TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A Major Project Report

On

LIVE CAMERA FEED SCENE DESCRIPTOR FOR VISUALLY IMPAIRED

Submitted By:

AADITYA MANI SUBEDI (PUL075BCT001)

ARPAN GYAWALI (PUL075CT014)

BIDHAN KHATIWADA (PUL075BCT018)

BIJAYA SHRESTHA (PUL075BCT020)

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN COMPUTER ENGINEERING

(April 30, 2023)

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A Major Project Report

On

LIVE CAMERA FEED SCENE DESCRIPTOR FOR VISUALLY IMPAIRED

Submitted By:

AADITYA MANI SUBEDI (PUL075BCT001)

ARPAN GYAWALI (PUL075CT014)

BIDHAN KHATIWADA (PUL075BCT018)

BIJAYA SHRESTHA (PUL075BCT020)

Submitted To:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

(April 30, 2023)

PAGE OF APPROVAL

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **"LIVE CAMERA FEED SCENE DESCRIPTOR FOR VISUALLY IMPAIRED"** submitted by **Aaditya Subedi**, **Arpan Gyawali**, **Bidhan Khatiwada**, **Bijaya Shrestha** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

.....	.....
Supervisor	Internal examiner
Daya Sagar Baral	
Assistant Professor	Assistant Professor
Department of Electronics and Computer Engineering,	Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.	Pulchowk Campus, IOE, TU.

.....

External examiner

Assistant Professor

Department of Electronics and Computer Engineering,

Pulchowk Campus, IOE, TU.

Date of approval:

COPYRIGHT

ACKNOWLEDGEMENTS

Working with a variety of people whose thoughts and ideas have directly or indirectly aided or motivated us has been a wonderful pleasure. We'd want to express our gratitude to everyone who helped us finish this project, which was truly an academic experience. We'd like to express our gratitude to our supervisor Assistant Prof. Daya Sagar Baral for giving us the information, invaluable suggestions, and assistance we needed to complete this project. We appreciate and respect the initiative taken by the Department of Electronics and Computer Engineering, Pulchowk Campus in indulging the students in this self-practicing method of education to familiarize us with the tackling of real-world problems, meanwhile also enhancing our capabilities to work as a team.We'd also like to thank all of our friends who have supported us with this project, both directly and indirectly.

ABSTRACT

Every visually impaired people wants to interact with their nature, surrounding and people. They want to feel the event happening on the nature but are naturally deprived. Our product aims at assisting visually impaired individuals in navigating their way around Pulchowk Campus and describing the actions happening inside the campus. The product utilizes a live camera feed and visual transformer techniques to generate a descriptive caption and its audio output, providing the user with a proper and timely description of their surroundings. The product is designed to work on some landmark of the campus and wide range of activities. We suggest a model that is fine tuned on the pre-trained Git-base-Vatex model in our campus video datasets to describe the surrounding scene.

Keywords: *Video Captioning, Generative Image-To-Text Transformer, processor, Pulchowk Campus, Vatex dataset, WebRTC*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

TTS Text-To-Speech

NLP Natural Language Processing

LSTM Long Short Term Memory

CNN Convolutional Neural Network

AI Artificial Neural Network

ROUGE Recall-Oriented Understudy for Gisting Evaluation

GIT Generative Image to Text

WEBRTC Web Real-Time Communication)

W3C Wide Web Consortium

RTP Real Time Protocol

SVC Scalable Video Coding

BLEU Bilingual Evaluation Understudy

METEOR Metric for Evaluation of Translation with Explicit ORdering

API Application Programming Interface

CIDEr Consensus-based Image Description Evaluation

1. INTRODUCTION

1.1. Background

Describing visual content with natural language text has recently received increased interest, especially describing images with a single sentence[1, 2].Video description has so far seen less attention despite its important applications in human-robot interaction, video indexing, and describing movies for the blind. Various mobile apps have been developed to help especially visually impaired person. In the verge of implementing numerous and wide range of features on a single application, rose a problem in user interactivity with increased complexity. Furthermore, none of the pre-existing application captions video in real time. Rather they capture an image of real time process it and only describes the object on the scene rather than the action. While image description handles a variable length output sequence of words, video description also has to handle a variable length input sequence of frames. Related approaches to video description have resolved variable length input by holistic video representations[3], pooling over frames[4], or sub-sampling on a fixed number of input frames [5].

This work introduces a product that is capable of taking real-time video as an input and generating a description of the scene in real-time. The output of the product includes a caption that is both displayed on the screen and spoken aloud in a way that can be heard by humans. In contrast to other approaches, this technology provides a real-time solution for generating captions for live scene.

1.2. Objectives

The objectives of the project are:

1. Design an application that a visually impaired person can use to get informed about the actions happening around Pulchowk Campus
2. Design an intuitive and user-friendly interface that visually impaired individuals can use to interact with the system
3. Develop an algorithm or deep learning model that can extract features from live camera feed and generate a natural language description of the scene.
4. Ensure that the system operates in near real-time to provide immediate feedback to the user.

1.3. Problem Statement

There aren't any affordable and applicable technology that can assist visually impaired people to make them aware of the surrounding. Therefore, there is a need for a system that can automatically describe the video scene around a person.
Our automated solution can provide an assist to visually impaired people making them aware of the surrounding.Our product focuses to a visually impaired person visiting a Pulchowk Campus. One can easily be aware of the action taking place in the Campus premises.

1.4. Scope of Project

The prime focus of the application is provide an aid to visually impaired person to describe about the live scene happening around Pulchowk Campus. The user can be a student from a Campus or any individual visiting a Pulchowk Campus. This product can be scaled up so that it can be used by visually impaired person to get informed about general live action in abstract surrounding.
Furthermore, it can help in object detection by identifying and localizing objects in the scene, such as cars, buildings, people, animals, and other relevant entities. It also assist in recognizing and categorizing actions that are taking place in the scene, such as walking, running, jumping, waving, or other relevant movements, and also events that are occurring in the scene, such as a protest, a car accident, a fire, or other significant occurrences.

2. LITERATURE REVIEW

The problem of generating descriptions in open domain videos is difficult not just due to the diverse set of objects, scenes, actions, and their attributes, but also because it is hard to determine the salient content and describe the event appropriately in context.

Early work on scene descriptor considered tagging videos with metadata [6] and clustering captions and videos [7] for retrieval tasks. Several previous methods for generating sentence descriptions used a two stage pipeline that first identifies the semantic content (subject, verb, object) and then generates a sentence based on a template. This typically involved training individual classifiers to identify candidate objects, actions and scenes. They then use a probabilistic graphical model to combine the visual confidences with a language model in order to estimate the most likely content (subject, verb, object, scene) in the video, which is then used to generate a sentence. While this simplified the problem by detaching content generation and surface realization, it requires selecting a set of relevant objects and actions to recognize. Moreover, a template-based approach to sentence generation is insufficient to model the richness of language us ed in human descriptions – e.g., which attributes to use and how to combine them effectively to generate a good description.

In contrast, our approach avoids the separation of content identification and sentence generation by learning to directly map videos to full human-provided sentences, learning a language model simultaneously conditioned on visual features.

2.1. Related Works

In [4], LSTMs are used to generate video descriptions by pooling the representations of individual frames. Their technique extracts CNN features for frames in the video and then mean-pools the results to get a single feature vector representing the entire video. They then use an LSTM as a sequence decoder to generate a description based on this vector.

The approach in [1] also generates video descriptions using an LSTM; however, they employ a version of the two-step approach that uses CRFs to obtain semantic tuples of activity, object, tool, and locatation and then use an LSTM to translate this tuple into a sentence. Moreover, the model in [1] is applied to the limited domain of cooking videos.

Another recent project [8] uses LSTMs to predict the future frame sequence from an encoding of the previous frames.

Various Mobile Application like "Be My Eyes", "My Eyes", "Envision AI", "Seeing AI" are available for visually impaired people that can help with a variety of tasks.

'Seeing AI' developed by Microsoft uses CNN model that take recent image as an input and describe about the object on the scene.

'TapTapSee' uses the camera on a smartphone or tablet to identify objects and read aloud descriptions of them.

'Envision AI' is a mobile app that uses CNN Model to assist individuals, help users recognize objects, read text, and navigate their environment.

Most of these application, in context of describing a scene lacks a feature of captioning a real-time action. Rather, they capture an image process on it and provides an outcome in the form of detected object or a person. In contrast, our product rather focuses on captioning a live action and provides with an audio output that makes any visually impaired person to know about what is actually happening in his surrounding inside Pulchowk Campus.

3. THEORY

3.1. Artificial Intelligence

Artificial Intelligence (AI) is the field of computer science that focuses on creating intelligent machines that can perform tasks that typically require human intelligence. AI has seen tremendous growth and development over the past few decades, with breakthroughs in areas such as machine learning, natural language processing, computer vision, and robotics. These technologies are being applied to a wide range of industries and use cases, from healthcare and finance to manufacturing and transportation.The goal of AI is to create algorithms and systems that can learn from data, reason, make predictions, and take actions.

3.1.1. Types of Artificial Intelligence:

1. Supervised learning: This type of AI is trained on labeled data, where the desired output is known. The machine is taught to recognize patterns in the data and map them to the correct output. For example, in image recognition, a supervised learning algorithm would be trained on labeled images of cats and dogs, and then would be able to classify new images as either a cat or a dog based on what it has learned.

2. Unsupervised learning: This type of AI is trained on unlabeled data, where the desired output is unknown. The machine is taught to identify patterns and structures in the data without any specific guidance. For example, in clustering, an unsupervised learning algorithm could be used to group similar customers together based on their buying habits, without knowing in advance what specific characteristics to look for.

3. Reinforcement learning: This type of AI is trained through trial and error, where the machine receives rewards for certain actions and punishments for others. The machine learns to make decisions based on the consequences of its actions. For example, in game playing, a reinforcement learning algorithm would be trained to maximize its score based on the rewards it receives for different moves. In robotics, reinforcement learning can be used to train robots to perform tasks in the real world.

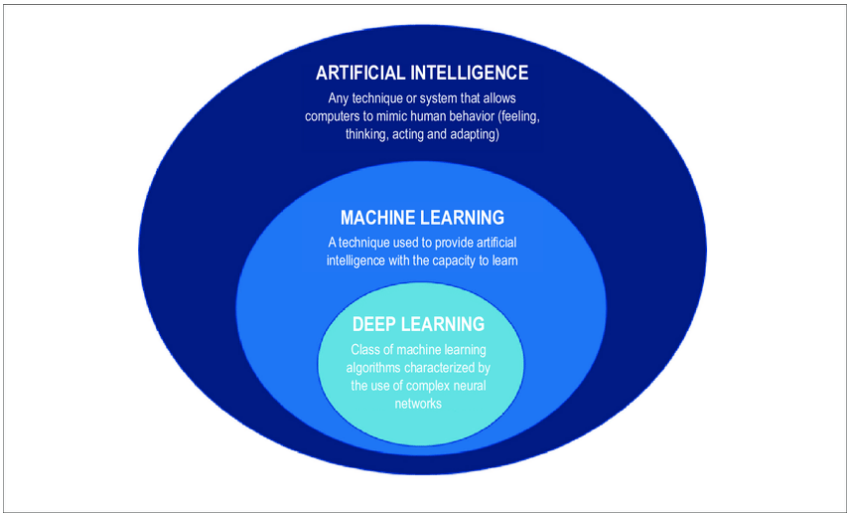

Figure 3.1: Artificial Intelligence

3.2. Machine Learning

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy, without being explicitly programmed.

