CS4681 - Advanced Machine Learning

Progress Report

SLM004

Parameter-Efficient Neural Networks: Enhancing the Sparsely-Gated Mixture-of-Experts Layer in LSTMs

Ravichandran Abineyan

210017V

Table of Contents

1. Introduction

1.1 Background

Deep learning models have shown remarkable performance gains by scaling parameters, with architectures such as LSTMs and Transformers achieving state-of-the-art results across language modeling, translation, and multimodal tasks. However, this scaling comes with high computational and memory costs. The Mixture-of-Experts (MoE) framework offers a promising solution by activating only a subset of experts for each input, enabling massive parameter counts without proportional compute requirements. Early work by Shazeer et al. (2017) [13] demonstrated the effectiveness of MoE layers within LSTM-based language models, achieving unprecedented capacity at manageable computational budgets.

1.2 Problem Statement

Despite its success, the original LSTM + MoE framework suffers from parameter redundancy and inefficiency. Experts are large and largely independent, leading to high memory usage and training instability. In recent years, parameter-efficient MoE approaches (e.g., Switch routing, low-rank decomposition, and LoRA-based experts) have significantly improved efficiency in Transformer architectures. However, these advances have not been systematically applied or evaluated in the recurrent (LSTM) MoE setting, leaving an important research gap.

1.3 Objectives

This project aims to bridge that gap by exploring parameter efficiency techniques in LSTM + MoE architectures, with the following objectives:

1. Re-implement the baseline LSTM + MoE model as described by Shazeer et al. (2017).
2. Integrate modern parameter-efficient strategies (e.g., low-rank factorization, shared expert layers, Switch-style gating).
3. Compare the performance, efficiency, and stability trade-offs of these approaches.
4. Identify best practices for designing lightweight MoE layers in recurrent models.

2. Literature Review

2.1 Introduction: The Challenge of Scaling Efficient AI Models

The exponential growth of model sizes in natural language processing and other domains has introduced significant challenges in terms of training efficiency, memory usage, and deployment feasibility. Models such as BERT and GPT-4 demonstrate the remarkable potential of scaling, but their dense architectures activate all parameters for every input, leading to unsustainable computational costs. To address this, researchers have turned to sparse and modular neural architectures, with the Mixture of Experts (MoE) paradigm emerging as a leading framework [1]. MoE models activate only a fraction of parameters per input, thereby decoupling model capacity from computational cost. This enables scaling to billions or even trillions of parameters without proportionally increasing compute overhead.

2.2 Early MoE Foundations and Specialization

The concept of MoE was first formalized by Jacobs et al. [1], who proposed adaptive mixtures of local experts as a way to achieve modular specialization. Their work introduced the key idea of a gating mechanism, which determines which expert sub-models are activated for a given input. This framework allowed different experts to specialize in particular subspaces of data, fostering diversity and efficiency.

Eigen et al. [2] extended this idea into Deep Mixtures of Experts (DMoE), where multiple layers of experts and gates are stacked hierarchically. This architecture demonstrated that experts at lower layers tend to specialize in local attributes (such as spatial or positional features), while higher layers capture more abstract or semantic properties. Such hierarchical specialization highlighted the expressive power of modular sparse models and established a basis for modern MoE research.

2.3 Sparsely-Gated Mixture of Experts: A Breakthrough in Scaling

The seminal work by Shazeer et al. [3] introduced the Sparsely-Gated Mixture of Experts (MoE) layer, which represented a turning point in large-scale model design. This architecture employed a Noisy Top-k gating mechanism, which activates only a handful of experts (e.g., 4) out of thousands for each input, achieving sparsity levels exceeding 99.99%. This innovation allowed neural networks to scale to over 100 billion parameters while maintaining high computational efficiency.

A critical insight from this work was that experts naturally specialize in syntactic and semantic features without explicit supervision, leading to improved performance in both language modeling and machine translation. Shazeer et al. also addressed practical challenges such as expert imbalance (preventing some experts from dominating), the shrinking batch problem (caused by distributing tokens among many experts), and communication overhead in distributed training. Their empirical results showed perplexity reductions of 24–39% and BLEU score improvements across multiple translation tasks, firmly establishing sparse conditional computation as a foundation for extreme model scaling.

2.4 Routing, Training Stability, and Balanced Assignment

Despite the promise of sparse activation, MoE architectures face significant challenges in ensuring balanced expert utilization. If routing mechanisms assign too many inputs to a small subset of experts, both specialization and efficiency suffer. Shazeer et al. [3] mitigated this with auxiliary balancing losses, but these required delicate hyperparameter tuning.

