TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

SAGARMATHA ENGINEERING COLLEGE

A

PROJECT REPORT

ON

VISUALLY IMPAIRED READING ASSISTANT

BY

ARJUN KOIRALA 36403

SUSHMIT PAUDEL 36419

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

June, 2025

VISUALLY IMPAIRED READING ASSISTANT

BY

Arjun Koirala 36403

Sushmit Paudel 36419

Project Supervisor

Er.Baikuntha Acharya

Deputy Head of Department of Electronics and Computer Engineering

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 “VISUALLY IMPAIRED READING ASSISTANT” 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 of Engineering 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.

Arjun Koirala 36403

Pawan Kandel 36408

Sushmit Paudel 36419

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 **VISUALLY IMPAIRED READING ASSISTANT**, submitted by **Arjun koirala, Pawan Kandel and Sushmit Paudel** in partial fulfillment of the requirement for the degree of "Bachelor in Electronics, Communication and Information Engineering".

....

Supervisor: Er. Baikuntha Acharya,
Department of Electronics and Computer Engineering,
Sagarmatha Engineering College

....

External Examiner: Prof. Dr. Nand Bikram Adhikari
Reader
Pulchowk Campus, Pulchowk

DEPARTMENTAL ACCEPTANCE

The project work entitled **"VISUALLY IMPAIRED READING ASSISTANT"**, submitted by **Arjun koirala, Pawan Kandel and Sushmit Paudel** 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.

ACKNOWLEDGEMENT

Foremost, we would like to express our heartfelt gratitude to **Er. Baikuntha Acharya**, our project supervisor and lecturer and Deputy Head of Electronics and Computer Engineering. His constant support, patience, and motivation have been invaluable throughout this journey. From guiding us in developing our project to helping us compile this report and acquire necessary resources, his support is behind our progress.

A big thank you to our Project Coordinator, **Er. Bipin Thapa Magar**, for always being available when we faced challenges. His valuable insights and willingness to assist at any time made a significant difference in overcoming obstacles and refining our work.

We also extend our appreciation to **Er. Bharat Bhatta**, the Head of the Department of Electronics and Computer Engineering, for his encouragement, insightful advice, and constant support. His guidance helped us shape our project with a more structured and thoughtful approach.

Moreover, we are grateful to all the faculty members of the Department of Electronics and Computer Engineering for sharing their knowledge and expertise, which played a crucial role in completing our project.

ABSTRACT

Reading printed books remains a huge challenge especially when it comes to Nepali texts written in the Devanagari script. This project is designed to help visually impaired people by creating a system that converts printed Nepali text into speech, making books more accessible to them. The system works by capturing images of book pages, extracting the text, and converting it into spoken words. This system integrates three advanced technologies to process images into meaningful speech. First, Optical Character Recognition (OCR) powered by image captioning, extracts text from images. Then, Natural Language Processing (NLP) structures the text by tokenizing and refining it for better readability. Finally, the processed text is transformed into clear, natural-sounding speech. At the core of this system is a deep learning model that processes images step by step. A Convolutional Neural Network (CNN), specifically ResNet-50, first analyzes the image to recognize characters. Then, a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) acts as a smart reader, interpreting the extracted text and converting it into meaningful speech. A custom-built tokenizer ensures the system understands different words accurately. The system is tested with key performance measures like precision, BLEU score to make sure it delivers high-quality results. This project is about creating real-world impact by making Nepali books accessible through speech visually impaired individuals can learn, explore, and experience the world of written knowledge.

Keywords: Visually impaired, Optical Character Recognition, Text-to-Speech, Convolutional Neural Network, Recurrent Neural Network, Long Short-Term Memory.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolution Neural Network
DNN	Deep Neural Network
EEPROM	Electrically Erasable Programmable Read-Only Memory
IDE	Integrated Development Environment
LSTM	Long Short Term Memory
ML	Machine Learning
NLG	Natural Language Generation
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
OCR	Optical Character Recognition
ReLU	Rectified linear unit
RNN	Residual Neural Network
SSML	Speech Synthesis Markup Language
TTS	Text to speech
RTC	Real-Time Clock
SoC	System on Chip

CHAPTER 1

INTRODUCTION

1.1 Background

Visually impaired individuals face significant challenges in reading Nepali printed text, which affects their independence and quality of life. Access to printed materials is essential for education, employment, and daily activities. Traditional methods of accessing printed text, such as Braille and audiobooks, have limitations. Braille books are bulky and not all materials are available in Braille, while Nepali audiobooks can be limited in availability. To address these challenges, various technologies have been developed. Many systems have been designed using ML and different AI algorithms. This system aims accessibility for visually impaired people around the world. The main issue is about the Nepali text reading system as there are many forms of Nepali texts (*sorwarna,benjan barna,rashwo,dirgha, etc.*) Some applications utilize these technologies but when it comes to the accurate recognition of non-Latin scripts, they often require powerful hardware and constant Internet access. Our system with integrated OCR and TTS technology is particularly promising in solving the issue.

1.2 Problem Definition

Visually impaired individuals struggle to access Nepali printed text, limiting their independence and quality of life. An integrated and user-friendly tool that can convert printed text into spoken words is needed, allowing visually impaired users to read any Nepali book independently. This system uses OCR and TTS technologies to address these challenges faced by visually impaired people, especially Nepalese people.

1.3 Objectives

The objective of this project is as follows.

- To develop an OCR-based tool that captures images of book pages containing Devanagari script and converts the extracted text into audible speech.

1.4 Features

The features of our project are as follows:

- Hardware is involved to ensure the system is lightweight and portable.
- Optimized software for devices with limited computational power and low-end laptops.
- Converts predicted text into spoken words using TTS technology, allowing users to listen to the content of the Nepali book.
- Accessible tool designed specifically for visually impaired users.

1.5 System Requirements

1.5.1 Software Requirements

The software requirements for our project are as follows:

(a) Programming Language:

- Python: The primary language used for implementing Image digitization, training custom models, and performing data pre-processing.

1.6 Required Libraries and Frameworks

1.6.1 Deep Learning Frameworks

PyTorch, torchvision, tqdm

1.6.2 Image Handling and Visualization

The Python Imaging Library (PIL), now known as Pillow, is responsible for managing image file input and converting images into formats compatible with the deep learning model. It handles various image formats, essential for preprocessing the input data. Matplotlib is visualization library used to plot training data, model evaluation metrics, and output results. It aids in monitoring the model's performance and understanding the trends in training.

1.6.3 Evaluation and Metrics

scikit-learn is a library providing essential evaluation metrics such as accuracy, precision, recall, and F1-score. These metrics are fundamental in assessing the model's performance, particularly in the validation phase of the project.

1.6.4 Model Optimization

To improve training efficiency and reduce resource consumption, the system utilizes advanced optimization techniques. These methods are particularly beneficial for handling large-scale models and ensuring faster training without compromising accuracy.

Mixed-Precision training techniques is optimization approach that leverages mixed-precision computations to balance memory usage and computational speed. By selectively using lower precision for certain operations on compatible GPUs, it enhances training efficiency while preserving the model's overall accuracy. This method is essential for achieving faster performance in resource-intensive tasks.

1.6.5 Custom Components

A custom tokenizer is implemented to handle vocabulary creation, as well as encoding and decoding captions. It is tailored to the dataset, ensuring efficient text processing for the image-captioning task.

1.6.6 ESP32/ESP32-CAM Board Libraries

The ESP32/ESP32-CAM board utilizes several important libraries to ensure smooth functionality and enhance the overall performance of the system. These libraries include:

Camera Library that Provides essential functions for interfacing with the ESP32 camera module and capturing image frames, enabling real-time image acquisition.

RTC I/O Control Library which manages the Real-Time Clock (RTC) I/O pins on the ESP32, which are used for power management and other specialized input/output operations.

EEPROM Library enables reading and writing of data to the device's EEPROM, allowing persistent storage of configurations and other critical data across system reboots.

Wi-Fi Connectivity Library facilitates connection to Wi-Fi networks, enabling communication with other devices and systems over the internet. This is critical for remote monitoring and data transfer.

Web Server Library allows the implementation of a lightweight web server on the ESP32, providing a user interface for system control and real-time data visualization.

(b) Key Functional Modules:

- Image Pre-processing: Image resizing, normalization, and transformation using 'torchvision transforms'.

  Conversion of image files into tensors for model training.

- Model Training and Evaluation:

  Training loop implemented using PyTorch for custom neural networks. CUDA Toolkit which supports GPU-based acceleration for faster model training and inference. Tracking and saving metrics including loss, accuracy, precision, recall, and F1-score.

- Data Handling:

  Custom dataset class to handle image-caption pairs. Train-validation split for accuracy on performance and evaluation.

1.6.7 Hardware Requirements

This project involves hardware with a laptop as the remote server for processing data and keeping the system running smoothly. A micro SD card to store captured images and other necessary information. A battery providing a reliable and portable power source. The ESP-32 CAM module for capturing images and performing basic processing tasks. Speakers are used to provide clear audio output. Wi-Fi or Ethernet connection is essential for communication between the ESP-32 CAM module and the laptop.

1.6.8 Functional Requirements

The functional requirements for the project are designed to ensure smooth operation and user-friendliness. This system utilizes the ESP-32 Cam module to capture high-resolution images providing the foundation for accurate processing. These images are then securely transmitted to a remote server using a Wi-Fi-based protocol for reliable and efficient data transfer. For word prediction, the system applies advanced pre-processing techniques like feature extraction and tokenization to convert the image data into structured input. A trained language model is then used to predict the most relevant words from the processed data.

To make the output accessible to visually impaired users, the project incorporates a neural-network-based TTS engine. This engine generates clear and natural speech to deliver a highly performance and good user experience.

Finally, RESTful APIs are developed to enable communication between the ESP-32 Cam and the server. These APIs handle image uploads, as well as retrieving word predictions ensuring the entire system works efficiently.

CHAPTER 2
LITERATURE REVIEW

Reviewing about the feature extraction and character recognition, Ia and Goodfellow proposed a system that combined edge detection techniques such as Sobel and Canny to highlight gradients in images and identify text boundaries. Their work [1] emphasized the application of Convolutional Neural Networks (CNNs) for recognizing and interpreting textual patterns. This study demonstrated the effectiveness of CNNs in character recognition, providing a solid foundation for OCR systems.

Similarly, Pant and Nirajan showed that CNNs trained on diverse datasets of Nepali and Sanskrit text achieve high accuracy in character recognition. Their research underscored the importance of developing OCR systems tailored to specific languages like Nepali and Sanskrit for accurate text-to-speech conversion [2].

Otsu and his team [3] highlighted the application of edge detection techniques, particularly Sobel and Canny, for image preprocessing in OCR. They also emphasized the critical role of CNNs in feature extraction and character recognition. Their work explored advanced methods such as Grapheme-to-Phoneme (G2P) Conversion and prosody modeling to generate natural-sounding speech. Furthermore, they discussed modern speech synthesis techniques like WaveNet and the Griffin-Lim algorithm for creating high-quality speech waveforms. On further transforming the OCR system, Meduri and Goyal developed an OCR system for Sanskrit using CNNs to address the challenges posed by its complex script. Their research demonstrated improved accuracy in recognizing Sanskrit text compared to traditional methods, showcasing the potential of deep learning techniques for script-specific OCR [4]. Similar to devanagari Nepali text, hindi text is also one of the complicated text for OCR for which, Bairagi and Dulal focused on creating an OCR system for the Hindi language, which tackled challenges like the complexity of the script and font variability. Their work [5] employed machine learning algorithms, resulting in significant improvements in recognition accuracy. This research highlighted the need for language-specific OCR solutions to achieve high performance. Hengaju worked on improving the Tesseract-OCR engine for Nepali text by implementing an image preprocessing pipeline with eight steps. Their pipeline significantly enhanced recognition accuracy for text in Nepali documents across different quality levels, demonstrating the importance of preprocessing in OCR systems [6].

