FW-Merging: Scaling Model Merging with Frank-Wolfe Optimization

Hao (Mark) Chen[1] Shell Xu Hu[2] Wayne Luk[1] Timothy Hospedales[2] Hongxiang Fan[1]
[1]Imperial College London, UK [2]Samsung AI Center, Cambridge, UK
hc1620@ic.ac.uk shell.hu@samsung.com w.luk@imperial.ac.uk t.hospedales@samsung.com
hongxiang.fan@imperial.ac.uk *

Abstract

*Model merging has emerged as a promising approach for multi-task learning (MTL), offering a data-efficient alternative to conventional fine-tuning. However, with the rapid development of the open-source AI ecosystem and the increasing availability of fine-tuned foundation models, existing model merging methods face two key limitations: (i) They are primarily designed for in-house fine-tuned models, making them less adaptable to diverse model sources with partially **unknown** model and task information, (ii) They struggle to scale effectively when merging **numerous** model checkpoints. To address these challenges, we formulate model merging as a constrained optimization problem and introduce a novel approach: **Frank-Wolfe Merging** (`FW-Merging`). Inspired by Frank-Wolfe optimization, our approach iteratively selects the most relevant model in the pool to minimize a linear approximation of the objective function and then executes a local merging similar to the Frank-Wolfe update. The objective function is designed to capture the desired behavior of the target-merged model, while the fine-tuned candidate models define the constraint set. More importantly, `FW-Merging` serves as an orthogonal technique for existing merging methods, seamlessly integrating with them to further enhance accuracy performance. Our experiments show that `FW-Merging` scales across diverse model sources, remaining stable with 16 irrelevant models and improving by **15.3%** with 16 relevant models on 20 CV tasks, while maintaining constant memory overhead—unlike the linear overhead of data-informed merging methods. Compared with the state-of-the-art approaches, `FW-Merging` surpasses the data-free merging method by **32.8%** and outperforms the data-informed Adamerging by **8.39%** when merging 20 ViT models. Our code is open-sourced at here.*

1. Introduction

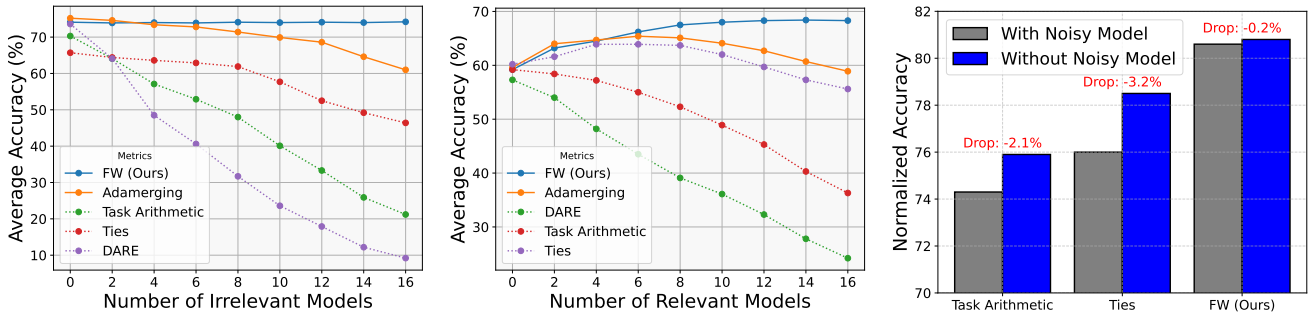
Multi-task learning (MTL)-based fine-tuning adapts a single pre-trained Large Language Model (LLM) for multiple downstream applications, reducing the deployment overhead of separately fine-tuning multiple models [53]. However, it still demands a large amount of high-quality data, which might only exist in the private domain [32], and significant compute resources [36]. To mitigate these issues, model merging has emerged as a promising technique for fusing fine-tuned models within the parameter space [14, 48, 50]. Existing model merging methods can be broadly classified into two categories: data-free methods [14, 15, 48], and data-informed methods [50, 51], which optimize merge coefficients based on additional data.

While these approaches have proven effective, several key limitations hinder their scalability and broader adoption. First, these methods adjust merging coefficients based on the known capabilities of the models on specific tasks to optimize performance, which is less robust when dealing with diverse model sources with unknown information[1]. This is primarily caused by the inability to distinguish high-quality models from poorly fine-tuned ones in an unknown model setting. Second, when scaling up these approaches to a large number of unknown models, these methods struggle and can result in significant performance degradation. As shown in Figure 1a, our profiling study demonstrates a performance reduction ranging from 18.9% to 64.4%. These limitations are **further amplified** by the fast-growing open-source AI ecosystem, where platforms such as Hugging Face have driven a surge in the release of powerful LLMs with many lacking complete information. Given that merging open-source models has repeatedly shown the potential to produce top-ranking LLMs on major model leaderboards [13], developing scalable and robust merging techniques is essential to harness the growing number of open-source models, further enhancing performance and widening the adoption of model merging.

To effectively scale model merging and leverage the vast collection of open-sourced models with unknown capabilities, the new model merging method must exhibit two fundamental scaling properties: *1) as more irrelevant models*

*Corresponding Authors: Shell Xu Hu and Hongxiang Fan

[1]This paper refers unknown information to: *1)* when open-source models are partially assessed on limited benchmarks, leaving their performance on other tasks unknown and costly to evaluate, and *2)* when models contain misleading information, polluting the merging process.



(a) 4 CV tasks with irrelevant model addition. (b) 20 CV tasks with relevant model addition. (c) NLP tasks with noisy model addition.

