TIE THE KNOTS: MODEL MERGING WITH SVD

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent model merging methods demonstrate that the parameters of fully-finetuned models specializing in distinct tasks can be combined into one model capable of solving all tasks without retraining. Yet, this success does not transfer well when merging LoRA finetuned models. We study this phenomenon and observe that the weights of LoRA finetuned models showcase a lower degree of alignment compared to their fully-finetuned counterparts. We hypothesize that improving this alignment is key to obtaining better LoRA model merges, and propose KnOTS to address this problem. KnOTS uses the SVD to jointly transform the weights of different LoRA models into an aligned space, where existing merging methods can be applied. Together with resetting the singular values to better represent the space relevant to each task, KnOTS improves LoRA merging by up to 13% across several vision benchmarks.

1 INTRODUCTION

Model merging (Matena & Raffel, 2021; Wortsman et al., 2022a; Choshen et al., 2022) introduced a surprising ability to create a single general model by combining weights of multiple task-specific models. This allows creating multi-task models by accumulating skills (Stoica et al., 2024; Ilharco et al., 2022; Yadav et al., 2023; Ortiz-Jimenez et al., 2024), a desirable trait in various scenarios such as recycling models shared on hubs (Choshen et al., 2023), patching model weaknesses (Cai et al., 2023; Zaman et al., 2023) and collaborating to improve models (Don-Yehiya et al., 2022; Raffel, 2023).

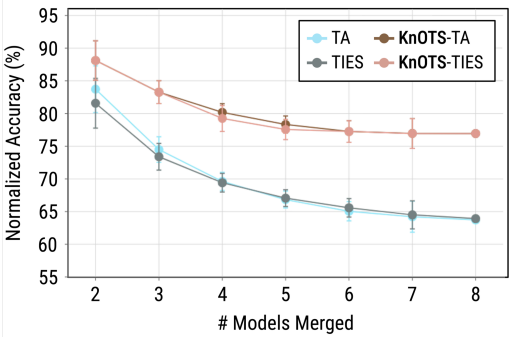


Figure 1: **KnOTS performs and scales significantly better** than prior work (Task Arithmetic (TA) (Ilharco et al., 2022) and TIES (Yadav et al., 2023)) when merging an increasing number of tasks. Performance is the average normalized accuracy with 95% confidence intervals over merging all possible combinations of the tasks each LoRA model was trained on, as defined in Ilharco et al. (2022); Yadav et al. (2023).

Model merging approaches have found substantial success when merging models that are full-rank finetuned (i.e., all parameters are tuned with maximum rank, we denote it as FFT) to solve distinct tasks from the same pretrained checkpoint, into one model capable of solving all (Ilharco et al., 2022; Matena & Raffel, 2021; Yadav et al., 2023; Ortiz-Jimenez et al., 2024; Wortsman et al., 2022b). In many cases, strong performance can be achieved by performing a linear sum over the fine-tuned model weights, without further training (Wortsman et al., 2022b; Ilharco et al., 2022; Yadav et al., 2023).

Recent analysis attributes this success to the updates on the parameters of a pretrained model made from finetuning on a new task (we term these "task-updates") being aligned to those achieved from finetuning on a different task, despite each being used to solve different problems (Gueta et al., 2023). This implies that their contributions to the intermediate outputs at each layer in the pretrained model are correlated and share the same representation space, for the same inputs (Entezari et al., 2021; Stoica et al., 2024). However, these same approaches do not always transfer well when applied to models finetuned with parameter efficient finetuning (PEFT) strategies (Tang et al., 2024), such as the extremely popular Low-Rank Adaptation (LoRA) (Hu et al., 2021). We probe into this phenomenon (Sec. 3) and find that while the task-updates between FFT models are *highly correlated*, those of

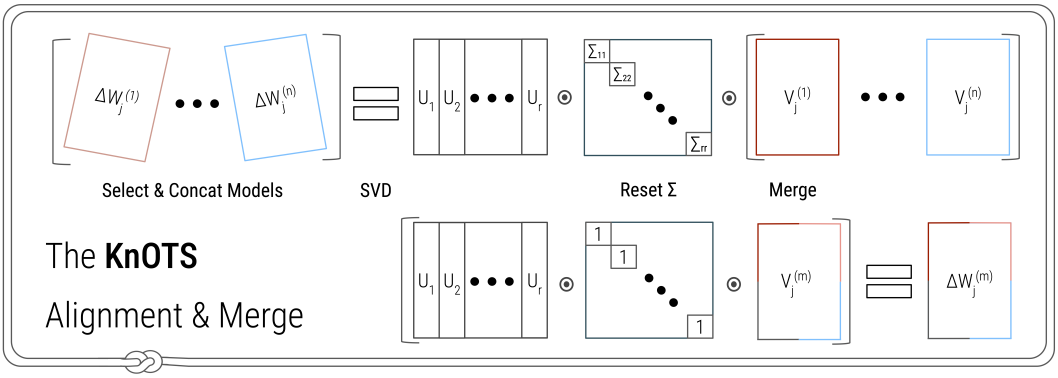


Figure 2: **The KnOTS method** for merging "task-updates" from an arbitrary layer-j of a model. Each task-update is denoted by $\Delta W_j^{(i)}$, where i is the i^{th} task-update. KnOTS first concatenates the updates together and applies the SVD, to obtain U, Σ and a set of concatenated $V^{(i)}$ matrices that each correspond to a particular task. KnOTS then "resets" Σ to the identity matrix, and merges the V's into a single $V^{(m)}$ matrix. Finally, KnOTS multiplies the U, reset Σ and $V^{(m)}$ to obtain a merged-update to be added to the pretrained model.

LoRA models are *not*. This suggests that the task-updates between LoRA models process inputs in different spaces, and are unaligned.

Our method, termed KnOTS (*Kn*owledge *O*rientation *T*hrough *S*VD), uses the singular value decomposition (SVD) to transform the task-updates of different LoRA models into a shared space, where merging methods can be applied (Fig. 2). We do this by first concatenating all the task-updates for a layer, and then decomposing it with the SVD to obtain: $U\Sigma V^T$. We observe that in this construction, U consists of vectors that form the orthonormal basis for a *shared* representation space between all task updates, the non-negative diagonal values in Σ represent the scale associated with each basis vector in the aligned space, and V^T is a set of concatenated matrices (one for each task-update) that are all aligned to the common U. KnOTS then applies existing merging methods, such as those of Ilharco et al. (2022); Yadav et al. (2023), to the V task-matrices and obtains a merged V matrix that is still aligned to U. After merging, KnOTS constructs a merged model by multiplying the resulting SVD components at every layer, and adding it to the corresponding parameter matrix from the pretrained model.

Interestingly, by analyzing the impact of Σ on a merged model (§4), we find that it may emphasize certain basis vectors that only benefit certain tasks. This may cause substantial performance degradation on other tasks when merging. Coupled with additional observations, namely, that other components in the SVD better distribute the basis vectors, KnOTS "resets" Σ to the identity matrix. This can then be appropriately scaled by existing merging methods to obtain more *balanced* merges. We validate our approach by conducting extensive experiments across a variety of settings, and find that KnOTS can be used to dramatically improve existing merging methods (§5). Our experiments range from merging eight image classification models to achieve a *performance gain of up to* 13% over prior work (§5.2), demonstrating how KnOTS is more *robust with scale* (Fig. 1), to showing how it yields *better general merged models* (§5.4).

