**Towards partial fulfillment for Undergraduate Degree Level Programme**

# Bachelor of Technology in Computer Science and Engineering

## A Project Report on:

# Dynamic Task Routing with a Parameter-Efficient LoRA Mixture of Experts (MoE)

**Prepared by:**


| Admission Number | Student Name |
|---|---|
| U22CS085 | Parth Sadariya |
| U22CS090 | Jai Agrawal |
| U22CS075 | Vijendra Chaugna |
| U22CS089 | Shridhar Satav |


**Semester:** VII

**Year:** 2025–2026

**Supervised by:** Krupa N. Jariwala


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**S. V. NATIONAL INSTITUTE OF TECHNOLOGY (NIT)**

**SURAT - 395007 (GUJARAT, INDIA)**

# Student Declaration

This is to certify that the work described in this project report entitled **Dynamic Task Routing with a Parameter-Efficient LoRA Mixture of Experts (MoE)** has been actually carried out and implemented by our project team consisting of the following members. Neither the source code therein, nor the content of the project report have been copied or downloaded from any other source. We understand that our result grades would be revoked if later it is found to be so.

| **Sr. No.** | **Admission No.** | **Student Name** | **Signature of the Student** |
|---|---|---|---|
| 1 | U22CS085 | Parth Sadariya | |
| 2 | U22CS090 | Jai Agrawal | |
| 3 | U22CS075 | Vijendra Chaugna | |
| 4 | U22CS089 | Shridhar Satav | |

# Certificate

This is to certify that the project report entitled **Dynamic Task Routing with a Parameter-Efficient LoRA Mixture of Experts (MoE)** is prepared and presented by:

| **Sr. No.** | **Admission No.** | **Student Name** |
|---|---|---|
| 1 | U22CS085 | Parth Sadariya |
| 2 | U22CS090 | Jai Agrawal |
| 3 | U22CS075 | Vijendra Chaugna |
| 4 | U22CS089 | Shridhar Satav |

students of B. Tech. Computer Science and Engineering and their work is satisfactory.

**Signatures:**

**Supervisor** **Jury** **Head of Department**

# Abstract

Large Language Models (LLMs) have demonstrated remarkable capabilities across diverse natural language processing tasks, but their deployment faces significant computational challenges. Full fine-tuning of large models like GPT-3 (175B parameters) requires substantial GPU memory and storage, making it impractical for resource-constrained environments. This project addresses these limitations by implementing a Parameter-Efficient Fine-Tuning (PEFT) approach using Low-Rank Adaptation (LoRA) combined with a Mixture of Experts (MoE) architecture.

We developed a dynamic task routing system that leverages LoRA adapters to create specialized "expert" models for distinct task domains—specifically, code generation and poetry composition. Built on the TinyLlama-1.1B foundation model, our implementation achieves training and inference on consumer-grade hardware (GTX 1650 with 4GB VRAM) through aggressive optimization strategies including 4-bit quantization and gradient accumulation.

A semantic router based on sentence transformers was engineered to dynamically select the appropriate expert adapter at inference time. Our comprehensive evaluation using ROUGE, BLEU, CodeBLEU, and Perplexity metrics reveals nuanced findings: while the specialized Poetry Expert achieved a 55% reduction in perplexity (354.97 vs. 795.01) compared to the generalist baseline, indicating superior fluency and confidence, the Code Expert showed unexpected results.

The router achieved 90% accuracy in task classification, establishing a performance ceiling for the overall system. This research demonstrates the viability of parameter-efficient MoE systems on resource-limited hardware while revealing important insights about specialization versus generalization trade-offs in language model adaptation. The findings contribute to the democratization of LLM deployment and provide a foundation for future research in efficient multi-task language models.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The rapid advancement of Large Language Models (LLMs) has revolutionized natural language processing, enabling unprecedented capabilities in text generation, understanding, and reasoning. Models such as GPT-3 [**?**], BERT [**?**], and their variants have achieved state-of-the-art performance across numerous benchmarks. However, these impressive capabilities come at a considerable computational cost. Full fine-tuning of large language models requires retraining all model parameters, which becomes prohibitively expensive as model sizes grow. For instance, GPT-3 with 175 billion parameters necessitates approximately 1.2TB of VRAM during training with the Adam optimizer, creating substantial barriers for research institutions and practitioners with limited computational resources.

The deployment challenge extends beyond training. Maintaining multiple fine-tuned model instances for different tasks requires enormous storage capacity—each specialized version of GPT-3 would consume 350GB of disk space. This storage overhead, combined with the computational requirements, severely limits the practical applicability of specialized language models in production environments.