Figure 1. Performance scaling of `FW-Merging` across CV tasks. (a) demonstrate robustness to irrelevant models, while (b) show improved performance with relevant models. (c) analyzes performance degradation when incorporating a noisy model initialized from a different pre-trained checkpoint. Detailed results and experimental setup are discussed in Section 4.4.

are added to the merging pool, the performance should remain unaffected, and *2) as more relevant models are added to the merging pool, the performance should steadily increase, converging towards the optimal performance*. To this end, we revisit model merging and formulate it as a constrained optimization problem, where the objective function dictates the desirable behavior of the final merged model, and fine-tuned checkpoints form the constraint set. Inspired by Frank-Wolfe optimization, we introduce **Frank-Wolfe Merging** (`FW-Merging`), an iterative algorithm designed to enhance merging efficiency while maintaining robustness at scale. `FW-Merging` comprises three principal stages in each iteration: (1) **Relevance Evaluation**: Instead of merging models arbitrarily, we obtain the linear approximation of the objective function using gradients of the current model, revealing the most beneficial direction for improvement. (2) **Model Selection**: The most relevant checkpoints are selected from the constraint set by minimizing the linear approximation, ensuring that each step incorporates task-specific knowledge with minimal interference. (3) **Knowledge Integration**: The selected checkpoint is integrated using an orthogonal merging method, striking a balance between adaptation and stability in the merged model.

We demonstrate the effectiveness of `FW-Merging` with a diverse pool of fine-tuned checkpoints across various language and vision tasks, compared to both data-free and data-informed model merging methods as well as traditional MTL-based fine-tuning. As shown in Figure 1, `FW-Merging` satisfy our two fundamental scaling properties: accuracy performance does not drop when 16 irrelevant models are added (compared to a 49% drop in task-arithmetic) and steadily improves by 15.3% when 16 relevant models are included. Additionally, `FW-Merging` requires only constant memory overhead, as it selects and merges a fixed number of models at a time. In contrast, methods that optimize merging coefficients [50] or resolve parameter interference [48] must store all models in memory, leading to linear overhead. Moreover, `FW-Merging` exhibits greater robustness to noisy models lacking critical information, such as their initialization point. As shown in Figure 1c, `FW-Merging` experiences minimal performance degradation when a misinitialized model is introduced, whereas Ties suffers a performance drop of up to 3.2%. `FW-Merging` outperforms state-of-the-art data-free merging method by **32.8%** and the data-informed method Adamerging by **8.39%** when merging 20 ViT models. On the language benchmarks, `FW-Merging` achieves **6.3%** improvement over the best model merging method across discriminative and generative tasks, while even surpassing the performance of traditional MTL using only **3.4%** of its required data. Our results position `FW-Merging` as an effective solution to scale model merging to the next level.

Our contributions can be summarized as follows:

- Identify scalability and robustness issues in existing model merging techniques through experiments, highlighting the urgent need for large-scale model merging.
- Formulate model merging as a constrained optimization problem with an objective function that explicitly captures the desired behavior of the final merged model.
- Introduce Frank-Wolfe Merging, a novel iterative method that autonomously guides the merged model toward an optimized direction, even with large sets of black-box open-source checkpoints.
- Evaluate our proposed approach on extensive benchmarks, demonstrating its effectiveness and scalability.

2. Related Work

Efficient Multi-Task Learning. In traditional Multi-Task Learning (MTL), a single model is trained on a dataset containing multiple tasks to enable the model to acquire diverse capabilities [2]. However, a significant challenge in traditional MTL is the issue of negative transfer [16]. To mitigate this, architecture-based approaches have been developed, such as parameter sparsification [26, 39] and shared structure modularization [29, 30]. On the optimization side,

methods to resolve gradient conflicts [6, 54] and domination of gradient or learning rate [5, 24] have been proposed. With the rise of Large Language Models (LLMs), MTL faces additional challenges, particularly the high computational costs. To address these challenges, strategies like parameter-efficient fine-tuning [11, 20, 21] and memory-efficient fine-tuning [8, 22, 31] have been introduced to minimize both memory and computational resource usage. More recently, model merging has emerged as a promising approach to make MTL more compute- and data-efficient.

Model Merging. While pre-merging methods prepare favorable conditions for merging, during-merging techniques combine multiple neural networks into a single model while retaining or enhancing their capabilities [49]. In this work, we focus on during-merging methods. Early insights into neural network landscapes [10] revealed that linear interpolation between models exposes useful loss surface properties, laying the foundation for weight averaging—a core merging technique. Simple averaging widens optima and improves generalization [15], evolving into advanced methods like model soups [46] and heterogeneous model merging. Recent advances introduce more structured approaches, such as Fisher-Weighted Averaging [37], which incorporates Fisher information to weight parameters more effectively, and Permutation Alignment methods like Git Re-Basin [1], which address weight permutation symmetries. Interference Resolution techniques, including TIES [25] and DOGE [45], mitigate parameter conflicts either through explicit alignment or projective gradient descent. Task Arithmetic [33] enables weight-space operations to combine task-specific behaviors in language models, while Diversity-Aware Merging, such as DARE [23], leverages model diversity to improve sparse-to-dense integration. In contrast to the data-free methods mentioned above, data-informed methods [40, 50, 51] optimize merging coefficients using additional data. Model merging is impactful for LLMs, enabling efficient knowledge integration without full retraining, facilitating distributed fine-tuning [44], multi-task learning [35], and cost-effective model adaptation.

