LoRA-Mini: Adaptation Matrices Decomposition and Selective Training

Ayush Singh†, Rajdeep Aher†, Shivank Garg

Vision and Language Group, Indian Institute of Technology Roorkee, Roorkee, Uttarakhand, India - 247667
ayush_s@mt.iitr.ac.in , aher_rp@ma.iitr.ac.in , shivank_g@mfs.iitr.ac.in

Abstract

The rapid advancements in large language models (LLMs) have revolutionized natural language processing, creating an increased need for efficient, task-specific fine-tuning methods. Traditional fine-tuning of LLMs involves updating a large number of parameters, which is computationally expensive and memory-intensive. Low-Rank Adaptation (LoRA) has emerged as a promising solution, enabling parameter-efficient fine-tuning by reducing the number of trainable parameters. However, while LoRA reduces the number of trainable parameters, LoRA modules still create significant storage challenges. We propose LoRA-Mini, an optimized adaptation of LoRA that improves parameter efficiency by splitting low-rank matrices into four parts, with only the two inner matrices being trainable. This approach achieves upto a 20x reduction compared to standard LoRA in the number of trainable parameters while preserving performance levels comparable to standard LoRA, addressing both computational and storage efficiency in LLM fine-tuning.

Introduction

The rapid growth of large language models (LLMs) such as GPT-3 (Brown et al. 2020), GPT-4 (OpenAI et al. 2024), Llama (Dubey, Jauhri, and Pandey 2024), and Mistral (Jiang et al. 2023) has led to various advancements in the field of natural language processing, enabling LLMs to achieve high performance across various benchmarks such as IFEval (Zhou et al. 2023), SQuAD (Rajpurkar et al. 2016) etc. However, the computational and memory costs involved in the training of these models from scratch are substantial, posing significant challenges. Fine-tuning (Zhang et al. 2024) is a popular method to adapt pre-trained models for specific downstream tasks, but requires extensive resources when tuning all the model parameters, making it impractical for many applications. To address these limitations, researchers developed Parameter-Efficient Fine-Tuning (PEFT) (Han et al. 2024) techniques, which limit the number of adjustable parameters during fine-tuning, achieving accuracy comparable to full finetuning while significantly reducing both computational and memory costs. Among PEFT approaches, Low-Rank Adaptation(LoRA) (Hu et al. 2021) has emerged as a widely adopted technique. The technique adapts large models by adding trainable low-rank matrices to the model's layers. It builds on the observation that fine-tuning updates often have low intrinsic dimensionality. By incorporating this, LoRA achieves high efficiency without a major compromise in model performance, allowing for more scalable adaptation of LLMs to diverse tasks. These advances highlight LoRA's role in balancing efficiency and effectiveness, enabling widespread application of large-scale models in resource-constrained environments.

Our approach builds on existing Parameter-Efficient Fine-Tuning (PEFT) methods by targeting an even lower parameter count, aiming to maintain model performance while further minimizing memory usage.

Although deep networks may have a vast number of parameters, but only a small number of parameters significantly impact the learning process. Hence, we refine the intrinsic rank within the current LoRA framework by decomposing the standard matrices A and B into two sub-matrices each, with only one matrix from each pair being trainable.

Our experiments cover a range of models, including BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019), and T5 (Raffel et al. 2023), with a primary focus on performance metrics from the GLUE (Wang et al. 2019) benchmark and English-Romanian translation task from WMT16 (Bojar et al. 2016) dataset. Our results show that LoRA-Mini attains accuracy comparable to full fine-tuning approaches while significantly reducing memory requirements. Our key contributions include:

1. Evaluating the effectiveness of selectively freezing parameters within LoRA layers while maintaining model quality.
2. Minimizing the number of trainable parameters relative to earlier PEFT methods, while achieving performance on par with full fine-tuning and LoRA.
3. Evaluating the scalability of our approach on diverse set of tasks and models.

Related Work

Parameter-efficient fine-tuning has emerged as an essential approach for adapting large language models (LLMs) without incurring high computational and memory costs. Techniques such as Adapter-based methods (Houlsby et al.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