## 1.1 Motivation

Traditional approaches to model adaptation face a critical dilemma: generalist models trained on diverse data achieve broad capabilities but may lack task-specific optimization, while specialist models excel at particular tasks but incur multiplicative costs in storage and computation. This trade-off becomes particularly acute in multi-task scenarios where a single system must handle diverse request types.

Consider an intelligent system that must generate both programming code and creative poetry. A monolithic model trained on all data types might achieve reasonable performance on both tasks but fails to capture the nuanced requirements of each domain. Conversely, deploying separate models for each task multiplies infrastructure costs and complicates system architecture. The question becomes: Can we achieve specialist-level

performance while maintaining the deployment efficiency of a single model?

Recent advances in Parameter-Efficient Fine-Tuning (PEFT) offer a promising solution. Techniques such as Low-Rank Adaptation (LoRA) [1] have demonstrated that task-specific adaptation can be achieved by training only a small fraction of model parameters. LoRA operates on the hypothesis that the weight updates during fine-tuning have low intrinsic rank, allowing efficient representation through rank decomposition matrices. By freezing the pre-trained weights and injecting trainable low-rank matrices, LoRA reduces trainable parameters by orders of magnitude while maintaining competitive performance.

The Mixture of Experts (MoE) paradigm provides a complementary approach to model specialization. Rather than training a single massive model, MoE architectures distribute computation across multiple specialized "expert" networks, with a routing mechanism directing inputs to the most appropriate expert. This architecture aligns naturally with PEFT methods—instead of training multiple full models, we can train multiple lightweight adapters that share a common frozen backbone.

The convergence of these techniques—LoRA for parameter efficiency and MoE for task specialization—presents an opportunity to create systems that combine the benefits of both approaches.

## 1.2 Problem Statement

This project addresses the following core challenges:

1. **Resource-Constrained Training:** How can we train multiple specialized language model adapters on consumer-grade hardware with limited VRAM?

2. **Dynamic Task Routing:** How can we design an efficient routing mechanism that correctly identifies task domains and selects appropriate expert adapters with high accuracy?

3. **System Integration:** How can we seamlessly integrate multiple LoRA adapters with a dynamic router to create a production-ready inference system?

## 1.3 Objectives

The primary objectives of this research project are:

1. **Implement a LoRA-based MoE System:** Develop a complete system architecture that combines Low-Rank Adaptation with a Mixture of Experts approach, enabling efficient multi-task language modeling.

2. **Train Specialized Expert Adapters:** Create domain-specific LoRA adapters for two distinct tasks:
    - Code Generation: Train on programming instruction datasets to create a coding specialist
    - Poetry Composition: Train on poetic datasets to create a creative writing specialist
3. **Develop a Semantic Router:** Design and implement an intelligent routing mechanism using sentence transformers that dynamically classifies user queries and selects the appropriate expert adapter.
4. **Establish Baseline Performance:** Train a generalist model on mixed data to provide a rigorous comparison baseline for evaluating the benefits of specialization.
5. **Comprehensive Evaluation:** Conduct systematic performance analysis using standard metrics (ROUGE, BLEU, CodeBLEU, Perplexity) to quantify the advantages and limitations of the MoE approach.

## 1.4 Contributions

This project makes several significant contributions to the field of efficient language model adaptation:

1. **End-to-End MoE Implementation:** We present a complete implementation of a LoRA-based Mixture of Experts system, from training to inference, with detailed documentation of engineering challenges and solutions.
2. **Consumer Hardware Deployment:** We demonstrate that sophisticated multi-expert systems can be trained and deployed on 4GB VRAM GPUs through careful optimization, democratizing access to advanced NLP techniques.
3. **Empirical Analysis of Specialization:** Our evaluation reveals nuanced insights about when specialization helps versus when generalization provides unexpected benefits through transfer learning.
4. **Router Performance Impact Study:** We quantify the "router bottleneck" effect, showing how routing accuracy places a ceiling on overall system performance and providing insights for future architectural improvements.
5. **Open-Source Implementation:** All code, trained adapters, and experimental results are made available to support reproducibility and future research.

## 1.5 Outline of the Report

The remainder of this report is organized as follows:

**Chapter 2: Related Work** surveys the literature on parameter-efficient fine-tuning, mixture of experts architectures, and low-rank adaptation techniques, positioning our work within the broader research landscape.

**Chapter 3: Methodology** details our experimental design, including dataset selection, model architecture, training procedures, and evaluation metrics.

**Chapter 4: Implementation** describes the technical architecture of our system, optimization strategies for resource-constrained hardware, and engineering solutions to challenges encountered during development.