Lewis et al. [6] proposed a more principled solution with BASE layers, which treat expert assignment as a linear assignment problem. This approach guarantees balanced token-to-expert allocation without requiring balancing hyperparameters. BASE layers simplify training, reduce communication overhead through local shuffling, and demonstrate improved training stability. Importantly, they showed that even a single BASE layer can yield strong performance, confirming the viability of algorithmically constrained routing as an alternative to loss-based balancing.

2.5 Scaling MoE: From Billion to Trillion Parameters

Scaling MoE architectures from hundreds of millions to trillions of parameters has been made possible by a combination of algorithmic and systems innovations. Fedus et al. [5] introduced the Switch Transformer, which simplifies routing by selecting only a single expert per input (k=1). This reduces router computation, halves expert capacity requirements, and simplifies communication patterns, enabling models with over a trillion parameters. Their results demonstrated a 7× speedup in pretraining compared to dense baselines, while maintaining or improving model quality.

On the systems side, Lepikhin et al. [4] developed GShard, a framework for automatic sharding and distributed training of MoE models across thousands of TPU cores. By introducing SPMD (Single Program Multiple Data) compilation, GShard was able to train a 600-billion parameter multilingual model efficiently across 2,048 TPU cores in only four days. This work highlighted how systems-level design is crucial for practical extreme-scale training.

More recently, Ludziejewski et al. [7] introduced scaling laws for MoE architectures, focusing on the concept of expert granularity. They demonstrated that optimal efficiency depends not only on the number of experts but also on their internal structure, showing that the efficiency gap between dense and sparse models grows larger as scale increases. Together, these works established MoE as a practical and scalable approach to trillion-parameter modeling.

2.6 Parameter Efficiency and Fine-Tuning in MoE

While scaling addresses training efficiency, parameter-efficient fine-tuning (PEFT) methods have emerged as a complementary line of research for adapting large pre-trained models. Methods such as LoRA (Low-Rank Adaptation) introduce small trainable matrices into pre-trained weights, enabling fine-tuning with only a fraction of parameters. Extensions like TT-LoRA (Tensor Train LoRA) further compress these adapters using tensor decomposition, reducing storage and inference costs [11].

MoE architectures have incorporated similar parameter-sharing techniques. Gao et al. [10] applied Matrix Product Operator (MPO) decompositions to experts, factorizing weight matrices into shared and auxiliary components. This significantly reduces redundant parameter growth in MoE while preserving expert specialization. Zadouri et al. [9] and Kunwar et al. [11] explored lightweight expert designs, combining LoRA-style adapters or low-dimensional scaling vectors with MoE routing. By decoupling expert training from routing, these approaches avoid instability while enabling scalable multi-task fine-tuning.

These PEFT-enhanced MoE models demonstrate strong performance while updating only a small fraction of parameters, making them attractive for real-world applications where computational resources are limited.

2.7 Synthesis and Research Gaps

Across decades of research, MoE architectures have evolved from early adaptive gating mechanisms [1] to sparsely-gated LSTM language models [4], to trillion-parameter Transformer systems [5], [6], and now to parameter-efficient hybrids [9]–[14]. Collectively, these works demonstrate that sparse activation through expert selection provides a compelling solution to the dual challenges of scaling capacity and reducing computation.

For this project, the focus lies at the intersection of these trajectories: bringing modern efficiency techniques (such as Switch-style gating [6], low-rank factorization [9], and LoRA-based expert fusion [11], [12]) back into the original LSTM + MoE setting [4], where efficiency has been comparatively underexplored. This synthesis highlights that while most efficiency gains have been realized in Transformer-based MoEs, the foundational LSTM framework still offers valuable ground for applying and validating these strategies.

Nevertheless, important gaps remain:

1. **Expert Utilization and Balancing**
   Shazeer et al. [4] introduced load-balancing losses to prevent experts from collapsing, but

imbalance remained a persistent issue. In parameter-efficient settings, where expert capacity is deliberately reduced (e.g., low-rank experts [9], shared projections [13]), this imbalance may waste scarce expert capacity, reducing the effectiveness of efficiency gains.

2. **Training Stability under Sparse Gating**
   The original MoE design [4] required noisy gating and auxiliary losses for stable convergence. When combined with parameter-efficient modifications such as Switch gating [6] or adapter-based experts [14], stability issues can worsen in recurrent models, where LSTMs already suffer from vanishing/exploding gradients.