Over the last couple of decades, the technological advances in storage and processing power have enabled some innovative products based on machine learning, such as Netflix's recommendation engine and self-driving cars.It is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, and to uncover key insights in data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase. They will be required to help identify the most relevant business questions and the data to answer them.

Machine learning algorithms are typically created using frameworks that accelerate solution development, such as TensorFlow and PyTorch.

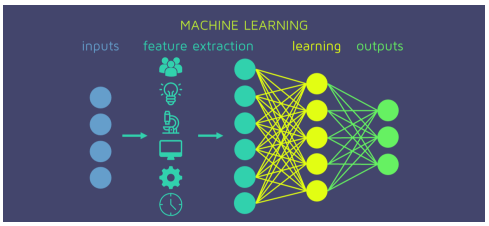

Figure 3.2: Machine Learning

3.3. Neural Network

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity. Tasks in speech recognition or image recognition can take minutes versus hours when compared to the manual identification by human experts. One of the most well-known neural networks is Google's search algorithm.

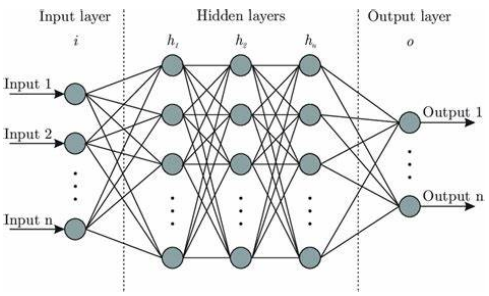

Figure 3.3: Neural Network

3.3.1. Types of Neural Network:

1. Multi-layer perceptrons (MLPs) : This type of network is composed of an input layer, one or more hidden layers, and an output layer. The information flows from the input layer through the hidden layers to the output layer in a forward direction, with each layer performing a nonlinear transformation of the input data. These networks are used for classification and regression tasks, such as image recognition or predicting stock prices.

2. Recurrent neural networks: This type of network is designed to process sequential data, where the order of the input data is important. The network has loops that allow information to be passed from one step to the next, and can learn from previous inputs to predict future outputs. Recurrent neural networks are widely used in natural language processing tasks such as text generation, machine translation, and speech recognition.

3. Convolutional neural networks: This type of network is designed to process image and video data. They use filters that can identify features such as edges, corners, and textures in the image. The filters are applied to overlapping regions of the image to generate feature maps that are used to make predictions. Convolutional neural networks are widely used in computer vision tasks such as object detection, image classification, and facial recognition.

3.4. Deep Learning

Deep Learning, is a more evolved branch of machine learning, and uses layers of algorithms to process data, and imitate the thinking process, or to develop abstractions.

It is often used to visually recognize objects and understand human speech. Information is passed through each layer, with the output of the previous layer providing input for the next layer. The first layer in a network is called the input layer, while the last is called an output layer.

All the layers between input and output are referred to as hidden layers. Each layer is typically a simple, uniform algorithm containing one kind of activation function.In a fully connected Deep neural network, there is an input layer and one or more hidden layers connected one after the other. Each neuron receives input from the previous layer neurons or the input layer. The output of one neuron becomes the input to other neurons in the next layer of the network, and this process continues until the final layer produces the output of the network. The layers of the

neural network transform the input data through a series of nonlinear transformations, allowing the network to learn complex representations of the input data.

It is capable of learning complex patterns and relationships within data. In deep learning, we don't need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain's biological neurons, and they are designed to learn from large amounts of data.

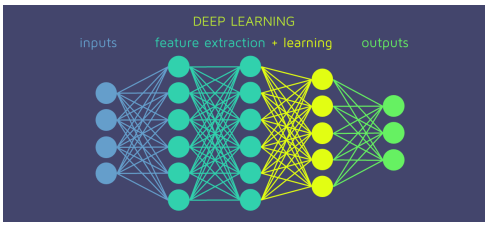

Figure 3.4: Deep Learning

3.5. NLP

NLP is a subfield of Artificial Intelligence (AI) that focuses on enabling machines to understand, interpret, and generate human language. It involves the use of statistical and machine learning techniques to analyze and model natural language data, including text and speech.It is used in a wide variety of everyday products and services. Some of the most common ways NLP is used are through voice-activated digital assistants on smartphones, email-scanning programs used to identify spam, and translation apps that decipher foreign languages.

3.5.1. Natural language techniques:

NLP encompasses a wide range of techniques to analyze human language. Some of the most common techniques you will likely encounter in the field includes:

1. Sentiment analysis: An NLP technique that analyzes text to identify its sentiments, such as "positive," "negative," or "neutral." Sentiment analysis is commonly used by businesses to better understand customer feedback.

2. Summarization: An NLP technique that summarizes a longer text, in order to make it more manageable for time-sensitive readers. Some common texts that are summarized

include reports and articles.

3. Keyword extraction: An NLP technique that analyzes a text to identify the most important keywords or phrases. Keyword extraction is commonly used for search engine optimization (SEO), social media monitoring, and business intelligence purposes.

4. Tokenization: The process of breaking characters, words, or subwords down into "tokens" that can be analyzed by a program. Tokenization undergirds common NLP tasks like word modeling, vocabulary building, and frequent word occurrence.

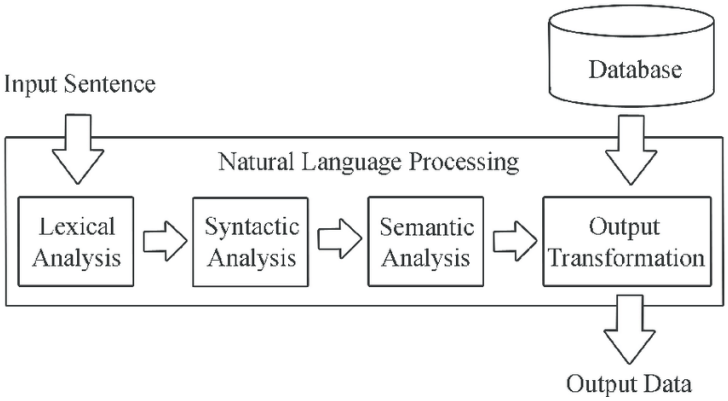

Figure 3.5: Natural Language Processing

3.6. RNN

A recurrent neural network (RNN) is a special type of artificial neural network adapted to work for time series data or data that involves sequences. Ordinary feedforward neural networks are only meant for data points that are independent of each other. However, if we have data in a sequence such that one data point depends upon the previous data point, we need to modify the neural network to incorporate the dependencies between these data points. RNNs have the concept of "memory" that helps them store the states or information of previous inputs to generate the next output of the sequence.The basic building block of an RNN is the recurrent cell, which is a module that takes an input, updates its internal state, and produces an output. The output is then fed back into the cell as input for the next time step. This allows the cell to maintain a memory of previous inputs, which can be used to influence the output at each time step.

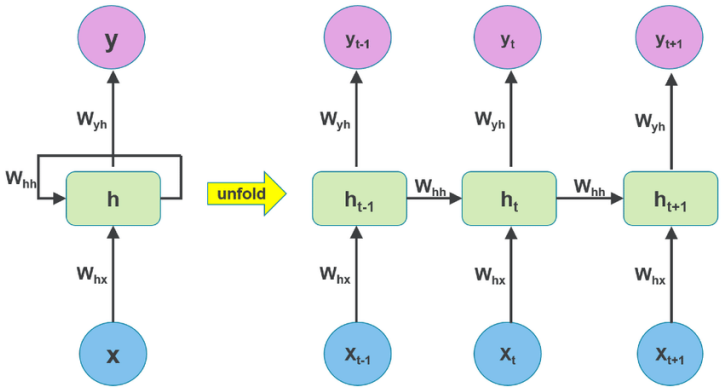

Figure 3.6: RNN

3.7. LSTM

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter Schmidhuber (1997), and were refined and popularized by many people in following work.1 They work tremendously well on a large variety of problems, and are now widely used. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn! The central role of an LSTM model is held by a memory cell known as a 'cell state' that maintains its state over time. The cell state is the horizontal line that runs through the top of the below diagram. It can be visualized as a conveyor belt through which information just flows, unchanged.

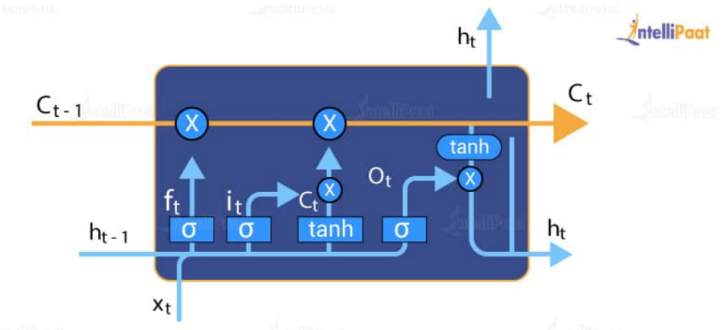

Figure 3.7: LSTM Model

Information can be added to or removed from the cell state in LSTM and is regulated by gates. These gates optionally let the information flow in and out of the cell. It contains a pointwise multiplication operation and a sigmoid neural net layer that assist the mechanism.

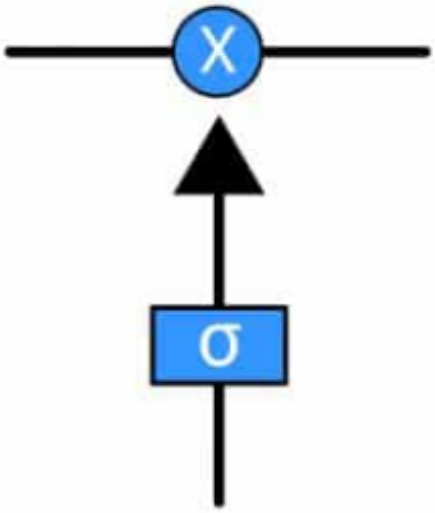

Figure 3.8: Gate in LSTM

The sigmoid layer gives out numbers between zero and one, where zero means 'nothing should be let through,' and one means 'everything should be let through.' LSTM networks are indeed an improvement over RNNs as they can achieve whatever RNNs might achieve with much better finesse. As intimidating as it can be, LSTMs do provide better results and are truly a big step in Deep Learning. With more such technologies coming up, you can expect to get more accurate predictions and have a better understanding of what choices to make

3.8. Transformer Model

The transformer model was introduced to address some of the limitations of traditional recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, which have been widely used in natural language processing (NLP) tasks.

