A

Project Report

on

MULTI MODAL ASSISTIVE SYSTEM FOR PEOPLE WITH DISABILITIES

Submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

by

Chithanuru Chandhana

(20EG105310)

Kandula Sri Pavani

(20EG105323)

Medoju Vamshi Krishna

(20EG105328)

Sunchikala Jhansi

(20EG105348)

Under the guidance of

Mr. G. Kiran Kumar

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ANURAG UNIVERSITY
VENTAKAPUR (V), GHATKESAR (M), MEDCHAL (D), T.S - 500088
TELANGANA
(2023-2024)

ANURAG UNIVERSITY | SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGNIEERING

CERTIFICATE

This is to certify that the project entitled **“Multi-Modal Assistive System for People with Disabilities”** being submitted by **Chithanuru Chandhana** bearing the Hall Ticket number **20EG105310, Kandula Sri Pavani** bearing the Hall Ticket number **20EG105323, Medoju Vamshi Krishna** bearing the Hall Ticket number **20EG105328** and **Sunchikala Jhansi** bearing the Hall Ticket numbers **20EG105348** in partial fulfilment of the requirements for the award of the degree of the **Bachelor of Technology** in **Computer Science and Engineering** in **Anurag University** is a record of bonafide work carried out by them under my guidance and supervision in academic year 2023 to 2024.

The results presented in this project have been verified and found to be satisfactory. The results embodied in this project report have not been submitted to any other University for the award of any other degree or diploma.

Signature of The Supervisor
G. Kiran Kumar
Assistant Professor

Signature Dean,
Dr. G. Vishnu Murthy
Department of CSE

External Examiner

ACKNOWLEDGEMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **G. KIRAN KUMAR**, **Assistant Professor, Department of Computer Science and Engineering**, Anurag University for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved my grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

We would like to acknowledge my sincere gratitude for the support extended by **Dr. G. VISHNU MURTHY**, **Dean, Department of Computer Science and Engineering,** Anurag University.

We also express my deep sense of gratitude to **Dr. V. V. S. S. S. BALARAM**, Academic coordinator. **Dr. T. SHYAM PRASAD,** Assistant Professor**,** Project Coordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of my project work.

We would like to express special thanks to **Dr. V. VIJAYA KUMAR, Dean School of Engineering,** Anurag University, for his encouragement and timely support in our B. Tech program.

Chithanuru Chandhana

20EG105310

Kandula Sri Pavani

20EG105323

Medoju Vamshi Krishna

20EG105328

Sunchikala Jhansi

20EG105348

DECLARATION

We hereby declare that the project work entitled **"Multi-Modal Assistive System for Enhanced Accessibility and Communication for People with Disabilities"** submitted to the **Anurag University** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology (B. Tech)** in Computer Science and Engineering is a record of an original work done by us under the guidance of **Mr. G. Kiran Kumar, Assistant Professor** and this project work have not been submitted to any other university for the award of any other degree or diploma.

Place: Anurag University, Hyderabad

Date:

Chithanuru Chandhana
20EG105310

Kandula Sri Pavani
20EG105323

Medoju Vamshi Krishna
20EG105328

Sunchikala Jhansi
20EG105348

ABSTRACT

Millions of individuals grappling with disabilities encounter daily hurdles. When it comes to navigating communication and accessibility obstacles. Deaf communities often find it challenging to bridge the gap with the hearing populace, while those with limited speech or writing capabilities struggle to find platforms for self-expression. Additionally, individuals with visual or mobility impairments face daunting tasks when navigating their surroundings. Unfortunately, current assistive technologies tend to address these challenges separately, lacking a cohesive solution. Traditionally, the focus has been on single modalities such as sign language translation apps or text-to-speech software. While these tools offer some assistance, they only scratch the surface, leaving significant gaps in usability and effectiveness. Moreover, navigation aids typically rely on visual maps or preset routes, which can impede the autonomy of those with mobility limitations. These fragmented solutions fail to provide a comprehensive answer to the multifaceted challenges individuals with disabilities encounter. Enter the Multi-Modal Assistive System (MAS), presenting a paradigm shift in addressing these issues. This innovative platform integrates five interconnected modules: sign language translation, text-to-speech/speech-to-text conversion, image-to-speech capabilities, Visual Question Answering and a customizable user interface. By tackling communication and accessibility complexities simultaneously, MAS offers a comprehensive solution that surpasses existing fragmented approaches. This holistic approach not only fosters greater independence and communication but also enhances engagement for individuals with disabilities, seamlessly integrating them into society. By removing communication barriers and empowering individuals to navigate their environments confidently, MAS lays the groundwork for a more inclusive and accessible future for all.

Keywords – Visual impairment, sign translation, Image/Text to speech, Sign Language Translation, Multi-Modal Assistive System, Visual Question Answering

TABLE OF CONTENTS

S. No.	CONTENT	Page No.
1.	Introduction	1
	1.1.Motivation	2
	1.2. Problem Definition	3
	1.3.Problem Illustration	3
	1.4. Objective of the Project	8
2.	Literature Document	9
3.	Proposed Method	12
	3.1 Visual Question and Answering	12
	3.1.1 Exploratory Data Analysis	12
	3.1.2 Data pipelining and generation	14
	3.1.3 Fine tuning and training the model	15
	3.2. Speech/text to Sign Language	16
	3.2.1 Data gathering	16
	3.2.2 Input speech/text preprocessing	17
	3.2.3 Translating Text to Sign Language Videos	19
	3.3. Image/text to Speech	20
	3.3.1 Setting up OCR Environment with Pytesseract	20
	3.3.2 Image Preprocessing	21
	3.3.3 Speech Generating Function	22
4.	Implementation	24
	4.1. Functionalities	24
	4.1.1 Visual Question and Answering	24
	4.1.2 Speech/text to Sign Language	26
	4.1.3 Image/text to Speech	29
	4.2. Attributes	30

4.3. Dataset 33

4.4. Experimental Screenshot 34

5. Experimental Results 37

5.1. Experiment setup 37

5.1.1. Setup Jupyter Notebook 37

5.1.2 Setting Up Visual Studio Code 38

5.1.3. Setup Flask 39

5.2. Libraries used 40

5.3. Parameters with formulas 42

6. Discussion of Results 45

6.1. Visual Question and Answering 45

6.2. Speech/text to Sign Language 46

6.3. Image/text to Speech 46

7. Summary, Conclusion and Recommendation 47

8. Future Enhancements 48

9. References 49

List of Figures

Figure No.	Figure Name	Page No.
1.1	Problems faced by people with disabilities	2
1.2	Percentage of adults with functional disability types	4
1.3	Datasets used for visual question answering task	5
1.4	Stats of different methodologies used for visual question answering task.	5
1.5	Comparison of continuous sign language translation datasets	6
1.6	Translation scores for a random sample of 291 pairs from ISL Translate	7
1.7	The results of the performance tests conducted for the OCR engines.	8
1.8	Objective of the project	8
3.1	Distribution of Image Height & Width	12
3.2	Distribution of Question's Character count & Word count	13
3.3	Distribution of answer type	13
3.4	Model Architecture for Visual question and answering	16
3.5	Speech to text preprocessing	19
3.6	Speech/text to sign Translation	20
3.7	Image preprocessing for text extraction	22
3.8	Image/text to Speech Translation	23
4.1	MS Coco dataset	33

4.2	ISL dataset	34
4.3	A web interface	34
4.4	A web interface integrated with multiple features	35
4.5	Visual question and answering	35
4.6	Text to sign language translator	35
4.7	Speech to sign language translator	36
4.8	Image to speech translator	36
5.1	Jupyter Notebook	38
5.2	Visual Studio Code	39
5.3	Installing the Flask and its dependencies	40
6.1	BLEU Score Comparison of VQA model	45
6.2	Completeness and time comparison	46
6.3	Precision Comparison of OCR model	46

List of Tables

Table No.	Table Name	Page No.
2.1	Comparison of Existing Method from selected Strategies	11
5.1	Parameter comparison table between existing methods and proposed methods	44
6.1	comparison of BLEU score for VQA between existing methods and proposed methods	45
6.1	comparison of Time and completeness between existing methods and proposed methods for Speech/ text to sign language	46
6.2	comparison of precision between existing methods and proposed methods	46

List of Abbreviations

Abbreviations	**Full Form**
VQA	Visual Question Answering
SLT	Sign Language Translation
TTS	Text-to-Speech
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
ASR	Automatic Speech Recognition
ISL	Indian Sign Language
OCR	Optical Character Recognition
MS COCO	Microsoft Common Objects
BLEU	Bilingual Evaluation Understudy
API	Application Program Interface

1. Introduction

Individuals with disabilities often encounter significant hurdles when it comes to accessing information and communicating effectively, which can profoundly impact their daily lives. Traditional assistive technologies have provided some support, but there remains a gap in addressing the diverse needs of this population. This gap has sparked a growing interest in the development of multi-modal assistive systems, which leverage a combination of technologies to enhance accessibility and communication for individuals with disabilities. These systems aim to provide a more inclusive and empowering experience by integrating features such as Sign Language Translation, Visual Question Answering, and Image/Text to Speech capabilities.

Central to the design and development of these multi-modal assistive systems is a user-centered approach that prioritizes the input and feedback of individuals with disabilities. By actively involving end-users in the co-design and evaluation phases, these systems ensure that the final product is tailored to their specific requirements and preferences. This collaborative approach not only enhances the usability and effectiveness of the system but also fosters a sense of ownership and empowerment among users. By positioning individuals with disabilities as active participants in the design and development process, these systems aim to address their unique challenges more comprehensively.

This project aims to explore the concept of Multi-Modal Assistive Systems (MMAS) and their potential to revolutionize the way people with disabilities interact with the world around them. By integrating various modes of input and output, these systems have the capacity to bridge the gap between individuals with disabilities and their environment. Through a thorough examination of the current state of Multi-Modal Assistive Systems, their applications in different domains, and ongoing research and development efforts, this project seeks to contribute to the ongoing discourse on inclusive technology. Ultimately, by shedding light on the advancements and potential of these systems, we hope to pave the way for a more inclusive and accessible future for individuals with disabilities.

1.1. Motivation

This project is motivated by a strong commitment to inclusivity and equality for individuals with disabilities. Despite technological advancements, many still encounter significant barriers in accessing information and communication, hindering their full participation in society and limiting personal and professional growth.

Developing Multi-Modal Assistive Systems aims to directly confront these challenges, empowering individuals with disabilities to navigate the world with increased independence and confidence. These systems have the potential to revolutionize how people with disabilities interact with their environment, offering new opportunities for learning, socialization, and employment.

By actively involving end-users in the design process, we ensure these systems are tailored to their specific needs and capabilities. Ultimately, our goal is to contribute to a more inclusive society where individuals with disabilities can thrive, breaking down communication barriers and enhancing accessibility to create equal opportunities for all**.**

fig 1.1: Problems faced by people with disabilities

1.2. Problem definition

The current landscape of assistive technologies presents significant challenges due to its fragmented nature. Visuals Question Answering, Sign Language Translation, and Text-to-Speech/Speech-to-Text tools, though individually beneficial, lack a cohesive framework. This fragmentation results in users grappling with inefficiencies, increased complexity, and potential barriers to widespread adoption. Furthermore, the lack of seamless synchronization between these tools hinders their effectiveness in addressing the diverse needs of individuals with disabilities. Therefore, there is a critical need to develop an integrated assistive technology solution that unifies Visual Question Answering, Sign Language Translation, and Text-to-Speech/Speech-to-Text technologies. Such an approach aims to streamline accessibility, enhance user experience, and foster inclusivity for individuals with disabilities, thereby addressing the shortcomings of the existing assistive technology ecosystem.

1.3. Problem Illustration

Listening and speaking are natural abilities of humans. Unfortunately, there are many people who do not have these abilities and cannot easily communicate with others. The World Health Organization stated that approximately 70 million people in the world are deaf-mutes. A total of 360 million people is deaf, and 32 million of these individuals are children. Also, by 2050, it is expected that one in every four people will face some degree of hearing loss. One of the main problems that deaf-mutes suffer from is communicating with others. Reliance on lip-reading and using sign language interpreters are two solutions for communication between deaf-mutes and non-deaf-mutes. However, lip-reading, in some cases, is useless, especially when the parties of the dialogue need to wear masks. In addition, the usage of sign language interpreters is costly.

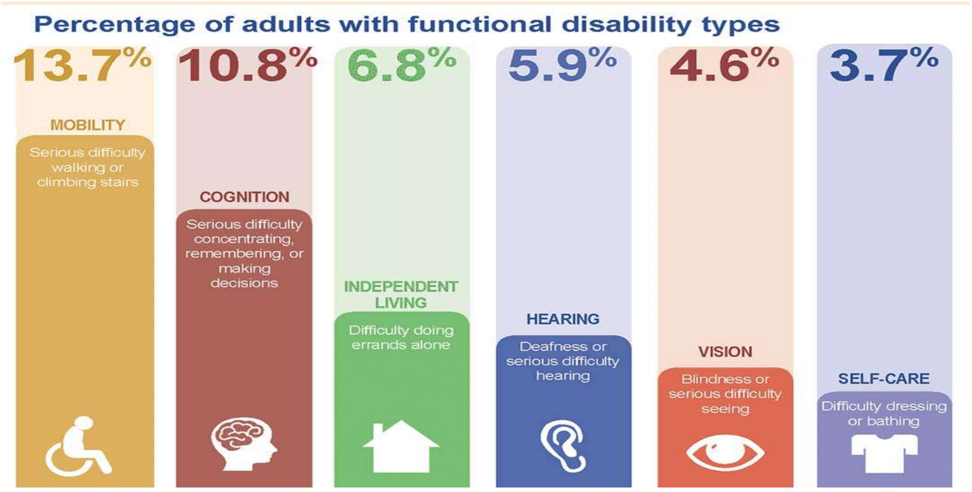
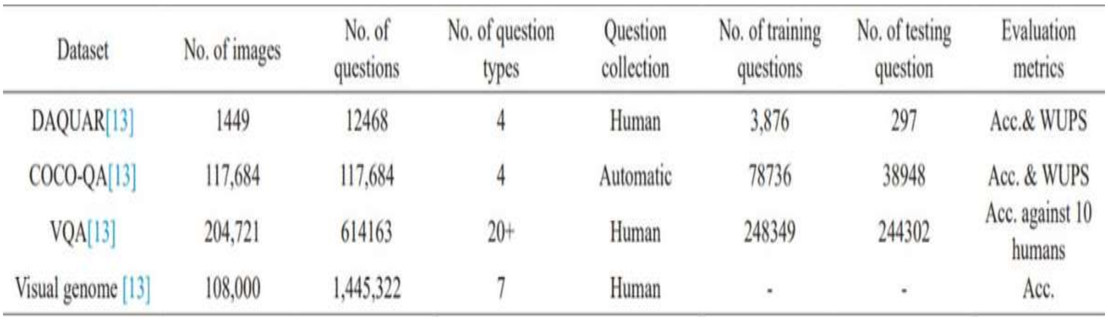


fig1.2: Percentage of adults with functional disability types

Visual Question Answering (VQA)