The review over text processing and summarization showed that Mihalcea et al. [7, 8] introduced TextRank, a text summarization algorithm based on the PageRank model, for ranking sentences in extractive summarization tasks. They also highlighted the importance of the transformer architecture introduced by Vaswani et al., which revolutionized NLP by enabling efficient text processing through attention mechanisms. These advancements are crucial for building sophisticated text summarization systems. The evolution of the NLP was summarized by Nadkarni and colleagues explored the historical evolution of NLP and summarized its various subfields, including text analysis and summarization. They discussed machine learning approaches used for medical NLP applications, emphasizing the impact of modern NLP architectures and the potential of systems like IBM Watson in healthcare [9].

Khurana reviewed the evolution and current trends in NLP, focusing on different levels of natural language understanding and generation. Their study [10] provided insights into NLP applications and the challenges faced in implementing these technologies, making it relevant for human-assistive systems.

The techniques regarding speech synthesis was reviewed where, Jurafsky [11] detailed the process of converting text to speech, highlighting key components such as text analysis, phoneme extraction, Grapheme-to-Phoneme Conversion, and prosody modeling. These elements ensure that the generated speech has accurate pitch and natural intonation, making them vital for TTS systems.

Oord and his collaborators introduced WaveNet, a deep generative model for high-quality speech synthesis. Their research [12] demonstrated the model's ability to produce highly realistic speech, significantly advancing the capabilities of modern TTS systems. They also explored the Griffin-Lim algorithm for further improving speech quality.

Thierry Dutoit provided an in-depth study of speech synthesis techniques, including concatenative synthesis, parametric synthesis, and unit selection. His research covered essential aspects such as prosody modeling, pitch contour generation, and speech signal processing, offering a comprehensive understanding of TTS systems [13].

Ungurean presented [14] a TTS system for the Romanian language, emphasizing the integration of Speech Synthesis Markup Language (SSML) as a standard for document authoring and inter-module communication in TTS systems. This work highlights the importance of standardization for efficient and scalable TTS systems.

Some of the vision-based captioning and accessibility research paper were focused on improving the prevailing systems among which Ganeshan and his team proposed a model combining CNNs for feature extraction with LSTM networks for image digitization. By leveraging CNN architectures such as GoogleNet, AlexNet, and VGG16, their system captured detailed image features. The LSTM network then generated captions, which were converted into voice output using a TTS API, enabling visually impaired users to interact with visual content [15].

Safiya and Pandian developed [16] a real-time image captioning system based on a VGG16-LSTM model, integrated with a Raspberry Pi 4B for efficient processing. Their system, tested on datasets like Flickr8k, Flickr30k, and VizWiz, demonstrated high accuracy in generating captions. This approach improved accessibility for visually impaired individuals.

Along with the applications and techniques in computer vision, Richard Szeliski's work explored the foundational algorithms in computer vision, including feature extraction, object recognition, and motion estimation. He emphasized the application of these techniques in real-world domains such as augmented reality, robotics, and medical imaging, making his research essential for advancing vision-based systems [17].

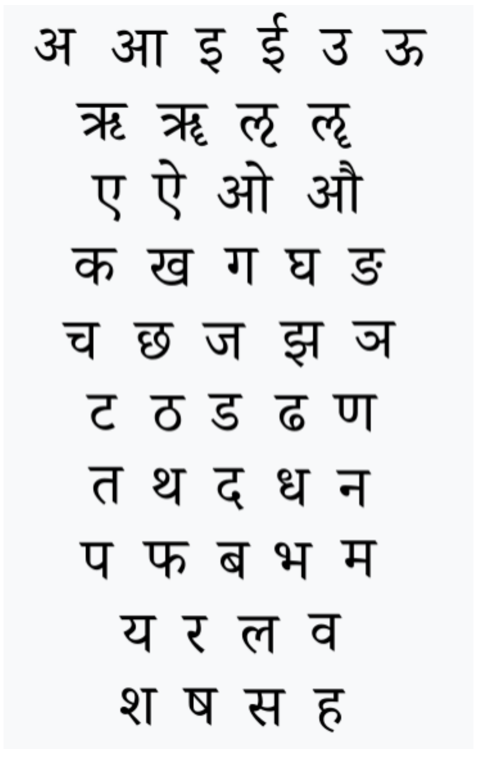
Sethi et al. [18] compared machine learning models—Neural Networks (NN), k-Nearest Neighbors (KNN), and Support Vector Machines (SVM) for accuracy in classification tasks. Their analysis showed that SVM outperformed the other models with a 99.38% accuracy, demonstrating its suitability for high-accuracy predictive modeling.

CHAPTER 3

RELATED THEORY

3.1 Devanagari Script

Nepali is one of the widely used language all over Nepal. Devanagari script is used in Nepali language. This language is unique from other languages spoken all over the world. As there are a lot of modifiers and half words in Nepali language. It is not as easy as learning english language. The Devanagari words has 36 different consonants and 12 vowels which are the basic characters. Not only the basic characters, there are special symbols called modifiers which are kept right, left, below and above of the consonants and the half words also plays huge role in the development of the Nepali words. The compound characters called "sabda" which are formed from combination of basic characters and the compound words called "waakya".



अ आ इ ई उ ऊ
ऋ ॠ ऌ ॡ
ए ऐ ओ औ
क ख ग घ ङ
च छ ज झ ञ
ट ठ ड ढ ण
त थ द ध न
प फ ब भ म
य र ल व
श ष स ह

Figure 3.1: Devanagari vowels and consonants

3.2 Machine Learning

The subset of AI which deals with scientific approach of computation from the statistical models that a computer system uses to perform the specific tasks without using explicit instructions,relying on patterns and inference instead. From the training set of data the mathematical models are built based on the algorithm and makes the predictions without being explicitly programmed to perform the certain tasks. Generally there are 3 types of machine learning. They are:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

3.3 Deep Learning

Deep learning is a subset of machine learning that uses complex algorithms to model high-level abstractions in data. Traditional machine learning requires extensive feature engineering but deep learning automates the process of feature extraction by training models end-to-end. The architecture of deep learning models typically consists of multiple layers of ANNs, where each layer learns different levels of abstraction from raw data. Deep learning algorithms based on neural networks utilize a back propagation method to adjust weights based on error gradients and optimizing the network for tasks like classification, regression, or generation. These models excel in tasks such as image recognition, speech processing, and natural language understanding but, traditional algorithms struggle to perform efficiently.

3.4 CNN

A CNN is a deep learning model specifically designed to process and analyze visual data. CNNs looks after recognizing patterns in images and other grid-like data through the use of multiple layers that automate feature extraction and classification. The architecture is composed of several key layers, each with a unique role in transforming and processing the data. CNNs have become the first step of computer vision applications that provides impressive performance in tasks like object detection, facial recognition, and semantic segmentation. Convolutional operations applies filters to input data to learn hierarchical patterns, followed by pooling to reduce dimensionality and improve computational efficiency.

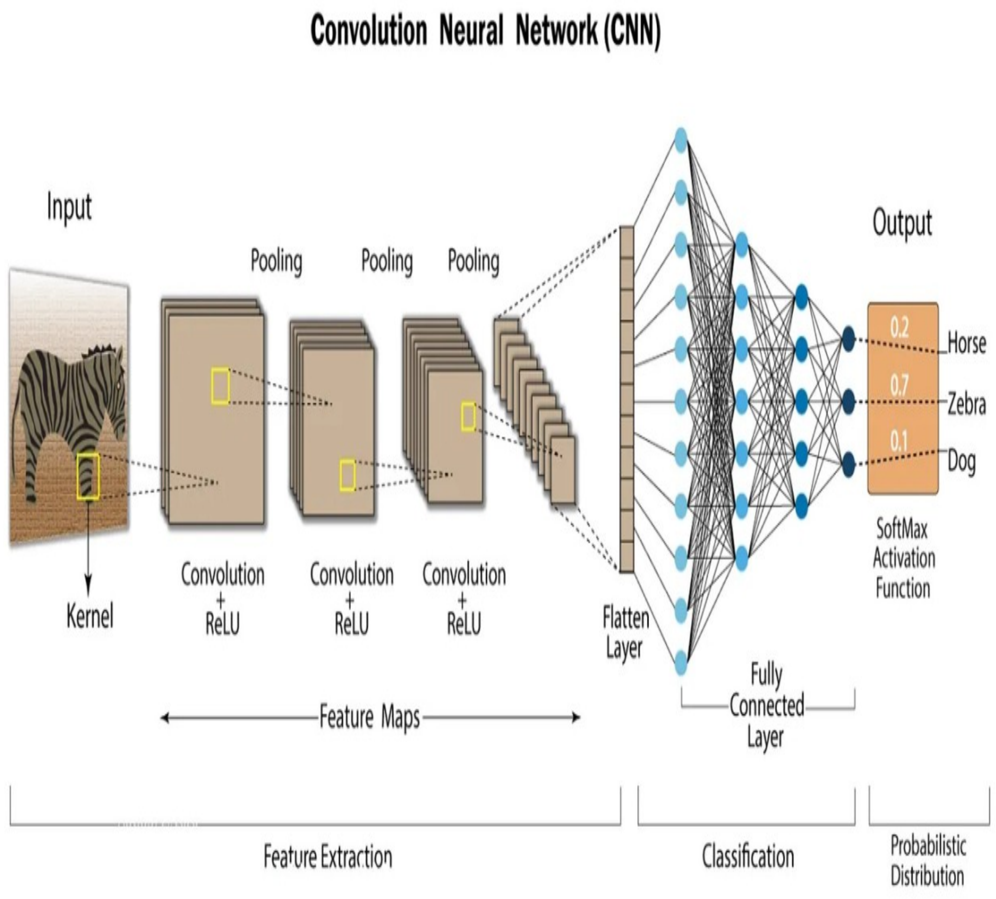


Figure 3.2: CNN Architecture

3.4.1 Convolutional Layer

The convolutional layer applies convolutional filters (also called kernels) to input data, extracting local features from the input image. Each filter detects specific patterns such as edges, textures, or shapes, which are essential for higher-level understanding. The convolution operation is performed by sliding the filter over the input image (or feature map) and computing the element-wise product followed by a sum to produce a single output. As the filter moves across the input, it generates a feature map, which highlights the presence of detected patterns. The number of filters and their size determines the model's ability to learn complex features, but it also increases the computational cost. A deeper CNN with more filters can capture more intricate patterns but requires more memory and processing power.

3.4.2 Pooling Layer

The pooling layer is responsible for down-sampling the feature maps produced by the convolutional layer that reduces their spatial dimensions while retaining the most important information. The most common pooling operation is max pooling, where a sliding window is applied to the feature map and the maximum value within the window is selected. This reduces the size of the outputand makes the model more computationally efficient and robust to small translations or distortions in the input. Pooling helps prevent overfitting by introducing spatial invariance and reducing the number of parameters that need to be learned in subsequent layers.

3.4.3 Flatten Layer

The flatten layer is a simple but crucial component in CNNs that converts multi-dimensional input data or 2D vector into a one-dimensional vector. This transformation is necessary because typically fully connected (dense) layers expect 1D data as input. The network can be transitioned from the feature extraction phase to the decision-making phase by flattening the multi-dimensional data. Flattening is essentially a reshaping operation that allows the model to process the learned hierarchies into features for classification or regression.

3.4.4 Dense Layer

The dense layer is a fully connected layer where each neuron is connected to every neuron in the previous layer. It plays a significant role in integrating information extracted by earlier layers. Each neuron in the dense layer computes a weighted sum of its input and applies a bias and passes the result through an activation function (ReLU or sigmoid) to introduce non-linearity. Dense layers are typically used for tasks like classification, where they map the learned features to the final output. The weights and biases in the dense layers are learned through back propagation by which gradients of the loss function with respect to each parameter are calculated and updated iteratively.

