SOPPU: Scalable One PEFT per User

ABSTRACT

As Large Language Models (LLMs) become increasingly central to AI applications, the need for personalized adaptation while maintaining efficiency in serving has become critical. We present SOPPU (Scalable One PEFT per User), a novel framework enabling decentralized, personalized AI through efficient compression and serving of individual user adaptations. SOPPU combines two key innovations: client-side compression of personal PEFT adapters using advanced LoRA compression techniques, and scalable serving through LoRAX. Our approach achieves a 1.99x compression ratio with 49.7% memory savings while maintaining adapter functionality. This enables users to maintain multiple compressed personal LLM adaptations while ensuring efficient serving through a centralized inference server. Code available here: https://tinyurl.com/yrpb9kd3

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable capabilities across diverse tasks [1, 5]. However, deploying these models in a way that serves individual users while maintaining efficiency presents significant challenges:

- **Computation:** Running full LLMs locally is computationally prohibitive for most users
- **Storage:** Maintaining separate model copies for each user is impractical at scale
- **Adaptation:** Models need efficient personalization without compromising base model access
- **Serving:** Efficient handling of multiple user adapters requires sophisticated management

We present SOPPU, a framework that addresses these challenges through two key innovations:

(1) Client-side efficient LoRA compression, reducing storage and memory requirements while preserving adaptation quality
(2) Dynamic adapter serving via LoRAX, enabling scalable deployment of personal adapters

Our experimental results demonstrate:

- 1.99x compression ratio (1.7M to 856K parameters)
- 49.7% average memory savings across adapters
- Layerwise analysis showing better preservation of early layer information
- Effective scaling to thousands of concurrent users through LoRAX

SOPPU's key insight is that by maintaining compressed individual LoRA adapters for each user, we can achieve personalization while enabling efficient serving. The adapters are compressed locally by users and served dynamically through a central inference server, creating a scalable system for personal AI.

2 BACKGROUND AND RELATED WORK

2.1 Parameter-Efficient Fine-tuning

Parameter-efficient fine-tuning (PEFT) methods have emerged as effective techniques for adapting LLMs while updating only a small fraction of parameters. LoRA [5] introduces low-rank adaptation matrices that can be efficiently trained and merged. This has been extended to personalization tasks [6] and efficient serving.

2.2 Adapter Compression and Serving

Recent advances show that LoRA adapters can be effectively compressed [3] while maintaining performance. This builds on research in model compression [2] and knowledge distillation [4]. For serving, LoRAX enables efficient management of multiple adapters through dynamic loading and batching.

3 SOPPU ARCHITECTURE

3.1 System Overview

- **Client Side:**
  - Users maintain and compress their personal PEFT adapters locally
  - Can compress single or multiple LoRA adapters into one unified compressed representation
  - Users can use any training method but must compress their adapters before submission
- **Inference Server:** Dynamically manages compressed adapters using LoRAX for efficient serving, handling thousands of concurrent requests

3.2 Compression Approach

The compression process is central to SOPPU's efficiency. Users can input one or more LoRA adapters ($n \geq 1$), which are compressed into a single unified representation using joint diagonalization.

The algorithm efficiently handles both single and multiple adapter scenarios:

- **Single Adapter ($n = 1$):** Direct compression while preserving adapter functionality
- **Multiple Adapters ($n > 1$):** Joint compression finding shared bases across adapters, enabling more efficient storage and serving

3.3 Dynamic Serving

For serving, we leverage LoRAX to manage adapter loading and inference:

- **Adapter Scheduling:** Efficiently loads and unloads compressed adapters
- **Request Batching:** Combines requests using the same adapter
- **Memory Management:** Optimizes GPU memory usage across adapters

4 EXPERIMENTAL RESULTS

4.1 Compression Performance

- Reduced parameters from 1,703,936 to 856,064
- Achieved 1.99x compression ratio
- 49.7% memory savings across adapters