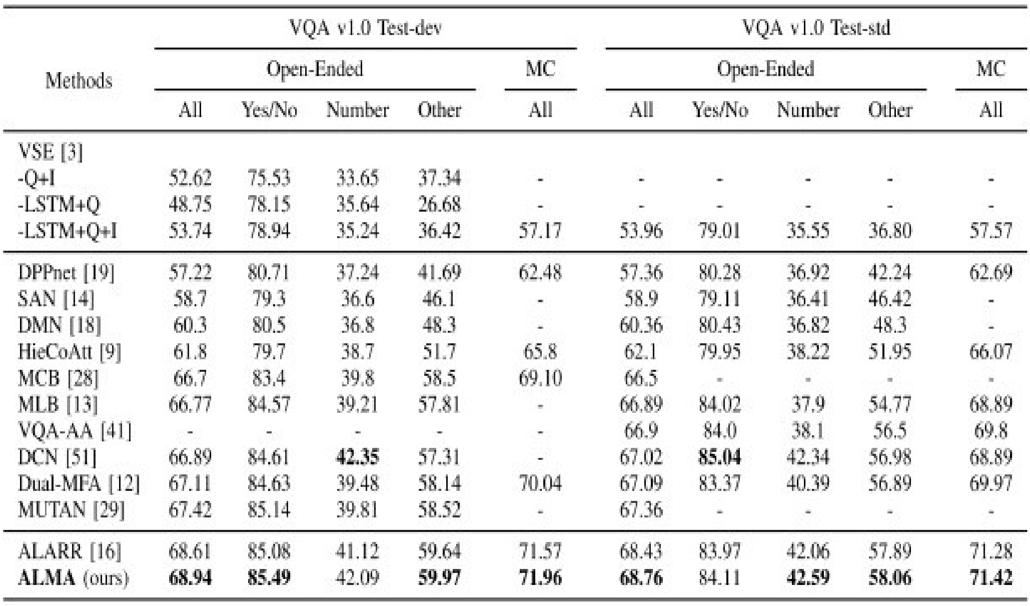
Visual Question Answering (VQA) is a cutting-edge field in Artificial Intelligence where machines are trained to comprehend and respond to questions posed about images. This interdisciplinary area combines Computer Vision and Natural Language Processing, aiming to develop systems capable of understanding both visual content and textual queries.

Visual Question Answering (VQA) would outline the diverse datasets and methodologies pivotal for implementation. Datasets such as VQA2.0, COCO-QA, and Visual Genome offer varied challenges for model training and evaluation, figure 1.3 illustrates the different datasets used for visual question answering task. Methodologies encompass both traditional approaches like Convolutional Neural Networks (CNNs) for image processing and Recurrent Neural Networks (RNNs) for text analysis, as well as advanced techniques like attention mechanisms and multimodal fusion. These methodologies aim to enable the model to understand the intricate relationship between images and corresponding textual questions, thereby facilitating accurate and contextualized responses in VQA systems, figure 1.4 illustrates the stats of different methodologies used for Visual Question Answering.



Dataset	No. of images	No. of questions	No. of question types	Question collection	No. of training questions	No. of testing question	Evaluation metrics
DAQUAR[13]	1449	12468	4	Human	3,876	297	Acc.& WUPS
COCO-QA[13]	117,684	117,684	4	Automatic	78736	38948	Acc. & WUPS
VQA[13]	204,721	614163	20+	Human	248349	244302	Acc. against 10 humans
Visual genome [13]	108,000	1,445,322	7	Human	-	-	Acc.

fig1.3: Datasets used for Visual Question Answering task



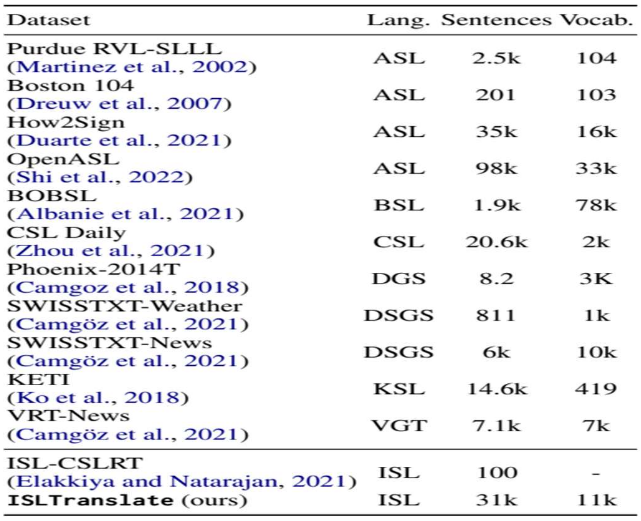
Methods	VQA v1.0 Test-dev					VQA v1.0 Test-std				
	Open-Ended				MC	Open-Ended				MC
	All	Yes/No	Number	Other	All	All	Yes/No	Number	Other	All
VSE [3]										
-Q+I	52.62	75.53	33.65	37.34	-	-	-	-	-	-
-LSTM+Q	48.75	78.15	35.64	26.68	-	-	-	-	-	-
-LSTM+Q+I	53.74	78.94	35.24	36.42	57.17	53.96	79.01	35.55	36.80	57.57
DPPnet [19]	57.22	80.71	37.24	41.69	62.48	57.36	80.28	36.92	42.24	62.69
SAN [14]	58.7	79.3	36.6	46.1	-	58.9	79.11	36.41	46.42	-
DMN [18]	60.3	80.5	36.8	48.3	-	60.36	80.43	36.82	48.3	-
HieCoAtt [9]	61.8	79.7	38.7	51.7	65.8	62.1	79.95	38.22	51.95	66.07
MCB [28]	66.7	83.4	39.8	58.5	69.10	66.5	-	-	-	-
MLB [13]	66.77	84.57	39.21	57.81	-	66.89	84.02	37.9	54.77	68.89
VQA-AA [41]	-	-	-	-	-	66.9	84.0	38.1	56.5	69.8
DCN [51]	66.89	84.61	**42.35**	57.31	-	67.02	**85.04**	42.34	56.98	68.89
Dual-MFA [12]	67.11	84.63	39.48	58.14	70.04	67.09	83.37	40.39	56.89	69.97
MUTAN [29]	67.42	85.14	39.81	58.52	-	67.36	-	-	-	-
ALARR [16]	68.61	85.08	41.12	59.64	71.57	68.43	83.97	42.06	57.89	71.28
ALMA (ours)	**68.94**	**85.49**	42.09	**59.97**	**71.96**	**68.76**	84.11	**42.59**	**58.06**	**71.42**

fig1.4: Stats of different methodologies used for Visual Question Answering task

Speech/Text to Sign Translation

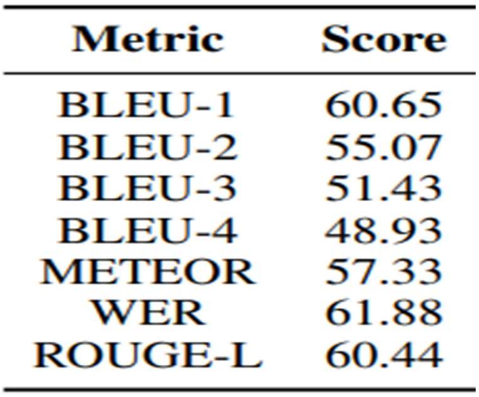
The initiative aims to develop a system for Speech/Text to Sign Translation, facilitating communication for the hearing impaired. Leveraging advanced Machine Learning techniques, it seeks to bridge the gap between spoken or written language and sign language, enabling seamless and accurate translation in real-time interactions.

Speech/Text to Sign Translation initiative outlines essential datasets and methodologies for implementation. figure 1.4. illustrates the different datasets like Purdue RVL-SLL, Boston 104, RWTH-PHOENIX-Weather 2014T, VRT-News and RWTH-PHOENIX-Weather 2014P provide annotated recordings of German sign language, while Sign Language MNIST offers a collection of hand gestures. Methodologies encompass Deep Learning techniques such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for feature extraction and sequence modelling, as well as attention mechanisms for capturing temporal dependencies. The fig 1.5 translation scores for a random sample of 291 pairs from ISL Translate aim to develop a robust system capable of accurately translating spoken or written language into sign language, enhancing communication accessibility for the deaf and hard of hearing.



Dataset	Lang.	Sentences	Vocab.
Purdue RVL-SLLL (Martinez et al., 2002)	ASL	2.5k	104
Boston 104 (Dreuw et al., 2007)	ASL	201	103
How2Sign (Duarte et al., 2021)	ASL	35k	16k
OpenASL (Shi et al., 2022)	ASL	98k	33k
BOBSL (Albanie et al., 2021)	BSL	1.9k	78k
CSL Daily (Zhou et al., 2021)	CSL	20.6k	2k
Phoenix-2014T (Camgoz et al., 2018)	DGS	8.2	3K
SWISSTXT-Weather (Camgöz et al., 2021)	DSGS	811	1k
SWISSTXT-News (Camgöz et al., 2021)	DSGS	6k	10k
KETI (Ko et al., 2018)	KSL	14.6k	419
VRT-News (Camgöz et al., 2021)	VGT	7.1k	7k
ISL-CSLRT (Elakkiya and Natarajan, 2021)	ISL	100	-
ISLTranslate (ours)	ISL	31k	11k

fig1.5: Comparison of continuous sign language translation datasets



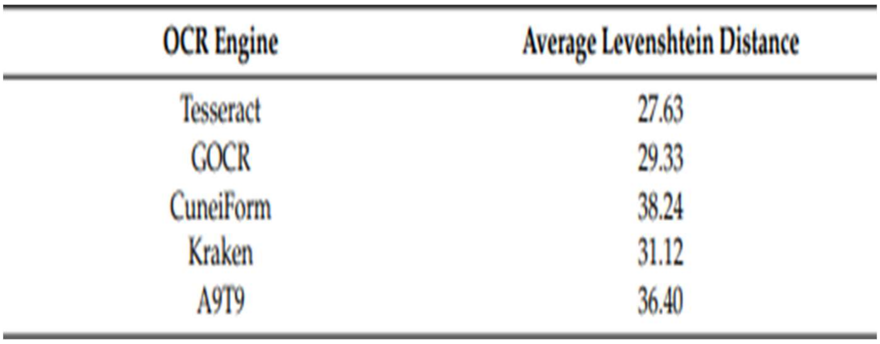
Metric	Score
BLEU-1	60.65
BLEU-2	55.07
BLEU-3	51.43
BLEU-4	48.93
METEOR	57.33
WER	61.88
ROUGE-L	60.44

fig1.6: translation scores for a random sample of 291 pairs from ISL Translate

Image/Text to Speech translation

The initiative focuses on developing a system for Image/Text to Speech translation, facilitating accessibility and communication for individuals with visual impairments. Leveraging advanced machine learning techniques, it aims to convert visual content or textual information into spoken language, enabling seamless and intuitive interaction with digital content.

The initiative of Image/Text to Speech translation fig 1.6 describes various OCR (Optical Character Recognition) models play a pivotal role in deciphering text within images. Models like Tesseract, Google Cloud Vision OCR, and Amazon Textract utilize Deep Learning techniques and trained datasets to accurately extract text from images, regardless of font, size, or orientation. These models employ advanced algorithms, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to interpret text regions and convert them into machine-readable formats. Through their collective capabilities, these OCR models enable efficient and precise extraction of text content from diverse image sources, facilitating tasks ranging from document digitization to text-based search and analysis



OCR Engine	Average Levenshtein Distance
Tesseract	27.63
GOCR	29.33
CuneiForm	38.24
Kraken	31.12
A9T9	36.40

fig 1.7: The results of the performance tests conducted for the OCR engines

1.4. Objective

Imagine a world without communication barriers. This project aspires to do just that by building bridges for those with visual, auditory, or reading difficulties. We achieve this through a technological marvel: a trilogy of tools. Visuals transform into text, sign language finds spoken voice, and text seamlessly flows to speech, and vice versa. This unlocks information, empowers expression, and fosters empathy between all communities. Together, we're weaving a richer world, where every voice becomes a vibrant thread in the magnificent tapestry of humanity.

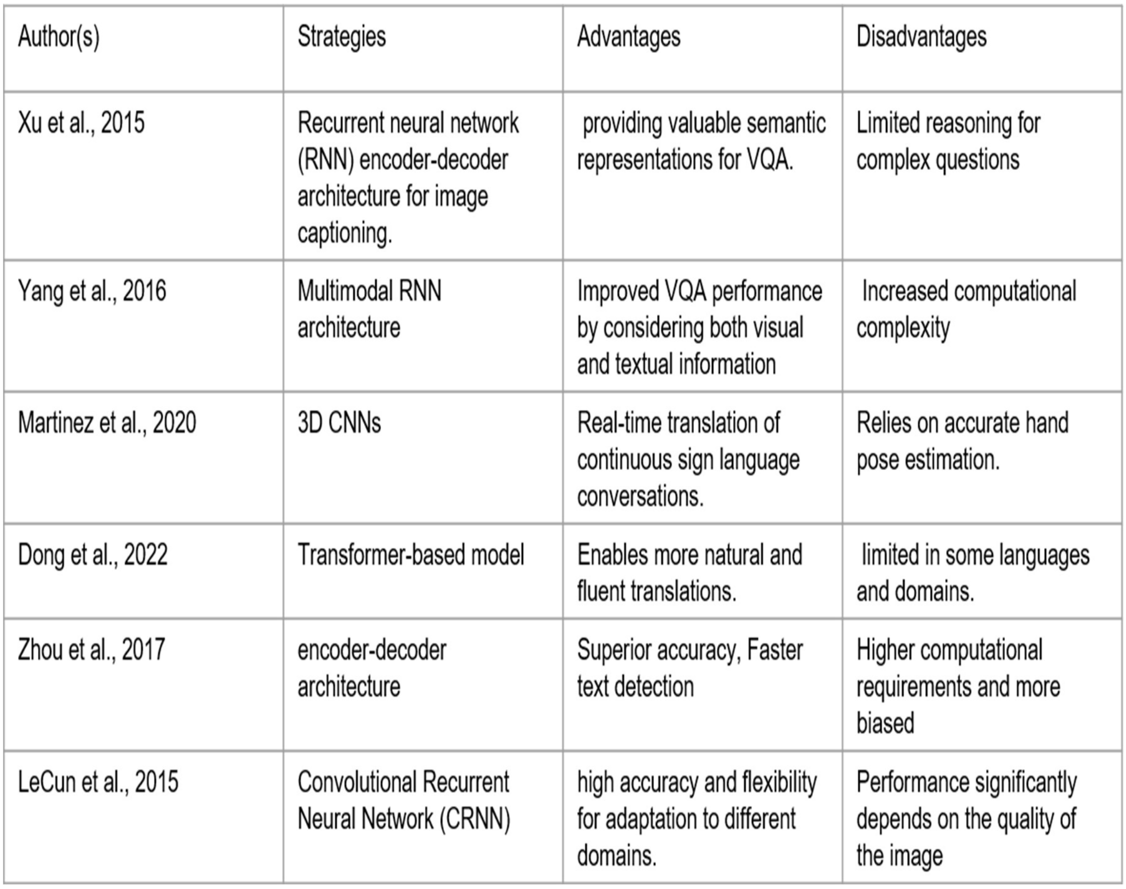
fig1.8: Objective of the project

2. Literature Survey

Visual Question Answering (VQA) has become a focal point in the intersection of computer vision and Natural Language Processing, especially within the context of Multi-Modal Assistive systems. These systems are designed to comprehend and respond to questions posed about visual content, typically images or videos, using natural language. The essence lies in enabling machines to understand and interpret visual information in a way that mirrors human comprehension. One prevalent approach in VQA involves leveraging deep learning architectures, such as Convolutional Neural Networks (CNN) for image processing and Recurrent Neural Networks or transformers for language understanding. These models are trained on large-scale annotated datasets, such as VQA v2, which contain pairs of images and corresponding questions with human-generated answers. By learning from such data, VQA systems can effectively associate visual features with textual information to generate accurate responses to questions. The integration of attention mechanisms further enhances the capability of these systems to focus on relevant regions of the image while processing questions. Recent advancements in VQA research have explored techniques such as fine- grained attention and co-attention mechanisms to improve performance and robustness. Moreover, efforts have been made to incorporate external knowledge sources, such as text- based corpora, to augment the understanding of complex questions. VQA holds immense promise in multi-modal assistive systems by enabling users to interact with their environment through natural language queries, thereby enhancing accessibility and usability for individuals with diverse needs.

