Interpolated Layer-Wise Merging for NeurIPS 2024 LLM Merging Competition

Rio Akizuki Nozomu Yoshinari Yuya Kudo Yoichi Hirose
Toshiyuki Nishimoto Kento Uchida Shinichi Shirakawa
Yokohama National University
akizuki-rio-pd@ynu.jp yoshinari-nozomu-ry@ynu.jp
kudo-yuya-dr@ynu.jp hirose-youichi-kc@ynu.jp
nishimoto-toshiyuki-gf@ynu.jp uchida-kento-fz@ynu.ac.jp
shirakawa-shinichi-bg@ynu.ac.jp

Abstract

Model merging is an emerging technique to generate models by combining multiple models, leveraging their individual strengths in a unified model. Because the configuration hyperparameters of model merge techniques have a significant effect on the merged models, optimizing them using evolutionary algorithms is promising to enhance the model merge results. In our participation in the LLM Merging Competition in NeurIPS 2024, we introduced a novel search space, *interpolated layer-wise space*, for optimizing merging configurations using evolutionary algorithms. This paper explores the potential for performance enhancement in merged models and the efficiency of our search space for evolutionary model merge.

1 Introduction

Rapid advancement of LLM has significantly impacted natural language processing, enabling breakthroughs in tasks such as machine translation, text summarization, and conversational AI. As these models become increasingly complex and huge, merging multiple LLMs to create more powerful and efficient systems has emerged as a crucial research area. Evolutionary algorithms can be powerful tools to optimize merging configurations in this context [1]. The literature of [1] is a pioneer work of optimizing merging configurations, where two merging approaches are considered: the parameter space (PS) merging and the data flow space (DFS) merging. In the PS merging, the weights of multiple candidate models are integrated into a merged model with the same architecture. Several merging techniques have been developed, such as TIES-Merging [2]. In [1], the merging configurations of DARE-TIES [2, 3] are optimized using the covariance matrix adaptation evolution strategy (CMA-ES) [4].

The performance and optimization efficiency of evolutionary algorithms heavily depend on the search space. Typically, when optimizing the PS merging configurations, the *model-wise* search space in which the entire model is treated as single units during the optimization process is used. Namely, the common merging configuration parameters are used for all weights in each candidate model. This model-wise search space is simple and can reduce the search space size but may not fully leverage the potential of combining different models. A straightforward extension to enhance granularity is the *layer-wise* approach, which optimizes the merging configurations for each layer. While layer-wise merging offers a more fine-grained search space, our preliminary experiments revealed that this approach leads to an excessively large search space, leading to difficult and computationally expensive optimization problems.

To address these challenges, we propose a novel search space called *interpolated layer-wise space*. In our search space, the merging parameters for a part of the layers are optimized, and the merging

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

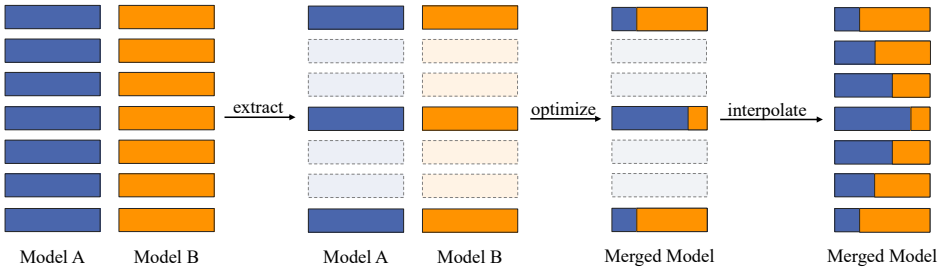


Figure 1: Computation process of merge parameters on the interpolated layer-wise space.

parameters for other layers are computed by interpolation methods. The proposed search space has a higher expressive capability than the model-wise approach while keeping the number of design variables to be optimized lower than the layer-wise approach. Therefore, optimizing merging configurations using our search space is expected to improve the search efficiency without sacrificing the performance of the merged models compared to the layer-wise approach.

We conducted experiments to evaluate the effectiveness of our search space. The results demonstrate that the CMA-ES converges earlier on the interpolated layer-wise space than the layer-wise search space without sacrificing performance. These findings suggest that our search space offers a practical trade-off between the performance of the merged model and the computational budget for optimization. We submitted the model merging code for the competition using the merging configuration parameters obtained by optimization in our interpolated layer-wise search space.

2 Interpolated Layer-Wise Space

We introduce our proposed search space, the *interpolated layer-wise space*, for evolutionary model merge. The general optimization process on our search space is illustrated in Figure 1.

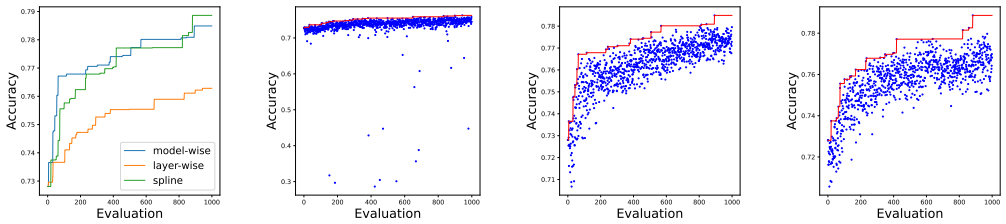
2.1 Design Variable on Interpolated Layer-Wise Space

In interpolated layer-wise space, n layers are selected to construct the search space for evolutionary algorithms. We determined these layers by dividing the model's entire layers with even intervals. Then, in evolutionary model merge, the merge parameters for n pre-selected layers are sampled by evolutionary algorithm.

Given the number of layers in the model l, the number of models m, and the number of merge parameters p for each layer, the number of possible configuration parameters on our interpolated layer-wise search space is $n \times m \times p$, which reduces the search space dimensions n/l times compared to $l \times m \times p$ for the layer-wise space.