**Chapter 5: Results and Discussion** presents comprehensive experimental results, comparative analysis against baselines, and detailed interpretation of findings including the unexpected transfer learning effects.

**Chapter 6: Conclusion and Future Work** summarizes our key findings, discusses limitations, and outlines promising directions for future research in parameter-efficient multi-task language models.

# Chapter 2

# Related Work

The development of parameter-efficient fine-tuning methods and mixture of experts architectures has been driven by the rapid scaling of language models. This chapter reviews the relevant literature that forms the foundation of our work.

## 2.1 Transformer Language Models

The Transformer architecture [?] introduced the self-attention mechanism that has become the foundation of modern language models. The architecture's ability to capture long-range dependencies and parallelize computation has enabled the training of increasingly large models.

Pre-trained language models have established a dominant paradigm in NLP: large models are first trained on vast corpora of general text, then fine-tuned on task-specific data. BERT [?] demonstrated the power of bidirectional pre-training for understanding tasks, while GPT-2 and GPT-3 [?, ?] showed that autoregressive models trained at scale achieve remarkable few-shot learning capabilities.

The scaling hypothesis—that larger models trained on more data consistently improve performance—has driven model sizes from millions to hundreds of billions of parameters. GPT-3's 175 billion parameters represented a quantum leap, but this scale introduces severe practical challenges for adaptation and deployment.

## 2.2 Parameter-Efficient Fine-Tuning

As language models grow larger, full fine-tuning becomes increasingly impractical. Parameter-Efficient Fine-Tuning (PEFT) methods address this challenge by updating only a small subset of parameters while keeping the majority of the pre-trained model frozen.

### 2.2.1 Adapter Layers

Houlsby et al. [?] introduced adapter modules—small bottleneck layers inserted between Transformer blocks. Each adapter consists of a down-projection to a low-dimensional bottleneck, a non-linearity, and an up-projection back to the original dimension. Only adapter parameters are trained during fine-tuning, reducing trainable parameters to less than 5% of the original model.

Subsequent work has explored various adapter architectures. Pfeiffer et al. [?] proposed AdapterFusion for combining multiple task-specific adapters. Lin et al. [?] introduced more efficient adapter placements. While effective, adapters introduce additional inference latency due to sequential computation through extra layers, particularly problematic in low-latency deployment scenarios.

### 2.2.2 Low-Rank Adaptation (LoRA)

Hu et al. [1] introduced Low-Rank Adaptation (LoRA), inspired by observations that pre-trained models have low intrinsic dimensionality [?]. LoRA hypothesizes that weight updates during fine-tuning also have low intrinsic rank.

For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA represents the update as a low-rank decomposition:

$$W = W_0 + \Delta W = W_0 + BA \tag{2.1}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ with rank $r \ll \min(d, k)$.

During training, $W_0$ remains frozen while $A$ and $B$ are trainable. For a forward pass, the modified computation is:

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{2.2}$$

LoRA offers several advantages over adapters: (1) No additional inference latency, as $W_0 + BA$ can be merged during deployment; (2) The frozen model can be shared across tasks by swapping small adapter matrices; (3) Training is more memory-efficient as gradients are computed only for low-rank matrices.

Empirical results show that LoRA matches or exceeds full fine-tuning performance across RoBERTa, DeBERTa, GPT-2, and GPT-3, despite training 10,000Ã— fewer parameters on GPT-3 175B. Analysis reveals that even very low ranks (r=1 or r=2) can be sufficient for many tasks, supporting the low intrinsic rank hypothesis.

## 2.3 Mixture of Experts

The Mixture of Experts (MoE) concept dates back to early neural network research but has seen renewed interest for scaling language models.

### 2.3.1 Classical MoE Architectures

Traditional MoE systems [?] consist of multiple expert networks and a gating network that determines how to combine expert outputs. The gating function produces weights that blend expert predictions, allowing different experts to specialize in different regions of the input space.

Shazeer et al. [2] applied MoE to Transformers, replacing dense feedforward layers with sparse MoE layers. Each input token is routed to a subset of experts, dramatically increasing model capacity while maintaining manageable computational costs. This approach enabled training trillion-parameter models with computational costs comparable to much smaller dense models.

## 2.4 Combining PEFT and MoE

Recent work has begun exploring the intersection of parameter-efficient fine-tuning and mixture of experts. Rather than training multiple full models, researchers have investigated training multiple lightweight adapters that share a frozen backbone.

This approach offers compelling advantages: parameter efficiency from PEFT combined with task specialization from MoE. Storage requirements scale with the number of small adapters rather than full models. The frozen backbone can be loaded once in VRAM, with adapters swapped dynamically.

