Survival of the Smallest: Genetic Algorithms for LLM Compression

Anirudh G. Joshi*
Department of Computer Science
PES University
Bengaluru, India
janirudhg@gmail.com

Badri Prasad V R
Associate Professor
Department of Computer Science
PES University
Bengaluru, India
badriprasad@pes.edu

***Abstract*—Large Language Models demonstrate exceptional capabilities in language comprehension and generation, but their significant size presents challenges for deployment and inference. This research investigates the compression of LLMs by employing structural pruning, specifically targeting the removal of redundant attention heads. Genetic algorithms are used to efficiently search for optimal sub-models within a vast combinatorial space. The fitness of sub-models is assessed based on performance metrics, ensuring minimal degradation from the original LLM. The method uses only forward passes of the model, and eliminates the need for the resource-intensive task of gradient calculations. This novel method seeks to make LLMs more accessible by facilitating their deployment on resource-constrained devices. Our method achieves accuracy losses of less than 4% to provide a speedup of 1.28x.**

***Index Terms*—Large Language Models, Structural Pruning, Genetic Algorithms, Model Compression**

I. Introduction

Large Language Models (LLMs) have revolutionized natural language processing with their remarkable ability to understand and generate human-like text. However, their practical deployment in real-world applications presents a significant challenge due to their massive size. This substantial size leads to the usage of huge computational resources and memory, hindering their integration into resource-constrained environments such as edge devices or personal computers. As state-of-the-art models grow increasingly compute-intensive, democratizing LLMs requires effective compression techniques that can reduce their size without compromising performance.

We explore the compression of LLMs using structural pruning, a technique that targets the removal of entire structural units within a model – in our case, attention heads. Structural pruning leverages the inherent redundancy often present in LLMs, recognizing that not all modules contribute equally to a model's performance [8]. By identifying and removing these less impactful modules, the model is compressed without drastically impacting its functionality.

Large Language Models are built using transformer architectures, which rely on multi-head attention layers to process and understand text. A sub-model is a subset of the original LLM that has a subset of the attention heads pruned. To efficiently navigate the vast combinatorial space of sub-models resulting from structural pruning, this research proposes a novel approach utilizing genetic algorithms. Genetic algorithms are powerful for searching through complex solution spaces.

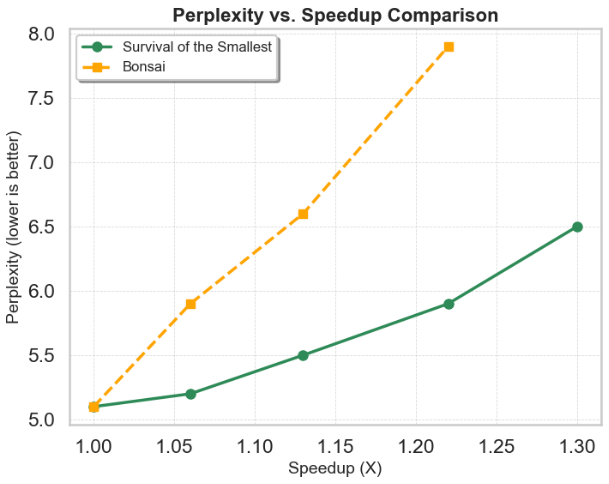

Fig. 1. A comparison of perplexity vs speedup with Bonsai over LLaMA-2 7B on the Wikitext-2 dataset. **Survival of the Smallest offers lower perplexity at every sparsity rate of the LLM**

Each sub-model is represented as a chromosome, and the genetic algorithm iteratively evolves a population of these chromosomes, guided by a fitness function that evaluates the performance of the sub-models. The fitness function incorporates metrics such as sparsity and accuracy in downstream tasks, ensuring that the compressed model not only achieves a desired reduction in size, but also maintains acceptable performance. Our objective is to use only forward passes of the model, therefore, avoiding the resource-intensive task of training the model. Our approach eliminates the need for post-pruning fine-tuning of the LLM, as the obtained results are convincing.

