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

**Anonymous Author(s)**
Affiliation
Address
email

## 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.

## 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. The 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, the concept of model merging—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 are 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

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

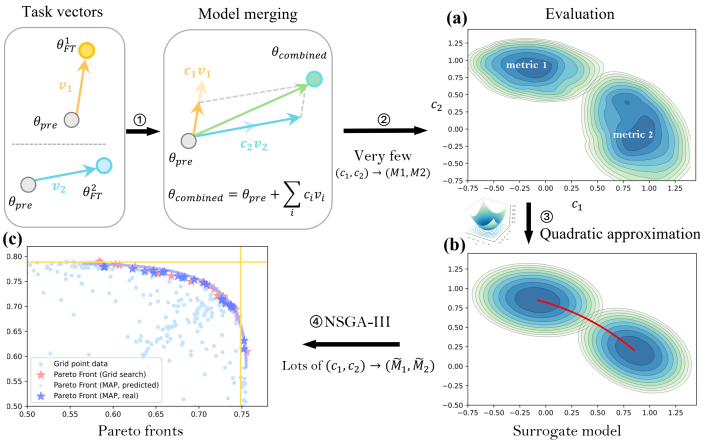



Figure 1: The overall process of MAP. Step 1: select 2 tasks and their corresponding task vectors. Step 2: sample few scaling weights $\mathbf{c}$ to query the task metrics accordingly. Step 3: use Quadratic model to approximate the $\mathbf{c} \rightarrow$ *metrics* mapping as a surrogate model. Step 4: Use NSGA-III to find amortized Pareto fronts. Figure (a) shows 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 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 Pareto Front. Pareto front (Grid search) is regarded as the ground truth given the enough number of grid points. Pareto front (MAP, predicted) is the amortized Pareto front. Pareto front (MAP, real) is the Pareto front involving the same $\{(c_1, c_2)\}$ but re-evaluated as to get the ground truth metrics as a 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 utizes quadratic surrogate models to approximate the evaluation metric functions, which amortize the computation of Pareto fronts;
- **C2** Propose a nested-merging MAP to decrease computational complexity from $O(N \cdot 2^N)$ to $O(N \log_2^N)$;
- **C3** Propose Bayesian MAP, which efficiently queries the computationally expensive evaluations according to loss information.
- **C4** 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.

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,

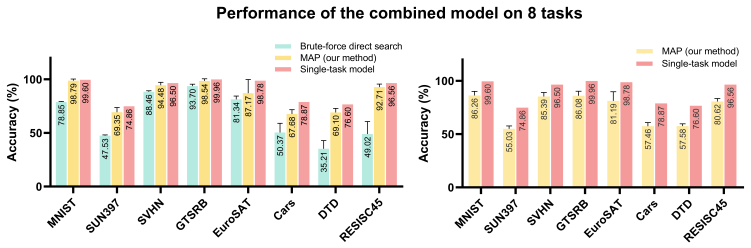



Figure 2: **(a)** Using the results of inference runs of merged ViT-B-32 models [45] obtained from 250 combinations of scaling coefficients, our method outperforms the direct search method in finding a set of diverse solutions for eight tasks based on the same computation expense. For both methods, we 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.

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 pairs of solution directly against each other (known as brute force search). Since brute force search compares all possible solutions and finds all the Pareto optimums, 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 released here `https://anonymous.4open.science/r/MAP-NEURIPS/`.

# 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 $c_i$ is some scaling factors and has 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 the 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 Methods

## 3.1 Quadratic approximation of evaluation metric

Given the task vectors $\{\mathbf{v}_n\}_{n=1,...,N}$ 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)[1]:

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

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 ($e_n$, $\mathbf{b}_n$, $\mathbf{A}_n$)[2] 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.

[1]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)

[2] 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.

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

### 3.2 Model merging with amortized pareto fronts

In this section, we introduce our generalized algorithm for estimating the amortized Pareto fronts. As mentioned in Section 3.1, 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 4 and the algorithm details in Algorithm 1. Different from other Pareto Multi-task learning algorithms, our algorithm

**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 $\Omega$.
  **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

### 3.3 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 5 for the detailed computational complexity comparison. Algorithm details are presented in Algorithm 2.

