Advancing Amharic Text Summarization with a Tailored Parameter-Efficient Fine-Tuning Technique

By

Dagim Melkie Haile

Supervised by: Dr. Fantahun Bogale

SCHOOL OF INFORMATION TECHNOLOGY AND EGINEERING – SiTE

College of Technology and Built Environment

ADDIS ABABA UNIVERSITY

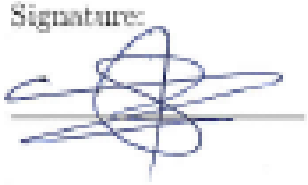
August, 2025

Approval

This is to certify that this thesis titled "Custom PEFT Module for an Improved Amharic Text Summarization" is prepared by Dagim Melkie Haile and submitted in partial fulfillment of the thesis-option requirements for the Degree of Master of Science in Artificial Intelligence at School of Information Technology and Engineering, Addis Ababa Institute of Technology.

Name:	Signature:	Date:

(Advisor)

Name:	Signature:	Date:
DR. MESFIN ABEBE		13/10/25

(External Examiner)

Name:	Signature:	Date:

(Internal Examiner)

Name:	Signature:	Date:

(Chairperson)

Abstract

While recent progress in Large Language Models (LLMs) has revolutionized the field of Natural Language Processing (NLP), applying these models to low-resource languages such as Amharic presents considerable difficulties. Key obstacles include the scarcity of available data and the intensive computational cost associated with conventional fine-tuning methods. To overcome these issues, this thesis introduces a specialized parameter-efficient fine-tuning (PEFT) framework developed specifically for Amharic text summarization. This new framework combines a dynamic low-rank adaptation component (DyLoRA-Amharic) with an adaptive activation method (AdaptAmharic), which work together to improve the model's flexibility and optimize its resource allocation during training.

The methodology involves injecting these custom modules into the mT5-small encoder–decoder architecture, allowing dynamic adjustment of DyLoRA-Amharic ranks and AdaptAmharic activation levels based on gradient signals. A joint optimization objective incorporating regularization terms for both rank and activation was employed to manage model complexity and ensure training stability. Comparative experiments were conducted against standard PEFT LoRA and Houlsby Adapter baselines on a curated Amharic summarization dataset.

Experimental results demonstrate that the proposed DyLoRA-Amharic and AdaptAmharic framework significantly outperforms the baselines across ROUGE, BLEU, and BERTScore metrics, achieving the lowest evaluation loss. Specifically, it improved ROUGE-L by 30.5% and BLEU by 52.4% over the strongest baseline. This superior performance validates the efficacy of a densely injected, dynamic, and regularized architecture, challenging the conventional emphasis on maximal sparsity in PEFT. While the framework utilizes a higher proportion of trainable parameters (13.42%) compared to the baselines, this trade-off is justified by the substantial performance gains.

This research contributes to advancing PEFT methodologies for low-resource NLP, providing a robust and adaptable solution for Amharic text summarization. The findings lay a foundation for developing more efficient and effective LLMs for diverse and linguistically underrepresented communities.

Keywords: Parameter-Efficient Fine-Tuning, PEFT, Low-Resource NLP, Amharic, Text Summarization, Dynamic LoRA, LoRA, Dynamic Masking, Adapter.

Acknowledgments

I am profoundly grateful to my advisor, Dr. Fantahun Bogale, for his invaluable guidance, encouragement, and intellectual insight throughout the course of this study. His patience, constructive feedback, and continuous support have been instrumental in shaping the direction and quality of this thesis.

My sincere appreciation also goes to the School of Information Technology and Engineering (SITE) and the Addis Ababa Institute of Technology (AAiT) for providing the academic environment and resources necessary to conduct this research.

I would also like to extend my heartfelt thanks to my friends at the Artificial Intelligence Department, whose discussions, technical advice, and collaboration have greatly enriched my understanding of this work.

Special thanks are due to the Open Source and Hugging Face communities, whose tools and documentation made it possible to experiment with advanced models such as mT5, LoRA, and adapters for Amharic text summarization.

Finally, I wish to acknowledge all those who contributed directly or indirectly to the success of this research. Your support, in its many forms, is sincerely appreciated and will always be remembered.

Dedication

This work is dedicated to everyone who checked in; Your words mattered more than you know.

Table of Contents

Abstract ii
Acknowledgments iii
Dedication iv
List of Abbreviations viii
List of Figures ix
List of Tables x

1 Introduction **1**
1.1 Motivation 2
1.2 Statement of the Problem 2
1.3 Research Question 3
1.4 Objective 3
1.4.1 General Objective 3
1.4.2 Specific Objective 3
1.5 Significance 4
1.6 Scope and Delimitation 4
1.6.1 Scope 4
1.6.2 Delimitations 5
1.7 Contribution 5
1.7.1 DyLoRA-Amharic 5
1.7.2 AdaptAmharic 6
1.7.3 Joint Optimization Objective 7
1.7.4 Research Contributions Summary 8
1.8 Thesis Structure 9

2 Literature Review and Related works **10**
2.1 Background 10
2.2 Parameter-Efficient Fine-Tuning (PEFT) Techniques 11
2.2.1 Overview of Parameter-Efficient Fine-Tuning (PEFT) 11
2.2.2 Low-Rank Adaptation (LoRA) 12
2.2.3 Adapters 14
2.2.4 Other PEFT Techniques 16
2.3 Hybrid and Dynamic PEFT Approach 17

2.3.1 Concept and Overview . . . 17
2.3.2 Implementation and Empirical Studies . . . 19
2.4 Summary of PEFT Approaches and Challenges . . . 20
2.4.1 Summary of PEFT Approaches . . . 20
2.4.2 Challenges and Opportunities . . . 20
2.5 Existing Approaches for Amharic Summarization . . . 25
2.6 Related Works . . . 26
2.6.1 Similar Efforts in Amharic Summarization . . . 27
2.6.2 Gap and Contribution . . . 29

3 Methodology **31**
3.1 Overall Research Design . . . 31
3.2 Proposed PEFT Module . . . 32
3.3 Data Collection and Preparation . . . 37
3.4 Base Pre-trained Language Model . . . 40
3.5 Evaluation Metrics . . . 41
3.6 Baselines . . . 42

4 Experiments and Results **43**
4.1 Experimental Setup . . . 43
4.1.1 Experimental Setup . . . 43
4.1.2 mT5-small + LoRA Baseline . . . 43
4.1.3 mT5-small + Adapter Baseline . . . 45
4.1.4 mT5-small + DyloraAmharic + AdaptAmharic . . . 46
4.2 Results . . . 50
4.2.1 Generation Quality and Predictive Accuracy . . . 50
4.2.2 Semantic Similarity Evaluation (BERTScore) . . . 51
4.2.3 Advanced Semantic and Character-Level Evaluation . . . 51
4.2.4 Parameter Efficiency . . . 52
4.2.5 Training Dynamics and Convergence . . . 52
4.2.6 Dynamic Module Behavior . . . 54
4.2.7 Training Progress and Regularization Dynamics . . . 57
4.3 Discussion . . . 58
4.3.1 Interpretation of Performance Differences . . . 58
4.3.2 Efficacy of a Regularized Dynamic Architecture . . . 59
4.3.3 Architectural Implications: Dense vs. Sparse Adapters . . . 59
4.3.4 Predictive Loss vs. Generation Quality . . . 60
4.3.5 Semantic Evaluation Insights . . . 60
4.4 Ablation Experiment: Evaluating Module Contributions . . . 61

4.5 Limitations 63

5 Conclusion **65**

References **65**

Appendices **70**

A Training Algorithm **70**

List of Abbreviations

AI	Artificial Intelligence
BLEU	Bilingual Evaluation Understudy
DSRP	Design Science Research Process
FFN	Feed-Forward Network
GPT	Generative Pre-training Transformer
LLMs	Large Language Models
LoRA	Low-Rank Adaptation
mBERT	Multilingual BERT
Meteor	Metric for Evaluation of Translation with Explicit Ordering
MHA	multi-head attention
mT5	Text-to-Text Transfer Transformer
mT5	Multilingual Text-to-Text Transfer Transformer
NLP	Natural Language Processing
NN	Neural Network
PEFT	Parameter-Efficient Fine-Tuning
PLMs	Pretrained Language Models
ReLU	Rectified Linear Unit
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
UNIPELT	Unified Framework for Parameter-Efficient Language Model Tuning

List of Figures

2.1 Basic Transformer block 13
2.2 LoRA reparameterization. 14
2.3 Architecture of the adapter module and its integration with the Transformer. 15
2.4 Prompt tuning. 17
2.5 Illustration of UNIPELT. 18
2.6 Performance of different methods on the XSum summarization task. . . 24

3.1 Histogram of token counts 39

4.1 Training and validation loss curves for the three setups. 53
4.2 DyLoRA-Amharic rank evolution for selected encoder and decoder layers. 54
4.3 AdaptAmharic activation value evolution for selected encoder and decoder layers. 55
4.4 Grad norm and EMA grad norm value evolution for selected encoder and decoder layers. 56
4.5 Training and validation loss curves for the Ablation Experiment. 62

List of Tables

2.1 Summary of Key PEFT Methods (Part 1 of 3: Adapter, LoRA, Prefix Tuning) . . . 21
2.2 Summary of Key PEFT Methods (Part 2 of 3: Prompt-Tuning, BitFit, UniPELT) . . . 22
2.3 Summary of Key PEFT Methods (Part 3 of 3: Unified View, DyLoRA) . . . 23
2.4 Summary of Related Works . . . 30

3.1 Descriptive Statistics of the Amharic-3 Dataset . . . 39
3.2 Train/Validation/Test Split for the Amharic-3 Dataset . . . 40

4.1 Comparative Performance Metrics on the Test Set . . . 50
4.2 Average BERTScore for Sampled Summaries . . . 51
4.3 COMET and chrF++ Scores for Sampled Summaries . . . 51
4.4 Comparison of Trainable Parameters . . . 52
4.5 Training Progress per Epoch (Custom Dynamic Adapter) . . . 57
4.6 Model Parameters and Training Time . . . 61
4.7 Model Performance Metrics . . . 62

Chapter 1

Introduction

Historically, the field of Natural Language Processing (NLP) has been characterized by a strong focus on English, which has resulted in a lack of resources and representation for many other world languages. This situation is now changing as multilingual NLP becomes a more prominent area of research. Two major factors are driving this evolution: the creation of powerful Pretrained Language Models (PLMs) like Multilingual BERT (mBERT), Generative Pre-training Transformer (GPT), and the expansion of multilingual data resources, including the Universal Dependencies treebanks[1, 2].

A major breakthrough in multilingual NLP is the ability of these PLMs to facilitate cross-lingual transfer learning, enabling knowledge acquired in one language to be effectively applied to another—even without explicit cross-lingual supervision. This phenomenon has transformed multilingual NLP by allowing models to generalize across languages, even in cases where direct training data is unavailable [3]. While this advancement has significantly expanded language coverage, the benefits remain unevenly distributed. Many languages, particularly those with limited digital resources, continue to lag behind English in performance and representation, underscoring the need for further innovation in multilingual model design and adaptation strategies[4].

To bridge this gap, researchers have explored Parameter-Efficient Fine-Tuning (PEFT) as a promising approach to adapting Large Language Models (LLMs) for low-resource languages such as Amharic. Unlike traditional fine-tuning—which requires extensive datasets and computational power— PEFT methods, including Low-Rank Adaptation (LoRA) [5], Adapter layers [6], and Prefix Tuning [7], allow for efficient model customization without modifying core parameters. These techniques enhance accessibility and make it feasible to tailor LLMs to underrepresented languages.

Despite their effectiveness, existing PEFT methods often struggle to fully capture the morphological richness and syntactic complexity of languages like Amharic [8, 5]. To overcome these challenges, this research introduces a novel PEFT framework that integrates low-rank adaptation mechanisms with non-linear transformations inspired by Adapter modules. Additionally, by incorporating dynamic parameter selection based

on task complexity, this approach enhances adaptability and scalability, demonstrating superior performance over static PEFT techniques such as LoRA and fixed Adapters in low-resource and morphologically rich language settings.

1.1 Motivation

Among the reasons for conducting this research includes improving NLP accessibility through parameter efficiency. Fine-tuning LLMs typically requires significant computational resources, which can be a barrier in resource-limited settings. PEFT techniques like LoRA or Adapters help to reduce these demands, making advanced NLP more accessible and cost-effective for applications in Amharic. This can directly benefit local institutions or enterprises that may not have access to extensive computational infrastructure. In addition, developing Amharic summarization capabilities is also valuable for the Artificial Intelligence (AI) and NLP community, as it contributes to the push for more inclusive, multilingual NLP. Effective fine-tuning of LLMs for Amharic not only enhances machine summarization for the language but could provide transferable insights into other low-resource languages as well, especially in Ethiopia. Further more, this research may help bridge the performance gap in NLP between high- and low-resource languages, demonstrating effective ways to adapt transfer learning to unique linguistic and contextual challenges in Amharic. Insights from this study could be useful for refining models for other morphologically complex, low-resource languages.

1.2 Statement of the Problem

Despite the rapid advancements in Parameter-Efficient Fine-Tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA) and Adapters, significant challenges persist in effectively adapting Large Language Models (LLMs) to low-resource languages like Amharic[9]. While pre-trained models offer a foundational understanding, their inherent linguistic representations frequently fall short for Amharic, particularly in capturing intricate morphological variations without specific task-driven guidance [8].

Furthermore, fine-tuning efficiency remains a critical concern in low-resource environments. The scarcity of high-quality labeled data exacerbates the challenge of effectively applying current PEFT methods, often necessitating extensive data augmentation or complex transfer learning strategies to achieve even modest performance improvements [10]. Many existing PEFT techniques, developed predominantly on high-resource languages, tend to rely on fixed-capacity or sparsely injected modules, which may not provide sufficient adaptive flexibility or expressive power to adequately address the complexities of morphologically rich languages or the specific demands of tasks like abstractive summarization. Moreover, these static methods typically do not inherently account for dynamic,

layer-wise adaptation based on the evolving learning signals during fine-tuning, nor do they adequately address cross-domain adaptability [11].

To bridge these critical gaps and enable more effective adaptation of LLMs for low-resource languages, there is a pressing need for novel methodologies that:

1. Combine low-rank mechanisms with non-linear transformations to enhance flexibility in adapting pre-trained models.

2. Introduce dynamic parameter selection based on task complexity, enabling superior adaptation compared to static approaches like LoRA or fixed Adapters.

3. Explore dense, yet regularized, injection strategies: This challenges the conventional emphasis on maximal sparsity by investigating whether a more comprehensive, parallel injection of adaptive modules, coupled with effective regularization, can yield superior performance for complex tasks without sacrificing training stability.

Without such advancements, LLMs will continue to exhibit suboptimal performance in low-resource language settings and domain-specific tasks, undermining their potential for equitable and diverse applications. Addressing these issues is imperative to bridge the gap in linguistic and domain-specific adaptability, ensuring broader inclusivity and efficiency in LLM utilization[12].

1.3 Research Question

RQ1: How can a custom parameter-efficient fine-tuning method tailored for domain adaptation in low-resource languages like Amharic improve task-specific performance while maintaining computational efficiency?
RQ2: How does the proposed PEFT method perform in comparison to standard approaches (e.g., LoRA, Adapters) on benchmark datasets for tasks like summarization?

1.4 Objective

1.4.1 General Objective

Designing a custom task-specific modules inspired by methods like Adapters and LoRA but tailored for domain adaptation in low-resource languages like Amharic.

1.4.2 Specific Objective

- Develop Custom Modular Architectures
  - Design and implement task-specific custom modules that can effectively integrate with pre-trained language models for Amharic text summarization.

- Develop Baseline Modules
  - Design and implement baseline modules using Houlsby Adpater and LoRA.
- Evaluate the Module
  - Investigate and compare the performance of the proposed custom modules against existing methods, focusing on key metrics like summarization quality, model size, and computational efficiency.
- Dateset Preparation
  - Preparation or curation of a high-quality dataset relevant to the Amharic language and text summarization tasks

1.5 Significance

The rapid progress in LLMs has significantly advanced NLP, but most improvements are concentrated on high-resource languages. Low-resource languages like Amharic remain underrepresented, with existing models struggling due to limited training data, inefficient parameter adaptation, and high computational costs. Traditional fine-tuning approaches require updating millions of parameters, making them impractical for many researchers and organizations working with Amharic. To bridge this gap, there is a critical need for PEFT techniques tailored to Amharic text summarization, ensuring better performance without excessive computational overhead.

Recent advancements in LoRA and Adapter-based fine-tuning have shown promise in reducing the number of trainable parameters, but existing methods do not fully account for the structural differences in language representation across layers of deep models. Moreover, most PEFT techniques are developed and evaluated on high-resource languages, leaving uncertainty about their effectiveness in low-resource settings. This research aims to explore and extend PEFT techniques, focusing on how dynamic parameter selection and layer-wise adaptation can improve Amharic text summarization while maintaining computational efficiency. By addressing these gaps, this study contributes to making state-of-the-art NLP models more accessible, efficient, and effective for underrepresented languages.

1.6 Scope and Delimitation

1.6.1 Scope

1. Focus on Amharic Language: This research specifically targeted the Amharic language, exploring the unique linguistic features, challenges, and requirements for effective text

summarization in this low-resource language.
2. Selected LLMs: The study was evaluated and fine-tune a limited number of pre-existing LLMs that have shown promise for language processing tasks.
3. Evaluation Metrics: The study focused on specific performance metrics for text summarization, such as Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scores, which are commonly used to evaluate summarization quality. User studies may also be included to gauge the effectiveness of the summaries produced.

1.6.2 Delimitations

1. Exclusion of Other Languages: The study did not explore text summarization for other languages, even though similar techniques may be applicable. The focus remained strictly on Amharic to maintain depth and relevance.
2. Limited Dataset Scope: The research utilized specific datasets for training and evaluation, which may limit the generalizability of the results. The datasets chosen was based on their relevance to the Amharic language and text summarization tasks.
3. Technical Constraints: The study did not delve into hardware constraints or scalability issues associated with deploying the fine-tuned models in production environments. The focus will be on the theoretical and empirical outcomes of the fine-tuning process.
4. Non-exploration of Full LLMs Training: The research did not involve the training of LLMs from scratch, which requires vast computational resources and extensive datasets.
5. Research time frame: The research was limited to the time frame allocated for the thesis, which may restrict the number of experiments carried out or the breadth of comparative analyses with other techniques or models.

1.7 Contribution

The research aims to contribute significantly to the field of PEFT and its applications in low-resource languages, particularly Amharic. The expected contributions include:

1.7.1 DyLoRA-Amharic

DyLoRA-Amharic: Adaptive Rank Reduction

To optimize LoRA for low-resource languages like Amharic, we introduce a dynamic rank allocation mechanism. The DyLoRA-Amharic module dynamically adjusts its rank based on the Exponential Moving Average (EMA) of its gradient norm. For a given DyLoRA-Amharic module, the current_rank is updated as follows:

$$\text{current\_rank} = \min\left(\max\left(\lfloor \beta_{\text{scale}} \cdot \text{EMA\_grad\_norm} \rfloor, 1\right), \text{max\_rank}\right) \tag{1.1}$$

Where:

- EMA_grad_norm is the Exponential Moving Average of the Frobenius norm of the gradients of the DyLoRA-Amharic's `up` projection layer's weights (W_{up}).
- β_{scale} is a scaling hyperparameter (e.g., 0.1), controlling the sensitivity of rank adjustment to gradient magnitude.
- max_rank is the predefined maximum allowed rank (e.g., 16), ensuring the rank stays within a reasonable bound.
- $\lfloor \cdot \rfloor$ denotes the floor function, converting the raw rank to an integer.

