BREAKING LANGUAGE BARRIERS: FINE-TUNING WHISPER FOR BENGALI AND TELUGU AUTOMATIC SPEECH RECOGNITION

ISHMITA BASU, IMON KALYAN GHOSH, and BATHULA VEERA RAGHAVULU

Automatic Speech Recognition (ASR) has become an essential component of modern human-computer interaction, enabling systems to convert spoken language into text with remarkable accuracy. However, these advances have been largely limited to high-resource languages such as English, Mandarin, and Spanish. Low-resource languages like Bengali and Telugu remain significantly under-represented in existing ASR systems due to the scarcity of large, annotated, and high-quality datasets. This disparity limits access to speech-enabled technologies for millions of native speakers and presents a critical challenge in democratisation of artificial intelligence.

In this research, we address the performance gap in ASR systems for low-resource languages by fine-tuning OpenAI's Whisper model using parameter-efficient fine-tuning (PEFT) techniques. We focus on the Whisper Small Model and implement four PEFT methods: Low-Rank Adaptation (LoRA), Adapter Layers, SpecAugment, and BitFit in two distinct language datasets: Bengali and Telugu. Each method was evaluated for its effectiveness in reducing Word Error Rate (WER), offering insights into their relative performance and computational efficiency. The results show substantial improvements in WER over the zero-shot baseline, with each technique contributing differently based on language characteristics and model configuration.

Beyond the experimental evaluation, we developed a modular and extensible training pipeline that allows users to fine-tune Whisper on their own datasets by simply providing a dataset link. This framework streamlines the training process for low-resource languages and opens up possibilities for community-led ASR improvements across diverse linguistic landscapes. By combining lightweight fine-tuning methods with reproducible tooling, our work contributes toward building equitable speech recognition systems that break language barriers and foster inclusion in voice technology.

ACM Reference Format:
Ishmita Basu, Imon Kalyan Ghosh, and Bathula veera Raghavulu. 2025. BREAKING LANGUAGE BARRIERS: FINE-TUNING WHISPER FOR BENGALI AND TELUGU AUTOMATIC SPEECH RECOGNITION. 1, 1 (April 2025), 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Over the past decade, ASR has revolutionised human-computer interaction, but its advances are largely confined to high-resource languages, leaving low-resource languages like Bengali and Telugu behind due to limited annotated data and resources. OpenAI's Whisper, a transformer-based ASR model trained on extensive multilingual audio, shows strong generalization but still struggles with under-represented languages because of insufficient domain-specific data.

To address this, our research investigates PEFT methods—specifically LoRA, Adapter Layers, Prefix Tuning, and BitFit—to adapt Whisper for Bengali and Telugu ASR. These approaches reduce the number of trainable parameters, making fine-tuning feasible even with limited computational resources. Using publicly available datasets, we trained the Whisper small model with each PEFT method and evaluated improvements using Word

Authors' Contact Information: Ishmita Basu; Imon Kalyan Ghosh; Bathula veera Raghavulu.

ACM XXXX-XXXX/2025/4-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Error Rate (WER). Results show notable accuracy gains, confirming the effectiveness of PEFT in low-resource settings.

Additionally, we developed a user-friendly training pipeline that streamlines data preprocessing, model configuration, training, and evaluation, lowering technical barriers for ASR research in low-resource languages. This paper provides a comparative analysis of PEFT methods for ASR in Bengali and Telugu and advocates for scalable, accessible speech technology to support linguistic diversity

2 Literature Review

The development of ASR has significantly progressed over the last two decades, becoming a vital component of many AI-powered systems. ASR technology aims to convert spoken language into written text using computational algorithms. It has found widespread application in areas such as virtual assistants, transcription tools, and voice-based user interfaces [7][4]. Early ASR systems were predominantly hybrid models, combining acoustic, pronunciation, and language models. However, recent advancements have led to the adoption of End-to-End (E2E) architectures due to their efficiency and improved accuracy [8].

Three prominent E2E approaches include Connectionist Temporal Classification (CTC), Attention-based Encoder-Decoder (AED), and Recurrent Neural Network Transducer (RNN-T). These methods eliminate the need for complex pre-processing pipelines and are especially beneficial in multilingual and multitask environments. One of the most prominent E2E ASR systems is Whisper, developed by OpenAI. It is trained on 680,000 hours of multilingual and multitask speech data and is capable of performing speech recognition, language identification, and translation [10].

Despite its robustness, Whisper's performance on low-resource languages—languages with limited digital and annotated data—remains suboptimal. The Javanese language, for instance, falls into this category and exhibits high WER when processed in a zero-shot setting [9] [11]. This discrepancy underlines the necessity of adapting ASR models through fine-tuning to specific linguistic and acoustic characteristics of low-resource languages.

Fine-tuning has emerged as a powerful technique to improve model performance on specialised tasks by retraining a pre-trained model on task-specific data. This approach has proven especially effective when combined with PEFT techniques such as LoRA, which allow models to be updated with minimal computational resources [6][2]. LoRA achieves this by introducing trainable low-rank matrices into the model's architecture, drastically reducing the number of trainable parameters while maintaining performance. Recent studies have demonstrated that fine-tuning Whisper on low-resource languages, such as Javanese, can substantially reduce WER. For instance, [9] showed that fine-tuning the Whisper Large-V2 model on a high-quality Javanese dataset reduced WER from 89.40% to 13.77%. Their implementation of PEFT using LoRA achieved significant model compression, training with less than 1% of the total parameters while maintaining accuracy.