One of the main drawbacks of RNNs and LSTMs is that they struggle to capture long-term dependencies in language, as they process sentences sequentially, one word at a time. This means that the model's ability to understand the relationship between words that are far apart in a sentence is limited.

The transformer model overcomes this limitation by introducing the concept of self-attention, which allows the model to weigh the importance of different words in a sentence when making predictions. This means that the model can better understand the relationship between words that are far apart in a sentence, and capture long-term dependencies in language more effectively.

The transformer model is highly parallelizable, which allows it to be trained more efficiently than RNNs and LSTMs. This is because the self-attention mechanism can be computed in parallel for all words in a sentence, whereas RNNs and LSTMs need to process words sequentially.

3.8.1. Architecture

The Transformer architecture follows an encoder-decoder structure but does not rely on recurrence and convolutions in order to generate an output. The task of the encoder, on the left half of the Transformer architecture, is to map an input sequence to a sequence of continuous representations, which is then fed into a decoder. And the decoder, on the right half of the architecture, receives the output of the encoder together with the decoder output at the previous time step to generate an output sequence.

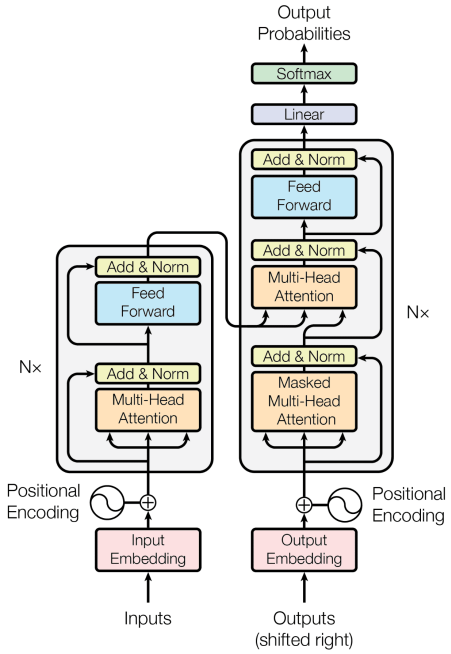

Figure 3.9: Architecture of Transformer Model

3.8.2. Encoder

The encoder, on the left half of the Transformer architecture which is shown below as:

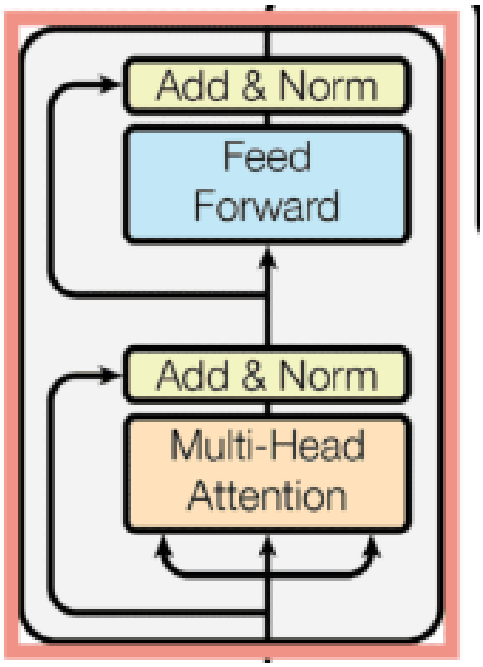

Figure 3.10: Encoder

The encoder of the Transformer model is responsible for processing the input sequence, which could be a sequence of words in natural language, an audio signal, or any other sequence-based input. The encoder consists of a stack of N = 6 identical layers, where each layer is composed of two sublayers: a multi-head self-attention mechanism and a fully connected feed-forward network.

The multi-head self-attention mechanism is the key innovation of the Transformer model. It allows the model to weigh the importance of different words in a sentence when making predictions. Specifically, the self-attention mechanism generates multiple "heads," each of which receives a linearly projected version of the input sequence, i.e., queries, keys, and values. These linear projections are learned weights that allow the model to attend to different aspects of the input sequence. The self-attention mechanism then computes a weighted sum of the values based on the similarity between the queries and the keys. The outputs of the different heads are concatenated and passed through a linear transformation to produce the final output of the self-attention mechanism. This process is repeated for each position in the input sequence.

The fully connected feed-forward network is another sublayer of each Transformer encoder layer. It consists of two linear transformations with ReLU activation in between. The purpose of this sublayer is to introduce nonlinearity and enable the model to learn more complex representations of the input sequence.

Each of the two sublayers in each Transformer encoder layer has a residual connection around it. This means that the output of the sublayer is added to the input of the sublayer, and the result is passed through a normalization layer to mitigate the effect of vanishing gradients during training. The normalization layer performs layer normalization, which normalizes the sum computed between the sublayer input and the output generated by the sublayer itself.

3.8.3. Decoder

The encoder, on the right half of the Transformer architecture which is shown below as:

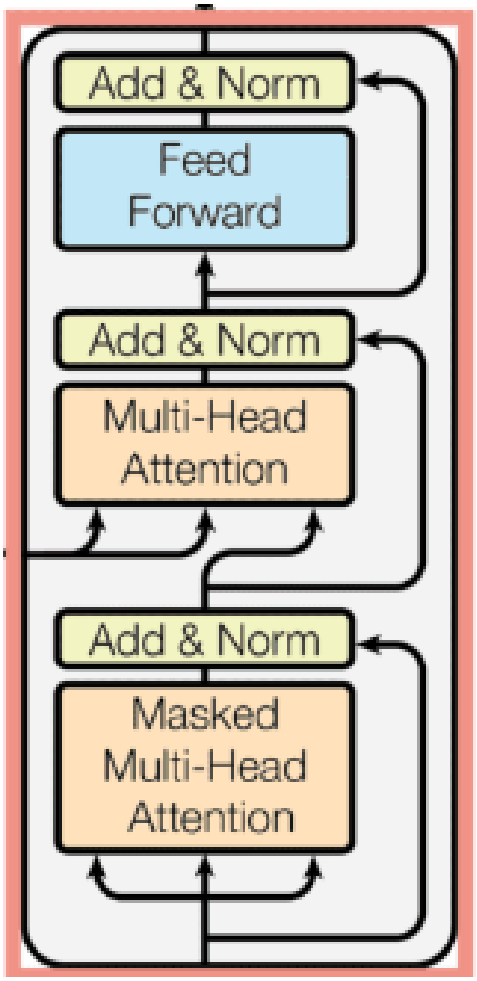

Figure 3.11: Decoder

The decoder of the Transformer model is responsible for generating the output sequence given the encoded input sequence. It consists of a stack of N = 6 identical layers, each of which is composed of three sublayers: a multi-head self-attention mechanism, a multi-head attention mechanism over the encoder output, and a fully connected feed-forward network.

The first sublayer in each decoder layer is a multi-head self-attention mechanism, which is

similar to the one used in the encoder. However, there is a key difference: during training, the decoder has access to the entire output sequence generated so far, including the current time step, while during inference, the decoder only has access to the previously generated output. Therefore, the self-attention mechanism in the decoder uses masked attention, where the attention weights for the current time step are set to zero, so that the decoder does not attend to itself.

The second sublayer in each decoder layer is a multi-head attention mechanism over the encoder output. This sublayer allows the decoder to attend to different parts of the input sequence while generating the output. Specifically, each head receives a linearly projected version of the decoder input, i.e., queries, and the encoder output, i.e., keys and values. The attention mechanism then computes a weighted sum of the values based on the similarity between the queries and the keys. The outputs of the different heads are concatenated and passed through a linear transformation to produce the final output of the multi-head attention mechanism.

The third sublayer in each decoder layer is a fully connected feed-forward network, which is similar to the one used in the encoder. Its purpose is to introduce nonlinearity and enable the model to learn more complex representations of the input sequence.

3.8.4. Training process

The training process of the Transformer architecture involves optimizing a loss function that measures the discrepancy between the predicted output sequence and the ground truth sequence. This loss function is typically the cross-entropy loss, which measures the difference between the predicted probability distribution over the output vocabulary and the true distribution. The training process of the Transformer architecture can be divided into two phases: pre-training and fine-tuning. In the pre-training phase, the model is trained on a large corpus of text data using an unsupervised learning objective, such as language modeling or masked language modeling. The pre-training objective is designed to encourage the model to learn useful representations of the input sequence, which can then be finetuned on downstream tasks. In the fine-tuning phase, the pre-trained model is further trained on a specific downstream task using supervised learning. The fine-tuning process involves replacing the final layer(s) of the model with task-specific layers and optimizing the model parameters to minimize the task-specific loss.

3.9. GIT

The Generative Image to Text Transformer model (GIT) [9] is a neural network model that generates natural language descriptions of images. It is based on the Transformer architecture, consisting of three main components: an image encoder, a text encoder, and a text decoder.

The image encoder is based on the contrastive pre-trained model. The input is the raw image and the output is a compact 2D feature map, which is flattened into a list of features. With an extra linear layer and a layernorm layer, the image features are projected into D dimensions, which are the input to the text decoder. It is based on a convolutional neural network (CNN) that extracts features from the input image. The features are then fed into a self-attention mechanism, which allows the model to attend to different parts of the image and extract relevant visual information.

In the text encoder, the text is tokenized and embedded into D dimensions, followed by an addition of the positional encoding and a layernorm layer. This gives a text embeddings which allows the model to attend to different parts of the text and extract relevant linguistic information.

The text decoder is a transformer module to predict the text description. The transformer module consists of multiple transformer blocks, each of which is composed of one self-attention layer and one feed-forward layer. Here, the image features are concatenated with the text embeddings as the input to the transformer module. The text begins with the [BOS] token, and is decoded in an auto-regressive way until the [EOS] token or reaching the maximum steps. The seq2seq attention mask as in figure below is applied such that the text token only depends on the preceding tokens and all image tokens, and image tokens can attend to each other. This is different from a unidirectional attention mask, where not every image token can rely on all other image tokens.

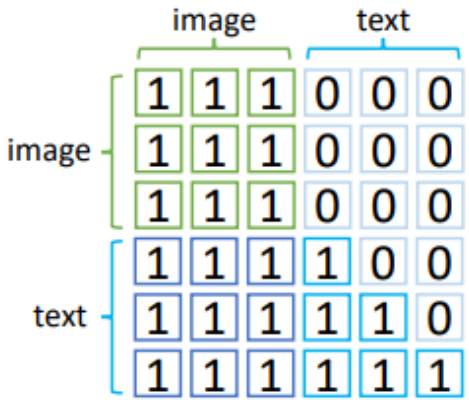

Figure 3.12: Seq2Seq Attention Mask

Here, If (i, j) is 1, the i-th output can depend on the j-th input; otherwise, not. For each image-text pair, let I be the image, y_i, $i \in 1,\ldots,N$ be the text tokens, y_0 be the [BOS] token, and y_{N+1} be the [EOS] token. We apply the language modeling (LM) loss to train the model:

$$\mathscr{L}_{\mathrm{LM}} = \frac{1}{N+1}\sum_{i=1}^{N+1}\mathrm{CE}(y_i, p(y_i|I,\{y_j\}_{j=0}^{i-1}))$$

where CE is the cross-entropy loss with label smoothing of 0.1.