3.5 NLP

NLP refers to the field of AI, concerned with the interaction between computers and natural languages which performs wide range of tasks like text classification, sentiment analysis, machine translation, named entity recognition, and text generation[10]. In a NLP model, text is first tokenized by breaking down into smaller components like words, characters, or subwords. Each token is then mapped to a numerical representation using techniques such as word embeddings. These embeddings capture semantic relationships and contextual meanings of words. Models like RNNs, LSTM networks and transformer architectures are commonly used to process sequential data. In tasks like Image digitization, CNNs are typically employed to extract visual features from images and combined with NLP models for generating text descriptions. The training process often involves back propagation and optimization techniques to minimize loss functions like cross-entropy. Evaluation metrics like accuracy, precision, recall, and F1-score are used to assess model performance.

3.6 OCR

OCR is a technology that enables the conversion of text from images, scanned documents, or handwritten notes into machine-readable and editable digital text. The process involves several key steps: first, the input image is preprocessed to enhance text visibility through techniques like binarization, noise reduction, etc. Next, text detection algorithms identify regions containing text, often using methods like edge detection or deep learning models such as CNNs. Character recognition is performed, where individual characters are extracted and classified using machine learning models like Support Vector Machines (SVMs) or deep neural networks. Finally, post-processing techniques, such as language modeling and spell-checking, refine the output to improve accuracy. Modern OCR systems uses advanced machine learning and deep learning techniques, making them highly accurate and capable of handling diverse fonts, languages, and complex layouts.

3.7 Hardware Tools

3.7.1 ESP 32-S CAM module

The ESP32-S CAM module is a powerful and compact IoT device with the ESP32-S microcontroller at its core. This module combines a dual-core LX6 processor running at up to 240 MHz with integrated Wi-Fi and Bluetooth capabilities. It is equipped with a camera interface and supports OV2640 and OV7670 cameras which helps to capture image and video streaming. Also its 520 KB of SRAM and 4 MB of PSRAM provide better memory for handling image processing tasks. The ESP32-S CAM is widely used in applications such as surveillance systems, AI-powered image recognition, and IoT-based automation due to its fast and flexible processing power and connectivity features.

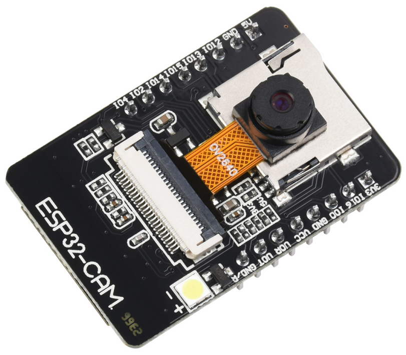


Figure 3.3: ESP32-S CAM module

3.7.2 MicroSD Card

The micro SD card allows for extended storage capacity which is essential for applications requiring data logging or multimedia storage. The microSD cards formatted with the FAT32 file system can be integrated with ESP32-S CAM module. Cards up to 4 GB or more, depending on firmware support can be used for data storage. This feature is particularly advantageous in scenarios where images or logs need to be stored locally for later retrieval or processing. The microSD interface ensures reliable read and write operations which makes it an indispensable component for projects with significant data handling requirements.

3.7.3 Speakers or Headset

For audio output the system employs speakers or headsets.It is essential for delivering text-to-speech feedback or audible alerts. These are connected via a 3.5mm audio jack or directly to the ESP32-S CAM's DAC pins. When using the DAC pins an external amplifier may be added to boost signal for clear and loud audio output. The speakers typically have an impedance range of 4 to 16 ohms to ensure compatibility with the ESP32's audio output specifications. This capability is especially important in accessibility-focused projects where audio serves as the primary medium of interaction.

3.7.4 Push button Switch

The simple hardware component used for user input or control in electronic systems. It operates as a momentary switch, closing the circuit only when pressed but default open state upon release. In the context of an ESP32-S CAM based project, the push-button switch can serve as initiating device operations or allowing to click or capture the photos. The switch connects to one of the GPIO pins of the ESP32-S micro-controller, with pull-up or pull-down resistors configured to ensure stable logic levels and avoid floating states. when the button is pressed capturing an image or activating a feature can be the better use of it. Its compact design makes it a valuable interface component in embedded systems like as this project.

3.8 Software Tools

3.8.1 PyTorch

PyTorch is an open-source deep learning framework used for implementing and training neural networks. Its dynamic computation graph and seamless integration with GPUs make it highly efficient for research and development. In this project, PyTorch is utilized to design and train a neural network model that processes images and generates meaningful predictions, forming the backbone of the system.

3.8.2 Torchvision

Torchvision is a complementary library to PyTorch, providing pre-trained models, datasets, and image transformation utilities. It is used in this project for essential image processing tasks such as resizing, normalization, and converting images into formats compatible with PyTorch models.

3.8.3 Progress Monitoring

Progress monitoring during computational tasks is facilitated through a library that provides real-time feedback via progress bars. This functionality enhances the user experience by offering visual updates during iterative processes, such as model training and validation. By allowing developers to track the system's progress efficiently, it contributes to better monitoring and debugging during model development.

3.8.4 Dataset Management

Efficient dataset handling is achieved through a utility module that supports large-scale data operations. This tool provides essential features such as batch processing, data shuffling, and splitting datasets into training and validation subsets. These capabilities are critical for ensuring efficient data preprocessing and robust model training. By streamlining data flow, the module optimizes the performance of the neural network and enhances overall system reliability.

3.8.5 Arduino IDE

The Arduino Integrated Development Environment (IDE) is an open-source platform used for writing and uploading code to microcontrollers, such as the ESP32 CAM module employed in this project. The ESP32 CAM, with its integrated camera, is programmed to perform configuring resolution, brightness, contrast, and other parameters for optimal image capture. Capturing images for image processing and analysis tasks. The ESP32 CAM bridges hardware and software, enabling efficient image acquisition and processing, making it a critical component of the system.

CHAPTER 4

METHODOLOGY

4.1 System Block Diagram

The top-level block diagram of our system is as follows:

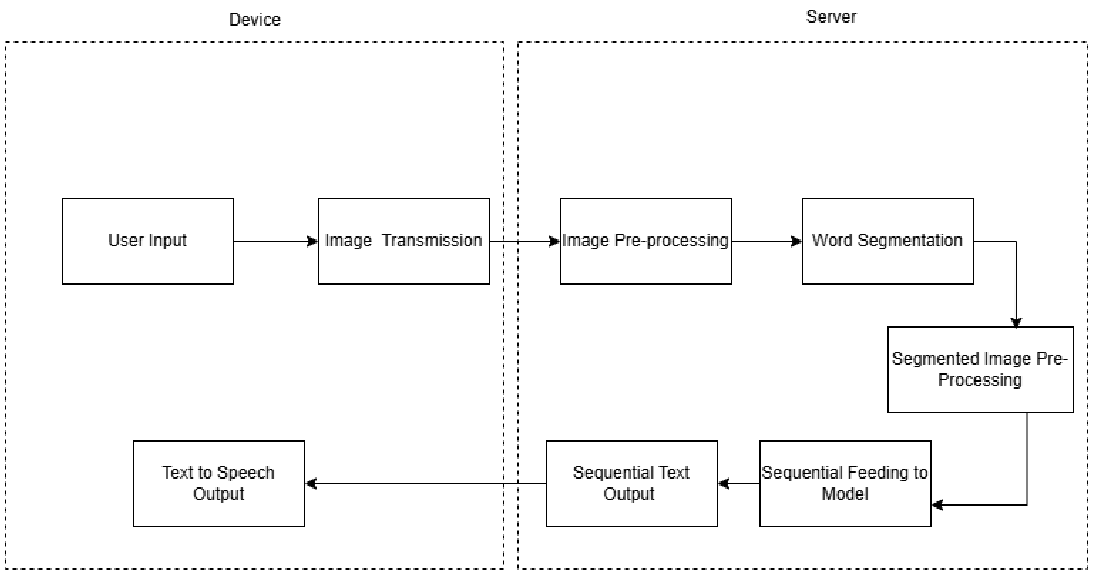


Figure 4.1: Top level system block diagram

The system consists of device and Server. The process is divided into several key stages, which work together to achieve accurate text extraction and speech output.

4.1.1 Device Component

The device is responsible for capturing the input image and delivering the final speech output. It consists of the following stages:

- User Input: The user captures an image that contains Nepali text from the ESP32-CAM module by pressing the push button switch.
- Image Transmission: The captured image is transmitted to the server via HTTP (over WiFi) protocol where ESP32 acts as an HTTP server and laptop acts as an HTTP client.

- Text-to-Speech Output: Once the text is predicted, as per the predicted caption the audio is searched in the device and converted into speech for the user through the speaker.

4.1.2 Server Component

The server is responsible for processing the received image and extracting text. It consists of the following stages:

- Image Pre-processing: The received image is enhanced by scaling the image, removing noise, adjusting contrast, and binarizing for better OCR performance.
- Word Segmentation: The image is divided into individual words to improve recognition accuracy. Word segmentation is done after contour detection and bounding box creation for every words on a sentence.
- Segmented Image Pre-processing: Each segmented portion undergoes additional refinement before text extraction. For this the image are resized to 64x64 and converted to tensor.
- Sequential Feeding to Model: The processed image segments are sequentially fed into the model. The model recognizes the text from each segment, and these recognized texts are then mapped to the respective images for caption pairing.
- Sequential Text Output: Once the model is trained, it generates the recognized text for each image segment in order, making sure the text matches the sequence of the images.

4.2 Hardware Setup

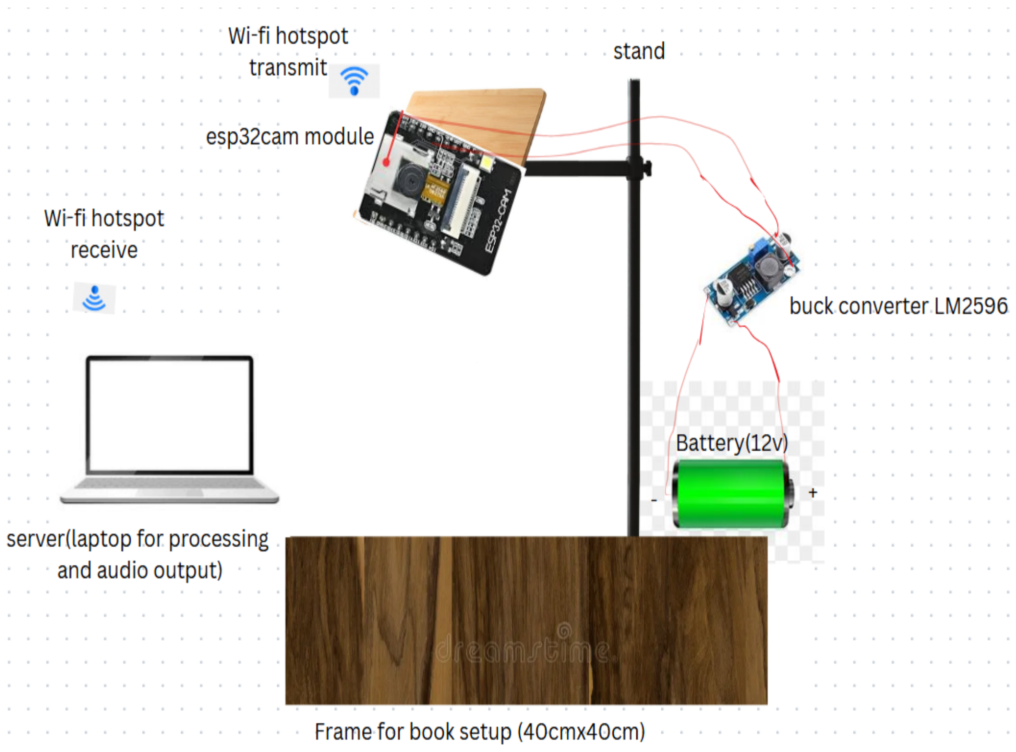


Figure 4.2: Hardware Setup

The core component of this system is the ESP32-CAM module, which is mounted on a 40 cm tall stand and angled downward to capture images of a document or book placed on a 40 cm x 40 cm wooden frame below. The ESP32-CAM acts as a wireless camera, transmitting captured images to a server (a laptop) via Wi-Fi hotspot for further processing. To power the ESP32-CAM, a 12V battery is used, but since the module operates at a lower voltage, a buck converter (LM2596) is integrated into the circuit. The buck converter steps down the voltage from 12V to 5V, ensuring safe and efficient power delivery to the camera module. The ESP32-CAM then captures an image of the text and wirelessly transmits it to a laptop, which serves as the processing unit. The laptop receives the image, runs OCR (Optical Character Recognition) software to extract the text, and then uses a Text-to-Speech (TTS) engine to convert the recognized text into audio output.