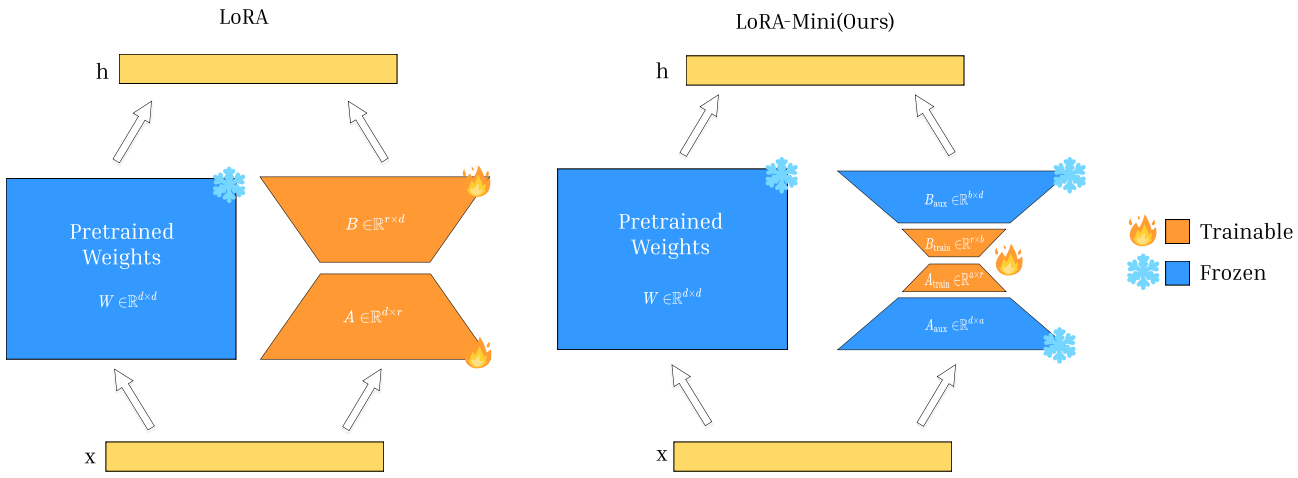

Figure 1: A visual comparison of the LoRA and LoRA-Mini techniques.

2019), Prefix-tuning (Li and Liang 2021), and Prompt-tuning (Lester, K., and T. 2021) provide alternatives to conventional fine-tuning, allowing for task-specific adaptation by inserting trainable parameters at selective layers. Adapter-based methods introduce small bottleneck layers to facilitate training while keeping core model parameters fixed. However, most adapter layers can add inference latency, as they must be processed sequentially due to their additional depth within each Transformer block. Variants of adapters attempt to mitigate this, such as by using a single adapter layer per block with an additional LayerNorm (Yang, Z., and M. 2020), or by reducing latency through layer pruning and multi-task adaptation (Rücklé, H., and Z. 2020; Pfeiffer et al. 2021).

On the other hand, prompt-based methods like Prefix-tuning face optimization challenges. Optimizing prompt-based adaptations can be difficult, with performance sensitivity to the number of trainable parameters and prone to non-monotonic behavior (Li and Liang 2021). Additionally, reserving part of the sequence length for adaptation may reduce the effective input length available for the downstream task, potentially limiting performance.

Consequently, LoRA (Hu, K., and P. 2021) provides an efficient approach to fine-tuning large language models by focusing on low-rank updates to the model weights rather than modifying the model parameters directly or adjusting input sequences. Unlike adapter-based methods, which add additional layers that can introduce latency in inference, or prompt-based methods, which often struggle with prompt optimization, it preserves the model's original architecture and sequence length, ensuring that the models' structure remains unchanged during deployment. Nevertheless, it faces a significant drawback: the large memory overhead associated with storing low-rank matrices, which presents challenges for deployment in resource-constrained environments.

Recent works have tried improving the original LoRA framework for efficient language model adaptation. Some significant works include LoRA-FA (Zhang et al. 2023) which achieves 1.4x memory reduction by freezing the projection-down matrix and training only the projection-up matrix. Bayesian-LoRA (Yang et al. 2024) further enhances efficiency by dynamically allocating ranks to layers based on data requirements.

Our technique, however, introduces additional parameter reduction from previous methods by decomposing matrices A and B into auxiliary and trainable components $(A_{aux}, A_{train}, B_{train}, B_{aux})$, with only the middle matrices being trainable. This substantially reduces the parameter count.

Methodology

LoRA-Mini focuses on the introduction of selective training within decomposed matrices. We divide each LoRA matrix into a trainable and frozen part, allowing us to limit the update space and execute controlled updates, where the frozen outer matrices guide the training process. The mathematical formulation of our approach can be expressed as follows:
Let $AB \in \mathbb{R}^{d \times k}$ represent our target weight matrix where d is input and k is output dimension. We decompose the LoRA matrices into:

- Outer auxiliary matrices: $A_{\text{aux}} \in \mathbb{R}^{d \times \text{a}}$, $B_{\text{aux}} \in \mathbb{R}^{\text{b} \times k}$ (frozen)
- Inner trainable matrices: $A_{\text{train}} \in \mathbb{R}^{a \times r}$, $B_{\text{train}} \in \mathbb{R}^{r \times b}$ (trainable)

Let x be the input to a layer. Then the output h from the layer after applying LoRA-Mini becomes :

$$h = (W + A_{\text{aux}} \cdot A_{\text{train}} \cdot B_{\text{train}} \cdot B_{\text{aux}}) \cdot x \qquad (1)$$

In LoRA, given a pre-trained weight matrix $W \in \mathbb{R}^{d \times k}$, the weight update ΔW is through two matrices, $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times k}$, where $r \ll \min(d, k)$. This parameterization leads to $\Delta W = AB$, resulting in $r \times (d + k)$ trainable parameters, reducing the training overhead.

