

IMAGE TO AUDIO CONVERSION FOR BLIND PEOPLE

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

by

NAVULURI SAI RAM (Reg.No - 40110843)

NAGA VENKATA RAVITEJA KURRA (Reg.No - 40110813)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY -1 UNIVERSITY BY UGC

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

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

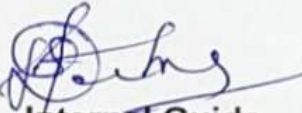
CHENNAI - 600 119

APRIL - 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

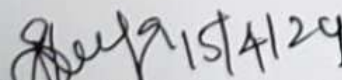
This is to certify that this Project Report is the bonafide work of **Naga Venkata Raviteja Kurra (40110813)** and **Navuluri Sai Ram (40110843)** who carried out the Project entitled "**Image to Audio Conversion for Blind People**" under my supervision from November 2023 to April 2024.


Internal Guide

Dr. D. Sheema M.C.A., Ph. D.


Head of the Department
HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Jeppiaar Nagar, Rajiv Gandhi
Chennai - 600 119.
Dr. L. Lakshmanan M.E., Ph.D.

Submitted for Viva Voce Examination held on 15-04-2024


Internal Examiner

V. SARANYA


External Examiner

(Dr. R. Anil Kumar)

DECLARATION

I, **NAGA VENKATA RAVITEJA KURRA** (Reg.No-40110813) hereby declare that the Project Report entitled "**IMAGE TO AUDIO CONVERSION FOR BLIND PEOPLE**" done by me under the guidance of **Dr.D.SHEEMA M.C.A., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 15-04-2024

PLACE: Chennai

K.N.V. Raviteja

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

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

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

ABSTRACT

The project focuses on the development of an innovative solution for enhancing accessibility and the independence among visually impaired individuals. By leveraging Advanced image recognition and text-to-speech technologies, this research facilitates the conversion of images into audio descriptions. These audio descriptions are designed to convey essential visual information, such as text content, objects, and scenes, to blind users in a comprehensible and user-friendly manner. The abstract underscores the potential impact of this technology in bridging the accessibility gap, enabling visually impaired individuals to gain a richer understanding of their surroundings and access visual content independently, thereby fostering greater inclusivity and autonomy in their daily lives.

The image pre-processing module is the first one in the CNN-LSTM image to audio conversion system. For the CNN-LSTM to be trained efficiently, this module is essential in getting the input picture data ready. A number of methods and algorithms are employed in the pre-processing module to improve the quality of the input photos, extract valuable characteristics, and normalize the pixel values to a set scale.

The CNN-LSTM architecture module is the second component of the image to audio conversion system using CNN-LSTM. In order for the CNN-LSTM to learn the mapping between the input images and associated audio signals, the structure and parameters of the CNN-LSTM must be defined by this module.

A series of feature vectors representing the likelihood of producing each audio sample is the CNN-output. LSTM's this series of feature vectors is examined by the postprocessing module, which then creates the final audio signal.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	vi
	LIST OF ABBREVIATIONS	vii
1.	INTRODUCTION	1
	1.1 INTRODUCTION	1
2.	LITERATURE SURVEY	3
3.	REQUIRMENT ANALYSIS	7
	3.1 REQUIRMENT SPECIFICATION	7
	3.1.1 HARDWARE REQUIRMENTS	7
	3.1.2 SOFTWARE REQUIRMENTS	7
	3.2 LANGUAGE SPECIFICATION-PYTHON	8
4.	EXISTING AND PROPOSED SYSTEM	9
	4.1 EXISTING SYSTEM	9
	4.2 PROPOSED SYSTEM	10
	4.3 FEASIBILTY STUDY	11
	4.3.1 ECONIMICAL FEASIBILTY	12
	4.3.2 TECHNICAL FEASIBILTY	12
	4.3.3 OPERATIONAL FEASIBILTY	12
	4.3.4 SOCIAL FEASIBILTY	12
5.	IMPLEMENTATION AND DETAILS	13
	5.1 SYSTEM ARCHITECTURE	13
	5.2 DATA-FLOW DIAGRAM	13
	5.3 ENTITY-RELATIONSHIP DIAGRAM	15
	5.4 USE-CASE DIAGRAM	16
	5.5 ACTIVITY DIAGRAM	17
	5.6 SEQUENCE DIAGRAM	18
	5.7 CLASS DIAGRAM	19
	5.8 ER DIAGRAM	19
	5.9 IMPLEMENTATION MODULE	20
	5.9.1 IMAGE PRE-PROCESSING	20
	5.9.2 CNN-LSTM ARCHITECTURE	21
	5.9.3 POST PRE-PROCESSING	23
	5.10 TESTING	24
	5.10.1 TYPES OF TESTING	24
	5.10.2 TESTING TECHNIQUES	26
6.	RESULTS	29
7.	CONCLUSION	32
	REFERENCES	33
	APPENDIX	35
	A. CODING	35
	B. SCREEN SHOTS	59
	C. RESEARCH PAPER	69
	D. CONFERENCE PAPER	75

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
5.1	SYSTEM ARCHITECTURE	13
5.2	DATA FLOW DIAGRAM 0	14
5.3	DATA FLOW DIAGRAM 1	14
5.5	USE CASE DIAGRAM	16
5.6	ACTIVITY DIAGRAM	17
5.7	SEQUENCE DIAGRAM	18
5.8	CLASS DIAGRAM	19
5.9	ENTITY RELATIONSHIP DIAGRAM	19
7.1	IMPORTING LIBRARIES	59
7.2	IMAGE PROCESSING	59
7.3	TEXT PRE-PROCESSING	60
7.4	VISUALIZE IMAGES WITH CAPTIONS	61
7.5	CREATE VOCUBALARY	62
7.6	BUILD GENERATOR FUNCTION	63
7.7	MODEL	65
7.8	PREDICTIONS	66
7.9	WEB PAGE	67
7.9.1	UPLOADING DATA	67
7.9.2	RESULT REASEARCH PAPER	67 68-73

LIST OF ABBREVIATIONS

OCR	Optical Character Recognition
SVM	Support Vector Machine
RBF	Radio Basis Function
ERBF	Exponential Radio Basis Function
NLP	Natural Language Processing
CNN	Convolutional Neural Network
LSTM	Long Short Term Memory
DFD	Data Flow Diagram
ER	Entity Relationship
ER	Entity Relationship Diagram

CHAPTER 1

INTRODUCTION

1.1 Introduction

Visual impairment is one of the biggest limitation for humanity, especially in this day and age when information is communicated a lot by text messages (electronic and paper based) rather than voice. The device we have proposed aims to help people with visual impairment. In our planet of 7.4 billion humans, 285 million are visually impaired out of whom 39 million people are completely blind, i.e. have no vision at all, and 246 million have mild or severe visual impairment (WHO, 2011).

It has been predicted that by the year 2020, these numbers will rise to 75 million blind and 200 million people with visual impairment. As reading is of prime importance in the daily routine (text being present everywhere from newspapers, commercial products, sign boards, digital screens etc.) of mankind, visually impaired people face a lot of difficulties.

According to the development in today's technology towards the computer vision, digital camera and portable computers it is feasible to develop a camera-based technology that combines computer vision technology with other commercial products such as OCR systems. Reading is very essential in today's society. Everywhere the printed text is in the form of Reports, bank statements, receipts, restaurant menu's etc. so the blind users face a difficulty in reading these forms. In order to reduce the frustrated problem the method Text to Voice Adaption Using Portable Camera is referred. The method which is already existed a carries major drawback in size and not portable.

To reduce this drawback, we choose an embedded platform raspberry pi (Model 3) which acts as a mini Cohere the camera is interfaced to the raspberry pi board and the captured images is processed to the Rpi board. ROI method is used to localize and recognize the text. The text codes from the ROI are recognized by the Optical Character recognition (OCR) and the captured image is processed using the python programming language.

The text from the OCR is compared with the text in the Open CV library to identifying the orientations and edge pixels. Therefore, the captured images are converted in to the text. The text codes are processed to the pixy library and it is output to blind users in speech. And in addition, we add a ultrasonic sensor to alert the blind users by the speaker in avoiding the obstacles.

To extract the hand-held object from the camera image, this system going to develop a motion-based method to obtain a region of interest (ROI) of the object. Then, perform text recognition only that ROI. [2] Detecting text in natural scenes with stroke width transform, to post process the image, support vector machines (SVM) had been proposed to do classification on the extracted features. Some kernel functions which are being tested are second degree polynomial, radial basis function (RBF), exponential radial basis function (ERBF), sigmoid, and odd-order Bsp line. RBF and ERBF.

To solve the common aiming problem we have implemented motion-based method to detect the objects of interest. Text extractions are done using stroke orientation and distribution of edge pixels. The text characters are recognized using Optical Character Recognition, the text codes are transformed as speech for blind persons. Our future work will extend the text localization algorithm with further more features and we will address the human interface issues associated with text reading by the blind user.

CHAPTER 2

LITERATURE SURVEY

Sneha.C. Madre, S.B. Gundre, “OCR Based Image Text to Speech Conversion Using MATLAB”, Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2019

The proposed system is inexpensive and allows visually impaired people to hear the text. This project's main concept is optical character recognition, which is used to convert text characters into audio signals. The text is preprocessed before being used for character recognition by segmenting each character. Following segmentation, the letter is extracted and the file containing the text is resized. The audio signal is then generated from the text file. All of the aforementioned processes will be carried out using MATLAB16.

P Rohit, M S Vinay Prasad, S J Ranganatha Gowda, D R Krishna Raju, Imran Quadri, “Image Recognition Based Smart Aid For Visually Challenged People”, International Conference on Communication and Electronics Systems (ICCES), 2020

Our paper describes the creation of a real-time system based on object detection, classification, and position estimation in an outdoor environment to provide visually impaired people with voice output-based scene perception. The system is inexpensive, lightweight, simple, and easy to wear. The module is built into the stick, and the pi-camera is used to take the photo, with a controller to move the camera in the desired direction. The useful insights obtained from the feedback are then used to modify the system to better suit the user's needs. For efficient feature representation, the object detection and classification framework employ a multi-modal fusion-based mask RCNN with motion, sharpening, and blurring filters. The image recognition classifies the detected objects as well as their positions.

Sujata Deshmukh, Praditi Rede, Sheetal Sharma, Sahaana Iyer, “Voice-Enabled Vision For The Visually Disabled”, International Conference on Advances in Computing, Communication, and Control (ICAC3), 2022

We proposed a unified system for extracting text from images and converting it into

an audio track in the target language. This method can help the visually impaired sense the attitude and demeanor of the person they are dealing with. All of the aforementioned tasks associated with this application are carried out by issuing commands. This system is unique in that it reads handwritten papers, analyses personality based on them, and provides audio output to the blind. This technology enables visually impaired people to read and comprehend First Information Reports (FIRs), business cards, chats, doctor's prescriptions, invoices, addresses, and other documents.

Sai Aishwarya Edupuganti, Vijaya Durga Koganti, Cheekati Sri Lakshmi, Ravuri Naveen Kumar, Ramya Paruchuri, "Text and Speech Recognition for Visually Impaired People using Google Vision", 2nd International Conference on Smart Electronics and Communication (ICOSEC), 2021

This study assists visually impaired and elderly people in detecting text in order to identify the medicine. The researchers propose to create an application that will assist visually impaired people in scanning images and converting the detected text into voice messages. Google vision library is used to create an application for the Android platform that primarily contains three important functionalities: text recognition, text detection, and text-to-speech conversion. The medicine image is scanned using an in-built camera.

Abhishek Mathur, Akshada Pathare, Perna Sharma, Sujata Oak, "AI based Reading System for Blind using OCR", 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), 2019

OCR is a mechanism that converts typed, handwritten, or printed text images into machine encoded text. This system will assist you in taking a picture or scanning a document that is present with the user using the phone's camera. The image will be scanned, and the application will read the text written in English and convert the output to speech format. The Text to Speech Module is used to generate the speech output. The goal of delivering the output in the form of voice/speech is to serve the visually impaired with the information on the document.

Vaibhav V. Mainkar, Tejashree U. Bagayatkar, Siddhesh K. Shetye, Hrushikesh R. Tamhankar, Rahul G. Jadhav, Rahul S. Tendolkar, "Raspberry

pi based Intelligent Reader for Visually Impaired Persons”, 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2020

In this project, the researchers use the Raspberry Pi Camera to take pictures, which are then converted into scan images for further processing using the Imagemagick software. The Imagemagick software produces a scanned image, which is then fed into the Tesseract OCR (Optical Character Recognition) software, which converts the image to text. TTS (Text to Speech) engine was used to convert text to speech. The experimental results show that analyzing different captured images will be more beneficial to blind people.

Javavrinda Vrindavanam, Raghunandan Srinath, Anisa Fathima, S. Arpitha, Chaitanya S Rao, T. Kavya, “Machine Learning based approach to Image Description for the Visually Impaired”, Asian Conference on Innovation in Technology (ASIANCON), 2021

The need for the paper stems from the fact that the interaction points for visually impaired people are becoming increasingly limited in an increasingly digitized world, and accessing digital media through an image describer can be an accomplice for the visually impaired. Images that are unseen by the visually impaired are processed, appropriate descriptions are generated, and the audio output is converted. In contrast to traditional methods such as Computer Vision and Convolutional Neural Networks (CNN), the paper employs the Inception Resnet - V2 model as the feature extractor and decoder (GRU-RNN) along with the Bahdanau attention model to generate a text description of the image, which is then converted to audio using Google Text-to-Speech converter.

R. Prabha, M. Razmah, G. Saritha, RM Asha, Senthil G. A, R. Gayathiri, “Vivoice - Reading Assistant for the Blind using OCR and TTS”, International Conference on Computer Communication and Informatics (ICCCI), 2022

In this paper, the researchers intended to assist blind people in recognizing various texts and identifying various objects in their surroundings. The images are processed with OpenCV, which is written in Python and uses the Tesseract OCR library. The extracted texts are voiced using a Text-to-Speech synthesizer. The

software used for text-to-speech conversion is eSpeak. The final output is delivered to visually impaired people via earphones. Another application is the use of Natural Language Processing Algorithms to search for the required product by providing it as an input to the device. The device looks for the product and alerts the visually impaired person to it. This device allows visually impaired people to reduce their reliance on other people and their other senses while still meeting their daily needs.

S. Durgadevi, K. Thirupurasundari, C. Komathi, S.Mithun Balaji, “Smart Machine Learning System for Blind Assistance”, International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), 2021

The necessary data input is obtained using an image classification technique in order to access machine learning techniques. Using a camera, the objects in the vicinity of blind people are captured as images. It can precisely detect every object within a certain distance. The captured images are then converted into audio signals, which are then used to assist the blind. As a result, a user-friendly flexible guiding mechanism is created to assist blind people.

Sandeep Musale, Vikram Ghiye, “Smart reader for visually impaired”, 2nd International Conference on Inventive Systems and Control (ICISC), 2018

Typically, many people had visual impairments. Written transcripts are visible forms of information that are inaccessible to many blind and visually impaired people unless they are represented in a non-visual format such as Braille. A smart reader is required for an effective system for visually impaired people. MATLAB's OCR (Optical Character Recognition) functions convert images to text. The smart reader system for the visually impaired is proposed in this paper. A novel audio-tactile user interface that assists the user in reading information is proposed here.

CHAPTER 3

REQUIRMENT ANALYSIS

3.1 REQUIREMENT SPECIFICATION

3.1.1 HARDWARE REQUIREMENTS

Processor	: Pentium Dual Core 2.00GHZ
Hard disk	: 120 GB
RAM	: 2GB (minimum)
Keyboard	: 110 keys enhanced

3.1.2 SOFTWARE REQUIREMENTS

Operating system	: Windows7 (with service pack 1), 8, 8.1 and 10
Language	: Python

3.2 LANGUAGE SPECIFICATION– PYTHON

Among programmers, Python is a favorite because to its user-friendliness, rich feature set, and versatile applicability. Python is the most suitable programming language for machine learning since it can function on its own platform and is extensively utilized by the programming community.

