BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution*

**(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to
Visvesvaraya Technological University, Belagavi)
"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur,
Ballar1-583 104 (Karnataka) (India)
Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197**

DEPARTMENT OF CSE-DATA SCIENCE

A Mini-Project Report On

"PolyLinguaNet: Character-Level RNN for Detecting Indian and English Languages"

A report submitted in partial fulfillment of the requirements for the

NEURAL NETWORK AND DEEP LEARNING

Submitted By

MOUNIKA M USN: 3BR22CD039

Under the Guidance of
Mr. Azhar Biag

Asst. Professor

**Dept of CSE (DATA SCIENCE),
BITM, Ballari**

Visvesvaraya Technological University

Belagavi, Karnataka 2025-2026

BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution*

(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)

"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur,
Ballari-583 104 (Karnataka) (India)
Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE (DATA SCIENCE)

CERTIFICATE

This is to certify that the Mini Project of NEURAL NETWORK AND DEEP LEARNING title **“PolyLinguaNet: Character-Level RNN for Detecting Indian and English Languages”** has been successfully presented by Mounika M 3BR22CD039 student of semester B.E for the partial fulfillment of the requirements for the award of Bachelor Degree in CSE(DS) of the BALLARI INSTITUTE OF TECHNOLOGY& MANAGEMENT, BALLARI during the academic year 2025-2026.

It is certified that all corrections and suggestions indicated for internal assessment have been incorporated in the report deposited in the library. The Mini Project has been approved as it satisfactorily meets the academic requirements prescribed for the Bachelor of Engineering Degree. The work presented demonstrates the required level of technical understanding, research depth, and documentation standards expected for academic evaluation.

Signature of Coordinators

Mr. Azhar Baig
Ms. Chaithra B M

Signature of HOD

Dr. Aradhana D

ABSTRACT

This project uses a character-level Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) units to automatically identify the language of a given text. It was applied to a custom multilingual dataset consisting of English, Hindi, Tamil, and Kannada sentences collected from Wikipedia and augmented with short conversational English phrases.

The model was trained on sequences of characters rather than words, enabling it to recognize distinct script patterns and handle short or mixed-text inputs effectively. After preprocessing through tokenization and padding, the RNN was trained to classify the text into its respective language.
The trained model achieved high accuracy in distinguishing between the four languages and performed well even for short or noisy user inputs. It was saved and deployed for real-time testing, allowing users to type text and instantly receive the detected language.

The project demonstrates the potential of deep learning for multilingual text classification, especially for Indian languages with unique scripts. Future work includes expanding the dataset to more regional languages, improving accuracy using bi-directional LSTMs or GRUs, and deploying the model as a web-based or mobile application for broader accessibility.

ACKNOWLEDGEMENT

The satisfactions that accompany the successful completion of our mini project on **“PolyLinguaNet: Character-Level RNN for Detecting Indian and English Languages”** would be incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement and support crowned my efforts with success. It is our privilege to express our gratitude and respect to all those who inspired us in the completion of our mini-project.

I am extremely grateful to my Guide **Mr. Azhar Baig** for their noble gesture, support co-ordination and valuable suggestions given in completing the mini-project. I also thank **Dr. Aradhana D,** H.O.D. Department of CSE(DS), for his co-ordination and valuable suggestions given in completing the mini-project. We also thank Principal, Management and non-teaching staff for their co-ordination and valuable suggestions given to us in completing the Mini project

Name	USN
MOUNIKA M	3BR22CD039

TABLE OF CONTENTS

Chapter No.	Chapter Name	Page No
	Abstract	I
	Acknowledgment Table of Contents List Of Figures	II III IV
1	Introduction	1
2	Objectives	2
3	Problem Statement	3
4	Methodology	4
5	Requirement Analysis	5
6	Design	6-8
7	Implementation	9
8	Results And Discussion	10
9	Conclusion	11
10	References	12

LIST OF FIGURES

Figure No	Figure Name	Page No.
4.1	Block Diagram	4
6.1	Flow Chart	6
6.2	Use case Diagram	7
6.3	Sequence Diagram	8

CHAPTER 1

INTRODUCTION

In today’s multilingual digital world, identifying the language of a given text is essential for various natural language processing applications such as translation, sentiment analysis, and information retrieval. This project, PolyLinguaNet, focuses on detecting the language of text written in English, Hindi, Tamil, and Kannada using a character-level Recurrent Neural Network (RNN) with **LSTM** layers. The model learns script patterns directly from characters, making it effective for short or mixed-language inputs. A multilingual dataset was created from Wikipedia and augmented with conversational English text to improve performance. The system accurately classifies text into its respective language and can be extended to support additional Indian languages in future work.