Our method extends this reduction further by limiting the trainable parameters to $r \times (a + b)$, where a and b are predefined dimensional constraints, as detailed in later sections.

Method	Parameters	Rank	STSB	COLA	MRPC	RTE
FFT	125M	-	**90.77**	80.54	**89.46**	*74.01*
LoRA	0.90M	8	88.49	80.53	87.50	62.81
LoRA	1.80M	16	89.11	80.24	87.50	67.14
LoRA	3.50M	32	89.87	82.35	85.78	68.95
Ours(D)	0.04M	8	89.62	81.00	86.52	69.31
Ours(D)	0.07M	16	89.51	81.50	*88.23*	72.92
Ours(D)	0.15M	32	89.55	82.17	86.76	74.72
Ours(D+A)	0.08M	8	89.62	82.35	87.01	70.75
Ours(D+A)	0.15M	16	89.67	*82.84*	86.02	69.31
Ours(D+A)	0.30M	32	*90.36*	**83.00**	86.52	68.23

Table 1: In this experiment, we use **RoBERTa**$_{\text{base}}$. We report Pearson correlation for the STSB task and accuracy for the remaining tasks. For all metrics, higher values indicate better performance. Green(or bold) entries denote the best and Blue(or italics) denote the second best entries of a column

Method	Parameters	Rank	STSB	COLA	MRPC	RTE
FFT	110M	-	**88.74**	*81.69*	82.11	64.98
LoRA	0.90M	8	86.44	*81.69*	76.47	63.90
LoRA	1.80M	16	87.00	79.96	81.62	61.73
LoRA	3.50M	32	86.96	79.29	80.15	**68.95**
Ours(D)	0.04M	8	86.21	**81.78**	82.33	*67.87*
Ours(D)	0.07M	16	*88.35*	80.82	*83.57*	64.26
Ours(D)	0.15M	32	87.13	81.40	82.59	61.01
Ours(D+A)	0.08M	8	88.02	79.29	78.92	67.14
Ours(D+A)	0.15M	16	88.25	79.58	81.37	64.26
Ours(D+A)	0.30M	32	87.98	80.82	**84.31**	65.70

Table 2: In this experiment, we use **BERT**$_{\text{base}}$: We report Pearson correlation for the STSB task and accuracy for the remaining tasks. For all metrics, higher values indicate better performance. Green(or bold) entries denote the best and Blue(or italics) denote the second best entries of a column

All matrices are initialized using the Kaiming method, with a Kaiming coefficient of $\sqrt{5}$, ensuring weight variance is scaled appropriately for stable gradient propagation. Our approach yields several advantages. First, by constraining updates to the inner matrices, while keeping outer matrices fixed, we achieve substantial memory savings without sacrificing the models efficiency. Second, the weight update, ($\Delta W = A_{aux}A_{train}B_{train}B_{aux}$) operates within a controlled subspace(within inner matrices), enabling efficient parameter optimization while maintaining model stability.

Experiments and Results

To do a robust evaluation of our approach, we broadly divided our experiment into two categories.

First, we evaluate our approach on the NLU and NLI tasks using the GLUE Benchmark (Wang et al. 2019) on RoBERTa (Liu et al. 2019) and BERT (Devlin et al. 2019). We particularly chose four tasks from the GLUE Benchmark: STSB (Cer et al. 2017), CoLA (Warstadt, Singh, and Bowman 2019), MRPC (Dolan and Brockett 2005), and RTE (Giampiccolo et al. 2007). The experiments were conducted using the following configurations :

- Full fine-tuning of the entire model
- Standard LoRA applied to dense and attention layers (rank values $r = 4, 8, 16$)
- LoRA-Mini applied only to dense layers
- LoRA-Mini applied to both dense and attention layers

We tested rank values of 8, 16, and 32, with a and b as 8, 16, 32, and 64. This enabled an effective comparison of LoRA-Mini's impact on feed-forward versus attention mechanisms. We did not experiment with applying our approach exclusively to attention layers, as previous research (Geva et al. 2021) has demonstrated that this would not yield significant improvements in overall accuracy. From our experiments, we observed that the configuration with a and b as 64 performs consistently well. Thus, we reported results with varying ranks and keeping a and b constant in the main paper. From Table 1 and Table 2, we can see that our approach performs consistently at par or greater than full-fine-tuning and LoRA. A comparison of rank with accuracy when LoRA-Mini is applied to both dense and attention layers of RoBERTa is shown in Figure 2.

We also evaluate the performance of our approach on generative tasks, with a particular focus on language translation. It is a significant benchmark for assessing model performance in real-world applications, as it involves both under-

standing and generating human language. To better understand how our approach scales with increasing model size, we select two models with different capacities: T5-Small and T5-Base. We fine-tuned these models on the English-Romanian subset of the WMT16 dataset, a widely used resource for machine translation tasks.

