StatsMerging: Statistics-Guided Model Merging via Task-Specific Teacher Distillation

Anonymous Author(s)
Affiliation
Address
email

Abstract

As large models are increasingly deployed across various tasks, the limited GPU memory available for storing and executing task-specific models presents a growing bottleneck. Model merging has emerged as a promising solution to accommodate multiple large models within constrained memory budgets. While traditional multi-task learning methods attempt to merge shared layers, they require labor-intensive annotated labels and incur significant computational overhead. Recent merging techniques aim to address this issue by combining models at inference time; however, these approaches often rely on simplistic heuristics, ignore weight distribution characteristics, assume architectural identity, or require access to test samples to infer merging coefficients, thereby limiting their generalization capability and scalability. We present *StatsMerging*, a novel lightweight learning-based model merging method guided by weight distribution statistics without requiring ground truth labels or test samples. *StatsMerging* offers three key advantages: (1) It uniquely leverages singular values from singular value decomposition (SVD) to capture task-specific weight distributions, serving as a proxy for task importance to guide task coefficient learning; (2) It employs a lightweight learner *StatsMergeLearner* to model the weight distributions of task-specific pre-trained models, improving generalization and enhancing adaptation to unseen samples; (3) It introduces *Task-Specific Teacher Distillation* for merging vision models with heterogeneous architectures, a merging training paradigm that avoids costly ground-truth labels by task-specific teacher distillation. Notably, we present two types of knowledge distillation, (a) distilling knowledge from task-specific models to train *StatsMergeLearner*; and (b) for the first time, distilling knowledge from models with different architectures prior to merging, following a distill-then-merge paradigm. Extensive experiments across eight tasks demonstrate the effectiveness of *StatsMerging*. Our results show that *StatsMerging* outperforms state-of-the-art techniques in terms of overall accuracy, generalization to unseen tasks, and robustness to image quality variations.

1 Introduction

Computer vision has witnessed transformative progress fueled by deep learning, particularly through the development and adoption of large-scale pre-trained models. Architectures like Convolutional Neural Networks (CNNs) (Krizhevsky et al., 2012; He et al., 2016; Simonyan and Zisserman, 2014), Vision Transformers (ViTs) (Dosovitskiy et al., 2021b; Touvron et al., 2021), and hybrid approaches (Liu et al., 2022) pre-trained on massive datasets have become the cornerstone of modern vision applications. Large-scale models leveraging multi-modal pre-training, such as CLIP (Radford et al., 2021)) or generative models like GANs (Goodfellow et al., 2014) and Diffusion Models (Ho et al., 2020; Rombach et al., 2022) have further pushed the boundaries of visual understanding and synthesis,

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

enabling the use of pre-trained backbones to a wide range of downstream vision applications. The dominant practice is to fine-tune these powerful base models to computer vision tasks, including image classification (He et al., 2016), object detection (Ren et al., 2015; Carion et al., 2020a), semantic segmentation (Long et al., 2015; Xie et al., 2021), image restoration (Zhang et al., 2017; Saharia et al., 2022), and image generation (Mirza and Osindero, 2014). This success, however, leads to a practical challenge: the proliferation of numerous specialized pre-trained weights and model checkpoints (Cao et al., 2024a, 2025), most of which share the same foundational ViT or CNN backbones. Managing this growing collection incurs significant storage overhead, complicates deployment, and represents a missed opportunity to consolidate the related, yet specialized, knowledge contained within these models (Wortsman et al., 2022), particularly on compute-constrained platforms such as edge devices (Cao et al., 2024b; Singh et al., 2024). While Multi-Task Learning (MTL) (Vandenhende et al., 2022b) aims to create versatile single models for vision tasks, it often demands complex joint training strategies, concurrent access to diverse datasets, and careful architecture design to balance performance across disparate tasks.

Model merging offers a compelling post-hoc alternative, seeking to combine independently trained models without expensive retraining. However, while techniques for model merging have gained traction, particularly in Natural Language Processing (NLP) (Yadav et al., 2023a; Ilharco et al., 2023), adapting these techniques in computer vision domain has far less explored. A straightforward approach of simple weight averaging (Wortsman et al., 2022) often fails in vision tasks due to the complex, hierarchical visual feature representations, task-specific optimizations, and the presence of intricate noise patterns that lead to sharp, non-convex loss minima (Izmailov et al., 2018). Recent methods in this direction (Matena and Raffel, 2022; Jin et al., 2023; Yang et al.; Padmanabhan et al., 2023) neglect the importance of weight distribution.

This paper introduces a novel model merging framework specifically designed to address the aforementioned challenges within computer vision. We propose *StatsMerging*, a weight distribution statistics-guided merging approach that moves beyond simple parameter averaging or task-vector manipulation. *StatsMerging* leverages the statistical features models pre-trained on prior tasks for merged. In particular, we compute salient statistics extracted by leverage Singular Value Decomposition (SVD) to capture the dominant properties of the learned feature spaces. This statistical information, intrinsically capturing aspects of the pre-trained model distributions, guides the merging process by learning a compact Multilayer Perceptron (MLP), coined *StatsMergeLearner* that predicts adaptive merging coefficients (λ) shown in Fig. 1. This allows the merging to be guided by the weight landscape, rather than treating coefficients as free parameters requiring external tuning data.