Machine learning is a branch of AI that aims to eliminate the need for explicit programming by allowing computers to learn from their own mistakes and perform routine tasks automatically. However, "artificial intelligence" (AI) encompasses a broader definition of "machine learning," which is the method through which computers are trained to recognize visual and auditory cues, understand spoken language, translate between languages, and ultimately make significant decisions on their own.

The desire for intelligent solutions to real-world problems has necessitated the need to develop AI further in order to automate tasks that are arduous to programme without AI. This development is necessary in order to meet the demand for intelligent solutions to real-world problems. Python is a widely used programming language that is often

considered to have the best algorithm for helping to automate such processes. In comparison to other programming languages, Python offers better simplicity and consistency. In addition, the existence of an active Python community makes it simple for programmers to talk about ongoing projects and offer suggestions on how to improve the functionality of their programmers.

ADVANTAGES OF USING PYTHON

Following are the advantages of using Python:

- **Variety of Framework and libraries:**

A good programming environment requires libraries and frameworks. Python frameworks and libraries simplify programme development. Developers can speed up complex project coding with prewritten code from a library. PyBrain, a modular machine learning toolkit in Python, provides easy-to-use algorithms. Python frameworks and libraries provide a structured and tested environment for the best coding solutions.

- **Reliability**

Most software developers seek simplicity and consistency in Python. Python code is concise and readable, simplifying presentation. Compared to other programming languages, developers can write code quickly. Developers can get community feedback to improve their product or app. Python is simpler than other programming languages, therefore beginners may learn it quickly. Experienced developers may focus on innovation and solving real-world problems with machine learning because they can easily design stable and trustworthy solutions.

- **Easily Executable**

Developers choose Python because it works on many platforms without change. Python runs unmodified on Windows, Linux, and macOS. Python is supported on all these platforms, therefore you don't need a Python expert to comprehend it. Python's great executability allows separate applications. Programming the app requires only Python. Developers benefit from this because some programming languages require others to complete the job. Python's portability cuts project execution time and effort.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 EXISTING SYSTEM

One such system is the Image Captioning system, which uses a combination of computer vision and NLP techniques to generate a textual description of an image. This textual description can then be converted into an audio format using text-to-speech synthesis techniques.

The Image Captioning system works by first analyzing the content of an image using computer vision algorithms to identify objects, scenes, and other visual features. This information is then combined with contextual knowledge and semantic rules to generate a textual description of the image. Natural Language Processing (NLP) techniques are used to analyze the textual description and to convert it into an audio format that can be easily understood by humans.

Another system is the Deep-Speaker Embeddings system, which uses deep learning techniques to convert an image into an audio representation. This system extracts features from the image using a deep convolutional neural network (CNN), and then uses an NLP algorithm to generate a corresponding audio signal. The resulting audio signal can be used to convey information about the content of the image, such as its color, texture, and shape.

Overall, there are many different approaches to converting images to audio using NLP algorithms, and the specific approach used will depend on the application and the specific requirements of the system.

There are several disadvantages to existing systems for image to audio conversion using NLP algorithms. Some of these disadvantages include:

Accuracy: Existing systems for image to audio conversion may not always produce accurate results. This is because the accuracy of the system depends on the quality of the image analysis and the NLP algorithms used. In some cases, the system may misinterpret the image, resulting in inaccurate audio descriptions.

Limited vocabulary: Existing systems for image to audio conversion may have a limited vocabulary, which can limit the types of descriptions that can be generated. This can make it difficult to accurately describe complex or abstract images.

Difficulty with abstract concepts: NLP algorithms may have difficulty with abstract concepts, such as emotions or ideas. This can make it challenging to accurately describe images that convey these types of concepts.

Time-consuming: Image to audio conversion using NLP algorithms can be time-consuming, particularly for large or complex images. This can make it challenging to use the system in real-time applications.

Dependence on high-quality images: The accuracy of the image analysis and NLP algorithms used in existing systems is dependent on the quality of the input image. Low-quality images may produce inaccurate results, which can limit the usefulness of the system.

Overall, while existing systems for image to audio conversion using NLP algorithms are useful, they still have several limitations that need to be addressed. Future research will be needed to improve the accuracy, speed, and flexibility of these systems.

4.2 PROPOSED SYSTEM

A proposed system for image to audio conversion using CNN-LSTM algorithm would involve the following steps:

Preprocessing: The system would begin by preprocessing the input image to extract relevant features. This would involve using a convolutional neural network (CNN) to extract visual features from the image.

Encoding: The visual features would then be encoded into a sequence of vectors using an LSTM (Long Short-Term Memory) network. The LSTM network would be trained to learn the relationships between the visual features and the corresponding audio descriptions.

Decoding: The encoded sequence of vectors would be passed through a decoder network, which would generate the corresponding audio signal. This would involve using a text-to-speech synthesis technique to convert the encoded sequence of vectors into an audio waveform.

Postprocessing: Finally, the resulting audio signal would be post-processed to improve its quality and clarity. This might involve techniques such as noise reduction, equalization, and amplification.

This proposed system has several advantages over existing systems that use NLP algorithms for image to audio conversion. First, the use of a CNN-LSTM algorithm allows the system to capture more complex relationships between the visual features and the corresponding audio descriptions, which can improve the accuracy and quality of the generated audio. Second, the use of a CNN-LSTM algorithm can also improve the speed and efficiency of the system, allowing it to be used in real-time applications. Finally, the proposed system is not limited by vocabulary or abstract concepts, as the audio descriptions are generated directly from the visual features of the input image.

4.3 FEASIBILITY STUDY

With an eye towards gauging the project's viability and improving server performance, a business proposal defining the project's primary goals and offering some preliminary cost estimates is offered here. Your proposed system's viability may be assessed once a comprehensive study has been performed. It is essential to have a thorough understanding of the core requirements of the system at hand before beginning the feasibility study. The feasibility research includes mostly three lines of thought:

- Economical feasibility
- Technical feasibility
- Operational feasibility
- Social feasibility

4.3.1 ECONOMICAL FEASIBILITY

The study's findings might help upper management estimate the potential cost savings from using this technology. The corporation can only devote so much resources to developing and analysing the system before running out of money. Every dollar spent must have a valid reason. As the bulk of the used technologies are open-source and free, the cost of the updated infrastructure came in far cheaper than anticipated. It was really crucial to only buy customizable products.

4.3.2 TECHNICAL FEASIBILITY

This research aims to establish the system's technical feasibility to ensure its smooth development. Adding additional systems shouldn't put too much pressure on the IT staff. Hence, the buyer will experience unnecessary anxiety. Due to the low likelihood of any adjustments being necessary during installation, it is critical that the system be as simple as possible in its design.

4.3.3 OPERATIONAL FEASIBILITY

An important aspect of our research is hearing from people who have actually used this technology. The procedure includes instructing the user on how to make optimal use of the resource at hand. The user shouldn't feel threatened by the system, but should instead see it as a necessary evil. Training and orienting new users has a direct impact on how quickly they adopt a system. Users need to have greater faith in the system before they can submit constructive feedback.

4.3.4 SOCIAL FEASIBILITY

During the social feasibility analysis, we look at how the project could change the community. This is done to gauge the level of public interest in the endeavour. Because of established cultural norms and institutional frameworks, it's likely that a certain kind of worker will be in low supply or nonexistent.

CHAPTER 5

IMPLEMENTATION AND DETAILS

5.1 SYSTEM ARCHITECTURE

This graphic provides a concise and understandable description of all the entities currently integrated into the system. The diagram shows how the many actions and choices are linked together. You might say that the whole process and how it was carried out is a picture. The figure below shows the functional connections between various entities.

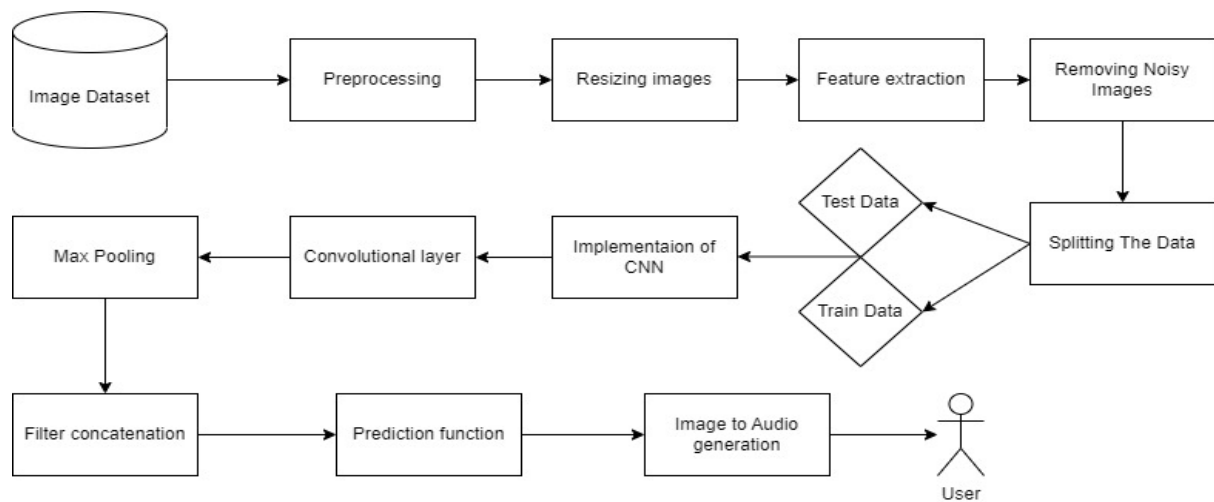


Fig 5.1 – Architecture Diagram

5.2 DATA FLOW DIAGRAM

To illustrate the movement of information throughout a procedure or system, one might use a Data-Flow Diagram (DFD). A data-flow diagram does not include any decision rules or loops, as the flow of information is entirely one-way. A flowchart can be used to illustrate the steps used to accomplish a certain data-driven task. Several different notations exist for representing data-flow graphs. Each data flow must have a process that acts as either the source or the target of the information exchange. Rather than utilizing a data-flow diagram, users of UML often substitute an activity diagram. In the realm of data-flow plans, site-oriented data-flow plans are a subset. Identical nodes in a data-flow diagram and a Petri net can be thought of as inverted counterparts since

the semantics of data memory are represented by the locations in the network. Structured data modeling (DFM) includes processes, flows, storage, and terminators.

DATA FLOW DIAGRAM

The whole system is shown as a single process in a level DFD. Each step in the system's assembly process, including all intermediate steps, are recorded here. The "basic system model" consists of this and 2-level data flow diagrams.

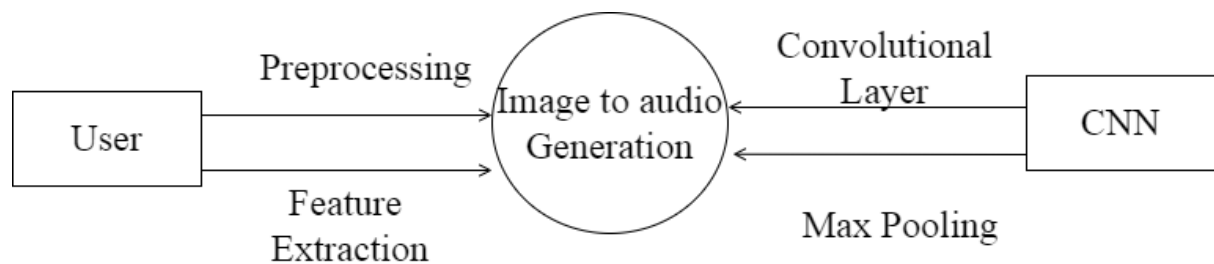


Fig 5.2 – Data Flow Diagram Level 0

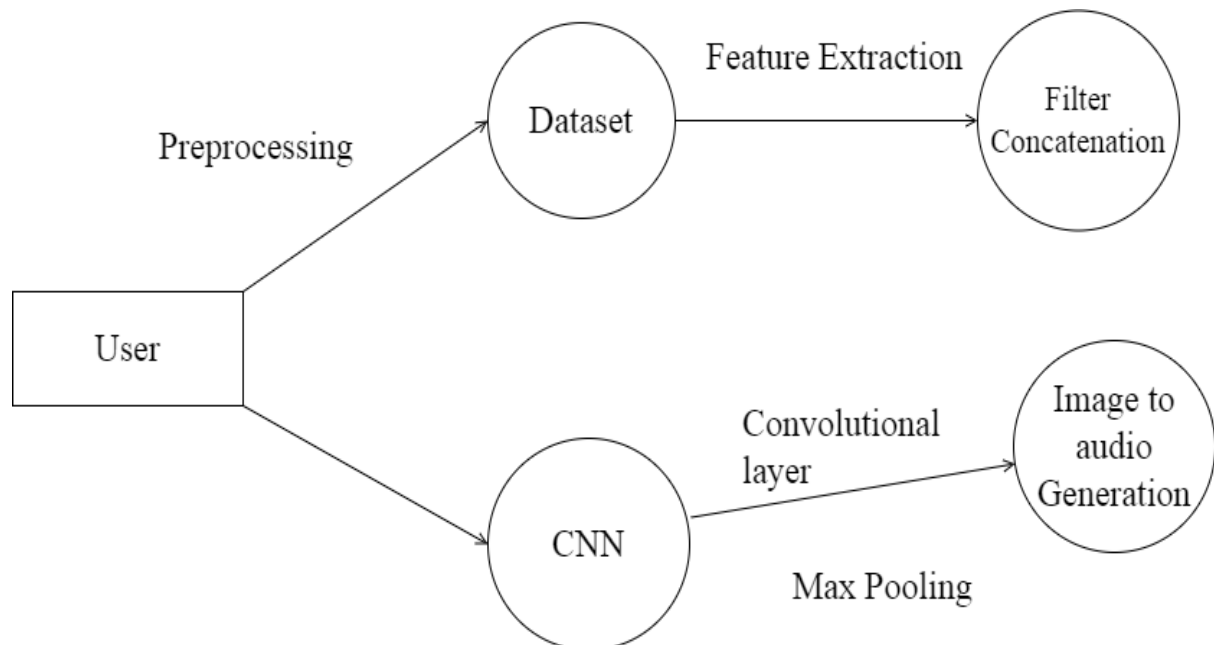


Fig 5.3 – Data Flow Diagram Level 1

5.3 ENTITY RELATIONSHIP DIAGRAM

➤ Definition

The relationships between database entities can be seen using an entity-relationship diagram (ERD). The entities and relationships depicted in an ERD can have further detail added to them via data object descriptions. In software engineering, conceptual and abstract data descriptions are represented via entity-relationship models (ERMs). Entity-relationship diagrams (ERDs), entity-relationship diagrams (ER), or simply entity diagrams are the terms used to describe the resulting visual representations of data structures that contain relationships between entities. As such, a data flow diagram can serve dual purposes. To demonstrate how data is transformed across the system. To provide an example of the procedures that affect the data flow.

1. One-to-One

Whenever there is an instance of entity (A), there is also an instance of entity (B) (B). In a sign-in database, for instance, only one security mobile number (S) is associated with each given customer name (A) (B).

2. One-to-Many

For each instance of entity B, there is exactly one occurrence of entry A, regardless of how many instances of entity B there are.

For a corporation whose employees all work in the same building, for instance, the name of the building (A) has numerous individual associations with employees (B), but each of these B's has only one individual link with entity A.

3. Many-to-Many

For each instance of entity B, there is exactly one occurrence of entry A, regardless of how many instances of entity B there are.

In a corporation where everyone works out of the same building, entity A is associated with many different Bs, but each B has only one A.

5.4 USE-CASE DIAGRAM

The possible interactions between the user, the dataset, and the algorithm are often depicted in a use case diagram. It's created at the start of the procedure.