We then evaluated the generated translations using the BLEU (Papineni et al. 2002) and ROUGE-L (Lin 2004) metrics. The experimental setup closely follows the previous set, but in this case, we applied LoRA-Mini to both the attention and dense layers of the models. The detailed results from these evaluations are presented in Table 3 and Table 4. These show that LoRA-Mini consistently matches the performance of both base LoRA and full fine-tuning on both BLEU and ROUGE-L metrics.

These experiments clearly indicate that LoRA-Mini is a reliable approach for both language generation and understanding tasks. It not only delivers competitive results in these tasks but also reduces the parameter count considerably compared to previous methods, which can lead to improved efficiency and scalability. Results of all configurations and additional analysis are provided in the Appendix.

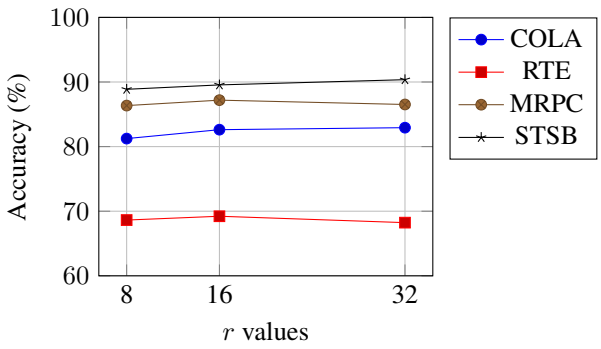

Figure 2: Accuracy comparison across different values of r for four tasks using LoRA-Mini applied to the Attention and Dense layers of RoBERTa-Base. The graph reports average of all combinations of a and b in that rank.

Method	Params	Rank	BLEU	ROUGEL
FFT	60M	-	26.11	45.95
LoRA	0.14M	1	26.17	45.86
LoRA	0.29M	2	26.1	46.04
LoRA	0.59M	4	26.1	46.05
LoRA	1.2M	8	26.11	45.87
LoRA	2.3M	16	26.08	45.91
Ours	0.08M	8	26.01	45.62
Ours	0.16M	16	26.02	45.88
Ours	0.32M	32	25.82	45.54

Table 3: In this experiment, we use **T5 Small**. We report BLEU, and ROUGEL. For all metrics, higher values indicate better performance.

Method	Params	Rank	BLEU	ROUGEL
FFT	223M	-	27.27	47.20
LoRA	0.38M	1	26.89	47.08
LoRA	0.76M	2	27.12	47.23
LoRA	1.50M	4	27.01	47.21
LoRA	3.0M	8	27.02	47.14
LoRA	6.1M	16	26.90	47.08
Ours	0.16M	8	26.83	46.98
Ours	0.32M	16	26.99	46.77
Ours	0.64M	32	26.92	46.78

Table 4: In this experiment, we use **T5 Base**. We report BLEU and ROUGEL. For all metrics, higher values indicate better performance.

Conclusion

We present LoRA-Mini as a parameter-efficient fine-tuning approach, demonstrating significant improvements in memory efficiency and task performance. Our results indicate that LoRA-Mini can achieve comparable or superior results compared to LoRA and full fine-tuning while substantially reducing the number of trainable parameters across a wide range of tasks, making it an ideal solution for finetuning LLMs in resource-constrained environments.

Limitations and Future Works

Due to computational constraints, we were unable to evaluate our LoRA approach on larger tasks of GLUE benchmark as well as on larger models such as LLaMA3.1-8B (Dubey, Jauhri, and Pandey 2024), Mistral-7B (Jiang et al. 2023) etc, or on more extensive benchmarks datasets like MMLU (Hendrycks et al. 2021), Math10k (Hu et al. 2023), COMMONSENSE170K (Hu et al. 2023) etc. Expanding to these larger models and benchmarks could provide a more comprehensive understanding of our method's scalability and performance in diverse domains and model sizes.

Future research could explore alternative matrix decomposition methods, such as partitioning LoRA matrices into more smaller components and experimenting with various combinations of trainable and frozen sections. Another approach might involve freezing randomly selected matrices (Zhu et al. 2024). Techniques like QR decomposition or singular value decomposition (SVD) could be used to initialize these matrices, potentially enhancing performance. Furthermore, this approach could be tested to approximate feedforward networks (FFNs) (Zeng and Lee 2024). Investigating a latent representation between LoRA matrices might also increase parameter efficiency.

References

Bojar, O. r.; Chatterjee, R.; Federmann, C.; Graham, Y.; Haddow, B.; Huck, M.; Jimeno Yepes, A.; Koehn, P.; Logacheva, V.; Monz, C.; Negri, M.; Neveol, A.; Neves, M.; Popel, M.; Post, M.; Rubino, R.; Scarton, C.; Specia, L.; Turchi, M.; Verspoor, K.; and Zampieri, M. 2016. Findings

of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation*, 131–198. Berlin, Germany: Association for Computational Linguistics.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165.

