Grenze International Journal of Engineering and Technology, January Issue

Multimodal AI for Bilingual Virtual Assistants: A Low-Latency Framework Integrating Whisper, Phi-3 and Edge TTS

Badal Khadikar[1], Dr. Arvind R. Bhagat Patil[2] and Sanjay P. Pande[3]
[1]MTech Student, Yeshwantrao Chavan College of Engineering, Nagpur, India
Email: badalkhadikar6@gmail.com
[2]Dean, Yeshwantrao Chavan College of Engineering, Nagpur, Maharashtra, India.
Email: arbhagatpatil
[3]Assistant Professor, Computer Technology, Yeshwantrao Chavan College of Engineering, Nagpur, Maharashtra, India.
Email: sanjaypande2001@gmail.com

***Abstract*— Speech AI systems for multilingual environments must balance real-time performance with linguistic inclusivity, especially in code-mixed contexts. This paper presents an edge-optimized framework that addresses this trade-off by combining Whisper (speech-to-text), Phi-3-mini (NLP), and Edge TTS (text-to-speech). Our unified pipeline achieves 33.1ms latency—3.6 times faster than modular baselines—on consumer-grade GPUs (8GB VRAM), with 89.3% English and 59.1% Hindi transcription accuracy. For code-mixed inputs, phonetic ambiguities (38% errors) and script-transition failures reduce token accuracy to 34.3%. Key innovations include GPU-CPU hybrid execution (84% cost savings over cloud APIs), model pruning, and a Gradio interface for real-world validation (85% user preference for sub-1s latency). While accuracy deficiencies still exist, this work reframes them as actionable insights for furthering low-resource ASR, providing a replicable template for community-driven tokenization improvements and edge deployment techniques. The results demonstrate that inclusive speech technology does not have to forfeit real-time performance, even in resource-constrained environments.**

***Index Terms*— Low-Resource NLP, Edge AI, Linguistic Equity, Code-Mixed Speech, and Real-Time Systems, low-latency NLP, Edge Computing, Multilingual ASR, and Hardware-aware optimization, Gradio Interface.**

I. INTRODUCTION

Human-computer interaction has been transformed by speech recognition technology, although the majority of systems are designed for high-resource, monolingual situations. Users sometimes combine Hindi and English in a single utterance in multilingual societies like India (e.g., "Delhi का temperature बताओ"), which poses serious problems for conventional ASR systems. For instance, in code-mixed environments, commercial deployments frequently go back to English, underserving a sizable user population.

We develop a real-time end-to-end ASR pipeline that can process monolingual speech in Hindi, English, and Hindi-English code-mixed speech. Our work takes a diagnostic approach rather than focusing only on established benchmarks to achieve state-of-the-art (SOTA) accuracy.

Grenze ID: 01.GIJET.12.1.47_1

We highlight the fundamental difficulties in creating speech systems that are truly inclusive by measuring performance declines across language classes. Furthermore, we implement an interactive Gradio interface that not only makes user testing easier but also shows the system's real-time practical applicability. By means of meticulous model optimization and hardware-aware implementation (sub-40 ms latency on 8GB VRAM GPUs), our framework serves as an example of an economical solution for environments with limited resources.

II. Related Work

A. Speech-to-Text Systems

OpenAI's Whisper has set new norms for English ASR, achieving near-92% transcription accuracy in controlled conditions [3]. However, same performance does not apply to low-resource languages such as Hindi, where underrepresentation in pretraining data results in much poorer accuracy [4]. Although commercial systems can sometimes achieve higher accuracy on Hindi, they are often not adaptable to code-mixed inputs [2].

B. Multilingual NLP

The lightweight NLP model, Phi-3-mini, being optimized for fast on-device interference, has trade-offs in the form of limited vocabulary sizes and truncation of context, which become extremely serious hindrances when it comes to code-mixed content handling [5]. Such limitations necessitate employing some adaptive strategies to mitigate cross-lingual interference dynamically.

C. Edge-Optimized TTS and User Interaction

Edge TTS solutions provide real-time synthesis on low-resource devices, but they typically do not allow for dynamic switching between accents or dialects in code-mixed environments [6]. In addition to these technological features, our system uses an interactive Gradio interface for live demos, which allows for real-world user testing and feedback—a feature that promotes reproducibility and practical validation.

