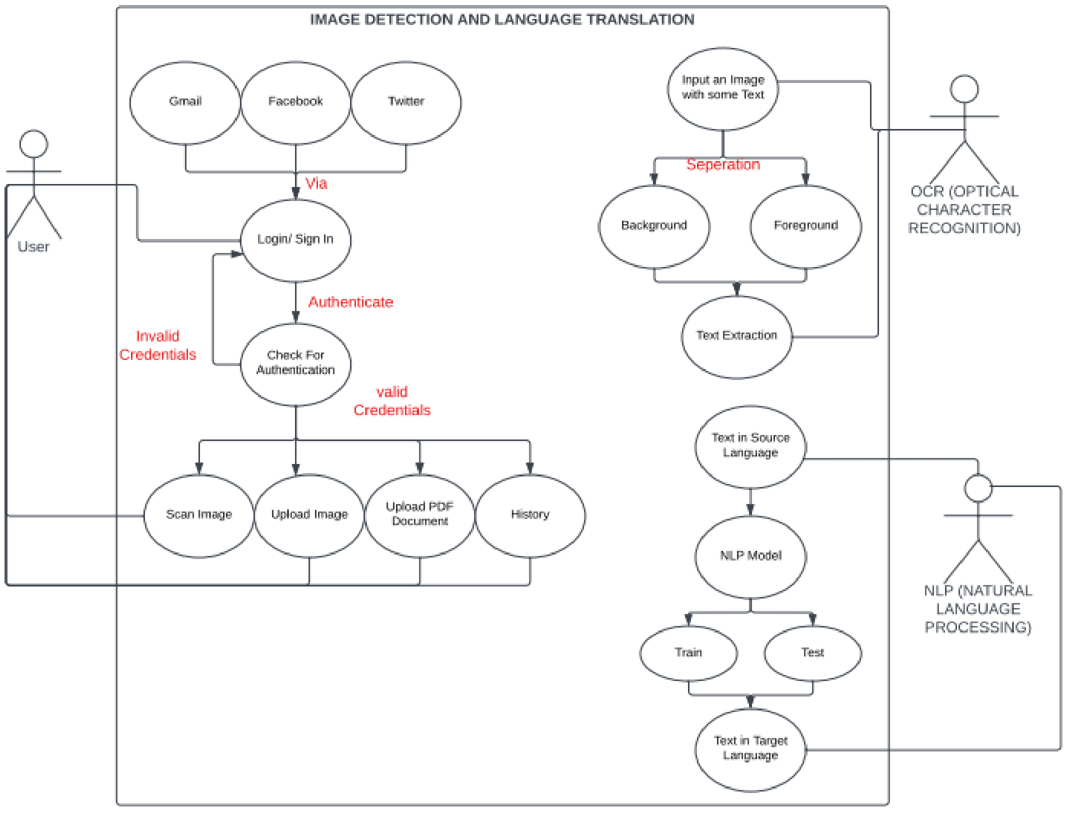


Figure 3: Use Case Diagram

Actor is the main concept in use case diagram. Therefore, our App consist of three types of actors. First is User itself, login or sign up into the app and can get the job done by uploading an image via PDF or Scanning that extract the text. Then this app will translate the text to target language. Next Actor is OCR (An image recognition software) that works tesseract library of python. This helps the user to extract the text from an image. The Last is NLP (A AI based technique) that works as machine translation. It is about building a model using transformers and then train the model by giving them input data of the languages and then tested for the output (Text into target language).

5. Software Design and Modeling

5.1 PROJECT ARCHITECTURE:

5.1.1 INTRODUCTION:

Project Architecture is based on OCR (Optical Character Recognition). It is based on the Image processing technique to play with the images and design or manipulate in different ways. Text recognition, often referred to as optical character recognition (OCR), involves the extraction and reuse of data from various sources such as scanned documents, camera photos, and image-only PDFs. An OCR application enables access to and editing of the original content by converting isolated letters on the image into words and then assembling those words into sentences. This advanced software eliminates the need for manual data entry, streamlining the process of transforming images into machine-readable and editable text.

5.1.2 HISTORY

Ray Kurzweil founded Kurzweil Computer Products, Inc. in 1974. This company's Omni-font optical character recognition (OCR) equipment could read text that was written in almost any typeface. He came to the conclusion that the ideal use of this technology would be a machine-learning aid for the blind, so he developed a reading machine that could convert text into speech. In 1980, Kurzweil sold his business to Xerox, which was keen to advance the sale of text conversion from paper to computers.

While digitizing old newspapers in the early 1990s, OCR technology gained popularity. The technology has advanced much since then. The technologies of today are capable of providing OCR accuracy that is almost perfect. Complex document-processing operations are automated using cutting-edge techniques.

The only way to digitally format documents prior to the development of OCR technology was to manually retype the text. This took a lot of time and unavoidably contained typographical and factual errors. OCR services are now widely accessible to the general public. For instance, documents can be scanned and stored on your smartphone using Google Cloud Vision OCR.

5.1.3 WORKING OF THE ARCHITECTURE

Optical character recognition (OCR) processes a document's physical structure using a scanner. After all pages have been duplicated, OCR software converts the document into a two-color or black-and-white form. Bright and dark areas are identified in the scanned-in image or bitmap, with the light sections being categorised as background and the dark areas being characters that need to be recognised.

Alphabetic or numerical digits are found after processing the black areas. You typically concentrate on one character, word, or portion of text at a time during this stage. The characters are then recognised using one of two algorithms: pattern recognition or feature recognition.

OCR systems combine hardware and software to convert physical, printed documents into machine-readable text. Hardware, such as an optical scanner or specialised circuit board, is used to copy or read text; the advanced processing is then often performed by software. OCR is a tool that we are employing to extract text from images.

It works in a way that an input image is passed through 3 main ways, Scanned Image from Camera, Upload Image from your Phone and you can upload PDF document if you want. The Process is like the input image is passed in the pytesseract and Opencv (Libraries for the implementation of the OCR) and they extract text from image and display it on the output screen.

5.1.4 BENEFITS OF USING THIS ARCHITECTURE:

The OCR has a lot more than we can think of, it is basically a very handy tool to continue with your daily life works and can make your working so smooth enough without wasting time. It has a lot of advantages like:

5.1.4.1 Cost Reduction.

5.1.4.2 Accelerates Workflow.

5.1.4.3 Automate documentation.

5.1.4.4 Easy to Use.

5.1.4.5 Works well with data.

5.1.4.6 Secure your data.

5.1.4.7 Handles accurate information.

5.1.4.8 Less time consuming.

5.1.4.9 Easy Data Handling and storing.

5.1.5 DESIGNING DIAGRAMS

5.1.5.1 Class Diagram

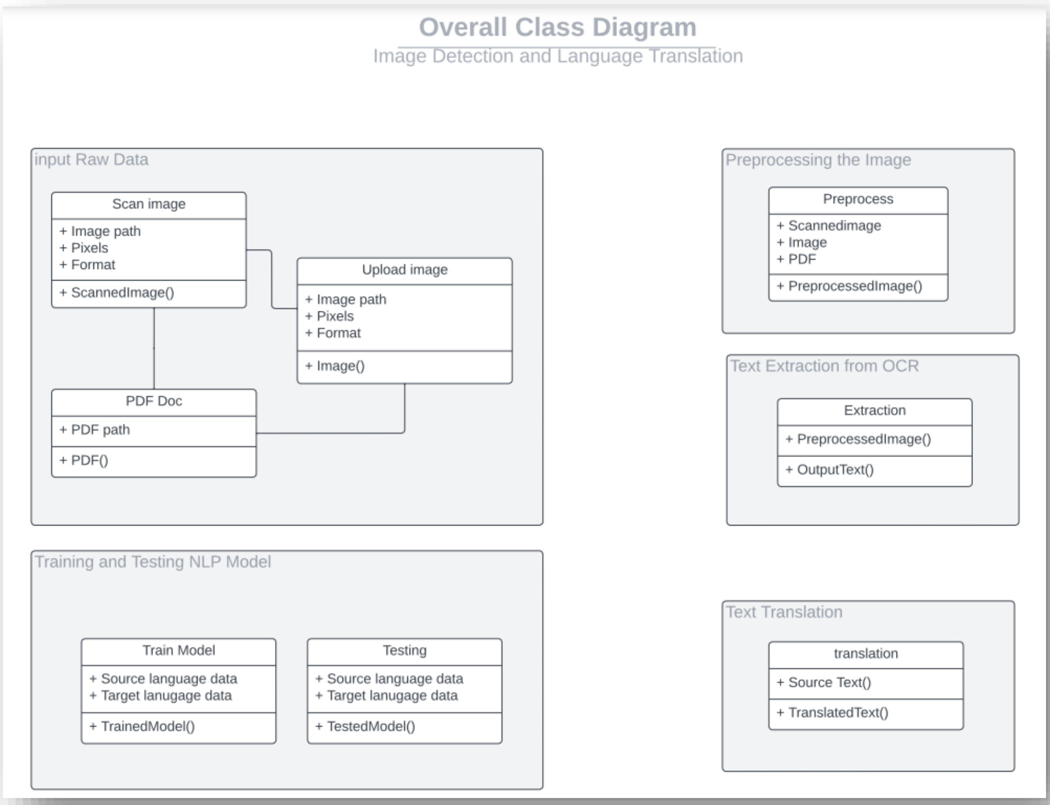


Figure 5.1.5.1: Class Diagram

5.1.5.2 Object Diagram

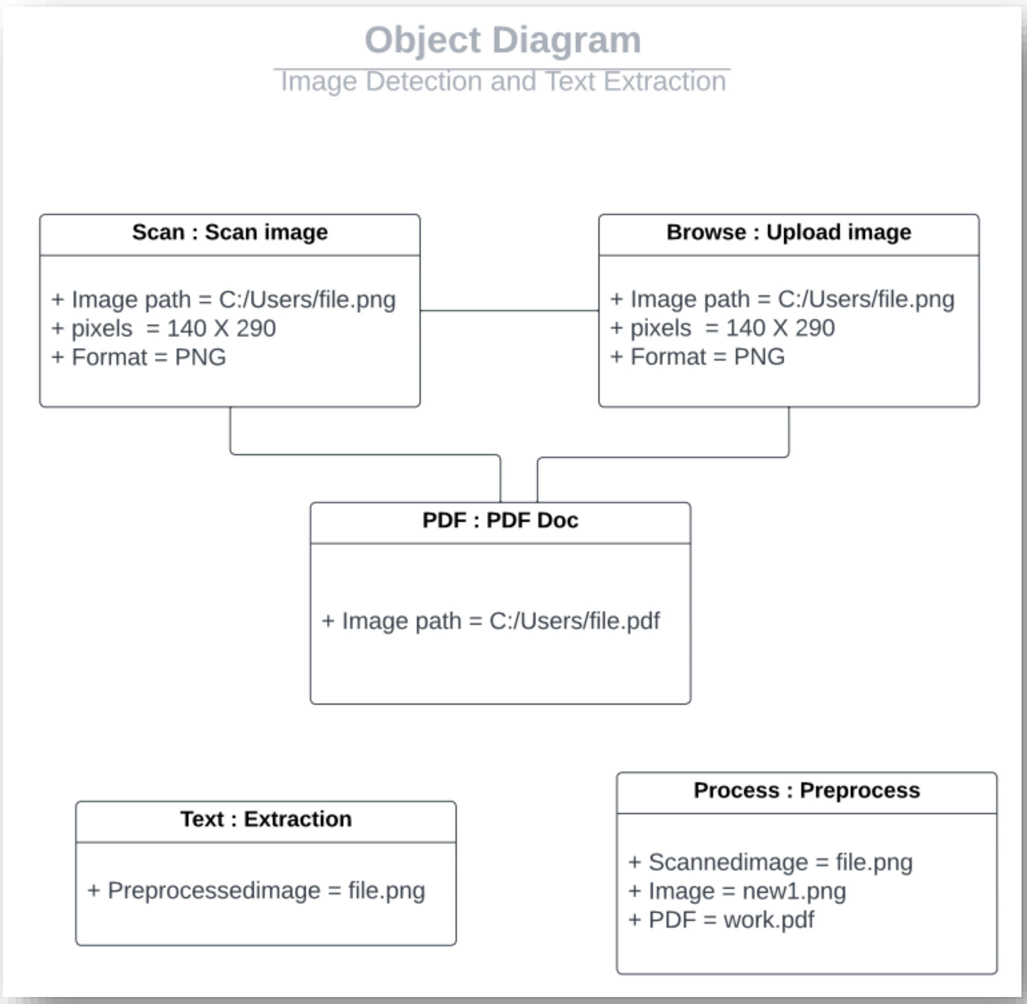


Figure 5.1.5.2: Object Diagram

5.1.5.3 ER Diagram

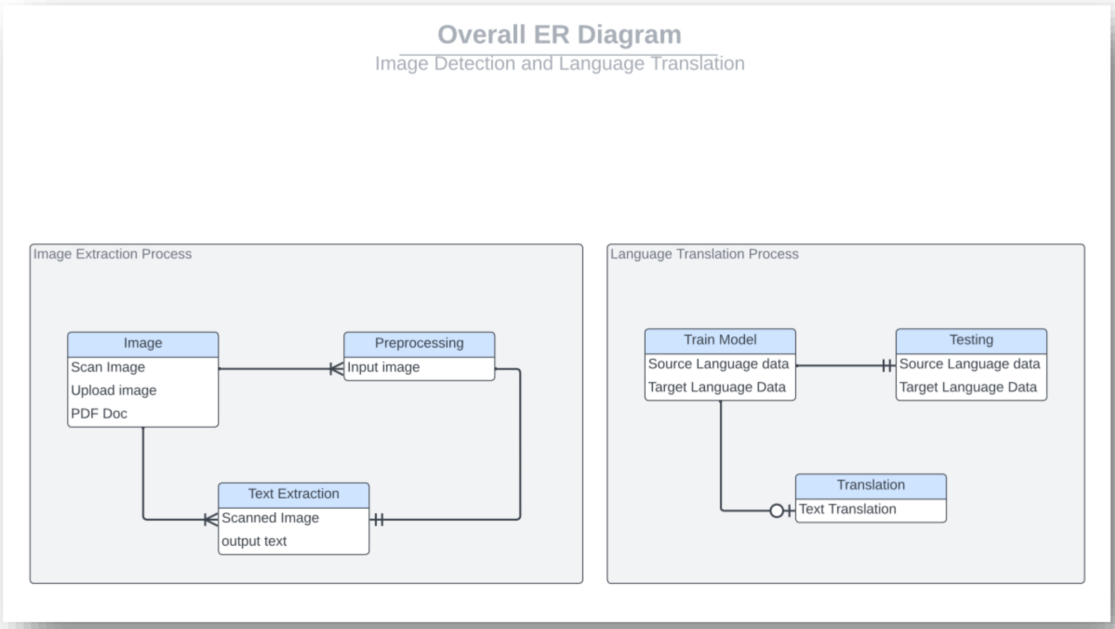


Figure 5.1.5.3: ER Diagram

5.1.6 BEHAVIORAL DIAGRAMS:

5.1.6.1 Sequence Diagram

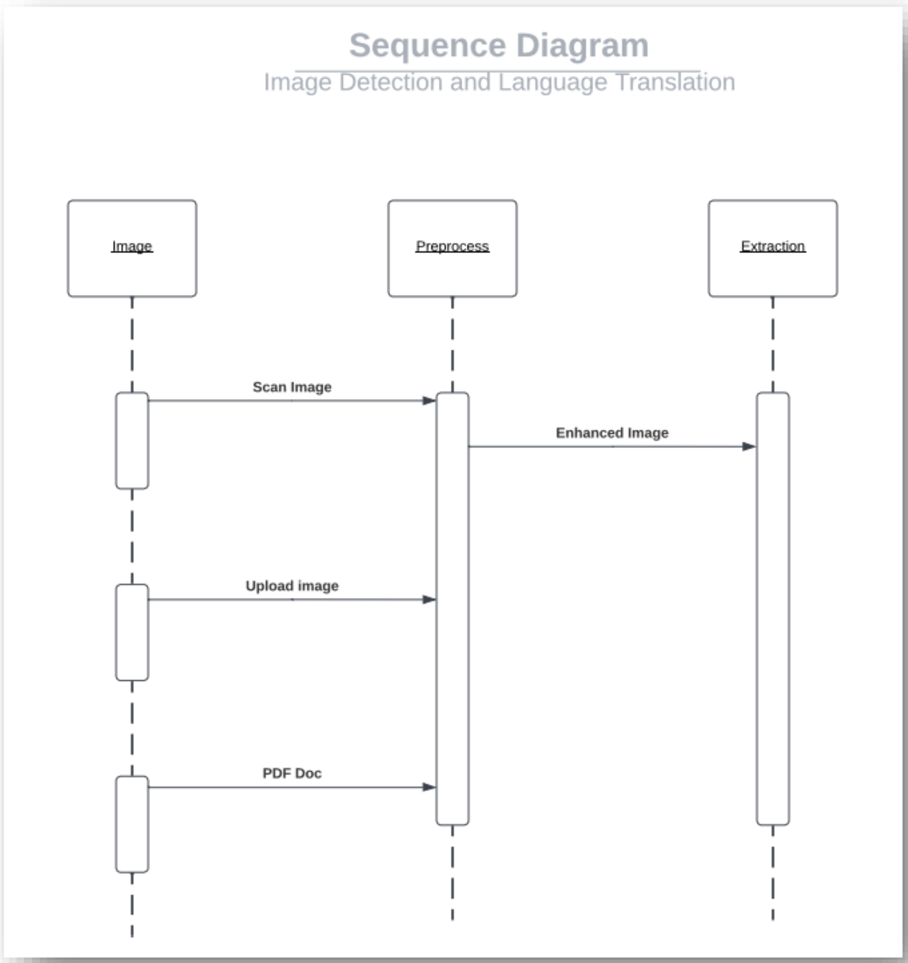


Figure 5.1.6.1: Sequence Diagram

5.1.6.2 Timing diagram

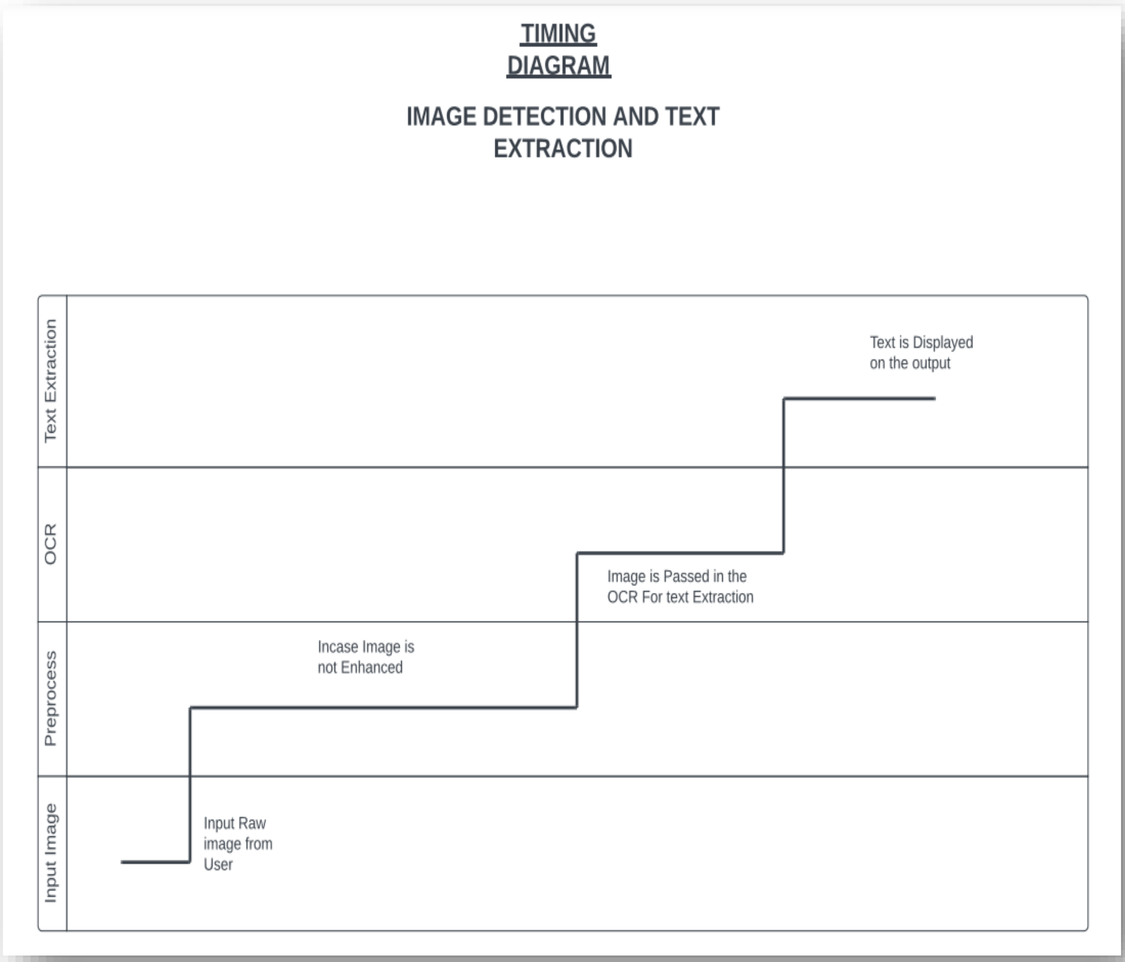

TIMING DIAGRAM
IMAGE DETECTION AND TEXT EXTRACTION
Text Extraction
OCR
Preprocess
Input Image
Text is Displayed on the output
Image is Passed in the OCR For text Extraction
Incase Image is not Enhanced
Input Raw image from User

Figure 5.1.6.2: Timing Diagram

5.1.6.3 Activity diagram

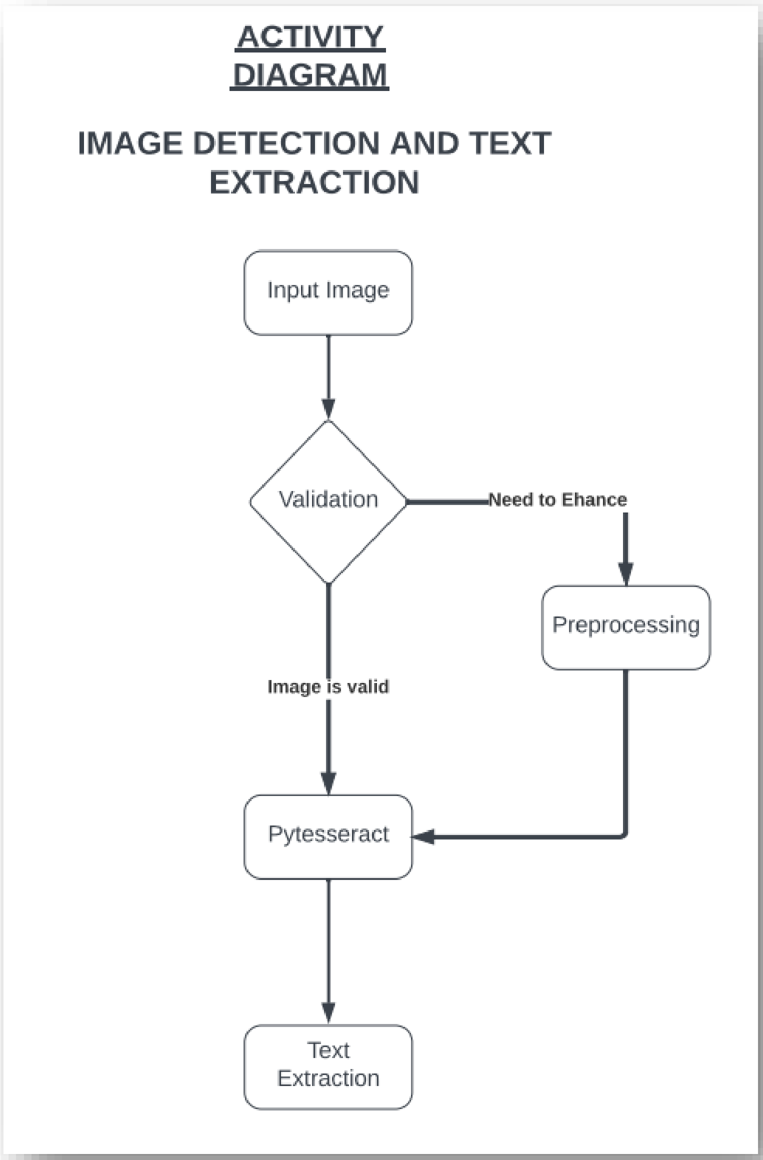


Figure 5.1.6.3: Activity Diagram

5.1.6.4 Composite Diagram

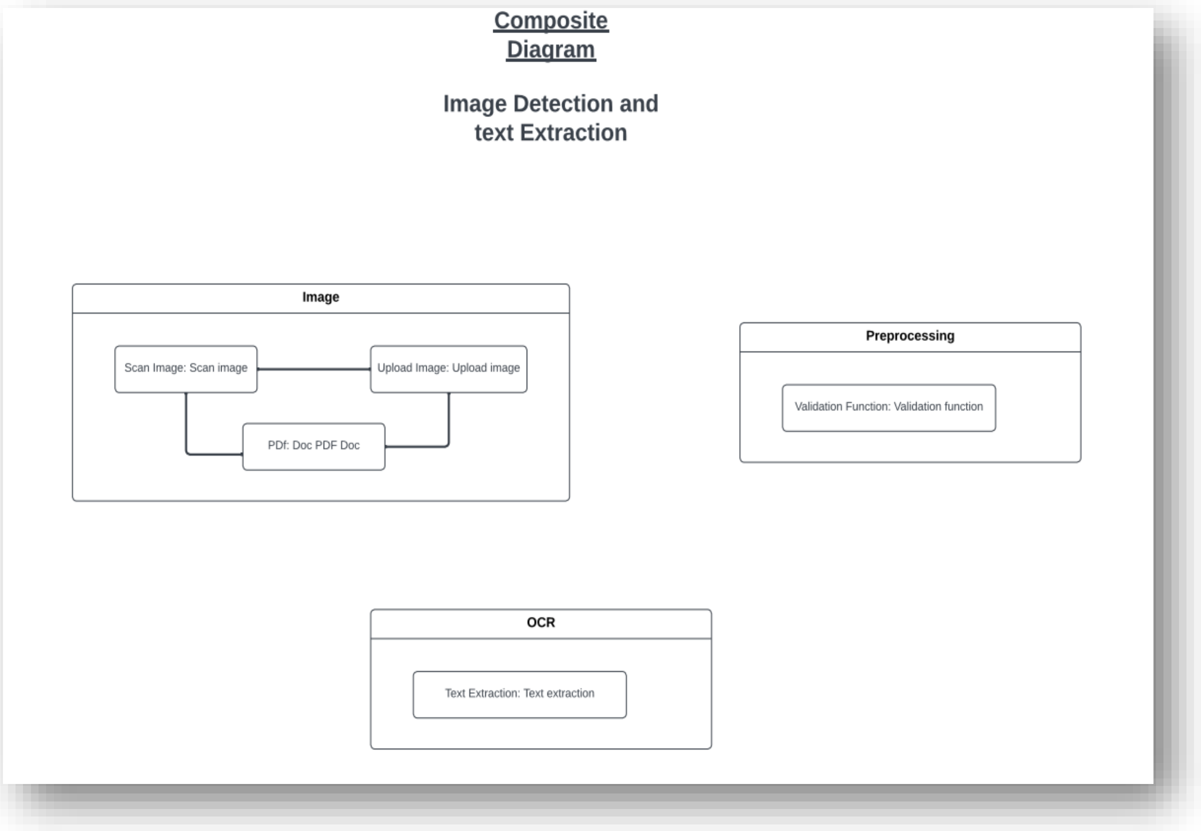
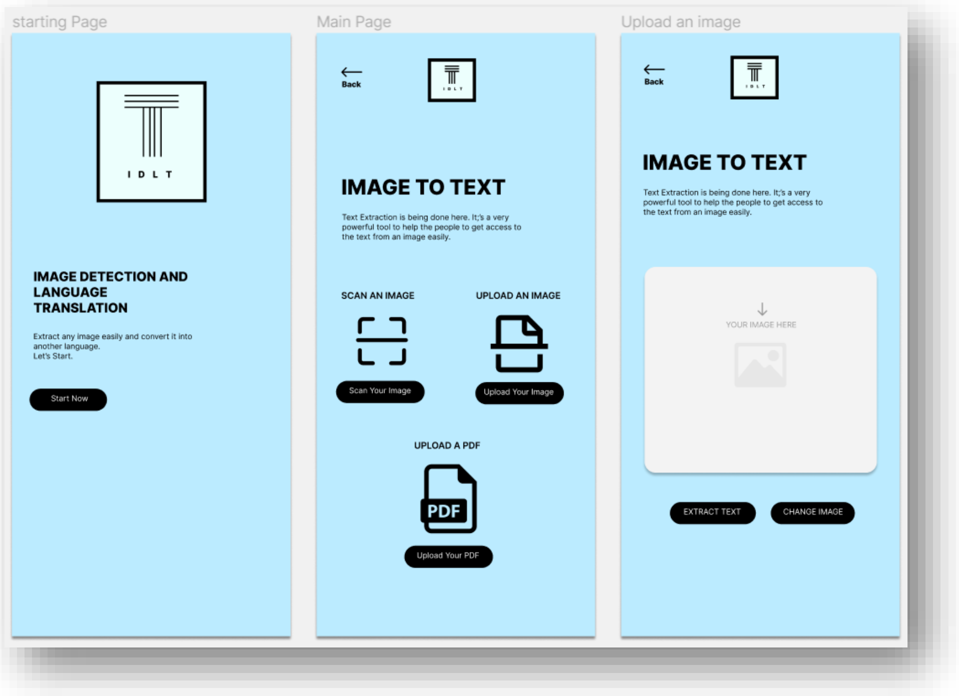

Composite Diagram
Image Detection and text Extraction
Image
Scan Image: Scan image
Upload Image: Upload image
PDf: Doc PDF Doc
Preprocessing
Validation Function: Validation function
OCR
Text Extraction: Text extraction

Figure 5.1.6.4: composite Diagram

5.1.7 FIEDILITY PROTOTYPE



Extract any image easily and convert it into another language.
Let's Start.

Text Extraction is being done here. It;s a very powerful tool to help the people to get access to the text from an image easily.

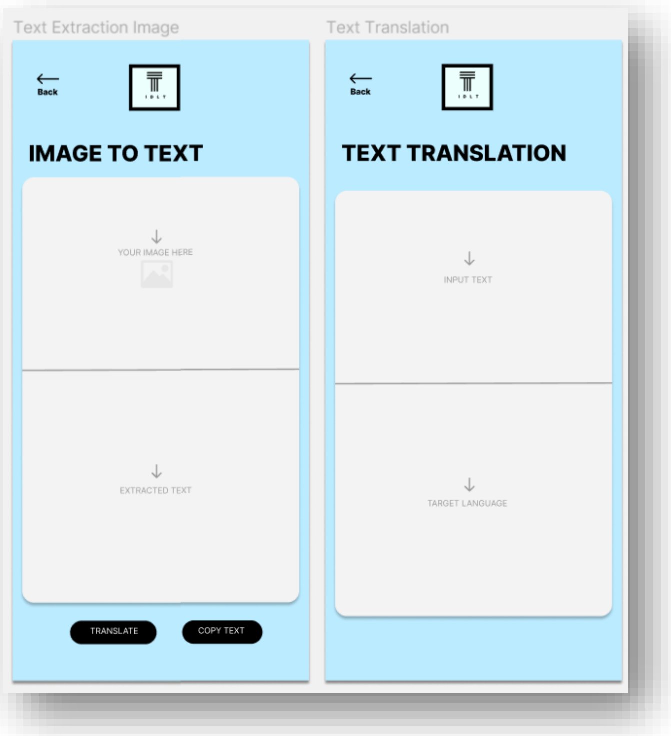


Figure 5.1.7: Fidelity Prototype

6. Algorithm analysis and complexity

.

6.1 ALGORITHM ANALYSIS

Algorithm analysis is a major part of the project, it initializes what and how the work is going to happen. There are various ways to implement one thing but very few are being considerable in action to perform such things. As per our project we are using three algorithms which are listed below:

- Histogram Equalization
- CLAHE (contrast Limited Adaptive Histogram Equalization)
- Otsu Thresholding

6.1.1 Histogram Equalization:

6.1.1.1 INTRODUCTION

"Image pre-processing" describes operations carried out on images at the most fundamental level of abstraction. If entropy is a measure of information, then these operations reduce the information content of the image rather than enhancing it. Pre-processing aims to enhance image data by reducing unwanted distortions and emphasizing specific visual features that play a crucial role in subsequent processing and analysis tasks.
Histogram equalisation is one approach for changing pixel brightness. It is a well-known contrast enhancing technique because it works on almost all types of photos.

6.1.1.2 DESCRIPTION

A histogram serves as an example of a frequency distribution. It forms the basis for a number of spatial domain processing techniques. Histogram editing can be used to enhance photographs.

Contrast is the term used to describe the difference in brightness between two objects in an image. Two objects can't be separated from one another when the contrast is too low; they are instead viewed as a single thing.

Histogram equalisation is a popular contrast-enhancement approach in image processing due to its high efficiency and simplicity. It is one of the trickier methods of modifying the dynamic range and contrast of an image by modifying its intensity histogram until it takes the appropriate form.

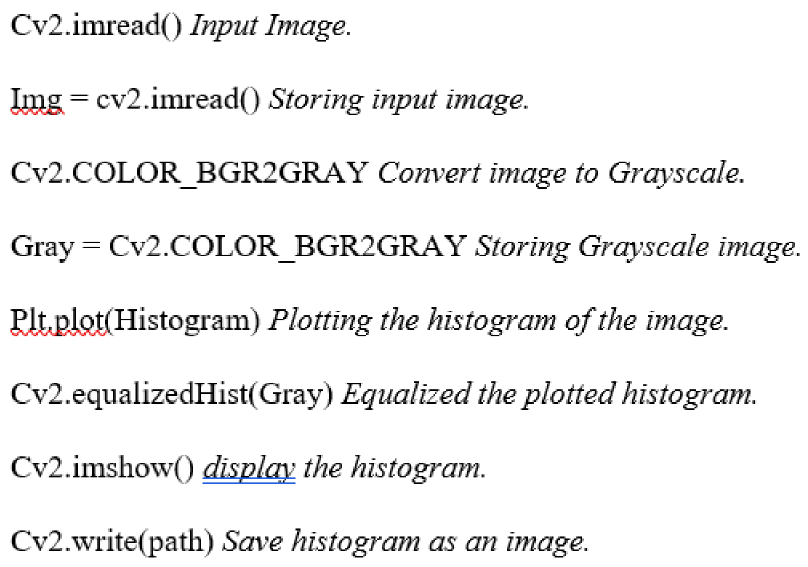
6.1.1.3 PURPOSE

Histogram Images' contrast is increased using a technique called equalisation in computer image processing. This is achieved by expanding the image's intensity range and distributing the most common intensity levels effectively. By doing so, this technique enhances the overall contrast of the images, particularly in cases where valuable information is represented by closely related contrast values. Consequently, regions with insufficient local contrast can gain additional contrast through this process.

6.1.1.4 SIGNIFANCE

Histogram equalization is a technique used to improve the contrast of images. Its main advantage is that it can increase the overall contrast of the image, making it look more vivid and clearer. Additionally, it can also help to reveal details that were previously obscured by low contrast. Additionally, Histogram Equalization can also be used to balance the intensity levels of an image, making it useful for image processing tasks such as image segmentation, feature extraction, and more.

6.1.1.5 PSESUDOCODE



```
Cv2.imread() Input Image.

Img = cv2.imread() Storing input image.

Cv2.COLOR_BGR2GRAY Convert image to Grayscale.

Gray = Cv2.COLOR_BGR2GRAY Storing Grayscale image.

Plt.plot(Histogram) Plotting the histogram of the image.

Cv2.equalizedHist(Gray) Equalized the plotted histogram.

Cv2.imshow() display the histogram.

Cv2.write(path) Save histogram as an image.
```

6.1.2 CLAHE (Contrast Limited Adaptive Histogram Equalization):

6.1.2.1 INTRODUCTION

"Image pre-processing" describes operations carried out on images at the most fundamental level of abstraction. If entropy is a measure of information, then these operations reduce the information content of the image rather than enhancing it. By minimising undesirable distortions or boosting particular visual features that are crucial for later processing and analysis activities, pre-processing seeks to enhance the picture data.

Histogram equalisation is one approach for changing pixel brightness. It is a well-known contrast enhancing technique because it works on almost all types of photos.

6.1.2.2 DESCRIPTION

The histogram equalization described above takes the image's overall contrast into account, which is frequently a bad idea. The output of the histogram equalization incorrectly displays the human image from the original diagram. Even though the image's contrast had been increased, too much brightness caused us to lose some of the information. The histogram is not limited to the local area, which is the cause.