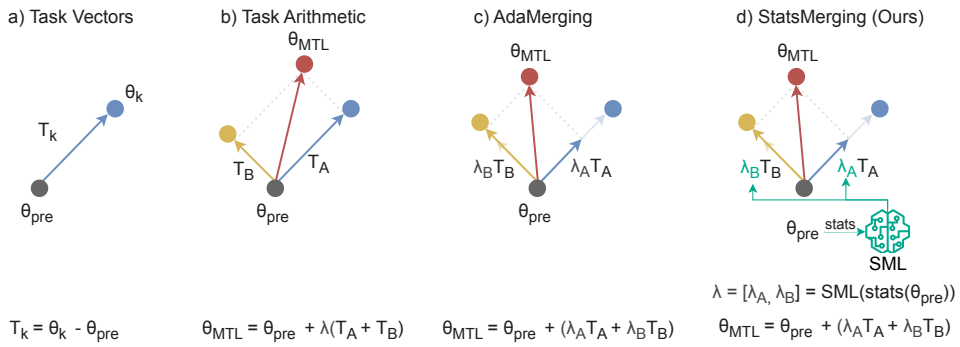

Figure 1: Compared to prior works, *StatsMerging* uniquely learns the merging coefficients using *StatsMergeLearner*, taking advantage of statistical features of weigts pre-trained on prior tasks. Notably, while both AdaMerging and *StatsMerging* are presented in the task-wise level in c) and d) for simplicity of illustration, the same principle can be applied at the layer-wise level for fine-grained adaptation.

We make four significant contributions summarized as follows:

- We propose *StatsMerging*[1], a novel model merging framework guided by model weight statistics, leveraging singular values extracted via Singular Value Decomposition (SVD) to predict merging coefficients λ.

[1]Our code is available at https://github.com/statsmerging/statsmerging.

- We design the lightweight *StatsMergeLearner* to learn model merging coefficients λ estimation based on weight statistical features, through a newly proposed Task-Specific Teacher Distillation training paradigm without manually-annotated labels.
- We introduce the first heterogeneous architectural merging method, which distills knowledge from models with non-identical architectures into the unified target architecture.
- Extensive experiments demonstrate the effective of our proposed *StatsMerging*, achieving 84.5% average accuracy on merging models from eight tasks, outperform the state-of-the-art AdaMerging (81.1%) by a substantial margin of 3.4%.

2 Related Work

2.1 Multi-Task Learning

Multi-Task Learning (MTL) (Zhang and Yang, 2021; Vandenhende et al., 2022a) represents a paradigm for training a single model to perform multiple tasks concurrently. While MTL aims to create unified models capable of handling diverse objectives, it typically requires careful design of network architectures, computationally expensive training, access to large and diverse datasets, and intricate task balancing strategies (Zhang and Yang, 2021). Furthermore, MTL necessitates joint training from the outset, which can be computationally expensive and may not be feasible when dealing with a collection of pre-trained, specialized models. Model merging offers a compelling alternative by enabling the combination of independently trained models, without the need for extensive retraining or simultaneous access to multi-task datasets. Our work distinguishes from MTL by focusing on efficiently transferring existing knowledge within specialized models through weight statistics-guided merging rather than joint training.

2.2 Multi-Task Merging

Early approaches to model merging often involved simple heuristics like Weight Averaging (Wortsman et al., 2022), Ties-Merging (Yadav et al., 2023a), and Arithmetic Merging (Ilharco et al., 2023). While straightforward to implement, these methods (Ye et al., 2023; Akiba et al., 2025; Tang et al., 2025) typically lack awareness of the weight distributions and learned representations within the models, leading to suboptimal performance in the merged model compared to individually fine-tuned models or unified models trained from scratch. For instance, Wortsman et al. (Wortsman et al., 2022) demonstrated that naive weight averaging could significantly degrade performance, highlighting the challenges in consolidating knowledge from independently trained networks. More recent methods, such as those explored in natural language processing (Yadav et al., 2023b; Ilharco et al., 2023), have shown promise by learning interpolation weights. However, these often treat the weights as free parameters, potentially requiring significant tuning data and not explicitly leveraging the weight distribution of the models being merged, a key distinction from our proposed approach. The gap often lies in effectively unifying the diverse and task-specific feature representations learned by individual models into a single, high-performing entity without extensive retraining.

2.3 Merging Methods in Computer Vision

The application of model merging techniques in computer vision is relatively less explored compared to natural language processing (Yadav et al., 2023b; Ilharco et al., 2023). Computer vision models, particularly deep convolutional neural networks (CNNs) (Krizhevsky et al., 2012; He et al., 2016; Simonyan and Zisserman, 2014) and Vision Transformers (ViTs) (Dosovitskiy et al., 2021a; Touvron et al., 2021), learn complex, hierarchical feature representations that are highly sensitive to task-specific optimizations (Izmailov et al., 2018). Simple averaging techniques often fail due to the non-convex nature of the loss landscape and the divergence of learned feature spaces across different visual tasks. Recent advancements (Matena and Raffel, 2022; Yang et al.) have shown potential, but often lack explicit mechanisms to account for the unique properties inherent in visual data and architectures, such as spatial relationships in CNNs or attention mechanisms in ViTs. Furthermore, the effectiveness of these methods across the broad spectrum of computer vision tasks, including low-level restoration (Zhang et al., 2017; Saharia et al., 2022), mid-level detection (Ren et al., 2015; Carion et al., 2020b), and high-level classification (He et al., 2016), has not been comprehensively validated. Our work addresses these limitations by introducing a novel merging framework that

leverages internal model weight statistics to guide the merging process, making it more adaptable and effective across diverse computer vision tasks and architectures.

