TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

SAGARMATHA ENGINEERING COLLEGE

A

PROJECT REPORT

ON

SMART AI SPECTACLE

BY

GOKUL SUBEDI 36404

KRISH GURUNG 36405

SAMIR BHATTARAI 36414

A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR IN ELECTRONICS, COMMUNICATION AND INFORMATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

SANEPA, LALITPUR, NEPAL

April, 2025

SMART AI SPECTACLE

BY

GOKUL SUBEDI 36404

KRISH GURUNG 36405

SAMIR BHATTARAI 36414

Project Supervisor

Er. Bharat Bhatta

Lecturer

A project submitted to the Department of Electronics and Computer Engineering in partial fulfilment of the requirements for the degree of Bachelor in Electronics, Communication and Information Engineering

Department of Electronics and Computer Engineering

Institute of Engineering, Sagarmatha Engineering College

Tribhuvan University Affiliate

Sanepa, Lalitpur, Nepal

COPYRIGHT ©

DECLARATION

We hereby declare that the report of the project work entitled "SMART AI SPECTACLE" which is being submitted to the Sagarmatha Engineering College, IOE, Tribhuvan University, in the partial fulfillment of the requirements for the award of the Degree of Bachelor in Electronics, Communication and Information Engineering, is a bonafide report of the work carried out by us. The material contained in this report has not been submitted to any University or Institution for the award of any degree.

GOKUL SUBEDI 36404
KRISH GURUNG 36405
SAMIR BHATTARAI 36414

CERTIFICATE OF APPROVALS

The undersigned certify that they have read and recommended to the Department of Electronics and Computer Engineering for acceptance, a project entitled **"SMART AI SPECTACLE"**, submitted by **Gokul Subedi, Krish Gurung, Samir Bhattarai** in partial fulfillment of the requirement for the degree of "Bachelor in Electronics, Communication and Information Engineering".

.....

Supervisor: Er. Bharat Bhatta,
Department of Electronics and Computer Engineering,
Sagarmatha Engineering College

.....

External Examiner: Prof. Dr. Ram Krishna Maharjan
Professor
Pulchowk Campus

DEPARTMENTAL ACCEPTANCE

The project work entitled **"SMART AI SPECTACLE"**, submitted by **Gokul Subedi, Krish Gurung, Samir Bhattarai** in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering in Electronics, Communication and Information Engineering"** has been accepted as a bonafide record of work independently carried out by team in the department.

.....

Er. Bharat Bhatta

Head of Department

Department of Electronics and Computer Engineering,

Sagarmatha Engineering College,

Tribhuvan University Affiliate,

Sanepa, Lalitpur

Nepal.

ACKNOWLEDGMENT

First of all, we would like to express our deepest gratitude to **Er. Bharat Bhatta**, our supervisor, for helping us with our project. His support, guidance, and encouragement made it possible for us to complete our work, write our report, and find the resources we needed.

We sincerely thank our our project coordinator, **Er. Bipin Thapa Magar**, **Er. Baikuntha Acharya**, along with our Head of Department, **Er. Bharat Bhatta**, for their continuous support, valuable advice, and guidance throughout our project. We are also grateful to the Department of Electronics and Computer Engineering for providing financial support and resources that helped turn our ideas into reality. Additionally, we appreciate Sagarmatha Engineering College for creating a great learning environment and encouraging academic excellence, which has contributed to our growth.

Our sincere appreciation goes to our colleagues and families for their help in data collection, valuable discussions, feedback, and encouragement throughout the research process.

ABSTRACT

Visually impaired individuals face challenges in perceiving their surroundings, recognizing people, and handling financial transactions. Existing assistive technologies often suffer from bulky hardware or limited functionality, making daily interactions difficult. To address this, we developed the Smart AI Spectacle, a wearable device that integrates computer vision and deep learning to provide real-time scene description, currency identification, and face recognition. The system utilizes a ResNet50-based encoder and an LSTM decoder for image captioning, enabling natural language descriptions of the environment. A deep learning classifier accurately identifies currency, while a face recognition model matches individuals using embedding-based comparisons. These tasks are executed in real time via an on-device AI unit, ensuring fast and efficient responses. The hardware setup includes an ESP32 camera for image capture, a PC for processing, and audio feedback for user interaction. This project aims to enhance accessibility for visually impaired users by providing a seamless and context-aware assistive experience, allowing them to navigate environments, recognize individuals, and handle transactions with confidence.

Keywords: AI, ResNet50, Artificial Intelligence, Computer Vision, Accessibility, Image Captioning, Smart Spectacles, Assistive Technology, ESP32, LSTM.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CNN	Convolution Neural Network
GNMT	Google's Neural Machine Translation
GRU	Gated Recurrent Unit
LSTM	Long Short Term Memory
ML	Machine Learning
NER	Named Entity Recognition
NLP	Natural Language Processing
POS	Part-Of-Speech
ReLU	Rectified Linear Unit
ResNet	Residual Networks
ROI	Region of Interest
RPN	Region Proposal Networks
TTS	Text to Speech
VGG	Visual Geometry Group

CHAPTER 1

INTRODUCTION

1.1 Background

Blindness and visual impairment are significant global health challenges, affecting millions of people worldwide. According to the World Health Organization (WHO), Globally, at least 2.2 billion people have a near or distance vision impairment. From data of 18 April 2012 – Using the most up-to-date studies, WHO estimates that the number of people with visual impairment (presenting vision) is 285 million (65% of whom are aged over 50 years). Of these, 246 million have low vision (63% over 50) and 39 million are estimated to be blind (82% over 50)[1]. For these individuals, recognizing objects, environmental awareness, face recognition, and currency identification pose daily challenges that can severely impact their quality of life and independence.

According to reports from various NGOs in Nepal, as well as research centers and hospital websites, it is estimated that 0.84% of Nepalese are blind, while an additional 1.7% are visually impaired. In numbers, around 252,000 individuals are blind, and approximately 510,000 have some form of visual impairment[2]. This data is outdated, and with the increasing screen time and exposure to various environmental radiations, the actual numbers are likely much higher today, affecting millions of lives and their ability to live independently. The loss of potential contributions from young individuals and those socially and mentally affected represents a significant setback to global innovation and development. In most cases, visual impairments are incurable. However, by developing innovative solutions that help these individuals perceive the world in a completely new way, their quality of life can be significantly improved.

This project aims to develop smart spectacles equipped with cameras and speakers, utilizing AI algorithms to provide real-time auditory feedback about the user's surroundings. The device is designed to help blind individuals gain a more intuitive and detailed understanding of their environment, enhancing their ability to navigate and interact with the world.

Additionally, the device is being developed to be lightweight, energy-efficient, and easy to use, offering versatile and reliable functionality. This innovation aims to significantly improve the lives of blind and visually impaired individuals.

1.2 Problem Definition

Blindness and visual impairment create major challenges for individuals, limiting their independence and affecting their daily lives. Many visually impaired people struggle with navigation, object recognition, and social interactions, making them highly dependent on others. The lack of accessible and intuitive assistive technologies further exacerbates these difficulties.

1.2.1 Challenges with Existing Assistive Technologies

Existing assistive technologies such as canes, guide dogs, and specialized mobile applications provide some level of aid but have notable limitations:

1. Canes and Guide Dogs: These aids primarily assist with obstacle detection and navigation. While useful, they do not provide comprehensive information about the surroundings, such as identifying specific objects, recognizing people, or understanding contextual details of the environment.

2. Mobile Applications: These can offer detailed environmental information but require continuous user interaction, which can be cumbersome and distracting. Users must frequently handle their devices and navigate through app interfaces, undermining the practicality and convenience of these solutions.

3. Wearable Devices: Although some advanced wearable devices exist, they often suffer from issues such as excessive heat generation, considerable weight, and high costs. These factors make them uncomfortable for prolonged use and inaccessible to many users due to their price.

1.2.2 Specific Issues in Nepal

As discussed in Section 1.1, visual impairment is a significant concern in Nepal, affecting a considerable portion of the population. Beyond these statistics, Nepal faces unique challenges in addressing visual impairment, including limited access to eye care services, inadequate awareness, and geographic barriers that hinder early diagnosis and treatment. Additionally, increasing screen time and environmental factors may further contribute to worsening conditions, emphasizing the need for effective and accessible solutions.

1.2.3 Impact of Visual Impairment

The challenges faced by visually impaired individuals include:

a) Object Recognition: Difficulty in identifying everyday objects, which can hinder basic tasks and activities.

b) Environmental Awareness: Limited understanding of their surroundings, leading to reduced mobility and increased risk of accidents.

c) Face Recognition: Challenges in recognizing people, affecting social interactions and personal relationships.

d) Currency Identification: Difficulty in distinguishing between different denominations, complicating financial transactions.

These challenges result in a substantial loss of potential contributions from visually impaired individuals, impacting global innovation and development. The social and mental effects further exacerbate their isolation and dependence on others.

1.2.4 Need for an Innovative Solution

Current solutions primarily focus on obstacle detection and basic navigation, leaving a significant gap in delivering detailed environmental information. The need for an innovative, user-friendly, and affordable solution that provides comprehensive real-time environmental descriptions is evident. Such a solution should enhance the independence

and quality of life for visually impaired individuals, allowing them to navigate and interact with their surroundings more effectively.

This project aims to address these gaps by developing a smart spectacle equipped with cameras and speakers that uses AI algorithms to deliver real-time auditory feedback. The goal is to create a lightweight, low-power and easy-to-use device that significantly improves the lives of blind and visually impaired individuals, fostering greater independence and enhancing their ability to perceive the world around them.

1.3 Objectives

The objective of this project is:

- To design a spectacle that can recognize face, identify Nepalese currency, and describe the scene user is facing toward.

1.4 Features

The features of the project are as follows.

a) Detect Nepalese currency.

b) Recognize known faces.

c) Provide a general description of scene user is facing toward.

d) Provide audio feedback for the user of the described scene or identified information

1.5 Feasibility

1.5.1 Technical Feasibility

The development of smart compact spectacles is technically feasible due to significant advances in hardware and software technologies. High-resolution, compact cameras are readily available, and processor which are available in PC provide the necessary computational power in a small, energy-efficient form factor. These components are well-suited for the integration into our devices.

In terms of software, machine learning frameworks such as TensorFlow and PyTorch, along with image processing libraries like OpenCV, enable the creation of sophisticated AI models for real-time object detection and scene description ensuring that our device can provide comprehensive real-time information to users.

Modern smartphones are able to communicate with these electronic equipment via wireless means making these device technically more feasible to create compact and lightweight spectacle, providing some extensive computation be done by mobile phone if necessary.

1.5.2 Operational Feasibility

Collaborations with NGO and INGO and other organizations that support blind individuals will provide critical insights and testing opportunities, ensuring that the product is compatible for fulfill the user needs.

1.6 System Requirements

The system requirements for our project are categorized into software and hardware requirements, each essential for the development and functionality of the intelligent system for visually impaired individuals.

1.6.1 Software Requirements

The software requirements for our project include:

(a) Python and its Libraries: Used for developing and deploying deep learning models. Essential libraries include TensorFlow, Keras, OpenCV, and NumPy.

(b) C and C++: Employed for performance-critical components and hardware interfacing, providing low-level control and efficiency specially for the hardware responsible for listening and interacting with user and core system.

(c) Code Editor: Any code editor capable of editing python script for deep learning model development, as well as C and C++ for hardware programming will be ideal for our use case. Ideally, Visual Studio Code and Arduino IDE were choose as for Python scripting and Hardware programming respectively.

1.6.2 Hardware Requirements

The hardware requirements for our project are:

1. ESP-32 Cam Module: To capture real-time images and transmit them wirelessly to the processor along with the user's selected model.
2. Processor: The main processing unit, managing software and hardware interactions.
3. Button: To take user input and toggles the hardware module on or off as needed.
4. Buck Converter: To regulate voltage provided to the ESP-32 Cam module.
5. Wire and connectors: Wires like Jumper wire and ribbon cable for power transmission and user interaction with ESP-32 Cam module.
6. Eye-wear: To mount hardware component for rigidity.

1.6.3 Functional Requirements

(a) Image Capture and Processing:

- The system should be able to capture images and perform real-time face recognition.
- The system should be able to identify currency.
- The system should be able to describe environment around the user.

(b) Audio Feedback:

- The system must convert all detected information into clear and understandable speech.

(c) User Interaction:

- The system should be functionally controllable with buttons.

(d) Software Updates:

- The system should be capable of updating the software whenever an update is available.

(e) Data Storage and Security:

- The system should be capable of storing relevant information for further use and safety.
- The system should be secure and operable with minimum risk.

(f) Performance:

- The system should be highly accurate and reliable.
- The system should be able to process the request and provide the information with very less latency (less then 2 second).
- The system should be capable of operating for at least 8 hours on single charge.
- The system should be capable of going to sleep and wake on interrupt to preserve the power.

(g) Connectivity:

- The system should support Bluetooth for connectivity with smartphones or other devices for strong functionality.
- The system should enable updates and data synchronization wirelessly.

(h) Durability and Environment Adaptability:

- The system should be durable and capable of operating under various environmental conditions, including different lighting and weather.

1.6.4 Non-Technical Feasibility

(a) Market Feasibility:

- Identify the primary users of the smart spectacles, specially the visually impaired individuals.
- The actual demand of the spectacles should be identified.
- Analyze market trends and potential for growth in the assistive technology sector.

(b) Economic Feasibility:

- Estimate the costs involved in the development, production, and distribution of the smart spectacles.
- Include expenses for hardware components, software development, labor, marketing, and distribution.
- Identify potential funding sources such as government grants, non-profit organizations, and crowdfunding.
- Develop a budget plan to manage the financial aspects of the project effectively.
- Determine an affordable pricing strategy for the end-users, considering the economic conditions of the target market.
- Explore options for subsidies or financial aid to make the device accessible to a broader audience.

(c) Operational Feasibility:

- Ensure the availability of required resources, such as hardware components and software tools.
- Develop a realistic timeline for the project's development, testing, and deployment phases.
- Set milestones and deadlines to track progress and ensure timely completion.

(d) Legal and Regulatory Feasibility:

- Ensure the device complies with relevant regulations and standards for electronic devices and assistive technologies.
- Address privacy and data protection laws to secure user data and maintain user trust.
- Protect the intellectual property rights of the developed technology through patents and trademarks.
- Conduct a freedom-to-operate analysis to avoid potential legal issues.

(e) Social Feasibility:

- Conduct usability testing to refine the design and functionality based on user input.
- Assess the potential social impact of the project on the lives of visually impaired individuals and their communities.
- Highlight the benefits of increased independence, improved quality of life, and enhanced social inclusion.
- Consider cultural factors that may influence the acceptance and use of the device.
- Adapt the device and its features to align with the cultural context of the target market.

CHAPTER 2

LITERATURE REVIEW

In the paper published by Bogdan Mocanu, Ruxandra Tapu and Titus Zaharia titled "DEEP-SEE FACE: A Mobile Face recognization System Dedicated to Visually Impaired People", they focus on visual impairment (VI), which affects over 285 million people worldwide, including 39 million who are blind and 246 million with low vision. The World Health Organization estimates that by 2020, these numbers will significantly increase. Visually impaired individuals typically rely on traditional aids like white canes and guide dogs to navigate their daily lives. The white cane is particularly favored due to its ease of use, affordability, and widespread acceptance among the blind community. However, the white cane has limitations in diverse urban environments, as it cannot provide information about the danger level of obstacles or recognize people in the vicinity. This lack of information forces VI individuals to stick to familiar paths and guess the identity of people they encounter, often interrupting conversations to identify those present.