To address this problem, adaptive histogram equalization is used. This technique involves breaking the image into smaller blocks and histogram equalizing each of them.

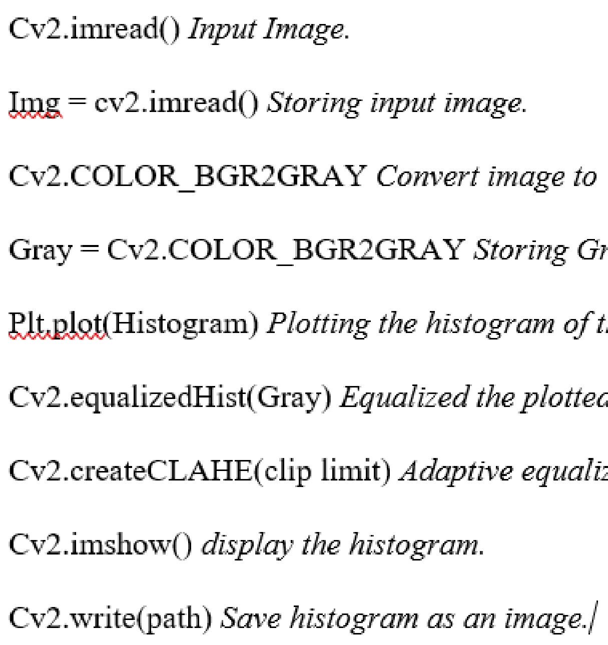
6.1.2.3 PURPOSE

The histogram equalization method described earlier considers the overall contrast of the image, which may not always be suitable. This can lead to undesired outcomes, such as the human image from the original diagram being inaccurately represented. While the contrast is enhanced through histogram equalization, an excessive increase in brightness can result in the loss of some information. The issue lies in the histogram's lack of restriction to local areas.
To overcome this limitation, adaptive histogram equalization is employed. This technique involves dividing the image into smaller blocks and applying histogram equalization to each of these blocks separately. By locally adjusting the contrast in smaller regions, adaptive histogram equalization helps to preserve more details and achieve a better representation of the image without compromising on information loss due to global contrast enhancement.

6.1.2.4 SIGNIFANCE

Photographs with poor contrast, as those taken in low light or with a low-end camera, or images that have been captured in similar conditions, benefit the most from it. The visibility of noise in an image can be diminished with the use of CLAHE. Additionally, it can aid in highlighting aspects of an image that might otherwise be challenging to see, such as tiny cancers in photographs used in medicine.

6.1.2.5 PSESUDOCODE



```
Cv2.imread() Input Image.
Img = cv2.imread() Storing input image.
Cv2.COLOR_BGR2GRAY Convert image to Grayscale.
Gray = Cv2.COLOR_BGR2GRAY Storing Grayscale image.
Plt.plot(Histogram) Plotting the histogram of the image.
Cv2.equalizedHist(Gray) Equalized the plotted histogram.
Cv2.createCLAHE(clip limit) Adaptive equalized histogram
Cv2.imshow() display the histogram.
Cv2.write(path) Save histogram as an image.
```

6.1.3 Otsu Thresholding:

6.1.3.1 INTRODUCTION

Let's first grasp the relationship between the technique and image thresholding before delving into its specifics, Simple thresholding has the drawback that the threshold value must be manually specified. We can manually test a threshold's performance by attempting various settings, but this is time-consuming and may not work in practice. Therefore, we require a method for computing the threshold automatically. A nice illustration of auto thresholding is the Nobuyuki Otsu-created Otsu method.

6.1.3.2 DESCRIPTION

To binarize a picture based on pixel intensities, image thresholding is utilized. This thresholding technique involves using a threshold value and a grayscale image as inputs to create a binary image as the output. In the resulting binary image, pixels that match the intensity threshold are designated as white (foreground), while pixels with intensities less than or equal to the threshold are represented as black (background).

6.1.3.3 PURPOSE

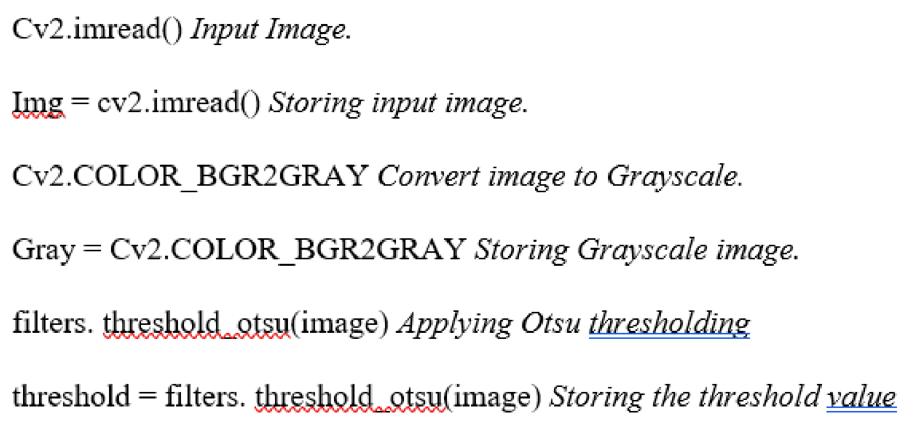
Thresholding is a technique used in image segmentation to manipulate the pixel arrangement of an image, making it more amenable to analysis. This process involves converting a color or grayscale image into a binary image, wherein only black and white pixels are present. By employing thresholding, we can isolate specific regions or objects of interest, simplifying further analysis and processing of the image.The basic purpose of this is to modify the image more correctly so that the image is visible to the user.

6.1.3.4 SIGNIFANCE

Otsu thresholding has a number of benefits, including:

- It is an easy and effective technique for thresholding images.
- It can deal with images that have different foreground and background intensities.
- It can handle photos with numerous histogram peaks, which can happen when there are more than two classes of pixels in an image.
- It is simple to use because it doesn't need any more input or parameters.
- As it is unaffected by image noise or minute image alterations, it is a reliable approach.
- For color photos, it can also be used to establish a thresholding value.
- Object identification, segmentation, and feature extraction are just a few of the image processing and computer vision applications that make extensive use of it.

6.1.3.5 PSESUDOCODE



```
Cv2.imread() Input Image.

Img = cv2.imread() Storing input image.

Cv2.COLOR_BGR2GRAY Convert image to Grayscale.

Gray = Cv2.COLOR_BGR2GRAY Storing Grayscale image.

filters. threshold_otsu(image) Applying Otsu thresholding

threshold = filters. threshold_otsu(image) Storing the threshold value
```

6.2 COMPARISON OF ALGORITHMS

6.2.1 Histogram Equalization:

Histogram equalization is similar to stretching algorithm but works in a different way and has more accuracy than other algorithms like:
You want to flatten the histogram into a uniform distribution while performing histogram equalization. Stretching, in contrast, allows you to adjust the complete spectrum of intensity values. similar to your actions in Normalization. Keep in mind that histogram stretching is another name for contrast stretching.

6.2.2 CLAHE (Contrast Limited Adaptive Histogram Equalization):

Histogram equalization and other techniques are available to enhance your image into a better form and these are the most widely used techniques out there but there is CLAHE an algorithm that used to enhance the image nearly to accurate colors of nature or the input picture. CLAHE (Contrast Limited Adaptive Histogram Equalization) is a modified version of Adaptive Histogram Equalization (AHE) that tackles the problem of excessive contrast enhancement. Unlike AHE, which processes the entire image, CLAHE operates on distinct sections called tiles. By applying bilinear interpolation to combine neighboring tiles, CLAHE effectively removes artificial borders, resulting in a more balanced and visually pleasing contrast enhancement.

6.2.3 Otsu Thresholding:

Otsu's method, like all other global thresholding techniques, performs poorly in the presence of loud noise, tiny object size, uneven lighting, and greater intra-class than inter-class variance. Local Otsu technique adaptations have been created for those circumstances.
A further benefit of Otsu's approach is that its mathematical foundation models the image's histogram as a combination of two Normal distributions with equal variance and size.
However, Otsu's thresholding may still produce satisfactory results even if these conditions are not met, much like how statistical tests—to which Otsu's method is closely related—can function properly even if the underlying assumptions are not entirely met.

6.3 COMPLEXITIES OF ALGORITHMS

ALGORITHM	**TIME COMPLEXITY** **(Best,worst,Average)**	**SPACE COMPLEXITY** **(Best,worst,Average)**
Histogram Equalization	O(N) Whereas, N is no. of pixels in the image	O(K) Whereas, K is no. of intensity levels in the image
CLAHE	O(N) Whereas, N is no. of pixels in the image	O(N) Whereas, N is no. of pixels in the image
Otsu Thresholding	O(N) Whereas, N is no. of pixels in the image	O(K) Whereas, K is no. of intensity levels in the image

Table 8.1: Complexities of Algorithms

6.4 REFERENCES

https://www.ibm.com/cloud/blog/optical-character-recognition

https://www.mygreatlearning.com/blog/histogram-equalization-explained/

https://towardsdatascience.com/histogram-equalization-5d1013626e64

https://learnopencv.com/otsu-thresholding-with-opencv/

https://www.geeksforgeeks.org/clahe-histogram-eqalization-opencv/

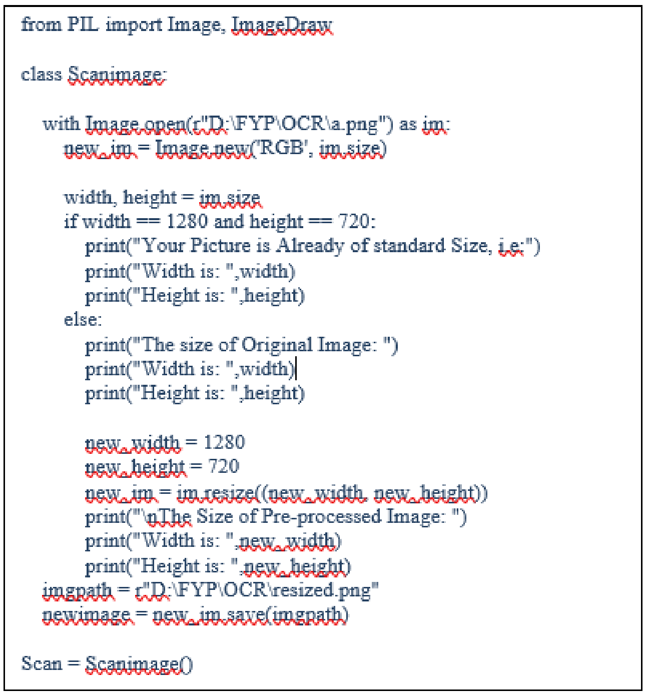
https://en.wikipedia.org/wiki/Otsu%27s_method#:~:text=Otsu's%20method%20performs%20well%20when,class%20than%20inter%2Dclass%20variance.

7. IMPLEMENTATION

7.1 CODE

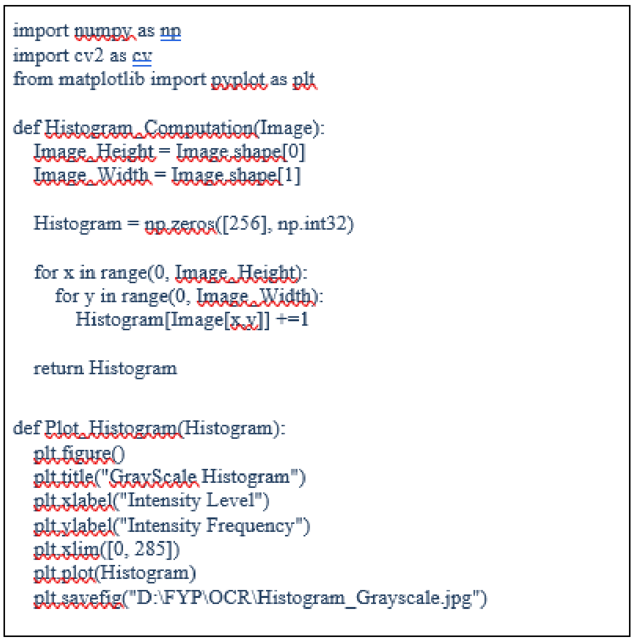
7.1.1 PREPROCESSING OF THE IMAGE

Here, first we scan the image from the dataset through giving the path of the image and then resizing the image to the standard size (i.e. 1280 X 720).

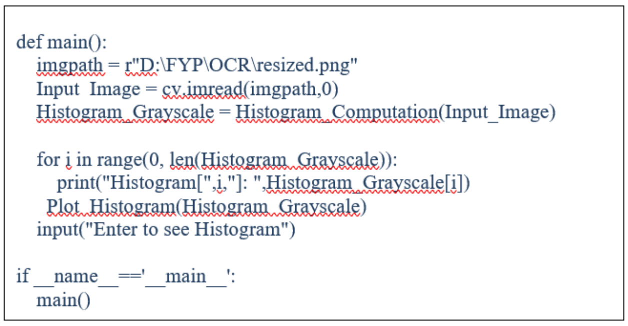


```
from PIL import Image, ImageDraw

class Scanimage:

    with Image.open(r"D:\FYP\OCR\a.png") as im:
        new_im = Image.new('RGB', im.size)

        width, height = im.size
        if width == 1280 and height == 720:
            print("Your Picture is Already of standard Size, i.e:")
            print("Width is: ",width)
            print("Height is: ",height)
        else:
            print("The size of Original Image: ")
            print("Width is: ",width)
            print("Height is: ",height)

            new_width = 1280
            new_height = 720
            new_im = im.resize((new_width, new_height))
            print("\nThe Size of Pre-processed Image: ")
            print("Width is: ",new_width)
            print("Height is: ",new_height)
    imgpath = r"D:\FYP\OCR\resized.png"
    newimage = new_im.save(imgpath)

Scan = Scanimage()
```

We convert the image to grayscale and plot a histogram of the standard sized image. This is to check the intensity level of the image. The level of brightness and darkness of the image.

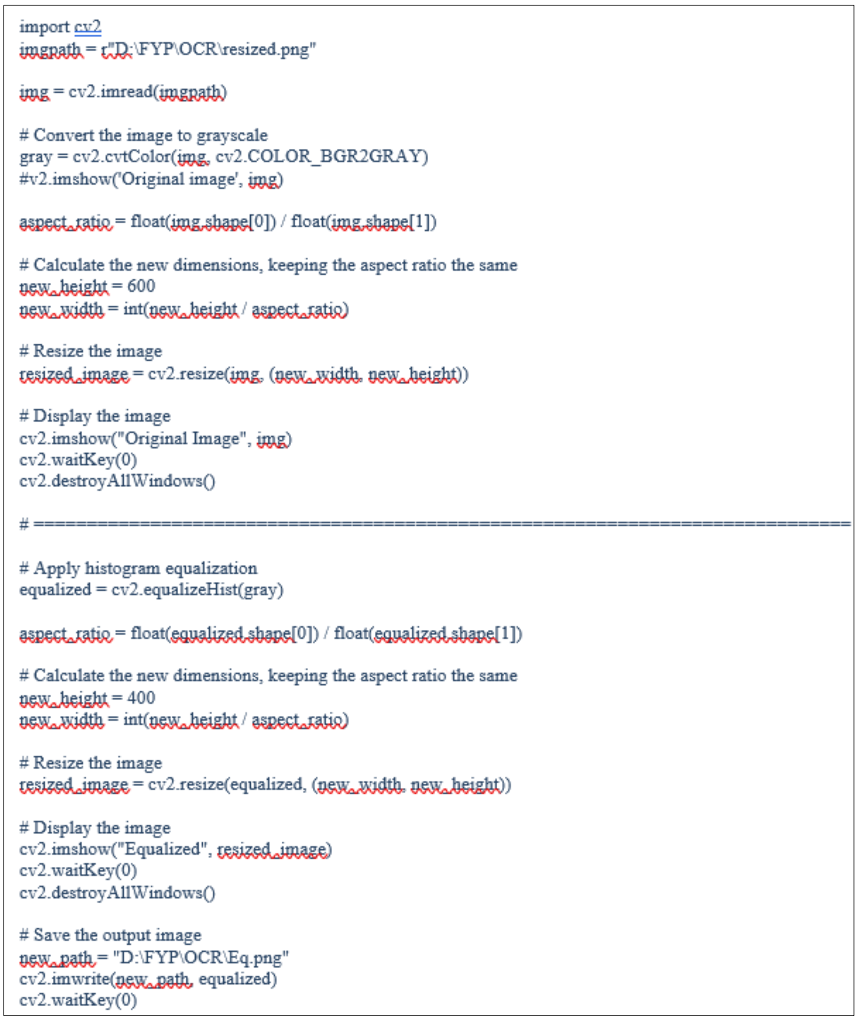


```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

def Histogram_Computation(Image):
    Image_Height = Image.shape[0]
    Image_Width = Image.shape[1]

    Histogram = np.zeros([256], np.int32)

    for x in range(0, Image_Height):
        for y in range(0, Image_Width):
            Histogram[Image[x,y]] +=1

    return Histogram


def Plot_Histogram(Histogram):
    plt.figure()
    plt.title("GrayScale Histogram")
    plt.xlabel("Intensity Level")
    plt.ylabel("Intensity Frequency")
    plt.xlim([0, 285])
    plt.plot(Histogram)
    plt.savefig("D:\FYP\OCR\Histogram_Grayscale.jpg")
```



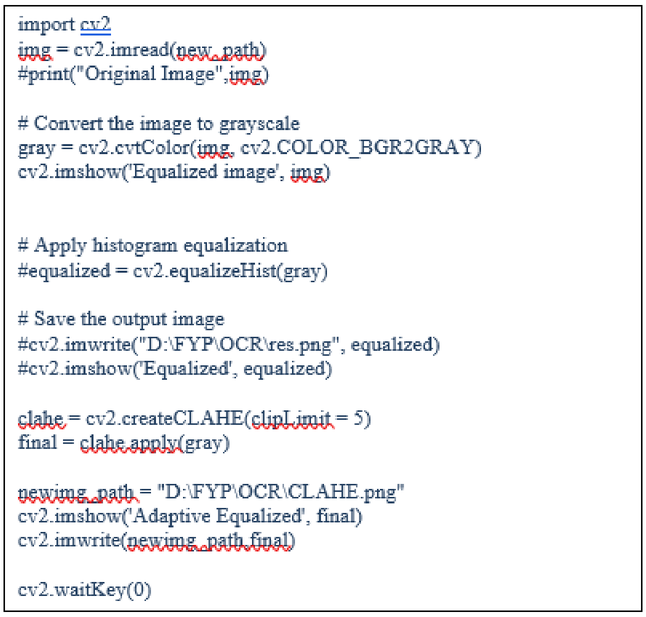
```
def main():
   imgpath = r"D:\FYP\OCR\resized.png"
   Input_Image = cv.imread(imgpath,0)
   Histogram_Grayscale = Histogram_Computation(Input_Image)

   for i in range(0, len(Histogram_Grayscale)):
      print("Histogram[",i,"]: ",Histogram_Grayscale[i])
    Plot_Histogram(Histogram_Grayscale)
   input("Enter to see Histogram")

if __name__=='__main__':
   main()
```

To fix the intensity of the image we applied an algorithm of equalizing the histogram.



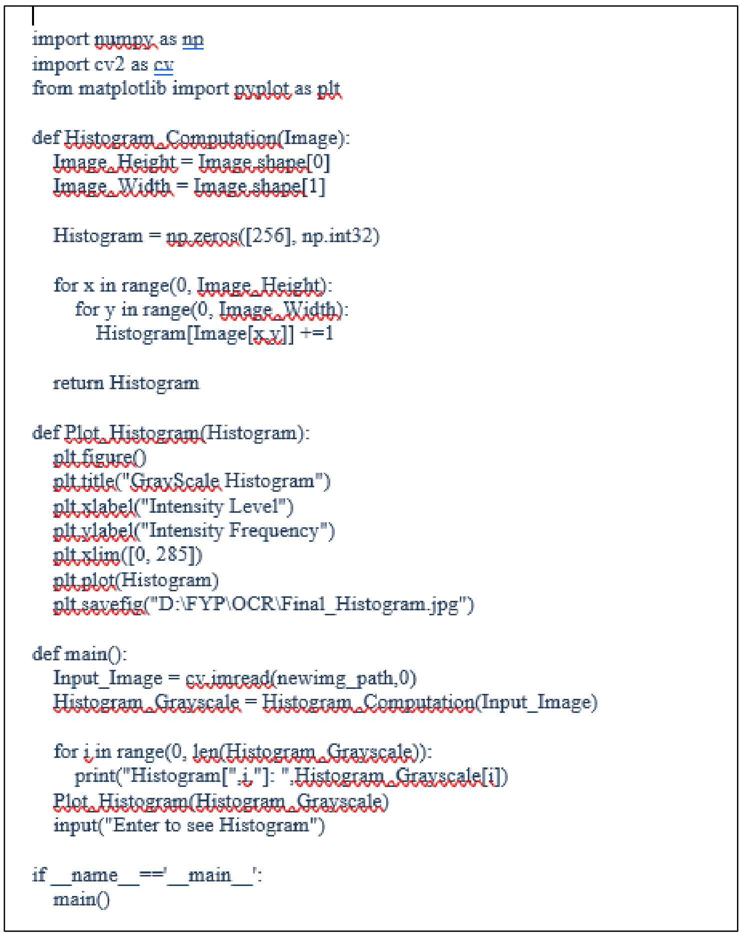
```
import cv2
imgpath = r"D:\FYP\OCR\resized.png"

img = cv2.imread(imgpath)

# Convert the image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#v2.imshow('Original image', img)

aspect_ratio = float(img.shape[0]) / float(img.shape[1])

# Calculate the new dimensions, keeping the aspect ratio the same
new_height = 600
new_width = int(new_height / aspect_ratio)

# Resize the image
resized_image = cv2.resize(img, (new_width, new_height))

# Display the image
cv2.imshow("Original Image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()

# ==========================================================================

# Apply histogram equalization
equalized = cv2.equalizeHist(gray)

aspect_ratio = float(equalized.shape[0]) / float(equalized.shape[1])

# Calculate the new dimensions, keeping the aspect ratio the same
new_height = 400
new_width = int(new_height / aspect_ratio)

# Resize the image
resized_image = cv2.resize(equalized, (new_width, new_height))

# Display the image
cv2.imshow("Equalized", resized_image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Save the output image
new_path = "D:\FYP\OCR\Eq.png"
cv2.imwrite(new_path, equalized)
cv2.waitKey(0)
```

Our result is not very clear yet, so we are trying another technique for the equalization of the histogram which is called CLAHE (contrast Limited Adaptive Equalization).



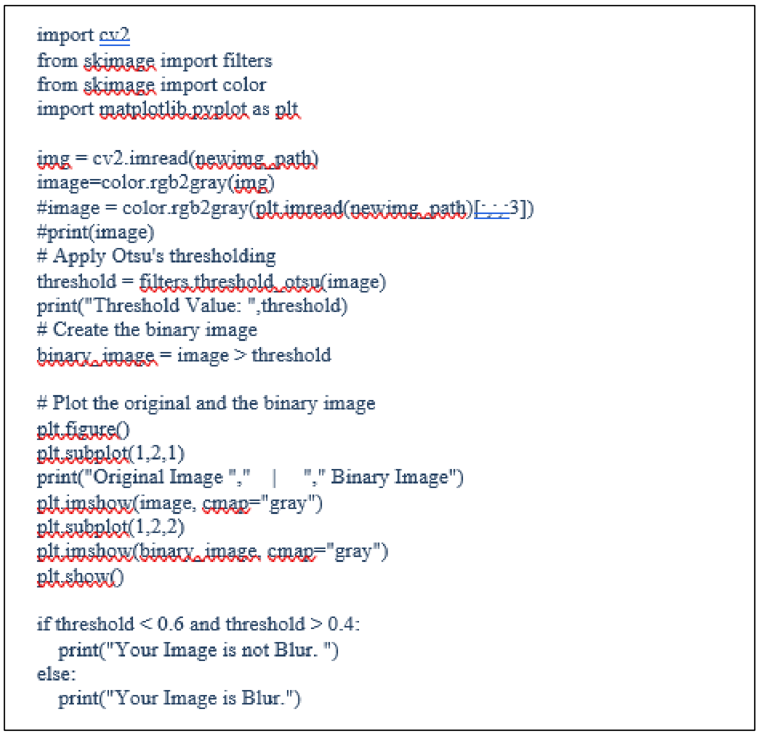
```
import cv2
img = cv2.imread(new_path)
#print("Original Image",img)

# Convert the image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow('Equalized image', img)


# Apply histogram equalization
#equalized = cv2.equalizeHist(gray)

# Save the output image
#cv2.imwrite("D:\FYP\OCR\res.png", equalized)
#cv2.imshow('Equalized', equalized)

clahe = cv2.createCLAHE(clipLimit = 5)
final = clahe.apply(gray)

newimg_path = "D:\FYP\OCR\CLAHE.png"
cv2.imshow('Adaptive Equalized', final)
cv2.imwrite(newimg_path,final)

cv2.waitKey(0)
```

Now to verify our result we will plot histogram once more to double check our image if any changes being made after applying the algorithm to our image.



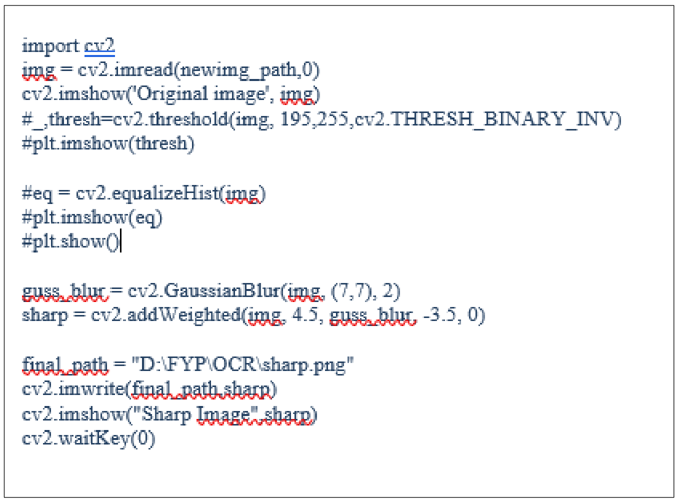
```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

def Histogram_Computation(Image):
    Image_Height = Image.shape[0]
    Image_Width = Image.shape[1]

    Histogram = np.zeros([256], np.int32)

    for x in range(0, Image_Height):
        for y in range(0, Image_Width):
            Histogram[Image[x,y]] +=1

    return Histogram

def Plot_Histogram(Histogram):
    plt.figure()
    plt.title("GrayScale Histogram")
    plt.xlabel("Intensity Level")
    plt.ylabel("Intensity Frequency")
    plt.xlim([0, 285])
    plt.plot(Histogram)
    plt.savefig("D:\FYP\OCR\Final_Histogram.jpg")

def main():
    Input_Image = cv.imread(newimg_path,0)
    Histogram_Grayscale = Histogram_Computation(Input_Image)

    for i in range(0, len(Histogram_Grayscale)):
        print("Histogram[",i,"]: ",Histogram_Grayscale[i])
    Plot_Histogram(Histogram_Grayscale)
    input("Enter to see Histogram")

if __name__=='__main__':
    main()
```

Now to make the image clearer and to check for the blur intensity into the image. We are going with the Otsu thresholding Method.



```
import cv2
from skimage import filters
from skimage import color
import matplotlib.pyplot as plt

img = cv2.imread(newimg_path)
image=color.rgb2gray(img)
#image = color.rgb2gray(plt.imread(newimg_path)[:,:,:3])
#print(image)
# Apply Otsu's thresholding
threshold = filters.threshold_otsu(image)
print("Threshold Value: ",threshold)
# Create the binary image
binary_image = image > threshold

# Plot the original and the binary image
plt.figure()
plt.subplot(1,2,1)
print("Original Image ","   |    "," Binary Image")
plt.imshow(image, cmap="gray")
plt.subplot(1,2,2)
plt.imshow(binary_image, cmap="gray")
plt.show()

if threshold < 0.6 and threshold > 0.4:
   print("Your Image is not Blur. ")
else:
   print("Your Image is Blur.")
```

The Result is our image is blur, so to cater this issue we try to maintain the thresholding value of the image.

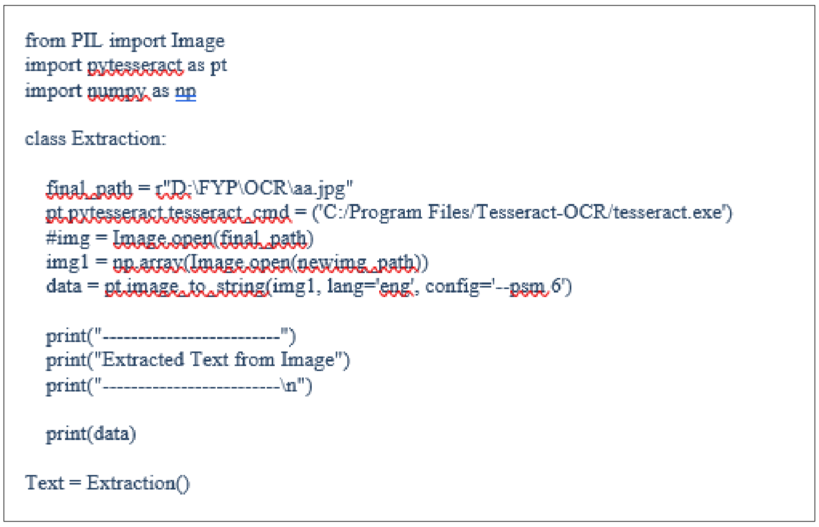


```
import cv2
img = cv2.imread(newimg_path,0)
cv2.imshow('Original image', img)
#_,thresh=cv2.threshold(img, 195,255,cv2.THRESH_BINARY_INV)
#plt.imshow(thresh)

#eq = cv2.equalizeHist(img)
#plt.imshow(eq)
#plt.show()

guss_blur = cv2.GaussianBlur(img, (7,7), 2)
sharp = cv2.addWeighted(img, 4.5, guss_blur, -3.5, 0)

final_path = "D:\FYP\OCR\sharp.png"
cv2.imwrite(final_path,sharp)
cv2.imshow("Sharp Image",sharp)
cv2.waitKey(0)
```

7.1.2 EXTRACTION OF TEXT FROM IMAGE

Now our image is done with preprocessing and ready to text extraction phase.

We are using a module of python Pytesseract.

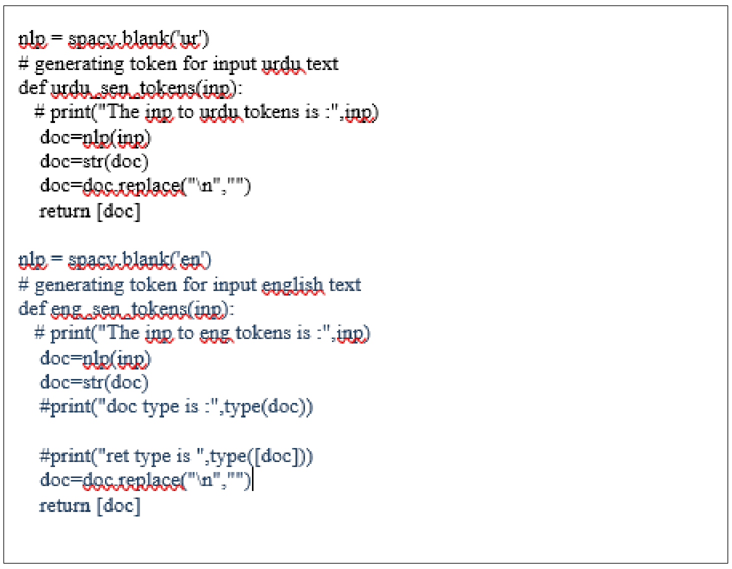


```
from PIL import Image
import pytesseract as pt
import numpy as np

class Extraction:

    final_path = r"D:\FYP\OCR\aa.jpg"
    pt.pytesseract.tesseract_cmd = ('C:/Program Files/Tesseract-OCR/tesseract.exe')
    #img = Image.open(final_path)
    img1 = np.array(Image.open(newimg_path))
    data = pt.image_to_string(img1, lang='eng', config='--psm 6')

    print("------------------------")
    print("Extracted Text from Image")
    print("------------------------\n")

    print(data)

Text = Extraction()
```

f

7.1.3 TOKENIZATION

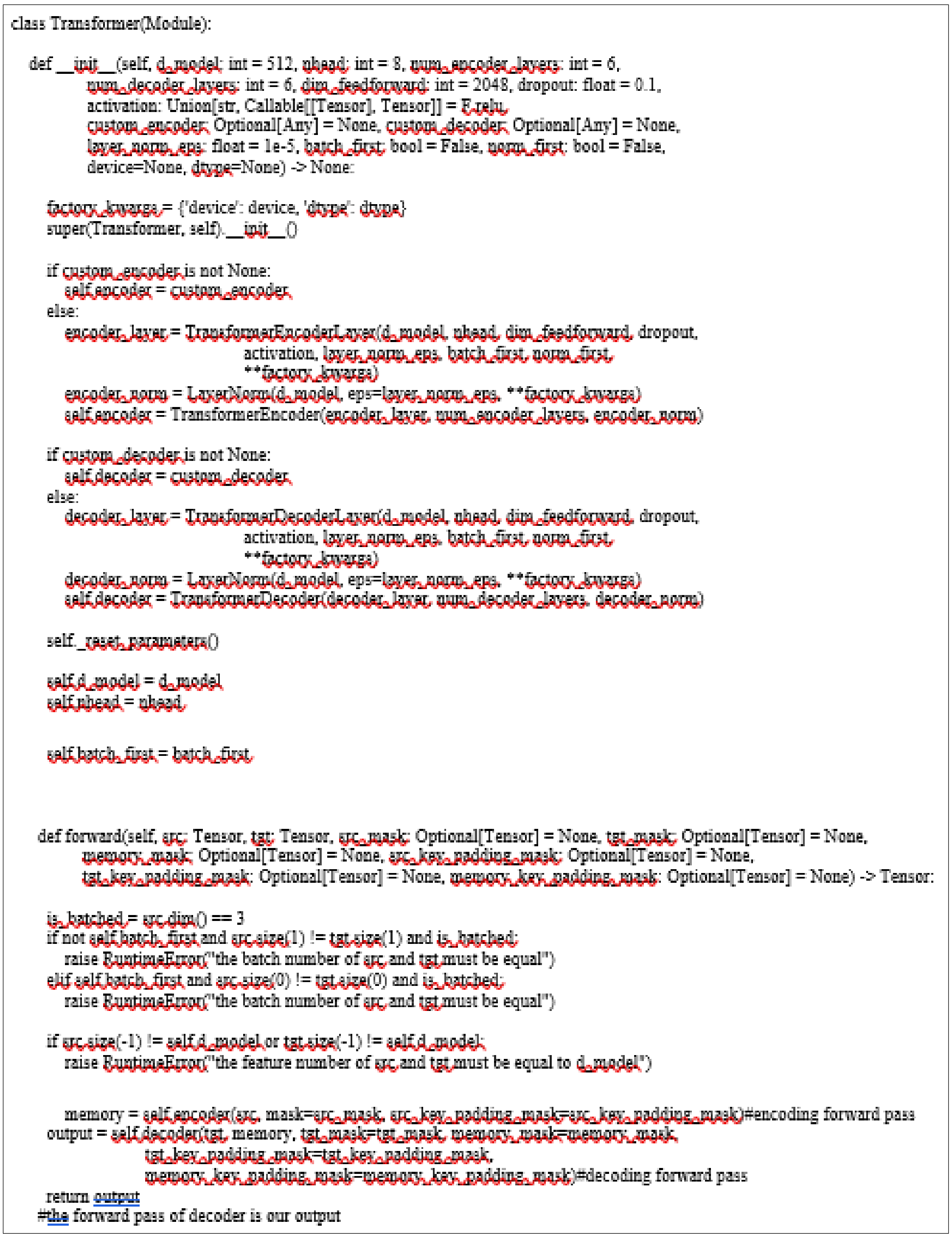
Now we have the Text, so first we make tokens for the NLP.



```
nlp = spacy.blank('ur')
# generating token for input urdu text
def urdu_sen_tokens(inp):
  # print("The inp to urdu tokens is :",inp)
   doc=nlp(inp)
   doc=str(doc)
   doc=doc.replace("\n","")
   return [doc]

nlp = spacy.blank('en')
# generating token for input english text
def eng_sen_tokens(inp):
  # print("The inp to eng tokens is :",inp)
   doc=nlp(inp)
   doc=str(doc)
   #print("doc type is :",type(doc))

   #print("ret type is ",type([doc]))
   doc=doc.replace("\n","")
   return [doc]
```

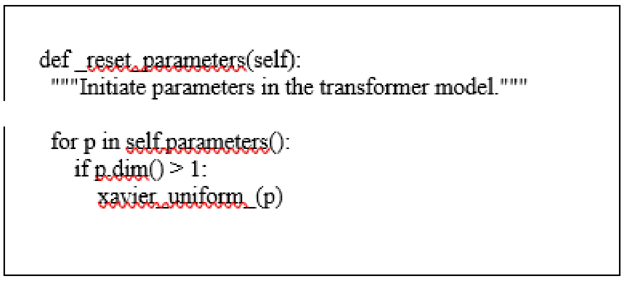
7.1.4 TRANSFORMER MODULE

Now we build the transformer for the translation and training the dataset.