Method	No Manual Label	No TT Samples	Layer Level	TT Adaptability	Heterogeneous Architecture
Traditional MTL	✗	✗	*	✗	✗
Task Arithmetic	✓	✓	✗	✗	✗
Ties-Merging	✓	✓	✗	✓	✗
Fisher Merging	✓	✓	✗	✗	✗
RegMean	✓	✓	✗	✗	✗
AdaMerging	✓	✗	✓	✓	✗
StatsMerging (Ours)	✓	✓	✓	✓	✓

Table 1: Summary of system characteristics in recent works. *: Optional. TT: Test-Time.

In summary, our method *StatsMerging* enjoys several advantages compared to prior works: (1) no human annotated labels are required to construct the training set; (2) no validation samples are needed to compute the weight coefficients for merging; (3) it works in the Layer-Wise level; (4) it allows for test-time adaptability; (5) it can extend to heterogeneous architectures.

3 Methodology

3.1 Preliminaries

Notations: A deep neural network is parameterized by a set of weights $\theta = \{\theta_1, \theta_2, \ldots, \theta_L\}$ that learns the mapping from an input data $x_i \in \mathbb{R}^d$ to a predicted value $\hat{y_i} \in \mathbb{R}^D$: $f_\theta(x_i) \rightarrow \hat{y_i}$. Of these, θ^l represents the l-th $l \in \{1, 2, \ldots, L\}$ layer weights where L is the number of layers of the model f_θ, d denotes an input data x_i's dimension. For classification problems, y_i is the class label and D is the number of classes, while for regression problems, D is the dimension of the output vector y_i.

The weights of a pre-trained model (e.g., ViT or ResNet) are denoted as $\theta_{pre} = \{\theta^1_{pre}, \theta^2_{pre}, \ldots, \theta^L_{pre}\}$. The weights fine-tuned on a specific training data $\{x_i, y_i\}_{i=1}^{N_k^{tr}}$ for task k is recorded as $\theta_k = \{\theta^1_k, \theta^2_k, \ldots, \theta^L_k\}$ where N_k^{tr} is the number of training samples.

Problem Formulation: The problem of *model merging* is formulated as given K tasks' training data, find a way to combine weights $\{\theta_k\}_{k=1}^K$ fine-tuned for K tasks previously to obtain a new weight θ_m without undergoing the retraining process, while the new model f_{θ_m} is capable of performing well on K tasks jointly.

It is assumed that all K fine-tuned weights and the merged weight share the same neural network architecture. Therefore, the core question is how to *linearly combine* $\{\theta_k\}_{k=1}^K$ to obtain θ_m. In the task level, the model merging problem is finding a set of coefficients $\lambda_k \in \{\lambda_1, \lambda_2, \ldots, \lambda_K\}$ such that the merged model weights $\theta_m = \sum_{k=1}^K \lambda_k \theta_k$ for model f_{θ_m} perform well on all K tasks. In the layer level, it becomes searching for a set of coefficients $\lambda^l_k \in \{\lambda^1_1, \lambda^2_1, \ldots, \lambda^L_1, \lambda^1_2, \lambda^2_2, \ldots, \lambda^L_2, \ldots, \lambda^L_K\}$ to obtain the merged model $\theta_m = \sum_{k=1}^K \sum_{l=1}^L \lambda^l_k \theta^l_k$ that maintain high performance on K tasks.

3.2 Weight Statistics-Guided Model Merging

In this section, we describe the main intuition and techniques of our proposed method: *StatsMerging*. Our core idea is that given the distribution of pre-trained weights θ_k, we can learn a function $g(\theta_k) \rightarrow \lambda_m$ to predict the merging coefficients λ_m. We argue that *weight distribution* plays an important role in model merging. However, directly using the raw weights θ_k as input is impractical due to the high dimension of θ_k. We posit that such information can be represented by weight statistics. These statistical features contain key information regarding the amount of weights θ_k for a task k to be merged to the final model. We highlight the key differences with prior works in Fig. 1 d).

Weight Statistics: For a pre-trained weight θ_k on task k, we compute the mean μ_{θ_k} and variance $\sigma^2 = Var(\theta_k)$ to represent its center and breadth, as well as its magnitude $m = ||\theta_k||$. In addition,

we extract the singular values σ_i' from Singular Value Decomposition (SVD):

$$W_k = U_k \Sigma_k V_k^\top \tag{1}$$

where W_{θ_k} represents the matrix of the model parameter θ_k. By default, we use rank 3 from Σ_k to form weight statistics. We hypothesize that singular values compress the key information regarding weight distribution that can benefit the decision of assigning the amount of weights from θ_k for merging. Combining all together, the weight statistics feature vector S_k is formed as

$$S_k = stats(\theta_k) = [\mu, \sigma^2, m, \sigma_r'] \tag{2}$$

where $stats()$ extracts the statistical features from the weight θ_k, σ_r represents the singular value vector given rank r: $\sigma_r' = [\sigma_1', \sigma_2', \ldots, \sigma_r']$.

Notably, the Equation 3 above is task-wise while we also introduce layer-wise formulation for layer l:

$$S_k^l = stats(\theta_k^l) = [\mu, \sigma^2, m, \sigma_r']^l \tag{3}$$

where the layer-wise statistics features of pre-trained model from task k layer l is computed.

StatsMergeLearner (SML): We adopt a multilayer perceptron (MLPs) to learn to predict the merging coefficients λ given weight statistics feature vector S_k as input. In the task-wise mode, the *StatsMergeLearner* is denoted as $SML(S_k)$:

$$\lambda_k = SML(S_k) = g(stats(\theta_k)) \tag{4}$$

where λ_k is a scalar representing the merging coefficient of Task k model. In the layer-wise mode, the *StatsMergeLearner* is denoted as $M(S_k)$:

$$\lambda_k^l = SML(S_k^l) = g(stats(\theta_k^l)) \tag{5}$$

where λ_k is a vector containing L layers' coefficients and λ_k^l refers to the coefficient of layer l in the k pre-trained model. By default, we use a two-layer MLP to implement the *StatsMergeLearner*.