Moreover, preprocessing steps such as audio resampling to 16kHz, tokenisation, and log-mel spectrogram extraction have proven essential in optimising ASR performance [10] [12]. The Whisper tokeniser, which supports over 90 languages including Javanese, helps ensure that the model effectively captures the lexical structure of under-represented languages. Overall, the literature underscores the effectiveness of combining pre-trained large-scale ASR models with efficient fine-tuning techniques to extend speech recognition capabilities to low-resource languages. This approach not only promises improved accessibility but also paves the way for inclusive AI systems capable of serving linguistically diverse populations.

3 Dataset Preparation

Preprocessing is a fundamental component in any speech recognition pipeline, especially when fine-tuning large-scale models such as Whisper for low-resource languages such as Bengali and Telugu. Our preprocessing methodology focused on ensuring the quality, balance, and manageability of training and evaluation data, all while simulating a realistic low-resource setting. The key components of this process are described below.

3.1 Original Dataset Description

We employed publicly available ASR datasets for Bengali and Telugu from Hugging Face to fine-tune Whisper. The Bengali dataset, "imonghose/bengali-asr-data," and a structurally similar Telugu dataset consisted of 16 kHz audio recordings paired with text transcripts, reflecting a range of speaker accents and acoustic conditions. The inclusion of both clean and noisy samples provided the variability necessary for robust model adaptation.

3.2 Subsampling for Low-Resource Simulation

To replicate low-resource settings and optimise computational efficiency, we selected 9% of the training set and 40% of the test set for both languages. Datasets were shuffled with a fixed random seed prior to subsampling to ensure balanced and unbiased data distribution, preserving linguistic and acoustic diversity within the reduced corpus.

3.3 Custom Dataset Creation: Deletion and Filtering

Additional refinement involved discarding samples with missing or malformed transcripts and filtering out audio clips with token length greater than 448. Texts were normalised through lowercasing, special character removal, and whitespace trimming, while audio files were validated for consistent 16 kHz sampling. These steps improved data quality, minimised noise, and ensured a uniform, reliable dataset for fine-tuning. After subsampling and filtering, the data was organised into a consistent and modular format using Hugging Face's DatasetDict, with "train" and "test" splits.

3.4 Language and Task Configuration

Each language was explicitly configured in the model setup. The language parameter was set to either "Bengali" or "Telugu", and the ASR task was defined as "transcribe". This ensures that the Whisper tokeniser and decoder are optimised for the target language during fine-tuning and inference.

Following preprocessing, the dataset was inspected to confirm the correct number of samples per split, the presence of audio and transcript fields, and overall sample balance. This validation was crucial to prevent run-time errors and to ensure high-quality inputs for fine-tuning. However, the challenges inherent to low-resource ASR datasets remained, including inconsistent audio quality, speaker diversity, and sample imbalance. Addressing these issues is vital for building robust and accurate models.

4 Methodology

4.1 Data pre-processing

Each entry in the initial dataset included the raw audio waveform, the corresponding transcription (sentence), and the length of the audio clip. A custom preprocessing function was applied to each data point. This function first resampled the audio to 16 kHz (if not already), then extracted log-Mel spectrogram features using Whisper's feature extractor. The associated transcription was tokenised into label IDs using a language-specific pre-trained tokeniser, where the target language was manually specified by the user. After preprocessing, the dataset was transformed to contain only the input_features and their corresponding labels. In addition, a custom data collator was employed during training to separately pad audio inputs and tokenised labels, replace padding in labels with -100 for proper loss masking, and remove redundant beginning-of-sequence tokens. This ensured seamless integration with the Whisper model's sequence-to-sequence architecture and consistent training behaviour, making it suitable for training with the Whisper model architecture.

4.2 Fine-tuning approaches employed

- **Low-Rank Adaptation (LoRA)** is a parameter-efficient fine-tuning method introduced by [6] to reduce the number of trainable parameters required when adapting large pre-trained models to downstream tasks. The core idea behind LoRA is to decompose the weight update matrix into the product of two lower-rank matrices, thereby introducing trainable rank-constrained adapters into the model while freezing the original weights. Formally, for a weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d\times k}$, LoRA reparameterises it during training as:

  $$\mathbf{W} = \mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + BA$$

  where $\mathbf{A} \in \mathbb{R}^{r\times k}, \quad \mathbf{B} \in \mathbb{R}^{d\times r}$ and $r \ll \min(d,k)$ is the rank of the adaptation. By training only A and B, LoRA drastically reduces the number of trainable parameters without sacrificing performance. This approach is particularly useful in low-resource and multilingual scenarios, such as fine-tuning Whisper for under-represented languages, as it enables efficient adaptation while maintaining the benefits of large-scale pretraining.
- **BitFit** is a minimalistic fine-tuning approach proposed by [3] that restricts training to only the bias terms of a pre-trained model while keeping all other weights frozen. Despite its simplicity, BitFit has been shown to achieve competitive performance on various downstream tasks. The rationale is based on the observation that bias terms can shift the activation distributions sufficiently to adapt pre-trained representations to new tasks. Mathematically, if a layer's output is represented as

