MAP: Low-compute Model Merging with Amortized Pareto Fronts via Quadratic Approximation

Lu Li[1*], Tianyu Zhang[2,3*], Zhiqi Bu[4*], Suyuchen Wang [2], Huan He[1], Jie Fu[4], Jiang Bian[5], Yonghui Wu[5], Yong Chen[1], Yoshua Bengio[2]

[1] University of Pennsylvania, [2] MILA [3] ServiceNow [4] AWS [5] HKUST [6] University of Florida

{luli1, huanhe, ychen}@upenn.edu

{tianyu.zhang, yoshua.bengio}@mila.quebec

zhiqibu@aws, jiefu@hkust, {bianjiang, yonghuiwu}@uflorida.edu

Abstract

Model merging has emerged as an effective approach to combine multiple single-task models into a multitask model. However, existing methods focus on enhancing average task accuracy, often neglecting the trade-offs between different tasks. We introduce Model Merging with Amortized Pareto Front (MAP), a novel low-compute algorithm that efficiently identifies a Pareto set of scaling coefficients for merging multiple models. MAP uses a quadratic approximation surrogate model to estimate task metrics, enabling amortized inference. Our approach is particularly valuable in federated learning scenarios, where it can balance performance across diverse client datasets while respecting privacy constraints and minimizing communication overhead. Experimental results on vision and natural language processing tasks demonstrate MAP's ability to accurately identify the Pareto front, offering practitioners a range of solutions with various trade-offs. This makes MAP a promising approach for optimizing multitask performance in both centralized and distributed learning environments, addressing the challenges of task conflicts and privacy preservation in model merging.

1 Introduction

Large pre-trained foundation models have become available in many real-world applications [60; 54; 13]. This increasing availability has led to a popular practice of fine-tuning these pre-trained models to adapt to a wide range of downstream tasks. Practitioners can independently fine-tune the same pre-trained model, such as CLIP style models [45; 63; 69], large language models [6; 47; 55; 28], etc., and then release the fine-tuned models without releasing the training data. As the deployment of such fine-tuned models increases, combining models with identical architectures and initializations has emerged as a promising approach to combine their respective capabilities. This is useful, especially in scenarios where the training data for each task is private and cannot be shared, such as individual-level patient data in a hospital and behavior data in social media recommendation systems.

Existing methods for merging models typically involve calculating a weighted average of the parameters from multiple models to enhance performance uniformly across various tasks. However, this approach often overlooks the conflicts among the diverse objectives of these tasks, which can lead to trade-offs in terms of model performance on various tasks. In real-world applications, it is often useful to obtain a set of Pareto optimal solutions rather than a single model. Such solutions allow practitioners to choose among different trade-offs, depending on their specific needs. For

[*]Equal contribution.

Preprint. Under review.

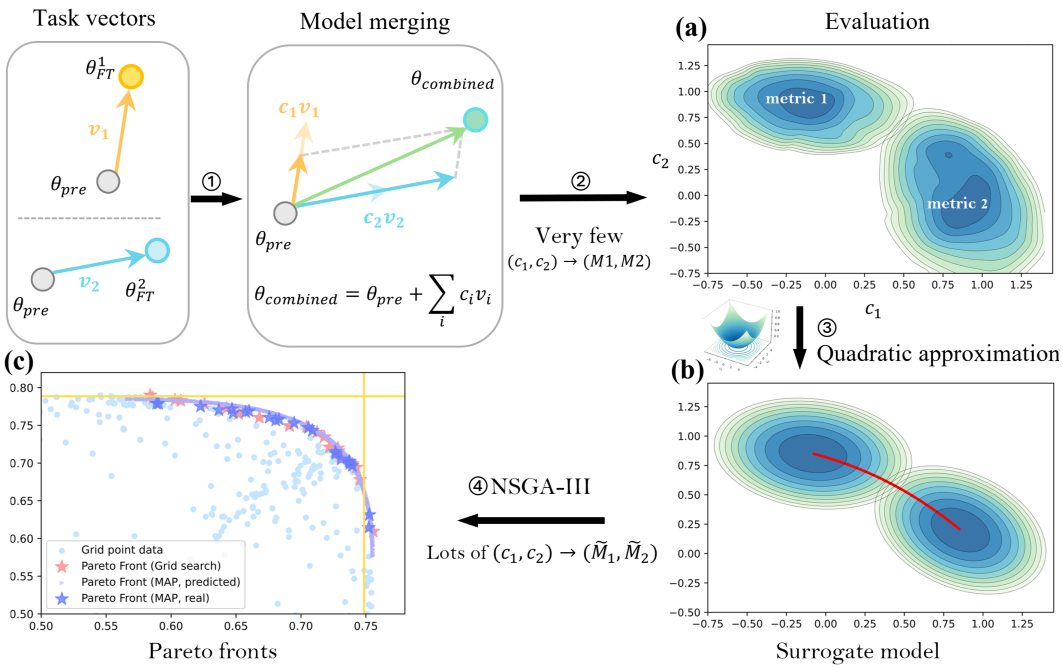


Figure 1: The overall process of MAP. Step 1: select 2 tasks and their corresponding task vectors. Step 2: sample a few scaling weights $\mathbf{c}$ to query the task metrics accordingly. Step 3: use the Quadratic model to approximate the $\mathbf{c} \rightarrow$ *metrics* mapping as a surrogate model. Step 4: Use NSGA-III to find amortized Pareto fronts. Figure (a) shows a contour plot of the actual accuracy landscape for the ViT models [17] obtained from 100 scaling coefficients (sampled uniformly) evaluated on the SUN397[64] and Cars[31]. Figure (b) shows a contour plot of the fitted quadratic functions. Red lines represent the Pareto front in the decision variable (c_1, c_2) space. Figure (c) shows an example of the Pareto Front. The Pareto front (Grid search) is regarded as the ground truth given the sufficient number of grid points. The Pareto front (MAP, predicted) is the amortized Pareto front. The Pareto front (MAP, real) is the Pareto front involving the same $\{(c_1, c_2)\}$ but re-evaluated to obtain the ground truth metrics for comparison. The yellow lines are the fine-tuned single-task models' evaluated performance.

example, hospitals specializing in certain areas might prefer a model that excels in tasks relevant to their specialty while maintaining adequate performance across a broader spectrum of diseases.

In this paper, we introduce a novel method that identifies the Pareto front without retraining the models to be merged. Our algorithm utilizes a quadratic approximation of the evaluation metric. Furthermore, we enhance it with a Bayesian adaptive sampling method and a nested merging scheme, which further brings down the computational cost. We validate our method across a diverse set of tasks, spanning from vision to natural language processing, and demonstrate its applicability to a variety of architectures, including ResNets [20], ViT [17], and large language models [6; 47; 55; 28]. Our results confirm that this novel approach supports the seamless integration of diverse model capabilities and aligns more closely with various real-world preference by providing a set of optimal fronts across the tasks.

Contributions The main contributions of this paper are:

- **C1** Design the MAP algorithm that utilizes quadratic surrogate models to approximate the evaluation metric functions, which amortize the computation of Pareto fronts;
- **C2** To our best knowledge, this paper is the first work, that estimates the Pareto front for task-vector-based model-merging methods with low compute.

Related work This paper is related to many existing works on multi-objective optimization, Pareto optimality, task arithmetic, federated/private learning, Bayesian methods, etc. We kindly refer the readers to Appendix A due to the page limit.

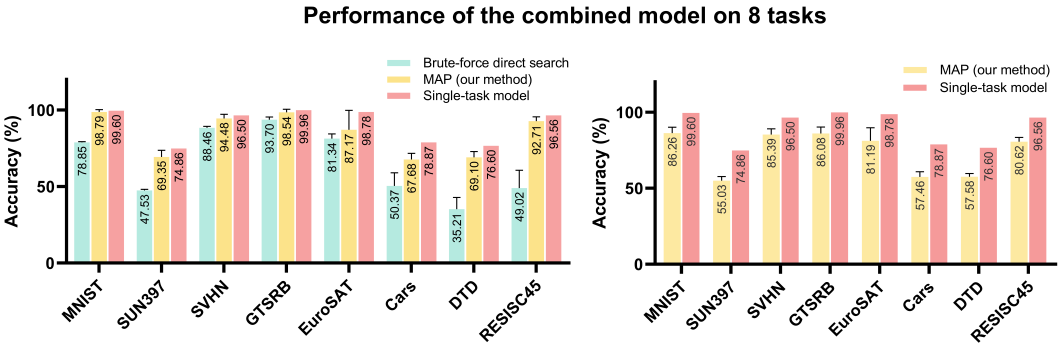


Figure 2: **(a)** Using the results of inference runs of merged ViT-B-32 models [45] obtained from 250 combinations of scaling coefficients, our method outperforms the direct search method in finding a set of diverse solutions for eight tasks based on the same computation expense. For both methods, we maximize a specific task while requiring all tasks to have an evaluation metric above a certain threshold (40%). The bar plot for each task shows the corresponding maximized accuracy. **(b)** We increase the requiring bounds to (65%) of the single-task model. With the same computational budget, the brute-force direct search method cannot find a feasible solution.

In Figure 2, we compare our method to direct search. We want to emphasize that, in our assumed setting, evaluation models are computationally expensive. Thus, we can only query the evaluation pipeline a limited number of times. We choose to compare to direct brute force search rather than Evolutionary algorithms like NSGA-III, etc. Evolutionary algorithms either need to query the evaluation metrics more than our computational budget or select the Pareto solutions from the pre-evaluated models. Suppose we already have a set of solutions and know their evaluation metrics, and we use evolutionary algorithms to pick the Pareto solutions among them. In that case, the set we find with the algorithms might not include all possible Pareto solution. Instead, it will likely be just a part of what we would find if we compared every possible pair of solutions directly against each other (known as brute force search). Since brute force search compares all possible solutions and finds all the Pareto optimal solutions, it's considered a complete method. Therefore, when evaluating the effectiveness of our algorithm, it is better to compare its results to those obtained from brute force search in the case of a limited set of already-known solutions. Our code is available at `https://github.com/luli-git/MAP`.

2 Background

2.1 Model merging

Model merging aims to find a way to combine multiple models with the same architecture $\{\boldsymbol{\theta}_{ft}^n\}_{n\in[N]}$ with their initialization $\boldsymbol{\theta}_{\text{pre}}$ in a way such that the merged model parameterized by $\boldsymbol{\theta}_m = \boldsymbol{\theta}_{\text{pre}} + \sum_{n=1}^{N} c_n \boldsymbol{\theta}_{ft}^n$ perform reasonably well on all N tasks [61].

A recent work by Ilharco et al. [24] introduced *task arithmetic* as a simple and effective way for performing model merging. The task vector for task n is defined as $\mathbf{v}_n = \boldsymbol{\theta}_{ft}^n - \boldsymbol{\theta}_{\text{pre}}$, which is the element-wise difference between the pre-trained weights and the fine-tuned weights for task N. To perform model merging with task vectors, we can compute $\boldsymbol{\theta}_{\text{pre}} + \sum_{n=1}^{N} c_n \mathbf{v}_n$, where c_i are scaling factors that have shown to be essential to the performance of the merged model [66; 67].

Denoting the metric of task n as M_n, most of the existing approaches for model merging aim to improve an equal weight average metric $\frac{1}{N}\sum_{n=1}^{N} M_n$. This target implies that user of the algorithm has equal preferences between tasks. However, in real-world applications, users might have biased preferences for the importance of tasks, necessitating trade-offs. In such cases, the goal of model merging is no longer the equal weight average metric. Instead, we need to answer a broader question:

Given any preferences over the set of tasks, what is the best way to merge the models?

2.2 Pareto fronts

Pareto dominance Let X be a set representing the solution space, where each element $x \in X$ is a possible solution to the multi-objective optimization problem. Let there be n objectives. Define an evaluation loss function $f_i : X \to \mathbb{R}$, where $i \in \{1, 2, \cdots, n\}$. Given two solutions $x, y \in X$, we define that x *Pareto dominates* y, denoted by $x \succ_P y$, if and only if:

$$\forall i \in \{1, 2, \ldots, n\}, f_i(x) \leq f_i(y) \text{ and } \exists j \in \{1, 2, \ldots, n\}, f_j(x) < f_j(y)$$

Pareto optimal solutions The Pareto front is the set of solutions in the solution space X that are not Pareto dominated by any other solutions in X.

$$\text{PF} = \{x \in X \mid \nexists y \in X \text{s.t.} y \succ_P x\} \tag{1}$$

Pareto optimal solutions have been studied in multi-task (multi-objective) learning (MTL) [49; 35]. However, in most of the studies, approximating the Pareto front is computationally expensive and data inefficient. We introduce our method, MAP, a computationally efficient method to find the Pareto front for model merging.

3 Related Work

Federated/Private Learning Because the model merging is a post-training approach, it can naturally be used in federated learning, which protects both the model privacy and data privacy. To be specific, for data privacy, the only operation that requires the data is line 6 in Algorithm 1, where each dataset can be kept at one site without sharing with other sites: the i-th site holding its own data will take in the model $\boldsymbol{\theta}(\mathbf{c})$ and only share $M_n(\boldsymbol{\theta}(\mathbf{c})) \in \mathbb{R}$; for model privacy, the only operator that requires an aggregation is line 5 in Algorithm 1, where each model can be kept at one site and the aggregation is privacy-preserving using a safe center site, or the secure multi-party computation without a center site.

Pareto fronts for Multi-task Learning Recent advancements in multi-task learning (MTL) have seen diverse approaches to Pareto Front learning in machine learning.

Lin et al. [35] developed a constrained optimization approach to find multiple Pareto optimal solutions representing different trade-offs among tasks in multi-task learning and solving decomposed subproblems in parallel using gradient-based optimization. It requires separate training runs for each solution. Navon et al. [39] and Lin et al. [34] employed hypernetworks for continuous Pareto Front approximation, generating target network weights based on preferred trade-offs. However, the size of these hypernetworks can become prohibitively large and need to be properly trained as well. Ruchte and Grabocka [48] introduced a memory-efficient method by augmenting network inputs with desired trade-offs, although this may obscure the network's conditioning due to nonlinear dynamics, and it is also based on the gradient updating method.

For more detailed discussions on other related work, please find Appendix A.

4 Methods

4.1 Quadratic approximation of evaluation metric

Given the task vectors $\{\mathbf{v}_n\}_{n=1,\ldots,N}$ and the initialization $\boldsymbol{\theta}_{\text{pre}} \in \mathbb{R}^d$, we denote the merged model parameters as $\boldsymbol{\theta}_m(\mathbf{c}) = \boldsymbol{\theta}_{\text{pre}} + \mathbf{V}\mathbf{c} = \boldsymbol{\theta}_{\text{pre}} + \sum_{n=1}^{N} c_n \mathbf{v}_n$, where $\mathbf{V} = \text{concat}(\mathbf{v}_1, ..., \mathbf{v}_N) \in \mathbb{R}^{d \times N}$ is the task matrix and $\mathbf{c} = \text{concat}(c_1, ..., c_N) \in \mathbb{R}^N$ is the scaling coefficients for the task vectors.

We aim to optimize the evaluation metric for all tasks, denoted by M_n, via the multi-objective optimization (MOOP)[2]:

$$\min_{c_1, \cdots, c_N} M_1(\mathbf{c}), \cdots, M_N(\mathbf{c}) \tag{2}$$

[2]The evaluation metric M can be differentiable (e.g. the mean square loss or the cross-entropy/perplexity) or not necessarily (e.g. the classification accuracy, F1 score, BLEU or Rouge)

This problem has N variables and N objectives, and we seek the Pareto optimal solutions.

To do so effectively and efficiently, we use the second-order Taylor expansion to approximate M_n:

$$\begin{aligned} M_n(\mathbf{c}) \equiv M_n(\boldsymbol{\theta}_{\text{merge}}(\mathbf{c})) &= M_n(\boldsymbol{\theta}_{\text{pre}}) + \nabla M_n(\boldsymbol{\theta}_{\text{pre}})^\top (\boldsymbol{\theta}_m(\mathbf{c}) - \boldsymbol{\theta}_{\text{pre}}) \\ &+ \frac{1}{2}(\boldsymbol{\theta}_m(\mathbf{c}) - \boldsymbol{\theta}_{\text{pre}})^\top \mathbf{H}_n(\boldsymbol{\theta}_{\text{pre}})(\boldsymbol{\theta}_m(\mathbf{c}) - \boldsymbol{\theta}_{\text{pre}}) + R_n(\boldsymbol{\theta}_m(\mathbf{c}) - \boldsymbol{\theta}_{\text{pre}}) \\ &\approx M_n(\boldsymbol{\theta}_{\text{pre}}) + \nabla M_n(\boldsymbol{\theta}_{\text{pre}})^\top \mathbf{V}\mathbf{c} + \frac{1}{2}(\mathbf{V}\mathbf{c})^\top \mathbf{H}_n(\boldsymbol{\theta}_{\text{pre}})\mathbf{V}\mathbf{c} \end{aligned}$$

where $\mathbf{H}_n(\boldsymbol{\theta}_{\text{pre}}) = \nabla^2 M_n(\boldsymbol{\theta}_{\text{pre}}) \in \mathbb{R}^{d\times d}$ is the Hessian matrix and $R_n(\boldsymbol{\theta}_m(\mathbf{c}) - \boldsymbol{\theta}_{\text{pre}}) = R_n(\mathbf{V}\mathbf{c})$ is the third-order remainder, which is negligible when $||\mathbf{V}\mathbf{c}||^3 = ||\boldsymbol{\theta}_m(\mathbf{c}) - \boldsymbol{\theta}_{\text{pre}}||^3$ is small. Note that the second-order Taylor expansion is widely used and provides a close approximation (see Figure 2 (a) and (b) and more discussion in Appendix A), as neural networks tend to converge close to their initialization, especially for fine-tuned models: it directly motivates Newton's method, and supports the analysis of convergence [37; 7] and the scaling laws for large models [30; 22].

Leveraging this quadratic approximation, we can define surrogate models for each task N, $\tilde{M}_n(\mathbf{c}; \mathbf{A}_n, \mathbf{b}_n, e_n) \equiv e_n + \mathbf{b}_n^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_n \mathbf{c}$ and optimize the proxy problem of (2) via

$$\min_{c_1,\cdots,c_N} \tilde{M}_1(\mathbf{c}), \cdots, \tilde{M}_N(\mathbf{c}) \tag{3}$$

where

$$\mathbf{A}_n = \mathbf{V}^\top \mathbf{H}_n(\boldsymbol{\theta}_{\text{pre}})\mathbf{V} \in \mathbb{R}^{N\times N}, \mathbf{b}_n = \mathbf{V}^\top \nabla M_n(\boldsymbol{\theta}_{\text{pre}}) \in \mathbb{R}^N, e_n = M_n(\boldsymbol{\theta}_{\text{pre}}) + R_n \tag{4}$$

Importantly, (3) is parameterized in contrast to (2), with $\frac{(N+1)(N+2)}{2}$ unknown coefficients in (e_n, $\mathbf{b}_n$, $\mathbf{A}_n$)[3] Thus, we can further leverage existing methods to learn the coefficients by minimizing the empirical risk over multiple $\mathbf{c}$. For instance,

$$\mathbf{A}_n^*, \mathbf{b}_n^*, e_n^* = \arg\min_{\mathbf{A}_n,\mathbf{b}_n,e_n} \sum_{\mathbf{c}\in\Omega} |M_n(\boldsymbol{\theta}_{\text{merge}}(\mathbf{c})) - \tilde{M}_n(\mathbf{c}; \mathbf{A}_n, \mathbf{b}_n, e_n)|^2 \tag{5}$$

where $\Omega = \{\mathbf{c}^{(1)}, ..., \mathbf{c}^{(K)}\}$ is the set of $\mathbf{c}$ and $M_n(\boldsymbol{\theta}_{\text{merge}}(\mathbf{c}))$ is the corresponding evaluation metric.

If the evaluation metric spans the whole real-number axis $\mathbb{R}$, we use the vanilla form of the surrogate model: $\tilde{M}_n(\mathbf{c}; \mathbf{A}_n, \mathbf{b}_n, e_n) \equiv e_n + \mathbf{b}_n^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_n \mathbf{c}$. When the evaluation metric is restricted to a specific range $[l, u]$, e.g. accuracy is restricted to $[0, 1]$, we would warp the quadratic part with a sigmoid function, i.e. $\tilde{M}_n(\mathbf{c}; \mathbf{A}_n, \mathbf{b}_n, e_n) \equiv (u - l)\sigma(e_n + \mathbf{b}_n^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_n \mathbf{c}) + l$. Similarly, if the evaluation metric is restricted to $[l, +\infty)$, a softplus function as a wrapper would be beneficial: $\tilde{M}_n(\mathbf{c}; \mathbf{A}_n, \mathbf{b}_n, e_n) \equiv \text{softplus}(e_n + \mathbf{b}_n^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_n \mathbf{c}) + l$. Please note that u and l mentioned here are not learnable parameters. They are specific to the feasible area of the metric.

4.2 Model merging with amortized Pareto fronts

In this section, we introduce our generalized algorithm for estimating the amortized Pareto fronts. As mentioned in Section 4.1, we approximate the evaluation metric $M_n(\cdot)$ by a surrogate quadratic model $\tilde{M}_n(\cdot)$. We then utilize $\tilde{M}_n(\cdot)$ to compute the amortized Pareto fronts. Please see the detailed experiments in Section 5 and the algorithm details in Algorithm 1. Different from other Pareto Multi-task learning algorithms, our algorithm

4.3 Additional Methods to bring down the computation cost

To further reduce the required computation of MAP, we propose (1) a Bayesian adaptive sampling algorithm and (2) a nested merging scheme with multiple stages. The details can be found in Appendix G.2 and G.3.

[3] 1 coefficient for e_n, N coefficients in $\mathbf{b}_n$ and $N(N+1)/2$ coefficients in $\mathbf{A}_n$ due to its symmetry.

Algorithm 1 MAP

Prepare models $\{\boldsymbol{\theta}_{ft}^n\}$ and compute task vectors $\{\mathbf{v}_n = \boldsymbol{\theta}_{ft}^n - \boldsymbol{\theta}_{\text{pre}}\}$.
for $n \in [N]$ **do**
  Sample K vectors of $\mathbf{c} \in \mathbb{R}^N$. Denote the set as Ω.
  for $\mathbf{c} = [c_1, ..., c_N] \in \Omega$ **do**
    Compute $\boldsymbol{\theta}(\mathbf{c}) = \boldsymbol{\theta}_{\text{pre}} + c_1\mathbf{v}_1 + \ldots c_N\mathbf{v}_N$.
    Derive the evaluation metric $M_n(\boldsymbol{\theta}(\mathbf{c}))$.
  Fit the quadratic approximation surrogate model $\tilde{M}_n$ by learning $\mathbf{A}_n^*, \mathbf{b}_n^*, e_n^*$ in (5).
Apply MOOP algorithm (e.g. NSGA-III) to $\{\tilde{M}_n\}$ and get the Pareto front

5 Experiments

In this section, we present our empirical findings across a wide range of tasks including zero-shot classification (CLIP-based) and LLMs (Llama3-based).

5.1 Experiment setup

Table 1: Experiment setup in terms of tasks and models

Task type	Metric	# of total tasks	Model type
Zero-shot Classification (normal)	Accuracy	8	ViT-B/32 (CLIP)
Zero-shot Classification (medical)	Accuracy	2	ViT-B/32 (CLIP)
Language Generation	Loss/Perplexity	4	Llama3-8B
Image Classification	Accuracy	3	ResNet-18

Datasets and models We study multi-task model merging on eight normal zero-shot image classification datasets following [24]: SUN397 [64], Cars [31], GTSRB [51], MNIST [33], EuroSAT [21], SHVN [40], DTD [11], RESISC45 [9]. We use the ViT-B/32 architecture in CLIP [45] as the pre-trained model for the experiments on vision tasks in the main text. We show the result of them in the main pages. For the rest of the experiments, due to the page limits, we show them in the appendix. The results of the datasets and tasks we experimented on are as follows: a zero-shot medical chest X-ray image classification task to show our model merging scheme also works in a real-world application domain [58]; 4 fine-tuned Llama3 models in different languages: French, Arabic, Chinese, and Japanese [4]; 3 vision tasks on the ResNet18 architecture: CIFAR-10 [32], Flowers-102 [41], and GTSRB [51].

5.2 Baseline and metrics

Baselines Our baseline method for obtaining Pareto fronts is the brute-force direct search. When the number of tasks is low ($N < 4$), we can sample enough grid points of $\mathbf{c}$ and query the corresponding evaluation metrics $\{M_n(\theta(\mathbf{c}))\}_{n=1}^N$. We can then directly use the resulting evaluation metrics to find the Pareto front by direct comparisons of each task performance of the merged model by $c_i \in \mathbf{c}$. When the number of tasks is low, we can regard the resulting Pareto front as the ground truth Pareto front. However, when the number of tasks grows, the required number of $(\mathbf{c}, \{M_n(\theta(\mathbf{c}))\}_{n=1}^N)$ pairs grows exponentially, which is way larger than the points we evaluated. Thus, when the number of tasks is high ($N \geq 4$), the results from the brute-force direct search can no longer be considered as ground truth.