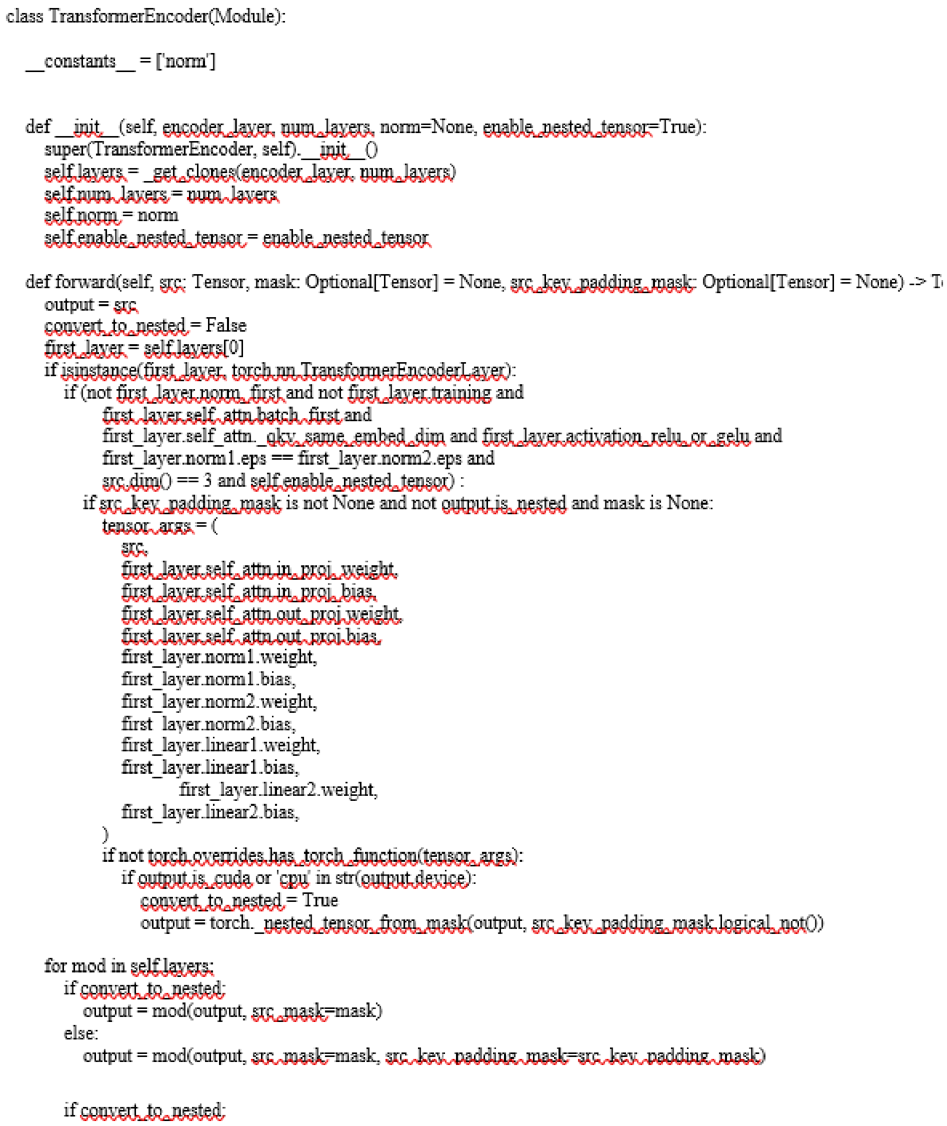
```
class Transformer(Module):

  def __init__(self, d_model: int = 512, nhead: int = 8, num_encoder_layers: int = 6,
             num_decoder_layers: int = 6, dim_feedforward: int = 2048, dropout: float = 0.1,
             activation: Union[str, Callable[[Tensor], Tensor]] = F.relu,
             custom_encoder: Optional[Any] = None, custom_decoder: Optional[Any] = None,
             layer_norm_eps: float = 1e-5, batch_first: bool = False, norm_first: bool = False,
             device=None, dtype=None) -> None:

    factory_kwargs = {'device': device, 'dtype': dtype}
    super(Transformer, self).__init__()

    if custom_encoder is not None:
      self.encoder = custom_encoder
    else:
      encoder_layer = TransformerEncoderLayer(d_model, nhead, dim_feedforward, dropout,
                                activation, layer_norm_eps, batch_first, norm_first,
                                **factory_kwargs)
      encoder_norm = LayerNorm(d_model, eps=layer_norm_eps, **factory_kwargs)
      self.encoder = TransformerEncoder(encoder_layer, num_encoder_layers, encoder_norm)

    if custom_decoder is not None:
      self.decoder = custom_decoder
    else:
      decoder_layer = TransformerDecoderLayer(d_model, nhead, dim_feedforward, dropout,
                                activation, layer_norm_eps, batch_first, norm_first,
                                **factory_kwargs)
      decoder_norm = LayerNorm(d_model, eps=layer_norm_eps, **factory_kwargs)
      self.decoder = TransformerDecoder(decoder_layer, num_decoder_layers, decoder_norm)

    self._reset_parameters()

    self.d_model = d_model
    self.nhead = nhead


    self.batch_first = batch_first



  def forward(self, src: Tensor, tgt: Tensor, src_mask: Optional[Tensor] = None, tgt_mask: Optional[Tensor] = None,
          memory_mask: Optional[Tensor] = None, src_key_padding_mask: Optional[Tensor] = None,
          tgt_key_padding_mask: Optional[Tensor] = None, memory_key_padding_mask: Optional[Tensor] = None) -> Tensor:

    is_batched = src.dim() == 3
    if not self.batch_first and src.size(1) != tgt.size(1) and is_batched:
      raise RuntimeError("the batch number of src and tgt must be equal")
    elif self.batch_first and src.size(0) != tgt.size(0) and is_batched:
      raise RuntimeError("the batch number of src and tgt must be equal")

    if src.size(-1) != self.d_model or tgt.size(-1) != self.d_model:
      raise RuntimeError("the feature number of src and tgt must be equal to d_model")


      memory = self.encoder(src, mask=src_mask, src_key_padding_mask=src_key_padding_mask)#encoding forward pass
    output = self.decoder(tgt, memory, tgt_mask=tgt_mask, memory_mask=memory_mask,
                  tgt_key_padding_mask=tgt_key_padding_mask,
                  memory_key_padding_mask=memory_key_padding_mask)#decoding forward pass
    return output
  #the forward pass of decoder is our output
```

```
@staticmethod
def generate_square_subsequent_mask(sz: int) -> Tensor:
    return torch.triu(torch.full((sz, sz), float('-inf')), diagonal=1)
```



```
def _reset_parameters(self):
  """Initiate parameters in the transformer model."""

  for p in self.parameters():
     if p.dim() > 1:
        xavier_uniform_(p)
```

7.1.5 ENCODING IN TRANSFORMER MODULE

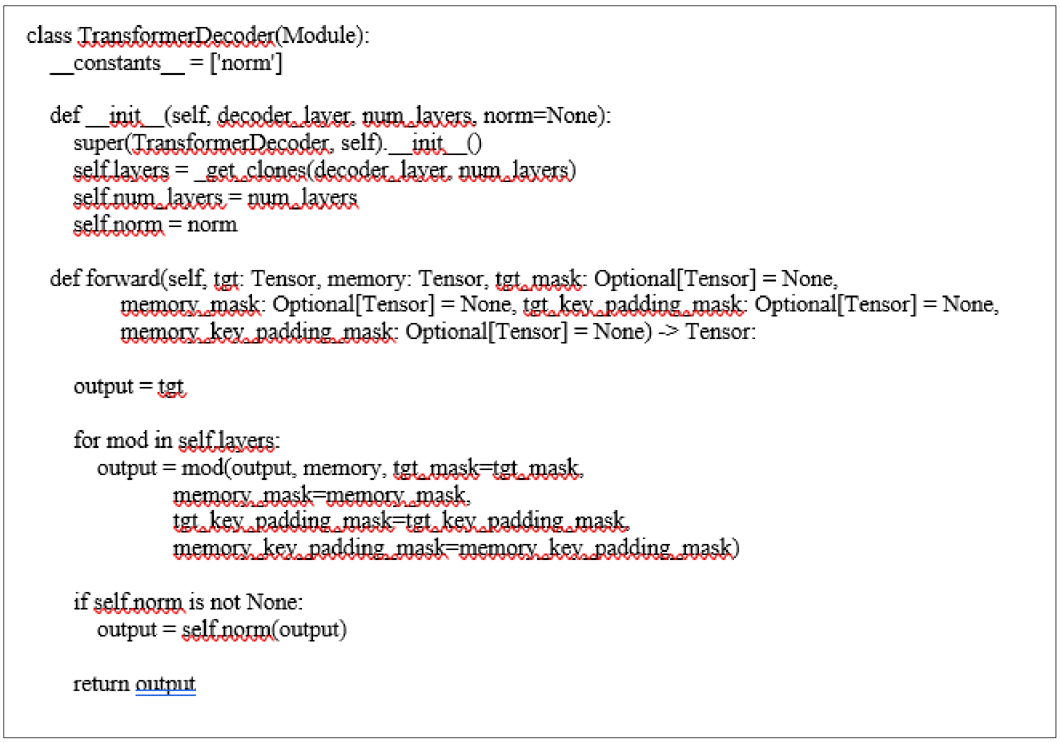


```
class TransformerEncoder(Module):

   __constants__ = ['norm']


   def __init__(self, encoder_layer, num_layers, norm=None, enable_nested_tensor=True):
      super(TransformerEncoder, self).__init__()
      self.layers = _get_clones(encoder_layer, num_layers)
      self.num_layers = num_layers
      self.norm = norm
      self.enable_nested_tensor = enable_nested_tensor

   def forward(self, src: Tensor, mask: Optional[Tensor] = None, src_key_padding_mask: Optional[Tensor] = None) -> T
      output = src
      convert_to_nested = False
      first_layer = self.layers[0]
      if isinstance(first_layer, torch.nn.TransformerEncoderLayer):
         if (not first_layer.norm_first and not first_layer.training and
               first_layer.self_attn.batch_first and
               first_layer.self_attn._qkv_same_embed_dim and first_layer.activation_relu_or_gelu and
               first_layer.norm1.eps == first_layer.norm2.eps and
               src.dim() == 3 and self.enable_nested_tensor) :
            if src_key_padding_mask is not None and not output.is_nested and mask is None:
               tensor_args = (
                  src,
                  first_layer.self_attn.in_proj_weight,
                  first_layer.self_attn.in_proj_bias,
                  first_layer.self_attn.out_proj.weight,
                  first_layer.self_attn.out_proj.bias,
                  first_layer.norm1.weight,
                  first_layer.norm1.bias,
                  first_layer.norm2.weight,
                  first_layer.norm2.bias,
                  first_layer.linear1.weight,
                  first_layer.linear1.bias,
                        first_layer.linear2.weight,
                  first_layer.linear2.bias,
               )
               if not torch.overrides.has_torch_function(tensor_args):
                  if output.is_cuda or 'cpu' in str(output.device):
                     convert_to_nested = True
                     output = torch._nested_tensor_from_mask(output, src_key_padding_mask.logical_not())

      for mod in self.layers:
         if convert_to_nested:
            output = mod(output, src_mask=mask)
         else:
            output = mod(output, src_mask=mask, src_key_padding_mask=src_key_padding_mask)


         if convert_to_nested:
```

```
                                                            output =
output.to_padded_tensor(0.)

    if self.norm is not None:
            output = self.norm(output)

        return output
```

7.1.6 TRANSFORMER DECODER MODULE

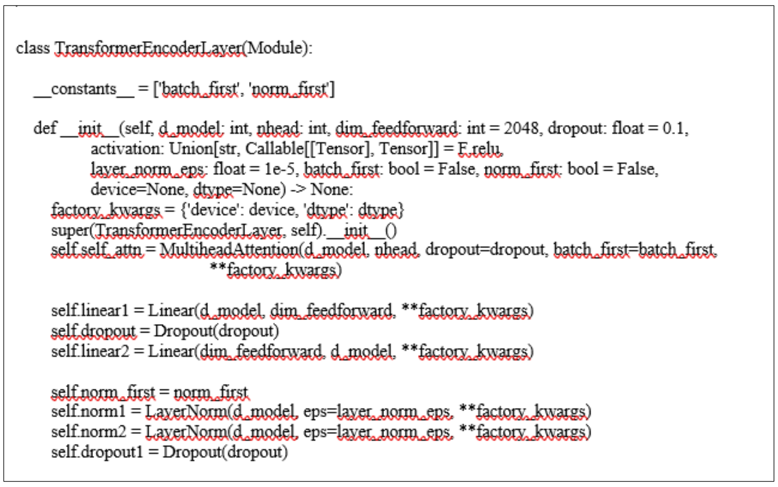
Now we creating transformer module for decoder.
For Decoding the encoding data.



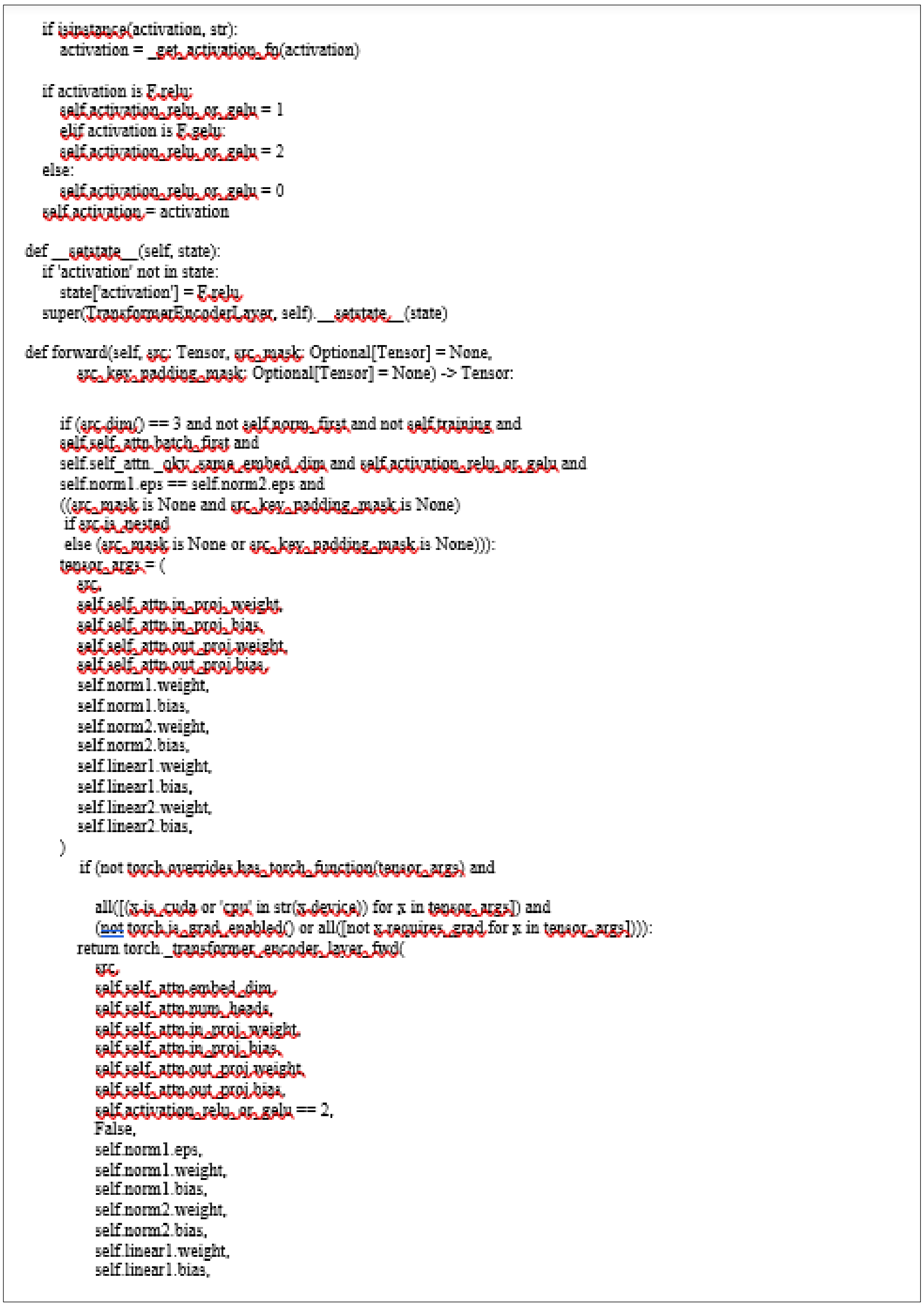
```
class TransformerDecoder(Module):
    __constants__ = ['norm']

    def __init__(self, decoder_layer, num_layers, norm=None):
        super(TransformerDecoder, self).__init__()
        self.layers = _get_clones(decoder_layer, num_layers)
        self.num_layers = num_layers
        self.norm = norm

    def forward(self, tgt: Tensor, memory: Tensor, tgt_mask: Optional[Tensor] = None,
                memory_mask: Optional[Tensor] = None, tgt_key_padding_mask: Optional[Tensor] = None,
                memory_key_padding_mask: Optional[Tensor] = None) -> Tensor:

        output = tgt

        for mod in self.layers:
            output = mod(output, memory, tgt_mask=tgt_mask,
                         memory_mask=memory_mask,
                         tgt_key_padding_mask=tgt_key_padding_mask,
                         memory_key_padding_mask=memory_key_padding_mask)

        if self.norm is not None:
            output = self.norm(output)

        return output
```

7.1.7 ENCODER LAYERS OF TRANSFORMER

This is the Encoding layers module of the Transformers.



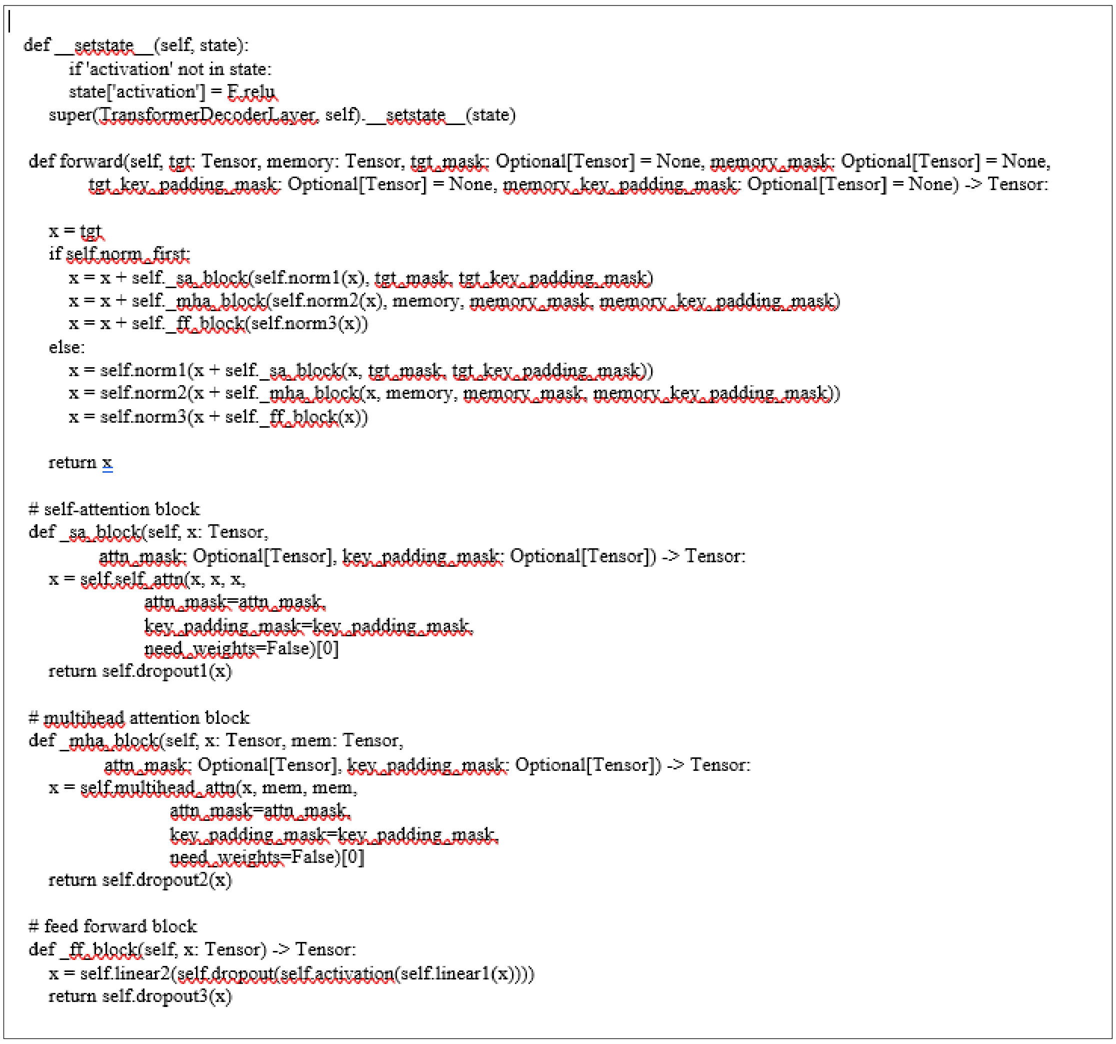
```
class TransformerEncoderLayer(Module):

    __constants__ = ['batch_first', 'norm_first']

    def __init__(self, d_model: int, nhead: int, dim_feedforward: int = 2048, dropout: float = 0.1,
                 activation: Union[str, Callable[[Tensor], Tensor]] = F.relu,
                 layer_norm_eps: float = 1e-5, batch_first: bool = False, norm_first: bool = False,
                 device=None, dtype=None) -> None:
        factory_kwargs = {'device': device, 'dtype': dtype}
        super(TransformerEncoderLayer, self).__init__()
        self.self_attn = MultiheadAttention(d_model, nhead, dropout=dropout, batch_first=batch_first,
                                            **factory_kwargs)

        self.linear1 = Linear(d_model, dim_feedforward, **factory_kwargs)
        self.dropout = Dropout(dropout)
        self.linear2 = Linear(dim_feedforward, d_model, **factory_kwargs)

        self.norm_first = norm_first
        self.norm1 = LayerNorm(d_model, eps=layer_norm_eps, **factory_kwargs)
        self.norm2 = LayerNorm(d_model, eps=layer_norm_eps, **factory_kwargs)
        self.dropout1 = Dropout(dropout)
self.dropout2 = Dropout(dropout)
```



```
    if isinstance(activation, str):
        activation = _get_activation_fn(activation)

    if activation is F.relu:
        self.activation_relu_or_gelu = 1
        elif activation is F.gelu:
        self.activation_relu_or_gelu = 2
    else:
        self.activation_relu_or_gelu = 0
    self.activation = activation

def __setstate__(self, state):
    if 'activation' not in state:
        state['activation'] = F.relu
    super(TransformerEncoderLayer, self).__setstate__(state)

def forward(self, src: Tensor, src_mask: Optional[Tensor] = None,
            src_key_padding_mask: Optional[Tensor] = None) -> Tensor:


        if (src.dim() == 3 and not self.norm_first and not self.training and
        self.self_attn.batch_first and
        self.self_attn._qkv_same_embed_dim and self.activation_relu_or_gelu and
        self.norm1.eps == self.norm2.eps and
        ((src_mask is None and src_key_padding_mask is None)
         if src.is_nested
         else (src_mask is None or src_key_padding_mask is None))):
        tensor_args = (
            src,
            self.self_attn.in_proj_weight,
            self.self_attn.in_proj_bias,
            self.self_attn.out_proj.weight,
            self.self_attn.out_proj.bias,
            self.norm1.weight,
            self.norm1.bias,
            self.norm2.weight,
            self.norm2.bias,
            self.linear1.weight,
            self.linear1.bias,
            self.linear2.weight,
            self.linear2.bias,
        )
            if (not torch.overrides.has_torch_function(tensor_args) and

                all([(x.is_cuda or 'cpu' in str(x.device)) for x in tensor_args]) and
                (not torch.is_grad_enabled() or all([not x.requires_grad for x in tensor_args]))):
            return torch._transformer_encoder_layer_fwd(
                src,
                self.self_attn.embed_dim,
                self.self_attn.num_heads,
                self.self_attn.in_proj_weight,
                self.self_attn.in_proj_bias,
                self.self_attn.out_proj.weight,
                self.self_attn.out_proj.bias,
                self.activation_relu_or_gelu == 2,
                False,
                self.norm1.eps,
                self.norm1.weight,
                self.norm1.bias,
                self.norm2.weight,
                self.norm2.bias,
                self.linear1.weight,
                self.linear1.bias,
                    self.linear2.weight,
                self.linear2.bias,
                src_mask if src_mask is not None else src_key_padding_mask,
            )
        x = src
        if self.norm_first:
            x = x + self._sa_block(self.norm1(x), src_mask, src_key_padding_mask)
            x = x + self._ff_block(self.norm2(x))
        else:
            x = self.norm1(x + self._sa_block(x, src_mask, src_key_padding_mask))
            x = self.norm2(x + self._ff_block(x))

        return x


    def _sa_block(self, x: Tensor,
                  attn_mask: Optional[Tensor], key_padding_mask: Optional[Tensor]) -> Tensor:
        x = self.self_attn(x, x, x,
                           attn_mask=attn_mask,
                           key_padding_mask=key_padding_mask,
                           need_weights=False)[0]
        return self.dropout1(x)


    def _ff_block(self, x: Tensor) -> Tensor:
        x = self.linear2(self.dropout(self.activation(self.linear1(x))))
        return self.dropout2(x)
```

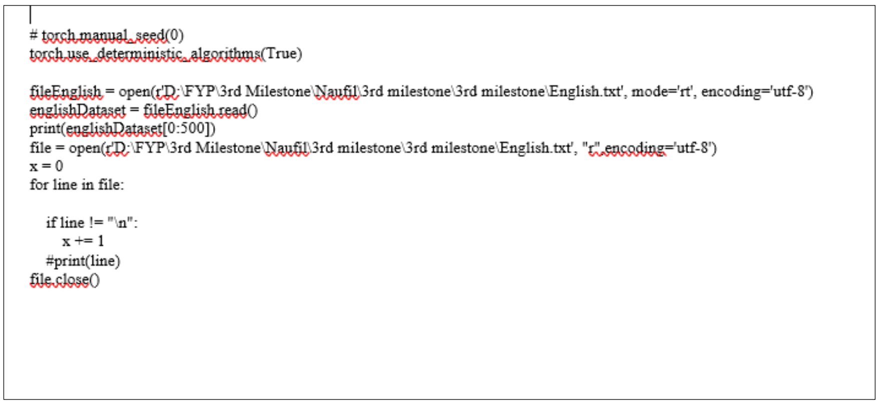
7.1.8 DECODER LAYERS OF TRANSFORMER

```
class TransformerDecoderLayer(Module):
    __constants__ = ['batch_first', 'norm_first']

    def __init__(self, d_model: int, nhead: int, dim_feedforward: int = 2048, dropout: float = 0.1,
                 activation: Union[str, Callable[[Tensor], Tensor]] = F.relu,
                 layer_norm_eps: float = 1e-5, batch_first: bool = False, norm_first: bool = False,
                 device=None, dtype=None) -> None:
        factory_kwargs = {'device': device, 'dtype': dtype}
        super(TransformerDecoderLayer, self).__init__()
        self.self_attn = MultiheadAttention(d_model, nhead, dropout=dropout, batch_first=batch_first,
                                            **factory_kwargs)
        self.multihead_attn = MultiheadAttention(d_model, nhead, dropout=dropout, batch_first=batch_first,
                                                 **factory_kwargs)

        self.linear1 = Linear(d_model, dim_feedforward, **factory_kwargs)
        self.dropout = Dropout(dropout)
        self.linear2 = Linear(dim_feedforward, d_model, **factory_kwargs)

        self.norm_first = norm_first
        self.norm1 = LayerNorm(d_model, eps=layer_norm_eps, **factory_kwargs)
        self.norm2 = LayerNorm(d_model, eps=layer_norm_eps, **factory_kwargs)
        self.norm3 = LayerNorm(d_model, eps=layer_norm_eps, **factory_kwargs)
            self.dropout1 = Dropout(dropout)
        self.dropout2 = Dropout(dropout)
        self.dropout3 = Dropout(dropout)


        if isinstance(activation, str):
            self.activation = _get_activation_fn(activation)
        else:
            self.activation = activation
```



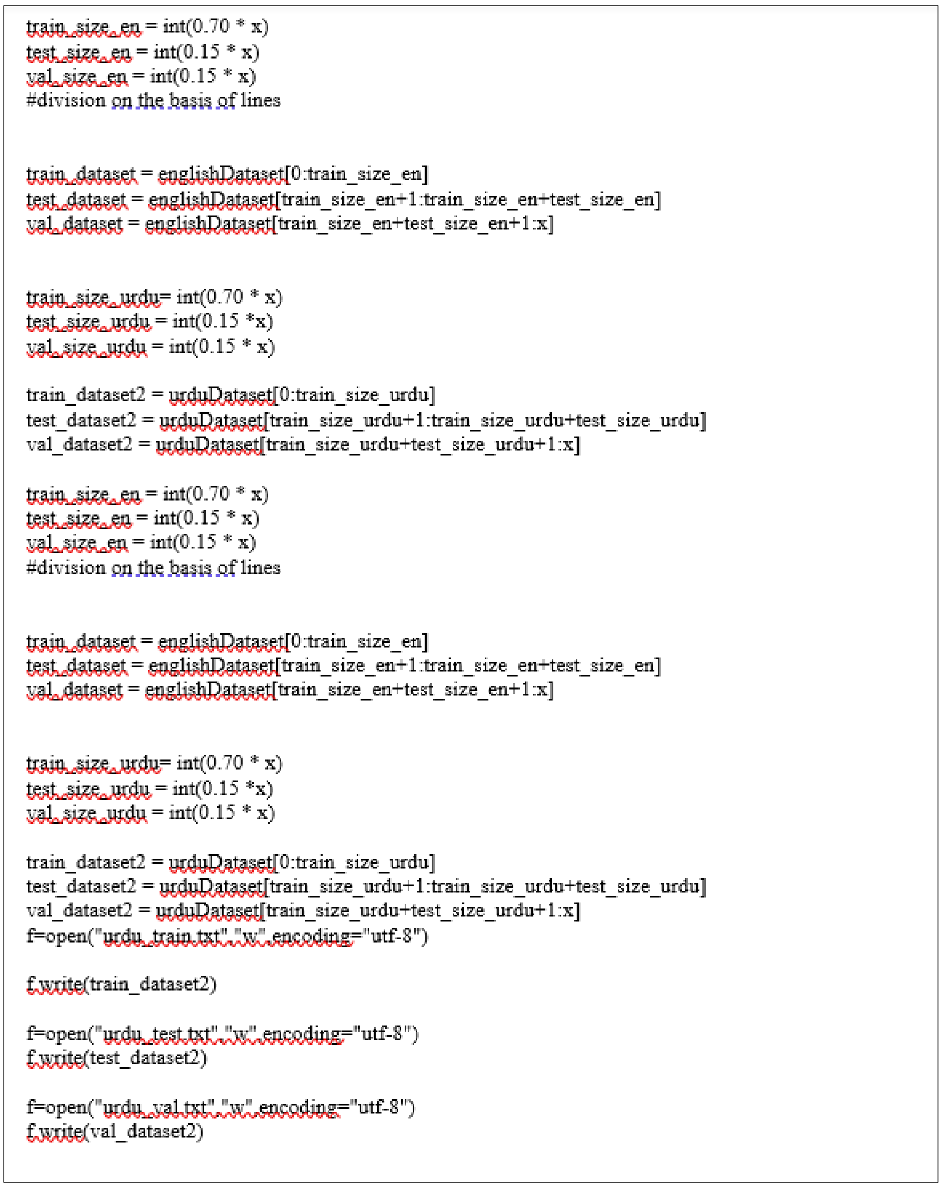
```
 def __setstate__(self, state):
        if 'activation' not in state:
        state['activation'] = F.relu
    super(TransformerDecoderLayer, self).__setstate__(state)

  def forward(self, tgt: Tensor, memory: Tensor, tgt_mask: Optional[Tensor] = None, memory_mask: Optional[Tensor] = None,
          tgt_key_padding_mask: Optional[Tensor] = None, memory_key_padding_mask: Optional[Tensor] = None) -> Tensor:

    x = tgt
    if self.norm_first:
      x = x + self._sa_block(self.norm1(x), tgt_mask, tgt_key_padding_mask)
      x = x + self._mha_block(self.norm2(x), memory, memory_mask, memory_key_padding_mask)
      x = x + self._ff_block(self.norm3(x))
    else:
      x = self.norm1(x + self._sa_block(x, tgt_mask, tgt_key_padding_mask))
      x = self.norm2(x + self._mha_block(x, memory, memory_mask, memory_key_padding_mask))
      x = self.norm3(x + self._ff_block(x))

    return x

  # self-attention block
  def _sa_block(self, x: Tensor,
           attn_mask: Optional[Tensor], key_padding_mask: Optional[Tensor]) -> Tensor:
    x = self.self_attn(x, x, x,
                attn_mask=attn_mask,
                key_padding_mask=key_padding_mask,
                need_weights=False)[0]
    return self.dropout1(x)

  # multihead attention block
  def _mha_block(self, x: Tensor, mem: Tensor,
           attn_mask: Optional[Tensor], key_padding_mask: Optional[Tensor]) -> Tensor:
    x = self.multihead_attn(x, mem, mem,
                   attn_mask=attn_mask,
                   key_padding_mask=key_padding_mask,
                   need_weights=False)[0]
    return self.dropout2(x)

  # feed forward block
  def _ff_block(self, x: Tensor) -> Tensor:
    x = self.linear2(self.dropout(self.activation(self.linear1(x))))
    return self.dropout3(x)
```

7.1.9 READING THE DATASET



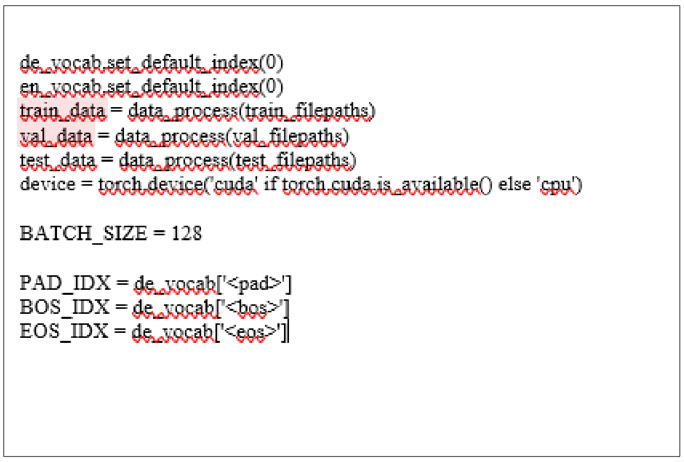
```
# torch.manual_seed(0)
torch.use_deterministic_algorithms(True)

fileEnglish = open(r'D:\FYP\3rd Milestone\Naufil\3rd milestone\3rd milestone\English.txt', mode='rt', encoding='utf-8')
englishDataset = fileEnglish.read()
print(englishDataset[0:500])
file = open(r'D:\FYP\3rd Milestone\Naufil\3rd milestone\3rd milestone\English.txt', "r", encoding='utf-8')
x = 0
for line in file:

  if line != "\n":
    x += 1
  #print(line)
file.close()
```