The EMA_grad_norm is updated at each step using:

$$\text{EMA\_grad\_norm}_{\text{new}} = \alpha_{\text{ema}} \cdot \text{current\_step\_grad\_norm} + (1 - \alpha_{\text{ema}}) \cdot \text{EMA\_grad\_norm}_{\text{old}} \tag{1.2}$$

Where α_{ema} is the EMA decay factor (e.g., 0.1). This approach makes rank selection data-driven, allowing more capacity to layers exhibiting stronger learning signals.

1.7.2 AdaptAmharic

AdaptAmharic: Gradient-Weighted Adapter Activation

To improve adaptation for Amharic text summarization, we introduce a dynamic activation mechanism for adapter layers. The AdaptAmharic module controls its contribution via a continuous activation value, active, which is updated based on the Exponential Moving Average (EMA) of its gradient norm. The active value is computed using a sigmoid function:

$$\text{active} = \sigma\left(\gamma_{\text{gating}} \cdot \text{EMA\_grad\_norm} + \beta_{\text{activation}}\right) \tag{1.3}$$

Where:

- σ is the sigmoid function, ensuring the activation value is between 0 and 1.
- EMA_grad_norm is the Exponential Moving Average of the Frobenius norm of the gradients of the AdaptAmharic's `up` projection layer's weights.

- γ_{gating} is a scaling factor for the EMA gradient norm (e.g., 1.0), controlling the steepness of the sigmoid.
- $\beta_{\text{activation}}$ is a bias term (e.g., 0.1), shifting the sigmoid curve to influence the baseline activation level.

The EMA_grad_norm is updated at each step using:

$$\text{EMA\_grad\_norm}_{\text{new}} = \alpha_{\text{ema}} \cdot \text{current\_step\_grad\_norm} + (1 - \alpha_{\text{ema}}) \cdot \text{EMA\_grad\_norm}_{\text{old}} \tag{1.4}$$

Where α_{ema} is the EMA decay factor (e.g., 0.1). This smooth gating mechanism allows for nuanced and continuous control over the adapter's influence, leading to more stable and effective learning.

1.7.3 Joint Optimization Objective

To optimize both DyLoRA-Amharic and AdaptAmharic parameters jointly, we introduce a composite loss function that combines the main summarization task loss with regularization terms for the dynamic rank and activation values. This objective balances dynamic capacity adaptation with controlled module influence.

The total loss L_{total} is defined as:

$$L_{\text{total}} = L_{\text{task}} + \lambda_{\text{rank}} \sum_{l=1}^{N_{\text{DyLoRAAmharic}}} \text{current\_rank}_l + \lambda_{\text{activation}} \sum_{l=1}^{N_{\text{AdaptAmhric}}} \text{active}_l \tag{1.5}$$

Where:

- L_{task} is the main summarization loss (e.g., cross-entropy loss for sequence-to-sequence tasks).
- λ_{rank} is the regularization weight for DyLoRA-Amharic ranks (e.g., 1×10^{-5}). This term penalizes higher ranks, encouraging the model to maintain parameter efficiency.
- current_rank_l is the dynamically determined integer rank of the l-th DyLoRA-Amharic module.
- $N_{\text{DyLoRAAmharic}}$ is the total number of DyLoRA-Amharic modules injected across the model.
- $\lambda_{\text{activation}}$ is the regularization weight for AdaptAmharic activation values (e.g., 1×10^{-5}). This term encourages the model to use lower activation levels unless truly beneficial.

- active_l is the dynamically determined continuous activation value (between 0 and 1) of the l-th AdaptAmharic module.
- $N_{\text{AdaptAmharic}}$ is the total number of AdaptAmharic modules injected across the model.

This objective ensures that the model dynamically allocates resources (rank and activation) while being regularized against unnecessary complexity, promoting an efficient and stable fine-tuning process.

1.7.4 Research Contributions Summary

To address the challenges of low-resource NLP for Amharic text summarization, we propose several novel contributions by developing a custom Parameter-Efficient Fine-Tuning (PEFT) framework. This framework integrates two core components: DyLoRA-Amharic and AdaptAmharic, designed to enhance model flexibility and resource allocation within the mT5-small encoder-decoder architecture.

Our key contributions are summarized as follows:

1. **Gradient-Driven Dynamic Rank Allocation (DyLoRA-Amharic):** We introduce a novel mechanism for DyLoRA-Amharic where the effective rank of LoRA modules is dynamically adjusted based on the Exponential Moving Average (EMA) of the gradients flowing through their respective `up` projection layers. This approach allows for a data-driven allocation of model capacity, enabling layers that exhibit stronger learning signals to utilize higher ranks, while maintaining efficiency in less critical layers. Crucially, this dynamicity is achieved through **rank masking**, where the output of the fixed-size low-rank matrices is truncated to the dynamically determined rank, rather than physically resizing the matrices. This design ensures computational efficiency and avoids overhead associated with dynamic memory allocation.

2. **Gradient-Weighted Smooth Adapter Activation (AdaptAmharic):** We develop an adaptive activation mechanism for adapter layers, where the adapter's contribution is smoothly controlled by a continuous activation value. This value is dynamically determined by passing the EMA of the adapter's gradient norm through a sigmoid function. This "soft" gating allows for nuanced and continuous control over the adapter's influence, preventing abrupt changes and promoting stable optimization. This mechanism ensures that adapters contribute more significantly only when their gradients indicate high utility for the task.

3. **Full Model Injection and Joint Regularized Optimization:** Unlike many PEFT methods that focus on sparse injection (e.g., only attention layers), our

framework employs a **dense, parallel injection strategy**, augmenting every linear layer in both the encoder and decoder with our custom dynamic modules. This comprehensive injection provides a richer capacity for task-specific adaptation. Furthermore, we integrate both DyLoRA-Amharic's dynamic rank and AdaptAmharic's dynamic activation into a unified optimization framework. This joint optimization incorporates explicit regularization terms that penalize excessive rank and activation growth, thereby stabilizing the training of the densely injected dynamic network and preventing over-parameterization.

In summary, these contributions aim to advance the state of PEFT methodologies by demonstrating that a densely injected, dynamic, and regularized architecture can yield superior performance for complex tasks in low-resource settings, challenging the conventional emphasis on maximal sparsity. This ultimately drives innovation in the practical deployment of LLMs for diverse and linguistically underrepresented communities.

1.8 Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2, reviews existing literature and contextual studies that form the foundation for this work. Chapter 3, describes the methodological framework, including the model architectures and training strategies adopted. Chapter 4, presents the experiments conducted and the corresponding results and analysis. Chapter 5, summarizes the major findings, outlines the research contributions, and proposes potential directions for further exploration. Supplementary resources, such as algorithmic descriptions, are included in the appendices.

Chapter 2

Literature Review and Related works

The rapid growth of LLMs has revolutionized NLP tasks, enabling significant progress in applications such as machine translation, text generation, and summarization. Despite these advancements, fine-tuning LLMs is computationally expensive, particularly for languages with limited resources such as Amharic. This challenge has sparked interest in parameter-efficient fine-tuning techniques that allow for high-quality performance with reduced computational and memory demands [6]. In this literature review, we explore various approaches to parameter-efficient fine-tuning of LLMs, particularly in the context of low-resource languages like Amharic. We also discuss the current state of Amharic NLP research and summarize key techniques that supports improved text summarization for the language.

2.1 Background

The field of NLP has recently been revolutionized by the development of LLMs. These models, often pre-trained on massive amounts of text data to acquire general-purpose "knowledge," have demonstrated remarkable capabilities across a wide range of NLP tasks. This trend towards increasingly large models, such as GPT-3, has led to substantial gains in performance on many NLP tasks [13]. However, the fine-tuning of these LLMs for specific applications remains a significant challenge, particularly for languages with limited resources such as Amharic.

The traditional approach to adapting pre-trained language models for specific tasks has been full fine-tuning. This involves updating all the model's parameters using a task-specific supervised dataset. While effective, particularly for achieving high accuracy, full fine-tuning becomes increasingly challenging and inefficient as LLMs grow in size. The limitations of full fine-tuning include substantial computational demands, high storage

requirements, a greater risk of overfitting when training data is limited, and the necessity of large, task-specific datasets. These factors collectively hinder the practical application of large models, especially in resource-constrained research environments.

To overcome these challenges, recent research has explored PEFT strategies that update only a limited subset of model parameters during adaptation. This selective tuning approach retains most of the pre-trained weights, thereby lowering the overall memory footprint and training cost without sacrificing task performance.

Multiple frameworks have emerged under the PEFT paradigm, such as adapter-based tuning, prefix or prompt-based conditioning, low-rank decomposition (e.g., LoRA), and bias-only adaptation (BitFit). These approaches consistently demonstrate competitive or superior results compared to full fine-tuning, while training only a small portion of model parameters.

The effectiveness of PEFT highlights the potential for efficient adaptation of large pre-trained models. As LLMs continue to increase in size and are applied to a wider range of applications, efficient fine-tuning techniques are becoming crucial for their practical and scalable deployment.

2.2 PEFT Techniques

2.2.1 Overview of PEFT

A rapid growth in research has led to the publication of over a hundred PEFT papers in recent years, with comprehensive surveys summarizing the most widely adopted approaches, including LoRA, Adapters, BitFit, Compacter, Pre-fixtuning, and Soft-Prompts. These diverse methods are distinguished by the trade-offs they present across several key areas: memory consumption and parameter use, training velocity, the quality of the final model, and any subsequent overhead incurred during inference.

Considering that the Transformer architecture represents the largest class of neural networks ever developed, it follows that a significant portion of PEFT strategies have been specifically created for this model design[14].

The fundamental unit of the Transformer architecture is structured as a MHA mechanism followed by FFN, as detailed in Figure 2.1. Both the attention and fully-connected components integrate residual connections and Layer Normalization steps to significantly boost model training stability.

The attention mechanism can be described by the equation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

where:

- Q, K, V: Represent the Query, Key, and Value matrices, derived from the input to the attention layer.
- Q^T: The transpose of the Query matrix.
- d_k: The dimensionality of the key vectors.
- softmax: A function that normalizes the output of the dot product into a probability distribution.

The Feed-Forward Network (FFN) typically uses two linear transformations with a Rectified Linear Unit (ReLU) activation function, often represented as:

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{2.2}$$

where:

- $\text{ReLU}(x)$: The rectified linear unit activation function, which outputs the input if it's positive, and zero otherwise ($\text{ReLU}(x) = \max(0, x)$).
- W_1, b_1, W_2, b_2: Weight and bias matrices that transform the input.

In simpler terms, the attention mechanism calculates attention weights by comparing queries to keys and then uses these weights to combine values. The FFN performs a non-linear transformation on the output of the attention mechanism, further processing the information.

Specific Transformer model designs can differ, for example, by adding a cross-attention component in sequence-to-sequence networks or by employing Layer Normalization prior to the sublayers Pre-LN). Nevertheless, the majority of PEFT techniques designed for these models primarily focus on the fundamental MHA + FFN structure. This approach allows these methods to be easily adjusted to accommodate different architectural configurations.

2.2.2 LoRA

A standard NN is constructed with numerous dense layers that execute matrix multiplications. The corresponding weight matrices within these layers are typically full-rank. However, when fine-tuning for a new task, research by [15] suggests that pre-trained language models exhibit a small intrinsic dimension. This insight indicates that the models retain the ability to learn effectively, even if their parameters are projected randomly into a reduced subspace.

LoRA [5] is a reparameterization-based PEFT technique that has gained significant popularity. Instead of directly fine-tuning the dense weight matrices of a pre-trained

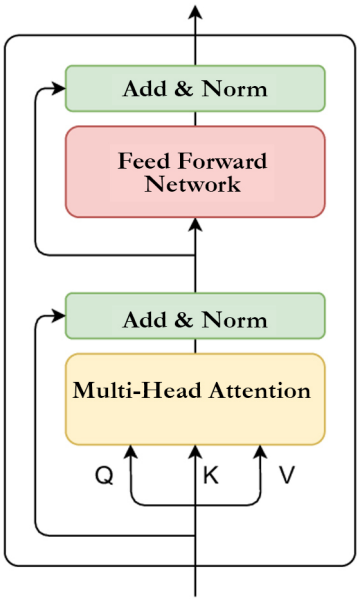


Figure 2.1: Basic Transformer block

model, LoRA introduces low-rank decomposition matrices into the layers of the Transformer architecture. As described in [5], a standard LoRA layer modifies the original weight matrix W_0 by adding a low-rank update ΔW:

$$W = W_0 + \Delta W = W_0 + AB \tag{2.3}$$

where $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times d}$ are low-rank matrices, and $r \ll d$ is the rank. During fine-tuning, only the parameters of the small, low-rank matrices A and B are trained, while the original pre-trained weights W_0 remain frozen. This significantly reduces the number of trainable parameters.

As mentioned in [16], LoRA is often applied to the query and value matrices in the multi-head attention (MHA) modules of the Transformer. One of the key advantages of LoRA is that the learned low-rank weights AB can be merged back into the original weight matrix W_0 during inference, resulting in no additional inference latency, Figure 2.2 visually represents the addition of these low-rank modules within a Transformer block.

When comparing LoRA to full fine-tuning and other PEFT methods experimental result demonstrated that LoRA can achieve performance comparable to or even better than full fine-tuning while training a significantly smaller number of parameters. In addition, LoRA scales effectively with larger models.

Nonetheless, [5] acknowledges limitations and suggests future research directions, including the need for more principled methods for rank selection, as the optimal rank 'r' is often determined empirically. A deeper theoretical understanding of why LoRA works could also lead to further improvements. However, LoRA remains a significant contribution to the field of PEFT. By introducing low-rank approximations for weight updates,

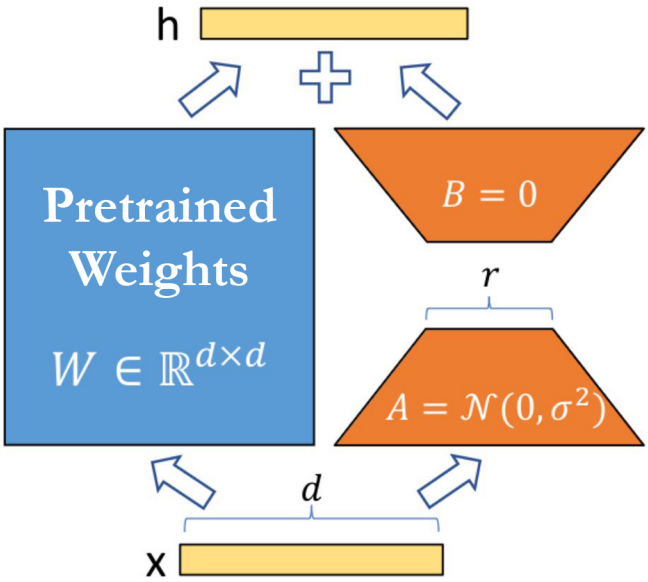


Figure 2.2: LoRA reparameterization.

it offers an efficient way to adapt LLMs for downstream tasks, addressing the limitations of full fine-tuning. The experimental results validate the effectiveness of the method and has paved the way for more accessible and scalable fine-tuning of large language models, particularly in resource-constrained scenarios.

2.2.3 Adapters

Adapter techniques [6] involve inserting small Neural Network (NN) modules, known as adapters, into each layer of the pre-trained model. These adapters typically have a bottleneck architecture, reducing the dimensionality of the input, processing it through a non-linear activation function (e.g., ReLU), and then projecting it back to the original dimensionality. Only the parameters of these newly introduced adapter modules are trained, while the original pre-trained weights are kept frozen. Figure 2.3 illustrates the insertion of adapter modules within Transformer blocks.

The Adapter module design is centered around two core characteristics: a minimal parameter count and near-identity initialization.

The requirement for the modules to be significantly smaller than the layers in the base network ensures that the overall model's size only increases gradually, even as new tasks are integrated. Furthermore, stable training of the adapted model necessitates the use of a near-identity initialization. By setting up the adapters this way, the original network's function is left undisturbed at the start of the training process. During the course of training, these modules are allowed to become active, potentially modifying the distribution of activations across the network. A final advantage is that these adapter

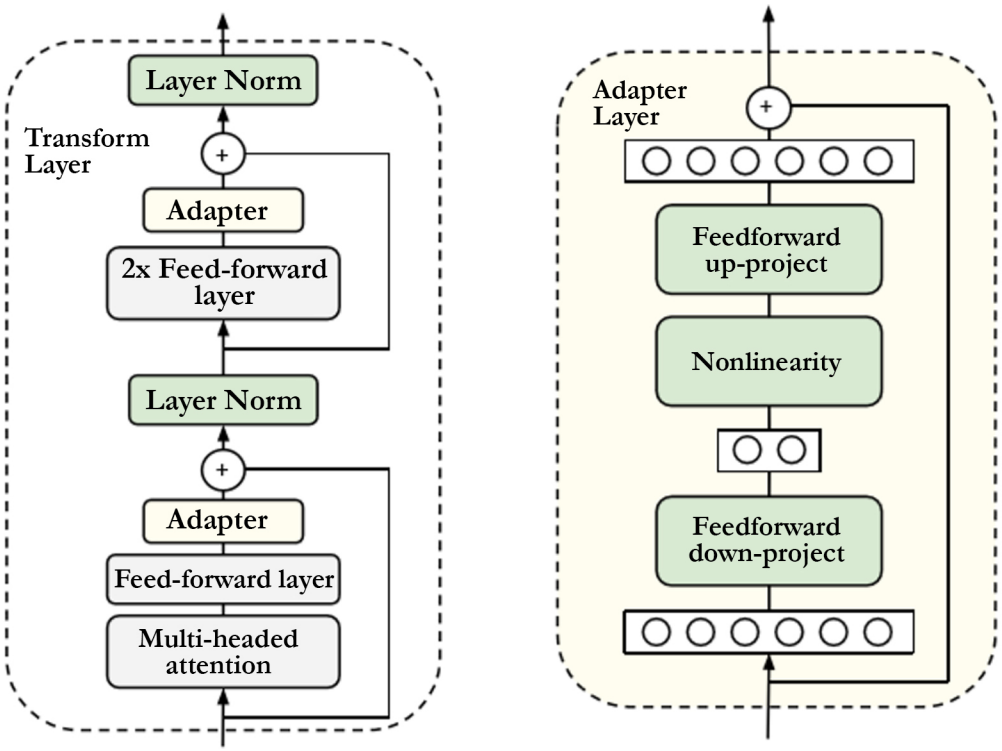


Figure 2.3: Architecture of the adapter module and its integration with the Transformer.

units can also be bypassed entirely when not needed for a specific task.

$$h' = h + W_{up} \cdot \sigma(W_{down} \cdot h) \tag{2.4}$$

Where:

- h is the input to the adapter.
- h' is the output of the adapter.
- $W_{down} \in \mathbb{R}^{d \times r}$ is the matrix that reduces the dimensionality from d to r.
- $W_{up} \in \mathbb{R}^{r \times d}$ is the matrix that restores the dimensionality from r to d.
- $\sigma(\cdot)$ is a non-linear activation function, such as ReLU or GELU.

In order to constrain the quantity of parameters being introduced, the adapter mechanism operates by initially mapping the input d-dimensional features to a smaller m-dimensional space. This is followed by the application of a nonlinearity, and the result is subsequently mapped back to d-dimensions. The overall count of new parameters incorporated into each layer, which accounts for biases, is calculated as $\mathbf{2md + d + m}$. By ensuring the bottleneck dimension $\mathbf{m}$ is significantly smaller than the original dimension $\mathbf{d}$ ($\mathbf{m} \ll \mathbf{d}$), the addition of parameters per task is effectively minimized.