To address these challenges, they introduce DEEP-SEE FACE, an innovative assistive device that uses computer vision algorithms and an offline-trained deep convolutional neural network (CNN) for face recognition. DEEP-SEE FACE extends the previously proposed DEEP-SEE architecture by adding a face recognition module. This system can identify familiar faces in real-time from video streams, helping visually impaired users navigate both indoor and outdoor environments and recognize people in social and media contexts. The hardware setup includes a smartphone for video acquisition, a lightweight processing unit with an Nvidia GPU, and bone conduction headphones, making the system portable, wearable, and affordable for a wide range of visually impaired users.

The face detection component of DEEP-SEE FACE uses the Faster R-CNN algorithm with Region Proposal Networks (RPN) to identify faces in video frames, trained on the VGG model pre-trained on ImageNet. This is followed by a face tracking module based on the ATLAS algorithm, which is optimized for tracking multiple faces under dynamic conditions. For face recognition, the system employs a CNN, specifically the VGG16

architecture, to extract features from each detected face. These features are then processed to create a compact, global face descriptor that adapts to varying image qualities and conditions through a learning-based weight adaptation scheme. This enhances recognition accuracy despite challenges like motion blur and occlusions. The system provides acoustic feedback to the user via bone conduction headphones, informing them of the presence and identity of recognized individuals, thereby improving social interactions and media consumption for visually impaired users [3].

In the paper published by Ahmed Yousry, Mohamed Taha and Mazen Selim titled "Currency recognition System for Blind people using ORB Algorithm", they propose a mobile system for currency recognition capable of identifying Egyptian currency notes from various perspectives and scales. Recognizing currency in diverse environments presents a complex challenge due to the influence of numerous uncontrolled conditions on image quality. To address this, they develop a smartphone application designed to identify currencies, even when they are partially visible, folded, wrinkled, or worn due to usage.

The proposed system can recognize Egyptian currency denominations ranging from 0.5 to 200 Egyptian pounds. It utilizes the ORB Algorithm, known for its speed and efficiency in feature detection and description. ORB, or Oriented FAST and Rotated BRIEF Algorithm, is particularly advantageous for its robustness to illumination, rotation, and scaling. Previous research has demonstrated the versatility of the ORB Algorithm in various applications, including detecting moving objects in dynamic backgrounds, identifying copy-move attacks in digital images, and automatic landmark recognition in aerial imagery.

Related works in the field of currency recognition include methods based on texture, color, and shape analysis, as well as those utilizing scale-invariant feature transform (SIFT) algorithms. Various approaches have been developed for different currencies, such as Jordanian, Saudi Arabian, and Mexican banknotes, each with their own recognition accuracies and computational efficiencies. One notable method achieved 100% classification accuracy for normal banknotes and 99.8% accuracy for defiled banknotes by leveraging size information and correlation matching of multiple templates.

The proposed method consists of two phases: offline and online. In the offline phase, a dataset is constructed from a collection of Egyptian currency images. In the online phase,

the system detects and recognizes unknown currency images captured by the device's camera. This phase involves pre-processing techniques to remove noise, segmentation to extract the currency from the background, application of the ORB Algorithm for feature extraction, and matching the results with the dataset. The output of the system is delivered as a voice informing the user of the currency value, making it accessible and user-friendly [4].

CHAPTER 3

RELATED THEORY

As human needs and lifestyles evolve, electronic devices are AI and Machine Learning are enhancing the usability and reliability of everyday devices, making complex decisions easier. In this project, they play a key role in improving accessibility for visually impaired individuals by interpreting and describing the visual world. Through advanced algorithms, the system provides real-time assistance and information to users. In this project, AI and ML are pivotal for enhancing accessibility and independence for visually impaired individuals.

Deep learning, a subset of machine learning, uses multi-layered neural networks to learn hierarchical representations of data, making it ideal for tasks like image and speech recognition. Computer vision methods, such as object detection and face recognition, analyze and interpret visual data, while natural language processing (NLP) enables systems to convert visual information into speech and understand user commands through speech recognition. These technologies enhance accessibility for visually impaired users by providing auditory feedback and hands-free operation.

During the project development phase, we used specific machine learning frameworks to ensure functionality. These frameworks were instrumental in optimizing performance and enhancing accuracy of the project. They are explained in detail below:

3.1 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a deep learning algorithm designed to process visual data by extracting features through multiple layers, making images more manageable for predictions without losing key details. The CNN consists of several layers that progressively capture higher-level features from raw image data.

3.1.1 How Do CNNs Work?

CNNs use three primary layers to process images:

Convolutional Layer

The convolutional layer applies filters (kernels) to the input image to detect specific features. It generates a feature map by sliding the filter across the image, highlighting essential patterns like edges or textures.

Pooling Layer

Pooling layers reduce the dimensionality of the data, helping to decrease computation and prevent overfitting. It applies an aggregation function (max or average pooling) to the input, retaining significant features while reducing size.

Fully-Connected Layer

In the fully-connected layer, each node is connected to every node in the previous layer. This layer performs the final classification based on the features extracted from previous layers, ensuring an accurate decision by integrating all the extracted information.

These three types of layers work together in a hierarchical manner to enable CNNs to effectively process and analyze complex visual data, making them essential tools in various applications such as image recognition, object detection, and scene understanding [5].

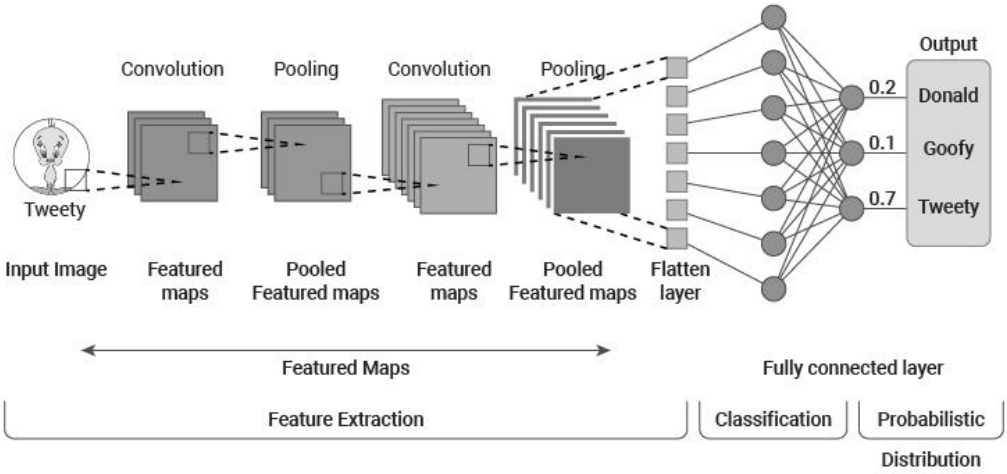

Figure 3.1: Architecture of Convolutional Neural Network(CNN)

[6]

3.2 Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is a class of artificial neural networks designed to recognize patterns in sequences of data, such as time series or natural language. Unlike traditional feedforward neural networks, RNNs have connections that form directed cycles, allowing them to maintain a memory of previous inputs in their hidden states. This memory capability makes RNNs well-suited for tasks where the order and context of data are important.

3.2.1 How Do Recurrent Neural Networks Work?

RNNs operate through a series of layers where the output from a previous step is fed back into the network, creating a loop that allows information to persist over time. The primary components of an RNN are:

Recurrent Layer

The recurrent layer is the core of an RNN, where the network's temporal dynamics are captured. It consists of a set of neurons that maintain a hidden state representing

information from previous time steps. At each time step, the network processes the current input and updates its hidden state based on both the current input and the previous hidden state. This update is typically performed using the following equations:

$$h_t = f(W_h \cdot [h_{t-1}, x_t] + b_h) \tag{3.1}$$

where h_t is the hidden state at time t, W_h is the weight matrix, x_t is the input at time t, and b_h is the bias term. The function f is usually a non-linear activation function such as tanh or ReLU.

Output Layer

The output layer in an RNN produces the final prediction based on the hidden states. This layer transforms the hidden state into a desired output format, such as a probability distribution for classification tasks or a sequence of values for regression tasks. The output y_t at time t is computed as:

$$y_t = g(W_y \cdot h_t + b_y) \tag{3.2}$$

where W_y is the weight matrix for the output layer, b_y is the bias term, and g is the activation function used to generate the output.

Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU)

While standard RNNs can suffer from issues like vanishing and exploding gradients, specialized variants such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU) address these problems. Both LSTM and GRU architectures introduce gating mechanisms to better manage the flow of information and maintain long-term dependencies. For example, LSTM networks use forget, input, and output gates to regulate the information stored in the cell state, while GRUs use update and reset gates to control the hidden state updates.

These components enable RNNs to capture temporal dependencies and make predictions based on sequential data. As a result, RNNs are widely used in applications such as

natural language processing, time series forecasting, and speech recognition.

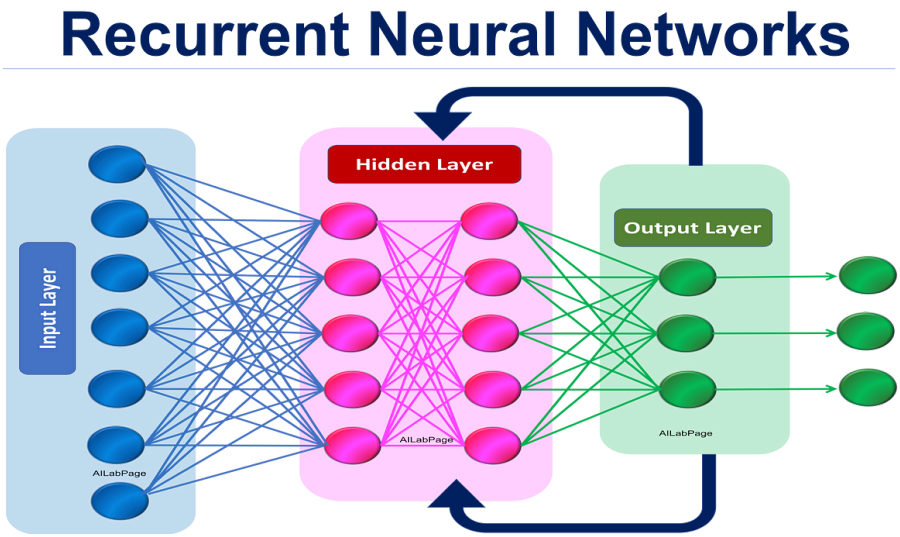

Figure 3.2: Architecture of a Recurrent Neural Network (RNN) [7]

3.3 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a branch of artificial intelligence that focuses on the interaction between computers and humans through natural language. The goal of NLP is to enable computers to understand, interpret, and generate human languages in a way that is both meaningful and useful. NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. This integration allows machines to comprehend the nuances of language, including context, sentiment, and intent.

3.3.1 How Does Natural Language Processing Work?

NLP systems operate through several key stages and techniques to process and understand natural language. The main components involved in NLP are:

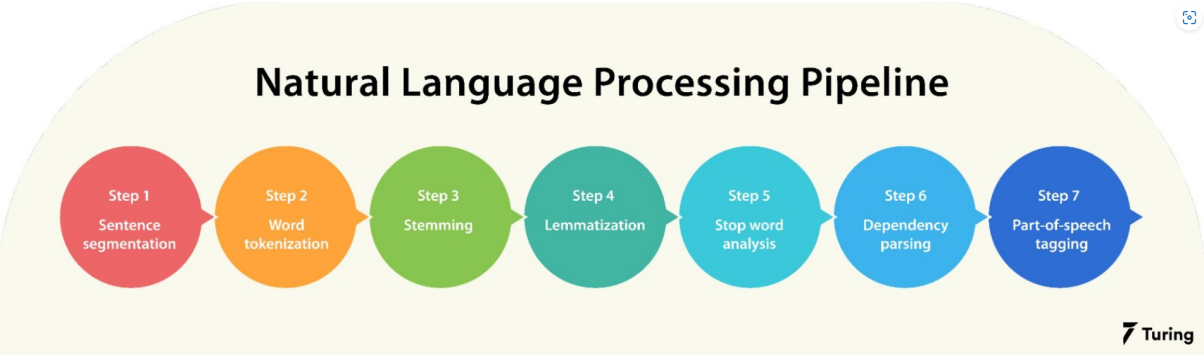

Figure 3.3: Flowchart of NLP Process

[8]

Tokenization

Tokenization is the process of breaking down a text into smaller units, such as words or phrases, known as tokens. This step is crucial because it simplifies the text into manageable pieces for further analysis. Tokenization can be done at the word level (splitting a sentence into individual words) or at the sentence level (splitting a paragraph into individual sentences). It is the first step in most NLP pipelines, laying the groundwork for subsequent processing.

Part-of-Speech Tagging

Part-of-speech (POS) tagging involves labeling each token with its appropriate part of speech, such as nouns, verbs, adjectives, and adverbs. This process helps in understanding the syntactic structure of the text, which is essential for tasks like parsing, semantic analysis, and machine translation. POS tagging uses statistical models and algorithms to predict the correct tags based on the context of each token.

Machine Translation

Machine translation is the automatic translation of text from one language to another. It relies on advanced algorithms and deep learning models to understand the meaning of the source text and generate an accurate translation in the target language. Popular NLP models like Google's Neural Machine Translation (GNMT) and OpenAI's GPT are used to achieve high-quality translations.

Text Generation

Text generation involves creating coherent and contextually relevant text based on a given input. This can range from generating simple responses in chatbots to creating entire articles or stories. Advanced NLP models, such as GPT-3, leverage large datasets and deep learning techniques to generate human-like text that is contextually appropriate and linguistically accurate.

By integrating these components, NLP systems can perform a wide range of tasks that involve understanding and generating human language. This capability makes NLP an essential technology for applications like virtual assistants, automated customer support, language translation, and content creation [9].

3.4 ResNet Architectures

Residual Networks (ResNets) are a class of convolutional neural networks (CNNs) designed to mitigate the vanishing gradient problem in very deep networks. They achieve this by incorporating residual connections, which enable the training of deeper networks by allowing gradients to flow more easily through the network. In this section, we discuss ResNet-18 and ResNet-50, two variants used in different applications in our project.

3.4.1 ResNet-18

ResNet-18 is an 18-layer deep residual network. It is designed to address the vanishing gradient problem, making it easier to train deeper neural networks. The architecture of ResNet-18 includes residual blocks with skip connections that allow the network to learn residual mappings rather than direct mappings. This design helps preserve gradient flow during back-propagation, enhancing the training process.

Application: Money Classification

In this project, ResNet-18 is utilized for the task of money classification. Due to its relatively shallow depth compared to other ResNet variants, ResNet-18 provides a good balance between complexity and performance for tasks that do not require excessively

deep networks. Its architecture allows for efficient training and accurate classification of various denominations and types of currency.

3.4.2 ResNet-50

ResNet-50, a deeper variant with 50 layers, extends the concept of residual learning by employing bottleneck blocks. These blocks use three convolutional layers instead of the two used in ResNet-18, which helps reduce computational complexity while maintaining deep network benefits. The use of bottleneck blocks in ResNet-50 allows it to capture more complex features and patterns in data.

Application: Scene Description

For the task of scene description, ResNet-50 is employed due to its deeper architecture, which enables it to capture more nuanced and complex features in images. This depth is particularly useful for understanding and describing complex scenes with multiple objects and varying contexts. The advanced feature extraction capabilities of ResNet-50 contribute to more accurate and detailed scene descriptions.

3.4.3 How ResNet Works?

The key innovation in ResNet architecture is the use of residual (skip) connections. These connections bypass one or more layers and add the input directly to the output of the bypassed layers. This approach creates shortcut paths for the gradient to flow through, reducing the risk of vanishing gradients and facilitating the training of very deep networks. The residual connections effectively enable the network to learn the residual mapping, which improves the optimization process and overall performance.

