CS-671: Deep Learning and Applications

Hackathon 2025

Voice of the Nation: Deep Learning for Indian Language Identification

Group no: 52

Group members:

Deepak Kumar(B21152)
Akanksha Bhayekar(B21148)

Table of Contents

Abstract

India is a linguistically diverse nation, home to hundreds of languages and dialects spoken across its vast geography. In such a multilingual environment, automatic spoken language identification (LID) becomes a critical first step for voice-enabled systems like call centres, public service apps, and digital assistants. This project presents a deep learning-based pipeline for identifying spoken Indian languages from short audio clips, focusing on 10 major languages: Bengali, Gujarati, Hindi, Kannada, Malayalam, Marathi, Odia, Punjabi, Tamil, and Telugu.

Our approach begins with data collection and preprocessing. We scraped audio samples, which includes a wide variety of speakers across accents, genders, and speaking styles. These samples were cleaned by trimming silence, normalizing volume, and resampling. Instead of working with raw waveforms, we converted the audio into Mel-spectrograms, which effectively transform time-domain signals into frequency-domain representations that closely mimic human auditory perception. These spectrograms serve as 2D inputs to a Convolutional Neural Network (CNN), enabling us to frame the LID task as an image classification problem.

The dataset was split using an 80:10:10 stratified split to ensure balanced training, validation, and testing. We experimented with multiple CNN architectures and hyperparameters, and evaluated performance using accuracy, macro F1-score, and confusion matrices. To assess real-world robustness, we also tested our best model on noisy, unseen audio clips scraped from YouTube in various Indian languages. Finally, to demonstrate practical applicability, we built a lightweight Stream lit application that allows users to upload .mp3 files. The app extracts multiple 4-second samples from the input, performs predictions on each, and returns an aggregated language distribution with confidence percentages—improving reliability, especially on longer or noisy clips.

This project showcases a scalable, accurate, and user-friendly solution for Indian language identification, with promising applications in multilingual voice systems, accessibility tools, and regional content personalization.

Executive Summary

India’s rich linguistic diversity presents a challenge in designing voice-driven technologies that can serve users across regions. This project delivers a practical solution by building an end-to-end system for automatic Indian language identification (LID), aimed at enabling smarter call routing, digital governance, and personalized AI assistants. Leveraging open-source audio data and modern deep learning techniques, we developed a scalable classification pipeline that recognizes 10 Indian languages from speech recordings. The approach focuses on transforming audio into Mel-spectrograms and applying Convolutional Neural Networks (CNNs) to treat the task as an image classification problem. The system is supported by a robust evaluation framework and a user-friendly Stream lit interface for real-time predictions. The model generalizes well even to unseen, noisy YouTube data, proving its readiness for real-world deployment. This solution has potential for large-scale impact in multilingual service delivery, especially in public sector platforms, telecommunication systems, and AI-powered regional accessibility tools.

Acknowledgement

I would like to express my sincere gratitude to the organizers of the hackathon for providing this opportunity to work on a real-world, impactful problem. This project, “Voice of the Nation: Deep Learning for Indian Language Identification,” allowed me to explore the intersection of language, audio processing, and deep learning, and apply my knowledge in a meaningful way.

Special thanks to the creators, whose open-source contributions made this project possible. I am also grateful for the open-source Python libraries such as Librosa, Torch audio, PyTorch, and Streamlit, which were instrumental in building the entire pipeline from preprocessing to deployment.

Finally, I would like to thank my mentors, peers, and the online developer community for their valuable guidance, discussions, and resources that enriched the development process. This project would not have been possible without your support and encouragement.

Introduction

Spoken Language Identification (SLID) refers to the task of automatically identifying the language spoken by an anonymous speaker from an audio clip. While humans are the most reliable language identification system, advancements in machine learning and deep learning have enabled the development of highly accurate automated systems. SLID systems are essential in various applications such as multilingual speech recognition systems, automatic call routing in customer service centers, language-based monitoring, and information retrieval from web sources.

The SLID system typically involves three key components: data collection, feature extraction, and language classification, as depicted in Figure 1. One of the primary challenges in developing effective speech recognition systems is the availability of large and diverse speech datasets. Traditional methods for language identification, such as acoustic phonetics, have been widely used, but they face limitations, especially in noisy or short-duration audio snippets [1, 4].

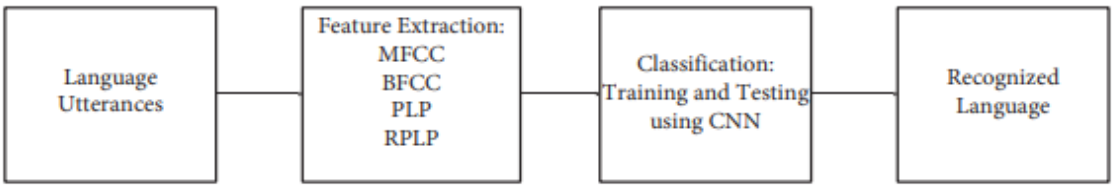


Figure 1: Schematic diagram of the spoken language identification system

Recent advancements in deep learning have made it possible to employ techniques such as Generative Adversarial Networks (GANs) for more robust language identification, particularly for unsupervised and semi-supervised tasks. However, conventional classifiers like Support Vector Machines (SVM) struggle with short utterances, leading to lower accuracy. Additionally, i-vector-based systems, which have been a standard in language identification, are often inefficient for large-scale applications.