Sign Language Translation (SLT) has emerged as a critical component of Multi-Modal Assistive Systems, aiming to bridge the communication gap between individuals who are deaf or hard of hearing and those who use spoken language. The core challenge lies in accurately recognizing and interpreting sign language gestures and subsequently translating them into textual or spoken language output. Deep learning techniques, particularly Convolutional Neural Networks and Recurrent Neural Networks, have played a pivotal role in advancing the accuracy and efficiency of SLT systems. These models are trained on annotated sign language datasets, which contain videos of sign gestures along with their corresponding textual translations. By leveraging large-scale datasets and sophisticated architectures, SLT systems can effectively recognize and translate a wide range of sign language expressions. However, challenges persist, including handling variations in sign language gestures across different individuals and regions, as well as ensuring real-time translation capabilities. Recent research efforts have explored novel approaches, such as transformer-based models, which have shown promising results in improving translation quality and efficiency. Additionally, there is ongoing work in integrating SLT with other modalities, such as text-based chat interfaces, to facilitate seamless communication between sign language users and non-signers. SLT holds immense promise in empowering individuals with hearing impairments by enabling them to communicate effectively and participate fully in various social and professional settings.

Speech-to-Text (STT) and Text-to-Speech (TTS) technologies serve as fundamental components of multi-modal assistive systems, facilitating seamless conversion between spoken language and text, thereby enabling communication and interaction for individuals with speech or hearing impairments. STT systems employ acoustic models, language models, and sequence-to-sequence architectures to accurately transcribe spoken language into text. These systems leverage deep learning techniques, such as recurrent neural networks (RNNs) and transformer models, to achieve high accuracy even in noisy environments. By analyzing audio signals and extracting linguistic features, STT systems can generate textual representations of spoken utterances, which can then be further processed or utilized for various applications. On the other hand, TTS systems utilize neural network-based architectures to synthesize natural-sounding speech from textual input. Techniques such as Wave Net have been instrumental in generating expressive and human-like speech waveforms. In the realm of multi-modal assistive systems, research in STT and TTS focuses on improving real-time performance, accommodating diverse accents and languages, and enhancing the naturalness and expressiveness of synthesized speech. Efforts are also directed towards seamlessly integrating STT and TTS functionalities into assistive devices and applications, thereby empowering individuals with speech or hearing impairments to communicate effectively and engage with their environment on their terms.



Author(s)	Strategies	Advantages	Disadvantages
Xu et al., 2015	Recurrent neural network (RNN) encoder-decoder architecture for image captioning.	providing valuable semantic representations for VQA.	Limited reasoning for complex questions
Yang et al., 2016	Multimodal RNN architecture	Improved VQA performance by considering both visual and textual information	Increased computational complexity
Martinez et al., 2020	3D CNNs	Real-time translation of continuous sign language conversations.	Relies on accurate hand pose estimation.
Dong et al., 2022	Transformer-based model	Enables more natural and fluent translations.	limited in some languages and domains.
Zhou et al., 2017	encoder-decoder architecture	Superior accuracy, Faster text detection	Higher computational requirements and more biased
LeCun et al., 2015	Convolutional Recurrent Neural Network (CRNN)	high accuracy and flexibility for adaptation to different domains.	Performance significantly depends on the quality of the image

Table 2.1 : Comparison of Existing Method from selected Strategies

3. Proposed Method

3.1. Visual Question and Answering:

3.1.1. Exploratory Data Analysis:

Our exploration of the Visual Question Answering (VQA) dataset reveals distinct characteristics between the visual data and the question-answer pairs.

Image Analysis:

The dataset leverages images from MS COCO, with varying sizes requiring pre-processing for uniformity. While sizes differ, all images share a consistent color depth of 3, indicating RGB format. This highlights the importance of image pre-processing to achieve a standardized format suitable for further analysis within the VQA model.

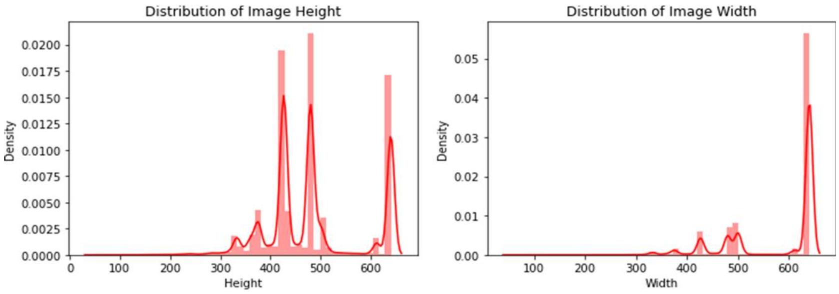


Fig3.1: Distribution of Image Height & Width

Question Analysis:

The dataset boasts a substantial number of questions (443,757) associated with the images. Interestingly, the questions themselves tend to be concise. Most questions fall within a 20–40-character range, with an average of 29 characters. Similarly, the word length distribution skews towards shorter questions, with a majority consisting of 4-8 words. While some questions reach a maximum of 22 words, a significant portion stays below 10 words. This suggests the task often focuses on identifying specific details within the image, requiring fewer words to formulate the question. Additionally, the dataset incorporates a rich vocabulary of over 13,332 unique words, allowing for diverse question phrasing.

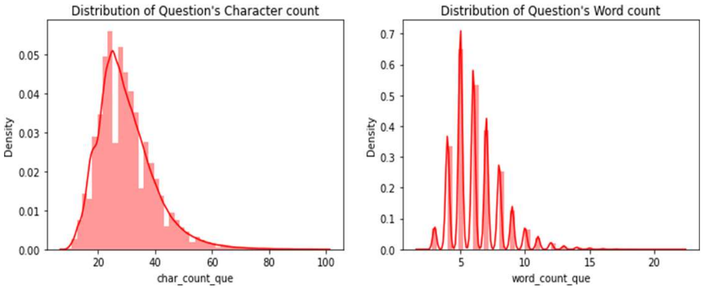


fig 3.2: Distribution of Question's Character count & Word count

Answer Analysis:

A striking difference emerges when analyzing the answer data. Unlike the questions, the answers overwhelmingly favor brevity. A staggering 99% of answers have three words or less, with a significant portion consisting of single words. This brevity extends to the character length, averaging only 4 characters with a maximum of 71. These findings suggest that the VQA task primarily deals with factual, single-step reasoning that can be addressed with concise answers

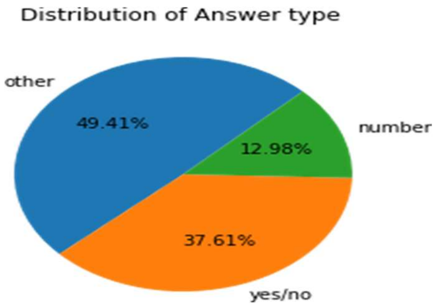


fig 3.3: Distribution of answer type

3.1.2. Data pipelining and generation

The data pipelining and generation process for training a Visual Question Answering (VQA) model involves meticulously managing both textual and image data to seamlessly integrate them into the model architecture. To begin with, textual data undergoes a series of preprocessing steps to ensure consistency and compatibility with the model. This includes lowercasing, decontraction to convert contractions into their natural form, removal of punctuation, and padding to standardize the length of questions. These steps are crucial for creating a uniform input format that the model can effectively process and interpret.

On the other hand, image data is prepared by resizing all images to a standardized dimension of 224x224 pixels and normalizing their pixel values to fall within the range of [0,1]. This resizing and normalization process optimizes the representation of images for the neural network, facilitating better feature extraction and interpretation. By standardizing the dimensions and pixel values of images, the model can effectively learn and extract relevant visual features across different images, enhancing its ability to comprehend visual content and generate accurate responses.

The data generation process involves the implementation of specialized classes, such as CustomDataGen and CustomDataGen_aug, to facilitate efficient batch-wise retrieval and augmentation of textual and image data during model training. These classes enable dynamic loading of data batches and incorporate random transformations, such as image augmentation techniques, to enhance model robustness and generalization. By applying random transformations to a fraction of images during training, the model learns to adapt to variations in visual content, improving its ability to handle diverse inputs and generate accurate responses. Overall, this comprehensive approach to data pipelining and generation lays a strong foundation for training a robust VQA model capable of performing effectively in real-world scenario.

3.1.3. Fine tuning and training the model

The process of fine-tuning and training the Visual Question Answering (VQA) model involves the design and configuration of a Deep Learning architecture tailored to the task's requirements. The model architecture begins by leveraging a pre-trained VGG19 Convolutional Neural Network (CNN) as the backbone for image feature extraction. This pre-trained model, initialized with weights from ImageNet, is configured to accept input images of size 224x224 pixels with three color channels. By setting include top=True, the model retains its fully connected layers, facilitating global pooling of features.

To adapt the pre-trained VGG19 model for the VQA task, its layers are made non-trainable, preserving the learned representations while preventing further modification. Regularization is applied to the kernel weights of each layer with a L2 regularization strength of 0.1, promoting model generalization and mitigating overfitting. This regularization technique helps control the complexity of the model and reduce the risk of learning noisy patterns from the training data.

The VGG19 architecture's output layer, labelled 'fc2,' serves as the starting point for integrating textual information. Textual input in the form of questions is processed using an embedding layer, initialized with pre-trained word embeddings to capture semantic relationships. Subsequent bidirectional Long Short-Term Memory (LSTM) layers are employed to extract sequential features from the embedded text data, enhancing the model's understanding of context and temporal dependencies. Dropout layers are strategically inserted to mitigate overfitting by randomly deactivating neurons during training, promoting better generalization.

The textual features extracted by the LSTM layers are combined with the image features using pointwise multiplication, facilitating the fusion of visual and textual information at a semantic level. This fusion mechanism enables the model to correlate image content with textual context, enhancing its ability to generate accurate responses to visual questions. Finally, the fused features are passed through a dense SoftMax layer to produce probability distributions over the vocabulary of potential answers. The model is compiled using the Adam optimizer with a learning rate of 0.0001 and trained to minimize categorical cross-entropy loss, with accuracy as the evaluation metric.

Overall, the architecture integrates deep CNNs and Recurrent Neural Networks (RNNs) to effectively process both visual and textual inputs, enabling the model to learn rich representations and perform robustly on the VQA task.

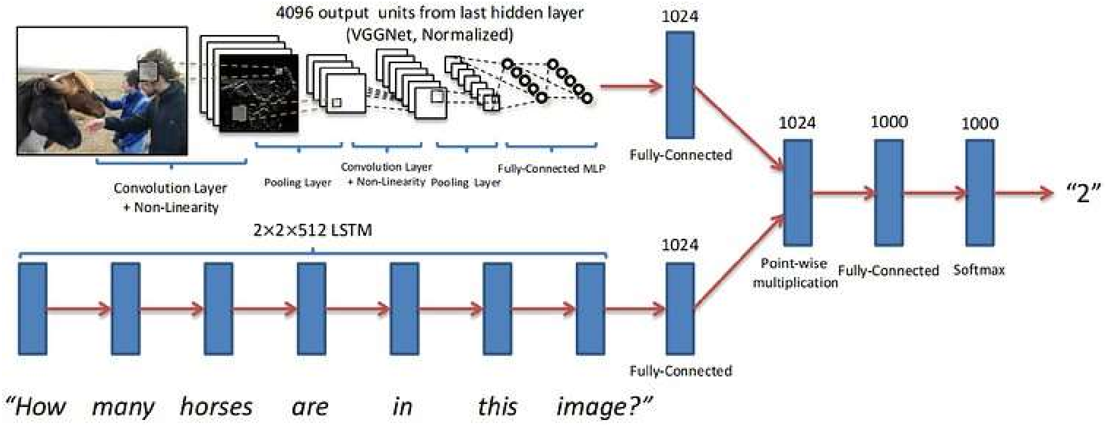


fig 3.4: Model Architecture for Visual question and answering

3.2. Speech/ Text to Sign language

3.2.1. Data Gathering:

The "Sign Data Gathering" phase in the Speech/Text to Sign Language Translation project involves collecting real-world sign language data at different levels: sentence-level, word-level, and character-level. This crucial step lays the foundation for developing an accurate and comprehensive translation system capable of converting spoken or written language into sign language effectively.

At the sentence-level, the focus is on capturing the natural expressions and gestures used by sign language speakers to convey complete thoughts or ideas. This entails recording individuals proficient in sign language as they communicate sentences or phrases across various contexts and topics. By gathering a diverse range of sentence-level sign language data, the translation system can learn to recognize and interpret the nuances of language expression, including grammar, syntax, and semantics.