Adapters offer flexibility in terms of where they can be inserted within the Transformer architecture, such as after the attention mechanism or the feed-forward network. Various adapter architectures exist, including sequential adapters, parallel adapters, and scaled adapters [12].

Adapters achieve performance comparable to full fine-tuning while training a significantly smaller number of parameters. The results from [6] highlight the parameter efficiency and effectiveness of Adapters.

While Adapters offer efficient training, they present certain limitations and challenges, including potential inference overhead, particularly in multi-task learning scenarios where multiple adapters are employed. Furthermore, the effectiveness of Adapters is influenced by design choices, necessitating careful tuning of their architecture and hyperparameters, such as the bottleneck dimension.

Although the introduction of small, modular Adapter layers enables effective adaptation of pre-trained models with a reduced number of trainable parameters, contributing to the development of more efficient and scalable techniques for adapting large language models to various downstream tasks. [6] suggests several directions for future research, including optimizing Adapter architecture by exploring different configurations, developing methods for dynamic Adapter selection for various inputs or tasks, and investigating the combination of Adapters with other parameter-efficient fine-tuning techniques.

2.2.4 Other PEFT Techniques

Besides LoRA and Adapters, several other parameter-efficient fine-tuning methods have been proposed [12, 16].

- Prefix-tuning (PF)[7]: This method prepends a sequence of trainable vectors to the input of the Transformer. Only these prefix vectors are optimized during fine-tuning. Prefix-tuning has shown effectiveness in generation tasks.

- Prompt tuning (PT) [17]: Similar to prefix-tuning, prompt tuning involves adding trainable ”soft” prompt tokens to the input embeddings. These tokens are learned to guide the pre-trained model to perform the desired task. Prompt tuning’s performance often scales with the size of the pre-trained model.

- BitFit[18]: This method focuses on fine-tuning only the bias parameters of the pre-trained model while keeping all weight matrices completely frozen. Although it is remarkably simple and involves updating a minimal number of parameters, BitFit has demonstrated surprisingly high performance on certain tasks. The advantage for multi-task scenarios is that each new task only requires the storage of the bias term vectors (which constitute less than 0.1% of the total parameters) along with the task-specific final linear classifier layer.

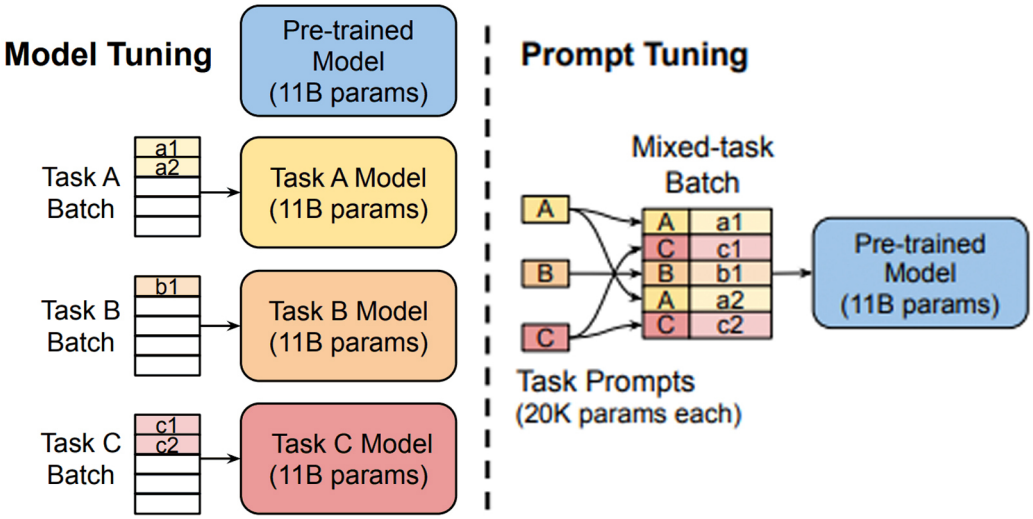


Figure 2.4: Prompt tuning.

2.3 Hybrid and Dynamic PEFT Approach

While individual PEFT methods have demonstrated effectiveness, the field is rapidly evolving, with a growing number of new techniques and tasks[19]. This proliferation makes it non-trivial to select the most appropriate method for a specific task. Furthermore, each standard PEFT method typically optimizes a separate set of parameters for each language or task, potentially leading to a large collection of fine-tuned checkpoints that need to be stored and deployed individually [20].

2.3.1 Concept and Overview

The concepts of hybrid and dynamic PEFT approaches represent a potential next step in addressing these limitations and further enhancing the efficiency and effectiveness of adapting large language models.

Hybrid PEFT can be conceptualized as the combination of two or more distinct PEFT methods within a single fine-tuning framework. The rationale behind such an approach is that different PEFT techniques might be effective at capturing different types of task-specific information or might interact synergistically to yield improved performance or efficiency. For instance, one could envision a hybrid approach that leverages the strengths of both prompt tuning (for task specification) and adapter layers (for feature adaptation). The unified view of PEFT methods, which reframes them as modifications to specific hidden states and defines design dimensions[12], provides a theoretical basis for understanding how different PEFT methods could be combined effectively.

Dynamic PEFT refers to approaches where the PEFT strategy itself, or its configura-

tion, is adapted during the fine-tuning process or based on the specific input or task context. This could involve dynamically activating or weighting different PEFT submodules, adjusting the number of tuned parameters, or even switching between different PEFT methods based on data characteristics or training progress. The Unified Framework for Parameter-Efficient Language Model Tuning (UNIPELT) framework, which incorporates different PEFT methods as submodules and learns to dynamically activate them using a gating mechanism[19],serves as a prime example and inspiration for dynamic PEFT. Such dynamic adaptation aims to enhance the model's ability to learn and generalize across diverse data and tasks without the need for extensive manual method selection.

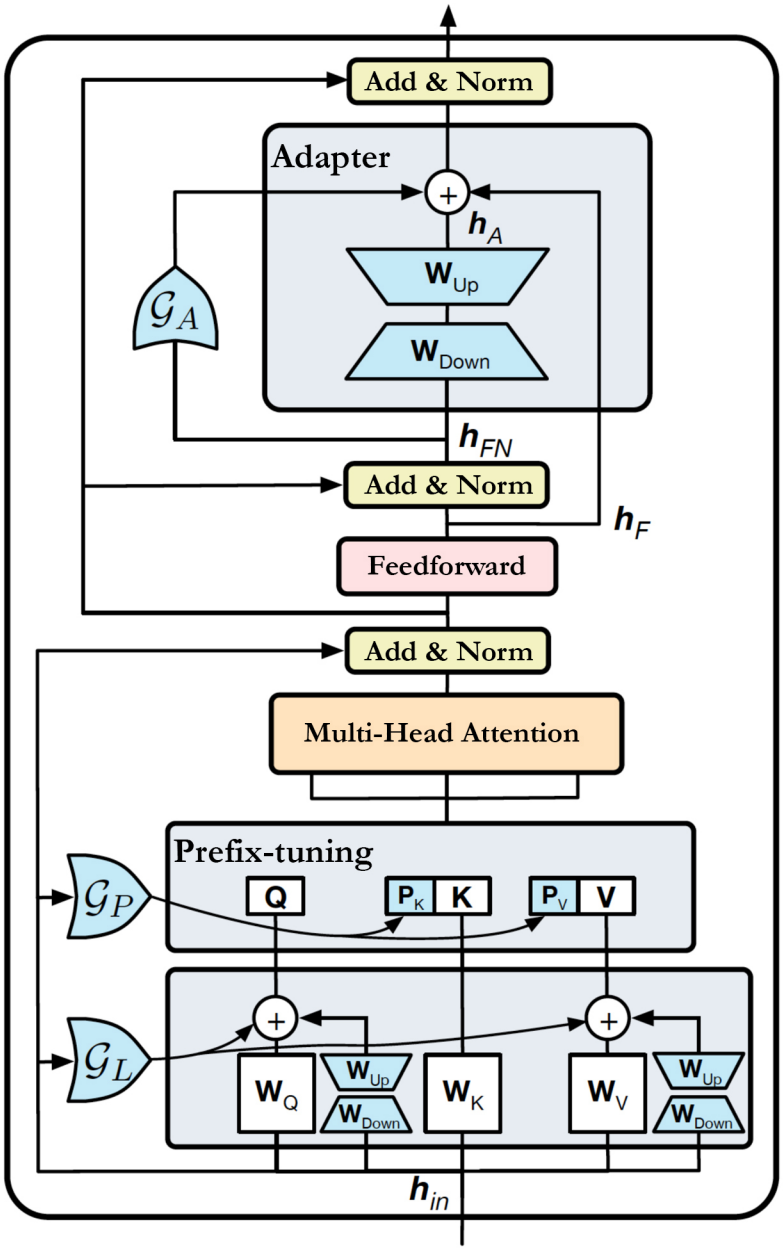


Figure 2.5: Illustration of UNIPELT.

The potential benefits of hybrid and dynamic PEFT approaches are manifold. By

combining the strengths of different methods, hybrid approaches could achieve higher performance than individual PEFT techniques. Dynamic approaches could offer greater flexibility and robustness by adapting to the specific demands of the data or task, potentially leading to more efficient learning and better generalization, especially in scenarios with diverse or low-resource data [20].

2.3.2 Implementation and Empirical Studies

Although we do not currently have numerous studies focusing solely on "hybrid and dynamic PEFT" as a distinct, mature area of research, what we find provide crucial building blocks and examples that illustrate the underlying principles and the current state of exploration in this direction.

The UNIPELT structure [19] marks a notable advancement toward achieving dynamic PEFT. It proposes a unified structure that integrates various Parameter-Efficient Language Model Tuning (PELT) methods as modular components. A key feature is its capacity to learn to dynamically enable the combination of submodules best suited for the specific task or dataset. This dynamic selection process is managed by a gating mechanism that is trained to allocate greater weight to submodules that positively contribute to performance. By fusing multiple PELT techniques while maintaining high efficiency, UNIPELT aims to achieve superior, more consistent performance without requiring manual selection of the best method. Empirical findings for UNIPELT indicate that it can surpass both traditional full fine-tuning and the performance of the individual submodules it combines, thereby highlighting the advantages of a dynamically adaptive PEFT strategy.

The unified perspective on parameter-efficient transfer learning presented in [12] also provides a foundational understanding for conceptualizing and potentially creating hybrid PEFT techniques. This research reconceives PEFT methods as specific modifications applied to the hidden states of pretrained models and establishes a set of design dimensions along which different methods can be compared. This framework helps identify relationships between existing approaches and could facilitate the transfer of design components across them, potentially leading to novel hybrid PEFT techniques. The possibility of designing new parameter-efficient fine-tuning methods by combining existing design decisions suggests significant potential for developing more effective hybrid models.

Among methods that can be considered dynamic approaches to PEFT, even if it don't explicitly use the term "hybrid" in the context of combining different methods from different paradigms, but rather combining or dynamically selecting parameters within the PEFT space is DyLoRA [21]. This method focuses on low-rank adapters, such as LoRA. It proposes a way to train low-rank adapter modules for multiple ranks simultaneously rather than training a single-rank adapter at a time. DyLoRA addresses the challenge of

selecting the optimal rank size for low-rank adapters, a problem that their performance is very sensitive to. It samples from a predefined random distribution during each iteration to truncate the up-projection and down-projection matrices in the LoRA objective. This concept introduces dynamism into the rank selection and training process for low-rank adaptation techniques.

Empirical studies comparing different mainstream PEFT methods, such as prompt tuning, prefix-tuning, LoRA, and adapter, provide insights into the strengths and weaknesses of individual techniques. For instance, [16] studies analyze their performance, convergence speed, and efficiency across a diverse set of NLP tasks. These comparisons serve as a motivation for exploring hybrid approaches that could potentially mitigate the limitations of individual methods. The exploration of the combinability of different PEFT methods [16] further directly investigates the potential of hybrid PEFT.

2.4 Summary of PEFT Approaches and Challenges

PEFT strategies are typically classified based on either their structural methodology or intended optimization goals. Structurally, these methods may involve augmenting the model with additional parameters, adjusting pre-existing ones, or transforming parameter representations. Functionally, they are designed to reduce memory usage, enhance storage compactness, and promote modular integration within larger systems[16].

2.4.1 Summary of PEFT Approaches

Given the diverse approaches that have emerged within PEFT research, a clear understanding of their underlying mechanisms, advantages, and limitations is essential. The following Tables 2.1, 2.2, 2.3 provide a synthesized overview of some of the key foundational PEFT techniques and prominent frameworks developed in this field. Due to the comprehensive nature of the comparison across multiple dimensions, the synthesis is presented in three parts for clarity and readability, categorizing methods by their core strategies and objectives.

2.4.2 Challenges and Opportunities

Despite the progress in PEFT research, the development and application of both the standard and ”hybrid and dynamic” approaches present several challenges and promising opportunities.

Table 2.1: Summary of Key PEFT Methods (Part 1 of 3: Adapter, LoRA, Prefix Tuning)

Feature	**Adapter** [6]	**LoRA** [5]	**Prefix Tuning** [7]
PEFT Category (Approach)	Additive (New Parameters)	Reparameterization	Additive (New Parameters)
Core Idea / Mechanism	Insert small bottleneck feed-forward layers within transformer blocks.	Represents weight matrix updates (ΔW) with a low-rank decomposition (AB).	Prepends a sequence of trainable continuous vectors (prefix) to the input embeddings.
Where Typically Applied	After attention and feed-forward layers within each transformer block.	Applied to weight matrices (e.g., attention weights: W_q, W_k, W_v, W_o).	Applied at the beginning of each transformer layer's input sequence.
Parameters Tuned	New small matrices (W_down, W_up) and biases per layer.	New low-rank matrices (A, B) added per selected weight matrix.	New continuous vector sequence trained.
Key Advantages	Modularity, Task-specific, Easy to combine/stack for multi-tasking; Can be inserted/removed easily.	High efficiency & speed; Low memory usage; Easy to implement; Does not increase inference latency much.	Good performance, especially for generation tasks; More expressive than Prompt Tuning due to per-layer prefixes; Does not add inference latency per token.
Key Limitations/Considerations	Can add inference latency per token (sequential processing). May require more parameters than some other methods.	Performance highly depends on choosing the correct rank (r) empirically; Less modularity than Adapters for some multi-task setups.	Can increase the total sequence length processed by each layer, potentially impacting efficiency; May require careful hyperparameter tuning for prefix length and initialization.
Main Goal of Paper	Propose a parameter-efficient alternative to full fine-tuning by adding small, modular layers to pre-trained models.	Introduce a highly parameter-efficient method for adapting large language models by injecting low-rank learnable matrices into existing weight matrices.	Propose training a short sequence of continuous vectors (a prefix) prepended to the input of each transformer layer for efficient fine-tuning on generation tasks.

Table 2.2: Summary of Key PEFT Methods (Part 2 of 3: Prompt-Tuning, BitFit, UniPELT)

Feature	**Prompt-Tuning** [17]	**BitFit** [18]	**UniPELT** [19]
PEFT Category	Additive (New Parameters)	Selective (Existing Subset)	Hybrid/Unified (Combines methods)
Core Idea / Mechanism	Learns a small set of continuous vectors (prompt) prepended to the input embedding.	Only fine-tunes the bias terms in the pre-trained model, freezes weights.	Unifies multiple PEFT methods (like Adapters, LoRA, Prefix Tuning) under one framework, learns to activate best using a gating mechanism.
Where Typically Applied	Prepended to the initial input embedding sequence.	Biases across layers (e.g., in linear layers, Layer Norm).	Can incorporate and apply various PEFT submodules at different strategic points within the model, guided by the gating.
Parameters Tuned	New continuous vector sequence trained.	Existing bias vectors of the pre-trained model are made trainable.	Parameters of the included PEFT submodules (e.g., Adapter weights, LoRA matrices) and gating mechanism parameters are tuned.
Key Advantages	Extreme parameter efficiency; Scales well with increasing model size (minimal task-specific params); Simple to implement; Can be used without modifying the core model architecture.	Extremely parameter efficient; Very simple to implement; Can be a strong baseline or effective in low-data regimes.	Can potentially combine and leverage the strengths of different PEFT methods; Learns the best approach dynamically; Offers flexibility.
Key Limitations	Less expressive/powerful than methods that modify internal layers or use larger additive components; Performance can be highly sensitive to initialization and hyperparameters.	Limited overall model capacity due to tuning only biases; May not perform as well as more expressive PEFTs on complex tasks or with ample data.	Increased complexity (managing multiple submodules and training the gating mechanism effectively); Requires a suitable gating strategy.
Main Goal of Paper	Demonstrate that training a minimal continuous prompt prepended to the input can achieve high performance comparable to other PEFT methods, especially at large model scales.	Show that simply making only the bias terms of a Transformer model trainable can be a surprisingly effective and extremely parameter-efficient fine-tuning method.	Develop a unified framework that can incorporate different existing PEFT methods as submodules and learn to activate the most effective ones based on the input or task via a gating mechanism.

Table 2.3: Summary of Key PEFT Methods (Part 3 of 3: Unified View, DyLoRA)

Feature	**Unified View** [19]	**DyLoRA** [21]
PEFT Category (Approach)	Framework (Categorization and Analysis)	Reparameterization (Dynamic)
Core Idea / Mechanism	Provides a framework to understand and classify PEFT methods based on design dimensions and how they modify hidden states, revealing connections.	Allows dynamic adjustment of LoRA's rank (r) during fine-tuning based on adaptation needs, potentially improving efficiency and adaptivity over fixed-rank LoRA.
Where Typically Applied	Analyzes where modifications/tuning happen (e.g., inputs, hidden states, weights, biases, outputs), across different layers.	Applied to weight matrices, similar to LoRA, but with the rank potentially changing per layer or based on fine-tuning progress.
Parameters Tuned	N/A (Focus is on analyzing existing methods, not tuning parameters of the framework itself during model adaptation).	New low-rank matrices (A, B) per selected weight matrix, with a dynamically changing number of parameters determined by the rank r.
Key Advantages	Helps understand fundamental differences and relationships between various PEFT methods; Provides a structured approach and guidance for designing new PEFT methods.	Can potentially achieve better adaptation and performance over fixed-rank LoRA by dynamically adjusting the model's adaptation capacity during training.
Key Limitations/Considerations	It's an analytical framework, not a deployable PEFT method itself; Requires effort to apply its insights to design or choose methods.	More complex than standard LoRA; Requires a mechanism or strategy to effectively determine the dynamic rank based on training progress or other factors.
Main Goal of Paper	Provide a systematic framework to classify, understand, and relate existing PEFT methods based on their underlying design choices.	Introduce a variant of LoRA that allows the rank of the low-rank matrices to change dynamically during the fine-tuning process, aiming for better adaptation efficiency.

Challenges

- Performance Gap: While PEFT methods can achieve comparable results to full fine-tuning on some benchmarks like GLUE [12, 18], a noticeable performance gap can still exist on more complex or high-resource tasks, such as summarization and machine translation [12]. Figure 2.6 demonstrates this performance difference on the XSum summarization task. As noted in [16], PEFT methods generally underperform compared to full fine-tuning under most cases

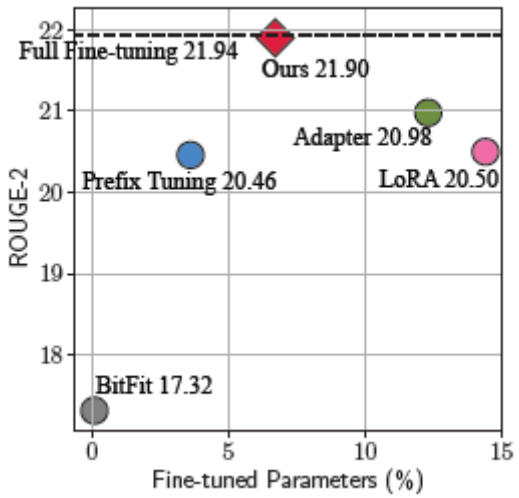


Figure 2.6: Performance of different methods on the XSum summarization task.