4.3 Input section

First the input (image) will be collected from the ESP-32 CAM module and sent to the remote server (laptop) which will then be pre-processed using transform functions and fed to the OCR based encoder decoder model and prediction will be done through the trained model which further will be processed for audio output using the TTS model and the output will be originated from the speaker of the remote server.

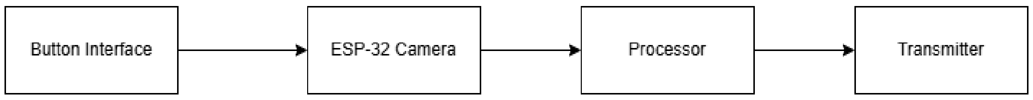


Figure 4.3: Image input block diagram

The system comprises four interconnected modules that enable image capture, processing, and wireless transmission.

Button Interface converts physical button presses into digital signals for the ESP-32 Camera or Processor. ESP-32 Camera combines an ESP32 microcontroller (with Wi-Fi hotspot) and a camera module, handles initial data compression and wireless communication setup. Processor does image pre-processing (noise reduction, edge detection). Running machine learning OCR and managing data flow between the camera and transmitter. Wireless communication module sends processed data to external systems using HTTP protocol.

4.3.1 Word Segmentation

The images captured from the camera undergo word segmentation that allows for precise captioning and text processing. The segmentation process begins with resizing the image, followed by grayscale conversion, Gaussian blurring, and thresholding. These pre-processing steps enhance the image quality, making it easier to detect and isolate individual words. After this, contour detection is applied to identify distinct regions in the image, and bounding boxes are created around these regions. These boxes define the boundaries of each word. The word segmentation process is crucial for the accuracy of NLP tasks. This segmented text can then be used for tasks such as captioning or other NLP applications where clear boundaries between words are essential for accurate interpretation.

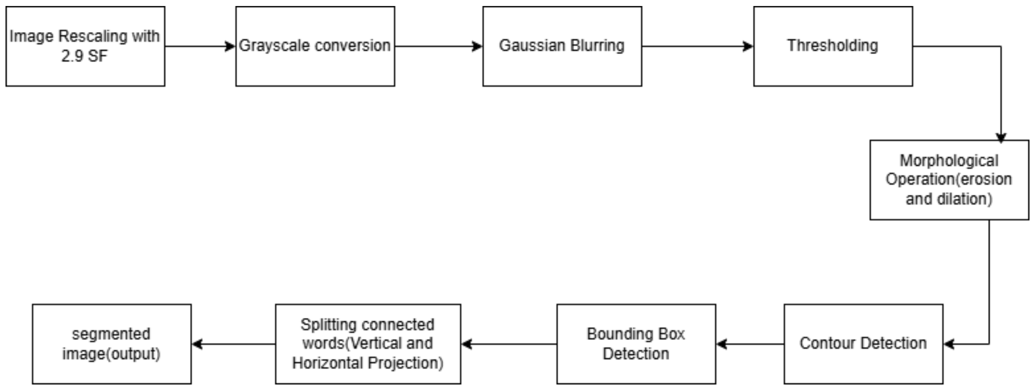


Image Rescaling with 2.9 SF
Grayscale conversion
Gaussian Blurring
Thresholding
Morphological Operation(erosion and dilation)
Contour Detection
Bounding Box Detection
Splitting connected words(Vertical and Horizontal Projection)
segmented image(output)

Figure 4.4: Word Segmentation Block Diagram

The above block diagram shows the process of word segmentation.

- Image Rescaling with 2.9 SF: The main purpose is to adjust the image size to standardize dimensions and improve processing efficiency. Scaling the image by a Scale Factor (SF) of 2.9. It is done as rescaling compensates for low-resolution inputs or aligns text size for consistent segmentation.
- Gray-scale Conversion: It simplifies the image by reducing color complexity. Converting RGB images to a single-channel grayscale using luminance weighting. Gray-scale conversion eliminates color noise and reduces computational load for subsequent steps.
- Gaussian Blurring: It smooths the image to reduce high-frequency noise. Applying a Gaussian filter kernel (5×5 kernel) to blur the image. Gaussian blurring prepares the image for thresholding by suppressing minor artifacts and enhancing text edges.
- Thresholding: It is done to binarizes the image to separate text (foreground) from the background. Using adaptive thresholding (Otsu's algorithm) to dynamically determine the optimal threshold value. Binary image where text pixels are white (255) and background is black (0) is obtained.
- Morphological Operations (Erosion and Dilation): It refines the binary image to remove noise and merge fragmented text regions. Erosion is the process shrinking white regions using a structuring element (3×3 kernel) to disconnect

loosely joined characters. Dilatio is the process to expand white regions to fill gaps within characters/words. Morphological operation corrects over-segmentation or under-segmentation caused by thresholding.

- Contour Detection: It precisely outlines word/character boundaries.It uses edge detection algorithms to trace object boundaries.It is done to handle irregularly shaped text or skewed layouts missed by bounding boxes.
- Bounding Box Detection: It localizes segmented words/characters. It is done by drawing rectangles around regions identified by projection profiles.
- Splitting Connected Words (Vertical and Horizontal Projection): It identifies word/character boundaries using pixel density analysis. Horizontal projection is the process which sums pixel intensities row-wise to detect lines of text. Vertical projection is the process sums pixel intensities column-wise to identify gaps between words/characters.
- Segmented Image (Output): A cleaned binary image where text regions are isolated and background is suppressed. This image feeds into word-splitting algorithms.

4.3.2 Image Pre-processing

After the image of segmented word is obtained the pre-processing of the segmented words are carried out where the image are resized to 64x64 and normalization and tensor conversion is done by which the model can be initialized with the Image captioning process.

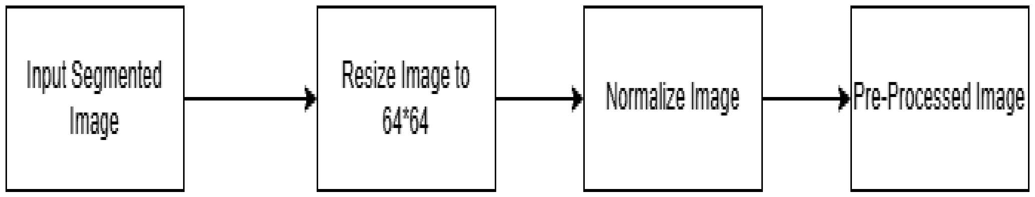


Figure 4.5: Image pre-processing of segmented words block diagram

The above figure shows the step-by-step process of pre-processing of segmented word images. These steps ensure that the images are standardized and optimized for further processing, such as feature extraction or model training.

- Input Segmented Image: The process starts with an image containing a single word that has been separated from a larger document or page. This image serves as the input and contains the necessary details for further processing.
- Resize Image to 64×64: Since images can come in different sizes, the segmented word image is resized to a fixed 64×64 pixels. Keeping all images the same size helps maintain consistency, making it easier for the model to process them. Simple resizing techniques are used to ensure the image remains clear and readable.
- Normalize Image: After resizing, the image is adjusted so that its pixel values fall within a standard range, usually between 0 and 1. This step helps reduce the effect of differences in brightness, lighting, or contrast, ensuring that all images are processed in a similar way.
- Pre-Processed Image: At this stage, the image is now properly sized and adjusted, making it ready for further analysis. With a uniform size and balanced pixel values, the image is well-prepared for use in machine learning models or other processing tasks, ensuring smoother and more accurate results.

4.4 Flowchart

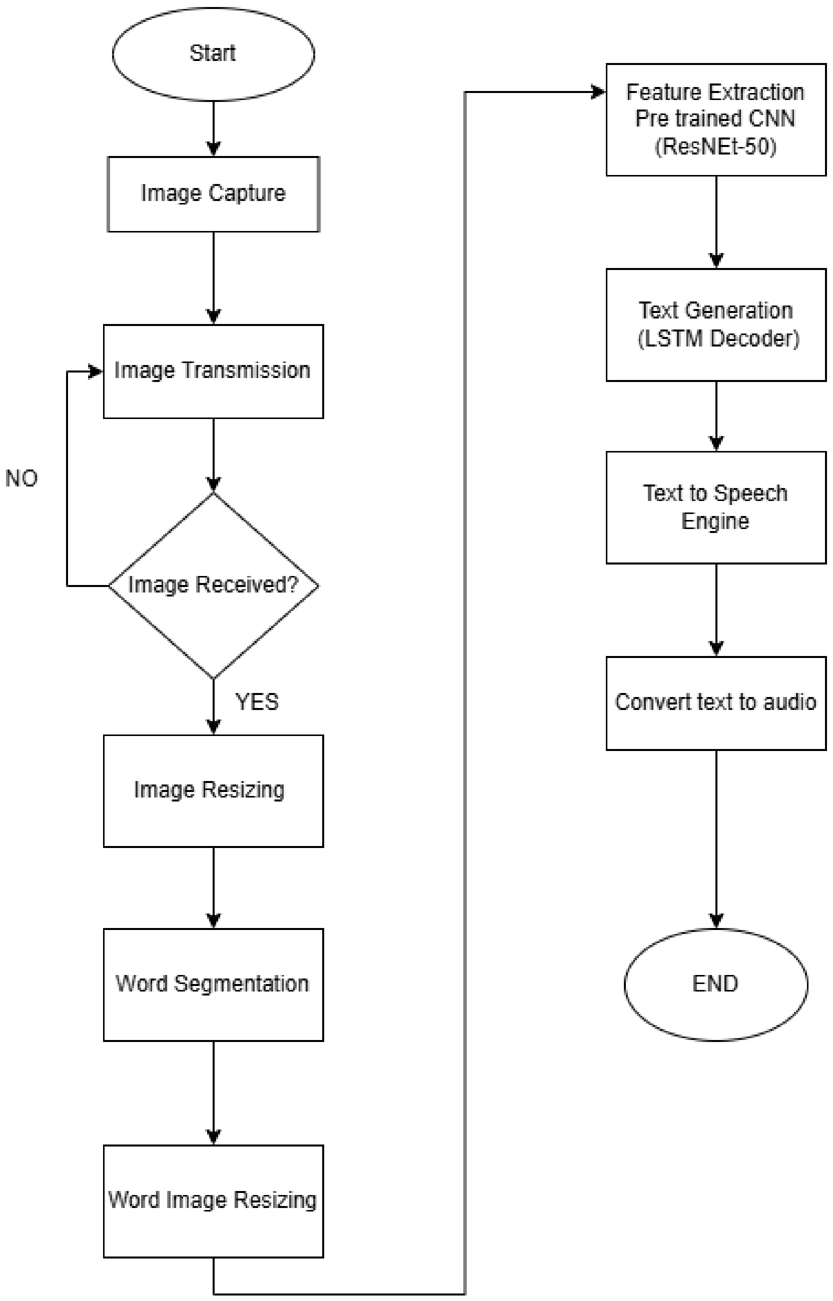


Figure 4.6: System Working Flow Chart

4.4.1 System Working Flowchart

The flowchart shows the working of our system. The workflow begins with image capture, where a camera module ESP-32 CAM snaps a photo of text-containing content, such as a document or signage. The captured image is wirelessly transmitted to a processing unit, typically a remote server, using protocols like Wi-Fi or Bluetooth.