3.9.1. Git-base-vatex

Git-base-vatex is a transformer model that has been obtained by fined tuning git-base model on Vatex dataset for video captioning, which generates captions for video sequences based on visual information.

VATEX [10] is a new large-scale multilingual video description dataset, which contains over 35k videos and about 10 captions for each video. VATEX is characterized by the following major unique properties. First, it contains both English and Chinese descriptions at scale, which can support many multilingual studies that are constrained by monolingual datasets. Secondly, VATEX has the largest number of clip-sentence pairs with each video clip annotated with multiple unique sentences, and every caption is unique in the whole corpus. Thirdly, VATEX contains more comprehensive yet representative video content, covering 600 human activities in total. Furthermore, both the English and Chinese corpora in VATEX are lexically richer and thus can empower more natural and diverse caption generation.

Six frames of each data is taken from the video and given to the image encoder as explained

above. Then their temporal embedding are extracted. These features are then concatenated with text encoding and given to text decoder.

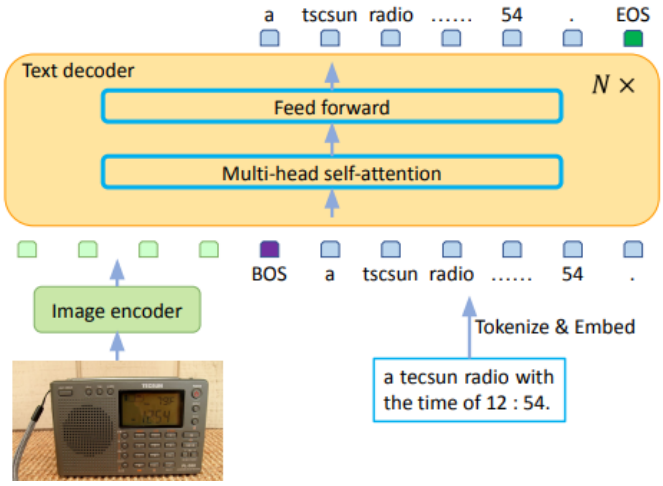

Figure 3.13: Image Captioning

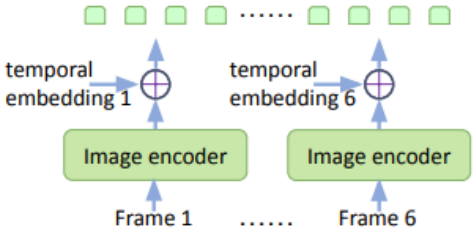

Figure 3.14: Video Captioning

Figure 3.15: GIT architecture

3.10. WebRTC

WebRTC (Web Real-Time Communication) is an open-source technology and framework that allows real-time communication, such as audio and video calls, between web browsers and mobile applications. WebRTC enables peer-to-peer communication without requiring any plugins, downloads, or installations, making it an attractive option for real-time applications such as video conferencing, online gaming, and file sharing.

WebRTC was first developed by Google in 2011 and is now a standard supported by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF). WebRTC uses a combination of JavaScript APIs and open-source protocols such as STUN (Session Traversal Utilities for NAT) and ICE (Interactive Connectivity Establishment) to establish and maintain a direct communication channel between peers.

Major components of WebRTC include several JavaScript APIs:

- **getUserMedia** acquires the audio and video media (e.g., by accessing a device's camera and microphone).
- **RTCPeerConnection** enables audio and video communication between peers. It performs signal processing, codec handling, peer-to-peer communication, security, and bandwidth management.

- **RTCDataChannel** allows bidirectional communication of arbitrary data between peers. The data is transported using SCTP over DTLS.[22] It uses the same API as WebSockets and has very low latency.

The WebRTC API also includes a statistics function:

- **getStats** allows the web application to retrieve a set of statistics about WebRTC sessions. These statistics data are being described in a separate W3C document.

The WebRTC API includes no provisions for signaling, that is discovering peers to connect to and determine how to establish connections among them. Applications use Interactive Connectivity Establishment for connections and are responsible for managing sessions, possibly relying on any of Session Initiation Protocol, Extensible Messaging and Presence Protocol, Message Queuing Telemetry Transport, Matrix, or another protocol. Signaling may depend on one or more servers.

You can group WebRTC applications into 4 broad categories:

1. **Conversational voice and video:** Applications that need the ability to have a person communicate with others in real time and in a conversational manner. These will more often than not end up using WebRTC
2. **Live streaming:** while WebRTC isn't the most popular choice for streaming, it is one of the best technologies available for low-latency live streaming. If you need to stream something to one or more users and maintain really low latency to enhance the interactivity (things like cloud gaming, gambling, auctions, webinars, etc) – then WebRTC might be a great choice
3. **Data transfer:** you can send voice and video with WebRTC, but you can also send arbitrary data. This can be used to share huge files between machines with little need for server space for example. Or it can be used to create a bittorrent like experience
4. **Privacy:** since WebRTC runs direct between browsers, it is sometimes used to enhance privacy. Doing this by simply not sending the media or data via servers at all

3.11. Fine-tuning with transfer learning

Fine-tuning with transfer learning is a technique in mac hine learning where a pre-trained model is adapted to a new task by updating some of its parameters. Transfer learning is the practice

of using knowledge gained from solving one problem to help solve another related problem.

In the case of fine-tuning with transfer learning, a pre-trained model is used as a starting point, and then the model is fine-tuned on a new task using a smaller amount of data than would be required to train the model from scratch. Fine-tuning typically involves updating the final layers of the pre-trained model to suit the new task, while freezing the weights of the lower layers, which capture more general features.

Fine-tuning with transfer learning can significantly reduce the amount of time and resources required to train a model, as well as improve its performance on new tasks. It is a powerful technique that has enabled significant progress in various areas of machine learning, including computer vision, speech recognition, and natural language processing.

3.12. Multithreading

Multithreading is a programming concept that allows multiple threads of execution to run concurrently within a single process. A thread is a lightweight unit of execution that consists of a program counter, a stack, and a set of registers. Multiple threads can exist within a single process, and they can share the same memory space.

Multithreading can improve the performance of programs that have a lot of I/O or that need to perform multiple tasks simultaneously. By allowing multiple threads to run concurrently, a program can take advantage of modern CPUs that have multiple cores or hyperthreading capabilities.

Even though the processor executes only one instruction at a time, threads from multiple programs are executed so fast that it appears multiple programs are executing concurrently.

Each CPU cycle executes a single thread that links to all other threads in its stream. This synchronization process occurs so quickly that it appears all the streams are executing at the same time.Each thread contains information about how it relates to the overall program. While in the asynchronous processing stream, some threads are executed while others wait for their turn.

3.13. Text-to-Speech

Text-to-Speech (TTS) is a technology that has become increasingly popular in recent years. It is a process that converts written text into spoken words, making digital content accessible to

people with visual impairments or reading difficulties. TTS works by analyzing the text and using natural language processing algorithms to determine the pronunciation of each word. The system then generates a waveform that represents the spoken words, which can be played back through a speaker or headphones. TTS has a wide range of applications, including accessibility, language learning, navigation, and assistive technology. For instance, it can help people who have difficulty reading to access digital content such as books, articles, or websites. It can also be used to improve pronunciation and listening skills in language learning, provide audible directions in navigation systems, or serve as an assistive technology for people with speech disabilities.

3.14. Technology Used

- Programming Languages:
  - Python3 (For Deep Learning and Backend):Python is a powerful, high-level programming language that is simple to read and has a clean structure. It has an interpreted nature, meaning you can run code without the intermediate compilation step. Python is rapidly being adopted by fortune organizations and has upcoming libraries that support machine learning, data science, database manipulation, and AI. Python 3 is an interpreted language, which means that code written in Python is executed line by line, rather than being compiled first. This makes it easier to write and test code, and it also allows for more rapid development cycles.It is a dynamically typed language, which means that variables can change their data type during runtime. This makes it more flexible and easier to use than statically typed languages like C++ or Java.

    Python 3 is an object-oriented language, which means that it supports object-oriented programming principles such as encapsulation, inheritance, and polymorphism. This allows developers to write modular and reusable code, which can help to reduce development time and improve code quality.It has a large number of third-party libraries that make it a popular choice for data science, machine learning, web development, and more. Some popular libraries include NumPy, Pandas, Matplotlib, Django, Flask, and TensorFlow.

    Python is a cross-platform language, which means that code written in Python can be run on different operating systems like Windows, Mac, and Linux. This makes

it a flexible and versatile choice for developers who need to create applications that run on multiple platforms.It comes with a comprehensive standard library that provides built-in support for a wide range of tasks such as file I/O, networking, regular expressions, and more. This can help to reduce development time and make it easier to write high-quality code.

- Flutter (For Frontend):Flutter is an open-source mobile app development framework developed by Google. It is designed to help developers build high-performance, natively compiled mobile apps for iOS, Android, and the web from a single codebase.It uses the Dart programming language, which is a statically typed language that has a concise and expressive syntax. This makes it easy for developers to write and read code, which can help to reduce development time and improve code quality.One of the key features of Flutter is its widget-based architecture. Widgets are the basic building blocks of a Flutter app, and they can be combined in various ways to create complex and dynamic user interfaces. Flutter provides a large number of built-in widgets, as well as support for custom widgets, which allows developers to create unique and visually appealing interfaces.

  Flutter provides a hot-reload feature, which allows developers to see the changes they make to their code in real-time, without having to recompile the entire app. This makes it easy to experiment and iterate quickly during the development process.It also provides a rich set of tools and libraries, including the Flutter SDK, the Dart SDK, and the Flutter plugin for Android Studio or Visual Studio Code. These tools make it easy to develop, test, and deploy apps for a wide range of devices and platforms.

- WebRTC (For Integrating Layer):WebRTC (Web Real-Time Communications) is an open-source technology that enables real-time communication between web browsers and mobile applications. It provides a set of APIs that allow developers to add voice and video chat, file sharing, and data transfer capabilities to their applications without requiring the installation of additional software or plugins. WebRTC uses peer-to-peer networking technology to establish secure and efficient connections between devices, and it supports multiple platforms, including desktop and mobile

devices. Overall, WebRTC simplifies real-time communication and collaboration for developers and end-users.

- Frameworks:
  - PyTorch (For Deep Learning):PyTorch is an open-source machine learning library developed by Facebook's AI Research team that is widely used for building and training deep learning models. It supports both CPU and GPU computation and provides a variety of tools and modules for building and training neural networks. PyTorch is known for its dynamic computational graph capabilities, which make it easy to build complex models with variable-length inputs. Additionally, PyTorch has a user-friendly interface for debugging and visualizing the training process, with many resources and tutorials available online.

  - Aiohttp (For Backend Server):Aiohttp is an open-source Python library for building asynchronous HTTP-based web services and clients. It is built on top of asyncio, a Python library for writing concurrent code, and provides an easy-to-use API for building and consuming HTTP APIs. Aiohttp supports both client and server-side implementations, making it a versatile tool for web developers. It also provides support for HTTP/2, WebSockets, and secure HTTPS connections. Overall, aiohttp simplifies the process of building high-performance and scalable web applications in Python.

