IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 April 2026, accepted 11 May 2026, date of publication 13 May 2026, date of current version 19 May 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3693119

RESEARCH ARTICLE

End-to-End Adaptation of LLMs for Low-Resource Languages

P. K. UDITH I. SANDARUWAN[1], NIMESH M. A. FONSEKA[1], PAMITH C. SALWATHURA[1], (Student Member, IEEE), DON SANJEEWA ALWIS[2], CHATHURA WANIGASEKARA[3], (Senior Member, IEEE), AND LOGEESHAN VELMANICKAM[1], (Member, IEEE)

[1]Department of Electrical Engineering, University of Moratuwa, Moratuwa 10400, Sri Lanka
[2]Decryptogen LLC, Cheyenne, DCG 82001, USA
[3]Institute of Maritime Technologies and Propulsion Systems, German Aerospace Center (DLR), 21502 Geesthacht, Germany

Corresponding authors: Chathura Wanigasekara (chathura.wanigasekara@dlr.de) and Logeeshan Velmanickam (logeeshanv@uom.lk)

ABSTRACT While Large Language Models (LLMs) have revolutionized information processing, their benefits are disproportionately skewed toward high-resource languages, leaving languages like Sinhala behind. Building on our earlier work, "An Approach to Training and Fine-Tuning Large Language Models for Low-Resource Languages," this extended version presents the complete development and evaluation of a Sinhala Large Language Model (LLM) obtained by training a LLaMA 3.1 base model for this underrepresented language. Developing the Sinhala LLM required addressing challenges inherent to low-resource language modelling. The scarcity of online Sinhala text and the prevalence of printed-only literature required extensive efforts in corpus creation, translation, and noise reduction. Sinhala's complex orthography further necessitated the design of a custom tokenization strategy aligned with existing pre-trained architectures. Additionally, the process of continued pre-training, fine-tuning, and quantization was constrained by limited hardware and memory resources. Despite these limitations, the trained Sinhala LLM demonstrates significant progress in adapting large-scale architectures to a low-resource context. Experimental results highlight consistent improvements in text generation quality, contextual understanding, and response coherence. This study demonstrates that effective LLMs can be built for low-resource languages even with limited hardware, providing a reproducible framework for researchers facing similar constraints.

INDEX TERMS Continued pre-training, data scarcity, large language models (LLMs), LoRA fine-tuning, low-resource languages, quantization, Sinhala language processing, transfer learning.

I. INTRODUCTION

Building upon our prior work, "An Approach to Training and Fine-Tuning Large Language Models for Low-Resource Languages" [1], this study presents the development and evaluation of a Sinhala Large Language Model (LLM) fine-tuned from the LLaMA 3.1 model. This journal extension expands on our earlier findings by deepening the methodological analysis, refining the training pipeline, and providing a more comprehensive evaluation of model behavior under real-world constraints.

The rapid evolution of large language models keeps pushing what we think AI can do. OpenAI's GPT-5, introduced in 2025, marked a big step forward in reasoning, multimodal understanding, and overall capability, setting a new bar for general-purpose AI [2]. Anthropic's Claude 3.7 series has also shown impressive progress, especially in reasoning-heavy and code-intensive tasks [3]. Meta's Llama 4 continues to shape the open-weight ecosystem, offering strong performance and efficiency that make it practical for both research and real-world applications.

Yet, while these frontier models dominate research and development, their capabilities do not extend equally across all languages. In particular, they struggle to perform effectively on Sinhala, the primary language spoken by over

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng Chen.

23 million people in Sri Lanka [4]. This gap highlights a systemic failure in current foundation models: the neglect of low-resource languages. Sinhala is particularly affected due to the scarcity of high-quality digital corpora and standardized datasets. Because the language is virtually absent from major pre-training mixtures, it remains unsupported by state-of-the-art models, effectively excluding millions of speakers from the benefits of modern NLP.

Addressing this gap is more than a technical challenge; it is a matter of linguistic equity and digital inclusion. Sinhala presents distinct hurdles for LLM development. There's a scarcity of high-quality digital text, complex morphology, non-Latin orthography, and limited standardized datasets. These factors complicate tokenization, data preprocessing, and model adaptation. Compounding this are the financial and computational barriers faced by research groups in developing countries like Sri Lanka, further widening the divide between high-resource and low-resource AI ecosystems.

In this work, we seek to narrow that divide. By systematically addressing dataset construction, tokenizer design, continued pre-training, and parameter-efficient fine-tuning within constrained computational environments, we demonstrate that meaningful progress in Sinhala language modeling is achievable. More broadly, this study aims to contribute toward a more inclusive and linguistically diverse AI landscape, where advances in large language models extend beyond high-resource languages and reach communities that have historically been left behind.

To clearly define how this full study advances beyond our preliminary framework, the following subsection outlines the specific technical expansions and performance gains realized in this work.

A. KEY TECHNICAL CONTRIBUTIONS BEYOND PRIOR WORK

This article represents a significant extension of our preliminary research presented in [1]. While the earlier work introduced the fundamental framework for Sinhala adaptation during its initial training phases, this manuscript provides the finalized development lifecycle and a rigorous empirical validation. The primary technical novelties and quantified advancements of this work are as follows:

- **Quantified Tokenizer Efficiency and OOV Elimination:** Unlike the preliminary study, we provide a detailed ablation of the vocabulary expansion. By introducing ∼35,000 Sinhala-specific tokens (∼30% of the final vocabulary), we achieved a 74.4% reduction in average token length per Sinhala sentence (from 91.4 to 23.4 tokens). This strategy effectively eliminated byte-level fallback (OOV rate) from 97.5% to 0.0%, ensuring the model captures the morphologically rich structure of the language rather than relying on fragmented subwords.
- **Stabilized Continued Pre-training (CP) and Fine-Tuning:** We transitioned to a strategy where training was strictly localized to input/output embeddings during the CP phase and low-rank adapters during instruction tuning. This architectural constraint, paired with a 60/40 Sinhala-to-English data mixture, proved essential in maximizing linguistic fluency while successfully preventing "catastrophic forgetting" of the core reasoning capabilities inherent in the base LLaMA 3.1 model.
- **Rigorous Multidimensional Evaluation Framework:** This manuscript introduces a complete suite of quantitative and qualitative results, moving beyond the "work-in-progress" status of [1]. We address the limitations of surface-level overlap metrics (e.g., ROUGE/BLEU) by introducing:
  - **Contextual Perplexity Analysis:** We demonstrate an 89.53% reduction in token-level perplexity. We clarify that the high initial perplexity (2.313E8) stems from the random initialization of the expanded vocabulary embeddings, which our training successfully regularized to 2.422E7.
  - **Structured LLM-as-a-Judge:** We implemented a blind evaluation using DeepSeek-V3 guided by a 0–5 scoring rubric (focusing on coherence and syntax) across 20 human-verified reference prompts. This recorded a quality improvement from a baseline of 1/5 to 4.5/5.
- **Reproducible Hardware and Hyperparameter Recipe:** We provide the first detailed technical report on training Sinhala LLMs under resource-constrained environments. This includes specific disclosures on FP16 mixed-precision usage, 4-bit quantization, and the precise learning rate warmup schedules required to achieve convergence on a single-GPU setup.

II. LITERATURE REVIEW

Sinhala remains structurally underserved in every major pre-training corpus. Frontier models handle it only as a side effect of broad multilingual exposure, not as a language intentionally optimized for. This creates the same pattern across all benchmark results: reasonable performance from massive proprietary models, partial or inconsistent performance from open-source models, and no model that treats Sinhala as a first-class target. Prior studies already indicate the gap. Claude 3 Sonnet and GPT-4o hold up across translation, summarization, and QA despite never being tuned specifically for Sinhala, while models like Llama 3 and Mistral 7B would show noticeable improvement only after task-aligned or instruction-aligned fine-tuning. Even then, the ceiling remains lower than what these models achieve in high-resource languages. The primary constraint is data representation: Sinhala is almost absent from pre-training data, and the surrounding ecosystem (tokenizers, datasets, evaluation frameworks) lags behind languages with significant representation [5].