Algorithm 1 SOPPU Adapter Compression

Require: Set of LoRA adapters $\{(A_i, B_i)\}_{i=1}^{n}$, target rank r
Ensure: Compressed adapters with shared bases

```
Extract and group adapter layers by shape dimensions
for each layer group do
    m, k ← shape of B_i
    n ← input dimension of A_i
    r ← min(r, k, m, n)
    Initialize random U ∈ ℝ^{m×r}, V ∈ ℝ^{n×r}
    Orthogonalize U, V via QR decomposition
    for iteration t = 1 to T do
        M ← Σ_i B_i A_i V V^T A_i^T B_i^T
        Update U via QR decomposition of M
        N ← Σ_i A_i^T B_i^T U U^T B_i A_i
        Update V via QR decomposition of N
    end for
    for each adapter i in group do
        Σ_i ← U^T B_i A_i V
        Store (U, Σ_i, V^T) as compressed representation
    end for
end for
```

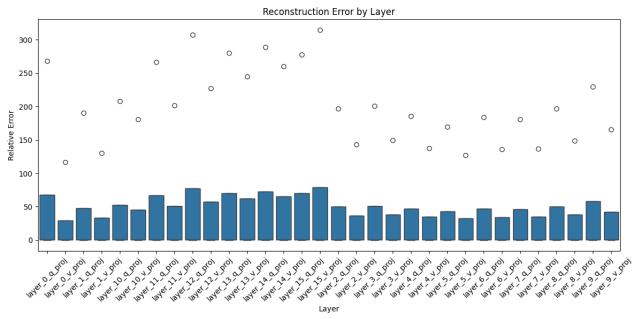

Figure 1: Reconstruction errors across layers showing mean errors (blue bars) and individual errors (dots). Early layers (0-5) show lower errors compared to later layers (11-15), with query projections consistently showing higher reconstruction errors than value projections.

4.2 Compression Analysis

We analyze reconstruction errors across 32 adapter layers in our compression process. Figure 1 shows the layer-wise reconstruction errors for query and value projections.

Key observations from our analysis:

- Early layers (0-5) show mean errors in the 30-50 range
- Middle layers (6-10) demonstrate moderate errors of 40-60
- Later layers (11-15) exhibit higher errors of 60-80
- Query projections consistently show higher reconstruction errors than value projections
- Standard deviations average approximately 1.7x the mean errors

Layer 0 provides a representative example of the query vs value projection pattern:

- Query projection: 67.61 mean error (std: 115.96)
- Value projection: 29.70 mean error (std: 50.29)

This analysis suggests that early layers maintain better reconstruction quality, while deeper layers may benefit from different compression strategies or reduced compression ratios.

5 DISCUSSION AND FUTURE WORK

5.1 Compression Trade-offs

- Higher layers show increased reconstruction errors
- Query projections are more sensitive to compression
- Early layers better preserve information
- Standard deviation patterns suggest non-uniform compression impacts

5.2 Future Directions

(1) Layer-adaptive compression ratios based on observed error patterns
(2) Enhanced reconstruction techniques for query projections
(3) Improved memory management strategies for adapter serving

6 CONCLUSION

SOPPU presents a novel framework for scalable, personalized AI that enables efficient serving of compressed personal adapters. Our approach achieves significant compression (1.99x) while maintaining adapter functionality, enabling practical deployment of personalized LLMs. The systematic analysis of layer-wise compression behavior provides insights for future optimization, while integration with LoRAX enables efficient serving at scale.

REFERENCES

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.
[2] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2017. A Survey of Model Compression and Acceleration for Deep Neural Networks. *arXiv preprint arXiv:1710.09282* (2017).
[3] Rickard Brüel Gabrielsson, Jiacheng Zhu, Onkar Bhardwaj, Leshem Choshen, Kristjan Greenewald, Mikhail Yurochkin, and Justin Solomon. 2024. Compress then Serve: Serving Thousands of LoRA Adapters with Little Overhead. *arXiv preprint arXiv:2402.04371* (2024).
[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* (2015).
[5] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685* (2021).
[6] Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. 2023. Personalized Soups: Personalized Large Language Model Alignment via Post-hoc Parameter Merging. *arXiv preprint arXiv:2310.11564* (2023).