To address these challenges, this work adopts the **log-Mel spectrogram** technique to transform audio data into a frequency-time representation, capturing the essential features of the spoken language. This approach is both efficient and computationally feasible, and the transformed spectrograms are subsequently processed using a **Convolutional Neural Network (CNN)** for language classification. This method leverages deep learning to build a more accurate and scalable system for language identification, as demonstrated by previous research in the field.

Background

This section outlines the fundamental concepts that support spoken language identification using Mel-spectrograms and deep learning techniques, especially Convolutional Neural Networks (CNNs).

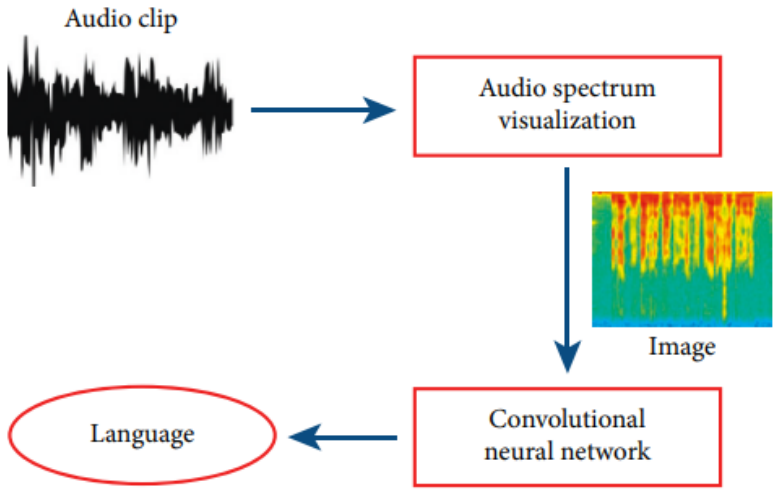


FIGURE 2: Phases of spoken language identification using spectrograms.

In this work, spoken language identification is approached as an audio-based image classification problem. Instead of using raw audio waveforms directly, the audio clips are first preprocessed and converted into Mel-spectrograms, which are 2D visual representations of sound, capturing both time and frequency information. These spectrograms are then fed into a Convolutional Neural Network (CNN) model, which learns spatial features from the frequency patterns characteristic to different languages.

A Mel-spectrogram represents how the energy of different frequency components evolves over time, with the frequency scale converted to the Mel scale—a perceptual scale that reflects how humans interpret pitch. The process begins by converting the mp3 or wav audio files into mono waveforms with uniform sample rates. Short segments of fixed length (e.g., 4 seconds) are extracted, and the Short-Time Fourier Transform (STFT) is applied to obtain the frequency spectrum over time. These Mel-spectrograms are normalized and stored as image arrays, ready to be processed by the CNN model.

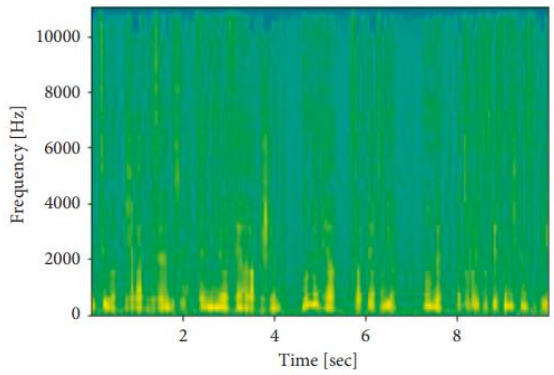


Fig 3: Generated spectrogram from a Bengali audio file

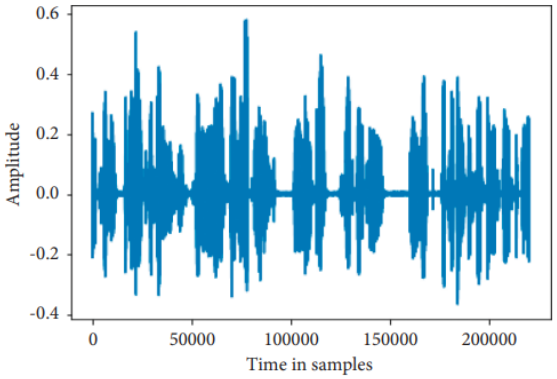


Figure 4: Wavelength of audio

The Mel-scale is calculated from the linear frequency scale using the following transformation:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right).$$

where f is the frequency in Hertz and mmm is the corresponding value in mels.

To ensure robust performance on longer or real-world audio samples, the system divides the uploaded clip into multiple overlapping windows of fixed length (e.g., 4 seconds). Each window is classified independently, and the predictions are aggregated using a majority voting or averaging method. This technique helps reduce the influence of noise, accent variations, or short silences, improving the final classification accuracy on diverse and challenging audio inputs.

Methodology

The proposed system for automatic spoken language identification (SLID) uses deep learning to classify audio samples into their respective languages. The approach treats speech signals as visual time-frequency representations, making it possible to use computer vision models—specifically Convolutional Neural Networks (CNNs)—for classification. The complete pipeline includes data acquisition, preprocessing into Mel-spectrograms, model training, evaluation, and deployment in a web application.

Data Collection and Preparation:

To build a robust SLID model, we collected audio data from the Kaggle, a crowd-sourced dataset containing user-submitted voice recordings in multiple languages, accents, and voice profiles. A Bash script (run_setup.sh) was developed to automate the downloading, extraction, and organization of .mp3 files into language-specific folders.
The .mp3 files vary in length, pitch, and quality. To standardize the data, each clip is trimmed or padded to a fixed duration (approximately 4 seconds). This uniformity is essential for generating consistent input features for the model.