  $$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

  BitFit fine-tunes only the bias b, while keeping the weight matrix W fixed. This results in a drastic reduction in the number of trainable parameters (often less than 0.1% of the total model size), making it highly appealing for resource-constrained settings. In the context of Whisper and low-resource language ASR, BitFit allows efficient adaptation without the risk of overfitting, particularly when training data is limited.
- **Adapter Layers** are lightweight neural modules inserted within the layers of a pre-trained model, allowing task-specific fine-tuning while keeping the original model weights largely frozen. This approach was introduced by [5] to enable efficient transfer learning for large-scale transformers. The core idea is to add a small bottleneck architecture to each transformer layer: a down-projection from the hidden size ddd to a lower dimension r, followed by a non-linear activation and an up-projection back to d. Formally, given a hidden state $\mathbf{h} \in \mathbb{R}^d$, an adapter computes:

  $$\text{Adapter}(h) = h + W_{\text{up}}\, f\big(W_{\text{down}}\, h\big)$$

  $where \quad W_{\text{down}} \in \mathbb{R}^{r\times d},\ W_{\text{up}} \in \mathbb{R}^{d\times r}$ and f is typically a ReLU activation. Only these adapter parameters are updated during fine-tuning. This significantly reduces computational cost and memory footprint while enabling the model to learn task-specific representations. In the context of fine-tuning Whisper for low-resource ASR, adapter layers offer a flexible and scalable alternative to full-model updates, promoting better generalisation and faster convergence.
- **SpecAugment** is a data augmentation technique introduced by [1] that operates directly on log-Mel spectrograms, applying time warping, frequency masking, and time masking to simulate variations in speech patterns without altering linguistic content. Its primary goal is to improve model generalisation by enhancing robustness to distortions and unseen audio conditions. In fine-tuning, particularly for low-resource languages, SpecAugment mitigates overfitting by artificially expanding training data variability, which is crucial for large, over-parameterised models like Whisper. Although no prior work has directly

combined LoRA with SpecAugment, there is theoretical motivation to do so. LoRA enables parameter-efficient adaptation by injecting low-rank trainable matrices into frozen models but does not inherently address data scarcity. SpecAugment complements LoRA by acting as a regularisation mechanism at the input level, potentially improving the stability and generalisation of LoRA updates, particularly in noisy or diverse linguistic settings typical of low-resource ASR tasks.

4.3 Hyperparameter selection

Table 1. Training Hyperparameters and Their Significance

Hyperparameter	**Value**	**Significance**
Train batch size	2	Small due to GPU constraints
Eval batch size	2	Matches training batch size
Gradient accumulation steps	2	Simulates effective batch size of 4
Learning rate	1e−5	Medium setting, stable training
Warmup steps	50	Helps stabilize early training
Epochs	2	Chosen due to small dataset size
Evaluation strategy	Steps (every 142)	Fine-grained tracking
Save strategy	Steps (every 284)	Save checkpoints regularly
Optimizer	AdamW (Torch)	With weight decay
Weight decay	0.01	Regularization
FP16	True	Enables mixed-precision training
Max grad norm	1.0	Gradient clipping
Metric for best model	Eval loss	For model selection
Logging	TensorBoard	For monitoring

To ensure efficient and stable training on a small-scale dataset, we carefully selected training hyperparameters. The batch size was set to 2 per device, with gradient accumulation over two steps to simulate a larger effective batch size within memory constraints. A learning rate of 1×10^{-5} was chosen to balance convergence speed and stability, which is critical for fine-tuning large models like Whisper with PEFT methods. Training was conducted over two epochs, with 50 warmup steps to stabilise early dynamics. Evaluation and checkpointing were performed every 142 and 284 steps, respectively, providing regular feedback without excessive overhead. Mixed-precision training (fp16=True) was enabled to optimise memory and computational efficiency. The AdamW optimiser with weight decay (0.01) and epsilon 1×10^{-8} was used for regularisation and numerical stability, with gradient clipping set at a maximum norm of 1.0. TensorBoard logging was employed for real-time monitoring. Hyperparameters were kept consistent across all methods to ensure fair comparison and reproducibility.

4.4 Implementation

- **LoRA** : We fine-tuned the Whisper model using LoRA via the PEFT library to enable parameter-efficient training. A Lora configuration was defined with rank r=32, scaling factor $\alpha = 64$, and a dropout rate of 0.05 to regularise updates. Adaptation was limited to the attention projection layers (q_proj, k_proj, v_proj, out_proj), which are critical to the model's representational capacity. The bias terms were left frozen, further minimising the number of trainable parameters. The LoRA modules were injected accordingly, after which we confirmed that only the injected parameters were trainable. This setup preserved the core pre-trained weights while allowing efficient adaptation to the downstream ASR task, consistent with best practices in low-resource fine-tuning.
- **BitFit** : For the BitFit fine-tuning setup, we adopted the approach proposed by [3], which restricts training to only the bias parameters of the model. We first froze all model parameters to prevent gradient updates. Then,

we selectively re-enabled gradient computation for parameters whose names included .bias, effectively limiting fine-tuning to bias terms across the model. This minimalist strategy significantly reduces the number of trainable parameters—often to less than 0.1% of the total—while still allowing the model to adapt its internal representations. Such a setup is especially advantageous in low-resource scenarios, as it mitigates overfitting and enables efficient training with limited data.