3. **Parameter Redundancy vs. Efficiency**
   Shazeer et al. demonstrated scaling to billions of parameters, but efficiency was not a primary concern. Recent approaches [9]–[12] show that large portions of expert weights can be compressed or shared without major performance loss. However, the right balance between compression and expressiveness in LSTM-based MoEs remains unexplored.

4. **Generalization and Transferability**
   While [4] showed strong results on language modeling and translation, it is unclear how efficiently adapted MoEs (e.g., LoRA-experts [11], [12]) transfer to downstream NLP tasks in recurrent architectures. Parameter-efficient methods could either improve adaptability or restrict generalization if capacity is overly constrained.

5. **Extension Beyond Transformers**
   Most recent efficiency innovations were developed in Transformer-MoE systems [6], [15], [16]. The LSTM + MoE framework has not been revisited with modern efficiency tricks, leaving a clear research gap: can lightweight MoEs deliver similar gains in recurrent settings, or are these benefits unique to Transformer architectures?

3. Methodology

The methodology for this project is organized into four stages: Baseline Reproduction, Parameter-Efficient Extensions, Evaluation, and Analysis. Each stage is designed to progressively build towards a systematic investigation of parameter efficiency in LSTM + MoE architectures.

3.1 Baseline Reproduction: LSTM + MoE (Shazeer et al., 2017)

The first step is to faithfully reproduce the Sparsely-Gated MoE layer integrated with LSTM language models:

- **Model Components**
    - Experts: Each expert is a feed-forward network (two fully connected layers with ReLU).
    - Gating Network: A softmax-based router assigns each token to the top-k experts (as in Shazeer et al.). Noise injection is applied to encourage balanced usage.
    - Load Balancing Loss: An auxiliary term is added to reduce expert collapse (when only a few experts dominate).
    - LSTM Backbone: The MoE layer is inserted between LSTM layers for conditional computation.
- **Dataset**
    - A benchmark language modeling dataset (e.g., WikiText-103, One Billion Word Benchmark) will be used to replicate conditions similar to the original work.
- **Implementation**
    - The baseline is implemented in PyTorch/TensorFlow, ensuring modularity for later modifications.

3.2 Parameter-Efficient MoE Extensions

To address the inefficiency of the baseline, modern parameter efficiency strategies will be integrated and tested:

(a) Switch-Style Gating (Fedus et al., 2021)

- Replace noisy top-2 gating with top-1 routing, simplifying expert selection.

- Reduces communication overhead and stabilizes training.

(b) Low-Rank Factorization (Gao et al., 2022)

- Factorize expert weight matrices into low-rank components.
- Reduces per-expert parameter counts significantly (e.g., ×27.2 reduction with MPO).

(c) Shared and Lightweight Experts (Bai et al., 2022)

- Share parameters across multiple experts (e.g., partial weight tying).
- Investigate "tiny experts" where only a fraction of full parameters are unique.

(d) LoRA-Based MoE (Wu et al., 2024; Sun et al., 2025)

- Replace full expert layers with **low-rank adapters (LoRA modules)**.
- Multiple LoRA experts can be fused dynamically, offering modular efficiency.
- This design allows updating <1% of parameters during fine-tuning.

(e) Hybrid Designs

- Combine techniques (e.g., Switch gating + LoRA experts) to test trade-offs.
- Explore TT-LoRA MoE (Kunwar et al., 2025) for extreme efficiency with tensorized adapters.

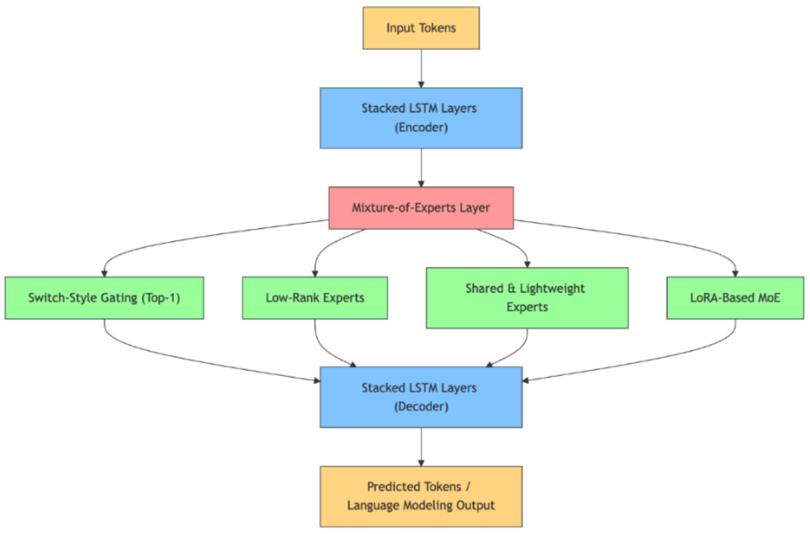
Input Tokens
Stacked LSTM Layers (Encoder)
Mixture-of-Experts Layer
Switch-Style Gating (Top-1)
Low-Rank Experts
Shared & Lightweight Experts
LoRA-Based MoE
Stacked LSTM Layers (Decoder)
Predicted Tokens / Language Modeling Output