2 Related Work

The geometric landscape of non-convex loss functions used to train neural networks remains largely uncharted. However, Draxler et al. (2018); Garipov et al. (2018); Simsek et al. (2021); Frankle & Carbin (2018) reveal that the parameter values of independently trained neural networks can be interpolated without increasing the test loss, a phenomenon known as mode-connectivity. Neyshabur et al. (2020), shows that models are often mode-connected when they are finetuned on the same task from the same pretrained initialization. In parallel, it has become common practice to finetune models using pretrained weights (Dosovitskiy et al., 2020; Huang et al., 2023; Oquab et al., 2023; Hsu et al., 2021; Touvron et al., 2023; Radford et al., 2021). Together with recent discoveries (Wortsman et al., 2022b; Matena & Raffel, 2021) that even the parameters of models finetuned on different tasks from

the same initialization may be merged to create strong multitask models, there has been a surge of model merging approaches for finetuned models.

Model merging methods. Several approaches improve robustness or scores by averaging finetuned weights (Choshen et al., 2022; Wortsman et al., 2022b; Rame et al., 2022; Ramé et al., 2024). Task arithmetic (TA) (Ilharco et al., 2022) first subtracts the parameter values of the pretrained model from those of the finetuned models, creating a set of "task-vectors." These are then linearly summed to create a merged model, without any gradient calculations or retraining. Ortiz-Jimenez et al. (2024) theoretically grounds TA, by showing that its success is dependent on the task-vectors governing disjoint regions in the pretrained model's function space—a concept known as "weight disentanglement." In practice, they find that the weights of different finetuned models heavily conflict, and propose a finetuning strategy that disentangles finetuned models to merge better with TA. Daheim et al. (2023); Shah et al. (2023) also find that TA-style merging can improve with finetuning and access to large amounts of training data. TIES (Yadav et al., 2023) improves TA by resolving the parameter interference between models when merging. Specifically, it prunes low-magnitude weights and then only averages weights which share the dominant sign. DARE (Yu et al., 2023) further explores this issue by randomly dropping fine-tuned weights and rescaling the remaining ones to create sparse task-vectors.

Merging LoRA models. LoRA (Hu et al., 2021) has quickly become a very popular finetuning technique. However, existing merging methods (Ilharco et al., 2022; Yadav et al., 2023; Yu et al., 2023) do not transfer well to merging LoRA models (Tang et al., 2024). Tang et al. (2024) suggest this is due to increased weight-entanglement between the models, and propose to a finetuning method similar to Ortiz-Jimenez et al. (2024) which improve the performance of each merging method. Following Ortiz-Jimenez et al. (2024); Tang et al. (2024), as KnOTS transforms task-updates to a shared orthonormal basis, it can be thought of as disentangling updates *without* finetuning. To the best of our knowledge, we present the first framework capable of achieving strong merged LoRA models *without finetuning*.

3 Background and Motivation

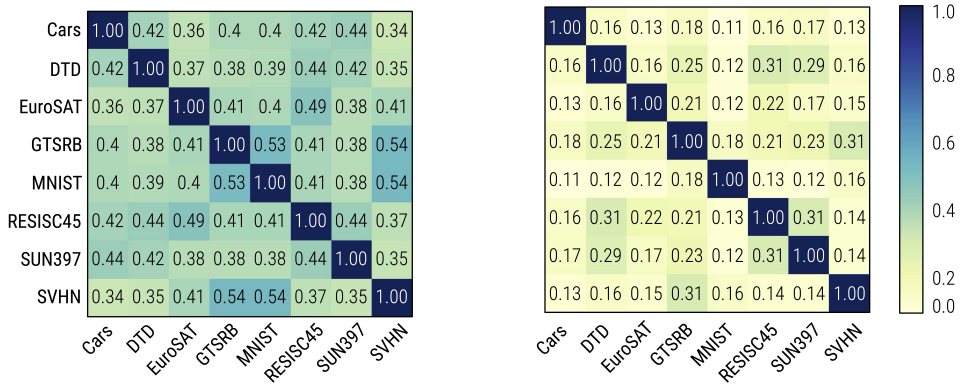
Problem setting. Suppose that we have n tasks (e.g., different image classification tasks) and a set of already finetuned models with the same architecture $\{f^{(1)}, f^{(2)}, \ldots, f^{(n)}\}$ for each task. We assume all models are finetuned from the same pretrained model $f^{(pt)}$. We seek to create a single multitask model, $f^{(m)}$, capable of solving all n tasks. However, we *do not* assume access to sufficient data or the compute needed to jointly train a single multitask model for all tasks. Fortunately, alternative methods exist that are both *data-efficient* and *training-free*. These instead fuse parameters of every finetuned model together into one, unified model, a process known as *model merging* (Wortsman et al., 2022a; Choshen et al., 2022; Stoica et al., 2024; Yadav et al., 2023; Ilharco et al., 2022). We are interested in merging LoRA finetuned models (Hu et al., 2021).

Model definitions. Let $f^{(pt)}$ have l layers, each containing weights and an optional bias. Assume all biases are concatenated in respective weights, and let weight of the j^{th} layer be represented as $W_j^{(pt)}$. Let $\theta^{(pt)}$ be the collection of all model weights, such that $\theta^{(pt)} = \{W_1^{(pt)}, \ldots, W_j^{(pt)}, \ldots, W_l^{(pt)}\}$. Finetuned models are obtained by applying updates to each weight in $\theta^{(pt)}$. We denote these updates by $\tau^{(i)} = \{\Delta W_1^{(i)}, \ldots, \Delta W_j^{(i)}, \ldots, \Delta W_l^{(i)}\}$, where $\Delta W_j^{(i)}$ is the update corresponding to the j^{th} layer in the i^{th} model. Thus, the parameters of $f^{(i)}$ are given by $\theta^{(pt)} + \tau^{(i)} = \{W_1^{(pt)} + \Delta W_1^{(i)}, \ldots, W_j^{(pt)} + \Delta W_j^{(i)}, \ldots, W_l^{(pt)} + \Delta W_l^{(i)}\}$. We study pretrained models finetuned with Low-Rank Adaptation (LoRA) (Hu et al., 2021) which enforces low-rank task-updates $\Delta W_j^{(i)}$.

Here, every $\Delta W_j^{(i)} \in \mathbb{R}^{O \times I}$ processes the output of the previous layer $z_{j-1}^{(i)} \in \mathbb{R}^I$ to receive the output $z_j^{(i)} \in \mathbb{R}^O$ as follows: $z_j^{(i)} = \Delta W_j^{(i)} z_{j-1}^{(i)}$.

Merging models. Existing methods (Ilharco et al., 2022; Yadav et al., 2023) within our problem setting merge models as follows. First, all $\tau^{(1)}, \ldots, \tau^{(n)}$ are flattened into vector representations—

termed task-vectors (Ilharco et al., 2022). Second, these task-vectors are stacked as rows, and each column (weight across models) is passed to a function (e.g., sum in (Ilharco et al., 2022)). Third, each transformed task-vector is uniformly scaled via a unique "scaling coefficient", denoted by $\lambda^{(i)} \geq 0$. Fourth, the scaled and transformed task-vectors are summed to create a single merged vector. This is then un-flattened into a collection of merged updates denoted by $\tau^{(m)} = \{\Delta W_1^{(m)}, \ldots, \Delta W_j^{(m)}, \ldots, \Delta W_l^{(m)}\}$, where $\Delta W_j^{(m)}$ is the result from merging the corresponding $\Delta W_j^{(i)}$ from each $\tau^{(i)}$. Afterwards, $\tau^{(m)}$ is added to $\theta^{(pt)}$ to generate the parameters for the merged model $f^{(m)}$. Optimal values for $\lambda^{(i)}$ are selected according to the performance of $f^{(m)}$ on any available data (Ilharco et al., 2022; Yadav et al., 2023). Although, LoRA updates factorize into smaller matrices, we utilize the full-matrix representation $\Delta W_j^{(i)}$ in our merges. We explain this decision in App. A.