Both ResNet-18 and ResNet-50 leverage these residual connections to enhance their learning capabilities and achieve high performance in their respective applications[10].

ResNet18 Architecture

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

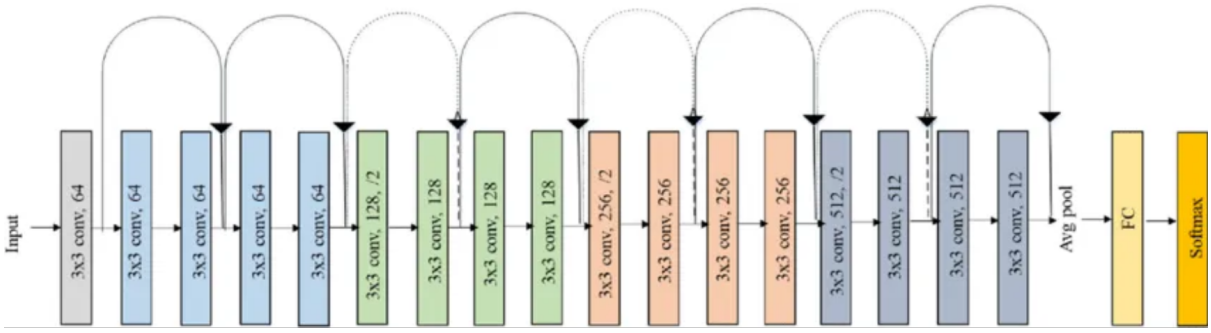

Figure 3.4: ResNet-18 Architecture

[11]

3.5 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) specifically designed to address the vanishing gradient problem that affects traditional RNNs. They are well-suited for sequence prediction problems due to their ability to capture long-term dependencies in sequential data.

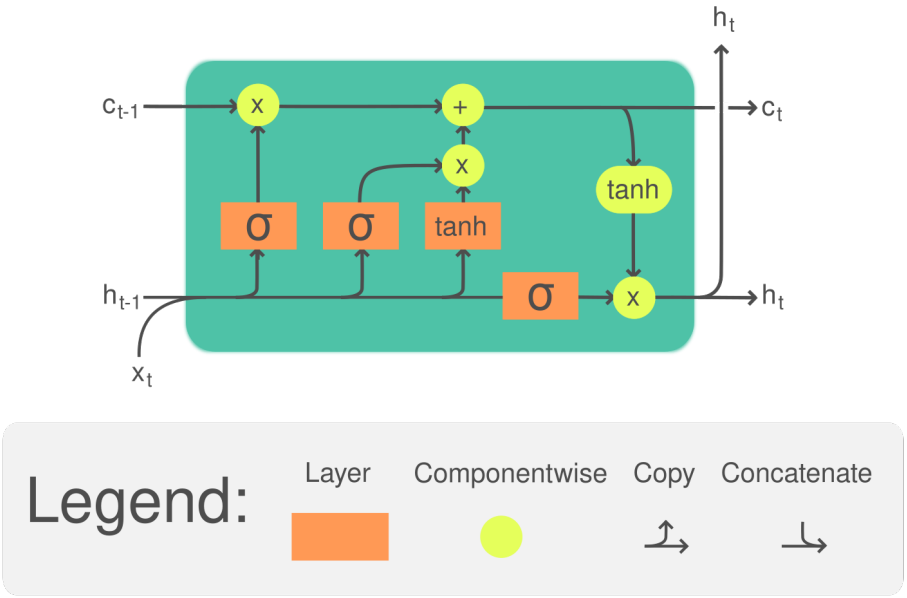

Figure 3.5: Long Short-Term Memory (LSTM) network architecture showing the key components: input gate, forget gate, and output gate.

[12]

LSTMs introduce memory cells that can retain information over long sequences. These cells are equipped with three main gates—input gate, forget gate, and output gate—that control the flow of information in and out of the cell.

3.5.1 Components of LSTM

Input Gate

- The input gate determines which information will be added to the cell state. It takes the current input x_t and the previous hidden state h_{t-1}, and outputs a value between 0 and 1 for each component of the cell state. A value of 1 means "completely keep this" and 0 means "completely discard this."

Forget Gate

- The forget gate decides what information from the cell state should be discarded. It consists of two parts:

a. A sigmoid layer that outputs values between 0 and 1 for each component of the cell state to indicate how much of the previous cell state should be forgotten.

b. A tanh layer that generates a vector of candidate values to be added to the cell state.

Output Gate

- The output gate determines the next hidden state h_t based on the updated cell state. It filters the cell state through a sigmoid activation function to decide which parts of the cell state should be output.

Applications of LSTM Networks

LSTM networks are highly effective for various sequential tasks, including:

- Time Series Prediction: Forecasting future values based on historical data, such as stock prices and weather conditions.
- Natural Language Processing (NLP): Tasks like language modeling, machine translation, and sentiment analysis, where context and sequential information are crucial.
- Speech Recognition: Converting spoken language into text by capturing temporal dependencies in audio signals.
- Video Analysis: Analyzing video sequences to detect actions and recognize objects over time.

By controlling and memorizing information over long sequences, LSTMs effectively mitigate the problems of vanishing and exploding gradients, enabling better training and more accurate predictions in sequential data tasks [13].

3.6 Image Captioning

Image Captioning is an example of deep learning on mixed data modalities (texts and images). It is the task of describing the content of an image in words. This task lies at the

intersection of computer vision and natural language processing. Image captioning adopts an encoder-decoder framework consisting of two principal components, a convolutional neural network (CNN) for image feature extraction and a recurrent neural network (RNN) for language caption generation.

Encoder-Decoder Framework

1. **Encoder (CNN):** The encoder is typically a convolutional neural network (CNN), such as ResNet, VGG, or Inception, which is used to extract high-level features from the input image. These features are a compact representation of the image, capturing essential aspects such as objects, scenes, and their relationships. The CNN processes the image and generates a fixed-size vector or a set of feature maps that encapsulate the visual information.

   In this project, the encoder employs ResNet-50, a widely used convolutional neural network (CNN), to extract meaningful features from the input image. ResNet-50, with its 50-layer deep structure, uses residual connections to overcome the vanishing gradient problem, ensuring efficient training and the ability to learn complex patterns. When an image is passed through ResNet-50, it is first resized and normalized to meet the network's input requirements, typically to a fixed size like $224 \times 224 \times 3$. This preprocessing ensures the image can be consistently interpreted.

   The network processes the image through a series of convolutional, pooling, and residual layers, extracting hierarchical features that range from low-level edges and textures in the early stages to high-level semantic concepts like object shapes and relationships in the deeper layers. At its final stage, ResNet-50 produces a global feature vector that encapsulates the image's most critical characteristics in a high-dimensional space. To make this feature vector compatible with the decoder, it is further passed through a linear layer that reduces its dimensionality, creating a fixed-size embedded representation that encapsulates the visual essence of the image. This feature vector is then used as the input for the decoder, enabling it to generate descriptive captions.

2. **Decoder (RNN):** The decoder is generally a recurrent neural network (RNN), such as a Long Short-Term Memory (LSTM) or a Gated Recurrent Unit (GRU),

which generates the image caption word by word. The RNN takes the encoded image features as its initial input and sequentially predicts the next word in the caption until it generates a complete sentence. The decoder needs to capture the temporal dependencies and structure of the language, ensuring the generated caption is grammatically correct and contextually appropriate [14].

In this project, the decoder utilizes a Long Short-Term Memory (LSTM) network, which excels in generating sequential data, such as textual descriptions. The LSTM begins with the embedded feature vector from the encoder, which serves as its initial hidden state. This initialization ensures that the decoder starts with a comprehensive understanding of the image's content. The caption generation process begins with a special <start> token, which is embedded into a vector and fed into the LSTM. The LSTM processes this token along with the hidden state, generating the probability distribution for the first word of the caption. The word with the highest probability is selected, forming the first word of the sentence.

The decoder then proceeds in a sequential manner, where the predicted word from the previous step (or the ground truth during training) is embedded and input into the LSTM at each subsequent step. The LSTM updates its hidden state with each new word, effectively capturing the context and maintaining coherence throughout the caption. This process continues until the model predicts a special <end> token, signaling the conclusion of the caption. The decoder is equipped with a fully connected linear layer that maps the LSTM's hidden state to the vocabulary space, enabling it to generate probabilities for each word in the vocabulary at every time step.

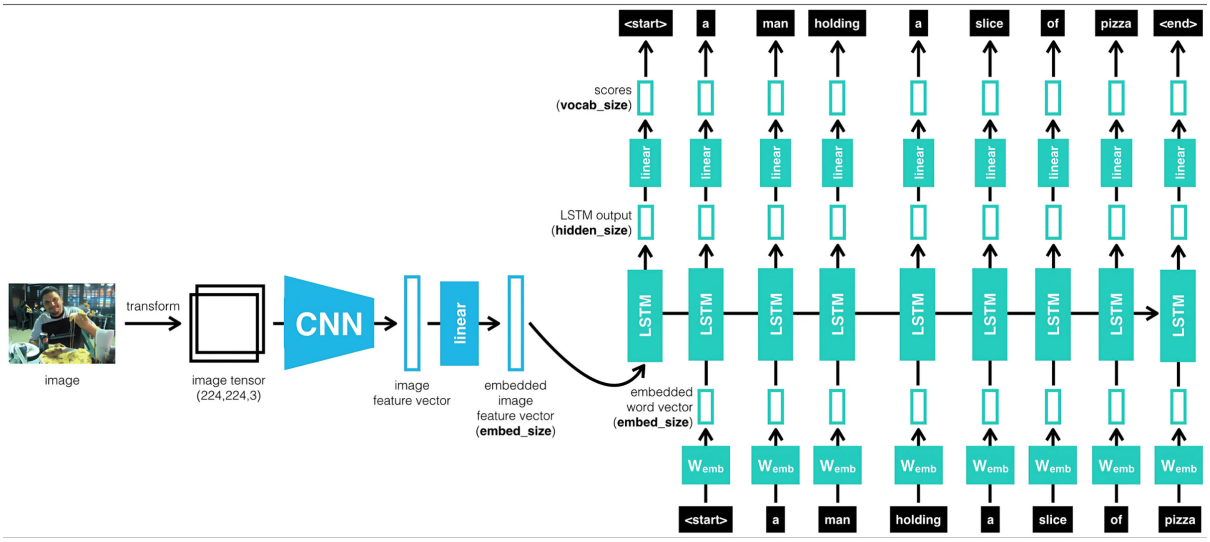

Figure 3.6: Encoder Decoder Framework

[15]

Figure 3.7: Examples of Image Caption

[16]

3.7 Hardware Description

The system integrates various hardware components for operations, based on user request. The primary components include a esp32 cam, PC processor, push buttons for user input, and an earphone for audio output. Below is a detailed description of each hardware component and its role in the system:

ESP32 Camera

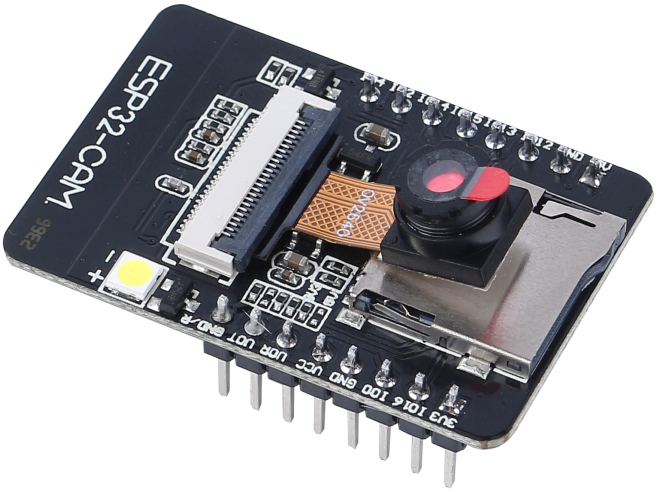

Figure 3.8: ESP32 cam
[17]

- The esp23 camera serves as the primary input device in the system.
- Its role is to capture video, which is then snipped at the rate of 12 to 20 frames per second to obtain real-world images for the system.
- Each snipped frame is then fed to the processor for further processing.

Buttons

Figure 3.9: Push Switch
[18]

- Three push buttons act as the control input to the system.

- Each button represents one of the following requests:
  - Face Recognition
  - Currency Identification
  - Scene Description
- The user can press one button at a time to make a request.

Laptop for processing

Figure 3.10: Laptop
[19]

- The laptop acts as the processing unit of the system.
- It is responsible for computing and processing all user requests.
- It receives input data from both the esp32 camera and the pressed push button flag.
- The processor from laptop processes the image data according to the specific command requested by the user through the button.
- It employs algorithms for:

- Face Recognition: Uses OpenCV and the haarcascade-frontalface-default model to crop the face of the person in the frame and feed it to the face recognition model. The model predicts the name of the person if known, or outputs "unknown" if the person is not recognized.
- Currency Identification: Directly processes the snipped frame through the currency identification model to identify the amount from among the seven currencies currently used in Nepal.
- Scene Description: Feeds the images directly to the image captioning model, which generates a caption of the image.

- After processing the data, it converts the output into a clear audio message using natural language processing algorithms.

Buck Converter

Figure 3.11: buckconverter
[20]

- A buck converter is a DC-DC power converter that steps down a higher input voltage to a lower output voltage efficiently. It uses a switch (like a transistor), a diode, an inductor, and a capacitor to regulate the output voltage, making it ideal for our battery-powered spectacle.

CHAPTER 4

METHODOLOGY

4.1 Software Development Life Cycle (SDLC)

The Agile Model is the most suitable SDLC for our "Smart AI Spectacle" project due to its flexibility, adaptability, and iterative nature. The project involves integrating multiple AI models (face recognition, scene captioning, and money classification) with hardware components such as a laptop processor and a camera. Agile supports continuous feedback, real-world testing, and iterative improvements, enabling refinements based on performance, usability, and hardware integration. This iterative approach ensures effective development of each model and their seamless integration into the final system.