Figure 3.1 High-level Architecture Diagram

3.3 Evaluation Strategy

Metrics

- **Performance**:
    - Perplexity (language modeling quality).
    - BLEU (for translation tasks, if dataset permits).
- **Efficiency**:
    - Number of trainable parameters.
    - Memory footprint during training and inference.
    - Training time per epoch and inference latency.
- **Stability**:
    - Training convergence curves.
    - Expert utilization distribution (to detect collapse).

Baselines for Comparison

- Original LSTM (no MoE) model.
- LSTM + MoE baseline (Shazeer et al., 2017).
- Each parameter-efficient extension (Switch, Low-rank, LoRA, etc.).

Experimental Setup

- Controlled training runs on identical hardware settings.
- Grid search for critical hyperparameters (learning rate, rank size, gating noise).
- At least three seeds per experiment to report stable averages.

3.4 Analysis and Deliverables

- **Comparative Analysis**

    - Quantitative tables comparing perplexity vs. parameter count.
    - Visualizations of expert utilization (heatmaps, distribution plots).
    - Trade-off curves (accuracy vs. efficiency).
- **Insights**
    - Which parameter-efficient technique works best in LSTM + MoE?
    - How much efficiency can be gained without harming performance?
    - Do trends in Transformer-based MoEs carry over to recurrent settings?
- **Deliverables**
    - Full implementation of baseline and extended models.
    - A comparative evaluation report on performance, training stability, and parameter efficiency.
    - A research paper synthesizing findings with broader MoE literature.

4. Feasibility Study

The feasibility of this project is supported by three factors:

1. **Technical Feasibility**
    - The baseline model (LSTM + MoE) has been publicly documented [4] and can be reimplemented using standard deep learning frameworks (PyTorch/TensorFlow).
    - Parameter-efficient strategies (e.g., Switch gating [6], low-rank factorization [9], and LoRA-based experts [11]) are modular and can be integrated into the MoE layer without requiring fundamental changes to the LSTM backbone.

2. **Resource Feasibility**
    - While full-scale trillion-parameter MoEs are computationally prohibitive, the project will evaluate smaller-scale LSTM + MoE models on accessible datasets (e.g., WikiText-103).
    - Efficiency methods reduce compute and memory requirements, making experiments viable on limited university hardware or cloud GPU resources.

3. **Research Feasibility**

    - The literature indicates clear gaps in applying efficiency techniques to recurrent MoEs.

This ensures that findings, even at modest scale, contribute meaningful insights into an underexplored area.

5. Project Timeline

Week 1 – Project Setup, Planning & Initial Literature Review

The first week establishes the foundation of the research. Alongside project setup, you will begin the literature review to frame your study in the context of prior work:

- Define research objectives: improving parameter efficiency in Mixture-of-Experts (MoE).
- Set up the environment (PyTorch/TensorFlow, Colab, experiment tracking).
- Begin literature review: cover Shazeer et al. (2017) MoE, Switch Transformers (Fedus et al., 2021), low-rank approximation techniques, and parameter-sharing methods.
- Document initial insights on where existing approaches succeed and where efficiency gaps remain.

Week 2 – Methodology Planning, Evaluation Setup & Extended Literature Review

This week extends the literature review and begins drafting the methodology and evaluation framework:

- Extend review to include parameter-efficient neural networks (structured sparsity, routing, compression, distillation).
- DraFeft methodology outline: baseline LSTM model, MoE integration plan, and efficiency-focused modifications.
- Define evaluation metrics: perplexity, BLEU (if MT), training time, parameter count, GPU memory usage.
- Establish progressive evaluation setup to compare baseline → MoE → efficiency-optimized models.