- Benchmark Diversity: Many PEFT methods are primarily evaluated on encoder-only models and relatively simpler generation benchmarks [12]. There is a need for more comprehensive evaluations across diverse tasks, model architectures (including encoder-decoder models like T5 [16] and BART [7], and datasets [16]. The authors of [16] advocate for reporting results on more diverse benchmarks to provide a more complete picture of their performance profile.

- Convergence and Stability: As highlighted in [16], the convergence rate of different PEFT methods can vary. For instance, Prefix-tuning can face convergence difficulties [16]. Understanding and improving the convergence and stability of PEFT methods is crucial for their practical application.

- Designing Effective Dynamic Mechanisms: For dynamic PEFT, a significant challenge lies in designing robust and efficient mechanisms for determining when and how to adapt the PEFT strategy. The gating mechanism in UNIPELT is a promising approach, but further research is needed to explore other dynamic adaptation strategies based on various factors such as data characteristics, task complexity, and training progress.

- Optimizing Hybrid Architectures: The design space for hybrid PEFT is vast, with numerous possible combinations of existing methods. Efficiently exploring this space and identifying optimal hybrid architectures for different types of tasks and data remains a significant challenge. Automated methods for searching and evaluating hybrid PEFT configurations could be valuable.

- Computational Overhead of Dynamic Approaches: While PEFT methods are generally efficient, dynamic approaches that involve evaluating and switching between different submodules or configurations might introduce some computational overhead. Balancing the benefits of dynamic adaptation with the potential increase in computational cost is an important consideration.

- Complexity of Implementation and Deployment: Implementing and deploying systems with dynamically configured PEFT methods can be more complex than using a single static PEFT technique. Developing user-friendly frameworks and tools that simplify the creation and management of hybrid and dynamic PEFT models is crucial for their wider adoption.

Opportunities

- Enhanced Performance and Efficiency: Hybrid PEFT offers the opportunity to achieve performance gains beyond what individual PEFT methods can offer by leveraging their complementary strengths. Dynamic PEFT can lead to more efficient resource utilization by adapting the model's capacity based on the demands of the input or task.

- Facilitating Learning in Low-Resource Settings: Hybrid and dynamic PEFT approaches could be particularly beneficial in low-resource scenarios by allowing the model to dynamically leverage different parameter-efficient strategies that are most effective with limited data.

- Model-as-a-Service and Efficient Deployment: The ability to dynamically adapt model behavior with a small number of parameters through hybrid and dynamic PEFT can lead to more efficient deployment scenarios, particularly in model-as-a-service platforms where serving a large number of customized models can be resource-intensive

2.5 Existing Approaches for Amharic Summarization

Text summarization is a critical task in Natural Language Processing, aiming to condense large volumes of text into concise and informative summaries, thereby mitigating

information overload. Applying text summarization techniques to low-resource languages like Amharic presents unique challenges, stemming primarily from the scarcity of publicly available large-scale datasets and limited linguistic resources compared to widely spoken languages. Despite these hurdles, researchers have explored various methodologies to develop effective text summarization systems for Amharic. Early efforts often focused on extractive techniques leveraging statistical [22] or graph-based methods, while more recent work has begun to explore the potential of neural network models[23] and fine-tuned pre-trained transformers. This section reviews the landscape of existing approaches for Amharic text summarization, examining their methodologies, contributions, and limitations in addressing the specific characteristics and resource constraints of the language.

Among the efforts focused on extractive techniques is [22] an Amharic text summarization model specifically tailored for news items posted on social media platforms like Twitter and Facebook. The primary challenge addressed by the study is the prevalence of duplicate and the need for users to quickly assess summaries of important posts in Amharic. The proposed solution involves a three-step process: firstly, computing similarity between documents, secondly, clustering these documents based on similarity using the K-means algorithm, and thirdly, summarizing each cluster using the TF-IDF algorithm. This extractive summarization approach focuses on selecting high-ranked sentences from the posts to form concise summaries. The results of the study show that increasing the size of the summary directly correlates with a higher extraction rate.

[23] presents a work on enhancing Amharic text summarization through abstractive methods that use deep learning architectures to understand the meaning of sentences to create coherent summaries. In this study, scheduled sampling, which combines curriculum learning with deep learning, is applied. This method starts training with both input text and reference summaries but gradually introduces the model to its own outputs, decreasing reliance on reference sentences. This approach helps address the problem of exposure bias common in sequence-to-sequence models. In this paper, the authors importantly highlight the challenge of working with Amharic as it is a low-resource language. They describe their effort to build a custom dataset and word-embedding model for Amharic. For this study, they compiled a dataset by scraping over 50,000 articles from various Amharic news websites. This dataset, along with the developed word-embedding model, is provided as open-source. The model's performance is evaluated using metrics like ROUGE and BLEU, showing promising results.

2.6 Related Works

Recent research has explored PEFT techniques for improving large language model performance on downstream tasks, particularly for low-resource languages. [20] proposed

composing language and task-specific PEFT modules to enhance zero-shot cross-lingual transfer.

The combination of transfer learning from multilingual models and parameter-efficient fine-tuning techniques could enable high-quality summarization models for Amharic without the need for extensive computational resources[24].

2.6.1 Similar Efforts in Amharic Summarization

[25] presents a crucial effort to improve the performance of the LLaMA-2 model for this under-resourced language. Recognizing that the unavailability of comprehensive task-specific and generative datasets hinders the effective fine-tuning of LLMs, the authors propose a methodology centered on the creation and integration of diverse datasets to enhance the existing LLaMA-2-Amharic model. Their approach involved a meticulous process of dataset creation, starting by leveraging existing Amharic NLP datasets designed for specific tasks like sentiment analysis, news classification, named entity recognition, question answering, text summarization (using resources like XL-Sum), machine translation (drawing from WMT19 and NLLB), and even exploring spelling correction.

Crucially, these varied task-specific datasets were transformed into an instruction-following format by developing task-specific instruction templates and employing a data creation pipeline to merge instructions with appropriate data, thereby making them suitable for instruction fine-tuning of LLMs [25]. Supplementing these converted datasets, the researchers also undertook the creation of entirely new generative datasets tailored for Amharic, focusing on areas such as music lyrics generation, folktale story generation, poem generation, and news title generation to endow the model with broader generative capabilities. To further enrich the training data, they incorporated machine-translated versions of widely used instruction datasets (like Alpaca and Dolly), acknowledging the practicality of leveraging existing resources while carefully managing data balance and applying thresholding to prevent any single task from dominating the training process.

With these carefully curated and compiled datasets, the authors proceeded to fine-tune the LLaMA-2-Amharic model using SFT, often utilizing PEFT techniques like LoRA to manage the computational demands associated with training large models [25]. The fine-tuned models were then subjected to evaluation across a variety of NLP tasks, and the results demonstrated promising performance, indicating that the integration of these tailored datasets effectively enhances the model's abilities for Amharic. The paper's contributions are significant, particularly in the context of low-resource NLP research, as they not only provide a detailed methodology for creating high-quality instruction datasets for Amharic but also open-source their dataset creation pipeline, the instruction datasets themselves, the trained models, and evaluation outputs, actively encouraging further language-specific studies and development in this space. However, the authors

also candidly discuss limitations encountered during their work, including the challenge of finding reliable generation metrics for Amharic tasks, noting that models sometimes produced overly verbose outputs despite instruction design efforts, and they reported using multiple metrics to best capture task ability while acknowledging high word error rates in tasks like spell correction and NER across evaluated models, including powerful ones like GPT-4, highlighting inherent difficulties in these specific Amharic tasks; they also mentioned being limited by computational resources which prevented them from pre-training the original LLaMA 2 model from scratch. Overall, the "Walia-LLM" paper represents a vital step forward in making advanced LLM technology more accessible and effective for underrepresented languages like Amharic, providing valuable resources and insights for future research aimed at bridging the digital language divide.

[26] tackles the specific challenge of automatically generating news headlines for the Amharic language, a task complicated by the scarcity of high-quality linguistic datasets. Recognizing this limitation, the authors embark on a study to leverage the power of pre-trained transformer models, opting to fine-tune the relatively smaller checkpoint of the T5v1.1 model, t5-small, to perform this abstractive summarization task for Amharic news articles. Their methodology commenced with the crucial step of data collection, amassing an Amharic dataset comprising over 70,000 news articles paired with their corresponding headlines sourced from Amharic news websites. This collected data underwent necessary preprocessing steps, including text cleaning to ensure quality, and a technique involving the TF-IDF algorithm was employed for news article size optimization, a potentially important step given the input constraints of many models. Furthermore, prior to feeding the data into the transformer model for feature extraction and the headline generation task, a dedicated tokenizer model was developed using the byte pair encoding (BPE) algorithm, a standard approach for handling the vocabulary of languages, especially those with complex morphology like Amharic.

With the dataset prepared and tokenized, the core of their work involved fine-tuning the t5-small model on this Amharic news dataset, adapting its vast pre-trained knowledge to the specific task of generating concise headlines from longer article bodies [26]. To evaluate the performance of their fine-tuned model, the authors employed standard metrics commonly used in text generation and summarization tasks, namely Rouge-L, BLEU, and Meteor, assessing the quality of the generated headlines against the human-written reference headlines on a test partition of the dataset containing 7,230 instances. Their initial evaluation yielded Rouge-L, BLEU, and Meteor scores of 0.5, 0.24, and 0.71 respectively, which they considered good relative to the performance of the base, non-fine-tuned T5 model that achieved significantly lower scores (0.1 Rouge-L, 0.03 BLEU, and 0.14 Meteor).

Seeking to further enhance the quality of the generated summaries, the researchers incorporated a postprocessing technique utilizing a rule-based approach, which proved

effective in boosting the performance metrics, resulting in improved scores of 0.72 for Rouge-L, 0.52 for BLEU, and 0.81 for Meteor [26]. These postprocessed results were highlighted as being relatively better not only than the non-fine-tuned T5 but also surpassing the results reported in previous studies focused on abstractive text summarization for the Amharic language, which had reported a Rouge-L score of 0.27. The paper concludes by emphasizing that their findings offer valuable insight for the practical application and future improvement of Amharic news headline generation, suggesting avenues for further research including increasing the article length processed by the model, utilizing larger volumes of training data, exploring machine learning-based adaptive postprocessing techniques, and fine-tuning other available pre-trained models, contributing a concrete step towards advancing generative NLP for Amharic despite its low-resource status.

Table 2.4 below summarizes these approaches and demonstrate the potential of PEFT for enhancing Amharic text summarization.

2.6.2 Gap and Contribution

Effective and robust PEFT methods are particularly critical in scenarios characterized by limited labeled data, such as those encountered when working with languages that have fewer resources. The significant challenges in developing NLP tools for Amharic, largely due to the scarcity of resources, underscore the potential impact of efficient transfer learning techniques[27]. Extending this, the development of potentially more powerful hybrid or dynamically adapted PEFT methods holds particular promise for addressing the unique difficulties in such low-resource settings[1].

While the concept of explicitly defined "Hybrid and Dynamic PEFT Approach" is still in its nascent stages within the current research landscape, the underlying ideas are clearly emerging from efforts to unify and dynamically adapt PEFT methods. Frameworks like UNIPELT and the unified view of PEFT represent significant steps towards realizing the potential of these more sophisticated and adaptive fine-tuning strategies. Future research focused on understanding the interactions between different PEFT techniques, designing effective dynamic adaptation mechanisms, and rigorously evaluating the performance and efficiency of hybrid and dynamic approaches will be crucial for advancing the field of parameter-efficient transfer learning for large language models.

To contribute to this promising avenue of research and address the challenges in Amharic NLP, this study proposes a novel custom parameter-efficient fine-tuning framework that is inspired by LoRA and Adapters with a dynamic rank adjustment and activation mechanism for enhanced performance on low-resource text summarization tasks. The methodology section that follows outlines the design and implementation of this framework, the experimental setup, and the evaluation procedures.

Table 2.4: Summary of Related Works

Feature	**Walia-LLM: Enhancing Amharic-LLaMA by Integrating Task-Specific and Generative Datasets** [25]	**Fine-Tuned Pre-trained Transformer for Amharic News Headline Generation**[26]
Primary Goal/Focus	Enhance Amharic LLaMA-2 model performance by integrating diverse instruction-following and generative datasets.	Generate news headlines for the Amharic language by fine-tuning a pre-trained transformer.
Language	Amharic	Amharic
Base Model Used	LLaMA-2-Amharic (based on LLaMA-2)	T5v1.1 small (t5-small)
Fine-tuning Approach	Supervised Fine-Tuning (SFT) using instruction format. Used PEFT (specifically LoRA)	Full Fine-tuning of t5-small.
Main Task(s)	Multiple NLP tasks framed as instruction following (includes summarization, classification, QA, generation, etc.).	News Headline Generation (a form of abstractive summarization).
Dataset Type/Source	Compiled instruction datasets (converted from existing task data + new generative data + translated instruction data).	Dataset of over 70k Amharic news articles and headlines collected from news websites.
Key Metrics Used	Evaluated on various task-specific metrics (details less prominent in summaries, focus on overall enhancement).	Rouge-L, BLEU, Meteor.
Notable Results	Shows promising results across different NLP tasks; demonstrates enhancement of Amharic LLAMA model.	Achieved specific scores (e.g., 0.72 Rouge-L with postprocessing) better than non-tuned T5 and previous work.
Relevance to Our Research	Uses LLM for Amharic; addresses low-resource challenges; employs instruction tuning (related to adaptation); covers summarization as a task.	Focuses on Amharic text summarization task; addresses low-resource challenges; uses fine-tuned Transformer (provides a baseline type).

Chapter 3

Methodology

This chapter provides a comprehensive account of the methodology used for the research described in this thesis. It covers the complete research design, detailing the structure and major elements of the novel Parameter-Efficient Fine-Tuning framework module. Furthermore, the section describes the specific Amharic text summarization dataset that was used, including its properties and preparation, and outlines the thorough experimental setup, which details the base pre-trained model, hyperparameters, and the chosen evaluation metrics. The approaches defined here were formulated to successfully tackle the difficulties inherent in adapting large language models for low-resource Amharic text summarization, and to evaluate the empirical results of the proposed method against suitable comparative baselines. A key feature of this work involves integrating dynamic, gradient-driven adaptation mechanisms within our specialized PEFT modules, where the effective rank of the DyLoR-Amharic elements is regulated through masking, and the influence of AdaptAmharic contributions is gradually controlled based on their utility signals.

3.1 Overall Research Design

This research adopts the Design Science Research Process (DSRP) paradigm, a recognized methodology in Information Systems and related fields focused on creating innovative artifacts to address real-world problems. The research process is guided by the widely accepted DSRP model proposed by Peffers et al. [28], which provides a cyclical framework for conducting and presenting design science research. The alignment of this research with the core steps of the DSRP model is as follows:

- **Problem Identification and Motivation:** Addressed in the Introduction (Chapter 1.2) and further elaborated in the Literature Review (Chapter 2.6.2), defining the specific challenges of low-resource Amharic text summarization that necessitate a novel solution.

- **Objectives for a Solution:** Clearly articulated as the research questions and objectives in the Introduction (Chapter 1.4), outlining the desired capabilities and performance of the proposed framework.
- **Design and Development:** Detailed in the subsequent subsections of this Methodology chapter, specifically in the Proposed PEFT Framework section (Section 3.2, where the architecture and mechanics of the novel framework are presented.
- **Demonstration:** Executed through the implementation and application of the developed framework to the task of Amharic text summarization, showcasing its functionality in generating summaries. This is part of the Experiments and Results chapter (Chapter 4).
- **Evaluation:** Conducted by empirically assessing the performance of the developed framework against established metrics and relevant baselines, as presented and analyzed in the Experiments and Results chapter (Chapter 4).
- **Communication:** Fulfilled through this thesis document itself, which communicates the problem, artifact, evaluation, and findings to the academic community.

This DSR approach is particularly well-suited for this research as it involves the systematic design, development, and rigorous evaluation of a novel IT artifact – a custom parameter-efficient fine-tuning framework – specifically engineered to address the practical challenges of text summarization for the low-resource Amharic language.

3.2 Proposed PEFT Module

This research introduces a novel custom Parameter-Efficient Fine-Tuning module designed to enhance the adaptation of large pre-trained language models for low-resource text summarization, specifically for Amharic. The proposed framework combines the strengths of LoRA and Adapter methods with gradient-based dynamic activation mechanisms, allowing for a more flexible and data-driven allocation of model capacity during fine-tuning. The framework consists of two main components: **DyLoRA-Amharic** for dynamic LoRA rank allocation and **AdaptAmharic** for dynamic adapter activation.

The core idea behind the dynamic mechanisms is to leverage gradient norms computed during backpropagation as signals of parameter importance. Layers or components that exhibit larger gradient norms are considered more important for the task at hand, as the model is actively trying to adapt them. This gradient-based importance informs the dynamic adjustment of PEFT parameters.

DyLoRA-Amharic: Adaptive Rank Reduction for Low-Resource NLP

To optimize LoRA for low-resource languages like Amharic, we introduce a dynamic rank allocation mechanism. The DyLoRA-Amharic module dynamically adjusts its rank based on the Exponential Moving Average (EMA) of its gradient norm. For a given DyLoRA-Amharic module, the current_rank is updated as follows:

$$\text{current\_rank} = \min\left(\max\left(\lfloor \beta_{\text{scale}} \cdot \text{EMA\_grad\_norm} \rfloor, 1\right), \text{max\_rank}\right) \tag{3.1}$$

Where:

- EMA_grad_norm is the Exponential Moving Average of the Frobenius norm of the gradients of the DyLoRA-Amharic's `up` projection layer's weights (W_{up}).
- β_{scale} is a scaling hyperparameter (e.g., 0.1), controlling the sensitivity of rank adjustment to gradient magnitude.
- max_rank is the predefined maximum allowed rank (e.g., 16), ensuring the rank stays within a reasonable bound.
- $\lfloor \cdot \rfloor$ denotes the floor function, converting the raw rank to an integer.

The EMA_grad_norm is updated at each step using:

$$\text{EMA\_grad\_norm}_{\text{new}} = \alpha_{\text{ema}} \cdot \text{current\_step\_grad\_norm} + (1 - \alpha_{\text{ema}}) \cdot \text{EMA\_grad\_norm}_{\text{old}} \tag{3.2}$$

Where α_{ema} is the EMA decay factor (e.g., 0.1). This approach makes rank selection data-driven, allowing more capacity to layers exhibiting stronger learning signals.

AdaptAmharic: Gradient-Weighted Adapter Activation

To improve adaptation for Amharic text summarization, we introduce a dynamic activation mechanism for adapter layers. The AdaptAmharic module controls its contribution via a continuous activation value, active, which is updated based on the Exponential Moving Average (EMA) of its gradient norm. The active value is computed using a sigmoid function:

$$\text{active} = \sigma\left(\gamma_{\text{gating}} \cdot \text{EMA\_grad\_norm} + \beta_{\text{activation}}\right) \tag{3.3}$$

Where:

- σ is the sigmoid function, ensuring the activation value is between 0 and 1.

- EMA_grad_norm is the Exponential Moving Average of the Frobenius norm of the gradients of the AdaptAmharic's `up` projection layer's weights.
- γ_{gating} is a scaling factor for the EMA gradient norm (e.g., 1.0), controlling the steepness of the sigmoid.
- $\beta_{\text{activation}}$ is a bias term (e.g., 0.1), shifting the sigmoid curve to influence the baseline activation level.