4. METHODOLOGY

4.1. Process Model

The Agile software development methodology is a straightforward and effective approach to develop software solutions from a system requirement. The Agile process model includes the following steps:

1. **Planning:** A comprehensive plan was developed to determine the time and effort required to build the project, and to assess the technical and economic feasibility of the project.
2. **Requirement Analysis:** The requirements were defined and analyzed to design the various elements of the project.
3. **Designing:** The design phase began, and the project elements were designed based on the requirements. This involved constructing user flow diagrams or high-level UML diagrams.
4. **Building:** The work on the project started, and a working product was deployed that had minimal functionality. The product underwent several stages of improvement.
5. **Testing:** The product's quality and performance were assessed, and bugs were identified and resolved during this phase.

The project was carried out using a process model that consisted of five iterations. The initial iteration involved thorough planning and research to establish the project cycles, which was followed by the execution of the process. For the subsequent iterations, the project started from the analysis phase of the process model and progressed to testing the final product. The first two iterations were completed within the first 2 months of the project's conception, while the last three iterations were done after this initial six-month period.

4.2. Planning and research

Our application employs 3 major tasks i.e image stream transmission, caption generation, and text to speech generation. so appropriate research on these tasks needed to be done.
For the real-time video transmission from one device to another device in a real-time various cutting-edge technologies were available like Real Time Protocol(RTP), H.264, VP9, Scalable Video Coding (SVC), WebRTC. Among them, we choose webRTC because of its high-quality

Video , in-built security, low-latency and easier integration it provides through simple APIs
For the task of generating captions from a stream of images, we need to account the temporal information between the image frames. Hugginface provides an easy-to-integrate transformer model that employs an attention mechanism. Microsoft provides an open-source GIT-BASE pre-trained on 4M images and GIT-LARGE pre-trained on 14M images. GIT-BASE is suitable for our simpler application as the model is not heavy and thus offers quicker inference processing. Various GIT-BASE models pre-trained on different image datasets like GIT-BASE-COCO, GIT-BASE-VATEX, and GIT-BASE-MSRVTT were available . Among them, GIT-BASE-VATEX [1] with the CIDEr score of 91.5 seems to be promising for our application. We thus chose it, for fine-tuning with our custom datasets of Pulchowk Campus domain.
Similarly, we need to employ a text to speech mechanism inorder to generate the speech from the predicted caption. Various flutter packages offer text-to-speech translation, flutter_azure_-tts, rw_tts, text_to_speech, flutter_tts. We choose flutter_tts for its easier integration and easier customization it provides.

4.3. Training and Inferencing of model

4.3.1. Data Collection and pre-processing

For this project, we collected video data inside the Pulchowk Campus boundary. The location covered three main areas: the Dean's office, the ICTC building, and the library. We captured various actions of people, including walking, riding, talking on cell phones, and talking to each other. We captured total of 232 videos where 52 videos describing actions on the ICTC buildings, 45 videos describing actions near Dean office,40 videos describing actions near library, 34 videos describing actions near campus cafe and 61 videos with general actions in other places inside campus without taking location into consideration. From 232 videos, 200 are was used for training and 32 for testing.

To capture the video, we used a mobile with a frame rate of 30 frames per second. We took care to ensure that the camera was placed in a location that provided a clear view of the scene, while at the same time minimizing any obstructions or distortions in the video.

After capturing the video, we split it into short 5-6 second video clips. For each clip, we wrote a caption manually. The captions were designed to describe the actions taking place in the video and provide context to the viewers. On average, each caption contained approximately

[1] `https://huggingface.co/microsoft/git-base-vatex`

10 words.

4.3.2. Fine tuning

We used PyTorch and Google Collab environment to fine-tune the pre-trained GIT-base-vatex model on our custom datasets. We first initialized the model with the pre-trained weights and biases of GIT-base-vatex model. GITAutoprocessor of git-base-vatex, encodes the images frames and corresponding caption into the encoding vector. The encoding vector is BatchEncoding class. We use AdamW optimiser with the learning rate of 5e-6. The batch size is set to 2. We trained the model for 20 epochs. The average training time was 4 hours. The loss at 1st epoch was 8.57 which reached 3.70 at the end of 20 epoch.

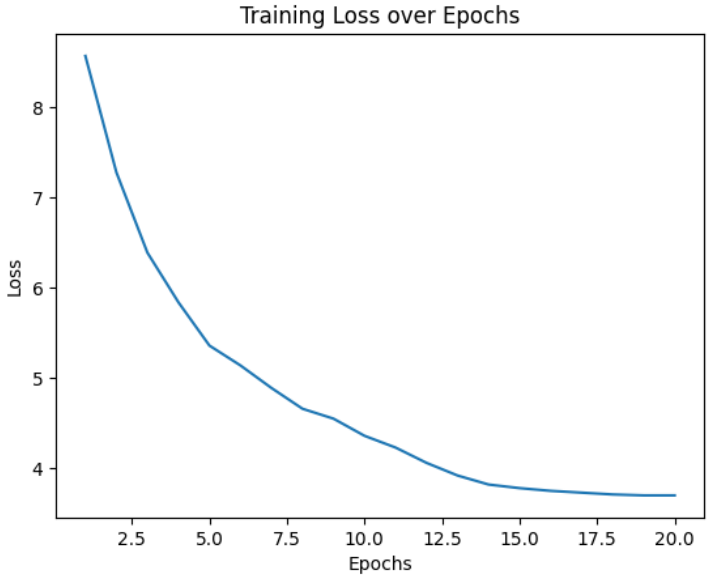

Figure 4.1: Epoch vs Loss

Adam Optimizer: Adam is a popular optimization algorithm used in deep learning for optimizing stochastic gradient descent (SGD). It combines the advantages of two other optimization algorithms: Adagrad and RMSProp. The Adam optimizer keeps an exponentially decaying average of past gradients and squared gradients, respectively. It is particularly effective when dealing with large datasets and high-dimensional parameter spaces. Transformers require a lot of computational resources to train, and the Adam optimizer has been found to be particularly effective for optimizing the large number of parameters in these models.

4.3.3. Inferencing

The fine-tuned model resulted from the above fine-tuning process is then integrated with the backend server to perform the inference task. The inferencing model(refers to fine-tuned model on our custom datasets) takes in 6 image frames sampled from the 3 seconds continuous image streams. The frames are encoded by the GITAutoProcessor to produce an enconding vector, BatchEncoding class. This encoding vector is fed inside the inferencing model that generates the caption output of maximum length 20.

4.4. Working Principle of our application

A typical live camera feed scene descriptor for visually impaired, which takes a continuous video stream as input and give a description as output, contains the following key modules.

4.4.1. Video Description Interface

A simple camera feed interface is developed using flutter for a quick demo of the model and its function. The interface ask for the user consent to capture the video. A user can start the process by pressing START button. Our App then ask for the user consent to use the device camera. When the user consent for camera use, the app then initializes a WebRTC peer connection instance, sends the offer message to the backend server. When the backend server is ready for RTCConnection, backend replies the answer message and the connection is said to be established. The interface then keep forwarding the camera feed stream to the backend server using webRTCtechnology, particularly flutter_webrtc [2]in frontend. It keeps listening to the predicated caption from the backend through RTCData channel.
For each received caption, the frontend then generates the voice through the device's speaker using a text-to-speech transformation mechanism.

4.4.2. WebRTC

WebRTC is an integrating layer for data communication between frontend and backend in our application. Frontend and backend establish peer connection between them through offer and answer message transmission. Sending and receiving offers and answers for peer connection is done through **HTTP API**. On successful peer connection, the images stream is propagated from frontend to backend through **RTC track**. The predicted caption is delivered from backend to frontend through **RTC data channel**.

[2]`https://pub.dev/packages/flutter_webrtc`

4.4.3. Backend Server

The backend server for the application was implemented using the aiohttp[3] framework in Python. When the server is started, the configuration for the trained model is loaded from a initialization.py file. Before listening to requests from the designated port, the server loads the model using the configuration and stores it as a global variable. This prevents the loading of the model from the disk to the memory of the GPU for each request. Although this method reserves a certain portion of the memory of GPU constantly, this seems to be the right option because it saves memory allocation and deallocation time for the model for each request to the server. Then the server listens to the offer requests and sends answer responses in the form of API for webRTC peer connection. The information backend receives on the offer message is discussed on earlier section. When the backend is ready, it answers the frontend with the relevant information required for peer connection. The information on the answer message is similar to the offer message discussed earlier.

On succesful peer connection, backend keeps listening for a stream of video feed from frontend, and in response it sends predicated caption from model to the frontend through webRTC data channel. Every time a new data is sent for prediction, the server replaces the data to be predicted in the model and recomputes the results.

The backend server continuously accumulates the captured frames for 3 seconds. All the frames captured for 3 seconds are sampled to extract only 6 frames. These 6 sampled frames are passed into the ML model for inference. Each inference processes run on the background in a new thread. Running the inference on background thread enables backend server to continuously collect the frames in real-time.

4.4.4. Inference Model

Fined tuned GIT-Base-Vatex model(pretrained on the vatex datasets) takes sampled 6 frames from backend. It performs the inference task on a seperate thread from main thread as well as other threads where inference process is running i.e. each chunk of frames runs inference model on seperate thread. Once the inference process completes, it generates caption and then triggers the backend server to inform that new caption is predicted. Each inference process run on the background so that backend server process is freed to captured new frames while inferencing is taking place.