3. Method

3.1. Preliminary: Frank-Wolfe algorithm

The Frank-Wolfe (FW) algorithm [9], also known as the conditional gradient method, is an iterative optimization algorithm for constrained optimization problems of the form:

$$\min_{x \in \mathcal{C}} f(x) \tag{1}$$

where f is a continuously differentiable function, and $\mathcal{C}$ is a compact convex set. The algorithm follows an elegant geometric intuition: at each iteration t, FW first identifies which vertex of $\mathcal{C}$ yields the steepest descent direction and then moves towards this vertex to decrease the value of the objective function. More specifically, FW algorithm:

1. Constructs a linear subproblem of the original optimization (a.k.a. *linear minimization oracle*) using first-order Taylor expansion at the point x_t:
$$\mathrm{LMO}(\mathcal{C}, x_t) := \arg\min_{s \in \mathcal{C}} \langle s, \nabla f(x_t) \rangle \tag{2}$$
2. Finds the vertex s_t of the feasible set $\mathcal{C}$ by picking $s_t \in \mathrm{LMO}(\mathcal{C}, x_t)$.
3. Takes a careful step from the current point x_t towards this direction $s_t - x_t$, maintaining feasibility through the convex combination: $x_{t+1} = (1 - \gamma_t)x_t + \gamma_t s_t$.
The step size $\gamma_t \in [0, 1]$ can be chosen by line search
$$\gamma_t = \arg\min_{\gamma \in [0,1]} f\Big((1-\gamma)x_t + \gamma s_t\Big), \tag{3}$$

which ensures a sufficient decrease in $f(.)$ at each FW step.

To determine when to stop, the *FW gap* is used to measure the suboptimality in terms of the proximity to the best solution of LMO:

$$g_t := \max_{s \in \mathcal{C}} \langle -\nabla f(x_t), s_t - x_t \rangle, \tag{4}$$

which is non-negative by definition.

3.2. Frank-Wolfe Model Merging

We consider the problem of fine-tuning a pre-trained foundation model on new tasks. Given a pre-trained model θ_0 and previously fine-tuned models $\{\theta_1^*, \theta_2^*, \cdots, \theta_n^*\}$, we aim to fine-tune the $(n+1)$-th model θ_{n+1} on new tasks as a convex combination of the previous models with optimal merging coefficients.

To this end, we propose a Frank-Wolfe based model merging framework, which is described as follows.

$$\min_{\lambda} \ell\Big(\sum_{i=1}^{n} \lambda_i \theta_i^*\Big) \quad \text{s.t.} \quad \sum_{i=1}^{n} \lambda_i = 1, \quad \lambda_i \geq 0, \tag{5}$$

where $\lambda = \{\lambda_1, \ldots, \lambda_n\}$ are the merging coefficients, and ℓ is a loss function formulated for a specific goal, such as aligning the merged and individual models [45] or satisfying a task objective [50].

Potentially, a scaling issue of this formulation appears when the number of fine-tuned models n is large, since we need to keep all the fine-tuned models in memory. To address this, we propose a reformulation of the problem eq. (5) as follows:

$$\min_{\theta \in \mathcal{M}} \ell(\theta), \tag{6}$$

where $\mathcal{M} := \mathrm{conv}(\{\theta_i^*\}_{i=1}^n)$ is the convex hull of the set of previously fine-tuned models.

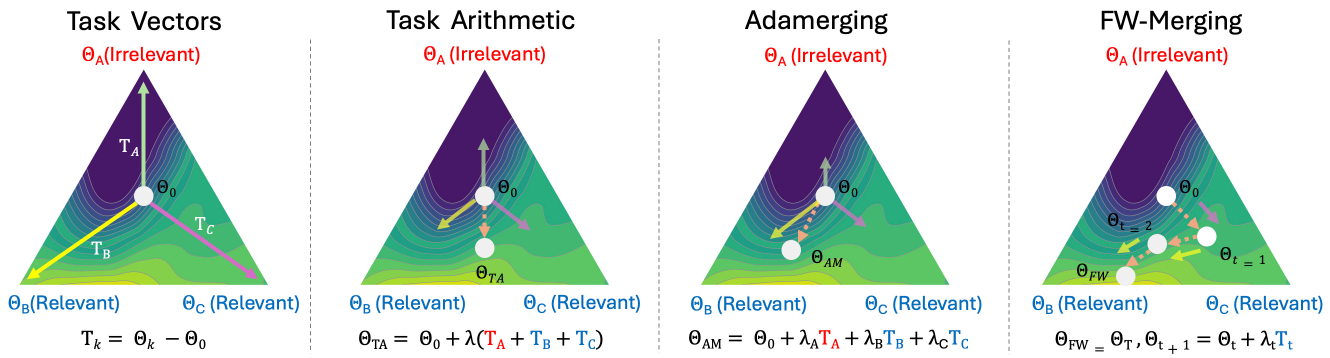


Figure 2. Illustration of model merging methods. Θ_A is an irrelevant model, while Θ_B and Θ_C are relevant models. Darker regions indicate higher objective function loss. Task Arithmetic treats all task vectors equally, failing to move optimally. Adamerging assigns different coefficients, moving towards more desirable direction but suffer from slow convergence due to interference from $\Theta_a A$. `FW-Merging` iteratively selects the most relevant model to merge and adapts step sizes, efficiently reaching the optimum after T iterations.

Proposition 1. *The optimization problems in equations* (5) *and* (6) *are equivalent.*

Proof. By definition of convex hull, any point $\theta \in \mathcal{M}$ can be written as a convex combination of the vertices $\{\theta_i^*\}_{i=1}^n$, i.e., $\theta = \sum_{i=1}^n \lambda_i \theta_i^*$ where $\lambda \in \Delta^n := \{\lambda \in \mathbb{R}^n \mid \sum_{i=1}^n \lambda_i = 1, \lambda_i \geq 0\}$. Therefore:

$$\min_{\theta \in \mathcal{M}} \ell(\theta) = \min_{\lambda \in \Delta^n} \ell\Big(\sum_{i=1}^n \lambda_i \theta_i^*\Big).$$

This shows that any solution of one problem can be mapped to a solution of the other problem with the same objective value. □

Since the FW algorithm requires the initial solution to be an interior point of the constraint set, we add the initial solution θ_0 to form a new constraint set $\mathcal{M} := \text{conv}(\mathcal{M} \cup \{\theta_0\})$, which we still denote as $\mathcal{M}$ for simple notations. A nice property of this reformulation is that the LMO can be simplified to

$$\text{LMO}(\{\theta_i^*\}_{i=1}^n, \theta_t) = \underset{s \in \{\theta_1^*, \ldots, \theta_n^*\}}{\arg\min} \langle \nabla \ell(\theta_t), s \rangle \qquad (7)$$

This is because for linear programming problems over convex sets, the optimal solution is always attained at the vertices of the constraint set. Algorithm 1 details the steps, and Figure 2 provides an overview.