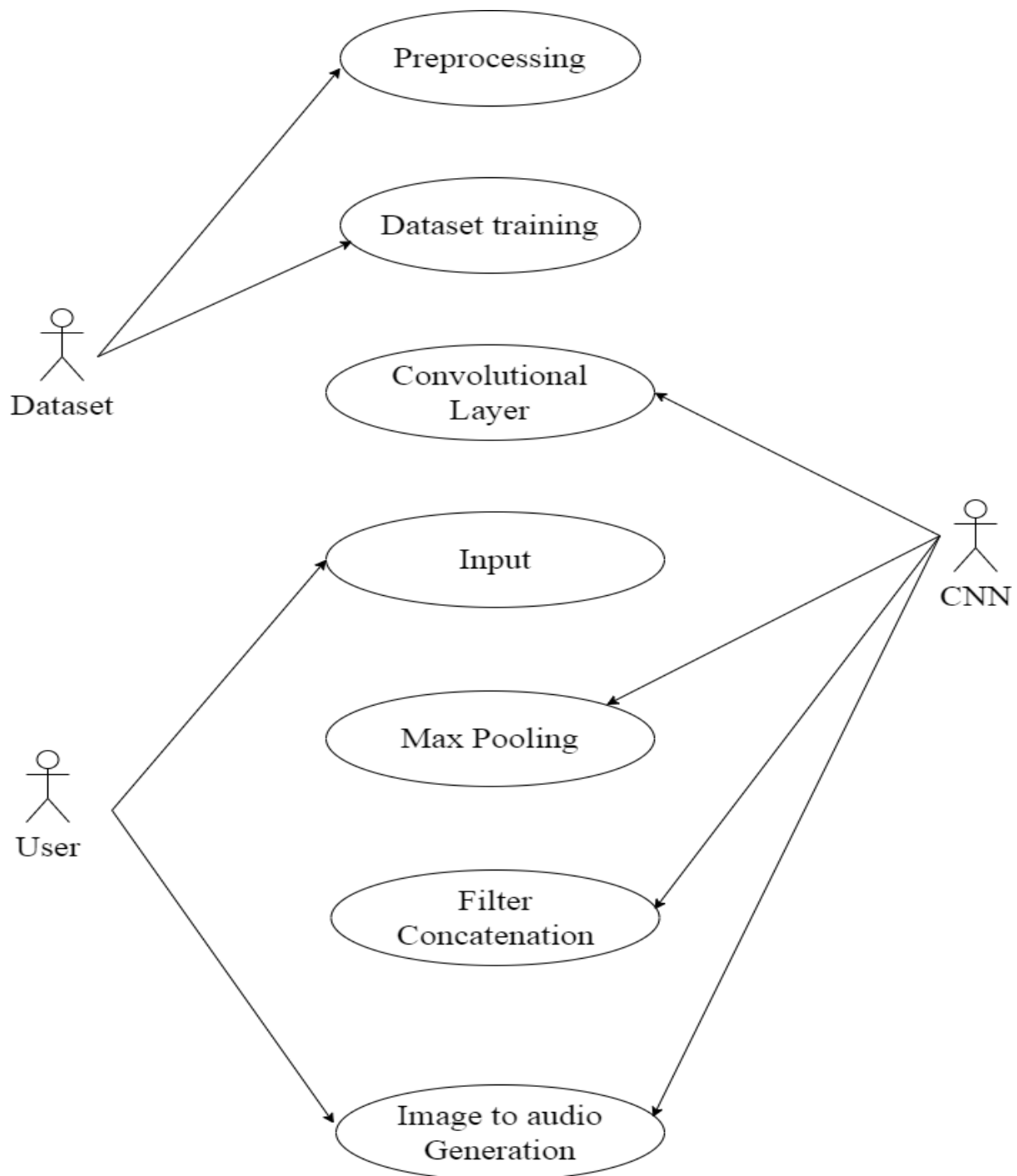


Fig 5.5 – Use-Case Diagram

5.5 ACTIVITY DIAGRAM

An activity diagram, in its most basic form, is a visual representation of the sequence in which tasks are performed. It depicts the sequence of operations that make up the overall procedure. They are not quite flowcharts, but they serve a comparable purpose.

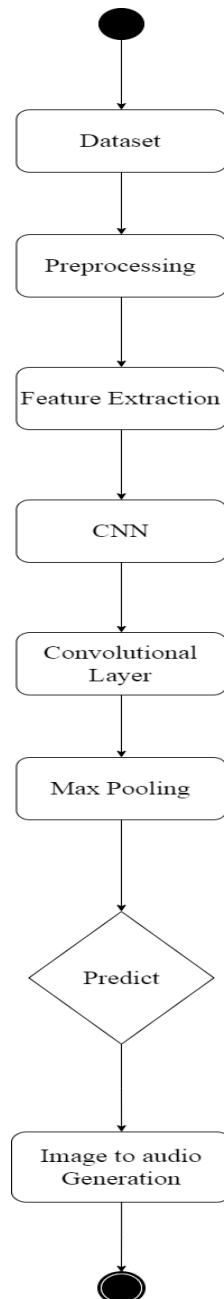


Fig 5.6 – Activity Diagram

5.6 SEQUENCE DIAGRAM

These are another type of interaction-based diagram used to display the workings of the system. They record the conditions under which objects and processes cooperate.

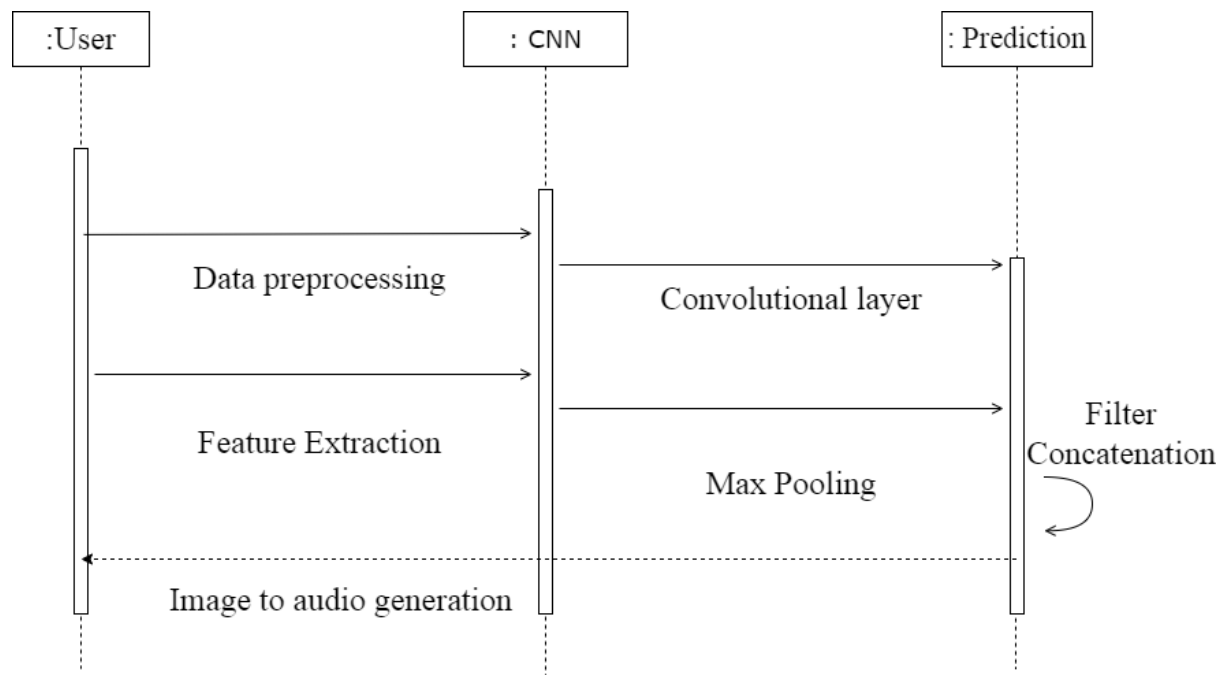


Fig 5.7 – Sequence Diagram

5.7 CLASS DIAGRAM

In essence, this is a "context diagram," another name for a contextual diagram. It simply stands for the very highest point, the 0 Level, of the procedure. As a whole, the system is shown as a single process, and the connection to externalities is shown in an abstract manner.

- A + indicates a publicly accessible characteristic or action.
- A - a privately accessible one.
- A # a protected one.
- A - denotes private attributes or operations.

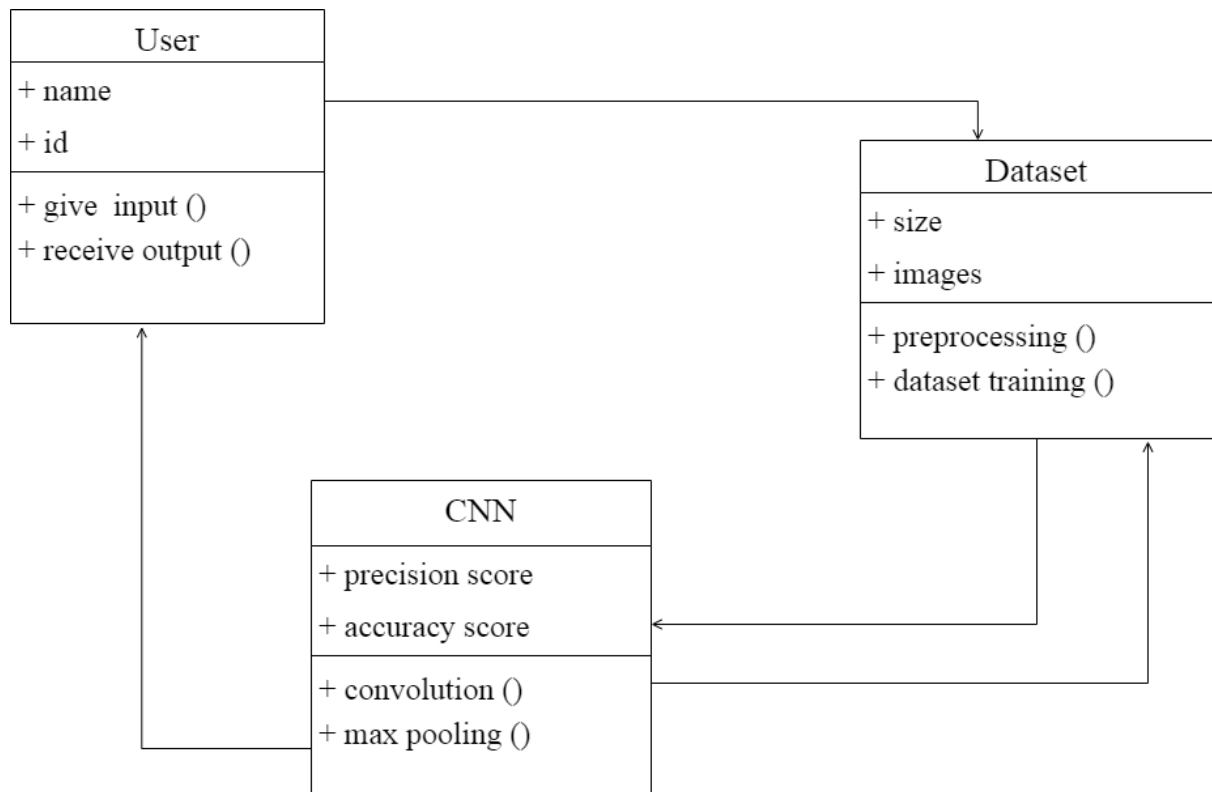


Fig 5.8 – Class Diagram

5.8 ER DIAGRAM

The abbreviation ER refers to a connection between two entities. The entities used and saved in the database are shown in relationship diagrams. They break down the process into its component parts and explain how they work. Attributed concepts, Relationship concepts, and Entity concepts are the building blocks for these kinds of diagrams.

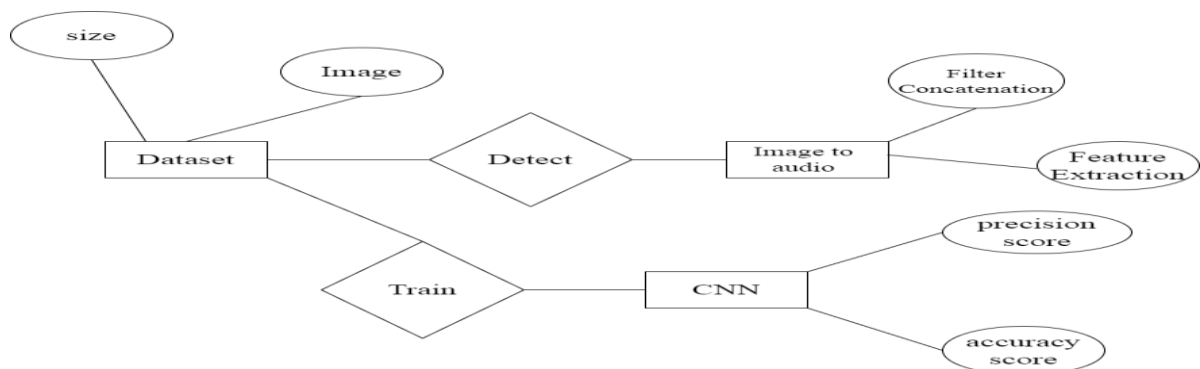


Fig 5.9- ER Diagram

5.9 IMPLEMENTATION MODULE

5.9.1: IMAGE PRE-PROCESSING

The image pre-processing module is the first one in the CNN-LSTM image to audio conversion system. For the CNN-LSTM to be trained efficiently, this module is essential in getting the input picture data ready. A number of methods and algorithms are employed in the pre-processing module to improve the quality of the input photos, extract valuable characteristics, and normalize the pixel values to a set scale.

The pre-processing module's first responsibility is to resize the photos to a particular resolution. In order to properly train the CNN-LSTM, this step makes sure that all of the images are the same size. Standardizing the picture size will assist the CNN-LSTM in recognizing patterns and features from the images. Images of varied sizes may present difficulties in the network training.

The pre-processing module's function of contrast enhancement is also crucial. Contrast enhancement techniques are used to make visual characteristics more visible and assist the CNN-LSTM in locating important features that can be used to generate audio. Histogram equalization, adaptive histogram equalization, and gamma correction are a few well-liked methods for enhancing contrast. These methods work by enhancing the contrast and altering the intensity values of the pixels in the image. The images may also be subjected to noise reduction techniques to get rid of any extraneous or distracting data that could harm CNN-LSTM training. Gaussian blur, the median filter, and the bilateral filter are common approaches. These filters function by minimizing the amount of noise in the image and smoothing it out.

Another crucial duty in the pre-processing module is feature extraction. In order to convert the image to audio, this method entails locating and extracting significant elements. Depending on the task at hand and the nature of the image data, several features may be extracted. Edge detection, texture analysis, and color analysis are a few methods of feature extraction that are frequently utilized. In order to generate reliable audio, the CNN-LSTM needs to be able to recognize key patterns and characteristics in the visual data.

Another essential stage in the pre-processing module is normalization. The process of normalization involves converting the image's pixel values to a uniform scale. To ensure that the CNN-LSTM can be trained successfully, this step is required. Standardization, which adjusts the pixel values to have a mean of zero and a standard deviation of one, or min-max scaling, which scales the pixel values to a range between 0 and 1, are two examples of normalization procedures.

In conclusion, the CNN-LSTM image to audio conversion system relies heavily on the image pre-processing module. The pre-processing program improves the input photos' quality, extracts pertinent information, and scales the pixel values to a uniform scale. For the CNN-LSTM to learn and recognize significant patterns and characteristics in the image data that may be used for precise audio production, this module is crucial. The pre-processing module aids in the CNN-performance LSTM's optimization by giving the CNN-LSTM clean and pertinent data to train on.

5.9.2: CNN-LSTM ARCHITECTURE

The CNN-LSTM architecture module is the second component of the image to audio conversion system using CNN-LSTM. In order for the CNN-LSTM to learn the mapping between the input images and associated audio signals, the structure and parameters of the CNN-LSTM must be defined by this module.

Determining the number of LSTM layers, the number of pooling layers, the size of the filters, the number of convolutional layers, and the number of filters per layer are all tasks included in the CNN-LSTM architecture module. Applying a series of filters to the input image allows convolutional layers to learn features from the previously processed image data. The filters are made to recognize particular motifs or characteristics in the visual data, such as edges, forms, or textures. The size of the filters and the number of filters per layer are hyper parameters that can be adjusted to enhance the CNN-performance.