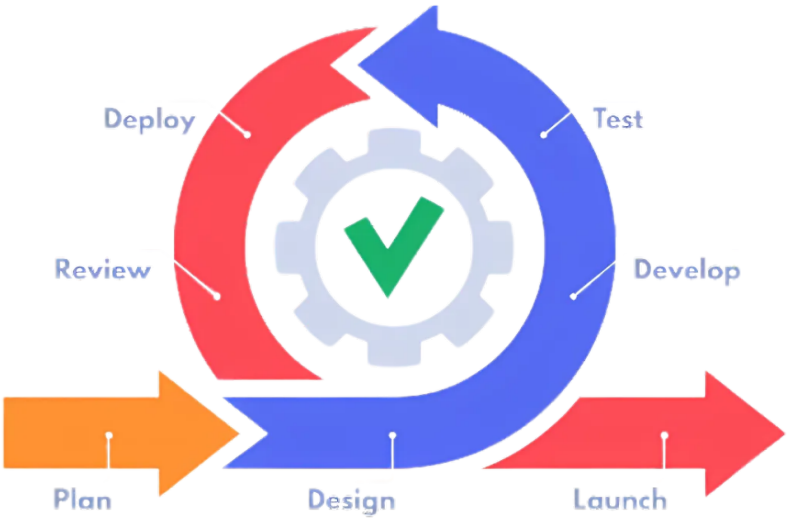

Figure 4.1: Agile software development model

[21]

4.2 System Block Diagram

The block diagram of our system is as follows:

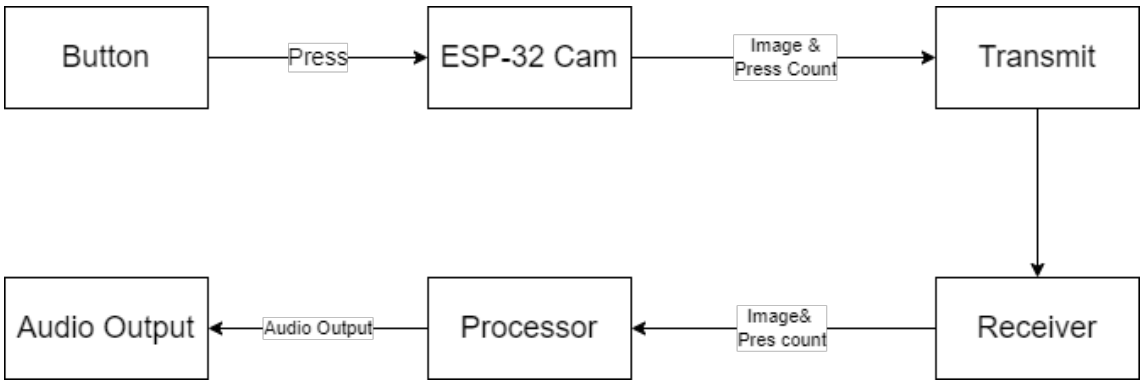

Figure 4.2: System Block Diagram

The system described in Section 4.2 integrates the various components to facilitate the spectacle working based on user request.

The overall system has, two sub-system, Sender system, facilitated by ESP-cam module, responsible for user interaction and a Receiver system facilitated by laptop, responsible for deep learning model operation, prediction and generation of output.

In the sender system, the camera and the button facilitate for the primary input device, where the button facilitate the user interaction with the system, and the camera for the image input.

When user press the button for specific number of time, the ESP transmits the current live image from the built-in camera as well as the button press count to the processing system currently known as Receiver system.

Now, the receiver system, upon receiving the image and press-count or the model choice, runs the specific model over the provided image and outputs the result.

This result is then converted to audio using readily available text-to-speech engine and played back to user using speaker driver available.

Here, the communication between transmitter and receiver system are done by using WIFI where sender system create hotspot for the WIFI and the receiver connects with this hotspot for seamless communication

4.3 System Flowchart

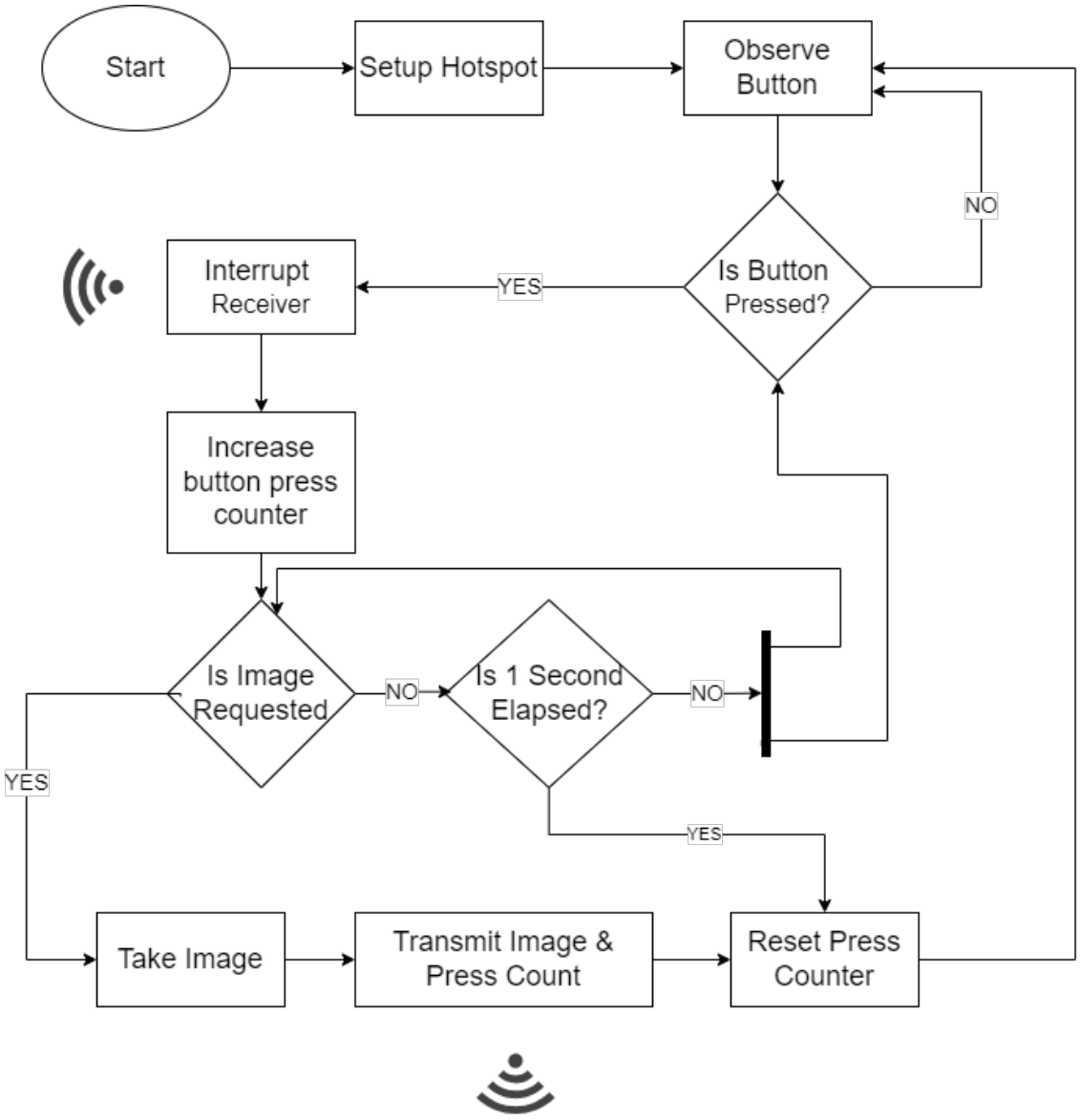

Figure 4.3: Transmitter System Flowchart

The chart of the transmitter system shown in figure 4.3 illustrate the sequential process of the transmitter system from start to end and the loop it makes. The system first setup the hotspot, where receiver can connect for seamless connection.

Once the connection is established, the system enters a continuous loop, monitoring for button presses. If user press the button. it now interrupt the receiver system indicating the user interaction. Then the system will wait for 0.9 second until the receiver request for the image and press count.

Every time the user press button. it increments the button press count, and loops through above mentioned loop.

If the receiver requests image, then it captures image and transmit it over the WIFI, with press count in the header. And finally reset the press count and go back on loop for observing the button.

And if the receiver didn't request for image within 1 second, it reset the press count and go back on button observation loop.

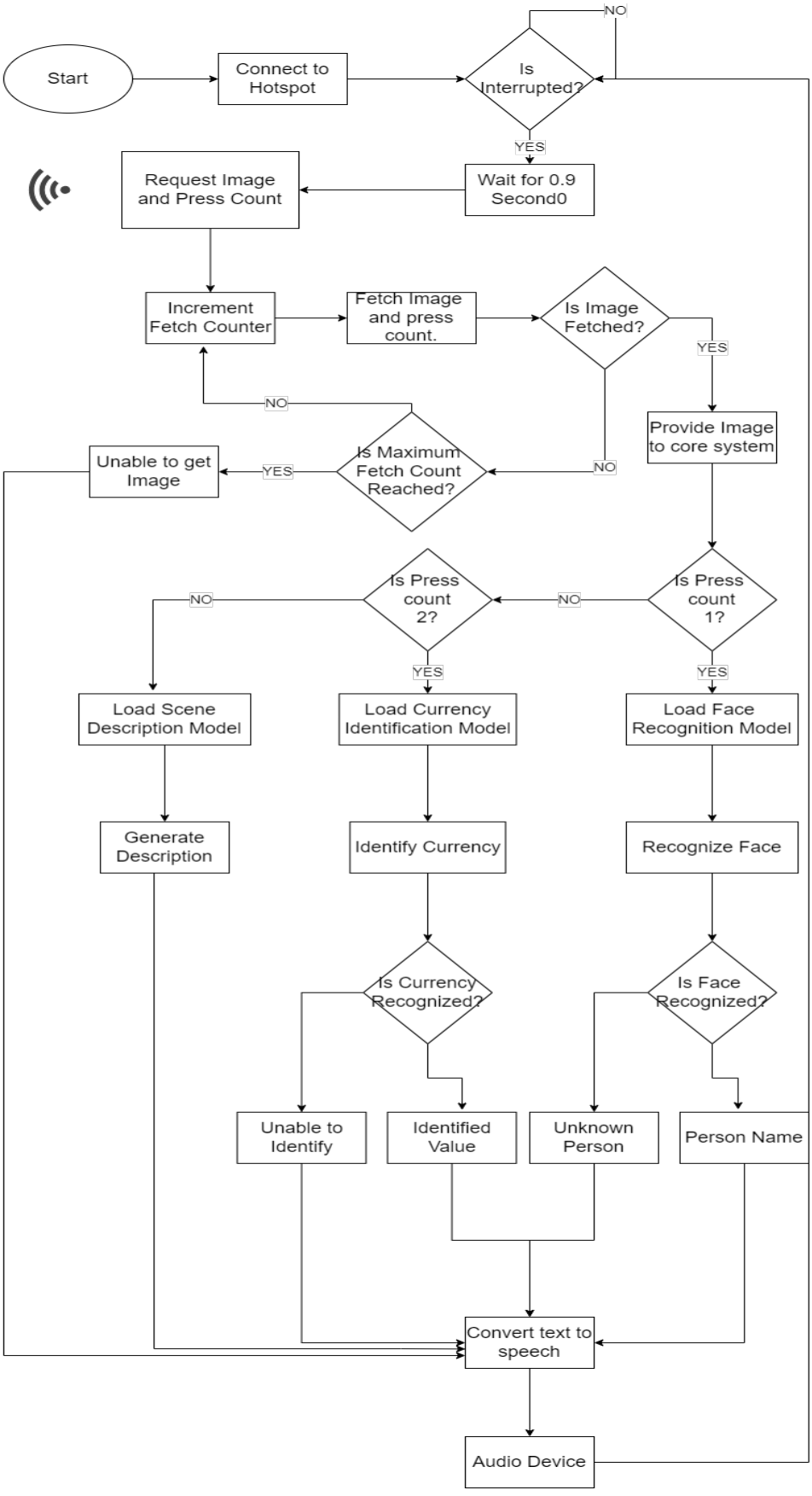

Figure 4.4: Receiver System Flowchart

The flow chart of the receiver system presented in figure 4.4 illustrates the sequential process of the receiver system from start to end and the loop it makes. The system begins by connecting to the hotspot created by the sender system.

Upon initialization the system will keep waiting for the sender for interrupt, virtually doing nothing.

When user press a button, the sender send a interrupt signal to the receiver through API call over the WIFI network they are connected to, indicating the button pressed by the user. Now both the sender and receiver waits for 0.9 seconds for user to provide necessary command.
Here by necessary command, the user can press button multiple time, based on what they want the system to do.

After 0.9 second, the receiver system request the image and the button press count with the sender. Now the sender system capture the image of current instance and send image to the receiver via a API call with press count in it's header.

Upon getting image, the receiver system decode the press count, convert the image into necessary format for image pre-processing and pass the image and press count to the master function.

The master function then take the press count and with switch case, run the appropriate model for the result user is requesting for.

If press count is 1:
Now the face recognition model will get executed, based on the explanation on face recognition model 4.6.2, this model performs the inference on pre-processed image and result an output.

If press count is 2:
Now the currency identification model will get executed, based on the explanation on this model 4.5.2, this model performs the inference on pre-processed image and result an output.

If press count more then 2:
The scene description model will get executed. And based on explanation of this model 4.7.2 Inference, this model generates a description using the pre-processed image and

result an output

Now finally the output from any given model is converted to audio using readily available text-to-speech engine in python then output to the speaker for the user.

4.4 Model and Dataset

4.4.1 Dataset

This system relies on a supervised machine learning algorithm, requiring a large dataset with diverse scenarios, lighting conditions, and well-structured labels and captions. To support this, project team members are actively engaged in the process of collecting and labeling images to meet the system's requirements, thereby enhancing the algorithm's ability to generalize across different real-world conditions.

- Image Captioning: Approximately 150,000 images, each annotated with five different captions.
- Money Identification: Approximately 25,500 images, with each currency amount stored in separate folders
- Face Recognition:Approximately 161 individuals with a minimum of 9 images and a maximum of 120 images per person.

Image Sizes:

- Images: Images were present in various sizes, including portrait (1080x1920 pixels), landscape (1920x1080 pixels), and square (1080x1080 pixels).

Data Count:

Table 4.1: Scene image form online source

S.No.	Data Label	data count
1	Scene Images	142,000

Table 4.2: Currency image form kaggle source

S.No.	Data Label	data count
1	5 Rs	1530
2	10 Rs	1630
3	20 Rs	1529
4	50 Rs	1530
5	100 Rs	1579
6	500 Rs	1509
7	1000 Rs	1529
8	unknown	1567

Table 4.3: Face image form kaggle source

S.No.	Data Label	data count
1	All images	1625

Custom Dataset:

In addition to the Kaggle dataset, we have collected our own images.

- Nepali Currency Images: Approximately 2000 images of each currency.
- Environment Images: Over 5,586 images from within the Kathmandu valley.
- Face Images: 81 person each having 10-40 images.

Image Sizes:

- Environment Images: Various sizes, including portrait (1080x1920 pixels), landscape (1920x1080 pixels), and square (1080x1080 pixels).
- Face and Currency Images: Standardized to 120x120 pixels.

Data Count:

Table 4.4: Scene image form custom source

S.No.	Data Label	data count
1	Scene Images	6,123

Table 4.5: Currency image form Custom source

S.No.	Data Label	data count
1	5 Rs	2130
2	10 Rs	2230
3	20 Rs	2129
4	50 Rs	2180
5	100 Rs	2176
6	500 Rs	2109
7	1000 Rs	2129
8	unknown	2000

Table 4.6: Face image form custom source

S.No.	Data Label	data count
1	All images	1370

4.5 Currency Identification model Architecture

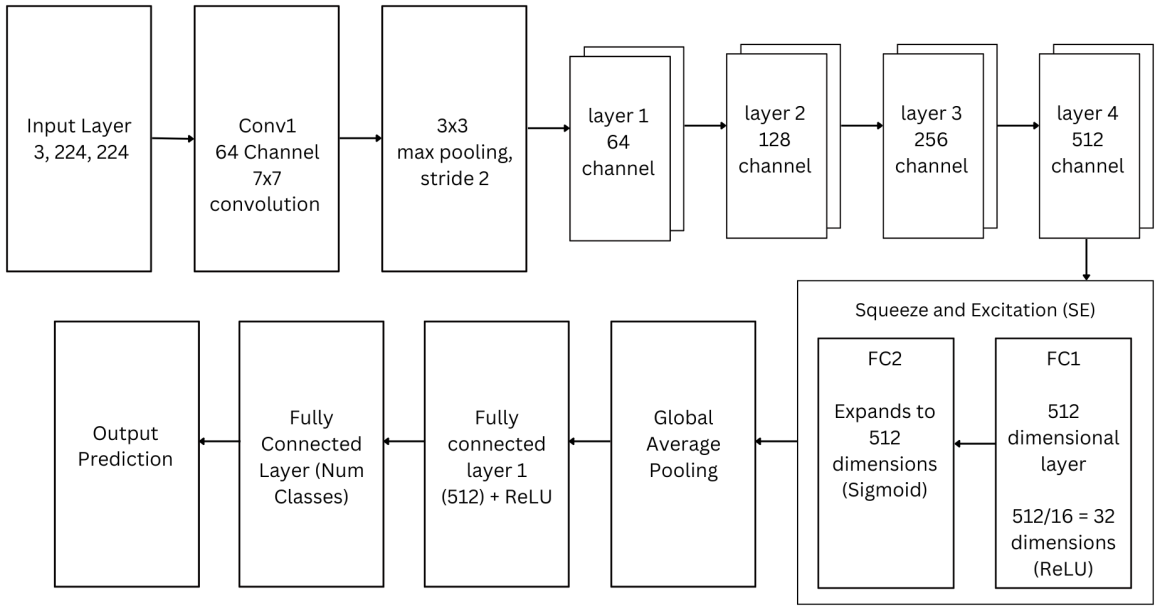

Figure 4.5: Currency Identification Model architecture.