A critical checkpoint follows: the system verifies whether the image was successfully received. If transmission fails, it retries or alerts the user; if successful, preprocessing begins.Next, the image undergoes resizing to standardize its dimensions, ensuring compatibility with downstream machine learning models. This step reduces computational overhead and prepares the image for word segmentation, where advanced computer vision techniques—like thresholding, morphological operations, and projection profiling—isolate individual words or text blocks. Each segmented word is resized again to a uniform format, optimizing it for feature extraction.The core machine learning phase starts with feature extraction using ResNet-50, a pre-trained convolutional neural network. This model analyzes each word image, distilling it into high-level numerical features that capture essential text characteristics. These features are then fed into an LSTM decoder, a recurrent neural network adept at sequence prediction. The LSTM deciphers the features sequentially, generating coherent text strings. Finally, the recognized text is passed to a text-to-speech (TTS) engine, which synthesizes natural-sounding audio using models. The audio output is delivered through speakers.

4.4.2 Training Flowchart

Flowchart shown in figure 6.7 shows the basic training workflow. The training process begins by initializing essential libraries and frameworks, such as TensorFlow or PyTorch, which provide tools for deep learning, data handling, and image processing. Next, paired datasets of images and captions (e.g., COCO or Flickr30k) are loaded. These datasets are cleaned and standardized—images are resized and normalized, while captions are preprocessed to remove noise like punctuation or uppercase letters.The data is then split into training and validation sets, ensuring the model learns generalizable patterns without overfitting to specific examples. A typical split balances learning and evaluation.The model architecture is built using an encoder-decoder framework.The encoder (a pre-trained CNN like ResNet-50) converts images into compact feature vectors, capturing visual details like objects and scenes.The decoder (an LSTM or Transformer) generates descriptive captions word-by-word, treating the task as a sequence prediction problem.During training and validation, batches of images and captions are fed to the model. The model learns by minimizing a loss function (e.g., cross-entropy) that measures how well its generated captions match the ground truth. Validation metrics like BLEU score simultaneously track caption quality on unseen data.To optimize training efficiency, early stopping monitors validation loss. If performance plateaus for several epochs, training halts automatically, preventing wasted resources and overfitting.Finally, the trained model weights, tokenizer (which maps words to numerical IDs), and training metrics (loss curves, accuracy scores) are saved. These outputs enable future deployment, such as auto-generating alt-text for images on websites or assisting visually impaired users through real-time captioning.

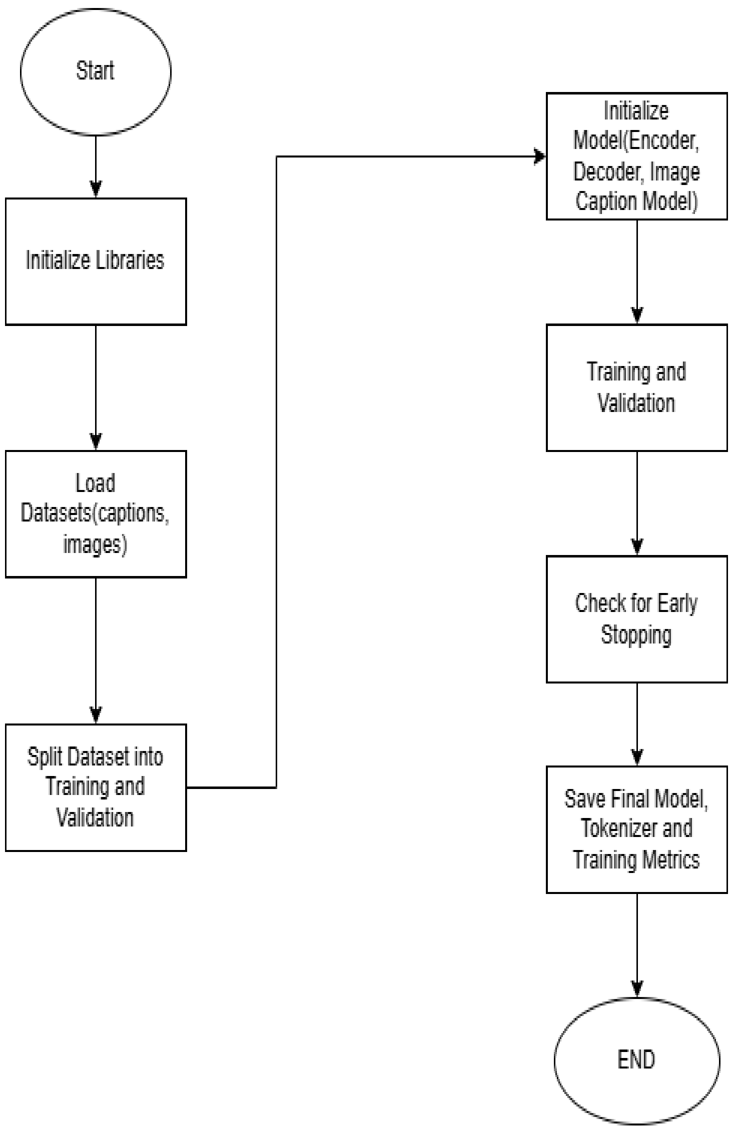

Start
Initialize Libraries
Load Datasets(captions, images)
Split Dataset into Training and Validation
Initialize Model(Encoder, Decoder, Image Caption Model)
Training and Validation
Check for Early Stopping
Save Final Model, Tokenizer and Training Metrics
END

Figure 4.7: Training Flow Chart

4.4.3 Image captioning

The image captioning system is a sophisticated integration of computer vision and natural language processing (NLP) techniques, designed to generate descriptive textual narratives from visual inputs.

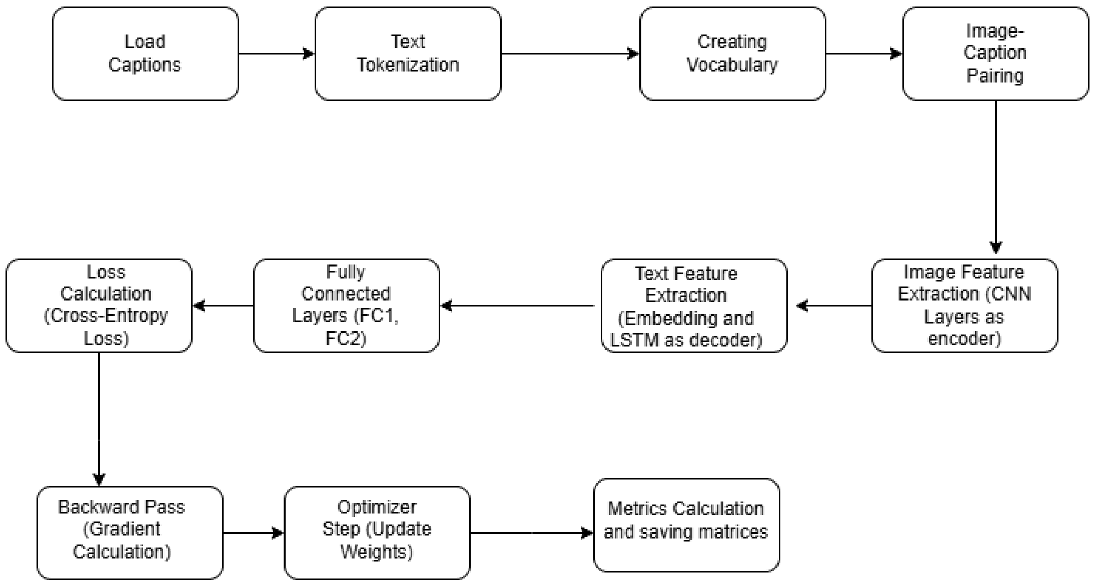

Load Captions
Text Tokenization
Creating Vocabulary
Image-Caption Pairing
Image Feature Extraction (CNN Layers as encoder)
Text Feature Extraction (Embedding and LSTM as decoder)
Fully Connected Layers (FC1, FC2)
Loss Calculation (Cross-Entropy Loss)
Backward Pass (Gradient Calculation)
Optimizer Step (Update Weights)
Metrics Calculation and saving matrices

Figure 4.8: Image captioning block diagram

The process begins with data preparation, where paired image-caption datasets are loaded and preprocessed. Captions undergo tokenization, breaking text into discrete units and a vocabulary is constructed to map these tokens to numerical identifiers. This step ensures linguistic consistency and enables the model to process language computationally. Images are standardized through resizing and normalization, while image-caption pairs are structured to align visual content with contextual descriptions, forming the foundation for supervised learning.Central to the architecture is an encoder-decoder framework. The encoder leverages a pre-trained convolutional neural network (CNN), such as ResNet-50, to extract hierarchical visual features from images. These features encapsulate object identities, spatial relationships, and scene semantics, condensing the image into a compact feature vector. The decoder, typically a long short-term memory (LSTM) network or Transformer, processes these visual features alongside embedded textual inputs. An embedding layer converts tokenized captions into dense vector representations, capturing semantic relationships between words. The LSTM then sequentially predicts the next word in the caption, leveraging contextual dependencies to generate coherent and grammatically sound descriptions. During model training, the system optimizes its parameters through iterative forward and backward propagation. A cross-entropy loss function quantifies the disparity between predicted and ground-truth captions, guiding the model to refine its predictions. Optimization algorithms, such as Adam, compute gradients and adjust weights to minimize this loss. To enhance generalization, a validation loop evaluates performance on unseen data, employing metrics like BLEU and METEOR to assess caption relevance and fluency. Early stopping mechanisms halt training if validation performance plateaus, balancing computational efficiency with model robustness.Upon convergence, the finalized model—alongside its tokenizer and training metrics—is archived for deployment.

4.5 Software Development

As part of the project, we developed algorithms and APIs to ensure smooth interaction between the Natural Language Processing (NLP) and Text-to-Speech (TTS) modules. The ESP32-CAM module was configured to capture input from books containing Devanagari scripts, enabling efficient text extraction. Additionally, we designed a user-friendly interface to make the system easy to control and allow users to access the converted text effortlessly.

4.5.1 Datasets

Dataset collection and creation is the most important part in every machine learning process. Due to the unlabeled dataset over the internet, we created the data and labeled on our own over 65000 data. we split the dataset in such a way that training set contains 80 percent images and validation set contains 20 percent images of each class. Here, image size was adjusted with scaling factor 2.9 and the segmentation was done after contour and bounding box detection.

4.5.2 Data Augmentation

It is used to increase the training data set by applying various transformations to existing images. This approach helps improve the model generalization by introducing variations in the data, making it less prone to overfitting. The transformations implemented include adding grain (noise) to the images, which introduces Gaussian noise to simulate real-world imperfections. Then inverts the colors of the images, providing variation in color representation. Additionally, adjusts the brightness of the images, simulating different lighting conditions. These augmented images are saved with specific prefixes to distinguish them from the original images. The captions corresponding to these augmented images are also updated and saved, ensuring each augmented image is properly labeled. This process allows the model to train on a more diverse set of images, improving its ability to generalize and perform well in real-world scenarios.