Optimization Objective. To train *StatsMergeLearner*, in the standard supervised training paradigm, we freeze the weights for each task θ_k and apply the cross-entropy loss function L_{CE} on the aggregated dataset:

$$\mathcal{L}_{\text{CE}}^{SL} = -\sum_{c=1}^{C_m} y_c \log(\hat{y}_c)) \tag{6}$$

where $\hat{y}_c$ is the prediction from the merged model for class c, C_m is the total number of classes in the aggregated dataset.

3.3 Task-Specific Teacher Distillation

We present a novel Task-Specific Teacher Distillation training paradigm to train the *StatsMerge-Learner* (SML) for model merging as illustrated in Fig. 2 and detailed in Algorithm 1. Our key intuition is that each pre-trained model θ_k is already good at its own task dataset $\{x_i, y_i\}_k \in D_k$, therefore we regard it (θ_k) as the Task-Specific Teacher T_k. Subsequently, the predictions $\hat{y}_{i,k}$ from the model trained on task k is decent enough as pseudo labels when it comes to its pre-trained dataset sample $\{x_i, y_i\}_k$. We aggregate such pairs $\{x_i, \hat{y}_{i,k}\}_k$ to construct the merged dataset to train $SML()$. We highlight the key benefit of this approach that enables dataset preparation without relying on human-annotated labels. The predicted class label in one-hot encoded format. Therefore, the cross-entropy loss is applied:

$$\mathcal{L}_{\text{CE}} = -\sum_{c=1}^{C_m} \hat{y}_{c,k} \log(\hat{y}_c)) \tag{7}$$

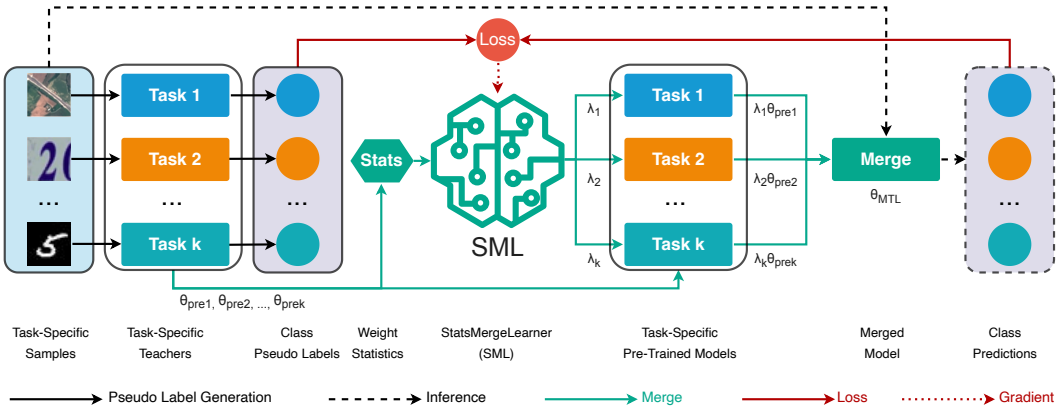

Figure 2: Knowledge Distillation Diagram. *StatsMergeLearner* (SML) learns the merging coefficients λ by minimizing the loss between the merged model's predictions and pseudo labels generated by task-specific teacher models. During inference, only the merged model in *StatsMerging* is used to predict class labels.

We present a novel Task-Specific Teacher Distillation training paradigm to train the *StatsMergeLearner* (SML) for model merging as illustrated in Fig. 2 and detailed in Algorithm 1. Our key intuition is that each pre-trained model θ_k is already good at its own task dataset $\{x_i, y_i\}_k \in D_k$, therefore we regard it (θ_k) as the Task-Specific Teacher T_k. Subsequently, the predictions $\hat{y}_{i,k}$ from the model trained on task k is decent enough as pseudo labels when it comes to its pre-trained dataset sample $\{x_i, y_i\}_k$. We aggregate such pairs $\{x_i, \hat{y}_{i,k}\}_k$ to construct the merged dataset to train $SML()$. We highlight the key benefit of this approach that enables dataset preparation without relying on human-annotated labels. The predicted class label in one-hot encoded format. Therefore, the cross-entropy loss is applied while such loss function simplicity helps extend to other vision tasks and architectures.

Algorithm 1. Unified Statistics-Guided Model Merging via Task-Specific Teacher Model Distillation

Input: Set of pre-trained models $\{M_1, M_2, \ldots, M_k\}$ with weights $\{\theta_1, \theta_2, \ldots, \theta_k\}$ for K tasks.
Output: Merged model M_{merged} with weights θ_{merged}
// Prepare K pre-trained models
if Same architecture A for all M_i **then**
    Set M_{target} to the shared architecture
else
    Select a target architecture M_{target}
    for $i = 1$ to k **do**
        if $A(M_i) \neq A(M_{\text{target}})$ **then**
            Distill M_i into M_{target} to obtain updated θ_i
        end if
    end for
end if
// Merge K models
for $k = 1$ to K **do**
    // mean μ, std σ^2, norm m, singular value σ'_r
    Extract statistics $S_k = [\mu, \sigma^2, m, \sigma'_r]$ from θ_k
    Predict coefficients $\lambda_k = \text{SML}(S_k)$
    Merge layer weights: $\theta^l_{\text{merged}} = \sum_{i=1}^{k} \lambda_k \theta_k$
end for
return M_{merged} with weights θ_{merged}

4 Experiments

4.1 Experimental Setup

In this section, we present the experimental setup and evaluation results used to compare our method against recent baselines.

