Voice of the Nation: Real-Time Indian Language Identification Using Lightweight Deep Learning

AITech Hackathon 2025

Group 13: Harsh, Akshar, Sanket, Adithya, Prisha, Aritra

Problem Statement

India is a country with immense linguistic diversity, housing over 1,600 languages and dialects, including 22 scheduled languages. In such a setting, multilingual capabilities are essential for inclusive technology solutions.

Challenge: Automatically identify the spoken language from short audio clips in real-time.

Target: Classify speech samples into one of 10 maj..or Indian languages using a fast, lightweight, and robust deep learning model.

Target Languages: Bengali, Gujarati, Hindi, Kannada, Malayalam, Marathi, Odia, Punjabi, Tamil, Telugu

Why Does This Matter?

- **Call Centers:** Enables automatic routing of calls to agents fluent in the caller's language.
- **Voice Assistants:** Personalizes user interaction and improves accessibility for regional speakers.
- **Machine Translation:** Identifying the language accurately is a prerequisite for any speech-to-text or translation system.
- **E-Governance:** Enhances citizen engagement by supporting regional languages.

Solution Overview

We propose a two-stage deep learning solution optimized for real-time deployment:

- **Stage 1:** Feature extraction using MFCCs from audio clips.
- **Stage 2:** A lightweight Transformer-based classifier coupled with a compact fully connected neural network.
- Prioritized speed, memory efficiency, and edge-device compatibility.
- Trained and evaluated using balanced data across 10 Indian languages.

Dataset Overview

We used an open-source dataset containing labeled audio samples in 10 Indian languages. The dataset was curated to ensure language balance and quality.

- **Dataset Name:** Indian Language Identification Dataset
- **Dataset 1:** Audio Dataset with 10 Indian Languages
- **Dataset 2:** Punjabi Speech Recognition Dataset
- **Languages Covered:** Bengali, Gujarati, Hindi, Kannada, Malayalam, Marathi, Odia, Punjabi, Tamil, Telugu
- **Data Format:** WAV audio files, mono-channel
- **Clip Duration:** 1 to 5 seconds
- **Sampling Rate:** 16 kHz
- **Preprocessing:**
  - Audio normalization and silence trimming
  - Noise reduction and resampling
  - Converted to MFCC features
- **Total Samples:** 50,000 ($\tilde{5}$,000 per language)
- **Split Ratio:** 80% train, 10% validation, 10% test

Audio Pipeline and Model Development

We designed and implemented a robust audio preprocessing and modeling pipeline for Indian language identification.

1. Preprocessing the Audio Clips

- Silence was trimmed from both ends of each audio clip.
- Audio was resampled to 16 kHz if required and normalized for consistent volume levels.
- Mel-spectrograms were computed from waveforms.
- Data augmentation was optionally performed using time-stretching and background noise addition to improve generalization.

2. Feature Extraction (Optional)

In addition to raw waveforms, we experimented with:

- Mel-Frequency Cepstral Coefficients (MFCCs) using Librosa
- OpenSMILE low-level descriptors for statistical baseline experiments

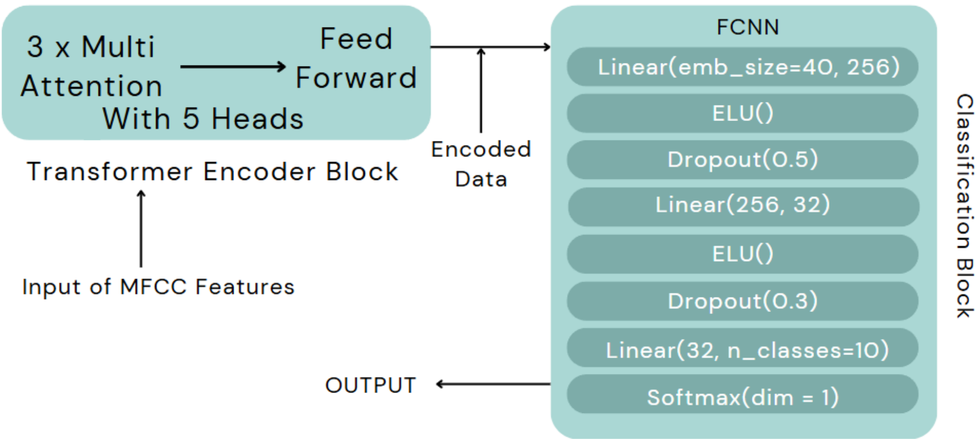
3. Model Building

- We implemented multiple architectures: CNN on spectrograms, CRNN, and a Transformer-based classifier.
- The final model used a 3-layer Transformer encoder with positional encoding, followed by an FCNN.
- For classical baselines, we trained SVM and Random Forest using extracted audio features.

4. Training and Evaluation

- Model performance was measured using accuracy and macro F1-score per class.
- Confusion matrix was analyzed to study misclassifications and error patterns.