Building a large model from scratch for Sinhala is not feasible. The data volume alone makes it unrealistic. GPT-3's

175-billion-word corpus illustrates the scale required for modern pre-training [6]. Models like BERT, GPT-2, and every major successor rely on this broad, general-purpose pre-training to internalize linguistic structure before any task-specific learning happens. The entire field now operates on this two-stage pipeline: large-scale pre-training, then targeted adaptation. This setup gives a model enough general capability to handle diverse downstream tasks without rebuilding it from the ground up, which would be computationally expensive. Transfer learning makes this pipeline workable [7]. A pre-trained model carries forward the linguistic and world knowledge learned from massive datasets, so adapting it to a new domain becomes a matter of fine-tuning rather than full retraining. This cuts down both compute and data demands, since the task-specific dataset only needs to steer an already capable model rather than create one from scratch [8].

The first intervention point is continued pre-training. Because Sinhala barely appears in the original corpora, the model's internal representation of the language is weak or inconsistent. Continued pre-training widens this representation by exposing the model to large quantities of Sinhala text [9]. It teaches structure, syntax, morphology, discourse flow, and code-mixing patterns that the base model never sees at scale. This step corrects the imbalance created during initial multilingual training. But it introduces a second problem: catastrophic forgetting. When the model absorbs new data distributions, earlier capabilities may deteriorate, especially if the new corpus is narrow or repetitive [10], [11]. Sinhala-heavy pre-training must therefore be balanced to strengthen the language without erasing useful general-world and multilingual knowledge.

Instruction fine-tuning forms the next layer. Continued pre-training improves internal representation, but it does not teach the model how to behave as an assistant. Instruction data provides structure. Well-formed prompts and aligned outputs create consistent task behavior across reasoning, translation, classification, summarization, and open-ended generation [12].

Full fine-tuning is generally impractical for teams without large compute budgets. Updating all parameters increases cost, extends training time, and amplifies the risk of overfitting – especially in low-resource contexts where data diversity is limited [13]. Parameter-Efficient Fine-Tuning (PEFT) resolves this by modifying only small, strategically selected parts of the model while freezing the rest. LoRA attaches low-rank update matrices to targeted weight matrices, letting the model absorb new patterns without rewriting its entire parameter space. In Transformer-based LLMs, LoRA commonly targets the self-attention projections (W_q, W_k, W_v, W_o). These layers control how tokens interact across sequences, so even small updates produce meaningful shifts in language-specific reasoning and generation. This yields a resource-efficient training framework that is computationally light, stable even with smaller datasets, and capable of real gains in Sinhala performance without degrading overall capability.

III. ENVIRONMENT AND HARDWARE SETUP

All experiments in this study were carried out with fairly constrained computational resources. We didn't have access to massive GPU clusters, so the workflow had to be carefully optimized to make the most of what was available, covering tokenizer training, continued pre-training, LoRA fine-tuning, and hyperparameter tuning. These limitations shaped several decisions: batch sizes and sequence lengths had to be smaller, each training stage ran for a limited duration, and the number of hyperparameter search iterations was restricted. Despite these constraints, the model steadily improved at every stage. We expect that with more compute, longer training, larger batches, and a deeper hyperparameter sweep, the performance could be even stronger. The table below summarizes the hardware setup and the key training details for each stage.

TABLE 1. Hardware and training setup.

Resource	Details
GPU	NVIDIA A40 (48 GB VRAM)
System Memory	50 GB
vCPUs	9
Pre-training	2 GPUs (occasional parallel scaling)
Fine-tuning	1 GPU
Hyperparameter Tuning	2 weeks (16 iterations)
Continued Pre-training	2 weeks
LoRA Fine-tuning	∼1 week

IV. METHODOLOGY

Developing a Sinhala Large Language Model required a structured pipeline to overcome challenges associated with data scarcity, tokenization, and computational constraints. Our approach began with dataset collection, curating diverse text sources, and applying careful preprocessing to make the data as clean and useful as possible. We then explored tokenization strategies, tailoring them to Sinhala's unique script and linguistic properties. The pre-training and fine-tuning phase involved adapting a pre-trained model using techniques like PEFT and LoRA to optimize performance within resource limitations. Finally, we applied quantization and other optimizations to reduce memory usage and make inference more efficient. The sections that follow walk through each of these steps in detail, showing how the pipeline comes together to produce a functional Sinhala LLM.

A. DATASET COLLECTION

Before selecting Llama 3.1 as the base model for fine-tuning, we experimented with several prominent open-source models, including Qwen 2 (7B), Phi-3, and Mistral (7B), due to their popularity within the research community. These

models were evaluated using diverse Sinhala queries. Despite some prior exposure to Sinhala data, their generated outputs were largely incomprehensible. LLaMA 3.1 (8B) performed slightly better, but it still consistently failed to produce coherent Sinhala responses.

We further determined that, without extensive pre-training on Sinhala-specific data, advanced prompting techniques such as Retrieval Augmented Generation (RAG) were ineffective in producing meaningful Sinhala outputs. These observations made it clear that if we wanted a model that truly understood Sinhala, fine-tuning on targeted datasets was unavoidable.

Given the scarcity of publicly available digital resources and NLP tools for Sinhala, we had to create custom datasets. Two high-quality, purpose-built datasets were constructed:

- **Raw Sinhala Dataset** −− Used for continued language model pre-training to build foundational linguistic understanding.
- **Instruction Fine-Tuning (IFT) Dataset** – Developed specifically to improve task-oriented, instruction-following capabilities, fine-tuned using LoRA.

1) RAW SINHALA DATASET

The Raw Sinhala Dataset consists of authentic, culturally relevant Sinhala texts obtained primarily through web scraping from reputable Sinhala news websites and publicly accessible sources such as books and literary content. We used frameworks like `BeautifulSoup` and `Scrapy` to automate collection and corpus creation.

To ensure high dataset quality, rigorous preprocessing steps were conducted, comprising:

- **Duplicate removal** to eliminate redundant textual samples.
- **Length-based filtering** to exclude samples that were either too short (lacking context) or excessively long (potentially off-topic).
- **Null and incomplete entry removal** to maintain structural consistency.
- **Profanity and low-quality content filtering** using a carefully curated, language-specific blocklist of inappropriate Sinhala terms.
- **Random character repetition filtering** to discard nonsensical or spam-like content.

The final Raw Sinhala Dataset combines scraped content with the instruction dataset, giving a corpus much larger than the IFT dataset alone. This ensured broad linguistic coverage for continued pre-training.

2) INSTRUCTION FINE-TUNING DATASET

The Instruction Fine-Tuning dataset was primarily generated using a structured pipeline based on the Multilingual Reverse Instructions (MURI) technique [14]. This approach was strategically selected due to its demonstrated ability to create culturally relevant, linguistically rich, and diverse instruction-response pairs.

We started by collecting Sinhala content from trusted sources like `BBC Sinhala science`, `Vidusara science`, and `Anroidwadakaroyo`. This gave us millions of authentic texts across domains and writing styles.

The MURI pipeline then followed these steps:

1) **Content Selection:** Pick high-quality Sinhala samples from the scraped sources.
2) **English Translation:** Convert these samples into English to serve as intermediate representations.
3) **Instruction Generation:** Use a pre-trained LLaMA model to generate task-oriented English instructions from the English text.
4) **Back-translation:** Translate the generated instructions back into Sinhala to preserve cultural and linguistic fidelity.
5) **Pair Formation:** Match each Sinhala instruction with its original Sinhala response to form cohesive instruction-response pairs.