(a) Full-rank finetuning alignment (b) LoRA finetuning alignment

Figure 3: **Finetuning strategy impacts weight alignments between tasks**. The figure shows the average pairwise absolute value (e.g., magnitudes) correlations (using (Li et al., 2015)) between the outputs *solely given by finetuning updates* (e.g., a $\Delta W_j^{(i)}$) across every attention layer, from models finetuned on different tasks (defined in §5.2). High correlation corresponds to high weight alignment (Entezari et al., 2021; Ainsworth et al., 2022; Stoica et al., 2024). (a) Full-rank finetuned models exhibit high correlation magnitudes in their finetuned-updates despite specializing in different tasks. (b) LoRA finetuned models show highly uncorrelated updates, indicating low weight alignment.

LoRA feature misalignment. Existing full-rank finetuned (FFT) model merging approaches perform well without additional data, even on different tasks (Ilharco et al., 2022; Wortsman et al., 2022a; Yadav et al., 2023). (Tang et al., 2024; Gueta et al., 2023) attribute these successes to matched knowledge (Zaman et al., 2023) and no loss barriers (Garipov et al., 2018; Entezari et al., 2021) between them, i.e. the loss is non-decreasing between a merged model and the original models. In our context, it means that their weights perform a similar task and are *aligned*. Rather than, for example, permuted weights between models (Entezari et al., 2021; Ainsworth et al., 2022; Stoica et al., 2024; Jordan et al., 2022). Fig. 3a illustrates this phenomenon: the average outputs between two FFT models are highly correlated across all layers. App. B further explains this figure.

However, Ilharco et al. (2022); Yadav et al. (2023) perform substantially worse when merging LoRA models (Tang et al., 2024). This may seem surprising as LoRA models are also finetuned from the same pretrained model. We probe into this observation by reproducing our correlation analysis on LoRA models rather than FFT ones, finding LoRA models to be far less correlated (Fig. 3b). Similar results are found (but not discussed) in (Zhu et al., 2024), and suggest that the updates between LoRA models finetuned on different tasks are *not aligned*.

Aligning updates. Entezari et al. (2021); Stoica et al. (2024) link aligned weights between models to their outputs (from the same input) lying in the same *representation space*. Unlike traditionally finetuned models, our results suggest that updates from LoRA models *do not* lie in the same spaces. We posit this is likely due to their low-rank constraint: each update can only impact a small portion (i.e., subspace) of the full representation space. Thus, we may align their updates by aligning them to the same representation space.

Our objective. We seek to align LoRA models by transforming their updates to lie in the same representation space, enabling the application of existing merging approaches. We aim to do this without requiring additional finetuning, or data.

4 METHOD: KNOTS

We propose a simple data and gradient-free method for aligning LoRA models that dramatically improves the performance of existing model merging methods. Our method, termed KnOTS (*Kn*owledge *O*rientation *T*hrough *S*VD) , uses the singular value decomposition (SVD) to align the representation spaces between LoRA models (see illustration in Fig. 2).

Representation space alignment. KnOTS aligns the updates across different LoRA models layer-wise. Thus, it suffices to consider how we align the updates for an arbitrary layer j. Recall that the parameter updates are denoted by: $\Delta W_j^{(1)}, \ldots, \Delta W_j^{(i)}, \ldots, \Delta W_j^{(n)}$, each outputting to a (different) subspace in the full representation space. By simply concatenating each update and obtaining an SVD decomposition, we align each to the *same* representation space:

$$\left[\Delta W_j^{(1)}; \Delta W_j^{(2)}; \ldots; \Delta W_j^{(n)}\right] = U\Sigma V^T \tag{1}$$

$$= U\Sigma \left[V^{(1)}; V^{(2)}; \ldots; V^{(n)}\right]^T \tag{2}$$

where, $\Delta W_j^{(1)} = U\Sigma[V^{(1)}]^T; \Delta W_j^{(2)} = U\Sigma[V^{(2)}]^T; \ldots; \Delta W_j^{(n)} = U\Sigma[V^{(n)}]^T$. The SVD jointly decomposes the parameter updates into 3 separate matrices (Fig. 2. First, $U \in \mathbb{R}^{O\times r}$, where $r \leq O$ is the total rank of the concatenated updates, is a matrix consisting of orthonormal column vectors. Importantly, these vectors span the entire output space achievable from any parameter update, regardless of which task the input belongs to. The other two SVD components, $\Sigma \in \mathbb{R}^{r\times r}$ and $V \in \mathbb{R}^{r\times nI}$, are trivially aligned to this shared representation space. Second, Σ is a diagonal matrix whose non-negative values emphasize which basis vectors in U contribute the most to the aligned space across *all* parameter updates. Third, V is a transformation matrix that maps from a concatenated input space to the aligned output space represented by U. By splitting V into n matrices: $\{V^{(1)}, \ldots, V^{(i)}, \ldots, V^{(n)}\}$ each $\in \mathbb{R}^{I\times r}$, we obtain a matrix particular to each task that is intrinsically aligned to the *same* subspace spanned by U—thereby making all matrices aligned. We colloquially refer to these matrices as "sub-parameters."

Applying merge methods. Because all sub-parameters are now aligned, we can now apply existing merging methods on them without any modifications. Specifically, each $V^{(i)}$ can be fused to create a merged $V^{(m)}$ according to the merging procedure. We can then compute the merged update by, $\Delta W_j^{(m)} = U\Sigma V^{(m)}$, which is added to the pre-trained layer weights W_j.

The impact of Σ. The diagonal elements of Σ weigh how important a basis vector in U is to the shared representation space across all models. Yet, our analysis shows that they are not the only features to weigh importances. Instead, importances are found dynamically per separate model, and further depend on $V^{(i)}$ and the outputs from previous layers $z_{j-1}^{(i)}$:

$$z_j^{(i)} = \Delta W_j^{(i)} z_{j-1}^{(i)} = U\left(\Sigma[V_j^{(i)}]^T z_{j-1}^{(i)}\right) = \sum_{k=1}^{r} u_k < \left(\Sigma[v_j^{(i)}]^T\right)_k, z_{j-1}^{(i)} > \tag{3}$$

We observe that the importance of the vectors in U is given by the scales (i.e., magnitudes) of elements of $\left(\Sigma[V_j^{(i)}]^T z_{j-1}^{(i)}\right) \in \mathbb{R}^r$. If $z_{j-1}^{(i)}$ has bounded magnitude (as is the case in transformer models), we can obtain an upper bound on the importance of an arbitrary u_k: $\|\left(\Sigma[v_j^{(i)}]^T\right)_k z_{j-1}^{(i)}\| \leq \|\left(\Sigma[v_j^{(i)}]^T\right)_k\| \|z_{j-1}^{(i)}\|$. Thus, $\|\left(\Sigma[v_j^{(i)}]^T\right)_k\| = \Sigma_{kk}\|[v_j^{(i)}]_k^T\|$ is its *data agnostic* importance.