**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 the $\mathbf{c}^* = (c_i^*, c_j^*) \in \mathbb{R}^2$ 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 $l_{ij} = \mathit{pref}_i l_i + \mathit{pref}_j l_j$
  Update $\tau$ 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** $\boldsymbol{\theta}_{\text{merge}}^{1,2,\cdots,N}$

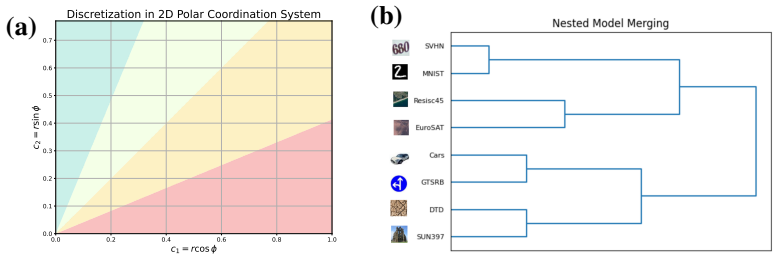



Figure 3: (a) Discretizing of two task scaling coefficients along the angular dimension in 2D polar coordinate system (please refer to Figure 12 for an example of 3D spherical discretization.); (b) An example of nested model merging for $N = 6$ models (see more details in Appendix G);

### 3.4 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 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 discretizing along the angular dimensions. Please refer to figure 12 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 $n$.

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

## 4 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).

### 4.1 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 experiments, due to the page limits, we show them in the appendix. The result of dataset and task we experiment on are as follows: a zero-shot medical chest x-ray image classification tasks to show our model merging scheme also works in a read world application domain [58]; 4 fine-tuned Llama3 models in different languages: French, Arabic, Chinese, and Japanese. [3]; 3 vision tasks on the ResNet18 architecture, CIFAR-10 [32], Flowers-102 [41], and GTSRB [51].

### 4.2 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.

**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 sample $K = \min(100, |PF_{baseline}|)$ 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 compare $M_n(\theta(\mathbf{c}_k^{MAP}))$ 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 $PF_{MAP}$.

### 4.3 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.

[3] All the fine-tuned language Llama3 models can be found on huggingface. The IDs of the model 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.

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% ($\pm$0.30) | 0.953 ($\pm$0.018) |
| 3 | 300 | 6.69 | 50 | 46.90% ($\pm$0.71) | 0.980 ($\pm$0.003) |
| 4 | 300 | 4.16 | 60 | 50.67% ($\pm$2.44) | 0.984 ($\pm$0.004) |
| 5 | 500 | 3.47 | 85 | 53.00% ($\pm$1.88) | 0.941 ($\pm$0.019) |
| 6 | 500 | 2.82 | 100 | 60.71% ($\pm$1.34) | 0.941 ($\pm$0.030) |
| 7 | 1000 | 2.68 | 140 | 63.42% ($\pm$1.91) | 0.891 ($\pm$0.024) |
| 8 | 1000 | 2.37 | 250 | 65.58% ($\pm$0.94) | 0.868 ($\pm$0.028) |

### 4.4 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 4. In addition, we show that MAP is 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 4, we show the Pareto front found by MAP-DARE.

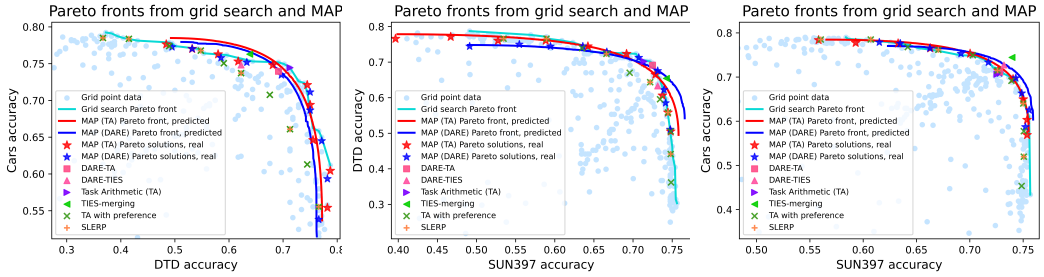



Figure 4: The Pareto fronts obtained using MAP with Task Arithmetic, MAP with Task Arithmetic and DARE. We sampled 10 Pareto solutions on the Pareto front predicted by MAP and evaluated them to obtain the real values. We plot the results obtained using TIES-merging, Task Arithmetic (TA) with a single 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 on within the Pareto front found by MAP. In addition, we sampled 10 points from the Pareto front predicted by MAP and evaluated them (MAP Pareto solutions, real), and 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 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 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.