In addition to the brute-force method, we compare with other model merging methods including SLERP [50], TIES-merging [66], Task Arithmetic [24] with a single scalar, and Task Arithmetic with preferences as scalars, DARE combined with Task Arithmetic [68], and DARE combined with TIES-merging.

[4] All the fine-tuned language Llama3 models can be found on Huggingface. The IDs of the models are: French: jpacifico/French-Alpaca-Llama3-8B-Instruct-v1.0; Arabic: MohamedRashad/Arabic-Orpo-Llama-3-8B-Instruct; Chinese: shenzhi-wang/Llama3-8B-Chinese-Chat; Japanese: haqishen/Llama-3-8B-Japanese-Instruct.

Win rate We used the win rate to measure how often the Pareto front found by MAP outperforms the Pareto front found by the baseline in terms of multiple objectives. Let PF_{MAP} and $PF_{baseline}$ represent the set of solutions in the Pareto fronts obtained from the MAP and the baseline methods, respectively. Each solution in these sets is a vector in $\mathbb{R}^N$, where N is the number of objectives or tasks. We sampled $K = \min(100, |PF_{baseline}|)$ points from each of the two Pareto fronts, denoted as $\mathbf{c}_k^{MAP}$ and $\mathbf{c}_k^{baseline}$, $k = 1, \ldots, K$. Then, we compared $M_n(\theta(\mathbf{c}_k^{MAP}))$ and $M_n(\theta(\mathbf{c}_k^{baseline}))$ pairwise for $k = 1, \ldots, K$ and $n = 1, \ldots, N$, resulting in $K \times K \times N$ comparisons. The ratio of instances where $M_n(\theta(\mathbf{c}_k^{MAP})) > M_n(\theta(\mathbf{c}_k^{baseline}))$ is computed as the win rate of our amortized Pareto front PF_{MAP}.

5.3 Win rate of MAP over the direct search method

Table 2 shows the results for the win rate. In conclusion, when the number of tasks ($N < 4$), we can use only query 30 and 50 scaling coefficients to get similar performance with the ground truth Pareto front. For the high number of task scenarios ($N \geq 4$), MAP performs much better than the brute-force (not ground truth) Pareto solutions.

Table 2: Win rate of the amortized Pareto front and the Pareto front by brute-force direct search. The number of points is the number of points each algorithm used to find the Pareto front. Note that the number of evaluations by each method is the number of points multiplied by the number of tasks. The number of points per dimension is the number of points raised to the power of $\frac{1}{N}$, which measures the sparsity of points the direct search method uses per dimension to approximate the Pareto front.

# of tasks	# of pts direct search	# of pts per dim	# of pts (MAP)	Win rate (MAP)	R^2 (MAP)
2	200	14.14	30	49.81% (±0.30)	0.953 (±0.018)
3	300	6.69	50	46.90% (±0.71)	0.980 (±0.003)
4	300	4.16	60	50.67% (±2.44)	0.984 (±0.004)
5	500	3.47	85	53.00% (±1.88)	0.941 (±0.019)
6	500	2.82	100	60.71% (±1.34)	0.941 (±0.030)
7	1000	2.68	140	63.42% (±1.91)	0.891 (±0.024)
8	1000	2.37	250	65.58% (±0.94)	0.868 (±0.028)

5.4 MAP offers a well-distributed set of Pareto optimal solutions compared with other baseline methods

We compared the performance of MAP with TIES-merging [65], TIES-merging with DARE [68], Task Arithmetic with DARE, Task Arithmetic with normalized preference as scaling coefficients [25], SLERP [50]. For dimension 2, we can directly visualize the results, as shown in Figure 3. In addition, we show that MAP is a plug-in that can be directly combined with other task vector-based model merging methods where the users can control the preferences using some scalars. In Figure 3, we show the Pareto front found by MAP-DARE.

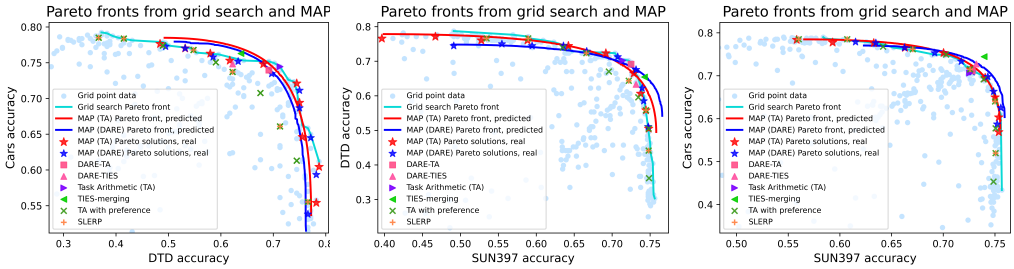


Figure 3: The Pareto fronts obtained using MAP with Task Arithmetic, MAP with Task Arithmetic and DARE. We sampled 10 Pareto solutions from the predicted front by MAP and evaluated them to obtain the real values. We plotted the results obtained using TIES-merging, Task Arithmetic (TA) with a single scalar for all tasks, Task Arithmetic with preferences as scalars, TA combined with DARE (DARE-TA), TIES-merging combined with DARE (DARE-TIES), and SLERP.

We can see that none of the baseline methods can directly dominate all Pareto solutions found by MAP, and most of them lie either within the Pareto front found by MAP. In addition, we sampled 10 points from the predicted Pareto front by MAP and evaluated them (MAP Pareto solutions, real)

to confirm that they indeed lie close to the predicted Pareto front. This evidence further confirms the usefulness of MAP as a general strategy for finding a set of diverse and well-distributed Pareto solutions that none of the existing model merging methods can substitute.

For dimensions higher than 2, we use two metrics to compare the Pareto front found by MAP with other baseline methods. We sample 20 preference vectors and normalize them. Then, we pick the solution by MAP that maximizes the preference-weighted sum of accuracies and compare it with those by baseline methods. The results are shown in Table 3. We also compute the win rate, which is the proportion of preferences where the preference-weighted sum of accuracies of MAP is higher than that of all the baseline methods. We can observe that even in higher dimensions, when the user has certain preferences, MAP is still useful and can better accommodate user preferences. Please note that when using MAP, we do not require the user to pre-specify their preference vector. This preference vector based evaluation is only for evaluation purposes.

Table 3: We compared MAP with a set of baseline methods by sampling a set of 20 normalized preference vectors and computing the preference-weighted sum of accuracies and win rate. The win rate is defined as the proportion of the 20 preference vectors where the preference-weighted sum of accuracies of MAP is higher than those of the baseline methods. $\uparrow$ indicates higher is better. The number after $\pm$ is the standard deviation. Please refer to Table 2 for the number of scaling coefficient vectors used for different number of tasks.

Preference weighted sum of accuracies ($\uparrow$)							
# tasks	2	3	4	5	6	7	8
Single task models	75.84±1.76	77.03±1.84	82.43±4.40	87.69±4.50	88.52±4.02	89.26±3.58	90.62±2.52
MTL	73.63±0.30	75.13±1.00	80.10±2.79	84.93±3.58	86.78±2.94	87.40±2.56	89.11±2.36
MAP	70.70±0.21	69.05±1.41	72.84±1.05	77.31±0.83	74.26±0.52	73.40±-0.14	72.96±0.73
Model soups	67.79 ± 1.46	64.25 ± 2.15	66.04 ± 3.22	67.01 ± 3.42	63.11 ± 1.99	63.35 ± 2.17	64.36 ± 2.77
TIES-merging	69.30±0.33	67.60±0.58	71.79±2.93	76.49±3.10	73.74±2.96	72.54±2.87	72.24±1.91
DARE-TIES	67.62±1.65	66.49±2.34	71.39±4.45	74.55±4.55	73.34±4.10	71.43±3.84	71.89±2.86
Task Arithmetic	70.73±1.84	61.15±2.33	52.69±4.23	61.58±4.62	51.37±3.84	39.79±3.97	60.77±2.84
TA with preference as weights	69.22 ± 1.4	66.88 ± 2.37	68.73 ± 5.48	71.92 ± 5.5	68.13 ± 4.69	68.14 ± 4.2	68.17 ± 2.89
DARE-TA	70.61±0.22	64.18±1.24	58.04±8.19	65.39±7.03	56.76±7.01	46.75±5.73	64.51±3.81
Win rate of MAP over baseline methods ($\uparrow$)							
# tasks	2	3	4	5	6	7	8
MAP vs TIES-merging	75±9.80	60±10.93	70±10.49	75±9.39	65±10.85	65±11.66	70±10.25
MAP vs DARE-TIES	90±6.81	80±9.11	75±9.87	90±6.70	65±10.59	60±11.38	75±9.70
MAP vs TA	65±10.91	100±0	100±0	100±0	100±0	100±0	100±0
MAP vs DARE-TA	65±10.64	100±0	100±0	100±0	100±0	100±0	100±0
MAP vs TA with preference	85.0 ± 0.08	85.0 ± 0.08	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
MAP vs Model soup	90.0 ± 0.07	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0

6 Conclusion and Limitations

Our method (MAP) can serve as an out-of-the-box plug-in that is very easy to combine directly with other task-vector based model-merging methods, such as tasks arithmetic [24] and DARE [68], etc. From Section 4 and Section 5, we have shown that our methods Algorithm 1, algorithm 2, and Algorithm 3 can efficiently reduce the computational cost from $O(N2^N)$ to $O(N \log_2 N)$. However, we would like to point out some limitations of our method. First, during the design of Algorithm 1, we do not handle the situation of a non-convex Pareto front. In the case of two tasks, it is likely that the Pareto fronts with proper trade-offs would have a convex Pareto front. However, if one deals with more tasks, the ground truth Pareto fronts are likely non-convex, and Algorithm 1 ends up approximating a convex hull. A quick but naive solution would be to implement Algorithm 2 in this case, so that the Pareto fronts between two tasks would be more likely to be convex. Second, we do not have a detector algorithm to detect if two tasks have proper trade-offs. Approximating a Pareto "front" with only a single Pareto solution likely results in very unstable and poorly-performed results. In this case, practitioners should be informed.

References

[1] Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International conference on machine learning*, pages 242–252. PMLR, 2019.

[3] Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point. In *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*, pages 87–102, 2009.

[4] Oscar Brito Augusto, Fouad Bennis, Stephane Caro, et al. Multiobjective optimization involving quadratic functions. *Journal of optimization*, 2014, 2014.

[5] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Zhiqi Bu, Xinwei Zhang, Mingyi Hong, Sheng Zha, and George Karypis. Pre-training differentially private models with limited public data. *arXiv preprint arXiv:2402.18752*, 2024.

[8] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[9] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[10] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

[11] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[12] Carlos A Coello Coello and Nareli Cruz Cortés. Solving multiobjective optimization problems using an artificial immune system. *Genetic programming and evolvable machines*, 6:163–190, 2005.

[13] Haotian Cui, Chloe Wang, Hassaan Maan, Kuan Pang, Fengning Luo, Nan Duan, and Bo Wang. scgpt: toward building a foundation model for single-cell multi-omics using generative ai. *Nature Methods*, pages 1–11, 2024.

[14] Nico Daheim, Thomas Möllenhoff, Edoardo Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In *The Twelfth International Conference on Learning Representations*, 2024.

[15] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013.

[16] Kalyanmoy Deb, Manikanth Mohan, and Shikhar Mishra. Towards a quick computation of well-spread pareto-optimal solutions. In *Evolutionary Multi-Criterion Optimization: Second International Conference, EMO 2003, Faro, Portugal, April 8–11, 2003. Proceedings 2*, pages 222–236. Springer, 2003.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[18] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR, 2019.

[19] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

[22] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[23] Evan J Hughes. Evolutionary many-objective optimisation: many once or one many? In *2005 IEEE congress on evolutionary computation*, volume 1, pages 222–227. IEEE, 2005.

[24] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[25] Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[26] Moksh Jain, Emmanuel Bengio, Alex Hernández-García, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with gflownets. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9786–9801. PMLR, 2022.

[27] Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *arXiv preprint arXiv: 2310.11564*, 2023.

[28] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *arXiv preprint arXiv: 2401.04088*, 2024.

[29] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[30] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[31] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

[32] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[33] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[34] Xi Lin, Zhiyuan Yang, Qingfu Zhang, and Sam Kwong. Controllable pareto multi-task learning. *arXiv preprint arXiv: 2010.06313*, 2020.

[35] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *Advances in neural information processing systems*, 32, 2019.

[36] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

[37] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

[38] Henry B. Moss, David S. Leslie, Daniel Beck, Javier Gonzalez, and Paul Rayson. Boss: Bayesian optimization over string spaces. In *Advances in Neural Information Processing Systems*, 2020.

[39] Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. *arXiv preprint arXiv: 2010.04104*, 2020.

[40] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7. Granada, Spain, 2011.

[41] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008.

[42] National Institutes of Health et al. Nih clinical center provides one of the largest publicly available chest x-ray datasets to scientific community, 2017.

[43] Guillermo Ortiz-Jiménez, Alessandro Favero, and P. Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Neural Information Processing Systems*, 2023.

[44] E. O. Pyzer-Knapp. Bayesian optimization for accelerated drug discovery. *IBM Journal of Research and Development*, 62(6):2:1–2:7, 2018.

[45] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[46] Alexandre Ramé, Guillaume Couairon, Mustafa Shukor, Corentin Dancette, Jean-Baptiste Gaya, L. Soulier, and M. Cord. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *Neural Information Processing Systems*, 2023.

[47] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *arXiv preprint arXiv: 2308.12950*, 2023.

[48] Michael Ruchte and Josif Grabocka. Scalable pareto front approximation for deep multi-objective learning. *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1306–1311, 2021.

[49] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[50] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, page 245–254, New York, NY, USA, 1985. Association for Computing Machinery.

[51] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pages 1453–1460. IEEE, 2011.

[52] Hui Su, Zhi Tian, Xiaoyu Shen, and Xunliang Cai. Unraveling the mystery of scaling laws: Part i. *arXiv preprint arXiv:2403.06563*, 2024.

[53] Kei Terayama, Masato Sumita, Ryo Tamura, and Koji Tsuda. Black-box optimization for automated discovery. *Accounts of Chemical Research*, 54(6):1334–1346, 2021.

[54] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

[55] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv: 2307.09288*, 2023.

[56] David Allen Van Veldhuizen. *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. Air Force Institute of Technology, 1999.

[57] Haoxiang Wang, Yong Lin, Wei Xiong, Rui Yang, Shizhe Diao, Shuang Qiu, Han Zhao, and Tong Zhang. Arithmetic control of llms for diverse user preferences: Directional preference alignment with multi-objective rewards. *arXiv preprint arXiv: 2402.18571*, 2024.

[58] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[59] James T. Wilson, Riccardo Moriconi, Frank Hutter, and Marc Peter Deisenroth. The reparameterization trick for acquisition functions, 2017.

[60] Michael Wornow, Rahul Thapa, Ethan Steinberg, Jason Fries, and Nigam Shah. Ehrshot: An ehr benchmark for few-shot evaluation of foundation models. 2023.

[61] Mitchell Wortsman, Gabriel Ilharco, S. Gadre, R. Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Y. Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *International Conference on Machine Learning*, 2022.

[62] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7959–7971, 2022.

[63] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*, pages 1–5. IEEE, 2023.

[64] Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119:3–22, 2016.

[65] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Neural Information Processing Systems*, 2023.

[66] 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.

[67] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*, 2023.

[68] 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. *arXiv preprint arXiv:2311.03099*, 2023.

[69] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. *IEEE International Conference on Computer Vision*, 2023.

[70] Yifan Zhong, Chengdong Ma, Xiaoyuan Zhang, Ziran Yang, Haojun Chen, Qingfu Zhang, Siyuan Qi, and Yaodong Yang. Panacea: Pareto alignment via preference adaptation for llms. *arXiv preprint arXiv: 2402.02030*, 2024.

[71] Zhanhui Zhou, Jie Liu, Chao Yang, Jing Shao, Yu Liu, Xiangyu Yue, Wanli Ouyang, and Yu Qiao. Beyond one-preference-fits-all alignment: Multi-objective direct preference optimization. *arXiv preprint arXiv: 2310.03708*, 2023.

Appendix

A Related Work **13**

B Performance of the quadratic approximation **15**

C Nested-Merging MAP **15**

C.1 Time Complexity of Nested-Merging MAP 15

C.2 Pareto Fronts Obtained Using Nested Merging 16

C.3 Nested merging with rule-based preferences 17

D Bayesian MAP **18**

E Proof: Negligibility of the Remainder in Multivariate Taylor Series **19**

F Additional Experiment Results **20**

F.1 Zero-shot Medical Image Classification 20

F.2 Merging language model 21

F.3 Additional experiment results on ResNet 21

G More details about algorithms **22**

G.1 Algorithm 1 22

G.2 Nested merging scheme 22

G.3 Bayesian adaptive sampling 22

G.4 Algorithm 2 24

G.5 Algorithm 3 24

H Potential QA **25**

I NeurIPS 2024 reviewer comments **27**

A Related Work

Multi-objective optimization Multi-objective optimization (MOOP) aims at identifying a variety of Pareto solutions, each offering different trade-offs, instead of just a single solution. In fact, the multi-task problem is approached from a MOOP view point [49; 35], which leverages a rich literature of algorithms including MGDA, IMTL, GradNorm, RLW, PCGrad, scalarization, and so on. We note that these training algorithms iteratively optimize over the $\mathbb{R}^d$ model parameters and can be computationally costly (in order to instantiate and modify the per-task gradient). In contrast, our algorithm is a post-training MOOP over the $\mathbb{R}^N$ scaling coefficients, which is significantly low-compute.

Many methods have been proposed to solve MOOP and derive the Pareto optimal solutions. One may leverage the scalarization technique to transform a multi-objective problem into many single-objective ones, aiming to find one solution for each problem and approximate the Pareto front. Additionally, one can solve the Karush-Kuhn-Tucker conditions, which are easy to work with, given that the parameters have a closed-form solution under our quadratic approximation.

To select an appropriate MOOP algorithm, we need to take into account the computational costs and the quality of the Pareto front. For instance, many indicator-based and selection-based methods (e.g.

the hypervolume indicator [3]) have the time complexity that grows super-polynomially with the number of objectives. Indeed, for $N \geq 3$, MOOP is termed as many-objective optimization, which is significantly challenging for well-established algorithms (e.g. NSGA-II and SPEA2) and requires more advanced algorithms such as ϵ-MOEA [16], MSOPS [23], SMS-EMOA [5], and NSGA-III [15], or the KKT approach [4].

Task arithmetic Task arithmetic, a method of model merging that has drawn increasing attention, applies a weighted average of models (controlled by the scaling coefficients) to obtain high performance on multiple tasks simultaneously. Several existing works have studied ways to select the scaling coefficients [24; 66; 67]: [24] simply uses equal scaling, similar to the Model Soup approach, which is heuristic and suboptimal, plus the applicability is limited to certain tasks and small scales; [67] aims to learn the optimal scaling weights by using the Shannon entropy as a surrogate objective function to the cross entropy loss. However, it requires the use of unlabeled test data from all tasks. In most real-world applications, data cannot be directly shared, which is what makes model merging appealing initially [36]. In addition, such an approach only works for classification tasks due to the use of cross-entropy loss.

[62] proposed 'WiSE-FIT' to perform robust fine-tuning by computing the weighted average of the pre-trained model parameters with the parameters of the models fine-tuned on different tasks, with different weights determining different trade-offs between pre-training and fine-tuning task performance. Task Arithmetic [24] adds the weighted sum of the differences between the fine-tuned model and the pre-trained model to the weights of the pre-trained model. Again, the weights are used to distinguish the importance of various models. In addition, [43] explores how fine-tuning models in their tangent space enhances weight disentanglement, leading to improved performance in task arithmetic for editing pre-trained vision-language models.

Different from task arithmetic based model merging approaches, Ainsworth et al. [1] show that even if the models are not finetuned from the same pretrained model, practitioners could still merge the models by permutations of the rows and columns of the weight matrices. Similarly, Jin et al. [29] also introduce a dataless knowledge fusion method which is not based on task vectors that merges fine-tuned language models in parameter space without requiring access to their original training data, outperforming existing baselines and offering an efficient alternative to multi-task learning. Daheim et al. [14] proposes an uncertainty-based gradient matching method for model merging that improves performance and robustness by reducing gradient mismatches between models trained on different datasets.

Second-order Taylor expansion In deep learning, the second-order Taylor expansion and thus the quadratic approximation on the evaluation metric is an important technique, that characterizes the metric landscape. For example, the Newton-Ralphson method is derived from it: $\mathbf{w}_t - \mathbf{w}_{t+1} = \mathbf{H}^{-1}\mathbf{G} = \text{argmin}_{\mathbf{v}} M(\mathbf{w}_t - \mathbf{v})$ given that $M(\mathbf{w}_t - \mathbf{v}) = M(\mathbf{w}_t) - \mathbf{v}\mathbf{G} + \frac{1}{2}\mathbf{v}^\top \mathbf{H}\mathbf{v}$. The quadratic approximation is also combined with Lipschitz smoothness (L) in the classic convergence analysis of SGD [8; 19], through $M(\mathbf{w}_t - \eta\mathbf{G}) \leq M(\mathbf{w}_t) - \eta||\mathbf{G}||^2 + \frac{L\eta^2}{2}||\mathbf{G}||^2$. Interestingly, although the approximation is only accurate in the local neighborhood, it can be used to indicate the global convergence over the iterations. One reason is that state-of-the-art neural networks are usually over-parameterized and undergo lazy training, meaning that the converging parameters are close to the initialization [10; 18; 2]. Thus, the local approximation informs the global convergence behavior. In particular, this approximation has played important roles in the scaling laws of neural networks (e.g. Equation 3.3 in [52]) that predict the performance and help select hyperparameters before training actually takes place.

Model-merging Applications in LLM In the realm of model merging applications for language model preferences, recent research has made significant progress.Rame et al. [46] and Jang et al. [27] introduced "rewarded soups" and "personalized soups", which utilize model soup to interpolate weights of networks fine-tuned on diverse rewards to achieve Pareto-optimal generalization across preference spaces and address the challenge of aligning large language models with individual perspectives. Zhong et al. [70] developed "Panacea", which reframes alignment as a multi-dimensional preference optimization problem, using singular value decomposition-based low-rank adaptation to guide model behavior with a low-dimensional preference vector. To address the limitations of scalar rewards in RLHF, Zhou et al. [71] introduced Multi-Objective Direct Preference Optimization, an RL-free algorithm that extends Direct Preference Optimization to handle multiple alignment objectives

efficiently. Finally, Wang et al. [57] proposed the Directional Preference Alignment framework, which incorporates multi-objective reward modeling to represent user preferences, offering intuitive arithmetic control over LLM generation for diverse user preferences.

Bayesian Optimization Bayesian optimization has been widely used in scenarios that require efficient exploration of parameter spaces, particularly when evaluating the performance of each configuration is costly or time-consuming. This method is especially advantageous in machine learning and hyperparameter tuning, where traditional optimization techniques may be computationally prohibitive. Popular Bayesian optimization methods and applications are [59; 26; 38; 44; 53].

B Performance of the quadratic approximation

We further validated the quadratic approximation of the surrogate models and the true performance by measuring the out-of-sample R^2. Figure 4 shows the relationship between the average R^2 across all tasks and the number of total points used to fit the quadratic function. As we can see from Figure 4 (a), the average R^2 values are close to 1 when the dimension of the decision space is low ($N < 4$) when using around 30 points. However, as the dimension of the decision space increases ($N \geq 4$), the average R^2 depends more heavily on the number of points (Figure 4 (b)).

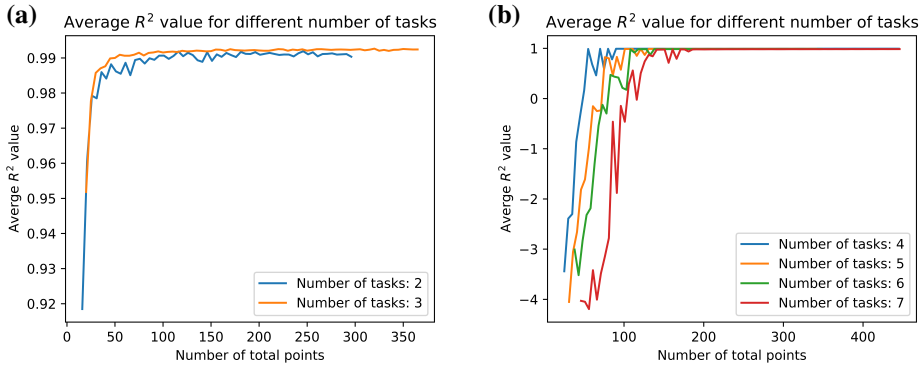


Figure 4: Average R^2 value as a measure of the quality of fitting the surrogate model for merging different numbers of tasks. (a) Number of tasks is 2 and 3; (b) Number of tasks is 4, 5, 6, and 7.

C Nested-Merging MAP

C.1 Time Complexity of Nested-Merging MAP

Table 4: Computational cost of model merging for N models.