Interestingly, when examining the $\|[V_j^{(i)}]_k^T\|$, we observe that different basis vectors are important for different tasks. For example Figure 4a demonstrates that the initial few basis vectors are important for the model updates fine-tuned on Cars (Krause et al., 2013) dataset while the latter few are important for the updates on GTSRB (Stallkamp et al., 2011).

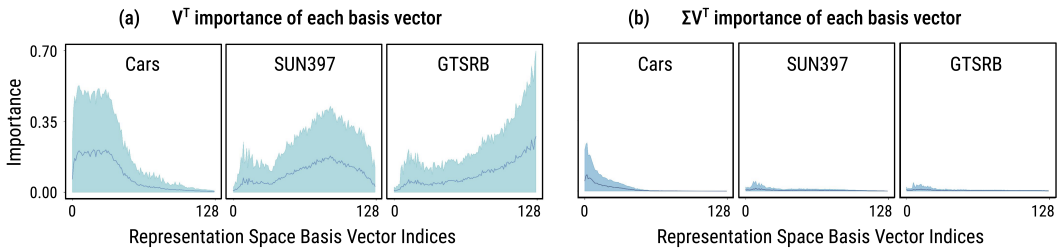


Figure 4: **while V shows importance per task, Σ shows global importance.** The importance distribution on the joint-basis from ViT-B/32 models finetuned on three tasks changes when Σ is multiplied or not with the V's. The x-axis is the ordered indices of basis vectors which span the shared representation space between all models. The y-axis is the weight each task assigns to every basis vector. Each plot shows the mean importance across all LoRA layers $\pm$ standard deviation. (a): Importance distribution according to the V's. (b): Importance distribution when Σ is multiplied in.

We find this intuitive: different parts of the shared representation space may be useful to each model to solve its required tasks. However, when adding Σ_{kk} into $\|[V_j^{(i)}]_k^T\|$—Figure 4b, we find that the importance distributions in each task become very homogeneous. We posit that this may yield importance weights that favour some tasks' skills while overlooking the needs of others.

Thus, we propose to alter Σ to give similar weight to every task, ideally achieving a better multitask model across all tasks. Specifically, we "reset" the diagonal values in Σ by replacing it with the identity matrix, thereby retaining the distinct emphasis each task assigns to the representation space in U. We further analyze the merit of the specific choice of resetting in Section 5.5, and compare it against other sophisticated choices. Note that merging methods (Wortsman et al., 2022b) typically re-scale the task updates, hence it is the relative weight in Σ that matters and not the absolute value.

5 Experiments and Results

We validate KnOTS in two different settings spanning six experiments. §5.2 evaluates KnOTS in the popular eight vision tasks setting introduced by Ilharco et al. (2022), and shows how KnOTS: (1) improves merging methods by an *average of 13%*, (2) scales *dramatically* better with more merges, (3) *improves* with larger pretrained models, and (4) creates *significantly* better "general" models than the ensemble in a novel setting specifically designed to measure generalization. The second setting (§5.4) shows how KnOTS generalizes better than prior work when merging models finetuned on *different domains*. Despite being this setting being significantly more challenging (numbers reflecting it), KnOTS outperforms prior work in both in-domain and out-of-domain (OOD) generalization—nearly achieving the ensemble. We also further analyze the different facets of KnOTS (§5.5).

5.1 Experimental Details

Models. We use ViT-B/32 or ViT-L/16 (Dosovitskiy et al., 2020) for all our vision experiments. These are two variants of the CLIP vision encoder (Radford et al., 2021) that are finetuned separately on a variety of tasks. Our natural language experiments are conducted with a LLama3-8B model (AI, 2024) PEFT with LoRA separately on different natural language inference (NLI) tasks. As in (Hu et al., 2021), we only apply LoRA on the weight matrices in the attention layers: namely the key, query, value and output layers. Unless otherwise specified, each LoRA has a rank of 16. Further training details are found in Appendix D.

Merging methods. The focus of this paper is improving merging performance *without further finetuning* on LoRA models. To the best of our knowledge, there are only three merging methods that do not rely on finetuning for our setting: Task-Arithmetic (TA) (Ilharco et al., 2022), TIES (Yadav et al., 2023) and RegMean (Jin et al., 2023). Both TA and TIES merge models in their original feature spaces; that is, they do not align the representation spaces of each weight before merging.

TA (Ilharco et al., 2022) merges models finetuned on different tasks with the same pretrained checkpoint by linearly summing their parameters. They apply a summation-weight (termed "scaling coefficient") to the parameters of each model that achieves the best merged model performance over a

Table 1: **Eight models per-task results.** We merge eight ViT-B/32 models finetuned with LoRA on different image classification datasets. "Finetuned" refers to the accuracy of each finetuned model on the dataset it was trained on. We report the per-task (including the average) normalized-accuracies across other merging baselines. These describe how close they get to the "Finetuned" accuracy. KnOTS-TIES improves over baselines by up to 13% average accuracy.

Method	Per-Task Normalized Accuracies (%)								
	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC45	SUN397	SVHN	Avg
Finetuned	74	58.3	99	92.7	99.3	88.4	64.5	96.2	84.1
RegMean	80.2	71.3	37.9	47.3	43.1	70.5	93.9	43.0	60.9
TA	82.1	73.6	48.8	42.3	53.1	71.5	97.5	41.2	63.8
TIES	82.6	72.8	49.5	38.3	56.9	69.9	97.1	44.1	63.9
Knots-TA	82.9	84.5	46.2	78.3	67.3	82.8	98.5	50.7	73.9
Knots-TIES	**83.9**	**86.9**	**50.8**	**79.5**	**78.4**	**82.9**	**98.6**	**54.2**	**76.9**

held-out validation set. TIES (Yadav et al., 2023) extends TA by reducing noise and conflicts between parameters when merged. It reduces noise by pruning k parameters with the lowest magnitudes, and reduces parameter conflicts by those that lie in outlying directions. TIES then linearly sums the model parameters with scaling coefficients (and k-pruning) that best merge over the held-out set. In contrast, RegMean (Jin et al., 2023) is a merging approach that simultaneously also *aligns* the representation spaces between the weights of each model. It does this by solving a closed-form locally linear regression problem at every model layer. KnOTS can be coupled with merging methods such as TA and TIES by applying them directly on the sub-parameters. This doesn't disrupt the hyperparameter tuning process, and enables each to operate over aligned representation spaces. In some experiments, we add the "Ensemble" as a baseline, which consists of all models used in a particular merging evaluation. The ensemble passes an input in parallel to all models, and predicts its class according to the highest confidence prediction across all models. Finally, we test out DARE (Yu et al., 2023) in junction with other models merging methods and find no improvement in performance, refer App. E for more details.

Metrics. We report the *absolute* accuracy of all "Finetuned" models on their respective datasets, and color them grey to indicate this. Similar to Ilharco et al. (2022); Yadav et al. (2023), we compare our merging methods via "normalized-accuracy" wherever applicable. This is obtained by dividing the performance of a merged model on a task (e.g., Cars (Krause et al., 2013)) by the performance of the original model finetuned on the task. For instance, the normalized accuracy for a given task-i is computed as $\frac{\text{Accuracy of merged model on task-i}}{\text{Accuracy of finetuned model on task-i}}$. This metric shows *how close* the merged model gets to the original finetuned model for each task. Certain experiment settings study different generalization properties and thus have different metrics. For the remainder of this paper, we assume all accuracy measurements are *normalized*, and define other metrics in their relevant sections.