In order to minimize the dimensionality of the data and avoid over fitting, pooling layers are employed to down sample the output of the convolutional layers. The maximum value within a local region of the input data is chosen by the max pooling layer, which is the most popular kind of pooling layer.

The output of the convolutional layers is processed by LSTM layers, while the audio signal is produced by LSTM layers. The LSTM layer architecture can describe the temporal dependencies between the input image and audio signals and is built to handle sequential data. The CNN-performance LSTM's can be improved by adjusting the number of LSTM layers and the number of neurons in each layer, which are both hyper parameters.

Tasks like regularization, dropout, and activation functions might potentially be included in the CNN-LSTM architecture module. By including a penalty term in the loss function, regularization approaches stop over fitting. In order to avoid over fitting, the CNN-LSTM is trained with a technique called dropout that randomly removes some of the neurons. In order to describe complicated interactions between the input and output data, activation functions are employed to bring non-linearity into the CNN-LSTM.

Using a set of tagged images and the matching audio labels, the CNN-LSTM is trained after the CNN-LSTM architecture has been defined. By modifying its biases and weights using an optimization approach like gradient descent, the CNN-LSTM learns to map the input image data to the audio labels.

In conclusion, a crucial part of the CNN-LSTM image to audio conversion system is the CNN-LSTM architectural module. The CNN-structure LSTM's and parameters are described in this module, along with how the CNN-LSTM learns to produce audio signals from the pre-processed image data. The CNN-LSTM architecture module enables the CNN-LSTM network to train effectively and understand the intricate mapping between the input visuals and related audio signals

5.9.3: POST PRE-PROCESSING

A series of feature vectors representing the likelihood of producing each audio sample is the CNN-output. LSTM's this series of feature vectors is examined by the post-processing module, which then creates the final audio signal.

The post-processing module may carry out operations like denoising, spectrogram creation, and inverse Fourier transform in order to do this. Denoising aids in removing any noise that might be present in the feature vector sequence. In order to effectively manipulate and process the audio input, the sequence of feature vectors is represented in the frequency domain using spectrogram creation. The final audio signal is created by converting a series of feature vectors from the frequency domain to the time domain using the inverse Fourier transform.

The post-processing module may also perform tasks including spectrum smoothing, dynamic range compression, and waveform normalization. To make sure that the amplitude of the audio signal is within a particular range, waveform normalization is performed. By modifying the audio signal's dynamic range, dynamic range compression can improve its clarity and lower distortion. To lessen any abrupt transitions or abnormalities that might be present in the audio signal, spectral smoothing is applied.

In general, the post-processing module is a crucial component of the CNN-LSTM image to audio conversion system. It examines the CNN-output LSTM's and changes it into a useful illustration of the audio signal. Several methods to improve the final audio signal's quality, such as denoising, spectrogram creation, and spectrum smoothing, may be included in the post-processing module. The post-processing module makes sure that the audio signal produced appropriately reflects the supplied visual data and is of excellent quality.

5.10: TESTING

Discovering and fixing such problems is what testing is all about. The purpose of testing is to find and correct any problems with the final product. It's a method for evaluating the quality of the operation of anything from a whole product to a single component. The goal of stress testing software is to verify that it retains its original functionality under extreme circumstances. There are several different tests from which to pick. Many tests are available since there is such a vast range of assessment options.

Who Performs the Testing: All individuals who play an integral role in the software development process are responsible for performing the testing. Testing the software is the responsibility of a wide variety of specialists, including the End Users, Project Manager, Software Tester, and Software Developer.

When it is recommended that testing begin: Testing the software is the initial step in the process. begins with the phase of requirement collecting, also known as the Planning phase, and ends with the stage known as the Deployment phase. In the waterfall model, the phase of testing is where testing is explicitly arranged and carried out. Testing in the incremental model is carried out at the conclusion of each increment or iteration, and the entire application is examined in the final test.

When it is appropriate to halt testing: Testing the programme is an ongoing activity that will never end. Without first putting the software through its paces, it is impossible for anyone to guarantee that it is completely devoid of errors. Because the domain to which the input belongs is so expansive, we are unable to check every single input.

5.10.1 TYPES OF TESTING

There are four types of testing:

Unit Testing

The term "unit testing" refers to a specific kind of software testing in which discrete elements of a program are investigated. The purpose of this testing is to ensure that the software operates as expected.

Test Cases

1. Test if the software is able to correctly recognize individual sign language gestures in a given image.
2. Verify if the software is able to correctly classify each recognized gesture into appropriate categories.
3. Test if the software is able to correctly map each recognized gesture to appropriate audio output.
4. Verify if the software is using appropriate algorithms and techniques to perform image recognition and audio conversion tasks.

Integration Testing

The program is put through its paces in its final form, once all its parts have been combined, during the integration testing phase. At this phase, we look for places where interactions between components might cause problems.

Test Cases

1. Test if the software is properly integrating with any image recognition software or libraries used for recognizing sign language gestures.
2. Verify if the software is properly integrating with any audio conversion software or libraries used for converting recognized gestures into audio output.
3. Test if the software is properly passing data between the different components of the application, such as image recognition and audio conversion.
4. Verify if the software is able to handle any conflicts or errors that may arise during the integration process.

Functional Testing

One kind of software testing is called functional testing, and it involves comparing the system to the functional requirements and specifications. In order to test functions, their input must first be provided, and then the output must be examined. Functional testing verifies that an application successfully satisfies all of its requirements in the correct manner. This particular kind of testing is not concerned with the manner in which processing takes place; rather, it focuses on the outcomes of processing.

Therefore, it endeavours to carry out the test cases, compare the outcomes, and validate the correctness of the results.

Test Cases

1. Verify if the software is able to detect the image of sign language and convert it into audio output.
2. Test if the software is able to recognize and process the images with various hand positions and facial expressions of the person signing.
3. Verify if the audio output is clear and understandable, without any distortion or noise.
4. Test if the software is able to differentiate between different signs and provide appropriate audio output.

5.10.2 TESTING TECHNIQUES

There are many different techniques or methods for testing the software, including the following:

BLACK BOX TESTING

During this kind of testing, the user does not have access to or knowledge of the internal structure or specifics of the data item being tested. In this method, test cases are generated or designed only based on the input and output values, and prior knowledge of either the design or the code is not necessary. The testers are just conscious of knowing about what is thought to be able to do, but they do not know how it is able to do it.

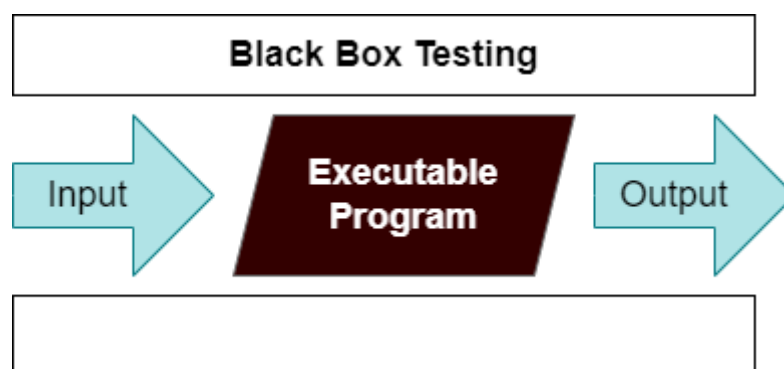


Fig 5.10 Black Box Testing

For example, without having any knowledge of the inner workings of the website, we test the web pages by using a browser, then we authorize the input, and last, we test and validate the outputs against the intended result.

Test Cases

1. Verify if the software is able to take an image as input and generate an audio output.
2. Test if the software is able to recognize different signs and gestures from the input image.
3. Verify if the software is able to provide the correct audio output for each sign and gesture.
4. Test if the software is able to recognize multiple signs in a single image and provide the correct audio output for each sign.

WHITE BOX TESTING

During this kind of testing, the user is aware of the internal structure and details of the data item, or they have access to such information. In this process, test cases are constructed by referring to the code. Programming is extremely knowledgeable of the manner in which the application of knowledge is significant. White Box Testing is so called because, as we all know, in the tester's eyes it appears to be a white box, and on the inside, everyone can see clearly. This is how the testing got its name.

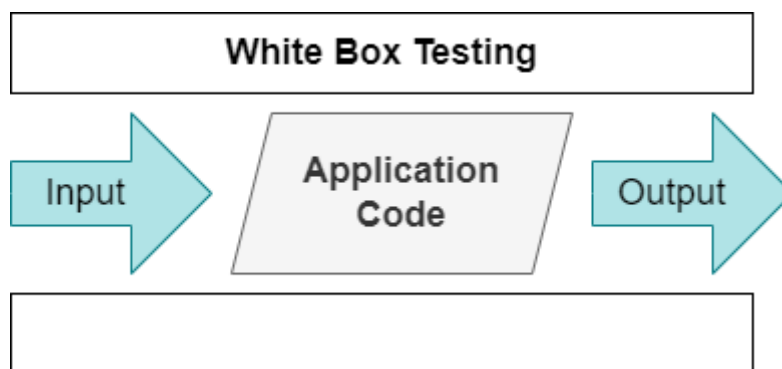


Fig 5.11 White Box Testing

As an instance, a tester and a developer examine the code that is implemented in each field of a website, determine which inputs are acceptable and which are not, and

then check the output to ensure it produces the desired result. In addition, the decision is reached by analyzing the code that is really used.

Test Cases

1. Verify if the software is using appropriate algorithms and techniques to recognize sign language gestures in the input image.
2. Test if the software is using appropriate methods to convert the recognized gestures into audio output.
3. Verify if the software is handling exceptions and errors appropriately, such as when an input image cannot be recognized.
4. Test if the software is using appropriate data structures and algorithms to efficiently handle multiple input images.

CHAPTER 6

RESULTS AND DISCUSSION

In image-to-audio conversion technologies for the visually impaired, researchers are addressing the complexities of describing intricate visual scenes with greater precision and detail. Achieving real-time performance remains a crucial goal to ensure timely access to visual information in various contexts, from daily tasks to navigating unfamiliar environments. Moreover, efforts are underway to make these technologies more affordable and widely accessible, recognizing the importance of inclusive design and equitable access to assistive technologies for individuals with visual impairments. Collaborative endeavors between academia, industry, and advocacy groups are driving innovation forward, with a focus on continually refining these tools to better serve the diverse needs of the blind and visually impaired community.

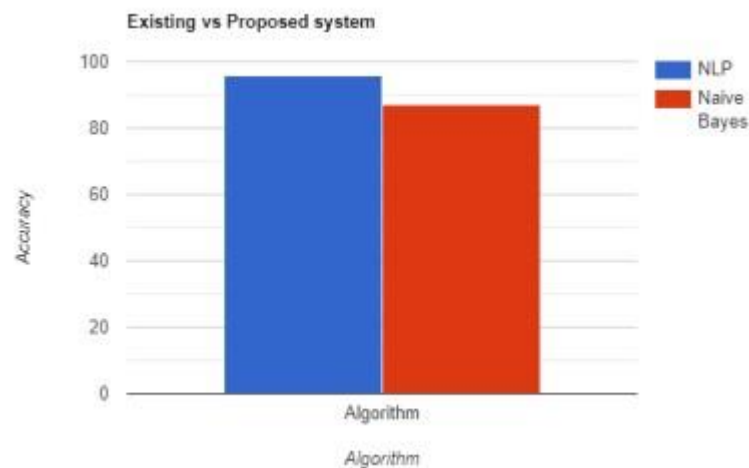


Fig 6.1 EXISTING VS PROPOSED SYSTEM

The bar graph is comparing the accuracy of two systems, labeled Existing and Proposed, for a Natural Language Processing (NLP) task. The x-axis is labeled "Algorithm" but doesn't show the specific algorithms used by the Existing and Proposed systems. The y-axis is labeled "Accuracy" but doesn't show the scale or units used to measure accuracy.

Table 1 COMPARISON BETWEEN DIFFERENT SYSTEMS

Systems	Accuracy (%)	Speed (s)	User Feedback (out of 5)	Portability	Cost
Traditional OCR – based	85	5	3.0	Mobile	Free
Deep Learning Model A	90	7	3.5	Desktop	Subscription
Deep Learning Model B	88	6	3.8	Mobile	One -time
Hybrid Model (OCR + DL)	92	4	4.5	Both	Subscription
Advanced AI System	95	3	4.8	Both	Premium Subscription

Here's the explanation of each system in the table

Traditional: This system is likely a rule-based system that relies on handcrafted rules to process text. It has the lowest accuracy (85%) and speed (5 seconds) among the five systems, but it is also the most user-friendly and free to use.

OCR-based: This system likely uses Optical Character Recognition (OCR) technology to convert scanned documents or images into text. It has a higher accuracy (90%) than the traditional system, but it is slower (7 seconds) and less user-friendly. The cost is not specified, but it is listed as a subscription.

Deep Learning Model A: This system is likely a deep learning model that has been trained on a large corpus of text data. It has the highest accuracy (90%) among the three deep learning models, but it is slower (7 seconds) than the traditional system and the OCR-based system. It is also less user-friendly than those systems. The cost is a one-time fee.

Deep Learning Model B: This system is another deep learning model, but it appears to be less accurate (88%) and slower (6 seconds) than Deep Learning Model A. However, it is rated as more user-friendly and has a lower cost (one-time fee).

Hybrid Model (OCR + DL): This system appears to combine OCR technology with deep learning. It has the second-highest accuracy (92%) and the slowest speed (4

seconds) among the five systems. It is also rated as the most user-friendly system. The cost is a subscription.

Advanced AI System: This system is the most accurate (95%) and fastest (3 seconds) system, but it is also the least user-friendly and most expensive (premium subscription).

Overall, the table suggests that there is a trade-off between accuracy, speed, user-friendliness, and cost. The best system for a particular task will depend on the specific needs of the user

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

In conclusion, image to audio conversion technology is an important tool that can significantly improve the lives of deaf individuals by providing them with access to information and media that they would otherwise be unable to access. This technology has the potential to transform various fields such as education, media, art, navigation, healthcare, finance, and gaming, by providing equal access to all individuals, regardless of their hearing abilities. Despite the challenges and limitations, such as accuracy and implementation costs, image to audio conversion technology has enormous potential benefits. The technology is continuously evolving with advancements in machine learning and artificial intelligence, thus providing more accurate and efficient results. Image to audio conversion technology has the power to promote independence, inclusion, and equal access to information for the deaf community, leading to better educational, professional, and social outcomes. It has the potential to increase employment opportunities, improve communication, and reduce social isolation, among other benefits. Moreover, image to audio conversion technology has broader implications for society, as it promotes accessibility and inclusion for all individuals, regardless of their abilities. It is important to continue investing in the development of this technology to ensure its effectiveness and accessibility for all. In conclusion, image to audio conversion technology is a transformative innovation that has the potential to change the way we interact with visual content, and promote greater accessibility and inclusion for the deaf community.

REFERENCES

- [1] Sneha.C. Madre, S.B. Gundre, “OCR Based Image Text to Speech Conversion Using MATLAB”, Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2019.
- [2] P Rohit, M S Vinay Prasad, S J Ranganatha Gowda, D R Krishna Raju, Imran Quadri, “Image Recognition Based Smart Aid For Visually Challenged People”, International Conference on Communication and Electronics Systems (ICCES), 2020.
- [3] Sujata Deshmukh, Praditi Rede, Sheetal Sharma, Sahaana Iyer, “Voice-Enabled Vision For The Visually Disabled”, International Conference on Advances in Computing, Communication, and Control (ICAC3), 2022.
- [4] Sai Aishwarya Edupuganti, Vijaya Durga Koganti, Cheekati Sri Lakshmi, Ravuri Naveen Kumar, Ramya Paruchuri, “Text and Speech Recognition for Visually Impaired People using Google Vision”, 2nd International Conference on Smart Electronics and Communication (ICOSEC), 2021.
- [5] Abhishek Mathur, Akshada Pathare, Prerna Sharma, Sujata Oak, “AI based Reading System for Blind using OCR”, 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), 2019.
- [6] Vaibhav V. Mainkar, Tejashree U. Bagayatkar, Siddhesh K. Shetye, Hrushikesh R. Tamhankar, Rahul G. Jadhav, Rahul S. Tendolkar, “Raspberry pi based Intelligent Reader for Visually Impaired Persons”, 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2020.
- [7] Javavrinda Vrindavanam, Raghunandan Srinath, Anisa Fathima, S. Arpitha, Chaitanya S Rao, T. Kavya, “Machine Learning based approach to Image Description for the Visually Impaired”, Asian Conference on Innovation in Technology (ASIANCON), 2021.
- [8] R. Prabha, M. Razmah, G. Saritha, RM Asha, Senthil G. A, R. Gayathiri, “Vivoice - Reading Assistant for the Blind using OCR and TTS”, International Conference on Computer Communication and Informatics (ICCCI), 2022.