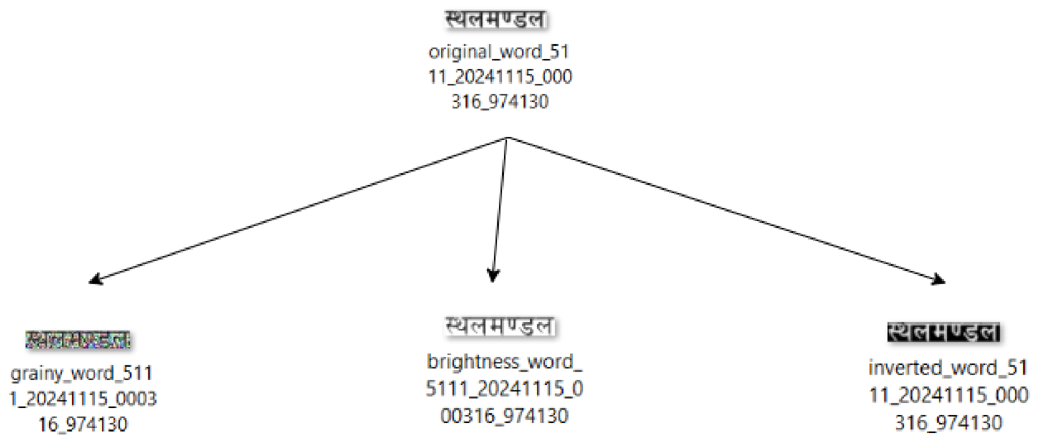

स्थलमण्डल
original_word_5111_20241115_000316_974130
स्थलमण्डल
grainy_word_5111_20241115_000316_974130
स्थलमण्डल
brightness_word_5111_20241115_000316_974130
स्थलमण्डल
inverted_word_5111_20241115_000316_974130

Figure 4.9: Data augmentation

4.5.3 Model Architecture

The model is built using a classical encoder-decoder architecture with an attention mechanism, specifically designed to handle sequence-to-sequence learning with visual inputs. At its core, it uses ResNet-50 as the visual encoder, a powerful convolutional neural network (CNN) that has been pre-trained on ImageNet. This encoder extracts essential features from input images, processing them through multiple convolutional layers until reaching the final feature map. To ensure a fixed-size representation, an adaptive average pooling layer compresses this feature map, followed by a fully connected layer that projects it into a lower-dimensional space. This step is crucial as it aligns the image features with the embedding dimension required for the decoder. On the text generation side, the decoder is built using LSTM (Long Short-Term Memory) networks, which are well-suited for handling sequential data. It takes the processed image features along with embedded text sequences as inputs and predicts the next word at each time step. The decoder includes an embedding layer that converts words into dense vector representations, allowing the model to understand their relationships. The LSTM then processes these vectors, learning dependencies between words and ensuring that the generated sequence is contextually accurate. Finally, a fully connected layer maps the LSTM's outputs to a vocabulary, predicting words one by one to form a meaningful sentence. To enhance the interaction between visual and textual data, the model employs an adaptive multimodal fusion mechanism, which blends both types of information effectively. During training, images pass through the ResNet-50 encoder, and the extracted features are merged with the embedded text representations before entering the LSTM decoder. The model learns using a cross-entropy loss function and follows the teacher forcing technique, where the actual captions are provided during training to improve accuracy. The training process is optimized using the Adam optimizer with a learning rate of 0.0001, and the model's performance is evaluated using BLEU scores, which compare generated captions to reference captions. To prevent overfitting and ensure stable training, the model incorporates techniques like early stopping, scheduled learning rate adjustments, and dropout regularization. The dataset is carefully prepared using a custom tokenizer, which maps words to indices and structures them appropriately for training. On the image side, pre-processing steps such as resizing and normalization ensure consistency across batches. Additionally, to maintain hardware efficiency during long training sessions, periodic system cooling mechanisms are implemented.

Layer Type	**Output Shape**	**Param #**
Encoder.ResNet.Conv2d	-	9,408
Encoder.ResNet.BatchNorm2d	-	128
Encoder.ResNet.ReLU	-	0
Encoder.ResNet.MaxPool2d	-	0
Encoder.ResNet.Sequential	-	215,808
Encoder.ResNet.Sequential	-	1,219,584
Encoder.ResNet.Sequential	-	7,098,368
Encoder.ResNet.Sequential	-	14,964,736
Encoder.AdaptiveAvgPool2d	[1,1,2048]	0
Encoder.Projection	[256]	524,544
Decoder.Embedding (30, 256)	[None, 256]	7,680
Decoder.LSTM (256→1024)	[None, 1024]	5,251,072
Decoder.FC (1024→30)	[30]	30,750
Total Trainable Parameters:		29,322,078
Encoder Parameters:		24,032,576
Decoder Parameters:		5,289,502

Table 4.1 Model Architecture

4.5.4 Convolutional Neural Network (CNN) Architecture

The CNN architecture in this project is specifically designed to process 64*64 RGB images for image-captioning tasks. The architecture comprises two convolutional layers.

First Convolutional Layer:

- Applies 64 filters with a 3×3 kernel size.
- Includes padding to preserve the spatial dimensions of the image.
- ReLU activation introduces non-linearity, allowing the network to learn basic image features like edges and textures.

Second Convolutional Layer:

- Applies 128 filters with a 3×3 kernel size and padding.
- ReLU activation is applied to learn more complex features.

The output of the convolutional layers is flattened and passed through a fully connected layer (fc1) to generate a vector representation of the image features. These features are then integrated with a tokenized and processed caption sequence using an LSTM network.

This custom CNN architecture ensures a balance between computational efficiency and feature extraction, making it well-suited for tasks involving smaller image sizes.Simultaneously, the captioning data is tokenized and embedded. The embeddings are processed through an LSTM network, capturing temporal dependencies and the context of the captions. The final hidden state of the LSTM is concatenated with the image feature vector to generate a combined feature representation. This vector is passed through another fully connected layer (fc2) to generate the final predictions.This integration of CNN for image processing and LSTM for sequence modeling makes the model robust and suitable for generating accurate results.

4.5.5 TTS

The TTS process begins with text input where the system receives textual data processed from earlier stages. The audio recorded is then mapped as per the predicted caption in the directory and is played as audible speech through audio output, which can be played back via speakers or headphones.

4.5.6 Earphone/Speaker

Finally, the generated speech is output through earphones or speakers, allowing the visually impaired user to listen to the content of the captured book page. This block represents the audio output device used by the user to hear the synthesized speech.

4.5.7 Hardware Circuit Diagram

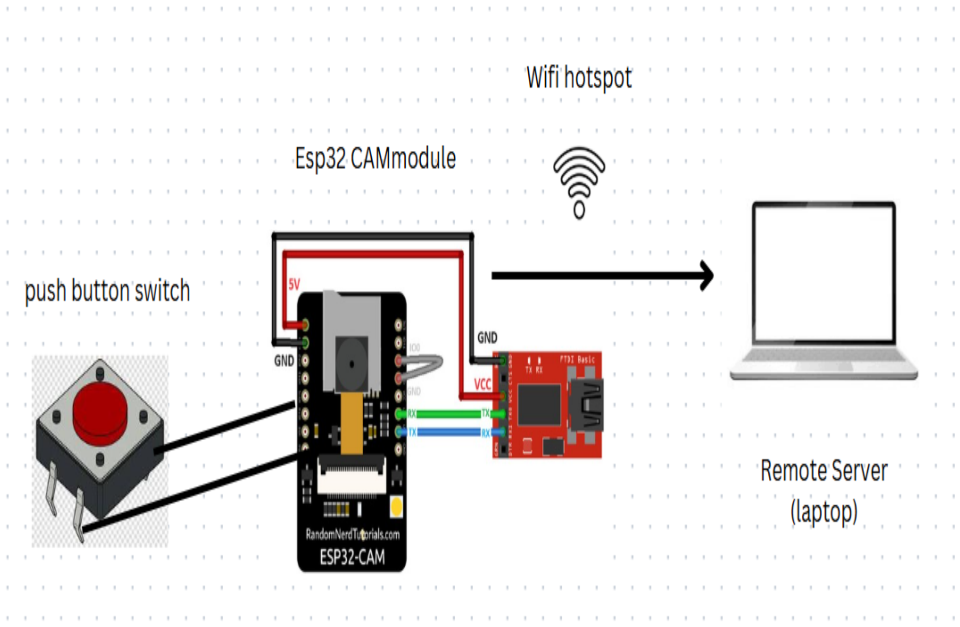
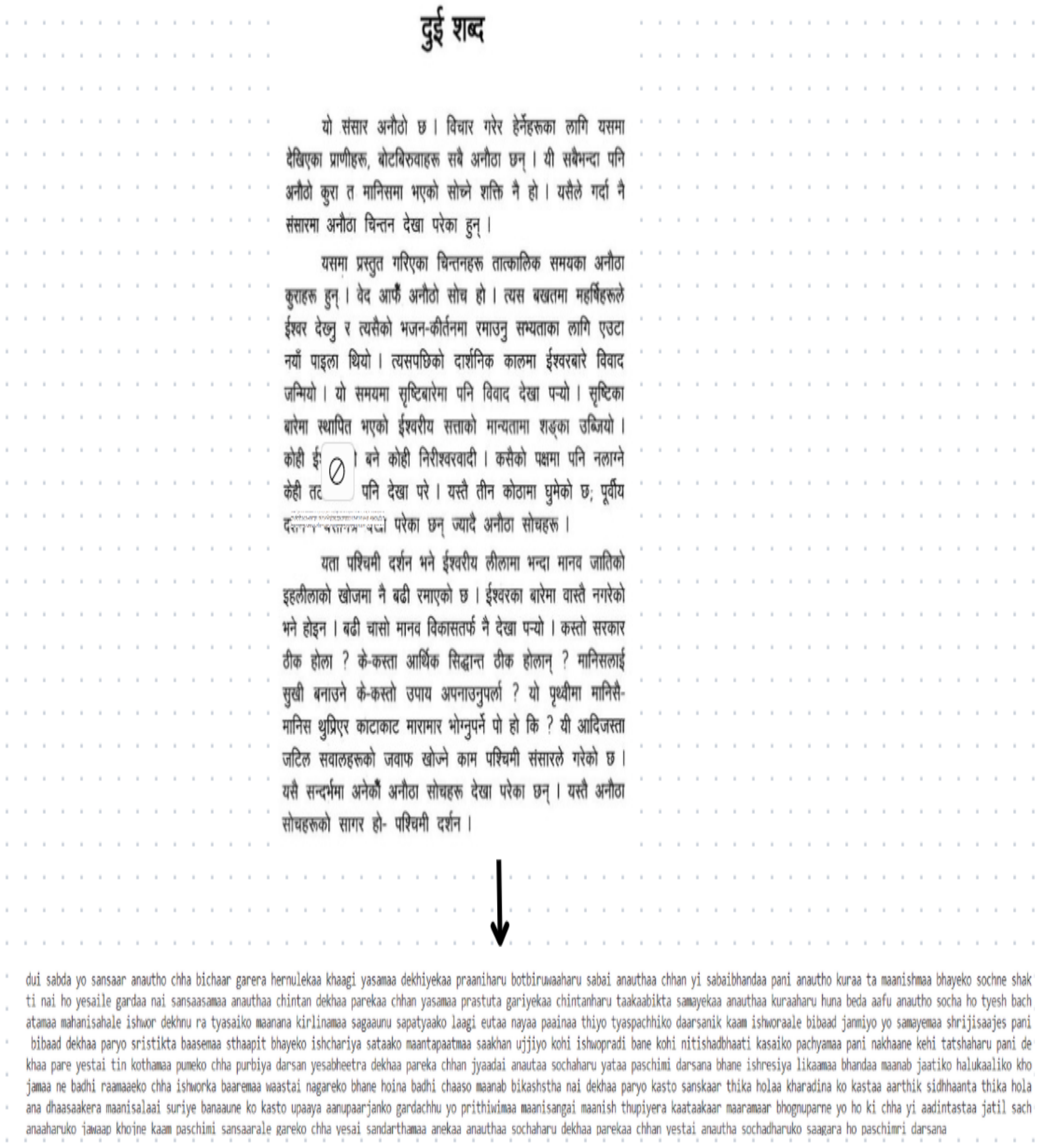


Figure 4.10: Hardware Circuit Diagram

As per the instruction from the push button switch the esp32 CAM module captures the image and through its Wi-Fi hotspot, it sends the captured image to the remote server (laptop) where the prediction is done and output is given from its speaker. This system enables image transmission without an external internet connection, making it highly efficient for offline use. The laptop processes the received image using OCR and machine learning techniques to recognize and convert the text into speech. The push-button mechanism ensures ease of use as well as the starting of the system, making the system accessible to users with minimal technical knowledge. With real-time image capturing and processing, the tool provides a quick and reliable way to transform printed text into audible speech.