2.2 Interpolation of Merge Parameters

For non-selected layers, the merge parameters are computed by interpolation using the merge parameters for the pre-selected layers. We expect that this interpolation ensures that all layers receive fine-grained merge parameters, improving the performance of the merged model compared to the model-wise space. In the optimization process of evolutionary model merge, the evaluation of the merged model is performed with interpolated merge parameters. Although several interpolation methods can be applied to our search space, we used cubic spline-interpolation provided by `scipy` in this report. This interpolation approach implicitly assumes that the good merging configuration parameters smoothly change between those of pre-selected layers.



(a) All Methods (b) layer-wise space (c) model-wise space (d) **ours**

Figure 2: Optimization histories across different evolutionary model merge methods.

3 Experiments

3.1 Experimental Setting

Source Models We used suzume-llama-3-8B-multilingual-orpo-borda-top75, Barcenas-Llama3-8b-ORPO, Llama-3-8B-Ultra-Instruct-SaltSprinkle, MAmmoTH2-8B-Plus [5], Daredevil-8B.[1] These five models are fine-tuned from Meta-Llama-3-8B-Instruct [6].

Dataset For optimization, we used tinyBenchmarks [7] in lm-evaluation-harness [8], which is open-source and consists of six tasks.

Optimization We used the CMA-ES [4] with default hyperparameter setting as the optimizer. CMA-ES optimized the DARE-TIES [2, 3] parameters for 5 layers per model, resulting in the dimension of the design variables being 50. The other layers were spline-interpolated based on the parameters of the 5 layers. The population size was 15, and the maximum number of evaluations was 1000. The fitness value for each solution was the average of the evaluation values for each task in tinyBenchmarks [7]. The mean vector of the search distribution in CMA-ES after all evaluations was used as the optimization result.

Baseline Methods We used the merged models with the DARE-TIES [2, 3] using optimized parameters by CMA-ES on model-wise and layer-wise spaces as baselines.

3.2 Results

Figure 2a displays the optimization histories of all methods, and Figures 2b, 2c, and 2d display the optimization history and evaluation scores for each method.

We observe in Figure 2a that the merged model on our search space achieved a higher score than that on the layer-wise space. In addition, the improvement of evaluation values by layer-wise space was slower than other search spaces due to the large search space. Comparing the optimization histories on the layer-wise space (in Fig. 2b) and the proposed search space (in Fig. 2d), the worse solutions were generated in the layer-wise space than in our search space. This indicates that the evolutionary model merge on our search space is more efficient without sacrificing model performance by reducing the number of parameters to be optimized.

On the other hand, the performance of merged models on our search space and the model-wise space were competitive in Figure 2a. One reason is that the proposed search space had an unnecessarily larger number of parameters for this task. Figure 3 shows the optimized merge parameters, which implies that our method can set the fine-grained merge parameters for each layer.

Table 1 shows the results of validation set (kaggle leaderboard) for each method. We note that we modified the prompt of the sample code for evaluation.

[1]These models are obtained from `Hugging Face` `https://huggingface.co/`.

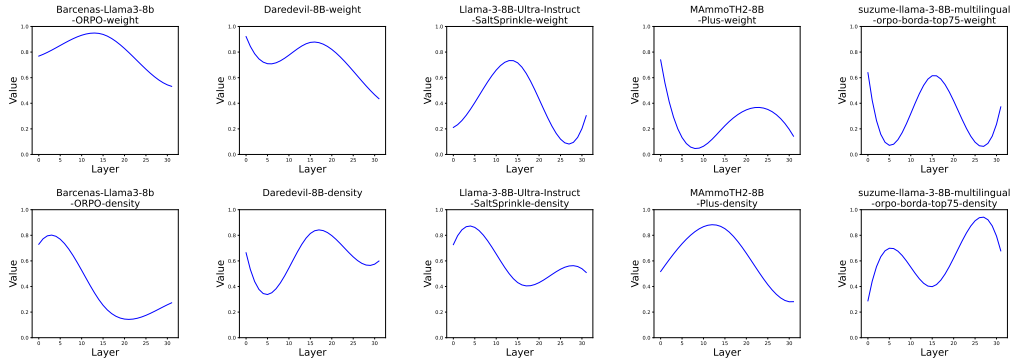


Figure 3: Weight parameters (upper row) and density parameters (lower row) obtained by evolutionary model merge on our search space.

Table 1: Validation scores on the open leaderboard of the LLM Merging Competition across different evolutionary model merging methods.

No.	Method	Valid Score
1	model-wise	0.59
2	layer-wise	0.51
3	**ours**	0.54

4 Conclusion

In this report, we present the interpolated layer-wise search space for the NeurIPS 2024 LLM Merging Competition. The proposed search space improved the performance of evolutionary model merge compared with the layer-wise. While optimizing merging configurations is an attractive approach, depending on search space, it requires a large number of evaluations for candidate merged models, leading to a high computational cost. We believe that it is essential to investigate more efficient optimization methods and sophisticated search space design in future work.

References

[1] Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *arXiv preprint arXiv:2403.13187*, 2024.

[2] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36, 2024.

[3] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *International Conference on Machine Learning*, 2024.

[4] Nikolaus Hansen. *The CMA Evolution Strategy: A Comparing Review*, pages 75–102. Springer Berlin Heidelberg, 2006.

[5] Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. MAmmoTH2: Scaling instructions from the web. *arXiv preprint arXiv:2405.03548*, 2024.

[6] AI@Meta. Llama 3 model card, 2024. `https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md`.

[7] Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating LLMs with fewer examples. In *International Conference on Machine Learning*, 2024.

[8] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2024.