Cer, D.; Diab, M.; Agirre, E.; Lopez-Gazpio, I.; and Specia, L. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In Bethard, S.; Carpuat, M.; Apidianaki, M.; Mohammad, S. M.; Cer, D.; and Jurgens, D., eds., *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 1–14. Vancouver, Canada: Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Dolan, W. B.; and Brockett, C. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Dubey, A.; Jauhri, A.; and Pandey, A. 2024. The Llama 3 Herd of Models. arXiv:2407.21783.

Geva, M.; Schuster, R.; Berant, J.; and Levy, O. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. arXiv:2012.14913.

Giampiccolo, D.; Magnini, B.; Dagan, I.; and Dolan, B. 2007. The Third PASCAL Recognizing Textual Entailment Challenge. In Sekine, S.; Inui, K.; Dagan, I.; Dolan, B.; Giampiccolo, D.; and Magnini, B., eds., *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 1–9. Prague: Association for Computational Linguistics.

Han, Z.; Gao, C.; Liu, J.; Zhang, J.; and Zhang, S. Q. 2024. Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey. arXiv:2403.14608.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Houlsby, N.; Gane, A.; Al-Mamun, R. P. S.; S., E. D. S. D.; S., J. O. E.; B., D. M.; and C., T. D. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 1720–1731. PMLR.

Hu, E.; K., Y.; and P., J. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *Proceedings of the 2021 International Conference on Learning Representations (ICLR)*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685.

Hu, Z.; Lan, Y.; Wang, L.; Xu, W.; Lim, E.-P.; Lee, R. K.-W.; Bing, L.; and Poria, S. 2023. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. *arXiv preprint arXiv:2304.01933*.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. arXiv:2310.06825.

Lester, B.; K., R. B.; and T., J. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 190–200. ACL.

Li, H.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 5457–5467. PMLR.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692.

OpenAI; Achiam, J.; Adler, S.; Agarwal, S.; and Ahmad, L. 2024. GPT-4 Technical Report. arXiv:2303.08774.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In Isabelle, P.; Charniak, E.; and Lin, D., eds., *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics.

Pfeiffer, J.; K., T.; A., R.; and B., J. 2021. Multi-Task Prompt Tuning for Natural Language Processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 7450–7461. ACL.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2023. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In Su, J.; Duh, K.; and Carreras, X., eds., *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2383–2392. Austin, Texas: Association for Computational Linguistics.

Rücklé, M.; H., T.; and Z., V. 2020. Parameter-Efficient Multi-Task Learning for NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2155–2163. ACL.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. GLUE: A Multi-Task Benchmark

and Analysis Platform for Natural Language Understanding. arXiv:1804.07461.

Warstadt, A.; Singh, A.; and Bowman, S. R. 2019. Neural Network Acceptability Judgments. arXiv:1805.12471.

Yang, A. X.; Robeyns, M.; Wang, X.; and Aitchison, L. 2024. Bayesian Low-rank Adaptation for Large Language Models. arXiv:2308.13111.

Yang, Z.; Z., Z.; and M., K. 2020. LayerNorm is All You Need. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*.

Zeng, Y.; and Lee, K. 2024. The Expressive Power of Low-Rank Adaptation. arXiv:2310.17513.

Zhang, L.; Zhang, L.; Shi, S.; Chu, X.; and Li, B. 2023. LoRA-FA: Memory-efficient Low-rank Adaptation for Large Language Models Fine-tuning. arXiv:2308.03303.

Zhang, S.; Dong, L.; Li, X.; Zhang, S.; Sun, X.; Wang, S.; Li, J.; Hu, R.; Zhang, T.; Wu, F.; and Wang, G. 2024. Instruction Tuning for Large Language Models: A Survey. arXiv:2308.10792.

Zhou, J.; Lu, T.; Mishra, S.; Brahma, S.; Basu, S.; Luan, Y.; Zhou, D.; and Hou, L. 2023. Instruction-Following Evaluation for Large Language Models. arXiv:2311.07911.

Zhu, J.; Greenewald, K.; Nadjahi, K.; de Ocáriz Borde, H. S.; Gabrielsson, R. B.; Choshen, L.; Ghassemi, M.; Yurochkin, M.; and Solomon, J. 2024. Asymmetry in Low-Rank Adapters of Foundation Models. arXiv:2402.16842.

Experimental Setup Details

Models :

- **BERT** [1]- A bidirectional transformer model with 110 million parameters, pre-trained on masked language modeling and designed for natural language understanding tasks.
- **RoBERTa Base** [2]- An optimized version of BERT with 125 million parameters, featuring more extensive pre-training, improving performance on a wide range of language tasks.
- **T5-Small** [3]- A smaller version of the Text-To-Text Transfer Transformer (T5) with 60 million parameters, optimized for efficiency on generative tasks.
- **T5-Base** [4]- The base variant of T5 with 223 million parameters.