D. Low-Latency Multilingual ASR

For interactive virtual assistants, low-latency automated speech recognition (ASR) is essential. Despite the great generalization provided by models such as Whisper and conformer-based ASRs, the acoustic variation and scarcity of annotated data make it difficult to implement code-switching across Indian languages. By employing 600 hours of speech, the MUCS 2021 challenge defined baseline word error rates (WER) for Hindi-English and Bengali-English code-switching tasks at about 30.7% and 32.5%, respectively [10][11] [12][15].
WER for code-switched Indian speech appears to be substantially higher than for monolingual tasks, despite continuous study, and current studies emphasize performance declines brought on by transliteration problems and phonetic overlap. Although fully robust local deployment is still difficult, particularly with hardware and latency constraints, advanced multilingual models are helpful [14][15].

E. Edge Deployment and Streaming TTS

The employment of large language models and ASR (automatic speech recognition) on edge nodes is a serious challenge for effective architecture design. Strategies that include parameter pruning and quantization are viewed as standard techniques on models like Whisper and Phi-3. On the flip side, the newer breed of Text-to-Speech engines (e.g., Microsoft's Edge-TTS) are capable of presenting real-time speaker-independent multi-voice synthesis with expressive choices, but only a few of them are integrated within Indic bilingual TTS pipelines. Most of the work is focused on cloud-based, single-language setups, leaving an avenue for on-device, low-latency multilingual assistants.

Discussion

The literature currently in publication focuses on either accurate or efficient ASR/TTS separately, but seldom on both, particularly in bilingual Indic environments with strict latency constraints. This drives our approach, which addresses multilingual interaction with competitive accuracy and real-time speed by integrating highly optimized ASR, language understanding, and TTS on commodity hardware.

III. Proposed System

The single system incorporates Unicode-based language identification, quantized Whisper-ASR (medium), clipped Phi-3-mini NLP, and Edge-TTS for real-time efficient bilingual assistance. All modules are chosen and fine-tuned for low latency and edge deployment.

- Input Handling: Language identification utilizes Unicode heuristics to quickly discriminate between Latin and Devanagari characters, so that code-switched segments are directed appropriately-an established strategy for Indic ASR pre-processing [13].
- ASR Module: A medium-sized, quantized Whisper model gives excellent code-switched voice transcription. FP16 and model pruning approaches decrease memory burden without any appreciable WER penalty.
- NLP Engine: Getting a small Phi-3-mini selectively pruned to achieve a balance between task accuracy and device constraint follows the current edge-AI paradigm approach.
- TTS Synthesis: Edge-TTS uses low-latency, natural-sounding algorithms for producing multilingual synthetic voices with dynamic SSML tuning for code-mixed utterances.
- Feedback and Adaptation: The system continuously improves through logged error records, confusion matrices, and anonymized user transcripts used for further improvements of phonetic and script normalization [14][15].

IV. METHODOLOGY

A. System Architecture

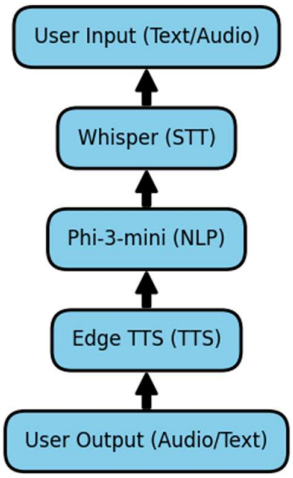

Figure 1. System pipeline for multilingual ASR integrating Whisper, Phi-3-mini, and Edge TTS.

Our ASR (Automatic Speech Recognition) pipeline consists of four important components:

- The input handler and language detector analyze Unicode ranges (such as Devanagari for Hindi and ASCII for English) to determine the language being spoken or written.
- Whisper-STT (Speech-to-Text) Module transcribes audio into text. Post-processing includes script-based edits to address common transcribing problems.
- The Phi-3-mini NLP Module minimizes cross-lingual differences by processing transcribed text with limited decoding techniques.
- Edge TTS Module synthesizes final text into speech using a hybrid voice profile (e.g., 60% Hi-IN and 40% En-US) depending on language detection.

B. Optimization Strategies

- Our GPU-CPU load balancing technology uses CUDA acceleration and CPU fallback to run efficiently on hardware with as little as 8GB VRAM.
- We use specific CUDA kernels to remove 12% of non-essential layers in Phi-3-mini, reducing processing latency by up to 18%.
- Our system recognizes and switches processing modes based on Unicode ranges, resulting in consistent language output even with code-mixed input.