7.1.10 TRAINING AND TESTING SPLIT



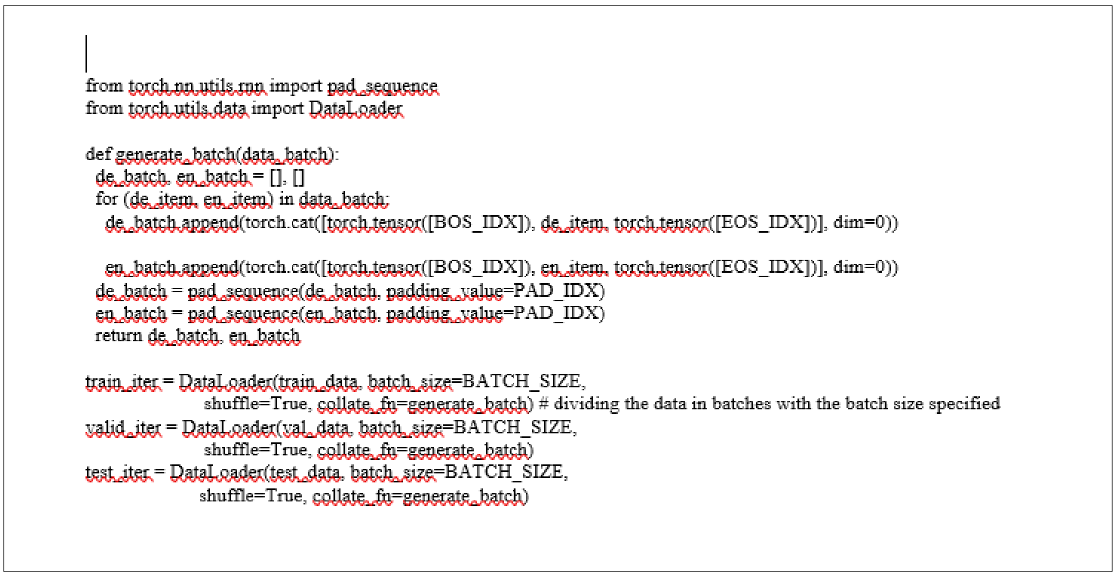
```
train_size_en = int(0.70 * x)
test_size_en = int(0.15 * x)
val_size_en = int(0.15 * x)
#division on the basis of lines


train_dataset = englishDataset[0:train_size_en]
test_dataset = englishDataset[train_size_en+1:train_size_en+test_size_en]
val_dataset = englishDataset[train_size_en+test_size_en+1:x]


train_size_urdu= int(0.70 * x)
test_size_urdu = int(0.15 *x)
val_size_urdu = int(0.15 * x)

train_dataset2 = urduDataset[0:train_size_urdu]
test_dataset2 = urduDataset[train_size_urdu+1:train_size_urdu+test_size_urdu]
val_dataset2 = urduDataset[train_size_urdu+test_size_urdu+1:x]

train_size_en = int(0.70 * x)
test_size_en = int(0.15 * x)
val_size_en = int(0.15 * x)
#division on the basis of lines


train_dataset = englishDataset[0:train_size_en]
test_dataset = englishDataset[train_size_en+1:train_size_en+test_size_en]
val_dataset = englishDataset[train_size_en+test_size_en+1:x]


train_size_urdu= int(0.70 * x)
test_size_urdu = int(0.15 *x)
val_size_urdu = int(0.15 * x)

train_dataset2 = urduDataset[0:train_size_urdu]
test_dataset2 = urduDataset[train_size_urdu+1:train_size_urdu+test_size_urdu]
val_dataset2 = urduDataset[train_size_urdu+test_size_urdu+1:x]
f=open("urdu_train.txt","w",encoding="utf-8")

f.write(train_dataset2)

f=open("urdu_test.txt","w",encoding="utf-8")
f.write(test_dataset2)

f=open("urdu_val.txt","w",encoding="utf-8")
f.write(val_dataset2)
```

7.1.11 MAPPING AND TESTING



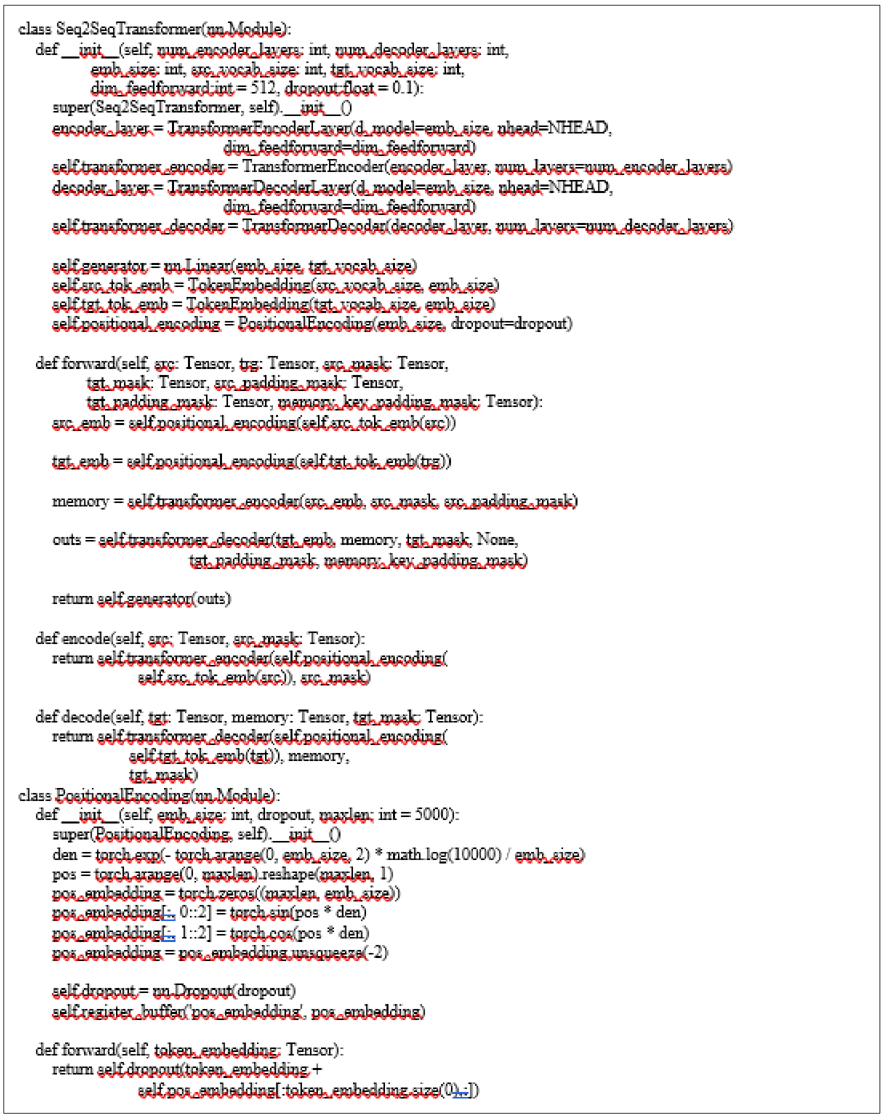
```
de_vocab.set_default_index(0)
en_vocab.set_default_index(0)
train_data = data_process(train_filepaths)
val_data = data_process(val_filepaths)
test_data = data_process(test_filepaths)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

BATCH_SIZE = 128

PAD_IDX = de_vocab['<pad>']
BOS_IDX = de_vocab['<bos>']
EOS_IDX = de_vocab['<eos>']
```

7.1.12 LOADING DATASET



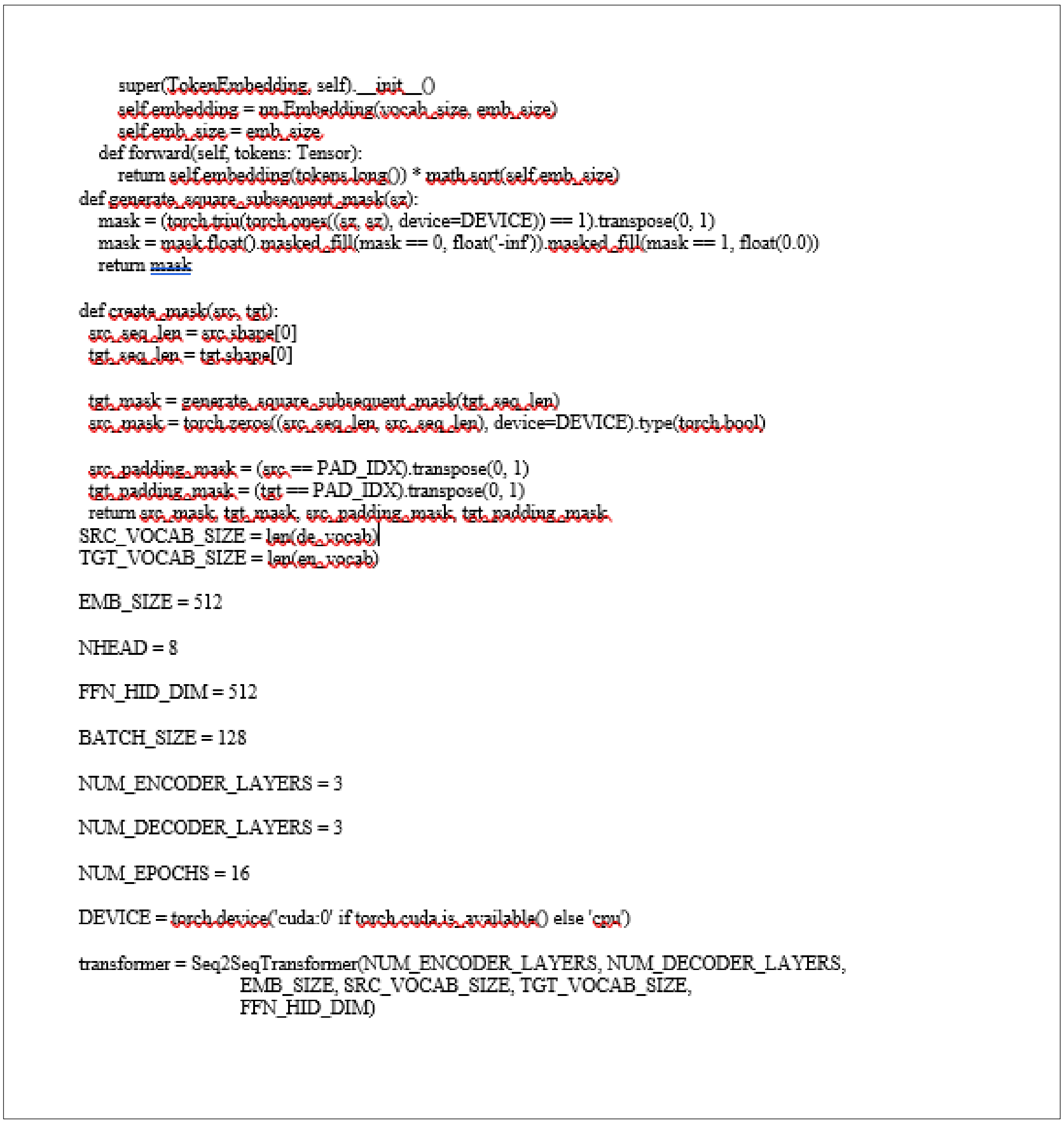
```
from torch.nn.utils.rnn import pad_sequence
from torch.utils.data import DataLoader

def generate_batch(data_batch):
  de_batch, en_batch = [], []
  for (de_item, en_item) in data_batch:
    de_batch.append(torch.cat([torch.tensor([BOS_IDX]), de_item, torch.tensor([EOS_IDX])], dim=0))

    en_batch.append(torch.cat([torch.tensor([BOS_IDX]), en_item, torch.tensor([EOS_IDX])], dim=0))
  de_batch = pad_sequence(de_batch, padding_value=PAD_IDX)
  en_batch = pad_sequence(en_batch, padding_value=PAD_IDX)
  return de_batch, en_batch

train_iter = DataLoader(train_data, batch_size=BATCH_SIZE,
                        shuffle=True, collate_fn=generate_batch) # dividing the data in batches with the batch size specified
valid_iter = DataLoader(val_data, batch_size=BATCH_SIZE,
                        shuffle=True, collate_fn=generate_batch)
test_iter = DataLoader(test_data, batch_size=BATCH_SIZE,
                       shuffle=True, collate_fn=generate_batch)
```

7.1.13 TRANSFORMER MODULE SEQUENCE TO SEQUENCE

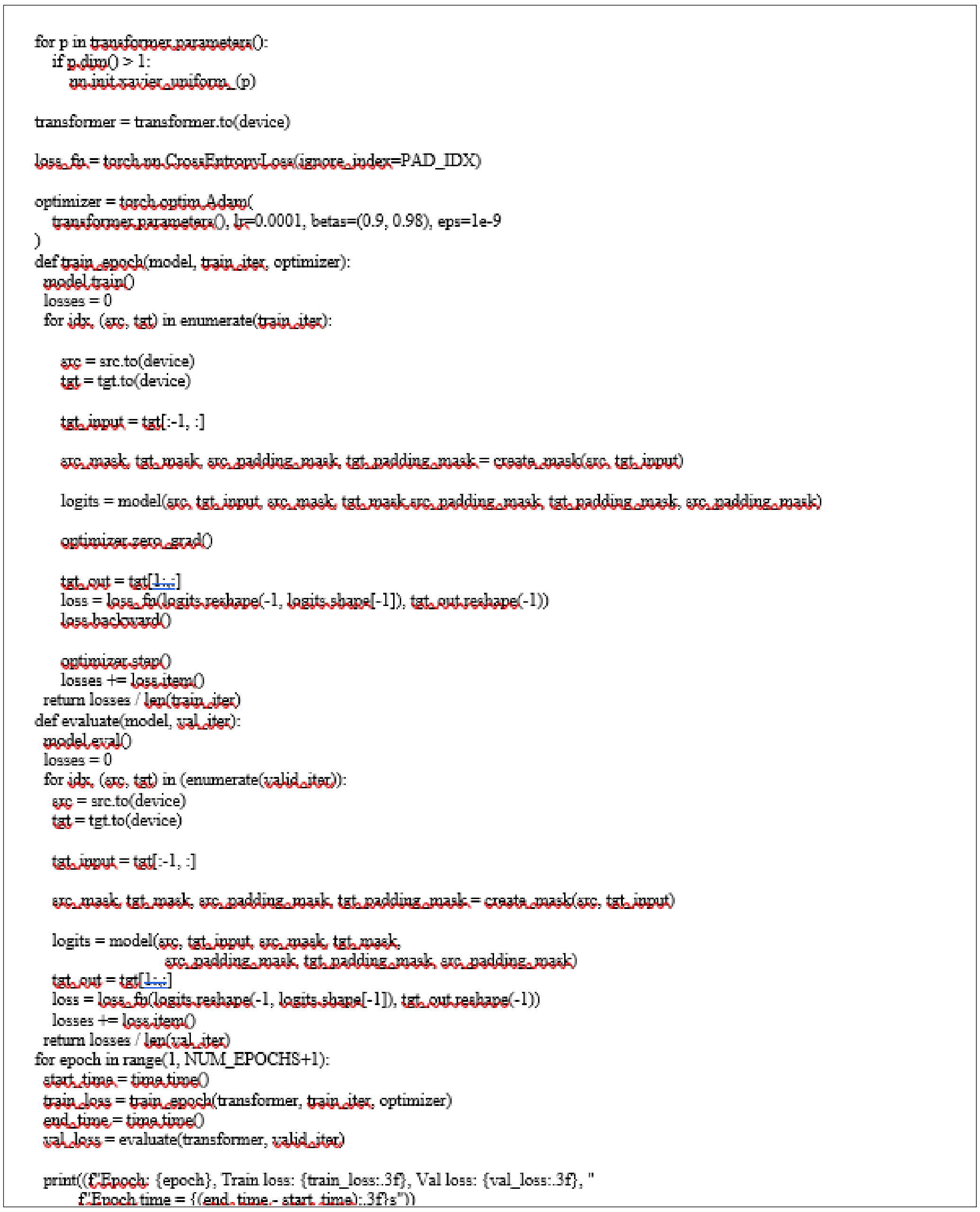


```
class Seq2SeqTransformer(nn.Module):
    def __init__(self, num_encoder_layers: int, num_decoder_layers: int,
                 emb_size: int, src_vocab_size: int, tgt_vocab_size: int,
                 dim_feedforward:int = 512, dropout:float = 0.1):
        super(Seq2SeqTransformer, self).__init__()
        encoder_layer = TransformerEncoderLayer(d_model=emb_size, nhead=NHEAD,
                                                dim_feedforward=dim_feedforward)
        self.transformer_encoder = TransformerEncoder(encoder_layer, num_layers=num_encoder_layers)
        decoder_layer = TransformerDecoderLayer(d_model=emb_size, nhead=NHEAD,
                                                dim_feedforward=dim_feedforward)
        self.transformer_decoder = TransformerDecoder(decoder_layer, num_layers=num_decoder_layers)

        self.generator = nn.Linear(emb_size, tgt_vocab_size)
        self.src_tok_emb = TokenEmbedding(src_vocab_size, emb_size)
        self.tgt_tok_emb = TokenEmbedding(tgt_vocab_size, emb_size)
        self.positional_encoding = PositionalEncoding(emb_size, dropout=dropout)

    def forward(self, src: Tensor, trg: Tensor, src_mask: Tensor,
                tgt_mask: Tensor, src_padding_mask: Tensor,
                tgt_padding_mask: Tensor, memory_key_padding_mask: Tensor):
        src_emb = self.positional_encoding(self.src_tok_emb(src))

        tgt_emb = self.positional_encoding(self.tgt_tok_emb(trg))

        memory = self.transformer_encoder(src_emb, src_mask, src_padding_mask)

        outs = self.transformer_decoder(tgt_emb, memory, tgt_mask, None,
                                        tgt_padding_mask, memory_key_padding_mask)

        return self.generator(outs)

    def encode(self, src: Tensor, src_mask: Tensor):
        return self.transformer_encoder(self.positional_encoding(
                            self.src_tok_emb(src)), src_mask)

    def decode(self, tgt: Tensor, memory: Tensor, tgt_mask: Tensor):
        return self.transformer_decoder(self.positional_encoding(
                          self.tgt_tok_emb(tgt)), memory,
                          tgt_mask)
class PositionalEncoding(nn.Module):
    def __init__(self, emb_size: int, dropout, maxlen: int = 5000):
        super(PositionalEncoding, self).__init__()
        den = torch.exp(- torch.arange(0, emb_size, 2) * math.log(10000) / emb_size)
        pos = torch.arange(0, maxlen).reshape(maxlen, 1)
        pos_embedding = torch.zeros((maxlen, emb_size))
        pos_embedding[:, 0::2] = torch.sin(pos * den)
        pos_embedding[:, 1::2] = torch.cos(pos * den)
        pos_embedding = pos_embedding.unsqueeze(-2)

        self.dropout = nn.Dropout(dropout)
        self.register_buffer('pos_embedding', pos_embedding)

    def forward(self, token_embedding: Tensor):
        return self.dropout(token_embedding +
                            self.pos_embedding[:token_embedding.size(0),:])
```

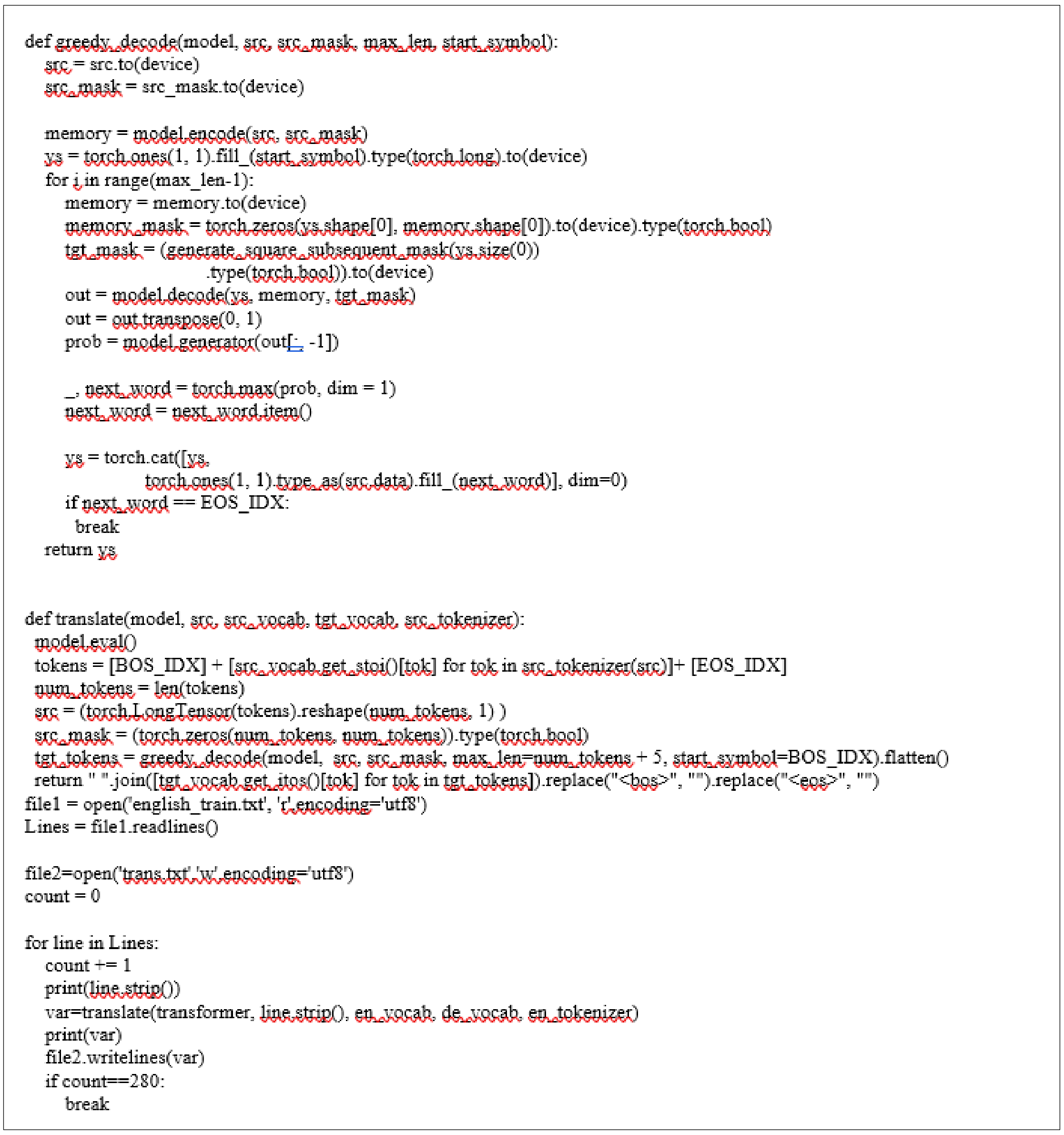
```
class TokenEmbedding(nn.Module):
    def __init__(self, vocab_size: int, emb_size):
```



```
        super(TokenEmbedding, self).__init__()
        self.embedding = nn.Embedding(vocab_size, emb_size)
        self.emb_size = emb_size
    def forward(self, tokens: Tensor):
        return self.embedding(tokens.long()) * math.sqrt(self.emb_size)
def generate_square_subsequent_mask(sz):
    mask = (torch.triu(torch.ones((sz, sz), device=DEVICE)) == 1).transpose(0, 1)
    mask = mask.float().masked_fill(mask == 0, float('-inf')).masked_fill(mask == 1, float(0.0))
    return mask

def create_mask(src, tgt):
  src_seq_len = src.shape[0]
  tgt_seq_len = tgt.shape[0]

  tgt_mask = generate_square_subsequent_mask(tgt_seq_len)
  src_mask = torch.zeros((src_seq_len, src_seq_len), device=DEVICE).type(torch.bool)

  src_padding_mask = (src == PAD_IDX).transpose(0, 1)
  tgt_padding_mask = (tgt == PAD_IDX).transpose(0, 1)
  return src_mask, tgt_mask, src_padding_mask, tgt_padding_mask
SRC_VOCAB_SIZE = len(de_vocab)
TGT_VOCAB_SIZE = len(en_vocab)

EMB_SIZE = 512

NHEAD = 8

FFN_HID_DIM = 512

BATCH_SIZE = 128

NUM_ENCODER_LAYERS = 3

NUM_DECODER_LAYERS = 3

NUM_EPOCHS = 16

DEVICE = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

transformer = Seq2SeqTransformer(NUM_ENCODER_LAYERS, NUM_DECODER_LAYERS,
                                 EMB_SIZE, SRC_VOCAB_SIZE, TGT_VOCAB_SIZE,
                                 FFN_HID_DIM)
```



```
for p in transformer.parameters():
    if p.dim() > 1:
        nn.init.xavier_uniform_(p)

transformer = transformer.to(device)

loss_fn = torch.nn.CrossEntropyLoss(ignore_index=PAD_IDX)

optimizer = torch.optim.Adam(
    transformer.parameters(), lr=0.0001, betas=(0.9, 0.98), eps=1e-9
)
def train_epoch(model, train_iter, optimizer):
  model.train()
  losses = 0
  for idx, (src, tgt) in enumerate(train_iter):

      src = src.to(device)
      tgt = tgt.to(device)

      tgt_input = tgt[:-1, :]

      src_mask, tgt_mask, src_padding_mask, tgt_padding_mask = create_mask(src, tgt_input)

      logits = model(src, tgt_input, src_mask, tgt_mask,src_padding_mask, tgt_padding_mask, src_padding_mask)

      optimizer.zero_grad()

      tgt_out = tgt[1:,:]
      loss = loss_fn(logits.reshape(-1, logits.shape[-1]), tgt_out.reshape(-1))
      loss.backward()

      optimizer.step()
      losses += loss.item()
  return losses / len(train_iter)
def evaluate(model, val_iter):
  model.eval()
  losses = 0
  for idx, (src, tgt) in (enumerate(valid_iter)):
    src = src.to(device)
    tgt = tgt.to(device)

    tgt_input = tgt[:-1, :]

    src_mask, tgt_mask, src_padding_mask, tgt_padding_mask = create_mask(src, tgt_input)

    logits = model(src, tgt_input, src_mask, tgt_mask,
                              src_padding_mask, tgt_padding_mask, src_padding_mask)
    tgt_out = tgt[1:,:]
    loss = loss_fn(logits.reshape(-1, logits.shape[-1]), tgt_out.reshape(-1))
    losses += loss.item()
  return losses / len(val_iter)
for epoch in range(1, NUM_EPOCHS+1):
  start_time = time.time()
  train_loss = train_epoch(transformer, train_iter, optimizer)
  end_time = time.time()
  val_loss = evaluate(transformer, valid_iter)

  print((f"Epoch: {epoch}, Train loss: {train_loss:.3f}, Val loss: {val_loss:.3f}, "
          f"Epoch time = {(end_time - start_time):.3f}s"))
```



```
def greedy_decode(model, src, src_mask, max_len, start_symbol):
   src = src.to(device)
   src_mask = src_mask.to(device)

   memory = model.encode(src, src_mask)
   ys = torch.ones(1, 1).fill_(start_symbol).type(torch.long).to(device)
   for i in range(max_len-1):
      memory = memory.to(device)
      memory_mask = torch.zeros(ys.shape[0], memory.shape[0]).to(device).type(torch.bool)
      tgt_mask = (generate_square_subsequent_mask(ys.size(0))
                              .type(torch.bool)).to(device)
      out = model.decode(ys, memory, tgt_mask)
      out = out.transpose(0, 1)
      prob = model.generator(out[:, -1])

      _, next_word = torch.max(prob, dim = 1)
      next_word = next_word.item()

      ys = torch.cat([ys,
                  torch.ones(1, 1).type_as(src.data).fill_(next_word)], dim=0)
      if next_word == EOS_IDX:
        break
   return ys


def translate(model, src, src_vocab, tgt_vocab, src_tokenizer):
  model.eval()
  tokens = [BOS_IDX] + [src_vocab.get_stoi()[tok] for tok in src_tokenizer(src)]+ [EOS_IDX]
  num_tokens = len(tokens)
  src = (torch.LongTensor(tokens).reshape(num_tokens, 1) )
  src_mask = (torch.zeros(num_tokens, num_tokens)).type(torch.bool)
  tgt_tokens = greedy_decode(model,  src, src_mask, max_len=num_tokens + 5, start_symbol=BOS_IDX).flatten()
  return " ".join([tgt_vocab.get_itos()[tok] for tok in tgt_tokens]).replace("<bos>", "").replace("<eos>", "")
file1 = open('english_train.txt', 'r',encoding='utf8')
Lines = file1.readlines()

file2=open('trans.txt','w',encoding='utf8')
count = 0

for line in Lines:
   count += 1
   print(line.strip())
   var=translate(transformer, line.strip(), en_vocab, de_vocab, en_tokenizer)
   print(var)
   file2.writelines(var)
   if count==280:
      break
```

7.2 COMPONENT DIAGRAM

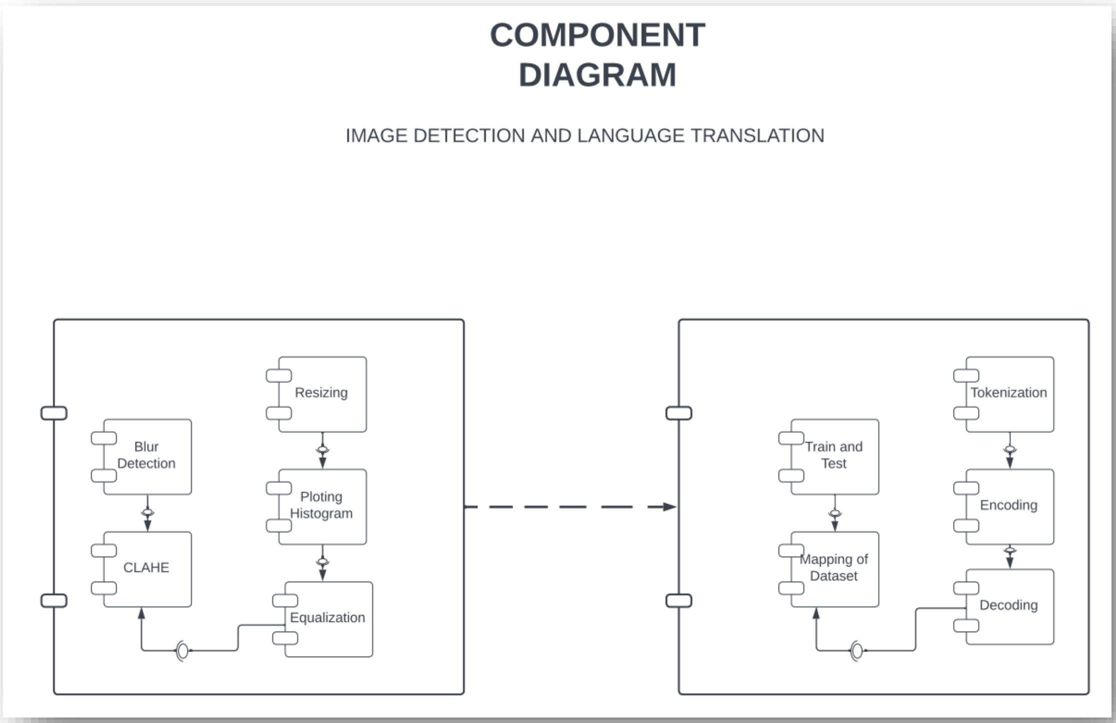


Figure 7.2: Component Diagram

Component Diagram is based on the interfaces that have been shown in the final output. It basically describes the importance of the key working aspects of the project. As per our project, we have seven components that show the core concepts of the project. The input image passes through the resizing process that resize to standard size of the image then it goes to the preprocessing phase where first we plot the histogram and check for the intensity levels of the image, and we apply algorithm to make our image readable and understandable. Equalization of histogram is done for the betterment of the image and more if required with the Adaptive Equalization. Then a final check for the blur detection of the image and after that our image is ready to proceed.

7.3 STATE TRANSITION DIAGRAM

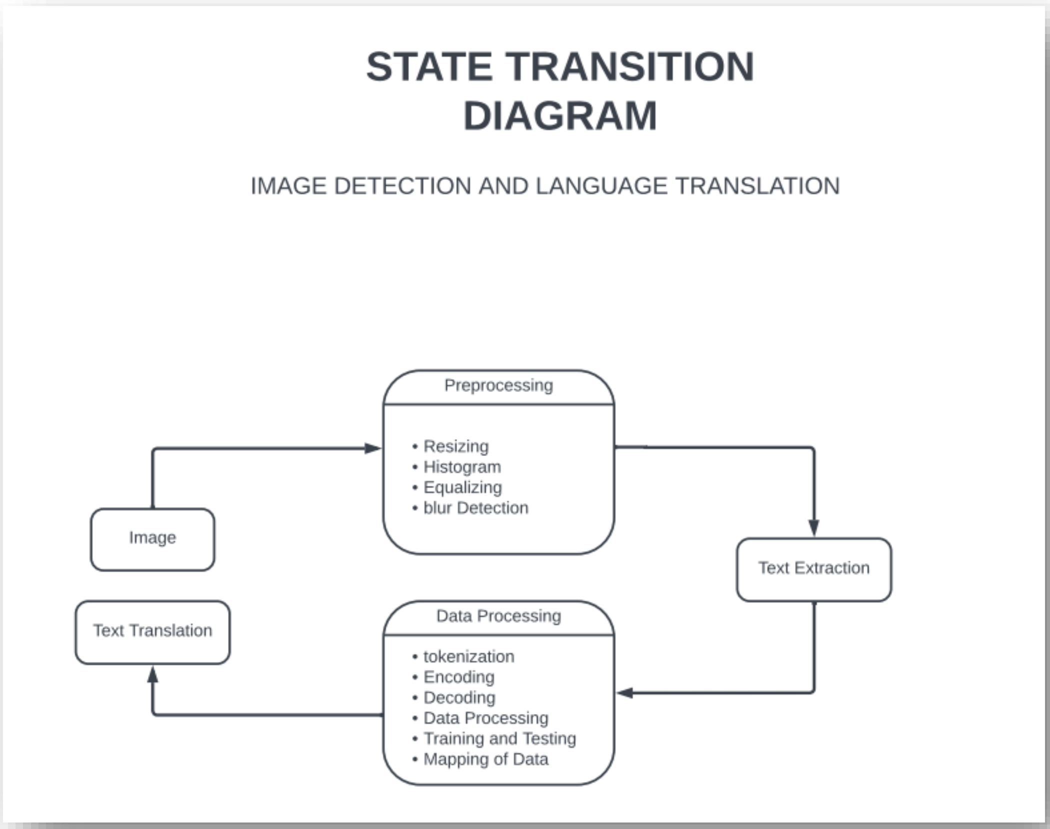


Figure 7.3: State Transition Diagram

State Transition diagram is basically used for the different states changing in the process. We have major two stages of the process one is text extraction and the second is text translation. These two steps are most important in the whole process. Data is being gathered, cleaned, enhanced and then text extracted through the image. Now as the data is being fetched from the image, so the translation process take place where the model of NLP is being trained as per the language so we can train and test our model to complete the process of Translation.

7.4 OPERATIONAL DIAGRAM

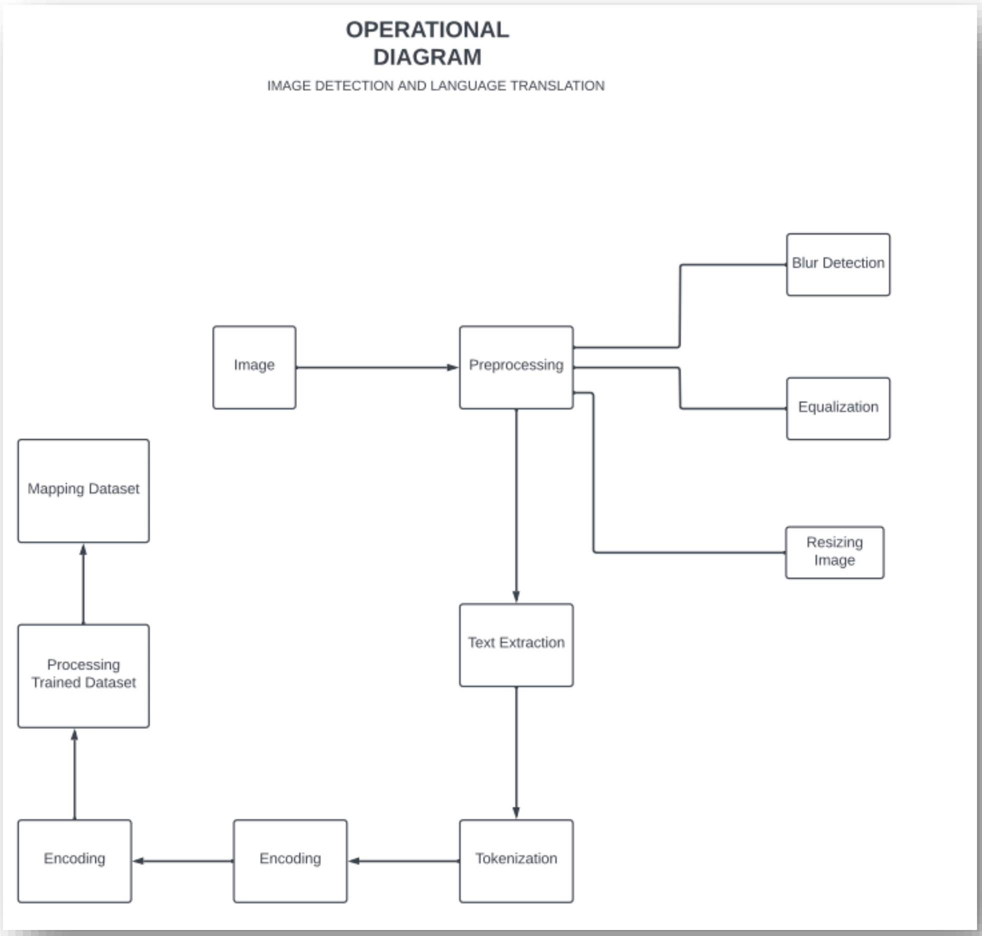


Figure 7.4: Operational Diagram

Operational Diagram is used as the fundamental representation of the project. It basically describes the major operations going on. In major we have two different processes that run through our project. Text Extraction from image and then Translation but as we go through deeper, we have many sub-processes going on. So, these are all the processes that make the project management stronger. Every aspect is being judged by the sub process, sub modules, libraries and helping functions that take part in the process.