### 4.5 Resulting on using nested merging to merge 8 models

We further evaluate the effectiveness of nested merging MAP. Supplemental Figure 6 shows the intermediate Pareto fronts obtained using nested merging MAP sequentially. In addition, we compared

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. ↑ indicates higher is better. The number after ± 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 (↑) | | | | | | | |
|---|---|---|---|---|---|---|---|
| # 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 (↑) | | | | | | | |
| # 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 |

Table 4: We compared nested-merging MAP with MAP and baseline methods in merging 8 models using a set of 10 preference vectors in terms of preference-weighted sum of accuracies.

| Metric | Single-task models | MAP-nested | MAP-plain | TIES-merging | DARE-TIES | Task Arithmetic (TA) | DARE-TA |
|---|---|---|---|---|---|---|---|
| Preference weighted sum (↑) | 90.05 ± 3.02 | 67.05 ± 3.89 | **72.12** ± 3.78 | 70.79 ± 3.81 | 70.51 ± 3.58 | 59.44 ± 5.68 | 63.14 ± 4.91 |

the performance of nested merging MAP (Algorithm 2) with MAP (Algorithm 1). The results are shown in Table 4. The results confirm that nested-merging MAP can obtain comparable results to those obtained by MAP while using much fewer evaluation runs. Specifically, for nested merging MAP (check Figure 6 for the process), we use 20 scaling coefficients for each pairwise merge. Thus, in the first round merging, the number of evaluations is $20 \times 2$ tasks $= 40$ times. In the second round merging, the number of evaluations is $20 \times 4$ tasks $= 80$ times. In the last round of merging, the number of evaluations is $20 \times 8$ tasks $= 160$ times. According to Figure 6, there are 4 first rounds in the first merging, 2 second rounds in the second merging and 1 last round. Thus, in total, we run $40 \times 4 + 80 \times 2 + 160 \times 1 = 480$ evaluations. In contrast, when running the vanilla MAP (Algorithm 1), we used 280 coefficients and evaluate $280 \times 8 = 2240$ times. In addition, while being suboptimal to MAP (Algorithm 1), nested-merging MAP can still outperform other baseline methods such as TA and DARE-TA in accommodating various user preferences with relatively low computational cost, especially when the number of tasks is high.

### 4.6 Bayesian MAP experiments

Please refer to Appendix D for Bayesian MAP experiments.

## 5 Conclusion and Limitations

Our method (MAP) can serve as an out-of-box plug-in that is very easy to directly combine with other task-vector based model-merging methods, for example, tasks arithmetic [24] and DARE [68]etc. From Section 3 and Section 4, we have shown that our method Algorithm 1, algorithm 2, Algorithm 3 can efficiently bring down 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 the Algorithm 1, we do not deal with the situation of non-convex Pareto front. In the case of 2 tasks, most likely the Pareto fronts with proper trade-offs would have convex Pareto front, but if one deals with more tasks, the ground truth Pareto fronts are likely non-convex and ending up approximating a convex hull by Algorithm 1. A quick but naive fix would be to implement Algorithm 2 in this case so that the Pareto fronts between 2 tasks would be more likely to be convex. Second, we don't have a detector algorithm to detect if 2 tasks have proper trade-offs. Approximating a Pareto 'front' with only a single Pareto solution inside likely results in very unstable and badly-performed results. The practitioners should be informed in this case.

# 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 $\mu$-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.

# 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 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 $N \geq 3$, the MOOP is termed as the many-objective optimization, which is significantly challenging for well-established algorithms (e.g. NSGA-II and SPEA2) and requires more advanced algorithms such as $\epsilon$-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 Model Soup paper, which is heuristic and sub-optimal, 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 go through lazy training, meaning that the converging parameters are close to the initialization [10, 18, 2], and thus the local approximation informs on 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 select the hyperparameters before the training actually takes place.