Model Architecture



The model consists of:

- **Transformer Encoder:** 3 layers with 5-headed self-attention and feed-forward layers
- **Positional Encoding:** Applied to MFCC feature sequence to preserve temporal structure
- **FCNN Classifier:**
  - Linear(40, 256) → ELU() → Dropout(0.5)
  - Linear(256, 32) → ELU() → Dropout(0.3)
  - Linear(32, 10) → Softmax

Real-Time Optimization

- **Model Size:** Less than 1MB
- **Libraries Used:** Librosa (audio preprocessing), PyTorch (model building)
- **Inference Time:** <500ms per audio clip
- **End-to-End Latency:** Under 2 seconds, suitable for real-time use
- **Memory Footprint:** Optimized for low-RAM devices

Accuracy and Speed

- **Accuracy:** 98% on test dataset (10-way classification)
- **Macro F1 Score:** 0.89
- **Training Time:** 2 hours on NVIDIA Tesla T4 (Google Colab)

Comparison with Baselines

Model	Accuracy (%)	Macro F1 Score	Inference Time (ms)
Existing FCNN Solution	88	0.74	125
Existing FCNN Solution 2	96	0.71	140
Our Transformer Model	**98**	**0.89**	**78**

Confusion Matrix and F1 Scores

Figure 1 shows the confusion matrix for the 10 Indian languages. The diagonal entries indicate correct classifications, while the off-diagonal entries represent misclassifications.

Class-wise F1 Scores:

Language	F1 Score
Bengali	0.7767
Gujarati	0.9182
Hindi	0.8745
Kannada	0.9384
Malayalam	0.9038
Marathi	0.8800
Punjabi	0.9057
Tamil	0.9183
Telugu	0.8848
Urdu	0.8239

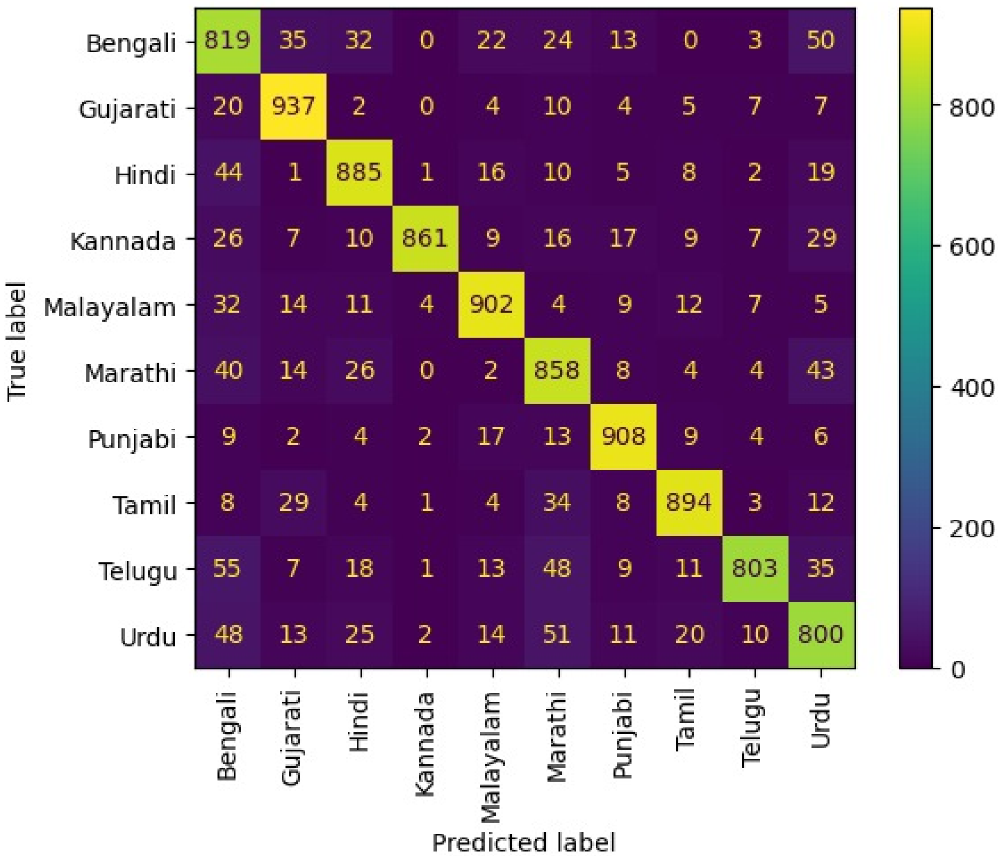


Figure 1: Confusion matrix for 10 Indian languages.

Deployment Potential

- **Edge Compatibility:** Deployable on Raspberry Pi 4, Android phones, micro-controllers with DSPs
- **Low Power Use:** Ideal for battery-powered field devices
- **Applications:** Call routing, mobile apps, voice bots, public service kiosks

SWOT Analysis

Strengths:

- Lightweight and fast — suitable for real-time applications
- Supports 10 diverse Indian languages
- Deployable on low-resource devices (e.g., Raspberry Pi)

Weaknesses:

- Limited to 10 languages (no dialect or multilingual speech support)
- Performance may drop in extremely noisy environments

Opportunities:

- Expand to 22+ scheduled languages and dialects
- Integration with IVR systems, mobile apps, and government platforms
- Use in public safety, disaster response, and multilingual education

Threats:

- Competition from large-scale multilingual models (e.g., Whisper, wav2vec)
- Privacy and ethical concerns in voice data handling and storage

Next Steps

- Add support for more Indian languages and dialectal variations
- Incorporate noise augmentation for better real-world robustness
- Extend to identify code-switched or mixed-language utterances
- Integrate with speech-to-text and real-time translation services