3.3. Design choices of the algorithm

The above algorithm illustrates the key ingredients of Frank-Wolfe merging: LMO, stopping criterion g_t, line search routine, and the merging function. We discuss in this section the design choices of these components.

Merging function The main deviation from the classical FW algorithm is the reinterpretation of the FW update $x_{t+1} = (1 - \gamma_t)x_t + \gamma_t s_t$ as a local merging between θ_t and s_t. We denote by MergeFn the customizable merging function as long as the merged model stays in the convex hull $\mathcal{M}$. The most straightforward merging function is the convex combination:

$$\text{MergeFn}(\theta_t, s_t, \gamma_t) := (1 - \gamma_t)\theta_t + \gamma_t s_t, \qquad (8)$$

which corresponds to the Task-Arithmetic [33] method. The step size γ_t makes sure the merged model stays in $\mathcal{M}$.

It is natural to ask whether other existing model merging methods, such as TIES-Merging [25] and DARE [23] could also be used as MergeFn. The problem with these sophisticated merging methods is that the merged model might leave the constraint set, and thus violate the assumption of maintaining feasibility required by the classical FW theory. We verified in practice that these less rigorous merging functions might achieve better performance in certain cases but they generally cause more stability issues. Therefore, we do not consider these merging functions from the current comparison.

Algorithm 1: Frank-Wolfe Merging:

Input : Initial solution θ_0; Fine-tuned checkpoints $\{\theta_i^*\}_{i=1}^n$; FW budget T.
Output: Merged model θ^*_{merged}.

1: **if** $\theta_0 \notin \mathcal{M}$ **then** $\mathcal{M} := \mathcal{M} \cup \{\theta_0\}$
2: **for** $t = 0 \ldots T$ **do**
3: Let $s_t := \text{LMO}(\theta_t)$ and $d_t := s_t - \theta_t$
4: **if** $g_t := \langle -\nabla \ell(\theta_t), d_t \rangle \leq \epsilon$ **then**
5: **return** $\theta^*_{\text{merged}} \leftarrow \theta_t$
6: **end if**
7: Line-search: $\gamma_t \in \underset{\gamma \in [0,1]}{\arg\min}\, \ell\left(\theta_t + \gamma d_t\right)$
8: Update: $\theta_{t+1} := \text{MergeFn}(\theta_t, s_t, \gamma_t)$
9: **end for**
10: **return** $\theta^*_{Merged} \leftarrow \theta_T$

Hard FW v.s. Soft FW In the case of deep learning, the optimization problem is non-convex, additional efforts are needed to better characterize the loss landscape and prevent the LMO from being dominated by one or a few fine-tuned models, which occurs because the linear approximation of $\ell(\theta)$ is an inaccurate sketch of the original objective function. Instead of relying on the *argmin* of linear subproblem, we fetch the top-k vertices of LMO, $\{\tilde{s}_j\}_{j=1}^k$. A more subtle top-k operation can be performed in a task-wise fashion if the original objective function involves multi-tasks.

Given the top-k vertices, we now go back to eq. (5) to obtain the optimal merging coefficients $\{\lambda_j^*\}_{j=1}^k$. Note that this inner optimization[2] is a reduced version of original eq. (5) because hosting k models in memory would not be a problem. We also remove the line search step as this gives a new merging function of the form

$$\text{MergeFn}(\theta_t, \{\tilde{s}_j\}_{j=1}^k, \{\lambda_j^*\}_{j=1}^k) := \theta_t + \sum_{j=1}^{k} \lambda_j^*(\tilde{s}_j - \theta_t). \tag{9}$$

We call this oracle *soft* LMO in comparison to the argmin version which we call *hard* LMO.

Task-wise LMO v.s. layer-wise LMO The naive implementation of `FW-Merging` would vectorize the whole model weights θ and then solve LMO. We call this *task-wise* LMO. Since different layers contribute differently to model performance [52], a *layer-wise* LMO may yield better model merging. To incorporate this, the constraint set is redefined as a Cartesian product of convex hulls for each layer: $\mathcal{M} := \mathcal{M}_1 \times \cdots \times \mathcal{M}_L$, where L is the number of layers and $\mathcal{M}_l := \text{conv}\left(\{\theta_i^{*,l}\}_{i=1}^n\right)$. The LMO is then conducted layer-wise:

$$\text{LMO}(\{\theta_i^{*,l}\}_{i=1}^n, D, \theta_t^l) = \underset{s^l \in \{\theta_1^{*,l}, \ldots, \theta_n^{*,l}\}}{\arg\min} \langle \nabla \ell(\theta^t)^l, s^l \rangle. \tag{10}$$

This version can be viewed as a block-coordinate Frank-Wolfe algorithm [19], which is applied when the problem has a natural decomposition into blocks.

4. Experiments

4.1. Experiment Setup

Benchmarks. Our primary objective is to evaluate the effectiveness of our method in scenarios where the number of models greatly exceeds the number of evaluation tasks, and each model's capabilities are unknown in advance.

[2]For inner optimization, we use projected gradient descent with a projection of $\{\lambda_j\}_{j=1}^k$ onto the simplex after each gradient update.