The architecture processes an input image systematically through a sequence of layers, each contributing to extracting features and enabling accurate predictions. The input to the model is an image of size $3 \times 224 \times 224$, where 3 corresponds to the RGB color channels, and 224×224 represents the image's height and width. This input is the starting point for the feature extraction process.

The first block in the pipeline is the Conv1 layer, which applies a 7×7 convolution operation with 64 filters. This convolutional layer captures low-level features such as edges, corners, and textures, which are fundamental for building higher-level representations. Following this, a 3×3 max-pooling operation with a stride of 2 reduces the spatial dimensions of the feature maps. This pooling operation minimizes computational complexity and ensures translation invariance while retaining the most important features.

The output of max pooling is passed through a series of convolutional layers, denoted as layer 1, layer 2, layer 3, and layer 4. These layers progressively increase the number of channels (from 64 in layer 1 to 512 in layer 4), allowing the model to capture increasingly abstract and high-level features. Layer 1 focuses on basic patterns, while layer 2 detects simple structures, layer 3 identifies object parts, and layer 4 captures complex object

representations. The hierarchical nature of these layers enables the model to learn multi-scale features critical for accurate predictions.

A squeeze-and-excitation (SE) block is integrated into the architecture after layer 4. The SE block introduces a channel-wise attention mechanism to recalibrate feature maps, emphasizing the most informative channels while suppressing less useful ones. Specifically, the SE block begins with global average pooling, which squeezes each channel into a single value, summarizing its spatial information. This is followed by two fully connected (FC) layers: the first layer reduces the dimensionality of the feature maps to $512/16 = 32$ dimensions using a ReLU activation, while the second FC layer restores the dimensionality to 512 using a Sigmoid activation. The Sigmoid function outputs attention weights that scale each channel, enhancing the network's focus on relevant features.

After channel recalibration, the recalibrated features are passed through a global average pooling (GAP) layer. GAP reduces the spatial dimensions of the feature maps to a single vector per channel, effectively compressing the feature maps into a compact representation. This operation is critical for preparing the features for the fully connected layers.

The output of GAP is fed into a fully connected layer with 512 units, followed by a ReLU activation function. This layer introduces non-linearity and further processes the features to make them suitable for the final classification step. The final fully connected layer maps these features to the number of classes in the dataset, producing a set of class scores. The class with the highest score is selected as the model's prediction.

The overall architecture effectively combines convolutional layers for feature extraction, the SE block for channel attention, and fully connected layers for classification, ensuring a robust and efficient flow of data through the network.

4.5.1 Training Process

The training process involves training an money classification model using the ResNet-18 architecture. The following sections detail the various stages of the training procedure:

Data Preparation

The training and testing datasets are loaded from directories containing images and are preprocessed by resizing them to 224x224 pixels. The images are then converted into

tensors and normalized using the mean and standard deviation values of pre-trained ResNet models.

$$\text{Mean} = [0.485, 0.456, 0.406], \quad \text{Std} = [0.229, 0.224, 0.225]$$

For efficient processing, data loaders are used to manage the datasets. The training and testing datasets are loaded with a batch size of 64. Training data is shuffled to improve model generalization, while testing data remains in its original order for consistent evaluation.

Model Initialization

The model is based on the ResNet-18 architecture and is initialized with the number of output classes set to 4. To compute the loss between predicted and true labels, the cross-entropy loss function is used. For optimization, the Adam optimizer is employed with a learning rate of 0.001, ensuring efficient weight updates during training.

Training Loop

The model is trained for 32 epochs, following a structured training and validation loop.

During the training phase, the model is set to training mode at the start of each epoch. For each batch of images and labels, the data is moved to the specified device (GPU or CPU), and the optimizer gradients are reset to zero. A forward pass is performed by passing the images through the model to generate predictions, followed by computing the loss and performing backpropagation. The optimizer then updates the model parameters based on the computed gradients. Throughout this process, running loss and accuracy are computed and accumulated. At the end of each epoch, the overall loss and accuracy are calculated and displayed.

In the validation phase, the model is switched to evaluation mode. For each batch of validation data, the images and labels are moved to the specified device, and a forward pass is performed to generate predictions. The loss is computed, and the validation loss and accuracy are accumulated. Finally, the computed validation loss and accuracy are displayed after each epoch.

Output layer

This is the final layer of our training process. After passing each image batch through the neural network for 32 epochs, we obtain a model that is ready for use on our system. This layer saves the model to the root directory for future use.

4.5.2 Inference

In this process, the system activates once it is provided with an image. The image is pre-processed, resized to a 240x240 BGR image, and fed into the trained model. The model now first extract the feature using ResNet-18 model, this process is accomplished by capturing the second to last output of ResNet-18 model architecture instead of the final output. Then this feature is fed forward to extra layers as defined above in architecture above. Finally, one among 8 class with highest probability is predicted as output, which is essentially the last layer of our model.

Then the model prediction being in the class label format, for example 0, 1, 2 to 7 instead of Rupees 50, Rupees 100. Now these classes output is mapped to their respective value as per following table. And this output is provided to the user as the output of the model.

Table 4.7: Class Mapping for Currency

Predicted Value	Mapped Output
0	Rupees 50
1	Rupees 5
2	Rupees 500
3	Rupees 100
4	Rupees 10
5	Rupees 1000
6	Rupees 20
7	Unknown

4.6 Face Recognition model Architecture

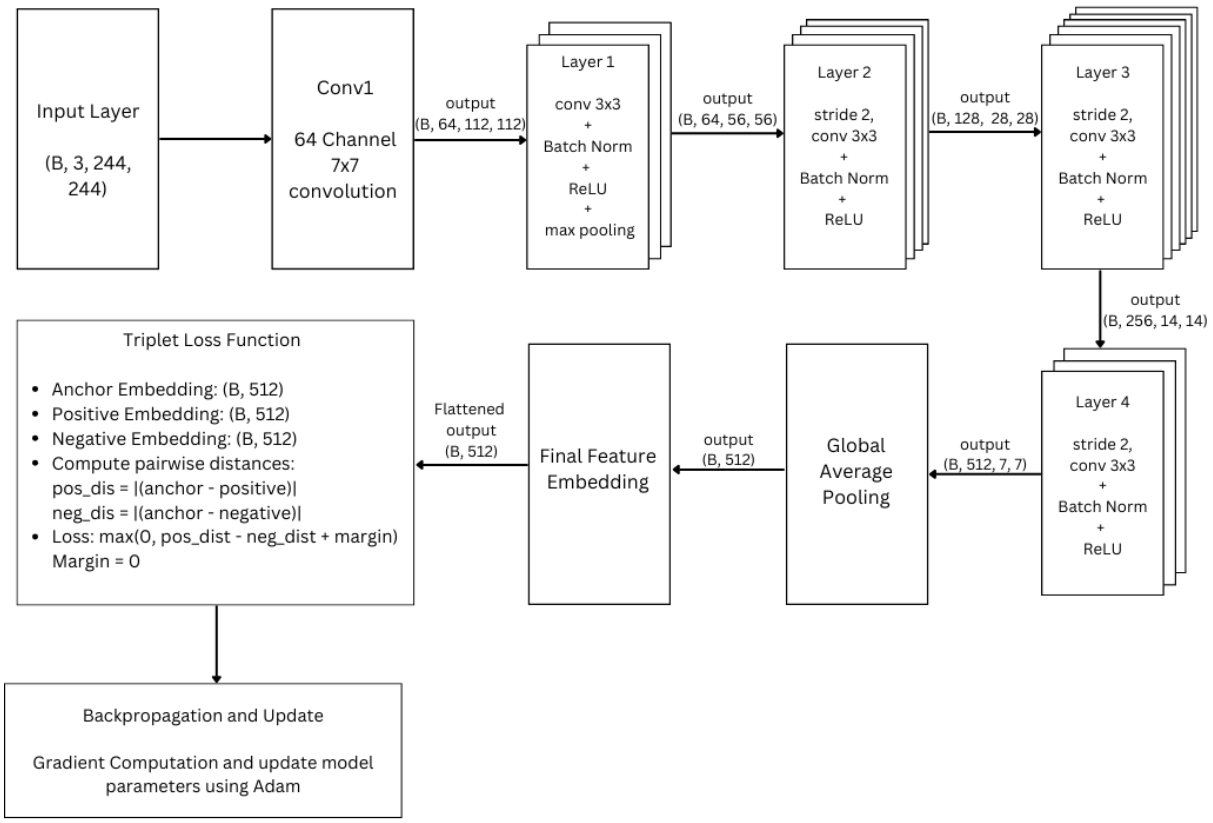

Figure 4.6: Face Recognition Model architecture.

The architecture begins with an Input Layer that processes images of size $B \times 3 \times 244 \times 244$, where B represents the batch size, 3 indicates the RGB color channels, and 244×244 corresponds to the spatial resolution of the image. This input is passed into the first convolutional block, Conv1, which applies 64 filters of size 7×7 to the image. This operation helps extract low-level features such as edges and textures. The output from this layer has dimensions $B \times 64 \times 112 \times 112$, where the spatial resolution is reduced due to the stride and convolutional operation.

Following Conv1, the output enters Layer 1, which applies 3×3 convolutions, followed by Batch Normalization to stabilize the training process and ReLU activation to introduce non-linearity. Additionally, a max pooling operation is applied to further reduce the spatial dimensions. The output from Layer 1 has dimensions $B \times 64 \times 56 \times 56$. The reduction in resolution allows the model to focus on larger receptive fields while preserving

essential spatial information.

Next, the output proceeds to Layer 2, where 3×3 convolutions with a stride of 2 are applied. This down-sampling operation reduces the spatial dimensions further while increasing the channel count to 128. Batch Normalization and ReLU activation are used here as well to ensure stable training and non-linear transformations. The output of Layer 2 has dimensions $B \times 128 \times 28 \times 28$, signifying a more compact representation of the input features.

In Layer 3, another set of 3×3 convolutions with a stride of 2 is applied, along with Batch Normalization and ReLU activations. This increases the number of channels to 256 while reducing the resolution to $B \times 256 \times 14 \times 14$. At this stage, the network captures more complex features, such as patterns and shapes, which are useful for high-level semantic understanding. The output is further passed to Layer 4, which again applies 3×3 convolutions with stride 2, Batch Normalization, and ReLU activation. Layer 4 produces feature maps of size $B \times 512 \times 7 \times 7$, where the spatial dimensions are reduced, but the depth of the features increases to 512 channels. These features contain rich semantic information.

The output from Layer 4 is then passed through a Global Average Pooling (GAP) layer. GAP reduces each 7×7 feature map into a single value by averaging the activations across the spatial dimensions. This operation converts the feature map into a compact vector of size $B \times 512$, which serves as the Final Feature Embedding of the input image. The embedding is a highly compressed and meaningful representation of the input that can be used for tasks such as classification or similarity measurement.

The model incorporates a Triplet Loss Function to optimize the feature embeddings. The triplet loss compares three types of embeddings: anchor, positive, and negative, all with dimensions $B \times 512$. It calculates the pairwise distances: $pos\_dis = ||\text{anchor} - \text{positive}||$ and $neg\_dis = ||\text{anchor} - \text{negative}||$. The loss is defined as:

$$\text{Loss} = \max(0, pos\_dis - neg\_dis + \text{margin}),$$

where the margin is set to 0. This ensures that the distance between the anchor and positive embeddings is minimized, while the distance between the anchor and negative

embeddings is maximized. By enforcing this constraint, the model learns to generate embeddings that are discriminative and suitable for tasks like face recognition or retrieval.

Finally, the Backpropagation and Update step uses the computed loss to update the model parameters. Gradients are calculated using the loss function, and the Adam optimizer is employed to adjust the weights of the network. The Adam optimizer ensures efficient and adaptive learning, allowing the model to converge faster while maintaining stability.

Overall, the architecture effectively extracts hierarchical features through multiple convolutional and pooling layers, compresses them using global average pooling, and learns robust embeddings using the triplet loss. This process enables the model to generate meaningful representations that are well-suited for various downstream tasks.

4.6.1 Training Process

The training loop follows a structured process to optimize the model using triplet loss. Each epoch consists of two main phases: training and validation.
During training, the model is set to train mode, and batches of triplets (anchor, positive, negative) are passed through it. The embeddings for these images are computed, and the triplet loss is calculated to ensure the anchor is closer to the positive than the negative. The optimizer updates the model parameters based on the loss, and performance metrics (accuracy, precision, recall, and F1-score) are computed.
In the validation phase, the model switches to evaluation mode, preventing weight updates. The same triplet loss computation is performed, and validation metrics are recorded to assess model performance. If the validation loss improves, the model's state is saved.
The process continues for the specified number of epochs or until early stopping is triggered, which prevents overfitting by stopping training when validation loss no longer improves. At the end of training, performance statistics are logged, and the best model is saved for further use.

4.6.2 Inference

During inference, the trained model processes input images to generate 512-dimensional embeddings. These embeddings are then compared using the Euclidean distance metric

to identify similar faces. The inference process for the face recognition model involves the following key steps:

The inference process begins by initializing the face detection model using MediaPipe and a ResNet50-based feature extractor. The system gets image and performs inference over it.

For provided image, the model detects faces and extracts the bounding box coordinates. If a face is detected, it is cropped and resized to match the training input size. Every second, a separate thread runs an inference function to extract facial embeddings using the feature extractor model. The extracted embedding is then compared against pre-saved embeddings using cosine similarity to determine the closest match.

The best match, if found within a similarity threshold, is displayed on the screen along with the confidence score. If no match is found, the system notifies the user. The program continuously runs, updating results dynamically, until the user exits by pressing the 'q' key.