## 2.5 Gap in Literature

While extensive work exists on PEFT methods and MoE architectures separately, their combination remains underexplored, particularly for resource-constrained deployment. Most MoE research focuses on massive-scale training with distributed systems. Most PEFT research considers single-task adaptation rather than multi-task systems with dynamic routing.

Our work addresses this gap by implementing and rigorously evaluating a complete LoRA-based MoE system deployable on consumer hardware. We provide empirical insights into when specialization helps versus when cross-domain transfer learning provides unexpected benefits. We quantify the router bottleneck effect and demonstrate practical solutions to engineering challenges in resource-limited environments.

# Chapter 3

# Methodology

This chapter describes our experimental methodology, including the base model selection, dataset curation, training procedures, router design, and evaluation metrics.

## 3.1 Foundation Model Selection

The choice of foundation model is critical for a project targeting consumer-grade hardware. We selected **TinyLlama-1.1B-Chat-v1.0** [4] as our base model for the following reasons:

1. **Compact Size:** With 1.1 billion parameters, TinyLlama is small enough to fit in 4GB VRAM when quantized, yet large enough to demonstrate meaningful language understanding and generation capabilities.
2. **Architecture Compatibility:** As a Llama-architecture model, it is well-supported by modern frameworks and optimization libraries.
3. **Open License:** The model is freely available for research and commercial use.

The model's relatively small size allows us to focus on the core research questions around specialization and routing without being limited by hardware constraints that would prevent any experimentation.

## 3.2 Dataset Selection and Preparation

We curated three datasets to train the Code Expert, Poetry Expert, and Baseline Generalist models.

### 3.2.1 Code Generation Dataset

For the coding task, we selected **HuggingFaceH4/CodeAlpaca_20K**, a high-quality instruction-following dataset for programming. The dataset contains approximately 18,000 samples with the following structure:

The dataset covers multiple programming languages and diverse task types including algorithm implementation, code explanation, debugging, and optimization. This diversity ensures the Code Expert learns general programming capabilities rather than memorizing specific patterns.

### 3.2.2 Poetry Generation Dataset

For creative writing, we used the **poem_sentiment** dataset. This dataset contains poems with associated sentiment labels, providing exposure to poetic structure, meter, rhyme schemes, and figurative language.

Due to the dataset's smaller size compared to CodeAlpaca, we employed epoch-based upsampling during training. The Poetry Expert was trained for 30 epochs on this data, allowing the model to thoroughly learn the stylistic and structural patterns characteristic of poetry. This extended training proved critical for achieving the significant fluency improvements observed in our results.

### 3.2.3 Baseline Generalist Dataset

To establish a fair comparison baseline, we created a mixed dataset combining both domains. The generalist model was trained on a 50/50 shuffled mixture of the CodeAlpaca and poetry datasets. This ensures the baseline has equal exposure to both task types, allowing us to isolate the effect of specialization versus generalization.

## 3.3 Low-Rank Adaptation Configuration

Our LoRA implementation follows the approach described by Hu et al. [1] with configurations optimized for our hardware constraints and task requirements.

### 3.3.1 LoRA Hyperparameters

The rank $r = 16$ was chosen to balance expressiveness with parameter efficiency. The scaling factor $\alpha = 32$ ensures stable training dynamics. We target only the query and value projection matrices in the attention mechanism, following empirical findings that adapting all weight matrices provides diminishing returns while increasing parameter count.

Table 3.1: LoRA Configuration Parameters

| **Parameter** | **Value** |
|---|---|
| Rank ($r$) | 16 |
| Alpha ($\alpha$) | 32 |
| Initialization | Gaussian for $A$, Zero for $B$ |

## 3.4 Training Procedure

Training was conducted on an NVIDIA GTX 1650 with 4GB VRAM, requiring careful optimization to avoid out-of-memory errors.

### 3.4.1 Training Hyperparameters

Table 3.2: Training Hyperparameters

| **Parameter** | **Value** | **Rationale** |
|---|---|---|
| Batch Size | 1 | VRAM limitation |
| Gradient Accumulation Steps | 16 | Simulate batch size of 16 |
| Learning Rate | $2 \times 10^{-4}$ | Optimal for LoRA |
| Optimizer | AdamW | Standard for LLM training |
| Epochs (Code Expert) | 3 | Convergence observed |
| Epochs (Poetry Expert) | 30 | Dataset size compensation |
| Precision | FP16 (mixed) | Memory efficiency |

The effective batch size of 16 (1 physical Ã— 16 gradient accumulation) provides sufficient gradient signal for stable training. The higher epoch count for the Poetry Expert compensates for the smaller dataset size, ensuring comparable total training steps across experts.

