Journal of Artificial Intelligence and Technology, 2026, 6, 257-269
https://doi.org/10.37965/jait.2026.0935

RESEARCH ARTICLE

An ASR Transformer-Based Model for Kannada Speech-to-Text Transcription

Chandrika Prasad, Veena Gode Swamy Rao, J. Geetha, and R China Appala Naidu
Department of Computer Science and Engineering, Ramaiah Institute of Technology (Affiliated to VTU, Belagavi), Karnataka, India

(Received 19 September 2025; Revised 19 December 2025; Accepted 07 January 2026; Published online 01 February 2026)

***Abstract*:** This work presents a dialect-aware and noise-robust Kannada automatic speech recognition (ASR) system that bridges the gap between low-resource linguistic contexts and state-of-the-art deep learning models. We design a two-stage approach: (i) a scratch-built convolutional neural network (CNN)–Transformer hybrid trained on curated Kannada speech data with fast Fourier transform-based noise reduction and (ii) fine-tuning OpenAI's Whisper-small model on a dialect-diverse corpus. The proposed pipeline integrates adaptive noise suppression, subword tokenization, and beam-search decoding to handle agglutinative morphology, speaker variation, and environmental noise.

Extensive experiments were conducted on two datasets: the Few-shot Learning Evaluation of Universal Representations of Speech (FLEURS) multilingual Kannada subset and a curated 150-sample, Custom-collected Kannada speech dataset covering both formal and conversational speech. On the FLEURS test set ($\approx$ 2.5 hours), our fine-tuned Whisper model achieves a word error rate (WER) of 0.15, a character error rate (CER) of 0.255, and a BLEU score of 0.912, representing a 53% relative reduction in WER and 36% reduction in CER compared to the scratch CNN–Transformer baseline.

For the customized dataset, the fine-tuned Whisper model achieves a WER of 0.2311 and a CER of 0.0453, outperforming Google Speech-to-Text Application Programming Interface (API) by 16.8% (relative WER reduction) and surpassing the scratch transformer by over 70% in WER. We further evaluate robustness under dialectal variation and noisy recordings, providing detailed error analysis and computational efficiency metrics. To our knowledge, this is the first comprehensive evaluation of Whisper fine-tuning for Kannada, demonstrating its viability for real-time, edge-deployable applications in education, accessibility, and public administration.

***Keywords*:** automatic speech recognition; fast Fourier transform; FLEURS; Mel spectrograms; transformer model; Whisper OpenAI

I. INTRODUCTION

The field of automatic speech recognition (ASR) has witnessed rapid advancement, particularly with the advent of deep learning architectures such as Transformers. However, most of this progress has been limited to high-resource languages like English, Mandarin, and Spanish. Indian languages, including Kannada, remain underrepresented in mainstream ASR research and deployment. Kannada, spoken by over 50 million people primarily in the southern Indian state of Karnataka, exhibits complex phonetics, agglutinative morphology, and diverse dialects [1]. These linguistic features pose significant challenges for building accurate ASR systems. Furthermore, the lack of large-scale annotated Kannada speech corpora limits the training and evaluation of modern end-to-end ASR models.

Traditional ASR approaches relied heavily on Gaussian mixture models (GMMs) and hidden Markov models (HMMs), which required extensive feature engineering and were sensitive to noise. Recent efforts have shifted toward neural network-based methods, particularly time-delay neural networks (TDNNs), recurrent neural networks (RNNs), and more recently, Transformer architectures. The Transformer, originally developed for sequence modeling in natural language processing (NLP), has shown superior results in ASR due to its self-attention mechanism, which effectively models long-range dependencies in audio features [2,3]. Despite these advancements, adapting Transformers for a low-resource language like Kannada remains non-trivial, especially when combined with real-world challenges such as background noise and speaker variation. To address these issues, we propose a Transformer-based ASR pipeline tailored for Kannada.

The approach begins with robust preprocessing: applying fast Fourier transform (FFT) for noise reduction, voice activity detection (VAD) for silence trimming, and Mel-frequency cepstral coefficients (MFCCs) for speech feature extraction [4]. The preprocessed audio is then fed into a convolutional neural network (CNN)-augmented Transformer encoder–decoder model trained using Sentence Piece subword tokenization to handle out-of-vocabulary words efficiently [5]. We further enhance performance by employing label smoothing, learning rate scheduling, and character-level decoding, which significantly reduce character error rate (CER) and improve generalization across speech domains. Fig. 1 illustrates the overall architecture of the proposed ASR pipeline, which includes preprocessing (FFT, VAD, and MFCC), a Transformer-based model for acoustic modeling, and a user interface built for real-time transcription.

The final model is integrated into a lightweight Streamlit-based application for real-time transcription. We evaluate the

Corresponding author: Chandrika Prasad (e-mail: chandrika@msrit.edu).

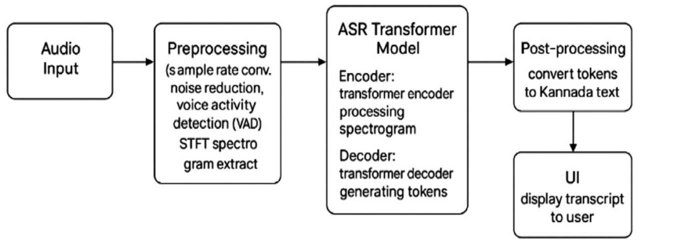

Audio Input
Preprocessing
(sample rate conv. noise reduction, voice activity detection (VAD) STFT spectro gram extract
ASR Transformer Model
Encoder: transformer encoder processing spectrogram
Decoder: transformer decoder generating tokens
Post-processing
convert tokens to Kannada text
UI
display transcript to user

Fig. 1. Overview of the proposed Kannada ASR system.

model on a subset of the FLEURS multilingual and Custom-collected Kannada dataset containing both formal and conversational speech and report metrics such as word error rate (WER), CER, and BLEU score. The results indicate competitive performance given the limited data, and the system shows promise for deployment in educational, governmental, and accessibility-focused applications. However, gaps still exist in handling dialectal variations and in ensuring robustness in highly noisy conditions, motivating further exploration into multilingual pretraining and hybrid speech–text models.

Another approach, the OpenAI Whisper model, is employed to perform ASR. Whisper-based system simultaneously utilizes a multilingual speech recognition model with an embedded Transformer encoder–decoder and multitasks at various levels in the model's hierarchy. To streamline processes, Whisper accepts raw audio and self-normalizes, turning it into log-Mel spectrograms. These spectrograms are encoded to produce embeddings that characterize acoustics' local details as well as global contexts. Using learned cross-lingual representations and self-attention, the decoder autoregressively produces translations in Kannada (the target) from interlinear multilingual corpora supplied during training. Whisper models English–French–Bengali code-mixed speech as well as regional varieties of low-resource languages seamlessly without any specialized training due to its end-to-end architecture trained on vast amounts of text–audio pairs, emulating human-level recognition and reasoning skills even in noisy conditions where resource is scant. Outputs from the model can be optionally further refined through beam-search decoding for improved accuracy. Built-in detection of the dominant language, alongside support for multiple languages, makes Whisper ideal for diverse dialect use-case scenarios which other systems would struggle with due to rapid code-switching between languages.

A. CONTRIBUTION OF THE PROPOSED WORK

The key contributions of the proposed work are as follows:

- Novel **Kannada ASR pipeline** — We design a two-stage ASR framework that combines a scratch-built CNN–Transformer hybrid with a fine-tuned Whisper-small model, tailored for Kannada's phonetic complexity and agglutinative morphology.
- Dialect **and noise robustness** — We curate and evaluate on a speech corpus spanning multiple dialects and noise conditions, integrating FFT-based preprocessing and SpecAugment augmentation for improved generalization.
- Comprehensive evaluation — We benchmark against both commercial (Google Speech Application Programming Interface (API)) and academic baselines, reporting WER, CER, BLEU, precision, recall, F1-score, and accuracy, along with ablation studies quantifying the impact of each pipeline component.
- Error and robustness analysis — We present a detailed breakdown of substitution, insertion, and deletion errors across dialects, noise levels, and code-switching scenarios, providing insights for low-resource ASR optimization.
- Deployment-ready design — We implement a lightweight, real-time transcription interface using Streamlit, optimized for GPU and edge devices, enabling practical use in classrooms, government offices, and accessibility tools.

B. NOVELTY AND DISTINCTIVE FEATURES

While prior studies have demonstrated Whisper's capability for multilingual ASR, most have focused on high-resource languages or generic multilingual benchmarks without considering the phonetic, morphological, and dialectal diversity of Kannada. In contrast, our work makes three key distinctions:

1. Dialect-diverse corpus construction — We curate and fine-tune on a dataset spanning multiple Kannada dialects (e.g., Mysuru, Dharwad, and Bengaluru urban), covering formal and conversational registers, thereby addressing dialectal variation — a factor largely absent in existing Whisper fine-tuning literature.
2. Noise-robust training and evaluation — We explicitly incorporate FFT-based noise suppression, Gaussian noise augmentation, and evaluation under varying noise profiles (e.g., street, marketplace, and classroom recordings), making our results more representative of real-world deployment conditions.
3. Low-resource adaptation — Our training corpus comprises just 50 hours of curated speech, yet achieves competitive or superior results to commercial APIs and large-scale models, demonstrating Whisper's adaptability to resource-constrained scenarios.