**Federated/Private Learning** Because the model merging is a post-training approach, it can be naturally used in federated learning which protects both the model privacy and data privacy. To be specific, for the 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 the 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** Recently 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.

**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 5 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 5 (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 5 (b)).

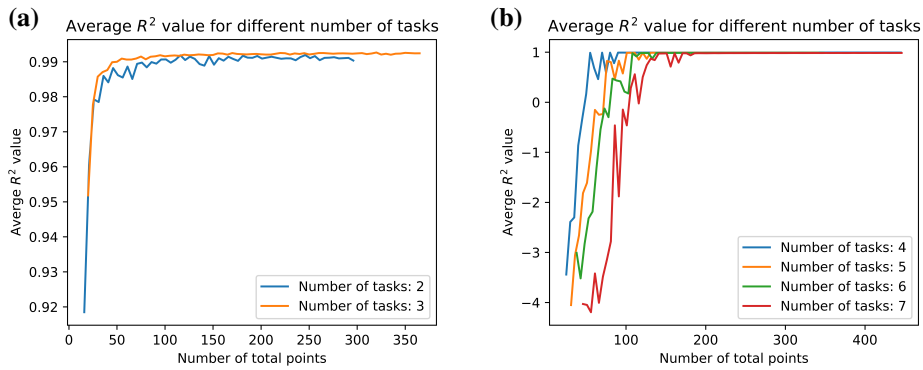



Figure 5: Average $R^2$ value as a measure of the quality of fitting the surrogate model for merging different number of tasks. (a) Number of tasks is 2 and 3; (b) Number of tasks is 4,5,6 and 7.

# C Nested-merging MAP

## C.1 Time complexity of nested-merging MAP

Table 5: 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 task. 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 NNested-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, 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 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 $2 \times 20 \times 4 = 160$ 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 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 $T$.