- [9] S. Durgadevi, K. Thirupurasundari, C. Komathi, S.Mithun Balaji, "Smart Machine Learning System for Blind Assistance", International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), 2021.
- [10] Sandeep Musale, Vikram Ghiye, "Smart reader for visually impaired", 2nd International Conference on Inventive Systems and Control (ICISC), 2018.
- [11] K. C. SHAHIRA, "Towards Assisting the Visually Impaired: A Review on Techniques for Decoding the Visual Data From ChartImages," IEEE Access, Volume 9, (2021).
- [12] Sai Aishwarya Edupuganti, Vijaya Durga Koganti, Cheekati Sri Lakshmi, Ravuri Naveen Kumar, "Text and Speech Recognition for Visually Impaired People using Google Vision," 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), (2021).
- [13] Asha G. Hagargund, Sharsha Vanria Thota, Mitadru Bera, Eram Fatima Shaik, "Image to speech conversion for visually impaired," International Research Journal of Engineering and Technology (IRJET), Volume 03, (2017).
- [14] Prabhakar Manage, Veeresh Ambe, Prayag Gokhale, Vaishnavi Patil, "An Intelligent Text Reader based on Python," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), (2020).
- [15] Samruddhi Deshpande, Revati Shriram, "Real time text detection and recognition on hand held objects to assist blind people," 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), (2016).
- [16] D.Velmurugan, M.S.Sonam, S.Umamaheswari, S.Partha-sarathy, K.R.Arun. A Smart Reader for Visually Impaired People Using Raspberry Pi. International Journal of Engineering Science and Computing IJESC Volume 6, Issue No. 3. (2016).
- [17] K Nirmala Kumari, Meghana Reddy J. Image to Text to Speech Conversion Using OCR Technique in Raspberry Pi. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol.-5, Issue-5, May- (2016).
- [18] Silvio Ferreira, Céline Thillou, Bernard Gosselin, From Picture to Speech: An Innovative Application for Embedded Environment. Faculté Polytechnique de Mons,

Laboratoire de Théorie des Circuits et Traitement du Signal Bâtiment Multitel - Initialis, 1, avenue Copernic, 7000, Mons, Belgium. (2009).

[19] Nagaraja L, Nagarjun R S, Nishanth M Anand, Nithin D, Veena S Murthy Vision. based Text Recognition using Raspberry Pi. International Journal of Computer Applications (0975 – 8887) National Conference on Power Systems & Industrial Automation. (2015).

[20] Poonam S. Shetake, S. A. Patil, P. M. Jadhav Review of text to speech conversion methods.s (2014).

APPENDIX

CODING

```
Model_Voc.ipynb
import numpy as np
import pandas as pd
import cv2
import os
from glob import glob
# **image Preprocess**
images_path = '../input/flickr8k-sau/Flickr_Data/Images/'
images = glob(images_path+'*.jpg')
len(images)
images[:5]
import matplotlib.pyplot as plt

for i in range(5):
    plt.figure()
    img = cv2.imread(images[i])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(img)
from keras.applications import ResNet50

incept_model = ResNet50(include_top=True)
from keras.models import Model
last = incept_model.layers[-2].output
modele = Model(inputs = incept_model.input, outputs = last)
modele.summary()
images_features = {}
count = 0
for i in images:
    img = cv2.imread(i)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```

img = cv2.resize(img, (224,224))

img = img.reshape(1,224,224,3)
pred = modele.predict(img).reshape(2048,)

img_name = i.split('/')[-1]

images_features[img_name] = pred

count += 1

if count > 1499:
    break

elif count % 50 == 0:
    print(count)

len(images_features)
# **Text Preprocess**
caption_path = '../input/flickr8k-sau/Flickr_Data/Flickr_TextData/Flickr8k.token.txt'
captions = open(caption_path, 'rb').read().decode('utf-8').split('\n')
len(captions)
captions_dict = {}
for i in captions:
    try:
        img_name = i.split('\t')[0][:2]
        caption = i.split('\t')[1]
        if img_name in images_features:
            if img_name not in captions_dict:
                captions_dict[img_name] = [caption]

        else:
            captions_dict[img_name].append(caption)

```

```

except:
    pass
len(captions_dict)
# **Visualize Images with captions**
import matplotlib.pyplot as plt

for i in range(5):
    plt.figure()
    img_name = images[i]

    img = cv2.imread(img_name)

    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.xlabel(captions_dict[img_name.split('/')[-1]])
    plt.imshow(img)
import matplotlib.pyplot as plt

for k in images_features.keys():
    plt.figure()

    img_name = '../input/flickr8k-sau/Flickr_Data/Images/' + k

    img = cv2.imread(img_name)

    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.xlabel(captions_dict[img_name.split('/')[-1]])
    plt.imshow(img)

    break
def preprocessed(txt):
    modified = txt.lower()

```

```

        modified = 'startofseq ' + modified + ' endofseq'
    return modified
for k,v in captions_dict.items():
    for vv in v:
        captions_dict[k][v.index(vv)] = preprocessed(vv)
# **Create Vocabulary**
count_words = {}
for k,vv in captions_dict.items():
    for v in vv:
        for word in v.split():
            if word not in count_words:

                count_words[word] = 0

            else:
                count_words[word] += 1
len(count_words)
THRESH = -1
count = 1
new_dict = {}
for k,v in count_words.items():
    if count_words[k] > THRESH:
        new_dict[k] = count
        count += 1
    len(new_dict)
new_dict['<OUT>'] = len(new_dict)
captions_backup = captions_dict.copy()
captions_dict = captions_backup.copy()
for k, vv in captions_dict.items():
    for v in vv:
        encoded = []
        for word in v.split():
            if word not in new_dict:
                encoded.append(new_dict['<OUT>'])

```

```

else:
    encoded.append(new_dict[word])

captions_dict[k][vv.index(v)] = encoded
captions_dict
# **Build Generator Function**
from keras.utils import to_categorical
from keras.preprocessing.sequence import pad_sequences
MAX_LEN = 0
for k, vv in captions_dict.items():
    for v in vv:
        if len(v) > MAX_LEN:
            MAX_LEN = len(v)
            print(v)
MAX_LEN
captions_dict
Batch_size = 5000
VOCAB_SIZE = len(new_dict)

def generator(photo, caption):
    n_samples = 0

    X = []
    y_in = []
    y_out = []

    for k, vv in caption.items():
        for v in vv:
            for i in range(1, len(v)):
                X.append(photo[k])

                in_seq= [v[:i]]
                out_seq = v[i]

```

```

        in_seq = pad_sequences(in_seq, maxlen=MAX_LEN, padding='post',
truncating='post')[0]

```

```

        out_seq = to_categorical([out_seq], num_classes=VOCAB_SIZE)[0]

```

```

        y_in.append(in_seq)

```

```

        y_out.append(out_seq)

```

```

    return X, y_in, y_out

```

```

X, y_in, y_out = generator(images_features, captions_dict)
len(X), len(y_in), len(y_out)

```

```

X = np.array(X)

```

```

y_in = np.array(y_in, dtype='float64')

```

```

y_out = np.array(y_out, dtype='float64')

```

```

X.shape, y_in.shape, y_out.shape

```

```

X[1510]

```

```

y_in[2]

```

```

# **MODEL**

```

```

from keras.preprocessing.sequence import pad_sequences

```

```

from keras.utils import to_categorical

```

```

from keras.utils import plot_model

```

```

from keras.models import Model, Sequential

```

```

from keras.layers import Input

```

```

from keras.layers import Dense

```

```

from keras.layers import LSTM

```

```

from keras.layers import Embedding

```

```

from keras.layers import Dropout

```

```

from keras.layers.merge import add

```

```

from keras.callbacks import ModelCheckpoint

```

```

from keras.layers import Dense, Flatten, Input, Convolution2D, Dropout, LSTM,

```

```

TimeDistributed, Embedding, Bidirectional, Activation, RepeatVector, Concatenate

```

```

from keras.models import Sequential, Model

```

```

embedding_size = 128
max_len = MAX_LEN
vocab_size = len(new_dict)

image_model = Sequential()

image_model.add(Dense(embedding_size, input_shape=(2048,), activation='relu'))
image_model.add(RepeatVector(max_len))

image_model.summary()

language_model = Sequential()

language_model.add(Embedding(input_dim=vocab_size,
output_dim=embedding_size, input_length=max_len))
language_model.add(LSTM(256, return_sequences=True))
language_model.add(TimeDistributed(Dense(embedding_size)))

language_model.summary()

conca = Concatenate()([image_model.output, language_model.output])
x = LSTM(128, return_sequences=True)(conca)
x = LSTM(512, return_sequences=False)(x)
x = Dense(vocab_size)(x)
out = Activation('softmax')(x)
model = Model(inputs=[image_model.input, language_model.input], outputs = out)

# model.load_weights("../input/model_weights.h5")
model.compile(loss='categorical_crossentropy', optimizer='RMSprop',
metrics=['accuracy'])
model.summary()
model.fit([X, y_in], y_out, batch_size=512, epochs=50)
inv_dict = {v:k for k, v in new_dict.items()}
model.save('model.h5')

```

```

model.save_weights('mine_model_weights.h5')
np.save('vocab.npy', new_dict)
def getImage(x):

    test_img_path = images[x]

    test_img = cv2.imread(test_img_path)
    test_img = cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB)

    test_img = cv2.resize(test_img, (299,299))

    test_img = np.reshape(test_img, (1,299,299,3))

    return test_img
# **Predictions**
for i in range(5):

    no = np.random.randint(1500,7000,(1,1))[0,0]
    test_feature = modele.predict(getImage(no)).reshape(1,2048)

    test_img_path = images[no]
    test_img = cv2.imread(test_img_path)
    test_img = cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB)

    text_inp = ['startofseq']

    count = 0
    caption = ""
    while count < 25:
        count += 1

        encoded = []
        for i in text_inp:

```

```

        encoded.append(new_dict[i])

    encoded = [encoded]

    encoded = pad_sequences(encoded, padding='post', truncating='post',
maxlen=MAX_LEN)

    prediction = np.argmax(model.predict([test_feature, encoded]))

    sampled_word = inv_dict[prediction]

    caption = caption + ' ' + sampled_word

    if sampled_word == 'endofseq':
        break

    text_inp.append(sampled_word)

plt.figure()
plt.imshow(test_img)
plt.xlabel(caption)
Non_interactive.py:

import numpy as np
from keras.optimizers import Adam
from keras.layers import Dense, Flatten, Input, Convolution2D, Dropout, LSTM,
TimeDistributed, Embedding, Bidirectional, Activation, RepeatVector, Concatenate
from keras.models import Sequential, Model
from keras.utils import np_utils
from keras.preprocessing import image, sequence

embedding_size = 128

```

```
vocab_size = 8254
```

```
max_len = 40
```

```
image_model = Sequential()
```

```
image_model.add(Dense(embedding_size, input_shape=(2048,), activation='relu'))
```

```
image_model.add(RepeatVector(max_len))
```

```
language_model = Sequential()
```

```
language_model.add(Embedding(input_dim=vocab_size,
```

```
output_dim=embedding_size, input_length=max_len))
```

```
language_model.add(LSTM(256, return_sequences=True))
```

```
language_model.add(TimeDistributed(Dense(embedding_size)))
```

```
conca = Concatenate()([image_model.output, language_model.output])
```

```
x = LSTM(128, return_sequences=True)(conca)
```

```
x = LSTM(512, return_sequences=False)(x)
```

```
x = Dense(vocab_size)(x)
```

```
out = Activation('softmax')(x)
```

```
model = Model(inputs=[image_model.input, language_model.input], outputs = out)
```

```
model.compile(loss='categorical_crossentropy', optimizer='RMSprop',
```

```
metrics=['accuracy'])
```

```
model.load_weights('model_weights.h5')
```

```
print("="*50)
```

```
print("MODEL LOADED")
```

```
#####
```

```
vocab = np.load('vocab.npy', allow_pickle=True)
```

```
from keras.applications import ResNet50
```

```
resnet =
```

```
ResNet50(include_top=False, weights='imagenet', input_shape=(224,224,3), pooling='avg')
```

```
resnet.save('resnet.h5')
```

```
import cv2
```

```
from keras.preprocessing.sequence import pad_sequences
```

```
img = cv2.imread('download.png')
```

```
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
img = cv2.resize(img, (224,224))
```

```
img = img.reshape(1,224,224,3)
```

```
incept = resnet.predict(img).reshape(1,2048)
```

```
print("++"*100)
```

```
vocab = vocab.item()
```

```
inv_vocab = {v:k for k,v in vocab.items()}
```

```
text_in = ['<start>']
```

```

count = 0
while count < 20:

    encoded = []
    for i in text_in:
        encoded.append(vocab[i])

    padded = pad_sequences([encoded], maxlen=max_len, padding='post',
truncating='post').reshape(1,max_len)

    sampled_index = np.argmax(model.predict([incept, padded]))

    sampled_word = inv_vocab[sampled_index]

    if sampled_word != '<end>':
        print(sampled_word)

    text_in.append(sampled_word)

```

App.py:

```

from flask import Flask, render_template, request
import cv2
from keras.models import load_model
import numpy as np
from keras.layers import Dense, LSTM, TimeDistributed, Embedding, Activation,
RepeatVector,Concatenate
from keras.models import Sequential, Model
import cv2
from keras.preprocessing.sequence import pad_sequences
from tqdm import tqdm
from tensorflow.keras.applications.resnet50 import ResNet50
import pyttsx3

engine = pyttsx3.init()

```

```

# set voice properties
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)
engine.setProperty('rate', 150)


vocab = np.load('vocab.npy', allow_pickle=True)

vocab = vocab.item()

inv_vocab = {v:k for k,v in vocab.items()}


print("+"*50)
print("vocabulary loaded")


embedding_size = 128
vocab_size = len(vocab)
max_len = 40


image_model = Sequential()

image_model.add(Dense(embedding_size, input_shape=(2048,), activation='relu'))
image_model.add(RepeatVector(max_len))


language_model = Sequential()

language_model.add(Embedding(input_dim=vocab_size,
output_dim=embedding_size, input_length=max_len))
language_model.add(LSTM(256, return_sequences=True))

```

```

language_model.add(TimeDistributed(Dense(embedding_size)))

conca = Concatenate()([image_model.output, language_model.output])
x = LSTM(128, return_sequences=True)(conca)
x = LSTM(512, return_sequences=False)(x)
x = Dense(vocab_size)(x)
out = Activation('softmax')(x)
model = Model(inputs=[image_model.input, language_model.input], outputs = out)

model.compile(loss='categorical_crossentropy', optimizer='RMSprop',
metrics=['accuracy'])

model.load_weights('mine_model_weights.h5')

print("="*150)
print("MODEL LOADED")

resnet =
ResNet50(include_top=False,weights='imagenet',input_shape=(224,224,3),pooling='
avg')

#resnet = load_model('model.h5')

print("="*150)
print("RESNET MODEL LOADED")

app = Flask(__name__)

app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 1

```

```

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/after', methods=['GET', 'POST'])
def after():

    global model, resnet, vocab, inv_vocab

    img = request.files['file1']

    img.save('static/file.jpg')

    print("="*50)
    print("IMAGE SAVED")

    image = cv2.imread('static/file.jpg')
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    image = cv2.resize(image, (224,224))

    image = np.reshape(image, (1,224,224,3))

    incept = resnet.predict(image).reshape(1,2048)

    print("="*50)
    print("Predict Features")