II. Related Works

A. *Structural Pruning of Large Language Models (LLMs)*

Structured Pruning of LLMs: Structured pruning [19], improves upon unstructured pruning by eliminating entire rows

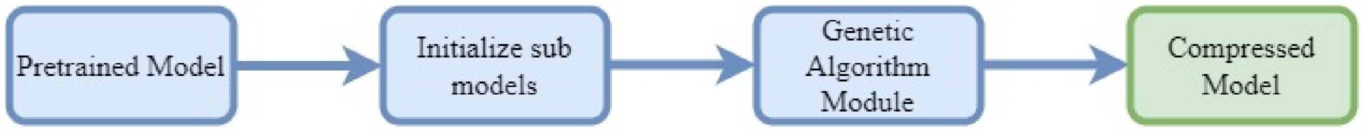

Fig. 2. Overview of the process of LLM compression using GAs

or columns of weights, leading to significant reductions in both model parameters and inference time without relying on specialized hardware. Unlike unstructured methods, which prune individual weights in isolation [3], structured pruning captures redundancy at a higher level, making it more effective for real-world deployment. Recent approaches such as LLM-Pruner [1] have attempted structured pruning, but their reliance on LoRA fine-tuning [22] introduces a trade-off between computational cost and pruning effectiveness, limiting their scalability for larger models. For optimal structured pruning in LLMs, three criteria are essential: (a) a structured importance metric to assess redundancy, (b) an adaptive mechanism for global model compression, and (c) a compensation strategy to mitigate performance degradation.

LLM-Pruner [1] identifies coupled structures within the network and assesses their impact on the model's predictions. It uses a dependency-based analysis, starting from an initial neuron and progressively identifying the neurons that rely on it. This iterative process leads to the pruning of those neurons with the least influence on the model's output.

LoRAShear [5] leverages dependency graphs to pinpoint minimally removable structures. It uses a progressive pruning technique applied to LoRA adapters, which are small, trainable matrices that preserve information stored in redundant structures and employs dynamic fine-tuning schemes to recover lost knowledge. However, these techniques rely on post-training the pruned models to recover their performance.

Bonsai [4] introduces a method that involves generating multiple sub-models, each containing a different subset of modules (such as attention heads or rows in feedforward projection) and choosing the best ones from them. Linear regression is used to estimate the importance of each module based on its contribution to the overall performance of the sub-model. The modules are then iteratively pruned until the desired sparsity level is reached. However, this approach requires a high runtime to perform the pruning process.

SLEB [7] leverages the fact that there is output similarity among consecutive transformer blocks to prune redundant blocks. Each block is assessed on the basis of its token prediction performance on the calibration data. The blocks which have a minimal effect are removed without much reduction in the model's generation abilities.

B. Genetic Algorithms for Model Compression

The primary motivation for us to explore the use of genetic algorithms (GAs) for LLM compression came from the work [2] in which GAs have been used to compress convolutional neural networks (CNNs). This approach uses chromosomes to represent model structures and an accuracy-based fitness function. It serves as a proof of concept for model compression. However, adapting these principles to the more complex architecture of LLMs requires careful consideration of chromosome representation, fitness function design, and genetic operators.

TABLE I
COMPARISON WITH OTHER APPROACHES

Approaches	**Component Pruned**	**Post Pruning**	**Back-Propagation**
Everybody Prune Now	Attention heads	Yes	No
SLEB	Transformer blocks	No	Yes
LLM-Pruner	Neuron structures	Yes	No
LoRAShear	Node Groups	Yes	Yes
Survival of the Smallest (Ours)	Attention heads	No	No

Other evolutionary methods like Artificial Immune Systems (AIS) and Ant Colony Optimization (ACO) lack the scalability and adaptability needed for problems in the scale of LLMs. While AIS suffers from slow convergence and hyperparameter sensitivity, ACO struggles with stagnation and inefficiency in large spaces. Genetic algorithms such as crossover and mutation offer ideal solutions for such high-dimensional tasks, enabling efficient exploration of complex solution spaces.

III. METHODOLOGY

Previous research [8] has shown that not all attention heads in the multi-head self-attention architecture of the LLM contribute equally to the model's understanding and generation capabilities. Therefore, attention heads are good targets for pruning.