The EMA_grad_norm is updated at each step using:

$$\text{EMA\_grad\_norm}_{\text{new}} = \alpha_{\text{ema}} \cdot \text{current\_step\_grad\_norm} + (1 - \alpha_{\text{ema}}) \cdot \text{EMA\_grad\_norm}_{\text{old}} \tag{3.4}$$

Where α_{ema} is the EMA decay factor (e.g., 0.1). This smooth gating mechanism allows for nuanced and continuous control over the adapter's influence, leading to more stable and effective learning.

Joint Optimization Objective

To optimize both DyLoRA-Amharic and AdaptAmharic parameters jointly, we introduce a composite loss function that combines the main summarization task loss with regularization terms for the dynamic rank and activation values. This objective balances dynamic capacity adaptation with controlled module influence.

The total loss L_{total} is defined as:

$$L_{\text{total}} = L_{\text{task}} + \lambda_{\text{rank}} \sum_{l=1}^{N_{\text{DyLoRA}}} \text{current\_rank}_l + \lambda_{\text{activation}} \sum_{l=1}^{N_{\text{Adapt}}} \text{active}_l \tag{3.5}$$

Where:

- L_{task} is the main summarization loss (e.g., cross-entropy loss for sequence-to-sequence tasks).
- λ_{rank} is the regularization weight for DyLoRA-Amharic ranks (e.g., 1×10^{-5}). This term penalizes higher ranks, encouraging the model to maintain parameter efficiency.
- current_rank_l is the dynamically determined integer rank of the l-th DyLoRA-Amharic module.
- N_{DyLoRA} is the total number of DyLoRA-Amharic modules injected across the model.

- $\lambda_{\text{activation}}$ is the regularization weight for AdaptAmharic activation values (e.g., 1×10^{-5}). This term encourages the model to use lower activation levels unless truly beneficial.
- active_l is the dynamically determined continuous activation value (between 0 and 1) of the l-th AdaptAmharic module.
- N_{Adapt} is the total number of AdaptAmharic modules injected across the model.

This objective ensures that the model dynamically allocates resources (rank and activation) while being regularized against unnecessary complexity, promoting an efficient and stable fine-tuning process.

Training Algorithm

The training process for the proposed framework follows a standard fine-tuning loop, augmented with integrated steps for computing gradient norms and dynamically updating the PEFT parameters. The algorithm for a single training step is as follows:

1. **Forward Pass and Loss Computation:** The input data is passed through the mT5-small model, which now includes the injected `LinearWithAdapters` wrappers. The output is used to compute the primary task loss (L_{task}), typically cross-entropy loss for summarization.
2. **Gradient Calculation:** Backpropagation is performed to compute gradients for all trainable parameters, which exclusively belong to the DyLoRA-Amharic and AdaptAmharic modules.
3. **Gradient Norm Tracking and EMA Update:** For each individual DyLoRA-Amharic and AdaptAmharic module, the Frobenius norm of the gradients of its respective `up` projection layer's weights is computed. These current step gradient norms are then used to update their Exponential Moving Averages (EMA). This EMA serves as a smoothed indicator of the module's importance and learning activity.
4. **Dynamic Parameter Update:**
    - **DyLoRA-Amharic Rank Update:** For each DyLoRA-Amharic module, the current_rank is dynamically determined based on its updated EMA_grad_norm and the β_{scale} hyperparameter, as defined in Equation 3.1. This rank dictates how much of the fixed-size low-rank matrices' output is effectively utilized.

Crucially, the rank is implemented via masking: the full $W_A W_B$ matrix is computed, but its effective rank is controlled by only considering the first current_rank dimensions during the matrix multiplication in the forward pass, effectively masking out the contribution of higher ranks. This avoids the computational overhead of resizing matrices.

- **AdaptAmharic Activation Update:** For each AdaptAmharic module, its continuous active value is dynamically updated by passing its updated EMA_grad_norm through a sigmoid function, scaled by γ_{gating} and biased by $\beta_{\text{activation}}$, as defined in Equation 3.3. This smooth gating mechanism controls the overall contribution of the adapter's output to the main model's forward pass.

5. **Regularization Loss Integration:** The total loss is augmented by adding regularization terms based on the sum of all current DyLoRA-Amharic ranks and the sum of all AdaptAmharic active values, weighted by λ_{rank} and $\lambda_{\text{activation}}$ respectively, as defined in Equation 3.5 This encourages parameter efficiency and training stability.

6. **Optimizer Step:** An optimizer step is performed based on the total loss, updating the trainable parameters of the DyLoRA-Amharic and AdaptAmharic modules.

7. **Gradient Zeroing:** Gradients are zeroed for the next iteration.

The overall algorithm is summarized in Algorithm 1.

Key Insights and Advantages for Low-Resource Amharic

This joint dynamic adaptation strategy offers several key advantages, particularly for low-resource and morphologically complex languages like Amharic.

- **Gradient-Driven Dynamicity:** The framework's core strength lies in its ability to leverage gradient signals as indicators of parameter importance. This allows the model to learn layer-wise sensitivity and dynamically determine where to focus adaptation efforts and how much capacity (via DyLoRA-Amharic rank) or activity (via AdaptAmharic gating) is needed per layer. This contrasts with static PEFT methods that pre-define these capacities.

- **Efficient Rank Adaptation via Masking:** For DyLoRA-Amharic, the dynamicity of rank is achieved through a masking approach. Instead of computationally expensive matrix resizing, the full maximum-rank low-rank matrices are maintained, but only the portion corresponding to the dynamically determined current_rank is actively utilized in the forward pass. This method ensures that the model can efficiently adapt its expressive power without incurring significant overhead.

- **Smooth and Stable Activation:** The AdaptAmharic module's smooth sigmoid gating ensures a continuous and differentiable control over its contribution. This prevents abrupt changes in the model's behavior, leading to more stable training and a nuanced integration of adapter knowledge, which is crucial for complex language structures.
- **Selective and Dense Adaptation:** By combining DyLoRA-Amharic (effective for attention weights) and AdaptAmharic (effective for FFNs and general layer-wise modulation), the framework comprehensively augments every linear layer in both encoder and decoder. This dense, yet selectively activated, adaptation helps avoid overfitting to limited data, a significant challenge in low-resource settings, by only activating capacity when and where it is most beneficial.
- **Holistic Fine-Tuning:** Unlike methods that statically inject or activate parameters, this approach learns the importance of different model components and PEFT mechanisms in real-time during training. This provides a more expressive and flexible fine-tuning strategy well-suited for capturing language-specific nuances in Amharic, from initial text encoding to final summary generation.

This dynamic and regularized approach provides a robust solution for adapting large language models to the unique linguistic and data challenges of Amharic text summarization.

3.3 Data Collection and Preparation

Dataset Origin and Rationale for Use

Given the challenges inherent to low-resource languages, the creation and curation of a high-quality dataset is pivotal for robust model performance. Instead of reinventing the wheel, this study adopts the meticulously curated **Amharic-3 dataset** from previous work on Amharic text summarization [29]. The following outlines how the Amharic-3 dataset was derived and prepared for fine-tuning the Multilingual Text-to-Text Transfer Transformer (mT5) model using parameter-efficient methods.

The Amharic-3 dataset represents the final evolution from earlier versions (Amharic-1 and Amharic-2), with a clear focus on maximizing quality over sheer volume. While the initial collections contained more entries, they exhibited variability in text lengths and contamination with non-Amharic elements. The Amharic-3 version addresses these issues by enforcing rigorous cleaning and normalization steps that ensure each entry is both linguistically coherent and fully compatible with the fixed token limits required by the model.

Aggregation and Cleaning Process

The development of the Amharic-3 dataset involved a multi-step procedure:

- **Data Aggregation:** Data from several open Amharic text summarization projects were compiled from prior studies [23, 30, 31].
- **Cleaning and Filtering:**
  - **Duplicate and Incomplete Entry Removal:** Eliminating duplicate entries and those with missing values improved the integrity of the dataset.
  - **Language Verification:** A regex-based filter was applied to ensure that at least 80% of the characters in each entry are Amharic, effectively excluding texts with significant contamination from other languages.
  - **Character-Level Normalization:** The inherent orthographic variations of the Amharic script were standardized by substituting variant representations with a common form.
  - **Extraneous Symbol Removal:** Non-Amharic characters and superfluous punctuation were removed to maintain consistency across the dataset.

Preprocessing Aligned with Model Requirements

After cleaning, the dataset was preprocessed to meet the input requirements of the mT5-small model:

- **Tokenization:** Both texts and summaries were tokenized using the mT5 tokenizer with a maximum token limit of 512 for texts and 128 for summaries. Such strict limits minimize truncation and preserve the essential context of each example.
- **Preservation of Context:** Enforcing these token limits ensures that the model processes examples that are as complete as possible, which in turn supports efficient fine-tuning.

Descriptive Statistics

Table 3.1 and Figure 3.1 summarizes the key descriptive statistics of the Amharic-3 dataset, offering insights into its structure and quality:

The table and figure indicates that the dataset is well curated, with text entries averaging approximately 352 tokens and summaries around 35 tokens. The relatively low standard deviations suggest manageable variability, and the imposed maximum token count (512) enforces consistency with the computational limits.

Table 3.1: Descriptive Statistics of the Amharic-3 Dataset

Statistic	Text Tokens	Summary Tokens
Count	29,366	29,366
Mean	352.43	34.89
Standard Deviation	96.32	11.32
Minimum	96	13
25th Percentile	275	27
Median (50th Percentile)	358	32
75th Percentile	434	40
Maximum	512	119

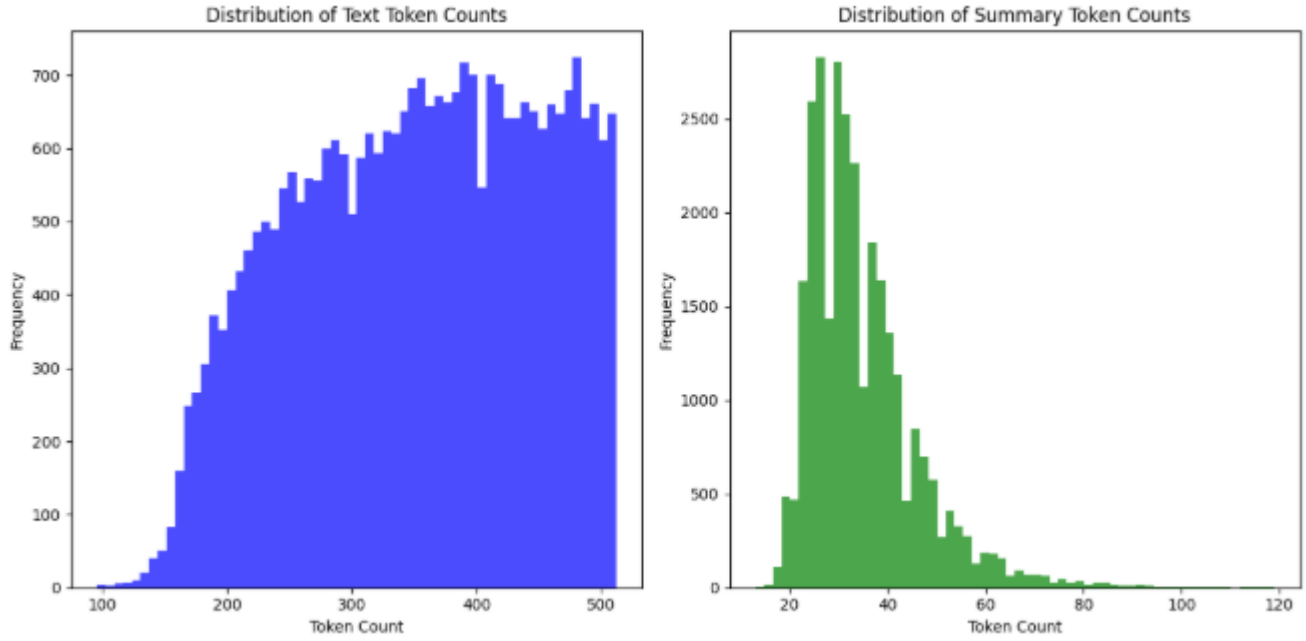


Figure 3.1: Histogram of token counts

Train/Validation/Test Split

After rigorous cleaning and preprocessing, the Amharic-3 dataset—comprising a total of 29,366 examples—was divided into three subsets to ensure robust model training and evaluation. Following standard practice, an 80% / 10% / 10% split was applied. This results in approximately 23,492 examples for training, 2,937 examples for validation, and 2,937 examples for testing. The 80-10-10 split is a widely accepted heuristic rather than a strict standard. It strikes a practical balance between maximizing the data available for training and preserving sufficient samples for validation and unbiased testing. This ratio has become conventional due to its empirical effectiveness across diverse domains and its early adoption in machine learning frameworks and literature[32].

Table 3.2 summarizes the distribution of examples:

This split strategy allows the training set to provide ample data for the model to learn the core linguistic features effectively, while the validation and test sets—being kept

Table 3.2: Train/Validation/Test Split for the Amharic-3 Dataset

Split	Percentage	Number of Examples
Training	80%	23,492
Validation	10%	2,937
Test	10%	2,937
Total	100%	29,366

separate from training—ensure unbiased hyperparameter tuning and reliable performance evaluation.

Impact on Model Fine-Tuning

The high quality and consistency of the Amharic-3 dataset are expected to contribute significantly to effective model fine-tuning. Its controlled token length distribution minimizes noise and ensures that the mT5-small model—fine-tuned using parameter-efficient methods is exposed to stable, representative examples. This ultimately enhances the model's ability to capture the essential semantics needed for generating relevant and coherent summaries.

3.4 Base Pre-trained Language Model

The selection of an appropriate base pre-trained language model is a critical step in the parameter-efficient fine-tuning process. For this research, the **mT5-small** model was chosen as the foundation for developing the custom PEFT framework. mT5 is a multilingual variant of the Text-to-Text Transfer Transformer (mT5) model, pre-trained by Google on a massive multilingual dataset known as mC4 [33]. The mC4 dataset is derived from a large web crawl and includes data from over 100 languages, crucially including Amharic. This pre-training exposure provides mT5 with an initial understanding of the Amharic language, making it a suitable starting point for downstream Amharic NLP tasks. The mT5 architecture follows the standard Transformer-based encoder-decoder structure of the original T5 model, making it inherently well-suited for sequence-to-sequence tasks such as text summarization. We specifically utilize the 'small' version of the mT5 model, which contains approximately 300 million parameters. This size strikes a balance between model capacity and computational feasibility for fine-tuning, aligning with the goals of parameter-efficient methods and the constraints often present when working with low-resource languages. The mT5-small model was accessed and utilized through the Hugging Face Transformers library [34], which provides convenient access to the pre-trained weights and tokenizer.

3.5 Evaluation Metrics

To quantitatively evaluate the performance of the proposed custom PEFT module and compare it against baseline methods for Amharic text summarization, a suite of widely accepted automatic evaluation metrics was employed. The chosen metrics are **ROUGE-1**, **ROUGE-2**, **ROUGE-L**, **BLEU**, **Meteor**, **BERTScore**, **COMET** and **chrF++**. These metrics measure the similarity between the machine-generated summaries and human-written reference summaries, capturing different aspects of content overlap and fluency.

- **ROUGE** [35]: evaluates the quality of automatically produced summaries by examining how much they overlap with human-written references in terms of shared words or phrase sequences.
- **Bilingual Evaluation Understudy (BLEU)** [36]: a metric first applied to machine translation, determines how closely generated text aligns with a reference by analyzing the frequency of matching word patterns while penalizing overly short outputs.
- **Metric for Evaluation of Translation with Explicit Ordering (Meteor)** [37]: extends BLEU by including linguistic flexibility—such as matching stems, synonyms, and paraphrases—making it more aligned with human judgment of translation or summarization quality.
- **BERTScore** [38]: BERTScore evaluates text generation by computing the similarity between each token in the candidate text and each token in the reference text using contextual embeddings from a pre-trained model (such as BERT). It then calculates precision, recall, and F1 scores based on the maximum similarity found for each token. BERTScore is valuable as it moves beyond simple n-gram matching to capture semantic similarity and paraphrasing.
- **COMET (Cross-lingual Optimization for MT Evaluation)** [39]: A neural metric that leverages pre-trained multilingual models to embed the source text, generated text, and reference text. It is trained to predict human judgments of quality, offering a more semantically aware and often human-correlated evaluation, and can consider source text for adequacy.
- **chrF++: words helping character n-grams** [40]: A metric primarily based on character n-gram F-score, which also incorporates word n-grams. It is particularly effective for morphologically rich languages like Amharic, as it is less sensitive to variations in word forms and tokenization issues, providing a robust measure of similarity at the character level.

Collectively, these metrics provide a comprehensive quantitative assessment of the generated summaries, evaluating both the presence of key content (ROUGE), the quality of the generated text in terms of n-gram overlap and fluency (BLEU), and the semantic similarity to the reference (BERTScore).

3.6 Baselines

To empirically evaluate the effectiveness of the proposed custom PEFT module, its performance is rigorously compared against several relevant baseline models and previously reported findings. These comparisons serve to highlight the advantages of the custom approach over simpler or established methods and position the research within the context of prior work in Amharic summarization. The baselines and comparison points for this research include:

- **Base Pre-trained Model (mT5-small):** The performance of the raw mT5-small model without any fine-tuning on the downstream summarization task is evaluated. This baseline, often referred to as zero-shot or few-shot performance depending on the prompting strategy, provides an indication of the base model's inherent capabilities on the task and highlights the gains achieved through fine-tuning.
- **mT5-small with LoRA:** This baseline utilizes the LoRA technique applied to the mT5-small model. This model is implemented and trained under the same experimental conditions (dataset, hyperparameters, training procedure) as the proposed custom framework to provide a direct comparison against a widely used standalone PEFT method.
- **mT5-small with Adapter:** This baseline employs the Adapter method for parameter-efficient fine-tuning of the mT5-small model. Similar to the LoRA baseline, this model is implemented and trained using the same experimental setup to evaluate the performance of standalone Adapter tuning.

By comparing the proposed custom PEFT framework's performance against this range of baselines, including standalone PEFT methods, the raw base model, and results from previous Amharic and multilingual summarization research, this study aims to provide a thorough evaluation of its effectiveness and position its contribution within the current research landscape.

Chapter 4

Experiments and Results

This chapter presents the empirical evaluation of the proposed novel custom PEFT framework for Amharic text summarization. It details the experimental setup, presents the quantitative and qualitative results obtained from fine-tuning the base pre-trained language model using the proposed method and several baseline approaches, and analyzes these results to assess the effectiveness of the framework in addressing the research objectives. The performance comparison focuses on key evaluation metrics relevant to text summarization, providing insights into the strengths and limitations of the different fine-tuning strategies in a low-resource context.

4.1 Experimental Setup

4.1.1 Experimental Setup

This section details the setup used to conduct the experiments for evaluating the proposed custom PEFT framework and the selected baseline models. All experiments were performed using the Amharic text summarization dataset described in Section 3.3 (Methodology Chapter), utilizing the mT5-small model as the base pre-trained language model (Section 3.4, Methodology Chapter).

4.1.2 mT5-small + LoRA Baseline

The primary objective of this experiment was to establish a performance baseline for abstractive text summarization on an Amharic language dataset. This was achieved by fine-tuning a pre-trained multilingual model using the standard Low-Rank Adaptation (LoRA) method from the PEFT library. The results from this experiment can serve as a benchmark against which more complex, custom fine-tuning methods can be compared.