- **Vision Tasks:** Following the setting of TALL [43], we use **20** ViT-B/32 models, each fine-tuned on a different vision task. The number of models to be merged is intentionally set to be significantly larger than the number of evaluation tasks, allowing us to assess the scalability of model merging methods. The evaluation benchmarks consist of four tasks: SUN397 [47], Stanford Cars [18], GTSRB [38], and DTD [7].
- **Language Discriminative Tasks:** We prepare **8** RoBERTa checkpoints [27] fine-tuned on eight tasks from the GLUE benchmark, following the practice in [28]. The merged model is then evaluated on four tasks from the GLUE benchmark [42]: MNLI, QNLI, QQP, and RTE.
- **Language Generative Tasks:** We collect **16** LLaMA2-7B models [41] fine-tuned with LoRA [12] on various tasks from Hugging Face. These models have unknown and uncontrolled capabilities, making them equivalently black-box models. Our goal is to evaluate the robustness of model merging methods in this challenging setting. The evaluation benchmarks include CNN/DM summarization [34], PubMedQA [17], and HumanEval [4].

Metrics. For vision tasks, we report the classification accuracy. Following [28], we report the average normalized score for language tasks to account for differences in task-specific score ranges to account for variations in task-specific score ranges. The normalized score is computed as $\text{Score}_{\text{normalized}} = \frac{1}{T}\sum_{t=1}^{T} \frac{\text{Score}(f(\boldsymbol{\theta}^*))}{\text{Score}(f_t(\boldsymbol{\theta}_t))}$.

Baselines. We compare `FW-Merging` with both data-informed and data-free model merging methods. For data-informed model merging, we compare `FW-Merging` against Adamerging [50], Surgery [51], and Concrete Merging [40]. To ensure a fair comparison, these methods are trained only on the same tasks as `FW-Merging`. For data-free model merging, we compare `FW-Merging` with Fisher Merging, Weight Averaging, RegMean Merging, Task Arithmetic [33], Ties-Merging [48], and DARE Merging [23] across both language and vision tasks. Additionally, we fine-tune one model on the discriminative language benchmark and another on the generative language benchmark to serve as additional baselines.

Implementations. We implement two `FW-Merging` variants: $\texttt{FW}_{\texttt{hard}}$, which uses hard FW with layer-wise LMO, and $\texttt{FW}_{\texttt{soft}}$, which employs soft FW with task-wise LMO (Section 3.3). For $\texttt{FW}_{\texttt{soft}}$, layer-wise coefficients are optimized via gradient descent on the training dataset to solve eq. 5, differing from Adamerging [50] by minimizing cross-entropy loss on training data rather than entropy on test samples. On language benchmarks, the training dataset consists of 100 samples per task, and $\texttt{FW}_{\texttt{hard}}$ runs

Table 1. Performance on 4 Discriminative Tasks when merging 8 RoBERTa and 3 Generative Tasks when merging 16 LLaMA2-7B.

Method	4 Disc. Tasks (8 Models)	3 Gen. Tasks (16 Models)	Avg. Normalized Score
Pretrained	49.6	77.1	63.4
Traditional MTL	73.1	81.2	77.2
Task Arithmetic (w/ DARE)	77.3	16.8	47.1
Ties-Merging (w/ DARE)	75.6	46.6	61.1
Task Arithmetic	80.8	75.9	78.4
Ties-Merging	64.3	78.5	71.4
$\texttt{FW}_{\texttt{hard}}$ (Ours)	**85.4**	**81.1**	**83.1**

Table 2. Performance on 4 CV Tasks when merging 20 ViT-B/32.

Method	SUN397	Cars	GTSRB	DTD	Avg.
Pretrained	62.3	59.7	32.6	43.8	49.6
DARE (TIES)	5.9	2.3	16.7	11.8	9.2
Task Arithmetic	20.4	12.2	29.8	22.3	21.2
Ties-Merging	51.0	36.2	57.7	40.6	46.4
Weight Averaging	64.2	59.6	43.1	46.5	53.4
Fisher Merging	64.6	63.8	43.0	46.9	54.6
RegMean	65.5	62.2	59.7	53.9	60.3
LW Concrete AM	62.5	60.3	88.0	54.7	66.3
Adamerging	66.4	70.1	95.1	64.0	73.9
Surgery	69.7	71.8	96.6	73.4	77.9
$\texttt{FW}_{\texttt{hard}}$ (Ours)	66.5	69.9	95.1	64.5	74.0
$\texttt{FW}_{\texttt{soft}}$ (Ours)	**72.9**	**74.8**	**96.8**	**76.0**	**80.1**

for 10 iterations, initialized with Task Arithmetic's merged model. For vision tasks, it runs for 3 iterations. $\texttt{FW}_{\texttt{soft}}$ runs for 15 iterations on vision benchmarks, initialized with the pre-trained model. Training datasets consist of 100 samples from MNLI, QNLI, QQP, and RTE [42] for discriminative tasks; CNN/DM [34], CodeAlpaca-20k [3], and PubMedQA [17] for generative tasks; and SUN397 [47], Stanford Cars [18], GTSRB [38], and DTD [7] for vision tasks.

4.2. Comparison with Model Merging Methods

We evaluate `FW-Merging` against both data-informed and data-free model merging approaches across language and vision benchmarks. Table 1 reports the results for language tasks, including both discriminative and generative settings, while Table 2 presents results on vision benchmarks.

Language Tasks. $\texttt{FW}_{\texttt{hard}}$ achieves the highest average normalized score across language benchmarks, consistently surpassing prior model merging baselines. Specifically, $\texttt{FW}_{\texttt{hard}}$ improves upon Task Arithmetic by 4.6 points, Ties-Merging by 11.7, and DARE (Ties) by 9.8 on discriminative tasks. Table 4 shows that $\texttt{FW}_{\texttt{hard}}$ also outperforms data-informed Adamerging by 5.9 points. For generative tasks, $\texttt{FW}_{\texttt{hard}}$ outperforms Task Arithmetic by 5.2 points, Ties-Merging by 2.6, and DARE (Ties) by 34.5. Interestingly, while Task Arithmetic outperforms Ties-Merging on discriminative tasks by a margin of 16.5 points, it lags behind by 2.6 points on the more challenging generative tasks. This discrepancy likely arises from increased interference among task vectors as more checkpoints are merged. Unlike Ties-Merging, which explicitly resolves merging conflicts, Task Arithmetic lacks a reconciliation mechanism, making it more susceptible to such interference. In contrast, $\texttt{FW}_{\texttt{hard}}$ consistently outperforms both Ties-Merging and Task Arithmetic by selectively merging only the most relevant model parameters in each iteration. This targeted approach effectively mitigates interference, leading to more stable and robust performance across both discriminative and generative tasks.