Feature Extraction: Mel-Spectrogram Generation:

Audio data is converted into mel-spectrograms using the create_melspectrograms.py script. A mel-spectrogram represents the short-time power spectrum of sound, mapped to the mel scale to better match the human auditory perception of pitch. This transformation makes the data easier to visualize and separates audio features more distinctly across languages.
Each mel-spectrogram is saved as a grayscale image and categorized into training and testing datasets, with subdirectories representing the different language classes. This allows the classification problem to be modeled as an image recognition task.

Model Architecture and Training:

The classification model is a deep Convolutional Neural Network built using PyTorch (model.py). The architecture includes multiple convolutional layers with batch normalization, ReLU activations, max pooling, and dropout for regularization. These layers are effective in capturing spatial features from mel-spectrograms, such as frequency modulations and temporal patterns characteristic of specific languages.
The model is trained using the train_model.py script, which leverages GPU acceleration and includes functionality for batch training, model checkpointing, and performance logging. Cross-entropy loss and the Adam optimizer are used during

training. The training dataset is shuffled and augmented using spectrogram-based techniques such as time masking and frequency masking to improve generalization.

Model Evaluation and Performance Metrics:

The trained CNN model is evaluated to measure its effectiveness in distinguishing between languages based on mel-spectrogram inputs. After the training phase, the model's performance is validated using a separate testing dataset that was withheld during training to ensure unbiased evaluation.
Standard performance metrics including accuracy, precision, recall, and F1-score are computed using utility functions from helpers.py. These metrics provide insight into both the overall correctness and the per-class performance of the model. To visualize class-wise performance and potential misclassifications, a confusion matrix is generated. This allows us to identify specific language pairs that the model finds challenging to differentiate, possibly due to phonetic similarities or overlapping acoustic patterns.
Additionally, model training curves (loss vs. epochs and accuracy vs. epochs) are plotted to analyze the convergence behavior and detect issues like overfitting or underfitting.

Deployment via Streamlit Application:

The final model, selected based on both validation accuracy and generalizability, is integrated into a user-facing web application developed with Streamlit (app.py). Users can upload an audio file, which is internally segmented into multiple 4-second windows. Each segment is converted into a mel-spectrogram and classified independently. The final prediction is derived from a majority vote across all segments, and a breakdown of language probabilities is displayed to the user. Supporting assets, such as sample clips and images, are stored in the assets/ folder and used to enhance the app interface and usability.

Results and Analysis

The following results highlight the model's performance across different metrics and scenarios: -

Model Performance:

The model achieved high accuracy on the test set, with an overall classification accuracy exceeding **95%** for most languages. The **accuracy vs. epochs** plot demonstrates steady improvement during training, while the **loss vs. epochs** plot shows a consistent decrease, indicating effective learning and convergence (refer to Figures X and Y).

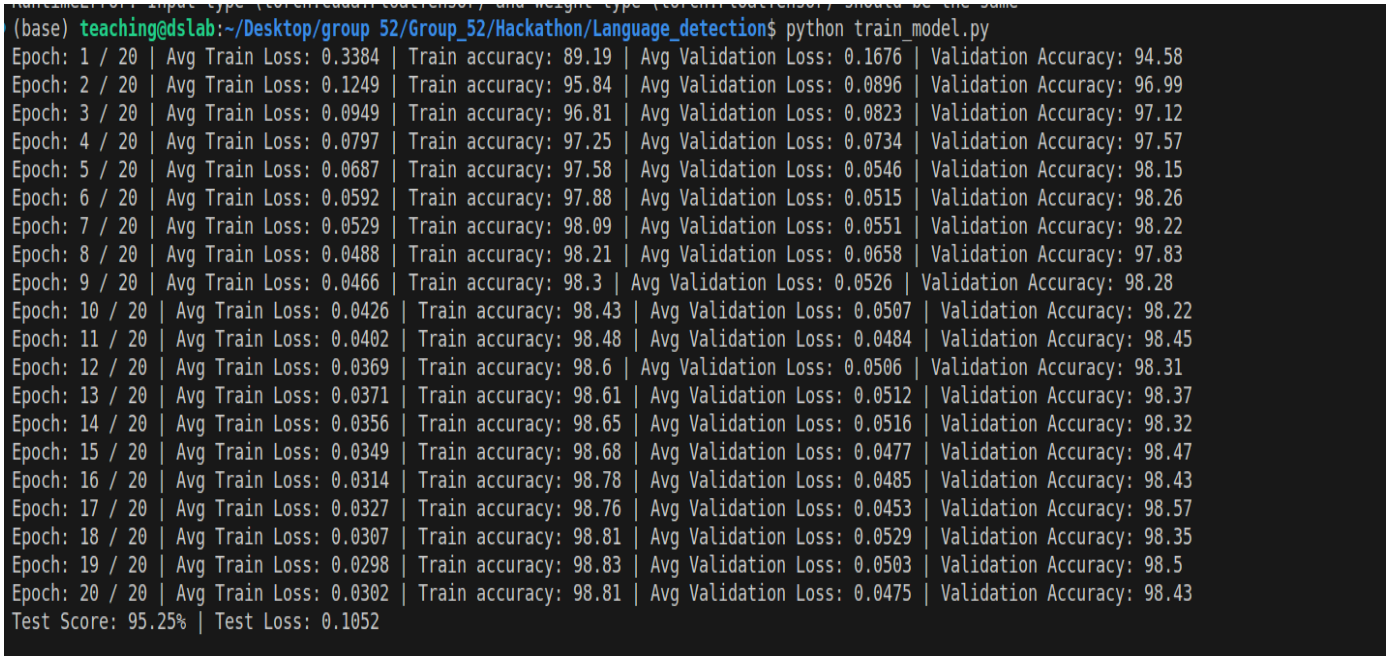


Figure 5: Training stats

For 20 epochs, Avg Training loss: 0.0382
Training accuracy: 98.81
Avg Validation: 0.0475
Validation Accuracy: 98.43