Model & Tokenizer

- **Base Model:** `google/mt5-small`. A multilingual T5 model pre-trained on the mC4 corpus, which includes Amharic.
- **Tokenizer:** The corresponding `MT5Tokenizer` for `google/mt5-small` was used for all text processing tasks.

Dataset and Preprocessing

The data was loaded as described in Section 3.3. The preprocessing was done using the `AutoTokenizer` for `google/mt5-small`.

- Input texts were tokenized with a maximum length of 512 tokens.
- Reference summaries were tokenized with a maximum length of 128 tokens.
- Truncation was applied to both inputs and labels.
- Labels were set to `-100` for padding tokens to be ignored during loss.

Fine-Tuning Methodology

This experiment used standard PEFT LoRA with the following configuration:

- **Task Type:** `SEQ_2_SEQ_LM`
- **Target Modules:** LoRA adapters on the query (`q`) and value (`v`) projection matrices in each attention block.
- **LoRA Rank (r):** 16
- **LoRA Alpha (α):** 32
- **LoRA Dropout:** 0.1
- **Bias:** No bias terms (`none`) in LoRA layers.

Training Hyperparameters

Training was performed with Hugging Face's `Seq2SeqTrainer`:

- **Epochs:** 10
- **Batch Sizes:** `per_device_train_batch_size`=4, `per_device_eval_batch_size`=1
- **Gradient Accumulation:** 1 steps
- **Learning Rate:** 1e-5
- **Weight Decay:** 0.01
- **Gradient Clipping:** max norm 1.0
- **Mixed Precision:** fp16=False

- **Evaluation:** every 500 steps
- **Checkpointing:** every 500 steps, keep last 3
- **Best Model:** loaded by `rougeL`
- **Generation Settings:** `predict_with_generate=True`, `num_beams`=4, `max_new_tokens`=128, `min_length`=20, `no_repeat_ngram_size`=3, `length_penalty`=2.0

Evaluation

- **Metrics:** ROUGE (ROUGE-1, ROUGE-2, ROUGE-L) and BLEU scores.
- **Strategy:** Evaluate every **1,000 steps** on the validation set.
- **Best Model:** Checkpoint with highest `eval_rougeL` was loaded at the end for final test evaluation.

4.1.3 mT5-small + Adapter Baseline

This experiment aimed to establish a second performance baseline for Amharic abstractive text summarization using the classic Houlsby adapter architecture. By fine-tuning a pre-trained model with this method, we created a robust point of comparison against both the standard LoRA baseline and more advanced custom architectures.

Environment and Dependencies

To ensure reproducibility and resolve known dependency conflicts (specifically the `transformers.model` import error), a clean environment was configured. The following commands were executed:

- pip uninstall -y transformers adapters
- pip install transformers==4.38.2
- pip install adapters

This guarantees compatibility between the `transformers` library and the `adapters` package, which provides the Houlsby adapter implementation.

Dataset and Preprocessing

The data was loaded as described in Section 3.3. The preprocessing was done using the `AutoTokenizer` for `google/mt5-small`.

- Input texts were tokenized with a maximum length of 512 tokens.
- Reference summaries were tokenized with a maximum length of 128 tokens.
- Truncation was applied to both inputs and labels.
- Labels are set to `-100` for padding tokens to be ignored during loss.

Fine-Tuning Methodology

We used the `adapters` library to implement Houlsby-style adapters:

- The base model's weights were frozen; only adapter weights were trained.
- **Adapter Configuration:** Predefined "pfeiffer" (Houlsby) recipe, which inserts two adapter modules per Transformer layer: one after multi-head attention, one after the feed-forward block.
- **Adapter Name:** `summarization_adapter`.
- **Training Call:** `model.train_adapter("summarization_adapter")` ensures only adapter parameters are updated.

Training Hyperparameters

Training was performed with Hugging Face's `Seq2SeqTrainer`:

- **Epochs:** 10
- **Batch Sizes:** `per_device_train_batch_size`=4, `per_device_eval_batch_size`=1
- **Gradient Accumulation:** 1 steps
- **Learning Rate:** 1e-5
- **Weight Decay:** 0.01
- **Gradient Clipping:** max norm 1.0
- **Mixed Precision:** fp16=False
- **Evaluation:** every 500 steps
- **Checkpointing:** every 500 steps, keep last 3
- **Best Model:** loaded by `rougeL`
- **Generation Settings:** `predict_with_generate=True`, `num_beams`=4, `max_new_tokens`=128, `min_length`=20, `no_repeat_ngram_size`=3, `length_penalty`=2.0

Evaluation

- Metrics: ROUGE-1, ROUGE-2, ROUGE-L F1 scores; BLEU score.
- Strategy: Evaluate on validation set every 1 000 training steps.
- Best Model Selection: Checkpoint with highest `eval_rougeL` was chosen for final evaluation on the test set.

4.1.4 mT5-small + DyloraAmharic + AdaptAmharic

This section details the methodology, model architecture, dataset, and training configurations employed for the text summarization experiments. The primary objective was

to evaluate the effectiveness of integrating custom DyLoRAAmharic and AdaptAmharic modules into a pre-trained mT5 model for Amharic text summarization.

Model Architecture

The core of our experimental setup was the `google/mt5-small` model, a multilingual T5-based sequence-to-sequence transformer, serving as the backbone for summarization. Its parameters were initially frozen to facilitate efficient fine-tuning through adapter modules. To introduce parameter-efficient fine-tuning and dynamic adaptation, two custom modules, `DyLoRAAmharic` and `AdaptAmharic`, were developed and directly injected into the linear projection layers of the base model.

DyLoRAAmharic Module

This module implements a dynamic low-rank adaptation strategy. It consists of a `down` projection layer (mapping `in_features` to `max_rank`) and an `up` projection layer (mapping `max_rank` to `out_features`). The rank of the adapter was dynamically adjusted during training based on the exponential moving average (EMA) of the gradients flowing through its `up` projection layer.

- **Parameters:** `max_rank` is set to 16, `beta` (for rank scaling) to 0.03, and `ema_alpha` (for EMA calculation) to 0.1.
- **Initialization:** The `down` layer weights are initialized using Kaiming uniform distribution, and `up` layer weights are initialized to zeros.

AdaptAmharic Module

This module introduces an adaptive activation mechanism. It comprises a `down` projection, a ReLU activation function, and an `up` projection. The module's activation status (active/inactive) was dynamically determined based on the EMA of gradients.

- **Parameters:** `beta_activation` (for activation probability scaling) was set to 0.05, and `ema_alpha` to 0.1.
- **Initialization:** Standard PyTorch `nn.Linear` default initialization.

Injection Strategy

Both `DyLoRAAmharic` and `AdaptAmharic` modules are injected directly into the `nn.Linear` layers within the `google/mt5-small` model's encoder and decoder blocks. Specifically, they replace the original linear layers for:

- **Self-Attention:** Query (`q`), Value (`v`), and Output (`o`) projections.
- **Cross-Attention (Decoder only):** Query (`q`), Key (`k`), Value (`v`), and Output (`o`) projections.

- **Feed-Forward Networks (DenseReluDense):** The two input linear layers (`wi_0`, `wi_1` for gated activation) and the output linear layer (`wo`).

All parameters of the original `google/mt5-small` model were explicitly frozen, ensuring that only the parameters of the injected `DyLoRAAmharic` and `AdaptAmharic` modules were trainable.

Dynamic Module Mechanisms

Both adapter modules were fully dynamic in the encoder and decoder, with their behavior updated at each logging step based on the Exponential Moving Average (EMA) of their gradients.

- **DyLoRAAmharic (Dynamic Rank):** The effective rank of the LoRA matrices was dynamically adjusted between 1 and a maximum of 16. The rank was determined by the EMA of the gradient norm of its `up` projection matrix, scaled by a β hyperparameter.
- **AdaptAmharic (Smooth Gating):** This module's contribution was controlled by a continuous "activation" value (a float between 0.0 and 1.0). This value was determined by passing the EMA of its gradient norm through a `sigmoid` function, allowing for smooth, gradual gating of the adapter's output.

Regularization for Complexity Control

A key feature of this experiment was the introduction of a custom regularization term added directly to the model's loss function. This term penalizes model complexity to prevent uncontrolled growth in rank and activation. The total loss was calculated as:

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{CrossEntropy}} + \mathcal{L}_{\text{Regularization}}$$

Where $\mathcal{L}_{\text{Regularization}}$ is defined as:

$$\mathcal{L}_{\text{Reg}} = \left(\lambda_{\text{rank}} \sum \text{ranks}\right) + \left(\lambda_{\text{activation}} \sum \text{activations}\right)$$

- $\lambda_{\text{rank}} : 1 \times 10^{-5}$
- $\lambda_{\text{activation}} : 1 \times 10^{-5}$

This regularization encourages the model to use the lowest possible rank and activation levels necessary to perform the task, promoting a more efficient and stable architecture.

Dataset and Preprocessing

The data was loaded as described in Section 3.3. The preprocessing was done using the `AutoTokenizer` for `google/mt5-small`.

- Input texts were tokenized with a maximum length of 512 tokens.
- Reference summaries were tokenized with a maximum length of 128 tokens.
- Truncation was applied to both inputs and labels.
- Labels were set to `-100` for padding tokens to be ignored during loss.

Training Configuration

Training was performed with Hugging Face's `Seq2SeqTrainer`:

- **Epochs:** 10
- **Batch Sizes:** `per_device_train_batch_size`=4, `per_device_eval_batch_size`=1
- **Gradient Accumulation:** 1 steps
- **Learning Rate:** 1e-5
- **Weight Decay:** 0.01
- **Gradient Clipping:** max norm 1.0
- **Mixed Precision:** fp16=False
- **Evaluation:** every 500 steps
- **Checkpointing:** every 500 steps, keep last 3
- **Best Model:** loaded by `rougeL`
- **Generation Settings:** `predict_with_generate=True`, `num_beams`=4, `max_new_tokens`=128, `min_length`=20, `no_repeat_ngram_size`=3, `length_penalty`=2.0

Evaluation Metrics

We use standard summarization metrics:

- **ROUGE:** ROUGE-1, ROUGE-2, ROUGE-L F1-scores
- **BLEU:** n-gram precision

All metrics were computed with the Hugging Face `evaluate` library, skipping special tokens and handling empty predictions with a placeholder.

Monitoring and Logging

A custom `DynamicUpdateMonitorCallback` tracks:

- Gradient norms on adapter `up.weight` hooks
- DyLoRA rank updates and AdaptAmharic activation toggles
- NaN/Inf checks in gradients and loss
- Periodic CUDA cache clears and garbage collection

Loss curves and adapter dynamics were plotted and saved, and a CSV (`dynamic_modules_history.csv`) logs detailed training histories.

4.2 Results

This section presents the empirical results from fine-tuning the `google/mt5-small` model on the Amharic text summarization task using three distinct PEFT methodologies: Standard LoRA, Houlsby Adapters, and the proposed Custom Dynamic Adapter. Performance was evaluated on generation quality, predictive accuracy, and parameter efficiency.

4.2.1 Generation Quality and Predictive Accuracy

Evaluation on the test set used ROUGE, BLEU, and evaluation loss (cross-entropy). Table 4.1 compares these metrics across setups.

Table 4.1: Comparative Performance Metrics on the Test Set

Model Architecture	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**BLEU**	**Eval Loss**
Standard PEFT LoRA	0.0001	0.0000	0.0001	0.0000	10.998
Houlsby Adapter (Baseline)	0.0426	0.0044	0.0430	0.0599	1.897
Custom Dynamic Adapter	**0.0559**	**0.0059**	**0.0561**	**0.0913**	**1.623**

The Custom Dynamic Adapter outperformed the baselines, with a 30.5% increase in ROUGE-L and a 52.4% increase in BLEU over the Houlsby Adapter. It also achieved the lowest eval loss (1.623), indicating better next-token prediction versus Houlsby (1.897) and Standard LoRA (10.998). The Standard LoRA baseline showed near-zero generation scores and high loss, suggesting instability or failure to converge.

Standard PEFT LoRA: This baseline performed exceptionally poorly, yielding near-zero ROUGE and BLEU scores and a very high evaluation loss. This suggests that the standard LoRA approach, with its limited number of trainable parameters (0.23%), was insufficient to adapt the mT5-small model for the Amharic summarization task, leading to a complete failure in generating meaningful summaries.

Houlsby Adapter Baseline: The Houlsby adapter demonstrated a significant improvement over standard LoRA. While its ROUGE and BLEU scores were still relatively low, they were substantially higher than LoRA's, indicating some capacity for summarization. The evaluation loss was also much lower. This model utilized a slightly higher percentage of trainable parameters (0.35%) compared to standard LoRA.

DyLoRA-Amharic & AdaptAmharic: The proposed combined approach achieved the best performance across all metrics. It recorded the lowest evaluation loss (1.6232) and the highest ROUGE and BLEU scores. Notably, its ROUGE-L F1 score of 0.0561 was

a substantial improvement over the other baselines. This model, by design, utilized a significantly larger proportion of trainable parameters (13.42%) due to the full injection of dynamic modules into both encoder and decoder layers, allowing for more extensive adaptation.

4.2.2 Semantic Similarity Evaluation (BERTScore)

To further assess the quality of generated summaries beyond n-gram overlap, BERTScore was computed for a small sample of five generated summaries from the DyLoRA-Amharic & AdaptAmharic model and the Houlsby Adapter baseline. BERTScore leverages contextual embeddings to provide a more semantically aware similarity measure.

Table 4.2: Average BERTScore for Sampled Summaries

Model	**Precision**	**Recall**	**F1**
Houlsby Adapter Baseline	0.9841	0.9864	0.9852
DyLoRA-Amharic & AdaptAmharic	0.9911	0.9923	0.9917

From Table 4.2, the DyLoRA-Amharic & AdaptAmharic model demonstrates a slightly higher average BERTScore F1 (0.9917) compared to the Houlsby Adapter baseline (0.9852) on the sampled summaries. This suggests that the proposed model generates summaries that were not only better in terms of lexical overlap (as indicated by ROUGE and BLEU) but also semantically more similar to the reference summaries.

4.2.3 Advanced Semantic and Character-Level Evaluation

To provide a more comprehensive evaluation beyond n-gram overlap and basic semantic similarity, COMET and chrF++ scores were computed for a small sample of five generated summaries from both the DyLoRA-Amharic & AdaptAmharic model and the Houlsby Adapter baseline. These metrics offer insights into human-like judgment and robustness to morphological complexity, respectively.

Table 4.3: COMET and chrF++ Scores for Sampled Summaries

Model	**chrF++ Score**	**COMET Mean Score**
Houlsby Adapter Baseline	36.7451	0.7519
DyLoRA-Amharic & AdaptAmharic	37.3326	0.7247

Table 4.3 presents the aggregate chrF++ score and the mean COMET score for the sampled summaries. The DyLoRA-Amharic & AdaptAmharic model achieved a slightly higher chrF++ score (37.3326) compared to the Houlsby Adapter Baseline (36.7451). This indicates that the proposed model generates summaries with a better character

n-gram overlap, which was particularly relevant for morphologically rich languages like Amharic where word forms can vary significantly.

Conversely, the Houlsby Adapter Baseline showed a slightly higher COMET mean score (0.7519) compared to the DyLoRA-Amharic & AdaptAmharic model (0.7247) on this small sample. While COMET is designed to correlate highly with human judgments, this specific observation on a limited sample suggests that the Houlsby adapter might, for these particular examples, produce summaries that are perceived as slightly more semantically aligned or fluent by the COMET model, despite the DyLoRA-Amharic & AdaptAmharic model's better performance on other metrics. It is important to reiterate that these results are based on a very small sample and should be interpreted as indicative trends rather than definitive conclusions about overall performance.

4.2.4 Parameter Efficiency

PEFT trades off trainable parameter count against performance. Table 4.4 lists parameter counts for each architecture.

Table 4.4: Comparison of Trainable Parameters

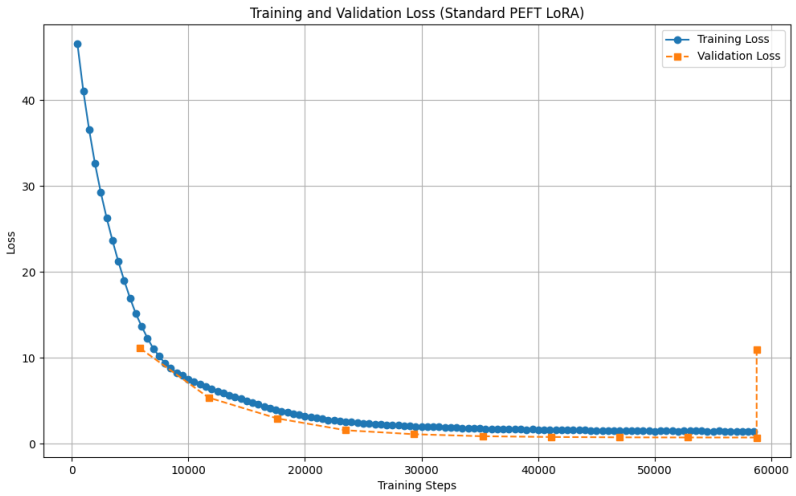
Model Architecture	**Trainable Parameters**	**Total Parameters**	**Trainable %**
Standard PEFT LoRA	688,128	300,864,896	0.23%
Houlsby Adapter (Baseline)	1,065,984	301,242,752	0.35%
Custom Dynamic Adapter	**46,531,072**	**346,707,840**	**13.42%**

The Custom Dynamic Adapter uses 13.42% of parameters versus 0.35% (Houlsby) and 0.23% (LoRA). This larger ratio stems from dense injection of two adapters per linear layer, yet still freezes 86.58% of core weights.

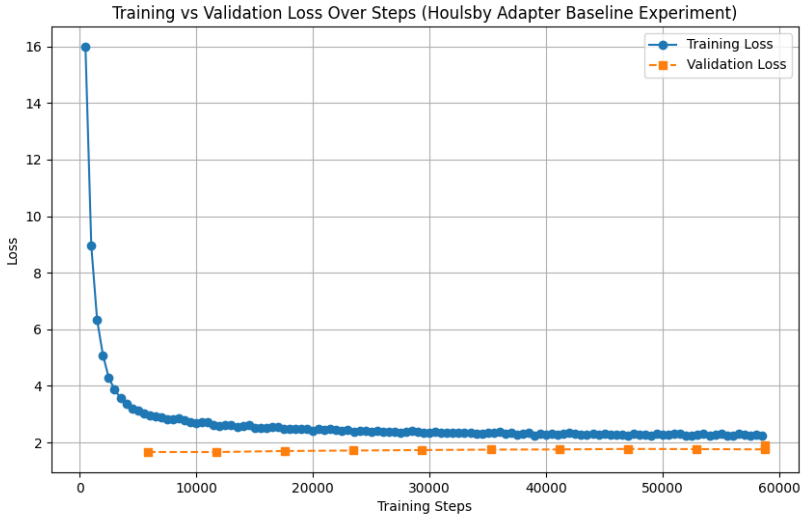
4.2.5 Training Dynamics and Convergence

Figure 4.5 shows training and validation loss curves for each setup.