5.2 Merging eight image classification models

We first follow the image classification benchmark from Ilharco et al. (2022) and merge models finetuned on eight different datasets: Cars (Krause et al., 2013), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), GTSRB (Stallkamp et al., 2011), MNIST (LeCun, 1998), RESISC45 (Cheng et al., 2017), SUN397 (Xiao et al., 2016) and SVHN (Netzer et al., 2011). We conduct several different experiments in this setting to validate KnOTS across diverse conditions.

KnOTS outperforms prior work. This experiment is the default setting of Ilharco et al. (2022); Yadav et al. (2023) and consists of recording how a merged model performs on each dataset *individually*. We term this experiment as "per-task evaluation" since the merged model is evaluated one-at-a-time on each dataset, using only its images and labels. Like Ilharco et al. (2022); Yadav et al. (2023), we report normalized per-task accuracies and their average. Tab. 1 reports merging performances. While TA and TIES both achieve similar performances, KnOTS significantly improves each by up to 13% on average. Notably, KnOTS substantially improves both TA and TIES, highlighting its applicability to different merging methods. We compare KnOTS against TA and TIES for the remainder of our experiments as they are the strongest baselines.

KnOTS benefits with larger models. We merge ViT-L/14 models finetuned with LoRA. Tab. 2 shows the averaged normalized accuracies of both ViT-B/32 and ViT-L/14 across all our merging methods. We observe that while the larger ViT-L/14 substantially improves performance on each task, a similar trend occurs when applying KnOTS on our baselines. This showcases how KnOTS can continue to improve merging methods, even as model size and capacity *increases*.

Table 2: **Normalized per-task avg accuracies** when merging different ViT models seperately finetuned with LoRA on eight image classification datasets. KnOTS consistently improves baselines, even as model size increases.

Metric	Method				
	Finetuned	TA	TIES	**KnOTS**-TA	**KnOTS**-TIES
ViT-B/32	84.1	63.8	63.9	73.9	76.9
ViT-L/14	92.3	74.5	74.7	84.0	84.4

KnOTS scales better with the number of models. Following Ilharco et al. (2022); Yadav et al. (2023), we also evaluate the performance of KnOTS as we increase the number of merged tasks. Fig. 1 illustrates the average normalized accuracies of the merged model as it's merged on increasing numbers of tasks. Dots denote the performance of a model evaluated *only* on the tasks involved in the merge, while the bars record 95% confidence intervals according to *all possible* combinations of tasks. The results use our ViT-B/32 models. Notably, KnOTS achieves *significantly* better merging results than baselines, and the performance gap *increases* as the number of tasks grow, highlighting its robustness.

KnOTS is robust to increased LoRA rank. Tab. 3 shows the per-task normalized accuracy results when merging 8 LoRA R128 models in the same setting as Tab. 1. We observe that even with the drastically higher rank LoRA, KnOTS is still capable of dramatically improving the existing baselines. Interestingly, the *scale* of improvement appears to diminish compared to that from the R16 variants. We hypothesize that as the rank increases, the difficulty of aligning their representation spaces without conflict increases–as there are more rank-components in each LoRA model. However, we expect the opposite to be true as the rank *decreases*. Nonetheless, we find that KnOTS is robust to very-large ranks and still significantly outperforms the baselines.

Table 3: **Normalized per-task avg accuracies** when merging ViT models seperately finetuned with LoRA R128 on eight image classification datasets. KnOTS consistently improves baselines, even as LoRA rank increases.

Metric	Method				
	Finetuned	TA	TIES	**KnOTS**-TA	**KnOTS**-TIES
LoRA128	87.2	47.2	59.8	59.5	67.2

KnOTS creates better general models. We further introduce a new experimental setting to the eight vision task benchmark that we call "joint-task." This task differs from the "per-task" regime in that it evaluates merged models over the union of all inputs and labels across *every* merged task. The joint-task is significantly more challenging than the per-task counterpart as it explicitly examines whether a merged model is a *general* model. Notably, after aggregating the labels across all eight tasks and removing duplicates (e.g., MNIST (LeCun, 1998) and SVHN (Netzer et al., 2011) have the same labels), we obtain 748 unique labels. As some labels in one task are hyponyms of those in another (e.g., "islet" in SUN397 (Xiao et al., 2016) and "island" in RESISC45 (Cheng et al., 2017)) making it challenging to distinguish labels, we report performance using "Hits@k." This refers to the number of times the expected label is within the top-k predictions of a model (e.g., Hits@1 is accuracy).

Table 4: **Eight models joint-task results.** We merge eight ViT-B/32 models finetuned with LoRA on different datasets. We report the joint-task "Union" performances for several merging methods. KnOTS-TIES performs the best.

Metric	Method				
	Ensemble	TA	TIES	**KnOTS**-TA	**KnOTS**-TIES
Hits@1	40.7	43.2	43.5	51.6	**54.6**
Hits@3	63.1	65.0	65.7	70.7	**74.0**
Hits@5	72.6	73.8	74.4	77.6	**81.0**

Tab. 4 shows the results over the "Union" of all images and labels across all tasks, when merging our ViT-B/32 LoRA models. Please see App. G for performances broken down by dataset. Overall, KnOTS-TIES significantly outperforms every baseline at all Hits@k levels on the extremely challenging "Union" evaluation—by up to 11.1%. Interestingly, we also observe that the ensemble performs particularly poorly in the joint setting. We posit this is due to certain models making over-confidently incorrect predictions on data from tasks they are not finetuned on, a notable issue in ensemble models (Kardan et al., 2021). We argue that the joint-task is an important benchmark for assessing a merged model's generality to any task, and we hope future work expands upon this setting.

Table 6: **ID and OOD performance on DomainNet.** We merge five LoRA ViT-B/32 models finetuned on the domains from the DomainNet dataset with different merging methods. We report normalized accuracy of each merged model in two settings: (a) in-domain (ID) and (b) out-of-domain (OOD). KnOTS-TIES outperforms all baselines in ID, and nearly matches the ensemble in OOD.

(a) **In-Domain (ID)** DomainNet accuracy.

Method	Per-Task Normalized Accuracies (%)					
	Clipart	Painting	Quickdraw	Real	Sketch	Avg
Finetuned	100.0	100.0	100.0	100.0	100.0	100.0
TA	22.7	22.0	7.9	24.7	21.8	19.8
TIES	29.3	25.0	9.7	31.3	29.4	24.9
Knots-TIES	23.9	**30.5**	**12.2**	**37.2**	**34.2**	**29.5**
	Reference Per-Task Raw Accuracies (%)					
Finetuned	71.5	61.9	61.3	75.7	61.2	66.3

(b) **Out-of-Domain (OOD)** DomainNet accuracy.