Benchmarks and Datasets :

- **GLUE Dataset** [5]: GLUE, or General Language Understanding Evaluation, is a benchmark designed to assess language understanding models across diverse NLP tasks, including CoLA (Corpus of Linguistic Acceptability), SST-2 (Stanford Sentiment Treebank), MRPC (Microsoft Research Paraphrase Corpus), STS-B (Semantic Textual Similarity Benchmark), QQP (Quora Question Pairs), MNLI (Multi-Genre Natural Language Inference), QNLI (Question Natural Language Inference), RTE (Recognizing Textual Entailment), and WNLI (Winograd NLI). Of these, we only train on 4 tasks, namely STSB, COLA, MRPC, and RTE.
- **WMT Dataset** [6]: The WMT-16 (Workshop on Machine Translation 2016) benchmark is a standardized evaluation framework for comparing translation models across multiple language pairs using reliable metrics like BLEU. In this research, we use the Romanian-to-English subset of WMT-16 to evaluate our model's translation accuracy and generalization across linguistic structures.

[1] huggingface.co/google-bert/bert-base-uncased
[2] huggingface.co/FacebookAI/roberta-base
[3] huggingface.co/google-t5/t5-small
[4] huggingface.co/google-t5/t5-base
[5] huggingface.co/datasets/nyu-mll/glue
[6] huggingface.co/datasets/wmt/wmt16

r	a	b	Accuracy(D)	Parameters(D)	Percentage(D)	Accuracy(D+A)	Parameters(A+D)	Percentage(A+D)
8	16	16	77.661%	11010	0.009%	80.058%	20226	0.016%
8	16	32	79.866%	15746	0.013%	81.016%	29570	0.024%
8	16	64	81.687%	25218	0.020%	81.592%	48258	0.039%
8	32	16	80.058%	15746	0.013%	80.345%	29570	0.024%
8	32	32	80.729%	20482	0.016%	80.345%	38914	0.031%
8	32	64	80.633%	29954	0.024%	82.934%	57602	0.046%
8	64	16	78.428%	25218	0.020%	79.866%	48258	0.039%
8	64	32	81.016%	29954	0.024%	82.646%	57602	0.046%
8	64	64	81.016%	39426	0.032%	82.359%	76290	0.061%
16	32	32	81.496%	39426	0.032%	82.359%	76290	0.061%
16	32	64	82.071%	58370	0.047%	83.317%	113666	0.091%
16	64	32	80.825%	58370	0.047%	81.975%	113666	0.091%
16	64	64	81.496%	77314	0.062%	82.838%	151042	0.121%
32	64	64	82.167%	153090	0.122%	82.934%	300546	0.240%

Table 5: RoBERTa-CoLA

r	a	b	Accuracy(D)	Parameters(D)	Percentage(D)	Accuracy(A+D)	Parameters(A+D)	Percentage(A+D)
8	16	16	67.148%	11010	0.009%	71.480%	20226	0.016%
8	16	32	67.870%	15746	0.013%	70.036%	29570	0.024%
8	16	64	70.036%	25218	0.020%	70.036%	48258	0.039%
8	32	16	64.982%	15746	0.013%	68.592%	29570	0.024%
8	32	32	67.148%	20482	0.016%	68.592%	38914	0.031%
8	32	64	75.090%	29954	0.024%	64.982%	57602	0.046%
8	64	16	65.704%	25218	0.020%	63.899%	48258	0.039%
8	64	32	67.870%	29954	0.024%	69.314%	57602	0.046%
8	64	64	69.314%	39426	0.032%	70.758%	76290	0.061%
16	32	32	68.953%	39426	0.032%	68.592%	76290	0.061%
16	32	64	71.119%	58370	0.047%	71.480%	113666	0.091%
16	64	32	71.119%	58370	0.047%	67.509%	113666	0.091%
16	64	64	72.924%	77314	0.062%	69.314%	151042	0.121%
32	64	64	74.729%	153090	0.122%	68.231%	300546	0.240%

Table 6: RoBERTa-RTE

r	a	b	Accuracy(D)	Parameters(D)	Percentage(D)	Accuracy(D+A)	Parameters(A+D)	Percentage(A+D)
8	16	16	84.069%	11010	0.009%	86.029%	20226	0.016%
8	16	32	84.804%	15746	0.013%	85.784%	29570	0.024%
8	16	64	87.255%	25218	0.020%	85.539%	48258	0.039%
8	32	16	83.578%	15746	0.013%	85.294%	29570	0.024%
8	32	32	86.029%	20482	0.016%	87.500%	38914	0.031%
8	32	64	87.500%	29954	0.024%	86.765%	57602	0.046%
8	64	16	84.804%	25218	0.020%	86.275%	48258	0.039%
8	64	32	85.539%	29954	0.024%	87.010%	57602	0.046%
8	64	64	86.520%	39426	0.032%	87.010%	76290	0.061%
16	32	32	87.745%	39426	0.032%	87.255%	76290	0.061%
16	32	64	85.294%	58370	0.047%	88.480%	113666	0.091%
16	64	32	85.784%	58370	0.047%	87.010%	113666	0.091%
16	64	64	88.235%	77314	0.062%	86.029%	151042	0.121%
32	64	64	86.765%	153090	0.122%	86.520%	300546	0.240%