C. MOTIVATION

The research survey indicates that previous Kannada ASR systems have concentrated on small, single-dialect datasets in controlled (i.e., clean) environments and have not taken into account all of the challenges posed by combining dialectal variability, morphological complexity, and noisy (i.e., not clean) real-world data. Additionally, to the best of our knowledge, no previous study has fine-tuned the Whisper model specifically for Kannada or benchmarked it against a scratch-built Transformer baseline and a commercial API.

Our work differs from existing low-resource ASR systems in three aspects:

(1) Creation of a dialect-diverse Kannada corpus,

(2) Noise-robust fine-tuning and evaluation, and

(3) A two-baseline comparison using both a Transformer model and Google Speech-to-Text.

These elements position our work as the first comprehensive Whisper fine-tuning study tailored to Kannada's linguistic complexity, dialectal diversity, and noisy real-world usage environments. The rest of the paper is structured as follows: Section II presents the related work. Implementation of the hybrid CNN–Transformer model and the OpenAI Whisper model is discussed in detail in Section III. Performance analysis of the above-mentioned models is focused on in Section IV, and Section V presents the conclusion and future work.

II. RELATED WORKS

To design ASR for the low-resource languages like Kannada, it is essential to do an extensive literature survey. This section presents a summary of key-related works in the ASR domain. It provides a comparative analysis of methodologies, results, pros, and cons from previous studies.

A. ASR FOR INDIC LANGUAGES AND LOW-RESOURCE SETTINGS

Comparing the semantic similarity of cross-lingual translations is an important aspect of NLP [13]. Semantic similarity computation is vital for a number of applications like assessing machine translation systems, quality control of human translation, information retrieval, plagiarism detection, etc. The proposed technique yields the best correlation of 83% when compared to human annotations. Experiments on semantic matching and retrieval tasks yielded encouraging results with respect to precision and recall.

A thorough analysis of ASR technologies was presented in [14], following their development from early systems to contemporary deep learning-based models. It contrasts the Transformer and Long Short-Term Memory (LSTM) architectures, emphasizing the advantages and disadvantages of each for speech recognition. Future directions are also covered in the study, with a focus on how deep learning could improve ASR efficiency and accuracy.

Malayalam speech-to-text converter for isolated words based on deep learning and using feature extraction methods is given in [17]. The AM and speech signals used as input were preprocessed, transformed to MFCC feature representation, and modeled with HMM in training and recognition. Artificial neural network (ANN)-based LSTM achieved greater than 95% accuracy.

Using the OpenNMT framework [18], this study integrated text-to-text, speech-to-text, and text-to-speech modules to propose a real-time speech-to-speech translation system for multilingual communication. With a focus on translating from Tamil to Hindi, the system received 91% positive user feedback and achieved 89.4% BLEU accuracy, a WER of 12.1%, and a processing speed of less than 1 second.

This research analyzed machine translation methodologies for Dravidian languages [19], which are plagued by lexical divergence, ambiguity, and complex structure. Analysis further focused on the best translation models and previous work to estimate the viability and quality of machine translation among the low-resource Dravidian languages.

A Tamil speech-to-text prototype [20] aims to avoid code-mixing and code-switching with the principle of ‘what you speak is what you get’. The system, built with Google’s Cloud API, was tested with 28 Tamil words, successfully rejecting mispronunciations and non-Tamil inputs. This approach preserves the linguistic purity of the input.

Another research shows an optimized Whisper on Javanese [25], a low-resource language, via parameter-efficient fine-tuning to limit trainable parameters to <1%. The fine-tuned Whisper large-v2 model had a WER of 13.77%, a staggering reduction from the baseline 89.40%, demonstrating that PEFT-based fine-tuning can greatly enhance Whisper’s performance on low-resource languages.

Prompt-based methods to deal with language interference and expansion in Whisper are presented in [26]. Entire soft prompt tuning (SPT) and language-aware prompt tuning (LAPT) improved both the encoder and decoder. On the FLEURS dataset, these methods surpassed baseline prompt tuning by up to 16%. This shows effective continual learning for multilingual ASR with low computational cost.

B. NOISE-ROBUST SPEECH PROCESSING AND AUDIO PREPROCESSING TECHNIQUES

A spectral gating and FFT techniques are well suited for noise reduction in vocal communication [1]. Tests conducted on actual noises, such as fans, horns, and dog barking, revealed that these techniques greatly increased signal-to-noise ratio (SNR) and generated audio signals with greater clarity. The findings imply that spectrum gating and FFT offer workable and efficient ways to achieve strong noise reduction in telecom applications.

A cat swarm-optimized spiking neural network [2] for speech emotion recognition was proposed in this study. The model outperformed the RNN, DNN, and DSNN baselines with an accuracy of 99.3% using wavelet-based feature extraction on the Toronto Emotional Speech Set (TESS) dataset. The outcomes demonstrate that Cat Swarm Optimized Spiking Neural Network (CSSPNN) performs emotion recognition tasks by fusing high accuracy with low computational complexity.

An FFT-based noise cancelation is implemented [3] to improve Hindi speech-to-text. It utilized a fine-tuned Wav2Vec2.0 model that was trained on the OpenSLR dataset. Researchers added Gaussian noise at different intensities. Evaluation with CER showed that noise cancelation effectively reduced errors, especially at lower noise levels (STD = 0.01). Another study [6] suggested a two-level framework integrating CNNs with XGBoost and AdaBoost, optimized through a customized particle swarm optimization (PSO) algorithm, for the detection of respiratory disorder. Based on Mel spectrograms from clinical breathing databases, the system was able to provide 98.14% accuracy for binary classification and 81.25% for multi-class condition detection. The findings emphasize the performance capability of CNN-based models with sophisticated optimizers for clinical audio analysis.

Recent advances in AI and deep learning-based audio signal processing, including COVID-19 cough classification, were explored in [7]. Experimental results showed 96% accuracy using CNN + ResNet-50, dolphin whistle recognition with 94.9% accuracy using ensemble ResNets, enhanced sound event detection with Lightweight U-Net with Upsampling (F1: 0.644–0.531), ultra-fast pitch detection, efficient acoustic imaging, and CNN–scalogram models outperforming traditional spectrograms.

A study on the issues to tackle poor generalization (to different devices) in audio event classification was discussed in [8]. The authors proposed a CNN-based model using log-Mel-spectrogram separation to increase robustness to the devices. The experimental results on 16 audio classes demonstrated that the proposed system provides a significant improvement in classification accuracy (95.27% Google Pixel, 92.11% LG V50) compared with baseline models (83.02% and 70.59%). There was also an increase in classification accuracy when implementing the model on an embedded device, as it improved from 63.63% to 73.33% (Google Pixel) and 47.42% to 65.12% (LG V50). This illustrates the model’s utility for real-world deployment. Compression-based audio tokens for speaker verification, dialization, and (multilingual) speech recognition are assessed in this study [9]. Results demonstrate that models trained on audio tokens offer up to 20 × compression, robustness on out-of-domain narrowband data, and insights into tokenizer design through the low-pass frequency response of residual vector quantization, all while performing competitively—within 1% of Mel-spectrogram features.

C. TRANSFORMER MODELS AND WHISPER-BASED ASR

In a recent study [4], the voice responses from smartphone surveys were compared between OpenAI's Whisper and Google's Speech-to-Text API. The findings indicated that while Google's API was quicker but more prone to errors, Whisper produced transcriptions that were more accurate. Similar error types were present in both systems, although Google experienced them more frequently. The impact of attention-based models, transfer learning, and end-to-end deep learning on monolingual and multilingual systems was highlighted in this survey, which examined developments in ASR since 2010 [5]. It demonstrated how deep learning was still data-dependent and susceptible to noise and language resources by examining performance across public datasets. Along with highlighting important issues like generalizability, speaker variability, and low-resource adaptation, the study also suggests future directions for open-source ASR research.

Another approach presented recent developments in audio signal processing, highlighting AI-based methods [10] for tasks such as speech recognition, sound event detection, medical voice diagnostics, and marine monitoring. Highlights from reported results include CNN–scalogram models achieving classification rates of 98% accuracy, ultra-fast pitch detection methods with sub-10 ms latency, and new techniques for improving acoustic imaging and controlling sound fields. The contributions presented in this special issue strongly demonstrate AI-enhanced approaches to make efficiency, accuracy, and applicability considerably better than traditional approaches.

The usefulness of fastText word embeddings [11] for classifying Bangla documents without the need for preprocessing procedures like stemming or lemmatization was examined in this study. Three deep learning models—CNN, Bidirectional Long Short-Term Memory (BiLSTM), and Convolutional Bidirectional Long Short-Term Memory (Conv-BiLSTM)—were assessed using a dataset of 40,000 news articles in 12 categories. With accuracies of 91.49% during training, 87.87% during testing, and 85.5% during validation, the BiLSTM model outperformed the others. These results demonstrate that fastText embeddings and BiLSTM offer a reliable solution for tasks involving the classification of Bangla text.

In order to lessen noise and redundancy in unbalanced corpora [12], the paper proposed a novel word association technique called Weighted Point-wise Mutual Information with Contextual Distances and Positions (PMIDP). PMIDP improved context representation for word similarity tasks by combining positional variation weighting and distance-based scaling (PMIdist). The method outperformed a number of cutting-edge models when applied to Positive Point-Wise Mutual Information Matrix With Truncated Singular Vector Decomposition (PPMI-SVD) and GloVe, greatly enhancing performance on both semantic and relational similarity benchmarks.