	# evals per task	minimum # evals (total)	# evals (total)
Naive model merging	$O(N^2)$	$O(N^3)$	$O(N \cdot 2^N)$
Nested model merging	$O(1)$	$O(N \log_2^N)$	$O(N \log_2^N)$

To estimate the computational complexity of nested-merging MAP, we denote N as the number of tasks. The number of total evaluations needed for nested-merging MAP is: $N/2 \times 2 + \cdots + N/2^m \times 2^m = O(N \log_2^N)$ where $2^{m-1} < N \leq 2^m$.

Detailed calculations are as follows. When the number of tasks is 8, estimating the surrogate model in MAP (Algorithm 1) with 8 degrees of freedom in scaling coefficient vectors could still be costly. In practice, we need to sample about 300 sets of scaling coefficient vectors, get their corresponding merged models and thus take $300 \times 8 = 2400$ times of evaluation to implement MAP (Algorithm 1). However, if we implement Nested-Merging (NMMAP) (Algorithm 2), we only need to evaluate 20

merged models for each round of 2-task merging. In the first rounds, there are 4 2-task mergings running in parallel, each of them evaluates 2 tasks. Thus, it takes $4 \times 20 \times 2 = 160$ times of evaluation. In the second round, there are 2 2-task mergings running in parallel, each of them evaluates 4 tasks. It is 4 tasks rather than 2 tasks because when we merge the models: model for task a, b ($model_{a,b}$) and model for task c, d ($model_{c,d}$), we need to evaluate the merged model: $model_{a,b,c,d}$ on Task a, b, c, d. Thus, it takes $2 \times 20 \times 4 = 160$ times of evaluation. In the last round, there is only one 2-task merging and evaluation on 8 tasks. It takes $1 \times 20 \times 8 = 160$ times of evaluation. In total, NMMAP takes 480 times of evaluations, which is far less than 2400 evaluations.

In conclusion, we can see the number of rounds can be calculated by $\log_2(N)$, in each round, we need to evaluate $T \cdot N/2^i \cdot 2^i = TN$ where T is the number of scaling coefficient vectors needed to be evaluated when the number of tasks is 2. In the above example, $T = 20$. Thus, the time complexity is $O(TN \log_2(N))$. We can rewrite it to $O(N \log_2(N))$ if ignoring T.

C.2 Pareto Fronts Obtained Using Nested Merging

In our experiments, we explored the strategy of decomposing tasks into pairs, identifying the Pareto front for each pair, and then sequentially merging these binary-combined models two at a time. Our goal was to determine if this incremental merging approach affects performance negatively when compared to merging multiple models in a single operation.In Figure 5, we show the intermediate Pareto fronts obtained when merging 8 models using nested-merging MAP, where we merge two models at a time. The figures show the intermediate Pareto fronts obtained, where A_B means the model obtained by merging model A and model B.

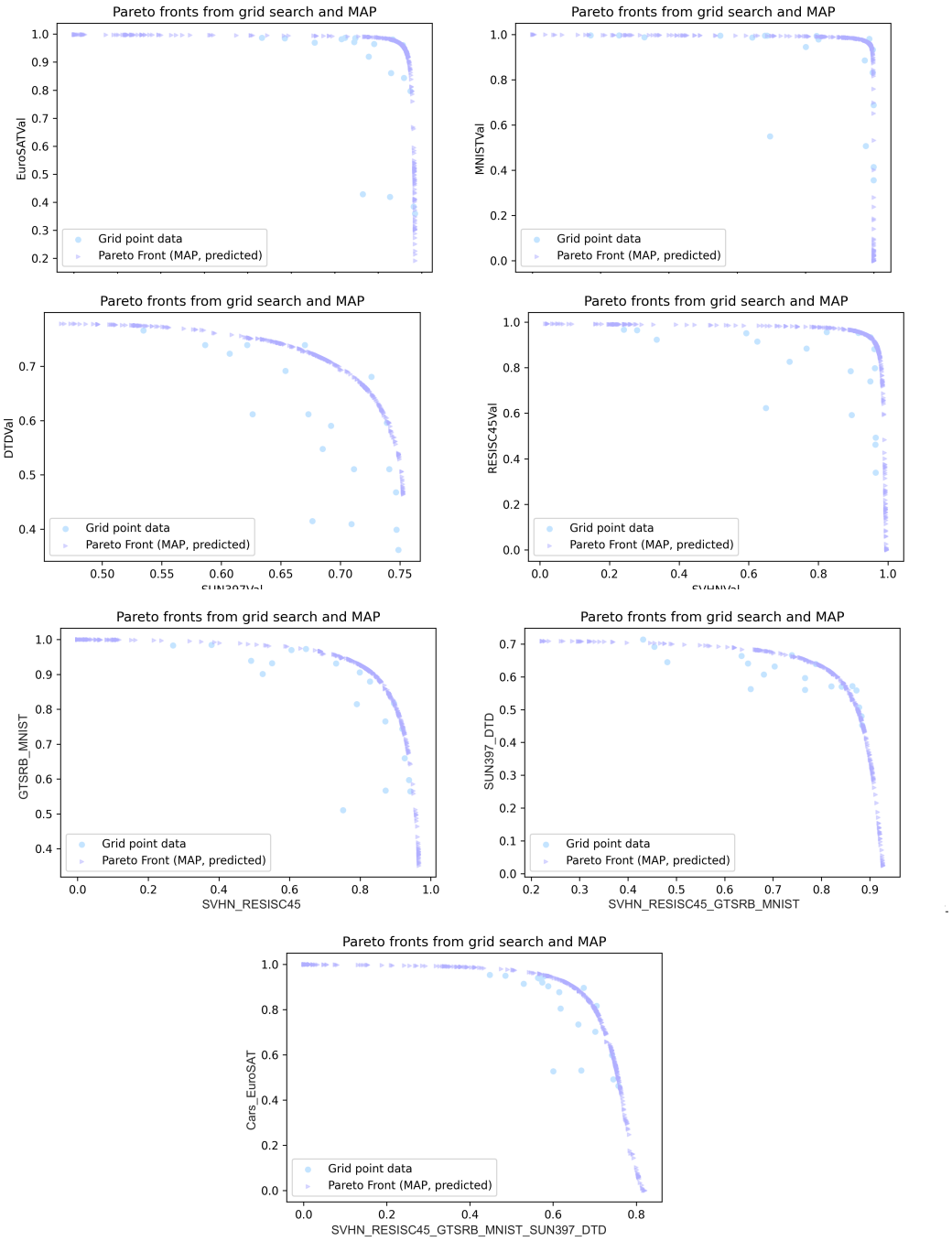


Figure 5: Illustration of the sequential steps involved in merging 8 models using nested-merging MAP (left to right, top to bottom). The figures show the intermediate Pareto fronts obtained, where A_B means the model obtained by merging model A and model B.

C.3 Nested merging with rule-based preferences

One of the main advantages of MAP is that a numeric preference vector is not required. In addition to the preference-based evaluation we performed in Table **??**, we performed another set of experiments merging four models using rule-based performance. Rule-based preference means preferences such as "I want the merged model to have performance above 70% for tasks A, B, and C, while maximizing the performance on task D". Such preferences can be easily accommodated using MAP or Bayesian

MAP, where the complete Pareto front can be obtained. However, for nested-merging MAP, since we merge two models at a time, obtaining the full Pareto front at one time is not possible. Thus, we conducted experiments on merging four models, where the merged model for two models is not picked by numeric preference vectors, but some rule-based preference.

We performed nested model merging on four tasks and compared its performance with merging four tasks directly. When merging two models (SUN397 and DTD, GTSRB and Cars), we used rule-based preferences, which requires all models to achieve 70% of the single-task fine-tuned model, while maximizing the performance on the first task (SUN397 (1) and GTSRB (3)).

Table 5: Accuracy on four tasks when using nested merging vs direct merging. We first merge models 1 and 2, and models 3 and 4 in parallel, using the preference that requires all tasks to have a normalized performance of at least 70% while optimizing the performance on tasks (1 and 3). We then compute the Pareto front for the two combined models. The table shows the highest accuracies on each task, while requiring all tasks to achieve a normalized performance of 65%.

	SUN397 (1)	DTD (2)	GTSRB (3)	Cars (4)
Nested merging	**73.24** ($\pm$1.03)	50.69 ($\pm$1.67)	96.26 $\pm$(0.31)	56.20 ($\pm$3.46)
Directly merge 4 models	68.36 ($\pm$2.63)	**61.70**($\pm$6.78)	**98.97** ($\pm$0.41)	**68.24**($\pm$3.38)

D Bayesian MAP

Generational distance and inverted generational distance We evaluated the quality of the Pareto front in capturing the shape of the ground truth Pareto front by measuring how much the predicted Pareto front converges to the ground truth Pareto front by calculating the generational distance (GD) [56] and how much the predicted Pareto front covers the ground truth Pareto front by calculating the inverted generational distance (IGD) [12]. GD and IGD are standard measures used in evolutionary multi-objective optimization to evaluate the solutions found by the evolutionary algorithms.Given two solution sets $PF_i = \{\tilde{M}_1^i(\theta_m(\mathbf{c})), \ldots, \tilde{M}_N^i(\theta_m(\mathbf{c}))\}, i = 1, 2$, the GD and IGD metrics are defined as

$GD(PF_1) \equiv \frac{1}{K}\left(\sum_{i=1}^{K} d_i^p\right)^{1/p}$ and $IGD(PF_1) \equiv \frac{1}{M}\left(\sum_{i=1}^{M} \tilde{d}_i^p\right)^{1/p}$ where d_i is the minimal Euclidean distance from $\tilde{M}_1^1(\theta_m(\mathbf{c}))$ to PF_2 and $\tilde{d}_i$ is the minimal Euclidean distance from $\tilde{M}_1^2(\theta_m(\mathbf{c}))$ to PF_1.

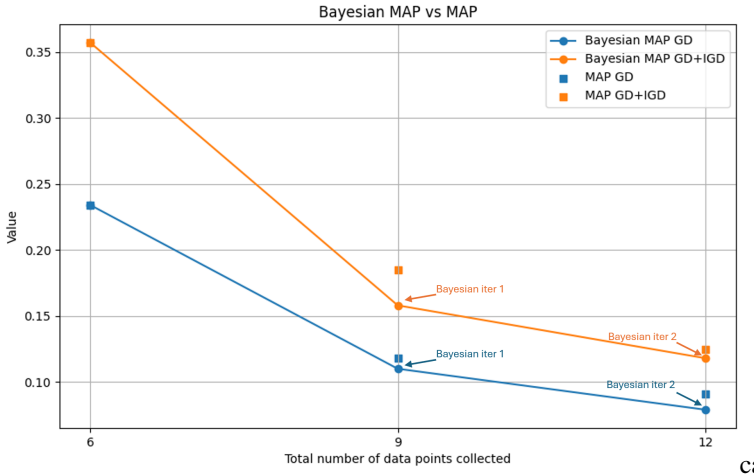
Performance of Bayesian MAP We further improve the efficiency of MAP by proposing an adaptive Bayesian sampling algorithm. This Bayesian approach samples the points from regions with the highest level of uncertainty, where uncertainty is quantified with the Upper confidence bound (UCB). Please refer to Algorithm 4 for more details about the algorithm.

Please refer to Appendix D as the experiment result comparing Bayesian MAP Algorithm 4 with MAP Algorithm 1. The very left column shows the name of the 2 tasks being merged. Below, we define points (pts) as scaling coefficients and evaluation metrics of the corresponding merged models. Please note that the result shown in this figure is the mean of 7 merging tasks that merge 2 models: DTD+Cars, DTD+RESISC45, EuroSAT+RESISC45, GTSRB+Cars, GTSRB+RESISC45, RESISC45+SVHN, SUN397+SVHN. Please also check the Pareto front estimated by Bayesian MAP with only one iteration (6+3 pts) in Figure 6.

The experiments are initialized with 6 pairs of $\mathbf{c}, \{M_n\}_{n=1}^N$ (iter 0). In every following iteration, we sample more points following the Bayesian adaptive sampling algorithm. We compare the Bayesian adaptive sampling beginning with 6 points and adding 3 additional points (6+3 pts) with running Algorithm 1 with 9 points in a row. We also compare the Bayesian adaptive sampling beginning with 6 points and adding 3 additional points for 2 times (6+2 $\times$ 3 pts) with running Algorithm 1 with 12 points in a row. We show that, under most cases, utilizing the same number of points, Bayesian adaptive sampling performs better than the run-in-a-row scheme in Algorithm 1.

In conclusion, when the number of data points (scaling coefficients and evaluation metrics of the corresponding merged models) is small, the Bayesian MAP Algorithm 4 performs better than MAP Algorithm 1. As the number of data points increases, their performance becomes closer. Thus, we

recommend implementing Bayesian MAP when the computational resource is very limited and the number of models (tasks) to merge is not high.



captionBayesian MAP compared to MAP. The x-axis represents the number of points used by either MAP or Bayesian MAP, while the y-axis represents the value for IGD or GD. We compared MAP with 6, 9, and 12 points, and Bayesian MAP with 6 initial points, sampling 3 more points each round for two rounds.

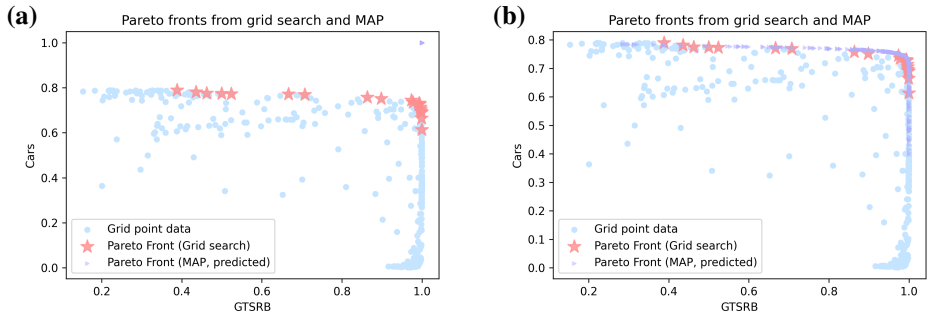


Figure 6: (a) This plot shows a failure case of the amortized Pareto front when we only have 6 initial randomly sampled pairs of $(\mathbf{c}, \{M_i(\theta_m(\mathbf{c}))\}_{i=1}^N)$ (b) After one iteration of Bayesian adaptive sampling for 3 more pairs (9 in total), the amortized Pareto front is much better than the initial Pareto front.

E Proof: Negligibility of the Remainder in Multivariate Taylor Series

Corollary: Accurate Local Approximation by Taylor Expansion If $f : \mathbb{R}^n \to \mathbb{R}$ is $(k+1)$ times continuously differentiable in a neighborhood around a point $a \in \mathbb{R}^n$, then the Taylor polynomial $T_k(x)$ of order k provides an accurate approximation of $f(x)$ when x is sufficiently close to a. Furthermore, the remainder term $R_k(x)$ becomes negligibly small as $||x - a||$ approaches zero, assuming that the $(k+1)$th derivatives of f are bounded by a constant M in the neighborhood between a and x.

Proof Consider the Taylor series expansion of f around the point a, truncated at order k:

$$T_k(x) = \sum_{|\alpha| \leq k} \frac{D^\alpha f(a)}{\alpha!}(x-a)^\alpha$$

where α is a multi-index of non-negative integers, $D^\alpha f(a)$ denotes the partial derivatives of f at a corresponding to α, and $(x-a)^\alpha = (x_1 - a_1)^{\alpha_1} \cdots (x_n - a_n)^{\alpha_n}$.

Assumptions

1. Proximity: $\|x - a\| \to 0$ where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^n$.
2. Bounded Derivatives: There exists a constant M such that for all multi-indices α with $|\alpha| \equiv k+1$, the norm of the tensor $D^\alpha f$ evaluated at any point ξ between a and x is bounded by M.

$$\|D^\alpha f(\xi)\| = \sup_{\|v_1\|=1,\ldots,\|v_{k+1}\|=1} |D^\alpha f(\xi)(v_1, \ldots, v_{k+1})| \leq M$$

The remainder term of the Taylor series expansion is given by:

$$R_k(x) = \sum_{|\alpha|=k+1} \frac{D^\alpha f(\xi)}{\alpha!}(x-a)^\alpha$$

Given the assumptions, we estimate:

$$|R_k(x)| \leq \sum_{|\alpha|=k+1} \frac{\|D^\alpha f(\xi)\|}{\alpha!}\|x-a\|^{k+1} \leq \sum_{|\alpha|=k+1} \frac{M}{\alpha!}\|x-a\|^{k+1}$$

As $\|x - a\| \to 0$, the term $\|x - a\|^{k+1}$ goes to zero. Thus, the remainder term $R_k(x)$ becomes arbitrarily small, making it negligible.

In conclusion, under the stated assumptions, the Taylor series truncated at order k, $T_k(x)$, provides an accurate approximation of $f(x)$ near a, and the remainder $R_k(x)$ can be ignored as $\|x - a\| \to 0$ and the higher-order derivatives remain bounded by M.

F Additional Experiment Results

F.1 Zero-shot Medical Image Classification

In addition to natural images, we used another dataset consisting of over 112,000 chest X-rays and 30,000 unique patients [42]. It originally contained 15 classes (14 diseases and 1 class for no finding). We split the dataset into two groups, where medical task 1 specifically tries to classify Atelectasis, Consolidation, Infiltration, Pneumothorax, and medical task 2 tries to classify Nodule, Mass, and Hernia. An example image taken from the dataset is shown in Figure 7 (a).

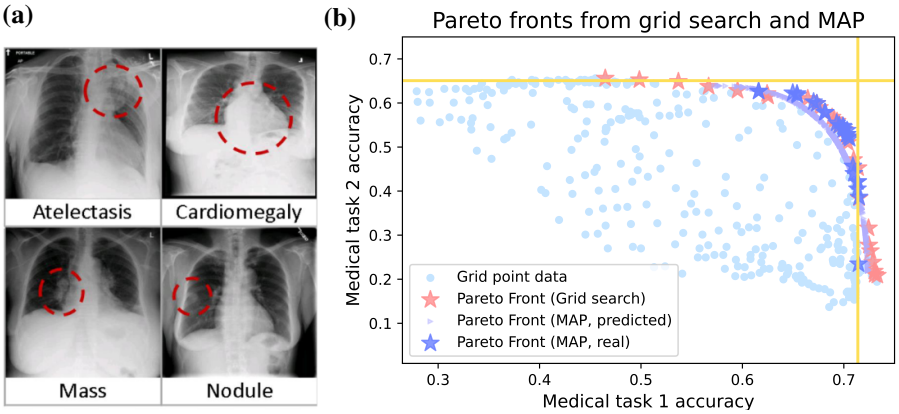


Figure 7: (a) Example figure from the NIH [42] dataset. (b) Pareto fronts found by brute-force direct search using 400 points and by MAP using 30 points. We randomly sampled 25 points from the predicted Pareto front by MAP. The resulting IGD is 0.016, and GD is 0.014.

F.2 Merging language model

We merged language models in French and Arabic, as well as Chinese and Japanese. Please refer to Table 6 and Appendix F.2. As we can see, the ground truth Pareto front is not in good shape. There are only a few points on the ground truth Pareto fronts which means few merged models could dominate the rest and the trade-off between the metrics of different languages might not be very significant. Even under this condition, our algorithm is still able to find out the Pareto fronts. We further try to merge 'Arabic+French' and 'Chinese+Japanese' in the nested scheme, Algorithm 2 with various preferences. However, the Pareto front usually only contains a single model which prevents us from predicting a Pareto front.

Table 6: Llama-3 fine-tuned models merging

	GD	IGD	GD+IGD
Arabic+French	$0.023_{0.010}$	$0.035_{0.018}$	$0.058_{0.028}$
Chinese+Japanese	$0.014_{0.013}$	$0.028_{0.017}$	$0.041_{0.026}$

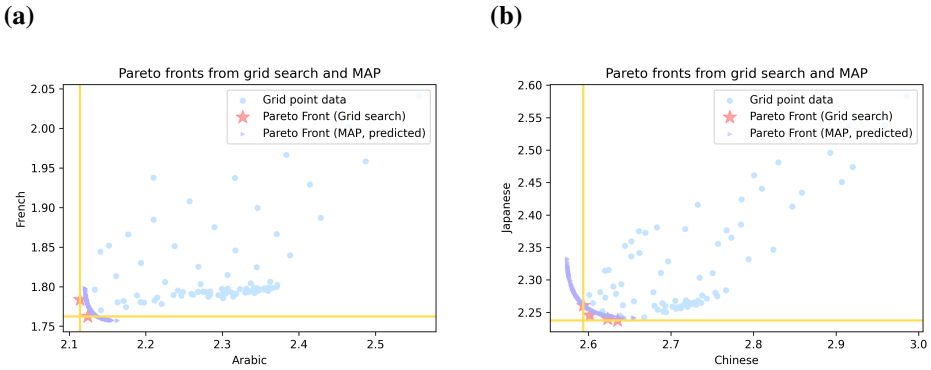


Figure 8: (a) Amortized Pareto front on merging Llama-3 fine-tuned on French with Llama-3 fine-tuned on Arabic; (b) Amortized Pareto front on merging Llama-3 fine-tuned on Chinese with Llama-3 fine-tuned on Japanese

F.3 Additional experiment results on ResNet

We performed additional experiments on ResNet18 [20] on merging two models finetuned on CIFAR10 [32] and Flowers102 [41] and show the Pareto front obtained in Figure 9. Unlike ViT models which perform zero-shot classification, ResNet requires additional fine-tuning of the classification head after model merging. We demonstrate that our method still applies to those models.

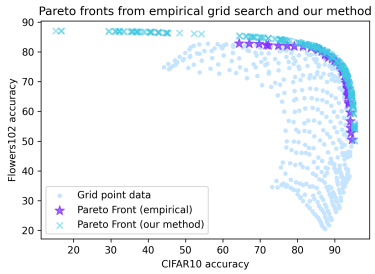


Figure 9: Pareto front obtained for two ResNet18 models on CIFAR10 and Flowers 102. We perform additional finetuning of the classification head after merging the model weights.

G More details about algorithms

G.1 Algorithm 1

In this section, we extend the discussion on line 7 of the algorithm and on Remark 1. In (5), the optimization problem is essentially minimizing the mean squared error (MSE) of a linear regression. The predictors are $\mathbf{C} = \text{concat}(c_1^2, c_2^2, \cdots, c_N^2, c_1c_2, c_1c_3, \cdots, c_{T-1}c_T, c_1, c_2, \cdots, c_N, 1)$, the variables are $(\mathbf{A}_n^*, \mathbf{b}_n^*, e_n^*)$, and the responses are $M_n(\boldsymbol{\theta}_{\text{m}}(\mathbf{c}))$.

We can certainly seek other designs to improve the accuracy and robustness of $(\mathbf{A}_n^*, \mathbf{b}_n^*, e_n^*)$. On one hand, we can use other loss functions like mean absolute error or Huber loss, while still using the linear regression. On the other, we can modify $\tilde{M}_n(\mathbf{c})$. If the metric M_n has a specific bounded region, we could restrict the fitting interval. For example, if the metric is the accuracy or F1 score between 0 and 1, we could switch the definition to $\tilde{M}_n(\mathbf{c}; \mathbf{A}_n, \mathbf{b}_n, e_n) \equiv \text{sigmoid}(e_n + \mathbf{b}_n^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_n \mathbf{c})$. This is equivalent to a logistic regression with the same definition of predictors and responses as the linear regression. If the metric is loss, which ranges from 0 to positive infinity, then we could switch to $\tilde{M}_n(\mathbf{c}; \mathbf{A}_n, \mathbf{b}_n, e_n) \equiv \text{softplus}(e_n + \mathbf{b}_n^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_n \mathbf{c})$. In summary, one may extend (5) to

$$\mathbf{A}_n^*, \mathbf{b}_n^*, e_n^* = \arg \min_{\mathbf{A}_n, \mathbf{b}_n, e_n} \text{MSE}\left(M_n(\boldsymbol{\theta}_{\text{merge}}(\mathbf{c})), \sigma(\tilde{M}_n(\mathbf{c}; \mathbf{A}_n, \mathbf{b}_n, e_n))\right),$$

and replace MSE and $\sigma = \mathbb{I}$ with other functions.

We emphasize that Algorithm 1 is an out-of-box plugin-like method. Many parts of it are kept generic: there are multiple ways to sample Ω in terms of the number of $\mathbf{c}$ and the style of sampling (Bayesian or not), thus creating a tradeoff between the quality of Pareto front and the computational time; the task vector can be computed on the more memory-capable CPU and possibly in a parameter-efficient manner (e.g. LoRA, Adapter, and BiTFiT): for example, computing $\mathbf{v}_n$ via the subtraction of two full 7B models requires $2 \times 23 = 46$GB of memory capacity, but the memory cost for PEFT models may reduce to 16MB by only subtracting the non-frozen components; the for-loops in line 2 and 4 can be computed in parallel, possibly using a much smaller subset of the data, and thus enjoying a computational speed-up.