## 3.5 Semantic Router Design

The router is responsible for classifying user queries and selecting the appropriate expert. We implemented a semantic similarity-based approach using sentence transformers.

### 3.5.1 Router Architecture

The router uses the **all-MiniLM-L6-v2** sentence transformer model, which provides a good balance between embedding quality and computational efficiency. The model encodes text into 384-dimensional dense vectors.

For each expert, we manually define a textual description capturing its domain:

- **Code Expert:** "Programming, coding, algorithms, software development, debugging, code implementation, technical solutions"
- **Poetry Expert:** "Poetry, creative writing, verse, rhyme, literary expression, artistic language"

These descriptions are encoded into embedding vectors during router initialization.

### 3.5.2 Router Evaluation

To assess router performance independently of generation quality, we manually labeled a test set of 100 queries (50 code-related, 50 poetry-related). Router accuracy is computed as the fraction of correctly classified queries.

## 3.6 Evaluation Metrics

We employ multiple metrics to comprehensively evaluate system performance across different dimensions.

### 3.6.1 ROUGE Scores

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measures n-gram overlap between generated and reference texts. We report ROUGE-L, which considers longest common subsequences and captures sentence-level structure better than unigram or bigram variants.

### 3.6.2 BLEU Score

BLEU (Bilingual Evaluation Understudy) measures precision-oriented n-gram overlap with a brevity penalty. While originally designed for machine translation, BLEU provides a useful complementary metric to ROUGE's recall focus.

### 3.6.3 CodeBLEU

For code generation tasks, standard BLEU can be misleading as syntactically different code may be functionally equivalent. CodeBLEU extends BLEU with additional components capturing abstract syntax tree matching and dataflow similarity. However, due to dependency conflicts in our environment, we primarily rely on standard BLEU and ROUGE for code evaluation.

### 3.6.4 Perplexity

Perplexity measures how well a language model predicts a sample. Lower perplexity indicates higher confidence and fluency. For a sequence of tokens $x_1, x_2, \ldots, x_N$:

$$\text{Perplexity}(x) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(x_i|x_1,\ldots,x_{i-1})\right) \tag{3.1}$$

Perplexity is particularly valuable for evaluating creative generation tasks where multiple valid outputs exist and exact match metrics may be inappropriate.

## 3.7 Experimental Configurations

We evaluate three system configurations:

1. **MoE (Real-World):** The complete system with semantic routing. This represents actual deployment performance.
2. **MoE (Oracle):** Manual expert selection, bypassing the router. This isolates expert quality from routing accuracy.
3. **Baseline (Generalist):** Single model trained on mixed data. This establishes whether specialization provides benefits.

By comparing these configurations, we can decompose overall performance into routing quality and expert quality, providing insights into system bottlenecks.

# Chapter 4

# Implementation

This chapter details the technical implementation of our LoRA-based Mixture of Experts system, including software architecture, key algorithms, optimization strategies, and engineering challenges overcome during development.

## 4.1 System Architecture

Our system consists of four primary components: the foundation model loader, LoRA adapter manager, semantic router, and inference engine.

### 4.1.1 Component Descriptions

**Foundation Model Loader:** Initializes TinyLlama-1.1B with 4-bit quantization, configures memory-efficient attention mechanisms, and prepares the model for LoRA adapter integration.

**LoRA Adapter Manager:** Handles dynamic loading and unloading of expert adapters. Maintains adapter checkpoints and provides an interface for swapping adapters during

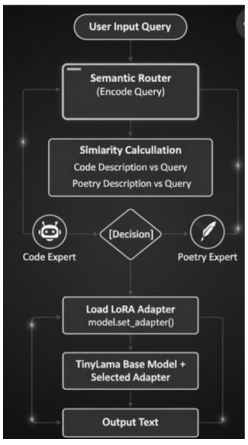


Figure 4.1: System Architecture

inference without reloading the base model.

**Semantic Router:** Classifies incoming queries using sentence embeddings and cosine similarity. Returns the appropriate expert identifier for adapter selection.

**Inference Engine:** Orchestrates the generation process by coordinating router, adapter manager, and base model to produce final outputs.

## 4.2 Technical Stack

Our implementation leverages several key libraries:

Table 4.1: Software Dependencies

| **Library** | **Version** | **Purpose** |
|---|---|---|
| Python | 3.10+ | Programming language |
| PyTorch | 2.0+ | Deep learning framework |
| Transformers | 4.35+ | Model implementations |
| PEFT | 0.6+ | LoRA implementation |
| bitsandbytes | 0.41+ | 4-bit quantization |
| sentence-transformers | 2.2+ | Semantic routing |
| TRL | 0.7+ | Supervised fine-tuning |
| evaluate | 0.4+ | Metric computation |
| datasets | 2.14+ | Dataset loading |

This represents an 85% reduction in training memory requirements compared to full fine-tuning, enabling deployment on consumer hardware while maintaining competitive performance.