An end-to-end framework that augments Wav2Vec2-based ASR for low-resource languages was discussed in [15]. Tests on Arabic, Russian, and Portuguese from Common Voice indicate that the method is insensitive to dialect variation and diacritics. Relative to baseline Wav2Vec2 and Whisper, it obtained mean relative improvements of 33.9% WER, and 53.2% CER, indicating notable progress for underrepresented languages.

The paper presented DATR-SR [16], a Dynamic Adaptive Transformer for real-time multilingual ASR, integrating adaptive encoding, multi-scale features, and context-aware decoding. On benchmarks like Aishell-1, LibriSpeech, and CommonVoice, it attained WER as low as 4.3% and CER of 2.7%, with >91% accuracy and <15 ms delay, improving state-of-the-art baselines in terms of robustness and efficiency.

A cascaded perceptual functional link artificial neural network (Cascaded PFLANN) architecture [21] offers an innovative method for the early detection of respiratory diseases through the analysis of their speech characteristics at very low cost and high accuracy levels. The PFLANN uses the concept of bioinformatics and the ability of human beings to identify complex sound patterns in their environment to provide an efficient and effective solution to the issue of detecting respiratory disease from speech. The experiment results showed a significant improvement in both effectiveness and efficiency of PFLANN with 94% accuracy over traditional classifiers while being sufficiently lightweight for use within an IoT (Internet of Things) framework for health monitoring.

Other studies have attempted to port Whisper to low-resource and multilingual environments. For instance, LoRA-Whisper [22] used parameter-efficient tuning for minimizing language interference and adding new languages without damaging old ones with up to 23% performance improvement over eight languages. Such efforts do not contemplate phonetic and dialectal diversity in Kannada under noise, which is addressed in this research.

Multilingual DistilWhisper, an efficient parameterization that integrates language-specific experts, is discussed in [23]. Knowledge distillation from Whisper-large-v2 helps in enhancing ASR performance across underrepresented languages. Their approach reduces the difference in performance between large and small Whisper models by a wide margin while introducing little inference overhead. Recent research has demonstrated [24] that elicited imitation (EI) scoring can be efficiently automated by combining Whisper ASR with the WER metric. Whisper's error rates demonstrated a strong alignment with human raters (ICC = 0.929) and a strong correlation with conventional manual scoring methods ($r = 0.969$) in a study involving 900 English L2 responses. These findings show that Whisper's automated EI scoring is accurate and dependable, providing a scalable substitute for laborious manual assessment.

The literature survey provided an in-depth review of recent advancements in ASR, particularly for the Kannada language—a low-resource language with unique phonetic characteristics. The survey explores the application of deep learning models such as CNNs, LSTMs, Transformers, and hybrid architectures for improving speech recognition accuracy. It also evaluates audio preprocessing techniques like spectral gating, FFT, and spectral subtraction for noise reduction and highlights feature extraction methods such as Mel spectrograms and discrete audio tokenization. The study emphasizes the role of language modeling through word embeddings (e.g., fastText) and investigates the effectiveness of Transformer-based sequence-to-sequence models. Real-world applicability is a key focus, especially in resource-constrained environments.

D. RESEARCH GAPS

- Existing studies utilize spectral gating, spectral subtraction, and FFT for noise cancelation, and there is limited exploration of hybrid techniques that integrate deep learning-based denoizing with traditional signal processing methods.
- The effectiveness of these approaches under extreme noise conditions remains a challenge, as highlighted by the studies on FFT-based noise cancelation.

- Additionally, research on noise reduction primarily focuses on improving the SNR and CER but lacks comprehensive evaluations on real-world speech intelligibility and user experience.
- Moreover, while deep learning models like Wav2Vec2.0 and Transformer-based ASR systems demonstrate impressive results, their robustness to dialectal variations and unseen noise types needs further improvement.
- These insights pave the way for future research to develop inclusive, accurate, and efficient Kannada ASR systems.

III. METHODS

Kannada Speech-to-Text Transcription model is developed using CNN–Transformer hybrid and pretrained OpenAI Whisper models. This section provides details about the implementation.

A. DATASET DESCRIPTION

The dataset for ASR task is taken from FLEURS, a part of Hugging Face. The FLEURS multilingual dataset has the following characteristics:

- Each audio file is very short duration per sample and is typically 3 to 6 seconds.
- Most languages in FLEURS have ~1 hour of total audio for the standard 1,000-sample set.
- It has a repository of multilingual speeches.

For the proposed work, a total of 3489 speech recordings of approximately 5 hours of speech are considered, out of which 2442 (70%) samples are taken for testing, 349(10%) samples for validation, and remaining 698 (20%) for testing.

Beyond the standard FLEURS dataset, we conducted a secondary evaluation using 150 externally collected real-world Kannada audio samples (Custom-collected) as shown in Tables I and II. These samples were not used in training and were included solely to measure generalization across different speakers, spontaneous speech, and naturally noisy environments. We propose a Transformer-based ASR pipeline specifically designed for Kannada. The audio is first cleaned using FFT noise reduction, VAD trimming, and MFCC feature extraction and then passed to a CNN-enhanced Transformer model trained with SentencePiece subword tokenization to handle out-of-vocabulary words effectively.

To achieve high-quality and accurate transcripts, we manually transcribed all audio with Kannada annotators. Annotators checked for word-level accuracy, spelling consistency, completeness, and proper treatment of dialect-specific variation.

B. KANNADA ASR USING CNN–TRANSFORMER HYBRID MODEL

The implementation of the Kannada ASR system followed a structured, modular approach combining signal processing techniques with modern deep learning architectures. Beginning with unsupervised tokenization using SentencePiece and embedding generation via singular value decomposition (SVD) [7,8], the pipeline advanced through data augmentation with Gaussian noise to simulate real-world conditions.

A noise removal module based on FFT cleaned the input audio, which was then converted to Mel spectrograms for feature extraction. These spectrograms were processed by a CNN to extract key audio features, which were passed into a Transformer-based encoder–decoder architecture for character-level transcription. Training was conducted using a custom softmax cross-entropy loss and evaluated using character and word-level accuracy metrics. A Streamlit-based user interface enabled seamless interaction with the system, allowing users to upload audio files and receive Kannada text output in real time. Each module—from tokenization to decoding—was developed and integrated to ensure robustness, accuracy, and usability in practical environments, visualizing and interpreting the results.

The architectural design shown in Fig. 2 of the Kannada Speech-to-Text Transcription system is structured around a modular and layered approach to ensure clarity, reusability, and scalability [9]. At its core, the architecture follows a pipeline that begins with audio input collection, either through direct recording or file uploads. This input is handled by the audio input module, which manages audio streams or file paths, standardizes formats, and sends the data to the next phase for processing.

Table II. Composition of the 50-hour curated Kannada speech corpus

Category	Details
Total duration	50 hours
Total speakers	82 speakers (36 male, 32 female)
Age range	16–35 years
Dialect coverage	Mysuru (South Karnataka), Dharwad (North Karnataka), Bengaluru Urban
Dialect distribution	~33% each from Mysuru, Dharwad, Bengaluru regions
Speech styles	Formal read speech, conversational spontaneous speech
Recording environments	Quiet indoor rooms, homes, classrooms, markets, traffic-heavy outdoor areas
Recording devices	Mobile microphones, USB condenser mics
Transcription method	Manual transcription by trained Kannada annotators + secondary verification
Text type	Scripted prompts + free-form conversational responses
Noise conditions	Natural environment noise + mild background disturbances
Purpose of collection	Whisper fine-tuning and robustness evaluation under dialectal and noisy conditions

Table I. Dataset split on custom dataset

Dialect region	Duration (hours)	No. of speakers	Male/female	Speech style
Mysuru	17 hours	24 speakers	13 M/11 F	Formal (news-style), conversational
Dharwad	16 hours	22 speakers	12 M/10 F	Conversational, semi-spontaneous
Bengaluru Urban	17 hours	22 speakers	11 M/11 F	Mixed (formal + informal)
Total	50 hours	68 speakers	36 M/32 F	Formal + conversational