We use the concept of a sub-model [4], which is a smaller model that contains a subset of attention heads from the original model. Combining this concept with the methods of genetic algorithms, we define a population, P, consisting of a set of sub-models. A fitness function is defined to evaluate the performance of sub-models. Our objective is to evolve the population so that the resulting sub-model not only requires less resources but also performs well on the data.

A. Chromosome representation

Each sub-model in the population is represented as a chromosome. A chromosome consists of multiple genes, where each gene corresponds to an attention head. Therefore, the number of genes present in all chromosomes of the population will be equal to the number of attention heads in the original model. The gene values are binary, indicating whether the respective component is retained (1) or pruned (0) for that sub-model.

B. Initializing the population

We start with a population of sub-models having a small, fixed ratio, $s_{initial}$, of attention heads removed. This is done in a random fashion [15]. The size of the population is a hyperparameter that can be varied based on the model chosen.

C. Fitness Function

The fitness of each chromosome is evaluated based on the performance of the pruned model configuration on a downstream task. The fitness function penalizes sub-models which have a significant degradation in performance as compared to the original model, while configurations that preserve performance are rewarded. The fitness value is also dependent on the sparsity of the sub-model, rewarding higher sparsities. For a sub-model p_i, the fitness of the sub-model, along with its sparsity, s_i can be based on metrics such as perplexity and BLEU score given by the model over a set of samples. For example:

$$f(p_i) = -\text{perplexity}(p_i) \cdot (1 - s_i) \tag{1}$$

Lower perplexity indicates better model performance. The equation is negated to ensure that sub-models with lower perplexity receive higher fitness scores. For metrics such as accuracy, the equation is not negated.

D. Genetic algorithms

Genetic algorithms are applied to the population, and the chromosomes are replaced by their offspring after each generation.

1) Selection: The parents required to create the next generation of sub-models are selected using the tournament selection method. The tournament size in our implementation was considered to be 3.

2) Crossover: We design an innovative algorithm that makes sure that the offspring sub-model inherits the best traits of both its parents. In this algorithm, the two parents are given weights based on their fitness. The choice of each gene for the child is random, with a higher weight given to the parent with the higher fitness. Therefore, for each gene, the child has a higher probability of inheriting the trait of the fitter parent. This emulates a real-life "survival of the fittest" situation.

3) Mutation: It must be ensured that the sub-models are not caught in a local optimum of the solution space. Mutation is performed to introduce randomness into the algorithm so that more exploration of the search space is possible. Flip-bit mutation is employed with mutation rate as the hyperparameter.

Mutation plays a dual role in our algorithm:

(a) ensuring that the chromosomes do not get stuck around the local maximum of the fitness function, introducing diversity into the search, and

(b) increasing the sparsity of the model by randomly flipping the bits set to 1 to 0.

Algorithm 1 Genetic Algorithm for Structural Pruning of LLMs

```
Require: Model weights $W$, initial population size $N$, number of generations $G$, fitness threshold $\tau$, maximum sparsity rate $S$
Ensure: Pruned model with weights $\hat{W}$
1: Initialize population $\mathcal{P}$ with $N$ random pruning configurations
2: for $g \leftarrow 1$ to $G$ do
3:     for each individual $p_i \in \mathcal{P}$ do
4:         Evaluate fitness $f(p_i)$
5:     end for
6:     Select parents using tournament selection based on $f(p_i)$
7:     Generate offspring $\mathcal{P}_{\text{new}}$ via crossover of the parents
8:     Introduce mutation into $\mathcal{P}_{\text{new}}$
9:     Update population: $\mathcal{P} \leftarrow \mathcal{P}_{\text{new}}$
10:    if $\max(f(p)) \geq \tau$ or $\max(\text{sparsity}) \geq S$ then
11:        break
12:    end if
13: end for
14: Identify the best sub-model: $p^* \leftarrow \arg\max f(p)$
15: Prune $W$ using $p^*$ to get $\hat{W}$
16: return $\hat{W}$
```