Table 7: RoBERTa-MRPC

r	a	b	Pearson (D)	Parameters (D)	Percentage (D)	Pearson (A+D)	Parameters (A+D)	Percentage (A+D)
8	16	16	86.598%	10241	0.008%	87.252%	19457	0.016%
8	16	32	88.103%	14977	0.012%	89.141%	28801	0.023%
8	16	64	89.513%	24449	0.020%	89.820%	47489	0.038%
8	32	16	87.523%	14977	0.012%	87.731%	28801	0.023%
8	32	32	88.570%	19713	0.016%	88.505%	38145	0.031%
8	32	64	89.677%	29185	0.023%	89.779%	56833	0.045%
8	64	16	87.221%	24449	0.020%	88.249%	47489	0.038%
8	64	32	88.112%	29185	0.023%	89.606%	56833	0.045%
8	64	64	89.652%	38657	0.031%	89.629%	75521	0.060%
16	32	32	89.326%	38657	0.031%	89.611%	75521	0.060%
16	32	64	89.731%	57601	0.046%	89.761%	112897	0.090%
16	64	32	88.807%	57601	0.046%	89.179%	112897	0.090%
16	64	64	89.517%	76545	0.061%	89.677%	150273	0.120%
32	64	64	89.559%	152321	0.122%	90.369%	299777	0.240%

Table 8: RoBERTa-STSB

r	a	b	Accuracy(D)	Parameters(D)	Percentage(D)	Accuracy(D+A)	Parameters(D+A)	Percentage(D+A)
8	16	16	78.619%	11010	0.010%	80.35%	20226	0.018%
8	16	32	81.112%	15746	0.014%	79.58%	29570	0.027%
8	16	64	80.633%	25218	0.023%	79.67%	48258	0.044%
8	32	16	78.715%	15746	0.014%	81.21%	29570	0.027%
8	32	32	80.633%	20482	0.019%	81.21%	38914	0.035%
8	32	64	81.016%	29954	0.027%	79.67%	57602	0.052%
8	64	16	77.661%	25218	0.023%	79.77%	48258	0.044%
8	64	32	80.153%	29954	0.027%	81.59%	57602	0.052%
8	64	64	81.783%	39426	0.036%	79.29%	76290	0.069%
16	32	32	79.195%	39426	0.036%	82.26%	76290	0.069%
16	32	64	81.208%	58370	0.053%	82.45%	113666	0.103%
16	64	32	80.345%	58370	0.053%	80.54%	113666	0.103%
16	64	64	80.825%	77314	0.070%	79.58%	151042	0.137%
32	64	64	81.400%	153090	0.139%	80.82%	300546	0.273%

Table 9: BERT-CoLA

r	a	b	Accuracy(D)	Parameters(D)	Percentage(D)	Accuracy(D+A)	Parameters(D+A)	Percentage(D+A)
8	16	16	64.260%	11010	0.010%	64.982%	20226	0.018%
8	16	32	58.123%	15746	0.014%	65.343%	29570	0.027%
8	16	64	66.426%	25218	0.023%	64.621%	48258	0.044%
8	32	16	61.733%	15746	0.014%	64.621%	29570	0.027%
8	32	32	63.899%	20482	0.019%	67.148%	38914	0.035%
8	32	64	61.733%	29954	0.027%	67.870%	57602	0.052%
8	64	16	62.816%	25218	0.023%	64.260%	48258	0.044%
8	64	32	63.538%	29954	0.027%	64.982%	57602	0.052%
8	64	64	67.870%	39426	0.036%	67.148%	76290	0.069%
16	32	32	62.455%	39426	0.036%	67.870%	76290	0.069%
16	32	64	61.372%	58370	0.053%	66.065%	113666	0.103%
16	64	32	64.260%	58370	0.053%	67.870%	113666	0.103%
16	64	64	64.260%	77314	0.070%	64.260%	151042	0.137%
32	64	64	61.011%	153090	0.139%	65.704%	300546	0.273%

Table 10: BERT-RTE

r	a	b	Accuracy(D)	Parameters(D)	Percentage(D)	Accuracy(D+A)	Parameters(D+A)	Percentage(D+A)
8	16	16	81.127%	11010	0.010%	84.559%	20226	0.018%
8	16	32	81.373%	15746	0.014%	82.598%	29570	0.027%
8	16	64	82.843%	25218	0.023%	81.127%	48258	0.044%
8	32	16	79.167%	15746	0.014%	80.637%	29570	0.027%
8	32	32	83.088%	20482	0.019%	82.353%	38914	0.035%
8	32	64	82.598%	29954	0.027%	81.373%	57602	0.052%
8	64	16	79.167%	25218	0.023%	84.559%	48258	0.044%
8	64	32	81.863%	29954	0.027%	80.882%	57602	0.052%
8	64	64	83.333%	39426	0.036%	78.922%	76290	0.069%
16	32	32	83.088%	39426	0.036%	83.578%	76290	0.069%
16	32	64	83.088%	58370	0.053%	79.902%	113666	0.103%
16	64	32	84.804%	58370	0.053%	80.637%	113666	0.103%
16	64	64	83.578%	77314	0.070%	81.373%	151042	0.137%
32	64	64	82.598%	153090	0.139%	84.314%	300546	0.273%