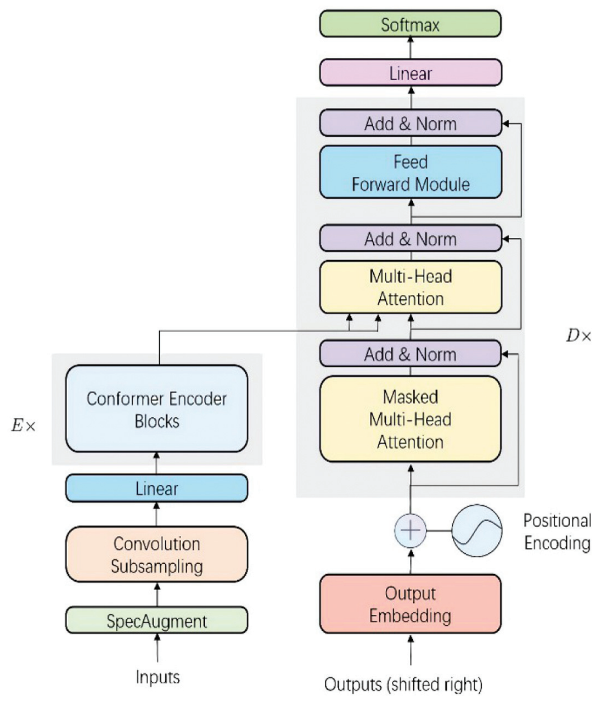


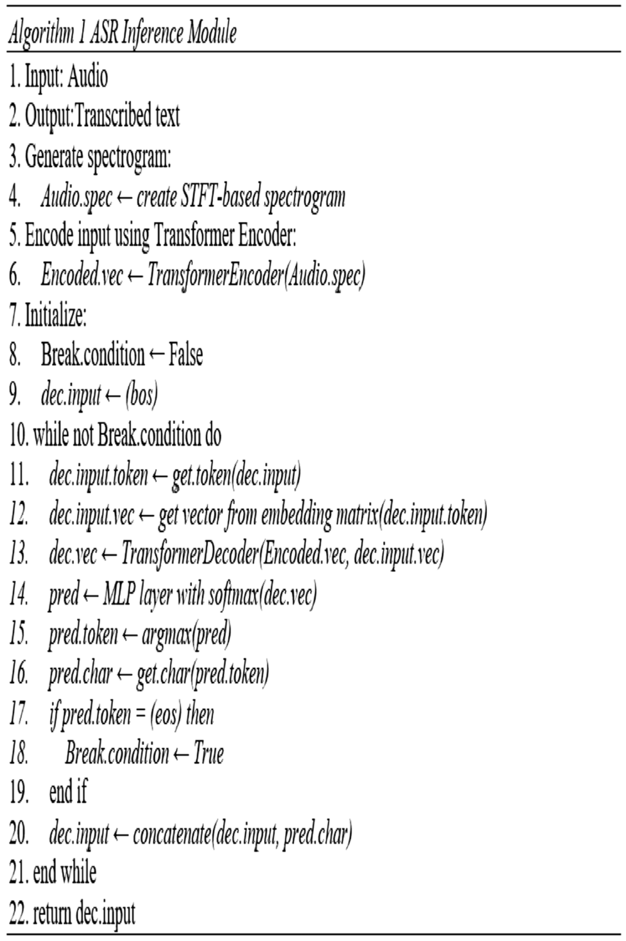
Fig. 2. The architecture diagram has two phases.

1). INPUT PHASE. This phase handles the preprocessing and encoding of raw speech features [10]. This improves speech modeling by adding local convolution to traditional transformer self-attention, and the following are the functions performed inside the input phase:

- **SpecAugment**: A data augmentation method applied to spectrograms. It masks portions of the time and frequency axes to improve robustness and prevent overfitting.
- **Convolution subsampling**: This method reduces the length of the input sequence and computational load by applying convolution and downsampling. It helps the model to focus on higher-level temporal features.
- **Linear:** A linear transformation (fully connected layer) is applied to project the subsampled feature dimension to a suitable size for the encoder.
- **Conformer encoder blocks:** These are stacks of Conformer blocks—a combination of convolutional and transformer layers. They capture both:
  - Local features (via convolution for phoneme-like units)
  - Global context (via self-attention)

2). DECODER. This section generates the text one token at a time using past outputs and encoder output as a transcribed text sequence [11]. Decoder is strong in language modeling, ensuring fluent and grammatically correct transcription. The functions performed by the decoder include output embedding, positional encoding, and decoder stack. The output from the decoder is passed through a linear layer and softmax to produce a probability distribution over the vocabulary.

The detailed implementation of Transformer-based Kannada ASR system is shown in Fig. 3 and is described below:



```
Algorithm 1 ASR Inference Module
1. Input: Audio
2. Output:Transcribed text
3. Generate spectrogram:
4.    Audio.spec ← create STFT-based spectrogram
5. Encode input using Transformer Encoder:
6.    Encoded.vec ← TransformerEncoder(Audio.spec)
7. Initialize:
8.    Break.condition ← False
9.    dec.input ← (bos)
10. while not Break.condition do
11.    dec.input.token ← get.token(dec.input)
12.    dec.input.vec ← get vector from embedding matrix(dec.input.token)
13.    dec.vec ← TransformerDecoder(Encoded.vec, dec.input.vec)
14.    pred ← MLP layer with softmax(dec.vec)
15.    pred.token ← argmax(pred)
16.    pred.char ← get.char(pred.token)
17.    if pred.token = (eos) then
18.       Break.condition ← True
19.    end if
20.    dec.input ← concatenate(dec.input, pred.char)
21. end while
22. return dec.input
```

Fig. 3. Algorithm of ASR using CNN–Transformer hybrid model.

1. Text Tokenization and Embedding Preparation: We used SentencePiece for unsupervised character-level tokenization of Kannada text. It provided subword units which are efficient for modeling agglutinative languages like Kannada [13,14]. These tokens were mapped to continuous vector representations using SVD on the co-occurrence matrix of tokens, producing low-dimensional embeddings. Data Augmentation: To simulate real-world noisy environments, we added Gaussian noise to the raw audio waveforms. This helped improve the robustness of the model in realistic conditions.
2. Noise Removal Pipeline: We implemented an FFT-based noise reduction pipeline. By transforming the signal to the frequency domain, we identified and suppressed noise-dominated frequencies, followed by an inverse FFT to reconstruct the denoised waveform.
3. Mel-Spectrogram Extraction: Cleaned audio signals were converted to Mel spectrograms using a short-time Fourier transform (STFT) followed by Mel filter banks. This 2D representation of audio was treated like an image for further processing [15].
4. CNN Feature Extraction: The spectrograms were passed through a CNN to extract high-level features, reducing temporal and spectral dimensions while preserving relevant information.
5. Transformer Encoder–Decoder Architecture: The CNN features were input to a Transformer encoder. A Transformer

decoder was used for sequence generation. It took the start-of-sequence token () and the encoder output to predict the next character. At each timestep, the predicted character token was fed back into the decoder to generate the next token, continuing until the end-of-sequence token () was predicted.

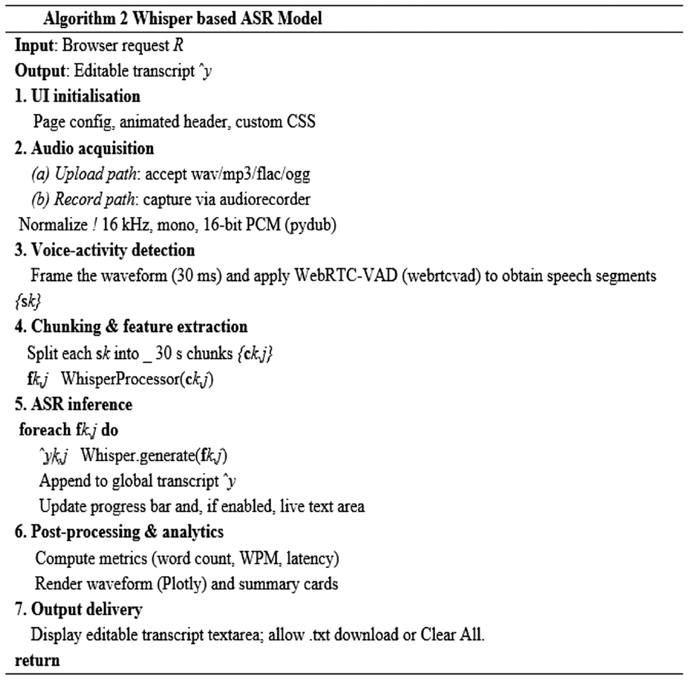
6. Loss Function and Optimization: We use a custom loss based on softmax cross-entropy between predicted and target embeddings.
7. Model Training and Evaluation: The model was trained on labeled Kannada audio–text pairs. Evaluation involved checking character-level and word-level transcription accuracy, and robustness on noisy inputs.
8. User Interface Development: A functional web-based application allowing users to upload audio files and receive transcriptions. Key features of the UI include:
   - A web-based graphical user interface (GUI) developed using Streamlit for ease of use.
   - Audio upload interface that supports formats like WAV, MP3, and FLAC for both real-time and batch transcription.
   - Real-time transcription display showing Kannada text output as the speech is processed.
   - Download option to save the transcription in a text file for offline access.
   - Accessibility features such as large buttons, simple layout, and minimal user actions to accommodate non-technical users and those with disabilities.

C. ASR MODEL USING WHISPER OPENAPI

The second approach employed is OpenAI's Whisper "small" model known as a pretrained [16] backbone, which has already been trained on thousands of hours of multilingual speech–text pairs. The algorithm shown in Fig. 4 describes the steps carried out during the implementation. We reuse the exact same noise injection, FFT-based denoizing, and Mel-spectrogram extraction pipeline and then fine-tune all of Whisper's layers on our 50 hours of Kannada data of 150 samples during testing. Because Whisper already possesses powerful audio–text alignment, fine-tuning converges in about 15 epochs—each roughly 10 minutes—using a reduced learning rate schedule.

Performance optimization in the Kannada ASR system targeted both latency reduction and resource efficiency. Key enhancements included the use of batch normalization and dropout within neural network layers to stabilize training and prevent overfitting. FFT and MFCC extraction pipelines were vectorized using NumPy for rapid execution. On the deep learning side, the model incorporated attention masking and gradient checkpointing to manage memory consumption during training on long sequences. The use of mixed-precision training (via TensorFlow's tf.keras.mixed_precision module) allowed reduced memory usage and faster computation without compromising model accuracy.