4.7 Scene Caption Generation model Architecture

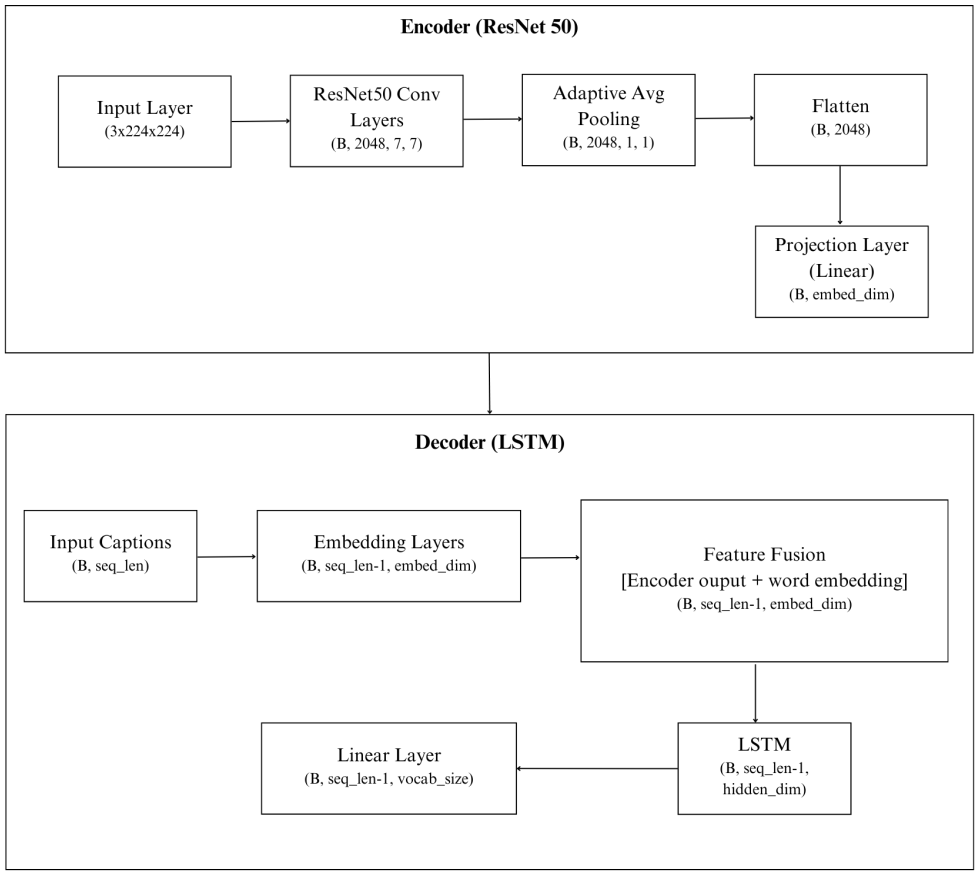

Figure 4.7: Scene Caption Generation Model architecture.

The image captioning model follows an encoder-decoder architecture. The encoder begins with a pretrained ResNet50 backbone, modified to exclude its final classification layers (average pooling and fully connected layers). This preserves the spatial feature maps generated by ResNet50's convolutional blocks, which capture hierarchical visual patterns—from edges and textures to objects and scenes. These features are then compressed into a fixed-size representation using an adaptive average pooling layer, which collapses the spatial dimensions of the feature map into a 1x1 grid. The pooled features are flattened into a 1D vector and passed through a linear projection layer, reducing their dimensionality to match the decoder's embedding space (512 dimensions). This final embedding serves as a distilled summary of the image's semantic content.

The decoder is an LSTM-based network tasked with generating grammatically coherent and contextually relevant captions. It starts by converting input caption tokens (integer

indices) into dense vector representations via an embedding layer. Crucially, the image embedding from the encoder is fused with these word embeddings at every timestep. This is achieved by expanding the image embedding to match the sequence length of the caption and adding it element-wise to the word embeddings. The fused features are then processed by the LSTM, which sequentially updates its hidden state to model dependencies between words. At each timestep, the LSTM's output is passed through a linear layer to produce logits (unnormalized scores) over the vocabulary, predicting the likelihood of the next word.

A key design choice is feature fusion: the image embedding is reused across all decoder timesteps, ensuring the model maintains awareness of the visual context while generating each word. During training, the decoder operates in teacher-forcing mode, where ground-truth captions (shifted by one token) are fed as input to accelerate convergence. At inference, the process becomes autoregressive: the decoder generates captions iteratively, starting with a <start> token and producing one word at a time until an <end> token is predicted.

Together, the encoder and decoder form a pipeline that translates images into text. The encoder's ResNet50 backbone provides robust visual understanding, while the decoder's LSTM enables sequential language modeling, with feature fusion bridging the two modalities. This architecture balances efficiency and performance, leveraging pretrained vision models and lightweight recurrent networks to achieve end-to-end caption generation.

4.7.1 Training Procedure

This training code defines a Trainer class for training and evaluating a scene description model. It initializes key components such as the model, tokenizer, data loaders, loss function, and optimizer. It also tracks performance metrics, including training loss, validation loss, and BLEU scores. The training loop processes image-caption pairs by tokenizing and padding captions, computing loss using cross-entropy, and updating model weights through backpropagation.

The validation phase evaluates model performance by generating captions and comparing them to ground-truth captions using the BLEU score. The model generates captions one token at a time until it reaches an end token or the maximum sequence length. The

training process includes early stopping to prevent overfitting, saving the best-performing model based on BLEU score improvements.

Throughout training, metrics are logged, saved, and visualized to track performance trends. The final trained model is saved, ensuring that the best version is available for inference. The structured design of the Trainer class makes it adaptable for various image captioning tasks while maintaining efficiency and clarity.

4.7.2 Inference

This inference pipeline is designed to perform real-time image captioning to provide image. When the program is executed, the provided image is then preprocessed—converted from BGR to RGB, resized to 224×224 pixels, and normalized—to match the model's training conditions. The preprocessed image is passed through an encoder-decoder model, where the encoder (a ResNet50-based network) extracts visual features and the decoder generates a caption using beam search or greedy decoding. A trained tokenizer maps between words and indices to form a coherent sentence as the final output.

When an image is processed, the generated caption is then retured as output of the system.

4.7.3 Data Pre-processing

Scene Dataset:

The scene dataset required extensive cleaning due to inconsistencies such as images without captions, mislabeled data, and missing images for certain labels. A Python script was used to filter out irrelevant images and labels while generating a list of images lacking associated labels. These images were manually reviewed, re-labeled, or re-captioned to maintain consistency. For the custom dataset, all images were uniformly re-captioned to match the format of the online dataset. The pre-processing steps for these images included horizontal and vertical flipping, vertical and horizontal shearing, conversion to BGR format, and noise reduction using denoising techniques.

Money Dataset:

The online dataset for currency images was already well-structured and required minimal pre-processing. To enhance the dataset size and diversity, the custom dataset was

integrated with the online dataset, creating a comprehensive and larger dataset for training.

Facial Images:

The facial image dataset was sourced entirely from online resources and social media platforms. To standardize the dataset, several pre-processing steps were applied, including vertical and horizontal flipping, vertical and horizontal shearing, conversion to BGR format, and transformation into black-and-white images. These steps ensured that the images were properly prepared for further processing and model training.

4.8 Hardware Block Diagram

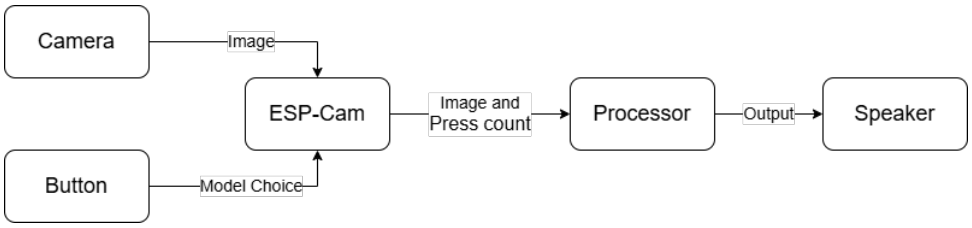

Figure 4.8: Hardware block Diagram.

Hardware block diagram description

This hardware block diagram shows a camera feeding live image data into a ESP32-Cam module, which performs task of counting button press number and transmitting the image and press count upon requested by receiver or Processor.

Then the processor process the image provided by ESP according to the request and produce an audio output, which is then played by a speaker unit.

CHAPTER 5

RESULT AND OUTPUT

5.1 Face Model Training Graph

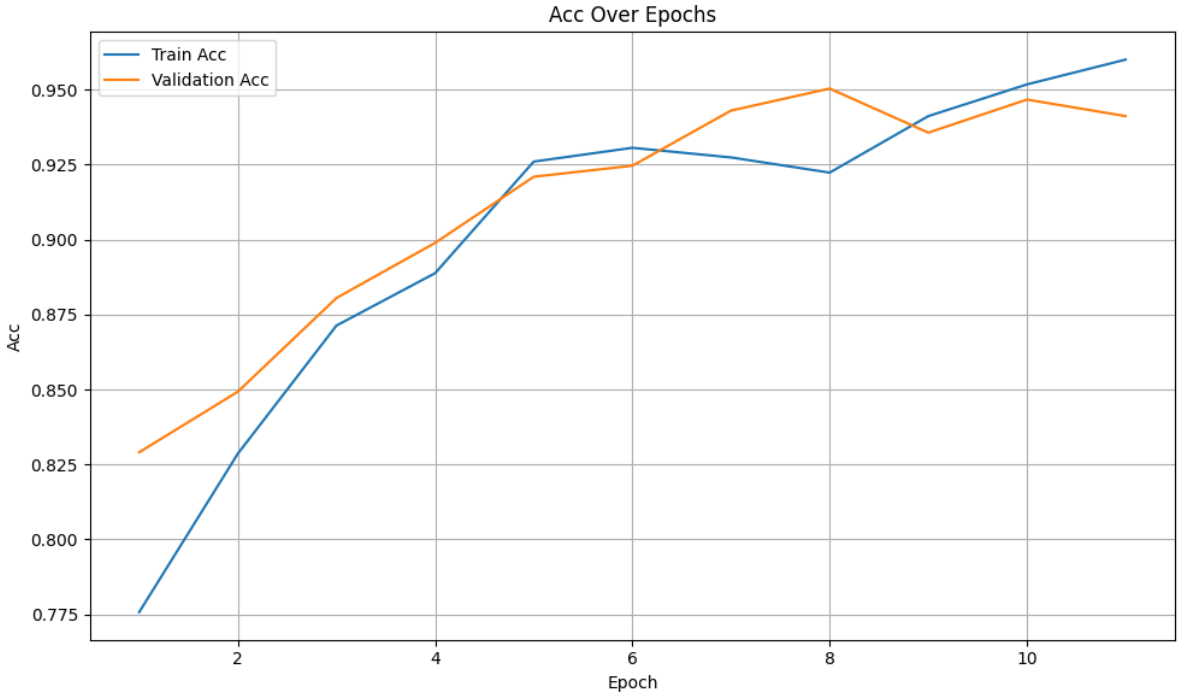

Figure 5.1: Accuracy graph

Up until the 8th epoch, the validation accuracy is higher than the training accuracy, indicating that the model is generalizing well to unseen data. However, after the 6th epoch, the training accuracy surpasses the validation accuracy, signaling potential overfitting. The model performs exceptionally well on training data but slightly worse on validation data.

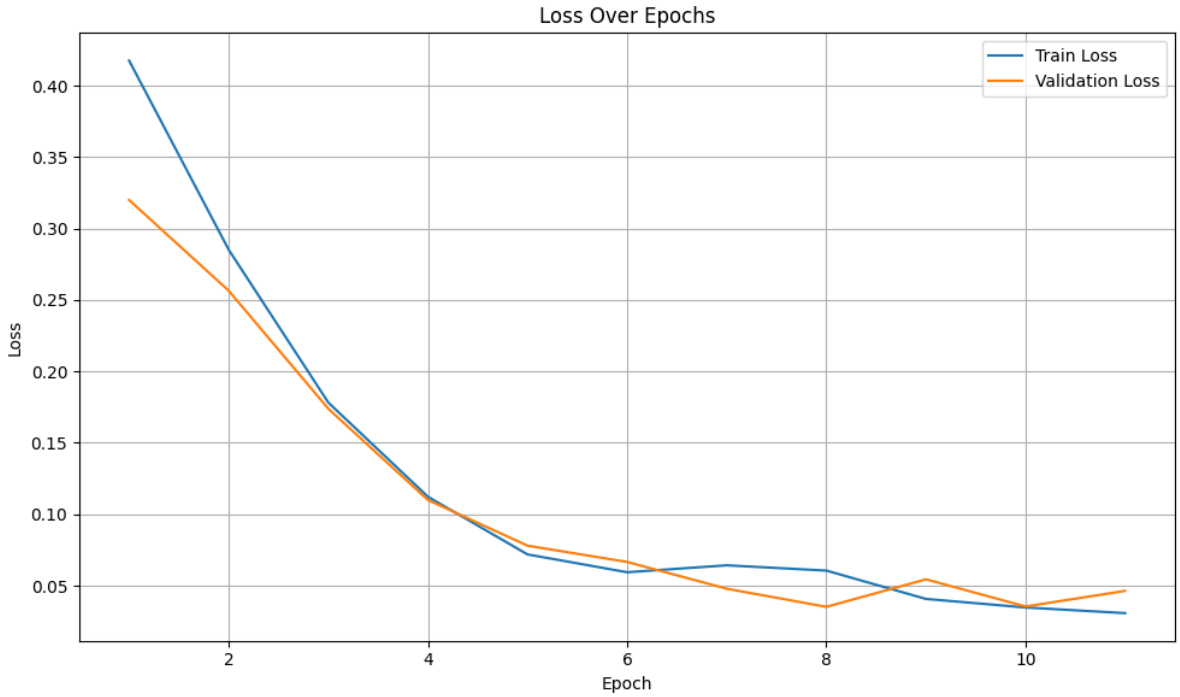

Figure 5.2: Loss graph

Both the training and validation losses converge around the 4th epoch and continue to decrease together. This suggests that the model is not overfitting and is generalizing well to the validation data. The consistent decrease in both training and validation losses indicates that the model is learning effectively and improving its accuracy over the training period.

5.2 Currency Identification Training Graph

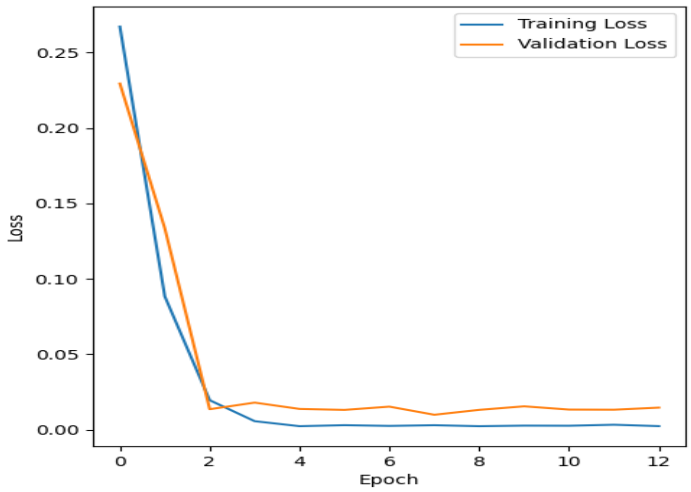

Figure 5.3: Loss graph

Both the training and validation losses show a rapid decrease initially, which signifies that the model is effectively learning and improving. The training loss continues to decrease and remains consistently lower than the validation loss throughout the epochs, suggesting that the model fits the training data very well. However, the slight fluctuations in the validation loss after stabilizing indicate that there might be minor overfitting, but overall the model generalizes well.

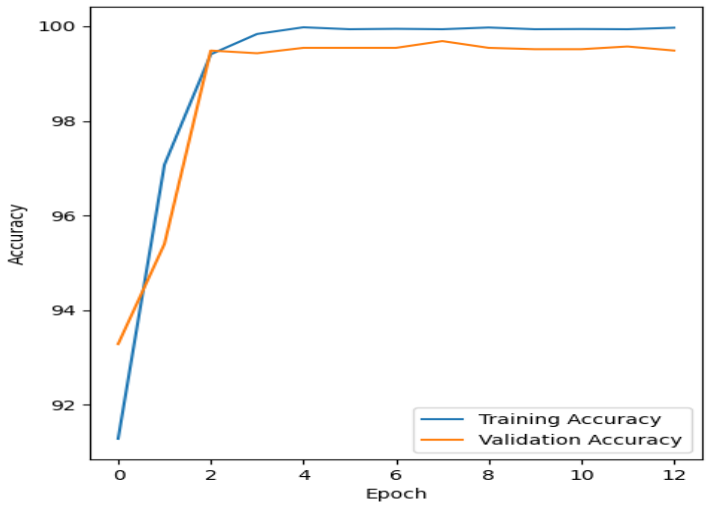

Figure 5.4: Accuracy graph

Both the training and validation accuracies increase quickly and plateau after the 3rd epoch.