- **Adapter Layers** : To enable PEFT, we extended the base Whisper model by injecting adapter layers into each encoder and decoder block. Each adapter consisted of a down-projection to 64 dimensions, followed by a GELU activation and an up-projection back to the original hidden size, with residual connections ensuring minimal disruption to the original representations. Parameters were initialised using Xavier uniform initialisation. Adapters were integrated by patching the forward methods of each transformer layer to apply the adapter transformation after the original computation. All base model parameters were frozen, allowing training only within the adapter modules. We verified that only adapter parameters were trainable, ensuring efficiency and modularity. This approach aligns with the work of [5], showing that adapters allow task-specific tuning with minimal overhead.
- **LoRA with SpecAugment** : To improve generalisation in low-resource fine-tuning, we combined SpecAugment with LoRA. SpecAugment was applied during training data preprocessing by masking random time and frequency regions in the log-Mel spectrograms (up to two masks each, with widths of 20 and 10, respectively). This was implemented within the dataset mapping function and applied only to training samples to avoid distorting evaluation inputs. The LoRA adaptation was implemented following the same procedure described previously, applying low-rank updates to the attention projection layers of the Whisper model. This setup combines input-level augmentation with PEFT, potentially enhancing robustness without increasing model complexity.

4.5 Low resource ASR fine-tuning pipelining using LoRA

To streamline and standardise the fine-tuning process, we developed a command-line interface (CLI)-based pipeline specifically tailored for LoRA-based adaptation of Whisper. This pipeline automates the entire workflow—from dataset sampling and preprocessing to model fine-tuning and evaluation—via a single command. Users can configure key parameters such as dataset path, target language, model size, training/testing split ratios, and Hugging Face credentials. Upon execution, the pipeline evaluates the baseline word error rate (WER) of the pre-trained Whisper model on the specified test subset. It then applies LoRA configuration, performs parameter-efficient fine-tuning, and re-evaluates the model to report the post-training WER. The implementation also includes

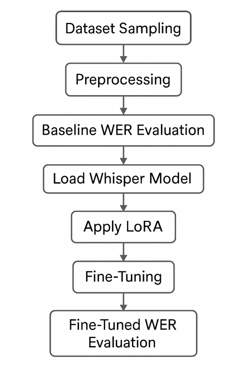

Fig. 1. End-to-end pipeline for LoRA-based fine-tuning of the Whisper model.

mixed-precision training, periodic checkpointing, and TensorBoard logging for monitoring. Post-training, the fine-tuned model and processor are pushed to the user's Hugging Face Hub for easy sharing and reuse. The modularity of the pipeline allows users to quickly adapt it to other languages or datasets by changing CLI arguments, offering a reproducible and extensible framework for low-resource ASR research.

5 Results and Analysis

Language	Method	WER (Before → After)	Training Loss (Start → End)	Evaluation Loss (Start → End)
Bengali	LoRA	192.836 → 96.533	2.779 → 1.003	1.673 → 0.867
Bengali	SpecAugment + LoRA	192.836 → 95.629	3.494 → 1.022	1.696 → 0.877
Bengali	BitFit	192.836 → 162.960	3.217 → 1.891	2.052 → 1.625
Bengali	Adapter Layers	192.836 → 100.000	11.178 → 2.186	4.179 → 1.929
Telugu	LoRA	196.254 → 96.581	1.652 → 0.837	1.611 → 0.871
Telugu	SpecAugment + LoRA	196.254 → 96.581	1.083 → 0.851	1.060 → 0.869
Telugu	BitFit	196.254 → 192.330	1.922 → 1.841	1.937 → 1.854
Telugu	Adapter Layers	196.254 → 100.000	7.535 → 2.009	6.159 → 1.987

Table 2. Comparison of WER and loss metrics for different fine-tuning methods on Bengali and Telugu.

- **LoRA** : LoRA consistently delivered the best results across both languages. For Bengali, it reduced the WER from 192.836 to 96.533, while for Telugu, it brought it down from 196.254 to 96.581. This indicates strong generalization and adaptability of the LoRA method across two typologically different low-resource languages.
- **SpecAugment + LoRA** : Adding SpecAugment to LoRA slightly enhanced performance for Bengali, with the WER improving marginally from 192.836 to 95.629—a minor yet consistent gain over using LoRA alone. However, on the Telugu dataset, SpecAugment did not show any noticeable effect, yielding the same final WER (96.581) as the plain LoRA setup. This suggests that the benefit of augmentation may be language- or data-specific.
- **BitFit** : BitFit offered modest improvements for both languages but fell significantly short of LoRA's performance. The WER for Bengali decreased to 162.960, and for Telugu to 192.330, indicating that while BitFit does help with adaptation to some extent, it lacks the capacity to achieve competitive performance in low-resource ASR scenarios.
- **Adapter Layers** : Adapter Layers showed better WER reductions than BitFit in both languages—down to 100.0 for both Bengali and Telugu. However, despite this quantitative improvement over BitFit, the overall performance remained suboptimal. The reasons for this, including training instability and potentially poorinitialisationn, are discussed further in the comparative analysis section.