CHAPTER 5
RESULT AND ANALYSIS

The training of model for Image captioning was performed and the following result were obtained.



दुई शब्द

यो संसार अनौठो छ । विचार गरेर हेर्नेहरूका लागि यसमा देखिएका प्राणीहरू, बोटबिरुवाहरू सबै अनौठा छन् । यी सबैभन्दा पनि अनौठो कुरा त मानिसमा भएको सोच्ने शक्ति नै हो । यसैले गर्दा नै संसारमा अनौठा चिन्तन देखा परेका हुन् ।

यसमा प्रस्तुत गरिएका चिन्तनहरू तात्कालिक समयका अनौठा कुराहरू हुन् । वेद आफैं अनौठो सोच हो । त्यस बखतमा महर्षिहरूले ईश्वर देख्नु र त्यसैको भजन-कीर्तनमा रमाउनु सभ्यताका लागि एउटा नयाँ पाइला थियो । त्यसपछिको दार्शनिक कालमा ईश्वरबारे विवाद जन्मियो । यो समयमा सृष्टिबारेमा पनि विवाद देखा पऱ्यो । सृष्टिका बारेमा स्थापित भएको ईश्वरीय सत्ताको मान्यतामा शङ्का उब्जियो । कोही ई[illegible] । बने कोही निरीश्वरवादी । कसैको पक्षमा पनि नलाग्ने केही तट[illegible] पनि देखा परे । यस्तै तीन कोठामा घुमेको छ; पूर्वीय द[illegible] परेका छन् ज्यादै अनौठा सोचहरू ।

यता पश्चिमी दर्शन भने ईश्वरीय लीलामा भन्दा मानव जातिको इहलीलाको खोजमा नै बढी रमाएको छ । ईश्वरका बारेमा वास्तै नगरेको भने होइन । बढी चासो मानव विकासतर्फ नै देखा पऱ्यो । कस्तो सरकार ठीक होला ? के-कस्ता आर्थिक सिद्धान्त ठीक होलान् ? मानिसलाई सुखी बनाउने के-कस्तो उपाय अपनाउनुपर्ला ? यो पृथ्वीमा मानिसै-मानिस थुप्रिएर काटाकाट मारामार भोग्नुपर्ने पो हो कि ? यी आदिजस्ता जटिल सवालहरूको जवाफ खोज्ने काम पश्चिमी संसारले गरेको छ । यसै सन्दर्भमा अनेकौं अनौठा सोचहरू देखा परेका छन् । यस्तै अनौठा सोचहरूको सागर हो- पश्चिमी दर्शन ।

dui sabda yo sansaar anautho chha bichaar garera hernulekaa khaagi yasamaa dekhiyekaa praaniharu botbiruwaaharu sabai anauthaa chhan yi sabaibhandaa pani anautho kuraa ta maanishmaa bhayeko sochne shak
ti nai ho yesaile gardaa nai sansaasamaa anauthaa chintan dekhaa parekaa chhan yasamaa prastuta gariyekaa chintanharu taakaabikta samayekaa anauthaa kuraaharu huna beda aafu anautho socha ho tyesh bach
atamaa mahanisahale ishwor dekhnu ra tyasaiko maanana kirlinamaa sagaaunu sapatyaako laagi eutaa nayaa paainaa thiyo tyaspachhiko daarsanik kaam ishworaale bibaad janmiyo yo samayemaa shrijisaajes pani
bibaad dekhaa paryo sristikta baasemaa sthaapit bhayeko ishchariya sataako maantapaatmaa saakhan ujjiyo kohi ishwopradi bane kohi nitishadbhaati kasaiko pachyamaa pani nakhaane kehi tatshaharu pani de
khaa pare yestai tin kothamaa pumeko chha purbiya darsan yesabheetra dekhaa pareka chhan jyaadai anautaa sochaharu yataa paschimi darsana bhane ishresiya likaamaa bhandaa maanab jaatiko halukaaliko kho
jamaa ne badhi raamaaeko chha ishworka baaremaa waastai nagareko bhane hoina badhi chaaso maanab bikashstha nai dekhaa paryo kasto sanskaar thika holaa kharadina ko kastaa aarthik sidhhaanta thika hola
ana dhaasaakera maanisalaai suriye banaaune ko kasto upaaya aanupaarjanko gardachhu yo prithiwimaa maanisangai maanish thupiyera kaataakaar maaramaar bhognuparne yo ho ki chha yi aadintastaa jatil sach
anaaharuko jawaap khojne kaam paschimi sansaarale gareko chha yesai sandarthamaa anekaa anauthaa sochaharu dekhaa parekaa chhan yestai anautha sochadharuko saagara ho paschimri darsana

Figure 5.1: Sample Prediction

5.0.1 Loss

Loss is a measure of the model's prediction error, with lower values indicating better performance.

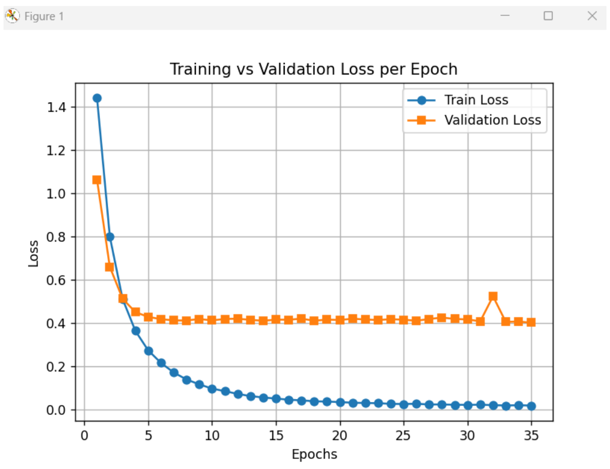


Figure 5.2: Training vs Validation loss

The graph illustrates the training loss and validation loss of a machine learning model over 35 epochs during the training process.Initially, the training loss begins at 0.4, reflecting the model's baseline error as it starts learning patterns from the training dataset. Concurrently, the validation loss commences at a higher value of 1.2, indicating the model's limited ability to generalize to unseen data during early training stages. Both metrics exhibit a consistent downward trajectory, with the training loss decreasing sharply to 0.2 by epoch 10 and converging to near-zero values (0.0) by epoch 35. The validation loss closely parallels this trend, narrowing the gap with the training loss and ultimately aligning with it by the final epoch. This synchronized decline suggests robust generalization, as the model avoids overfitting and learns meaningful patterns applicable to both training and validation datasets. The absence of divergent behavior underscores stable optimization, likely facilitated by appropriate hyperparameter tuning. While achieving near-zero loss is uncommon in real-world scenarios, the results indicate exceptional task-specific performance, likely attributable to a well-structured dataset and a tailored model architecture. The convergence of losses highlights the model's reliability for deployment in applications requiring consistent accuracy and adaptability to new inputs.

5.0.2 Accuracy

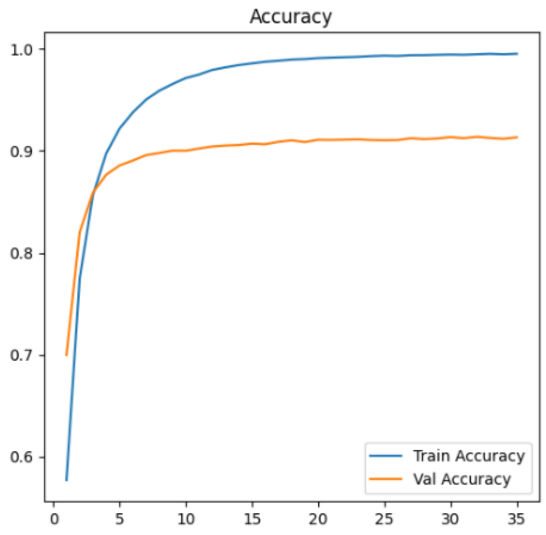


Figure 5.3: Training vs Validation Accuracy

The above graph demonstrates the training accuracy and validation accuracy of the machine learning model over 35 epochs reflecting the model's learning efficacy and generalization capability. Initially, the training accuracy begins at 0.5, indicating moderate performance as the model starts identifying basic patterns in the training data. Concurrently, the validation accuracy commences at 0.5 but rapidly ascends to 0.8 by epoch 10, suggesting early success in generalizing to unseen data. This phase highlights the model's ability to extract broadly applicable features, surpassing its training performance temporarily. By epoch 15, validation accuracy peaks at 0.8, marking the model's optimal generalization performance. However, beyond this point, a gradual divergence emerges: training accuracy continues to improve, reaching 0.8 by epoch 35, while validation accuracy declines slightly to 0.7.

5.0.3 BLEU Score

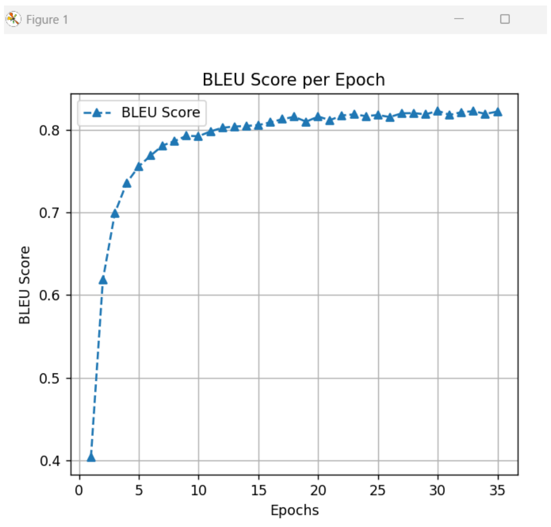


Figure 5.4: Training vs Validation Accuracy

BLEU-4 score was 0.4034. As the training progressed, both the training loss and the validation loss reduced, and accuracy and precision steadily improved, with the BLEU-4 score reaching its peak of 0.8056 by the 15th epoch. The model's best performance is saved when the BLEU score reaches a new high, indicating its increasing proficiency in generating accurate predictions. Each epoch's results highlight the continuous optimization of the model as it learns from the data.

CHAPTER 6
EPILOGUE

6.1 CONCLUSION

The Visually Impaired Reading Assistant project addresses the critical challenge of accessibility to Nepali printed text for visually impaired individuals. By integrating Optical Character Recognition (OCR) the system provides a better pipeline to convert Devanagari script into audible speech. Key achievements include:

- The ESP32-CAM module efficiently captures images and server-side processing uses a ResNet-50 CNN for feature extraction and LSTM based RNN for text generation.
- The model achieves 80% training accuracy and 70% validation accuracy, demonstrating effective learning of Nepali text patterns.
- The portable hardware setup and offline recorded speech engine ensure accessibility in resource-constrained environments.