Method	Out-of-Domain Normalized Accuracies (%)					
	Clipart	Painting	Quickdraw	Real	Sketch	Avg
Ensemble	100.0	100.0	100.0	100.0	100.0	100.0
TA	24.5	**29.4**	7.0	23.6	25.2	22.0
TIES	28.9	23.8	8.9	30.9	28.2	24.1
Knots-TIES	**32.1**	25.6	**10.8**	**33.6**	**30.5**	**26.5**
	Reference Per-Task Raw Accuracies (%)					
Ensemble	44.7	13.4	20.1	24.5	40.0	28.5

5.3 Merging six natural language inference models

Table 5: **Six models per-task results.**

Method	Per-Task Normalized Accuracies (%)						
	SNLI	MNLI	SICK	QNLI	RTE	SciTail	Avg
Finetuned	92.1	90.3	91.6	95.1	89.9	94.1	92.2
TA	73.0	72.2	46.8	64.5	91.9	42.2	65.1
TIES	-	-	-	-	-	-	-
Knots-TA	71.5	70.6	49.7	65.0	94.4	62.5	69.0
Knots-TIES	-	-	-	-	-	-	-

We also evaluate KnOTS in the NLI setting, by merging six PEFT llama3-8B (AI, 2024) models finetuned on SNLI (Bowman et al., 2015), MNLI (Williams et al., 2018), SICK (Marelli et al., 2014), QNLI and RTE (Wang et al., 2018), and SCITAIL (Khot et al., 2018). Each of these tasks involves performing a 3-way classification to determine whether a given "hypothesis" is true (entailment), false (contradiction) or inconclusive (neutral) compared to a given "premise." Although QNLI, RTE and SCITAIL only require two of the three classes, we maintain a 3-way classification head to ensure consistent layer sizes for each model, and pass loss over all labels. In the case of our baselines, we merge all parameters with the same method (e.g., TA or TIES). Similarly, when applying KnOTS, we merge all classification layer parameters with a baseline method (e.g., TA) and the LoRA parameters with the KnOTS variant (e.g., KnOTS-TA). Overall, we observe that KnOTS comprehensively outperforms our baselines in th NLI setting by up to 3.9% average normalized accuracy. This demonstrates the power of applying KnOTS on even dramatically larger models (8B parameters) and its robustness across modalities.

5.4 Measuring generalization with DomainNet

In many use cases, multitask models must be robust to different domains and generalize well to unseen domains. In this section, we conduct two experiments studying each of these generalization settings. Specifically, we finetune several ViT-B/32 models with LoRA on five domain-splits from the DomainNet (Muandet et al., 2013) benchmark: ClipArt, Painting, QuickDraw, Real, and Sketch. DomainNet is significantly more challenging than the previous eight-task setting. Not only is each domain very dissimilar to others, but their data distributions are more heterogeneous (Muandet et al., 2013). We then merge each model together using task-arithmetic, TIES, and our KnOTS-TIES – the best performing merge combination from previous experiments. The first experiment measures how well the merged models is capable of performing in-domain (ID), and reports normalized accuracy. The second experiment measures the out-of-domain (OOD) accuracy, which consists of merging models finetuned on a set of tasks and evaluating the merged model on a new held-out task. Specifically, if we have n models each finetuned on n tasks, OOD generalization measures the performance from merging every combination of $n-1$ models and evaluating the merged model on the remaining task.

KnOTS significantly outperforms prior work ID. Tab. 6a compares the performances of each merging algorithm across all domains, along with the performance from the Ensemble. While the overall performance of each merged model is significantly lower than the original "finetuned" models, this setting is also significantly more challenging than the previous eight-task setting. Despite these difficulties, KnOTS significantly improves baseline by up to 4.6% on average.

KnOTS nearly matches the ensemble for OOD generalization. Tab. 6b reports the results for OOD generalizations. This benchmark is *extremely* challenging, as each DomainNet domain has very little

distribution overlap with other domains—even the ensemble has substantial difficulty. We do not employ domain adaptation strategies such as (Chattopadhyay et al., 2023) to isolate any performance differences between methods as solely due to their robustness. Importantly, KnOTS-TIES significantly improves upon baselines across nearly every metric in this extraordinarily challenging setting and nearly matches the ensemble performance—emphasizing its robustness.

5.5 Understanding each component of KnOTS

We conduct ablation experiments to better understand each facet of KnOTS: (1) applying merging methods over the aligned sub-parameters, and (2) using different strategies to reset Σ. Each ablation is conducted using the KnOTS-TIES method as it performs the best across all our experimental settings. Table 7 summarizes the effect of each ablation, and more detailed results are in Table A1.

First, we observe that both merging over the sub-parameters and resetting Σ are crucial to achieving the best performance. That is, performing one and not the other substantially decreases accuracy by 5.0% and 12.5% respectively. Second, we posit that our strategy of resetting Σ may not be optimal. In fact, finding the "singular-values" for each model that achieve the best merged model by jointly fine-tuning on the training datasets of each task (a very expensive and impractical setting) increases the average merge accuracy to 93.3%. We consider this an upperbound as it illustrates the merging performance ceiling and further highlights the true gap to our best model.

Table 7: **Ablations on KnOTS Components.** We merge eight ViT-B/32 models on the eight vision datasets using different strategies to reset the Σ, and merging in the aligned V-space. KnOTS-TIES is the best non-finetuning method.

Method	Merge Configuration		Avg. Norm. Acc. (%)
	Over V's?	Σ	
KnOTS-TIES	✓	Reset	76.9
Only reset Σ	✗	Reset	71.9
Use original Σ	✓	Original	66.4
	✓	Finetuned	93.3
	✓	$\sim \mathcal{U}$	75.1
TIES	✗	✗	63.9

Interestingly, analyzing the distributions of the learned "singular-values" from the finetuning ablation shows them spread close to a uniform distribution. Thus, we conduct an additional ablation experiment, which resets Σ by sampling each value from a uniform distribution. This substantially improves performance compared to preserving the original Σ, yet underperforms compared to resetting it as in KnOTS. We find these results very interesting and find the large performance gap to the finetuning variant particularly motivating for future exploration.

6 Conclusion

In this paper, we study merging LoRA models sharing the same pretrained checkpoint and finetuned on different tasks without additional finetuning. We find that prior work does not transfer well in this setting, and observe that is due to parameter misalignment between LoRA models. We introduce KnOTS to tackle this problem by using the singular value decomposition (SVD) to map the parameters of each LoRA model into a shared representation space with aligned elements, over which prior work can be applied to merge the parameters. Through additional analysis, we observe that the computed singular values prune pieces in the representation space useful to certain models, which can negatively impact the merged model's performance on their tasks. Together with the SVD and "resetting" these singular values to one, KnOTS improves prior work by up to 13% across an extensive set of benchmarks. For limitations and potential impact, refer to App. H.