Vision Tasks. $\texttt{FW}_{\texttt{soft}}$ achieves state-of-the-art performance across multiple vision benchmarks, surpassing data-informed methods like Adamerging and Surgery. As shown in Table 2, $\texttt{FW}_{\texttt{hard}}$ surpasses Adamerging, Concrete Merging, and all data-free merging methods in overall performance. Additionally, $\texttt{FW}_{\texttt{soft}}$ attains the highest accuracy (80.1%), outperforming Adamerging by 6.2% and Surgery by 2.2%. Unlike Surgery, which requires additional task-specific parameters and multiple forward passes per inference, our approach efficiently adapts to diverse visual tasks without increasing storage or inference complexity.

In general, data-free merging methods show significantly lower performance compared to data-informed approaches while merging a large number of models, when the models' capabilities do not precisely align with the evaluation tasks. This limitation arises because data-free methods treat all models equally, merging them without considering their unique capabilities, which amplifies interference between models. In contrast, data-informed merging methods achieve superior performance by optimizing merging coefficients on datasets as they allow for explicit control over desirable capabilities. `FW-Merging`, in particular, enhances scalability via hard model selection based on the linear approximation minimization.

4.3. Comparison with Traditional MTL

We compare `FW-Merging` with models fine-tuned using traditional MTL on discriminative and generative tasks. In each case, one single model is fine-tuned across all tasks, with performance and computational cost reported

Table 3. Merging Methods' Performance vs. Number of Models when Adding Relevant vs. Irrelevant Models.

#Models	4 CV Tasks When "Irrelevant" Models Added					20 CV Tasks When "Relevant" Models Added				
	DARE	Task	Ties	AM	$\texttt{FW}_{\texttt{soft}}$	DARE	Task	Ties	AM	$\texttt{FW}_{\texttt{soft}}$
4	73.6	70.3	65.7	**75.2**	74.1	57.3	59.2	**60.2**	59.6	59.2
6	64.1	64.1	64.4	**74.6**	73.9	54.0	58.4	61.6	**64.0**	63.2
8	48.5	57.1	63.6	73.4	**74.0**	48.2	57.2	63.9	**64.7**	64.5
10	40.6	52.9	62.9	72.8	**73.9**	43.5	55.0	63.9	65.4	**66.2**
12	31.7	47.9	61.9	71.4	**74.1**	39.1	52.3	63.7	65.1	**67.5**
14	23.6	40.1	57.7	69.9	**74.0**	36.1	48.9	62.0	64.1	**68.0**
16	17.9	33.3	52.5	68.6	**74.1**	32.3	45.3	59.7	62.7	**68.3**
18	12.2	25.9	49.2	64.6	**74.0**	27.8	40.3	57.3	60.7	**68.4**
20	9.2	21.2	46.4	61.0	**74.2**	24.2	36.3	55.6	58.9	**68.3**

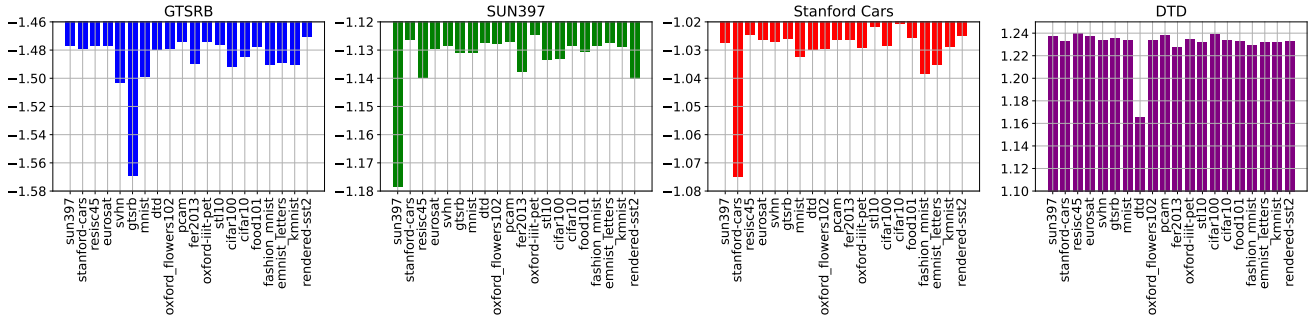


Figure 3. Linear Approximation of the Objective Function of Model Checkpoints Across Different Tasks in a Frank-Wolfe Iteration. The x-axis represents the checkpoints, and each graph shows the linear approximation result for each task.

Table 4. Costs and Perf. of methods on NLP discriminative tasks.

Method	Data Samples/Task	Time Cost	Perf.
Traditional MTL	2.9K	4.2h	73.1
Data-free Merging	0	0	80.8
Data-informed Merging	1.6K	2min	79.5
FW Merging	100	2min	85.4

in Table 1 and Table 4. Traditional MTL achieves an average score of 77.2, lower than that of FW-Merging (83.1). On discriminative tasks, MTL scores 73.1, trailing FW-Merging (85.4). For generative tasks, MTL scores 81.2, while FW-Merging closely matches it at 80.8, suggesting that FW-Merging's performance matches that of traditional MTL.

As shown in Table 4, FW-Merging demonstrates a substantial advantage in efficiency. Traditional MTL requires fine-tuning on 2.9K samples per task and takes 4.2 hours of training time, which is computationally intensive. In contrast, FW-Merging only requires 100 training samples per task and completes the merging process in just 2 minutes. This huge reduction in computational cost underscores the effectiveness of FW-Merging compared to traditional MTL. Moreover, FW-Merging has a key advantage over traditional MTL: while MTL requires a large volume of high-quality data for optimal performance, FW-Merging needs only a small set of data because: *1)* it optimizes merging coefficients based on models' characteristics, which simplifies the optimization space, and *2)* it uses model weights as inputs, which are much more information-dense representations than data, enabling more efficient objective learning.