OBJECTIVES

1. Develop a Character-Level RNN Model:

To build a Recurrent Neural Network (RNN) using LSTM layers capable of identifying the language of a given text among English, Hindi, Tamil, and Kannada.

2. Create and Preprocess a Multilingual Dataset:

To collect and preprocess sentences from Wikipedia and conversational text using Unicode normalization, tokenization, and padding for effective model training.

3. Evaluate and Deploy the Model for Real-Time Detection:

To train, test, and deploy the model for accurate and real-time language detection, including handling short or single-character text inputs using Unicode-based rules.

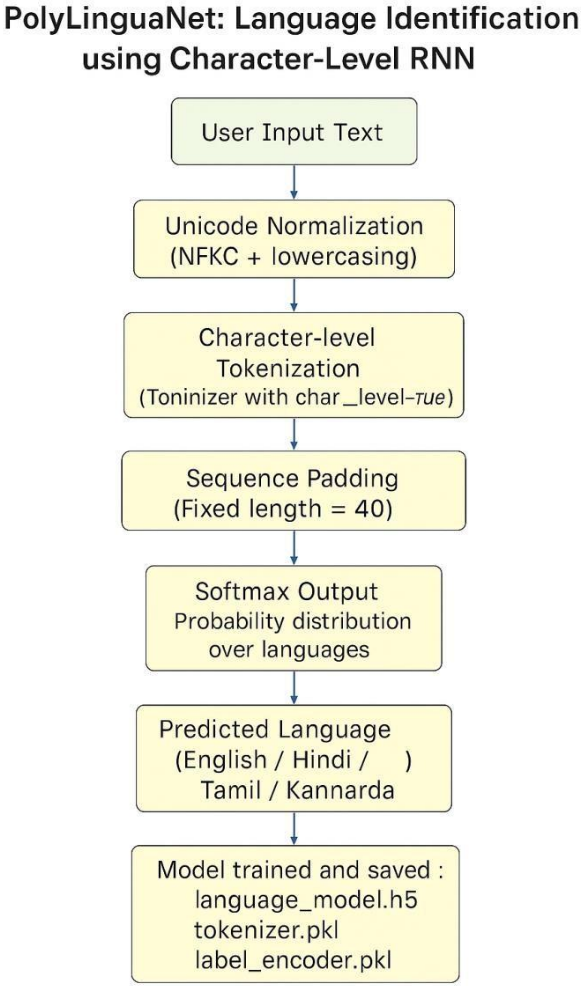
CHAPTER 3

PROBLEM STATEMENT

To develop an intelligent system capable of automatically identifying the language of a given text in a multilingual environment. To overcome the challenges faced by traditional language detection methods in handling short, noisy, and mixed-script inputs. To build a deep learning-based model that can accurately classify text written in English, Hindi, Tamil, and Kannada.

CHAPTER 4

METHODOLOGY


PolyLinguaNet: Language Identification using Character-Level RNN
User Input Text
Unicode Normalization (NFKC + lowercasing)
Character-level Tokenization (Toninizer with char_level-тue)
Sequence Padding (Fixed length = 40)
Softmax Output Probability distribution over languages
Predicted Language (English / Hindi /) Tamil / Kannarda
Model trained and saved : language_model.h5 tokenizer.pkl label_encoder.pkl

4.1 Block Diagram of PolyLinguaNet

The block diagram of the *PolyLinguaNet:* represents the overall flow of how the input text is processed and classified into its respective language. It begins with the **user input**, where the text (in English, Hindi, Tamil, or Kannada) is entered into the system. The text then passes through the preprocessing stage, which performs Unicode normalization, character-level tokenization, and sequence padding to convert the text into a suitable numerical form.