7.5 DEPLOYMENT DIAGRAM

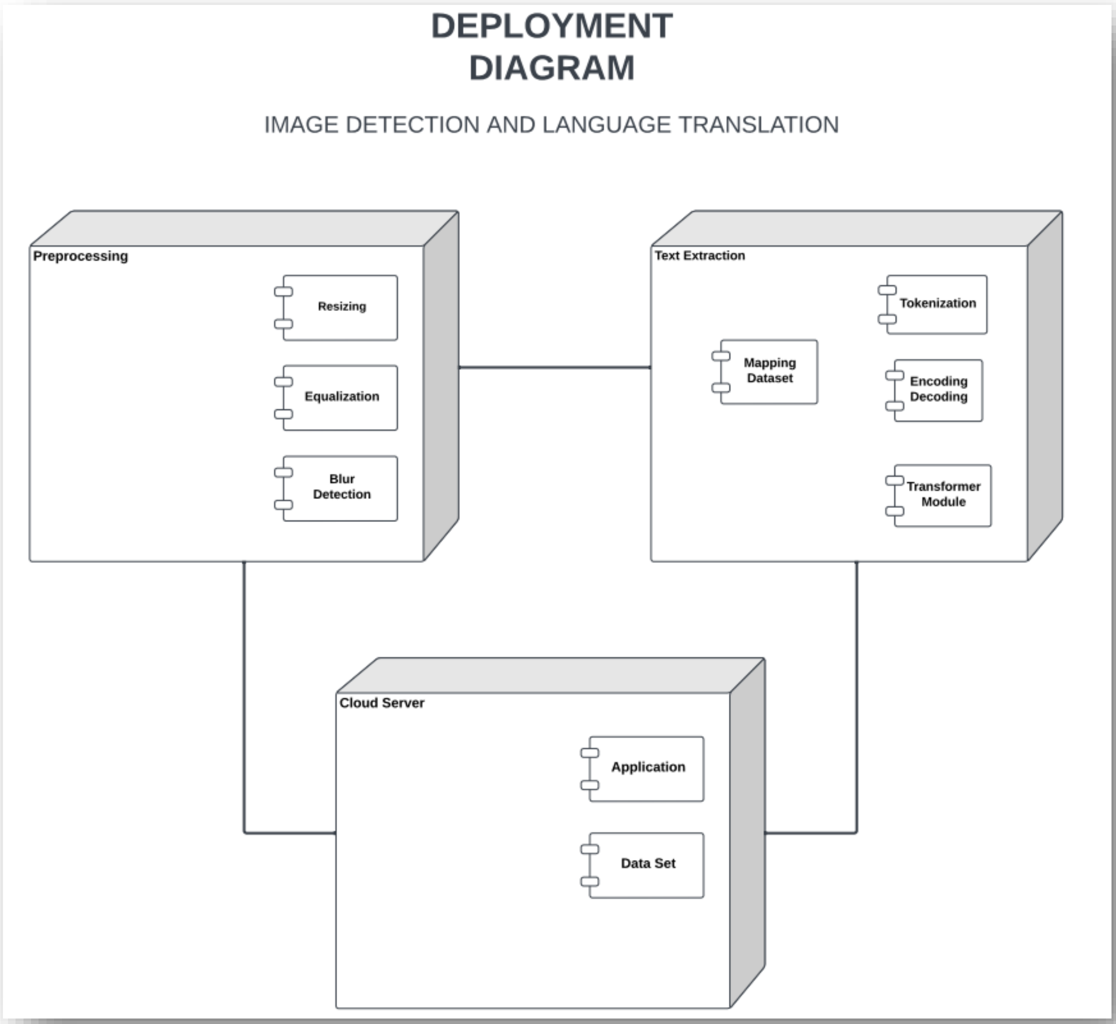


Figure 7.5: Deployment Diagram

Deployment diagram is the last stage of the project that shows the results of the sub domains of the project. It makes up the efforts to a final level and in a concise manner. Our application is the final stop where we deploy our scripts and models which we have trained and implemented. Three main parts of our project that are to be deploy is the server, Preprocessing and the Text Translation Phase. We store the dataset on the cloud server that will host on the web and the main translation phase of the project.

7.6 INTERFACE DESIGN

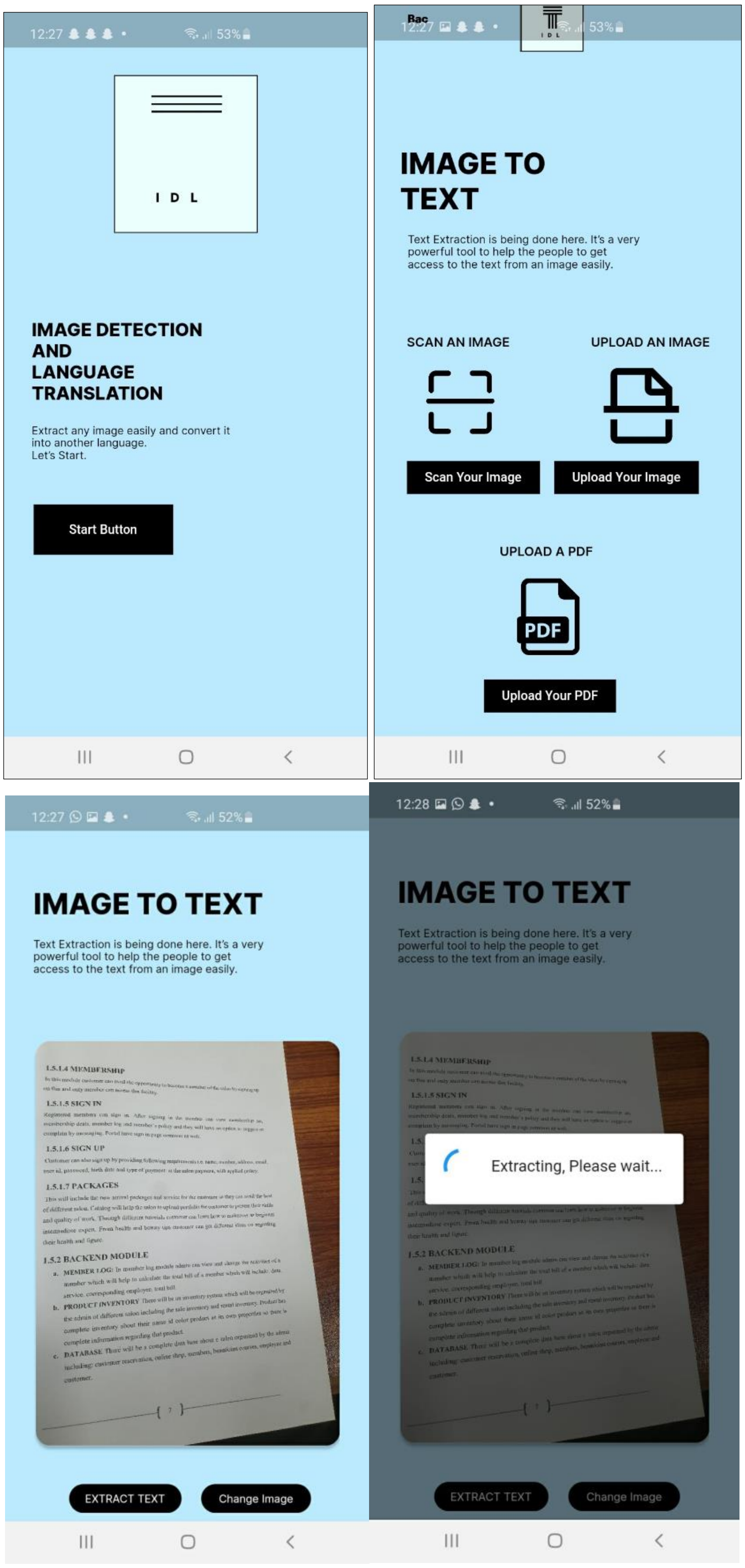


IMAGE DETECTION AND LANGUAGE TRANSLATION

Extract any image easily and convert it into another language.
Let's Start.

IMAGE TO TEXT

Text Extraction is being done here. It's a very powerful tool to help the people to get access to the text from an image easily.

IMAGE TO TEXT

Text Extraction is being done here. It's a very powerful tool to help the people to get access to the text from an image easily.

IMAGE TO TEXT

Text Extraction is being done here. It's a very powerful tool to help the people to get access to the text from an image easily.

Extracting, Please wait...

Figure 7.6: Interface Design

8. TESTING

Black box testing is a testing method in which the tester is not aware of how the software internally functions. The tester exclusively concentrates on the software's input and output. While with white box testing, the tester can test specific pieces of code, algorithms, and methods because they are familiar with the software's internal workings.

Black box testing's primary focus is testing the software's functionality to make sure it complies with requirements and specifications. White box testing is primarily concerned with making sure that the software's internal code is accurate and effective.

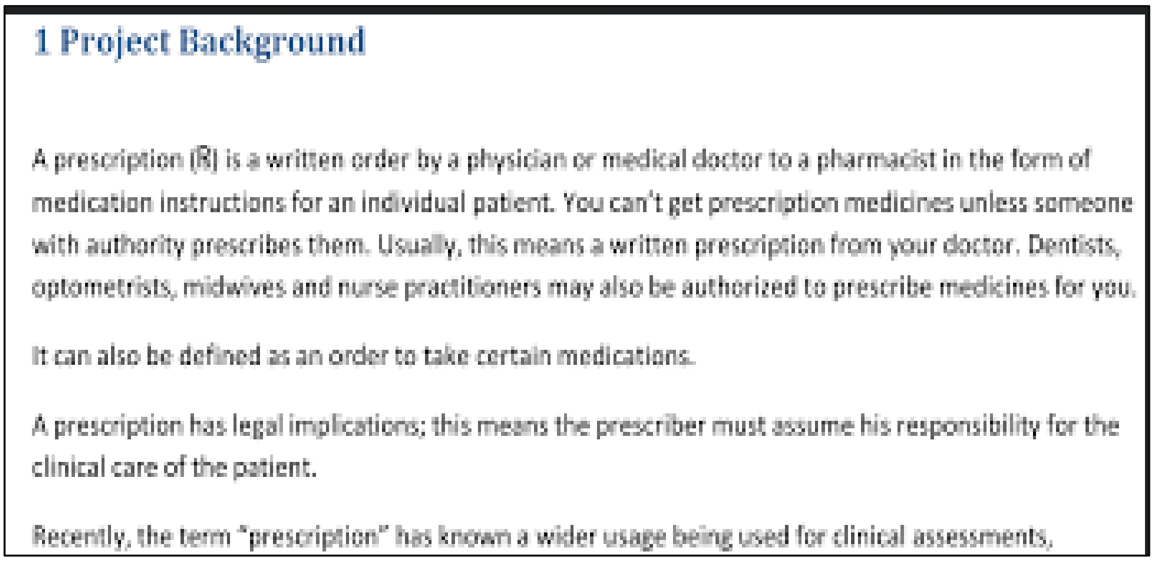
Black box testing can be carried out by testers who are unfamiliar with programming languages because it doesn't require any understanding of the inner workings of the software. Understanding of programming languages, software architecture, and design patterns are necessary for white box testing.

8.1 BLACK BOX TESTING:

Black Box Testing is a software testing method where the tester has no knowledge of the internal structure, design, or implementation of the system under test. Only the external behavior and design are evaluated during this testing approach.
Black box testing is user-friendly, as it does not require programming expertise, and it efficiently uncovers functional flaws. However, it may miss significant internal flaws that do not directly impact functionality. In contrast, White Box Testing, also known as glass box or clear box testing, is employed to identify internal flaws and ensure code effectiveness and readability. This method may be time-consuming and necessitates programming knowledge.
Both white box testing and black box testing are vital components of software testing, and the selection of the approach depends on the testing stage, objectives, and available resources. Black box testing is commonly used to assess software functionality, while white box techniques are applied at the unit, integration, and system levels.

Some of the Test Cases:

Original Image:


1 Project Background

A prescription (℞) is a written order by a physician or medical doctor to a pharmacist in the form of medication instructions for an individual patient. You can't get prescription medicines unless someone with authority prescribes them. Usually, this means a written prescription from your doctor. Dentists, optometrists, midwives and nurse practitioners may also be authorized to prescribe medicines for you.

It can also be defined as an order to take certain medications.

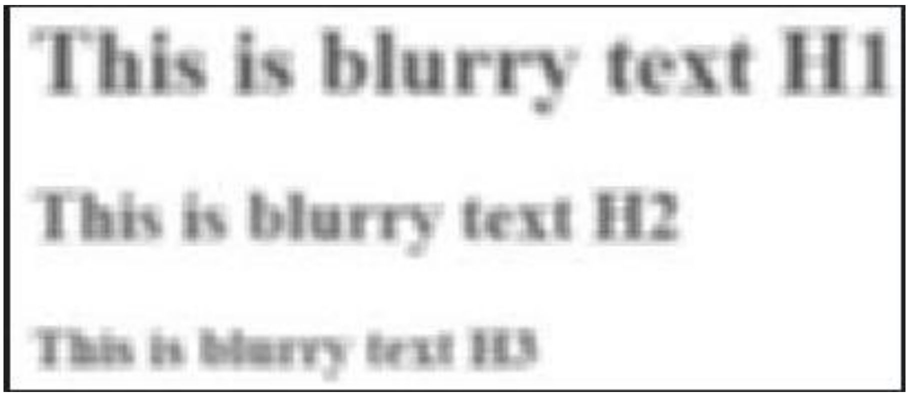
A prescription has legal implications; this means the prescriber must assume his responsibility for the clinical care of the patient.

Recently, the term "prescription" has known a wider usage being used for clinical assessments,

Output Text:

```
------------------------
Extracted Text from Image
------------------------

1 Project Background

Aprasaription (i)ia.a written oderky a physician or medical doctors a pharruactst Inthe toon of
medication Instructions for an Individual patient. You can't get prescription ened cine: ebers: sonmecna:
withauthority preseribartham. Usually, his meania writen proveription frem yourdactor, Ceantists,
wiprtcarreriarty, cheval Wiig Pur parevecRt terval aviary lity bo. lihvcLneel tes pore bm renmclbcriaa fiir EU,
lees ako be defined os om onder totale contain madications,

HA poesia: began Ug ce; Ka Pree es QM Dr TE TA Ppa ty hoe the
dinkcal care of the patient.

Recently, tha tern “praserption’ haa knewn a wider gange beng weed for clinical nzsacemarrt,
```

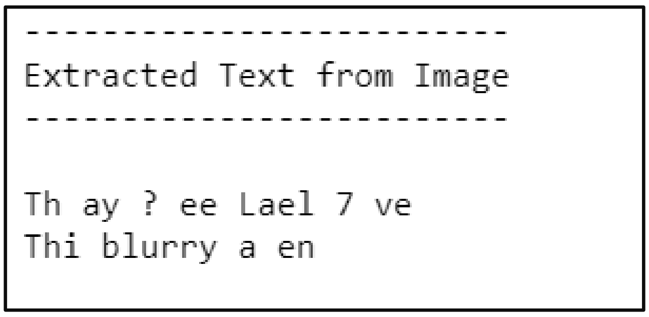
Original Image:


This is blurry text H1

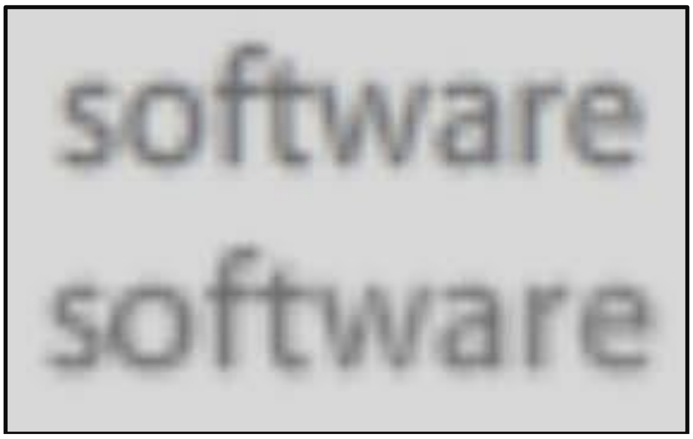
This is blurry text H2

This is blurry text H3

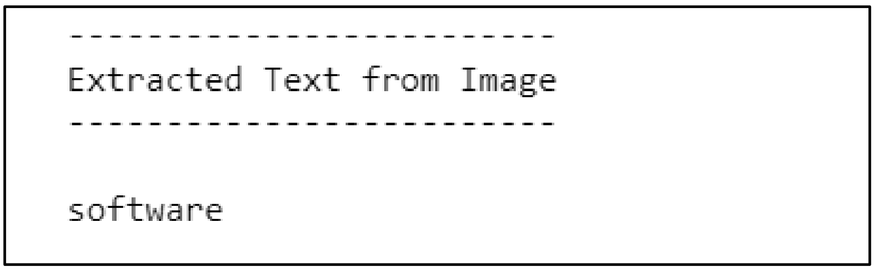
Output Text:


```
-------------------------
Extracted Text from Image
-------------------------

Th ay ? ee Lael 7 ve
Thi blurry a en
```

Original Image:

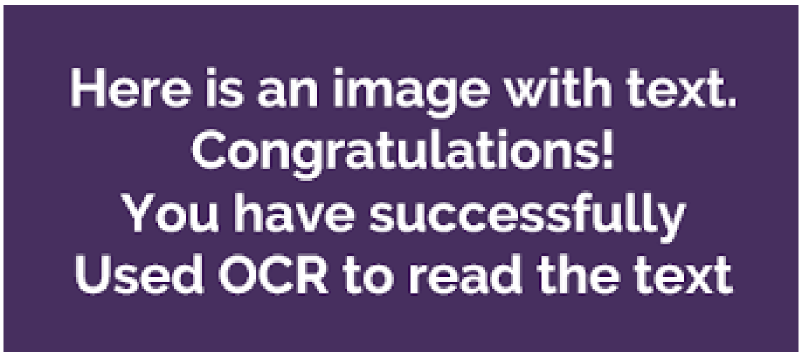


Output Text:

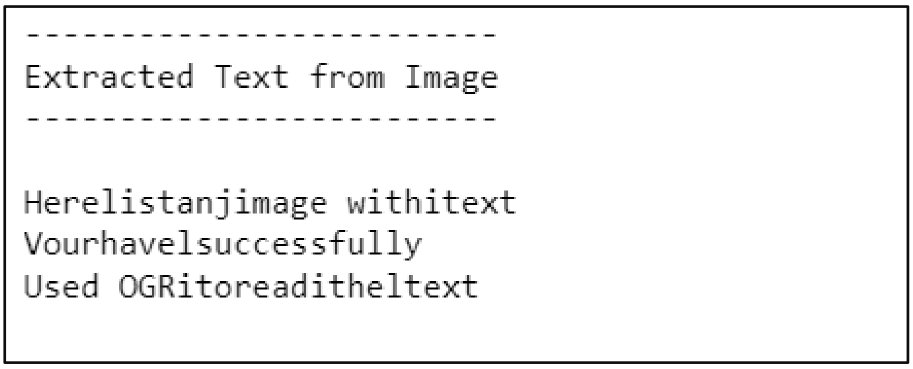


```
-------------------------
Extracted Text from Image
-------------------------

software
```

Original Image:


Here is an image with text.
Congratulations!
You have successfully
Used OCR to read the text

Output Text:


```
-------------------------
Extracted Text from Image
-------------------------

Herelistanjimage withitext
Vourhavelsuccessfully
Used OGRitoreaditheltext
```

8.2 WHITE BOX TESTING:

"White box testing" is a software testing method in which the tester possesses knowledge about the internal structure, design, and implementation of the system being tested. This approach involves examining the code's implementation and impact to conduct the tests effectively.
The White Box Test approach involves scrutinizing the product's code and organizational structure, utilizing this information to conduct the tests. While it can be applied in other phases, such as Integration Tests, it is primarily employed during the Unit Testing phase. For this method to yield accurate results, the tester must possess a profound understanding of the technology used in creating the software.

However, it might overlook certain significant internal flaws unrelated to functionality. White box testing helps find internal flaws and guarantees that the code is effective and readable. It can be time-consuming and requires programming knowledge.

- The White Box Test considers the system's internal operations while the Black Box Test just considers the system's external behavior.

- Contrary to White Box Testing, the use of Black Box Testing does not necessitate implementation knowledge.

- A Black Box Testing can be completed faster than a White Box Testing.

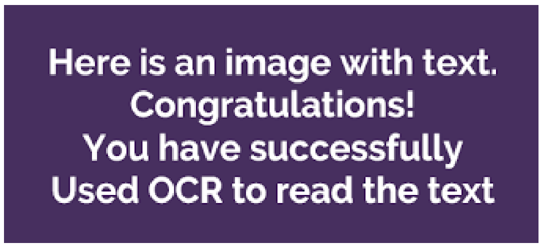
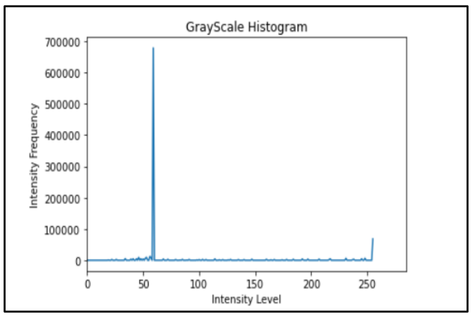
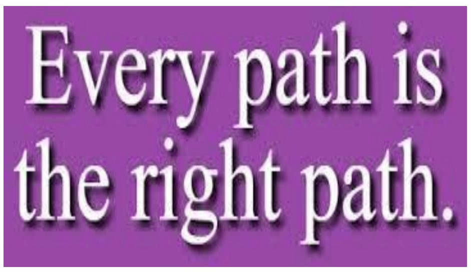
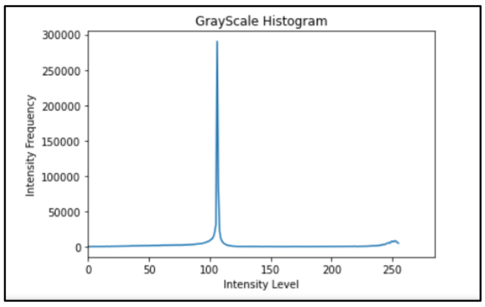
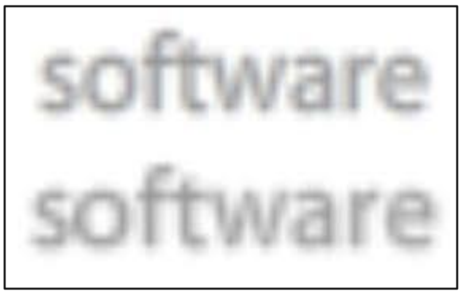
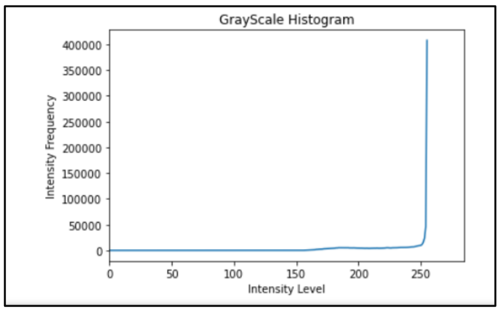
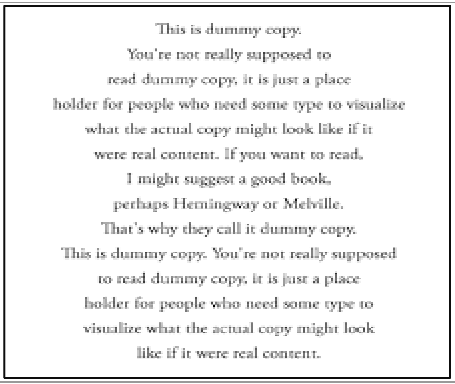
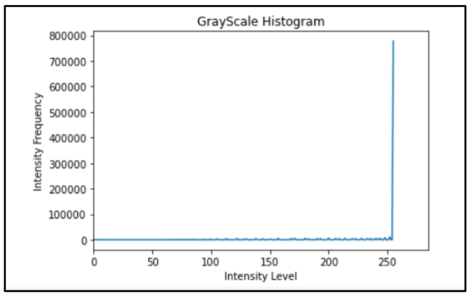
In conclusion, both white box testing and black box testing are crucial for software testing, and the choice of approach relies on the testing stage, the objectives, and the resources available.
Black box testing is typically performed to test the functionality of the software. Software is tested using white box techniques at the unit, integration, and system levels.

8.2.1 RESIZING OF THE INPUT IMAGE:

IMAGE NO	ORIGINAL SIZE	STANDARD SIZE
1	340 X 148	1280 X 720
2	120 X 720	1280 X 720
3	180 X 320	1280 X 720
4	1280 X 720	SAME AS ORIGINAL
5	420 X 120	1280 X 720
6	340 X 148	1280 X 720
7	1280 X 720	SAME AS ORIGINAL
8	180 X 320	1280 X 720
9	1280 X 720	SAME AS ORIGINAL
10	420 X 120	1280 X 720

Table 8.2.1: Resizing the input image.

8.2.2 PLOTING HISTOGRAM OF THE INPUT IMAGE:

IMAGE	HISTOGRAM
 Here is an image with text. Congratulations! You have successfully Used OCR to read the text	
 Every path is the right path.	
 software software	
 This is dummy copy. You're not really supposed to read dummy copy, it is just a place holder for people who need some type to visualize what the actual copy might look like if it were real content. If you want to read, I might suggest a good book, perhaps Hemingway or Melville. That's why they call it dummy copy. This is dummy copy. You're not really supposed to read dummy copy, it is just a place holder for people who need some type to visualize what the actual copy might look like if it were real content.	

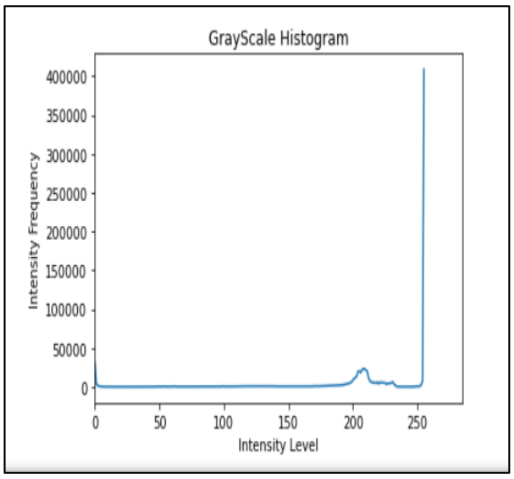
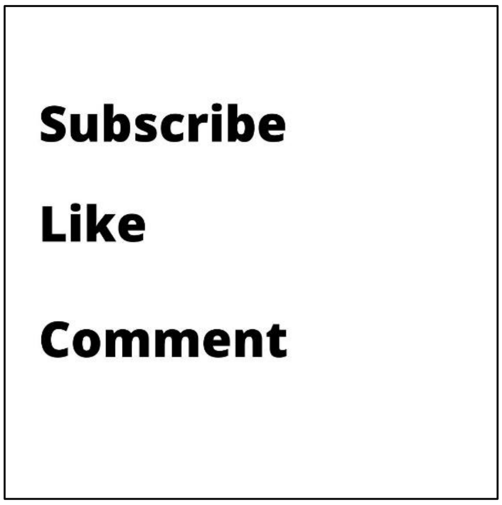
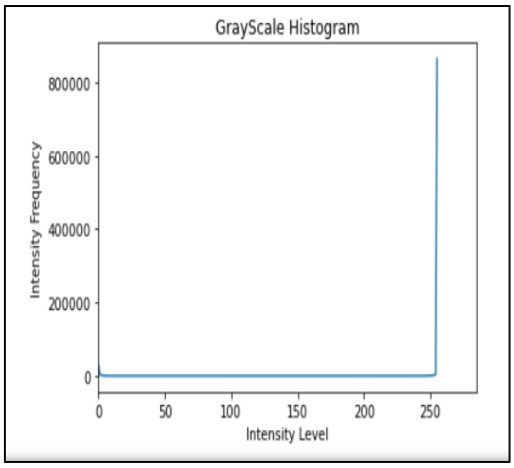
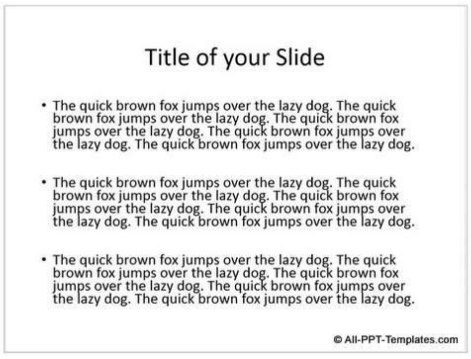
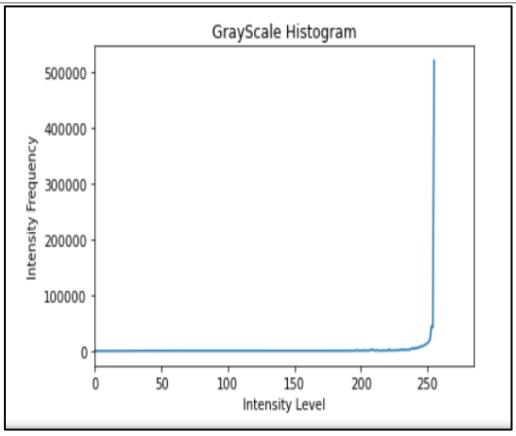
BLACK FRIDAY SALE!	
 Subscribe Like Comment	
 Title of your Slide • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. © All-PPT-Templates.com	

Table 8.2.2: Plotting Histogram of the input image.

8.2.3 APPLYING EQUALIZATION TO THE INPUT IMAGE:

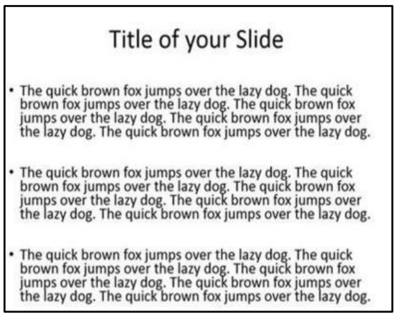
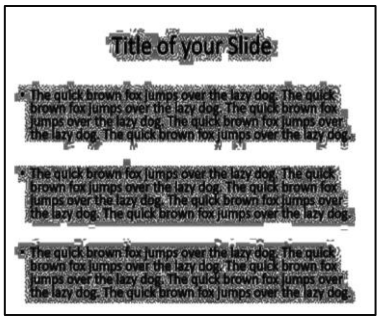
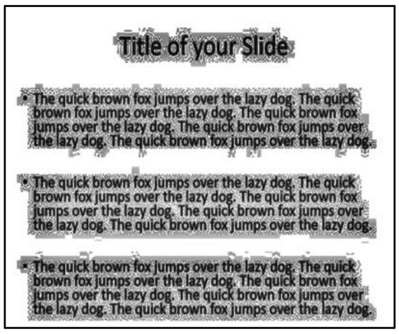
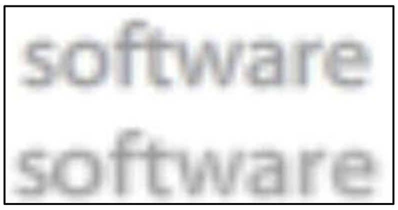
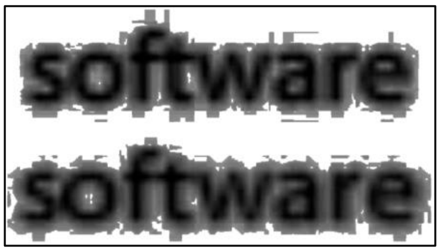
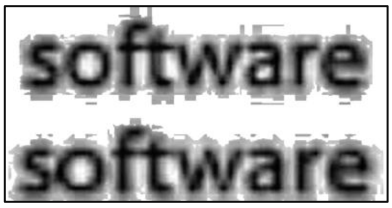
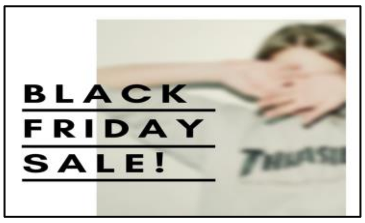
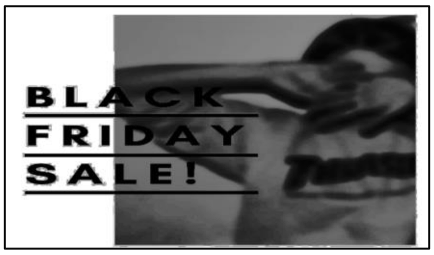
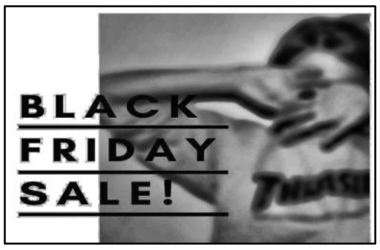
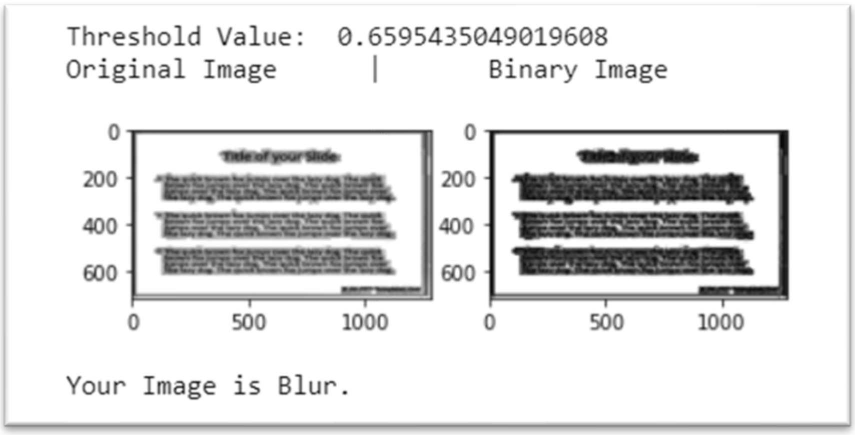
IMAGE	EQUALIZED	ADAPTIVE EQUALIZED
 Title of your Slide • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.	 Title of your Slide • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.	 Title of your Slide • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. • The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.
 software software	 software software	 software software
		

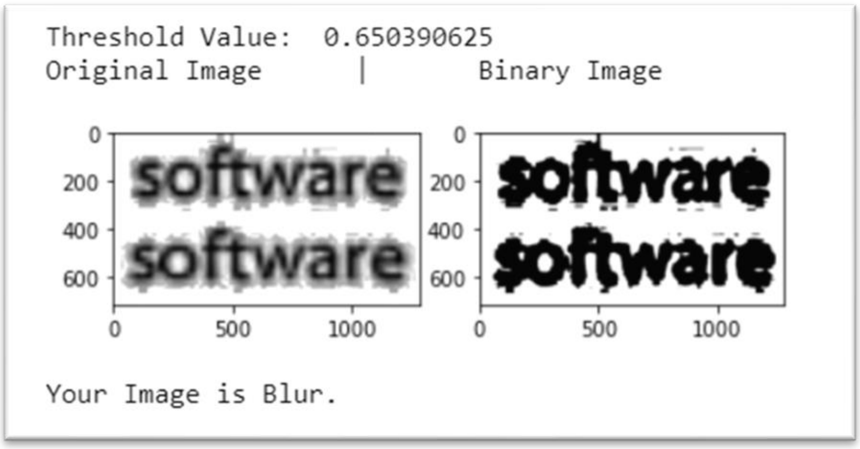
Table 8.2.3: Applying Equalization to the input image.