The training accuracy remains around 100%, while the validation accuracy stabilizes around 98%. This small gap between the training and validation accuracy indicates that the model is generalizing well to new data and is not overfitting.

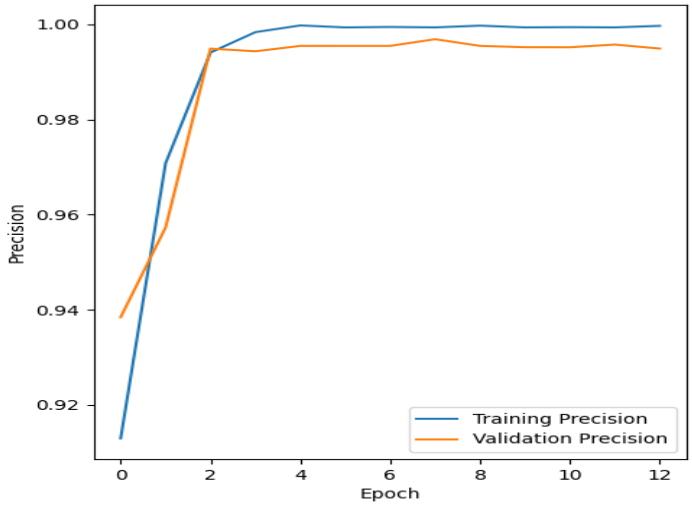

Figure 5.5: Precision graph

Both lines show a rapid increase in precision in the early epochs, indicating effective learning. The training precision remains slightly higher than the validation precision, but the difference is minimal, suggesting that the model is not overfitting significantly and generalizes well to new data.

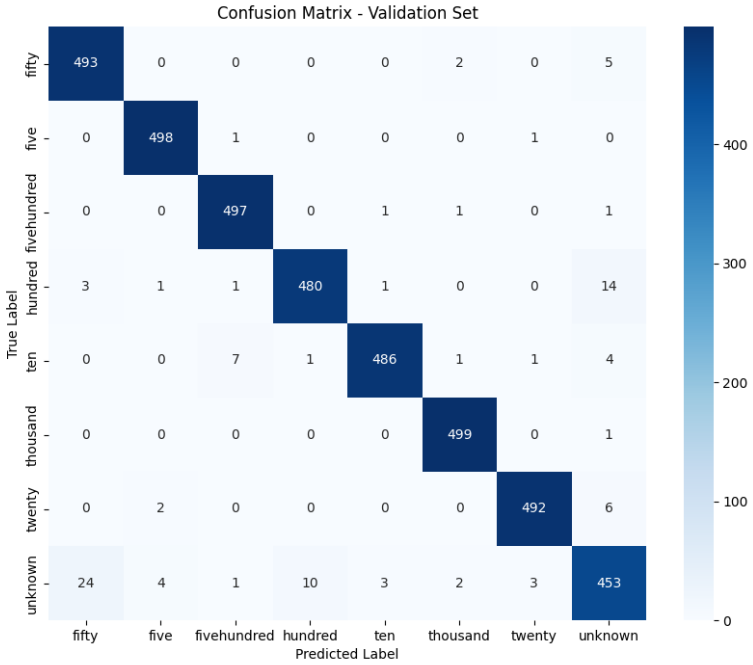

Figure 5.6: Currency Identification Confusion Matrix

The confusion matrix above indicates high accuracy in classifying various denominations. The model correctly identifies Fifty 493 times, Five 498 times, Five Hundred 497 times, Hundred 480 times, Ten 486 times, Thousand 499 times, Twenty 492 times, and Unknown 453 times. These high values suggest the model performs well in distinguishing different categories.

However, minor mis-classifications are observed. For example, Hundred was misclassified as Fifty 3 times and as Unknown 14 times, while Ten was misclassified as Five Hundred 7 times and Hundred once. Similarly, the Unknown category shows slight confusion, being misclassified as Fifty 24 times.

Overall, the model demonstrates strong generalization with minimal mis-classification, though slight improvements could be made in refining edge cases. The consistency in high correct classifications suggests the model is reliable but should be monitored for potential overfitting if the validation set is too similar to the training set.

5.3 Scene description Training Graph

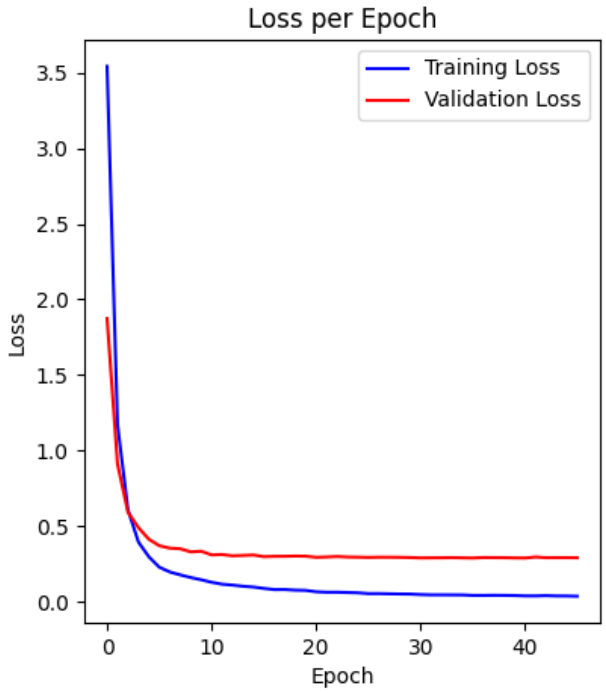

Figure 5.7: Loss graph

This graph illustrates the loss per epoch for both training and validation phases of a machine learning model. The x-axis represents the number of epochs, while the y-axis indicates the loss value. The blue curve corresponds to the training loss, and the red curve represents the validation loss.

At the beginning (early epochs), both losses are high, but they decrease rapidly, indicating that the model is learning and improving. The training loss continues to decline steadily, whereas the validation loss plateaus after a certain point. This suggests that while the model performs well on training data, it may not generalize as effectively to unseen validation data, possibly indicating overfitting. The gap between the training and validation loss in later epochs further supports this observation.

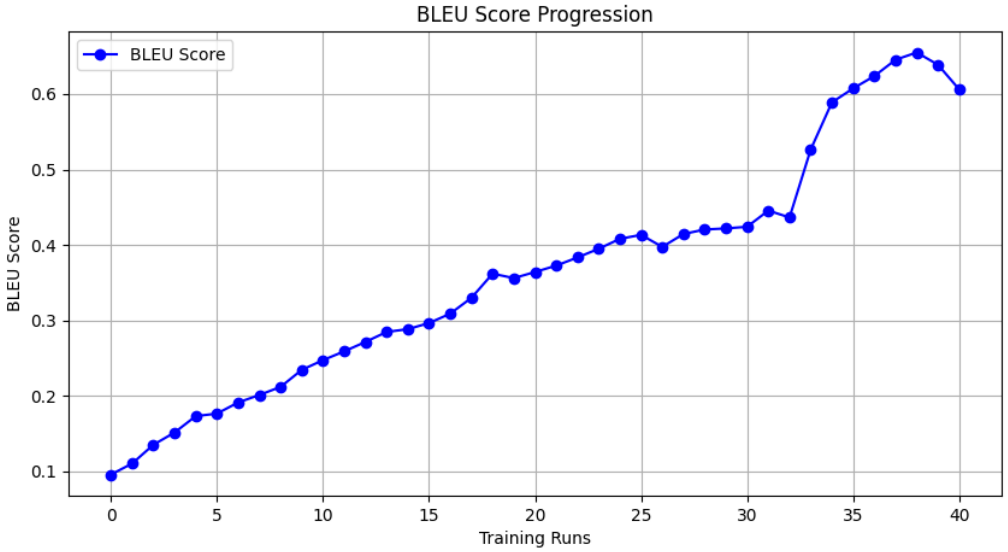

Figure 5.8: Scene Description model BLEU score

This plot illustrates the progression of BLEU scores over multiple training runs, highlighting the model's performance improvements. The x-axis represents sequential training runs, while the y-axis shows the BLEU score, which measures translation quality. Initially, the scores steadily increase, indicating consistent learning. Around run 32, there is a sharp rise, suggesting a significant performance boost, possibly due to effective model updates or improved learning. Toward the end, a slight decline appears, which might indicate overfitting or natural fluctuations in evaluation. Overall, the trend demonstrates a strong improvement in translation quality over time.

CHAPTER 6

EPILOGUE

6.1 CONCLUSION

The development of Smart AI Spectacles shows how technology can make life easier and more inclusive for everyone. This project is designed to assist visually impaired individuals by providing features like face detection, currency identification and scene description. With these capabilities, it helps people navigate their surroundings more independently and confidently.

Beyond its primary purpose, Smart AI Spectacles can also be useful for others, such as elderly individuals or those in low-visibility conditions. This project proves that with the right use of AI, we can create solutions that benefit a wide range of people and promote inclusiveness in technology.

Through this project, we have learned the importance of designing for accessibility and the potential impact of assistive technologies. While there is always room for improvement, Smart AI Spectacles is a step forward in making technology more helpful for everyone.

6.2 Limitations

While Smart AI Spectacles offer significant advantages in assisting visually impaired individuals, there are certain limitations that need to be addressed for further improvement.

6.2.1 Computational Speed

The system requires faster processing to provide real-time responses efficiently. Due to hardware constraints, there may be slight delays in processing and delivering outputs.

6.2.2 Scene Description Accuracy

The AI may not always describe every object present in the scene. Instead, it tends to focus on the main subject or provide a general description, which may sometimes miss important details.

6.2.3 Lighting Conditions for Money Recognition

The currency identification system struggles to function properly in low-light conditions, affecting its accuracy and reliability in such environments.

6.2.4 Face Recognition Challenges

Due to limitations of the ESP camera, face recognition does not work as efficiently as it does with a laptop webcam. The system may struggle with detection, especially in varying lighting conditions or from certain angles.

6.2.5 Scene Description with ESP Camera

Similar to face recognition, the scene description system also faces difficulties due to ESP camera limitations. It may not capture details as effectively as higher-quality cameras, impacting the overall performance.

Despite these limitations, Smart AI Spectacles demonstrate strong potential in assistive technology, and further enhancements in hardware and software can significantly improve its performance in the future.

6.2.6 Text-to-Speech Output

The Text-to-Speech (TTS) system used may not be the most advanced, which can affect the naturalness and clarity of the generated speech. In some cases, the TTS may struggle with proper pronunciation, intonation, and fluidity, making it less effective for conveying information naturally to the user.

6.3 Future Enhancements

As technology continues to advance, there are several improvements and additional features that can enhance the functionality and effectiveness of **Smart AI Spectacles**. Some key areas for future enhancements include:

6.3.1 Improved Scene Description for Navigation

With further advancements in scene description, the system can be enhanced to provide better contextual awareness, making it useful for both indoor and outdoor navigation. By accurately identifying objects, obstacles, and pathways, the spectacles can assist users in moving around unfamiliar environments more safely and efficiently.

6.3.2 Inclusive Navigation with Spatial Awareness

By incorporating more data about the surroundings and the user's position, the system can offer inclusive navigation assistance. This could include real-time directions, obstacle detection, and guidance based on environmental factors, making mobility easier for individuals with visual impairments.

6.3.3 Enhanced Computational Efficiency

Optimizing the hardware and software for faster processing can significantly improve the real-time performance of the system. This would ensure quicker responses for scene description, face recognition, and currency identification, making the user experience smoother and more reliable.

6.3.4 Better Performance in Low-Light Conditions

Future enhancements could focus on improving the system's ability to function in low-light conditions. By integrating more advanced image processing techniques or infrared-based vision support, the spectacles could provide accurate recognition even in dim environments.

6.3.5 Upgraded Camera and Sensor Integration

Replacing the current ESP camera with a higher-quality sensor can enhance both face recognition and scene description accuracy. Improved camera resolution, better light sensitivity, and wider field-of-view sensors can significantly boost the system's effectiveness.

6.3.6 AI Model Improvements with More Training Data

By training the AI on a larger and more diverse dataset, the accuracy of object recognition, scene description, and currency identification can be improved. This would help the system adapt better to real-world variations and enhance its overall usability.

With continuous development and integration of emerging technologies, **Smart AI Spectacles** have the potential to become a highly advanced assistive tool, making daily life more accessible and independent for users.