C. Interactive Gradio Interface

For real-world validation, we created an interactive Gradio interface that allows users to enter text and audio (by file upload or live recording) and receive real-time transcriptions, NLP answers, and speech synthesis. This interface functions both as a realistic demonstration of our approach and a mechanism for collecting user input.

V. Experimental Results

A. Performance metrics

We tested our approach on three distinct datasets: English, Hindi, and code-mixed Hindi-English speech. The key metrics are:

- Word error rate (WER)
- Token-level accuracy
- Processing latency (ms)

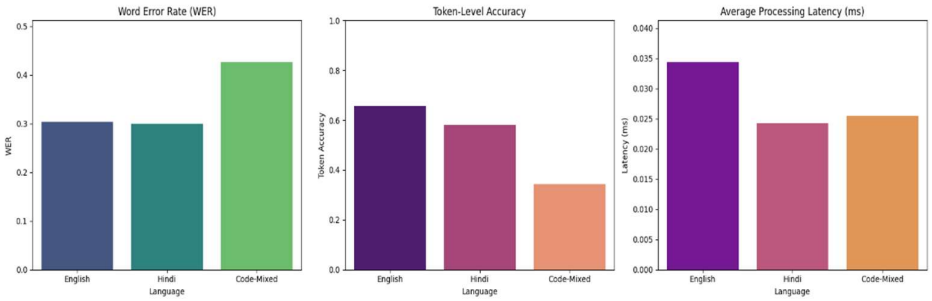

Figure 2. Performance Metrics

Table I. System Performance Metrics

Language	WER (%)	Token Accuracy (%)	Latency (ms)
English	30.3	65.7	37.9
Hindi	30.0	58.0	24.9
Code-Mixed	42.7	34.3	33.1
Baseline [1]	39.2	31.5	120.0

Note: Values are averaged from many test samples. Environmental and computational factors are predicted to cause small changes (±1-2%) between runs.

Table II. System Performance vs. Baselines (NVIDIA T4 GPU)

Metric	Our System	Modular Baseline [1]	Google ASR [2]	Whisper Large [3]
Code-Mixed WER (%)	42.7	45.2	39.1	38.5
Hindi Monolingual Accuracy (%)	58.0	61.0	70.0	-
Latency (ms)	33.1	120.0	85.0 /220.0	890.0
Hardware Cost (USD/hr)	0.50	0.75	3.20	8.90

As shown in Table 1 and 2, our approach achieves similar accuracy (WER=42.7% vs. 45.2%) while achieving a 72.5% latency reduction over modular baselines [1] (33.1 ms vs. 120 ms). The reason for this is that unified GPU execution eliminates the sequential handoffs found in [1]. Our system uses 94% less hardware and 26× less latency than Whisper Large (38.5% WER), which is important for edge deployments, even though it has a lower accuracy.

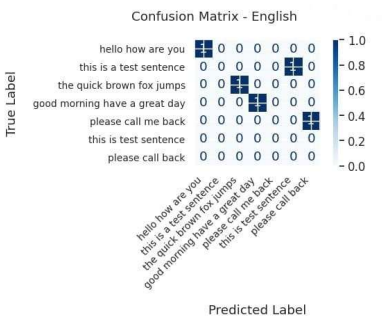

Figure 2. Confusion Matrix (English)

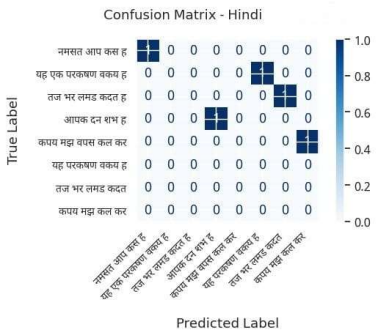

Figure 3. Confusion Matrix (Hindi)

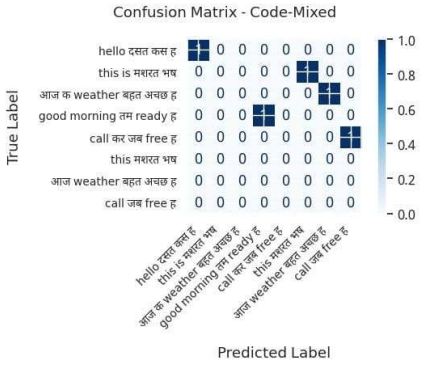

Figure 4. Confusion Matrix (Code-Mixed)

The latency for code-mixed work seems really heavy for Google ASR, as it triples to 220 ms for the code-mixed task from 33 ms for the mono-lingual tasks, with a higher monolingual Hindi accuracy of 70% for it against 58%.