Moving to the word-level, the objective is to acquire sign language representations for individual words commonly used in spoken or written language. This involves compiling a comprehensive vocabulary of sign language equivalents corresponding to each word in the target language(s). Sign language experts and native speakers contribute to this process by providing clear and standardized representations of words through video recordings or annotated datasets. Additionally, the data collection effort may involve crowd-sourcing techniques to ensure coverage of a wide range of vocabulary across different domains and dialects.

At the character-level, the focus shifts to capturing the intricate handshapes, movements, and facial expressions that constitute the building blocks of sign language communication. This involves gathering detailed annotations or recordings of individual signs, gestures, or phonemes that make up the Visual Language System. Each character-level data point provides insights into the fine- grained articulatory features and parameters essential for accurate sign language translation. Moreover, the data collection process may involve capturing variations in signing style, regional dialects, and cultural nuances to enhance the system's adaptability and inclusivity.

By gathering sign language data at different levels of granularity, the project aims to create a rich and diverse dataset that facilitates accurate translation from spoken or written language to sign language. The collected data serves as a valuable resource for training machine learning models, developing algorithms, and evaluating the performance of the translation system. Ultimately, the Sign Data Gathering phase plays a pivotal role in bridging the communication gap between individuals who use spoken or written language and those who rely on sign

3.2.2. Input Speech/ Text Pre-Processing

Several crucial steps are undertaken to prepare the input speech or text data for further processing and translation.

The process begins with setting up the microphone to capture the speech input effectively. This involves configuring the microphone settings, ensuring proper connectivity, and adjusting the recording parameters to optimize audio quality. By setting up the microphone correctly, the system can accurately capture spoken language inputs, laying the groundwork for accurate transcription and translation.

Ambient noise adjustment is performed to mitigate background noise interference that could affect the accuracy of speech recognition. This involves analyzing the ambient noise level and applying noise reduction techniques to enhance the clarity of the speech signal. By minimizing background noise, the system can improve the accuracy of speech recognition and subsequent text transcription.

Once the speech signal is captured and ambient noise is adjusted, the system proceeds to the text recognition phase, where the speech signal is converted into text format using Automatic Speech Recognition (ASR) technology. This involves analyzing the speech signal and transcribing it into textual form, enabling further processing and analysis.

Following text recognition, the pre-processed text undergoes additional pre- processing steps to enhance its compatibility with the translation model. This includes converting the text to lowercase to standardize the text format and remove variations in letter case. Additionally, punctuation removal is performed to eliminate non-alphanumeric characters from the text, ensuring that the input text is clean and consistent for subsequent processing.

By completing these pre-processing steps, the input speech or text data is refined and standardized, ready for further processing and translation into sign language. The pre-processing stage plays a crucial role in ensuring the accuracy and reliability of the translation process by optimizing the quality of the input data and removing potential sources of noise or inconsistencies. Ultimately, these efforts contribute to the development of a robust and effective speech/text to sign language translation system capable of bridging communication barriers between spoken or written language and sign language.

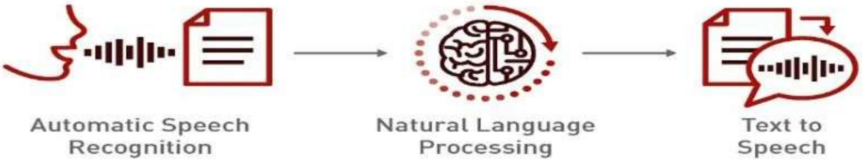


fig 3.5: Speech to text preprocessing

3.2.3. Translating Text to Sign Language Videos

In the " Translating Text to Sign Language Videos " phase of the Speech/Text to Sign Language Translation project, the primary objective is to translate textual inputs, comprising characters, words, or sentences, into corresponding Indian Sign Language (ISL) gestures. This process involves a meticulous mapping of each textual element to a predefined set of ISL gestures captured in video format.

Initially, the textual input, whether it be a single character, word, or sentence, undergoes segmentation into individual components for mapping. For instance, each character within a word or each word within a sentence is analyzed and associated with its corresponding ISL gesture. This segmentation enables precise mapping of each textual element to its respective sign language representation, ensuring accuracy in translation.

Subsequently, the mapped textual elements are matched with the pre- recorded ISL gesture videos obtained during the sign data gathering phase of the project. Each textual component is linked to the corresponding video depicting the ISL gesture representing that particular character, word, or sentence. This mapping process relies on a comprehensive database of ISL gesture videos meticulously curated during the sign data gathering stage.

Once the textual elements are successfully mapped to the ISL gesture videos, the next step involves combining these individual gesture videos to form a cohesive and coherent representation of the entire sign language sentence. This amalgamation process seamlessly stitches together the mapped videos in a sequential manner, ensuring fluid transitions between gestures and preserving the natural flow of sign language communication.

By accurately mapping textual inputs to ISL gesture videos and combining them to form complete sign language representations, the system effectively bridges the communication gap between spoken or written language and sign language. This mapping process is essential for facilitating effective communication for individuals with hearing impairments or those who primarily communicate using sign language, thereby promoting inclusivity and accessibility in communication channels. Ultimately, this phase of the project plays a pivotal role in enabling seamless speech/text to sign language translation, enhancing communication accessibility for diverse user group

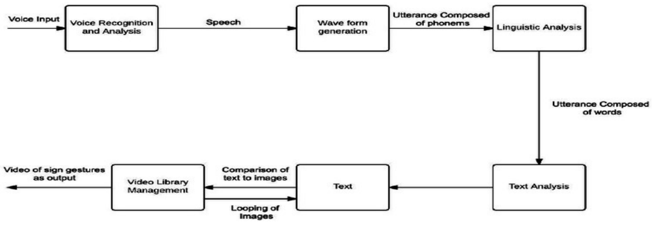

Voice Input
Voice Recognition and Analysis
Speech
Wave form generation
Utterance Composed of phonems
Linguistic Analysis
Utterance Composed of words
Text Analysis
Text
Comparison of text to images
Looping of Images
Video Library Management
Video of sign gestures as output

fig 3.6: Speech/Text to Sign Translation

3.3. Image/text to Speech:

3.3.1Setting up OCR Environment with Pytesseract

In this phase, the focus is on obtaining the necessary tools and configuring the environment for our project. Firstly, we need to acquire the OCR model compatible with pytesseract. There are several sources available for downloading OCR models, including pre-trained models provided by Tesseract or custom models trained on specific datasets. Depending on the requirements of our project, we'll choose the most suitable model and ensure it's compatible with pytesseract's version. We'll explore documentation and community forums to find the optimal model that meets our needs in terms of accuracy and language support.

Once we've obtained the OCR model, the next step is setting up the environment for the pytesseract module. This involves installing pytesseract along with its dependencies, which typically include Pillow for image processing and pyttsx3 or similar libraries for speech synthesis. We'll utilize package managers like pip to install these dependencies seamlessly. Additionally, we'll configure system environment variables to specify the path to the Tesseract executable, ensuring pytesseract can access the OCR engine efficiently.

Furthermore, we may need to install language packs and additional resources for pytesseract to support recognition in different languages and improve accuracy for specific character sets. This step is crucial for multilingual applications or projects dealing with non-standard text formats. By completing these setup procedures diligently, we ensure that our system is ready to perform optical character recognition effectively, paving the way for seamless integration of pytesseract into our project environment.

3.3.2. Image Preprocessing

Image preprocessing plays a vital role in enhancing the accuracy of text recognition within images. Before feeding images into the OCR system, it's essential to apply preprocessing techniques to optimize image quality. One of the primary preprocessing steps is noise reduction, which involves removing unwanted artifacts and imperfections from the image. This can be achieved through techniques such as Gaussian blur, median filtering, or morphological operations like erosion and dilation.

Another critical aspect of image preprocessing is resizing, where we adjust the dimensions of the image to a suitable scale for OCR processing. Resizing helps standardize the input image size, making it easier for the OCR engine to analyze and extract text accurately. Additionally, binarization is a common preprocessing technique that involves converting the image into a binary format by thresholding pixel values. This simplifies the image and improves contrast, making it easier to distinguish text from the background. Moreover, contrast adjustment is essential for enhancing the visibility of text within images. By adjusting the brightness and contrast levels, we can improve the readability of text and ensure better OCR results. Techniques such as histogram equalization or adaptive thresholding can be employed to enhance contrast effectively. Overall, image preprocessing aims to prepare images in a standardized format that maximizes the efficacy of the OCR process, ultimately leading to more accurate text recognition results.

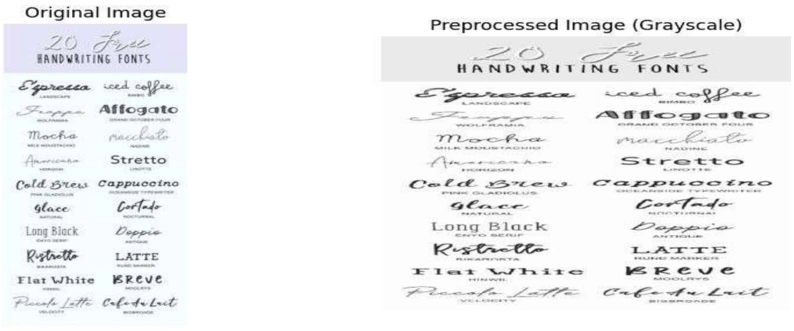


fig 3.7: Image preprocessing for text extraction

3.3.3. Speech Generating Function

The speech generating function is a crucial component of our project, enabling the conversion of text into audible speech. Leveraging pytesseract's text recognition capabilities, we can extract text from images or utilize text input provided by the user. The first step in implementing the speech generating function is selecting an appropriate text-to-speech (TTS) synthesis technique. There are various libraries and APIs available for TTS synthesis, each with its unique features and capabilities.

Once we've chosen a suitable TTS synthesis method, we'll integrate it with pytesseract to seamlessly convert recognized text into speech. This involves passing the extracted text to the TTS engine and generating corresponding audio output. We'll explore customization options such as voice selection, speech rate adjustment, and pronunciation control to enhance the naturalness and clarity of the generated speech.

Furthermore, error handling mechanisms will be implemented to ensure robustness and reliability of the speech generating function. This includes handling exceptions gracefully, providing informative error messages, and implementing

Fallback strategies in case of unexpected errors or failures. Additionally, we'll optimize performance by minimizing latency and resource consumption, allowing for real-time or near-real-time speech generation depending on the application requirements.

Overall, the speech generating function adds value to our project by providing users with a convenient and accessible way to consume textual content through audible speech. Whether it's converting text from images, documents, or user input, this functionality enhances the accessibility and usability of our application, making it more inclusive and user-friendly.

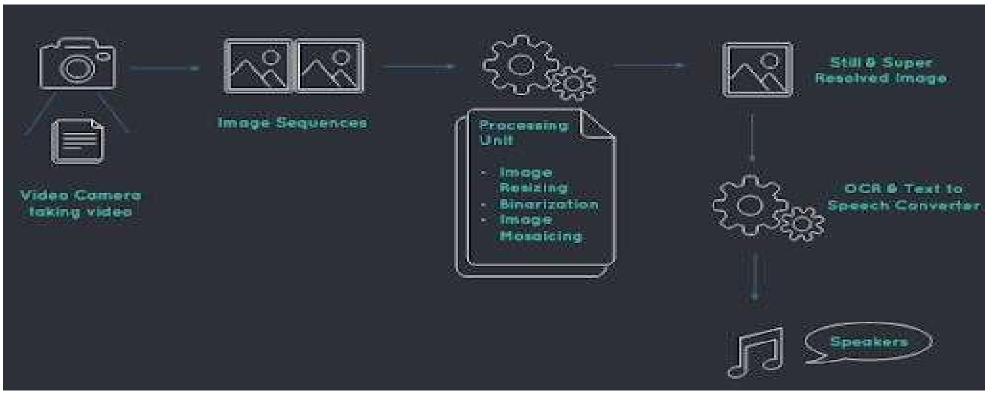


fig 3.8: Image/Text to Speech Translation

4. Implementation

4.1. Functionality:

4.1.1. Visual Question Answering (VQA):

Uploading image and questions:

Uploading image and questions marks the initial interaction point between users and the Visual Question Answering (VQA) system. This step involves users providing an image along with a corresponding question, serving as essential inputs for the VQA model. Users may input questions in natural language, such as "What color is the car?" or "How many people are in the image?" Simultaneously, they upload the corresponding image, which serves as the context for answering the question. This interaction facilitates the integration of both visual and textual information, enabling the system to comprehend the content of the image in conjunction with the query posed by the user. It sets the foundation for subsequent processing stages by ensuring that the VQA model receives the necessary inputs to generate accurate answers. The uploaded image and question initiate the VQA pipeline, guiding the system towards effectively analysing and interpreting the visual and textual components in tandem, ultimately providing meaningful responses to user queries.

Preprocessing the text:

In the pre-processing stage of textual data for a visual question answering (VQA), several steps are undertaken to ensure the text is in a suitable format for the model. Firstly, decontractions are resolved to maintain consistency in language representation. Next, the text is converted to lowercase to standardize input, facilitating uniformity in processing. Unnecessary characters or words are then removed, reducing noise and focusing on relevant information. Tokenization is subsequently applied, breaking down the text into individual tokens or words, thereby creating a representation that the model can effectively work with. Through these steps, the textual input undergoes refinement, ensuring coherence and consistency in format, which enhances the model's ability to understand and interpret the questions accurately. This pre-processing step is crucial as it lays the groundwork for subsequent stages of the VQA pipeline, enabling the model to process textual inputs efficiently and generate meaningful answers in conjunction with visual information from the images.

Preprocess the image:

In the preprocessing phase of images for a Visual Question Answering (VQA), two pivotal steps are resizing and converting to RGB format. Resizing involves adjusting the image dimensions to a standardized size, ensuring uniformity across all input images. This maintains the aspect ratio while adhering to the model's input specifications, promoting efficient processing. Converting the image to the RGB color space establishes consistent color representation, fostering compatibility across diverse image sources. The RGB format enables the model to interpret color information uniformly, bolstering its capacity to extract pertinent visual features. Through this preprocessing, the VQA system primes images for feature extraction and seamless integration with textual data during model inference. These preprocessing procedures are indispensable for optimizing the model's efficacy, empowering it to analyze visual content alongside textual inputs proficiently. Consequently, the model can provide accurate answers to user queries, thereby enhancing the overall VQA system performance.

Loading the model:

Loading the model is a crucial step in the implementation of a Visual Question Answering (VQA). At this stage, the pre-trained VQA model architecture is initialized and configured, setting the foundation for subsequent inference tasks. The model architecture is typically composed of two main components: one for processing visual inputs (such as a Convolutional Neural Network or CNN) and another for processing textual inputs (such as a Recurrent Neural Network or RNN). These components are integrated to form a unified framework capable of analyzing both images and text simultaneously. Additionally, if available, pre-trained weights may be loaded into the model, enabling it to leverage knowledge learned from large-scale datasets during training.