APPENDIX A

APPENDIX

Hardware

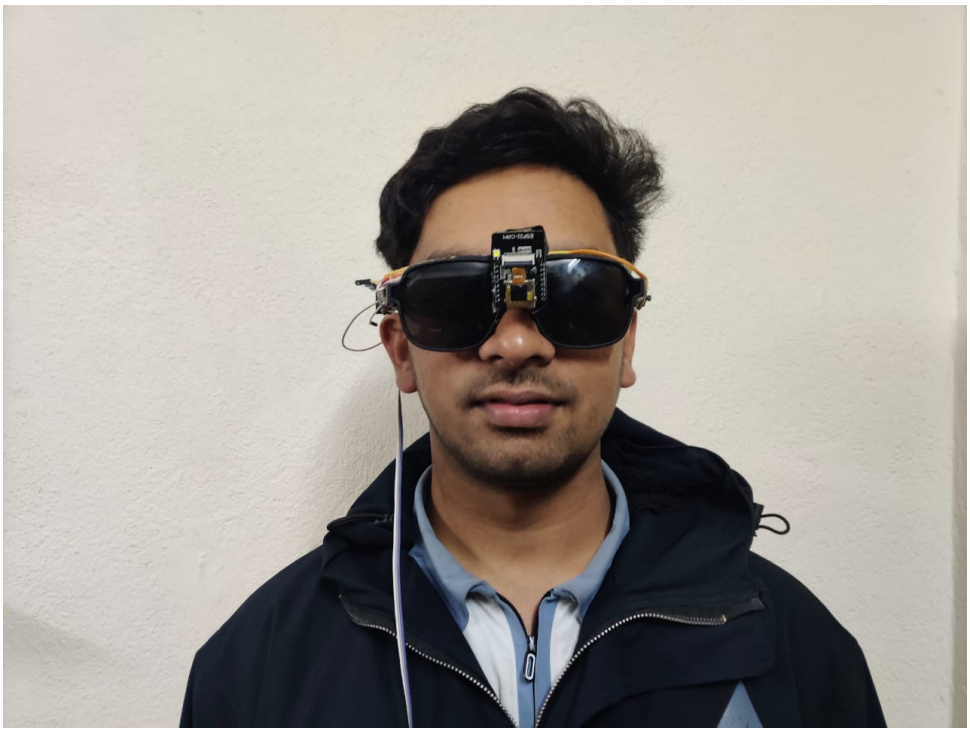

Figure A.1: System Hardware

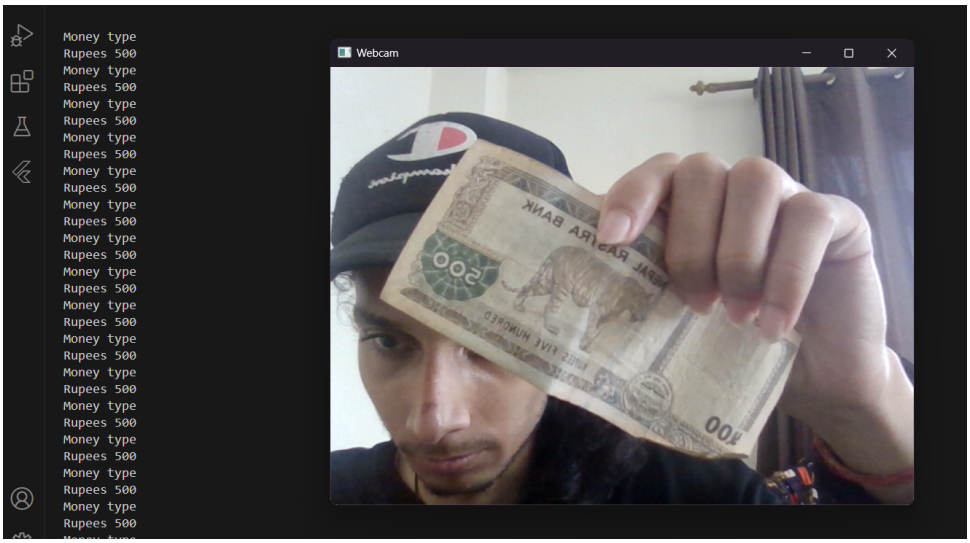

Figure A.2: Currency Identification Output

Model Output

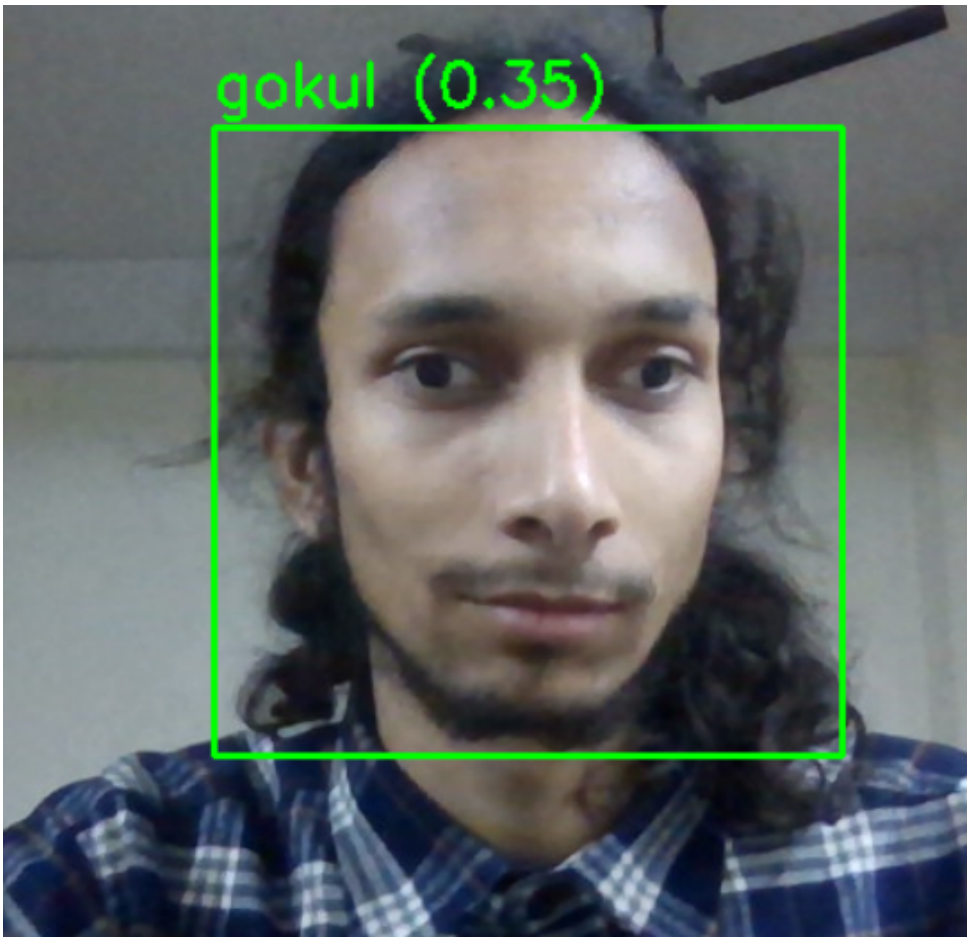

Figure A.3: Face identified

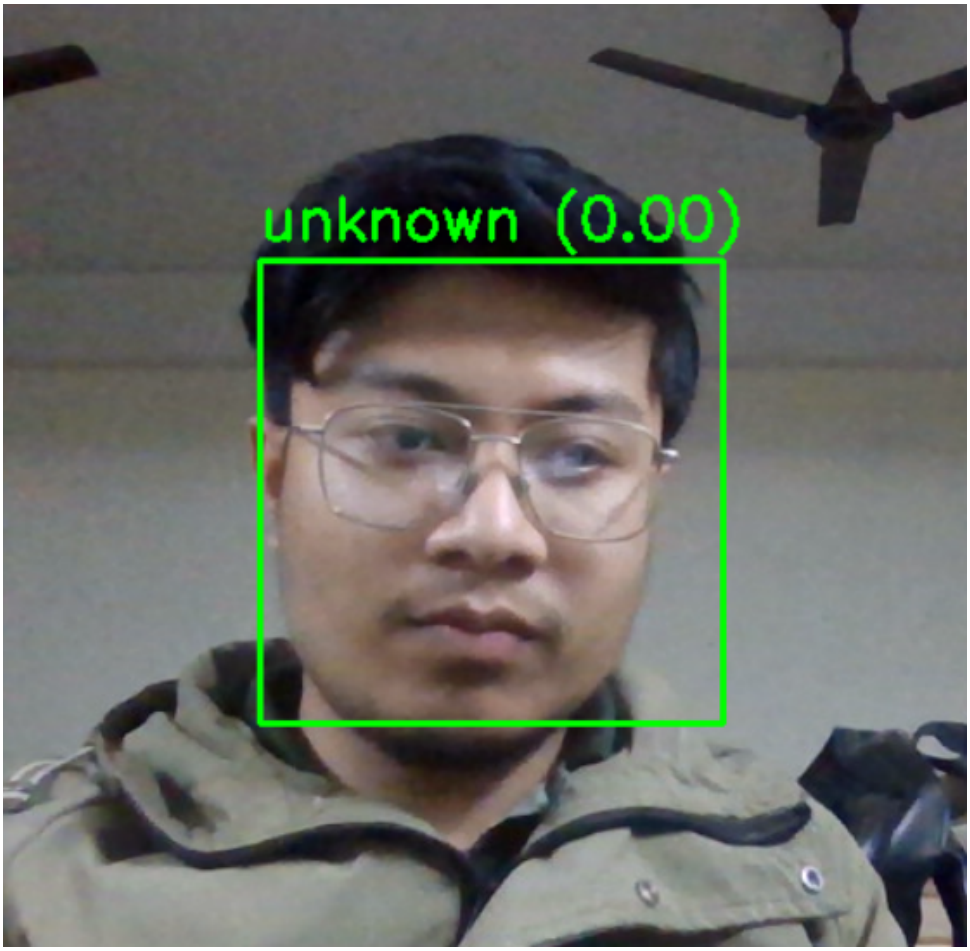

Figure A.4: UnKnown face output

Figure A.5: Scene description model output

Dataset:

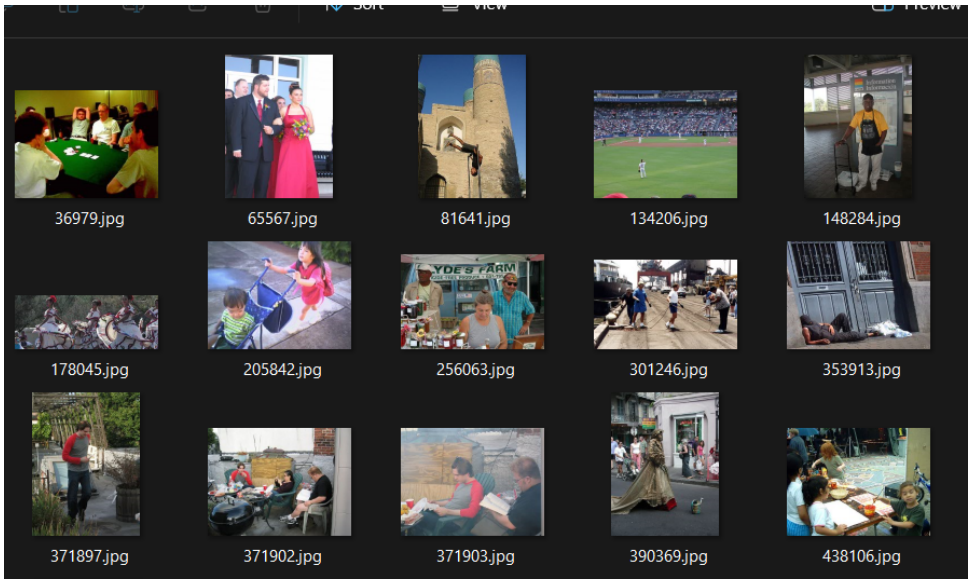

Figure A.6: Online Dataset for Scene description

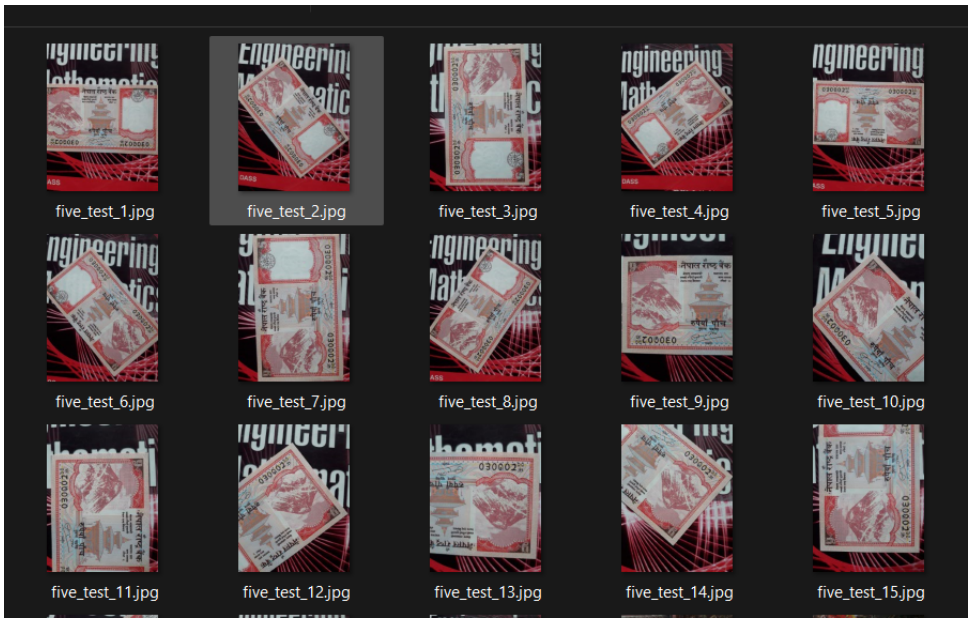

Figure A.7: Kaggle Dataset for Money Identification

Figure A.8: Custom dataset for Scene Description.

Figure A.9: Custom dataset for rupees 500

Model Summary

```
Total params: 11,476,071
Trainable params: 11,476,071
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 63.00
Params size (MB): 43.78
Estimated Total Size (MB): 107.35
----------------------------------------------------------------
```

Figure A.10: Summary of money classifier model.

```
Total params: 23,508,032
Trainable params: 23,508,032
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 286.55
Params size (MB): 89.68
Estimated Total Size (MB): 376.80
----------------------------------------------------------------
```

Figure A.11: Summary of face recognition model.

```
-------------------------------------------------
: ImageCaptioningModel
image_encoder: Sequential
image_encoder.0: Linear
image_encoder.1: ReLU
image_encoder.2: Linear
image_encoder.3: ReLU
embedding: Embedding
lstm: LSTM
attention: Attention
attention.attention: Linear
attention.softmax: Softmax
fc: Linear
dropout: Dropout
-------------------------------------------------
Total Parameters: 147796533
```

Figure A.12: Scene model summary.

REFERENCES

[1] WHO. Vision impairment and blindness. *World Health Organization*, 2023. https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment.

[2] L.B. Brilliant, R.P. Pokhrel, N.C. Grasset, J.M. Lepkowski, A. Kolstad, W. Hawks, R. Pararajasegaram, G.E. Brilliant, S. Gilbert, S.R. Shrestha, and J. Kuo. Epidemiology of blindness in nepal. *Seva Canada*, 1985. https://www.seva.ca/sites/default/files/epidemiology_of_blindness_in_nepal.pdf.

[3] Bogdan Mocanu, Ruxandra Tapu, and Titus Zaharia. Deep-see face: A mobile face recognition system dedicated to visually impaired people. *IEEE Access*, 6:51975–51985, 2018.

[4] Ahmed Yousry, Mohamed Taha, and Mazen Selim. Currency recognition system for blind people using orb algorithm. *International Arab Journal of Information Technology*, 5:34–40, 2018.

[5] Luís Fernando Torres. Convolutional neural network from scratch. *Medium.com*, 2023. https://medium.com/latinxinai/convolutional-neural-network-from-scratch-6b1c856e1c07.

[6] Convolutional neural network. https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-network-an-overview/.

[7] https://www.researchgate.net/publication/355125808/figure/fig5/AS:1076804416745476@1633741587342/Recurrent-Neural-Network.ppm.

[8] Flowchart of nlp process. https://images.prismic.io/turing/65980b21531ac2845a272614_Natural_language_processing_pipeline_e3608ff95c.webp.

[9] Turing. How does natural language processing function in ai? *Turing.com*, 2023. https://www.turing.com/kb/natural-language-processing-function-in-ai.

[10] Rohit Modi. Resnet — understand and implement from scratch. *Medium.com*, December 2021. https://medium.com/analytics-vidhya/resnet-understand-and-implement-from-scratch-d0eb9725e0db.

[11] https://www.researchgate.net/publication/341159618/figure/tbl1/AS:8878245\6804966 7@1588685280641/ResNet-18-architecture-and-layer-parameters.png.

[12] https://en.wikipedia.org/wiki/Long_short-term_memory#/media/File:LSTM_Cell.svg.

[13] Aakarsha Chug. What is lstm – long short term memory? *GeeksforGeeks*, 2024. https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/.

[14] Pin-Yu Chen and Cho-Jui Hsieh. Image captioning - an overview. *ScienceDirect*, 2023. https://www.sciencedirect.com/topics/computer-science/image-captioning.

[15] https://learnopencv.com/wp-content/uploads/2024/12/encoder-decoder.png.

[16] https://ars.els-cdn.com/content/image/3-s2.0-B9780128240205000156-f06-03-9780128240205.jpg.

[17] https://docs.sunfounder.com/projects/zeus-car/en/latest/_images/esp32_cam.png.

[18] https://cdn-reichelt.de/bilder/web/xxl_ws/C200/S_1323_RT.png.

[19] https://autolifethailand.tv/wp-content/uploads/2021/09/Lenovo-IdeaPad-Gaming-3-7.png.

[20] https://cdn11.bigcommerce.com/s-yo2n39m6g3/images/stencil/1280x1280/products/466/3407/lm2596-dc-dc-buck-converter__03171.1571733582.jpg?c=2?imbypass=on.

[21] https://www.cflowapps.com/wp-content/uploads/2022/08/agle_wrkflw.jpg.