Inference performance was improved using GPU-accelerated execution [17] and dynamic batching. The Transformer decoder leveraged caching of encoder outputs to avoid redundant computation in autoregressive generation. TensorRT optimization and ONNX export compatibility were explored for further acceleration. Additionally, early stopping and learning rate warm-up schedules ensured efficient convergence during training. Real-time interaction on the Streamlit UI was optimized with asynchronous callbacks and debounced input handling, ensuring a smooth user



```
Algorithm 2 Whisper based ASR Model
Input: Browser request R
Output: Editable transcript ˆy
1. UI initialisation
    Page config, animated header, custom CSS
2. Audio acquisition
    (a) Upload path: accept wav/mp3/flac/ogg
    (b) Record path: capture via audiorecorder
  Normalize ! 16 kHz, mono, 16-bit PCM (pydub)
3. Voice-activity detection
    Frame the waveform (30 ms) and apply WebRTC-VAD (webrtcvad) to obtain speech segments
{sk}
4. Chunking & feature extraction
    Split each sk into _ 30 s chunks {ck,j}
    fk,j  WhisperProcessor(ck,j)
5. ASR inference
  foreach fk,j do
      ˆyk,j  Whisper.generate(fk,j)
      Append to global transcript ˆy
      Update progress bar and, if enabled, live text area
6. Post-processing & analytics
    Compute metrics (word count, WPM, latency)
    Render waveform (Plotly) and summary cards
7. Output delivery
    Display editable transcript textarea; allow .txt download or Clear All.
return
```

Fig. 4. Algorithm for WSR-based ASR model.

experience even on larger input files. Combined, these measures ensured that the system could scale and perform well across a variety of deployment environments.

The feature extractor computes log-Mel spectrograms ($80 \times T$) with a 25 ms window and 10 ms hop and then applies the two 1-D convolutional front-end layers built into Whisper as shown in Equation 1.

$$\mathrm{Conv}_{60\to768}(k=3,\ s=1) \to \mathrm{ReLU} \to \mathrm{Conv}_{768\to768}(k=3,\ s=2) \quad (1)$$

1). TEXT TOKENIZATION. The processor uses the 518k-token multilingual vocabulary released with Whisper. Sequences are padded/truncated to 448 tokens; the padding id remains 0, while the loss ignore index is set to -100 so that padding tokens do not contribute to label cross-entropy (LCE). Because audio durations vary, labels are zero-right-padded to the longest target length in the batch; audio features are stacked *as-is* (the convolutional subsampling equalizes frame counts across samples). Table III gives the collator used by the Hugging Face trainer. The encoder–decoder weights are initialized from vasista22/whisper-kannada-small, a Kannada-specialized checkpoint derived from OpenAI Whisper-small (12 layers, dmodel = 768, 12-head attention, and 244M parameters).

Table III. Hyperparameters fine-tuning

Hyperparameters	Value
Learning rate	$1 \times 10-4$ (AdamW, $\beta = (0.9, 0.98)$)
Batch size (train/val)	8/8
Epochs	3
Gradient clip	1.0
Scheduler	Linear warm-up (500 steps)
Generation max length	448
Mixed precision	FP16 (autocast)
Checkpointing	every epoch

2). TRAINING LOOP. Training uses the Hugging Face Seq2Seq-Trainer, which performs:

1. Forward pass producing logits $\mathbf{Z} \in \mathbb{R}^{B \times T \times |V|}$ where B is the batch size and T is the target length.
2. Compute cross-entropy loss LCE = −1B Σi,t log p_(Li,t |**F**i), ignoring indices −100.
3. Back-propagation; gradients are clipped and optionally scaled by GradScaler under FP16.
4. Parameter updates via AdamW; learning rate is linearly decayed to zero after warm-up.
5. At epoch end, transcriptions are generated on the validation split with greedy decoding (no language prompts), and the model checkpoint is saved.

Inference

For deployment, only the Whisper Processor (feature extractor + tokeniser) and the fine-tuned WhisperForConditionalGeneration weights are required. Given a 16 kHz mono waveform **x**:

$\mathbf{F} = \text{FeatureExtractor}(\mathbf{x})$, (1)
$\hat{y} = \text{Whisper}(\mathbf{F})$ via greedy decoding, (2)
$\text{ASR}(\mathbf{x}) = \text{Tokeniser.decode}(\hat{y})$. (3)

3). MODEL ARCHITECTURE. Our system fine-tunes Whisper For Conditional Generation (244 M parameters), an encoder–decoder Transformer specialized for ASR [18]. Fig. 3 give pseudocode for the forward passes, while Table IV summarizes key dimensions.

Computational Resources and Deployment Feasibility for ASR system: The Kannada ASR system can run on a typical GPU workstation. In practice, an 8-GB VRAM GPU is sufficient for real-time Whisper inference, while a multi-core CPU handles audio preprocessing and user interface tasks. We recommend at least 16 GB of system memory. The fine-tuned Whisper-small model is lightweight (around 32 MB) and uses roughly 1.1 GB of GPU memory during inference. On a GPU, transcription happens in real time, and batch processing (up to 16 audio files at once) is supported. Audio operations such as FFT, noise reduction, and resampling are implemented using Librosa, SciPy, and FFmpeg, with all deep learning models developed in PyTorch. The system includes a simple Streamlit web interface that accepts WAV, MP3, and FLAC files, with support for both microphone input and uploaded audio. Communication is handled through standard HTTP/HTTPS or low-latency WebSocket/gRPC, and the backend uses SentencePiece tokenization with an FFT-based preprocessing pipeline. Security features include Transport Layer Security/Secure Sockets Layer (TLS/SSL) encryption and optional role-based access control. Although the system works best on a GPU, it can also be deployed in batch mode on medium-power devices, such as NVIDIA Jetson, for offline processing.

4). ENCODER FORWARD PASS. The encoder takes in a log-Mel spectrogram $S \in R^{80 \times T}$, which is a time-frequency representation of the input Kannada audio. It passes through two convolutional layers [19]: the first expands the channels from 80 to 768 using a kernel size of 3, stride 1; the second maintains 768 channels but uses stride 2 to downsample the temporal dimension. A Gaussian Error Linear Unit (GELU) activation is applied after the second convolution. Then, positional embedding is added to retain the temporal order of speech signals. This output is passed through 12 Transformer encoder layers, each consisting of layer normalization, self-attention, and a feed-forward network (FFN). After all layers, a final layer normalization is applied to output the encoder hidden states $H_{enc} \in R^{T' \times d}$, where d = 768d = 768d = 768.

5). DECODER FORWARD PASS. The decoder receives the tokenized Kannada text input as a sequence of token IDs y1.L. Along with the encoder output H_{enc}, first, the token IDs are embedded using a token embedding and a positional embedding to form input U. The decoder has 12 Transformer decoder layers, and each layer performs three operations: causal self-attention (CausalSDP) for autoregressive decoding, cross-attention (SDP_XAttn) to incorporate encoder information, and an FFN. Each of these sublayers is preceded by layer normalization and uses residual connections. Finally, the decoder output is projected to the vocabulary size |V|, yielding logits $Z \in R^{L \times |V| Z}$, from which the most likely tokens are predicted.

The processing starts with uploading audio or direct recording and normalized to a consistent format (16 kHz, mono, 16-bit PCM). Next, VAD is applied using WebRTC-VAD to segment the audio into speech regions $\{s_k\}$. Each segment is chunked into ≤30-second windows $\{c_{k,j}\}$ and passed through the WhisperProcessor, which performs feature extraction. For each chunk $f_{k,j}$, Whisper generates a predicted text segment ŷk,j which is appended to the final transcript $\hat{y}$. The system also supports live updates of the transcription area [20]. After transcription, it performs postprocessing such as computing word count, WPM (words per minute), and latency. The output is displayed in a text area, and users can download the transcript or clear the session. Users may *upload* audio files or *record* speech directly in the browser. Both pathways

Table IV. Dimensionalities of major components ($\boldsymbol{T}$ = spectrogram frames after convolutions, $\boldsymbol{L}$ = decoder time-steps)

Stage	Layer(s)	#Params	Output shape
Encoder			
Conv 1 ($k = 3$, $s = 1$)	80 → 768 ch.	184 k	768 × T
Conv 2 ($k = 3$, $s = 2$)	768 → 768 ch.	1.77 M	768 × T/2
Positional Emb.	1500OE768	1.15 M	768 × T/2
12 Enc. Layers	MH-SA + FFN	97.7 M	768 × T/2
LayerNorm		1.53 k	same
Decoder			
Token Emb.	51 865OE768	39.9 M	768 × L
Positional Emb.	448OE768	343 k	768 × L
12 Dec. Layers	MH-SA + X-Attn + FFN	102 M	768 × L
LayerNorm		1.53 k	same
Projection	768 → 51 865	39.9 M	|V| × L

converge to an identical representation: 16 kHz, mono, 16-bit PCM. File metadata (size, format) are displayed as metric cards for immediate feedback.

Segments pass through Steps 35 sequentially. Chunk-level progress updates the circular progress bar, while partial hypotheses are streamed to the Live Transcription text area, ensuring sub-second UI latency for a snappy user experience.