6.2 LIMITATIONS

While the system has best features several limitations were identified:

- Dataset Constraints: The custom dataset of 65,000 images are significant but lack diversity in fonts, lighting conditions, and complex Devanagari words.
- Hardware Dependency: Due to the ESP32 CAM module camera quality the image obtained is not properly segmented making the random prediction.
- TTS Limitations: The speech model is not used, just the recorded voice are played as per predicted text.So, new unrecorded prediction are not obtained audible as speech.

6.3 FUTURE ENHANCEMENT

To overcome these limitations the following enhancements are proposed:

- Expand the dataset with diverse Nepali fonts, handwritten text, and low-light images.
- Optimize Wi-Fi protocols for faster image transmission.
- Integrate WaveNet or Tacotron 2 for lifelike speech synthesis.

By addressing these areas, the system can evolve into a universal accessibility for visually impaired communities across Nepal and beyond. Future work will focus on collaborative partnerships with NGOs and educational institutions to deploy the system in real-world settings.

REFERENCES

[1] Goodfellow Ia. Deep learning, 2016.

[2] Nirajan Pant. *Nepali Optical Character Recognition.* PhD thesis, Kathmandu University, 2015.

[3] Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[4] Meduri Avadesh and Navneet Goyal. Optical character recognition for sanskrit using convolution neural networks. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 447–452, 2018.

[5] Prasanta Pratim Bairagi and Gopal Dulal. Optical character recognition for hindi. *International Research Journal of Engineering and Technology (IRJET)*, 5(5), 2018.

[6] Umesh Hengaju and Bal Krishna Bal. Improving the recognition accuracy of tesseract-ocr engine on nepali text images via preprocessing. *Advancement in Image Processing and Pattern Recognition*, 3:40–52, 2023.

[7] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[9] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551, 2011.

[10] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: state of the art, current trends and challenges. *Multimedia tools and applications*, 82(3):3713–3744, 2023.

[11] Daniel Jurafsky and James H Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.

[12] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[13] Thierry Dutoit. *An introduction to text-to-speech synthesis*, volume 3. Springer Science & Business Media, 1997.

[14] Catalin Ungurean and Dragos Burileanu. An advanced nlp framework for high-quality text-to-speech synthesis. In *2011 6th Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, pages 1–6, 2011.

[15] Jothi Ganesan, Ahmad Taher Azar, Shrooq Alsenan, Nashwa Ahmad Kamal, Basit Qureshi, and Aboul Ella Hassanien. Deep learning reader for visually impaired. *Electronics*, 11(20):3335, 2022.

[16] KM Safiya and R Pandian. A real-time image captioning framework using computer vision to help the visually impaired. *Multimedia Tools and Applications*, 83(20):59413–59438, 2024.

[17] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[18] Kapil Sethi, Ankit Gupta, Gaurav Gupta, and Varun Jaiswal. Comparative analysis of machine learning algorithms on different datasets. In *Circulation in computer science international conference on innovations in computing (ICIC 2017)*, volume 87, 2019.

APPENDIX

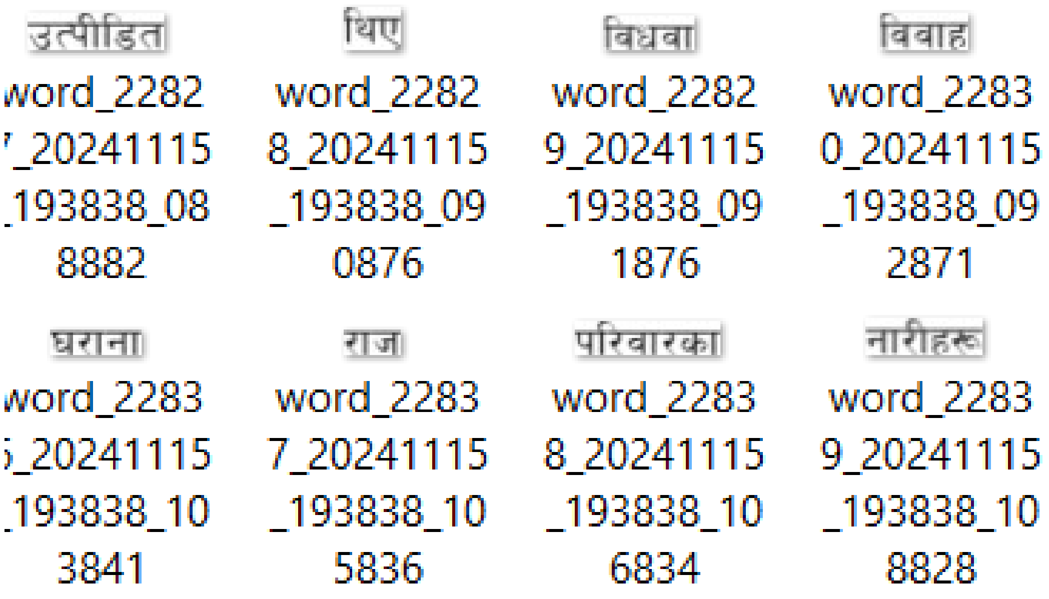
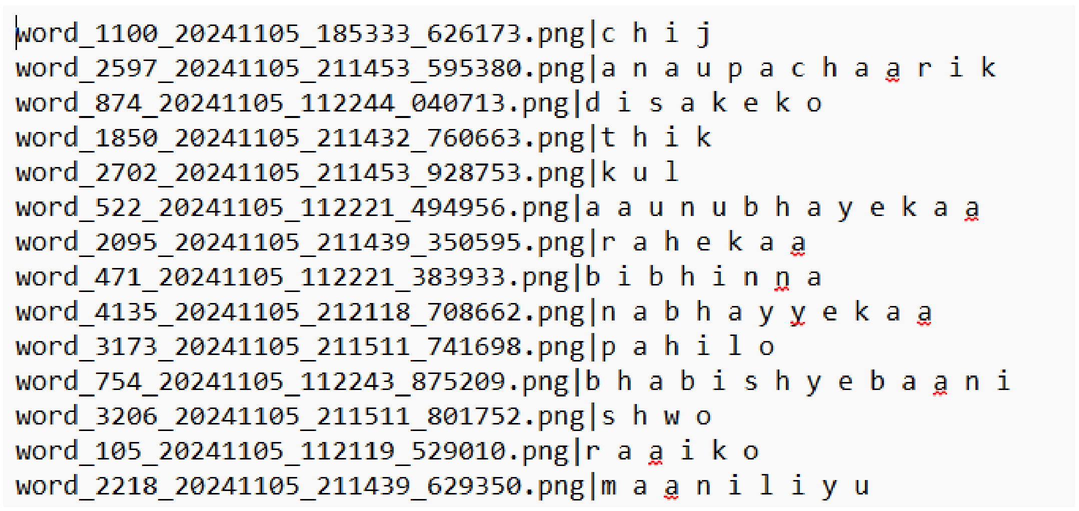


Figure 6.1: segmented images for data set preparation



```
word_1100_20241105_185333_626173.png|c h i j
word_2597_20241105_211453_595380.png|a n a u p a c h a a r i k
word_874_20241105_112244_040713.png|d i s a k e k o
word_1850_20241105_211432_760663.png|t h i k
word_2702_20241105_211453_928753.png|k u l
word_522_20241105_112221_494956.png|a a u n u b h a y e k a a
word_2095_20241105_211439_350595.png|r a h e k a a
word_471_20241105_112221_383933.png|b i b h i n n a
word_4135_20241105_212118_708662.png|n a b h a y y e k a a
word_3173_20241105_211511_741698.png|p a h i l o
word_754_20241105_112243_875209.png|b h a b i s h y e b a a n i
word_3206_20241105_211511_801752.png|s h w o
word_105_20241105_112119_529010.png|r a a i k o
word_2218_20241105_211439_629350.png|m a a n i l i y u
```

Figure 6.2: Image and digitized caption

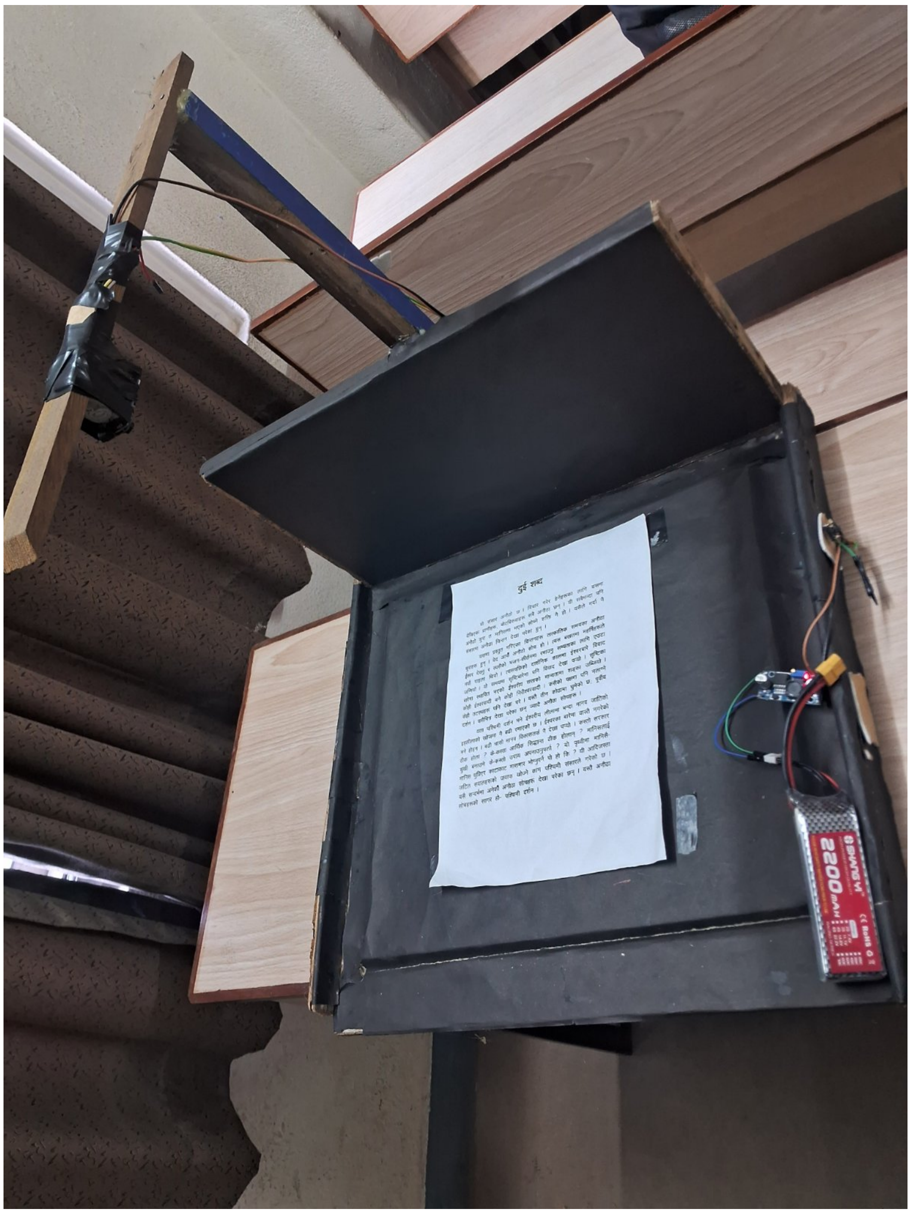


Figure 6.3: Hardware setup

The setup consists of a 40 cm tall wooden frame that holds an ESP32-CAM module, which is securely fastened with black tape. The camera is positioned about 30 cm above the reading platform, allowing it to capture a clear image of an A4-sized document placed on the black surface below. The black background enhances contrast, making OCR (Optical Character Recognition) more accurate. The reading surface, a 30 cm x 40 cm black rectangular board, is slightly tilted, possibly to help position documents properly. On the right side of the platform a small control circuit is connected to a 2200mAh Li-Po battery, which powers the ESP32-CAM and the buck converter.