Remark 1 (quadratic surrogate model fitting) *In Algorithm 1,* $\mathbf{A}_n^*, \mathbf{b}_n^*, e_n^*$ *in* (5) *can be readily solved by off-the-shelf methods, such as close-form solution (if applicable), LBFGS, Newton's method and gradient descent, since this is a convex problem. In Appendix G, we generalize the ways to learn the coefficients in* (4)*, besides minimizing the mean square error in* (5)*.*

G.2 Nested merging scheme

Empirically, we only need 30 pairs to get a good quality of Pareto front to merge 2 tasks, but due to the curse of dimensionality, we need exponentially more number of $(\mathbf{c}, \{\tilde{M}_n(\theta_m(\mathbf{c}))\}_{n=1}^N)$ pairs to get a good quality of Pareto front in the case of 8 tasks. Theoretically, in the best case, the points to get a good quality of Pareto front for N is $O(N^3)$. In the worst case, when the curse of dimensionality dominates, it could be $O(N \cdot 2^N)$. Using the nested merging scheme, we reduce the computation complexity from $O(N \cdot 2^N)$ to $O(N \log_2^N)$. Please refer to Table 4 for the detailed computational complexity comparison. Algorithm details are presented in Algorithm 2.

G.3 Bayesian adaptive sampling

We have discussed the strategy of bringing down the computation in the sense of big O notation. It is effective when the number of tasks involved in the Pareto fronts is high.

In the case of a relatively low number of tasks ($N \leq 4$), we consider using the Bayesian adaptive sampling method inspired by Bayesian optimization.

In Algorithm 1, we only sample a single round of scaling weights $\mathbf{c}$ and query the ground truth metric by evaluation through the $M_n(\boldsymbol{\theta}(\mathbf{c}))$. In contrast, as for the Bayesian adaptive sampling, we sample the scaling weights $\mathbf{c}$ in multiple rounds. Each round, the sampling strategy depends on the previous sampling $\mathbf{c}$ and its evaluation metrics. The process begins with an initial uniform sampling of scaling coefficients, $\{(c_1, \cdots, c_N)_i\}_{i=0}^{n_0}$, from $[0, 1]^N$. For simplicity, we define $\mathbf{c}_i$ as $(c_1, \cdots, c_N)_i$. We

Algorithm 2 Nested-merging MAP

Require: A predetermined preference *pref* $\in \mathbb{R}^N$ over the N tasks, the tuple of task, loss, task vector: $G_n = (\text{task}_n, l_n, \boldsymbol{\theta}_{ft}^n)$

Normalize *pref* to make sure the sum is 1
Initialize the set $\tau = \{G_1, \cdots, G_N\}$
while $|\tau| > 1$ **do**
  Find the pair of $(G_i, G_j) \in \tau$ that are closest to each other in terms of (l_i, l_j)
  Implement Algorithm 1 to find the Pareto front $\text{PF}_{i,j}$ between $(\tilde{M}_i, \tilde{M}_j)$
  Select $\mathbf{c}^* = (c_i^*, c_j^*) \in \mathbb{R}^2$ based on the Pareto front $\text{PF}_{i,j}$
  Merge the models by $\boldsymbol{\theta}_{\text{merge}}^{i,j} = \boldsymbol{\theta}_{\text{pre}} + c_i(\boldsymbol{\theta}_{\text{pre}} - \boldsymbol{\theta}_{ft}^i) + c_j(\boldsymbol{\theta}_{\text{pre}} - \boldsymbol{\theta}_{ft}^j)$
  Calculate the weighted average loss on the two tasks $l_{ij} = \textit{pref}_i l_i + \textit{pref}_j l_j$
  Update τ by replacing $\{G_i, G_j\}$ with $\{G_{ij}\}$, where $G_{ij} \equiv (\text{task}_{i,j}, l_{ij}, \boldsymbol{\theta}_{\text{merge}}^{i,j})$
return $\boldsymbol{\theta}_{\text{merge}}^{1,2,\cdots,N}$

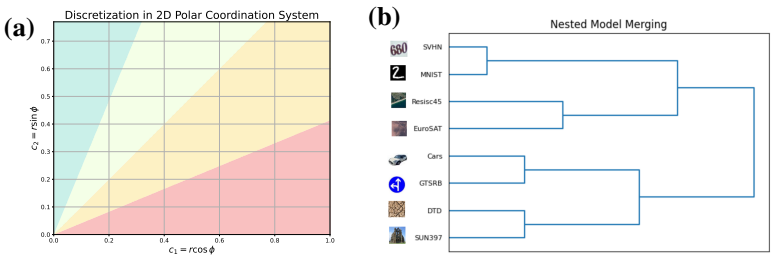


Figure 10: (a) Discretizing of two task scaling coefficients along the angular dimension in 2D polar coordinate system (please refer to Figure 11 for an example of 3D spherical discretization.); (b) An example of nested model merging for $N = 6$ models (see more details in Appendix G);

evaluate the merged model $\theta_m(\mathbf{c}_i)$ across tasks 1 to N and compute the corresponding $L2$ loss. We then discretize the regions of $\mathbf{c}$ into joint bins within a hyper-spherical coordinate system and discretize along the angular dimensions. Please refer to figure 11 as examples in 2-dimensional and 3-dimensional discretization. The posterior distribution is computed proportional to an acquisition function (e.g., upper confidence bound) of the approximation loss within specific bins.

We iteratively update each surrogate model for task n. See the algorithm 3 for a simplified version of the process. For more details, please refer to the Algorithm 4 in the appendix. After meeting the stopping criterion, we use the surrogate models for tasks $\{t\}_{n=1}^N$ to generate a lot of $(\mathbf{c}, \{\tilde{M}_n(\theta_m(\mathbf{c}))\}_{n=1}^N)$ and apply MOOP algorithm (e.g. NSGA-III) to $\{\tilde{M}_{n=1}^N\}$ to calculate the Pareto front.

Algorithm 3 Bayesian MAP

Require: Number of iterations J, Buffer $\mathcal{B}$, Pretrained model $\boldsymbol{\theta}_{\text{pre}}$, Task vectors $\mathbf{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_n)$, Evaluation function $M_n(\cdot)$ for task n.

Initialize $\mathcal{B}$ to an empty set.
for each iteration $j = 0$ to J **do**
  Sample scaling coefficients $\{\mathbf{c}_i\}_{i=1}^{n_j}$ based on updated distribution from previous results (uniformly sampled if $j = 0$).
  for each scaling coefficient $i = 1$ to n_j **do**
    Adjust the model with $\theta_m(\mathbf{c}_i) = \boldsymbol{\theta}_{\text{pre}} + \mathbf{c}_i \cdot \mathbf{V}$.
    Evaluate this model configuration $m_{n,i} = M_n(\theta_m(\mathbf{c}_i))$ and store $(\mathbf{c}_i, m_{n,i})$ in $\mathcal{B}$.
  Update the surrogate model based on the $(\mathbf{c}, \{\tilde{M}_n(\theta_m(\mathbf{c}))\}_{n=1}^N)$ pairs in $\mathcal{B}$.
  Estimate acquisition function (UCB) for different model configurations.
  Adjust the sampling strategy for the next iteration based on these evaluation metrics.
return

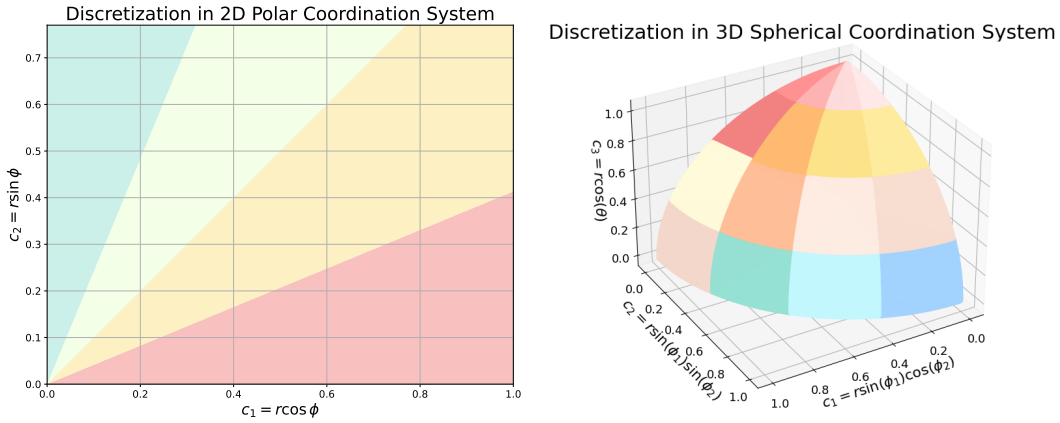
G.4 Algorithm 2

In this section, we explain the operations of the algorithm in Figure 10 in details. Here task 1 to 8 is Cars, GTSRB, DTD, SUN397, Resisc45, and SVHN. If we minimize (3) without the nested merging, we would need to estimate $\mathbf{A}_1, ..., \mathbf{A}_8 \in \mathbb{R}^{8\times 8}$, with hundreds of $\mathbf{c}$.

With the nested merging, for the first round, we merge $(\boldsymbol{\theta}^1_{ft}, \boldsymbol{\theta}^2_{ft})$into $\boldsymbol{\theta}^{1,2}_{\text{merge}}$, thus approximating $\mathbf{A}_1$ and $\mathbf{A}_2 \in \mathbb{R}^{8\times 8}$ by $\mathbf{A}_1[1:2, 1:2]$ and $\mathbf{A}_2[1:2, 1:2] \in \mathbb{R}^{2\times 2}$, respectively. That is, we only care about the interference between task 1 and 2, but not task 1 and 5. Similarly, we merge $(\boldsymbol{\theta}^3_{ft}, \boldsymbol{\theta}^4_{ft})$into $\boldsymbol{\theta}^{3,4}_{\text{merge}}$, and $(\boldsymbol{\theta}^5_{ft}, \boldsymbol{\theta}^6_{ft})$into $\boldsymbol{\theta}^{5,6}_{\text{merge}}$. Next, we merge $(\boldsymbol{\theta}^{1,2}_{\text{merge}}, \boldsymbol{\theta}^{3,4}_{\text{merge}})$into $\boldsymbol{\theta}^{1,2,3,4}_{\text{merge}}$, and finally into $\boldsymbol{\theta}^{1,2,3,4,5,6,7,8}_{\text{merge}}$.

G.5 Algorithm 3

Algorithm 4 is a detailed version of Algorithm 3. Figure 11 includes illustration of our discretization method (how we create bins) in 2D and 3D decision variable ($\mathbf{c}$) space.



(a) (b)

Figure 11: (a) Discretizing of two task scaling coefficients along the angular dimension in 2D polar coordinate system; (b) Discretizing of three task scaling coefficients along the angular dimensions in 3D spherical coordinate system;

Algorithm 4 Bayesian Adaptive of Surrogate Model

Require: Number of iterations J, Buffer $\mathcal{B}$, Pretrained model $\boldsymbol{\theta}_{\text{pre}}$, Task vectors $\mathbf{v}_n$, Evaluators for task N, $M_n(\cdot)$, Discretization bin number K, sample size for every iteration n_j, $j = 0$ to J, Bootstrap dropping rate $\alpha = 20\%$, Bootstrap sampling number $Q = 30$.

$\mathcal{B} \leftarrow \emptyset$
for $j = 0$ to J **do**
  if $j = 0$ **then**
    Sample n_0 scaling coefficients $\{\mathbf{c}_i\}_{i=1}^{n_j}$ from $U([0,1]^N)$
  else
    Sample n_j scaling coefficients $\{\mathbf{c}_i\}_{i=1}^{n_j}$ based on the posterior distribution
  for $i = 0$ to n_j **do**
    Merge the model $\theta_m(\mathbf{c}_i) = \boldsymbol{\theta}_{\text{pre}} + \mathbf{c}_i \cdot \mathbf{v}_n$
    Evaluate $m_{n,i} = M_n(\theta_m(\mathbf{c}_i))$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{c}_i, m_{n,i})\}$
  Fit the quadratic approximation surrogate model $\tilde{M}_n$ by learning $\mathbf{A}_n^*, \mathbf{b}_n^*, e_n^*$ in (5).
  Discretize the scaling coefficients along the angular dimensions in hyper-spherical coordinates (see figure 11 as examples)
  for $k = 0$ to K **do**
    Calculate the mean of $L2$ loss between $\tilde{M}_n(\mathbf{c}_i)$ and $M_t(\mathbf{c}_i)$, where $\mathbf{c}_i$ are in bin k, denoted as mean_k
    {Bootstrap to estimate the standard deviation of the losses.}
    for $q = 0$ to Q **do**
      Randomly (uniformly) drop α scaling coefficient in bin k
      Calculate the mean of $L2$ loss between $\tilde{M}_n(\mathbf{c}_i)$ and $M_t(\mathbf{c}_i)$ with the rest points and denoted with l_q
    Calculate the standard deviation of the $\{l_q\}_{q=0}^{Q}$ and denoted as std_k
    $\text{score}_k = \text{mean}_k + \frac{1}{2}\text{std}_k$
  Calculate probability distribution across the discretized bins by score_k as the posterior sampling strategy in the next round
return

H Potential QA

Q1: Why nested-merging MAP does not give a complete Pareto front? In nested-merging MAP, we merge the models in pair. For example, there are $model_i$ for task i as individual fine-tuned models. $i = 1, 2, 3, 4$. We first merge model 1 and model 2 given the preference of the user between task 1 and task 2. We then get $model_{1,2}$. At the same time, we merge model 3 and model 4 given the preference of the user between task 3 and task 4 and get $model_{3,4}$. Finally, we merge the $model_{1,2}$ and $model_{3,4}$ given the preference of user to get $model_{1,2,3,4}$. We output the $model_{1,2,3,4}$ as the output merged model of the algorithm. Please note that the merging order in the algorithm should be decided by the loss function clustering. It is a heuristic decision and does not always dominate other merging orders. Thus, we choose not to emphasize the contribution of this order. The practitioner may use any order they think can be helpful on the merging.

Q2: It seems nested-merging MAP performs worse than MAP. What is the meaning of it? In theory, the surrogate model can be easily fitted when the number of tasks is low. When it is high (e.g. 8), the fit of the surrogate model can no longer be a near-perfect approximation (please find the R2 in Table 2). Thus, even if the NMMAP can only find the suboptimal solution, it is still comparable to MAP Algorithm 1 according to **??**. We understand in general that NMMAP does not find the global optimum, but please kindly keep in mind that even if their performance is comparable, NMMAP takes way less computation of evaluation.

Q3: Why nested-merging MAP does not output the optimal solution? The solution found by nested-merging MAP (NMMAP) is indeed suboptimal given the limited search space compared with merging all models at once. However, it does not mean that it is not useful in all situations. When the number of tasks is high, it has comparable performance to MAP while consuming much fewer computations on evaluation.

Q4: The authors assume a setting where evaluation models are computationally expensive to query. But is this the reality? Yes, to get the Pareto front of tasks that have trade-offs. We need to have a lot of data points to show the trade-off. Here, each data point is the evaluation metric of a model. To get the ground truth evaluation metric of a task (e.g. let's say classification), we need to run the evaluation script of the model to determine the metric. If we have 4 tasks, and we set 5 possible values (let's say 0.2, 0.4, 0.6, 0.8, 1) for the scaling coefficient vector of each model specialized for one task. We will have to evaluate $5^4 = 625$ models on all 4 tasks. Assuming each evaluation takes 1 minute, evaluating all the models on all the 4 tasks will take $625 \text{ models} \times 4 \text{ tasks} \times 1 \text{ minute} = 2500 \text{ minutes} = 41.7$ hours which is expensive. In big O notation, assuming we have T tasks, the time of evaluation for each task is T. The time (computational) complexity is $O(TN2^N)$.

Q5: Why does quadratic approximation have a lower cost? For the MAP algorithm (Algorithm 1), the time complexity is the same as what we mentioned above, what is different is that we fitted an approximation of the evaluation metrics. We only need the scaling coefficient vectors to input to a quadratic function model to be able to get the estimated evaluation score. Run the quadratic function once take only $3.91 \times 10^{-6}s \pm 894 \times 10^{-9}s$. And thus, evaluating 2500 times takes $< 2500 \times 4 \times 10^{-6}s = 10^{-2}s$.

Q6: Why not compare to Git-Rebasin [1]? We didn't compare our method to Git Re-Basin because their study focuses on the scenarios of merging the models from different initializations, whereas our background works on the same initialization of different fine-tuned models.

Q7: How do you deal with gradient and Hessian in the second-order Taylor expansion? Notations:

- p as the number of parameters in the pre-trained model (also the number of parameters in each task vector).
- $\mathbf{V}$ is the matrix of task vectors of different N tasks. Thus, $\mathbf{V} \in \mathbb{R}^{p \times N}$.
- $\mathbf{c}$ is the scaling coefficient vector $\in \mathbb{R}^N$.
- M_n is the metric (e.g. accuracy) for the task n.

$$M_n(\mathbf{c}) = \underbrace{M_n(\boldsymbol{\theta}_{\text{pre}})}_{\in\mathbb{R}} + \underbrace{\nabla M_n(\boldsymbol{\theta}_{\text{pre}})^\top}_{\in\mathbb{R}^{1\times p}} \underbrace{\mathbf{V}}_{\in\mathbb{R}^{p\times N}} \underbrace{\mathbf{c}}_{\in\mathbb{R}^{N\times 1}} + \frac{1}{2} \underbrace{(\mathbf{V}\mathbf{c})^\top}_{\in\mathbb{R}^{1\times p}} \underbrace{\mathbf{H}_n(\boldsymbol{\theta}_{\text{pre}})}_{\in\mathbb{R}^{p\times p}} \underbrace{\mathbf{V}\mathbf{c}}_{\in\mathbb{R}^{p\times 1}} + \underbrace{R_n}_{\in\mathbb{R}} \tag{6}$$

$$= \underbrace{e_n}_{\in\mathbb{R}} + \underbrace{\mathbf{b}_n^\top}_{\in\mathbb{R}^{1\times N}} \underbrace{\mathbf{c}}_{\in\mathbb{R}^{N\times 1}} + \underbrace{\mathbf{c}^\top}_{\in\mathbb{R}^{1\times N}} \underbrace{\mathbf{A}_n}_{\in\mathbb{R}^{N\times N}} \underbrace{\mathbf{c}}_{\in\mathbb{R}^{N\times 1}} \tag{7}$$

$$\tag{8}$$

$$\tilde{M}_n(\mathbf{c}; \mathbf{A}_n, \mathbf{b}_n, e_n) \equiv e_n + \mathbf{b}_n^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_n \mathbf{c}$$

where

$$\mathbf{A}_n = \mathbf{V}^\top \mathbf{H}_n(\boldsymbol{\theta}_{\text{pre}})\mathbf{V} \in \mathbb{R}^{N\times N}, \mathbf{b}_n = \mathbf{V}^\top \nabla M_n(\boldsymbol{\theta}_{\text{pre}}) \in \mathbb{R}^N, e_n = M_n(\boldsymbol{\theta}_{\text{pre}}) + R_n \tag{9}$$

Please notice that $\mathbf{A}$ is a symmetric matrix. Specifically, when the number of tasks is 2, we have:

$$\tilde{M}_1(\mathbf{c}; \mathbf{A}_1, \mathbf{b}_1, e_1) \equiv e_1 + \mathbf{b}_1^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_1 \mathbf{c} \tag{10}$$

$$= \frac{1}{2}A_{1,11}c_1^2 + A_{1,12}c_1c_2 + \frac{1}{2}A_{1,22}c_2^2 + b_{1,1}c_1 + b_{1,2}c_2 + e_1 \tag{11}$$

$$\tilde{M}_2(\mathbf{c}; \mathbf{A}_2, \mathbf{b}_2, e_2) \equiv e_2 + \mathbf{b}_2^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_2 \mathbf{c} \tag{12}$$

$$= \frac{1}{2}A_{2,11}c_1^2 + A_{2,12}c_1c_2 + \frac{1}{2}A_{2,22}c_2^2 + b_{2,1}c_1 + b_{2,2}c_2 + e_2 \tag{13}$$

$$\tag{14}$$

We don't calculate gradient or Hessian to get $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{b}_1$, $\mathbf{b}_2$, e_1 and e_2. We use linear regression to estimate them. How? Given a (c_1, c_2) pair, we can define a merged model $\boldsymbol{\theta}_{\text{merge}}(c_1, c_2)$, we evaluate the merged model on task 1 and task 2 to get the metrics (e.g. accuracy). Given 20 pairs of (c_1, c_2), we would be able to evaluate get 20 corresponding (M_1, M_2) which are metric for task 1 and metric for task 2. Thus, we can fit the surrogate model $\tilde{M}_1(\mathbf{c}; \mathbf{A}_1, \mathbf{b}_1, e_1)$ and $\tilde{M}_2(\mathbf{c}; \mathbf{A}_2, \mathbf{b}_2, e_2)$.

I NeurIPS 2024 reviewer comments

MAP: Model Merging with Amortized Pareto Front Using Limited Computation

Lu Li, Tianyu Zhang, Zhiqi Bu, Suyuchen Wang, Huan He, Jie Fu, Yonghui Wu, Jiang Bian, Yong Chen, Yoshua Bengio

14 May 2024 (modified: 25 Sept 2024) Submitted to NeurIPS 2024 Conference, Senior Area Chairs, Area Chairs, Reviewers, Authors Revisions BibTeX CC BY 4.0

Keywords: model merging, transfer learning, multitask learning, task arithmetic, multi-objective optimization

TL;DR: We provide a computation-efficient algorithm for finding the Pareto front representing the trade-offs during model merging caused by conflicting objectives between different tasks.

Abstract:
Model merging has emerged as an effective approach to combine multiple single-task models, fine-tuned from the same pre-trained model, into a multitask model. This process typically involves computing a weighted average of the model parameters without any additional training. Existing model-merging methods focus on enhancing average task accuracy. However, interference and conflicts between the objectives of different tasks can lead to trade-offs during model merging. In real-world applications, a set of solutions with various trade-offs can be more informative, helping practitioners make decisions based on diverse preferences. In this paper, we introduce a novel and low-compute algorithm, Model Merging with Amortized Pareto Front (MAP). MAP efficiently identifies a Pareto set of scaling coefficients for merging multiple models to reflect the trade-offs. MAP approximates the evaluation metrics of the various tasks using a quadratic approximation surrogate model derived from a pre-selected set of scaling coefficients, enabling amortized inference. Experimental results on vision and natural language processing tasks show that MAP can accurately identify the Pareto front. To further reduce the required computation of MAP, we propose (1) a Bayesian adaptive sampling algorithm and (2) a nested merging scheme with multiple stages.

Corresponding Author: yoshua.bengio@mila.quebec
Reviewer Nomination: luli1@sas.upenn.edu
Primary Area: Other (please use sparingly, only use the keyword field for more details)
Submission Number: 8794

Paper Decision

Decision | Program Chairs | 25 Sept 2024, 11:19 (modified: 25 Sept 2024, 13:01) | Program Chairs, Authors | Revisions

Decision: Reject

Comment:
The paper proposes a scheme for merging models for multiple tasks, with the aim of obtaining a solution on the Pareto frontier of the tasks. The idea is to construct a suitable surrogate to the individual tasks' metrics, and then employ tools from multi-objective optimization.

Reviewer opinion was somewhat split on the paper. The main critiques were:

- potential limited novelty over existing works in multi-objective optimization, such as "Pareto multi-task learning". The authors' argument was that the algorithms for the two are fundamentally different, with the latter being gradient-based. From my reading, I tend to agree that the technical content could be more clearly expressed to evince its novelty:
  - Algorithm 1 involves a reduction to existing MOO algorithms, with the key ingredient appearing to be the construction of a suitable surrogate to the individual task metrics. It is not made *explicit* why such a surrogate is required: the title & contributions note the "amortized" improvements, but this does not appear to be explicitly discussed in Section 3.1.
  - It is not immediately apparent to what extent this component leverages specific structure of the model-merging problem; i.e., given a *generic* multi-objective optimization problem, would the same strategy of constructing a quadratic approximation suffice?
  - It is not immediately apparent why the proposed strategy is to be favored over a myriad of alternatives, e.g., linear scalarization, particularly given the apparent focus on convex Pareto fronts.
- lack of clarity in discussion of Nested merging scheme & Bayesian adaptive sampling. From my reading, I agree that these sections are too terse to be considered as part of the primary contributions. e.g., there is no discussion or intuition in the main body for any part of Algorithm 2.
- potentially unclear gains over some existing methods, like TIES-Merging.

Given these, we believe the paper may be best served by revision and a fresh round of review.

TL;DR: summarizing rebuttal/discussion phase

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 14 Aug 2024, 07:26 (modified: 14 Aug 2024, 07:44) | Program Chairs, Senior Area Chairs, Area Chairs, Authors | Revisions

Comment:
Dear AC,

I hope this message finds you well. Below is a summary of the rebuttal and discussion phase for our submission.

We have carefully incorporated the reviewers' feedback, made the necessary revisions, and expanded the related work section in the updated manuscript, which is available at the following link: https://anonymous.4open.science/api/repo/MAP-NEURIPS/file/revised_paper.pdf

P.S. If the LaTeX does not render correctly, please kindly refresh the browser.