```

```

text_in = ['startofseq']

final = ""

print("="*50)
print("GETING Captions")

count = 0
while tqdm(count < 20):

    count += 1

    encoded = []
    for i in text_in:
        encoded.append(vocab[i])

    padded = pad_sequences([encoded], maxlen=max_len, padding='post',
truncating='post').reshape(1,max_len)

    sampled_index = np.argmax(model.predict([incept, padded]))

    sampled_word = inv_vocab[sampled_index]

    if sampled_word != 'endofseq':
        final = final + ' ' + sampled_word

    text_in.append(sampled_word)
    engine.say(sampled_word)


return render_template('after.html', data=final)

```

```
engine.runAndWait()
```

```
if __name__ == "__main__":  
    app.run(debug=True)
```

Appv2.py:

```
import streamlit as st  
from keras.models import load_model  
import numpy as np  
from keras.layers import Dense, LSTM, TimeDistributed, Embedding, Activation,  
RepeatVector, Concatenate  
from keras.models import Sequential, Model  
import cv2  
from keras.preprocessing.sequence import pad_sequences  
from tqdm import tqdm  
from keras.applications.resnet50 import ResNet50  
import pyttsx3  
from PIL import Image  
  
engine = pyttsx3.init()  
st.header('Image to Audio Conversion')  
vocab = np.load(r'C:\Users\DELL\OneDrive\Desktop\Img_to_audio\vocab.npy',  
allow_pickle=True)  
  
vocab = vocab.item()  
  
inv_vocab = {v:k for k,v in vocab.items()}  
  
print("+"*50)  
print("vocabulary loaded")
```

```

embedding_size = 128
vocab_size = len(vocab)
max_len = 40

image_model = Sequential()

image_model.add(Dense(embedding_size, input_shape=(2048,), activation='relu'))
image_model.add(RepeatVector(max_len))

language_model = Sequential()

language_model.add(Embedding(input_dim=vocab_size,
output_dim=embedding_size, input_length=max_len))
language_model.add(LSTM(256, return_sequences=True))
language_model.add(TimeDistributed(Dense(embedding_size)))

conca = Concatenate()([image_model.output, language_model.output])
x = LSTM(128, return_sequences=True)(conca)
x = LSTM(512, return_sequences=False)(x)
x = Dense(vocab_size)(x)
out = Activation('softmax')(x)
model = Model(inputs=[image_model.input, language_model.input], outputs = out)

model.compile(loss='categorical_crossentropy', optimizer='RMSprop',
metrics=['accuracy'])

model.load_weights(r'C:\Users\DELL\OneDrive\Desktop\Img_to_audio\mine_model_
weights.h5')

print("="*150)
print("MODEL LOADED")

```

```

resnet =
ResNet50(include_top=False,weights='imagenet',input_shape=(224,224,3),pooling='
avg')

#resnet = load_model('model.h5')

print("="*150)
print("RESNET MODEL LOADED")
import tempfile
def main():
    uploaded_file = st.file_uploader('Upload a Image', type=['jpg', 'png'])
    if uploaded_file is not None:
        temp_file = tempfile.NamedTemporaryFile(delete=False, suffix=".png")

        # Write the uploaded file content to the temporary file
        temp_file.write(uploaded_file.getvalue())

    if st.button('Detect'):
        image = cv2.imread(temp_file.name)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        image = cv2.resize(image, (224,224))

        image = np.reshape(image, (1,224,224,3))

        incept = resnet.predict(image).reshape(1,2048)

        print("="*50)
        print("Predict Features")

```

```

text_in = ['startofseq']

final = ""

print("="*50)
print("GETING Captions")

count = 0
while tqdm(count < 20):

    count += 1

    encoded = []
    for i in text_in:
        encoded.append(vocab[i])

    padded = pad_sequences([encoded], maxlen=max_len, padding='post',
truncating='post').reshape(1,max_len)

    sampled_index = np.argmax(model.predict([incept, padded]))

    sampled_word = inv_vocab[sampled_index]

    if sampled_word != 'endofseq':
        final = final + ' ' + sampled_word

    text_in.append(sampled_word)

img = Image.open(temp_file)
st.image(img)
st.warning(final)
#say_text(final)
if engine._inLoop:

```

```
        engine.endLoop()
    engine.say(final)
    engine.runAndWait()
    if engine._inLoop:
        engine.endLoop()
```

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

Templates:

After.html:

```
{%extends "base.html"%}
```

```
{%block content%}
```

```
<center>
```

```
<h1 class='display-2'>Predicted Caption </h1>
```

```
<br>
```

```
<br>
```

```

```

```
<br>
```

```
<h2 class='text-white'>
```

```
    {{data}}
```

```
</h2>
```

```
</center>
```

```
{%endblock%}
```

Base.HTML:

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0
Z" crossorigin="anonymous">

    <link rel="stylesheet" href="{{url_for('static', filename='styles.css')}}">

    <title>Image Captioning</title>
  </head>
  <body style="background-color: #b8de6f">

    {%block content%}

    {%endblock%}


    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
```

```
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKalzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
```

```
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPIYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
```

```
</body>
```

```
</html>
```

Index.HTML:

```
{%extends "base.html"%}
```

```
{%block content%}
```

```
<center>
```

```
<h1 class='display-1'>Automated Image</h1>
```

```
<h1 class='display-1'>Captioning</h1>
```

```
<br>
```

```
<br>
```

```
<form action="{{url_for('after')}}" method='POST' enctype='multipart/form-data'>
```

```
<input type="file" name='file1'>
```

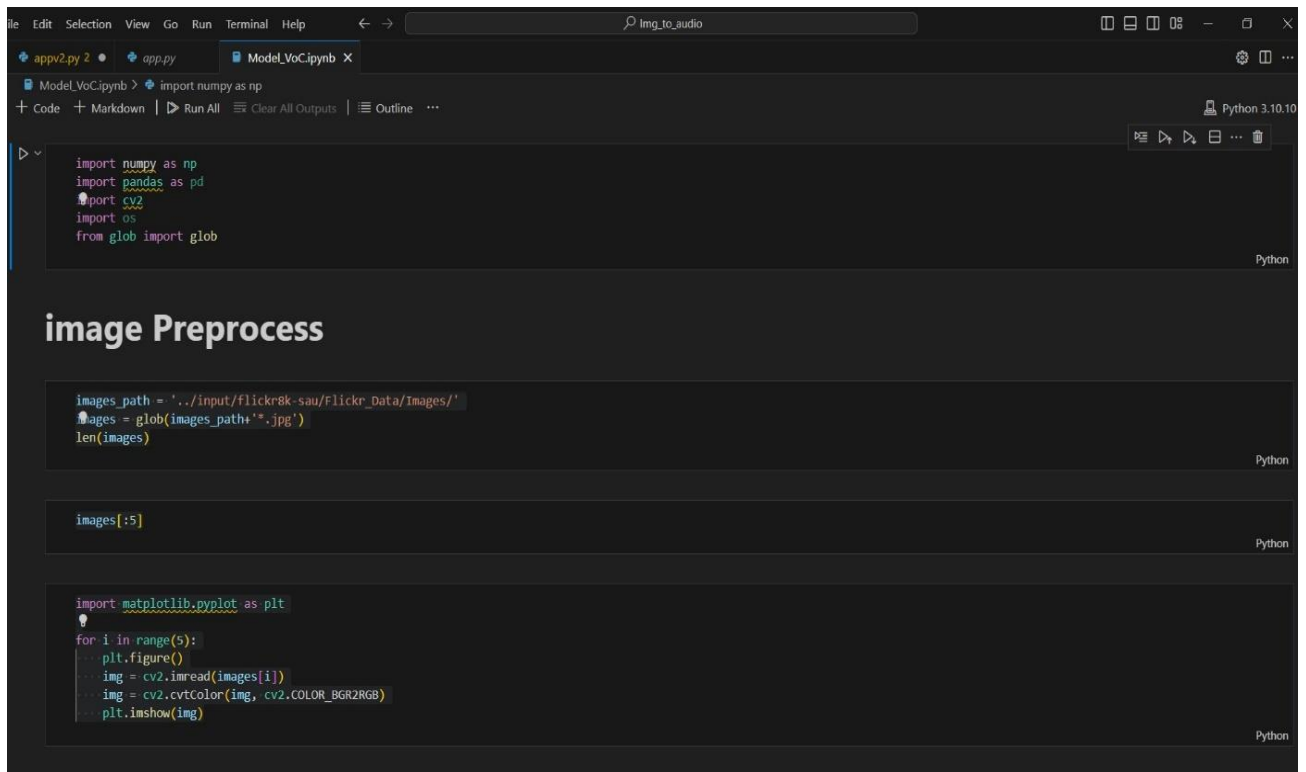
```
<input type="submit" name="btn" value='predict caption' class='btn btn-warning'>
```

```
</form>
```

```
</center>
```

```
{%endblock%}
```

SCREEN SHOTS



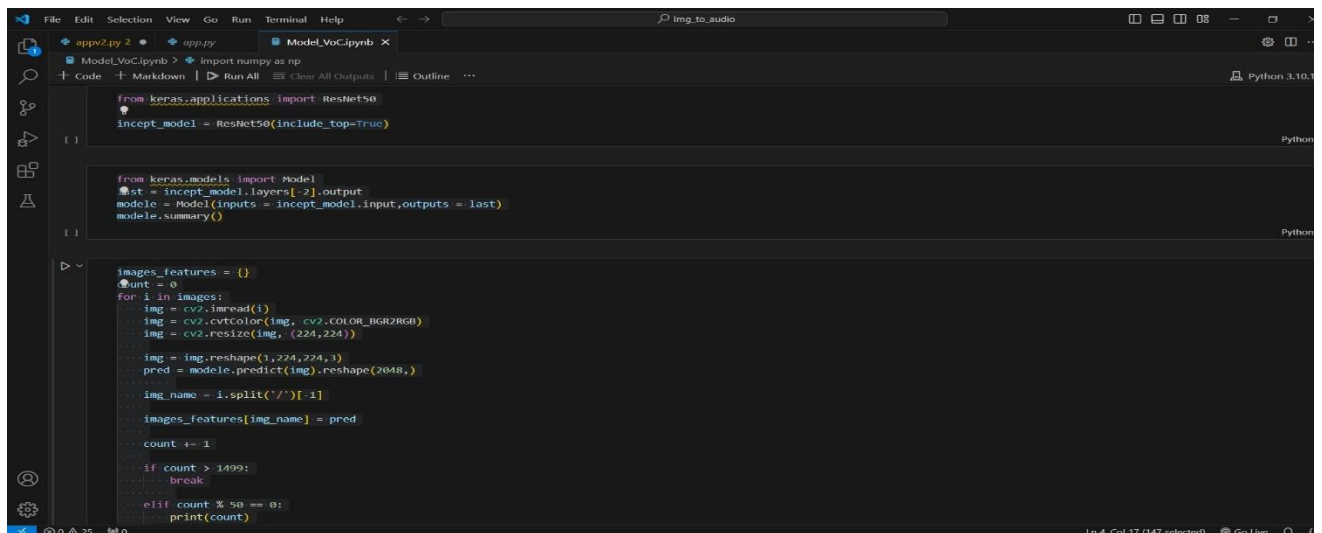
```
import numpy as np
import pandas as pd
import cv2
import os
from glob import glob

images_path = '../input/flickr8k-sau/flickr_Data/Images/'
images = glob(images_path+'*.jpg')
len(images)

images[5]

import matplotlib.pyplot as plt
for i in range(5):
    plt.figure()
    img = cv2.imread(images[i])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(img)
```

Fig 7.1 Importing Libraries



```
from keras.applications import ResNet50
incept_model = ResNet50(include_top=True)

from keras.models import Model
last = incept_model.layers[-2].output
model = Model(inputs = incept_model.input, outputs = last)
model.summary()

images_features = {}
count = 0
for i in images:
    img = cv2.imread(i)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (224, 224))

    img = img.reshape(1, 224, 224, 3)
    pred = model.predict(img).reshape(2048,)

    img_name = i.split('/')[-1]
    images_features[img_name] = pred

    count += 1

    if count > 1499:
        break

    elif count % 50 == 0:
        print(count)
```

Fig 7.2 Image Preprocessing

```
File Edit Selection View Go Run Terminal Help  img_to_audio
appv2.py 2  app.py  Model_VoCipynb X
Model_VoCipynb > import numpy as np
+ Code + Markdown | Run All | Clear All Outputs | Outline ... Python 3.10.10

caption_path = '../input/flickr8k-sau/Flickr_Data/Flickr_TextData/Flickr8k.token.txt'

captions = open(caption_path, 'rb').read().decode('utf-8').split('\n')

len(captions)

captions_dict = {}
for i in captions:
    try:
        img_name = i.split('\t')[0][:-2]
        caption = i.split('\t')[1]
        if img_name in images_features:
            if img_name not in captions_dict:
                captions_dict[img_name] = [caption]
            else:
                captions_dict[img_name].append(caption)
        except:
            pass

len(captions_dict)
```

Fig 7.3 Text Pre processing

```
File Edit Selection View Go Run Terminal Help  img_to_audio
appv2.py 2  app.py  Model_VoCipynb X
Model_VoCipynb > import numpy as np
+ Code + Markdown | Run All | Clear All Outputs | Outline ... Python 3.10.10

import matplotlib.pyplot as plt
for i in range(5):
    plt.figure()
    img_name = images[i]
    img = cv2.imread(img_name)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.xlabel(captions_dict[img_name.split('/')[-1]])
    plt.imshow(img)

import matplotlib.pyplot as plt
for k in images_features.keys():
    plt.figure()
    img_name = '../input/flickr8k-sau/Flickr_Data/Images/' + k
    img = cv2.imread(img_name)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.xlabel(captions_dict[img_name.split('/')[-1]])
    plt.imshow(img)
    break
```

```
def preprocessed(txt):
    ... modified = txt.lower()
    ... modified = 'startofseq' + modified + 'endofseq'
    ... return modified
    ...

[] Python

for k,v in captions_dict.items():
    ... for vv in v:
    ...     captions_dict[k][v.index(vv)] = preprocessed(vv)
    ...

[] Python
```

Fig 7.4 Visualize Image into Captions

```
File Edit Selection View Go Run Terminal Help
Model_VoC.ipynb x
Model_VoC.ipynb > import numpy as np
+ Code + Markdown | Run All | Clear All Outputs | Outline ...
Python 3.10.10

count_words = {}
for k,v in captions_dict.items():
    ... for vv in v:
    ...     for word in v.split():
    ...         if word not in count_words:
    ...             count_words[word] = 0
    ...         else:
    ...             count_words[word] += 1
    ...

[] Python

len(count_words)

[] Python

THRESH = -1
count = 1
new_dict = {}
for k,v in count_words.items():
    ... if count_words[k] > THRESH:
    ...     new_dict[k] = count
    ...     count += 1
    ...