5.1 Comparative Analysis (Bengali)

5.1.1 WER Progression Across Fine-Tuning Methods : This subsection examines the trajectory of the Word Error Rate (WER) during fine-tuning for each of the four parameter-efficient methods: LoRA, SpecAugment + LoRA, BitFit, and Adapter Layers. The analysis focuses primarily on the Bengali dataset, where detailed WER evaluations were recorded, and draws inferences based on comparative patterns and dataset characteristics.

- **LoRA** : The LoRA method exhibited the most favorable WER trajectory among all configurations. As illustrated in Figure 2 Quadrant 1, WER experienced a sharp decline from an initial value of approximately 97.3% to around 96.4% within the first 75 training batches, followed by a prolonged phase of stability. This rapid convergence and subsequent plateau suggest that LoRA effectively leveraged low-rank adaptation matrices to optimize performance in the early stages of training. The absence of significant oscillations further underscores the method's stability and reliability in fine-tuning large-scale models for low-resource ASR tasks.

- **SpecAugment + LoRA** : Integrating SpecAugment with LoRA produced a modest yet consistent improvement in WER, reducing it further to approximately 95.6% (Figure 2 Quadrant 2). The WER trajectory followed a pattern similar to that of LoRA, characterised by an initial rapid decline and later stabilisation. The incremental improvement is particularly notable given the larger training dataset for Bengali (approximately 8,000 samples), which likely enabled the model to benefit from the additional data diversity introduced through augmentation. Conversely, this benefit was not replicated for Telugu, where the training set was considerably smaller (approximately 4,000 samples), suggesting that the efficacy of data augmentation is sensitive to dataset size and variability.
- **BitFit** : BitFit resulted in a measurable but relatively modest reduction in WER, from approximately 172% to 162% over the training period (Figure 2 Quadrant 3). The observed curve suggests some initial learning, followed by a flattening trend that indicates limited further adaptation. This behaviour is consistent with the method's design: BitFit modifies only the bias terms in the network, dramatically reducing the number of trainable parameters. While efficient, this strategy appears insufficient for substantial performance gains in complex ASR tasks, particularly when adapting to phonetically rich and morphologically diverse languages like Bengali.
- **Adapter Layers** : The Adapter Layers configuration failed to produce any observable improvement in WER, which remained constant at 100% throughout training (Figure 2 Quadrant 4). Despite a theoretically sound design—injecting lightweight bottleneck adapters into both encoder and decoder layers of the Whisper model—the approach underperformed significantly in practice. Multiple factors could account for this stagnation: suboptimal initialisation of adapter parameters (even though Xavier uniform was used), overly restrictive adapter dimensionality (adapter_dim = 64), or ineffective gradient flow through the bottlenecked paths. Given that all other model parameters were frozen, the learning burden rested solely on these adapters, which may have lacked sufficient expressiveness or capacity to drive meaningful adaptation.

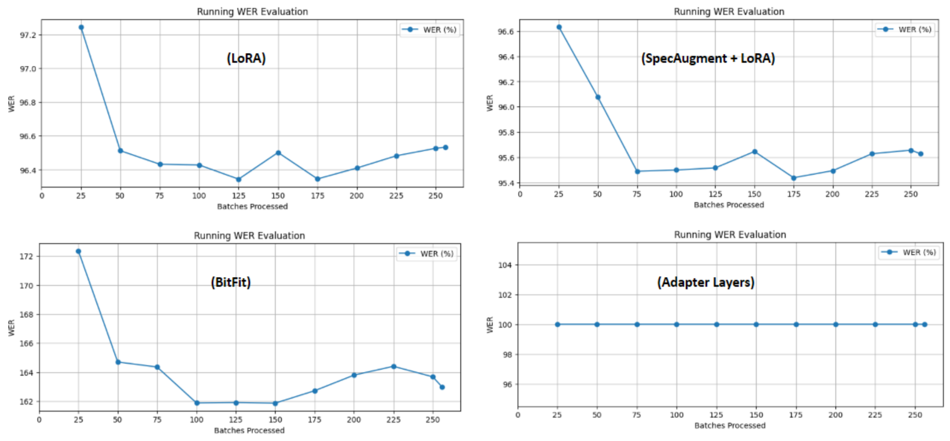

Fig. 2. WER Loss

5.1.2 Training and Evaluation Loss Analysis : In this section, we analyse the convergence characteristics and generalization behavior of each fine-tuning method through their respective training and evaluation loss curves.

These curves reveal not only optimization effectiveness but also offer insights into overfitting, learning stability, and the representational capacity of each parameter-efficient approach.