Summary:

- **Contribution:** We introduce three novel algorithms designed to estimate the Pareto front in **model-merging** scenarios with **limited computational resources** and without relying on gradient or Hessian information.
- **Motivation:** A Pareto front offers practitioners the flexibility to make decisions based on their specific preferences for different tasks.

Algorithm 1: MAP (Main Algorithm)

- We employ a surrogate model (quadratic model) to approximate task accuracy. Traditional methods require evaluating thousands of merged models to compute the Pareto front, which is time-consuming. By using the surrogate model, this process is significantly accelerated.
- **Efficiency:** Our method is, on average, **6x faster** while maintaining the same level of performance of direct search.

Algorithm 2: Nested-Merging MAP (NMMAP)

- This approach further reduces computation when **the number of models is high**.
- **Efficiency:** It achieves a **250x speedup** when the number of tasks is 8, while also outperforming direct search methods.
- **Advantages:** NMMAP reduces the number of evaluations from $O(TN2^N)$ to $O(N \log N)$.
- **Limitations:** It does not produce a complete Pareto front as it merges models in pairs midway through the algorithm, utilizing partial preference.

Algorithm 3: Bayesian MAP (BMAP)

- This algorithm uses Bayesian optimization to determine the distribution of scaling coefficients for task vectors when approximating the surrogate model in MAP.
- **Efficiency:** It is **16x faster** when the number of tasks is 2, with performance comparable to direct search methods.
- **Advantages:** BMAP offers a significant reduction in computation, particularly **when the number of tasks is low**.
- **Limitations:** It is less suitable for scenarios with a high number of tasks.

Compute required: direct search ($O(TN2^N)|T = 60s$) > MAP ($O(N2^N)$) > BMAP ($O(N2^N)$) >> NMMAP ($O(N \log N)$), where N is the number of tasks.

General Rebuttals

- Our MAP algorithm is based on the task arithmetic algorithm. In response to the reviewers' request for comparisons with additional model-merging baselines, we sampled a variety of preferences and compared performance on average. The results indicate that MAP generally outperforms TIES-merging, the current state-of-the-art algorithm in model merging.
- In response to requests for comparisons between NMMAP and other model-merging baselines, our findings show that NMMAP achieves a winning ratio of 45% while requiring significantly less computational resources than MAP.
- We have also addressed typos and clarified explanations that were previously unclear in the manuscript.

Responses to Reviewer-Specific Questions

Reviewer R2T3

- **Comment:** The reviewer requested a comparison with other Multi-Objective Optimization Problem (MOOP) methods.
  - **Response:** Our method differs fundamentally from traditional MOOP approaches in the Multi-Task Learning (MTL) literature because it is not gradient-based. Specifically, we do not require the calculation of any first-order information, such as gradients, to approximate the surrogate model. Our approach focuses on task vector-based model merging, which aims to combine multiple fine-tuned models originating from the same pre-trained checkpoint to create a multi-task model. While this method may generally be less powerful than gradient-based MOOP (MTL) methods, it is uniquely suited for scenarios where gradient information is unavailable or impractical to compute.
- **Comment:** The reviewer requested further clarification on the winning rate in the MAP vs. direct search experiment.
  - **Response:** We compared the performance of MAP with that of the direct search method. When the number of tasks N is low (e.g., $N = 2, 3, 4$), we consider the Pareto front found by the direct search method to be the ground truth. In these cases, MAP achieves a winning rate close to 50%, indicating its ability to effectively recover the ground truth Pareto front. However, as the number of tasks increases (e.g., $N = 5, 6, 7, 8$), the direct search method becomes less capable of finding the true Pareto fronts due to the curse of dimensionality. As a result, MAP's winning rates in these higher-dimensional scenarios are significantly above 50%, demonstrating its superior performance in complex settings.

Responses to Reviewer-Specific Questions

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 14 Aug 2024, 07:40 (modified: 14 Aug 2024, 07:46)

Program Chairs, Senior Area Chairs, Area Chairs, Authors | Revisions

Comment:

Responses to Reviewer-Specific Questions (continue)

Reviewer R2T3 (continue)

- **Comment:** The reviewer requested the calculation of a Pareto front for TIES-merging.
  - **Response:** It is not possible to calculate a Pareto front for TIES-merging because it does not employ different scaling coefficients for different task vectors. In other words, it lacks the necessary controls to represent user preferences.
- **Comment:** The reviewer suggested that our work is similar to Pareto MTL (https://arxiv.org/abs/1912.12854).
  - **Response:** We respectfully disagree and argue that our algorithm is fundamentally different. Pareto MTL requires the calculation of gradients, which makes it incompatible with model-merging scenarios where gradients are not available.

Reviewer V9hH

- **Comment:** The reviewer requested a comparison with the Git Re-Basin model merging algorithm.
  - **Response:** The Git Re-Basin study focuses on merging models initialized differently, whereas our work assumes models that share the same initialization but are fine-tuned differently. Therefore, we believe that a comparison with Git Re-Basin would not be relevant to our study.
- **Comment:** The reviewer requested a comparison with a baseline using a standard evolutionary algorithm without a surrogate model.
  - **Response:** Applying a standard evolutionary algorithm without a surrogate model in our context would be nearly infeasible due to the extremely high computational costs, which could exceed 500 hours for a single test. This is why we opted to use a surrogate model in our approach.

Reviewer 8WQN

- **Comment:** The reviewer requested more detailed explanations of our algorithm.
  - **Response:** We provided additional explanations, including examples and analyses, to clarify the workings of our algorithm.
- **Comment:** The reviewer requested a more detailed explanation of the time complexity calculation for NMMAP.
  - **Response:** We have included a thorough explanation, supported by examples and detailed analysis, to elucidate the time complexity of NMMAP.

Reviewer FQxa05

- **Comment:** The reviewer requested a more detailed explanation of time complexity.
  - **Response:** We have provided a comprehensive explanation addressing this concern.
- **Comment:** The reviewer was concerned about the computational cost of calculating the Hessian.
  - **Response:** We clarified that our approach avoids the need to calculate gradients and the Hessian by fitting a surrogate model, thus significantly reducing computational costs.

reducing computational costs.

- **Comment:** The reviewer requested validation of using Taylor expansion.
  - **Response:** We demonstrated the validity of our approach by showing that the L1 norm difference between fine-tuned models and pre-trained models is less than 1.66%.

Reviewer G7ak

- **Comment:** The reviewer requested an explanation of the NSGA-III algorithm.
  - **Response:** We have provided a detailed explanation of NSGA-III and included it in the appendix.
- **Comment:** The reviewer requested a comparison between MAP and Bayesian optimization without the surrogate model.
  - **Response:** We explained that our approach relies on the Upper Confidence Bound (UCB), which is based on the mean plus half the standard deviation of the fitting error of the surrogate model. Without the surrogate model, such a metric to guide sampling would not be available.
- **Comment:** The reviewer asked how the surrogate model is fitted.
  - **Response:** We provided a detailed explanation showing that the parameters of the surrogate model can be obtained through a closed-form regression solution.
- **Comment:** The reviewer asked how user preferences are included in the algorithm.
  - **Response:** We explained that in both MAP and BMAP, the algorithm generates a Pareto front, allowing users to select the optimal solution based on their preferences. In NMMAP, user preferences are utilized to merge models in pairs during the algorithm's execution.
- **Comment:** The reviewer noted that the performance in the BMAP vs. MAP experiment appeared less favorable compared to previous experiments.
  - **Response:** We clarified that the comparison between BMAP and MAP was conducted under conditions of **very limited** computational resources. In this experiment, we collected only 9 and 12 data points, whereas our typical experiments involve collecting 20 data points.

Could you please help us remind the reviewers for paper 8794 for responding to rebuttal?

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 11 Aug 2024, 16:45

Program Chairs, Senior Area Chairs, Area Chairs, Authors

Comment:

Dear AC,

I hope you are doing well! I'm writing because none of the 5 reviewers assigned to paper 8794 has responded to the rebuttals. Could you please kindly send a reminder to the reviewers, as the discussion period ends in two days?

Thank you very much! We really appreciate your help.

Best, Authors of paper 8794

Official Comment by Area Chair yW6h

Official Comment | Area Chair yW6h | 11 Aug 2024, 18:16 | Program Chairs, Senior Area Chairs, Area Chairs, Authors

Comment:

Dear authors, I have sent the reviewers a reminder.

Replying to Official Comment by Area Chair yW6h

Official Comment by Authors

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 11 Aug 2024, 20:42

Program Chairs, Senior Area Chairs, Area Chairs, Authors

Comment:

Thank you!

Replying to Official Comment by Authors

Could you please help us remind the remaining 4 reviewers for paper 8794 for responding to rebuttal?

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 12 Aug 2024, 15:20

Program Chairs, Senior Area Chairs, Area Chairs, Authors

Comment:

Dear AC,

Thank you very much for your help with sending a reminder to the reviewers. Reviewer G7ak has responded to our rebuttal, acknowledging our additional experiments and clarifications.

We sincerely appreciate the time and effort the reviewers have devoted to reviewing our work and understand that their schedule can be quite busy. However, **as the authors-reviewer discussion phase draws to a close**, we kindly request your help in encouraging the remaining four reviewers to review our responses.

Thank you very much for your continued support!

Best, Authors of paper 8794

Reference list of the rebuttal

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 07 Aug 2024, 06:43 (modified: 08 Aug 2024, 06:01)

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers, Reviewers Submitted, Authors | Revisions

Comment:

[1] Lin, Xi, et al. "Pareto multi-task learning." Advances in neural information processing systems 32 (2019).
[2] Navon, Aviv, et al. "Learning the pareto front with hypernetworks." arXiv preprint arXiv:2010.04104 (2020).
[3] Lin, Xi, et al. "Controllable pareto multi-task learning." (2020).
[4] Ruchte, Michael, and Josif Grabocka. "Scalable pareto front approximation for deep multi-objective learning." IEEE international conference on data mining (ICDM) (2021).
[5] Rame, Alexandre, et al. "Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards." Advances in Neural Information Processing Systems 36 (2024).
[6] Jang, Joel, et al. "Personalized soups: Personalized large language model alignment via post-hoc parameter merging." arXiv preprint arXiv:2310.11564 (2023).
[7] Zhong, Yifan, et al. "Panacea: Pareto Alignment via Preference Adaptation for LLMs." arXiv preprint arXiv:2402.02030 (2024).
[8] Zhou, Zhanhui, et al. "Beyond one-preference-for-all: Multi-objective direct preference optimization." arXiv preprint arXiv:2310.03708 (2023).

[9] Wang, Haoxiang, et al. "Arithmetic control of llms for diverse user preferences: Directional preference alignment with multi-objective rewards." arXiv preprint arXiv:2402.18571 (2024).
[10] Ainsworth, Samuel K., Jonathan Hayase, and Siddhartha Srinivasa. "Git re-basin: Merging models modulo permutation symmetries." arXiv preprint arXiv:2209.04836 (2022).
[11] Daheim, Nico, et al. "Model merging by uncertainty-based gradient matching." arXiv preprint arXiv:2310.12808 (2023).
[12] Ortiz-Jimenez, Guillermo, Alessandro Favero, and Pascal Frossard. "Task arithmetic in the tangent space: Improved editing of pre-trained models." Advances in Neural Information Processing Systems 36 (2024).
[13] Jin, Xisen, et al. "Dataless knowledge fusion by merging weights of language models." arXiv preprint arXiv:2212.09849 (2022).

Author Rebuttal by Authors

Author Rebuttal Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) 07 Aug 2024, 05:53 (modified: 07 Aug 2024, 11:28)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors Revisions

Rebuttal:

General response to all reviewers

We thank all the reviewers for their time and valuable comments. We are pleased that reviewers appreciate that this paper introduces a novel approach to tackle an important question (Reviewers V9hH, 8WQN, G7ak) and find MAP mathematically and empirically sound (Reviewers R2T3, V9hH). Below is our general response to all reviewers:

Contributions of MAP

We appreciate that Reviewers V9hH, 8WQN, and G7ak have explicitly acknowledged that the approach is novel. In summary, novel contributions of MAP are the following:

1. **CPU-level Pareto front approximation**: MAP approximates the whole Pareto front using only CPU level calculations. This is distinct from other approaches in the MOOP literature, such as Pareto MTL [1] or Panacea [7], which rely on GPU-level gradient-based training to adapt different preferences from practitioners. MAP is also distinct from conventional MOEA approaches which involves many queries of the merged models, which can be expensive and easily scales up as the number of tasks increases due to the curse of dimensionality. MAP uses a surrogate model to effectively amortize the cost.
2. Use a **surrogate model** to approximate the evaluation metric to determine the Pareto front based on seas of estimation of evaluation generated by the surrogate model. Previous methods require MGDA or finetuning or hypernetworks.
3. Implement **Bayesian Optimization on the surrogate model** to further bring down the evaluation cost for the surrogate model and proposed nested-merging scheme so that when the number of tasks is high, the number of evaluations of merged model of the surrogate model won't grow exponentially.
4. **Entire Pareto front approximation without needing user-prespecified preferences**: MAP can uncover the entire Pareto front without needing the users to input numeric preferences. Many existing algorithms for finding Pareto optimal solutions require the users to input their preferences. However, it has been noted as a limitation because it is unclear how to find the user's preference vector corresponding to the most suitable solution for him/her [2]. MAP can directly output the entire Pareto front, overcoming this limitation.
5. **Plug-in to other task vector-based model merging methods**: Our method serves as a plugin to all task vector-based model-merging methods where the practitioners' the degree of freedom to determine the scaling coefficient vectors (e.g. task arithmetic and DARE combined with task arithmetic). Details on experiments for combining MAP and DARE can be found in the one-page PDF supplement to this rebuttal.

New experiments

1. We compared with various existing model-merging methods, including TIES-merging, Task Arithmetic with a single scalar, Task Arithmetic with preferences as scalars, DARE-TIES, DARE-TA, and SLERP.
2. Visualizations of the results for dimension 2 can be found in the **Figure 1 in the 1-page PDF file**.
3. For higher dimensions, we evaluated the solutions found by MAP and baseline methods using a set of 20 preferences vectors. We then computed the sum of preference-weighted accuracies of all tasks. We also computed the win rate (percentage of preferences where the weighted sum of MAP is higher than those of baseline methods).

Preference weighted sum(↑)	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Single task models	75.84±1.76	77.03±1.84	82.43±4.40	87.69±4.50	88.52±4.02	89.26±3.58	90.62±2.52
MAP	70.70±0.21	69.05±1.41	72.84±1.05	77.31±0.83	74.26±0.52	73.40±-0.14	72.96±0.73
TIES-merging	69.30±0.33	67.60±0.58	71.79±2.93	76.49±3.10	73.74±2.96	72.54±2.87	72.24±1.91
DARE-TIES	67.62±1.65	66.49±2.34	71.39±4.45	74.55±4.55	73.34±4.10	71.43±3.84	71.89±2.86
Task Arithmetic	70.73±1.84	61.15±2.33	52.69±4.23	61.58±4.62	51.37±3.84	39.79±3.97	60.77±2.84
DARE-TA	70.61±0.22	64.18±1.24	58.04±8.19	65.39±7.03	56.76±7.01	46.75±5.73	64.51±3.81
Win rate of MAP over baseline methods(↑)	2	3	4	5	6	7	8
TIES-merging	75±0.62	60±0.33	70±7.06	75±6.54	65±6.61	65±5.83	70±3.42
DARE-TIES	90±44.43	80±50.26	75±47.02	90±44.43	65±48.94	60±51.30	75±47.02
TA	65±30.78	100±41.04	100±44.43	100±30.78	100±48.94	100±50.26	100±44.43
DARE-TA	65±48.94	100±0	100±0	100±0	100±0	100±0	100±0

Clarifications

The nested-merging MAP (NMMAP, Alg 3) does not give a complete Pareto front. We apologize for any confusion and we make it clear in the revised version that we only recommend using NMMAP when a complete Pareto front is not needed and when the user already has a preference vector in mind, and when the number of tasks are very high and computational resource given are very limited. We conducted more comprehensive experiments to compare merging 8 models using nested-merging MAP and MAP, the results can be found in the 1-page PDF.

For example, when the number of tasks is N, estimate the surrogate model in MAP (Alg 1) with N degree of freedoms in scaling coefficient vectors could still be costly. The magnitude of number of evaluation is $O(N2^N)$. However, when it comes to NMMAP, the complexity is $O(TNlog(N))$ where T is the number of merged models needs to be evaluated when number of task is 2. We use $T = 20$ in practice. More detailed analysis is in the appendix B but we will demonstrate for individual reviewers' rebuttal on this part. What is the cost? Since NMMAP is path dependent, the solution it get is not global. It estimates the 2-task-merging surrogate model well $(R^2 : 0.95)$ while 300 set of scaling coefficient vectors might not be able to estimate the surrogate model that well ($R^2 : 0.87$). Thus, mixing the positive side and negative side, sometime it might surpass the MAP but on average slightly lower than MAP.

Thanks to the reviewer's suggestions, we have added more related work to our paper. We have already incorporated reviewers' feedback and integrated all changes into the revised manuscript, available at https://anonymous.4open.science/r/MAP-NEURIPS/revised_paper.pdf.

PDF: pdf

Correction of the win rate std of the experiment results

Official Comment Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) 07 Aug 2024, 18:51 (modified: 08 Aug 2024, 21:17)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors Revisions

Comment:
Sorry for the miscalculation of the std of the experiment results. The previous std is not correct. Please kindly refer to the following corrected table. Thank you!

Experiment results with the correction of win rate std

Preference weighted sum(↑)	2	3	4	5	6	7	8
MAP	70.70±0.21	69.05±1.41	72.84±1.05	77.31±0.83	74.26±0.52	73.40±-0.14	72.96±0.73
Single task models	75.84±1.76	77.03±1.84	82.43±4.40	87.69±4.50	88.52±4.02	89.26±3.58	90.62±2.52
MTL	73.63 ± 0.30	75.13 ± 1.00	80.10 ± 2.79	84.93 ± 3.58	86.78 ± 2.94	87.40 ± 2.56	89.11 ± 2.36
Model soups	67.79±1.46	64.25±2.15	66.04±3.22	67.01±3.42	63.11±1.99	63.35±2.17	64.36±2.77
TIES-merging	69.30±0.33	67.60±0.58	71.79±2.93	76.49±3.10	73.74±2.96	72.54±2.87	72.24±1.91
DARE-TIES	67.62±1.65	66.49±2.34	71.39±4.45	74.55±4.55	73.34±4.10	71.43±3.84	71.89±2.86
Task Arithmetic	70.73±1.84	61.15±2.33	52.69±4.23	61.58±4.62	51.37±3.84	39.79±3.97	60.77±2.84
TA with preference as weights	69.22±1.4	66.88±2.37	68.73±5.48	71.92±5.5	68.13±4.69	68.14±4.2	68.17±2.89
DARE-TA	70.61±0.22	64.18±1.24	58.04±8.19	65.39±7.03	56.76±7.01	46.75±5.73	64.51±3.81

Win rate of MAP over baseline methods(↑)	2	3	4	5	6	7	8
MAP vs Model soup	90±07	100±0	100±0	100±0	100±0	100±0	100±0
MAP vs TIES-merging	75 ± 9.80	60 ± 10.93	70 ± 10.49	75 ± 9.39	65 ± 10.85	65 ± 11.66	70 ± 10.25
MAP vs DARE-TIES	90 ± 6.81	80 ± 9.11	75 ± 9.87	90 ± 6.70	65 ± 10.59	60 ± 11.38	75 ± 9.70
MAP vs TA	65 ± 10.91	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0
MAP vs TA with preference as weight	85±08	85±08	100±0	100±0	100±0	100±0	100±0
MAP vs DARE-TA	65 ± 10.64	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0

Official Review of Submission8794 by Reviewer R2T3

Official Review Reviewer R2T3 15 Jul 2024, 17:07 (modified: 25 Sept 2024, 11:58)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer R2T3 Revisions

Summary:
This paper introduces MAP, a low-compute algorithm aimed at identifying the Pareto front of solutions reflecting trade-offs between multiple tasks. Instead of relying on computationally expensive methods for multi-objective optimization, MAP approximates the evaluation metrics of different tasks using a quadratic approximation surrogate model derived from a pre-selected set of scaling coefficients. This allows for amortized inference and significantly reduces computational complexity. The authors further enhance MAP with a Bayesian adaptive sampling method and a nested merging scheme to further reduce computational cost. Experiments on various tasks, including image classification and language modeling, demonstrate that MAP accurately identifies the Pareto front and outperforms brute-force search methods, especially with an increasing number of tasks by nested merging scheme.

Soundness: 3: good
Presentation: 3: good
Contribution: 2: fair
Strengths:
1. The works is theoretically grounded.
2. The paper is well-written and easy to follow. The authors effectively utilize figures and algorithms to illustrate their method and results.
3. The paper provides a well-motivated and technically sound approach. The authors clearly explain the rationale behind their design choices, including the use of quadratic approximation, Bayesian adaptive sampling, and nested merging.

Weaknesses:
1. One of the main contribution of the paper is to obtain the approximate surrogate model which can be used to estimate the pareto front efficient with few actual evaluations. However, this only hold true for two task task, when performing the nesting merging there is no guarantee that the optimal model found via optimal c for pairwise models would result in a good pareto front. Is there a way to test that
2. Comparison with some specific methods that try to account for interference/conflict between model/objective of different tasks. For example, TIES-merging attempts to try to address this problem, hence it seems like a baseline that should be compared with. I understand that finding the pareto front is a different setting than what ties-merging operates in however it would be nice to see where it falls on the pareto front plot. Because if ties-merging lies at the optimal point then the number of eval combinations needed might be reduced drastically although I doubt that ties would be that good.
3. Novelty of the work is limited. Applying various strategies to estimate the pareto front for merging models is new thing however this topic has been extremely well studies in multitask learning literature and at many other places. For example, the current paper seems to follow the section order, plot style etc from the paper "Pareto Multi-Task Learning" (https://arxiv.org/pdf/1912.12854). Hence, I believe the original contribution of the paper is limited in that regard even though the application for which the perform experiments is new.

Questions:
1. I would like it if the author to differentiate the novelty of their methods when compared with other MOOP methods.
2. Table 4 should also contain methods like ties merging etc which try to directly address the issue of interference.
3. If two methods are equivalent, then would the win rate be 50%, if yes then I would like some clarifications on why first 4 rows of Table 2 have close to 50% win-rate?
4. L410 typo and missing citation.

Limitations:
N/A

Flag For Ethics Review: No ethics review needed.
Rating: 4: Borderline reject: Technically solid paper where reasons to reject, e.g., limited evaluation, outweigh reasons to accept, e.g., good evaluation. Please use sparingly.
Confidence: 4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.
Code Of Conduct: Yes

Rebuttal by Authors

Rebuttal by Authors

Rebuttal | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 07 Aug 2024, 06:33 (modified: 07 Aug 2024, 09:07)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors | Revisions

Rebuttal:

We thank the reviewer for these comments and questions. We appreciate that the reviewer recognized the strengths of MAP. Below we give a detailed reply to the reviewer's questions.

W1: Testing nested merging performance

Thank you for this insightful question. The solution found by nested-merging MAP (NMMAP) is indeed suboptimal given the limited search space compared with merging all models at once. However, when # of task is high, it has comparable performance to MAP while requiring much fewer evaluations.

To empirically test the performance of NMMAP algorithm, we conducted additional experiments merging 8 models using NMMAP and MAP and various baseline methods. We evaluated the solutions using a set of 20 preferences vectors. We then computed the sum of preference-weighted accuracies of all tasks. We also computed the win rate (percentage of preferences where the weighted sum of MAP is higher than those of baseline methods).

The results can be summarized below:

	Preference weighted sum(↑)	WinrateofMAP(↑)
Single-taskmodels	90.05±3.02	
MAP-nested	67.05±3.89	
MAP-plain	**72.12**±3.78	
TIES-merging	70.79±3.81	1.00±0.00
DARE-TIES	70.51±3.58	0.90±0.32
TaskArithmetic(TA)	59.44±5.68	1.00±0.00
DARE-TA	63.14±4.91	1.00±0.00

In general, we find that NMMAP can achieve comparable performance to MAP and outperforms TA and DARE TA, though underperforms TIES-merging and DARE-TIES.

Theoretically, NMMAP finds the suboptimal solution compared to the MAP because the solution it finds depend on the order of merging. The surrogate model can be easily fitted *when the number of tasks is low*. When it is high (e.g. 8), the fit of the surrogate model can *no longer* be near-perfect approximation (please find the R^2 in Table 2). The main advantage of NMMAP lies in achieving comparable performance while taking much fewer computation of evaluations when the number of tasks is high.

W2 & Q2: comparison with more baseline methods

Thank you for the valuable suggestions. We compared MAP with various existing model-merging methods, including TIES-merging (as suggested by you), along with Task Arithmetic with a single scalar, Task Arithmetic with preferences as scalars, DARE combined with TIES-merging, DARE combined with Task Arithmetic, and SLERP. For all the methods, we used the hyperparameters based on recommendation of the original paper.

When the number of tasks is 2, we draw 3 Pareto fronts and all the merging methods performance we have tested on the same figure. Visualizations of the results for dimension 2 can be found in Figure 1 in the 1-page PDF file. From the results, we can see that TIES-merging and other merging methods either lie on the Pareto front found by MAP or can only dominate very little proportion of the Pareto front. This evidence further confirms the usefulness of MAP as a way of finding diverse optimal solutions.