Datasets and Models : Our experiments include eight image classification tasks with datasets SUN397 (Xiao et al., 2016), Stanford Cars (Krause et al., 2013), RESISC45 (Cheng et al., 2017), EuroSAT (Helber et al., 2019), SVHN (Netzer et al., 2011), GTSRB (Stallkamp et al., 2011), MNIST (LeCun et al., 1998), DTD (Cimpoi et al., 2014), and CIFAR10 (Krizhevsky and Hinton, 2009) [2] We use ViT-B/32 CLIP (Radford et al., 2021) as the pre-trained backbone. Individual task-specific models are obtained by training on each dataset separately. For merging models with different architectures, we first distill them into a single backbone before applying our merging method.

[2]In the remainder of the paper, the abbreviations shown in brackets are used to denote each task dataset: SUN397 (SU), Cars (CA), RESISC45 (RE), EuroSAT (EU), SVHN (SV), GTSRB (GT), MNIST (MN) and DTD (DT).

Baselines and Metrics : We compare against standard baselines including Individual Training, Traditional Multi-Task Learning (MTL) (Zhang and Yang, 2021), Weight Averaging (Wortsman et al., 2022), Task Arithmetic (Ilharco et al., 2023), Fisher Merging (Matena and Raffel, 2022), RegMean (Jin et al., 2023), Ties-Merging (Yadav et al., 2023a) and AdaMerging (Yang et al.). The primary evaluation metric is the average accuracy (Avg Acc) on the test sets of all tasks. The evaluation is conducted on eight different vision classification tasks.

***StatsMergeLearner* Training Detail** : Our MLP-based *StatsMergeLearner* learns to predict layer-wise or task-wise merging weights coefficients (λ) based on weight statistics from individual task models. The *StatsMergeLearner* is trained for 500 epochs using Adam, with a learning rate of $1e-3$ and a StepLR scheduler (factor 0.1 every 100 epochs), which translates to around only 3 hours to merge 4 ViTs, offering the practicality and advantage of applying our technique for practitioners without spending days or weeks for training (Zhang and Yang, 2021; Padmanabhan et al., 2023). We train the *StatsMergeLearner* primarily using knowledge distillation from the aggregated dataset without human annotated labels described in Sec. 3.3, optimized with either Cross-Entropy (Mao et al., 2023) or KL Divergence (Kullback and Leibler, 1951) loss.

4.2 Merging Performance

In this section, we present a comprehensive evaluation of our approach in comparison to state-of-the-art task vector merging methods, assessing its superiority across several fundamental aspects: Multi-task merging performance, generalization to unseen tasks and heterogeneous architectures.

Substantially Higher Merging Performance. The main results of merging performance of ViT-B/32 models on eight tasks are presented in this section, detailed [3] in Table 2. We present two levels of granularity: Task-Wise (TW) and Layer-Wise (LW). Our method *StatsMerging* achieved an average accuracy (Avg Acc) of 76.5% and 84.5% in both TW and LW levels, outperforming the state-of-the-art (SOTA) method AdaMerging++ by a large margin of 3.3% and 3.4%. We attribute the improvements to the ability of *StatsMergeLearner* to adapt task-specific weights based on their weight statistics to the merged model. The use of pseudo labels from task-specific teachers $\{T_1, T_2, \ldots, T_k\}$ provides stronger signals for *StatsMergeLearner* to better assign weight coefficients λ than the entropy minimization approach in the AdaMerging++.

Table 2: Multi-task merging performance (Avg Acc $\%$) when merging ViT-B/32 models on eight tasks. Results of our method *StatsMerging* are shaded in gray. Bold and underscore indicate the highest and second-highest scores within the merging group below the double rules in each column, respectively. TW: Task-wise. LW: Layer-wise.

Method	SU	CA	RE	EU	SV	GT	MN	DT	**Avg Acc**
Pre-Trained	62.3	59.7	60.7	45.5	31.4	32.6	48.5	43.8	48.0
Individual	75.3	77.7	96.1	99.7	97.5	98.7	99.7	79.4	90.5
Traditional MTL	73.9	74.4	93.9	98.2	95.8	98.9	99.5	77.9	88.9
Weight Averaging	65.3	63.4	71.4	71.7	64.2	52.8	87.5	50.1	65.8
Task Arithmetic	55.2	54.9	66.7	78.9	80.2	69.7	97.3	50.4	69.1
Fisher Merging	**68.6**	69.2	70.7	66.4	72.9	51.1	87.9	59.9	68.3
RegMean	65.3	63.5	75.6	78.6	78.1	67.4	93.7	52.0	71.8
Ties-Merging	59.8	58.6	70.7	79.7	86.2	72.1	**98.3**	54.2	72.4
TW AdaMerging	58.0	53.2	68.8	85.7	81.1	84.4	92.4	44.8	71.1
TW AdaMerging++	60.8	56.9	73.1	83.4	87.3	82.4	95.7	50.1	73.7
TW *StatsMerging*	61.3	<u>70.0</u>	74.2	85.2	87.5	82.5	96.2	54.2	76.4 (+3.3)
LW AdaMerging	64.5	68.1	79.2	<u>93.8</u>	87.0	<u>91.9</u>	<u>97.5</u>	59.1	80.1
LW AdaMerging++	66.6	68.3	<u>82.2</u>	**94.2**	<u>89.6</u>	89.0	**98.3**	<u>60.6</u>	<u>81.1</u>
LW *StatsMerging*	<u>67.4</u>	**74.1**	**82.9**	91.1	**89.8**	**94.7**	**98.3**	**77.5**	**84.5 (+3.4)**