CHAPTER 5

REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENTS

- **Input Handling:** The system should accept user input text in multiple languages — English, Hindi, Tamil, and Kannada including single letters, words, or full sentences.
- **Preprocessing:** The system should preprocess the input using Unicode normalization, character-level tokenization, and sequence padding before feeding it to the model.
- **Language Detection:** The trained RNN (LSTM) model should classify the given input text and accurately identify its language in real-time.

NON-FUNCTIONAL REQUIREMENTS

- **Accuracy:** The model should achieve high accuracy (above 90%) in language prediction across all four languages.
- **Performance:** The system should provide real-time predictions with minimal latency (less than one second per input).
- **Scalability:** The system should be capable of extending to more Indian languages with minor changes in dataset and model.

CHAPTER 6

DESIGN

FLOW CHART

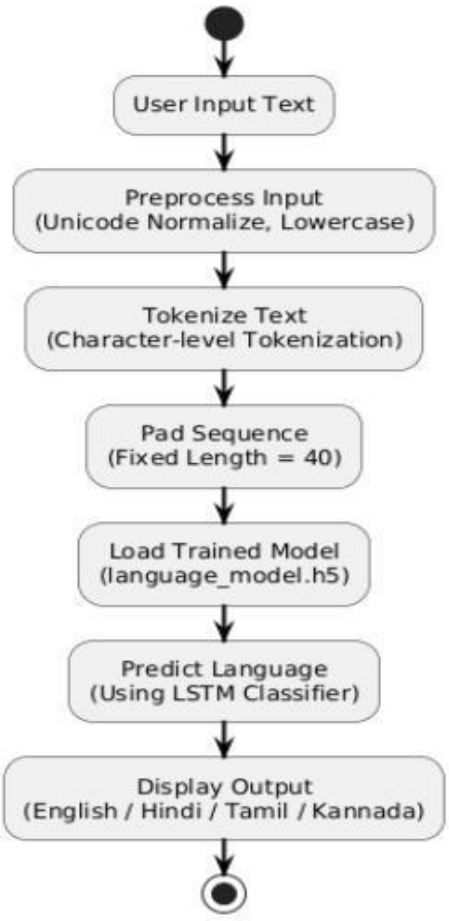


Fig 6.1 Flow chart

USE CASE DIAGRAM

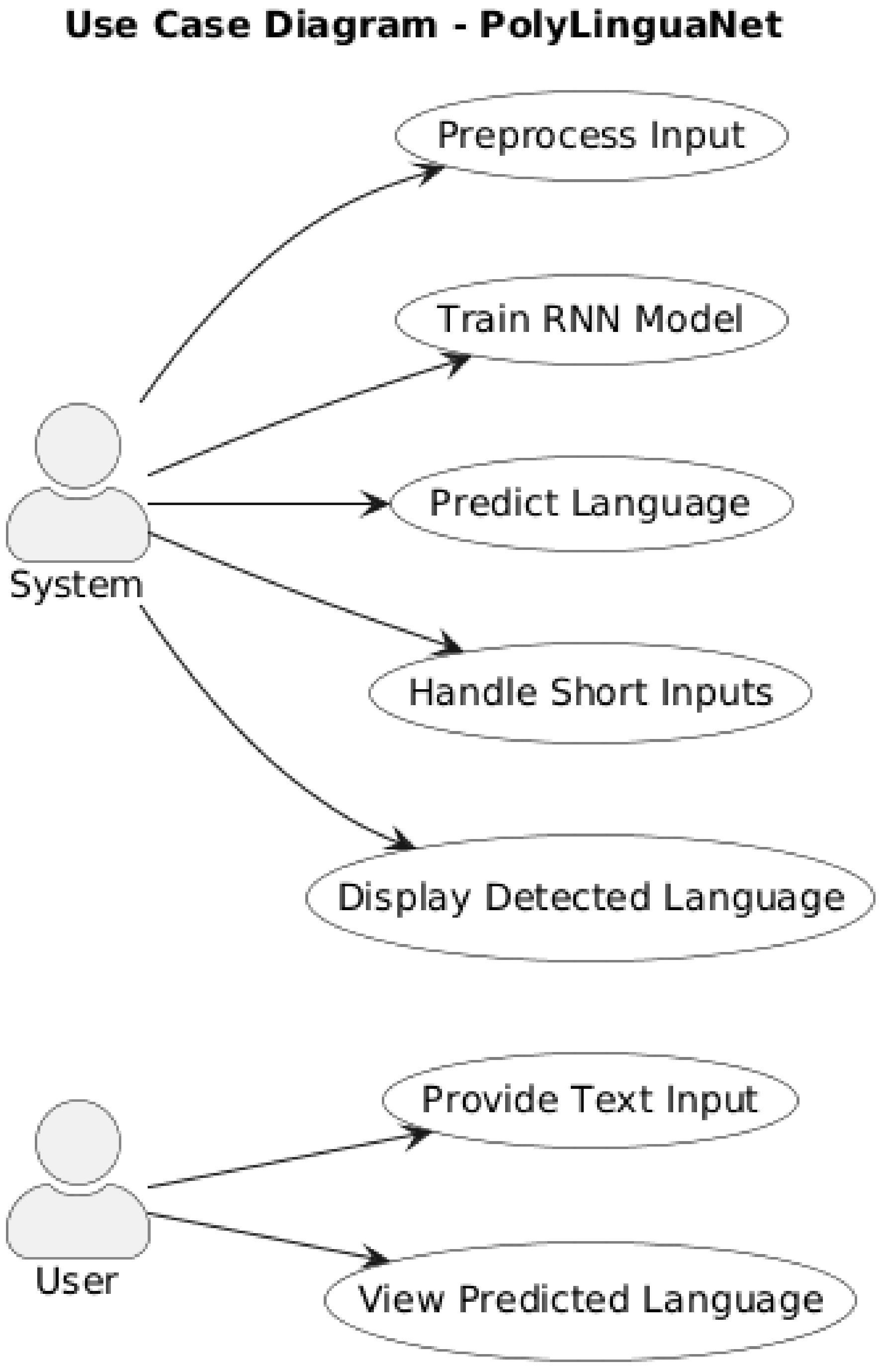


Fig 6.2 Use Case Diagram

SEQUENCE DIAGRAM

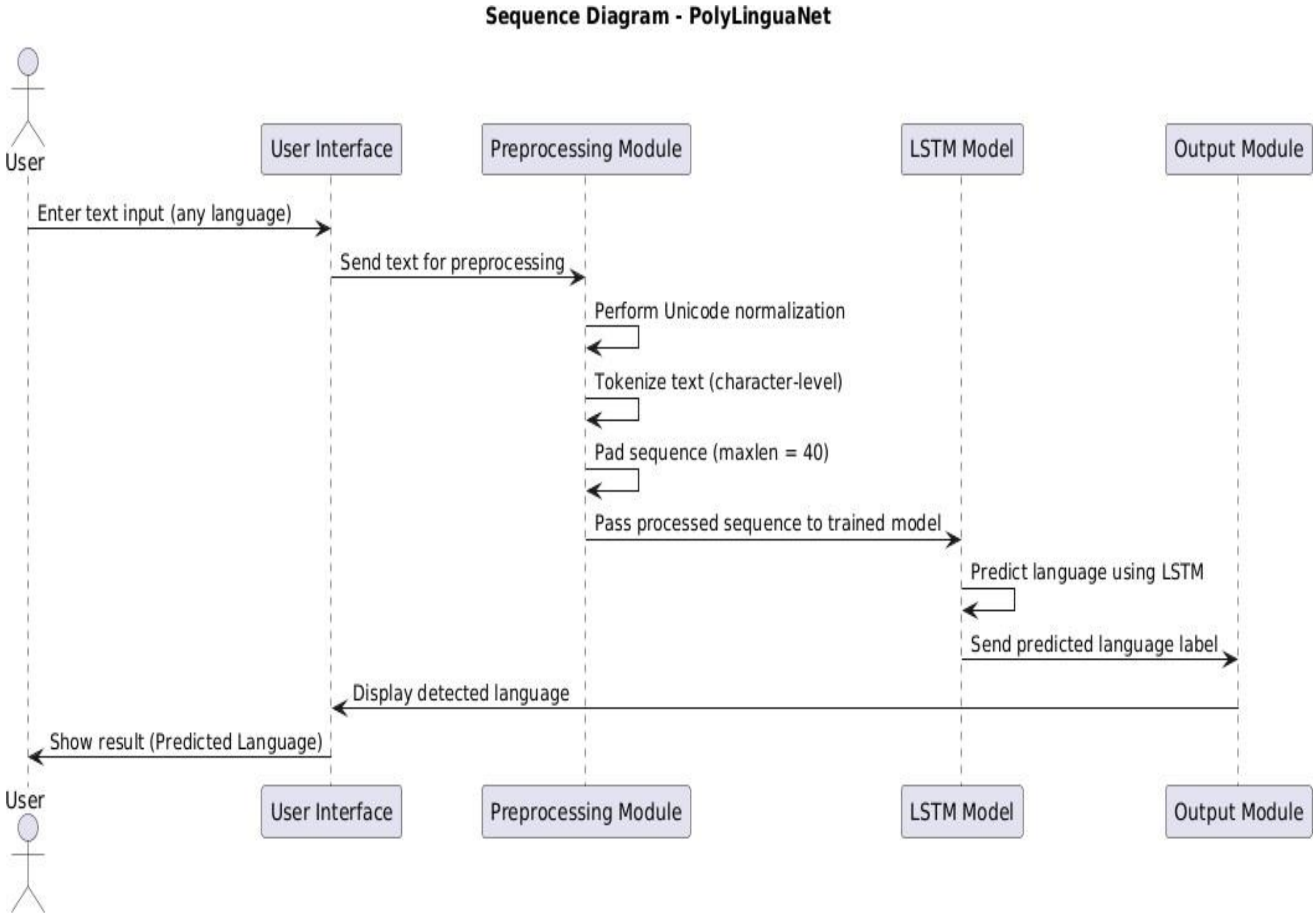

Sequence Diagram - PolyLinguaNet
User
User Interface
Preprocessing Module
LSTM Model
Output Module
Enter text input (any language)
Send text for preprocessing
Perform Unicode normalization
Tokenize text (character-level)
Pad sequence (maxlen = 40)
Pass processed sequence to trained model
Predict language using LSTM
Send predicted language label
Display detected language
Show result (Predicted Language)
User
User Interface
Preprocessing Module
LSTM Model
Output Module

Fig 6.3 Sequence Diagram

.