8.2.4 BLUR DETECTION OF THE INPUT IMAGE:



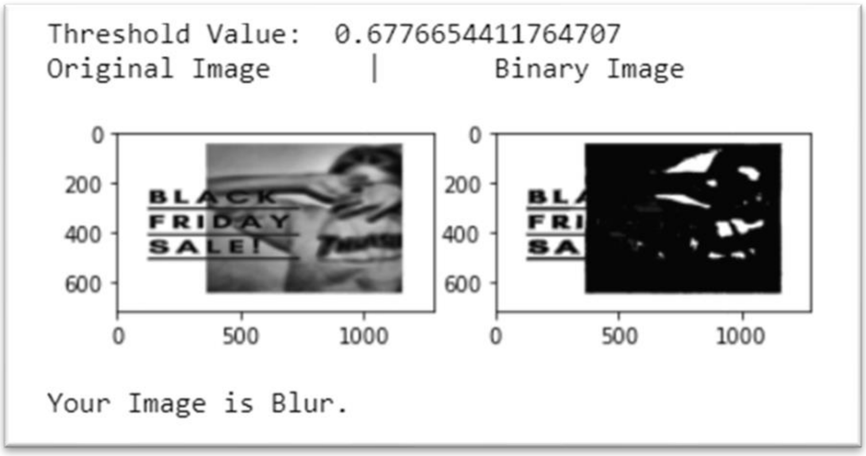
```
Threshold Value:  0.6595435049019608
Original Image      |       Binary Image
```

```
Your Image is Blur.
```



```
Threshold Value:  0.650390625
Original Image      |       Binary Image
```

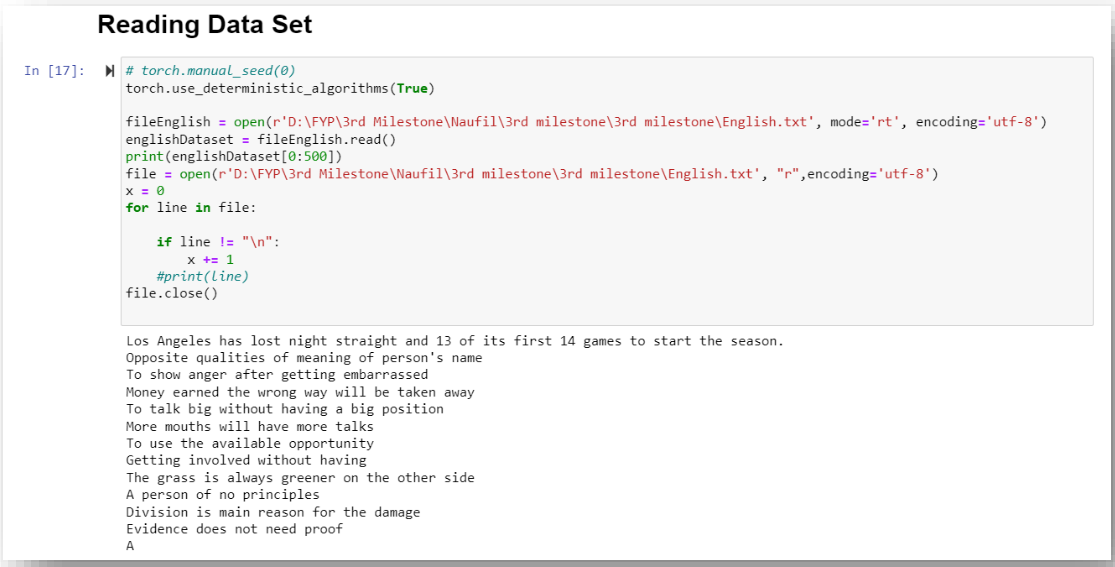
```
Your Image is Blur.
```



```
Threshold Value:  0.6776654411764707
Original Image      |       Binary Image
```

```
Your Image is Blur.
```

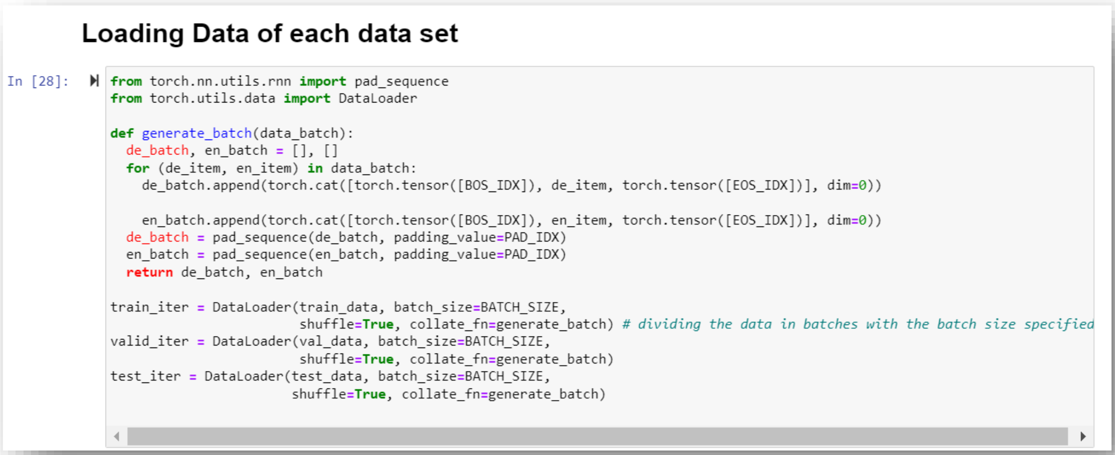
8.2.5 READING THE DATASET:



Reading Data Set

```
In [17]:  # torch.manual_seed(0)
          torch.use_deterministic_algorithms(True)

          fileEnglish = open(r'D:\FYP\3rd Milestone\Naufil\3rd milestone\3rd milestone\English.txt', mode='rt', encoding='utf-8')
          englishDataset = fileEnglish.read()
          print(englishDataset[0:500])
          file = open(r'D:\FYP\3rd Milestone\Naufil\3rd milestone\3rd milestone\English.txt', "r",encoding='utf-8')
          x = 0
          for line in file:

              if line != "\n":
                  x += 1
              #print(line)
          file.close()
```

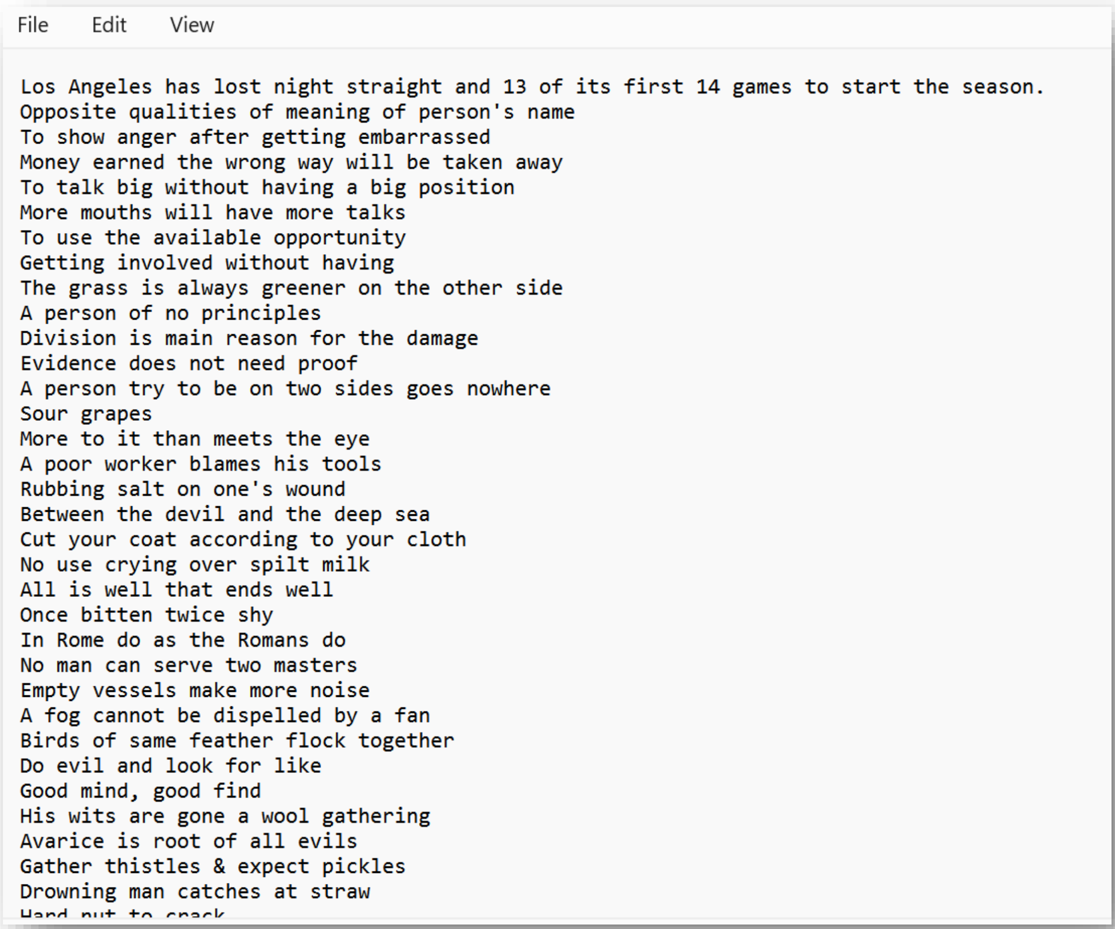
```
Los Angeles has lost night straight and 13 of its first 14 games to start the season.
Opposite qualities of meaning of person's name
To show anger after getting embarrassed
Money earned the wrong way will be taken away
To talk big without having a big position
More mouths will have more talks
To use the available opportunity
Getting involved without having
The grass is always greener on the other side
A person of no principles
Division is main reason for the damage
Evidence does not need proof
A
```



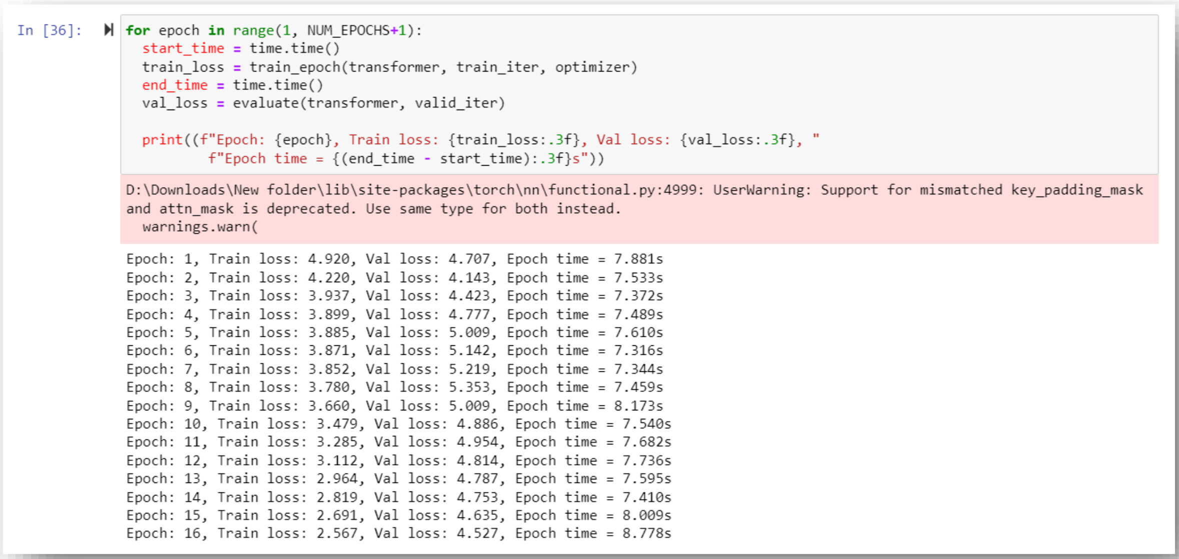
Loading Data of each data set

```
In [28]:  from torch.nn.utils.rnn import pad_sequence
          from torch.utils.data import DataLoader

          def generate_batch(data_batch):
            de_batch, en_batch = [], []
            for (de_item, en_item) in data_batch:
              de_batch.append(torch.cat([torch.tensor([BOS_IDX]), de_item, torch.tensor([EOS_IDX])], dim=0))

              en_batch.append(torch.cat([torch.tensor([BOS_IDX]), en_item, torch.tensor([EOS_IDX])], dim=0))
            de_batch = pad_sequence(de_batch, padding_value=PAD_IDX)
            en_batch = pad_sequence(en_batch, padding_value=PAD_IDX)
            return de_batch, en_batch

          train_iter = DataLoader(train_data, batch_size=BATCH_SIZE,
                                  shuffle=True, collate_fn=generate_batch) # dividing the data in batches with the batch size specified
          valid_iter = DataLoader(val_data, batch_size=BATCH_SIZE,
                                  shuffle=True, collate_fn=generate_batch)
          test_iter = DataLoader(test_data, batch_size=BATCH_SIZE,
                                 shuffle=True, collate_fn=generate_batch)
```

DATASET TO BE LOADED:



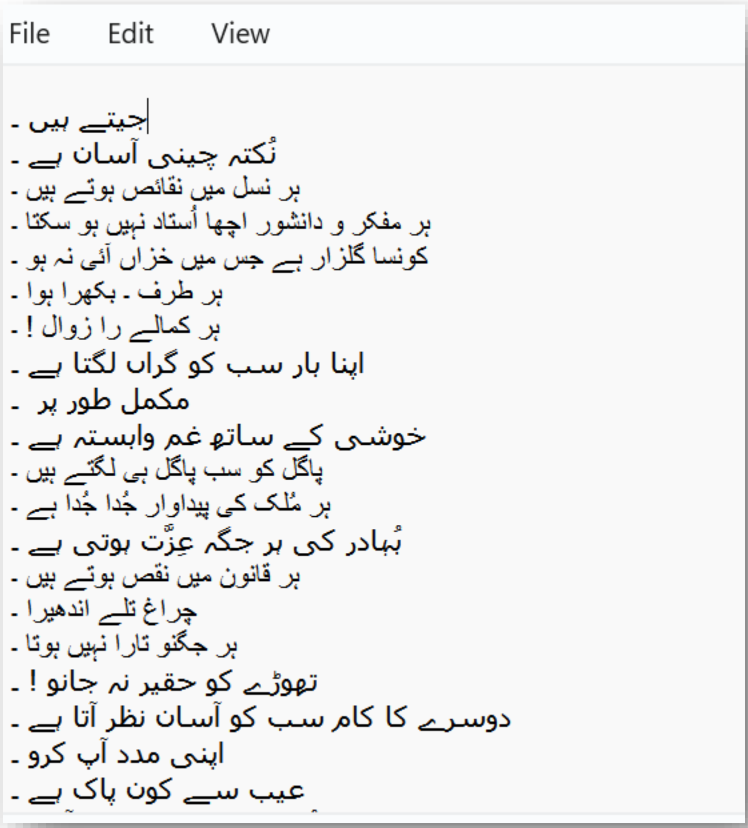
```
Los Angeles has lost night straight and 13 of its first 14 games to start the season.
Opposite qualities of meaning of person's name
To show anger after getting embarrassed
Money earned the wrong way will be taken away
To talk big without having a big position
More mouths will have more talks
To use the available opportunity
Getting involved without having
The grass is always greener on the other side
A person of no principles
Division is main reason for the damage
Evidence does not need proof
A person try to be on two sides goes nowhere
Sour grapes
More to it than meets the eye
A poor worker blames his tools
Rubbing salt on one's wound
Between the devil and the deep sea
Cut your coat according to your cloth
No use crying over spilt milk
All is well that ends well
Once bitten twice shy
In Rome do as the Romans do
No man can serve two masters
Empty vessels make more noise
A fog cannot be dispelled by a fan
Birds of same feather flock together
Do evil and look for like
Good mind, good find
His wits are gone a wool gathering
Avarice is root of all evils
Gather thistles & expect pickles
Drowning man catches at straw
Hard nut to crack
```



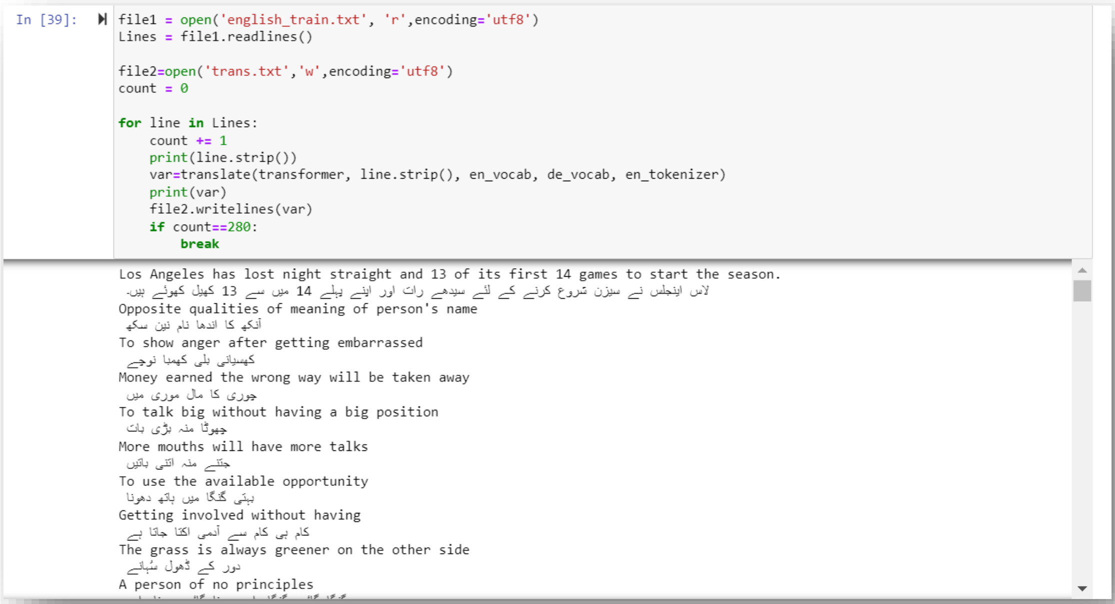
```
In [36]:  for epoch in range(1, NUM_EPOCHS+1):
              start_time = time.time()
              train_loss = train_epoch(transformer, train_iter, optimizer)
              end_time = time.time()
              val_loss = evaluate(transformer, valid_iter)

              print((f"Epoch: {epoch}, Train loss: {train_loss:.3f}, Val loss: {val_loss:.3f}, "
                      f"Epoch time = {(end_time - start_time):.3f}s"))
```

```
D:\Downloads\New folder\lib\site-packages\torch\nn\functional.py:4999: UserWarning: Support for mismatched key_padding_mask
and attn_mask is deprecated. Use same type for both instead.
  warnings.warn(
```

```
Epoch: 1, Train loss: 4.920, Val loss: 4.707, Epoch time = 7.881s
Epoch: 2, Train loss: 4.220, Val loss: 4.143, Epoch time = 7.533s
Epoch: 3, Train loss: 3.937, Val loss: 4.423, Epoch time = 7.372s
Epoch: 4, Train loss: 3.899, Val loss: 4.777, Epoch time = 7.489s
Epoch: 5, Train loss: 3.885, Val loss: 5.009, Epoch time = 7.610s
Epoch: 6, Train loss: 3.871, Val loss: 5.142, Epoch time = 7.316s
Epoch: 7, Train loss: 3.852, Val loss: 5.219, Epoch time = 7.344s
Epoch: 8, Train loss: 3.780, Val loss: 5.353, Epoch time = 7.459s
Epoch: 9, Train loss: 3.660, Val loss: 5.009, Epoch time = 8.173s
Epoch: 10, Train loss: 3.479, Val loss: 4.886, Epoch time = 7.540s
Epoch: 11, Train loss: 3.285, Val loss: 4.954, Epoch time = 7.682s
Epoch: 12, Train loss: 3.112, Val loss: 4.814, Epoch time = 7.736s
Epoch: 13, Train loss: 2.964, Val loss: 4.787, Epoch time = 7.595s
Epoch: 14, Train loss: 2.819, Val loss: 4.753, Epoch time = 7.410s
Epoch: 15, Train loss: 2.691, Val loss: 4.635, Epoch time = 8.009s
Epoch: 16, Train loss: 2.567, Val loss: 4.527, Epoch time = 8.778s
```



جیتے ہیں ۔
نُکتہ چینی آسان ہے ۔
ہر نسل میں نقائص ہوتے ہیں ۔
ہر مفکر و دانشور اچھا اُستاد نہیں ہو سکتا ۔
کونسا گلزار ہے جس میں خزاں آئی نہ ہو ۔
ہر طرف ۔ بکھرا ہوا ۔
ہر کمالے را زوال ! ۔
اپنا بار سب کو گراں لگتا ہے ۔
مکمل طور پر ۔
خوشی کے ساتھ غم وابستہ ہے ۔
پاگل کو سب پاگل ہی لگتے ہیں ۔
ہر مُلک کی پیداوار جُدا جُدا ہے ۔
بُہادر کی ہر جگہ عِزّت ہوتی ہے ۔
ہر قانون میں نقص ہوتے ہیں ۔
چراغ تلے اندھیرا ۔
ہر جگنو تارا نہیں ہوتا ۔
تھوڑے کو حقیر نہ جانو ! ۔
دوسرے کا کام سب کو آسان نظر آتا ہے ۔
اپنی مدد آپ کرو ۔
عیب سے کون پاک ہے ۔


```
In [39]:  file1 = open('english_train.txt', 'r',encoding='utf8')
          Lines = file1.readlines()

          file2=open('trans.txt','w',encoding='utf8')
          count = 0

          for line in Lines:
              count += 1
              print(line.strip())
              var=translate(transformer, line.strip(), en_vocab, de_vocab, en_tokenizer)
              print(var)
              file2.writelines(var)
              if count==280:
                  break
```

```
Los Angeles has lost night straight and 13 of its first 14 games to start the season.
 لاس اینجلس نے سیزن شروع کرنے کے لئے سیدھے رات اور اپنے پہلے 14 میں سے 13 کھیل کھوئے ہیں۔
Opposite qualities of meaning of person's name
 آنکھ کا اندھا نام نین سکھ
To show anger after getting embarrassed
 کھسیانی بلی کھمبا نوچے
Money earned the wrong way will be taken away
 چوری کا مال موری میں
To talk big without having a big position
 چھوٹا منہ بڑی بات
More mouths will have more talks
 جتنے منہ اتنی باتیں
To use the available opportunity
 بہتی گنگا میں ہاتھ دھونا
Getting involved without having
 کام ہی کام سے آدمی اکتا جاتا ہے
The grass is always greener on the other side
 دور کے ڈھول سُہانے
A person of no principles
```

8.3 SYSTEM TESTING:

The system integration testing is done by combining the back end and front end of the application and to integrate them. We must apply different type of test cases to check the system's availability and performance.

TEST CASES	VALID	NOT VALID
Page Redirection	✓	
Alignments of the elements		✓
Logo display	✓	
Action Buttons	✓	
Text Extraction	✓	
Language Translation	✓	
Image display alignment		✓

Table 8.3: system Integration Testing.

9. DISCUSSION (OPTIONAL)

9.1 PROJECT OVERVIEW:

The main purpose to serve the people with an enhanced and useful application that brings them much faster as the world is growing. To capture the idea of dealing with the android application is chosen due to the vast and hands on experience of the users to the android app. As far as project idea is concern, it gives us the full exposure to use of two languages in parallel.
The project is basically allowing the user to enter the text in the source language, and it will be translated into the target language. This is basically done by the image detection. First the user will scan the image containing some text. Then it will be converted into the readable text by the help of OCR (Optical Character Recognition). Then it will be translated into the target language by using a technique called NLP (Natural Language Processing).

9.2 PROJECT OBJECTIVES:

9.2.1 BRIDGING THE LINGUISTIC BARRIERS BETWEEN THE COUNTRIES:

When communicating with people from a different country and language that you don't understand, having an interpreter by your side can be incredibly helpful. Language translators are skilled in fluent and seamless translation from one language to another. With well-qualified translators, you can be confident that your messages will be accurately interpreted, minimizing the risk of any unintended miscommunications.
Professional translators have become essential for modern businesses, as they enable individuals and companies to broaden their horizons and tap into new markets. By speaking to their target audience in their native dialect, translators help businesses effectively connect with and engage specific markets, fostering successful communication and business expansion.

9.2.2 IMPROVED COMMUNICATION AND EXCHANGE OF IDEAS:

Language translation services play a vital role in facilitating seamless communication and idea-sharing between different parties, overcoming language barriers. With translation, complex subject matters can be effortlessly explained to individuals in their native language. Skilled professional translators ensure exceptional accuracy in their work, swiftly conveying messages without altering the intended meaning or overlooking important details.

9.3 PROJECT SCOPE:

This basically involves a dual method for the execution of an app through extracting the text from image and then converting the text into second language. This seems an easy task but can involve many complications, however we will manage through our enhanced techniques. Without losing the meaning of the original, translation converts a text from the source language (SL) into a proper and understandable version of the target language (TL). Many people believe that all it takes to be a translator is to be multilingual. This is untrue. It goes without saying that being bilingual is a crucial precondition, but translation skills are developed based on one's own extensive communicative and writing experiences in both languages. In actuality, translation is the process of taking a text's meaning out of its current form and reproducing it in a new form of a second language.

The following items are both included and excluded from the project's scope.

In of scope:

Text Extraction from image
Language Translation
Application Development

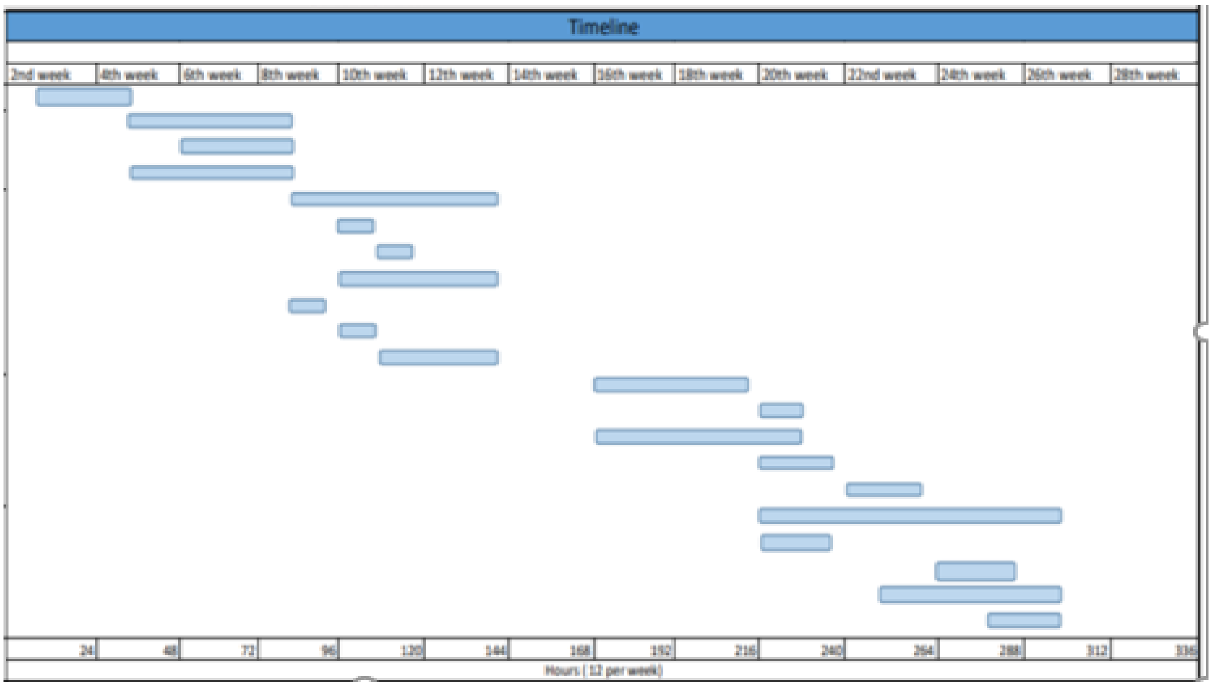
Out of scope:

Application Enhancement (Adding new Features)
Multiple Language Translation

9.4 PROJECT COST ESTIMATION:

- Web Domain
- Web Hosting
- Cloud Server
- Book Binding

9.5 GANTT CHART:



9.6 PROJECT ASSUMPTIONS:

To determine the necessary tasks and project timeline, it is essential to establish certain assumptions and premises. These assumptions, based on the current knowledge, are crucial for outlining the project plan. However, if any of these assumptions are proven incorrect at a later stage, the project plan should be promptly adjusted by revising activities and estimates to ensure accuracy and feasibility. Flexibility in accommodating potential changes is essential for successful project execution.

- Image is perfectly scanned through OCR.
- Data is shown on the mobile App.
- Language translates the source into target correctly.

9.7 PROJECT RISKS:

Project risks refer to specific characteristics, circumstances, or elements within the project environment that could potentially negatively impact the project or the quality of its outcomes. The known risks associated with this project have been listed below. To address these risks, a comprehensive plan will be devised and implemented to either minimize their effects or eliminate them altogether, ensuring the project's successful execution and delivery of high-quality results.

Risk Area	**Level (H/M/L)**	**Risk Plan**
1. OCR is being Made by us and it recognition is incorrect.	M	We try to use the standard library for the correct results.
2. Language is not accurately translated into the target language.	M	We try to figure out the issue and will train our model more accurately.
3. Training and testing is failed of the model.	H	We try to fix issue as fast as possible by Re-train our model or re-built it.

9.8 TOOLS AND TECHNOLOGIES:

Hardware:

- Database
- Server

Software(s):

- Android Studio
- Flutter
- NLP
- OCR

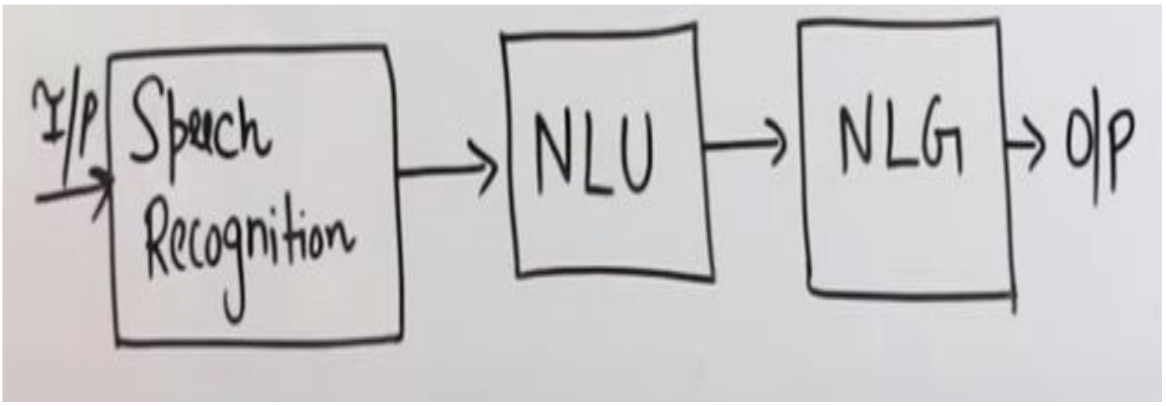
9.9 NATURAL LANGUAGE PROCESSING:

9.9.1 WHAT IS NLP:

Natural Language Processing (NLP) is a fundamental aspect of artificial intelligence (AI) that empowers computer programs to comprehend human language as it is spoken and written, commonly known as natural language. NLP enables machines to analyze and interpret human communication, opening up vast possibilities for seamless interactions between humans and technology.

9.9.2 APPLICATIONS OF NLP:

- SPEECH RECOGNITION: (siri , google assistant)
- SENTIMENTAL ANALYSIS: (tweents or facebook posts are good or bad e.t.c)
- MACHINE TRANSLATIONS: (google translate)
- CHAT BOTS E.T.C: (human communicate with machine)
- Automatic speech recognition.



9.9.3 CHALLENGE OF AMBIGIUTY:

1. **Lexical ambiguity**
   It is possible than there is multiple meaning of a single word
   Example: (the tank is full of water) now the problem is which tank war tank or any other tank?
2. **Syntactic ambiguity**
   Structure of the sentence should be correct
   Example: (old man and woman were taken to safe place).
3. **Semantic ambiguity**
   it depends on the meaning
   example : (the car hit the pole while it was moving)
4. **Pragmatic ambiguity**
   example: the police are coming

9.9.4 TYPES OF TOOLS:

1. MonkeyLearn | NLP made simple
2. Aylien | Leveraging news content with NLP
3. IBM Watson | A pioneer AI platform for businesses
4. Google Cloud NLP API | Google technology applied to NLP
5. Amazon Comprehend | An AWS service to get insights from text
6. NLTK | The most popular Python library
7. Stanford Core NLP | Stanford's fast and robust toolkit
8. TextBlob | An intuitive interface for NLTK
9. SpaCy | Super-fast library for advanced NLP tasks
10. GenSim | State-of-the-art topic modeling

9.10 Q/A DISCUSSION:

What is natural language processing based on language translaion?

Natural Language Processing (NLP) plays a crucial role in language translation, utilizing computer algorithms and machine learning techniques to automatically convert text from one language to another. As a subfield of artificial intelligence, NLP focuses on empowering machines to comprehend, interpret, and generate human language.
Language translation stands out as one of the most common and vital applications of NLP. This process involves training algorithms with substantial amounts of parallel text data in multiple languages, enabling automated text translation.
Various techniques are used in NLP-based language translation, including rule-based systems, statistical machine translation, and neural machine translation. Rule-based systems follow predefined rules to translate text, while statistical machine translation employs probabilistic models to predict the most probable translation based on patterns found in the training data. Currently, neural machine translation, driven by deep learning algorithms and extensive parallel data analysis, is the most favored approach.
The significance of NLP-based language translation extends to various industries like e-commerce, tourism, and global business, showcasing its substantial impact and relevance in the modern world.

what is xlm roberta?

XLM-Roberta (Cross-lingual Language Model - Roberta) is an advanced pre-trained language model developed by Facebook AI Research. Building upon the foundation of BERT (Bidirectional Encoder Representations from Transformers), it is tailored specifically for cross-lingual applications.
Through extensive pre-training on a vast dataset comprising monolingual content from over 100 languages, XLM-Roberta exhibits a remarkable ability to comprehend and generate text in multiple languages. This pre-training equips the model with a comprehensive language representation, making it adaptable for fine-tuning on various natural language processing tasks such as machine translation, sentiment analysis, and named entity recognition.
A distinctive feature of XLM-Roberta is its capability to handle code-switching and mixed-language inputs. This flexibility allows the model to grasp and generate text containing a combination of multiple languages, proving particularly advantageous in multilingual regions where people frequently communicate using different languages.
XLM-Roberta has demonstrated exceptional performance in cross-lingual natural language processing tasks, achieving state-of-the-art results in evaluations like the XTREME benchmark. This benchmark assesses models on 40 diverse natural language understanding tasks across 11 languages, reaffirming the model's impressive capabilities in handling cross-lingual complexities.

How can we implement xlm Roberta?

Implementing XLM-Roberta involves several steps, including data preparation, model training, and fine-tuning. Here is a general overview of the process:
Data Preparation: Gather the relevant data for your task and preprocess it for training. This may involve cleaning, tokenization, and converting the data into a suitable format for the model.
Model Training: Train the XLM-Roberta model on the preprocessed data. This involves setting up the training environment, defining the model architecture, and running the training process.
Fine-Tuning: Fine-tune the pre-trained XLM-Roberta model on your specific task. This involves adding a few layers to the model and training it on your task-specific data.
Evaluation: Evaluate the performance of the model on a test set to ensure that it is producing accurate results.
Here are some resources to get started with implementing XLM-Roberta:
Hugging Face Transformers Library: This is a popular open-source library that provides a wide range of pre-trained models, including XLM-Roberta. The library also provides a set of APIs and tools for fine-tuning and evaluating the models.
PyTorch: XLM-Roberta is implemented in PyTorch, a popular deep learning framework. PyTorch provides a flexible and efficient platform for training and fine-tuning deep learning models.
XTREME: This is a benchmark dataset that provides a standardized way to evaluate the performance of cross-lingual models like XLM-Roberta on a wide range of natural language understanding tasks.
Overall, implementing XLM-Roberta requires some knowledge of deep learning and natural language processing, but there are many resources available to help you get started.

what is transformer in language translation?

The transformer is a deep learning model architecture widely utilized in natural language processing (NLP) tasks, particularly language translation. Introduced in 2017 by Vaswani et al. in their paper "Attention Is All You Need," the transformer model's key innovation lies in its self-attention mechanism. This mechanism allows the model to assess the importance of different parts of the input sequence when generating each output word. This stands in contrast to traditional sequence-to-sequence models that rely on fixed order processing using recurrent neural networks (RNNs) or convolutional neural networks (CNNs).
In the context of language translation, a transformer model comprises an encoder and a decoder. The encoder processes the input sentence, generating a sequence of hidden representations, while the decoder utilizes these representations to generate the output sentence word-by-word.
The transformer architecture has proven to be highly effective, achieving state-of-the-art performance across a broad spectrum of NLP tasks, including language translation. Its self-attention mechanism enables more precise and context-aware word generation, contributing to its superiority in handling complex language-related tasks.
.

list the types of transformers for language translation?