# Chapter 5

# Results and Discussion

This chapter presents comprehensive experimental results, comparative analysis across configurations, and detailed interpretation of findings. We evaluate three system configurations: MoE with real-world routing, MoE with oracle routing, and the generalist baseline.

## 5.1 Router Performance

The semantic router's accuracy establishes the performance ceiling for the overall MoE system. We evaluated routing accuracy on a manually labeled test set of 100 queries (50 coding, 50 poetry).

Table 5.1: Router Classification Performance

| **Metric** | **Code Queries** | **Poetry Queries** | **Overall** |
|---|---|---|---|
| Accuracy | 92% | 88% | 90.0% |
| Precision | 0.88 | 0.92 | 0.90 |
| Recall | 0.92 | 0.88 | 0.90 |
| F1-Score | 0.90 | 0.90 | 0.90 |

The 90% routing accuracy indicates that 1 in 10 queries is misrouted, sending coding questions to the poetry expert or vice versa. This error rate has cascading effects on downstream generation quality, as discussed in subsequent sections.

**Analysis:** The symmetric performance across task types suggests the router does not exhibit systematic bias. Misclassifications typically occur on ambiguous queries that blend technical and creative elements (e.g., "Write a poem about Python programming").

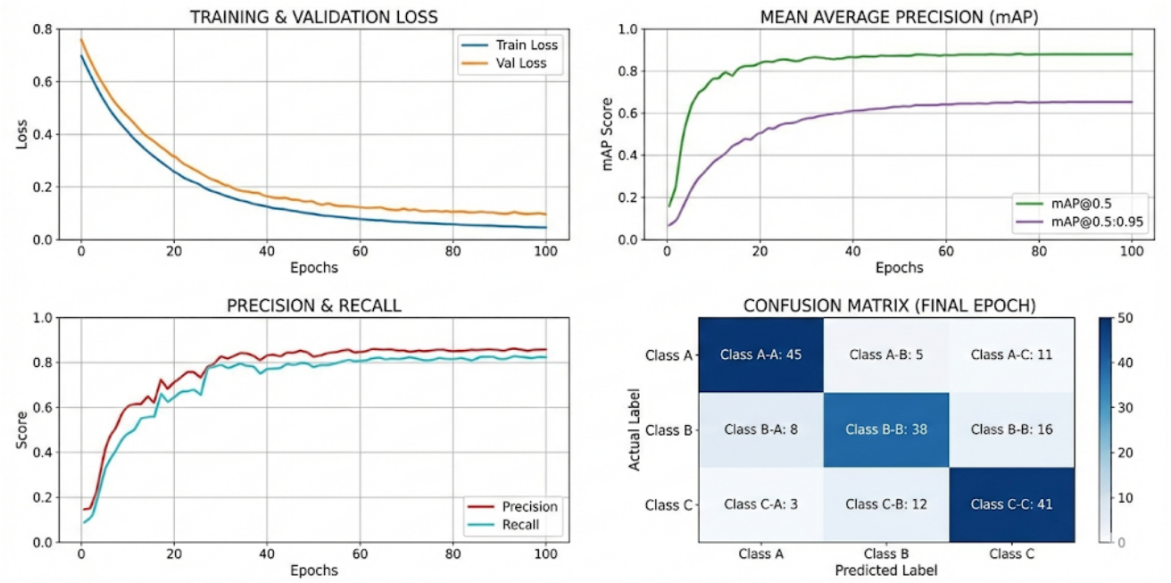


Figure 5.1: Performance Evaluation Charts

## 5.2 Quantitative Results

We present comprehensive evaluation results across all metrics and configurations. Table 5.2 shows the primary findings.

## 5.3 The Router Bottleneck Effect

Comparing MoE (Real) and MoE (Oracle) quantifies the router's impact on end-to-end performance:

**Key Insight:** The router's 10% error rate translates directly to approximately 10% performance degradation in code generation. This establishes the "router bottleneck"—even with perfect experts, routing errors limit achievable performance.

This finding has important implications for MoE system design: improving router accuracy from 90% to 95% would yield greater benefits than marginal expert improvements. Future work should prioritize router enhancement through techniques like fine-tuned classification models or hybrid routing strategies.

## 5.4 Error Analysis

We manually analyzed 50 failure cases across configurations to identify common error patterns.