Algorithm 2 Crossover Function for Genetic Algorithm

```
Require: Parents $\alpha_1$, $\alpha_2$; fitness scores $f(\alpha_1)$, $f(\alpha_2)$; crossover rate $C$;
Ensure: Children $\beta_1$, $\beta_2$
1: $F(\alpha) \leftarrow f(\alpha_1) + f(\alpha_2)$
2: $prob(\alpha_1) \leftarrow f(\alpha_1)/F(\alpha)$
3: Initialize empty children: $\beta_1 \leftarrow []$, $\beta_2 \leftarrow []$
4: if random number $r \in [0,1] <$ crossover_rate then
5:     for gene $g_1$ in $\alpha_1$ and gene $g2$ in $\alpha_2$ do
6:         if random number $r_1 \in [0,1] <$ $prob(\alpha_1)$ then
7:             add $g_1$ to $\beta_1$
8:         else
9:             add $g_2$ to $\beta_2$
10:        end if
11:    end for
12: else
13:    $\beta_1 \leftarrow \alpha_1$, $\beta_2 \leftarrow \alpha_2$
14: end if
15: return $\beta_1$, $\beta_2$
```

4) Termination Criteria: The algorithm iterates until a predefined convergence criterion is met, either when the model achieves a satisfactory balance between performance and compression or when the maximum number of generations is reached.

TABLE II
PERFORMANCE COMPARISON OF MODELS BEFORE AND AFTER OPTIMIZATION

Model	Metric	Before	After
RoBERTa-base	Accuracy (%)	61.78	59.69
	Execution Time (s)	4.62	**3.37**
DistilBERT-base	Accuracy (%)	98.8	98.1
	Execution Time (s)	2.67	**2.37**
Gemma - 2B	Precision (%)	80.32	76.94
	Execution Time (s)	85	**72**
LLaMA-2 7B	Perplexity	5.1	6.5
	Model Size (MB)	6676.51	**6042.51**

IV. RESULTS

All experiments were conducted on an NVIDIA GeForce RTX 3090 (24GB) GPU. The time taken to run 100 generations of a population of 10 individuals for the LLaMA-2 7B model was about 2 hours.

Pruning done on a RoBERTa-base model fine-tuned for multi-label topic classification on Twitter data showed encouraging results. Without pruning, the model achieved an accuracy of 61.78% with an execution time of 4.62 seconds. After pruning, the accuracy decreased slightly to 59.69%, while the execution time reduced to 3.37 seconds, reflecting a 27% decrease.

A DistilBERT-base model was fine-tuned for sentiment classification on the SST-2 dataset. The unpruned model achieved an accuracy of 98.8% with an execution time of 267 seconds. After pruning, the accuracy decreased marginally to 98.1%, while the execution time reduced to 237 seconds.

The base Gemma model had a size of 9560 MB and achieved a precision of 80.32% on text summarization tasks. Pruning resulted in a 10% reduction in model size, from 475 MB to 429 MB, while the accuracy decreased by only 3.38%. These results highlight the potential of pruning to significantly compress LLMs while maintaining a comparable level of performance.

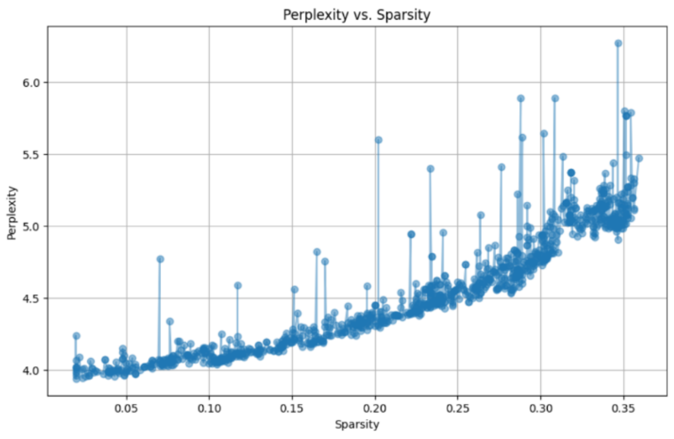

Fig. 3. Plot of perplexity versus sparsity for all the chromosomes across all generations for the LLaMA-2 7B model on the Wikitext-2 dataset. Perplexity spikes at different sparsities due to the randomness introduced by mutation and crossover. However, survival of the fittest ensures that perplexity is minimized.