References

Meta AI. Meta llama 3. https://llama.meta.com/llama3/, 2024. 6, 9

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv:2209.04836*, 2022. 4

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015. 9

Ruisi Cai, Zhenyu (Allen) Zhang, and Zhangyang Wang. Robust weight signatures: Gaining robustness as easy as patching weights? *ArXiv*, abs/2302.12480, 2023. URL https://api.semanticscholar.org/CorpusID:257205703. 1

Prithvijit Chattopadhyay, Kartik Sarangmath, Vivek Vijaykumar, and Judy Hoffman. Pasta: Proportional amplitude spectrum training augmentation for syn-to-real domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19288–19300, 2023. 10

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017. 7, 8, 17

Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. Fusing finetuned models for better pretraining. *arXiv preprint arXiv:2204.03044*, 2022. 1, 3

Leshem Choshen, Elad Venezian, Shachar Don-Yehiya, Noam Slonim, and Yoav Katz. Where to start? analyzing the potential value of intermediate models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1446–1470, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.90. URL https://aclanthology.org/2023.emnlp-main.90. 1

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*, pp. 3606–3613, 2014. 7, 17

Nico Daheim, Thomas Möllenhoff, Edoardo Maria Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. *arXiv preprint arXiv:2310.12808*, 2023. 3

Shachar Don-Yehiya, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Cold fusion: Collaborative descent for distributed multitask finetuning. *arXiv preprint arXiv:2212.01378*, 2022. 1

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv:2010.11929*, 2020. 2, 6

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *ICML*, 2018. 2

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv:2110.06296*, 2021. 1, 4

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv:1803.03635*, 2018. 2

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *NeurIPS*, 2018. 2, 4

Almog Gueta, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Knowledge is a region in weight space for fine-tuned language models. *arXiv:2302.04863*, 2023. 1, 4

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019. 7, 16

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021. 2

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. 1, 3, 6

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*, 2023. 2

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv:2212.04089*, 2022. 1, 2, 3, 4, 6, 7, 8, 15, 17

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. URL https://openreview.net/forum?id=FCnohuR6AnM. 6, 7

Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. Repair: Renormalizing permuted activations for interpolation repair. *arXiv:2211.08403*, 2022. 4, 15

Navid Kardan, Ankit Sharma, and Kenneth O Stanley. Towards consistent predictive confidence through fitted ensembles. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, 2021. 8

Tushar Khot, Ashish Sabharwal, and Peter Clark. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 9

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013. doi: 10.1109/ICCVW.2013.77. 5, 7, 16

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 7, 8, 16

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent Learning: Do different neural networks learn the same representations? *arXiv:1511.07543*, 2015. 4

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (eds.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 2014. URL http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf. 9

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. *arXiv:2111.09832*, 2021. 1, 2, 17

Krikamol Muandet, David Balduzzi, and Bernhard SchÃűlkopf. Domain generalization via invariant feature representation. In *ICML*, 2013. 9

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 4. Granada, 2011. 7, 8, 17

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *NeurIPS*, 2020. 2

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 2

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 3

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 16

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1406–1415, 2019. 17

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2, 6, 16

Colin Raffel. Building machine learning models like open source software. *Communications of the ACM*, 66(2):38–40, 2023. 1

Alexandre Rame, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. *Advances in Neural Information Processing Systems*, 35:10821–10836, 2022. 3

Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187*, 2024. 3

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084. 18

Viraj Shah, Nataniel Ruiz, Forrester Cole, Erika Lu, Svetlana Lazebnik, Yuanzhen Li, and Varun Jampani. Ziplora: Any subject in any style by effectively merging loras. *arXiv preprint arXiv:2311.13600*, 2023. 3

Berfin Simsek, François Ged, Arthur Jacot, Francesco Spadaro, Clement Hongler, Wulfram Gerstner, and Johanni Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021. 2

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*, pp. 1453–1460. IEEE, 2011. 5, 7, 16

George Stoica, Daniel Bolya, Jakob Brandt Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LEYUkvdUhq. 1, 3, 4, 15

Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. Parameter-efficient multi-task model fusion with partial linearization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=iynRvVVAmH. 1, 3, 4

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2

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. 9

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018. URL https://aclanthology.org/N18-1101. 9

Mitchell Wortsman, Gabriel Ilharco, Samir Ya 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. In *ICML*, 2022a. 1, 3, 4

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *CVPR*, 2022b. 1, 2, 3, 6

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 (IJCV)*, 119(1):3–22, August 2016. 7, 8, 17

Chen Xu, Weiwei Xu, and Kaili Jing. Fast algorithms for singular value decomposition and the inverse of nearly low-rank matrices. *National Science Review*, 10(6):nwad083, 03 2023. ISSN 2095-5138. doi: 10.1093/nsr/nwad083. URL https://doi.org/10.1093/nsr/nwad083. 16

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Resolving interference when merging models. *arXiv:2306.01708*, 2023. 1, 2, 3, 4, 6, 7, 8, 15, 17

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*, 2023. 3, 7, 17

Kerem Zaman, Leshem Choshen, and Shashank Srivastava. Fuse to forget: Bias reduction and selective memorization through model fusion. *ArXiv*, abs/2311.07682, 2023. URL https://api.semanticscholar.org/CorpusID:265158134. 1, 4

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in low-rank adapters of foundation models. *arXiv preprint arXiv:2402.16842*, 2024. 4

A WHICH LORA REPRESENTATION IS BEST FOR MERGING?

LoRA is a popular Parameter Efficient Finetuning (PEFT) method that finetunes models by applying low-rank updates to the pretrained parameters. For instance, let $W_j^{(i)}, \Delta W_j^{(i)} \in \mathbb{R}^{O \times I}$. With LoRA, each $\Delta W_j^{(i)}$ is rank $r << \min(O, I)$, and can be factorized into two low-rank matrices $A_j^{(i)} \in \mathbb{R}^{r \times I}, B_j^{(i)} \in \mathbb{R}^{O \times r}$, with $\Delta W_j^{(i)} = B_j^{(i)} A_j^{(i)}$. The existence of $A_j^{(i)}$, and $B_j^{(i)}$ for each $\Delta W_j^{(i)}$ may suggest that merging should be done on them separately, and the results multiplied to achieve the merged-update $\Delta W_j^{(m)}$. However, this immediately leads to issues when applying existing merging methods. For instance, let us apply task-arithmetic (TA) (Ilharco et al., 2022) on a single arbitrary layer (e.g., the j^{th} layer) across $f^{(1)}, \ldots, f^{(n)}$. First, TA merges the factorizations of $\Delta W_j^{(1)}, \ldots, \Delta W_j^{(n)}$ by computing $\sum_{i=1}^{n} \lambda^{(i)} A_j^{(i)}, \sum_{i=1}^{n} \lambda^{(i)} B_j^{(i)}$. Multiplying each to obtain $\Delta W_j^{(m)}$ yields,

$$\Delta W_j^{(m)} = \left(\sum_{i=1}^{n} \lambda^{(i)} B_j^{(i)} \right) \left(\sum_{i=1}^{n} \lambda^{(i)} A_j^{(i)} \right) \tag{4}$$

$$= \underbrace{\left(\sum_{i=1}^{l} \lambda^{(i)} \lambda^{(i)} B_j^{(i)} A_j^{(i)} \right)}_{\text{Products of aligned factorizations}} + \underbrace{\left(\sum_{i \neq k}^{l} \lambda^{(i)} \lambda^{(k)} B_j^{(i)} A_j^{(k)} \right)}_{\text{Products of misaligned factorizations.}} . \tag{5}$$

Notably, $\Delta W_j^{(m)}$ consists of two terms: one composed of $B_j^{(i)}$ and $A_j^{(i)}$ matrices from the updates of the same model, and one composed of $B_j^{(i)}$ and $A_j^{(k)}$ matrices from updates of different models. Unfortunately, there is no guarantee that $B_j^{(i)}$ expects the same input representation as the output of $A_j^{(k)}$ when $i \neq k$, and including these terms in the merged model can incur significant drop in model performance (Stoica et al., 2024). Similarly, this same issue persists as well when applying (Yadav et al., 2023). Thus, we only conduct merging on the original update representations: $\Delta W_j^{(i)}$ of each model. This trivially avoids mismatches when applying merging methods.