[3]https://docs.aiohttp.org/en/stable/

5. SYSTEM DESIGN

5.1. Block Diagram

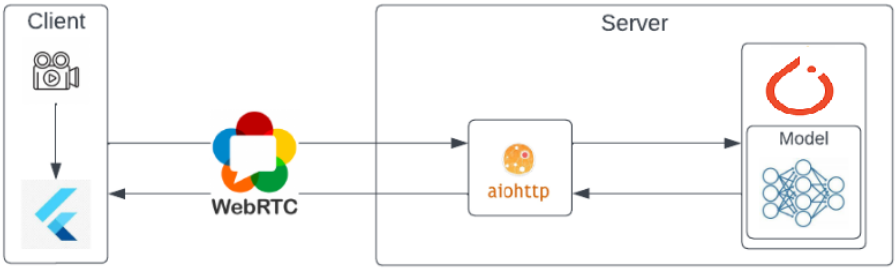

Figure 5.1: Block Diagram

5.2. Use Case Diagram

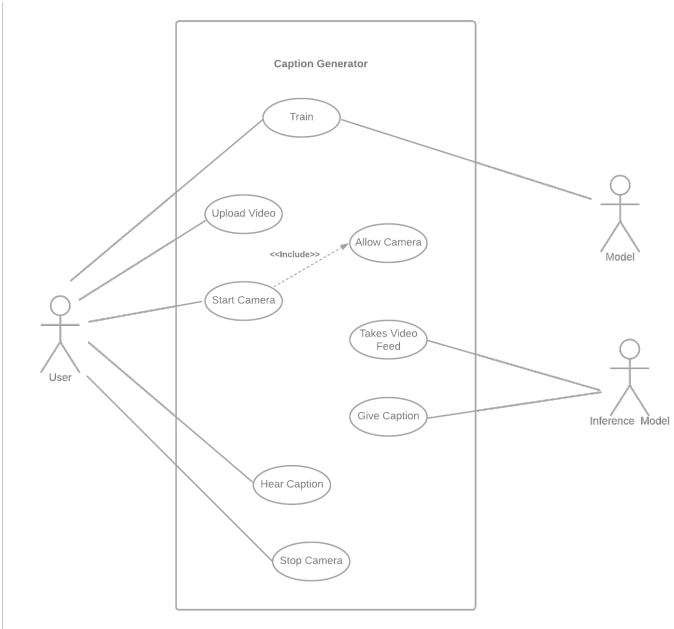

Figure 5.2: Use Case Diagram

5.3. Sequence Diagram

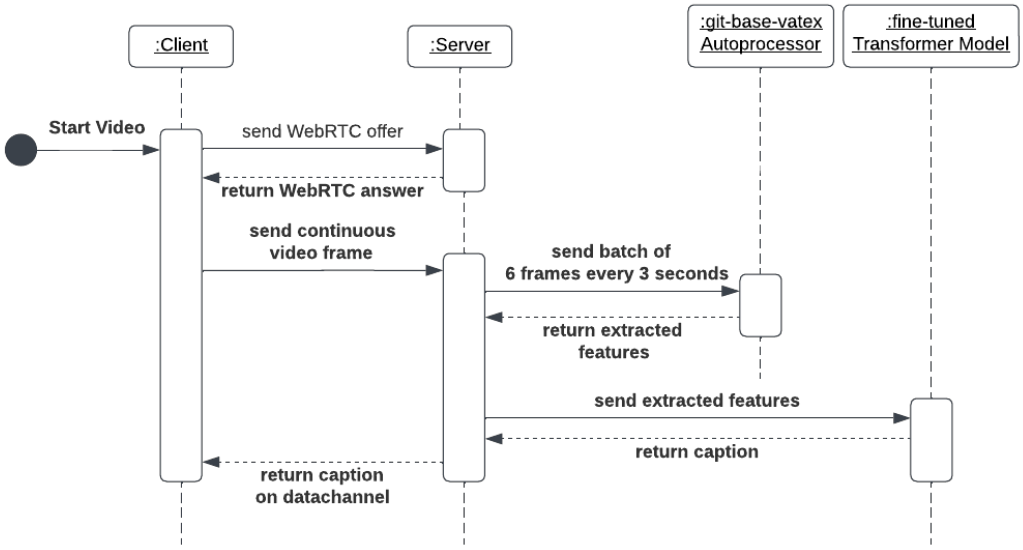

Figure 5.3: Sequence Diagram

6. RESULTS AND DISCUSSION

We trained our data in different model. Training an LSTM encoder-decoder sequence-to-sequence model in MSVD with 2k videos was unable to provide proper result due to less amount of data and simple architecture without any attention mechanism. While using VGG16 CNN model in encoder, we obtained the following results in test dataset:
BLUE-4: 0.192
METEOR: 0.230
CIDEr: 50.8

git-base-vatex: The git-base model by microsoft which was fine tuned in vatex dataset for video captioning. Vatex dataset contains 35k videos and each videos are given about 10 captions. 26k videos are taken as training set, 6k as test set and 3k videos as validation set. It was trained for 10 epochs with learning rate of 2.5×10^{-6}
CIDEr score obtained on test was 91.5

Our fine-tuned model: Finally, we fine tuned the git-base-vatex model on our own dataset collected at Pulchowk campus. Our dataset contains 232 videos at different landmark of pulchowk campus and each videos are given about 3 captions. 200 videos are taken as training set and 32 as test set. It was trained for 20 epochs with learning rate of 5×10^{-6}

The scored obtained on the test data on our fine tuned model are explained below:

Table 1: Evaluation Results

Metric	Score
BLEU_4	0.23935256374863303
METEOR	0.29055057593408057
ROUGE_L	0.5760558588613365
CIDEr	100.19187158496112

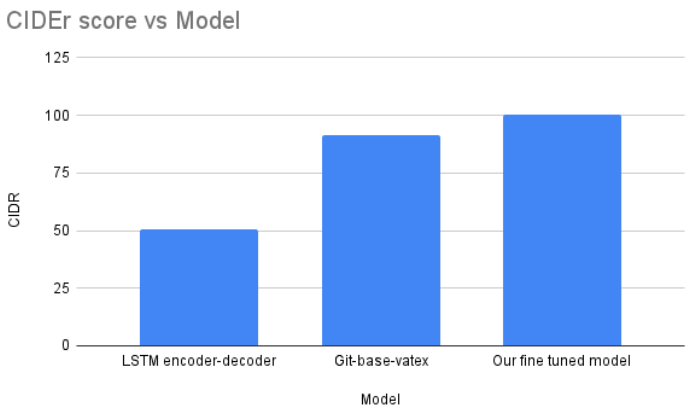

Figure 6.1: CIDEr score vs Model

BLEU-4 (Bilingual Evaluation Understudy) is a metric that measures the similarity between a machine-generated translation and a reference translation. Specifically, BLEU-4 computes the n-gram overlap between the machine translation and the reference translation, where n ranges from 1 to 4. BLEU-4 scores range from 0 to 1, with higher scores indicating better translations.

METEOR (Metric for Evaluation of Translation with Explicit ORdering) is another metric that measures the quality of machine-generated translations. METEOR combines various components to measure the similarity between a machine translation and a reference translation. These components include unigram precision, recall, and alignment scores, as well as various measures of paraphrasing. METEOR scores range from 0 to 1, with higher scores indicating better translations.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics that measure the similarity between a machine-generated summary and a reference summary. ROUGE includes several variants, such as ROUGE-1 (which measures unigram overlap), ROUGE-2 (which measures bigram overlap), and ROUGE-L (which measures longest common subsequence). Like BLEU-4 and METEOR, ROUGE scores range from 0 to 1, with higher scores indicating better summaries.

CIDEr (Consensus-based Image Description Evaluation) is a metric specifically designed for evaluating image captioning systems. CIDEr is based on a consensus-based model of human evaluation, where multiple reference captions are used to evaluate the quality of a machine-generated caption. CIDEr computes a weighted combination of n-gram similarity scores between the machine-generated caption and the reference captions. CIDEr scores range from 0 to infinity, with higher scores indicating better captions.