Loading the model involves instantiating its architecture and configuring parameters such as learning rates, optimizers, and loss functions. This process ensures that the model is initialized with the necessary resources and settings to efficiently process input data and generate accurate answers to user queries. Moreover, loading the model enables seamless integration with the preprocessing pipeline, allowing preprocessed image and text inputs to be fed into the model for inference.

Once the model is loaded, it is ready to process incoming image- question pairs and generate responses. By loading the model, the VQA system establishes the computational framework necessary for making predictions, facilitating efficient inference and enabling the system to provide meaningful answers to user queries based on the input image and textual question. Overall, loading the model is a critical step in the VQA pipeline, laying the groundwork for subsequent inference tasks and ensuring the system's readiness to handle user interactions effectively.

Make Predictions:

Making predictions is the culmination of the Visual Question Answering (VQA) implementation, where the model processes preprocessed image and textual inputs to generate accurate answers. Once the image and question are preprocessed, and the model is loaded with its architecture configured, predictions can be made. The model processes the preprocessed image and question through its architecture, extracting relevant features from both modalities and combining them to generate an answer. This integration of visual and textual information enables the model to understand the context of the question within the content of the image. Predictions are typically represented as probability distributions over possible answers, with the most probable answer selected based on the highest probability. By making predictions, the VQA system completes the inference process, providing a meaningful answer to the user's question based on the input image and textual query. This process is essential for delivering accurate responses and enhancing user interaction with the VQA system. Through continuous refinement and evaluation, the model's predictive capabilities can be further improved, ensuring robust performance across various visual and textual inputs. Overall, making predictions is the core functionality of the VQA system, enabling it to fulfil its purpose of answering questions based on visual content effectively.

4.1.2. Speech/ Text to Sign

Speech/ Text Input:

In the implementation of a Speech/Text to Sign, the Speech/Text input serves as the primary mode of communication between the user and the system. This input could be in the form of spoken language or written text. For speech input, users articulate their message verbally, which is then transcribed into text using Automatic Speech Recognition (ASR) technology. This transcription process converts the spoken words into a textual representation, enabling the system to process and analyze the linguistic content. On the other hand, for text input, users directly input their message through written text, eliminating the need for ASR conversion. Regardless of the input modality, the textual representation of the message forms the basis for subsequent processing stages in the system. The accuracy and clarity of the speech/text input are crucial factors that impact the system's ability to generate accurate sign language interpretations. Therefore, it is essential to employ robust ASR models and text processing techniques to ensure accurate transcription and interpretation of the user's message. By providing speech/text input, users initiate the communication process with the system, facilitating seamless interaction and enabling individuals with hearing impairments to effectively communicate using sign language.

Preprocessing the Text:

In the implementation of a Speech/Text to Sign, preprocessing the text plays a critical role in converting textual input into a format suitable for sign language interpretation. This preprocessing involves several essential steps aimed at enhancing the quality and interpretability of the textual content. Firstly, the text undergoes normalization, where inconsistencies in formatting, punctuation, and capitalization are standardized to ensure uniformity. Next, tokenization breaks down the text into individual words or tokens, facilitating easier processing and analysis by the system. Following tokenization, stop words removal eliminates common words such as "the," "and" or "is," which carry little semantic meaning and may introduce noise to the interpretation process. Additionally, stemming or lemmatization reduces words to their base forms, aiding in the recognition of common word variants. Furthermore, spell checking helps rectify typographical errors, ensuring accuracy in the textual representation. Lastly, encoding transforms the processed text into a numerical representation, enabling efficient input into the sign language generation model. Through these preprocessing steps, the textual input is refined and standardized, laying the groundwork for accurate interpretation and generation of sign language gestures. By optimizing the quality and consistency of the textual content, preprocessing facilitates seamless communication between users and the sign language interpretation system, empowering individuals with hearing impairments to access and convey information effectively.

Mapping the Input Text to gathered Videos:

In the implementation of a Speech/Text to Sign, mapping the input text to gathered videos involves a meticulous process aimed at converting textual input into coherent sign language interpretations. This mapping entails associating each character or word from the input text with pre-existing sign language videos or sequences. Firstly, the system identifies the individual characters or words in the text and retrieves corresponding sign language representations from a database or dataset. This database contains a collection of sign language videos depicting various gestures, movements, and expressions associated with different linguistic elements. Next, the system combines these individual sign language videos in a sequential manner to form a cohesive interpretation of the entire sentence or phrase. This combination involves blending transitions between consecutive sign language gestures to ensure fluidity and naturalness in the sign language output. Additionally, the system may incorporate facial expressions, body movements, and other non-manual markers to convey the nuances of the spoken or written message accurately. By mapping each character or word to gathered sign language data and integrating the videos into a cohesive sequence, the system generates a comprehensive and expressive interpretation of the input text in sign language. This process enables individuals with hearing impairments to access and understand spoken or written content effectively through sign language interpretation.

Final Video:

In the implementation of a Speech/Text to Sign, the final video is the culmination of the sign language interpretation process, representing the synthesized output of the system's conversion from speech/text input to sign language gestures. This final video seamlessly integrates the mapped sign language sequences derived from the input text, effectively conveying its intended meaning in sign language. The video composition ensures coherence and natural flow by blending transitions between individual sign language gestures and incorporating appropriate facial expressions, body movements, and non-manual markers to accurately convey the nuances of the spoken or written message. Additionally, the final video may incorporate graphical overlays or text annotations to provide supplementary information or context, further enhancing comprehension for users.

Throughout the generation of the final video, the system prioritizes accuracy, expressiveness, and accessibility to ensure that individuals with hearing impairments can effectively understand and engage with the interpreted content. By producing a comprehensive and expressive sign language representation of the input text, the final video empowers users to access and interact with spoken or written information effectively, bridging communication barriers and fostering inclusivity in diverse linguistic contexts.

4.1.2. Image/PDF to Speech

Uploading the Image/ PDF**:**

In the implementation of an Image/PDF to Speech, uploading the Image/PDF marks the initial interaction points between the user and the system, serving as the gateway for inputting visual content for conversion into speech. Users initiate this process by selecting and uploading an image or PDF file containing the desired content to be converted. This step is pivotal as it provides the necessary input for the system to commence the conversion process. The uploaded image or PDF is then processed by the system, which extracts the textual content using optical character recognition (OCR) technology in the case of images or directly parses the text from the PDF file. Once the textual content is extracted, it serves as the foundation for the subsequent speech synthesis phase. The clarity and quality of the uploaded image or PDF directly impact the accuracy and comprehensibility of the synthesized speech output. Therefore, users are encouraged to upload high-resolution images or well-formatted PDF files to ensure optimal performance. Overall, uploading the image or PDF file initiates the conversion process, facilitating the transformation of visual content into speech and enabling users to access information in an auditory format.

Extracting text from Image**:**

In the implementation of an Image/PDF to Speech, extracting text from images is a crucial initial step aimed at converting visual content into a textual format suitable for speech synthesis. This process involves leveraging Optical Character Recognition (OCR) technology to identify and extract text embedded within images or PDF documents. OCR algorithms analyze the visual content pixel by pixel, recognizing patterns and shapes that correspond to letters, numbers, and symbols. As the OCR algorithm identifies text elements, it reconstructs them into a coherent textual representation, preserving the original layout and formatting as much as possible. Additionally, preprocessing techniques such as noise reduction, binarization, and edge detection may be employed to enhance the accuracy of text extraction, particularly in cases of complex backgrounds or low image quality. Once the text is extracted, it is ready for further processing and analysis, such as language identification, text normalization, and semantic analysis. This extracted text serves as the foundation for subsequent stages of the project, enabling the synthesis of natural-sounding speech that accurately reflects the content of the original images or documents. Overall, extracting text from images forms a critical component of the Image/PDF to Speech project, facilitating the conversion of visual information into accessible and actionable textual content.

Creating the Audio File:

Creating the audio file in an Image/PDF to speech project involves a multi-step process aimed at transforming extracted text into natural- sounding speech. Initially, the extracted text undergoes preprocessing to ensure formatting consistency and linguistic correctness. This step is crucial for enhancing the overall quality and coherence of the spoken output. Subsequently, text-to-speech (TTS) technology is employed to convert the processed text into audible speech. TTS systems utilize sophisticated algorithms and linguistic models to generate human-like speech patterns, intonations, and accents, mimicking the nuances of natural language. Parameters such as speech rate, pitch, and volume are adjusted during the synthesis process to customize the audio output according to user preferences. Advanced TTS systems may leverage neural network architectures to further enhance speech synthesis accuracy and naturalness. Additionally, techniques such as prosody modelling are employed to infuse appropriate emotional cues and emphasis into the speech output, contributing to its expressiveness and comprehensibility. Once the audio file is generated, it can be saved in various formats for playback on different devices and platforms, providing users with convenient access to the converted textual content through spoken audio representations. Overall, the creation of the audio file is a crucial component of the Image/PDF to speech project, facilitating seamless accessibility and comprehension of textual information through spoken audio.

4.2. Attributes

Labelencoder:

In Visual Question Answering (VQA), LabelEncoder plays a crucial role in transforming text labels (like animal names) into numerical values for processing by machine learning models. This simplifies computations. More importantly, LabelEncoder allows for the inverse transformation, where a numerical prediction from the model can be converted back to the original class label. This lets us interpret the model's output in terms of the actual answer categories, making it easier to understand the VQA system's reasoning.

Tokenizer:

Within a Visual Question Answering (VQA) project, the tokenizer acts as a bridge between textual questions and the numerical world that machine learning models understand. It takes a cleaned question as input and performs text-to- sequence operations. This involves breaking the sentence down into its essential units (words, characters, or other predefined units) and converting them into a sequence of numerical identifiers. This allows the model to process the question's meaning in a structured way, ultimately facilitating the task of accurately answering questions based on visual content

Model:

The model attribute, typically stored as an .h5 file, is the heart of a VQA system. It encapsulates the weights learned by a trained deep learning model. Given an image and a question, the model doesn't directly process them. Instead, it relies on the weights to analyze the image features and the question's embedded meaning. These weights, Hasil from extensive training data, allow the model to make accurate predictions about the answer, essentially forming the knowledge base used for solving VQA tasks.

Input_image_VQA:

The input_image_VQA attribute represents the input image provided to a VQA model. This image is vital for the model's comprehension of the scene or object being described in the question. It typically undergoes pre-processing (resizing, normalization) before feeding into the model. The model analyses the extracted visual features from this image to understand the visual context of the question. Along with the question itself, the processed image plays a crucial role in enabling the model to accurately answer questions about the content of the image.

Input_question_VQA:

The input_question_VQA attribute represents the textual question posed to a VQA model. This question acts as a crucial guide, directing the model's attention towards relevant parts of the input image. In essence, the input_question_VQA attribute plays a vital role in bridging the gap between the visual world (image) and the textual world (question), enabling VQA models to effectively answer questions about the content of an image.

Predicted_ans:

The predicted_ans attribute embodies the answer generated by the VQA model in response to a given image and question. This answer represents the model's best attempt to provide a solution based on its analysis of the image and its understanding of the question's intent.

Sign_input_text:

The "Sign_input_text" attribute in the speech/text to sign translation project serves as the primary input mechanism for the translation system. It encapsulates the textual information provided directly by users or obtained through speech recognition. This inputs text forms the basis for generating corresponding sign language representations, enabling seamless communication between individuals using spoken or written language and those fluent in sign language.

Image_address:

The "Image_address" attribute in the speech/text to sign translation project signifies the location or address of an image or video within the dataset. This address serves as a reference point for accessing individual frames or segments of sign language data. By looping through and combining all the images or videos referenced by these addresses, the system generates a comprehensive final sign language video.

Input_image_OCR:

The "input_image_OCR" attribute in the image to speech translation project refers to the input image processed by an Optical Character Recognition (OCR) engine to extract textual content. This attribute enables the system to convert images containing text into machine-readable format, facilitating subsequent speech synthesis.

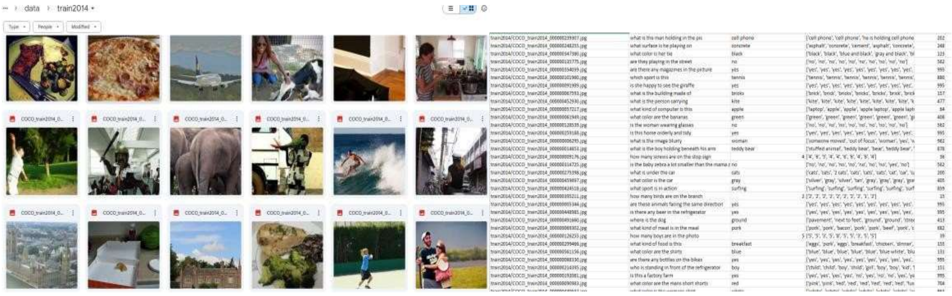
Extracted_text:

The "extracted_text" attribute in the image to speech translation project represents the text content extracted from images using an Optical Character Recognition (OCR) engine. This extracted text serves as the input for the subsequent step of converting text to speech. By accurately capturing and processing text from images, the OCR engine enables the system to transcribe visual information into a format that can be audibly communicated.