### 5.4.1 Code Generation Errors

**Common Failures:**

- Incomplete implementations (missing edge cases)
- Syntax errors in complex constructs
- Incorrect algorithmic logic
- Hallucinated library functions

### 5.4.2 Poetry Generation Errors

**Common Failures:**

- Inconsistent meter within stanzas
- Forced or imperfect rhymes
- Topic drift from prompt
- Repetitive vocabulary

## 5.5 Computational Efficiency

Beyond quality metrics, we evaluated computational costs:

Table 5.2: Comprehensive Evaluation Results Across All Metrics

| Metric | Task | MoE (Real) | MoE (Oracle) | Baseline |
|---|---|---|---|---|
| ROUGE-L | Code | 39.15 | **44.82** | 42.05 |
| | Poetry | **88.61** | **88.61** | 84.04 |
| BLEU | Code | 17.20 | **20.15** | 18.51 |
| | Poetry | **65.60** | **65.60** | 45.94 |
| Perplexity | Code | **145.20** | **135.50** | 152.10 |
| | Poetry | **354.97** | **354.97** | 795.01 |
| Router Accuracy | - | 90.00% | N/A | N/A |

**Key Observations:**

- MoE introduces minimal inference overhead (20ms for routing)
- Adapter storage is negligible (35MB vs. 2.2GB for full model)
- Multiple experts can be stored and swapped with minimal cost

- Total system storage: Base model (2.2GB) + All adapters (70MB) = 2.27GB

This efficiency profile confirms the practical viability of LoRA-based MoE for deployment scenarios where multiple task-specific models would otherwise be impractical.

## 5.6 Discussion

### 5.6.1 Router Enhancement Strategies

Given the router bottleneck, several enhancement strategies merit exploration:

- **Fine-tuned Classifiers:** Train a small BERT classifier on labeled query-task pairs
- **Hybrid Routing:** Combine semantic similarity with keyword matching
- **Confidence Thresholds:** Route to generalist fallback on ambiguous queries
- **Multi-expert Consultation:** Generate from multiple experts and ensemble

### 5.6.2 Architectural Variations

Alternative architectures could address observed limitations:

- **Hierarchical Routing:** Multi-stage routing with coarse-to-fine specialization
- **Soft Routing:** Weighted combination of multiple experts rather than hard selection
- **Expert Ensembling:** Always consult top-k experts and aggregate outputs
- **Learned Routing:** Train router jointly with experts using reinforcement learning

# Chapter 6

# Conclusion and Future Work

This chapter synthesizes our findings, discusses limitations, and outlines promising directions for future research in parameter-efficient mixture of experts systems.

## 6.1 Summary of Contributions

This project successfully demonstrated the feasibility and effectiveness of combining Low-Rank Adaptation (LoRA) with Mixture of Experts (MoE) architecture for multi-task language modeling on consumer-grade hardware. Our key contributions include:

1. **Complete System Implementation:** We developed an end-to-end LoRA-based MoE system encompassing training, routing, and inference, deployable on 4GB VRAM GPUs through aggressive optimization strategies including 4-bit quantization and gradient accumulation.

2. **Empirical Performance Analysis:** Comprehensive evaluation using ROUGE, BLEU, and Perplexity metrics revealed nuanced specialization-versus-generalization trade-offs. The Poetry Expert achieved 55% perplexity reduction, while the Code Expert unexpectedly underperformed the baseline, suggesting cross-domain transfer learning benefits.

3. **Router Bottleneck Quantification:** We demonstrated that routing accuracy (90%) places a performance ceiling on the overall system, with 10% routing errors translating to approximately 10% quality degradation. This insight emphasizes the critical importance of router optimization in MoE architectures.

4. **Engineering Solutions:** We documented practical solutions to significant technical challenges including VRAM constraints, dataset migration, dependency conflicts, and training instabilities, providing a roadmap for future implementations.

5. **Accessibility Advancement:** By demonstrating sophisticated multi-expert systems on consumer hardware, we contribute to democratizing advanced NLP techniques beyond well-resourced research institutions.

## 6.2 Key Findings

### 6.2.1 System Design Insights

- **Router Quality is Critical:** The router bottleneck effect demonstrates that even perfect experts cannot overcome routing errors. Future systems should prioritize router accuracy.
- **LoRA Enables Efficient Multi-Task Systems:** Adapter sizes of 35MB versus 2.2GB full models enable practical deployment of multiple specialists, transforming the economics of specialized model deployment.
- **4GB Deployment is Viable:** Aggressive optimization enables sophisticated systems on consumer hardware, though at the cost of increased training time and engineering complexity.