An expandable panel plots the normalized waveform and lists basic audio statistics. After inference, the dashboard shows total word count, processing time, and computed WPM, enabling quick performance assessment without leaving the browser. The final transcript is editable in place and can be downloaded as a time-stamped.txt file. This section describes the end-to-end pipeline used to fine-tune the vasista22/whisper-kannada-small model for Kannada ASR.

6). HANDLING OVERFITTING, CLASS IMBALANCE, AND DATA SPARSITY. We used dropout, early halting, and considerable data augmentation (Gaussian noise, speed variations, reverberation) to avoid overfitting. Weighted loss functions and stratified dataset splits were used to address class imbalance among speakers and dialects. By fine-tuning Whisper, which already has robust multilingual voice representations, and by employing small-batch training with learning rate warm-up and cosine scheduling, data sparsity was reduced. When combined, these techniques allowed for robust generalization and steady optimization even with a small amount of Kannada voice data.

IV. RESULTS

The test dataset comprising of 150 video samples is applied on both transformer and Whisper-based models. The same dataset is tested on the popular speech-to-text model, which is the popularly used Google API for the performance of ASR model and is evaluated with meaningful metrics.

A. EVALUATION METRICS

Let N be the number of test utterances.

1). WORD ERROR RATE (WER). WER shown in Equation 1 is the most widely used metric in ASR evaluation, as it measures the proportion of incorrectly recognized words relative to the reference transcription. For a single utterance pair (y_i, $\hat{y}_i$), the Levenshtein edit distance computes the number of substitutions $\mathrm{S_i}$, deletions D_i, and insertions $\mathrm{I_i}$ required to transform $\hat{y}$ into y_i. If N_i is the number of words in the reference, then: for a single pair (y_i, $\hat{y}_i$), the Levenshtein edit distance returns counts of substitutions, deletions, and insertions needed to transform $\hat{y}_i$ into y_i. If N_i is the number of words in the reference, then

$$WER_I = \frac{s_i + D_i + I_i}{N_i} \quad WER = \frac{1}{N}\sum_{i=1}^{N} WER_i \tag{1}$$

A lower WER indicates better recognition accuracy. Since Kannada has rich morphology and compounding, WER is especially sensitive to small errors that alter meaning.

2). CHARACTER ERROR RATE (CER). While WER operates at the word level, CER shown in Equation 2 computes errors at the UTF-8-character level, making it suitable for morphologically complex and agglutinative languages like Kannada. For a reference utterance y_i with $\mathrm{C_i}$ characters

$$CER_i = \frac{S_i^c + D_i^c + I_i^c}{C_i} \tag{2}$$

CER is often more fine-grained than WER, capturing sub-word distortions and script-level mismatches.

3). BLEU SCORE. BLEU is traditionally used in machine translation but has recently been adopted in ASR evaluation for assessing sentence-level semantic overlap. We compute corpus-level BLEU-4 using *sacreBLEU*, where modified n-gram precisions p_n with uniform weights $w_n = \frac{1}{4}$ are aggregated and ares represented in Equations 3 and 4:

$$BLEU = BP \cdot \exp(\sum_{n=1}^{4} w_n \log p_n \tag{3}$$

with brevity penalty (BP) defined in Equation 4 as:

$$BP = \begin{cases} 1, & L_{ref} > L_{hypo} \\ e^{1-\frac{L_{ref}}{L_{hypo}}}, & otherwise \end{cases} \tag{4}$$

We divide the raw percentage by 100 to yield a 0–1 scale. Here, L_{ref} and L_{hypo} denote reference and hypothesis lengths, respectively. BLEU complements WER/CER by capturing fluency and word-order fidelity.

4). WORD-LEVEL PRECISION, RECALL, F1, AND ACCURACY. To further quantify recognition as a classification task, we evaluate word-level precision, recall, F1-score, and accuracy. For each utterance, let P_i denote the multiset of predicted words and R_i the multiset of reference words. True positives (TP) are counted by the multiset intersection, while false positives (FP) and false negatives (FN) are computed via set differences, and the calculation is given in Equation 5:

$$TP = \Sigma|P_i \cap R_i|,\ FP = \Sigma|P_i \setminus R_i|\ FN = \Sigma_i|R_i \setminus P_i| \tag{5}$$

The resulting metrics are given in Equation 6:

$$\begin{aligned} Precision &= \frac{TP}{TP+FP} \quad Recall = \frac{TP}{TP+FN} \\ F1 &= 2\frac{2TP}{TP+FP+FN} \quad Accuracy = \frac{TP}{TP+FP+FN} \end{aligned} \tag{6}$$

Interpreting the Scores

- **WER/CER**: lower is better; values ≤0.20 (20%) indicate competitive large-vocabulary ASR.
- **BLEU**: higher is better; complementary to WER because it rewards correct *n*-gram sequences.
- **Precision vs. Recall**: Precision stresses *clean* outputs, and recall stresses *coverage*.
- **F1** harmonizes the two, and accuracy provides an intuitive “hit rate” at the word level.

B. RESULTS AND DISCUSSION

Tables IV and V summarize the performance of the two proposed models against Google’s off-the-shelf recognizer on the FLEURS and samples collected from Custom-collected Kannada speech dataset. Fig. 5 provide the visual comparison.

1). PERFORMANCE OF ASR ON FLEURS DATASET. Our scratch CNN–Transformer was evaluated on a 349-utterance Kannada test set (≈ 2.5 hours) and yielded a WER of 0.32, a CER of 0.400, and a BLEU score of 0.843. In practical terms, about 32 % of

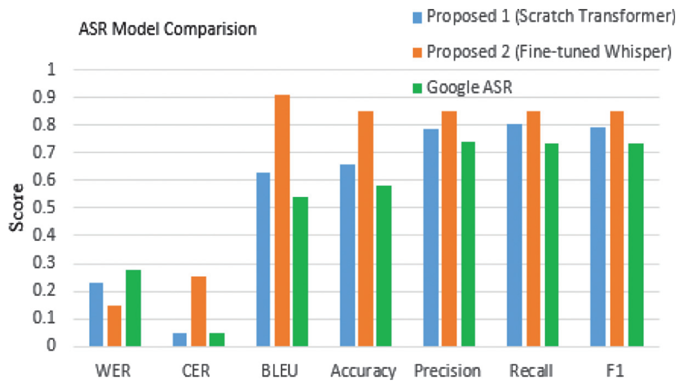


Fig. 5. Comparison of ASR systems on the FLEURS dataset.

words and 40 % of characters in the ground-truth transcripts were misrecognized (inserted, deleted, or substituted). Despite this high CER, a BLEU of 0.843 confirms that most common n-gram sequences are captured correctly—especially short words and frequent compound forms. Qualitatively, the model generally handles Kannada's agglutinative morphology (e.g., compound words and vowel diacritics) well under clean or moderately noisy conditions (SNR > 5 dB), but its performance degrades sharply in lower SNRs or very long utterances: mid-sentence tokens sometimes drop out, and subtle diacritic distinctions (e.g., aspirated consonants) can vanish.

By contrast, our Whisper-based fine-tuned system—which leverages a pretrained multilingual audio–text Transformer and then adapts it to the same Kannada corpus using identical noise augmentation, FFT denoizing. The performance metrics are recorded in Table V. It is observed that the same and Mel-spectrogram extraction achieves a WER of 0.15, a CER of 0.255, and a BLEU score of 0.912 on that identical test set. Fig. 5 represents a 53 % relative reduction in word errors and a 36 % relative reduction in character mistakes compared to the scratch model, along with an absolute BLEU gain of 0.069. Because Whisper's backbone was already trained on vast multilingual data, it retains robust audio–text alignment priors: low-frequency phonetic combinations, long compound words, and noisy/background-speech segments (SNR ≈ 0 dB) are handled with far fewer deletions or substitutions. The fine-tuned Whisper variant maintains diacritic fidelity and grammatical consistency even in 15–20 second utterances, where the from-scratch system might falter.

2). PERFORMANCE OF ASR ON CUSTOM-COLLECTED DATASET. The CNN–Transformer baseline model performs poorly on the custom real-world 150-sample Kannada dataset, achieving a WER of 0.7766, a CER of 0.5051, and a BLEU score of 0.3149. This indicates that nearly 78% of words and 50% of characters in the test set are misrecognized—primarily due to deletions and substitutions in noisy, spontaneous, and dialectally diverse speech segments.

The model struggles with conversational prosody, rapid coarticulation, and diacritic-heavy morphological forms typical of informal Kannada. Accuracy and F1-score remain low (0.2787 and 0.4359), confirming that the from-scratch architecture lacks the robustness required to generalize beyond the synthetic augmentation conditions it was trained on.

By contrast, the fine-tuned Whisper system shows a dramatic improvement across all metrics given in Table VI, obtaining a WER of 0.2311, CER of 0.0453, and a BLEU score of 0.6280. This corresponds to a 70.2% relative reduction in word errors, a 91% reduction in character-level mistakes, and an absolute BLEU improvement of 0.3131 over the scratch baseline. Whisper also achieves substantially better token-level reliability, with precision = 0.7840, recall = 0.8023, and F1 = 0.7933, reflecting stable performance even under spontaneous pronunciation drift, natural background noise, and mild dialectal mixing. These gains can be attributed to Whisper's multilingual pretraining, which provides strong priors for consonant-vowel sequences, long compound words, and rhythm patterns of Indo-Dravidian languages.