For higher dimensions, we evaluated the solutions found by MAP and baseline methods using a set of 20 preferences vectors. We then computed the sum of preference-weighted accuracies of all tasks. We also computed the win rate (percentage of preferences where the weighted sum of MAP is higher than those of baseline methods). The detailed table results are presented in the general response to reviewers.

W3 & Q1: comparison to other MOOP methods

Thank you for the great question. We indeed have cited the paper "Pareto Multi-Task Learning"[1] in our submission as a related work in finding Pareto fronts in the MTL literature, and our plotting styles were inspired by the paper "Multi-Task Learning as Multi-Objective Optimization"[2]. However, our method is distinct from this line of work from MTL literature because our method is not gradient based, i.e. we don't need to calculate any first order information to approximate the surrogate model. Task vector-based model merging aims to combine multiple finetuned models with the same pretrained checkpoint to obtain a multi-task model. This operation does not require any gradient-based training. More novelty and contributions of MAP are summarized in the general response to reviewers.

Q3: clarification on win rate

Thank you for the clarifying question. In section 4.3, we would like to show that MAP is capable of finding a good Pareto front. We compare the performance of it with the direct search method. When the number of tasks N is low, (when N is 2, 3 4), we regard the Pareto front found by the direct search method as the ground truth Pareto front because the search space is not intractable. For example, in the case of # tasks = 2, we used 200 points of evaluation metric (pts) and thus each dimension of search space would have $\sqrt{2}200 = 14.14$ pts which we claim is enough to search for the ground truth Pareto front. We include a visualization to illustrate in the 1-page PDF-file Figure 3.

Thus, *when N is 2, 3, 4*, we compare with the direct search which we regard as ground truth and reaches the near 50% winning rate which means MAP is able to find near ground truth Pareto front when the number of tasks is low.

When the number of tasks N is high (5, 6, 7, 8), we argue that because of the curse of dimensionality, the direct search method would no longer find the ground truth Pareto fronts. Thus, the winning rates are significantly higher than 50%. These experiments give us a sense that implementing MAP on a high dimension would find a Pareto front to a much better winning rate compared to direct search in the grid.

Conclusion

We hope our detailed response and added experiments better explain MAP's contribution. **We have incorporated reviewers' feedback and integrated all changes into the revised manuscript, available at https://anonymous.4open.science/r/MAP-NEURIPS/revised_paper.pdf**.

If any remaining questions/concerns make you hesitate to raise the score, we would be grateful if you let us know so we could further improve our work.

Replying to Rebuttal by Authors

Official Comment by Reviewer R2T3

Official Comment | Reviewer R2T3 | 13 Aug 2024, 00:44 | Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors

Comment:

Thank you to the authors for their rebuttal. I have read through the rebuttal and for Q1 it seems like TIES / DARE-TIES are better than NMMAP and for Q2 as well seems like TIES point lead to better performance in most cases, moreover, It would have been great to see the pareto front for the TIES merging method as well. Hence, I am not sure that the nested method works well for more tasks. Moreover, I still feel that this paper is extremely similar to "Pareto Multi-Task Learning" I agree with the authors that they do not need a gradient based estimation but I would say that is because of using model

"Pareto Multi-Task Learning" I agree with the authors that they do not need a gradient based estimation but I would say that is because of using model merging as opposed to some new innovation compared to "Pareto Multi-Task Learning" paper. They work with multitask training and need to train models etc but we do not need to train new models for merging and hence do not need gradient estimates even for obtaining the pareto front. Hence I still think that there is not much distinction between the two papers. Because of limited novelty compared to the paper I mentioned and my concerns on using NMMAP for more tasks, I would keep my score.

Clarifying reviewer's remaining concerns

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 13 Aug 2024, 01:43 (modified: 13 Aug 2024, 01:49)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer R2T3 | Revisions

Comment:

We thank the reviewer for expressing their remaining concerns. Below are our detailed replies to your concerns. We hope this would help clarify some misunderstandings.

TL;DR:

- **Response 1: Use MAP if you care more about quality rather than compute. Compute needed: existing baseline for model merging (direct search) ($O(TN2^N)|T=60s$) >> MAP ($O(TN2^N)|T=10^{-5}s$) >> NMMAP ($O(N\log N)$)**
- **Response 2: TIES-merging has been dominated by MAP in all number of tasks scenarios, ranging from 2 to 8.**
- **Response 3: There is not a way to calculate the TIES merging Pareto front because there are no controls from practitioners in this algorithm.**
- **Response 4: Comparing the algorithm of Pareto MTL and ours, they are completely different algorithms. There is no way to utilize Pareto MTL on model-merging without calculating gradients.**
- **Please kindly see the detailed reasoning of every response below.**

Response 1: for Q1 it seems like TIES / DARE-TIES are better than NMMAP and for Q2 as well seems like TIES point leads to better performance in most cases

TIES / DARE-TIES are better than NMMAP because **NMMAP is a method we put forward to further reduce the computation cost**. The whole theme of our algorithm is based on "how to get Pareto front with limited compute". If the practitioners care more about quality, they should consider using the MAP algorithm rather than NMMAP. We would like to highlight that NMMAP further reduces the computation cost compared to MAP, while MAP already reduces a lot of computing cost and only uses CPU-level computation compared to existing ways of computing Pareto front for model merging (direct search). We don't think the Pareto MTL algorithm has the same idea with us because one cannot implement Pareto MTL in the model-merging scenario (see more discussion about this in **Response 3**). Without MAP, a huge computational cost for fine-tuning the entire model is needed. Please refer to the following analyze.

Why is evaluating models computationally expensive?

To get the Pareto front of tasks which have trade-offs. We need to have a lot of data points to show the trade-off. Here, each data point is the evaluation metrics of a model. To get the ground truth evaluation metric of a task (e.g. let's say classification), we need to run the evaluation script of the model to determine the metric. If we have 4 tasks, and we set 5 possible values (let's say 0.2, 0.4, 0.6, 0.8, 1) for the scaling coefficient vector of each model specialized for one task. We will have to evaluate $5^4=625$ models on all 4 tasks. Assuming each evaluation takes 1 minute, evaluating all the models on all the 4 tasks will take 625 models x 4 tasks x 1 minute=2500 minute = 41.7 hours which is expensive. In big O notation, assuming we have T tasks, the time of evaluation for each task is T. the time (computational) complexity is $O(TN2^N)$.

Why does MAP (quadratic approximation as surrogate model) have a lower cost?

For the MAP algorithm (alg 1), the time complexity is the same as what I mentioned above, what is different is that we fitted an approximation of the evaluation metrics. We only need the scaling coefficient vectors to input to a quadratic function model to be able to get the estimated evaluation score. Running the quadratic function once takes only $3.91\times10^{-6}s\pm8.94\times10^{-9}s$. And thus, evaluating 2500 times takes $<2500\times4\times10^{-6}s<10{-}2s$.

Response 2: Q2 as well seems like TIES point lead to better performance in most cases

TIES-merging has been dominated by MAP in all number of tasks scenarios, ranging from 2 to 8.

Preference weighted sum(↑)	2	3	4	5	6	7	8
MAP	**70.70±0.21**	**69.05±1.41**	**72.84±1.05**	**77.31±0.83**	**74.26±0.52**	**73.40±-0.14**	**72.96±0.73**
Single task models	75.84±1.76	77.03±1.84	82.43±4.40	87.69±4.50	88.52±4.02	89.26±3.58	90.62±2.52
MTL	73.63 ± 0.30	75.13 ± 1.00	80.10 ± 2.79	84.93 ± 3.58	86.78 ± 2.94	87.40 ± 2.56	89.11 ± 2.36
Model soups	67.79±1.46	64.25±2.15	66.04±3.22	67.01±3.42	63.11±1.99	63.35±2.17	64.36±2.77
TIES-merging	69.30±0.33	67.60±0.58	71.79±2.93	76.49±3.10	73.74±2.96	72.54±2.87	72.24±1.91
DARE-TIES	67.62±1.65	66.49±2.34	71.39±4.45	74.55±4.55	73.34±4.10	71.43±3.84	71.89±2.86
Task Arithmetic	70.73±1.84	61.15±2.33	52.69±4.23	61.58±4.62	51.37±3.84	39.79±3.97	60.77±2.84
TA with preference as weights	69.22±1.4	66.88±2.37	68.73±5.48	71.92±5.5	68.13±4.69	68.14±4.2	68.17±2.89
DARE-TA	70.61±0.22	64.18±1.24	58.04±8.19	65.39±7.03	56.76±7.01	46.75±5.73	64.51±3.81

Continual

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 13 Aug 2024, 01:43 (modified: 13 Aug 2024, 01:45)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer R2T3 | Revisions

Comment:

Response 3: It would have been great to see the Pareto front for the TIES merging method as well.

We wish there was a way to calculate the TIES merging Pareto front. Unfortunately, there is no control left for the practitioner in the TIES merging left for the practitioner to calculate Pareto front (https://arxiv.org/abs/2306.01708, please check the algorithm 1, page 4). To be specific, the controls in DARE-TA and TA are the scaling coefficients. The controls in the model soup algorithm are the weights of the models. However, there is nothing that can be changed by practitioners in the TIES merging. **The scaling coefficient for every weight/parameter of the model is determined by the algorithm. And each parameter has a different scaling coefficient.** Thus, we cannot calculate the Pareto front of it.

Response 4: I still feel that this paper is extremely similar to "Pareto Multi-Task Learning".

We appreciate the reviewer's observation regarding similarities with 'Pareto Multi-Task Learning'. While both approaches aim to find the Pareto front, MAP introduces a fundamentally different methodology that leverages existing fine-tuned models rather than training from scratch. **The key innovation in MAP** lies in its ability to **utilize pre-existing fine-tuned models, which is particularly valuable given the vast number of such models available on platforms like Hugging Face**. **This approach significantly reduces computational requirements and allows for rapid exploration of the Pareto front.** We want to emphasize that the methodologies of MAP and Pareto MTL are fundamentally different: Pareto MTL requires gradient-based updates

to model parameters during training. (**https://arxiv.org/pdf/1912.12854, Please kindly see Algorithm 1 of Pareto MTL and compare with Algorithm 1 of MAP**) MAP employs a surrogate model and can approximate the entire Pareto front using only CPU-level compute, without the need for additional training or gradient computations.

Replying to Continual

Official Comment by Reviewer R2T3

Official Comment | Reviewer R2T3 | 13 Aug 2024, 02:03

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer R2T3

Comment:

Some quick and my last comments. After this I guess I have nothing to add I will let the AC/SAC decide the conclusion of this discussion.

Response 1: "The whole theme of our algorithm is based on “how to get Pareto front with limited compute”." -> I am also not sure of the utility of obtaining an approximate pareto front using NMMAP when one of the methods outperforms it while using much fewer evaluations. Response 2: "TIES-merging has been dominated by MAP in all number of tasks scenarios, ranging from 2 to 8." -> are these comparison compute equivalent? Response 3: "There is not a way to calculate the TIES merging Pareto front because there are no controls from practitioners in this algorithm." -> Similar to TA and DARE, ties also has a scaling parameter (check section 4 last line, or alg 1 second last line), it also has the density "k" which can be used.

Official Comment by Authors

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 13 Aug 2024, 03:23 (modified: 13 Aug 2024, 04:06)

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer R2T3 | Revisions

Comment:

We sincerely thank the reviewer for engaging this discussion. Below are our detailed replies: TL;DR:

- Response 1: **Ties-merging does not always dominate NMMAP. Winning ratio: 9/20 (=0.45). Leaving it as a low-compute choice based on a user preference does not hurt.**
- Response 2: **MAP is put forward as a plugin for any eligible task vector based model merging method. The goal is to compute a Pareto front which is bound to have multiple rounds of model-merging. It is not fair to compare a Pareto front calculating alg with a single shot model merging alg in the sense of compute**.
- Response 3: **λ in the TIES-merging cannot be separated as task-wise, since their task vector τ_m is unified and cannot be separated to $\sum_{t=1}^{n} \lambda_t$. Besides, we searched both λ and "k" and pick the best combination. TIES-merging is very sensitive to them. Not choosing the optimal ones leads to performance collapse.**

Response 1: Utility of obtaining an approximate Pareto front using NMMAP when one of the methods outperforms it

We thank the reviewer for the question. In fact, if one wants to use a better performing algorithm on average, MAP should be the choice. **However, as for ties-merging vs NMMAP, having a higher average score does not mean always domination.** NMMAP is a low-compute version of MAP. We emphasize that there is no guarantee that ties-merging are bound to be better than NMMAP. Below is the winning ratio data between NMMAP and other baselines. Even if NMMAP on average is worse than ties-merging, **the winning ratio is significantly near 50% (20 exps)**:

	Winrate(↑)
nested vs TIES	0.45
nested vs TA	**0.80**
nested vs DARE-TIES	0.45
nested vs DARE-TA	**0.70**

In conclusion, when a practitioner with preferences on different tasks and would like to merge the corresponding models. **She/he can always try first TIES-merging for a single shot and compare with NMMAP. If NMMAP is dominated by TIES-merging in the tasks and the practitioner still want a Pareto front to decision how to merge the models. She/he can utilize MAP. After all, these methods are not exclusive.**

Response 2: "TIES-merging has been dominated by MAP in all number of tasks scenarios, ranging from 2 to 8." -> are these comparison compute equivalent?

MAP is put forward as a plugin of model merging algorithms as long as the algorithm allows the users to have some controls corresponding to the preference (e.g. the scaling coefficient of task vectors in TA/DARE-TA). TIES-merging could only output one single model, while MAP outputs an estimation of Pareto front. There are scenarios that practitioners need a Pareto front to decide the scheme of merging models according to different preferences. Considering the fact that MAP is an algorithm that outputs Pareto fronts, it is not fair to compare a model-merging algorithm with a Pareto front finding model-merging algorithm. MAP contributes to reduce the computation complexity of calculating Pareto front in model-merging.

Response 3: Similar to TA and DARE, ties also has a scaling parameter (check section 4 last line, or alg 1 second last line), it also has the density "k" which can be used.

We would like to point out that **unlike** TA and DARE, the scaling parameter for TIES is for the **merged model as a whole** instead of for **individual task vectors** and hence **cannot be used for controlling the preference over tasks**. We searched for hyperparameters of "k" and λ. They are the hyperparameters that need to be optimized, rather than some controls that could lead to trade-off performance on different tasks. We tried different combinations of "k" and λ (searching step = 0.1). We end up using $\lambda = 1$ and $k = 20$ recommended by the original paper (P22, C.4) because this combination gives the best results and dominates other combinations. We would also like to point out that TIES-merging is **very sensitive to the hyper-parameters**. If we don't use this combination, their performance collapse quickly and dominated by both MAP and NMMAP.

Specifically, in Algorithm 1 in TIES-merging (https://arxiv.org/pdf/2306.01708), τ_t is the task vector for task t, they obtain $\tau_m^p = \frac{1}{|\mathcal{A}^p|} \sum_{t \in \mathcal{A}^p} \hat{\tau}_t^p$ for p in $1, \ldots, d$ and obtain the merged model as

$$\theta_m \leftarrow \theta_{\text{init}} + \lambda * \tau_m.$$

Note that here λ **is a scaling factor for the merged task vectors as a whole**, and its value needs hyperparameter search, and the original TIES-merging paper recommends to use $\lambda = 1$. (τ_m **cannot be separated to a sum of different task vector from different tasks.**)

Unlike in task arithmetic where the merged model is

$$\theta_m \leftarrow \theta_{\text{init}} + \sum_{t=1}^{n} \lambda_t * \tau_t$$

where the users can control the preference of task t by setting λ_t, the λ in TIES-merging is not task specific.

Official Review of Submission8794 by Reviewer V9hH

Official Review | Reviewer V9hH | 11 Jul 2024, 04:08 (modified: 25 Sept 2024, 11:58)

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer V9hH | Revisions

Summary:

The paper follows a series of model merging works based on task vectors, i.e. differences between models fine-tuned on specific tasks and their pretrained starting points. Such works usually obtain a multi-task model by simply doing a sum of the so defined vectors weighted by some coefficients. This paper focuses on choosing these coefficients so to obtain a Pareto front of models where each model works best on a specific task while maintaining satisfying performance on the others. The main contribution of the paper is to make this feasible by approximating the evaluation metrics with a surrogate model. The authors then propose to use a Bayesian sampling scheme together with a multi-stage nested merging schedule to increase the efficiency of the pipeline. The work considers both vision and natural language domains, using CLIP-style models on different vision datasets and LLaMa 3 on different languages.

Soundness: 3: good
Presentation: 2: fair
Contribution: 2: fair

Strengths:

- The work tackles an interesting and timely problem, as the number of fine-tuned models grows by the day and re-using them without incurring in huge costs becomes more and more a key priority.
- The proposed approach is novel and makes intuitive sense, offering a principled way to ammortize the burdening cost of evolutionary model merging. While the convex Pareto Front assumption may be limiting, the approach succeeds at rendering feasible a procedure that would otherwise be extremely cost.
- The paper is sound, both from a mathematical perspective and empirically. The experiments consider different datasets and domains.

Weaknesses:

- There is an overall lack of baselines: it is important to see whether the obtained models perform on par or better than those obtained with non-evolutionary methods such as TIES [1], DARE [2] or SLERP. Most importantly, there is no comparison with the multi-task model obtained using standard Task Vectors. While it is true that these approaches do not consider user preferences, it might just as well be that they work better overall and render the proposed framework basically useless.
- The nested scheme is not convincing, as it is not clear why the final obtained solutions would be in the Pareto Front of the N tasks if the algorithm considers pair-wise Pareto fronts with an arbitrary matching criterion such as having similar loss values. Indeed, the results in Table 4 don't seem too promising.
- Since the only considered baseline is brute force, it is hard to assess how much of the benefits are due to the proposed machinery or to the evolutionary algorithm being used. It would be interesting to see how much the obtained Pareto Front differs from one obtained from using a standard evolutionary algorithm without the surrogate model.
- Model merging is simplified as task vectors and follow-ups in the background, totally neglecting a vast corpus of literature on permutation-based approaches such as Git Re-Basin [3] etc.
- The work's clarity could be improved, both in the tables/figures and in the text, where typos abound.
  - Figure 4: the caption for (b) seems to refer to a different figure.
  - Table 3 is extremely messy and hard to read.
  - Table 7 is not intelligible, I advise finding another way to visualize that information

[1] Yadav, Prateek, et al. "Ties-merging: Resolving interference when merging models." Advances in Neural Information Processing Systems 36 (2024).

[2] Yu, Le, et al. "Language models are super mario: Absorbing abilities from homologous models as a free lunch." Forty-first International Conference on Machine Learning. 2024.

[3] Ainsworth, Samuel K., Jonathan Hayase, and Siddhartha Srinivasa. "Git re-basin: Merging models modulo permutation symmetries." NeurIPS. 2023.

Questions:

Typos

- L76: a recent work by \cite
- L120: if the evaluation metric spanning to the whole (..)
- Figure 5: "our amortized Pareto front much better than"
- Conclusions: "which is very easy to directly combined", "bad perform results"

Doubts

- why do c and v have different indexes in Algorithm 1?
- in the Pareto dominance definition, `f(x)` is a loss function and not a value function, right?

Limitations:

Yes

Flag For Ethics Review: No ethics review needed.
Rating: 5: Borderline accept: Technically solid paper where reasons to accept outweigh reasons to reject, e.g., limited evaluation. Please use sparingly.
Confidence: 4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.
Code Of Conduct: Yes

Rebuttal by Authors

Rebuttal | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 07 Aug 2024, 07:05 (modified: 07 Aug 2024, 09:07)

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors | Revisions

Rebuttal:

We have already **incorporated reviewers' feedback, integrated all changes and added more related work** into the revised manuscript, available at https://anonymous.4open.science/r/MAP-NEURIPS/revised_paper.pdf Please kindly feel free to check it. We thank the reviewer for these comments and questions. We appreciate that the reviewer recognized the strengths of MAP.

W1: experiments on more baselines

Thank you for the suggestions! We have added more experiments comparing MAP with various existing model-merging methods, including TIES-merging, along with Task Arithmetic with a single scalar, Task Arithmetic with preferences as scalars, DARE combined with TIES-merging, DARE combined with Task Arithmetic, and SLERP. For all the methods, we used the hyperparameters based on recommendation of the original paper.

When the number of tasks is 2, we draw 3 Pareto fronts and all the merging methods performance we have tested on the same figure. Visualizations of the results for dimension 2 can be found in Figure 1 in the 1-page PDF file. From the results, we can see that TIES-merging and other merging methods either lie on the Pareto front found by MAP or can only dominate very little proportion of the Pareto front. This evidence further confirms that MAP is a well-performing method useful for finding diverse optimal solutions.

For higher dimensions, we evaluated the solutions found by MAP and baseline methods using a set of 20 preferences vectors. We then computed the sum of preference-weighted accuracies of all tasks. We also computed the win rate (percentage of preferences where the weighted sum of MAP is higher than those of baseline methods). The detailed table results are presented in the general response to reviewers.

W2: Performance of nested merging

W2: Performance of nested merging

We apologize for the confusion. We would like to clarify that the solution found by the nested-merging MAP (NMMAP) is not guaranteed as the optimal solution of the Pareto front. We will make sure to emphasize this in the camera-ready version. The solution found by nested-merging MAP is suboptimal since the solution found depends on the order of merging models.

In theory, the surrogate model can be easily fitted *when the number of tasks is low*. When it is high (e.g. 8), the fit of the surrogate model can *no longer* be near-perfect approximation (please find the R^2 in table 2). We understand in general that NMMAP does not find the global optimum, but please kindly keep in mind that even if their performance is comparable, NMMAP takes way less computation of evaluation.

To empirically test the performance of NMMAP algorithm, we conducted additional experiments merging 8 models using NMMAP and MAP and various baseline methods. We evaluated the solutions using a set of 20 preferences vectors. We then computed the sum of preference-weighted accuracies of all tasks. We also computed the win rate (percentage of preferences where the weighted sum of MAP is higher than those of baseline methods).

The results can be summarized below:

	Preferenceweightedsum(↑)	WinrateofMAP(↑)
Single-taskmodels	90.05±3.02	
MAP-nested	67.05±3.89	
MAP-plain	**72.12**±3.78	
TIES-merging	70.79±3.81	1.00±0.00
DARE-TIES	70.51±3.58	0.90±0.32
TaskArithmetic(TA)	59.44±5.68	1.00±0.00
DARE-TA	63.14±4.91	1.00±0.00

As for the loss values, we apologize that we didn't mention the reason why we use the loss values as signals to merge the model. Our empirical study shows that tasks with similar difficulty would have clear trade-offs, if we merge a hard task (~~accuracy 70%) with an easy task (~~accuracy 99%), it is very likely that the weight of the task vector of easy task is not important at all. Thus, using the validation loss to represent the difficulties might be helpful. However, the comparison between them depends highly on the tasks, and sometimes the advantage of doing this is not significant. Thus, we didn't emphasize it as an important contribution and reserves for future study.

W3: compare with an algorithm without using surrogate model

We thank the reviewer for the question! In fact, directly applying the evolutionary algorithm to our setting without surrogate model is nearly infeasible due to the high computational cost. For example, using NSGA III algorithm, we start with a population of size 100 and use 150 generations. When there are two tasks, this requires 150 * 100 * 2 = 3,0000 evaluations of the merged models. However, as one evaluation typically costs ~ 1min, this could take as much as 500 hours, which is very impractical.

Please also note that when the number of tasks is low (e.g. 2,3), we regard the Pareto front found by the direct search method as the ground truth Pareto front because the search space is not intractable. For example, in the case of # tasks = 2, we used 200 points of evaluation metric (pts) and thus each dimension of search space would have $\sqrt{200} \approx 14.4$ pts which we claim is enough to search for the ground truth Pareto front.

W4: should have discussed another line of model merging such as Git-Rebasin

Thank you for the suggestion! We didn't compare our method to Git Re-Basin because their study focuses on the scenarios of merging the models from different initialization, whereas our background works on the same initialization of different fine-tuned models. However, it is indeed related to our work and we have included disucssion about it in the related work.

We have included the following work in our related work in the revised version pdf (anonymous.4open.science/r/MAP-NEURIPS/revised_paper.pdf) and will also included them in the camera ready version as well.

We include in "Task arithmetic" related work:

"Different from task arithmetic, Ainsworth et al. shows that even if the models are not finetuned by the pretrained model, practitioner could still merge the models by permutation of the model parameters to enclose the models."