- **LoRA** : The LoRA configuration demonstrates a classic well-behaved training trajectory. As shown in Figure 3 Quadrant 1, the training loss sharply declines from 2.7 to 0.65, while the evaluation loss follows a similar but slightly more gradual decline from 1.7 to 0.87. The parallel downward trends indicate that the model is generally effective without overfitting. This pattern affirms LoRA's ability to fine-tune large models in a stable and data-efficient manner, with both in-distribution (training) and out-of-distribution (validation) performance improving steadily over time.
- **SpecAugment + LoRA** : The addition of SpecAugment (Figure 3 Quadrant 2) results in a training curve that begins with a higher initial loss (3.5) but rapidly converges in tandem with the evaluation loss. Both losses decrease smoothly, with training loss reaching 0.6 and evaluation loss converging just below 0.9. The higher initial training loss is expected due to the regularization effect of augmentation, but the eventual convergence indicates enhanced robustness. Compared to standard LoRA, the augmented model converges slightly slower but ends with comparable if not slightly better generalization, especially visible in the smoother and slightly lower evaluation loss curve.
- **BitFit**: The BitFit loss curves (Figure 3 Quadrant 3) highlight a suboptimal convergence profile. Although both training and evaluation losses decrease consistently (from 3.2 and 2.0, respectively), the final values plateau at relatively high points (1.6 training and 1.65 evaluation). The relatively narrow gap between the two curves suggests the model is not overfitting, but the elevated final losses indicate limited representational power due to the restricted training of only bias parameters. This behavior matches the WER trends, confirming BitFit's trade-off: low complexity but also low efficacy in expressive fine-tuning.
- **Adapter Layers**: The most intriguing discrepancy arises in the Adapter Layers configuration (Figure 3 Quadrant 4). Despite the WER remaining completely flat at 100%, the training and evaluation loss curves display a textbook convergence trend, with the training loss dropping from 11.2 to 1.8 and evaluation loss from 6.7 to 2.0. This divergence between cross-entropy loss and WER points to a decoupling between model confidence and correctness of predictions. Several factors might explain this:
  - Loss Function vs. Metric Mismatch: Cross-entropy optimizes for likelihood, not WER. If the model confidently predicts consistently incorrect tokens (e.g., always decoding into the same wrong pattern), loss can improve while WER stagnates.
  - Bottleneck Overconstraint: With adapters limited to a 64-dimensional space and only they being trainable, the model might have learnt to "speak confidently in the wrong language", so to speak—optimizing fluency without correcting transcription accuracy.
  - Poor Initialisation or Activation Path: If adapters were poorly initialised or failed to inject meaningful transformations early in training, the model might have converged to suboptimal yet consistent output behaviours.

  This contrast underscores the need to complement loss analysis with direct output evaluation (e.g., decoding and manual inspection), especially when working with residual adapters in transformer-based ASR systems.

5.1.3 Parameter Efficiency across Fine-Tuning Methods: To evaluate the parameter efficiency of each fine-tuning method, we computed the ratio of trainable to total model parameters. As shown in Table 3, LoRA and SpecAugment + LoRA fine-tuned approximately 2.84% of the Whisper-small model, striking a balance between efficiency and flexibility. BitFit, the most lightweight method, updated just 0.09% of parameters, while Adapter Layers allowed training of around 0.98% through inserted bottleneck modules. This comparison illustrates the varying degrees of parameter reduction achieved, with LoRA offering strong performance at modest computational cost and BitFit trading off expressive power for extreme efficiency.

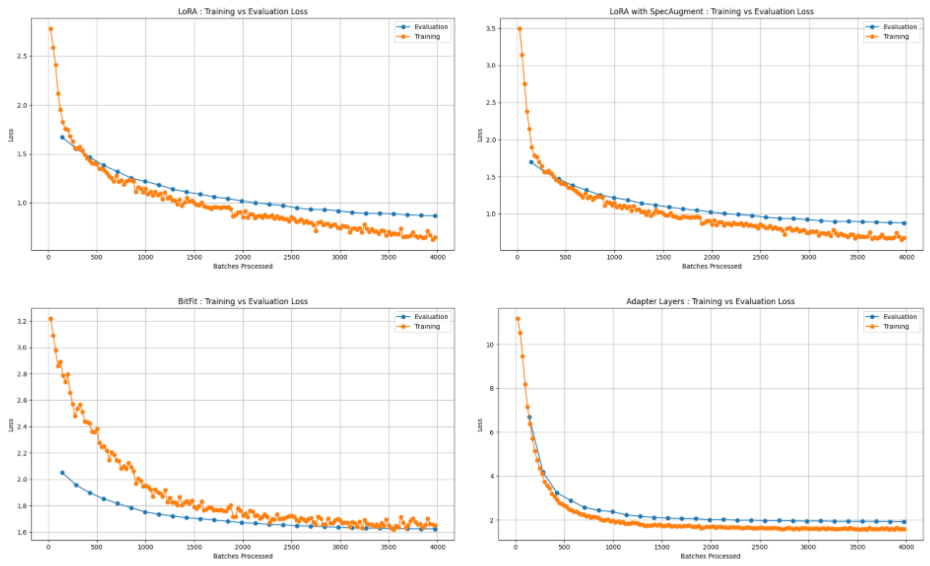

Fig. 3. Training vs Evaluation Loss

Method	Trainable Parameters	Total Parameters	Trainable Ratio %
LoRA	7,077,888	248,812,800	2.84%
SpecAugment + LoRA	7,077,888	248,812,800	2.84%
BitFit	224,256	241,734,912	0.09%
Adapter Layers	2,379,264	244,114,176	0.98%

Table 3. Trainable and total parameters with corresponding trainable ratios for each fine-tuning method.