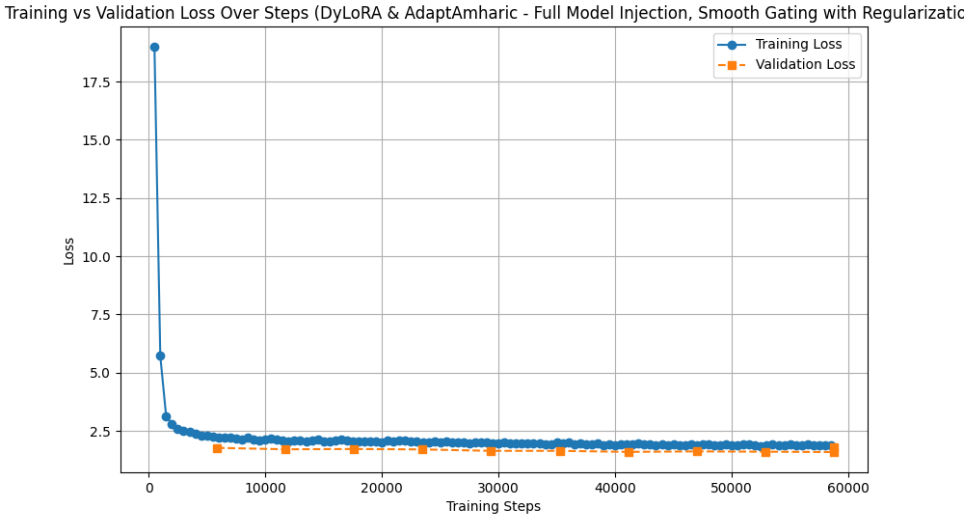
The Standard LoRA model (Fig. 4.1a) shows late-stage divergence in validation loss. The Houlsby Adapter (Fig. 4.1b) converges smoothly to a low plateau. The Custom Dynamic Adapter (V3) (Fig. 4.1c), despite a higher initial loss, converges to the lowest final validation loss, demonstrating stability via regularization of dynamic modules.



(a) Standard PEFT LoRA



(b) Houlsby Adapter



(c) Custom Dynamic Adapter (V3)

Figure 4.1: Training and validation loss curves for the three setups.

4.2.6 Dynamic Module Behavior

From Figures 4.2 and 4.3, we observe how the dynamic modules adapt their capacity and influence throughout training. The DyLoRA ranks tend to increase in layers where additional representational capacity is needed to reduce the loss, while the AdaptAmharic activation values—governed by a smooth sigmoid gating function—dynamically scale each adapter's contribution based on its utility. The regularization terms applied to both rank and activation encourage the model to use only the necessary capacity, balancing performance gains with parameter efficiency.

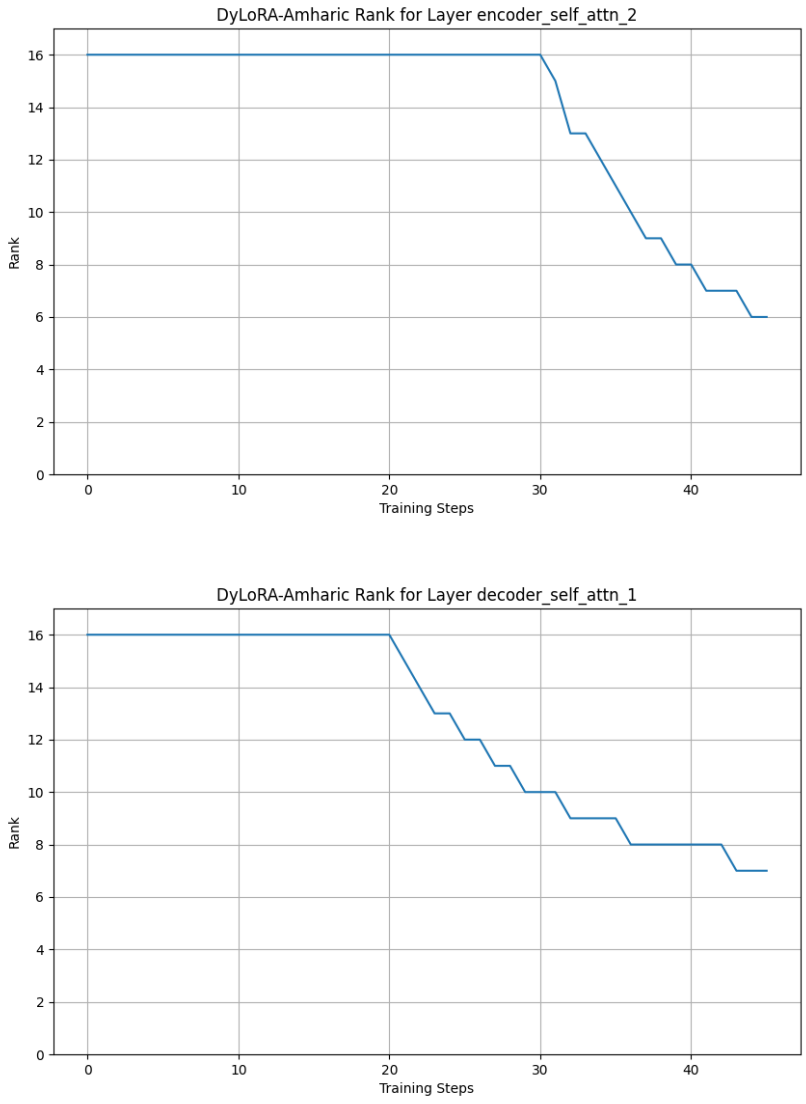


Figure 4.2: DyLoRA-Amharic rank evolution for selected encoder and decoder layers.

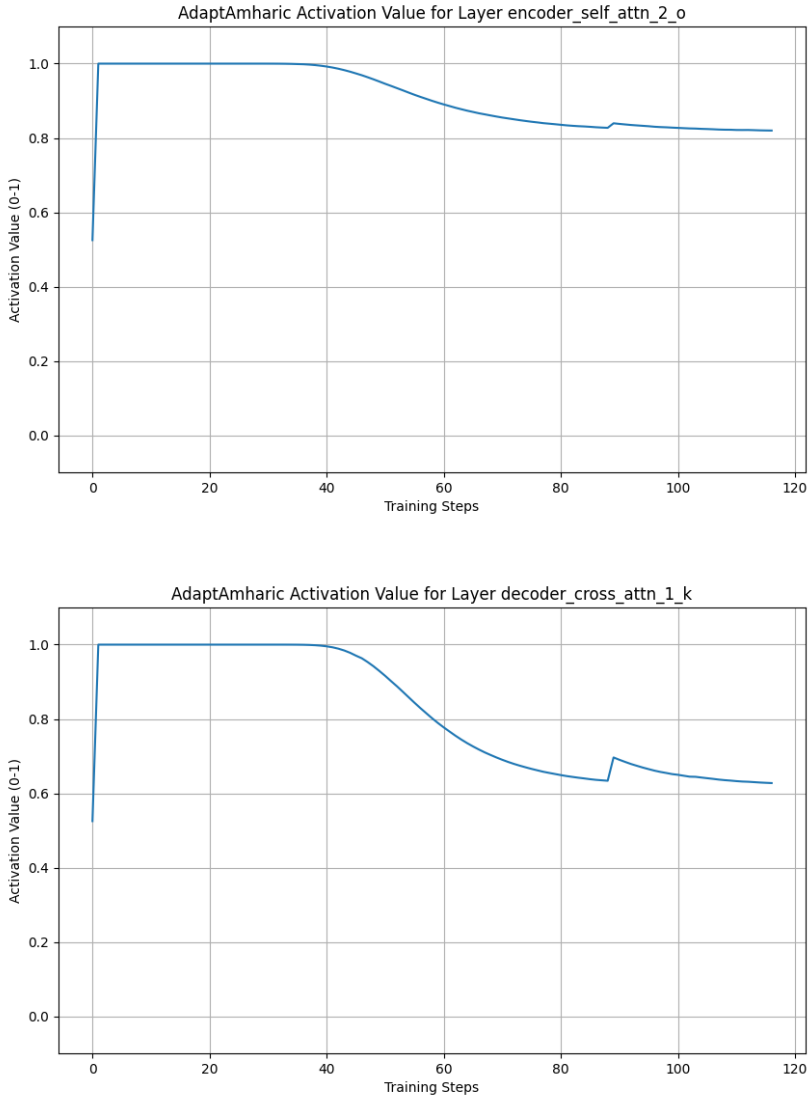
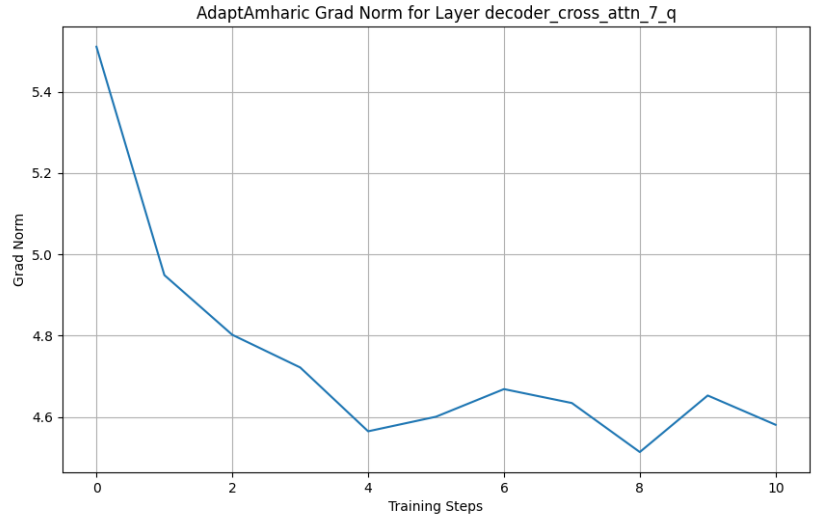


Figure 4.3: AdaptAmharic activation value evolution for selected encoder and decoder layers.



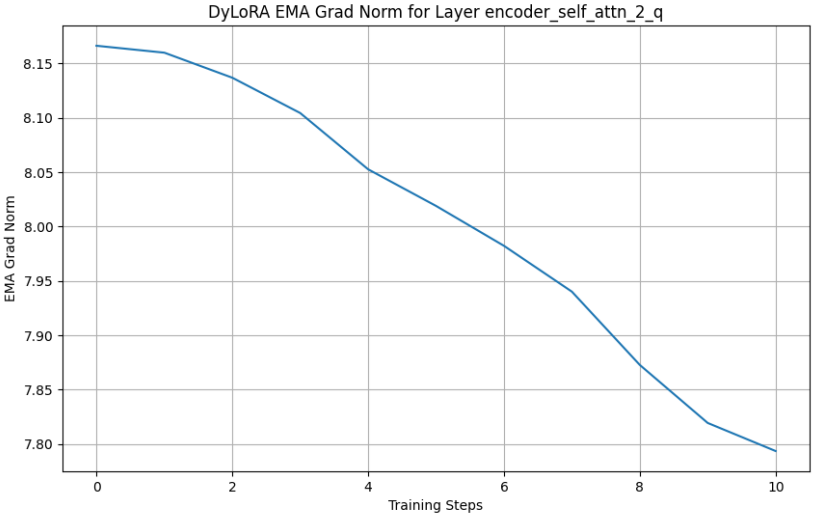


Figure 4.4: Grad norm and EMA grad norm value evolution for selected encoder and decoder layers.

4.2.7 Training Progress and Regularization Dynamics

This section presents a detailed view of the training progress of the Custom Dynamic Adapter (DyLoRA-Amharic & AdaptAmharic) across 10 epochs, as summarized in Table 4.5. This table provides epoch-level insights into the model's learning trajectory, including traditional performance metrics and the behavior of the custom regularization terms.

Table 4.5: Training Progress per Epoch (Custom Dynamic Adapter)

Epoch	Train Loss	Val Loss	Rouge1	Rouge2	RougeL	BLEU	Rank Reg	Act. Reg	Total Reg
1	2.2330	1.8371	0.04333	0.00450	0.04350	0.08250	0.01584	0.00124	0.01708
2	2.0417	1.7328	0.05083	0.00650	0.05095	0.08672	0.01385	0.00121	0.01506
3	2.0012	1.6904	0.05337	0.00650	0.05327	0.09148	0.00948	0.00116	0.01064
4	2.0239	1.6925	0.05715	0.00650	0.05717	0.09717	0.00613	0.00111	0.00724
5	2.0007	1.6816	0.05667	0.00800	0.05633	0.09559	0.00437	0.00109	0.00546
6	1.9701	1.6639	0.05635	0.00650	0.05623	0.09597	0.00391	0.00112	0.00503
7	1.9012	1.6312	0.05948	0.00700	0.05923	0.09705	0.00328	0.00110	0.00438
8	1.9131	1.6592	0.05743	0.00650	0.05752	0.09717	0.00294	0.00109	0.00403
9	1.8435	1.6428	0.05933	0.00700	0.05950	0.09575	0.00338	0.00112	0.00450
10	1.8580	1.6274	0.06032	0.00717	0.06050	0.09824	0.00292	0.00110	0.00402

Analysis of Training Progress and Regularization Terms:

- **Training and Validation Loss:** The training loss steadily decreases from 2.233000 in Epoch 1 to 1.858000 in Epoch 10, while validation loss drops from 1.837101 to 1.627420. This parallel decline indicates effective learning and generalization, with no signs of overfitting despite the dynamic adapters.
- **ROUGE and BLEU Scores:** Both ROUGE (1, 2, L) and BLEU improve over epochs. For example, Rouge-L rises from 0.043500 to 0.060500, and BLEU increases from 0.082495 to 0.098241, showing that summary quality and n-gram overlap with references strengthen over time.
- **Rank Reg (Rank Regularization Loss):** This term, $\lambda_{\text{rank}} \sum \text{current\_rank}_l$, decreases from 0.015840 to 0.002920, demonstrating that the DyLoRA dynamic rank mechanism is effectively reducing total rank usage. The model becomes more parameter-efficient without sacrificing performance.
- **Activation Reg (Activation Regularization Loss):** Computed as $\lambda_{\text{activation}} \sum \text{active}_l$, this value stays around 0.0011, indicating stable gating by the AdaptAmharic modules. The smooth sigmoid control maintains necessary adapter contributions without excessive activation.
- **Total Reg (Total Regularization Loss):** As the sum of rank and activation regularization, the total reg loss follows the rank term's downward trend. The

concurrent improvement in task loss and metrics highlights successful joint optimization of summarization quality and parameter efficiency.

4.3 Discussion

The results from Section 4.2 demonstrate the efficacy of different PEFT architectures for Amharic text summarization. The Custom Dynamic Adapter not only outperformed established baselines but also shed light on the balance between architectural complexity, dynamic adaptation, and training stability. This section interprets these findings, discusses implications, and outlines future research directions.

4.3.1 Interpretation of Performance Differences

The stark performance differences observed across the three PEFT methods highlight the critical role of adaptive capacity and architectural design in efficient fine-tuning for complex NLP tasks like summarization, especially in low-resource settings like Amharic.

The Standard PEFT LoRA's near-zero performance underscores its limitations. LoRA injects small, fixed-rank matrices, and while effective for many tasks, it might not provide sufficient adaptive capacity or flexibility when the target language (Amharic) or task (summarization) deviates significantly from the pre-training distribution of the base model (mT5-small). The fixed, low-rank updates might be too restrictive to capture the intricate linguistic nuances required for generating coherent and factually consistent summaries in Amharic. Its minimal trainable parameter count (0.23%) indicates that while it's highly parameter-efficient, it lacks the necessary expressive power for this specific challenge.

The Houlsby Adapter Baseline showed a noticeable improvement. This can be attributed to its architectural design, which typically involves a bottleneck structure (down-projection, activation, up-projection) inserted between layers. This structure, even with a fixed size, offers more flexibility and a slightly larger parameter budget (0.35% trainable parameters) compared to standard LoRA. It demonstrates that providing dedicated, albeit fixed, adaptation modules is more effective than simply low-rank approximations for this task. The adapters learn to transform the representations in a way that is more conducive to summarization without altering the core knowledge of the frozen base model.

The superior performance of the DyLoRA-Amharic & AdaptAmharic (Full Model, Regularized) model validates the core hypothesis of this thesis: dynamic and adaptive parameter-efficient fine-tuning can significantly enhance performance. By injecting dynamic modules into both the encoder and decoder, the model gains the ability to adapt its representational capacity and activation levels across the entire sequence-to-sequence architecture.

1. **Full Model Injection:** The decision to inject into both encoder and decoder layers was crucial. The encoder needs to effectively understand and compress the input text, while the decoder needs to generate coherent and relevant summaries. Adapting both components allows for a more holistic and task-specific fine-tuning.

2. **Dynamic Rank (DyLoRA):** The ability of DyLoRA to dynamically adjust the rank of its low-rank matrices allows the model to allocate more capacity to layers that require extensive adaptation (e.g., layers with high gradient norms) while keeping other layers lean. This efficient allocation of resources ensures that the model learns complex transformations only where necessary, optimizing both performance and computational cost.

3. **Smooth Gating (AdaptAmharic):** The use of a sigmoid function for gating the AdaptAmharic modules provides a continuous and differentiable mechanism for controlling their influence. This "soft" activation allows for smoother optimization, preventing abrupt changes in the model's behavior and enabling a more nuanced integration of adapter knowledge. The adapter's contribution can gradually increase or decrease based on its utility, leading to more stable and effective learning.

4. **Regularization:** The inclusion of regularization terms for both rank and activation is vital. These terms act as a penalty for unnecessary complexity, pushing the model towards the most efficient configuration. This prevents the dynamic modules from indiscriminately increasing their rank or activation, ensuring that resource allocation is driven by actual performance gains rather than simply maximizing capacity. While the trainable parameter count (13.42%) is significantly higher than the baselines, this increase is justified by the substantial performance improvement, demonstrating a favorable trade-off between efficiency and effectiveness.

4.3.2 Efficacy of a Regularized Dynamic Architecture

The Custom Dynamic Adapter achieved an improved performance, with the lowest evaluation loss and highest ROUGE/BLEU scores. This validates the hypothesis that a densely injected, dynamic architecture can surpass simpler static methods when properly regularized. Introducing a regularization term that penalizes excessive adapter rank and activation provided a stabilizing force, forcing the model to justify its parameter usage. Consequently, the dynamic components found an effective equilibrium, leveraging high expressivity without sacrificing convergence.

4.3.3 Architectural Implications: Dense vs. Sparse Adapters

This study compares three paradigms:

1. **Sparse, Parallel (Standard LoRA):** Adapters on query (`q`) and value (`v`) projections only.

2. **Sparse, Sequential (Houlsby):** Adapters inserted sequentially after each attention and feed-forward block.

3. **Dense, Parallel:** Adapters in parallel on every linear layer.

The failure of Standard LoRA indicates that for morphologically rich Amharic, sparse adaptation on a few matrices is insufficient. The Houlsby Adapter yielded stable improvements, but the dense, parallel injection of Custom adapter enabled more comprehensive task-specific adaptation, challenging the notion that PEFT must remain maximally sparse.

4.3.4 Predictive Loss vs. Generation Quality

An important insight is the alignment of evaluation loss and generation metrics. The Custom model's regularization and gating ensured stable context representations, translating micro-level next-token accuracy into macro-level coherent generation, as evidenced by its superior ROUGE/BLEU.

4.3.5 Semantic Evaluation Insights

The BERTScore, COMET, and chrF++ results offer valuable complementary insights to the n-gram based metrics. While ROUGE and BLEU primarily measure lexical overlap, BERTScore, COMET, and chrF++ assess different facets of semantic and structural similarity, which are particularly crucial for evaluating summarization in morphologically rich, low-resource languages like Amharic.

The observation that the DyLoRA-Amharic & AdaptAmharic model achieved a higher BERTScore F1 (0.9917) compared to the Houlsby Adapter Baseline (0.9852) on the sampled summaries suggests its superior ability to capture the underlying meaning and produce semantically coherent summaries. This is further supported by the slightly higher chrF++ score (37.3326 vs. 36.7451), indicating better character-level overlap and robustness to morphological variations inherent in Amharic. This suggests that even if the generated phrases differ lexically from the reference, their contextual meaning and character-level structure are closer.