## 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 6, 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.

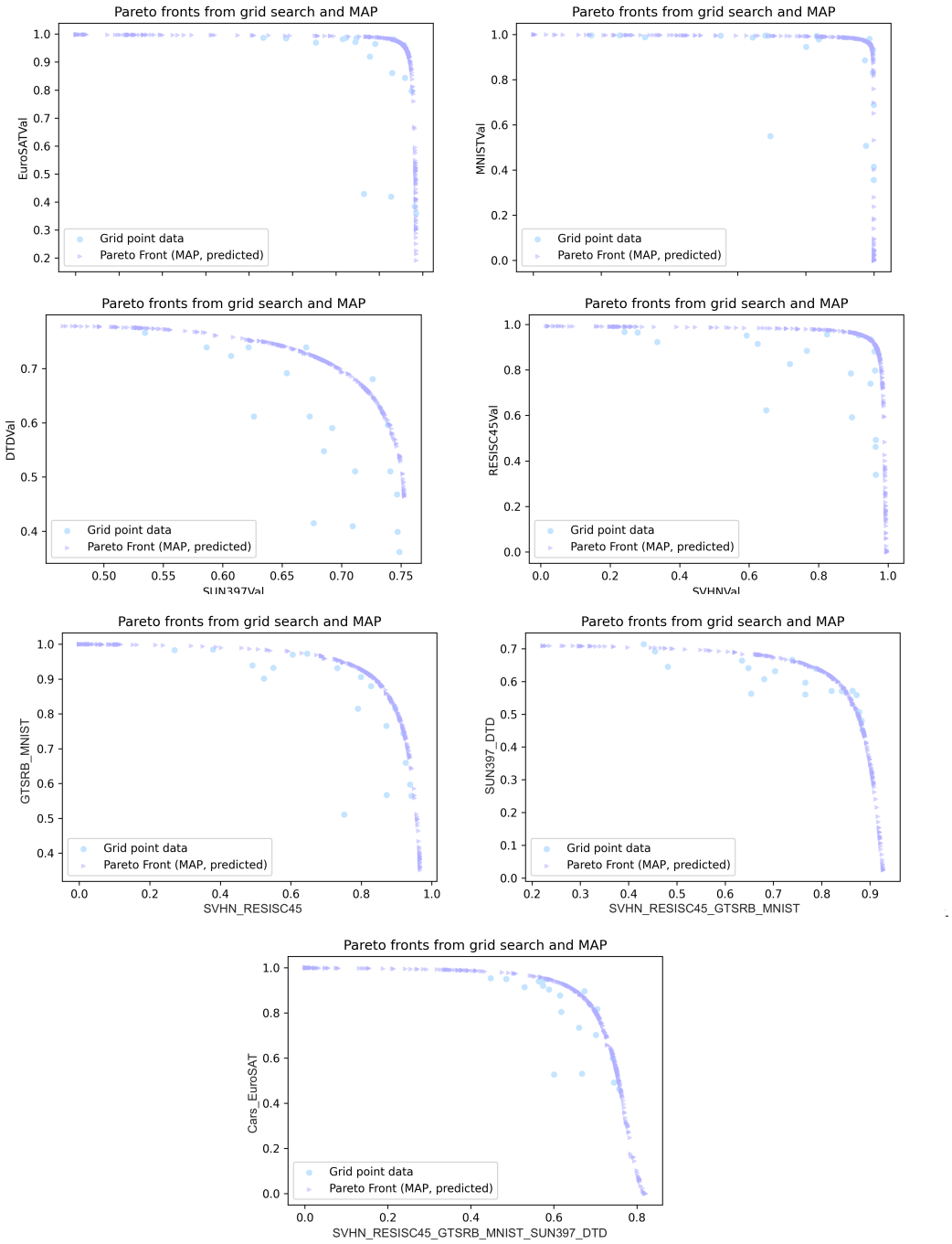



Figure 6: Illustration of the sequential steps involved in merging 8 models using nested-merging MAP (left to right, top to bottom). The figures show the intermediate Pareto fronts obtained, where $A\_B$ means the model obtained by merging model A and model B.

### 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 4, we performed another set of experiments merging four models using ruled-based performance. Rule-based preference means preference 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 preference 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 the rule-based preference, which requires all models should achieve 70% of the single-task fine-tuned model, while maximizing the performance on the first task (SUN397 (1) and GTSRB (3)).

Table 6: 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 require 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 Figure 7 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 10.

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.

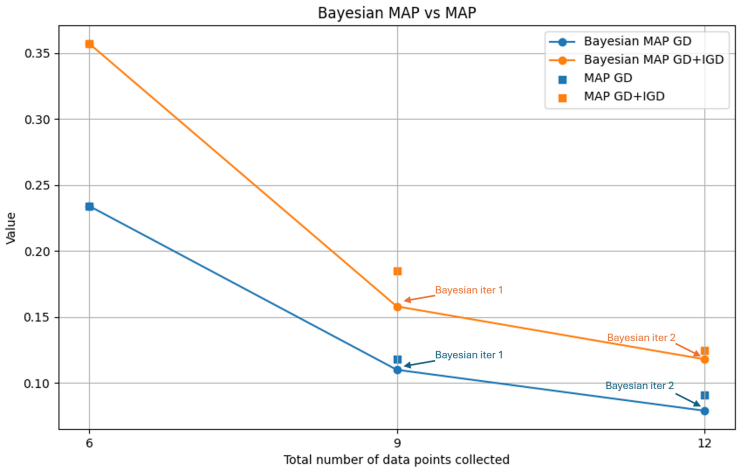



Figure 7: Bayesian MAP compared with MAP. The x-axis denotes the number of points used by either MAP or Bayesian MAP, and the y-axis denotes the value for IGD or GD. We compared MAP with 6, 9, 12 points and Bayesian MAP with 6 initial points, and sampled 3 more points each round for two rounds.