5.1.4 Runtime and Throughput Analysis: We assessed the computational efficiency of each fine-tuning strategy using two key metrics: evaluation runtime (Figure 4) and evaluation throughput in samples per second (Figure 11). These metrics provide insight into the practical cost of deploying models fine-tuned via different parameter-efficient methods.

- **Evaluation Runtime (Figure 4)** : LoRA consistently achieved the lowest evaluation runtimes across training, with values stabilising around 220–230 seconds, demonstrating its computational efficiency despite moderate parameter updates. BitFit and Adapter Layers exhibited moderately higher runtimes in the 320–460 second range, with Adapter Layers showing more fluctuation but generally stable performance. SpecAugment + LoRA introduced considerable instability in runtime, with periodic spikes exceeding 800 seconds. These spikes likely correspond to evaluation passes affected by data augmentation overhead, which adds variability and computational burden during decoding. Despite this, when augmentation is not applied, SpecAugment + LoRA reverts to LoRA-level efficiency, highlighting that the runtime penalty is not from LoRA itself but from the dynamic data transformations.
- **Evaluation Throughput (Figure 4)**: Throughput trends closely mirrored runtime observations. LoRA maintained a high and stable throughput of approximately 9.0–9.2 samples/sec, indicating efficient batched processing. SpecAugment + LoRA achieved similar peak throughput but with substantial drops during

augmentation-heavy steps, falling below 3 samples/sec in some evaluations. This fluctuation reinforces the runtime instability seen in Figure 10. BitFit and Adapter Layers sustained lower throughput values, ranging between 4.5 and 6.5 samples/sec, reflecting marginally heavier decoder-side operations or less efficient memory access patterns, possibly stemming from internal tensor updates associated with the adapter and bias modifications.

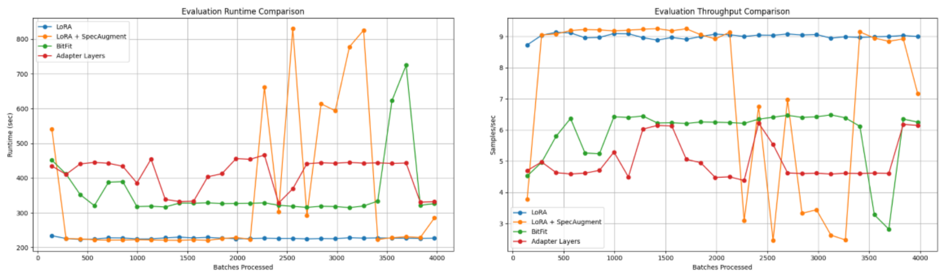

Fig. 4. Evaluation Runtime & Throughput

5.1.5 Gradient Norm and Training Stability Analysis: To assess training dynamics and optimisation health, we examined the gradient norm evolution throughout fine-tuning for all methods (Figure 4). Gradient norm reflects the magnitude of parameter updates and serves as an indicator of training activity, stability, and potential numerical issues.

- **LoRA and SpecAugment + LoRA** : Both LoRA and its SpecAugment-enhanced variant exhibited stable and healthy gradient behavior, with gradient norms typically ranging between 4 and 10. This range indicates consistent training signal flow without vanishing or exploding gradients. SpecAugment introduced slightly more fluctuation, which is expected due to augmented data variation. Nevertheless, the training remained well-regulated throughout, affirming the robustness of LoRA even under stochastic augmentation.
- **BitFit**: BitFit displayed consistently low gradient norms, hovering between 1 and 3. While this suggests high numerical stability, it also reflects limited gradient flow, which is consistent with its minimalist architecture that updates only bias terms. The low gradient magnitude reinforces the interpretation that BitFit learns slowly and conservatively, leading to modest performance gains despite stable training.
- **Adapter Layers**: Adapter Layers demonstrated high and erratic gradient norms, frequently spiking above 30 and reaching values beyond 50. This volatility suggests potential instability in optimisation, possibly due to poor initialisation, bottleneck constraints, or lack of gradient smoothing mechanisms. Although the loss curves showed apparent convergence, the chaotic gradient behavior may explain the model's failure to achieve meaningful WER improvements—suggesting that while loss decreased, the learning trajectory may have been unstable or misdirected.