The LLaMA-2 7B parameters model was used with the WikiText-2 dataset to prove the correctness of our algorithm. The size was reduced by about 27%, giving a speedup of 1.3x. The perplexity measure, which was initially 5.1 in the unpruned model, increased to about 6.5, which is a significantly smaller increase compared to the perplexity increase when the Bonsai method was used.

In the sub-model space explored using genetic algorithms, it was found that as more attention heads are pruned, the sparsity increases, and this leads to an increase in the perplexity of the pruned model.

A. *Effect of mutation on accuracy*

- We observed that about 1% mutation rate is ideal for introducing randomness into the search space. This amount ensures that the model retains most of its parents' "good" traits, while also changing a few genes to let the algorithm explore. It also plays the important role of increasing the sparsity of the model, as initially, the higher number of 1s are flipped to 0s.
- However, for mutation rates 5% and above, we observed a significant degradation in the performance of the sub-models. This could be attributed to the increased randomness, which also propagates to the future generations. Moreover, the sparsity also increased suddenly, without estimating the best sub-models at that sparsity level.

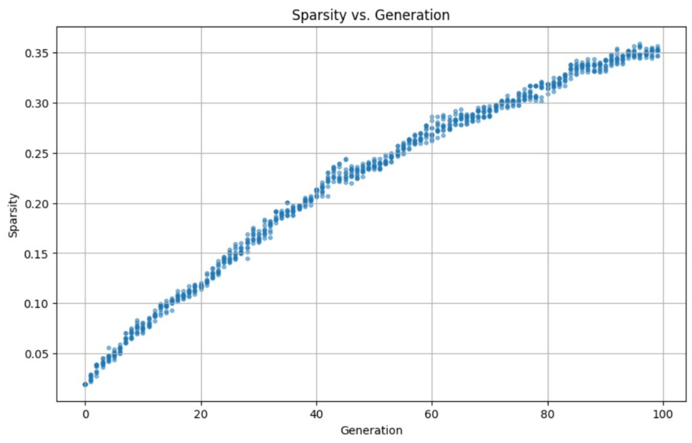

Fig. 4. Plot of sparsity versus the number of generations for the algorithm performed on the LLaMA-2 model on the Wikitext-2 dataset. Our method is designed to increase sparsity over generations - favouring individuals with higher sparsities.

V. CONCLUSION

In this work, we introduce a novel approach to prune LLMs using genetic algorithms. GAs efficiently search for the best sub-models, without relying on gradient-based updates. By pruning the redundant attention heads, our method efficiently navigates through the space to find optimal models that balance the trade-off between sparsity and performance. Experimental results demonstrate that our approach produces compact, computationally efficient models that retain strong downstream task performance.

Despite these advantages, limitations remain. The process of searching through multiple sub-models, while effective, introduces additional computational overhead compared to heuristic pruning strategies (about 200 more forward passes than [3]). Additionally, our method does not incorporate explicit fine-tuning post-pruning, which could further improve performance recovery.

Beyond pruning, genetic algorithms present exciting opportunities for broader model compression strategies, including hybrid approaches that combine pruning with quantization or distillation. This research direction holds promising implications for deploying LLMs in resource-constrained environments and making them more accessible for various applications.