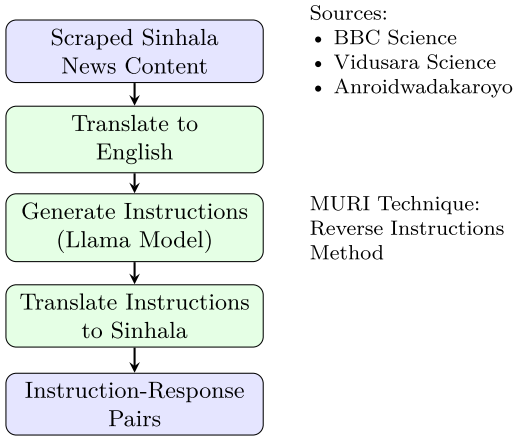

FIGURE 1. MURI pipeline for sinhala instruction dataset generation.

This approach ensures that, while instructions are systematically generated, the responses remain authentic, human-written Sinhala.

We further enriched the dataset using augmentation. Hindi language resources were particularly useful because of their linguistic similarity to Sinhala as they share Indo-Aryan roots, grammatical patterns, and vocabulary borrowings. Carefully translated Hindi instruction-response pairs added natural, culturally appropriate Sinhala examples. Other augmentation sources included:

- Extracting and translating existing multilingual datasets available on platforms such as Hugging Face.
- Collecting additional authentic Sinhala content through targeted web scraping of reputable Sinhala news websites and publicly available PDF documents.
- Incorporating a limited number of high-quality synthetic examples generated using advanced commercial models such as DeepSeek and Claude (limited in scale due to associated costs).

TABLE 2. Datasets created and used in this study.

Dataset Name	Purpose	HuggingFace Repository URL
MURI Source Datasets		
BBC Sinhala Science	MURI source content	[15]
Vidusara Science	MURI source content	[16]
Anroidwadakaroyo	MURI source content	[17]
MURI Test Sample	MURI technique validation	[18]
Raw Sinhala Dataset		
OScar Sinhala	Raw corpus for pre-training	[19]
Hindi-to-Sinhala Translation Datasets		
Hindi Translation 1	Augmentation (general)	[20]
Hindi Translation 2	Augmentation (general)	[21]
WikiHow Sinhala (translated from Hindi Wikihow dataset)	Augmentation (translated)	[22]
Anudesh Sinhala (translated Hindi Anudesh dataset)	Augmentation (translated)	[23]
Final Instruction Fine-Tuning Dataset		
Sinhala IFT Complete	Complete merged dataset	[24]

- Integrating English instruction-response examples sourced from the publicly accessible `JeanKaddour/minipile` dataset, selectively translated to Sinhala where appropriate.

The use of Hindi-derived instruction sets for augmentation is justified by the common Indo-Aryan roots shared with Sinhala, which minimizes the introduction of translation artifacts compared to more distant language families. This approach allowed for a broader range of task coverage while ensuring the model adhered to familiar syntactic structures. To ensure linguistic integrity, this augmented data was restricted to 2% of the total IFT corpus, serving as a secondary resource to the primary Sinhala-native datasets

While the instruction fine-tuning process incorporated both English and Sinhala samples, Sinhala content constituted over 90% of the total dataset, explicitly prioritizing fluency and linguistic depth in the target language. It is publicly available on Hugging Face and contains:

- **Total instances:** 3,627,982 instruction-response pairs
- **Dataset size:** 3.86 GB (raw), 1.42 GB (Parquet)
- **Token count:** ∼4 billion tokens
- **Data split:** 90% training, 10% evaluation

A representative set of instruction-response samples from the finalized IFT dataset is illustrated in Figure 2.

FIGURE 2. Sample instruction-response pairs from the instruction fine-tuning dataset.

By combining MURI-based generation, Hindi-based augmentation, and diverse scraping sources, this pipeline produced one of the largest and most comprehensive Sinhala instruction-following datasets to date.

B. TOKENIZATION APPROACH

Once the Sinhala dataset was compiled, we could go on to the next step and tokenize the dataset. Sinhala is severely underrepresented in most large-scale pre-training corpora, which means that the default tokenizer of a pretrained model does not meaningfully capture its structure. Without careful handling at this stage, even extensive fine-tuning would fail to produce coherent Sinhala outputs.

Naively adding Sinhala tokens to an existing pretrained tokenizer or retraining a tokenizer without preserving the original configuration consistently resulted in catastrophic failure. Models would generate incoherent or gibberish outputs and, in some cases, lose their ability to produce meaningful text altogether. These failures revealed a key insight: the tokenizer configuration itself is tightly coupled to the pretrained model's internal representations. Any mismatch disrupts how inputs are interpreted at the embedding level.

To address this, we adopted a Byte Level Byte Pair Encoding (BPE) tokenization strategy that aligns directly with the architecture of the LLaMA 3.1 (8B) model. This proved successful for Sinhala word tokenization since it ensured compatibility at the lowest level of token processing, particularly for a non-Latin, morphologically rich language like Sinhala.

Using the `Hugging Face` tokenizers library, we trained a new tokenizer from scratch on a Sinhala-only corpus extracted from both question and answer data. However, the most critical step was not the training itself, but how the tokenizer was initialized. Before training, we replicated the exact configuration of the pretrained LLaMA tokenizer. This included copying the pre-tokenizer, normalizer,

post-processor, and all special tokens (PAD, CLS, SEP, UNK, MASK). Matching these components precisely ensured that token boundaries, normalization behavior, and sequence formatting followed the same rules the model had learned during its original pre-training.

The new tokenizer was then trained using this configuration, producing Sinhala-specific vocabulary entries and merge rules. The pre-trained model's tokenizer could not be directly fine-tuned for our specific dataset since that caused the embedding layer to be tampered with and resulted in degraded performance when the model was inferenced afterward. Similarly, simply appending tokens without controlling ID assignment caused collisions and prevented the model from reliably distinguishing between Sinhala and English.

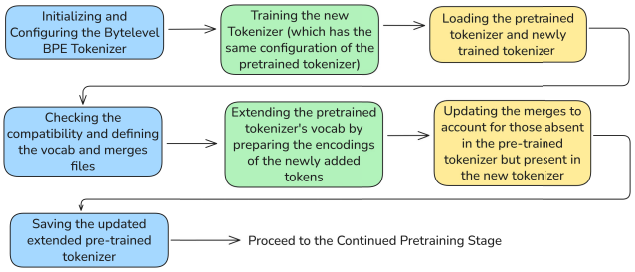

FIGURE 3. Tokenization approach.

To resolve this, we extended the model's vocabulary in a controlled manner instead of modifying it. Both tokenizer JSON files were loaded, and vocabularies and merge rules were extracted. Sinhala tokens that did not already exist in the pretrained vocabulary were appended, with new token IDs assigned starting from the maximum unused index. Merge rules were similarly extended, ensuring no conflicts with existing rules. Importantly, configuration files such as `special_tokens_map.json` and `tokenizer_config.json` were copied directly from the pretrained tokenizer to preserve behavior consistency.

TABLE 3. Tokenization efficiency: Original vs extended vocabulary.

Metric	Original LLaMA	Extended
Vocabulary size	128,256	163,963
Avg. tokens per Sinhala sentence	91.4	23.4
Byte-level fallback rate (%)	97.5%	0.0%
Token reduction (%)	–	74.4%

Here, approximately 35,000 Sinhala tokens were added to the tokenizer, which accounted for close to 30% of the total tokens of the modified tokenizer. Adding more tokens would have required proportionally more continued pre-training, as each new token introduces a fresh embedding that the model must learn. By extending the vocabulary rather than modifying it, the original embedding space was preserved to a large extent. Only the newly introduced Sinhala tokens required training, while existing English and multilingual representations remained intact. Empirically, this was reflected in stable English-language performance alongside steadily improving Sinhala generation quality, validating the success of our tokenizer fine-tuning process.