The LW *StatsMerging* achieved significantly higher $(+8.1\%)$ Avg Acc than TW *StatsMerging* with 84.5% and 76.4%, respectively. This improvement in layer-wise merging over task-wise aligns with observations in AdaMerging. We hypothesize that compared to the coarser task-level granularity, the

[3] Please refer to the Appendix for experimental details, including the full list of tasks, datasets, and baselines.

finer-grained layer-level offers greater flexibility of coefficients across various semantics regarding task-agnostic and task-specific features, which are often learned in various levels of a neural network.

Significantly Enhanced Generalization. A merged model is expected to generalize to unseen tasks by strategically transferring the knowledge from the combined set of old tasks. We benchmark such generalization ability of *StatsMerging* against four strong baselines: Task Arithmetic, Ties-Merging, AdaMerging, and AdaMerging++. We follow the same evaluation protocol in AdaMerging training on two groups of tasks, each group consisting of six seen tasks, and testing on two unseen tasks.

Table 3: Generalization results (Avg Acc %) on two unseen tasks when merging Layer-Wise ViT-B/32 models on six tasks. *StatsMerging*: shaded in gray. Bold: top score. Underscore: 2nd-highest score.

	Seen Tasks							**Unseen Tasks**		
Method	SU	CA	RE	DT	SV	GT	**Avg Acc**	MN	EU	**Avg Acc**
Task Arithmetic	63.3	62.4	75.1	57.8	84.6	80.4	70.6	77.2	46.2	61.7
Ties-Merging	67.8	66.2	77.2	56.7	77.1	70.9	69.3	75.9	43.3	59.6
AdaMerging	65.2	65.9	**88.5**	61.1	92.2	91.5	77.4	84.0	56.1	70.0
AdaMerging++	68.2	67.6	86.3	63.6	92.6	89.8	78.0	83.9	53.5	68.7
StatsMerging	**69.1**	**71.3**	86.7	**75.2**	**93.2**	**95.7**	**81.9 (+3.9)**	**85.1**	**56.4**	**70.8 (+0.8)**
Method	SU	CA	GT	EU	DT	MN	**Avg Acc**	RE	SV	**Avg Acc**
Task Arithmetic	64.0	64.0	75.2	87.7	57.0	95.7	73.9	52.3	44.9	51.1
Ties-Merging	68.0	67.1	67.7	78.4	56.5	92.8	71.8	**58.7**	49.2	53.9
AdaMerging	67.1	67.8	94.8	94.4	59.6	98.2	80.3	50.2	60.9	55.5
AdaMerging++	68.9	69.6	91.6	94.3	61.9	98.7	80.8	52.0	64.9	58.5
StatsMerging	**69.6**	**73.3**	**96.1**	**95.4**	**74.1**	**97.2**	**84.3 (+3.5)**	54.2	**67.1**	**60.7 (+2.2)**

Details are presented in Table 3, where in both groups our proposed *StatsMerging* achieved $70.8\%s$ and 60.7%, significantly outperforming the second best method AdaMerging by $+0.8\%$ and $+2.2\%$ margins. Such improvements are attributed to both (1) the careful feature design of weight statistics that captures the dominant information regarding weight distributions from pre-trained models, which potentially helps reduce noise from each task dataset; and (2) the joint training from all old tasks on the task-specific teacher-distilled labels, enabling the implicit learning of task-agnostic and task-specific features that can benefit the generalization ability.

Extension to Heterogeneous Architectures. Our *StatsMerging* offers the first and unique advantage without the assumption of architectural identity in prior works (Wortsman et al., 2022; Ilharco et al., 2023; Yadav et al., 2023a; Matena and Raffel, 2022; Jin et al., 2023). To verify the performance of varying architectures, we conduct experiments on ResNet50 (RN) and ViT-B/32 (VT) to represent Convolutional Neural Network (CNN) and Vision Transformer (ViT) architectures.

In particular, we distill fine-tuned VT teachers into a RN (Khanuja et al., 2021) student on three diverse tasks of CIFAR-10 (CI), EuroSAT (EU) and Stanford Cars (CA) with the distillation loss:

$$\mathcal{L} \;=\; \alpha\,\mathcal{L}_{\mathrm{CE}}(y,\hat{y}) \;+\; (1-\alpha)\,T^2\,\mathcal{L}_{\mathrm{KL}}\big(\sigma(\tfrac{z}{T}),\;\sigma(\tfrac{z_t}{T})\big), \tag{8}$$

where $\mathcal{L}_{\mathrm{KL}}$ denotes KL-Divergence, z is logit, $T = 4.0$ represents temperature, $\alpha = 0.7$ is the weight balance of two sub-losses. CI is used due to the available pre-trained RN weights. Remarkably, the distilled RN matches its VT teacher's accuracy, achieving 76.4% (VT: 77.7%) for CA and 94.5% for EU (VT: 99.7%) despite the architectural difference shown in Table 3. We then apply our *StatsMerging* to combine the CI–trained RN and its distilled variants. We merge multiple task models into a single RN using the merging coefficients inferred by *StatsMergeLearner*, yielding a 7.6% average improvement over the vanilla Task-Arithmetic of 73.7% and achieving 81.3% average accuracy.