4.3. Datasets:

MS COCO (For Visual Question and answering):

The MS COCO dataset stands as a cornerstone for research in Visual Question Answering (VQA). It provides a rich resource for training and evaluating models that can answer open-ended questions about images. This dataset encompasses a massive collection of real-world images (over 328,000) along with corresponding questions (millions) posed about the content within them. Each question is paired with multiple answer options, allowing for comprehensive evaluation of VQA models' ability to reason about visual information and translate it into accurate answers. The MS COCO dataset's diverse nature, encompassing various objects, scenes, and interactions, makes it a valuable tool for pushing the boundaries of VQA research.



train2014/COCO_train2014_000000239307.jpg	what is this man holding in the pic	cell phone	['cell phone', 'cell phone', 'he is holding cell phone	202
train2014/COCO_train2014_000000248255.jpg	what surface is he playing on	concrete	['asphalt', 'concrete', 'cement', 'asphalt', 'concrete',	348
train2014/COCO_train2014_000000347380.jpg	what color is her tie	black	['black', 'black', 'blue and black', 'gray and black', 'bl	123
train2014/COCO_train2014_000000135775.jpg	are they playing in the street	no	['no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no']	582
train2014/COCO_train2014_000000354099.jpg	are there any magazines in the picture	yes	['yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',	995
train2014/COCO_train2014_000000101980.jpg	which sport is this	tennis	['tennis', 'tennis', 'tennis', 'tennis', 'tennis', 'tennis',	880
train2014/COCO_train2014_000000091989.jpg	is she happy to see the giraffe	yes	['yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',	995
train2014/COCO_train2014_000000067593.jpg	what is the building made of	bricks	['brick', 'brick', 'bricks', 'bricks', 'bricks', 'brick', 'brick	157
train2014/COCO_train2014_000000452930.jpg	what is the person carrying	kite	['kite', 'kite', 'kite', 'kite', 'kite', 'kite', 'kite', 'kite', 'k	477
train2014/COCO_train2014_000000057217.jpg	what kind of computer is this	apple	['laptop', 'apple', 'apple', 'apple laptop', 'apple lapt	84
train2014/COCO_train2014_000000061949.jpg	what color are the bananas	green	['green', 'green', 'green', 'green', 'green', 'green', 'gr	408
train2014/COCO_train2014_000000128535.jpg	is the woman wearing glasses	no	['no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no']	582
train2014/COCO_train2014_000000259188.jpg	is this home orderly and tidy	yes	['yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',	995
train2014/COCO_train2014_000000006295.jpg	what is the image blurry	woman	['someone moved', 'out of focus', 'woman', 'yes', 'w	982
train2014/COCO_train2014_000000014853.jpg	what is the boy holding beneath his arm	teddy bear	['stuffed animal', 'teddy bear', 'bear', 'teddy bear', '	878
train2014/COCO_train2014_000000009176.jpg	how many screws are on the stop sign	4	['4', '6', '3', '4', '4', '6', '6', '4', '6', '4']	36
train2014/COCO_train2014_000000114725.jpg	is the baby zebra a lot smaller than the mama z	no	['no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'yes', 'no']	582
train2014/COCO_train2014_000000275398.jpg	what is under the car	cats	['cats', 'cats', '2 cats', 'cats', 'cats', 'cats', 'cat', 'cat', 'ca	200
train2014/COCO_train2014_000000459897.jpg	what color is the car	gray	['silver', 'gray', 'silver', 'tan', 'gray', 'gray', 'gray', 'gray	405
train2014/COCO_train2014_000000424518.jpg	what sport is in action	surfing	['surfing', 'surfing', 'surfing', 'surfing', 'surfing', 'surf	859
train2014/COCO_train2014_000000395211.jpg	how many birds are on the branch	2	['2', '2', '2', '2', '2', '2', '2', '2', '1', '2']	15
train2014/COCO_train2014_000000005344.jpg	are these animals facing the same direction	yes	['yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',	995
train2014/COCO_train2014_000000448985.jpg	is there any beer in the refrigerator	yes	['yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',	995
train2014/COCO_train2014_000000491660.jpg	where is the dog	ground	['pavement', 'next to feet', 'ground', 'ground', 'stree	413
train2014/COCO_train2014_000000068302.jpg	what kind of meat is in the meal	pork	['pork', 'pork', 'bacon', 'pork', 'pork', 'beef', 'pork', 'c	682
train2014/COCO_train2014_000000126255.jpg	how many boys are in the photo	5	['5', '5', '2', '5', '8', '5', '5', '2', '5', '5']	19
train2014/COCO_train2014_000000299466.jpg	what kind of food is this	breakfast	['eggs', 'pork', 'eggs', 'breakfast', 'chicken', 'dinner',	155
train2014/COCO_train2014_000000561156.jpg	what color are the shirts	blue	['blue', 'blue', 'blue', 'blue', 'blue', 'blue white', 'blu	131
train2014/COCO_train2014_000000088330.jpg	are there any bottles on the bikes	yes	['yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',	995
train2014/COCO_train2014_000000214395.jpg	who is standing in front of the refrigerator	boy	['child', 'child', 'boy', 'child', 'girl', 'boy', 'boy', 'kid', 'l	131
train2014/COCO_train2014_000000192081.jpg	is this a factory farm	yes	['yes', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'no', 'yes', 'ye	995
train2014/COCO_train2014_000000090983.jpg	what color are the mans short shorts	red	['pink', 'pink', 'red', 'red', 'red', 'red', 'red', 'red', 'fus	704

fig 4.1: MS Coco dataset

Indian Sign Language (ISL) data (for speech to sign language translation):

The ISL dataset caters specifically to text-to-sign language translation research. This dataset offers a rich collection of text-sign language pairs, focusing on Indian Sign Language (ISL). It functions as a cornerstone for researchers developing and evaluating machine translation systems for this domain. The dataset's strength lies in its comprehensive nature, encompassing a significant number of sentence or phrase correspondences, bridging the gap between written English and ISL expressions.

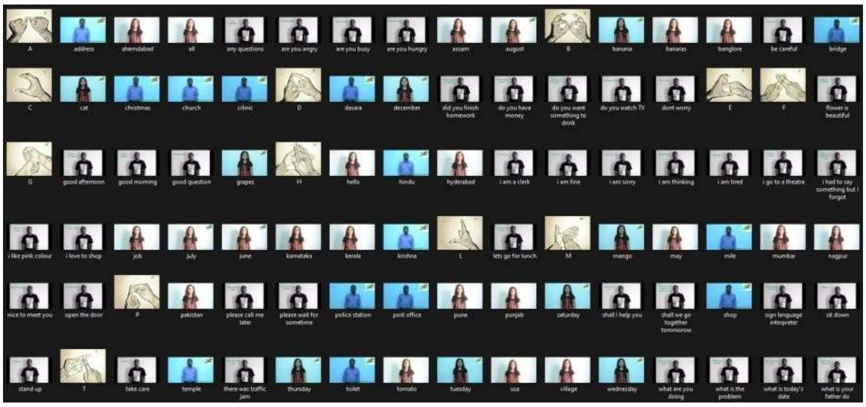
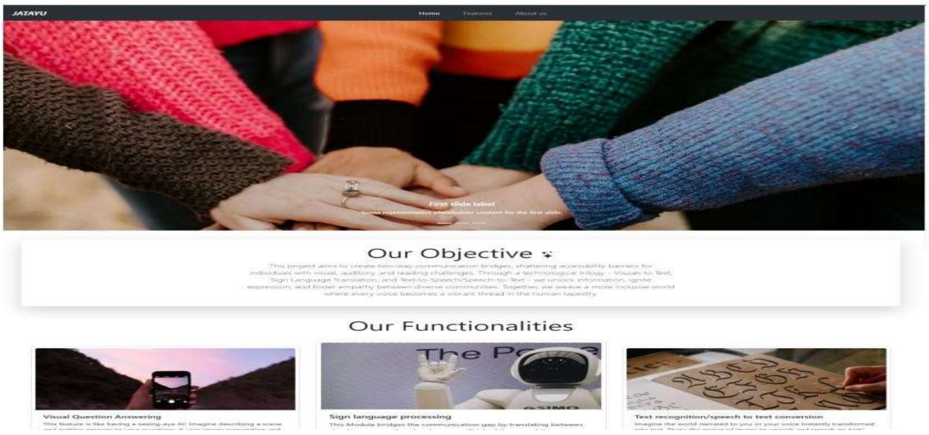


fig 4.2: ISL dataset

4.4. Experimental Screenshots:



First slide label

Some representative placeholder content for the first slide.

Our Objective

This project aims to create two-way communication bridges, shattering accessibility barriers for individuals with visual, auditory, and reading challenges. Through a technological trilogy - Visuals to Text, Sign Language Translation, and Text-to-Speech/Speech-to-Text - we unlock information, ignite expression, and foster empathy between diverse communities. Together, we weave a more inclusive world where every voice becomes a vibrant thread in the human tapestry.

Our Functionalities

Visual Question Answering

This feature is like having a seeing-eye AI! Imagine describing a scene

Sign language processing

This Module bridges the communication gap by translating between

Text recognition/speech to text conversion

Imagine the world narrated to you or your voice instantly transformed

fig 4.3: A web interface

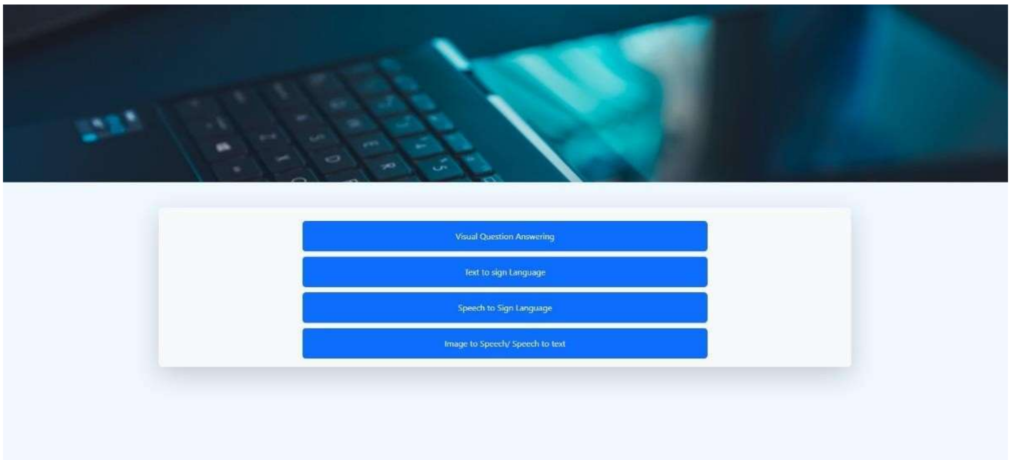
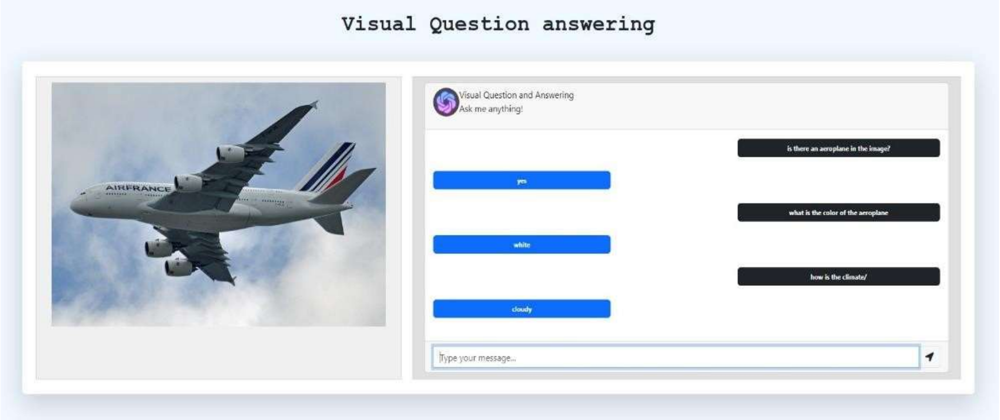


fig 4.4: A web interface integrated with multiple features



is there an aeroplane in the image?

yes

what is the color of the aeroplane

white

how is the climate/

cloudy

fig 4.5: Visual question and answering

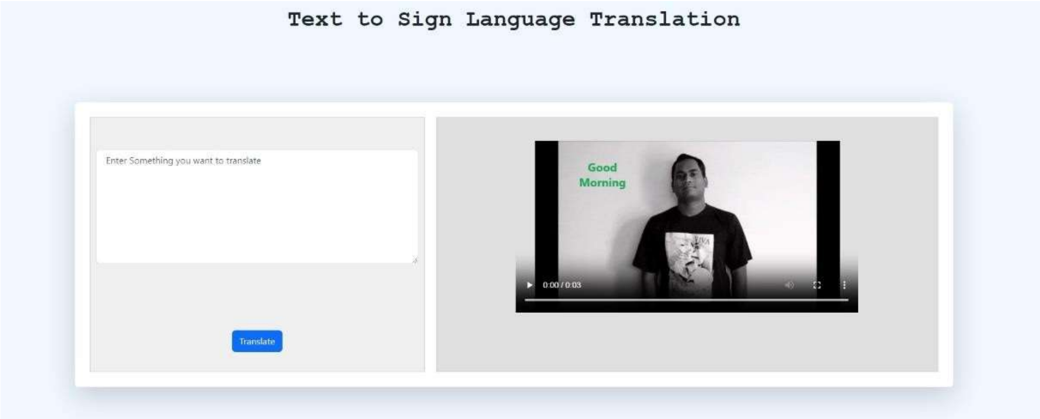
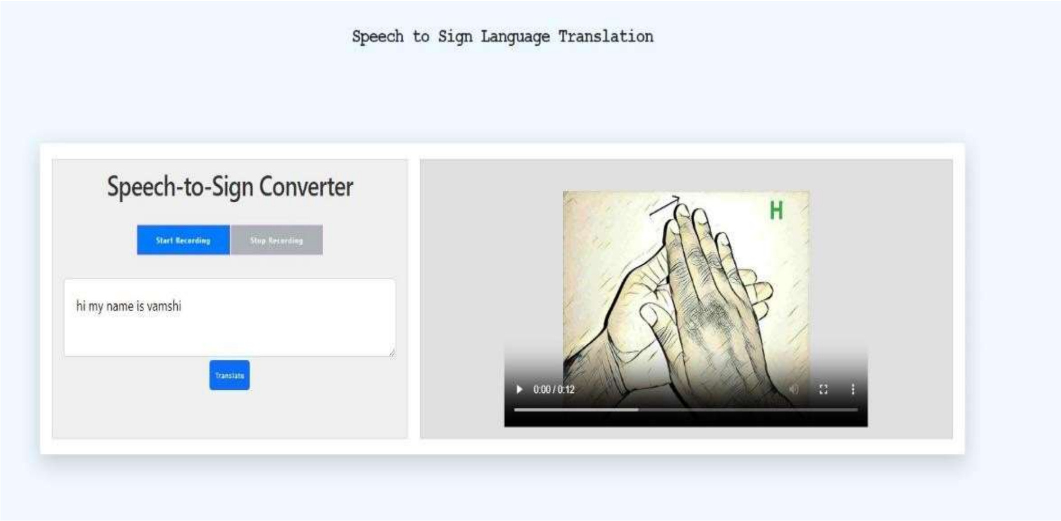

Text to Sign Language Translation
Enter Something you want to translate
Translate
Good Morning
0:00 / 0:03

fig 4.6: Text to sign language translator

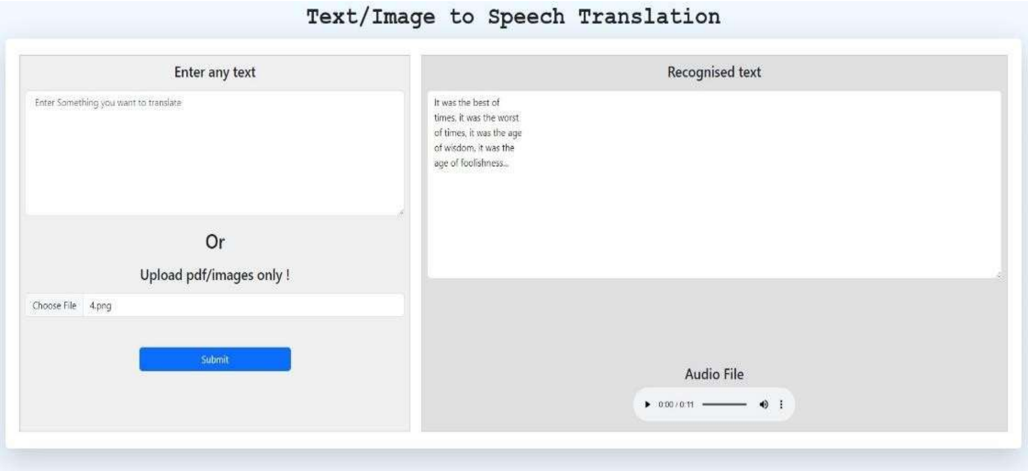


Speech to Sign Language Translation

Speech-to-Sign Converter

hi my name is vamshi

fig 4.7: Speech to sign language translator



Text/Image to Speech Translation

Enter any text

Enter Something you want to translate

Or

Upload pdf/images only !