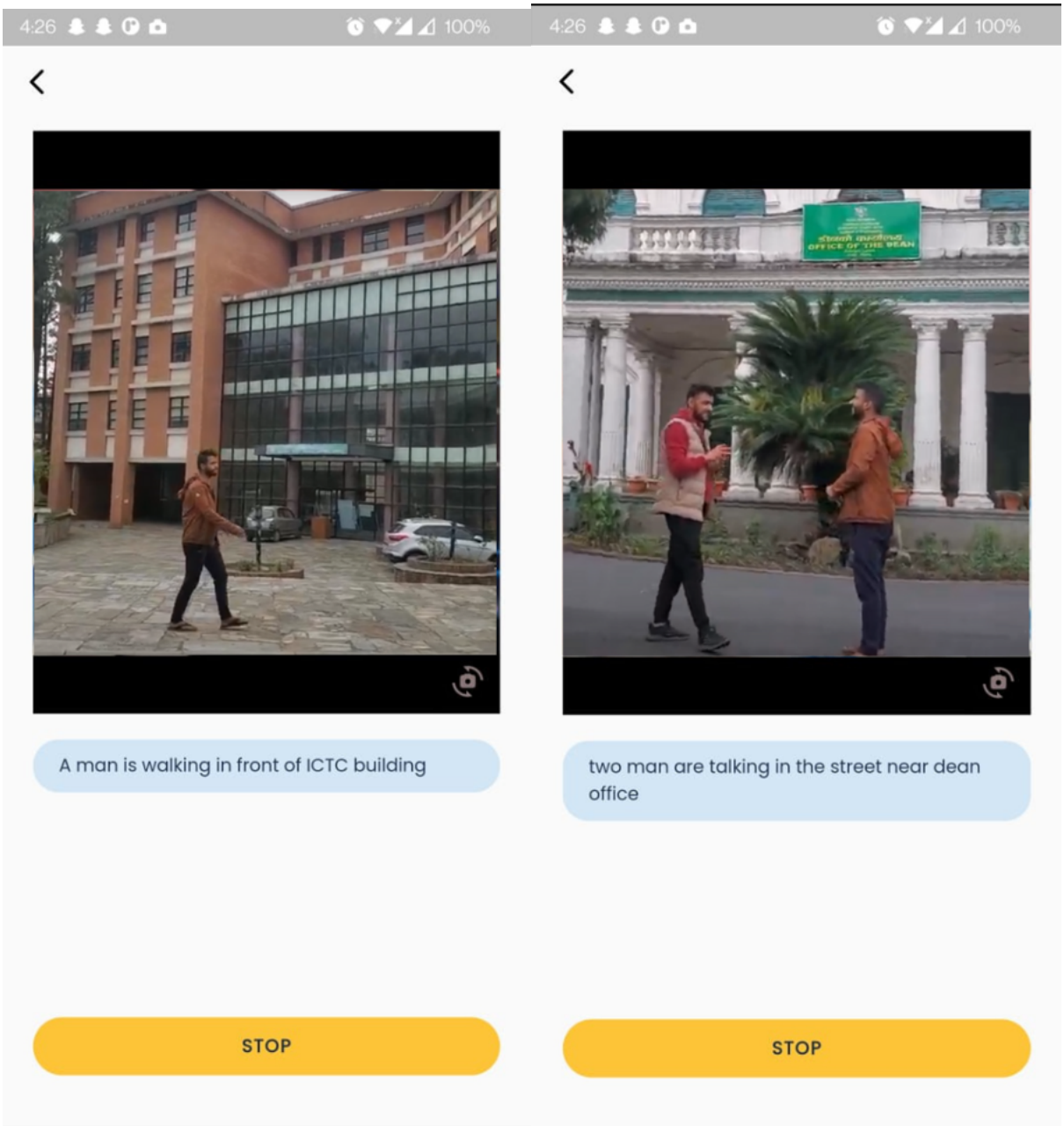

Figure 6.2: Example actions at Pulchowk Campus landmarks

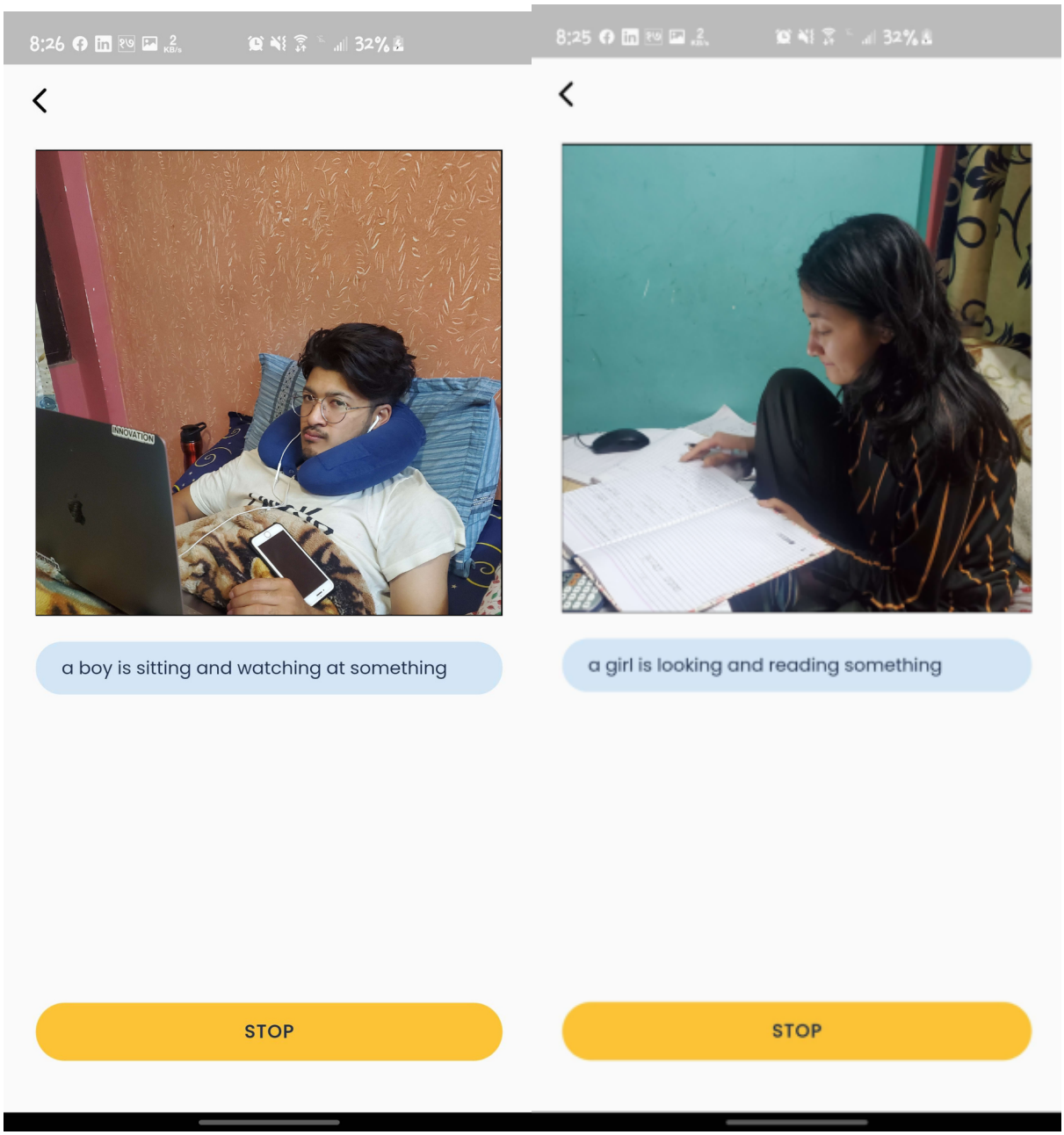

Figure 6.3: Example general actions

7. CONCLUSION

In conclusion, our mobile app that provides description of the scene in near real time for visually impaired people using live camera feed and helps them navigate through Pulchowk Campus. It can describe various activities happening around the campus. The transformer model used in the app has been trained on a large dataset, ensuring accurate and reliable descriptions of the environment. Also the app has been tested and has provided somewhat satisfying description of the scene. Overall, this app has great potential to improve the quality of life for visually impaired individuals, enabling them to navigate and explore their environment with more independence and confidence. Further improvements and enhancements can be made to the app in the future, based on feedback and advancements in machine learning technology.

8. LIMITATIONS AND FUTURE ENHANCEMENTS

8.1. Limitations

1. The model does not include all the location inside the Pulchowk campus
2. The model could not express all domain of complex actions and activities due to limited number of data.
3. Due to the lack of computational resources, the model may sometimes take longer than expected to return the caption.

8.2. Future works

1. By collecting more data, all the location and more actions inside the Pulchowk campus can be incorporated
2. High powerful GPU computing resources can be used, inorder to reduce the inference computation time delay.

REFERENCES

[1] Jeff Donahue et al. “Long-term recurrent convolutional networks for visual recognition and description”. In: (2015), pp. 2625–2634. DOI: 10.1109/CVPR.2015.7298935. URL: https://doi.org/10.1109/CVPR.2015.7298935 (cit. on pp. 1, 3).

[2] Andrej Karpathy and Li Fei-Fei. “Deep visual-semantic alignments for generating image descriptions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3128–3137 (cit. on p. 1).

[3] Sergio Guadarrama et al. “YouTube2Text: Recognizing and Describing Arbitrary Activities Using Semantic Hierarchies and Zero-shot Recognition”. In: *2013 IEEE International Conference on Computer Vision*. IEEE. 2013, pp. 2712–2719 (cit. on p. 1).

[4] Subhashini Venugopalan et al. “Translating Videos to Natural Language Using Deep Recurrent Neural Networks”. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015, pp. 1494–1504 (cit. on pp. 1, 3).

[5] Li Yao et al. “Describing Videos by Exploiting Temporal Structure”. In: *arXiv preprint arXiv:1502.08029v4* (2015) (cit. on p. 1).

[6] Hrushikesh Aradhye, George Toderici, and Jay Yagnik. “Video2Text: Learning to Annotate Video Content”. In: *2009 IEEE International Conference on Data Mining Workshops*. IEEE. 2009, pp. 263–270 (cit. on p. 3).

[7] Hai Huang et al. “A Multi-modal Clustering Method for Web Videos”. In: *2013 International Symposium on Computer Technology and Control & Automation*. IEEE. 2013, pp. 707–710 (cit. on p. 3).

[8] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. “Unsupervised learning of video representations using LSTMs”. In: *International Conference on Machine Learning*. JMLR.org. 2015, pp. 843–852 (cit. on p. 3).

[9] Jianfeng Wang et al. *GIT: A Generative Image-to-text Transformer for Vision and Language*. 2022. arXiv: 2205.14100 [cs.CV] (cit. on p. 17).

[10] Xin Wang et al. *VATEX: A Large-Scale, High-Quality Multilingual Dataset for Video-and-Language Research*. 2020. arXiv: 1904.03493 [cs.CV] (cit. on p. 18).