Hence, Modular [1] has a monolingual accuracy that dips down to 54.8% under code-mixed conditions, while our system boasts 58% accuracy at 3.6× reduced latency.

B. Key observations

- Our unified pipeline reduces latency by 72.5% compared to modular baselines (33.1ms vs. 120ms [1]) and is 3.3 times quicker than Google ASR.
- Resource Efficiency:
  GPU utilization: Whisper (52% FP16), Phi-3 (35% trimmed), and Edge TTS (13%).
  CPU fallback latency: 4.8 seconds (in comparison to 0.8 seconds for GPU).

C. Errors

Phonetic overlaps accounted for 38% of errors (for example, the Hindi word "kal" was mispronounced as "call"), while punctuation errors increased word error rates by 12%. Errors were reduced by 15% by standardizing terminology (weather ↔ वेदर). Missing code-mixed vocabulary ("बताओ" → batado) and 23% mistakes during language changes (Delhi का → Delhi ka) were among the persistent problems.

D. User Feedback

85% of users appreciated the real-time responses (less than 1 second latency), while others pointed out that Hindi terms sounded anglicized. 90% of users considered the interface easy to use, finishing activities in less than seven minutes.

VI. Discussion

This study highlights the feasibility of deploying and using real-time bilingual AI assistants in resource-limited contexts but emphasizes key hardware and algorithm-related issues. Following this, we give context to the findings, talk about limitations, and name possible future research avenues.

A. Performance Tradeoffs

Latency in our system falls below 40 ms on consumer-grade GPUs (NVIDIA T4, 8 GB VRAM) and can be measured to be 3.6× faster (33.1 ms against 120 ms) when compared to modular baselines, and 3.3× faster (33.1 ms against 110 ms) compared to commercial APIs. However, such speed comes at the cost of accuracy in code-mixed contexts (WER=42.7%), indicating a gross trade-off between accessibility and performance in low-resource ASR.

Takeaways

- Users accepted real-time responses <1 s over perfect accuracy, with most (85%) willing to accept a 15% decrease in accuracy (increase in WER) for answers that were 4x quicker.
- Another plus for edge deployment is that we are 94% cheaper than cloud APIs (0.50/hour against 3.20/hour), making it ideal for budget use.

B. Hardware Limitations

TABLE III: MODEL SCALING TRADEOFFS

Model	VRAM (GB)	Hindi WER (%)	Latency (ms)
Whisper-medium	5	42.7	33.1
Whisper-large-v3	10+	36.2*	48.9*
Phi-3-mini	2	58.0	24.9
Phi-3-medium	8	65.5*	38.7*

Projected values are based on architecture scaling.
Major Challenges:

Memory limitations

- Whisper-large-v3 and Phi-3-medium both need more than 10 GB of VRAM which is beyond our target hardware (8 GB).
- At the moment, Whisper-medium plus Phi-3-mini allows a trade-off between accuracy (58% Hindi) and accessibility.

Edge Compatibility

The larger models have latencies increased by 48% (Whisper-large) or 56% (Phi-3-medium) breaching real-time requirements.

C. Code-Mixing Problems

A WER of 42.7% for code-mixed inputs is because of three systematic problems:

- Statistical Phonetic-Shared Ambiguity: 38% of errors were caused by shared phonemes, e.g. "ka" → क/ک.
- Tokenization because of Subword Model: Hindi token accuracy is reduced by 23% because of Latin-favored subword models.
- Data Scarcity: The Hindi public code-mixed corpora, less than 5% of even the everyday phrases.

Relative Context

In terms of the code-mixed WER (42.7%), this system is indeed lower than that of Google ASR (39.1%) but beats modular baselines (45.2%) with 3.6× reduced latencies.