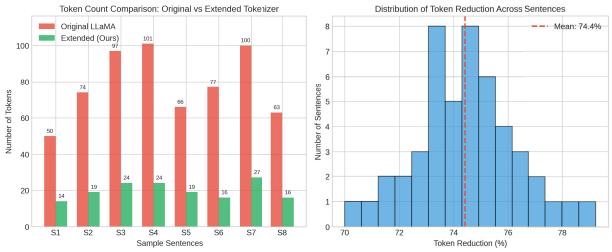

FIGURE 4. (Left) Token count comparison across sample Sinhala sentences showing consistent reduction with the extended tokenizer. (Right) Distribution of token reduction percentage across all test sentences.

The original LLaMA tokenizer required an average of 91.4 tokens per Sinhala sentence, with a byte-level fallback rate of 97.5%, indicating severe fragmentation where Sinhala characters were decomposed into individual bytes rather than meaningful subwords. After extending the vocabulary with approximately 35,707 Sinhala tokens, the average dropped to 23.4 tokens per sentence (a 74.4% reduction), with byte-level fallbacks reduced to 0.0%.

We chose vocabulary extension over full SentencePiece retraining because the latter would have required re-initializing all embedding weights from scratch, negating the transfer learning benefits from the pre-trained model. Our controlled extension approach preserves existing embeddings while adding new Sinhala-specific tokens that integrate naturally with continued pre-training. With a tokenizer capable of representing Sinhala text accurately and consistently, we proceeded to the next stage of the pipeline: continued pre-training.

C. MODEL CONTINUED PRE-TRAINING & LoRA FINE-TUNING

Enabling a pretrained model to handle Sinhala required more than instruction fine-tuning. The model first needed a stable internal representation of the language itself. For this reason, we adopted Domain-Adaptive Pre-Training (DAPT); Continued pre-training on raw Sinhala text so the model could absorb Sinhala-specific vocabulary, syntax, and distributional patterns that were largely absent from its original training data.

At this stage, we made a choice about what parts of the model to update. The input and output embedding layers were unfrozen so that the newly introduced Sinhala tokens could be integrated into the model's parameter space. All intermediate Transformer layers were kept frozen. Early experiments that unfroze attention layers resulted in immediate degradation of model behavior, including unstable generation and loss of previously learned capabilities. These observations

reinforced the need to constrain updates carefully when adapting a large model under limited data and compute.

A second failure mode emerged when we attempted continued pre-training exclusively on Sinhala text. While this strengthened Sinhala token activation, it also caused catastrophic forgetting of English and other multilingual capabilities. The model's responses deteriorated sharply outside Sinhala, indicating that the new data distribution was overpowering the representations learned during original pre-training.

The Reuse, Don't Retrain [8] (NVIDIA) paper recommends that continued pre-training should begin with a dataset similar to the model's original training distribution before transitioning to domain-specific data. Taking this insight into consideration, we used two datasets for the continued pre-training stage:

- **`JeanKaddour/minipile`** (40%) – Chosen for its similarity to the original LLaMA 3.1 training data, predominantly English and general-domain.
- **Raw Sinhala dataset** (60%) – Curated to strengthen Sinhala language representation.

This balance proved critical. Preliminary experiments indicated that ratios heavily skewed toward the target language (e.g., >80%) led to a degradation in the model's foundational reasoning capabilities. The 60/40 ratio was selected as an optimal balance to deepen Sinhala linguistic understanding while preserving the robust multilingual internal representations of the Llama 3.1 base. Without this anchoring, continued pre-training consistently destabilized the model.

Another important observation was the ordering of adaptation stages. Immediately after tokenizer extension, the model produced incoherent outputs containing fragmented Sinhala mixed with English. This confirmed that adding tokens alone is insufficient. Continued pre-training was necessary to align the new embeddings with meaningful linguistic structure. After this stage, the model began generating Sinhala-only outputs in response to Sinhala prompts, but with limited fluency due to short training duration. While the responses were not yet strong, they clearly indicated that the adaptation process was moving in the correct direction.

The LoRA fine-tuning stage saw the model being trained on the Instruction Fine-Tuning (IFT) dataset that included both English and Sinhala Q&A pairs. A full fine-tuning approach would have been computationally expensive and could have led to overfitting due to limited training resources. Instead, LoRA efficiently integrated Sinhala-specific knowledge by introducing low-rank parameter matrices into the model weights.

Rather than updating the full set of model parameters, LoRA freezes most of the pre-trained model weights and introduces small trainable matrices (LoRA Adapters) within selected layers. This approach significantly reduces the number of trainable parameters while still allowing effective adaptation to the new dataset. The following figure illustrates this mechanism:

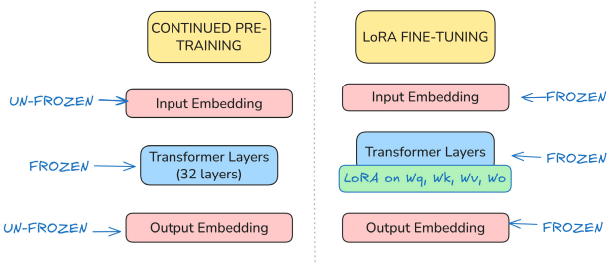

FIGURE 5. LoRA fine-tuning mechanism: Most model weights remain frozen while LoRA adapters introduce trainable low-rank updates.

During continued pre-training, all intermediate Transformer layers (32 layers) were frozen, and only the input embedding layer (`embed_tokens`) and the output projection layer (`lm_head`) were set as trainable. In the LLaMA 3.1 (8B) architecture, these two layers are not weight-tied and therefore maintain separate parameter matrices. This configuration resulted in approximately 770 million trainable parameters out of a total of 8.03 billion (9.6%), while the remaining 7.26 billion parameters (90.4%) were kept frozen.

Training was conducted for 10,000 steps with a per-device batch size of 2 and a gradient accumulation factor of 3, yielding an effective batch size of 6. With a maximum sequence length of 512 tokens, this corresponds to approximately 30.7 million training tokens ($10{,}000 \times 6 \times 512$).

The model was optimized using AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and a weight decay of 0.01. A cosine learning rate schedule was employed, with a linear warmup phase of 267 steps that increased the learning rate from 0 to 7.68×10^{-4}, followed by cosine decay to 0 over the remaining training steps. Gradient clipping was applied with a maximum norm of 1.0.

Training was performed in mixed precision (FP16), with gradient checkpointing enabled to reduce activation memory usage by approximately 70%.

During continued pre-training, mixed precision training with FP16 was employed to reduce memory consumption while maintaining numerical stability. For LoRA fine-tuning, the base model was loaded in 4-bit precision using the `NormalFloat4 (NF4)` quantization scheme via the `BitsAndBytes` library, with double quantization enabled to further reduce memory overhead. The compute data type was set to `BFloat16` when Flash Attention 2 was available, and `FP16` otherwise. The optimizer for the LoRA stage was Paged AdamW 8-bit, which offloads optimizer states to CPU memory when GPU memory is insufficient.

For deployment, post-training quantization was applied using Activation-Aware Weight Quantization (AWQ) with a 4-bit weight configuration (group size of 128, zero-point calibration enabled). This reduced the model size to approximately 4.5 GB while preserving inference quality, enabling deployment on consumer-grade hardware.

TABLE 4. Continued pre-training configuration.

Parameter	Value
Trainable parameters	770M / 8.03B (9.6%)
Trainable layers	Input embeddings + output projection
Frozen layers	32 Transformer layers
Weight tying (embed/lm_head)	No (separate matrices)
Total training steps	10,000
Per-device batch size	2
Gradient accumulation steps	3
Effective batch size	6
Sequence length	512 tokens
Total training tokens	∼30.7M
Optimizer	AdamW ($\beta = 0.9/0.999$, $\epsilon = 10^{-8}$)
Weight decay	0.01
Learning rate	7.68×10^{-4}
LR schedule	Cosine with linear warmup
Warmup steps	267
Max gradient norm	1.0
Precision	FP16 (mixed precision)
Gradient checkpointing	Yes

During the LoRA fine-tuning stage, LoRA adapters were applied to seven target modules within each Transformer layer: the query, key, value, and output projections of the self-attention mechanism (W_q, W_k, W_v, W_o), as well as the gate, up, and down projections of the feed-forward network (W_{gate}, W_{up}, W_{down}). The LoRA configuration used a rank of 16 with $\alpha = 24$ and a dropout rate of 0.1.