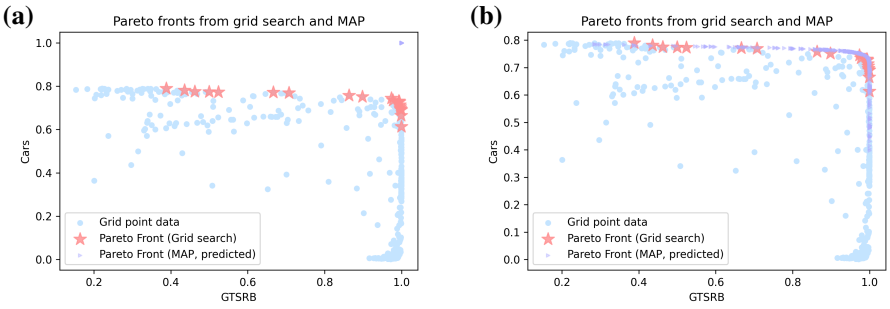



Figure 8: (a) This plot shows a failure case of amortized Pareto front when we only have 6 initial randomly sampled pairs of $(\mathbf{c}, \{M_i(\theta_m(\mathbf{c}))\}_{i=1}^N)$ (b) After one iteration of Bayesian adaptive sampling for 3 more pairs (9 in total), the amortized Pareto front is much better 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 $\alpha$ is a multi-index of non-negative integers, $D^\alpha f(a)$ denotes the partial derivatives of $f$ at $a$ corresponding to $\alpha$, 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 $\alpha$ with $|\alpha| \equiv k+1$, the norm of the tensor $D^\alpha f$ evaluated at any point $\xi$ 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 9 (a).

## F.2 Merging language model

We merged French and Arabic, as well as Chinese and Japanese. Please refer to Table 7 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.

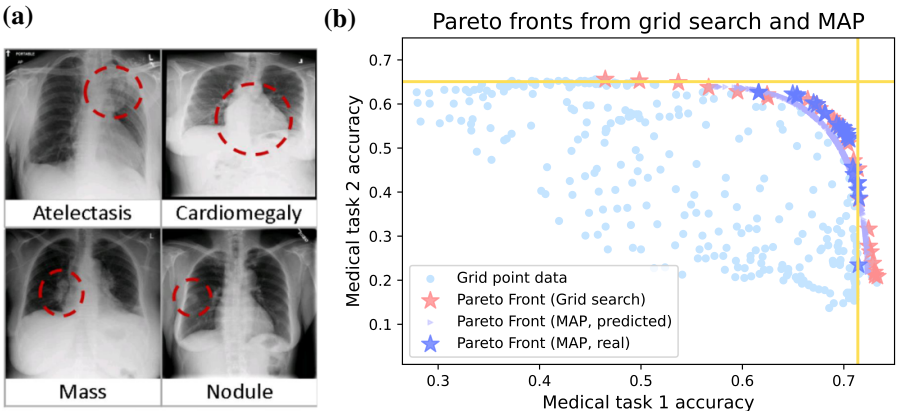



Figure 9: (a) Example figure from the NIH [42] dataset. (b) Pareto fronts found by brute-force direct search using 400 points and by MAP using 30 points. We randomly sampled 25 points from the predicted Pareto front by MAP. The resulting IGD is 0.016, and GD is 0.014.

Table 7: 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 | $0.014_{0.013}$ | $0.028_{0.017}$ | $0.041_{0.026}$ |

### 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 11. 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.

## 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 $\Omega$ 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

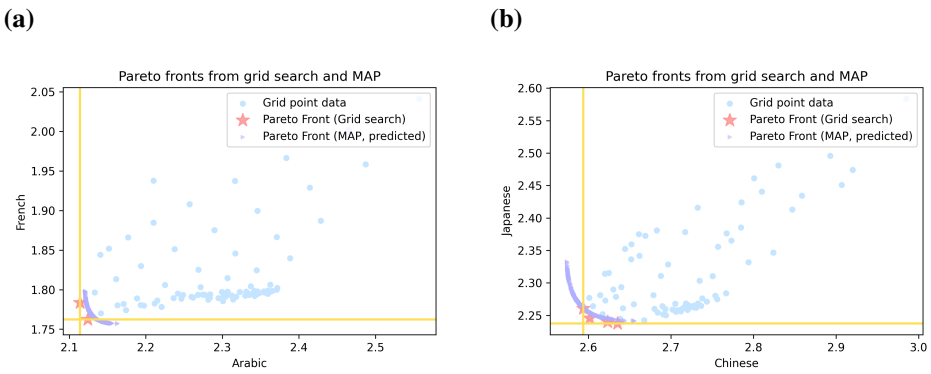



Figure 10: (a) Amortized Pareto front on merging Llama-3 fine-tuned on French with Llama-3 fine-tuned on Arabic; (b) Amortized Pareto front on merging Llama-3 fine-tuned on Chinese with Llama-3 fine-tuned on Japanese