FW-Merging is a post-training technique that does not require access to original training data, making it ideal for privacy-sensitive or data-scarce scenarios. Overall, the results suggest that FW-Merging is a scalable, efficient alternative to traditional MTL, providing comparable performance at a reduced computational cost.

4.4. Scaling to More Models and Tasks

We investigate the performance scaling of different merging methods with the number of models, as shown in Figure 1a, Figure 1b, and Table 3. In large-scale model merging, models from open-source platforms vary in quality. To simulate this, we use 20 ViT-B/32 models fine-tuned on tasks that are either included in the evaluation benchmark or not. A model is *irrelevant* if its fine-tuning dataset does not match the training split of the evaluation task, and *relevant* if it matches. To ensure fair comparison, the total number of training iterations run by $\texttt{FW}_{\texttt{soft}}$ is the same as that of Adamreging.

As shown in Table 3, adding *irrelevant* models sharply reduces the performance of data-free methods: DARE by

Table 5. Ablation on design variants of FW-Merging.

Coefficient λ	Method	LMO	Score
Vision Tasks			
Optimized	$\texttt{FW}_{\texttt{soft}}$	Layer-wise	79.7
Optimized	$\texttt{FW}_{\texttt{soft}}$	Task-wise	80.1
Unoptimized	$\texttt{FW}_{\texttt{soft}}$	Layer-wise	69.8
Unoptimized	$\texttt{FW}_{\texttt{soft}}$	Task-wise	70.3
-	$\texttt{FW}_{\texttt{hard}}$	Layer-wise	74.0
-	$\texttt{FW}_{\texttt{hard}}$	Task-wise	73.7
NLP Discriminative Tasks			
-	$\texttt{FW}_{\texttt{hard}}$	Layer-wise	85.4
-	$\texttt{FW}_{\texttt{hard}}$	Task-wise	78.2

64.4%, Task Arithmetic by 49.1%, and Ties by 19.1%, likely due to task interference and equal treatment of all models. Data-informed methods degrade less, with Adamerging dropping by 14.2%. In contrast, $\texttt{FW}_{\texttt{soft}}$ remains highly stable, fluctuating only from 73.9% to 74.1% as more models are added. In Figure 3, we examine the linear approximation of different checkpoints for a specific task and find that the model fine-tuned on the task consistently yields the most negative linear approximation. This indicates that in the Frank-Wolfe update, the most relevant checkpoint is chosen as the direction for merging, allowing FW-Merging to iteratively improve the merged model in the optimized direction within the constraint set. The inner product between gradients and model parameters serves as a reliable indicator of model relevance, with minized computational cost, further demonstrating FW-Merging 's scalability even in the presence of irrelevant models.

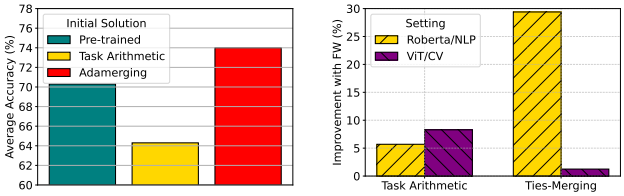
Adding *relevant* models should ideally improve performance, but data-free methods still degrade as shown in Table 3: DARE by 33.1%, Task Arithmetic by 22.9%, and Ties by 4.6%, with Ties performing best by mitigating parameter conflicts. Data-informed methods like Adamerging fluctuate between 58.9% and 64.7% as merging complexity increases, whereas $\texttt{FW}_{\texttt{soft}}$ steadily improves from 59.2% to 68.3% by iteratively selecting the most relevant models, facilitating smoother convergence. These results underscore FW-Merging 's effectiveness as a scalable solution for large-scale model merging.

4.5. Ablation Studies

Design variants. Table 5 compares the design variants of FW-Merging (Section 3.3). Task-wise LMO aligns better with $\texttt{FW}_{\texttt{soft}}$, improving performance slightly by 0.5 points over layer-wise LMO, while layer-wise LMO is more effective for $\texttt{FW}_{\texttt{hard}}$, especially on language tasks, yielding a 7.2-point gain. This is likely because $\texttt{FW}_{\texttt{soft}}$ optimizes layer-wise coefficients during merging, reducing the impact of layer-wise selection.

$\texttt{FW}_{\texttt{soft}}$ excels when merging a large number of models, outperforming $\texttt{FW}_{\texttt{hard}}$ by up to 6.7 points. Its ability to select multiple optimal directions per iteration allows it to navigate the parameter space efficiently.

Optimizing merging coefficients λ further improves performance by up to 9.9 points, underscoring the importance of weighting model parameters based on their relevance.



(a) Initial soluiton (b) Merging functions

Figure 4. Ablation on FW-Merging. (a) reports accuracies on the vision benchmark, while (b) on vision and language benchmarks.

Initial solution. We examine the effect of initialization on FW-Merging. An ideal initial solution should either (1) be closer to the global optimum or (2) expand the constraint set with a more meaningful search space. As shown in Figure 4a, initializing FW-Merging with the Adamerging result improves performance compared to starting from the pre-trained model, likely because Adamerging is closer to the optimal point. In contrast, task arithmetic leads to worse performance than the pre-trained model, potentially due to its poor performance on vision tasks (21.2%), suggesting it starts further from the optimum. Consequently, more FW iterations are required to achieve convergence.

Flexibility of merging functions. Although only a restricted set of merging functions ensure that FW-Merging remains within the convex hull, we demonstrate the flexibility of FW-Merging by showing its ability to enhance alternative merging functions. As shown in Figure 4b, applying FW-Merging with both Task Arithmetic and Ties-Merging improves performance on NLP and vision tasks, even though Ties-Merging does not necessarily stay within the convex hull. This suggests that FW-Merging remains effective across different merging functions.