References

[1] X. Ma, G. Fang, and X. Wang, "LLM-Pruner: On the structural pruning of large language models," *arXiv preprint*, arXiv:2305.11627, 2023. [Online]. Available: https://arxiv.org/abs/2305.11627

[2] M. Agarwal, S. K. Gupta, M. Biswas, and D. Garg, "Compression and acceleration of convolution neural network: a genetic algorithm based approach," Journal of Ambient Intelligence and Humanized Computing, vol. 14, pp. 13387–13397, 2023. [Online]. Available: https://doi.org/10.1007/s12652-022-03793-1

[3] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," arXiv preprint, arXiv:2306.11695, 2024. [Online]. Available: https://arxiv.org/abs/2306.11695

[4] L. Dery, S. Kolawole, J.-F. Kagy, V. Smith, G. Neubig, and A. Talwalkar, "Everybody Prune Now: structured pruning of LLMs with only forward passes," arXiv preprint, arXiv:2402.05406, 2024. [Online]. Available: https://arxiv.org/abs/2402.05406

[5] T. Chen, T. Ding, B. Yadav, I. Zharkov, and L. Liang, "LoRAShear: efficient large language model structured pruning and knowledge recovery," arXiv preprint, arXiv:2310.18356, 2023. [Online]. Available: https://arxiv.org/abs/2310.18356

[6] A. Vaswani et al., "Attention is All You Need," Advances in Neural Information Processing Systems, vol. 30, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[7] J. Song, K. Oh, T. Kim, H. Kim, Y. Kim, and J.-J. Kim, "SLEB: Streamlining LLMs through Redundancy Verification and Elimination of Transformer Blocks," arXiv preprint, arXiv:2402.09025, 2024. [Online]. Available: https://arxiv.org/abs/2402.09025

[8] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned," in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 5797–5808. [Online]. Available: https://arxiv.org/abs/1905.09418

[9] T. B. Brown et al., "Language Models are Few-Shot Learners," Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[10] E. Frantar and D. Alistarh, "SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot," arXiv preprint, arXiv:2301.00774, 2023. [Online]. Available: https://arxiv.org/abs/2301.00774

[11] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer Sentinel Mixture Models," arXiv preprint, arXiv:1609.07843, 2016. [Online]. Available: https://arxiv.org/abs/1609.07843

[12] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, "BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota, 2019, pp. 2924–2936. [Online]. Available: https://aclanthology.org/N19-1300/https://arxiv.org/abs/1905.10044

[13] F. Lagunas, E. Charlaix, V. Sanh, and A. M. Rush, "Block Pruning For Faster Transformers," in Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 1043–1059. [Online]. Available: https://arxiv.org/abs/2109.04838

[14] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Advances in Neural Information Processing Systems, vol. 32, 2019. [Online]. Available: https://arxiv.org/abs/1912.01703

[15] M. Santacroce, Z. Wen, Y. Shen, and Y. Li, "What Matters in the Structured Pruning of Generative Language Models?" arXiv preprint arXiv:2302.03773, 2023. [Online]. Available: https://arxiv.org/abs/2302.03773

[16] Z. Liu et al., "Deja Vu: Contextual Sparsity for Efficient LLMs at Inference Time," arXiv preprint arXiv:2310.17157, 2023. [Online]. Available: https://arxiv.org/abs/2310.17157

[17] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, "Can a suit of armor conduct electricity? A new dataset for open book question answering," arXiv preprint arXiv:1809.02789, 2018. [Online]. Available: https://arxiv.org/abs/1809.02789

[18] Y. An, X. Zhao, T. Yu, M. Tang, and J. Wang, "Fluctuation-based Adaptive Structured Pruning for Large Language Models," arXiv preprint arXiv:2312.11983, 2023. [Online]. Available: https://arxiv.org/abs/2312.11983

[19] Y. He and L. Xiao, "Structured Pruning for Deep Convolutional Neural Networks: A Survey," arXiv preprint arXiv:2303.00566, 2023. [Online]. Available: https://arxiv.org/abs/2303.00566

[20] T. Dettmers et al., "SpQR: A Sparse-Quantized Representation for Near-Lossless LLM Weight Compression," arXiv preprint arXiv:2306.03078, 2023. [Online]. Available: https://arxiv.org/abs/2306.03078

[21] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers," arXiv preprint arXiv:2210.17323, 2022. [Online]. Available: https://arxiv.org/abs/2210.17323

[22] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models," arXiv preprint arXiv:2106.09685, 2021. [Online]. Available: https://arxiv.org/abs/2106.09685