Sequence packing was enabled during training (via `SFTTrainer`), allowing multiple shorter examples to be combined into a single 2,048-token sequence. This improved GPU utilization and training efficiency without altering the underlying data distribution.

TABLE 5. LoRA fine-tuning configuration.

Parameter	Value
Total training steps	15,000
Per-device batch size	2
Gradient accumulation steps	2
Effective batch size	4
Sequence length	2048 tokens
Training quantization	4-bit NF4 (BitsAndBytes, double quantization)
Optimizer	Paged AdamW 8-bit
Weight decay	0.01
Learning rate	7.68×10^{-4}
LR schedule	Linear
Warmup steps	4000
Max gradient norm	0.3
Precision	BF16 (with Flash Attention) or FP16
Gradient checkpointing	Yes

Fine-tuning on a Sinhala-only instruction dataset led to another instance of catastrophic forgetting, with the model losing its ability to respond coherently in English. This was resolved by constructing a mixed instruction dataset containing both Sinhala and English question–answer pairs. The mixed setup allowed the model to retain general reasoning and English proficiency while steadily improving Sinhala instruction-following behavior.

To further optimize performance, we tuned the hyper-parameters of both continued pre-training and LoRA fine-tuning. We employed Bayesian Optimization, an iterative search strategy that efficiently explores the hyper-parameter space to maximize performance. We conducted 16 optimization iterations for each stage to determine the best hyper-parameter settings.

1) BAYESIAN OPTIMIZATION FOR HYPERPARAMETER TUNING

For our experiments, we employed Bayesian Optimization to optimize key hyperparameters for both *LoRA fine-tuning* and *continued pre-training*, ensuring efficient training while reducing computational costs.

For each training stage, we optimized the following hyperparameters:

- **LoRA Fine-Tuning:** LoRA alpha, LoRA rank, learning rate, dropout rate, gradient accumulation steps.
- **Continued Pretraining:** Learning rate, gradient accumulation steps, warm-up steps, and learning rate scheduler type.

To identify the best configuration, we performed 16 iterations of Bayesian Optimization for each stage, evaluating perplexity as the primary metric.

2) RESULTS OF THE HYPERPARAMETER TUNING

The results obtained from the above approach is outlined below for both stages. The identified configurations were used for the scaled-up training of the model thereafter to optimize the training and extract the best performance from the model using the limited resources.

TABLE 6. Best hyperparameter setting - CP.

Hyperparameter	Value
Learning Rate	0.00076818235
Gradient Accumulation Steps	3
Warmup Steps	267
LR Scheduler Type	Cosine

TABLE 7. Best hyperparameter setting - LoRA.

Hyperparameter	Value
LoRA Alpha	23
LoRA Dropout	0.14773642825
Learning Rate	0.00012023049
Gradient Accumulation Steps	2

The above plots illustrate the smooth and stable training behavior achieved using the above optimized hyperparameters derived from our methodology.

D. MODEL EVALUATION

The model was evaluated at the end of the final stage, following LoRA fine-tuning. Both quantitative metrics and qualitative inference-based analysis were used to assess performance, as relying solely on automated evaluation

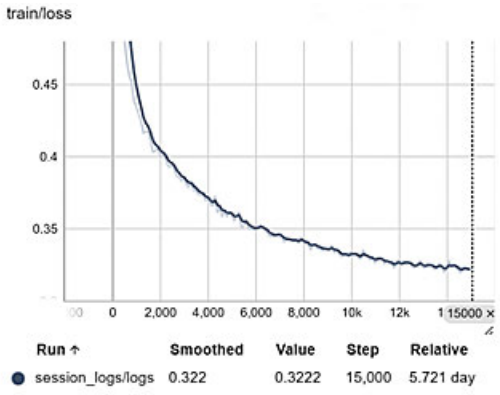

FIGURE 6. Training Loss during LoRA fine-tuning.

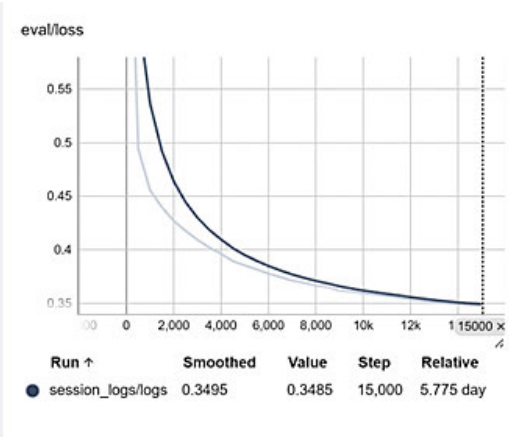

FIGURE 7. Evaluation Loss during LoRA fine-tuning.

metrics was found to be insufficient for capturing real-world behavior in Sinhala.

The quantitative evaluation was conducted on a held-out subset of 50 samples drawn from the evaluation split (10%) of the Sinhala IFT dataset (`Pamzyy/merged_final1_cleaned`). The test set consisted exclusively of Sinhala question-answer pairs. The reference answers were human-written and originated from the MURI pipeline described in Section IV-A2, where authentic Sinhala content sourced from reputable news outlets formed the basis for response generation.

For each sample, the model was provided with the question as input and generated a response with a maximum of 200 new tokens. The generated outputs were then compared against the reference answers using ROUGE, BLEU, METEOR, and perplexity.

Length normalization was intentionally not applied. Sinhala's agglutinative morphology and flexible word order make token-count-based normalization unreliable, as semantically equivalent responses can vary significantly in surface-level length depending on whether compound or analytic constructions are used.

We note that the evaluation sample size (50) is relatively small, and formal statistical significance testing (e.g., bootstrap confidence intervals or paired tests) was not conducted. This will be addressed in future work by expanding the evaluation set and incorporating rigorous statistical validation across all metrics.

Quantitatively, the Sinhala Model recorded lower scores across most traditional evaluation metrics, including ROUGE, and BLEU, when compared to the Base Model. However, our model achieved a substantial reduction in perplexity and a marginal improvement in the METEOR score, indicating significantly improved next-token prediction and semantic alignment for Sinhala text.

Perplexity was computed at the token level using the natural logarithm (base e), following the standard formulation:

$$\mathrm{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(x_i \mid x_{<i})\right),$$

where N denotes the number of non-padding tokens in each sequence. The metric was evaluated on 50 held-out samples from the evaluation split of the Sinhala Instruction Fine-Tuning dataset (`Pamzyy/merged_final1_cleaned`), using the reference answers as input sequences. To ensure a fair comparison, both models were evaluated using the same extended tokenizer.

The resulting perplexity values are significantly higher than those typically reported in English-language benchmarks, where values in the range of 10–100 are common. This difference is both expected and directly attributable to the tokenizer extension process. Approximately 35,000 Sinhala tokens were appended to the original LLaMA 3.1 vocabulary, expanding it to roughly 163,000 tokens. For the Base Model, the embeddings corresponding to these newly introduced tokens remain randomly initialized. As a result, when processing Sinhala text, the model assigns near-uniform probability across a large portion of the vocabulary. In the limiting case of a uniform distribution over V tokens, perplexity approaches V, which explains the observed Base Model perplexity of 2.313×10^8.