5. Conclusion

In this work, we extend model merging to a more challenging setting where the merging pool consists of a large number of black-box fine-tuned checkpoints. While existing methods require prior knowledge of model details to achieve optimized performance, our proposed Frank-Wolfe Merging (FW-Merging) scales effectively with a large number of black-box models, iteratively refining the merged model towards the optimal point defined by an objective function. Experiments demonstrate that FW-Merging achieves superior performance and scalability, paving the way for next-generation model merging.

6. Acknowledgement

The support of the United Kingdom EPSRC (grant number UKRI256, EP/V028251/1, EP/N031768/1, EP/S030069/1, and EP/X036006/1), Intel, and AMD is gratefully acknowledged.

References

[1] Samuel K Ainsworth, Jonathan Hayase, Siddhartha Srinivasan, Khaled Saab, and Stefano Fusi. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022. 3

[2] Rich Caruana. Multitask learning. *Machine learning*, 28: 41–75, 1997. 2

[3] Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca, 2023. 6

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021. 5

[5] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018. 3

[6] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33:2039–2050, 2020. 3

[7] 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. 5, 6

[8] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[9] Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956. 3

[10] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014. 3

[11] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 3

[12] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 5

[13] Hugging Face. Open LLM Leaderboard, 2024. Accessed: March 4, 2025. 1

[14] 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. 1

[15] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018. 1, 3

[16] Junguang Jiang, Baixu Chen, Junwei Pan, Ximei Wang, Dapeng Liu, Jie Jiang, and Mingsheng Long. Forkmerge: Mitigating negative transfer in auxiliary-task learning. *Advances in neural information processing systems*, 36:30367–30389, 2023. 2

[17] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, 2019. 5, 6

[18] 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. 5, 6

[19] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *International Conference on Machine Learning*, pages 53–61. PMLR, 2013. 5

[20] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021. 3

[21] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021. 3

[22] Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*, 2023. 3

[23] Fuzhao Liu, Xiaotian Hu, Chengyu Chen, Zhijian Wang, Shuxin Xiao, Zhen Wang, Zhongxiang Wang, Chuanxin Xie, Eric Xing, and Song Han. Dare: Diversity-aware model merging for sparse-to-dense mixture-of-experts. *arXiv preprint arXiv:2402.10887*, 2024. 3, 4, 5

[24] Liyang Liu, Yi Li, Zhanghui Kuang, J Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. iclr, 2021. 3

[25] Prateek Yadav Liu et al. Resolving interference when merging models. *arXiv preprint arXiv:2306.01708*, 2023. 3, 4

[26] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019. 2

[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 5

[28] Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. *Advances in Neural Information Processing Systems*, 37:78905–78935, 2025. 5

[29] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1930–1939, 2018. 2

[30] Jiaqi Ma, Zhe Zhao, Jilin Chen, Ang Li, Lichan Hong, and Ed H Chi. Snr: Sub-network routing for flexible parameter sharing in multi-task learning. In *Proceedings of the AAAI conference on artificial intelligence*, pages 216–223, 2019. 2

[31] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023. 3

[32] Priyanka Mary Mammen. Federated learning: Opportunities and challenges. *arXiv preprint arXiv:2101.05428*, 2021. 1

[33] Eric Mitchell, Kenton Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022. 3, 4, 5

[34] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016. 5, 6

[35] Shraman Prakash, Rohit Girdhar, Devamanyu Hazarika, Joao Carreira, and Andrew Zisserman. Unified model for image, video, audio and language tasks. *arXiv preprint arXiv:2307.16184*, 2023. 3

[36] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020. 1

[37] Sidak Pal Singh and Martin Jaggi. Merging models with fisher-weighted averaging. *arXiv preprint arXiv:2111.09832*, 2021. 3

[38] 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. 5, 6

[39] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33:8728–8740, 2020. 2

[40] Anke Tang, Li Shen, Yong Luo, Liang Ding, Han Hu, Bo Du, and Dacheng Tao. Concrete subspace learning based interference elimination for multi-task model fusion. *arXiv preprint arXiv:2312.06173*, 2023. 3, 5

[41] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 5

[42] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018. 5, 6

[43] Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. *arXiv preprint arXiv:2405.07813*, 2024. 5

[44] Zirui Wang, Daniel Duckworth, Shashank Nag, Kshitij Patil, Jonathon Shlens, Ekin D Cubuk, Barret Zoph, Joseph Campbell, Mitchell Wortsman, and Google Brain. lo-fi: distributed fine-tuning without communication. *arXiv preprint arXiv:2210.11948*, 2022. 3

[45] Yongxian Wei, Anke Tang, Li Shen, Feng Xiong, Chun Yuan, and Xiaochun Cao. Modeling multi-task model merging as adaptive projective gradient descent. *arXiv preprint arXiv:2501.01230*, 2025. 3

[46] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *arXiv preprint arXiv:2203.05482*, 2022. 3

[47] 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. 5, 6

[48] 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*, 36:7093–7115, 2023. 1, 2, 5

[49] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities, 2024. *URL https://arxiv.org/abs/2408.07666*, 2408. 3

[50] 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. 1, 2, 3, 5

[51] Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representa-

tion surgery for multi-task model merging. *arXiv preprint arXiv:2402.02705*, 2024. 1, 3, 5

[52] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014. 5

[53] Jun Yu, Yutong Dai, Xiaokang Liu, Jin Huang, Yishan Shen, Ke Zhang, Rong Zhou, Eashan Adhikarla, Wenxuan Ye, Yixin Liu, et al. Unleashing the power of multi-task learning: A comprehensive survey spanning traditional, deep, and pretrained foundation model eras. *arXiv preprint arXiv:2404.18961*, 2024. 1

[54] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020. 3