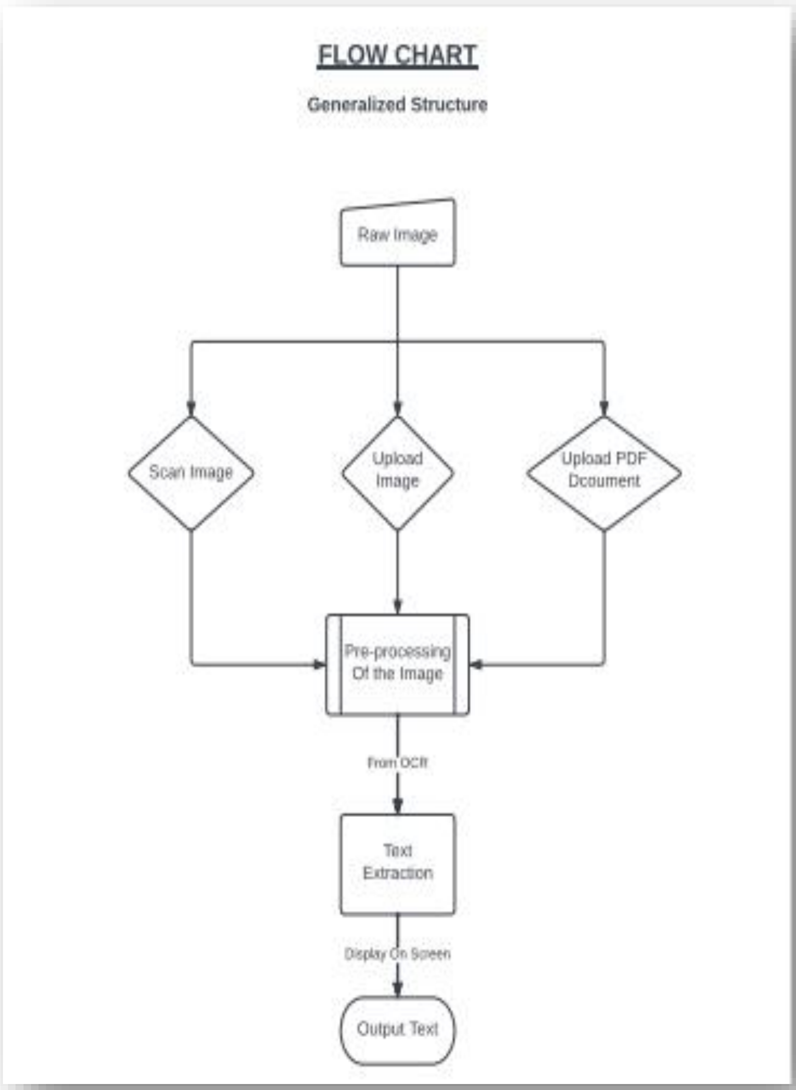
There are different types of transformers that can be used for language translation tasks. Here are some of the most common ones:
Standard transformer: This is the original transformer architecture introduced in the "Attention Is All You Need" paper. It consists of an encoder and a decoder, each with multiple layers of self-attention and feedforward neural networks.
Transformer-XL: This is an extension of the standard transformer that addresses the issue of limited context size. Transformer-XL uses a segment-level recurrence mechanism to allow the model to maintain longer-term dependencies.
BERT: BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained transformer model that can be fine-tuned for various NLP tasks, including language translation. BERT is trained on a masked language modeling task, where the model is trained to predict masked words in a sentence.
GPT (Generative Pre-trained Transformer): GPT is another pre-trained transformer model that can be fine-tuned for language translation. Unlike BERT, which is trained on a masked language modeling task, GPT is trained on a next-word prediction task.
T5 (Text-to-Text Transfer Transformer): T5 is a transformer model that is trained on a variety of NLP tasks using a unified text-to-text format. T5 can be fine-tuned for various language translation tasks by providing appropriate input-output pairs.
These are just a few examples of the types of transformers that can be used for language translation. There are many other variations and extensions of the transformer architecture that have been developed for different NLP tasks.

Which technique does XLM-RoBERTa use?

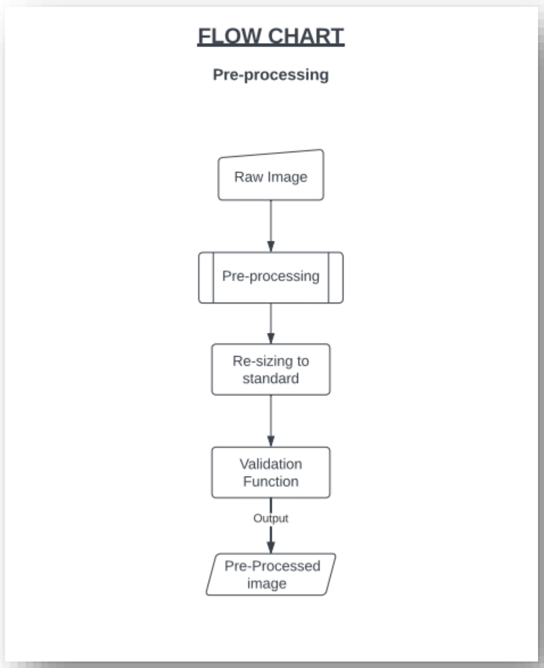
XLM-RoBERTa (Cross-lingual Language Model - RoBERTa) is a pre-trained transformer model developed by Facebook AI. It is an extension of the RoBERTa model that is pre-trained on large amounts of monolingual and parallel data in multiple languages.
XLM-RoBERTa uses a combination of several techniques to improve the performance of the RoBERTa model on cross-lingual tasks. These techniques include:
Masked language modeling (MLM): XLM-RoBERTa is pre-trained on a masked language modeling task, where the model is trained to predict masked words in a sentence in multiple languages.
Translation language modeling (TLM): XLM-RoBERTa is also pre-trained on a translation language modeling task, where the model is trained to predict the translation of a sentence from one language to another.
Multi-task learning: XLM-RoBERTa is trained on multiple tasks simultaneously, including MLM, TLM, and other language modeling tasks.
Larger training data: XLM-RoBERTa is pre-trained on a large corpus of monolingual and parallel data in multiple languages.
These techniques allow XLM-RoBERTa to learn rich representations that capture cross-lingual and language-specific information, which can be fine-tuned for various cross-lingual NLP tasks, such as language translation, cross-lingual document classification, and cross-lingual named entity recognition.

9.11 FLOW CHARTS:

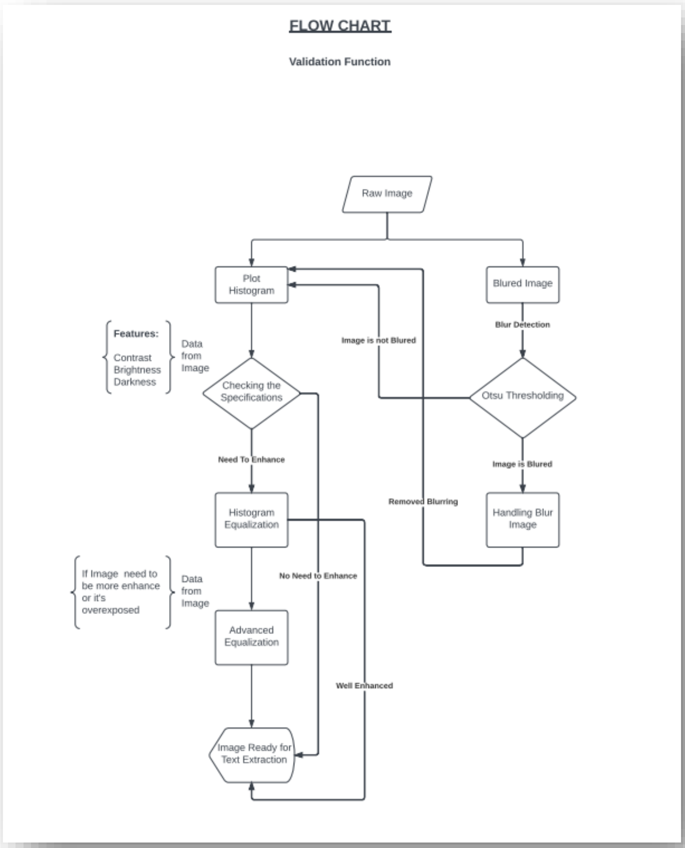
General Flow chart of the Project:



Preprocessing Flow chart of the Project:



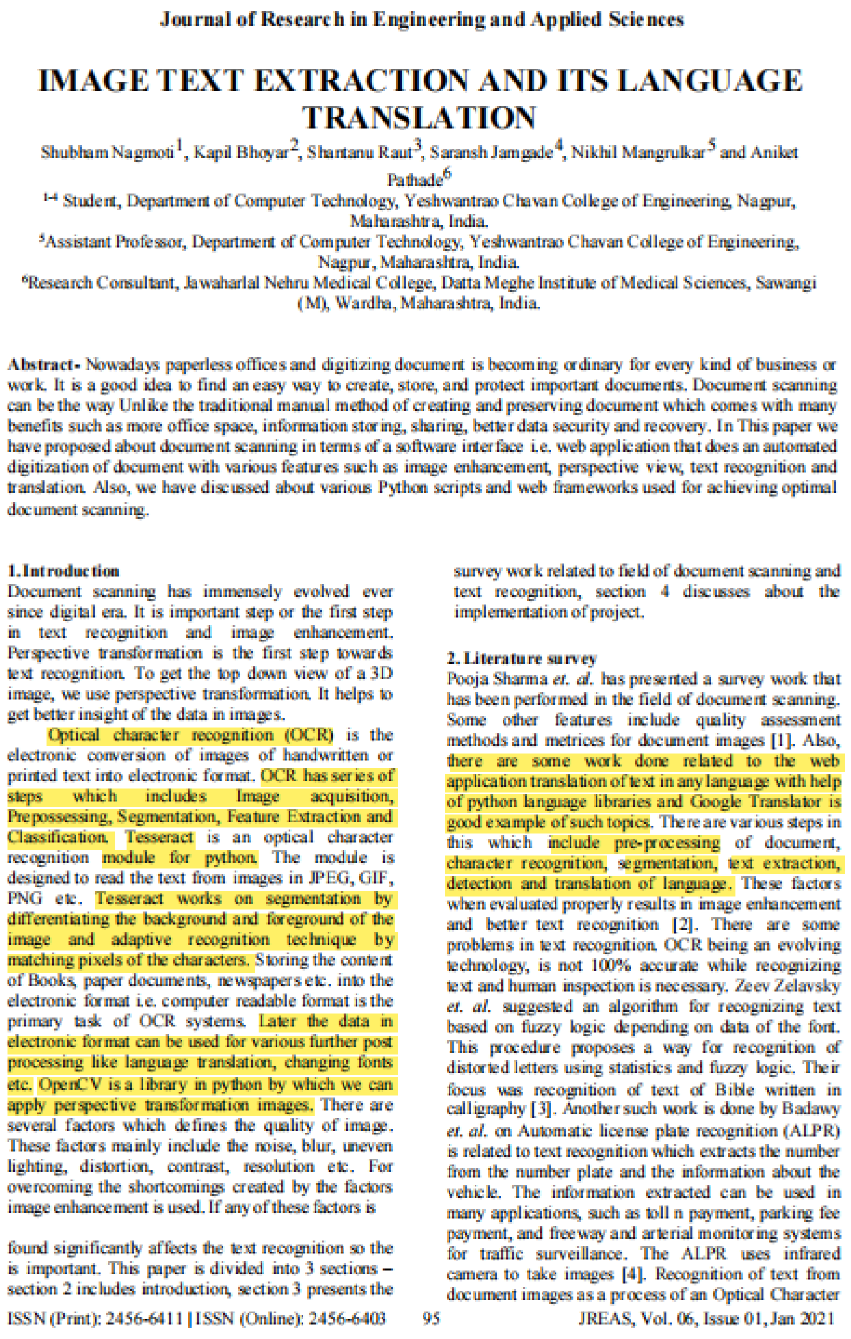
Validation Flow chart of the Project:



9.12 ARTICLES RELATED TO THE OPTICAL CHARACTER RECOGNITION:

ARTICLE #1

Read More:
https://www.researchgate.net/publication/351422903_IMAGE_TEXT_EXTRACTION_AND_ITS_LANGUAGE_TRANSLATION



Journal of Research in Engineering and Applied Sciences

IMAGE TEXT EXTRACTION AND ITS LANGUAGE TRANSLATION

Shubham Nagmoti[1], Kapil Bhoyar[2], Shantanu Raut[3], Saransh Jamgade[4], Nikhil Mangrulkar[5] and Aniket Pathade[6]
[1-4] Student, Department of Computer Technology, Yeshwantrao Chavan College of Engineering, Nagpur, Maharashtra, India.
[5]Assistant Professor, Department of Computer Technology, Yeshwantrao Chavan College of Engineering, Nagpur, Maharashtra, India.
[6]Research Consultant, Jawaharlal Nehru Medical College, Datta Meghe Institute of Medical Sciences, Sawangi (M), Wardha, Maharashtra, India.

Abstract- Nowadays paperless offices and digitizing document is becoming ordinary for every kind of business or work. It is a good idea to find an easy way to create, store, and protect important documents. Document scanning can be the way Unlike the traditional manual method of creating and preserving document which comes with many benefits such as more office space, information storing, sharing, better data security and recovery. In This paper we have proposed about document scanning in terms of a software interface i.e. web application that does an automated digitization of document with various features such as image enhancement, perspective view, text recognition and translation. Also, we have discussed about various Python scripts and web frameworks used for achieving optimal document scanning.

1. Introduction
Document scanning has immensely evolved ever since digital era. It is important step or the first step in text recognition and image enhancement. Perspective transformation is the first step towards text recognition. To get the top down view of a 3D image, we use perspective transformation. It helps to get better insight of the data in images.

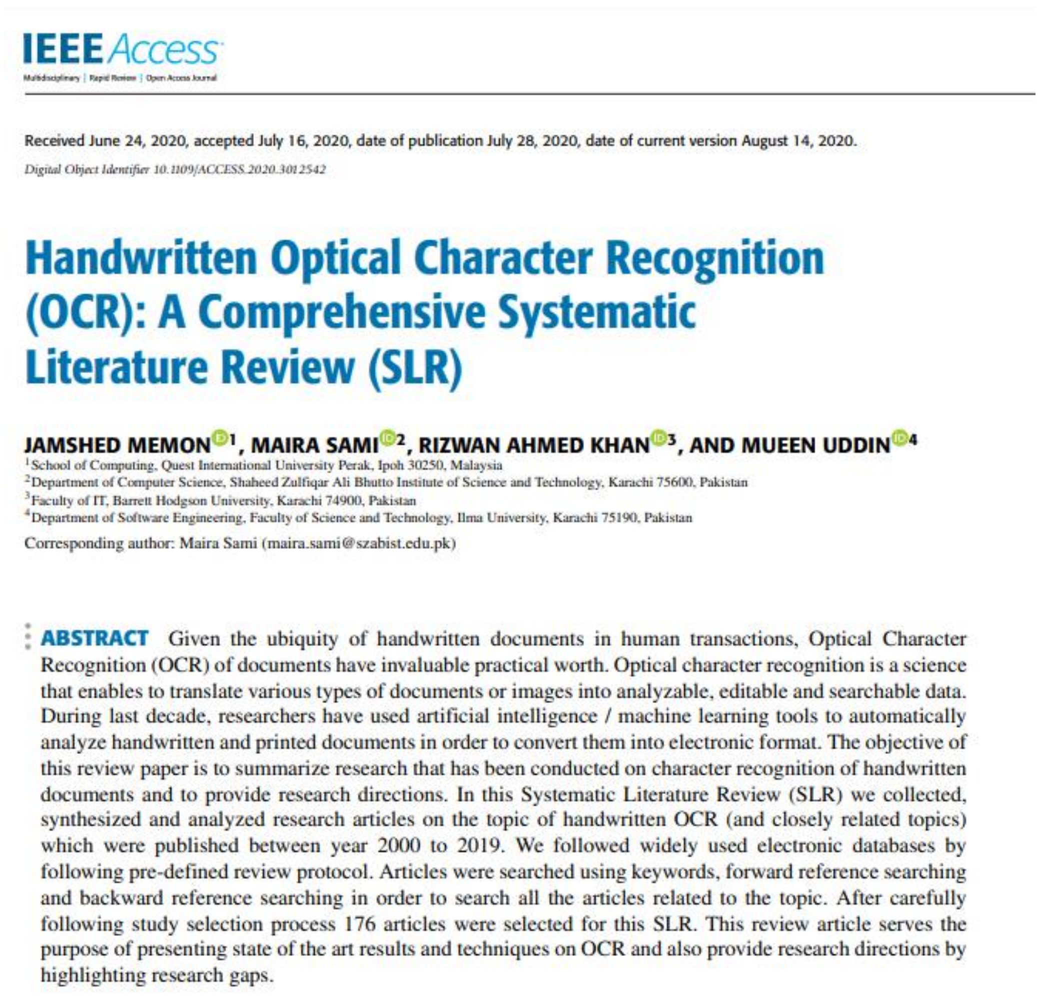
Optical character recognition (OCR) is the electronic conversion of images of handwritten or printed text into electronic format. OCR has series of steps which includes Image acquisition, Prepossessing, Segmentation, Feature Extraction and Classification. Tesseract is an optical character recognition module for python. The module is designed to read the text from images in JPEG, GIF, PNG etc. Tesseract works on segmentation by differentiating the background and foreground of the image and adaptive recognition technique by matching pixels of the characters. Storing the content of Books, paper documents, newspapers etc. into the electronic format i.e. computer readable format is the primary task of OCR systems. Later the data in electronic format can be used for various further post processing like language translation, changing fonts etc. OpenCV is a library in python by which we can apply perspective transformation images. There are several factors which defines the quality of image. These factors mainly include the noise, blur, uneven lighting, distortion, contrast, resolution etc. For overcoming the shortcomings created by the factors image enhancement is used. If any of these factors is

found significantly affects the text recognition so the is important. This paper is divided into 3 sections – section 2 includes introduction, section 3 presents the survey work related to field of document scanning and text recognition, section 4 discusses about the implementation of project.

2. Literature survey
Pooja Sharma *et. al.* has presented a survey work that has been performed in the field of document scanning. Some other features include quality assessment methods and metrices for document images [1]. Also, there are some work done related to the web application translation of text in any language with help of python language libraries and Google Translator is good example of such topics. There are various steps in this which include pre-processing of document, character recognition, segmentation, text extraction, detection and translation of language. These factors when evaluated properly results in image enhancement and better text recognition [2]. There are some problems in text recognition. OCR being an evolving technology, is not 100% accurate while recognizing text and human inspection is necessary. Zeev Zelavsky *et. al.* suggested an algorithm for recognizing text based on fuzzy logic depending on data of the font. This procedure proposes a way for recognition of distorted letters using statistics and fuzzy logic. Their focus was recognition of text of Bible written in calligraphy [3]. Another such work is done by Badawy *et. al.* on Automatic license plate recognition (ALPR) is related to text recognition which extracts the number from the number plate and the information about the vehicle. The information extracted can be used in many applications, such as toll n payment, parking fee payment, and freeway and arterial monitoring systems for traffic surveillance. The ALPR uses infrared camera to take images [4]. Recognition of text from document images as a process of an Optical Character

ARTICLE #2

Read More:
https://ieeexplore.ieee.org/document/9151144



IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 24, 2020, accepted July 16, 2020, date of publication July 28, 2020, date of current version August 14, 2020.
Digital Object Identifier 10.1109/ACCESS.2020.3012542

Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)

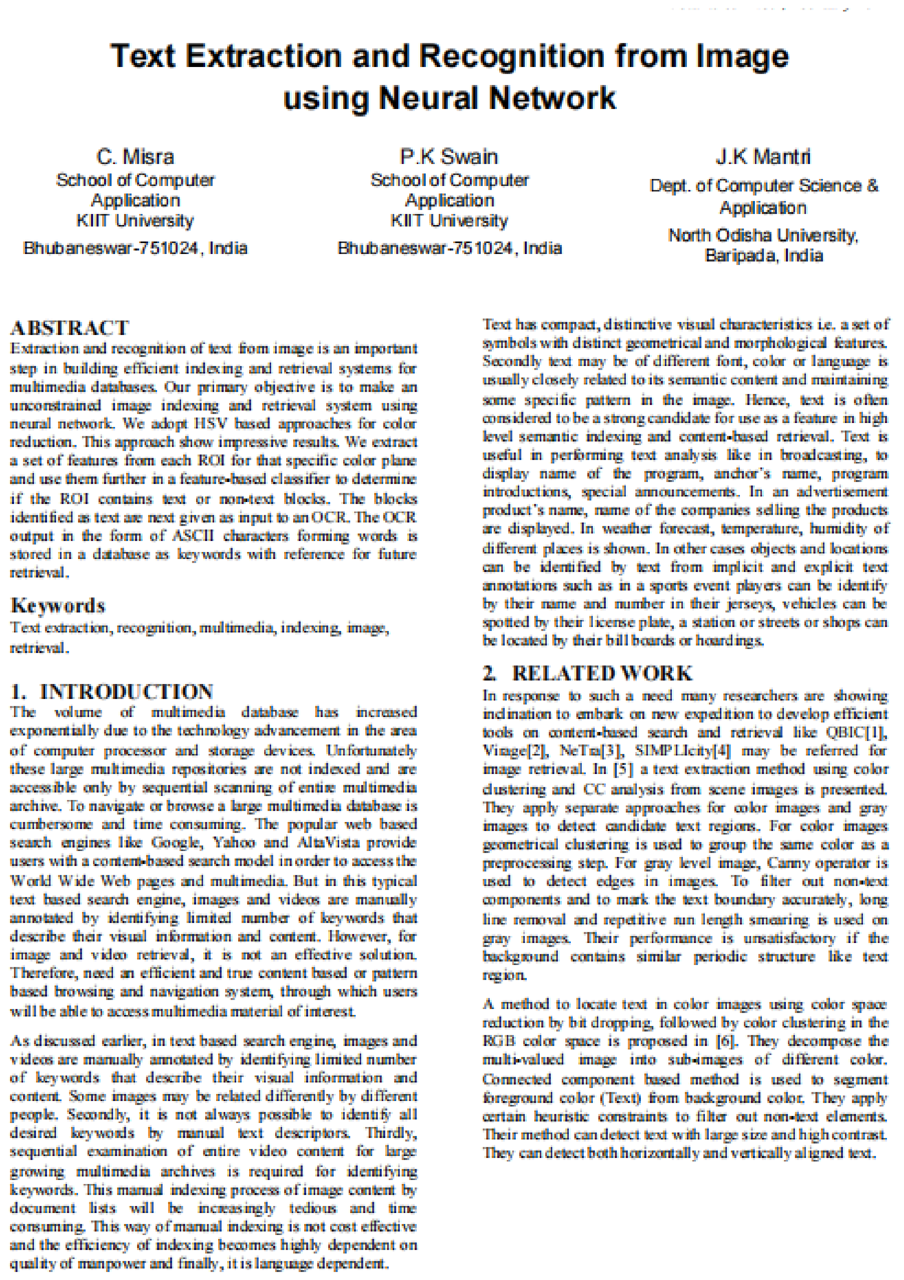
JAMSHED MEMON[1], MAIRA SAMI[2], RIZWAN AHMED KHAN[3], AND MUEEN UDDIN[4]
[1]School of Computing, Quest International University Perak, Ipoh 30250, Malaysia
[2]Department of Computer Science, Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, Karachi 75600, Pakistan
[3]Faculty of IT, Barrett Hodgson University, Karachi 74900, Pakistan
[4]Department of Software Engineering, Faculty of Science and Technology, Ilma University, Karachi 75190, Pakistan
Corresponding author: Maira Sami (maira.sami@szabist.edu.pk)

ABSTRACT Given the ubiquity of handwritten documents in human transactions, Optical Character Recognition (OCR) of documents have invaluable practical worth. Optical character recognition is a science that enables to translate various types of documents or images into analyzable, editable and searchable data. During last decade, researchers have used artificial intelligence / machine learning tools to automatically analyze handwritten and printed documents in order to convert them into electronic format. The objective of this review paper is to summarize research that has been conducted on character recognition of handwritten documents and to provide research directions. In this Systematic Literature Review (SLR) we collected, synthesized and analyzed research articles on the topic of handwritten OCR (and closely related topics) which were published between year 2000 to 2019. We followed widely used electronic databases by following pre-defined review protocol. Articles were searched using keywords, forward reference searching and backward reference searching in order to search all the articles related to the topic. After carefully following study selection process 176 articles were selected for this SLR. This review article serves the purpose of presenting state of the art results and techniques on OCR and also provide research directions by highlighting research gaps.

ARTICLE #3

Read More:

https://www.researchgate.net/publication/224890763_Text_Extraction_and_Recognition_from_Image_using_Neural_Network?enrichId=rgreq-38667e726b5b0e73bc046753ccdb1021-XXX&enrichSource=Y292ZXJQYWdlOzIyNDg5MDc2MztBUzo0NjEyNTQ0NzQxMTMwMjRAMTQ4Njk4MzA0NDgyNA%3D%3D&el=1_x_2&_esc=publicationCoverPdf



Text Extraction and Recognition from Image using Neural Network

C. Misra
School of Computer Application
KIIT University
Bhubaneswar-751024, India

P.K Swain
School of Computer Application
KIIT University
Bhubaneswar-751024, India

J.K Mantri
Dept. of Computer Science & Application
North Odisha University, Baripada, India

ABSTRACT
Extraction and recognition of text from image is an important step in building efficient indexing and retrieval systems for multimedia databases. Our primary objective is to make an unconstrained image indexing and retrieval system using neural network. We adopt HSV based approaches for color reduction. This approach show impressive results. We extract a set of features from each ROI for that specific color plane and use them further in a feature-based classifier to determine if the ROI contains text or non-text blocks. The blocks identified as text are next given as input to an OCR. The OCR output in the form of ASCII characters forming words is stored in a database as keywords with reference for future retrieval.

Keywords
Text extraction, recognition, multimedia, indexing, image, retrieval.

1. INTRODUCTION

The volume of multimedia database has increased exponentially due to the technology advancement in the area of computer processor and storage devices. Unfortunately these large multimedia repositories are not indexed and are accessible only by sequential scanning of entire multimedia archive. To navigate or browse a large multimedia database is cumbersome and time consuming. The popular web based search engines like Google, Yahoo and AltaVista provide users with a content-based search model in order to access the World Wide Web pages and multimedia. But in this typical text based search engine, images and videos are manually annotated by identifying limited number of keywords that describe their visual information and content. However, for image and video retrieval, it is not an effective solution. Therefore, need an efficient and true content based or pattern based browsing and navigation system, through which users will be able to access multimedia material of interest.

As discussed earlier, in text based search engine, images and videos are manually annotated by identifying limited number of keywords that describe their visual information and content. Some images may be related differently by different people. Secondly, it is not always possible to identify all desired keywords by manual text descriptors. Thirdly, sequential examination of entire video content for large growing multimedia archives is required for identifying keywords. This manual indexing process of image content by document lists will be increasingly tedious and time consuming. This way of manual indexing is not cost effective and the efficiency of indexing becomes highly dependent on quality of manpower and finally, it is language dependent.

Text has compact, distinctive visual characteristics i.e. a set of symbols with distinct geometrical and morphological features. Secondly text may be of different font, color or language is usually closely related to its semantic content and maintaining some specific pattern in the image. Hence, text is often considered to be a strong candidate for use as a feature in high level semantic indexing and content-based retrieval. Text is useful in performing text analysis like in broadcasting, to display name of the program, anchor's name, program introductions, special announcements. In an advertisement product's name, name of the companies selling the products are displayed. In weather forecast, temperature, humidity of different places is shown. In other cases objects and locations can be identified by text from implicit and explicit text annotations such as in a sports event players can be identify by their name and number in their jerseys, vehicles can be spotted by their license plate, a station or streets or shops can be located by their bill boards or hoardings.

2. RELATED WORK

In response to such a need many researchers are showing inclination to embark on new expedition to develop efficient tools on content-based search and retrieval like QBIC[1], Virage[2], NeTra[3], SIMPLIcity[4] may be referred for image retrieval. In [5] a text extraction method using color clustering and CC analysis from scene images is presented. They apply separate approaches for color images and gray images to detect candidate text regions. For color images geometrical clustering is used to group the same color as a preprocessing step. For gray level image, Canny operator is used to detect edges in images. To filter out non-text components and to mark the text boundary accurately, long line removal and repetitive run length smearing is used on gray images. Their performance is unsatisfactory if the background contains similar periodic structure like text region.

A method to locate text in color images using color space reduction by bit dropping, followed by color clustering in the RGB color space is proposed in [6]. They decompose the multi-valued image into sub-images of different color. Connected component based method is used to segment foreground color (Text) from background color. They apply certain heuristic constraints to filter out non-text elements. Their method can detect text with large size and high contrast. They can detect both horizontally and vertically aligned text.

9.13 BACKGROUND OF THE PROJECT AND REVIEWS:

Text extraction from images poses significant challenges in digital image processing. Detecting and recognizing text from images is a complex process. However, with sophisticated algorithms, computer software can extract text from images. Despite the complexity, such technology can find practical applications in various environments. Currently, different types of language translators exist, including voice-based and keyboard-based translators. However, these translators may not always be user-friendly. The goal of this work is to demonstrate the seamless integration of text and interactive visualization imagery. Leveraging the capabilities of Android device cameras, text extraction becomes feasible, and implementing the algorithm using Java language makes the process even more accessible.
Given the vast number of mobile users globally, the simplicity of capturing images to extract text using mobile devices is evident. This project aims to achieve text extraction from images and facilitate translation, making it a valuable tool for mobile users to understand and communicate with content in various languages.
Captured text information from camera in natural scene images can serve as indicative marks in many image based applications such as assistive navigation, auxiliary reading, image retrieval, scene understanding, etc. Extracting text from natural scene images is a more challenging problem as compared to scanned document because of complex backgrounds and also large variations of text patterns such as font, color, scale, intensity and orientation. Therefore, to extract text from camera captured images, text detection & extraction is an important and essential step which computes the sub regions of the images containing text characters or strings. Once the image is captured from camera, the image went through various processes whose task is to detect the text within the image and extract those texts then translates that text.

The main purpose to serve the people with an enhanced and useful application that brings them much faster as the world is growing. To capture the idea of dealing with the android application is chosen due to the vast and hands on experience of the users to the android app. As far as project idea is concern, it gives us the full exposure to use of two languages in parallel.

The project is basically allowing the user to enter the text in the source language and it will be translated into the target language. This work is done by the image detection. First the user will scan the image containing some text. Then it will be converted into the readable text by the help of OCR (Optical Character Recognition). Then it will be translated into the target language by using a technique called NLP (Natural Language Processing).Captured text information from camera in natural scene images can serve as indicative marks in many image based applications such as assistive navigation, auxiliary reading, image retrieval, scene understanding, etc. Extracting text from natural scene images is a more challenging problem as compared to scanned document because of complex backgrounds and also large variations of text patterns such as font, color, scale, intensity and orientation. Therefore, to extract text from camera captured images, text detection & extraction is an important and essential step which computes the sub regions of the images containing text characters or strings. Once the image is captured from camera, the image went through various processes whose task is to detect the text within the image and extract those texts then translates that text.

There was a lot of people who doesn’t know how to read English and Urdu so this application translates the text from English to Urdu or Urdu to English (Dual Zone) and also have a facility of voice command

As the problem states that there are most of the people which doesn’t understand and read the English and there are most of the people who doesn’t know how to read Urdu but understands Urdu so this app will be beneficial for those. This app solves a problem of three types of people:

1.) Once a person doesn’t Understand the English so this app converts document through image processing in Urdu
2.) Once a person doesn’t understand or read Urdu so this app converts Urdu document through image processing in English.
3.) Once a person doesn’t know how to read Urdu/English but he/she Understand English /Urdu so the app has also ability to read the document

Nowadays paperless offices and digitizing document is becoming ordinary for every kind of business or work. It is a good idea to find an easy way to create, store, and protect important documents. Document scanning can be the way Unlike the traditional manual method of creating and preserving document which comes with many benefits such as more office space, information storing, sharing, better data security and recovery. In This paper we have proposed about document scanning in terms of a software interface The main purpose to serve the people with an enhanced and useful application that brings them much faster as the world is growing. To capture the idea of dealing with the android application is chosen due to the vast and hands on experience of the users to the android app. As far as project idea is concern, it gives us the full exposure to use of two languages in parallel.

10. Future work

- We can enhance the ability of OCR on Handwritten Format.
- Different Types of fonts can be taken into consideration.
- More Advanced Techniques can be implemented for the image extraction.
- Post –processing techniques can be implemented on the Input image.
- New techniques can be made to produce highly definite results.
- Gamma correction Algorithm can be used to produce better results.
- Image segmentation in detail can be viewed.

11. Achievements

So as we are working on the image detection and language translation project that deals with the image detection and text extraction from OCR i.e. optical character recognition and the language translation is done by NLP i.e. Natural language processing.

11.1 OPTICAL CHARACTER RECOGNITION

We have achieved a blog reading that enhanced the ability to work and implement our project idea:

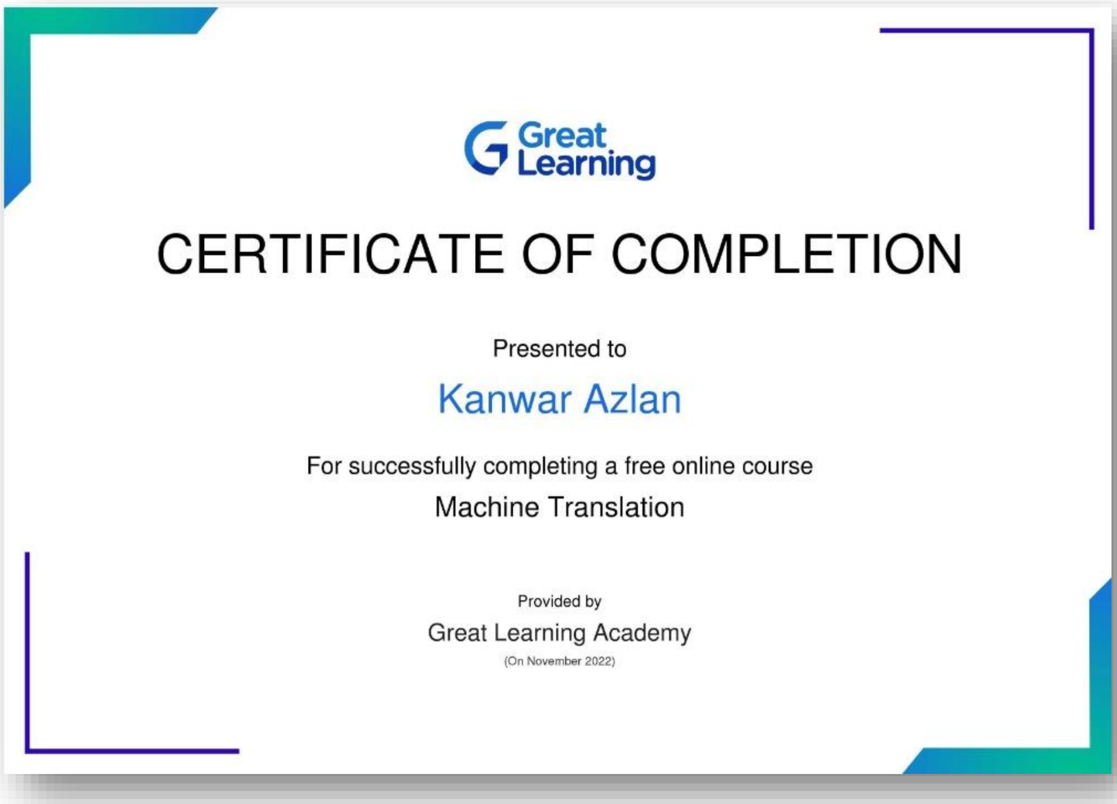
https://www.mygreatlearning.com/blog/introduction-to-image-pre-processing/

furthermore, we have researched about the OCR and came up with this research paper in which a similar project was made.

https://www.researchgate.net/publication/351422903_IMAGE_TEXT_EXTRACTION_AND_ITS_LANGUAGE_TRANSLATION

11.2 NATURAL LANGUGAE PROCESSING

As NLP is the core part of our project so we achieved a NLP Course completion certificate.