Choose File 4.png

Recognised text

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness...

Audio File

fig 4.8: Image to speech translator

5. Experimental Results

5.1. Experimental Setup

5.1.1. Setup Jupyter Notebook:

To install and set up Jupyter Notebook, you can follow these steps:

1. Install Python: First, you need to have Python installed on your system. You can download and install Python from the official Python website: https://www.python.org/. Make sure to check the option to add Python to your system PATH during installation.

2. Install Jupyter Notebook: Once Python is installed, you can install Jupyter Notebook using pip, which is the Python package manager. Open a terminal or command prompt and run the following command: **pip install jupyter**

3. Launch Jupyter Notebook: After the installation is complete, you can launch Jupyter Notebook by running the following command in your terminal or command prompt: **jupyter notebook**

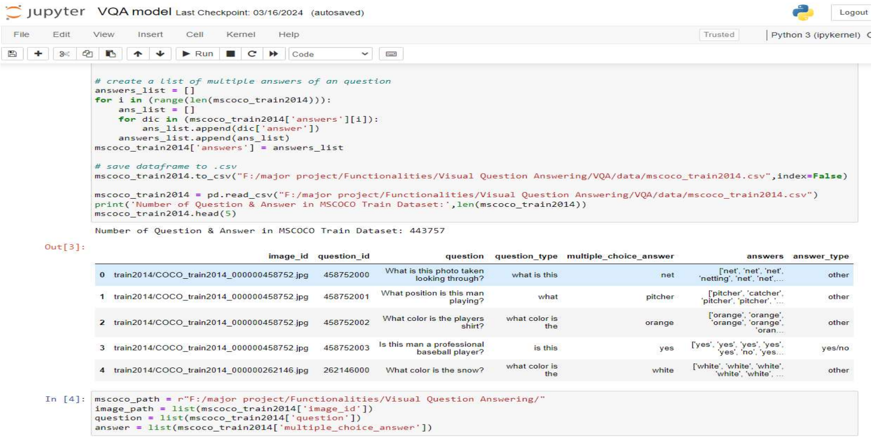
4. Accessing Jupyter Notebook: Once you run the command, your default web browser should open, and you'll be directed to the Jupyter Notebook dashboard. If it doesn't open automatically, you can manually open your web browser and go to http://localhost:8888/. Here, you'll see a file browser where you can navigate your filesystem and create or open Jupyter Notebook files.

5. Creating a New Notebook: To create a new notebook, click on the "New" button in the top right corner and select "Python 3" (or any other available kernel you want to use).

6. Using Jupyter Notebook: You can now start using Jupyter Notebook. Each notebook consists of cells where you can write and execute Python code, Markdown for documentation, and more. You can execute a cell by pressing

Shift+Enter or by clicking the "Run" button in the toolbar.

7. Saving and Closing: Make sure to save your work regularly by clicking the "Save" button or using the keyboard shortcut Ctrl+S. To close Jupyter Notebook, you can simply close the browser tab or stop the Jupyter Notebook server by pressing Ctrl+C in the terminal or command prompt where it's running.


jupyter VQA model Last Checkpoint: 03/16/2024 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3 (ipykernel)
Run
Code

```
# create a list of multiple answers of an question
answers_list = []
for i in (range(len(mscoco_train2014))):
    ans_list = []
    for dic in (mscoco_train2014['answers'][i]):
        ans_list.append(dic['answer'])
    answers_list.append(ans_list)
mscoco_train2014['answers'] = answers_list

# save dataframe to .csv
mscoco_train2014.to_csv("F:/major project/Functionalities/Visual Question Answering/VQA/data/mscoco_train2014.csv",index=False)

mscoco_train2014 = pd.read_csv("F:/major project/Functionalities/Visual Question Answering/VQA/data/mscoco_train2014.csv")
print('Number of Question & Answer in MSCOCO Train Dataset:',len(mscoco_train2014))
mscoco_train2014.head(5)
```

```
Number of Question & Answer in MSCOCO Train Dataset: 443757
```

Out[3]:

	image_id	question_id	question	question_type	multiple_choice_answer	answers	answer_type
0	train2014/COCO_train2014_000000458752.jpg	458752000	What is this photo taken looking through?	what is this	net	['net', 'net', 'net', 'netting', 'net', 'net',...	other
1	train2014/COCO_train2014_000000458752.jpg	458752001	What position is this man playing?	what	pitcher	['pitcher', 'catcher', 'pitcher', 'pitcher', '...	other
2	train2014/COCO_train2014_000000458752.jpg	458752002	What color is the players shirt?	what color is the	orange	['orange', 'orange', 'orange', 'orange', 'oran...	other
3	train2014/COCO_train2014_000000458752.jpg	458752003	Is this man a professional baseball player?	is this	yes	['yes', 'yes', 'yes', 'yes', 'yes', 'no', 'yes...	yes/no
4	train2014/COCO_train2014_000000262146.jpg	262146000	What color is the snow?	what color is the	white	['white', 'white', 'white', 'white', 'white', ...	other

In [4]:

```
mscoco_path = r"F:/major project/Functionalities/Visual Question Answering/"
image_path = list(mscoco_train2014['image_id'])
question = list(mscoco_train2014['question'])
answer = list(mscoco_train2014['multiple_choice_answer'])
```

Fig 5.1: Jupyter Notebook

5.1.2. Setting Up Visual Studio Code

Here's a guide to installing and configuring Visual Studio Code (VS Code) for a smooth development experience:

1. Download and Install: Head to the official VS Code download page: https://code.visualstudio.com/download Choose the installer suitable for your operating system (Windows, Mac, or Linux). Run the downloaded installer and follow the on-screen instructions.

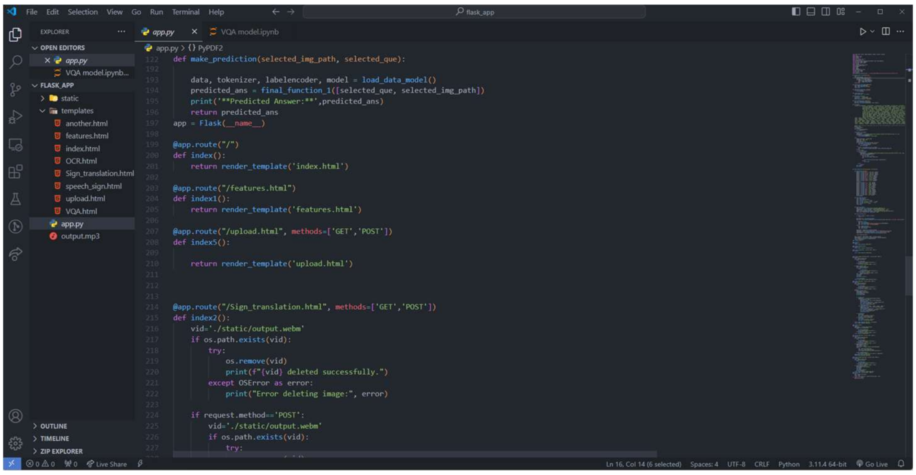
2. Open VS Code: Once installed, locate and launch VS Code from your applications list.

3. Install Extensions (Optional): VS Code is powerful with extensions. Explore the Extensions marketplace within VS Code and install language-specific extensions for syntax highlighting, code completion, and debugging support relevant to your

project needs.

4. Open Your Project: Navigate to your project folder using the File Explorer within VS Code (File > Open Folder).

You're now ready to start coding! Use the built-in features like syntax highlighting, code completion, and debugging to write and test your code efficiently.



```
def make_prediction(selected_img_path, selected_que):

    data, tokenizer, labelencoder, model = load_data_model()
    predicted_ans = final_function_1([selected_que, selected_img_path])
    print('**Predicted Answer:**',predicted_ans)
    return predicted_ans
app = Flask(__name__)

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

@app.route("/features.html")
def index1():
    return render_template('features.html')

@app.route("/upload.html", methods=['GET','POST'])
def index5():

    return render_template('upload.html')



@app.route("/Sign_translation.html", methods=['GET','POST'])
def index2():
    vid='./static/output.webm'
    if os.path.exists(vid):
        try:
            os.remove(vid)
            print(f"{vid} deleted successfully.")
        except OSError as error:
            print("Error deleting image:", error)

    if request.method=='POST':
        vid='./static/output.webm'
        if os.path.exists(vid):
            try:
```

Fig 5.2: Visual Studio Code

5.1.3. Setup Flask

To install and set up Flask, a popular Python library for creating interactive web applications, follow these steps:

1. Install Python: Ensure you have Python installed on your system. You can download and install Python from the official Python website: https://www.python.org/ . Make sure to check the option to add Python to your system PATH during installation.

2. Install Flask: You can install Flask using pip, the Python package manager. Open a terminal or command prompt and run the following command: **pip install Flask**

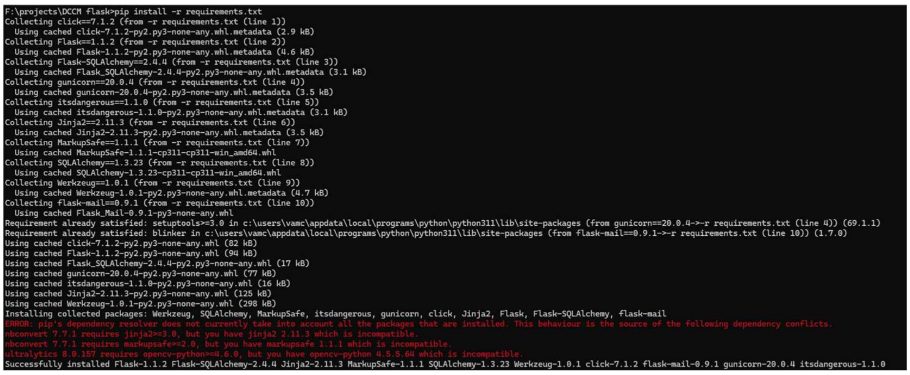
3. Create a Flask Application: Once Flask is installed, you can create a new Python script to define your Flask application.

4. Run Flask Application: In your terminal or command prompt, navigate to the directory containing `app.py` and run the following command **Python app.py**

5. Access Your Flask App: Once the Flask server starts, it will provide a local URL (usually http://127.0.0.1:5000/) where you can access your Flask application in your web browser.

6. Develop Your Application: You can continue to develop your Flask application by modifying `app.py`. Flask provides numerous components and features for building interactive data-driven applications, including widgets, charts, layouts, and more. Refer to the Flask documentation for detailed information: https://flask.palletsprojects.com/en/3.0.x/

7. Deploy Your Application: Once you're satisfied with your Flask application, you can deploy it to various platforms, including Flask Sharing, Heroku, AWS, or any other hosting provider.



```
F:\projects\DCCM flask>pip install -r requirements.txt
Collecting click==7.1.2 (from -r requirements.txt (line 1))
  Using cached click-7.1.2-py2.py3-none-any.whl.metadata (2.9 kB)
Collecting Flask==1.1.2 (from -r requirements.txt (line 2))
  Using cached Flask-1.1.2-py2.py3-none-any.whl.metadata (4.6 kB)
Collecting Flask-SQLAlchemy==2.4.4 (from -r requirements.txt (line 3))
  Using cached Flask_SQLAlchemy-2.4.4-py2.py3-none-any.whl.metadata (3.1 kB)
Collecting gunicorn==20.0.4 (from -r requirements.txt (line 4))
  Using cached gunicorn-20.0.4-py2.py3-none-any.whl.metadata (3.5 kB)
Collecting itsdangerous==1.1.0 (from -r requirements.txt (line 5))
  Using cached itsdangerous-1.1.0-py2.py3-none-any.whl.metadata (3.1 kB)
Collecting Jinja2==2.11.3 (from -r requirements.txt (line 6))
  Using cached Jinja2-2.11.3-py2.py3-none-any.whl.metadata (3.5 kB)
Collecting MarkupSafe==1.1.1 (from -r requirements.txt (line 7))
  Using cached MarkupSafe-1.1.1-cp311-cp311-win_amd64.whl
Collecting SQLAlchemy==1.3.23 (from -r requirements.txt (line 8))
  Using cached SQLAlchemy-1.3.23-cp311-cp311-win_amd64.whl
Collecting Werkzeug==1.0.1 (from -r requirements.txt (line 9))
  Using cached Werkzeug-1.0.1-py2.py3-none-any.whl.metadata (4.7 kB)
Collecting flask-mail==0.9.1 (from -r requirements.txt (line 10))
  Using cached Flask_Mail-0.9.1-py3-none-any.whl
Requirement already satisfied: setuptools>=3.0 in c:\users\vamc\appdata\local\programs\python\python311\lib\site-packages (from gunicorn==20.0.4->-r requirements.txt (line 4)) (69.1.1)
Requirement already satisfied: blinker in c:\users\vamc\appdata\local\programs\python\python311\lib\site-packages (from flask-mail==0.9.1->-r requirements.txt (line 10)) (1.7.0)
Using cached click-7.1.2-py2.py3-none-any.whl (82 kB)
Using cached Flask-1.1.2-py2.py3-none-any.whl (94 kB)
Using cached Flask_SQLAlchemy-2.4.4-py2.py3-none-any.whl (17 kB)
Using cached gunicorn-20.0.4-py2.py3-none-any.whl (77 kB)
Using cached itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Using cached Jinja2-2.11.3-py2.py3-none-any.whl (125 kB)
Using cached Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB)
Installing collected packages: Werkzeug, SQLAlchemy, MarkupSafe, itsdangerous, gunicorn, click, Jinja2, Flask, Flask-SQLAlchemy, flask-mail
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
nbconvert 7.7.1 requires jinja2>=3.0, but you have jinja2 2.11.3 which is incompatible.
nbconvert 7.7.1 requires markupsafe>=2.0, but you have markupsafe 1.1.1 which is incompatible.
ultralytics 8.0.157 requires opencv-python>=4.6.0, but you have opencv-python 4.5.5.64 which is incompatible.
Successfully installed Flask-1.1.2 Flask-SQLAlchemy-2.4.4 Jinja2-2.11.3 MarkupSafe-1.1.1 SQLAlchemy-1.3.23 Werkzeug-1.0.1 click-7.1.2 flask-mail-0.9.1 gunicorn-20.0.4 itsdangerous-1.1.0
```

Fig 5.3: Installing the Flask and its dependencies

5.2. Libraries Used:

TensorFlow:

TensorFlow is an open-source library from Google for building and training deep learning models. It uses data in multidimensional arrays and creates a computational graph to visualize the model's flow. TensorFlow offers high-level APIs to simplify complex deep learning tasks, making it a powerful tool for developers.