Table 11: BERT-MRPC

r	a	b	Pearson(D)	Parameters(D)	Percentage(D)	Pearson(D+A)	Parameters(D+A)	Percentage(D+A)
8	16	16	85.75%	10241	0.009%	86.14%	19457	0.018%
8	16	32	86.88%	14977	0.014%	87.53%	28801	0.026%
8	16	64	87.75%	24449	0.022%	87.41%	47489	0.043%
8	32	16	85.36%	14977	0.014%	86.58%	28801	0.026%
8	32	32	86.71%	19713	0.018%	87.05%	38145	0.035%
8	32	64	86.94%	29185	0.027%	87.86%	56833	0.052%
8	64	16	85.86%	24449	0.022%	86.91%	47489	0.043%
8	64	32	86.64%	29185	0.027%	87.34%	56833	0.052%
8	64	64	86.21%	38657	0.035%	88.02%	75521	0.069%
16	32	32	87.26%	38657	0.035%	87.64%	75521	0.069%
16	32	64	86.63%	57601	0.052%	87.06%	112897	0.103%
16	64	32	87.37%	57601	0.052%	87.36%	113666	0.103%
16	64	64	88.35%	76545	0.070%	88.25%	150273	0.137%
32	64	64	87.13%	152321	0.138%	87.98%	299777	0.273%

Table 12: BERT-STSB

r	a	b	Parameters	Percentage	bleu-1	bleu-2	bleu-3	rouge-1	rouge-2	rouge-L
8	16	16	20224	0.034%	31.90%	23.81%	18.17%	44.54%	26.57%	42.26%
8	16	32	30336	0.051%	33.63%	25.39%	19.57%	46.95%	28.60%	44.69%
8	16	64	50560	0.084%	34.23%	25.89%	20.01%	47.88%	29.42%	45.60%
8	32	16	30336	0.051%	32.66%	24.53%	18.84%	46.01%	27.81%	43.62%
8	32	32	40448	0.067%	34.03%	25.79%	19.94%	47.61%	29.31%	45.31%
8	32	64	60672	0.101%	34.36%	26.13%	20.29%	48.22%	29.92%	45.88%
8	64	16	50560	0.084%	33.19%	24.94%	19.12%	46.21%	27.83%	43.74%
8	64	32	60672	0.101%	34.15%	25.77%	19.88%	47.80%	29.31%	45.41%
8	64	64	80896	0.135%	34.27%	26.01%	20.15%	47.88%	29.56%	45.62%
16	32	32	80896	0.135%	33.92%	25.60%	19.75%	47.25%	28.95%	45.00%
16	32	64	121344	0.202%	34.54%	26.25%	20.34%	48.14%	29.77%	45.87%
16	64	32	121344	0.202%	34.07%	25.75%	19.89%	47.66%	29.36%	45.38%
16	64	64	161792	0.270%	34.27%	26.02%	20.19%	48.03%	29.89%	45.88%
32	64	64	323584	0.539%	34.09%	25.82%	19.98%	47.75%	29.52%	45.54%

Table 13: T5-Small

r	a	b	Parameters	Percentage	bleu-1	bleu-2	bleu-3	rouge-1	rouge-2	rouge-L
8	16	16	40192	0.018%	34.63%	26.32%	20.42%	48.22%	29.76%	45.87%
8	16	32	60288	0.027%	34.85%	26.70%	20.80%	48.79%	30.70%	46.52%
8	16	64	100480	0.045%	34.96%	26.80%	20.89%	49.07%	30.99%	46.76%
8	32	16	60288	0.027%	34.46%	26.11%	20.22%	48.26%	30.02%	45.93%
8	32	32	80384	0.036%	35.04%	26.81%	20.97%	49.07%	31.08%	46.77%
8	32	64	120576	0.054%	34.84%	26.62%	20.75%	49.10%	30.93%	46.79%
8	64	16	100480	0.045%	34.78%	26.53%	20.63%	48.71%	30.50%	46.35%
8	64	32	120576	0.054%	34.91%	26.73%	20.88%	48.92%	30.91%	46.69%
8	64	64	160768	0.072%	35.16%	26.83%	20.88%	49.20%	30.94%	46.98%
16	32	32	160768	0.072%	35.01%	26.84%	21.00%	48.95%	30.97%	46.72%
16	32	64	241152	0.108%	34.82%	26.62%	20.77%	48.91%	30.67%	46.59%
16	64	32	241152	0.108%	35.21%	27.05%	21.17%	49.21%	31.14%	46.98%
16	64	64	321536	0.144%	35.18%	26.99%	21.12%	49.07%	30.98%	46.77%
32	64	64	643072	0.288%	35.09%	26.92%	21.08%	48.96%	31.04%	46.78%

Table 14: T5-Base