Great Learning

CERTIFICATE OF COMPLETION

Presented to

Kanwar Azlan

For successfully completing a free online course

Machine Translation

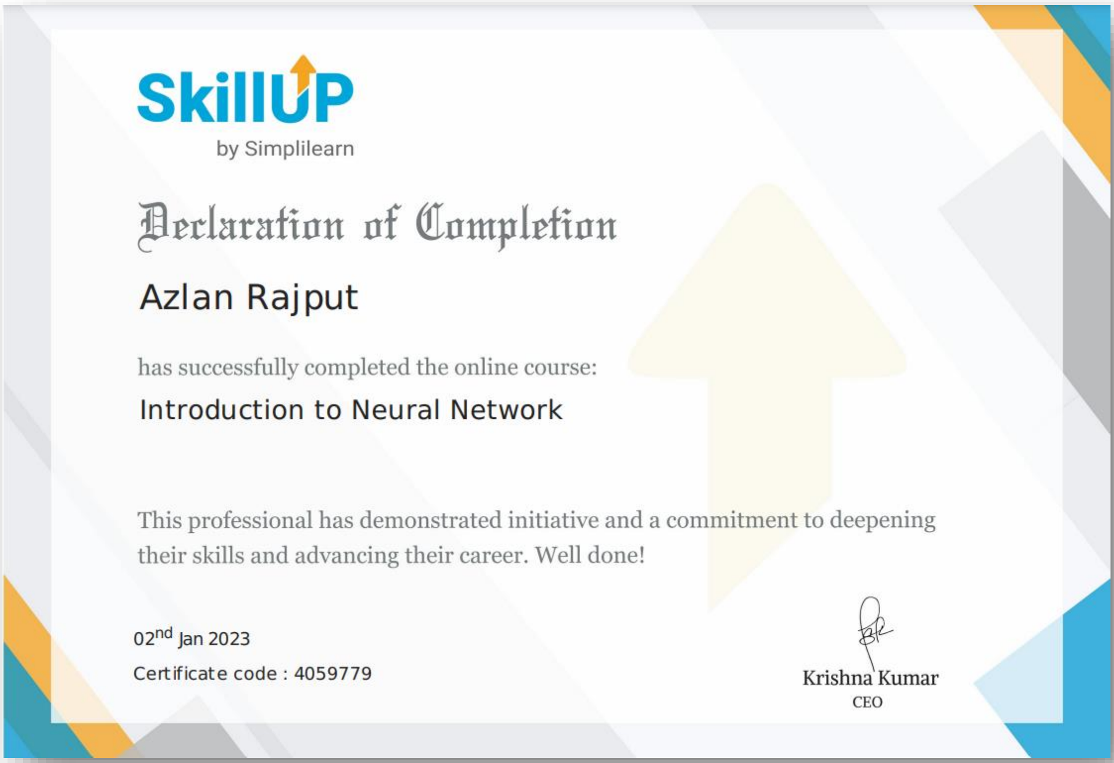
Provided by

Great Learning Academy

(On November 2022)

11.3 NEURAL NETWORKS

NLP is a subpart of Machine learning and ML is a subpart of Artificial Intelligence so neural networks can be very helpful for us to implement NLP and to make language translation more successful.



SkillUP by Simplilearn

Declaration of Completion

Azlan Rajput

has successfully completed the online course:

Introduction to Neural Network

This professional has demonstrated initiative and a commitment to deepening their skills and advancing their career. Well done!

02nd Jan 2023

Certificate code : 4059779

[illegible]

Krishna Kumar
CEO

12. Appendices

121 Acknowledgement

Final year project is the most significant part of every institute as well as students who willing take this opportunity to enhance their professional skills. This Report is also a way of presenting the efforts of students in the regard of FYP and can take a better understanding of the actual hard work. From the Head of Usman Institute to all the teachers as well as HOD’s and supervisor’s, they really give their precious time in order to make the FYP successful, So, from my whole FYP Group, I hereby thanks to all the supporting Team that make this happen.

12.2 Introduction

Text extraction from images is a complex endeavor in the field of digital image processing. Detecting and recognizing text within images poses significant challenges. Advanced computer software can achieve text extraction using intricate algorithms, but widespread implementation remains a challenge in the existing environment.
While language translators like voice-based and keyboard-based options exist, their ease of use is limited. This work aims to demonstrate the seamless connection between text and interactive visualization imagery. Leveraging the capabilities of Android device cameras, this project seeks to implement text extraction and translation with ease, utilizing Java language for efficient implementation.
The project's purpose is to enable text extraction from images and subsequent translation. Captured text information from natural scene images can serve as indicative markers in various image-based applications, including assistive navigation, auxiliary reading, image retrieval, and scene understanding.
Extracting text from images in natural scenes presents greater complexity compared to scanned documents, mainly due to intricate backgrounds and significant variations in text patterns, such as font, color, scale, intensity, and orientation.
To achieve text extraction from camera-captured images, the critical step involves text detection and extraction, calculating the regions containing text characters or strings. Upon capturing the image, various processes are initiated to detect and extract the text, which is then translated, offering a valuable tool for mobile users to effortlessly capture and understand text from images.

The main purpose to serve the people with an enhanced and useful application that brings them much faster as the world is growing. To capture the idea of dealing with the android application is chosen due to the vast and hands on experience of the users to the android app. As far as project idea is concern, it gives us the full exposure to use of two languages in parallel.

The project is basically allowing the user to enter the text in the source language and it will be translated into the target language. This work is done by the image detection. First the user will scan the image containing some text. Then it will be converted into the readable text by the help of OCR (Optical Character Recognition). Then it will be translated into the target language by using a technique called NLP (Natural Language Processing).

12.3 background and literature review chapter

Text extraction from images is a complex and challenging area within digital image processing. The process involves detecting and recognizing text from images, which necessitates the use of intricate computer software algorithms. However, due to the complexity involved, implementing such software for widespread use remains a challenge in the current environment.
Language translators, such as voice-based and keyboard-based options, are available, but their usability can be cumbersome. This project aims to showcase the close relationship between text and interactive visualization imagery. Leveraging the capabilities of Android device cameras, text extraction becomes feasible, with Java language providing an efficient implementation method.
The primary purpose of this project is to implement text extraction from images and subsequently translate the text. The captured text information from the camera in natural scene images holds potential for various image-based applications, including assistive navigation, auxiliary reading, image retrieval, and scene understanding.
Extracting text from images in natural scenes is more demanding compared to scanned documents due to intricate backgrounds and significant variations in text patterns, encompassing font, color, scale, intensity, and orientation.
As a crucial and essential step in text extraction from camera-captured images, text detection and extraction processes compute the regions containing text characters or strings. After capturing the image, various processes are employed to detect and extract the text, subsequently translating it.
Overall, this project aims to address the challenges of text extraction from images and leverage the potential applications of such technology in various domains.
There was a lot of people who doesn’t know how to read English and Urdu so this application translates the text from English to Urdu or Urdu to English (Dual Zone) and also have a facility of voice command

As the problem states that there are most of the people which doesn’t understand and read the English and there are most of the people who doesn’t know how to read Urdu but understands Urdu so this app will be beneficial for those. This app solves a problem of three types of people:

4.) Once a person doesn’t Understand the English so this app converts document through image processing in Urdu
5.) Once a person doesn’t understand or read Urdu so this app converts Urdu document through image processing in English.
6.) Once a person doesn’t know how to read Urdu/English but he/she Understand English /Urdu so the app has also ability to read the document

Nowadays paperless offices and digitizing document is becoming ordinary for every kind of business or work. It is a good idea to find an easy way to create, store, and protect important documents.

Document scanning can be the way Unlike the traditional manual method of creating and preserving document which comes with many benefits such as more office space, information storing, sharing, better data security and recovery.

In This paper we have proposed about document scanning in terms of a software interface The main purpose to serve the people with an enhanced and useful application that brings them much faster as the world is growing.

To capture the idea of dealing with the android application is chosen due to the vast and hands on experience of the users to the android app. As far as project idea is concern, it gives us the full exposure to use of two languages in parallel.

12.3.1 LITERATURE REVIEW

Now a day it’s a vast experience to handle a software where you can extract some information related to an image. To be more precise it’s a powerful work experience to build something like OCR that actually make your work easier. OCR is basically an Optical Character Recognition Tool that helps to extract a text from image. Scan your desired image and extract the text with the help of OCR.

It works really simple in a way that it separates the background and foreground of the image and extract the pixels out of the image, through which the text can be extracted. Now Comes the most important and vital part of our project. The Text translation is the legit work to be done. It basically involves AI and ML that include some transformers and a technique called NLP that creates the opportunity of the language translation or machine translation.

By processing natural language to translate it into another natural language, machine translation (MT) plays a significant role in helping linguists, sociologists, computer scientists, and other professionals. And with the massive interchange of information between various regions with various regional languages over the past few of years, this demand has increased enormously. There are many difficulties with machine translation, some of which include:

Two given languages may have entirely distinct structures because of the following reasons:
a) Not every word in one language has an equivalent word in another language; and
b) Words can convey multiple meanings.

Due to these difficulties and numerous more, MT has been a popular subject of study for more than 50 years. Numerous approaches have been put forth in the past with the intent of either enhancing the calibre of the translations produced by them or investigating the resilience of these systems by evaluating how well they work across a wide range of languages.

We discuss statistical methods (in particular word- and phrase-based methods) and neural methods in our survey of the literature since they have produced cutting-edge findings in a variety of important languages.

The process of transforming the original language's (the source language, or SL) structure into another language's (the target language, or TL) is known as translation. This procedure fits the definition of intra-lingual translation, which is the understanding of verbal cues using some other languages (Munday, 2016: 8). Larson (1998:3) offers a different definition, claiming that translation is the process of transferring meaning from texts written in the source language into texts written in the receptor language.

To complete the procedure, the semantic structure must be changed from the form of the ST to the form of the TT. Only the forms change; the meaning remains unchanged. According to Catford (1978: 20), translation is the process of changing text in one language (SL) by.

The term "textual material" emphasizes that in the process of translation, only specific parts of the source language text are replaced with their equivalent target language counterparts. For instance, when translating the Indonesian text "Berapa umurmu?" into English as "How old are you?", the source language grammar and lexis are substituted with the corresponding target language grammar and lexis. According to Newmark (1988: 21), there are two approaches in the translation process.
In a recent paper by N. Mangrulkar (2021), a proposed web application for document scanning is introduced. The application automates document digitization with various features, including image enhancement, perspective view, text recognition, and translation. The paper also delves into the utilization of Python scripts and web frameworks to achieve optimal document scanning.
In another research by Misra, Swain, & Mantri (2012), the primary objective is to create an unconstrained image indexing and retrieval system using neural networks. HSV-based approaches for color reduction were adopted to achieve impressive results.
In a study by Liu (2021) on transformers, a new approach called Re-Transformer is introduced, which modifies the basic architecture for improved system performance. The paper outlines four refinements made to the traditional approach.
An article by D. Bahdanau (2016) discusses neural machine translation as a recently proposed approach. Unlike traditional statistical machine translation, neural machine translation aims to build a single neural network for joint tuning to maximize translation performance.
The researchers, h. Sutskever (2014), conclude that while deep neural networks (DNNs) excel in tasks with large labeled training sets, they face challenges in mapping sequences to sequences. The paper presents a general end-to-end approach to sequence learning, making minimal assumptions on the sequence structure.
In another conference proceeding by B. Shi (2015), the problem of scene text recognition in image-based sequence recognition is investigated. Scene text recognition is considered a crucial and complex task within image-based sequence recognition.

Additionally, the Transformer translation model (Vaswani et al., 2017) is extended with a new context encoder to handle discourse phenomena problematic for Transformers. Zhang (2017) introduces this extension to incorporate document-level context into the original encoder and decoder.

[1] According to this paper (N.Mangrulkar, 2021),

[2] In this research (Misra, Swain, & Mantri, 2012),

[3] Regarding the transformers (Liu, 2021),

[4] In this Article (D.Bahdanau, 2016),

[5] The researchers conclude (h.Sutskever, 2014),

[6] According to this conference proceeding (B.Shi, 2015),

[7] In the addition of Transfomers (Zhang, 2017),

[8] The Article states (Baniata, 2021),

[9] Researchers describes as (Sefara, 2021),

Some research has been conducted on web application translation, where text in any language is translated using Python language libraries. A notable example is Google Translator, which demonstrates the effectiveness of such approaches. The process involves several steps, including document pre-processing, character recognition, segmentation, text extraction, language detection, and translation.
For details:

https://www.researchgate.net/publication/351422903_IMAGE_TEXT_EXTRACTION_AND_ITS_LANGUAGE_TRANSLATION

The above link is the similar project based on text Extraction and Language translation.

This is the similar image of OCR representation.

<u>12.3.2 DESCRIPTION</u>

This is the similar image of OCR representation.

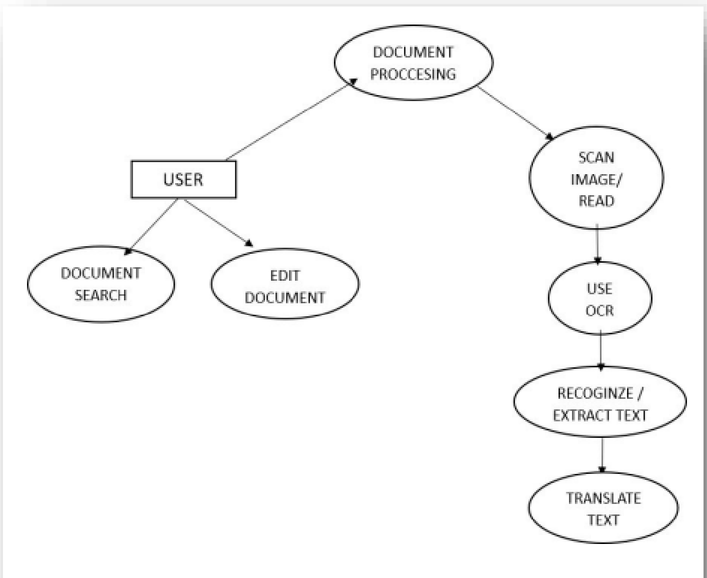


Fig. 1: Block diagram of OCR system

Figure #12.3.2: Block Diagram

The above Fig. 12.3.2 The block diagram illustrates the OCR system and provides an overview of its processing steps. The first step involves scanning (reading) the input, which serves as the input to the OCR methodology. Using the Tesseract module, text is extracted from the input image. This extracted text can then be translated into different languages.
Perspective transformation is the initial stage in text recognition. It enables us to obtain a top-down view of a 3D image, offering better insights into the image data. OpenCV, a Python library, allows us to apply perspective transformation to images. For text recognition, Tesseract is utilized as an optical character recognition module for Python. It is designed to read text from images in formats like JPEG, GIF, PNG, and more. Tesseract operates through segmentation, distinguishing between the background and foreground of the image, and employs adaptive recognition techniques by matching pixels of the characters.

12.4 HARDWARE AND SOFTWARE ANALYSIS

Front End:

2- **Flutter (Framework):**
We are using Flutter as our front-end tool. Basically all other frameworks are available for the mobile application development like Django, flask etc. But Flutter is widely used now a days and is created by Google.
A Platform for the Front end tool that helps us to take better approach in the mobile application development.
Dart is the language used by flutter framework. Flutter apps are written in the Dart language and make use of many of the language's more advanced features. While writing and debugging an application, Flutter runs in the Dart virtual machine, which features a just-in-time execution engine.

Back End:

3- **OCR (Optical Character Recognition):**
Optical character recognition (OCR) is sometimes referred to as text recognition. An OCR program extracts and repurposes data from scanned documents, camera images and image-only pdfs.

OCR software singles out letters on the image, puts them into words and then puts the words into sentences, thus enabling access to and editing of the original content. It also eliminates the need for manual data entry.

4- **NLP (Natural Language Processing):**
Natural Language Processing (NLP) is a combination of computer science, information engineering, and artificial intelligence (AI). While people use words to communicate, computers operate with the language of numbers.

5- **Python (Programming Language):**
Python is widely used and most effective high-level programming language. We Train our model with the data of source language and target language using python.

Requirements:

- As a Front-end developer, I want flutter framework to be used in our project so that it will help us in the easy execution of the mobile application.
- As a Back end developer, I want Python to be chosen as our programming language so that we have a great scope and it provides many libraries built in that will help us throughout the process.
- As a programmer or student, I want to first train or test our program so train a model that consists of desired data and expected to give desired output can do it.
- As a student, I want to learn NLP and its execution so that it will help us in the domain of language translation and it will be helpful in order to train our model.

12.5 PROJECT ARCHITECTURE:

12.5.1 INTRODUCTION:

The Project Architecture is centered around OCR (Optical Character Recognition), utilizing image processing techniques to interact with images and perform various manipulations. Text recognition, also known as optical character recognition (OCR), involves extracting and reusing data from scanned documents, camera photos, and image-only PDFs through an OCR application. The OCR software enables access and editing of the original content by isolating letters from the image, converting them into words, and then assembling the words into sentences. This eliminates the need for manual data entry.

12.5.2 HISTORY

Ray Kurzweil founded Kurzweil Computer Products, Inc. in 1974, introducing the Omni-font optical character recognition (OCR) equipment capable of reading text in various typefaces. Recognizing its potential, he envisioned its ideal application as a machine-learning aid for the blind, leading him to develop a reading machine that could convert text to speech. In 1980, Kurzweil sold his business to Xerox, which saw the value of advancing text conversion from paper to computers.
The popularity of OCR technology grew during the early 1990s when digitizing old newspapers. Over the years, OCR technology has made significant advancements, now offering nearly perfect accuracy. Cutting-edge techniques have automated complex document-processing operations using today's technologies.

The only way to digitally format documents prior to the development of OCR technology was to manually retype the text. This took a lot of time and unavoidably contained typographical and factual errors. OCR services are now widely accessible to the general public. For instance, documents can be scanned and stored on your smartphone using Google Cloud Vision OCR.

12.5.3 WORKING OF THE ARCHITECTURE

Optical Character Recognition (OCR) employs a scanner to process the physical form of a document. The OCR software analyzes the scanned document, converting it into a two-color or black-and-white version. The scanned image or bitmap is carefully examined to distinguish bright areas as the background and dark areas as characters that require recognition.

Once the black regions are processed, the OCR system identifies alphabetical or numerical digits. This recognition phase typically focuses on individual characters, words, or sections of text at a time. To accomplish character identification, one of two algorithms, namely pattern recognition or feature recognition, is utilized.

By integrating both hardware and software, OCR systems transform physical, printed documents into machine-readable text. Hardware components like optical scanners or dedicated circuit boards are employed to copy or read the text, while the more intricate processing tasks are handled by the OCR software. This combination of hardware and software enables the conversion of printed content into machine-readable text, streamlining document digitization and making it easier to work with the text in digital form.We are using OCR as a tool to extract text from image.

It works in a way that an input image is passed through 3 main ways, Scanned Image from Camera, Upload Image from your Phone and you can upload PDF document if you want. The Process is like the input image is passed in the pytesseract and Opencv (Libraries for the implementation of the OCR) and they extract text from image and display it on the output screen.

12.5.4 BENEFITS OF USING THIS ARCHITECTURE:

The OCR has a lot more than we can think of, it is basically a very handy tool to continue with your daily life works and can make your working so smooth enough without wasting time. It has a lot of advantages like:

9.5.4.1 Cost Reduction.

9.5.4.2 Accelerates Workflow.

9.5.4.3 Automate documentation.

9.5.4.4 Easy to Use.

9.5.4.5 Works well with data.

9.5.4.6 Secure your data.

9.5.4.7 Handles accurate information.

9.5.4.8 Less time consuming.

9.5.4.9 Easy Data Handling and storing.

12.6 ALGORITHM ANALYSIS

Algorithm analysis is a major part of the project, it initializes what and how the work is going to happen. There are various ways to implement one thing but very few are being considerable in action to perform such things. As per our project we are using three algorithms which are listed below:

- Histogram Equalization
- CLAHE (contrast Limited Adaptive Histogram Equalization)
- Otsu Thresholding

ALGORITHM	TIME COMPLEXITY (Best,worst,Average)	SPACE COMPLEXITY (Best,worst,Average)
Histogram Equalization	O(N) Whereas, N is no. of pixels in the image	O(K) Whereas, K is no. of intensity levels in the image
CLAHE	O(N) Whereas, N is no. of pixels in the image	O(N) Whereas, N is no. of pixels in the image
Otsu Thresholding	O(N) Whereas, N is no. of pixels in the image	O(K) Whereas, K is no. of intensity levels in the image

Table 9.6.1: Complexities of Algorithms

12.7 ACHIEVEMENT

So as we are working on the image detection and language translation project that deals with the image detection and text extraction from OCR i.e. optical character recognition and the language translation is done by NLP i.e. Natural language processing.

12.7.1 OPTICAL CHARACTER RECOGNITION

We have achieved a blog reading that enhanced the ability to work and implement our project idea:

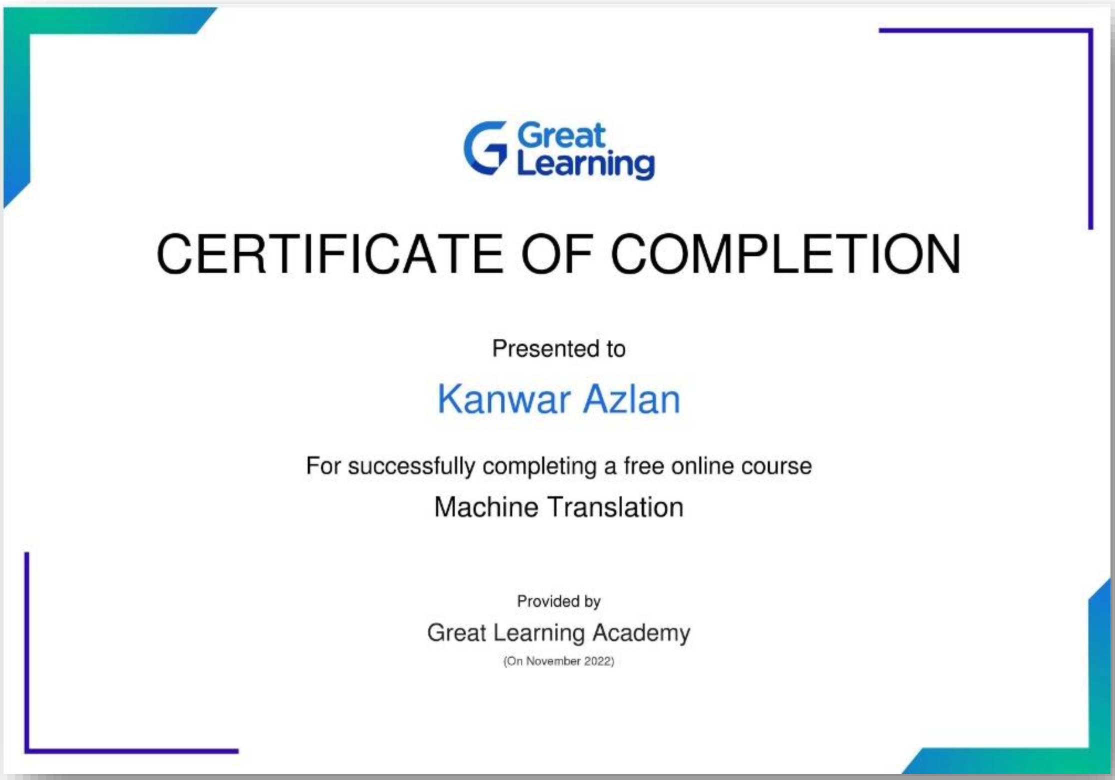
https://www.mygreatlearning.com/blog/introduction-to-image-pre-processing/

furthermore, we have researched about the OCR and came up with this research paper in which a similar project was made.

https://www.researchgate.net/publication/351422903_IMAGE_TEXT_EXTRACTION_AND_ITS_LANGUAGE_TRANSLATION

12.7.2 NATURAL LANGUGAE PROCESSING

As NLP is the core part of our project so we achieved a NLP Course completion certificate.



Great Learning

CERTIFICATE OF COMPLETION

Presented to

Kanwar Azlan

For successfully completing a free online course

Machine Translation

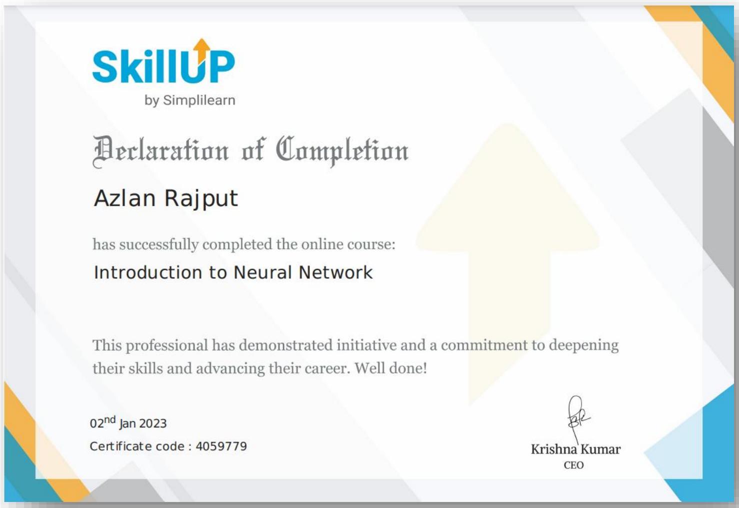
Provided by

Great Learning Academy

(On November 2022)

12.7.3 NEURAL NETWORKS

NLP is a subpart of Machine learning and ML is a subpart of Artificial Intelligence so neural networks can be very helpful for us to implement NLP and to make language translation more successful.



SkillUP
by Simplilearn

Declaration of Completion

Azlan Rajput

has successfully completed the online course:

Introduction to Neural Network

This professional has demonstrated initiative and a commitment to deepening their skills and advancing their career. Well done!

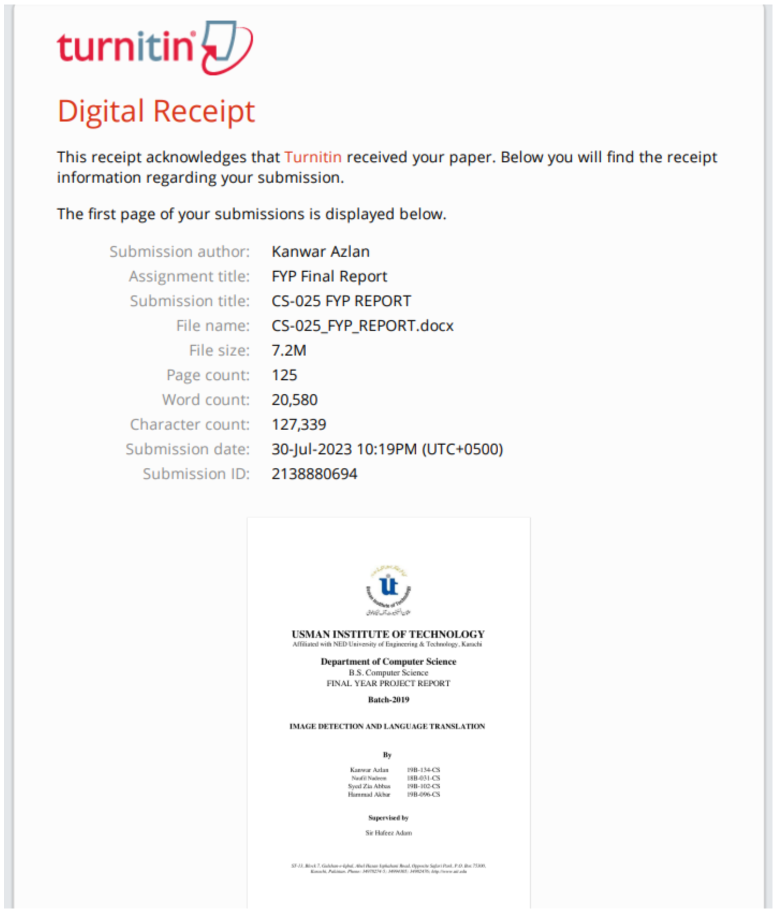
02nd Jan 2023

Certificate code : 4059779

[illegible]

Krishna Kumar
CEO

Copy of **Plagiarism check report**



turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Kanwar Azlan
Assignment title:	FYP Final Report
Submission title:	CS-025 FYP REPORT
File name:	CS-025_FYP_REPORT.docx
File size:	7.2M
Page count:	125
Word count:	20,580
Character count:	127,339
Submission date:	30-Jul-2023 10:19PM (UTC+0500)
Submission ID:	2138880694

USMAN INSTITUTE OF TECHNOLOGY
Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science
B.S. Computer Science
FINAL YEAR PROJECT REPORT

Batch-2019

IMAGE DETECTION AND LANGUAGE TRANSLATION

By

Kanwar Azlan	19B-134-CS
Naufil Nadeem	18B-031-CS
Syed Zia Abbas	19B-102-CS
Hammad Akbar	19B-096-CS

Supervised by

Sir Hafeez Adam

ST-13, Block 7, Gulshan-e-Iqbal, Abul Hasan Isphahani Road, Opposite Safari Park, P.O. Box 75300, Karachi, Pakistan. Phone: 34978274-5; 34994305; 34982476; http://www.uit.edu

12.8 REFERENCES

B.Shi. (2015). An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. *School of Electronic Information and Communications*, 1-9.

Baniata, l. H. (2021). A Transformer-Based Neural Machine Translation Model for Arabic Dialects That Utilizes Subword Units. *School of Computer Science and Engineering, Kyungpook National University*, 1-28.

D.Bahdanau. (2016). NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE. *JOINTLY LEARNING TO ALIGN AND TRANSLATE* (pp. 1-15). germany: ICLR.

h.Sutskever. (2014). Sequence to Sequence Learning with Neural Networks. -, 1-9.

Liu, H. (2021). Re-Transformer: A Self-Attention Based Model for Machine Translation. *5th International Conference on AI in Computational Linguistics*, 1-8.

Misra, C., Swain, P., & Mantri, J. (2012). Text Extraction and Recognition from Image using Neural Network. *International Journal of Computing Applications*, 13-19.

N.Mangrulkar. (2021). IMAGE TEXT EXTRACTION AND ITS LANGUAGE TRANSLATION. *Journal of Research in Engineering and Applied Sciences*, 1-4.

Sefara, T. (2021). Transformer-based Machine Translation for Lowresourced Languages embedded with Language Identification. *Council For Scientific and Industrial Research*, (pp. 1-7). South Africa: Joseph.

Zhang, J. (2017). Improving the Transformer Translation Model. *State Key Laboratory of Intelligent Technology and Systems*, 1-10.

13. WORK PRODUCTS

13.1 DEPOLYMENT ARCHITECTURE WITH STRATEGY

STAGE	TASK	ARCHITECTURE	STRATEGY
1st Stage	To link the back end files with the front end.	Flask (Flask is micro framework of python used to build web and mobile Apps)	• We build the API with Flask. • Then we provide port no and IP Address to which API hits. • When we run the flask's API, it generates a link (HTTP Request). • After this we integrate this link with the action button of our front end.
2nd Stage	Input the data from Database	Images Database, Language Database	• Import image file to the program in real time. • Verify for the relevant image to be taken for further processing, if not valid, capture the image again. • Same as, the languages data will be taken from the database. • The row data will be given to the model for the training and testing. • The model will map the data of both the languages and split the data by train and test.
3rd Stage	Extracting the Text from image	Optical Character Recognition (OCR) A technique used to extract the text from the documents.	• The input (RAW) data of the images were passed with the preprocessing. • After the preprocessing phase, the image is enhanced and ready for further proceedings. • Then the image is passed through the tesseract (Library of Python) for the text extraction.

4th Stage	Translation of the text into desired language	Natural Language Processing (NLP) Technique used to translate one language to another.	• The extracted text transferred to the trained model for the mapping of the translated language. • Then the text will be mapped with the desired language for gaining the translated text with the help of trained and tested data. • The output text in the desired language is shown on the output.

Table 10.1: Deployment Architecture with strategy.

13.2 BUG LIST

TYPE OF BUG	TASK	DESCRIPTION	SOLUTION
Functional Bug	Image doesn’t recognize	When we are trying to take the input image, the clarity of some sort of images doesn’t match the appropriate value.	We applied proper algorithm to avoid such behavior of the images.
User interface (UI) Bug	Visual elements and user interaction	Some of the buttons and alignments are out of order during the building of the application.	We lock the alignment and order of the elements.

compatibility Bug	Performance check	When we try to compare the performance of the application, It’s not compatible with some of the devices.	It requires a basic powered device with all its proper functionalities in running.
Localization Bug	Translation of the language	In the phase of translation, some texts are recognized wrong by the model.	We try to enhance the accuracy of the model.

Table 10.2: Bug List.

14. CONCLUSION

The primary objective of this project is to provide an enhanced and efficient application that caters to the growing needs of people in today's fast-paced world. To achieve this, we have opted to develop an Android application due to the widespread familiarity and extensive user experience with Android apps. This project offers us the opportunity to work with two languages simultaneously, enhancing our skills and exposure.

Text extraction from images is a complex area in digital image processing, requiring sophisticated algorithms to detect and recognize text accurately. While various language translators, such as voice-based or keyboard-based translators, exist, they may not be user-friendly. Our aim is to establish a dynamic connection between text and interactive visualization imagery.

Leveraging the capabilities of Android device cameras, our proposed algorithm, implemented in Java, will facilitate text extraction from images with ease. Given the widespread use of mobile devices, users can effortlessly capture images to extract the embedded text. This project's primary goal is to successfully implement text extraction from images and subsequently translate the extracted text.

The captured text from camera images can serve as valuable indicators in numerous image-based applications, including assistive navigation, auxiliary reading, image retrieval, and scene understanding.

Extracting text from images in natural scenes poses greater challenges than processing scanned documents due to complex backgrounds and variations in text patterns, such as font, color, scale, intensity, and orientation. To address this, the text detection and extraction process becomes crucial, as it computes the specific regions within the images containing text characters or strings. Once the image is captured, it undergoes various processes to detect and extract text, which is then translated.

In today's world, the concept of paperless offices and digitizing documents has become increasingly prevalent across various businesses and industries. This project aims to provide a user-friendly software interface for document scanning, offering benefits like optimized office space utilization, efficient information storage, easy sharing, enhanced data security, and reliable data recovery.

The main purpose to serve the people with an enhanced and useful application that brings them much faster as the world is growing. To capture the idea of dealing with the android application is chosen due to the vast and hands on experience of the users to the android app. As far as the project idea is concerned, it gives us the full exposure to use of two languages in parallel.

The project is basically allowing the user to enter the text in the source language, and it will be translated into the target language. This is basically done by image detection. First the user will scan the image containing some text.

Then it will be converted into the readable text by the help of OCR (Optical Character Recognition). Then it will be translated into the target language by using a technique called NLP (Natural Language Processing).

This basically involves a dual method for the execution of an app through extracting the text from image and then converting the text into second language. This seems an easy task but can involve many complications, however we will manage through our enhanced techniques.

15. REFERENCES

B.Shi. (2015). An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. *School of Electronic Information and Communications*, 1-9.

Baniata, l. H. (2021). A Transformer-Based Neural Machine Translation Model for Arabic Dialects That Utilizes Subword Units. *School of Computer Science and Engineering, Kyungpook National University*, 1-28.

D.Bahdanau. (2016). NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE. *JOINTLY LEARNING TO ALIGN AND TRANSLATE* (pp. 1-15). germany: ICLR.

h.Sutskever. (2014). Sequence to Sequence Learning with Neural Networks. -, 1-9.

Liu, H. (2021). Re-Transformer: A Self-Attention Based Model for Machine Translation. *5th International Conference on AI in Computational Linguistics*, 1-8.

Misra, C., Swain, P., & Mantri, J. (2012). Text Extraction and Recognition from Image using Neural Network. *International Journal of Computing Applications*, 13-19.

N.Mangrulkar. (2021). IMAGE TEXT EXTRACTION AND ITS LANGUAGE TRANSLATION. *Journal of Research in Engineering and Applied Sciences*, 1-4.

Sefara, T. (2021). Transformer-based Machine Translation for Lowresourced Languages embedded with Language Identification. *Council For Scientific and Industrial Research*, (pp. 1-7). South Africa: Joseph.

Zhang, J. (2017). Improving the Transformer Translation Model. *State Key Laboratory of Intelligent Technology and Systems*, 1-10.

Additional References:

https://www.mygreatlearning.com/blog/introduction-to-image-pre-processing/

https://www.researchgate.net/publication/351422903_IMAGE_TEXT_EXTRACTION_AND_ITS_LANGUAGE_TRANSLATION

https://huggingface.co/t5-base?text=My+name+is+Wolfgang+and+I+live+in+Berlin

https://www.youtube.com/watch?v=DkzbCJtFvqM&list=PLZoTAELRMXVOTsz2jZl2Oq3ntWPoKRKwv

https://pytorch.org/hub/huggingface_pytorch-transformers/

https://olympus.mygreatlearning.com/courses/74236/pages/course-outline-and-agenda

https://www.analyticsvidhya.com/blog/2021/06/language-translation-with-transformer-in-python/

https://pytorch.org/tutorials/beginner/translation_transformer.html

https://www.mdpi.com/2076-3417/12/14/7195/pdf

https://www.datacamp.com/tutorial/machine-learning-models-api-python

https://www.practitest.com/qa-learningcenter/resources/black-box-vs-white-box-testing/

https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/

https://www.ibm.com/cloud/blog/optical-character-recognition

https://www.mygreatlearning.com/blog/histogram-equalization-explained/

https://towardsdatascience.com/histogram-equalization-5d1013626e64

https://learnopencv.com/otsu-thresholding-with-opencv/

https://www.geeksforgeeks.org/clahe-histogram-eqalization-opencv/

https://en.wikipedia.org/wiki/Otsu%27s_method#:~:text=Otsu's%20method%20performs%20well%20when,class%20than%20inter%2Dclass%20variance.

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/

https://www.stickyminds.com/article/state-transition-diagrams

https://www.lawinsider.com/dictionary/operational-

https://www.youtube.com/results?search_query=operational+uml+diagram

https://www.youtube.com/results?search_query=component+uml+diagram