APPENDICES

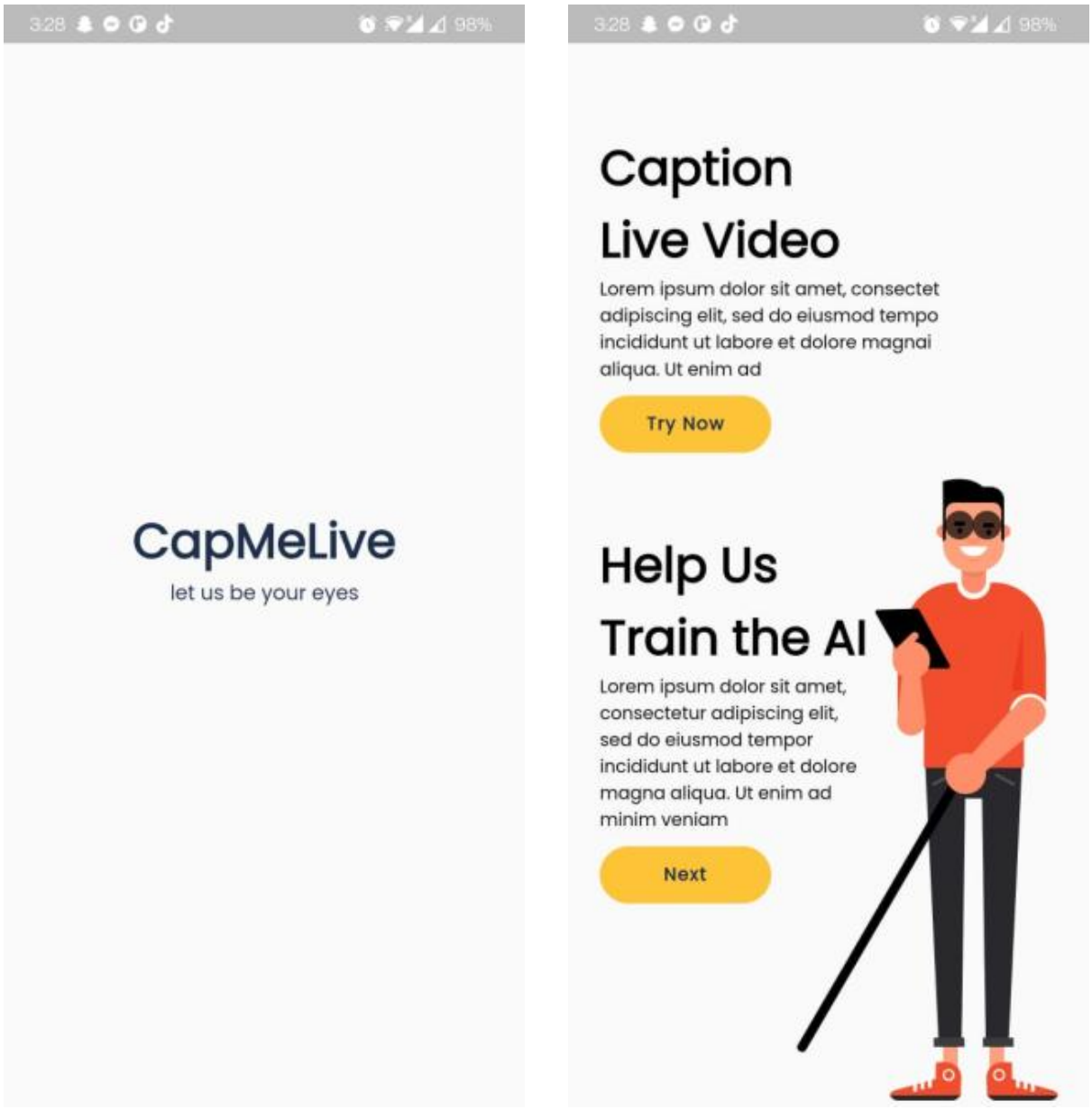

Figure 8.1: Splash Screen and Dashboard

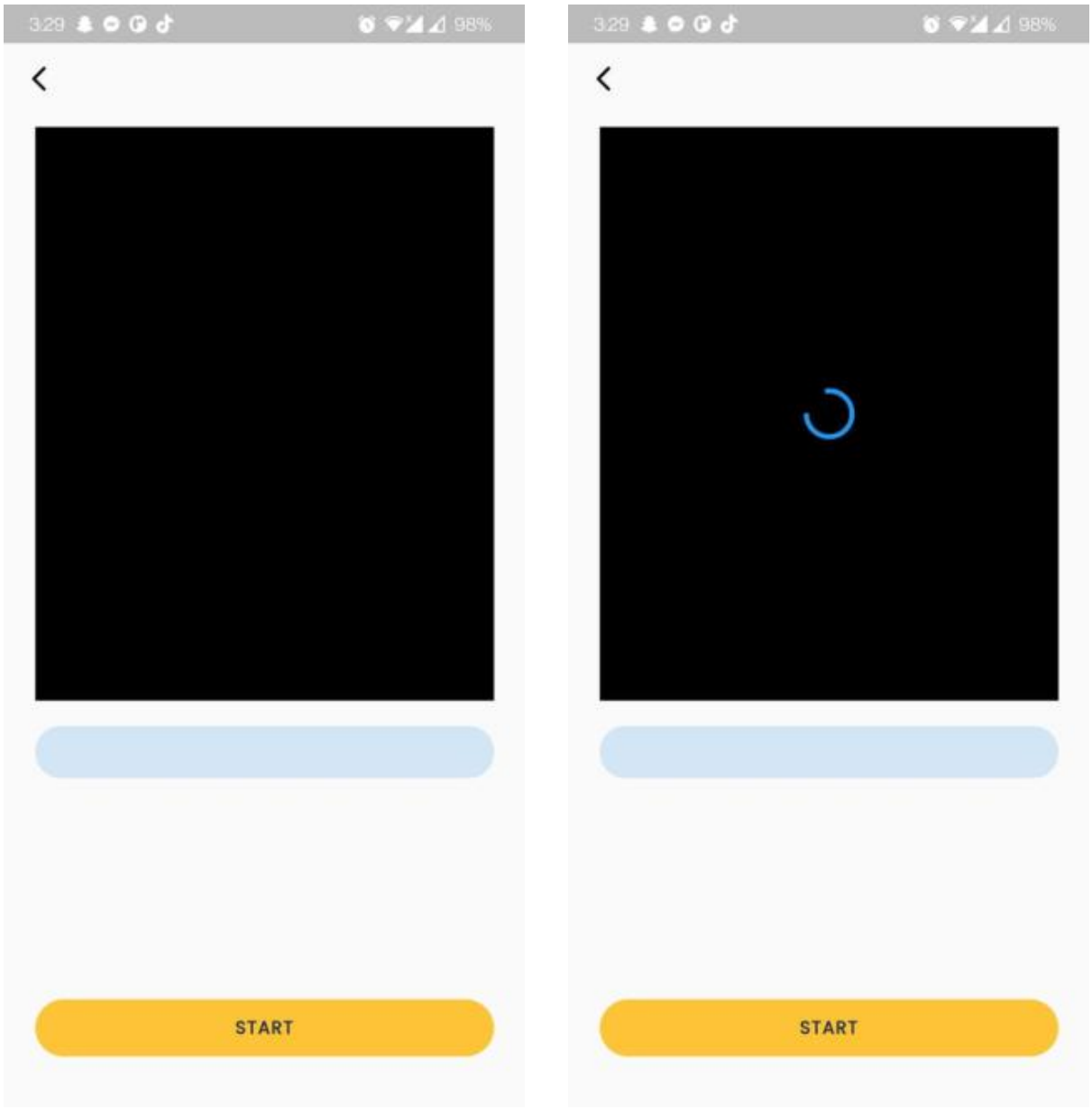

Figure 8.2: Start camera and waiting for connection

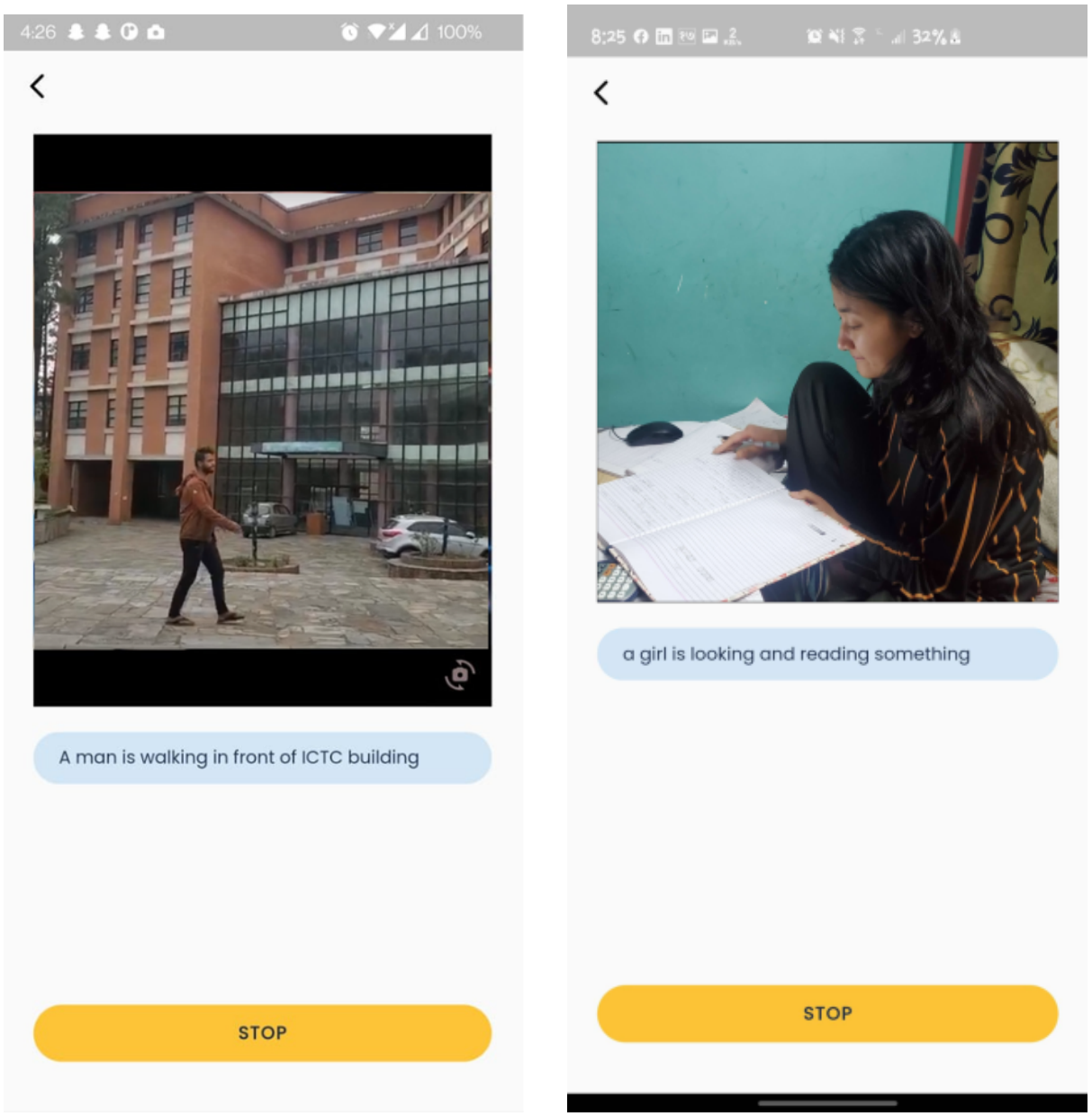

Figure 8.3: Result screenshots

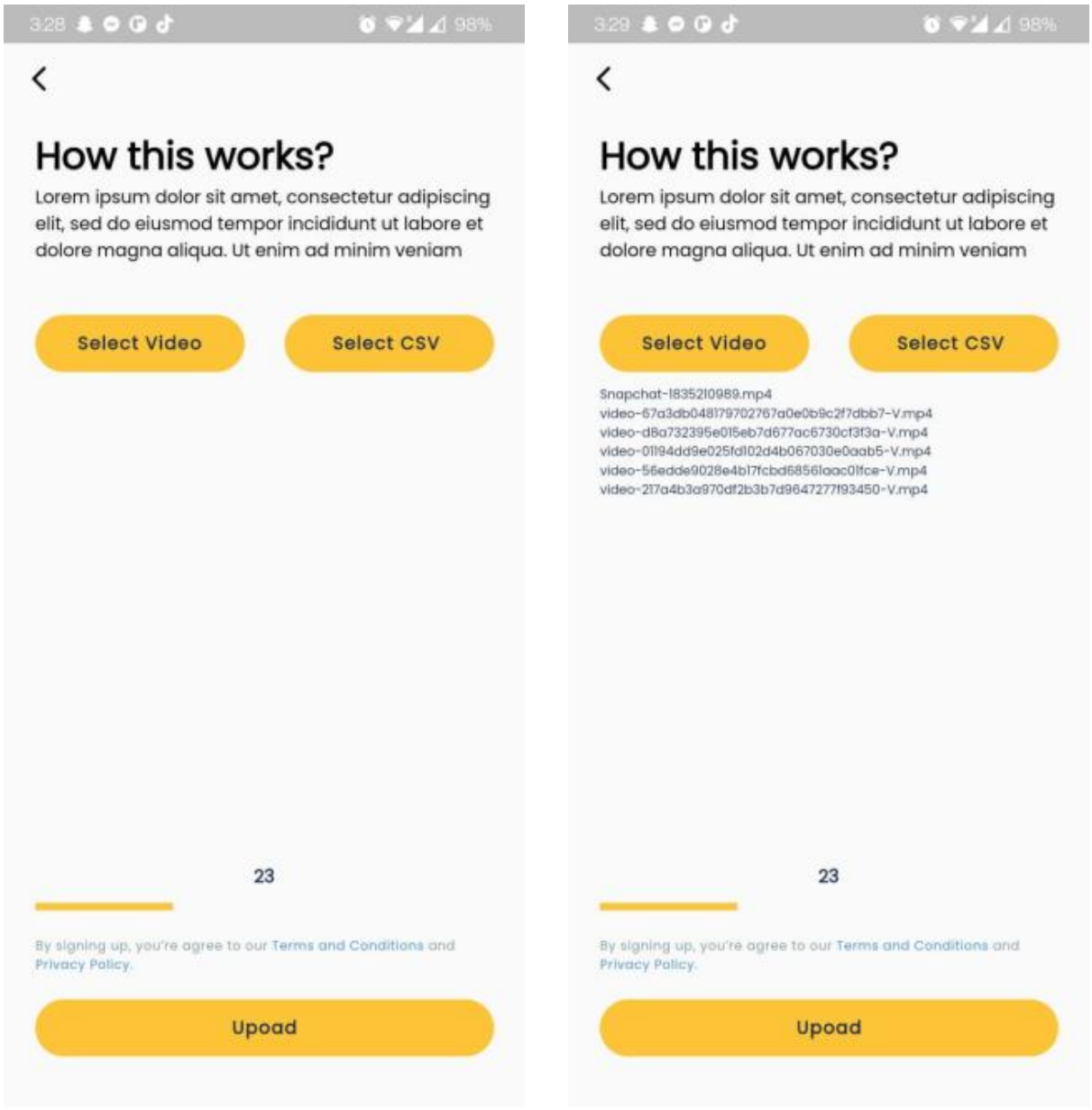

Figure 8.4: uploading videos