Table 3. Multi-task merging performance (Avg Acc %) of models in heterogeneous architectures: ResNet50 (RN) & ViT-B/32 (VT). *StatsMerging*: shaded in gray.

Method	CI	CA	EU	**Avg Acc**
Backbone	RN	VI	VI	-
Distilled	-	RN	RN	-
Individual	97.8	77.7	99.7	91.7
Distilled	-	76.4	94.5	-
Weight Averaging	77.1	56.4	64.9	59.4
Ties-Merging	76.5	52.8	80.1	69.8
Task Arithmetic	81.4	61.6	78.2	73.7
LW *StatsMerging*	87.2	68.4	88.4	81.3

4.3 *StatsMerging* Analysis

Label & Loss Function Study. We conduct a loss function study on ViT-B/32 (4) models merged from four tasks, as shown in Table 4. Observe that *StatsMerging* trained on pseudo labels via Task-Specific Teacher Distillation (KD) achieves similar performance to *StatsMerging* trained on ground-truth labels (GT), with 88.5% and 81.2% average accuracy in TW and 90.4% and 83.5% in LW levels.

Table 4. Multi-task performance (Avg Acc %) of *StatsMerging* when merging ViT-B/32 (4) models across four tasks. *StatsMerging* shaded in gray. GT: Ground Truth. KD: Knowledge Distillation. TW: Task-wise. LW: Layer-wise.

Loss	Level	CA	EU	RE	GT	**Avg Acc**
GT	TW	73.2	94.2	91.1	95.6	88.5
KD	TW	64.2	88.6	85.2	86.7	81.2
GT	LW	75.6	96.3	92.1	97.6	90.4
KD	LW	68.7	91.6	87.2	93.5	83.5

Statistical Feature Ablation Study. We conduct an ablation study on the statistical features. Results in Table 5 show that combining all statistical features improves merging performance, validating our design choice. Notably, the singular values σ' improve the multi-task performance in both same and different architecture settings by $+3.0$ and $+3.2$ increase of average accuracy, justifying our design choice of using SVD.

Table 5: Multi-task performance (Avg Acc %) of *StatsMerging* when ablating statistical features of ViT-B/32 (4) models on four tasks: CA, EU, RE & GT. Bold: top score. *StatsMerging*: shaded in gray.

Same Architecture					**Different Architecture**				
μ_{θ_k}	σ^2	m	σ'	**Avg Acc**	μ_{θ_k}	σ^2	m	σ'	**Avg Acc**
✓				83.4	✓				76.2
✓	✓			84.1 (+0.7)	✓	✓			77.5 (+1.3)
✓	✓	✓		87.2 (+3.1)	✓	✓	✓		78.1 (+0.6)
✓	✓	✓	✓	**90.2 (+3.0)**	✓	✓	✓	✓	**81.3 (+3.2)**

Coefficient Analysis. We visualize the heatmap of ViT-B/32 (4) across eight tasks in Fig. 3. We make several key observations: (1) the **common recurring pattern** of coefficients λ across all eight tasks from earlier (left) to deeper (right) layers aligns with the repeated self-attention blocks in the ViT architecture, e.g. Multi-Head Self-Attention (MHSA), MLP (Feed-Forward Network), and LayerNorm, etc, demonstrating the need of various coefficients for various types of layers; (2) The **sparse non-uniform coefficient distributions** (various colors like Layer 13, 19 or 25) suggests that merging layers can be more efficient at some specific layers instead of using one coefficient for an entire pre-trained model, justifying the our granularity choice of Layer-Wise over Task-Wise level; (3) some **task-specific coefficient distributions** verify the necessity of assigning distinct merging coefficients across tasks in various layers, such as in Layer 5 vs. 147. Such distributions reflect the various visual representations for different semantics learned across both layers and tasks. More visualizations are provided in the Appendix for in-depth analysis.

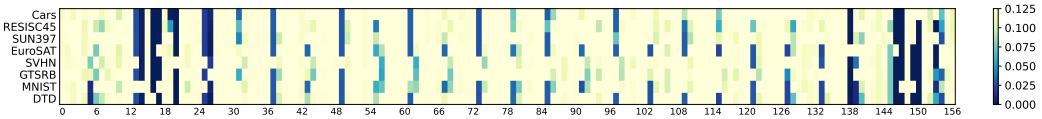

Figure 3: Heatmap of *StatsMerging* merging coefficients λ of ViT-B/32 (4) across eight tasks. X-axis: layer index. Y-axis: Tasks. Coefficients are normalized to sum to 1.

5 Conclusion

Model merging offers a compelling post-hoc advantage to reduce memory storage from a corpus of large pre-trained models. We propose *StatsMerging*, a novel merging technique without relying on simple heuristics, test-time samples or human annotated. The key intuition lies in the guidance of weight statistics using a lightweight MLP learner, dubbed *StatsMergeLearner*, to learn merging coefficient prediction. Exhaustive experiments demonstrate the effectiveness of our proposed *StatsMerging* in model mering from eight diverse tasks, achieving 84.5% average accuracy and surpassing the SOTA AdaMerging (81.1%) by a large margin of 3.4%.

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: Our abstract does reflect our contributions to large extent, we did thorough analysis and added more appropriate results in appendix as well which supports our claims.

   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 did mention limitations in the appendix.

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: Yes, we did provide experiments that support our proof and assumptions, more experiments are added in the appendix.

   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: Yes, we provided our code and model in github, that gives proof to reproduce results.

   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 provided access to code with proper readme.

   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 provided details in 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: Our experiements are easy to follow, reproducable.