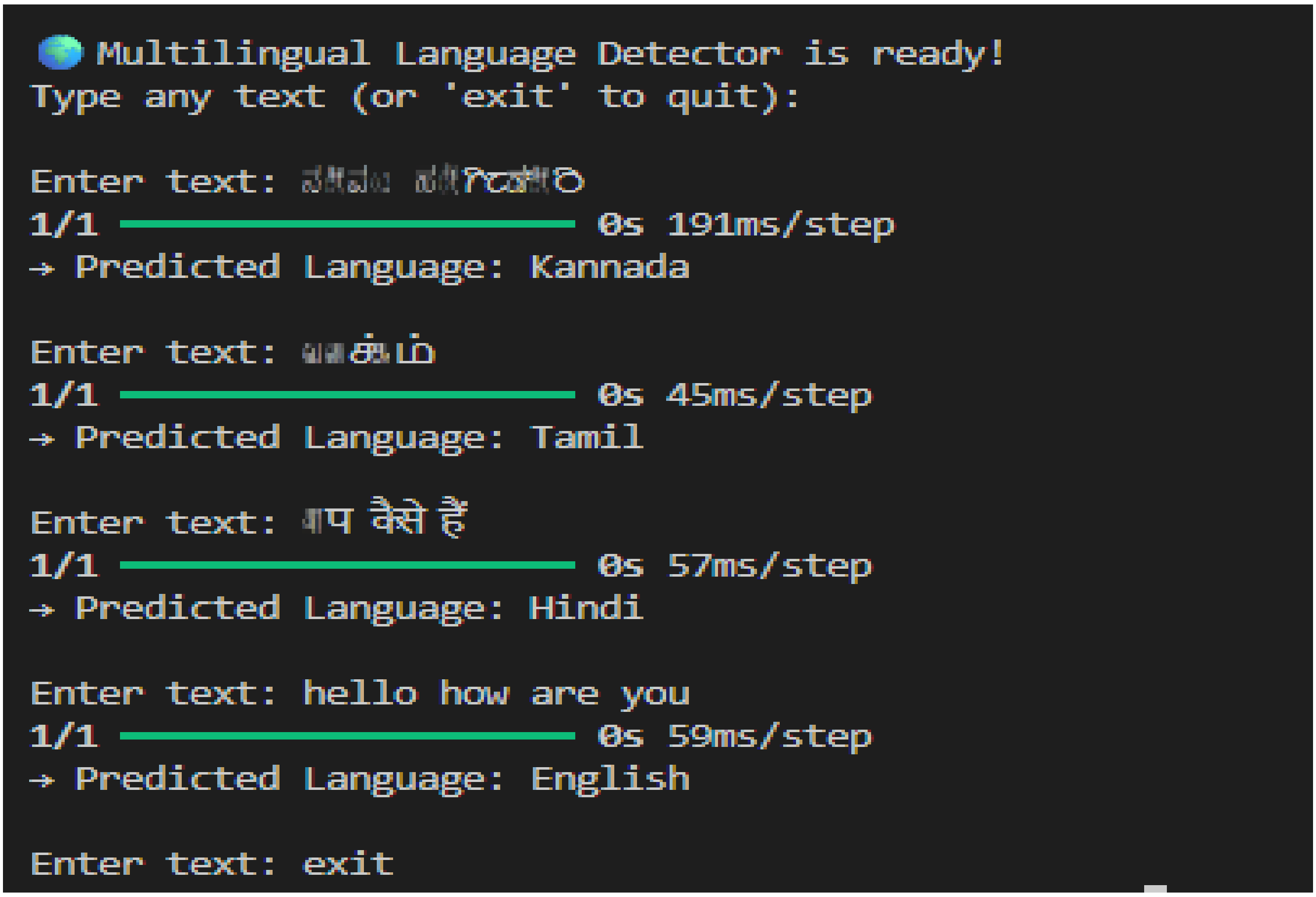
CHAPTER 7

IMPLEMENTATION

1. **Data Collection and Preprocessing:**
   - Dataset Preparation: A multilingual dataset was created using Wikipedia sentences in English, Hindi, Tamil, and Kannada. Around 100–200 sentences per language were collected.
   - Augmentation: Additional short English phrases and greetings (e.g., “hi,” “hello,” “ok”) were added to improve detection of short text inputs.
2. **Character-Level Tokenization and Sequence Formation:**
   - The text was tokenized at the character level, meaning each character was converted into a numeric token.
   - All sequences were padded or truncated to a fixed length (40 characters) to ensure uniform input size for the RNN model.
3. **Model Design and Training:**
   - **Model Architecture:**
     - Embedding Layer: Converts input characters into dense vector representations.
     - LSTM Layer: Captures sequential dependencies between characters to learn language-specific structures.
     - Dense + Dropout Layers: Used for classification and overfitting control.
     - Softmax Output: Produces probabilities for each target language.
   - **Training Process:**
     - The model was trained using categorical cross-entropy loss and the Adam optimizer.
     - It was trained for 10 epochs with a batch size of 16 and validated on 20% of the data.
4. **Model Evaluation and Saving:**
   - After training, the model achieved high accuracy in detecting all four languages.
5. **Real-Time Language Detection:**
   - During prediction, the user enters text through the console.