Following continued pre-training and LoRA fine-tuning, the model begins to learn meaningful probability distributions over the Sinhala token space, assigning higher likelihood to contextually appropriate tokens. This shift is reflected in the substantial 89.5% reduction in perplexity ($2.313 \times 10^8 \rightarrow 2.422 \times 10^7$). These values should therefore be interpreted in relative terms as an indicator of improvement in Sinhala language modeling capability, rather than being directly compared against absolute perplexity benchmarks derived from English-only models with stable, well-trained vocabularies.

From a qualitative standpoint, the Base Model struggled significantly when prompted in Sinhala. Responses were largely incoherent, frequently repetitive, and often deviated from the question intent. In many cases, the model failed to maintain focus, producing outputs with grammatical

TABLE 8. Quantitative evaluation results of base model and sinhala model.

Metric	Base Model	Sinhala Model	% Change
METEOR	0.210	0.222	**+5.71%**
ROUGE-1	0.120	0.071	-40.83%
ROUGE-2	0.092	0.040	-56.52%
ROUGE-L	0.110	0.070	-36.36%
ROUGE-L SUM	0.110	0.073	-33.64%
BLEU	0.100	0.059	-41.00%
PERPLEXITY	2.313E8	2.422E7	**-89.53%**

errors and fragmented structure, making it unsuitable for meaningful Sinhala inference. In contrast, the Sinhala Model demonstrated a clear qualitative improvement. The model was able to generate more structured and contextually relevant responses, with noticeable gains in factual correctness for certain questions. However, despite this improvement, limitations remained. Responses to more abstract or conceptual prompts, such as those related to the benefits of artificial intelligence or the nature of the human mind, often lacked sufficient depth or clarity. This indicates that while the model had begun to internalize Sinhala language patterns, higher-level reasoning and abstraction remained underdeveloped, likely due to training duration and computational constraints.

This discrepancy highlights a critical limitation of standard traditional metrics when applied to low-resource languages. Metrics such as ROUGE and BLEU are heavily dependent on surface-level token overlap and reference alignment, which may not accurately reflect semantic correctness, fluency, or contextual relevance in Sinhala. In practice, despite lower metric scores, our model consistently produced more meaningful, language-consistent, and task-relevant Sinhala responses than the Base Model.

1) MODERN EVALUATION: LLM AS A JUDGE

To address the inherent limitations of traditional overlap-based metrics like ROUGE and BLEU, which often fail to capture the semantic and morphological complexities of the Sinhala language, we adopted a modern "LLM-as-a-Judge" evaluation framework. This approach utilizes a high-performing, superior model to evaluate the semantic alignment and linguistic quality of generated responses beyond simple word-matching.

For this evaluation, we utilized the **DeepSeek** model as our primary evaluator. A test set consisting of 20 randomly sampled prompts was selected from the held-out dataset, covering a range of categories including general knowledge, conversational queries, and specific navigation commands. The judge model was tasked with comparing the responses generated by both the base model and our fine-tuned model.

Each response was assessed on a scale of 0 to 5 based on a defined rubric focused on three key linguistic dimensions:

- **Coherence:** Evaluates the logical flow and structural integrity of the generated Sinhala text.
- **Contextual Relevance:** Measures how effectively the model addressed the user's intent within the prompt.
- **Syntactic Correctness:** Checks for adherence to proper Sinhala grammar, orthography, and sentence structure.

The following is the system prompt that was passed to the Judge model, used to evaluate the responses given by both models for a given question.

Evaluation Prompt:

> "You are an expert linguist specializing in the Sinhala language. Your task is to evaluate the quality of two AI-generated responses based on a user prompt. Evaluate both responses based on: 1) Coherence, 2) Contextual Relevance, and 3) Syntactic Correctness. For each response, provide a brief justification for your assessment and then assign a final score from 0 to 5, where 0 is completely nonsensical and 5 is an excellent response."

a: EXAMPLE 1: IMPACT OF AI ON EDUCATION

Prompt (Sinhala): AI භාවිතයෙන් අධ්‍යාපනයට ඇති බලපෑම මොකක්ද?

Base Model Response: AI භාවිතයෙන් අධ්‍යාපනයට ඇති බලපෑම මොකක්ද? නිව්යෝක් විශ්වවිද්‍යාලයේ ප්‍රොෆෙසර් මරියා බ්‍රස්ට්ට් අපට පෙන්වයි. මෙම වාර්තාව සම්බන්ධයෙන් මොරියා බ්‍රස්ට්ට්

The Base Model failed to produce a meaningful answer. The response contained repetition of the input prompt, irrelevant references to foreign academics, corrupted characters, and abruptly terminated without conveying any substantive information.

Sinhala Model Response: 2023 දී ස්වාධීන අධ්‍යයන සේවා සමාගමක් වන ProQuest විසින් කරන ලද අධ්‍යයනයකින් හෙළි වූයේ A.I.(වාග්මාලාවාදී බුද්ධිය) වෙතින් ලබා දී ඇති අධ්‍යාපනික සම්පත් වැඩි වන විට, සිසුන් වඩාත් බුද්ධිමත් හා සාර්ථක වෙමින් පවතින බවයි. මෙම අධ්‍යයන දැනුම ලබා දෙන්නේ කෙසේද යන්න පැහැදිලි කරන්නේ නැත. කෙසේ වෙතත්, AI වෙතින් ලබා දී ඇති විවිධ සම්පත් සහ ඒවායේ වාසි සහ අවාසි පිළිබඳව ඔවුන් විස්තර කරයි. AI වෙතින් ලබා දී ඇති විවිධ සම්පත් අතරට ඇතුළත් වන

The model generated a coherent and contextually relevant explanation, referencing an external study (ProQuest, 2023) and discussing how AI-driven educational resources can enhance student learning outcomes. While slightly verbose,

the response maintained topical focus and demonstrated clear semantic understanding.

LLM Evaluation Score:

- Base Model: 1/5
- Sinhala Model: 4/5

b: EXAMPLE 2: SRI LANKAN EDUCATION SYSTEM

Prompt (Sinhala): ශ්‍රී ලංකාවේ අධ්‍යාපන ක්‍රමය ගැන විස්තර කරන්න

Base Model Response: ශ්‍රී ලංකාවේ අධ්‍යාපන ක්‍රමය ගැන විස්තර කරන්න ශ්‍රී ලංකාවේ අධ්‍යාපන ක්‍රමය ගැන විස්තර කරන්න අධ්‍යාපන ක්‍රමය ගැන විස්තර කරන්න මෙහි දී අනුකෘම්පනය කරන්න අධ්‍යාපන ක

The Base Model exhibited severe repetition, restating the prompt multiple times with no progression toward an actual explanation. The response lacked structure, factual content, and semantic coherence.

Sinhala Model Response: ශ්‍රී ලංකාවේ අධ්‍යාපන ක්‍රමය ව්‍යුහගත වූ පද්ධතියක් වන අතර එය සාමාන්‍යයෙන් විධිමත් අධ්‍යාපනය සහ ප්‍රාථමික අධ්‍යාපනය යන දෙකෙන් සමන්විත වේ. විධිමත් අධ්‍යාපනය සාමාන්‍යයෙන් උසස් පෙළ සහ වෘත්තීය පුහුණුව ඇතුළත් වන අතර, ප්‍රාථමික අධ්‍යාපනය සාමාන්‍යයෙන් පෙර පාසල සහ උසස් පෙළ ඇතුළත් වේ. ශ්‍රී ලංකාවේ අධ්‍යාපන ක්‍රමය ව්‍යුහගත වන්නේ විවිධ මට්ටම්වල අධ්‍යාපනය සඳහා විවිධ ආයතන ඇති බැවිනි. මෙයට රජයේ පාසල්, රජයේ උසස් අධ්‍යාපන ආයතන සහ පෞද්ගලික පාසල් ඇතුළත් වේ. මීට අමතරව, ශ්‍රී ලංකාවේ පුළුල් අධ්‍යාපන වැඩසටහන් සහ අධ්‍යාපනික සේවාවන් රාශියක් ඇත. මෙයට ජාතික අධ්‍යාපන වැඩසටහන්, අධ්‍යාපනික වැඩසටහන් සහ අධ්‍යාපනික සේවා ඇතුළත් වේ. සමස්තයක් වශයෙන්, ශ්‍රී ලංකාවේ අධ්‍යාපන ක්‍රමය විවිධ මට්ටම්වල

In contrast, our model produced a well-structured overview of Sri Lanka's education system, distinguishing between formal and primary education, and referencing government and private institutions. The response demonstrated logical flow and factual consistency.