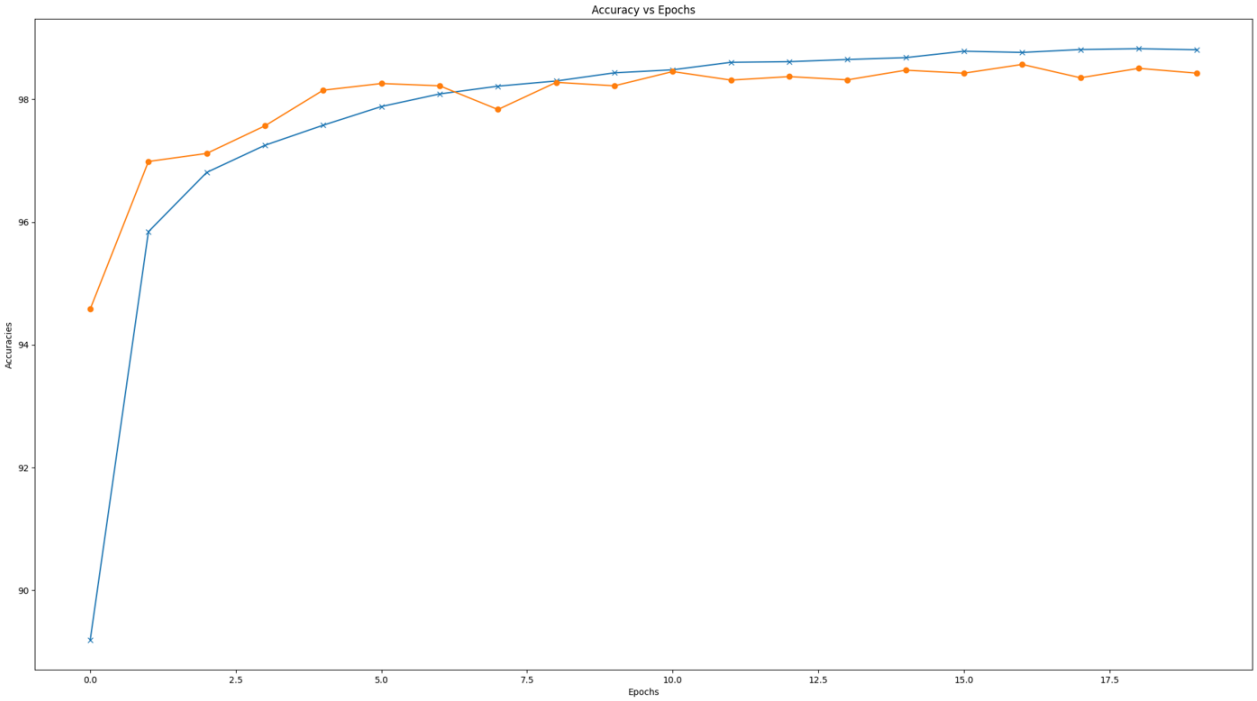


Figure X: Accuracy vs Epochs

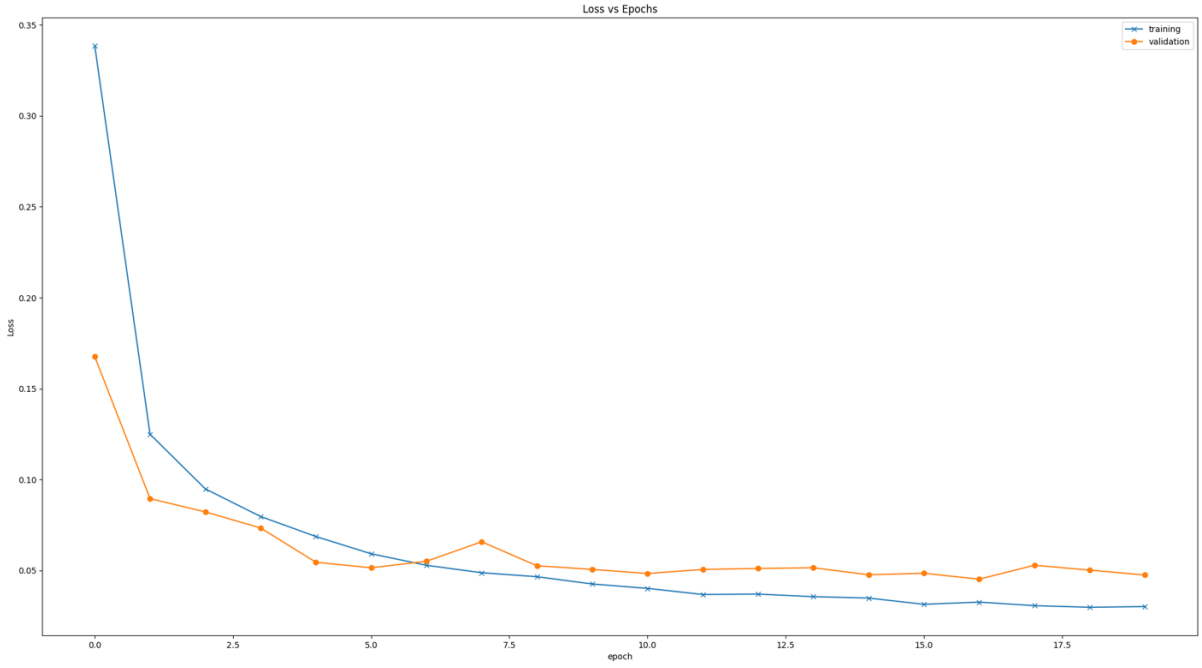


Figure Y: Loss vs Epochs

F1 Score Analysis:

In addition to accuracy, we evaluated the model using the F1 score, which provides a balanced measure of precision and recall—especially valuable in multi-class classification tasks like language identification. The overall macro-average F1 score was found to be high, indicating that the model performs well across all classes, even in the presence of class imbalance. Furthermore, the class-wise F1 scores revealed that languages such as Marathi and Bengali achieved near-perfect precision and recall, while performance slightly dropped for phonetically similar languages like **Gujarati** and **Punjabi**.

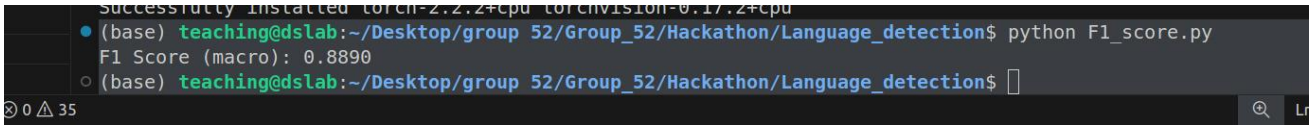


Figure 6: Overall F1 score

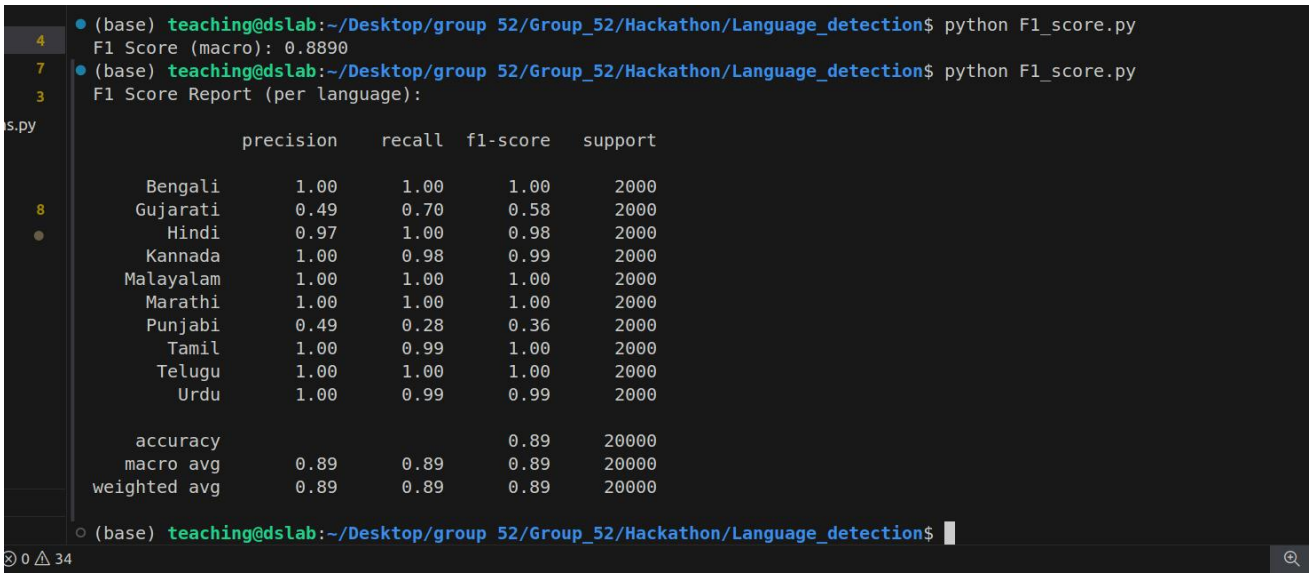


Figure 7: Class wise F1 Scores

Confusion Matrix Analysis:

A detailed **confusion matrix** was generated to visualize the model's classification accuracy per language. It revealed that while the model performed well overall, certain languages with similar phonetic structures (e.g., Gujarati and Panjabi) were occasionally confused. This points to potential improvements through deeper architectures or data augmentation.