Continual of rebuttal

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 07 Aug 2024, 08:06 (modified: 10 Aug 2024, 06:04)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer V9hH | Revisions

Comment:

W5: clarity could be improved

Q: Figure 4: the caption for (b) seems to refer to a different figure.
A: We are deeply sorry for the mistake. We have fixed in the revised version pdf, will also fix it in the camera ready version. The tentative fix is: how the average R^2 value as the quality of fitting the surrogate model change as more merged model is evaluated. (a) showing the case of number of task is 2 and 3.(b) showing the case of number of task is 4,5,6 and 7.
Q: Table 3 is extremely messy and hard to read.
A: We apologize for the confusion, this table shows the supremacy of Bayesian MAP (alg 2) comparing to the MAP (alg 1). We showed that if we only allow 12 set of scaling coefficient vectors to be sampled to mimic the performance of MAP under extreme low compute cases (usually we need about 20), Bayesian MAP is usually better than MAP. We run the Bayesian MAP for multiple iterations. Initially, we sample 6 scaling coefficient vectors and evaluate them. In iteration 1, we added 3 more scaling coefficient vectors and evaluate them. These 3 scaling coefficient vectors are sampled according to the uncertainty of the surrogate model fitting. Basically, we would sample more scaling coefficient vectors on where the fitting uncertainty is higher. From table 4, we can see BMAP iteration 1 (6+3) performs better than directly initialize with sampling 9 scaling coefficient vectors. Similarly, BMAP iteration 2 (6+3+3) performs better than directly initialize with sampling 12 scaling coefficient vectors.

Q: Table 7 is not intelligible, I advise finding another way to visualize that information.
A: We have fixed in the revised version pdf (https://anonymous.4open.science/api/repo/MAP-NEURIPS/file/revised_paper.pdf) **Appendix D, Page 19**, will also fix it in the camera ready version. Since it is a table including the detailed information of the table 4 exps, it is hard for us to post it here. We make a better illustrated version which is available in our anonymous code base as well. Please kindly check here if you are interested.

Q1: Typos

Many thanks for helping us find the typos. We have fixed them in the revised version pdf and we will have a thorough exam before posting the camera-ready version.

Q2: Doubts

- Sorry, that is a typo as well. We have fixed in the revised version pdf, will also fix it in the camera ready version. They should all be N.
- Yes, we will specify this in the camera-ready version. It is also **already fixed** in the revised version pdf (https://anonymous.4open.science/api/repo/MAP-NEURIPS/file/revised_paper.pdf). We will do a thorough inspection for the typos when finalizing the

paper. Thank you very much.

Replying to Rebuttal by Authors

Answer to Rebuttal

Official Comment | Reviewer V9hH | 13 Aug 2024, 02:12 | Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors

Comment:

I thank the Authors for their rebuttal.

I am glad to see the missing baselines. I am still not fully convinced, however, of the utility of the approach if the baselines reach similar results without all the added complexity and computation.

"Different from task arithmetic, Ainsworth et al. shows that even if the models are not finetuned by the pretrained model, practitioner could still merge the models by permutation of the model parameters to enclose the models." --> This sentence is confusing and wrong. What does it mean to be fine-tuned by a model? Also, what is permuted are the neurons, i.e. the rows of the weight matrices, not the individual parameters themselves.

Given the more thorough evaluation and trusting the authors will improve the clarity of the paper, I am raising my score from 4 to 5.

Official Comment by Authors

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 13 Aug 2024, 03:21 (modified: 13 Aug 2024, 03:22)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer V9hH | Revisions

Comment:

We thank the reviewer for expressing their remaining concerns.

Response 1: utility of the approach

We would like to point out that while MAP (based on task arithmetic) outperforms its based method (task arithmetic) for over 20%. Though **MAP (based on task arithmetic) outperforms TIES-merging by 1.4%**, we would like to point out that MAP is **a general plug-in** for task-vector based model merging methods where the practitioners have a way of controlling the scaling coefficients for different tasks. **Please also kindly notice that TIES-merging also outperforms task arithmetic for about 20% which is where MAP (based on task arithmetic) improved from.** If there are other better task-vector based model-merging method allowing controlling the scaling coefficients are out in the future, MAP can also serve as a plugin to help practitioner get a Pareto front and make decisions.

Clarity:

We apologize for the confusion and would like to rephrase it as follows: "Different from task arithmetic based approaches, Ainsworth et al. show that even if the models are not finetuned from the same pretrained model, practitioners could still merge the models by permutations of the rows and columns of the weight matrices." We have updated the corresponding part in the revised manuscript Line 490.

Replying to Official Comment by Authors

Looking forward to hearing from you

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 13 Aug 2024, 19:57 (modified: 13 Aug 2024, 19:57)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer V9hH | Revisions

[Deleted]

Official Review of Submission8794 by Reviewer 8WQN

Official Review | Reviewer 8WQN | 09 Jul 2024, 04:07 (modified: 25 Sept 2024, 11:58)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer 8WQN | Revisions

Summary:

The paper introduce a novel and low-compute algorithm, Model Merging with Amortized Pareto Front (MAP) which efficiently identifies a Pareto set of scaling coefficients for merging multiple models to reflect the trade-offs. The method approximates the evaluation metrics of the various tasks using a quadratic approximation surrogate model derived from a pre-selected set of scaling coefficients, enabling amortized inference.

Soundness: 3: good
Presentation: 3: good
Contribution: 4: excellent
Strengths:

1. Design quadratic surrogate models to approximate the evaluation metric functions, which amortize the computation of Pareto fronts;
2. Propose a nested merging scheme to decrease computational complexity from exponential to quadratic;
3. Propose Bayesian adaptive sampling, which efficiently queries the computationally expensive evaluations according to loss information.

Weaknesses:

1. Algorithm 4 has some problems, e.g. figure ??.
2. The variables and objectives in section 3.1 need to be described more detailed. As we know, not all variables in MOEA necessarily involve trade-offs.
3. Is the second-order Taylor expansion related to nesting? I might have a misunderstanding in this part. It would be beneficial to elaborate more on the role of the second-order Taylor expansion.
4. The time complexity in MOEA is a crucial topic for discussion. The nested method used in this article significantly reduces time complexity, which is an intriguing aspect. However, Appendix B needs to be reorganized to better address this, e.g. "...The results are summarized in the table below, which compares the outcomes of merging four models using... " which table? And, a more detailed analysis of the time complexity is essential.

Questions:

Described in weaknesses. I might improve the rating if my questions are well addressed.

Limitations:

Described in weaknesses. The method is intersting, and I might improve the rating if my questions are well addressed.

Flag For Ethics Review: No ethics review needed.
Rating: 7: Accept: Technically solid paper, with high impact on at least one sub-area, or moderate-to-high impact on more than one areas, with good-to-excellent evaluation, resources, reproducibility, and no unaddressed ethical considerations.
Confidence: 4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.
Code Of Conduct: Yes

Rebuttal by Authors

Rebuttal Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) 07 Aug 2024, 07:25 (modified: 07 Aug 2024, 09:07)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors Revisions

Rebuttal:

We have already incorporated reviewers' feedback, integrated all changes and added more related work into the revised manuscript, available at https://anonymous.4open.science/r/MAP-NEURIPS/revised_paper.pdf Please kindly feel free to check it. We thank the reviewer for these comments and questions.

W1: Algorithm 4

We are deeply sorry for the mistake. We will make sure to fix it in the camera ready version. The tentative fix is: Average R^2 value of fitting the surrogate model for various number of dimensions. (a) showing the case of number of task is 2 and 3.(b) showing the case of number of task is 4,5,6 and 7.

W2: clarifications in section 3.1

Thank you very much for the suggestion. Yes, generally not all variables in MOEA necessarily involve trade-offs. In this case, the Pareto front would collapse to few points or even a single point, which means one single merging recipe dominates the rest sampled candidates. In our experiments, we do spot that few tasks pairs does not have very strong trade-off.

W3: role of second-order Taylor expansion

Sure. The second-order Taylor expansion is directly related to the MAP (alg 1). We will improve the explanation in the camera ready version.

The detailed expansion is as follows: Denote $M(\theta)$ as the evaluation metric. The evaluation metrics depend on the validation set and the model to be evaluated. The model depends on the task vectors and scaling coefficient vectors. In our setting, the only variables are the scaling coefficient vectors. $M_n(c) \equiv M(\theta_{\text{merge}}(c)) = M(\theta_{pretrained}) + \nabla M(\theta_{pretrained})^\top(\theta_m(c) - \theta_{pretrained}) + \frac{1}{2}(\theta_m(c) - \theta_{pretrained})^\top H_n(\theta_{pretrained})(\theta_m(c) - \theta_{pretrained}) + R_n(\theta_m(c) - \theta_{pretrained})$ where $M(\theta_{pretrained})$ is the constant term, we define it as e.

It is fixed given the pretrained model. $\nabla M(\theta_{pretrained})^\top(\theta_m(c) - \theta_{pretrained})$ is the first order gradient term. $R_n(\theta_m(c) - \theta_{pretrained})$ is the remainder term which is the approximation error term.

$\theta_m(c) - \theta_{pretrained}$ term can be rewritten as the $V \cdot c$ which is the dot product of the task vector and the scaling coefficient vectors.

Therefore, the term $\nabla M(\theta_{pretrained})^\top(\theta_m(c) - \theta_{pretrained})$ can be represented as $\nabla M(\theta_{pretrained})^\top V \cdot c$.

$M(\theta_{pretrained})^\top$'s shape is $(1, p)$, V's shape is (p, N), where p is the number of parameters of the model which is a huge number. We define vector b s.t. $b^\top \equiv \nabla M(\theta_{pretrained})^\top V$. However, b's shape is $(1, N)$ which is only the number of tasks.

The term, $\frac{1}{2}(\theta_m(c) - \theta_{pretrained})^\top H_n(\theta_{pretrained})(\theta_m(c) - \theta_{pretrained})$,can be re-written as $\frac{1}{2}c^\top V^\top H_t\theta_{pretrained}Vc$.

Similarly, the $V^\top$, $H_t\theta_{pretrain}$, V has the corresponding shapes: (N, p), (p, p), (p, N). Denote $A \equiv V^\top H_t\theta_{pretrain}V$, A's shape is (N, N).

Thus, $M(c) = e + b^\top c + \frac{1}{2}c^\top Ac + R_n(\theta_m(c) - \theta_{pretrained})$. We want to avoid calculating $b^\top$ and A by their definition because we want to bypass the calculation of the gradient term and hessian term. Thus, we would set them as the parameters of the surrogate model.

For example, if the number of tasks is 2. In this case, $c = (c1, c2)$, $M(c) = e + b_1 \cdot c_1 + b_2 * c_2 + A_{11}c_1^2 + A_{12} \cdot c_1 \cdot c_2 + A_{22} \cdot c_2^2$, the surrogate model has 6 parameters: $(e, b_1, b_2, A_{11}, A_{12}, A_{22})$. If there are N tasks, then c is an N-dimensional vector, the number of parameters of the surrogate model is $1 + N + (N^2 - N)/2 + N = (N + 1)(N + 2)/2$.

W4: time complexity of MOEA

Thank you for pointing that out. We will definitely fix it in the camera-ready version. The table we are referring to is table 5. Below are the detailed analysis of the time complexity:

When the number of tasks is 8, estimate the surrogate model in MAP (alg 1) with 8 degree of freedoms in scaling coefficient vectors could still be costly. In practical, we need to sample about 300 set of scaling coefficient vectors, get their corresponding merged models and thus takes 300x8=2400 times of evaluation to implement MAP (alg 1). However, if we implement NMMAP (alg 3), we only need to evaluate 20 merged models for each round of 2-task-merging. In the first rounds, there are 4 2-task-merging running in parallel, each of them evaluates on 2 tasks. Thus, takes 4x20x2 = 160 times of evaluation. In the second round, there are 2 2-task-merging running in parallel, each of them evaluates on 4 tasks. It is 4 tasks rather than 2 task because when we merge the model: model for task a, b (m_{a,b}) and model for task c, d (m_{c,d}). We need to evaluate the merged model: model_{a, b, c, d} on Task a, b, c, d. Thus, it takes 2x20x4 = 160 times of evaluation. In the last round, there is only one 2-task-merging and evaluate on 8 tasks. It takes 1x20x8 = 160 times of evaluation. In total, NMMAP takes 480 times of evaluations, which is far less than 2400 evaluations.

In conclusion, we can see the number of rounds can be calculated by $\log_2(N)$, in each round, we need to evaluate $T \cdot N/2^i \cdot 2^i = TN$ where c is the number of scaling coefficient vectors needs to be evaluated when number of task is 2. In the above example, $T = 20$. Thus, the time complexity is $O(cN \log_2(N))$.

Continual of rebuttal

Official Comment Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) 07 Aug 2024, 07:48 (modified: 08 Aug 2024, 06:01)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer 8WQN Revisions

Comment:

W4: time complexity of nested-merging (typo-fixed, please kindly ignore the W4 from previous part)

Thank you for pointing that out. We will definitely fix it in the camera-ready version. The table we are referring to is table 5. Below are the detailed analysis of the time complexity:

When the number of tasks is 8, estimate the surrogate model in MAP (alg 1) with 8 degree of freedoms in scaling coefficient vectors could still be costly. In practical, we need to sample about 300 set of scaling coefficient vectors, get their corresponding merged models and thus takes 300x8=2400 times of evaluation to implement MAP (alg 1). However, if we implement NMMAP (alg 3), we only need to evaluate 20 merged models for each round of 2-task-merging. In the first rounds, there are 4 2-task-mergings running in parallel, each of them evaluates on 2 tasks. Thus, takes 4x20x2 = 160 times of evaluation. In the second round, there are 2 2-task-mergings running in parallel, each of them evaluates on 4 tasks. It is 4 tasks rather than 2 task because when we merge the model: model for task a, b ($model_{a,b}$) and model for task c, d ($model_{c,d}$). We need to evaluate the merged model: $model_{a,b,c,d}$ on Task a, b, c, d. Thus, it takes 2x20x4 = 160 times of evaluation. In the last round, there is only one 2-task-merging and evaluate on 8 tasks. It takes 1x20x8 = 160 times of evaluation. In total, NMMAP takes 480 times of evaluations, which is far less than 2400 evaluations.

In conclusion, we can see the number of rounds can be calculated by $\log_2(N)$, in each round, we need to evaluate $T \cdot N/2^i \cdot 2^i = TN$ where T is the number of scaling coefficient vectors needs to be evaluated when number of task is 2. In the above example, $T = 20$. Thus, the time complexity is $O(TN \log_2(N))$. We can rewrite it to $O(N \log_2(N))$ if ignoring T.

W5: writing clarity

- Figure 4: the caption for (b) seems to refer to a different figure.

We are deeply sorry for the mistake. We have fixed in the revised version pdf, will also fix it in the camera ready version. The fixed caption is: Average R^2 value of fitting the surrogate model for various number of dimensions. (a) showing the case of number of task is 2 and 3.(b) showing the case of number

of task is 4,5,6 and 7.

- Table 3 is extremely messy and hard to read.

We apologize for the confusion, this table shows the supremacy of Bayesian MAP (alg 2) comparing to the MAP (alg 1). We showed that if we only allow 12 set of scaling coefficient vectors to be sampled to mimic the performance of MAP under extreme low compute cases (usually we need about 20), Bayesian MAP is usually better than MAP. We run the Bayesian MAP for multiple iterations. Initially, we sample 6 scaling coefficient vectors and evaluate them. In iteration 1, we added 3 more scaling coefficient vectors and evaluate them. These 3 scaling coefficient vectors are sampled according to the uncertainty of the surrogate model fitting. Basically, we would sample more scaling coefficient vectors on where the fitting uncertainty is higher. From table 4, we can see BMAP iteration 1 (6+3) performs better than directly initialize with sampling 9 scaling coefficient vectors. Similarly, BMAP iteration 2 (6+3+3) performs better than directly initialize with sampling 12 scaling coefficient vectors.

- Table 7 is not intelligible, I advise finding another way to visualize that information.

We have fixed in the revised version pdf (anonymous.4open.science/r/MAP-NEURIPS/revised_paper.pdf), will also fix it in the camera ready version. Since it is a table including the detailed information of the table 4 exps, it is hard for us to post it here. We make a better illustrated version which is available in our anonymous code base as well. Please kindly check here if you are interested.

Continual of rebuttal

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 07 Aug 2024, 07:57 (modified: 07 Aug 2024, 07:58)
Program Chairs, Senior Area Chairs, Area Chairs, Authors, Reviewer 8WQN | Revisions

[Deleted]

Replying to Rebuttal by Authors

Supplement of the rebuttal

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 08 Aug 2024, 20:53
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors

Comment:

W3: role of second-order Taylor expansion. A Better illustration

Notations:

- p as the number of parameters in the pre-trained model (also the number of parameters in each task vector).
- $\mathbf{V}$ is the matrix of task vectors of different N tasks. Thus, $\mathbf{V} \in \mathbb{R}^{p \times N}$.
- $\mathbf{c}$ is the scaling coefficient vector $\in \mathbb{R}^N$.
- M_n is the metric (e.g. accuracy) for the task n.

$$
\begin{aligned}
M_n(\mathbf{c}) &= \underbrace{M_n(\theta_{pre})}_{\in\mathbb{R}} + \underbrace{\nabla M_n(\theta_{pre})^\top}_{\in\mathbb{R}^{1\times p}} \underbrace{\mathbf{V}}_{\in\mathbb{R}^{p\times N}} \underbrace{\mathbf{c}}_{\in\mathbb{R}^{N\times 1}} + \frac{1}{2} \underbrace{(\mathbf{V}\mathbf{c})^\top}_{\in\mathbb{R}^{1\times p}} \underbrace{\mathbf{H}_n(\theta_{pre})}_{\in\mathbb{R}^{p\times p}} \underbrace{\mathbf{V}\mathbf{c}}_{\in\mathbb{R}^{p\times 1}} + \underbrace{R_n}_{\in\mathbb{R}} \\
&= \underbrace{e_n}_{\in\mathbb{R}} + \underbrace{\mathbf{b}_n^\top}_{\in\mathbb{R}^{1\times N}} \underbrace{\mathbf{c}}_{\in\mathbb{R}^{N\times 1}} + \underbrace{\mathbf{c}^\top}_{\in\mathbb{R}^{1\times N}} \underbrace{\mathbf{A}_n}_{\in\mathbb{R}^{N\times N}} \underbrace{\mathbf{c}}_{\in\mathbb{R}^{N\times 1}}
\end{aligned}
$$

$$
\tilde{M}_n(\mathbf{c}; \mathbf{A}_n, \mathbf{b}_n, e_n) \equiv e_n + \mathbf{b}_n^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_n \mathbf{c}
$$

where

$$
\mathbf{A}_n = \mathbf{V}^\top \mathbf{H}_n(\theta_{pre}) \mathbf{V} \in \mathbb{R}^{N\times N}, \mathbf{b}_n = \mathbf{V}^\top \nabla M_n(\theta_{pre}) \in \mathbb{R}^N, e_n = M_n(\theta_{pre}) + R_n
$$

Please notice that $\mathbf{A}$ is a symmetric matrix. Specifically, when the number of tasks is 2, we have:

$$
\begin{aligned}
\tilde{M}_1(\mathbf{c}; \mathbf{A}_1, \mathbf{b}_1, e_1) &\equiv e_1 + \mathbf{b}_1^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_1 \mathbf{c} \\
&= \frac{1}{2}A_{1,11}c_1^2 + A_{1,12}c_1c_2 + \frac{1}{2}A_{1,22}c_2^2 + b_{1,1}c_1 + b_{1,2}c_2 + e_1 \\
\tilde{M}_2(\mathbf{c}; \mathbf{A}_2, \mathbf{b}_2, e_2) &\equiv e_2 + \mathbf{b}_2^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_2 \mathbf{c} \\
&= \frac{1}{2}A_{2,11}c_1^2 + A_{2,12}c_1c_2 + \frac{1}{2}A_{2,22}c_2^2 + b_{2,1}c_1 + b_{2,2}c_2 + e_2
\end{aligned}
$$

We don't calculate gradient or Hessian to get the parameters: $(\mathbf{A}_1, \mathbf{b}_1, e_1)$ and $(\mathbf{A}_2, \mathbf{b}_2, e_2)$

We use linear regression to estimate them. How? Given a (c_1, c_2) pair, we can define a merged model $\theta_{\text{merge}}(c_1, c_2)$, we evaluate the merged model on task 1 and task 2 to get the metrics (e.g. accuracy). Given 20 pairs of (c_1, c_2), we would be able to evaluate get 20 corresponding (M_1, M_2) which are metric for task 1 and metric for task 2. Thus, we can fit the surrogate model $\tilde{M}_1(\mathbf{c}; \mathbf{A}_1, \mathbf{b}_1, e_1)$ and $\tilde{M}_2(\mathbf{c}; \mathbf{A}_2, \mathbf{b}_2, e_2)$.

Replying to Supplement of the rebuttal

Looking forward to hearing from you

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 13 Aug 2024, 20:06
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors

Comment:

Dear Reviewer 8WQN,

Thank you again for your valuable suggestions on improving this manuscript.

We wanted to follow up to ensure that we have sufficiently addressed your questions and concerns. **As the discussion period deadline is ending in a few hours**, we would be grateful if you could kindly consider updating your evaluation if our revisions have resolved the issues you raised. We sincerely appreciate your time and effort in reviewing our submission.

Thank you for your consideration.

Best, Authors of Paper 8794

Official Review of Submission8794 by Reviewer FQxa

Official Review | Reviewer FQxa | 05 Jul 2024, 21:55 (modified: 25 Sept 2024, 11:58)

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer FQxa | Revisions

Summary:
This paper proposes Model Merging with Amortized Pareto Front (MAP), a method that identifies a Pareto set of scaling coefficients for merging multiple models to reflect the trade-offs. They utilize a quadratic approximation of the evaluation metric, and enhance it with a Bayesian adaptive sampling method and a nested merging scheme to reduce computational cost. Their main contributions are design of quadratic surrogate models to approximate evaluation models, a nested merging scheme, and a Bayesian adaptive sampling. The authors also conduct several experiments to support the effectiveness of their methods.

Soundness: 2: fair
Presentation: 2: fair
Contribution: 2: fair
Strengths:
The idea of approximating the ground truth evaluation model through quadratic functions is simple and can be effective. This simplifies the learning of PF. Also, the efforts to reduce computational cost are also appreciated. The authors have conducted several experiments to support the effectiveness of the algorithms.

Weaknesses:
Moving the detailed related work to the appendix is understandable, but it will be better if the authors provide a brief version in the main text. Besides, the related work is incomplete. There have been papers working on learning multiobjective optimization in similar scenarios with/without model merging [1,2,3,4,5]. The authors should discuss them and, even better, compare with them. Currently, the baselines in experiments are lacking. Comparing only with brute force search and not with other PSL approaches certainly needs justification.

Taylor expansion is only approximately accurate when in the affinity of the expansion point. In large model settings, this need to be better justified and empirically analyzed.

[1] Rewarded soups: towards Pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards

[2] Personalized Soups: Personalized Large Language Model Alignment via Post-hoc Parameter Merging

[3] Panacea: Pareto Alignment via Preference Adaptation for LLMs

[4] Beyond One-Preference-Fits-All Alignment: Multi-Objective Direct Preference Optimization

[5] Arithmetic Control of LLMs for Diverse User Preferences: Directional Preference Alignment with Multi-Objective Rewards

Questions:
The single sentence on the top of Page 2 is easy to be missed. The authors should improve their paper layout.

The computation cost of Pareto fronts is not well explained and motivated. The authors assume a setting where evaluation models are computationally expensive to query. But is this the reality? Why does quadratic approximation have a lower cost?

How to compute the Hessian in second-order Taylor expansion? Will that incur lots of computation and memory consumption?

I do not understand how $\frac{(N+1)(N+2)}{2}$ coefficients are derived in Line 116. What do the coefficients refer to? How is the number computed?

In nested merging scheme, the authors select pair-wise $\mathbf{c}^*$. This will result in only one solution. How can the Pareto front be represented? Currently, the nested merging scheme and the Bayesian adaptive sampling is not well explained, and is hard to understand. The authors should consider improving this part.

Limitations:
See weaknesses and questions.

Flag For Ethics Review: No ethics review needed.
Rating: 4: Borderline reject: Technically solid paper where reasons to reject, e.g., limited evaluation, outweigh reasons to accept, e.g., good evaluation. Please use sparingly.
Confidence: 4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.
Code Of Conduct: Yes

Rebuttal by Authors

Rebuttal | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 07 Aug 2024, 07:40 (modified: 07 Aug 2024, 09:07)

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors | Revisions

Rebuttal:
We have already **incorporated reviewers' feedback, integrated all changes and added more related work** into the revised manuscript, available at https://anonymous.4open.science/r/MAP-NEURIPS/revised_paper.pdf Please kindly feel free to check it.

We thank the reviewer for these comments and questions!

W1: Add related work to main text

Thanks for the insightful comments! We will include one more paragraph in the relative work: Model merging application in LLM. The rewarded soups and personalized soups are based on the model soup. We included it in the comparison. Please refer to the table above. Panacea is based on different setting. It requires fine-tuning when adapting the preference. Thus, we didn't compare with it. Beyond One-Preference-Fits-All Alignment and Arithmetic Control of LLMs for Diverse User Preferences are two very interesting papers. The latter incorporates multi-objective reward modeling to represents user preferences offering intuitive arithmetic control over LLM generation for diverse user preferences. However, because the application setting is different from ours. We are not able to compare with it. Instead, we have added various model merging based baselines. The details can be found in the 1-page PDF file, as well as the general response to all reviewers.