LLM Evaluation Score:

- Base Model: 0.5/5
- Sinhala Model: 4.5/5

c: EXAMPLE 3: CULTURAL HERITAGE OF SRI LANKA

Prompt (Sinhala): ශ්‍රී ලංකාවේ සංස්කෘතික උරුමයන් මොනවාද?

Base Model Response: ශ්‍රී ලංකාවේ සංස්කෘතික උරුමයන් මොනවාද? (විවාදයක් අරඹමු) ශ්‍රී ලංකාවේ සංස්කෘතික උරුමයන් මොනවාද? (විවාදයක් අරඹමු) දෙමළ උරුමයන් පිළිබඳව අපේක්ෂාවන් ඇති අය තවමත් ප්‍රමාද වෙමින් සිටිති. අනුන් විසින් අපේක්ෂා කරනු ලබන අන්තර්ජාල පිටු ගනු ලබන්නේ අපේක්ෂාවන් ඇති කරනු ලබන දෙමළ උරුමයන් පිළිබඳව සංස්කෘතික උරුමයන් පිළිබඳව විවාදයක

The response deviated from the question, introducing unrelated commentary and failing to enumerate or explain Sri Lankan cultural heritage. The output lacked relevance and coherence.

Sinhala Model Response: ශ්‍රී ලංකාවේ පොහොසත් සංස්කෘතික උරුමය ශ්‍රී ලාංකික සංස්කෘතියේ සුවිශේෂී සහ විවිධාකාර අංග ඇතුළත් වේ. මෙම උරුමය පොහොසත් වන අතර සංස්කෘතික අත්දැකීම් සහ සම්ප්‍රදායන් ඇතුළුව රටේ සංස්කෘතික උරුමයේ විවිධ අංශ ඇතුළත් වේ. ශ්‍රී ලංකාවේ සංස්කෘතික උරුමයන් ඇතුළත් වන්නේ: සාහිත්‍යය: ශ්‍රී ලංකාවේ සාහිත්‍යය සියවස් ගණනාවක් පුරා විහිදී ඇති අතර එහි පොහොසත් ඉතිහාසයක් ඇත. ශ්‍රී ලාංකික සාහිත්‍යයේ සුවිශේෂී අංගයන් අතරට සාහිත්‍ය විවිධත්වය, පොහොසත් සම්ප්‍රදායන් සහ සාහිත්‍ය හුවමාරුව ඇතුළත් වේ. ශ්‍රී ලංකාවේ සාහිත්‍යයේ ප්‍රධාන කාණ්ඩ කිහිපයක් ඇත, එනම් සාම්ප්‍රදායික සාහිත්‍යය, නූතන සාහිත්‍යය සහ දෙමළ සාහිත්‍යය ඇතුළුව.

The model delivered a detailed and organized response, outlining multiple aspects of Sri Lanka's cultural heritage, including literature, music, and historical traditions. The answer demonstrated depth, contextual awareness, and fluent Sinhala generation.

LLM Evaluation Score:

- Base Model: 1/5
- Sinhala Model: 5/5

These qualitative examples reinforce the earlier observation that traditional metrics alone are insufficient for evaluating Sinhala language models. Although our model achieved lower ROUGE and BLEU scores, its real-world inference behavior exhibits significantly improved fluency, semantic correctness, and task relevance.

2) QUALITATIVE EVALUATION – HUMAN REVIEW

After each stage, we prompted the model using a sample set of questions and evaluated its response. This qualitative review revealed a clear improvement in Sinhala language generation across each stage. The model's ability to understand, respond, and maintain fluency in Sinhala showed notable progression, validating the effectiveness of our training pipeline.

V. DISCUSSION

This work demonstrates that adapting a large language model to a low-resource language like Sinhala is feasible, even under constrained computational settings, provided that each stage of the pipeline is handled with care. Several key insights emerged from our experiments, particularly around tokenizer design, continued pre-training, and parameter-efficient fine-tuning.

A. KEY FINDINGS BY PIPELINE STAGE

1) TOKENIZER FINE-TUNING

Initial experiments demonstrated that pretrained tokenizers produced fragmented or incoherent token sequences for Sinhala leading to unstable and nonsensical model outputs. Naively retraining or modifying the tokenizer also proved harmful, often disrupting the model's learned representations.

To address this, we introduced a custom Byte-Level BPE tokenizer that strictly preserved the configuration of the pre-trained model while extending its vocabulary. Approximately 35,000 Sinhala tokens were added, accounting for roughly 30% of the final vocabulary. This approach significantly

improved token alignment with Sinhala's script and morphology while preserving the original embedding structure. As a result, Sinhala generation quality improved without degrading the model's English performance, confirming that careful tokenizer extension is both necessary and sufficient for stable multilingual adaptation.

2) CONTINUED PRE-TRAINING

Continued pre-training played a central role in enabling meaningful Sinhala generation. Training exclusively on Sinhala data initially appeared promising but quickly resulted in catastrophic forgetting, with the model losing much of its English and general-world competence. This failure highlighted the risks of narrow-domain adaptation in low-resource settings.

We mitigated this by adopting a domain-adaptive pre-training strategy that mixed 40% data from `JeanKaddour/minipile`, which closely resembles the original training distribution, with 60% raw Sinhala text. This balance proved critical. It allowed the model to internalize Sinhala-specific vocabulary and structure while anchoring its broader linguistic and reasoning capabilities. The transition from unstable, mixed-language outputs to consistently fluent Sinhala responses provided strong evidence that this staged, balanced approach is effective for low-resource language adaptation.

3) LoRA FINE-TUNING

The final stage, LoRA-based instruction fine-tuning, refined the model's task-following and response quality. Full fine-tuning was not feasible given our resource constraints, and early attempts confirmed that it increased the risk of overfitting. LoRA offered a practical alternative by enabling targeted updates through low-rank adapters while keeping the majority of the model frozen.

Fine-tuning on a mixed-language Instruction Fine-Tuning dataset, with over 90% Sinhala content, proved essential. When trained on Sinhala-only instructions, the model again exhibited signs of catastrophic forgetting. The mixed setup preserved bilingual performance while substantially improving coherence, contextual relevance, and instruction adherence in Sinhala. Importantly, these gains were achieved with minimal additional compute, underscoring the effectiveness of parameter-efficient methods for low-resource language modeling.

VI. CONCLUSION

This work presented a practical and systematic pipeline for adapting a large language model to Sinhala, a low-resource language that remains largely underserved by modern AI systems. Through careful tokenizer extension, balanced continued pre-training, and parameter-efficient instruction fine-tuning, we were able to substantially improve the model's ability to generate coherent, contextually relevant Sinhala text. Importantly, these gains were achieved under strict data and compute constraints, demonstrating that meaningful progress is possible even outside large-scale industrial settings.

Beyond the immediate performance improvements, this study highlights a broader insight: low-resource language modeling is less about brute-force scaling and more about disciplined design choices at each stage of the pipeline. Tokenizer configuration, data balance, and constrained adaptation strategies proved to be decisive factors in preserving model stability while extending linguistic coverage.