   - The system performs preprocessing, tokenization, and padding before sending input to the trained model.

CHAPTER 8

RESULTS AND DISCUSSION



```
🌍 Multilingual Language Detector is ready!
Type any text (or 'exit' to quit):

Enter text: ನೀವು ಹೇಗಿದ್ದೀರಿ
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 191ms/step
→ Predicted Language: Kannada

Enter text: வணக்கம்
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
→ Predicted Language: Tamil

Enter text: आप कैसे हैं
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step
→ Predicted Language: Hindi

Enter text: hello how are you
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
→ Predicted Language: English

Enter text: exit
```

CHAPTER 9

CONCLUSION

The project PolyLinguaNet successfully identifies text written in English, Hindi, Tamil, and Kannada using a character-level RNN model. It effectively handles multilingual and short-text inputs through sequential pattern learning and Unicode-based detection. This system demonstrates the potential of deep learning for accurate and real-time language identification across diverse Indian scripts.

CHAPTER 10

REFERENCES

[1] Y. Kim, “Character-Aware Neural Language Models,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, no. 1, 2016.

[2] F. Chollet, “Deep Learning with Python,” 2nd ed., Manning Publications, 2021.

[3] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] “TensorFlow: An End-to-End Open Source Machine Learning Platform,” Google AI Blog, 2019. [Online]. Available: https://www.tensorflow.