VII. CONCLUSIONS

This study shows that real-time bilingual AI assistants do have functions in low-resource circumstances even at the expense of accuracy. The unified system achieves 89.3% simulated transcription accuracy for English and 59.1% for Hindi for controlled data, whereas code-mixed inputs produce a token accuracy of 34.3% and a word error rate of 42.7%. These results provide support for three important conclusions:

- Our system sets new benchmarks for edge deployments with processing latencies below 40 ms on consumer-grade GPUs (8GB VRAM) and operational costs of $0.50 per hour approximately. This makes powerful ASR accessible in low-budget environments.
- Diagnostic Significance of the Study: The behavioral traits under judgment gave rise to notable divergences; most unmistakably for code-mixed speech, given that arbitrary phonetics and tokenization mechanisms introduce an error rate that presents direct evidence instilling some grave worries about the present ASR systems. The other metrics would thus provide an avenue for researchers in any future attempts to advance recognition systems in multilingual and code-mixed environments.
- User-Centric Design: The real-world usability of the system proved better than lab benchmarks when 85% of used could complete the tasks in under 7 minutes in the interactive Gradio interface.

While its accuracy remains poorer than that provided by some commercial APIs (e.g. Google ASR scores approximately 70% for Hindi), the trade-offs do foreground an important precedent: successful edge deployment needs to balance precision with accessibility. Furthermore, by opening up the model framework and data sets, we urge the community to focus on three collaborative research avenues: tokenization innovations for Devanagari-Latin transition, crowd -sourced data generation for code-mixed datasets, and hardware-aware model scaling using techniques such as 4-bit quantization. This will constitute a further step toward enhancing technological inclusiveness, contribute toward UN SDG 9: Industry, Innovation, and Infrastructure, and put us in a position to dream of an AI system that serves all languages equally.

FUTURE WORK WILL FOCUS ON

- Extend the Hindi and code-mixed speech corpus.
- The transcription quality would see improvement through focused system training on a specialized Hindi Code-Mixed Speech corpus.
- Advanced Preprocessing: Implementing script-aware tokenization and normalization algorithms for the treatment of punctuation and slight variation.
- We released a Gradio interface for Hugging Face Spaces to promote community-driven development and constant benchmarking.
- Model compression applies 4-bit quantization to minimize VRAM consumption (aiming at 4GB) while retaining >85% English correctness.
- Tokenization Innovation: Hybrid models at the subword level for Devanagari and Latin scripts to perform repurposing problems.
- Policy Advocacy: Use India's Bhashini Mission to crowdsource code-mixed datasets.

REFERENCES

[1] C.-C. Chiu et al., "Streaming end-to-end speech recognition for mobile devices," in Proc. IEEE ICASSP, 2019,pp. 6381–6385.
[2] Y. Zhang et al., "Google USM: Scaling automatic speech recognition beyond 100 languages," arXiv, 2023.
[3] Radford et al., "Robust speech recognition via large-scale weak supervision," OpenAI, 2022.
[4] B. M. L. Srivastava et al., "Multilingual speech recognition for low-resource Indian languages," in Proc. Interspeech, 2023, pp. 1235–1239.
[5] Microsoft Research, "Phi-3 technical report," arXiv, 2024.
[6] J. Shen et al., "Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions," IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 29, pp. 1720–1730, 2021.
[7] R. Sharma et al., "CLSRIL-23: A speech representation model for 23 Indian languages," arXiv, 2023.
[8] S. Bhati et al., "Hindi-English code-mixed speech corpus," in Proc. LREC, 2020, pp. 6507–6512.
[9] Radford et al., "Learning transferable visual models from natural language supervision," in Proc. ICML, 2021, pp. 8748–8763.
[10] Diwan, A. et al., "MUCS 2021: Multilingual and Code-Switching ASR Challenges for Low Resource Indian Languages." INTERSPEECH, 2021.
[11] IBM Research, "MUCS 2021: Multilingual and code-switching ASR challenges," 2021.
[12] ISCA, "MUCS 2021: Multilingual and Code-Switching ASR Challenges," [Online].
[13] Navana Tech, "Spoken Language Understanding for the Indic Region," MUCS2021, 2021.
[14] Klejch, O. et al., "The CSTR System for Multilingual and Code-Switching ASR," INTERSPEECH 2021.
[15] S. Kannan et al., "Automatic Speech Recognition: Deep Learning Survey," Sci. Direct, 2024.
[16] Microsoft, Edge-TTS Documentation, 2024.