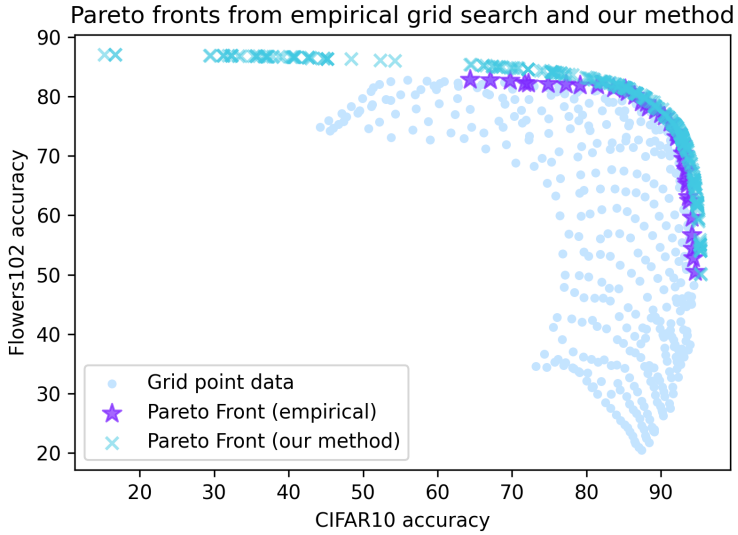



Figure 11: Pareto front obtained for two ResNet18 models on CIFAR10 and Flowers 102. We perform additional finetuning of the classification head after merging the model weights.

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 Algorithm 2

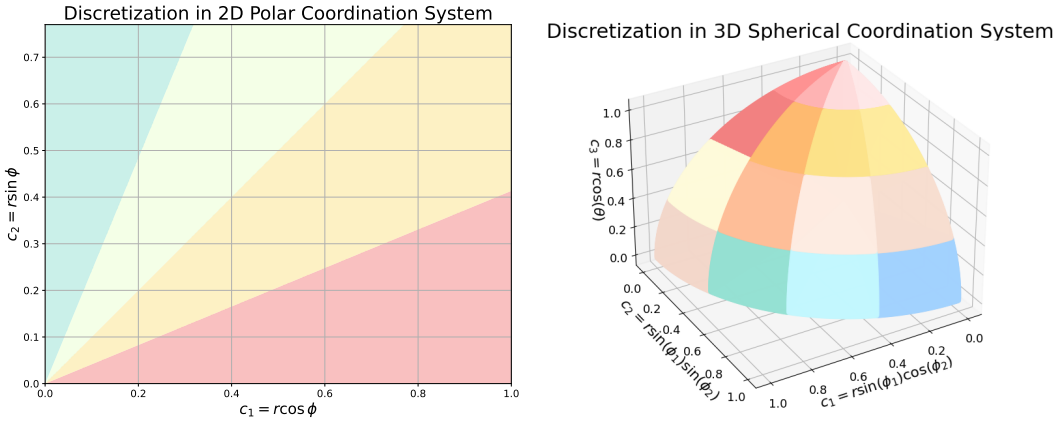
In this section, we explain the operations of the algorithm in Figure 3 in details. Here task 1 to 8 is Cars, GTSRB, DTD, SUN397, Resisc45, and SVHN. If we minimize (3) without the nested merging, we would need to estimate $\mathbf{A}_1, ..., \mathbf{A}_8 \in \mathbb{R}^{8\times 8}$, with hundreds of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

With the nested merging, for the first round, we merge $(\boldsymbol{\theta}_{ft}^1, \boldsymbol{\theta}_{ft}^2)$into $\boldsymbol{\theta}_{\text{merge}}^{1,2}$, 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

$(\boldsymbol{\theta}_{ft}^3, \boldsymbol{\theta}_{ft}^4)$into $\boldsymbol{\theta}_{\text{merge}}^{3,4}$, and $(\boldsymbol{\theta}_{ft}^5, \boldsymbol{\theta}_{ft}^6)$into $\boldsymbol{\theta}_{\text{merge}}^{5,6}$. Next, we merge $(\boldsymbol{\theta}_{\text{merge}}^{1,2}, \boldsymbol{\theta}_{\text{merge}}^{3,4})$into $\boldsymbol{\theta}_{\text{merge}}^{1,2,3,4}$, and finally into $\boldsymbol{\theta}_{\text{merge}}^{1,2,3,4,5,6,7,8}$.