   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: [NA]

   Justification: Our experiments can be done on any GPU that has memory up to 40 GB. We used NVIDIA RTX A6000.

   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: Yes, It does follow Neurips Code of Ethics.

   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: Our method solely addresses the new emerging solution for merging multi-models post training, so it doesn't comes under societal impacts.

    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 the responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This doesn't comes under the category of safeguards or anything related to scraped datasets.

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: [NA]

Justification: We are the sole owner of code and models, we mentioned credit wherever we used opensource models, or data.

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: We did have detailed documentation of the models and code we tried.

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: We didn't require any crowdsourcing, we used opensource data and required very less efforts for training.

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: Our work is independent work, not related to any organization.

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.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We used it for language and clarifying the formulation.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

References

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, pages 1–10, 2025.

Bryan Bo Cao, Abhinav Sharma, Lawrence O'Gorman, Michael Coss, and Shubham Jain. A lightweight measure of classification difficulty from application dataset characteristics. In *International Conference on Pattern Recognition*, pages 439–455. Springer, 2024a.

Bryan Bo Cao, Abhinav Sharma, Manavjeet Singh, Anshul Gandhi, Samir Das, and Shubham Jain. Representation similarity: A better guidance of dnn layer sharing for edge computing without training. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 2242–2244, 2024b.

Bryan Bo Cao, Lawrence O'Gorman, Michael Coss, and Shubham Jain. Few-class arena: A benchmark for efficient selection of vision models and dataset difficulty measurement. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL `https://openreview.net/forum?id=2ET561DyPe`.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision (ECCV)*, pages 213–229. Springer, 2020a.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020b.

Gong Cheng, Junwei Han, and Lei Guo. Remote sensing image scene classification: Benchmark and state of the art. *ISPRS Journal of Photogrammetry and Remote Sensing*, 124:157–173, 2017.

Mihai Cimpoi, Subhransu Maji, Iasonas Kokkinos, Scott Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 861–868, 2014.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Georgiou, et al. An image is worth 16x16 words: Transformers for image recognition at scale. 2021a.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021b.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems 27 (NIPS 2014)*, pages 2672–2680, 2014.

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.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset for sentinel-2 image classification. In *2019 IEEE International Geoscience and Remote Sensing Symposium IGARSS 2019*, pages 204–207. IEEE, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 6840–6851, 2020.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023.

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. UAI 2018.

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*, 2023.

Simran Khanuja, Melvin Johnson, and Partha Talukdar. Mergedistill: Merging pre-trained language models using distillation. *arXiv preprint arXiv:2106.02834*, 2021.

Jonathan Krause, Benjamin Dunn, and Li Fei-Fei. Collecting and annotating a dataset for fine grained recognition problems. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 139–146. IEEE, 2013.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Canada, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, volume 25, pages 1097–1105, 2012.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986, 2022.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3431–3440, 2015.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications. In *International conference on Machine learning*, pages 23803–23828. PMLR, 2023.

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

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. 2014.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with convolutional neural networks. Technical report, Google, 2011.

Arthi Padmanabhan, Neil Agarwal, Anand Iyer, Ganesh Ananthanarayanan, Yuanchao Shu, Nikolaos Karianakis, Guoqing Harry Xu, and Ravi Netravali. Gemel: Model merging for {Memory-Efficient},{Real-Time} video analytics at the edge. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 973–994, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems 28 (NIPS 2015)*, pages 91–99, 2015.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Manavjeet Singh, Sri Pramodh Rachuri, Bryan Bo Cao, Abhinav Sharma, Venkata Bhumireddy, Francesco Bronzino, Samir R Das, Anshul Gandhi, and Shubham Jain. Ovida: Orchestrator for video analytics on disaggregated architecture. In *2024 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 135–148. IEEE, 2024.

Johannes Stallkamp, Reda Mammeri, Mark Eckenfelder, Jonathan Richter, Didier Van Cauwelaert, Ulrich Schaefer, and Christian Igel. The german traffic sign recognition benchmark. In *The 2011 international joint conference on neural networks*, pages 1453–1460. IEEE, 2011.

Anke Tang, Enneng Yang, Li Shen, Yong Luo, Han Hu, Bo Du, and Dacheng Tao. Merging models on the fly without retraining: A sequential approach to scalable continual model merging. *arXiv preprint arXiv:2501.09522*, 2025.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Hervé Jégou, and Alexandre Sablayrolles. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, pages 10347–10357. PMLR, 2021.

Simon Vandenhende, Stamatios Georgoulis, Leander Arras, Luc Van Gool, and Radu Timofte. Multi-task learning for computer vision: Recent advances and future directions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6488–6513, 2022a.

Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3614–3633, 2022b.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 2022.

Jianxiong Xiao, James Hays, Kevin A Ehinger, Aude Oliva, and Antonio Torralba. Sun attribute dataset: Learning the joint distribution of object and attribute. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4145–4153, 2016.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, pages 12077–12090, 2021.

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, 2023a.

Sachin Yadav, Chitta Malaviya, Graham Neubig, and Puneet Agarwal. Merging transformers without training via a convex combination of parameter subsets. pages 41105–41125, 2023b.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*.

Peng Ye, Chenyu Huang, Mingzhu Shen, Tao Chen, Yongqi Huang, Yuning Zhang, and Wanli Ouyang. Merging vision transformers from different tasks and domains. *arXiv preprint arXiv:2312.16240*, 2023.

Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. In *IEEE transactions on image processing*, volume 26, pages 3142–3155. IEEE, 2017.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021.