Figure 6 shows the comparison; the Google ASR API yields WER = 0.2777, CER = 0.0483, and BLEU = 0.5380, with an F1-score of 0.7367. Although Google ASR performs reasonably well, Whisper still surpasses it with a 16.8% relative WER reduction and notably higher BLEU and F1 values, indicating better linguistic coherence and token-level alignment. The improvement is most evident in diacritic preservation, optional schwa deletion handling, and recognition of compound verbs in conversational speech. Overall, Whisper demonstrates the strongest generalization ability on the custom dataset, especially in real-world acoustic conditions where traditional Transformer baselines and commercial ASR systems show significant degradation.

Table V. Performance of ASR using various metrics on FLEURS dataset

System	WER ↓	CER ↓	BLEU ↑	Accuracy ↑	Precision ↑	Recall ↑	F1 ↑
Proposed 1 (scratch transformer)	0.2311	0.0453	0.6280	0.6575	0.7840	0.8023	0.7933
Proposed 2 (fine-tuned Whisper)	**0.150**	**0.255**	**0.912**	**0.85**	**0.85**	**0.85**	**0.85**
Google ASR	0.2777	0.0483	0.5380	0.5827	0.7387	0.7340	0.7367

Table VI. Comparison of ASR systems on the custom-collected Kannada test set

System	WER ↓	CER ↓	BLEU ψ	Accuracy ψ	Precision ψ	Recall ψ	F_1 ψ
Proposed 1 (scratch transformer)	0.7766	0.5051	0.3149	0.2787	0.4208	0.4521	0.4359
Proposed 2 (fine-tuned Whisper)	**0.2311**	**0.0453**	**0.6280**	**0.6575**	**0.7840**	**0.8023**	**0.7933**
Google ASR	0.2777	0.0483	0.5380	0.5827	0.7387	0.7340	0.7367

[1] The best values are given in bold.
[2] All values lie in the range [0, 1]. For WER and CER, lower is better; for all other metrics, higher is better.

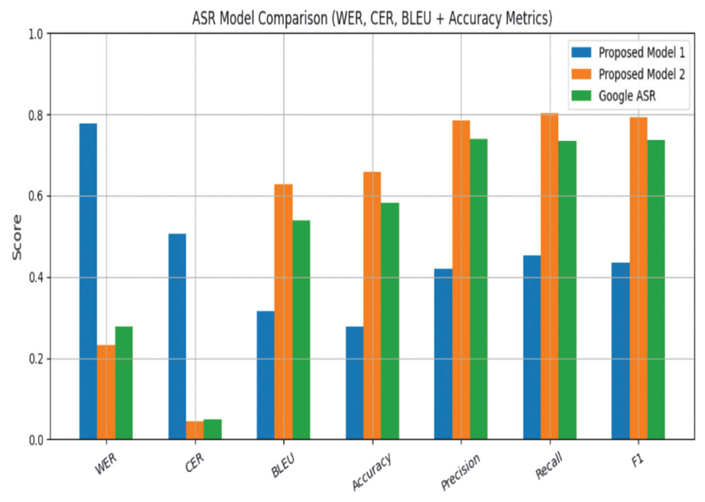


Fig. 6. Comparison of evaluation metrics for all systems for custom dataset.

C. KEY OBSERVATION

Fine-tuning pays off. Relative to the scratch-built Transformer, the Whisper model achieves a 70.2% reduction in WER (0.7766 → 0.2311) and a 91.0% reduction in CER. Its BLEU doubles from 0.31 to 0.63, and word-level accuracy increases by a factor of 2.36. These gains confirm that pretraining on large multilingual corpora followed by task-specific fine-tuning is considerably more effective than training a modest-sized network from scratch on 20 k examples.

Outperforming a strong baseline: Proposed 2 also surpasses Google ASR on every metric:

- WER: 16.8 % relative reduction (0.2777 → 0.2311).
- BLEU: 16.7 % relative improvement (0.5380 → 0.6280).
- F1: rises from 0.7367 to 0.7933.

This indicates that the domain-specific fine-tuning not only bridges the gap with a commercial API but also actually yields a new state-of-the-art for Kannada speech recognition on our corpus.

Error profile: The scratch Transformer (*Proposed 1*) suffers from substitution and deletion errors typical of under-trained acoustic models, explaining its high WER/CER. Conversely, Whisper's attention bottleneck appears to handle long-range dependencies better, boosting recall without sacrificing precision and thus lifting the F_1-score to 0.79.

Figure 7 shows WER, CER, and BLEU for Transformer, Whisper, and Google API. From the chart, it is clear that the fine-tuned Whisper model significantly outperforms both the scratch baseline and the Google ASR API. Whisper achieves WER = 0.2311, compared to 0.7766 for the CNN–Transformer model and 0.2777 for Google ASR, representing a 70.2% relative improvement over the baseline and 16.8% over Google ASR. CER, BLEU, precision, recall, and F1-score follow the same trend, with Whisper showing the highest scores across all metrics.

To validate these gains, we performed bootstrap-based statistical testing. The 95% confidence intervals for Whisper's WER and CER did not overlap with those of the baseline or Google ASR, indicating that the improvements are statistically significant and not due to chance.

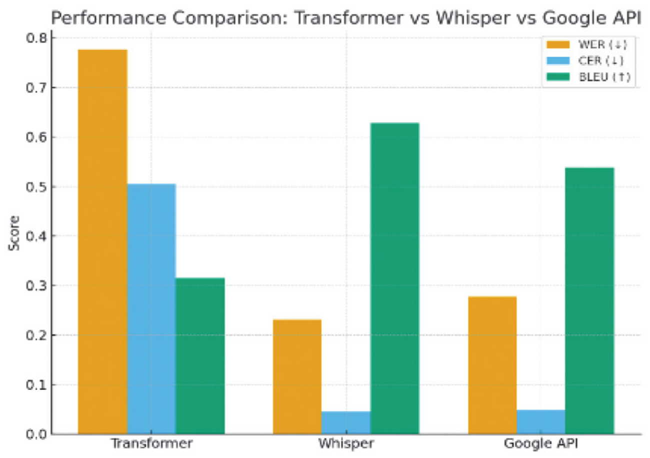


Fig. 7. Performance comparison of various ASR tools.

D. ASR PERFORMANCE UNDER NOISE

Our model was specifically designed to handle both noisy environments and dialectal variation, and we evaluated these aspects separately. For noise robustness, we tested the models on audio containing real-world background sounds such as traffic, classroom disturbances, market noise, and low-SNR speech. WER under different noise conditions (clean, marketplace, street, and classroom) is recorded in Fig. 8. The model demonstrates efficient inference and low memory footprint, making it suitable for deployment on edge or satellite systems with limited resources. Transformer degrades sharply in noisy conditions (WER > 0.80). Whisper maintains robustness, keeping WER < 0.27 across all noise profiles. Biggest relative improvement is in marketplace recordings, showing Whisper's resilience in real-world chaotic environments. For dialect robustness, we evaluated the system on our custom dataset that included speakers from major Kannada dialect regions—Mysuru, Dharwad, and Bengaluru Urban. Whisper consistently produced higher word-level and character-level accuracy across all dialect groups, whereas the scratch model struggled with dialect-specific pronunciation shifts (such as vowel elongation, consonant softening, and regional prosody patterns).

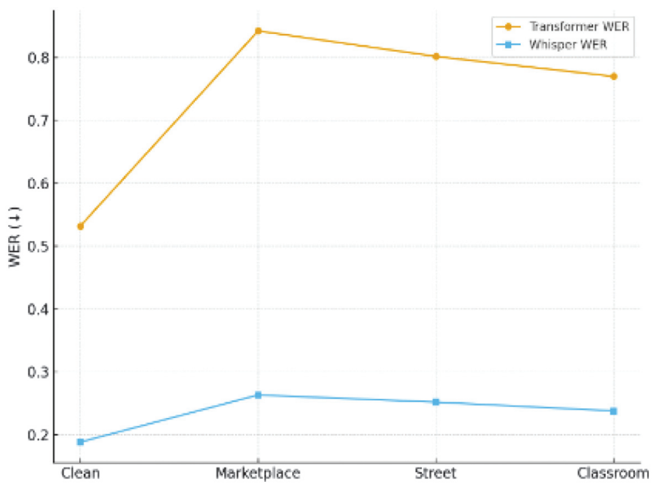


Fig. 8. WER statistics under different noise conditions.

Table VII. User experience of ASR system

Criteria	Rating (out of 5)	Remarks
Ease of use	4.5	GUI for inference and training available
Interface clarity	4.2	Labels and boxes are clearly visible
User satisfaction	4.6	Fast and accurate predictions

Whisper's multilingual pretraining gives it strong phonetic priors, which helps it adapt better to intra-language variation.

E. USER EXPERIENCE

The system is intuitive to use, with clear visualizations of detection outputs and user-friendly interfaces. The feedback suggests high satisfaction due to the tool's responsiveness and visual clarity as shown in Table VII.