### G.3 Algorithm 3

Algorithm 4 is a detailed version of Algorithm 3. Figure 12 includes illustration of our discretization method (how we create bins) in 2D and 3D decision variable (**c**) space.




(a) (b)

Figure 12: (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;

### G.4 NSGA-III

As the algorithm we used to find the Pareto front based on the surrogate models, NSGA-III is a multi-objective evolutionary algorithm for solving many-objective problems, typically involving more than three objectives. The algorithm's core intuition lies in maintaining population diversity while converging towards the Pareto-optimal front. It employs a population-based approach, evolving solutions over multiple generations, and utilizes non-dominated sorting to rank solutions based on their non-domination level, thus propelling the population towards the Pareto-optimal front. A key feature is the use of well-spread reference points on a normalized hyper-plane to maintain diversity in high-dimensional objective spaces. The algorithm generates offspring through genetic operators such as crossover and mutation, and implements elitism by preserving the best solutions from both parents and offspring in each generation. Objective values are normalized to handle varying scales, and solutions are associated with the closest reference points in this normalized space. Diversity is further maintained through niche preservation, selecting solutions associated with under-represented reference points. The main flow of NSGA-III involves initializing a population, then iterating through generations by creating offspring, combining parents and offspring, performing non-dominated sorting, selecting the best solutions for the next generation, and applying the reference point approach as needed to maintain diversity.

## 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

**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 12 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 $\text{mean}_k$
    {Bootstrap to estimate the standard deviation of the losses.}
    **for** $q = 0$ to $Q$ **do**
      Randomly (uniformly) drop $\alpha$ 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 $\text{std}_k$
    $\text{score}_k = \text{mean}_k + \frac{1}{2}\text{std}_k$
  Calculate probability distribution across the discretized bins by $\text{score}_k$ as the posterior sampling strategy in the next round
**return**

**Algorithm 5** NSGA-III Algorithm

**Input:** Population size $N$, number of generations $G$, set of reference points $\mathcal{R}$
**Initialize** the population $P$ with $N$ individuals
**for** $g = 1$ to $G$ **do**
  Generate offspring population $Q$ using crossover and mutation operators
  Combine parent and offspring populations: $R = P \cup Q$
  Perform non-dominated sorting on $R$ to identify fronts $F_1, F_2, \ldots$
  Initialize the new population $P = \emptyset$ and set $l = 1$
  **while** $|P| + |F_l| \leq N$ **do**
    Add the entire front $F_l$ to the population $P$
    $l = l + 1$
  **if** $|P| < N$ **then**
    Normalize objectives for individuals in $P \cup F_l$
    Associate each member of $P \cup F_l$ with a reference point in $\mathcal{R}$
    Compute niche count $\rho_j$ for each reference point $j$
    **while** $|P| < N$ **do**
      Find reference point $j_{min}$ with minimum $\rho_{j_{min}}$
      Select individual $i$ from $F_l$ associated with $j_{min}$
      Add $i$ to $P$ and remove from $F_l$
      Update niche count: $\rho_{j_{min}} = \rho_{j_{min}} + 1$
**Output:** Final population $P$

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 Table 4. 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 $625$ models $\times$ $4$ tasks $\times$ $1$ minute $=$ $2500$ 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 $\tilde{M}_2(\mathbf{c}; \mathbf{A}_2, \mathbf{b}_2, e_2)$.

# NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

**The checklist answers are an integral part of your paper submission.** They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We discuss the limitations throughout this work, including the inaccuracy of Taylor expansion in some cases and inefficiency of model merging when our nested and Bayesian techniques are not used.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: This paper has no theorem.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide the details in main text and appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will include open access in the camera-ready.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the details in main text and appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We include the errors in some tables.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the details in main text and appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the code.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite existing assets properly and checked the license.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper does not release new assets

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.