Q1: Fix single line above figure

Thank you for the suggestion! We will fix it in the camera-ready version.

Q2:

Why is evaluating models computationally expensive?

To get the pareto front of tasks which have trade-offs. We need to have a lot of data points to show the trade-off. Here, each data point is the evaluation metrics of a model. To get the ground truth evaluation metric of a task (e.g. let's say classification), we need to run the evaluation script of the model to determine the metric. If we have 4 tasks, and we set 5 possible values (let's say 0.2, 0.4, 0.6, 0.8, 1) for the scaling coefficient vector of each model specialized for one task. We will have to evaluate $5^4 = 625$ models on all 4 tasks. Assuming each evaluation takes 1 minute, evaluating all the models on all the 4 tasks will take 625 models x 4 tasks x 1 minute=2500 minute = 41.7 hours which is expensive. In big O notation, assuming we have T tasks, the time of evaluation for each task is T. the time (computational) complexity is $O(TN2^N)$.

Why does quadratic approximation have a lower cost?

For the MAP algorithm (alg 1), the time complexity is the same as what I mentioned above, what is different is that we fitted an approximation of the evaluation metrics. We only need the scaling coefficient vectors to input to a quadratic function model to be able to get the estimated evaluation score.

Run the quadratic function once take only $3.91 \times 10^{-6}s \pm 89410^{-9}s$. And thus, evaluating 2500 times takes < $2500 \times 4 \times 10^{-6}s < 10^{-2}s$.

Q3: computation of Hession?

We don't calculate the Hessian at all. Here is the explanation:

Denote $M(\theta)$ as the evaluation metric. The evaluation metrics depend on the validation set and the model to be evaluated. The model depends on the task vectors and scaling coefficient vectors. In our setting, the only variables are the scaling coefficient vectors.
$M_n(c) \equiv M(\theta_{\text{merge}}(c)) = M(\theta_{pretrained}) + \nabla M(\theta_{pretrained})^\top(\theta_m(c) - \theta_{pretrained}) + \frac{1}{2}(\theta_m(c) - \theta_{pretrained})^\top H_n(\theta_{pretrained})(\theta_m(c) - \theta_{pretrained}) + R_n(\theta_m(c) - \theta_{pretrained})$
where $M(\theta_{pretrained})$ is the constant term, we define it as e.

It is fixed given the pretrained model. $\nabla M(\theta_{pretrained})^\top(\theta_m(c) - \theta_{pretrained})$ is the first order gradient term. $R_n(\theta_m(c) - \theta_{pretrained})$ is the remainder term which is the approximation error term.

$\theta_m(c) - \theta_{pretrained}$ term can be rewritten as the $V \cdot c$ which is the dot product of the task vector and the scaling coefficient vectors.

Therefore, the term $\nabla M(\theta_{pretrained})^\top(\theta_m(c) - \theta_{pretrained})$ can be represented as $\nabla M(\theta_{pretrained})^\top V \cdot c$.

$M(\theta_{pretrained})^\top$'s shape is $(1, p)$, V 's shape is (p, N), where p is the number of parameters of the model which is a huge number. We define vector b s.t. $b^\top \equiv \nabla M(\theta_{pretrained})^\top V$. However, b's shape is $(1, N)$ which is only the number of tasks. The term, $\frac{1}{2}(\theta_m(c) - \theta_{pretrained})^\top H_n(\theta_{pretrained})(\theta_m(c) - \theta_{pretrained})$,can be re-written as $\frac{1}{2}c^\top V^\top H_t\theta_{pretrained}Vc$.

Similarly, the $V^\top$, $H_t\theta_{pretrain}$, V has the corresponding shapes: $(N, p), (p, p), (p, N)$. Denote $A \equiv V^\top H_t\theta_{pretrain}V$, A's shape is (N, N).

Thus, $M(c) = e + b^\top c + \frac{1}{2}c^\top Ac + R_n(\theta_m(c) - \theta_{pretrained})$. We want to avoid calculating $b^\top$ and A by their definition because we want to bypass the calculation of the gradient term and hessian term. Thus, we would set them as the parameters of the surrogate model.

For example, if the number of tasks is 2. In this case, $c = (c1, c2)$, $M(c) = e + b_1 \cdot c_1 + b_2 * c_2 + A_{11}c_1^2 + A_{12} \cdot c_1 \cdot c_2 + A_{22} \cdot c_2^2$, the surrogate model has 6 parameters: $(e, b_1, b_2, A_{11}, A_{12}, A_{22})$. If there are N tasks, then c is an N-dimensional vector, the number of parameters of the surrogate model is $1 + N + (N^2 - N)/2 + N = (N + 1)(N + 2)/2$.

Continual of rebuttal

Official Comment Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) 07 Aug 2024, 07:58 (modified: 08 Aug 2024, 20:46)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer FQxa Revisions

Comment:

Continuing the answer of the Q3 rebuttal

For the MAP algorithm, the lowest possible number of evaluation it needs is $O(N^3)$ because to get the solution of the surrogate model, we need at least $(N + 1)(N + 2)/2$ scaling coefficient vectors for the merged models. Each merged model requires N times of evaluations since there are N tasks. Thus, the time complexity is at least $O(N^3)$. When the dimension is high, we realized that the curse of dimensionality start to dominate the complexity. If we maintain each scaling coefficient discretization space (pts per dimension in the figure 3 in the rebuttal pdf) at least have 2 values inside. The number of scaling coefficient vectors required would be $O(2^N)$. Each corresponding model need to be evaluated N times. Thus, the number of total evaluation is $O(N2^N)$

W2: Are finetuned model affinity of the expansion point? (Taylor expansion approximation)

That's an important question! Yes, we run the calculated the L1 norm of the tasks vectors and the pretrained model. Please kindly refer to the table below.

Metric	SUN397	Cars	DTD	SVHN	RESISC45	MNIST	GTSRB	EuroSAT
pretrained_l1_norm	1270487	1270487	1270487	1270487	1270487	1270487	1270487	1270487
diff_l1_norm	21055.390	20127.460	13621.270	19349.623	18409.214	17578.563	16712.563	15941.990
ratio(%)	1.66%	1.58%	1.07%	1.52%	1.45%	1.38%	1.32%	1.25%

Q3: computation of Hession? Better illustration

Notations:

- p as the number of parameters in the pre-trained model (also the number of parameters in each task vector).
- $\mathbf{V}$ is the matrix of task vectors of different N tasks. Thus, $\mathbf{V} \in \mathbb{R}^{p \times N}$.
- $\mathbf{c}$ is the scaling coefficient vector $\in \mathbb{R}^N$.
- M_n is the metric (e.g. accuracy) for the task n.

$$
\begin{aligned}
M_n(\mathbf{c}) &= \underbrace{M_n(\theta_{pre})}_{\in\mathbb{R}} + \underbrace{\nabla M_n(\theta_{pre})^\top}_{\in\mathbb{R}^{1\times p}} \underbrace{\mathbf{V}}_{\in\mathbb{R}^{p\times N}} \underbrace{\mathbf{c}}_{\in\mathbb{R}^{N\times 1}} + \frac{1}{2}\underbrace{(\mathbf{V}\mathbf{c})^\top \mathbf{H}_n(\theta_{pre})}_{\in\mathbb{R}^{1\times p}\ \ \in\mathbb{R}^{p\times p}} \underbrace{\mathbf{V}\mathbf{c}}_{\in\mathbb{R}^{p\times 1}} + \underbrace{R_n}_{\in\mathbb{R}} \\
&= \underbrace{e_n}_{\in\mathbb{R}} + \underbrace{\mathbf{b}_n^\top}_{\in\mathbb{R}^{1\times N}} \underbrace{\mathbf{c}}_{\in\mathbb{R}^{N\times 1}} + \underbrace{\mathbf{c}^\top}_{\in\mathbb{R}^{1\times N}} \underbrace{\mathbf{A}_n}_{\in\mathbb{R}^{N\times N}} \underbrace{\mathbf{c}}_{\in\mathbb{R}^{N\times 1}}
\end{aligned}
$$

$$
\tilde{M}_n(\mathbf{c}; \mathbf{A}_n, \mathbf{b}_n, e_n) \equiv e_n + \mathbf{b}_n^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_n \mathbf{c}
$$

where

$$
\mathbf{A}_n = \mathbf{V}^\top \mathbf{H}_n(\theta_{pre})\mathbf{V} \in \mathbb{R}^{N\times N}, \mathbf{b}_n = \mathbf{V}^\top \nabla M_n(\theta_{pre}) \in \mathbb{R}^N, e_n = M_n(\theta_{pre}) + R_n
$$

Please notice that $\mathbf{A}$ is a symmetric matrix. Specifically, when the number of tasks is 2, we have:

$$
\begin{aligned}
\tilde{M}_1(\mathbf{c}; \mathbf{A}_1, \mathbf{b}_1, e_1) &\equiv e_1 + \mathbf{b}_1^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_1 \mathbf{c} \\
&= \frac{1}{2}A_{1,11}c_1^2 + A_{1,12}c_1c_2 + \frac{1}{2}A_{1,22}c_2^2 + b_{1,1}c_1 + b_{1,2}c_2 + e_1 \\
\tilde{M}_2(\mathbf{c}; \mathbf{A}_2, \mathbf{b}_2, e_2) &\equiv e_2 + \mathbf{b}_2^\top \mathbf{c} + \frac{1}{2}\mathbf{c}^\top \mathbf{A}_2 \mathbf{c} \\
&= \frac{1}{2}A_{2,11}c_1^2 + A_{2,12}c_1c_2 + \frac{1}{2}A_{2,22}c_2^2 + b_{2,1}c_1 + b_{2,2}c_2 + e_2
\end{aligned}
$$

We don't calculate gradient or Hessian to get the parameters: $(\mathbf{A}_1, \mathbf{b}_1, e_1)$ and $(\mathbf{A}_2, \mathbf{b}_2, e_2)$

We use linear regression to estimate them. How? Given a (c_1, c_2) pair, we can define a merged model $\theta_{\text{merge}}(c_1, c_2)$, we evaluate the merged model on task 1 and task 2 to get the metrics (e.g. accuracy). Given 20 pairs of (c_1, c_2), we would be able to evaluate get 20 corresponding (M_1, M_2) which are

metric for task 1 and metric for task 2. Thus, we can fit the surrogate model $\tilde{M}_1(\mathbf{c}; \mathbf{A}_1, \mathbf{b}_1, e_1)$ and $\tilde{M}_2(\mathbf{c}; \mathbf{A}_2, \mathbf{b}_2, e_2)$.

Replying to Continual of rebuttal

Looking forward to hearing from you

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 13 Aug 2024, 20:03

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer FQxa

Comment:

Dear Reviewer FQxa,

Thank you again for your thoughtful comments. Following your suggestions, we have tried our best to address the points raised in your review in our previous response and the updated manuscript (https://anonymous.4open.science/r/MAP-NEURIPS/revised_paper.pdf). We understand you might have a very busy schedule, but as the discussion period is **ending in a few hours**, we would love to hear your thoughts on our response. Is there any unclear point that we can further clarify? Many thanks!

Best, Authors of paper 8794

Official Comment by Reviewer FQxa

Official Comment | Reviewer FQxa | 14 Aug 2024, 04:18

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer FQxa

Comment:

Thank you for the rebuttal. I choose to maintain my score.

Official Comment by Authors

Official Comment | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 14 Aug 2024, 04:30 (modified: 14 Aug 2024, 04:51)

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer FQxa | Revisions

Comment:

Thanks for replying! We understand the discussion period is ending within 3 hours. We will still try our best to answer your questions in the rest of the limited time if you have any other concerns. Please kindly feel free to let us know. We appreciate your discussion and comments. Thanks!

Best,
Authors of paper 8794

Official Review of Submission8794 by Reviewer G7ak

Official Review | Reviewer G7ak | 05 Jul 2024, 19:10 (modified: 25 Sept 2024, 11:58)

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer G7ak | Revisions

Summary:

They propose a method for merging a model given a set of preferences. This method tries to find task vector coefficients and optimizes a quadratic approximation of this objective using Bayesian adaptive sampling. They show that the surrogate is more efficient

Soundness: 2: fair
Presentation: 3: good
Contribution: 3: good

Strengths:

- The idea of optimizing a surrogate is a very nice idea.
- The experiments show that the MAP is better than the brute force search and that the surrogate is a good approximation,

Weaknesses:

- Surrogate only considers optimization over the task vector scaling coefficients. There are many merging methods that do not just select task vector scaling coefficients. [3]
- Experiments do not show that MAP is a better merging method. No comparisons against other merging methods.

Questions:

- Can you include a high level overview of the algorithm used to optimize MOOP?
- Could you have applied bayesian adaptive sampling w/o a surrogate using the evaluation metric?
- How is the surrogate fit (i.e. A*n, b*n, e*n learned) in Alg 1 Line 7? Is it minimizing eq 6? How is it computed?
- Where does the preference vector show up in the objective in the equations?
- Section 4.2 - Though the baseline of direct search is included, other simple baselines are missing. For example,
  - what happens if using scaling factors for the task vectors corresponding to the preference?
  - Sweeping the scaling factor with a fixed value from 0 to 1 in increments of 0.1 as done in the original task vector paper
- Section 4.6 - Shouldn't the baseline include merging 8 models?
- I don't follow table 3. What are iter 9 and iter 12? Are those iterations 9 and 12? Why do they do poorly?
- In the experiments, how are the preference vectors chosen?
- Missing reference to [1] and [2], which also relate task vectors to building approximations around the pre-trained model.
- The caption for figure 4 seems to not match the figure?

[1] Model Merging by Uncertainty-Based Gradient Matching. Daheim et al. https://arxiv.org/abs/2310.12808 [2] Task Arithmetic in the Tangent Space: Improved Editing of Pre-Trained Models. Ortiz-Jiminez et al. https://arxiv.org/abs/2305.12827 [3] Dataless Knowledge Fusion by Merging Weights of Language Models. Jin et al. https://arxiv.org/abs/2212.09849

Limitations:

NA

Flag For Ethics Review: No ethics review needed.
Rating: 6: Weak Accept: Technically solid, moderate-to-high impact paper, with no major concerns with respect to evaluation, resources, reproducibility, ethical considerations.
Confidence: 4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.
Code Of Conduct: Yes

Rebuttal by Authors

Rebuttal | Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) | 07 Aug 2024, 07:24 (modified: 07 Aug 2024, 09:07)

Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors Revisions

Rebuttal:

We have already **incorporated reviewers' feedback, integrated all changes and added more related work** into the revised manuscript, available at https://anonymous.4open.science/r/MAP-NEURIPS/revised_paper.pdf Please kindly feel free to check it.

W1: only applies to task vector model mergeing method

Thanks for pointing out. Yes, our method serves as a plugin to all task vector-based model-merging methods where the practitioners' the degree of freedom to determine the scaling coefficient vectors (e.g. task arithmetic and DARE combined with task arithmetic). We will explore how to estimate the Pareto front on other non-task-vector based model-merging methods in future work.

W2: more exps on baselines

Thanks for the suggestion. We have added more exps to compare MAP algorithms with other model-merging methods. Please kindly refer to the table in the overall rebuttal. From the results, we can see that ties-merging and other merging method can only dominate very little proportion of the Pareto front. When the number of tasks is high, All the hyperparameters of the methods we compared are from the recommendation of the original paper. We did some hyperparameter search for them as well. The results show the hyperparameters they recommended would give the best performance. When the number of tasks is 2, we draw 3 Pareto fronts and all the merging methods performance we have tested on the same figure (please kindly check figure 1 in the rebuttal pdf).

Q1: NSGA-III

Yes, we have included an algorithm block in the supplement for the NSGA-III algorithm used to optimize the MOOP in the revised pdf (anonymous.4open.science/r/MAP-NEURIPS/revised_paper.pdf). In addition, we have included a brief overview of the algorithm:

1. Initialization:
   - The population is initialized with N individuals.
   - The total generation g.
   - A set of reference points $\mathcal{R}$ is defined, which guides the selection process based on diversity.
2. Evolution Process:
   - For each generation g, offspring are generated using crossover and mutation.
   - The parent and offspring populations are combined.
   - A non-dominated sorting is performed to identify the different fronts $F_1, F_2, \ldots$.
   - The new population P is filled front by front until the population size N is met.
   - If the population size exceeds N when adding a front, reference point-based selection is performed to maintain diversity and fill the remaining slots.
3. Output:
   - After G generations, the final population P is returned.

Q2: Bayesian without surrogate model?

Thank you for this very insightful question! Sorry, we are not able to apply the Bayesian adaptive sampling without the surrogate model because Bayesian method need to work with a metric to indicate which regions should be sampled more in the next stage. In our case, this metric is called the Upper confidence bound (UCB) which is mean + ½ std of the fitting error of the surrogate model. Without the surrogate model, we don't have such a metric to indicate where to sample.

Q3: How surrogate model learned?

Let's take a closer look at the surrogate model. For example, if the number of tasks is 2. In this case, $c = (c_1, c_2), \tilde{M}(c) = e + b_1c_1 + b_2c_2 + A_{11}c_1^2 + A_{12}c_1c_2 + A_{22}c_2^2$. The surrogate model has 6 parameters: $(e, b_1, b_2, A_{11}, A_{12}, A_{22})$.

Thus, in general, we can just implement linear regression based on the data $(c, M(c))$ pairs we collected. Denote $\mathbf{C} = \text{concat}(c_1^2, c_2^2, \cdots, c_N^2, c_1c_2, c_1c_3, \cdots, c_{N-1}c_N, c_1, c_2, \cdots, c_N, 1)$. The close form of the linear regression is $(C^\top C)^{-1}C^\top M(c)$. In general, when we composite an activation function σ within the definition of the surrogate model to restrict the prediction range. The surrogate model for task n is estimated by $A_n^\star, b_n^\star, e_n^\star = \arg\min_{A_n, b_n, e_n} \text{MSE}\left(M_n(\theta_{\text{merge}}(c)), \sigma(\tilde{M}_n(c; A_n, b_n, e_n))\right)$,

Q4: where is pref show?

In the MAP and BMAP (alg 1 and alg 2), the users do not need to specify a preference vector and can obtain the full pareto front, and then pick the solutions as they like. For example, they can pick a solution by using preference vectors, or using certain rules, such as "I want the performance on tasks A, B, and C to be at least 70% of the original task, and then make the performance of task D as high as possible". Indeed, this way of letting the users get the full Pareto front is ideal and arguably more useful than other MOOP algorithms that need users to specify a preference vector beforehand. This is because the users sometimes might not know about the preferences as numeric values.

In nested merging, we will illustrate how the preference works in the algorithm. For example, the there are $model_i$ for task i as individual finetuned models. $i = 1, 2, 3, 4$. We first merge model 1 and model 2 given the preference of user between task 1 and task 2. We then get $model_{1,2}$. At the same time, we merge model 3 and model 4 given the preference of user between task 3 and task 4 and get $model_{3,4}$. Finally, we merge the $model_{1,2}$ and $model_{3,4}$ given the preference of user to get $model_{1,2,3,4}$. We output the $model_{1,2,3,4}$ as the output merged model of the algorithm.

Q5: more exps and baselines

We thank the reviewer for the insightful suggestions of baselines! We added extensive experiments on the two settings. Please kindly see the table in the overall rebuttal.

Q6: nested-merging exps

Please kindly check the performance including nested-merging MAP when number of task is 8:

	Preference weighted sum(↑)	Win rate of MAP(↑)
Single-taskmodels	90.05±3.02	
MAP-nested	67.05±3.89	
MAP-plain	**72.12**±3.78	
TIES-merging	70.79±3.81	1.00±0.00
DARE-TIES	70.51±3.58	0.90±0.32
TaskArithmetic(TA)	59.44±5.68	1.00±0.00
DARE-TA	63.14±4.91	1.00±0.00

Continue of the rebuttal

Official Comment Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) 07 Aug 2024, 07:26 (modified: 11 Aug 2024, 01:00)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer G7ak Revisions

Comment:

Q7: Bayesian MAP performance

We apologize for the confusion, this table shows the supremacy of Bayesian MAP (alg 2) comparing to the MAP (alg 1). We showed that if we only allow 12 sets of scaling coefficient vectors to be sampled to mimic the performance of MAP under extreme low compute cases (usually we need about 20), Bayesian MAP is usually better than MAP. Please kindly check our revised version of paper (https://anonymous.4open.science/api/repo/MAP-NEURIPS/file/revised_paper.pdf). **Appendix D, Figure 7, Page 19**

We run the Bayesian MAP for multiple iterations. Initially, we sample 6 scaling coefficient vectors and evaluate them. In iteration 1, we added 3 more scaling coefficient vectors and evaluate them. These 3 scaling coefficient vectors are sampled according to the uncertainty of the surrogate model fitting. Basically, we would sample more scaling coefficient vectors on where the fitting uncertainty is higher. From table 4, we can see BMAP iteration 1 (6+3) performs better than directly initialize with sampling 9 scaling coefficient vectors. Similarly, BMAP iteration 2 (6+3+3) performs better than directly initialize with sampling 12 scaling coefficient vectors.

Why they perform bad?
This is because we compare BMAP with MAP in an situation that **computational resource are very limited**. Normally, when running MAP in 2-model-merging case, we need to sample 20 scaling coefficient vectors and evaluate their corresponding merged model. In the BMAP vs MAP experiment, we only take 9 (iter 1), 12 (iter 2) scaling coefficient vectors. The results showing that BMAP is on average better than MAP when the computation budget is low.

Q8: How pref are chosen in the exps?

In the experiments for MAP (alg 1), there's no need for choosing preference vectors since the algorithm directly output the whole Pareto front. The sampled c's are grid points. In the experiments for nested-merging MAP (alg 3), we used the following way to choose the preference vectors. Given n tasks, we list all the permutations from 1 through n (n! is the total permutations for n tasks), for example, for n = 3, we list: [1,2,3], [1,3,2], [2,3,1], [2,1,3], [3,1,2], [3,2,1], then we normalize the vector such that it sums to 1. For example, [2,1,3] becomes [0.33, 0.17, 0.5]. The normalized vectors then represent the different preferences the user might have. We performed nested merging using the preference vectors, and then compared with the plain MAP algorithm.

Q9: Missing related work

Thank you for pointing out the missing references. We have added them to the paper and related work (https://anonymous.4open.science/api/repo/MAP-NEURIPS/file/revised_paper.pdf).

Q10: Caption of figure 4

We are deeply sorry for the mistake. We have fixed in the revised version pdf (https://anonymous.4open.science/api/repo/MAP-NEURIPS/file/revised_paper.pdf), will also fix it in the camera ready version. The tentative fix is: how the average R^2 value as the quality of fitting the surrogate model change as more merged model is evaluated. (a) showing the case of number of task is 2 and 3.(b) showing the case of number of task is 4,5,6 and 7.

Replying to Continue of the rebuttal

Official Comment by Reviewer G7ak

Official Comment Reviewer G7ak 11 Aug 2024, 20:39
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer G7ak

Comment:
Thanks for the response and for the new experiments where it outperforms TIES and DARE. Some discussion of the extra compute this method requires to for evaluating the surrogates should be mentioned (compared to those baselines). I raise my score from a 5 to a 6.

Official Comment by Authors

Official Comment Authors (Suyuchen Wang, Jie Fu, Yong Chen, Lu Li, +6 more) 11 Aug 2024, 21:05 (modified: 11 Aug 2024, 21:33)
Program Chairs, Senior Area Chairs, Area Chairs, Reviewers Submitted, Authors, Reviewer G7ak Revisions

Comment:
Dear reviewer G7ak,

Thank you for your recognition of our work!

We apologize that due to the limited space, the discussion of the extra computation for fitting the surrogate model is discussion in the Appendix B of the original submission. In our revised version (https://anonymous.4open.science/api/repo/MAP-NEURIPS/file/revised_paper.pdf), it is in **Appendix C, Page 16**. Please feel free to check it.

A conclusion of the computation cost of fitting the surrogate model

	# evals per task	minimum # evals (total)	# evals (total)
Naive model merging	$O(N^2)$	$O(N^3)$	$O(N2^N)$
Nested model merging	$O(1)$	$O(N \log N)$	$O(N \log N)$

The empirical inference cost of the surrogate model

This part is discussed in **Appendix H, Q5, Page 25** (https://anonymous.4open.science/api/repo/MAP-NEURIPS/file/revised_paper.pdf):

For the MAP algorithm \Cref{alg:Taylor} , the time complexity is the same as what I mentioned above, what is different is that we fitted an approximation of the evaluation metrics. We only need the scaling coefficient vectors to input to a quadratic function model to be able to get the estimated evaluation score. Run the quadratic function once take only $3.91 \times 10^{-6}s \pm 894 \times 10^{-9}s$. And thus, evaluating 2500 times takes $< 2500 \times 4 \times 10^{-6}s = 10^{-2}s$.

Best,
Authors of paper 8794

About OpenReview Contact Frequently Asked Questions
Hosting a Venue Feedback Terms of Use
All Venues Sponsors Privacy Policy

OpenReview is a long-term project to advance science through improved peer review, with legal nonprofit status through Code for Science & Society. We gratefully acknowledge the support of the OpenReview Sponsors. © 2024 OpenReview