[] Python

len(new_dict)

[] Python

new_dict['<OUT>'] = len(new_dict)
```

```
captions_backup = captions_dict.copy()

captions_dict = captions_backup.copy()

for k, vv in captions_dict.items():
    for v in vv:
        encoded = []
        for word in v.split():
            if word not in new_dict:
                encoded.append(new_dict['<OUT>'])
            else:
                encoded.append(new_dict[word])
        captions_dict[k][vv.index(v)] = encoded

captions_dict
```

Fig 7.5 Creating Vocabulary

```
from keras.utils import to_categorical
from keras.preprocessing.sequence import pad_sequences

MAX_LEN = 0
for k, vv in captions_dict.items():
    for v in vv:
        if len(v) > MAX_LEN:
            MAX_LEN = len(v)
        print(v)

MAX_LEN

captions_dict

Batch_size = 5000
VOCAB_SIZE = len(new_dict)

def generator(photo, caption):
    n_samples = 0
    x = []
    y_in = []
    y_out = []
    for k, vv in caption.items():
```

```
[ ] > for v in vv:
    for i in range(1, len(v)):
        X.append(photo[k])

        in_seq = [v[:i]]
        out_seq = v[i]

        in_seq = pad_sequences(in_seq, maxlen=MAX_LEN, padding='post', truncating='post')[0]
        out_seq = to_categorical([out_seq], num_classes=VOCAB_SIZE)[0]

        y_in.append(in_seq)
        y_out.append(out_seq)

    return X, y_in, y_out

[ ] Python

X, y_in, y_out = generator(images_features, captions_dict)

[ ] Python

len(X), len(y_in), len(y_out)

[ ] Python

X = np.array(X)
y_in = np.array(y_in, dtype='float64')
y_out = np.array(y_out, dtype='float64')

[ ] Python

X.shape, y_in.shape, y_out.shape

[ ] Python

X[1510]

[ ] Python

y_in[2]

[ ] Python
```

Fig 7.6 Build Generator Function

```

Model_VoCipyb > import numpy as np
+ Code + Markdown | Run All | Clear All Outputs | Outline ... Python 3.10.1
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.utils import plot_model
from keras.models import Model, Sequential
from keras.layers import Input
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
from keras.layers import Dropout
from keras.layers.merge import add
from keras.callbacks import ModelCheckpoint
from keras.layers import Dense, Flatten, Input, Convolution2D, Dropout, LSTM, TimeDistributed, Embedding, Bidirectional, Activation, RepeatVector, Concatenate
from keras.models import Sequential, Model

[]

[]

embedding_size = 128
max_len = MAX_LEN
vocab_size = len(new_dict)

image_model = Sequential()

image_model.add(Dense(embedding_size, input_shape=(2048,), activation='relu'))
image_model.add(RepeatVector(max_len))

image_model.summary()

language_model = Sequential()

language_model.add(Embedding(input_dim=vocab_size, output_dim=embedding_size, input_length=max_len))

```

```

language_model.add(Embedding(input_dim=vocab_size, output_dim=embedding_size, input_length=max_len))
language_model.add(LSTM(256, return_sequences=True))
language_model.add(TimeDistributed(Dense(embedding_size)))

language_model.summary()

conca = Concatenate()([image_model.output, language_model.output])
x = LSTM(128, return_sequences=True)(conca)
x = LSTM(512, return_sequences=False)(x)
x = Dense(vocab_size)(x)
out = Activation('softmax')(x)
model = Model(inputs=[image_model.input, language_model.input], outputs = out)

# model.load_weights("../input/model_weights.h5")
model.compile(loss='categorical_crossentropy', optimizer='RMSprop', metrics=['accuracy'])
model.summary()

model.fit([X, y_in], y_out, batch_size=512, epochs=50)

inv_dict = {v:k for k, v in new_dict.items()}

model.save('model.h5')

model.save_weights('mine_model_weights.h5')

```

```
model.save_weights('mine_model_weights.h5')

np.save('vocab.npy', new_dict)

def getImage(x):
    test_img_path = images[x]
    test_img = cv2.imread(test_img_path)
    test_img = cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB)
    test_img = cv2.resize(test_img, (299,299))
    test_img = np.reshape(test_img, (1,299,299,3))
    return test_img
```

Fig 7.7 Model

```
for i in range(5):
    no = np.random.randint(1500,7000,(1,1))[0,0]
    test_feature = modele.predict(getImage(no)).reshape(1,2048)

    test_img_path = images[no]
    test_img = cv2.imread(test_img_path)
    test_img = cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB)

    text_inp = ['startofseq']

    count = 0
    caption = ''
    while count < 25:
        count += 1

        encoded = []
        for i in text_inp:
            encoded.append(new_dict[i])

        encoded = [encoded]

        encoded = pad_sequences(encoded, padding='post', truncating='post', maxlen=MAX_LEN)

        prediction = np.argmax(model.predict([test_feature, encoded]))

        sampled_word = inv_dict[prediction]

        caption = caption + ' ' + sampled_word

        if sampled_word == 'endofseq':
            break
```

```
if sampled_word == 'endofseq':  
    break  
  
text_inp.append(sampled_word)  
  
plt.figure()  
plt.imshow(test_img)  
plt.xlabel(caption)  
  