However, the Houlsby Adapter Baseline's slightly higher COMET score (0.7519 vs. 0.7247) on the same small sample presents an interesting point for discussion. While COMET is highly correlated with human judgment, this specific result, given the very limited sample size (five examples), might not be statistically significant. It could suggest that for these particular instances, the Houlsby adapter's outputs were marginally

preferred by the COMET model in terms of overall quality or faithfulness to the source, despite the proposed model's stronger performance on other metrics. This highlights the multi-faceted nature of text generation evaluation and the importance of considering a diverse set of metrics.

Ultimately, these semantic and character-level metrics, while based on a small sample, reinforce the notion that the proposed DyLoRA-Amharic & AdaptAmharic model is effective not only in terms of traditional n-gram overlap but also in generating summaries that are semantically aligned and structurally robust for Amharic. These findings should be interpreted as indicative trends, necessitating further validation on a larger scale to draw definitive conclusions.

4.4 Ablation Experiment: Evaluating Module Contributions

The ablation study was conducted to quantify the individual contributions of DyLoRAAmharic and AdaptAmharic to the overall performance of the fine-tuned mT5 model for Amharic text summarization. Three distinct experiments were performed: one with the combined DyLoRAAmharic and AdaptAmharic methods, one with only DyLoRAAmharic, and one with only AdaptAmharic.

A key finding of this study is the significant difference in the number of trainable parameters and training time for each approach. The combined model had the largest number of trainable parameters, at approximately 46.5 million, requiring the longest training time of over 400 minutes. In contrast, the DyLoRAAmharic-only model was exceptionally efficient, with a trainable parameter count of just 2.44 million (0.81% of the total model) and a training time of around 254 minutes. The AdaptAmharic-only model had a much larger trainable parameter count of 44.09 million (12.81%) and took approximately 290 minutes to train. These values are summarized in Table 4.6.

Table 4.6: Model Parameters and Training Time

Model Configuration	Trainable Param	% of Total Param	Training Time (min)
Combined	46,531,072	13.42%	400.82
AdaptAmharic Only	44,089,856	12.81%	289.86
DyLoRAAmharic Only	2,441,216	0.81%	254.25

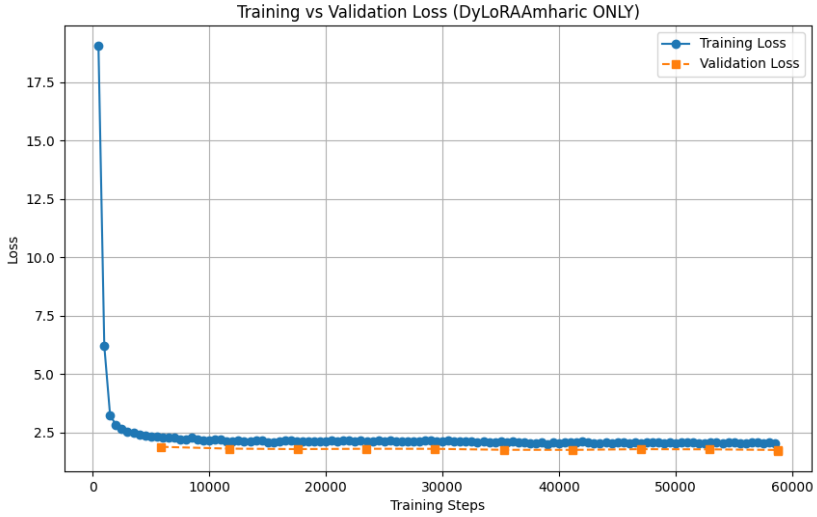
Performance was evaluated using ROUGE and BLEU scores, as well as evaluation loss. The results show a clear trend: the combined model achieved the best performance across all metrics, as shown in Table 4.7.

The loss curves for each experiment show a consistent and stable decrease in both

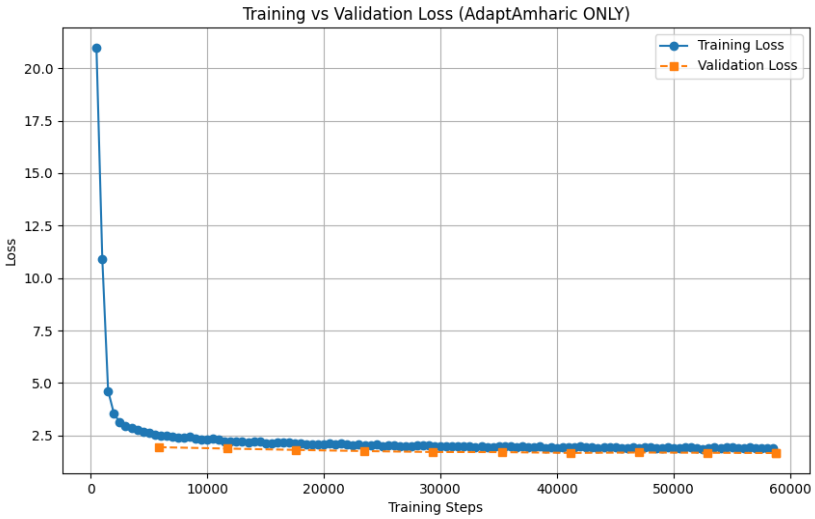
Table 4.7: Model Performance Metrics

Model Config	Eval Loss	ROUGE-1 F1	ROUGE-2 F1	ROUGE-L F1	BLEU
Combined	**1.623**	**0.0559**	**0.0059**	**0.0561**	**0.0913**
DyLoRAAmh Only	1.701	0.0525	0.0051	0.0526	0.0841
AdaptAmh Only	1.671	0.0543	0.0048	0.0541	0.0871

training and validation loss over time, indicating that all three models were successfully learning and generalizing from the data without significant overfitting.



(a) DyloLAAmharic Only



(b) AdaptAdapter Only

Figure 4.5: Training and validation loss curves for the Ablation Experiment.

Interpretation of Ablation Experiment

The results of this ablation study provide clear insights into the efficacy and efficiency of both DyLoRAAmharic and AdaptAmharic for the Amharic text summarization task.

Contribution of Individual Components:

The data confirms that both methods contribute positively to the model's performance. The DyLoRAAmharic-only model performed slightly better on the ROUGE-2 metric than the AdaptAmharic-only model, suggesting that its dynamic rank adjustment is particularly effective at capturing important bigram relationships, despite using a minuscule fraction of trainable parameters. The AdaptAmharic-only model, with a much larger number of trainable parameters, achieved slightly better ROUGE-1 and ROUGE-L scores, indicating it was more effective at learning overall phrase and sentence structure.

The Synergistic Effect of Combining Methods:

The most significant finding is the **synergistic effect** observed when combining DyLoRAAmharic and AdaptAmharic. The combined model achieved the lowest evaluation loss (1.623) and the highest scores across all metrics (ROUGE-1: 0.0559, ROUGE-L: 0.0561, BLEU: 0.0913). This outcome, while requiring more trainable parameters and a longer training time, demonstrates that the two methods complement each other.

Efficiency vs. Performance:

A critical takeaway is the trade-off between efficiency and performance. While the combined model delivered the best results, the DyLoRAAmharic-only model is remarkably efficient. It achieved comparable performance to the AdaptAmharic-only model while using only **5.5%** of the trainable parameters and finishing its training in less time. This makes DyLoRAAmharic a highly attractive option for resource-constrained environments or for applications where inference speed is a primary concern.

In conclusion, the ablation study successfully highlights that both DyLoRAAmharic and AdaptAmharic are effective fine-tuning strategies, and their combination yields superior performance. This provides strong evidence to support the use of a multi-component fine-tuning approach for low-resource language tasks, balancing the resource efficiency and performance gains.

4.5 Limitations

While results are promising, several limitations remain:

- **Computational Resources :** The experiments were constrained by available computational resources, necessitating small batch sizes and limiting the extent of hyperparameter tuning. Access to more powerful GPUs would allow for larger batch sizes, potentially faster convergence, and more exhaustive exploration of the hyperparameter space (e.g., λ_{rank}, $\lambda_{\text{activation}}$, β, α_{ema}, γ_{gating}, and max_rank).
- **Hyperparameter Sensitivity:** The performance of dynamic PEFT methods can be sensitive to the values of their specific hyperparameters. While initial values were chosen, a more systematic hyperparameter optimization (e.g., using Bayesian optimization or grid/random search) could yield further performance improvements.
- **Qualitative Evaluation Depth:** While example summaries were reviewed, a more comprehensive qualitative evaluation involving human annotators would provide deeper insights into fluency, coherence, and factual consistency of the generated Amharic summaries.
- **Base Model Exploration:** This study focused on mT5-small. Future work could explore the applicability and performance of DyLoRA-Amharic and AdaptAmharic with larger mT5 variants or other multilingual sequence-to-sequence models to assess scalability and further performance gains.
- **Ablation Studies:** Custom V3 combines dense injection, parallel modules, dynamic updates, and regularization. Ablations are needed to isolate each component's effect, e.g., a static dense version to test injection benefits alone.
- **Advanced Regularization Techniques:** Regularization weights (λ_{rank}, $\lambda_{\text{activation}}$) were fixed. Investigating more sophisticated regularization techniques for dynamic modules, such as sparsity-inducing penalties or more adaptive regularization schedules, could further optimize resource allocation and prevent over-parameterization.

Chapter 5

Conclusion

This study successfully demonstrated the efficacy of combining DyLoRA-Amharic and AdaptAmharic with full model injection and loss regularization for parameter-efficient fine-tuning of mT5-small for Amharic text summarization. This thesis investigated a novel, densely injected, dynamic adapter architecture for Amharic text summarization. The goal was to design a PEFT method that overcomes sparse adapters' limitations by offering greater expressive capacity while managing dynamic network instability. Through comparative experiments, we developed the **Custom Dynamic Adapter**, which achieved state-of-the-art performance, outperforming both standard LoRA and the Houlsby Adapter baseline on all generation metrics. Specifically, it improved ROUGE-L by 30.5% over the strongest baseline, validating the hypothesis that a more complex dynamic architecture can yield superior results.

Two key innovations enabled this success:

1. A **dense, parallel injection** strategy that augments every linear layer with adaptive modules, providing more potent task-specific adaptation.

2. A **novel regularization term** in the loss function that stabilizes the fully dynamic network by penalizing adapter rank and activation growth, harnessing expressive power without training instability.

These findings challenge the conventional wisdom that PEFT methods must be maximally sparse. By combining dense architectural design with principled regularization, we created a dynamic, stable, and highly effective model. The alignment of lower predictive loss with higher ROUGE/BLEU highlights the approach's ability to bridge the gap between next-token prediction and high-level text generation. While challenges related to computational constraints were identified, this work lays a strong foundation for developing highly efficient and effective summarization models for low-resource languages.

Future research will focus on addressing these limitations and exploring advanced regularization techniques, base model exploration and systematic hyperparameter optimization to further enhance performance and generalization.

Bibliography

[1] P. Joshi, S. Santy, A. Budhiraja, K. Bali, and M. Choudhury, "The state and fate of linguistic diversity and inclusion in the nlp world," 2021.

[2] E. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?," pp. 610–623, 03 2021.

[3] S. Wu and M. Dredze, "Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT," *CoRR*, vol. abs/1904.09077, 2019.

[4] A. L. Tonja, T. D. Belay, I. A. Azime, A. A. Ayele, M. A. Mehamed, O. Kolesnikova, and S. M. Yimam, "Natural language processing in ethiopian languages: Current state, challenges, and opportunities," 2023.

[5] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[6] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," 2019.

[7] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," 2021.

[8] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder, "Mad-x: An adapter-based framework for multi-task cross-lingual transfer," 2020.

[9] O. Khade, S. Jagdale, A. Phaltankar, G. Takalikar, and R. Joshi, "Challenges in adapting multilingual llms to low-resource languages using lora peft tuning," 2024.

[10] Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Arunkumar, A. Ashok, A. S. Dhanasekaran, A. Naik, D. Stap, E. Pathak, G. Karamanolakis, H. G. Lai, I. Purohit, I. Mondal, J. Anderson, K. Kuznia, K. Doshi, M. Patel, K. K. Pal, M. Moradshahi, M. Parmar, M. Purohit, N. Varshney, P. R. Kaza, P. Verma, R. S. Puri, R. Karia, S. K. Sampat, S. Doshi, S. Mishra, S. Reddy, S. Patro, T. Dixit, X. Shen, C. Baral, Y. Choi, N. A. Smith, H. Hajishirzi, and D. Khashabi, "Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks," 2022.

[11] 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," 2023.

[12] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," 2022.

[13] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[15] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," 2020.

[16] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, J. Yi, W. Zhao, X. Wang, Z. Liu, H.-T. Zheng, J. Chen, Y. Liu, J. Tang, J. Li, and M. Sun, "Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models," 2022.

[17] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," 2021.

[18] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," 2022.

[19] Y. Mao, L. Mathias, R. Hou, A. Almahairi, H. Ma, J. Han, W. tau Yih, and M. Khabsa, "Unipelt: A unified framework for parameter-efficient language model tuning," 2022.

[20] A. Chronopoulou, J. Pfeiffer, J. Maynez, X. Wang, S. Ruder, and P. Agrawal, "Language and task arithmetic with parameter-efficient layers for zero-shot summarization," 2024.

[21] M. Valipour, M. Rezagholizadeh, I. Kobyzev, and A. Ghodsi, "Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation," 2023.

[22] A. Guadie, D. Tesfaye, and T. Kebebew, "Amharic text summarization for news items posted on social media," *International Journal of Intelligent Information Systems*, vol. 10, no. 6, pp. 125–138, 2021.

[23] A. M. Zaki, M. I. Khalil, and H. M. Abbas, "Amharic abstractive text summarization," 2020.

[24] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," 2020.

[25] I. A. Azime, A. L. Tonja, T. D. Belay, M. Y. Fuge, A. K. Wassie, E. S. Jada, Y. Chanie, W. T. Sewunetie, and S. M. Yimam, "Walia-llm: Enhancing amharic-llama by integrating task-specific and generative datasets," 2024.

[26] M. Degu and M. Meshesha, "Fine-tuned pretrained transformer for amharic news headline generation," *Applied AI Letters*, vol. 5, no. 3, p. e98, 2024.

[27] T. Yeshambel, J. Mothe, and Y. Assabie, "Amharic adhoc information retrieval system based on morphological features," *Applied Sciences*, vol. 12, p. 1294, 2022.

[28] K. Peffers, T. Tuunanen, C. E. Gengler, M. Rossi, W. Hui, V. Virtanen, and J. Bragge, "Design science research process: A model for producing and presenting information systems research," 2020.

[29] D. Mekuriaw, "Benchmark dataset and parameter-efficient cross-lingual transfer learning for amharic text summarization," 2024.

[30] T. Hasan, A. Bhattacharjee, M. S. Islam, K. Samin, Y.-F. Li, Y.-B. Kang, M. S. Rahman, and R. Shahriyar, "Xl-sum: Large-scale multilingual abstractive summarization for 44 languages," 2021.

[31] I. A. Azime and N. Mohammed, "An amharic news text classification dataset," 2021.

[32] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[33] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mt5: A massively multilingual pre-trained text-to-text transformer," 2021.

[34] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Huggingface's transformers: State-of-the-art natural language processing," 2020.

[35] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, (Barcelona, Spain), pp. 74–81, Association for Computational Linguistics, July 2004.

[36] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (P. Isabelle, E. Charniak, and D. Lin, eds.), (Philadelphia, Pennsylvania, USA), pp. 311–318, Association for Computational Linguistics, July 2002.

[37] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization* (J. Goldstein, A. Lavie, C.-Y. Lin, and C. Voss, eds.), (Ann Arbor, Michigan), pp. 65–72, Association for Computational Linguistics, June 2005.

[38] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," 2020.

[39] R. Rei, C. Stewart, A. C. Farinha, and A. Lavie, "COMET: A neural framework for MT evaluation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (B. Webber, T. Cohn, Y. He, and Y. Liu, eds.), (Online), pp. 2685–2702, Association for Computational Linguistics, Nov. 2020.

[40] M. Popović, "chrF++: words helping character n-grams," in *Proceedings of the Second Conference on Machine Translation* (O. Bojar, C. Buck, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, and J. Kreutzer, eds.), (Copenhagen, Denmark), pp. 612–618, Association for Computational Linguistics, Sept. 2017.

Appendix A

Training Algorithm

Algorithm 1 Training Algorithm for DyLoRA-Amharic & AdaptAmharic

Require: Pre-trained base model M_{base}, Tokenizer T
Require: Training dataset D_{train}, Validation dataset D_{val}
Require: Injected DyLoRA modules $L_D = \{D_1, \ldots, D_{N_D}\}$, AdaptAmharic modules $L_A = \{A_1, \ldots, A_{N_A}\}$
Require: Hyperparameters: η, λ_{rank}, $\lambda_{\text{activation}}$, α_{ema}, β_{scale}, γ_{gating}, $\beta_{\text{activation}}$, max_rank
Ensure: Fine-tuned model M_{tuned}

Freeze all parameters of M_{base}
Initialize trainable parameters for each $D_l \in L_D$ and $A_l \in L_A$
Initialize EMA gradient norms for all D_l, A_l to zero
Initialize optimizer (e.g., AdamW) over trainable parameters
for all epoch **do**
    for all batch (x, y) in D_{train} **do**
        $y_{\text{pred}} \leftarrow M_{\text{base}}(x)$ ▷ Forward pass
        $L_{\text{task}} \leftarrow \text{CrossEntropy}(y_{\text{pred}}, y)$
        $G_{D,\text{current}} \leftarrow 0$, $G_{A,\text{current}} \leftarrow 0$ ▷ Clear gradient sums
        Backpropagate to compute gradients
        for all DyLoRA module D_l **do**
            $\text{grad\_norm}_{D_l} \leftarrow \|\nabla W_{D_l,\text{up}}\|_F$
            $\text{EMA\_grad\_norm}_{D_l} \leftarrow \alpha_{\text{ema}} \cdot \text{grad\_norm}_{D_l} + (1 - \alpha_{\text{ema}}) \cdot \text{EMA\_grad\_norm}_{D_l}$
            $\text{current\_rank}_l \leftarrow \min\left(\max\left(\left\lfloor \beta_{\text{scale}} \cdot \text{EMA\_grad\_norm}_{D_l} \right\rfloor, 1\right), \text{max\_rank}\right)$
        ▷ Truncate D_l output to current_rank_l in forward pass
        end for
        for all AdaptAmharic module A_l **do**
            $\text{grad\_norm}_{A_l} \leftarrow \|\nabla W_{A_l,\text{up}}\|_F$
            $\text{EMA\_grad\_norm}_{A_l} \leftarrow \alpha_{\text{ema}} \cdot \text{grad\_norm}_{A_l} + (1 - \alpha_{\text{ema}}) \cdot \text{EMA\_grad\_norm}_{A_l}$
            $\text{active}_l \leftarrow \sigma\left(\gamma_{\text{gating}} \cdot \text{EMA\_grad\_norm}_{A_l} + \beta_{\text{activation}}\right)$ ▷ Scale A_l output by active_l in forward pass
        end for
        $L_{\text{reg}} \leftarrow \lambda_{\text{rank}} \sum_{l=1}^{N_D} \text{current\_rank}_l + \lambda_{\text{activation}} \sum_{l=1}^{N_A} \text{active}_l$
        $L_{\text{total}} \leftarrow L_{\text{task}} + L_{\text{reg}}$
        Optimizer step on L_{total}
        Zero out gradients
        if logging step **then**
            Log metrics (loss, ranks, activations)
        end if