Pandas:

Pandas, a Python library, excels at data manipulation for analysis. It offers structures like DataFrames (similar to spreadsheets) to organize data. Pandas provides functions for cleaning, sorting, and filtering data, making it a go-to tool for data wrangling before diving into analysis or machine learning.

Pillow:

Pillow, a friendly fork of PIL, is a Python library for image processing. It lets you open various image formats, edit them (resize, crop, rotate), and save them in different formats. This makes Pillow handy for tasks like resizing photos, adding watermarks, or converting images to different formats.

MoviePy:

MoviePy is a Python library for script-based video editing. It allows you to easily combine video clips with cuts and concatenations in just a few lines of code. You can also use it for basic video processing tasks like changing the file extension. MoviePy works by writing commands to manipulate the video, making it accessible for automating video editing workflows.

SpeechRecognition:

The Web Speech API's SpeechRecognition object lets websites capture user speech through the microphone. You can configure it for continuous listening or single phrases. This enables features like voice dictation in forms or voice commands for website controls.

PyPDF2:

PDF2 is a free Python library for manipulating PDFs. It lets you access and read content by opening the PDF and extracting text using methods like. extractText (). You can also merge, split, crop pages, and even encrypt PDFs.

Pytesseract:

Pytesseract is a Python library that acts as a wrapper for Tesseract, a powerful open-source OCR (Optical Character Recognition) engine. It lets you extract text from images like receipts, scanned documents, or screenshots. Pytesseract can handle various image formats and integrates with Python's imaging libraries for pre-processing.

gTTS:

gTTS, short for Google Text-to-Speech, is a Python library that converts text into audio files (MP3). It acts as an interface for Google's Text-to-Speech API. You can use gTTS to create audio files in various languages with a simple command. It even handles splitting long sentences for natural-sounding speech.

Flask:

Flask is a Python web framework that simplifies building websites. Unlike bulkier options, Flask is lightweight and lets you choose the tools you need. You can define routes for handling user requests and use templates to create dynamic content. Flask's flexibility makes it popular for small websites and complex web applications alike.

Numpy:

NumPy, a Python library for numerical computing, excels at manipulating arrays. Images are essentially grids of pixels, which can be represented as NumPy arrays. This allows NumPy to perform basic image processing tasks like rotations, resizing, and grayscale conversion efficiently. While powerful for foundational tasks, consider Scikit- image for more advanced image processing applications.

5.3. Parameters:

BLEU Score:

BLEU (BiLingual Evaluation Understudy) is a metric for automatically evaluating the answer generated by VQA model. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high-quality reference translations. A value of 0 means that the machine-translated output has no overlap with the reference translation (low quality) while a value of 1 means there is perfect overlap with the reference translations (high quality).

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \times \log(p_n)\right)$$

Time:

The time parameter can estimate the generation time for a sign language video. It considers the video length for each word/character in the dataset and the number of words/characters in your input. It multiplies the average word/character video length by the total number of words/characters, providing an approximate generation time.

$$T = \sum_{i=1}^{n}(t_{wi} \times f_{wi}) + t_c$$

$$C = \frac{N_{mapped}}{N_{total}} \times 100\%$$

Precision:

precision is a metric used to assess how accurate an OCR model is at recognizing text. It focuses on the proportion of characters the model recognizes correctly; out of all the characters it identifies. Precision ranges from 0.0 (worst) to 1.0 (perfect).

$$AP = \sum_{k=0}^{k=n-1} [r(k) - r(k+1)] * p(k)$$

Parameter	Previous methods	Proposed method
Accuracy (VQA)	In the previously used method, the accuracy is low.	The accuracy is increased as we used transformers with customized learning rate and regularization.
Accuracy (image to speech)	In the previously used method, the accuracy is low.	The accuracy is increased as we used Pre-processing the Input Image combination of pytesseract with deep learning for text recognition.
Time	In the previously used method, the generation time of video is low.	The generation time of video is fast as we load the videos from the existence.
Completeness	In the previously used method, for some sentences they could not generate the complete sign representation.	Our method can completely generate any sign video for any sentence as we generate video by combining the individual letters or words of the sign representations

Table 5.1: Parameter comparison table between existing methods and proposed method

6. Discussion of Results

The Multi-Modal Assistive System showcases promising advancements across various domains. In the Text/Speech to Sign component, the system demonstrates effective translation of textual input into sign language, enhancing communication accessibility for the hearing impaired. Through Visual Question and Answering, the system efficiently processes visual queries, providing accurate responses and enabling seamless interaction with visual content. Additionally, the Image to Speech functionality showcases remarkable capabilities in converting images into spoken words, facilitating access to visual information for individuals with visual impairments. Overall, the project's results highlight the significant strides made in harnessing multi-modal technologies to empower individuals with diverse needs, bridging communication and accessibility gaps across different modalities. These achievements underscore the potential of such systems to revolutionize assistive technologies and enhance inclusivity in various settings, ranging from education and healthcare to everyday communication scenarios.

6.1. Visual question and answering:

	Existing Method	**Proposed Method**
BLEU Score	**30.0**	**33.4**

Table 6.1: comparison of BLEU score for VQA between existing methods and proposed method

Graph:

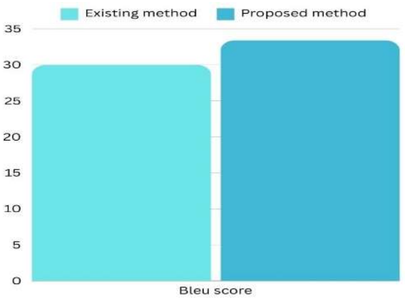


Figure 6.1. BLEU Score Comparison of VQA model

The proposed method has relative improvement of 8%

6.2. Text/Speech to Sign:

	Existing Method	Proposed Method
Completeness	80%	100%
Time	2 sec	1 sec

Table 6.2: comparison of Time and completeness between existing methods and proposed method for Speech/ text to sign language

Graph:

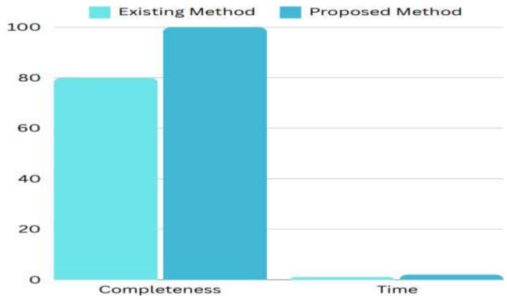


Figure 6.2. Completeness and time comparison

The proposed method is both completeness and time efficient compared to existing system.

6.3. Image to Speech:

	Existing Method	Proposed Method
Accuracy	75%	80%

Table 6.3: comparison of precision between existing methods and proposed method

Graph:

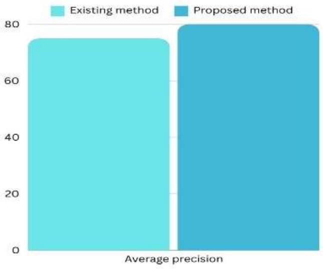


Figure 6.3. Precision Comparison of OCR model

The proposed method accuracy is increased as we used Pre-processing the Input Image and combination of pytesseract with deep learning for text recognition.

7. Summary, Conclusion and Recommendation

The development of a multimodal assistive system is a significant advancement in enhancing accessibility for individuals with diverse communication needs. By integrating Text/Speech to Sign functionality, the project bridges the communication gap between users of sign language and those who rely on spoken or written language, promoting seamless interaction and understanding. This feature empowers deaf or hard-of-hearing individuals to participate more fully in various social and professional contexts, fostering equality and integration. Moreover, the integration of Visual Question and Answering capabilities enhances the utility of the system by providing prompt and accurate responses to inquiries through visual input. This functionality streamlines information retrieval processes and supports independent learning and decision-making for users with visual impairments or those who benefit from visual cues, promoting autonomy and accessibility in education, work, and daily life. Additionally, the project's holistic approach underscores its commitment to addressing diverse communication needs comprehensively. By combining Text/Speech to Sign, Visual Question Answering, and Image to Speech functionalities, the project offers a multifaceted solution that accommodates various modes of communication and information processing. This integrative approach enhances accessibility, promotes social inclusion, and encourages equal participation across different domains of life. In conclusion, the multimodal assistive system project represents a pivotal advancement in fostering accessibility, inclusivity, and autonomy for individuals with diverse communication needs. Leveraging innovative technologies to support different communication modalities, the project empowers users to navigate and engage with the world more independently and effectively, holding immense promise in promoting equity and enhancing quality of life for individuals of all abilities.

8. Future Enhancements

Future enhancements for the Multi-Modal Assistive System for people with disabilities can significantly improve its performance and inclusivity. For the Visual Question Answering module, expanding the dataset to include the entire available data can enhance prediction accuracy. Additionally, resizing images to 448x448 and experimenting with different pre-trained CNN models for feature extraction could further improve model performance. Increasing the number of class labels (K) to 2000 or 3000 may also enhance the model's capability to answer a broader range of questions accurately. In the Speech to Sign Language Translation module, incorporating animated avatars to represent sign language can provide a more engaging and intuitive experience for users. Furthermore, extending the system to support multiple sign languages instead of just the Indian sign language can make the system accessible to a more diverse user base.For the Text Recognition module using Pytesseract, integrating it with a custom deep learning model can enhance efficiency and accuracy in recognizing text from images. This approach can leverage the strengths of both Pytesseract and deep learning to achieve better results, especially when dealing with complex or varied text format. Overall, these enhancements aim to make the multi-modal assistive system more robust, inclusive, and effective in meeting the diverse needs of user’s disabilities

9. References / bibliography

[1]. P. Anderson et al., “Bottom-up and top-down attention for image captioning and visual question answering,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, p. 6.

[2]. M. Malinowski and M. Fritz, “A multi-world approach to question answering about real-world scenes based on uncertain input,” in Proc. Neural Inf. Process. Syst. Conf., 2014, pp. 1682–1690.

[3]. M. Ren, R. Kiros, and R. S. Zemel, “Exploring models and data for image question answering,” in Proc. Neural Inf. Process. Syst. Conf., 2015, pp. 2953–2961.

[4]. M. Malinowski, M. Rohrbach, and M. Fritz, “Ask your neurons: A neural-based approach to answering questions about images,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp.1-9

[5]. yun liu , xiaoming zhang , feiran huang "adversarial learning with multi-modal attentionfor visual question answering"ieee transactions on neural networks and learning systems, vol. 32, no. 9, september 2021

[6]. Pallavi K S,Sonali M, Tanuritha , Vidhya "Visual Question Answering using Deep Learning" 2022 IJRTI | Volume 7, Issue 7 | ISSN: 2456-3315

[7]. K. P. Moholkar1, Ajay Pisharody2, Noorul Hasan Sayyed3, Rakesh Samanta4, Aadarsh Turkish: Visual Question Answering using Convolutional Neural Networks - Journal of Computer and Mathematics Education (TURCOMAT) Vol. 12No. 1S (2021)

[8]. Yash Srivastava, Shiv Ram Dubey, nehasis Mukherjee and Vaishnav Murali: Visual question answering using deep learning: A survey and performance analysis. - International Conference on Computer Vision and Image Processing (2021)

[9]. Kevin J Shih, Derek Hoiem and Saurabh Singh. Where to look: Focus areas for answering visual questions. The IEEE conference on computer vision and pattern recognition published the results., pages 4613–4621 (2020)

[10]. Vedanth P Bhar adwaj, Ananya B, Ms. Yashaswini B V "Sign Language Interpreter using Deep Learning Techniques"© 2022 IJRTI | Volume 7, Issue 6 | ISSN: 2456-3315

[11]. Shaik Khadar Sharif, Chava Sri Varshini, "Sign Language Recognition", Vol. 9, May-2020, Hyderabad-500090, Telangana, India.

[12]. Mehreen Hurroo, Mohammad Elham Walizad," Sign Language Recognition System using Convolutional Neural Network and Computer Vision ", IJERT, Vol. 9 Issue 12, December- 2020, 406-415, Delhi, India.

[13]. Shubhendu Apoorv, Sudharshan Kumar Bhowmick," Indian sign language interpreter using image processing and machine learning ", IOP Conference, Vol.7,2020, Issue 8, Chennai, India

[14]. Siming He, "Research of a Sign Language Translation System Based on Deep Learning", 2019 AIAM Conference, Canada.

[15]. Anshul Mittal, Pradeep Kumar an Team, "A Modified-LSTM Model for Continuous Sign- Language Recognition using Leap Motion", 2019 IEEE Sensors Journal, India.

[16]. Dr. prashantha, a.akash, b.jaidev , girish, jonna dileep "image text to speech conversion in desired language"2023 ijcrt | volume 11, issue 12 december 2023 | issn: 2320-2882.

[17]. H.Waruna H. Premachandra Information Communication Technology Center, Wayamba University of Sri Lanka, Makandura, Sri Lanka ; AnuradhaJayakody; HiroharuKawanaka; Converting high resolution multilingual printed document images in to editable text using image processing and artificial intelligence; 12-13 March 2022.

[18]. Nag, S., Ganguly, P. K., Roy, S., Jha, S., Bose, K., Jha, A., &Dasgupta, K. (2018). Offline Extraction of Indic Regional Language from Natural Scene Image using Text Segmentation and Deep Convolutional Sequence. arXiv preprint arXiv:1806.06208.

[19]. Sneha.C.Madre,Prof.S.B.Gundre "OCR Based Image Text To Speech Conversion Using MATLAB"IEEE Xplore Compliant Part Number: CFP18K74-ART; ISBN:978-1-5386-2842-3

[20]. K. Nirmala Kumari et al, Image Text to Speech Conversion using OCR Technique in Raspberry Pi, IJAREEIE, Vol 5, Issue 5,May 2016.

[21]. Carlos Merino-Gracia et al, A Head-mounted Device for Recognizing Text in Natural Scenes, International Workshop on Camera-Based Document Analysis and Recognition, Springer Berlin Heidelberg, 2011, Page: 29- 41.

[22]. Chucai Yi et al, Assistive Text Reading from Natural Scene for Blind Persons, Springer, Mobile Cloud Visual Media Computing. pp 219-241, edited book. ISBN: 978-3-319- 24700-7 (Print) 978-3-319-24702-1 (Online).