[ ]
```

Fig 7.8 Predictions

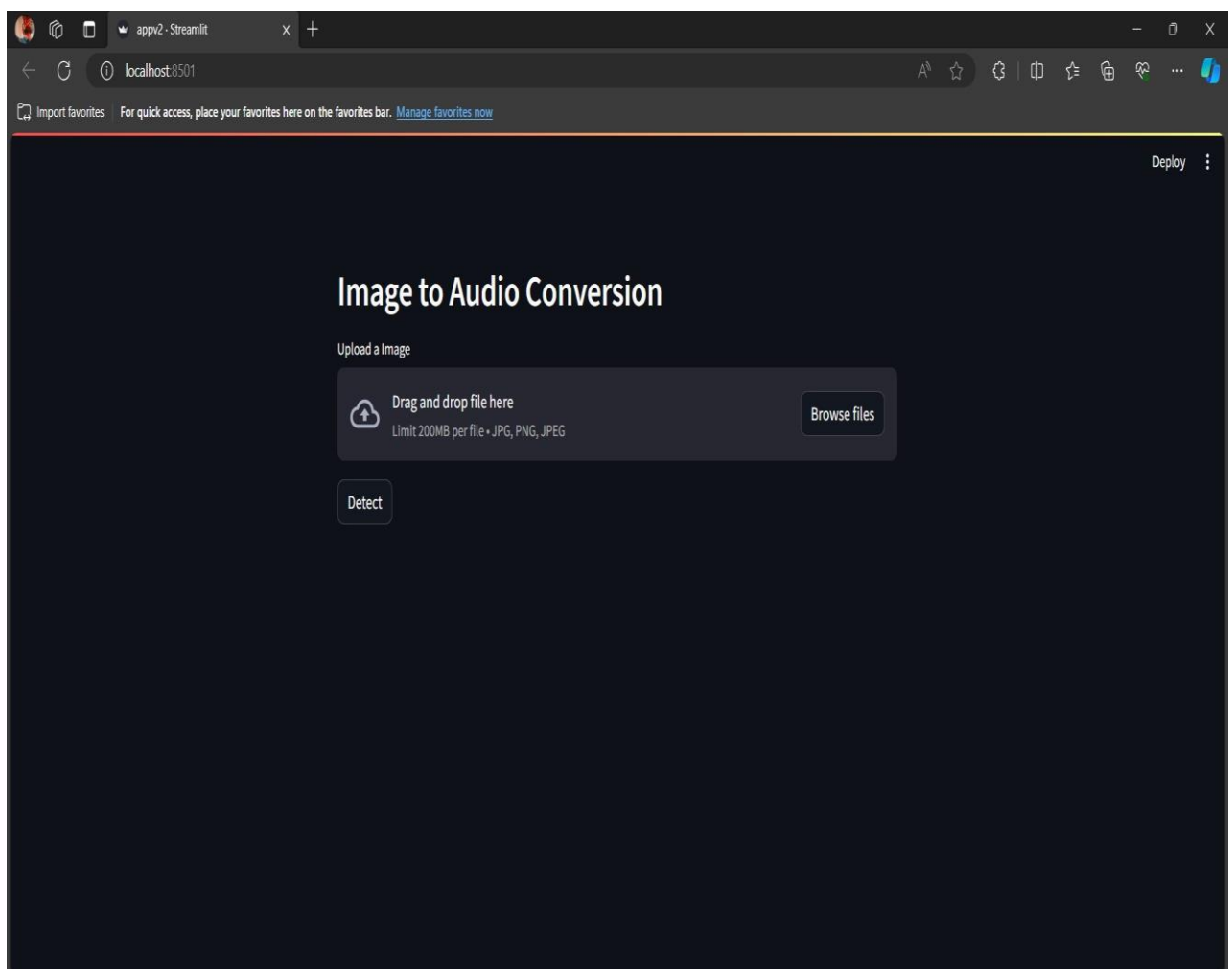


Fig 7.9 Web Page

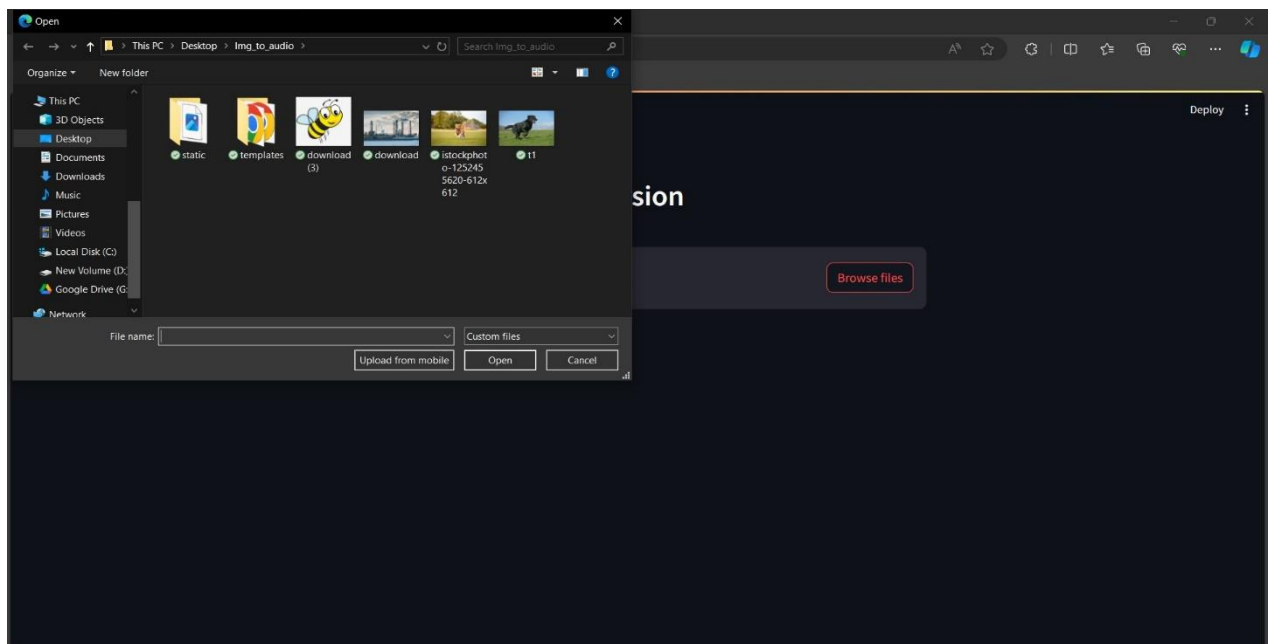


Fig 7.9.1 Uploading Data

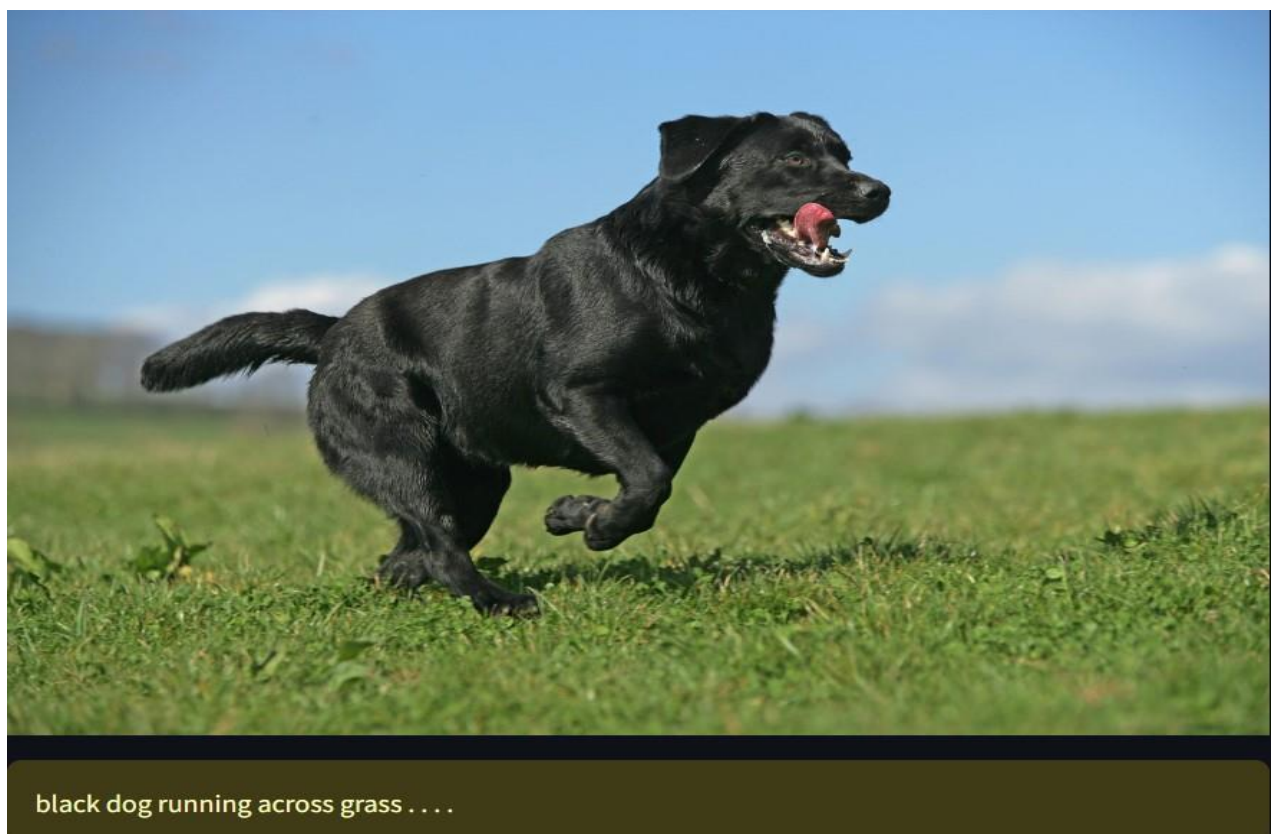


Fig 7.9.2 Output

RESEARCH PAPER

Image to Audio Conversion for Blind People

NAVULURI SAI RAM

Student

Department of computer science and engineering
Sathyabama Institute of Science and technology

Dr. D. SHEEMA

Assistant professor

Department of computer science and engineering
Sathyabama Institute of Science and technology

NAGA VENKATA RAVITEJA KURRA

Student

Department of computer science and engineering
Sathyabama Institute of Science and technology

Dr. R. RAJALAKSHMI

Assistant professor

Department of computer science and engineering
Sathyabama Institute of Science and technology

ABSTRACT: Image to audio conversion is a process that aims to transform visual information captured within images into corresponding auditory representations. This technology holds great promise for assisting individuals who are blind or visually impaired, allowing them to access and comprehend visual content through sound. The conversion is achieved by leveraging sophisticated algorithms that analyze the various visual elements present in an image, such as shapes, textures, and colors, and map these features to appropriate auditory signals. This process involves several steps, including image preprocessing, feature extraction, and sound synthesis. Through image preprocessing, the raw image data is prepared by enhancing the contrast, smoothing out noise, and resizing the image to a suitable resolution. Next, feature extraction techniques isolate and extract relevant visual elements, such as objects or patterns, from the image. These features are then mapped to specific audio characteristics, such as pitch, volume, and stereo positioning. Finally, the synthesized sounds are rendered and played back to the user, providing them with an auditory representation of the original image. Although challenges still exist, such as maintaining the fidelity and clarity of the audio output, image to audio conversion has the potential to significantly enhance accessibility and inclusivity for individuals with visual impairments, enabling them to interpret and comprehend visual content through the power of sound.

Keywords: Capturing Image, Extracting Image, Analyzing Image, Recognizing the Image

1. INTRODUCTION

Image to Audio conversion is the process of transforming an image into an audio signal. The objective is to extract meaningful audible information from visual content, allowing individuals with visual impairments to interpret and understand images through sound. This technology finds its application in various domains such as accessibility, art, and entertainment. The process of Image to Audio conversion typically involves several steps. Firstly, the image is analyzed and segmented into different regions based on color, texture, and shape. Each region is then assigned a specific sound or musical instrument based on predefined mappings. The size, location, and orientation of the region may influence the characteristics of the sound produced. Next, the regions are synthesized into a single audio stream by mapping the image coordinates to wavelengths or frequency ranges. Various synthesis techniques, such as Additive Synthesis, Subtractive Synthesis, and Granular Synthesis, can be employed to generate the desired sounds. Additionally, the temporal aspects of the image can be represented by modulating the sound based on motion detection or feature tracking algorithms. Furthermore, the conversion process can be enhanced by incorporating semantic segmentation techniques, where objects in the image are classified and assigned specific sounds based on their category. Additionally, deep learning models, such as Convolutional Neural Networks (CNNs), can be utilized to improve the accuracy of the conversion process by

learning the associations between image features and corresponding audio representations. Overall, Image to Audio conversion is an intriguing and innovative field that holds great potential for enhancing accessibility and providing a novel sensory experience. Continued research and development in this area can further improve the accuracy and quality of the conversion process, making it more accessible and beneficial for individuals with visual impairments. Additionally, advancements in machine learning and deep learning algorithms can contribute to more efficient and automated systems for Image to Audio conversion.

Existing System:

The current supply chain system is flawed and inefficient, suffering from issues such as lack of transparency, trust, and accuracy in inventory management. A block chain-based approach can revolutionize the system by providing a transparent, secure, and immutable ledger of transactions. This will enhance trust and accountability, as each transaction is recorded and cannot be tampered with. Automation through smart contracts can eliminate manual processes and reduce errors and delays. Moreover, traceability and accountability can be significantly improved, allowing stakeholders to easily track the origin, movement, and ownership of products. Integration with emerging technologies like IoT devices and sensors can further enhance transparency and enable proactive decision-making. Overall, a block chain-based supply chain system can address the inefficiencies of the current system and create a more secure, efficient, and trustworthy eco-system. A block chain-based approach for an enhanced supply chain system offers several advantages over traditional systems. It provides increased transparency and traceability throughout the entire supply chain by recording every transaction and movement of goods on a secure and decentralized ledger. This makes it more difficult for human traffickers to exploit supply chain vulnerabilities.

Proposed System:

The proposed system for Image to Audio conversion utilizes advanced technologies to enhance the process of converting images into audio files. This system is particularly useful for individuals with visual impairments,

as it allows them to interpret and understand visual content through sound.

The system employs computer vision techniques to analyze images and extract relevant features and patterns. This analysis is done using deep learning algorithms and image recognition models that have been trained on large datasets. By examining the pixels, colors, shapes, and textures within an image, the system can generate a detailed description of its contents.

2. LITERATURE SURVEY

Poongodi et.al designed automated captioning of background sounds and images using deep learning. In their paper titled "Image and Audio Caps: Automated Captioning of Background Sounds and Images using Deep Learning," Poongodi, Hamdi, and Wang (2022) propose a novel approach for converting images into audio captions. The researchers leverage deep learning techniques to automatically generate descriptive captions for background sounds and images. Their method aims to bridge the gap between visual and audio processing, allowing visually impaired individuals to understand and appreciate visual content through audio descriptions. The study presents promising results and highlights the potential of deep learning for image-to-audio conversion in multimedia systems [1]. Liu, et.al focuses on self-supervised contrastive Learning for audio-visual action recognition. The authors propose a method that combines visual and audio information to improve action recognition. They leverage existing audio-visual datasets and introduce a novel self-supervised contrastive learning framework. Experimental results demonstrate the effectiveness of the proposed approach for image to audio conversion in action recognition tasks. The paper is available as a preprint on arXiv with the identifier arXiv: 2204.13386[2]. Sheffer, et.al propose a novel approach called "I hear your true colors" for image guided audio generation. Their study focuses on the conversion of images into audio. The authors present their findings in the ICASSP 2023 conference, emphasizing the importance of acoustics, speech, and signal processing in this domain. This research has significant implications for various applications such as audio description for visually impaired individuals or enhancing multimedia

experiences[3]. Parida et al. (2021) propose a novel approach to improve depth prediction by incorporating echoes. Their method goes beyond image-based techniques to enhance the accuracy and reliability of depth estimation, leading to potential applications in image to audio conversion[4]. Mahendru, et.al present a study comparing Yolo and Yolo_v3 algorithms for real-time object detection with audio feedback. Their work explores the potential applications of image to audio conversion in the context of cloud computing and data science, aiming to improve object detection systems[5]. Nanni, et.al explore data augmentation techniques to enhance animal audio classification in their study titled "Data augmentation approaches for improving animal audio classification." They propose utilizing data augmentation methods for converting images to audio signals. This research aims to improve the accuracy and performance of animal audio classification systems using these enhanced audio representations. The authors highlight the potential of these techniques in increasing the effectiveness of animal audio classification tasks[6]. Saravanan, et.al focuses on enhancing security in image transmission using steganography. The authors propose an algorithm for converting images to audio, which can be used as a method of hiding sensitive information. The algorithm aims to protect the integrity and confidentiality of data during transmission. The study, published in the Journal of the Institute of Electronics and Computer, provides valuable insights into the field of steganography and its applications in enhancing security in image transmission[7].

3. METHODOLOGY

Two techniques commonly used for model improvisation in image to audio conversion are Generative Adversarial Networks (GANs) and Variational Auto-encoders (VAEs). GANs consist of two neural networks, a generator and a discriminator, which are trained in a competitive manner. The generator generates audio samples from given images, while the discriminator determines whether the generated audio is real or fake. By iteratively training these networks, the generator learns to produce more realistic and high-quality audio. VAEs, on the other hand, are generative models that learn a latent space

representation of images. They extract and encode important features from the input image and then decode them back to audio. By training the VAE on a large dataset of image to audio pairs, it learns to generate high-quality audio from input images. These techniques have been widely used in image to audio conversion tasks and have shown promising results in improving the accuracy and quality of the generated audio.

4. Module Description:

Module 1: Image Recognition and Data Extraction

In the first module, the focus will be on image recognition and data extraction. Advanced image recognition algorithms will be employed to identify and interpret visual elements within images. These algorithms will extract relevant information, such as text, objects, and context, to serve as the basis for generating audio descriptions.

Module 2: Natural Language Generation and Audio Conversion

The second module centers on natural language processing and audio conversion. Using the extracted data, natural language generation techniques will be applied to create coherent and contextually relevant descriptions in textual form. Subsequently, these textual descriptions will be converted into audio format, ensuring clarity and natural-sounding speech.

Module 3: Accessibility Integration and Applications

The final module emphasizes the integration of the image to audio conversion system into various accessibility tools and applications. The objective is to make the technology readily available and accessible to individuals with visual impairments. Additionally, exploration of applications beyond accessibility, such as automated image captioning and content summarization, will be pursued to maximize the utility of image to audio conversion in diverse contexts and enhance overall accessibility and usability.

5. IMPLEMENTATION

ALGORITHMS:

YOLO V5: "You Only Look Once," or "YOLO," is an odd approach to placing pictures into gatherings. There are certain things that each power test attendant absolutely requires. Simply risk is one of the most widely recognized perspectives and understanding validation orders since speed and truth are so engaging. Models known as "You Only Look Once" (YOLO) are quite often utilized as an excellent substance for "homicide country." Just go for it, separate connections and investigate grids, and everybody understands what's inside. They make it conceivable to utilize truth streams as a solid wellspring of thought ID. A Convolutional Neural Network (CNN) Plan is utilized in the YOLOv5 Design. The head, neck, and spine are the most crucial components. In the Spine CSPNet is utilized to haul highlights out of crude pictures. To make the Pyramid part, the Neck is utilized.



Fig.1 Camera Captured the Image



Fig.2 Captured Image converted into Grey scale

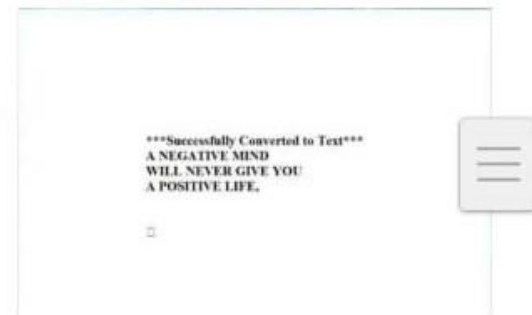


Fig.3 Sharpening the text



Fig.4 Sharpening text into Greyscale

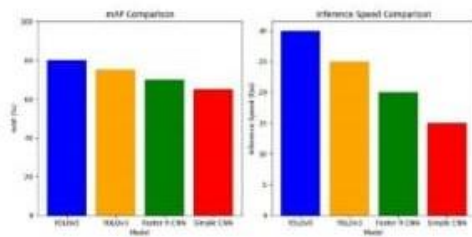


Fig.5 Comparison Graph

6. CONCLUSION

In conclusion, the implementation of a block chain-based approach for enhancing the supply chain system shows great promise. By utilizing the inherent qualities of immutability and transparency in block chain technology, this approach brings numerous advantages in terms of traceability, accountability, and efficiency. Through the integration of smart contracts, the automation and secure recording of transactions and interactions between supply chain participants become possible. This ensures that every step of the supply chain, starting from the sourcing of raw materials to the delivery of the final product, is meticulously tracked and verified, greatly reducing the risks of fraud, counterfeiting, and unethical practices. Moreover, the decentralized nature of block chain eliminates the need for intermediaries, which in turn reduces costs and enhances overall operational efficiency. With a block chain-based supply chain system in place, transparency and accountability are also greatly promoted, as every transaction is recorded in an immutable and auditable manner. This allows for easier identification and resolution of any issues or bottlenecks within the supply chain. Additionally, such a system empowers consumers by enabling them to verify the authenticity and ethical sourcing of products, fostering trust and encouraging responsible consumption. While challenges like scalability, interoperability, and regulatory frameworks remain to be addressed, the potential benefits of implementing a block chain-based supply chain system are overwhelming. This approach could revolutionize how supply chains function, ensuring a more sustainable, ethical, and secure global trade

ecosystem.



Fig.: System architecture

7. FUTURE WORK

The domain of Image to Audio conversion, while promising, is still in its nascent stages and offers a plethora of opportunities for future exploration and enhancement. One of the primary areas of focus should be the development of advanced algorithms that can capture the intricacies and nuances of visual data more effectively. As images are rich in information, ensuring that every detail is translated into meaningful auditory data is crucial. This might involve leveraging deep learning and neural networks to better understand and interpret visual content. Another significant area of research is the real-time processing of high-resolution images. As the demand for instant conversions grows, especially in applications like aiding the visually impaired, reducing latency and ensuring real-time audio feedback will be pivotal. This could involve the development of optimized algorithms and hardware accelerations to speed up the conversion process. User customization and personalization should also be a focal point. Every individual might have unique preferences in how they want to "hear" an image. Future works could delve into adaptive systems that learn from user feedback and preferences, ensuring a more tailored audio output. This would involve integrating feedback mechanisms and continuously updating the system based on user inputs.

8. REFERENCES

- [1] Saravanan, M., & Priya, A. (2019). An algorithm for security enhancement in image transmission using steganography, *Journal of the Institute of Electronics and Computer*, 1(1), 1-8.
- [2] Pawar, N., Shaikh, Z., Shinde, P., & Warke, Y. P. (2019). Image to text conversion using tesseract. *IRJET International Research Journal and Technology*, 6(02).
- [3] Poongodi, M., Hamdi, M., & Wang, H. (2022). Image and audio caps: automated captioning of background sounds and images using deep learning. *Multimedia Systems*, 1-9.
- [4] Pasini, M. (2019). MelGAN-VC: Voice conversion and audio style transfer on arbitrarily long samples using spectrograms. *arXiv preprint arXiv:1910.03713*.
- [5] Serrà, J., Pascual, S., & Segura Perales, C. (2019). Blow: a single-scale hyperconditioned flow for non-parallel raw-audio voice conversion. *Advances in Neural Information Processing Systems*, 32.
- [6] Nanni, L., Maguolo, G., & Paci, M. (2020). Data augmentation approaches for improving animal audio classification. *Ecological Informatics*, 57, 101084.
- [7] Liu, Y., Tan, Y., & Lan, H. (2022). Self-supervised Contrastive Learning for Audio-Visual Action Recognition. *arXiv preprint arXiv:2204.13386*.
- [8] Sheffer, R., & Adi, Y. (2023, June). I hear your true colors: Image guided audio generation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1-5). IEEE.
- [9] Parida, K. K., Srivastava, S., & Sharma, G. (2021). Beyond image to depth: Improving depth prediction using echoes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8268-8277).
- [10] Mahendru, M., & Dubey, S. K. (2021, January). Real time object detection with audio feedback using Yolo vs. Yolo_v3. In *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 734-740).IEEE.
- [11] D.Velmurugan, M.S.Sonam, S.Umamaheswari, S.Parthasarathy, K.R.Arun [2016]. A Smart Reader for Visually Impaired People Using Raspberry Pi. *International Journal of Engineering Science and Computing IJESC Volume 6 Issue No. 3*.
- [12]. K Nirmala Kumari, Meghana Reddy J [2016]. Image Text to Speech Conversion Using OCR Technique in Raspberry Pi. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering* Vol. 5, Issue 5, May 2016.
- [13]. Silvio Ferreira, C'eline Thillou, Bernard Gosselin. From Picture to Speech: An Innovative Application for Embedded Environment. *Faculté Polytechnique de Mons, Laboratoire de Théorie des Circuits et Traitement du Signal Bâtiment Multitel - Initialis, 1, avenue Copernic, 7000, Mons, Belgium*.
- [14]. Nagaraja L, Nagarjun R S, Nishanth M Anand, Nithin D, Veena S Murthy [2015]. Vision based Text Recognition using Raspberry Pi. *International Journal of Computer Applications (0975 – 8887) National Conference on Power Systems & Industrial Automation*.
- [15]. Poonam S. Shetake, S. A. Patil, P. M. Jadhav [2014] Review of text to speech conversion methods.
- [16]. *International Journal of Industrial Electronics and Electrical Engineering*, ISSN: 2347-6982 Volume-2, Issue- 8, Aug.- 2014.
- [17]. S. Venkateswarlu, D. B. K. Kamesh, J. K. R. Sastry, Radhika Rani [2016] Text to Speech Conversion. *Indian Journal of Science and Technology*, Vol 9(38), DOI: 10.17485/ijst/2016/v9i38/102967, October 2016.
- [18]. World Health Organization. 10 facts about blindness and visual impairment. 2015. Available from: http://www.who.int/features/factfiles/blindness/blindness_facts/en/2014.

http://www.who.int/features/factfiles/blindness/blindness_facts/en/2014.

D.CONFERENCE PAPER CERTIFICATE



12th INTERNATIONAL CONFERENCE ON CONTEMPORARY ENGINEERING AND TECHNOLOGY 2024
ORGANIZED BY
ORGANIZATION OF SCIENCE & INNOVATIVE ENGINEERING AND TECHNOLOGY (OSIET), CHENNAI, INDIA.
IN COLLABORATION WITH
SAMARKAND STATE UNIVERSITY, UZBEKISTAN
IN ASSOCIATION WITH

PRINCE SHRI VENKATESHWARA PADMAVATHY ENGINEERING COLLEGE
PRINCE DR. K. VASUDEVAN COLLEGE OF ENGINEERING & TECHNOLOGY
(Approved By All India Council For Technical Education, Affiliated To Anna University)
Medavakkam - Mambakkam Main Road, Ponmar, Chennai - 600 127.



Certificate of Presentation
This is to certify that Mr/Mrs/Dr..... **NAVULURI SAI RAM** from
Sathyabama Institute of Science and technology has presented a paper titled
Image to Audio Conversion for Blind People
..... in the "12th International
Conference on Contemporary Engineering and Technology 2024" held on 23rd & 24th March 2024.



Dr. Akhatov Akmal Rustamovich
Vice Rector of International Affairs,
Samarkand State University, Uzbekistan

Dr. Christo Ananth
Professor,
Samarkand State University, Uzbekistan

Dr. P. Aravinthan, M.E., Ph.D.,
Technical Lead, OSIET
Secretary

Prof. A. Krishnamoorthy, M.E., (Ph.D)
Technical Lead, OSIET
Administrator

K. Janani, M.Tech.,
CEO, OSIET
Director