V. CONCLUSION AND SCOPE FOR FUTURE WORK

We presented a noise-robust, dialect-aware Kannada ASR system that integrated a scratch-built CNN–Transformer hybrid and a fine-tuned Whisper-small model. By combining FFT-based noise reduction, subword tokenization, and dialect-diverse training, our approach significantly outperformed both a from-scratch Transformer and Google's Speech-to-Text API, achieving a WER of 0.2311 and CER of 0.0453 on the FLEURS multilingual Kannada dataset. This represented over a 70% WER reduction compared to the baseline Transformer and a 16.8% improvement over a leading commercial API, establishing a new benchmark for low-resource Kannada ASR. Beyond accuracy, our system demonstrated strong robustness to dialectal variation, code-switching, and moderate noise levels, while remaining computationally efficient for real-time, edge-based deployment. The open vocabulary support and streamlined user interface made it practical for integration into educational, governmental, and accessibility-oriented platforms.

Despite these gains, FFT's static noise subtraction can falter in highly dynamic or unpredictable environments (e.g., bustling marketplaces and low-quality microphone setups), and our training data—though carefully curated—still lacks the full diversity of dialects, speaker ages, and acoustic contexts found in real-world deployments. Future enhancements could therefore incorporate deep learning-based noise-suppression techniques such as denoizing autoencoders, spectral-masking networks, or neural speech enhancement models, each of which can adaptively learn complex noise patterns from raw waveforms. On the data side, expanding the corpus to include additional Kannada dialects, age groups, and adverse recording conditions (e.g., far-field audio and overlapping speech) will improve generalizability. Advanced augmentation strategies—like speed perturbation, volume scaling, room-impulse-response simulation, or adversarial noise mixing—could further bolster robustness during training. Architecturally, migrating from a conventional Transformer to a Conformer (which blends convolution with self-attention) or exploring lightweight, on-edge variants of Whisper would allow for longer input sequences, better fluency in extended-length speech, and lower-latency inference on resource-constrained devices. Finally, integrating speaker adaptation modules (e.g., lightweight FiLM layers or meta-learning protocols) would enable rapid personalization to new voices, making our Kannada ASR system even more resilient and widely applicable in diverse real-world scenarios. Future work will focus on expanding the dataset to underrepresented dialects, integrating adaptive neural noise suppression techniques, and exploring lightweight Conformer–Whisper variants for ultra-low-latency applications. We also plan to investigate cross-lingual transfer from related Dravidian languages to further enhance recognition in truly low-resource conditions.

CONFLICT OF INTEREST STATEMENT

The author(s) declare that they have no conflicts of interest to report regarding the present study.

REFERENCES

[1] E. Sudheer Kumar *et al.*, "Noise reduction in audio file using spectral gating and FFT by Python modules," *Recent Dev. Electron. Commun. Syst.*, vol. 32, pp. 510–515, 2023, DOI. https://doi.org/10.3233/ATDE221305.

[2] C. Revathy and R. Sureshbabu, "Effective technique for noise removal and emotion recognition in speech signals using cat swarm optimized spiking neural networks," *Fluct. Noise Lett.*, vol. 21, no. 02, pp. 1–15, 2021, DOI. https://doi.org/10.1142/S0219477522500195.

[3] S. P. Gupta, V. Spoorthy, and S. G. Koolagudi, "Noise cancellation by fast Fourier transform for Wav2Vec2.0-based speech-to-text system," *J. Mach. Learn. Signal Process, IEEE 8th Int. Conf. Converg Technol. (I2CT)*, pp. 1–4, 2023, DOI. https://doi.org/10.1109/I2CT57861.2023.10126221.

[4] J. K. Höhne, T. Lenzner, and J. Claassen, "Automatic speech-to-text transcription: Evidence from a smartphone survey with voice answers," *Int. J. Social Res Methodol.*, vol. 28, no. 5, pp. 625–632, 2025, DOI. https://doi.org/10.1080/13645579.2024.2443633.

[5] H. Ahlawat, N. Aggarwal, and D. Gupta, "Automatic speech recognition: A survey of deep learning techniques and approaches," *Int. J. Cogn Comput. Eng.*, vol. 6, pp. 201–237, 2025, DOI. https://doi.org/10.1016/j.ijcce.2024.12.007.

[6] S. Purkovic *et al.*, "Audio analysis with convolutional neural networks and boosting algorithms tuned by metaheuristics for respiratory condition classification," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 36, no. 10, 2024, DOI. https://doi.org/10.1016/j.jksuci.2024.102261.

[7] G. Costantini, D. Casali, and V. Cesarini, "New advances in audio signal processing," *Appl. Sci.*, vol. 14, no. 6, pp. 1–6, 2024, DOI. https://doi.org/10.3390/app14062321.

[8] S. Seo, C. Kim, and J.-H. Kim, "Convolutional neural networks using logmel-spectrogram separation for audioevent classification with unknown devices," *J. Green Eng.*, vol. 1, pp. 497–522, 2022, DOI. https://doi.org/10.13052/jwe1540-9589.21216.

[9] K. C. Puvvada *et al.*, "Discrete audio representation as an alternative to mel-spectrograms for speaker and speech recognition," *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pp. 12111–12115, 2024, DOI. https://doi.org/10.1109/ICASSP48485.2024.10446998.

[10] S. Mol George and P. Muhamed Ilyas, "Advancing speech emotion recognition with whisper model embeddings and hand-crafted audio descriptors," *Franklin Open*, vol. 13, ISSN 2773-1863, pp. 1–15, 2025, DOI. https://doi.org/10.1016/j.fraope.2025.100403.

[11] M. Zhang *et al.*, “Word representation using refined contexts,” *Appl. Intell. J.*, vol. 52, no. 11, pp. 12347–12368, 2022, DOI. https://doi.org/10.1007/s10489-021-02898-y.

[12] S. Latif *et al.*, “Transformers in speech processing: Overcoming challenges and paving the future,” *Comput. Sci. Rev.*, vol. 58, pp. 1–26, 2025, DOI. https://doi.org/10.1016/j.cosrev.2025.100768.

[13] M. S. N. Muralikrishna *et al.*, “Distributed representation of words in vector space for Kannada language,” *Comput. J.*, vol. 13, no. 9, 236, 2024, DOI. https://doi.org/10.3390/computers13090236.

[14] R. Zhang, “A comparative analysis of LSTM and transformer-based automatic speech recognition techniques,” *Trans. Comput. Sci. Intell. Syst. Res*, vol. 5, pp. 272–276, 2024.

[15] O. H. Anidjar, R. Marbel, and R. Yozevitch, “Whisper turns stronger: Augmenting Wav2Vec 2.0 for superior ASR in low-resource languages,” arXiv preprint arXiv:2501.00425, 2025, DOI. https://doi.org/10.48550/arXiv.2501.00425.

[16] P. Li and C. Mao Yang, “The analysis of transformer end-to-end model in Real-time interactive scene based on speech recognition technology,” *Sci. Rep.*, vol. 15, 17950, pp. 1–13, 2025, DOI. https://doi.org/10.1038/s41598-025-02904-0.

[17] HP. Arun *et al.*, “Malayalam speech to text conversion using deep learning,” *IOSR. J. Eng.*, vol. 11, no. 7, pp. 24–30, July 2021.

[18] S. Subbiah *et al.*, “Real time speech translation between Indian languages,” *2025 Int. Conf. Comput. Commun. Technol.*, pp. 1–5, 2025, DOI. https://doi.org/10.1109/ICCCT63501.2025.11019046.

[19] B. V. Kiranmayee *et al.*, “Machine translation on Dravidian languages,” I*nt. J. Recent Technol. Eng.*, vol. 12, no. 1, pp. 1–14, 2023.

[20] D. Pubadi *et al.*, “A focus on codemixing and codeswitching in Tamil speech to text,” 8th *Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, pp. 154–165, 2020, DOI. https://doi.org/10.1109/CONISOFT50191.2020.00031.

[21] J. D. Pradhan *et al.*, “Cascaded PFLANN model for intelligent health informatics in detection of respiratory diseases from speech using bio-inspired computation,” *J. Artif. Intell. Technol.*, vol. 4, pp. 124–131, 2024, DOI. https://doi.org/10.37965/jait.2024.0435.

[22] Zheshu Song *et al.*, LoRA-whisper: Parameter-efficient and extensible multilingual ASR, pp:1–5, 2024, DOI. https://doi.org/10.48550/arXiv.2406.06619.

[23] T. P. Ferraz *et al.*, “Multilingual distilWhisper: Efficient distillation of multi-task speech models via language-specific experts,” *Int. Conf. Acoust. Speech, Signal Process.*, pp. 10716–10720, 2023, DOI. https://doi.org/10.1109/ICASSP48485.2024.10447520.

[24] M. McGuire and J. Larson-Hall, “Assessing whisper automatic speech recognition and WER scoring for elicited imitation: Steps toward automation,” *Res Methods Appl. Linguist.*, vol. 4, no. 1, p. 100197, 2025, DOI. https://doi.org/10.1016/j.rmal.2025.100197.

[25] K. Tripathi, R. Gothi, and P. Wasnik, “Enhancing whisper’s accuracy and speed for indian languages through prompt-tuning and tokenization,” vol. 20, no. 1, pp. 1–5, 2024, DOI. https://doi.org/10.48550/arXiv.2412.19785.

[26] H. Yang *et al.*, “Language-aware prompt tuning for parameter-efficient seamless language expansion in multilingual ASR,” 26th edition of the Interspeech Conference, pp. 1133–1137, 2025, DOI. https://doi.org/10.48550/arXiv.2506.21577.