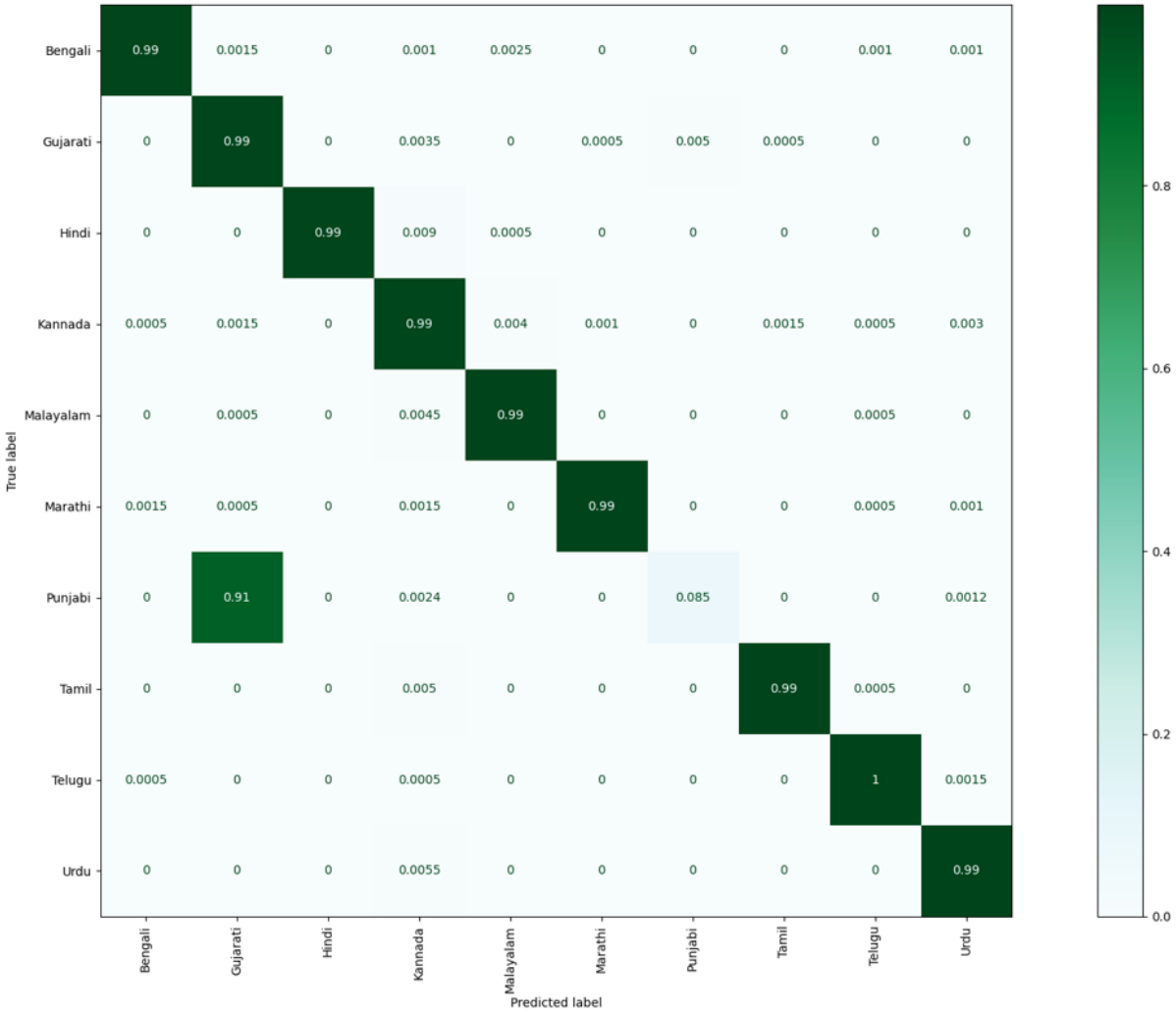
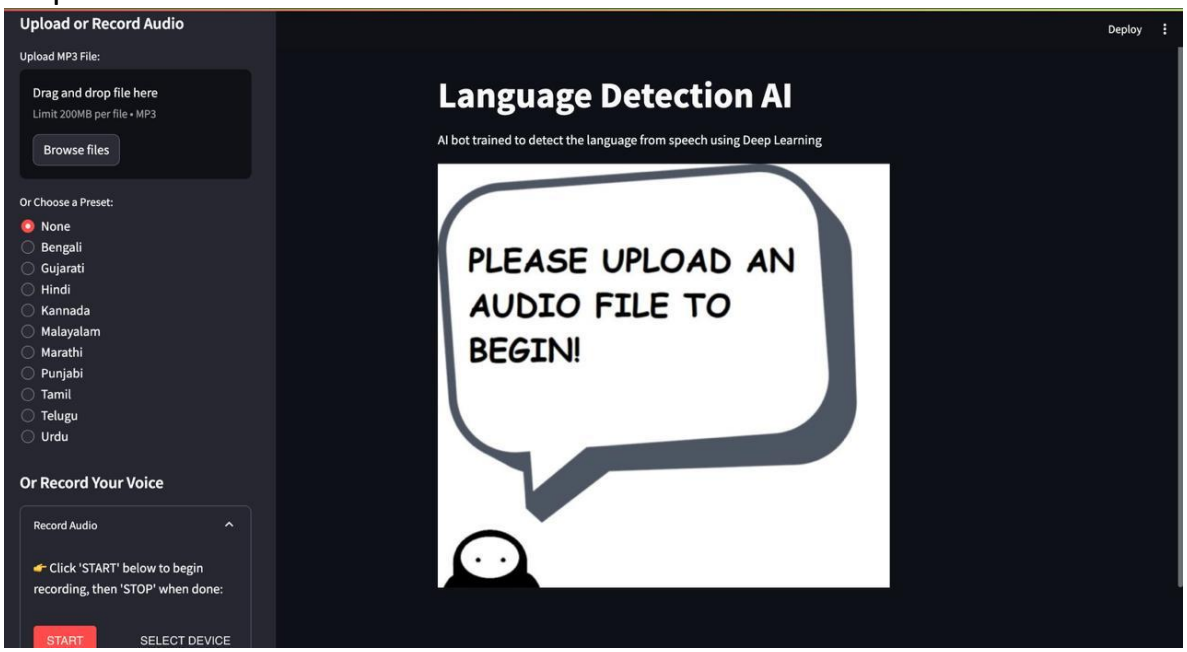