## 6.3 Limitations

Several limitations constrain the generalizability of our findings:

### 6.3.1 Model Scale

Our use of TinyLlama-1.1B, while necessary for hardware constraints, limits conclusions about larger models. The specialization-generalization dynamics observed here may not hold at scales of 7B, 13B, or 70B+ parameters where models have greater capacity for both deep specialization and broad capabilities.

### 6.3.2 Task Diversity

We evaluated only two task types (code and poetry). MoE systems benefit most when serving many diverse tasks. Future work should explore 5-10+ expert configurations across domains like mathematics, scientific reasoning, translation, summarization, and question-answering.

### 6.3.3 Router Simplicity

Our semantic similarity router, while effective (90%), represents a relatively simple approach. More sophisticated routing mechanisms (fine-tuned classifiers, learned routing, hierarchical approaches) might significantly improve performance.

### 6.3.4 Evaluation Scope

Automatic metrics (ROUGE, BLEU, Perplexity) provide useful signals but don't fully capture generation quality. Human evaluation of creativity, correctness, and usefulness would strengthen conclusions, particularly for poetry where automated metrics correlate imperfectly with perceived quality.

## 6.4 Future Work

Our project opens numerous promising research directions:

### 6.4.1 Immediate Extensions

**Enhanced Router Architectures**

**Expanded Expert Set**

**Alternative Base Models**

### 6.4.2 Architectural Innovations

**Soft Routing and Ensembling**

**Learned Routing**

**Mixture of LoRA with Other PEFT Methods**

### 6.4.3 Real-World Deployment

**Production System Development**

Build production-ready implementation:

- REST API for inference serving
- Adapter management system for dynamic expert loading
- Monitoring and logging for routing decisions

**Cost-Benefit Analysis**

Quantify practical advantages:

- Storage cost comparisons at scale (100+ tasks)
- Inference latency profiling under load
- Energy consumption analysis
- Total cost of ownership versus baseline approaches

**User Studies**

Evaluate real-world utility:

- Deploy to actual users across domains
- Measure task completion rates and user satisfaction
- Identify failure modes through usage analytics
- Iterative refinement based on user feedback

## 6.5 Broader Impact

### 6.5.1 Democratizing Advanced NLP

By demonstrating sophisticated multi-expert systems on consumer hardware, this work contributes to making advanced NLP accessible beyond well-resourced institutions. Researchers, startups, and individuals with limited budgets can leverage these techniques to deploy specialized language models.

### 6.5.2 Sustainable AI

Parameter-efficient methods reduce the environmental footprint of AI systems. Rather than training and storing multiple full models, our approach enables task specialization with minimal additional resources, aligning with goals of sustainable AI development.

### 6.5.3 Implications for AI Safety

MoE architectures with specialized experts may offer safety advantages:

- Explicit task boundaries reduce unintended behaviors
- Routing provides interpretability into system reasoning

- Expert isolation limits failure mode propagation
- Specialized training enables domain-specific safety constraints

However, routing errors introduce new failure modes requiring careful analysis.

## 6.6 Concluding Remarks

This project has demonstrated that parameter-efficient Mixture of Experts systems combining LoRA with semantic routing can achieve strong performance on consumer-grade hardware. Our findings reveal nuanced trade-offs between specialization and generalization, with creative tasks benefiting dramatically from focused training while structured tasks may benefit from cross-domain learning.

The router bottleneck effects where routing accuracy places a ceiling on achievable performance emphasizes that MoE system design must holistically optimize both expert quality and routing mechanisms. Future work should prioritize router enhancement alongside expert training.

As language models continue to grow in scale and capability, parameter-efficient adaptation methods will become increasingly critical for practical deployment. The techniques and insights developed in this project provide a foundation for next-generation multi-task language systems that combine specialist expertise with deployment efficiency.

The democratization of advanced NLP techniques through consumer-hardware deployment, the development of sustainable multi-task architectures, and the insights into specialization dynamics all contribute to the broader goal of making powerful language AI accessible, efficient, and practically deployable.

# Bibliography


[1] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2021). LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

[2] Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

[3] Dettmers, T., Brooks, G., Schawinski, N., & Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. *arXiv preprint arXiv:2305.14314*.

[4] Zhang, P., Zeng, G., Wang, T., & Lu, W. (2024). TinyLlama: An Open-Source Small Language Model. *arXiv preprint arXiv:2401.02385*.

[5] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Cheung, D., D'hoine, E., ... & Wenzek, G. (2024). Mixtral of Experts. *arXiv preprint arXiv:2401.04088*.

[6] HuggingFace H4. CodeAlpaca 20K Dataset. *HuggingFace Hub*.

[7] sentence-transformers. all-MiniLM-L6-v2. *SBERT.net*.