6 Conclusion

This study evaluated four parameter-efficient fine-tuning methods—LoRA, SpecAugment + LoRA, BitFit, and Adapter Layers—on the Whisper-small model for low-resource ASR in Bengali and Telugu. Our results demonstrate that LoRA consistently achieved the best trade-off between WER reduction, training stability, and computational efficiency, fine-tuning under 3% of model parameters while maintaining low runtime and high throughput. SpecAugment provided marginal WER improvements for Bengali, though its benefits were dataset-size dependent and came with additional evaluation overhead. BitFit, while highly efficient, underperformed due to limited

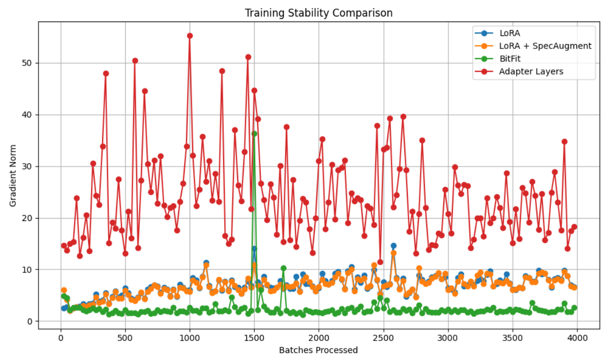

Fig. 5. Training Stability

update capacity, and Adapter Layers failed to improve WER despite seemingly converging loss curves, likely due to unstable training dynamics and poor optimisation flow. Overall, LoRA emerges as the most reliable and scalable approach for parameter-efficient fine-tuning in low-resource ASR, combining strong empirical performance with manageable computational cost.

7 Future Scope

The fine-tuning of the Whisper model for Bengali and Telugu using LoRA, has demonstrated promising results in extending multilingual ASR capabilities to low-resource Indian languages. Our current implementation focused on core ASR tasks—data preprocessing, feature extraction, and model adaptation—but there remain significant opportunities to extend this work further. Below, we outline key areas for future development

7.1 Speech Diarization Integration

Future work includes integrating speaker diarisation to attribute speech to individual speakers, which is crucial for multi-speaker scenarios like lectures or debates. Tools like PyAnnote or NVIDIA NeMo could be combined with the fine-tuned Whisper model for end-to-end multi-speaker transcription, enhancing real-world applications.

7.2 Interactive and Modular User Interface

Developing a modular web-based UI (using Gradio, Streamlit, or Flask) can allow users to upload audio, select languages, enable features like noise filtering or diarisation, and view/download timestamped transcripts. This would make ASR tools accessible to non-technical users in fields like academia, media, and government.

7.3 Dialectal Adaptation and Dataset Expansion

Given India's linguistic diversity, future efforts should focus on creating custom dialectal datasets through crowdsourcing or media sources. Integrating dialect tagging and semi-supervised learning will enable scalable fine-tuning for dialects like Sylheti and Telangana Telugu, improving inclusivity and ASR accuracy.

7.4 Multi-Modal and Real-Time Capabilities

Future research can explore real-time streaming transcription, cross-lingual transcription (transcribe in one language, translate to another), and multimodal learning by combining audio, text, and possibly video for enhanced context understanding.

References

[1] Francisca Acheampong, Henry Nunoo-Mensah, and Wenyu Chen. 2021. Transformer models for text-based emotion detection: a review of BERT-based approaches. *Artificial Intelligence Review* 54 (12 2021), 1–41. doi:10.1007/s10462-021-09958-2

[2] Charith Chandra Sai Balne, Sreyoshi Bhaduri, Tamoghna Roy, Vinija Jain, and Aman Chadha. 2024. Parameter Efficient Fine Tuning: A Comprehensive Analysis Across Applications. arXiv:2404.13506 [cs.LG] https://arxiv.org/abs/2404.13506

[3] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 1–9. doi:10.18653/v1/2022.acl-short.1

[4] Hong Chen and Bo Zhang. 2019. Application of Automatic Speech Recognition (ASR) Algorithm in Smart Home. *Journal of Physics: Conference Series* 1237 (06 2019), 022133. doi:10.1088/1742-6596/1237/2/022133

[5] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. arXiv:1902.00751 [cs.LG] https://arxiv.org/abs/1902.00751

[6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL] https://arxiv.org/abs/2106.09685

[7] A. Kumar and V. Mittal. 2019. Speech recognition: A complete perspective. *International Journal of Recent Technology and Engineering* 7 (04 2019), 78–83.

[8] Jinyu Li. 2022. Recent Advances in End-to-End Automatic Speech Recognition. *APSIPA Transactions on Signal and Information Processing* 11, 1 (2022), –. doi:10.1561/116.00000050

[9] Riefkyanov Pratama and Agit Amrullah. 2024. ANALYSIS OF WHISPER AUTOMATIC SPEECH RECOGNITION PERFORMANCE ON LOW RESOURCE LANGUAGE. *Jurnal Pilar Nusa Mandiri* 20 (03 2024), 1–8. doi:10.33480/pilar.v20i1.4633

[10] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust Speech Recognition via Large-Scale Weak Supervision. arXiv:2212.04356 [eess.AS] https://arxiv.org/abs/2212.04356

[11] Andrew Rouditchenko, Sameer Khurana, Samuel Thomas, Rogerio Feris, Leonid Karlinsky, Hilde Kuehne, David Harwath, Brian Kingsbury, and James Glass. 2023. Comparison of Multilingual Self-Supervised and Weakly-Supervised Speech Pre-Training for Adaptation to Unseen Languages. doi:10.48550/arXiv.2305.12606

[12] Yavuz Toraman and Baris Gecit. 2023. User Acceptance of Metaverse: An Analysis for e-Commerce in the Framework of Technology Acceptance Model (TAM). *Sosyoekonomi* 31 (01 2023), 85–104. doi:10.17233/sosyoekonomi.2023.01.05