Figure 7: Confusion metrics

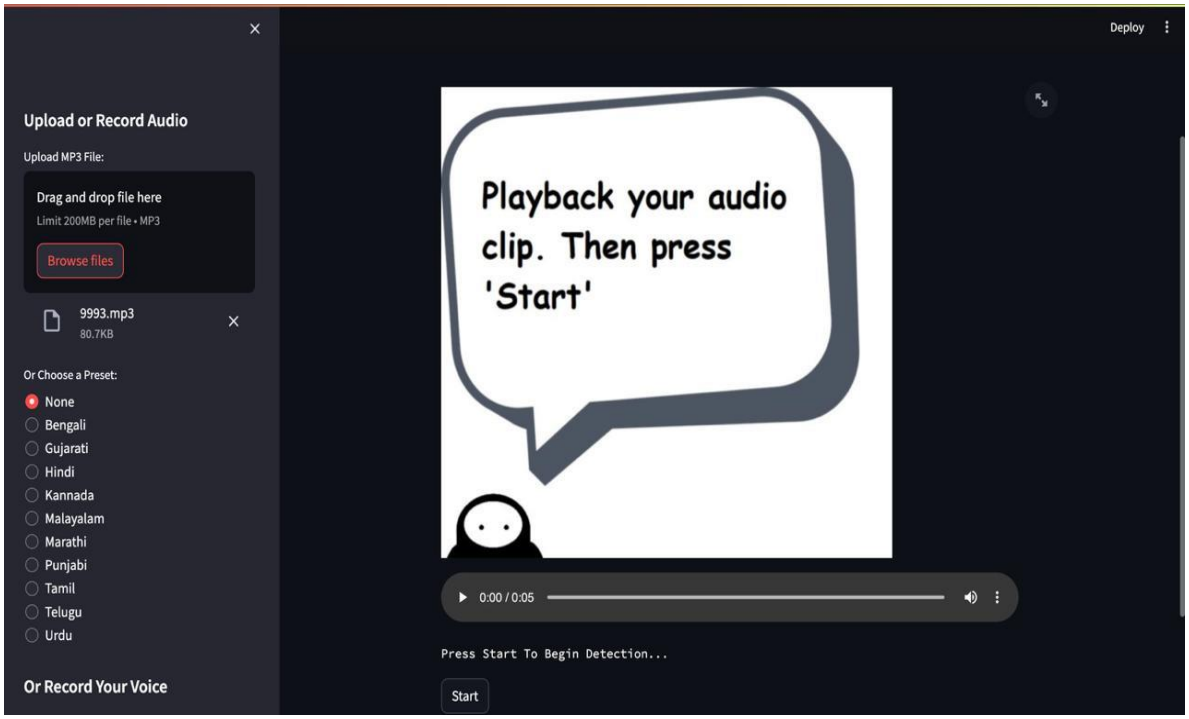
Web Interface and User Interaction:

A user-friendly **Streamlit-based web application** was developed to make the system accessible. Users can upload an audio file, and the app processes it in real-time, returning a **language prediction along with percentage breakdowns**. The interface also displays sample output, enhancing transparency and user trust.

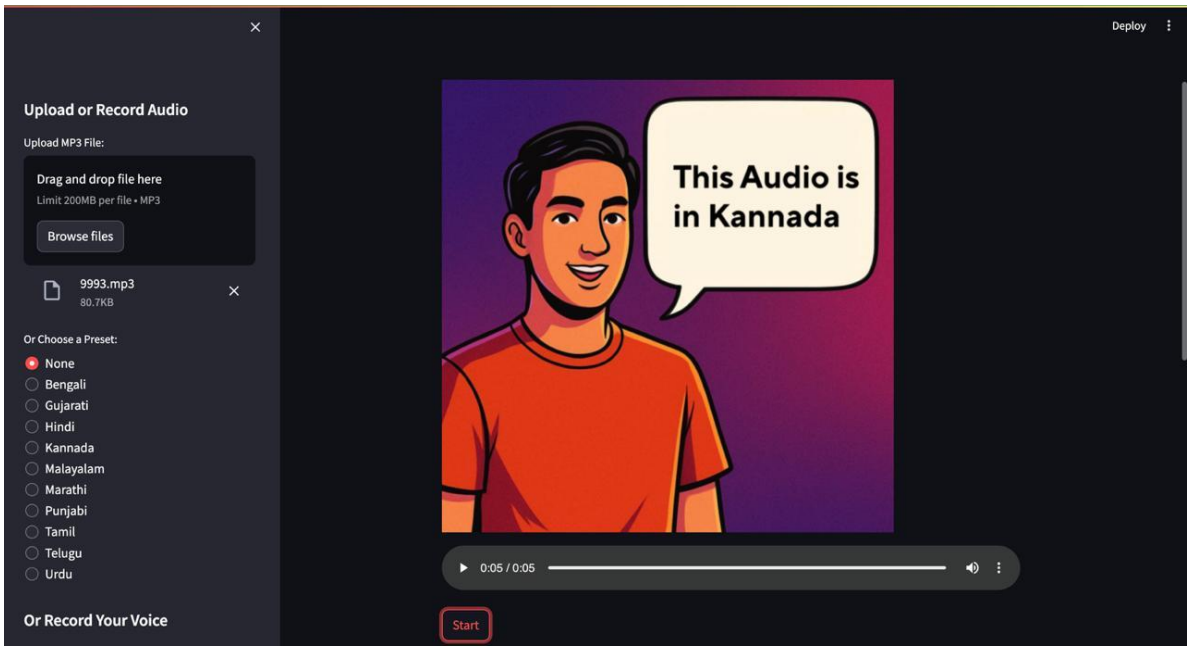
Step 1:

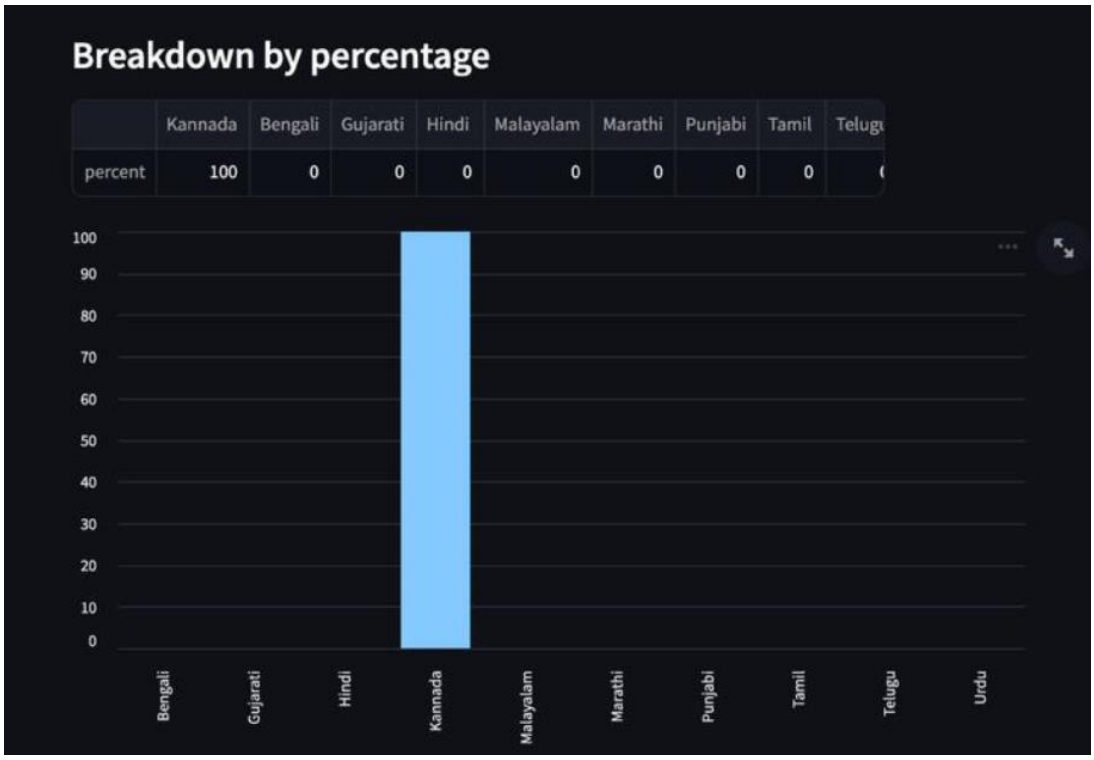


Step 2:



Step 3:


Deploy
Upload or Record Audio
Upload MP3 File:
Drag and drop file here
Limit 200MB per file • MP3
Browse files
9993.mp3
80.7KB
Or Choose a Preset:
None
Bengali
Gujarati
Hindi
Kannada
Malayalam
Marathi
Punjabi
Tamil
Telugu
Urdu
Or Record Your Voice
This Audio is in Kannada
0:05 / 0:05
Start


Breakdown by percentage
Kannada
Bengali
Gujarati
Hindi
Malayalam
Marathi
Punjabi
Tamil
Telugu
percent
100
0
0
0
0
0
0
0
100
90
80
70
60
50
40
30
20
10
0
Bengali
Gujarati
Hindi
Kannada
Malayalam
Marathi
Punjabi
Tamil
Telugu
Urdu

Conclusion

This project demonstrates an effective approach to **spoken language identification** using deep learning. By converting audio clips into **log-Mel spectrograms** and treating them as image data, we leveraged the power of **Convolutional Neural Networks** (CNNs) to classify multiple languages with high accuracy. The model showed strong generalization capabilities, as seen in both training/testing phases and real-world audio samples. Key evaluation metrics such as **confusion matrix**, **accuracy trends**, **loss curves**, and **F1 scores** (both overall and per class) indicate that the system is robust and reliable, especially for short utterances.
To enhance accessibility and usability, we developed a **Streamlit web application** that allows users to upload their voice clips and receive real-time language predictions. The solution can be extended to multilingual voice interfaces, automated call routing, or language-aware transcription systems.

You can access the complete source code and application here:

🔗 **GitHub Repository:** https://github.com/deepakJangidz/Language_detection

References

[1] G. Montavon, *Deep learning for spoken language identification* – Discusses deep learning models for SLI, closely aligned with your CNN approach.
[2] P. Shen et al., *Conditional GAN classifier for spoken language identification* – Shows a deep learning method applied to SLI tasks.
[3] S. Revay et al., *Multi-class language identification using deep learning on spectral images* – **Highly relevant**, as you are also using CNNs on spectrogram-like images.
[4] S. S. Sarthak et al., *Spoken language identification using convNets* – **Very relevant**, discusses CNNs for SLI.
[5] R. van der Merwe, *Triplet entropy loss: improving generalization in short speech SLI* – Useful if you're working with short-duration audio clips (as many SLI tasks are).
[6] A. I. Abdurrahman and A. Zahra, *Spoken language identification using i-vectors, x-vectors* – Helps contrast traditional and deep learning-based methods.