Week 3 – Baseline Model Implementation (LSTM)

This week transitions to implementation with a standard stacked LSTM language model as the baseline:

- Implement the baseline LSTM.
- Train on a small dataset to confirm feasibility.
- Record baseline metrics: perplexity, training time, parameter count.
- Document implementation choices and limitations for later comparison.

Week 4 – Integration of the MoE Layer

This week focuses on implementing the **Mixture-of-Experts layer**:

- Insert MoE between LSTM layers following Shazeer et al. (2017).
- Implement the gating mechanism and ensure proper routing and load balancing.
- Validate functionality on a small dataset subset.
- Compare performance with baseline LSTM.

Week 5 – Scaling Experiments with MoE

Here, the capacity and computational trade-offs of scaling MoE are examined:

- Experiment with different numbers of experts (e.g., 16, 32).
- Explore top-k gating with k=2 and k=4.
- Evaluate impact on perplexity, compute cost, and memory usage.
- Establish scaling baselines before introducing efficiency-focused modifications.

Week 6 – Method 1: Sparse Routing via Top-1 Gating

Introduce the first efficiency-oriented modification inspired by Switch Transformers:

- Replace top-k gating with top-1 routing (sparse expert activation).
- Compare against top-k routing in terms of efficiency (memory, training speed) and model performance.
- Analyze trade-offs in expressivity vs. parameter utilization.

Week 7 – Short Paper Draft & Method 2: Low-Rank Factorization of Experts

This week serves both an experimental and a documentation purpose:

- Implement low-rank factorization of expert weight matrices to reduce parameter counts.
- Train and evaluate the low-rank MoE variant.
- Draft a short research paper or workshop-style submission documenting progress:
    - Research motivation and literature review.
    - Methodology outline.
    - Preliminary results (baseline, MoE, sparse routing, low-rank experts).

Week 8 – Method 3: Parameter Sharing Across Experts

The third efficiency-focused modification will be explored:

- Implement shared input/output projections across experts, while preserving unique transformations internally.
- Evaluate effects on efficiency, redundancy reduction, and performance.
- Compare results against earlier variants.

Week 9 – Comprehensive Experimentation & Analysis

This week consolidates all experimental results:

- Train and evaluate baseline LSTM, vanilla MoE, and optimized MoE variants (sparse routing, low-rank factorization, parameter sharing).
- Collect metrics across perplexity, BLEU, parameter count, GPU memory, and training time.
- Perform comparative analysis of accuracy vs. efficiency trade-offs.
- Develop visualizations for clarity.

Week 10 – Final Paper Writing

The last week focuses on complete documentation and knowledge dissemination:

- Write the final paper including: introduction, literature review, methodology, results, analysis, discussion, and conclusion.
- Highlight contributions to parameter-efficient MoE design.

6. References

[1] C. Jacobs, M. Jordan, S. Nowlan, and G. Hinton, “Adaptive mixtures of local experts,” *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[2] D. Eigen, M. Ranzato, and I. Sutskever, “Learning factored representations in a deep mixture of experts,” *arXiv preprint arXiv:1312.4314*, 2014.

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

[4] M. Lepikhin et al., “GShard: Scaling giant models with conditional computation and automatic sharding,” in *Proc. ICLR*, 2021.

[5] W. Fedus, B. Zoph, and N. Shazeer, “Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity,” *arXiv preprint arXiv:2101.03961*, 2021.

[6] M. Lewis et al., “BASE Layers: Simplifying training of large, sparse models,” in *Proc. ICML*, 2021.

[7] B. Ludziejewski, C. Dietrich, and W. Samek, “Scaling laws for mixture-of-experts,” *arXiv preprint arXiv:2302.04676*, 2023.

[8] T. Mu and J. Lin, “A comprehensive survey of mixture-of-experts: Algorithms, theory and applications,” *arXiv preprint arXiv:2302.00676*, 2023.

[9] A. Zadouri, R. Strudel, H. Zhang, and M. Elhoseiny, “Ultra-lean mixture of experts,” *arXiv preprint arXiv:2310.13420*, 2023.

[10] M. Gao et al., “Parameter-efficient mixture-of-experts via matrix product operators,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[11] R. Kunwar et al., “Tensor-train low-rank adaptation for parameter-efficient fine-tuning,” *arXiv preprint arXiv:2302.09095*, 2023.

[12] Z. Meng, S. Sun, and A. P. Parikh, “Parameter-efficient multi-task fine-tuning of pre-trained transformers,” in *Proc. EMNLP*, 2021.

[13] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. Hinton, and J. Dean, “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer,” *arXiv preprint arXiv:1701.06538*, 2017.