Future work will focus on expanding both the breadth and diversity of Sinhala data, enabling longer and more extensive pre-training runs with higher-capacity hardware, and strengthening cross-lingual robustness without sacrificing Sinhala fluency. We also plan to evaluate the model on downstream NLP tasks and explore real-world deployments, including assistive technologies and conversational systems designed specifically for Sinhala-speaking users. Ultimately, we hope this work contributes toward a more inclusive AI ecosystem, where linguistic representation is treated as a fundamental requirement rather than an afterthought.

ACKNOWLEDGMENT

Portions of the manuscript text were refined using an AI language model (ChatGPT, OpenAI) for language clarity and grammar. The authors reviewed and verified all content.

REFERENCES

[1] U. Sandaruwan, N. Fonseka, P. Salwathura, D. M. S. Alwis, C. Thembiliyagoda, T. Amarathunga, C. Wanigasekara, and L. Velmanickam, "An approach to training and fine-tuning large language models for low-resource languages," in *Proc. IEEE World AI IoT Congr. (AIIoT)*, Seattle, WA, USA, May 2025, pp. 1061–1066, doi: 10.1109/AIIOT65859.2025.11105278.

[2] *Introducing GPT-5*, Accessed: Nov. 10, 2025. [Online]. Available: https://openai.com/index/introducing-gpt-5/

[3] *Claude 3.7 Sonnet and Claude Code*, Accessed: Mar. 14, 2026. [Online]. Available: https://www.anthropic.com/news/claude-3-7-sonnet

[4] (2025). *Sri Lanka Population (2025)—Worldometer*. [Online]. Available: https://www.worldometers.info/world-population/sri-lanka-population/

[5] R. Jayakody, and G. Dias, "Performance of recent large language models for a low-resourced language." In *IEEE Int. Conf. Asian Lang. Process. (IALP)*, pp. 162–167, 2024.

[6] T. B. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901.

[7] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 5485–5551, 2020.

[8] D. H. Hagos, R. Battle, and D. B. Rawat, "Recent advances in generative AI and large language models: Current status, challenges, and perspectives," *IEEE Trans. Artif. Intell.*, vol. 5, no. 12, pp. 5873–5893, Dec. 2024.

[9] J. Parmar, S. Satheesh, M. Patwary, M. Shoeybi, and B. Catanzaro, "Reuse, Don't retrain: A recipe for continued pretraining of language models," 2024, *arXiv:2407.07263*.

[10] A. V. Robins, "Catastrophic forgetting, rehearsal and pseudo rehearsal," *Connect. Sci.*, vol. 7, Apr. 1995, Art. no. 123146.

[11] R. French, "Catastrophic forgetting in connectionist networks," *Trends Cognit. Sci.*, vol. 3, no. 4, pp. 128–135, Apr. 1999.

[12] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A comprehensive overview of large language models," 2023, *arXiv:2307.06435*.

[13] G. Chen, F. Liu, Z. Meng, and S. Liang, "Revisiting parameter-efficient tuning: Are we really there yet?" 2022, *arXiv:2202.07962*.

[14] A. Köksal, M. Thaler, A. Imani, A. Üstün, A. Korhonen, and H. Schütze, "MURI: High-quality instruction tuning datasets for low-resource languages via reverse instructions," 2024, *arXiv:2409.12958*.
[15] Accessed: Nov. 10, 2025. [Online]. Available: https://huggingface.co/datasets/Pamzyy/BBC_science
[16] Accessed: Nov. 10, 2025. [Online]. Available: https://huggingface.co/datasets/Pamzyy/Vidusara_science2
[17] Accessed: Nov. 10, 2025. [Online]. Available: https://huggingface.co/datasets/Pamzyy/Anroidwadakaroyo
[18] Accessed: Nov. 10, 2025. [Online]. Available: https://huggingface.co/datasets/Pamzyy/testIFT
[19] Accessed: Nov. 10, 2025. [Online]. Available: https://huggingface.co/datasets/Pamzyy/OScar_sinhala
[20] Accessed: Nov. 10, 2025. [Online]. Available: https://huggingface.co/datasets/Pamzyy/Translated_Hindi_dataset_tosi1
[21] Accessed: Nov. 10, 2025. [Online]. Available: https://huggingface.co/datasets/Pamzyy/translatedhindi2
[22] Accessed: Nov. 10, 2025. [Online]. Available: https://huggingface.co/datasets/Pamzyy/Wikihow_sinhala
[23] Accessed: Nov. 10, 2025. [Online]. Available: https://huggingface.co/datasets/Pamzyy/anudesh_sinhala
[24] Accessed: Nov. 10, 2025. [Online]. Available: https://huggingface.co/datasets/Pamzyy/merged_final1

P. K. UDITH I. SANDARUWAN received the B.Sc. degree (Hons.) in electrical engineering with a minor in pattern recognition from the Department of Electrical Engineering, University of Moratuwa, Sri Lanka, in 2025. He was one of the authors of the paper titled "An Approach to Training and Fine-Tuning Large Language Models for Low-Resource Languages" (IEEE AIIoT, 2025) in Seattle, USA. His research interests include large language models, multi-agent architectures, and machine learning.

NIMESH M. A. FONSEKA received the B.Sc. degree (Hons.) in electrical engineering from the Department of Electrical Engineering, University of Moratuwa, Sri Lanka, in 2025. He was one of the authors of the paper titled "An Approach to Training and Fine-Tuning Large Language Models for Low-Resource Languages" (IEEE AIIoT, 2025) in Seattle, USA. His research interests include artificial intelligence, automation, and embedded systems.

PAMITH C. SALWATHURA (Student Member, IEEE) received the B.Sc. degree (Hons.) in electrical engineering with a minor in pattern recognition from the Department of Electrical Engineering, University of Moratuwa, Sri Lanka, in 2025. He was one of the authors of the paper titled "An Approach to Training and Fine-Tuning Large Language Models for Low-Resource Languages" (IEEE AIIoT, 2025) in Seattle, USA. His research interests include machine learning, embedded systems, and artificial intelligence.

DON SANJEEWA ALWIS received the B.Sc. degree in information technology from Sri Lanka Institute of Information Technology (SLIIT), Sri Lanka. He is currently the Founder and the Director of Decryptogen, where he leads AI-driven enterprise solutions, decentralized AI training systems, and cloud transformation initiatives. His research spans decentralized large language model (LLM) training, blockchain-integrated AI systems, distributed machine learning architectures, and AI governance. He has authored and co-authored research in decentralized LLM frameworks and blockchain-based incentive mechanisms for AI ecosystems. With industry experience across Europe, USA, and Australia, he has led large-scale AI and cloud transformation projects in finance, healthcare, education, and logistics. His work focuses on advancing scalable, transparent, and trustworthy AI systems through the integration of blockchain and cloud-native technologies.

CHATHURA WANIGASEKARA (Senior Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from the University of Peradeniya, Sri Lanka, in 2013, and the M.E. degree in electrical and electronic engineering and the Ph.D. degree in control engineering from The University of Auckland, New Zealand, in 2016 and 2020, respectively.

From 2020 to 2022, he was a Postdoctoral Researcher with the Centre for Industrial Mathematics, University of Bremen, Germany. Since 2022, he has been with German Aerospace Centre, Institute for the Protection of Maritime Infrastructures, Bremerhaven, Germany. He is currently with the Institute of Maritime Technologies and Propulsion Systems, Geesthacht, Germany. He has been a Lecturer with the University of Bremen, since 2023. He is the author and co-author of more than 70 scientific publications in various reputed journals and conferences. His current research interests include nonlinear system identification and control, machine learning, and networked control systems.

LOGEESHAN VELMANICKAM (Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from the University of Peradeniya, Sri Lanka, in 2014, and the Ph.D. degree from North Dakota State University, Fargo, ND, USA, in 2019. He is currently a Senior Lecturer with the Department of Electrical Engineering, University of Moratuwa, Sri Lanka. He has published several research papers in leading international conferences and journals. His research interests include power systems, AI and the IoT applications, instrumentation, smart sensor development, and lab-on-a-chip development. He has received numerous awards for his research contributions.