B COMPUTING FEATURE CORRELATIONS ACROSS TASKS

We create the correlation matrices presented in Fig. 3 from models that are finetuned on the different tasks defined in §5.2. Each matrix is generated using data sampled from a custom dataset we create using the existing validation data of each task. Specifically, our dataset consists of (1) the validation set of a task if it exists, or (2) a randomly sampled 20% of the test set. Importantly and for fair comparison, both correlation plots are created using the *same data*. Each correlation matrix shows the average pairwise correlations of the outputs between the finetuning-updates of two separate models, across all layers.

Pairwise correlations on full-rank finetuned models (Fig. 3a). We use the set of eight full-rank finetuned (FFT) ViT-B/32 models released by (Ilharco et al., 2022) that are finetuned from the same pretrained model on each of the eight tasks, and also obtain the pretrained model. We then collect the intermediate outputs of every key, query, value and output projection layer in each of the nine models, using all the data from our validation set. This yields eight different intermediate representations—each belonging to a distinct finetuned model—and one intermediate representation belonging to the pretrained model, for each layer. Using this information, we can obtain the *contribution* of a finetuning update on a respective layer by subtracting the pretrained model's representation at the layer from that of the respective finetuned model. Thus, doing this for the intermediate representations of all finetuned models across all layers gives the contributions of finetuning updates at each layer. Once these contributions are computed across all our data, we find the pairwise correlations between the contributions of two finetuned models at each layer, following the procedure of (Jordan et al., 2022; Stoica et al., 2024). Finally, we average the magnitudes (absolute values) of correlations across all layers for each model pair, and report the results in Fig. 3a.

Pairwise correlations on LoRA finetuned models (Fig. 3b). We use the eight LoRA finetuned ViT-B/32 models we merge in §5.2. We compute the correlations of LoRA finetuned models in the exact same way as their FFT counterparts, with one difference. Specifically, the finetuning-updates in a LoRA model is given entirely by the LoRA projection layer. Thus, we collect the contributions of these updates to the pretrained model by simply taking the outputs of each LoRA projection layer. The remainder of the procedure is equivalent to that used for FFTs. Results are reported in Fig. 3b.

C Full Ablation Results

Table A1: **Ablation KnOTS components.** We merge eight ViT-B/32 models on the eight vision datasets using different strategies to reset the Σ and merging on the sub-parameters. Merging both on the sub-parameters and resetting Σ is critical to achieving the best performance across all datasets, and is only surpassed when the elements of Σ are finetuned over the training data from all datasets.

Method	Configuration		Normalized Per-Task Accuracies (%)								
	Over V	Σ	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC45	SUN397	SVHN	Avg
KnOTS-TIES	✓	Reset	**83.9**	**86.9**	50.8	79.5	**78.4**	**82.9**	**98.6**	54.2	**76.9**
	✗	Reset	82.7	82.2	**52.5**	68.9	63.7	79.2	**98.6**	47.5	71.9
	✓	✗	83.4	72.0	49.9	38.4	66.5	70.5	96.2	54.2	66.4
	✓	Finetuned	92.5	78.5	95.4	93.4	98.2	95.4	97.6	95.4	93.3
	✓	$\sim \mathcal{U}$	82.2	83.0	44.0	**79.6**	78.2	81.4	97.1	**55.7**	75.1
TIES	✗	✗	82.6	72.8	49.5	38.3	56.9	69.9	97.1	44.1	63.9

D Training Details

In this section, we specify the details associated with training the LoRA models used across all experiments in 5.1.

We make use of the Open-CLIP based (Radford et al., 2021) ViT-B/32 and ViT-L/14 models from HuggingFace. These models are then LoRA fine-tuned using the PEFT LoRA library. Across all our experiments we initialize the LoRA layers across the query, key, value and output layer. Note that these are the only learnable layers. We set the LoRA rank to be 16, LoRA alpha to be 16, LoRA dropout to be 0.1 and disable the use of bias parameters. All models are trained using the AdamW optimizer, with a cosine learning rate scheduler using Cross-Entropy loss.

The ViT-B/32 models were fine-tuned on the 8 vision tasks using a standard learning rate of 1e-5, weight decay of 1e-1 and label smoothing set to 0.

The ViT-L/14 models were fine-tuned on the 8 vision tasks using a standard learning rate of 3e-4, weight decay of 1e-4 and label smoothing set to 0.

The ViT-B/32 models were fine-tuned on each domain in domainNet using a standard learning rate of 1e-3, weight decay of 1e-4 and label smoothing set to 0.1.

Across all our experiments the text encoder in the Clip model remains frozen and the text embeddings are obtained by passing the class labels through the text encoder.

Compute resources. All of our experiments were conducted on machines with one Nvidia A40 with 48GB of VRAM, and a CPU that has 8 workers. We trained all our models across these machines, and also applied every merging algorithm in this environment. KnOTS is capable of running entirely on the CPU in seconds, and uses at most 2GB RAM over the entire merging process for all our models. Specifically, our most time-intensive setting was that of §5.2. Here it takes KnOTS 49 seconds (and 1.18GB of RAM) to merge all the ViT-B/32 models, and 149 seconds (and 1.94GB of ram) to merge all ViT-L/16 models. We compute the SVD using the Pytorch (Paszke et al., 2019) "torch.linalg.svd" solver. However, we note that more efficient SVD algorithms can easily be employed with our approach, such as the recent Fast SVD (Xu et al., 2023) which is designed for low-rank matrices.

Datasets and licenses. This paper uses the following datasets and associated licenses. Cars (Krause et al., 2013) and GTSRB (Stallkamp et al., 2011) both use the Creative Commons License. EuroSAT (Helber et al., 2019) is under the MIT license and MNIST (LeCun, 1998) is under the Gnu General

Table A2: **Eight models per-task results including Fisher weight averaging.** We merge eight ViT-B/32 models finetuned with LoRA on different image classification datasets. "Finetuned" refers to the accuracy of each finetuned model on the dataset it was trained on. We report the per-task (including the average) normalized-accuracies across other merging baselines. These describe how close they get to the "Finetuned" accuracy.

Method	Per-Task Accuracies (%)								
	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC45	SUN397	SVHN	Avg
Finetuned	74	58.3	99	92.7	99.3	88.4	64.5	96.2	84.1
RegMean	80.2	71.3	37.9	47.3	43.1	70.5	93.9	43.0	60.9
Fisher	84.5	72.4	44.4	55..8	47.8	70.9	96.1	39.2	63.9
TA	82.1	73.6	48.8	42.3	53.1	71.5	97.5	41.2	63.8
TIES	82.6	72.8	49.5	38.3	56.9	69.9	97.1	44.1	63.9
Knots-TA	82.9	84.5	46.2	78.3	67.3	82.8	98.5	50.7	73.9
Knots-TIES	**83.9**	**86.9**	**50.8**	**79.5**	**78.4**	**82.9**	**98.6**	**54.2**	**76.9**

Public License. We could not find the licenses of DTD (Cimpoi et al., 2014), RESISC45 (Cheng et al., 2017), SVHN (Netzer et al., 2011) and SUN397 (Xiao et al., 2016). DomainNet (Peng et al., 2019) is distributed under a custom research-only, non-commercial license.

E Comparisons with DARE

In this section we discuss the merging method DARE(Yu et al., 2023). The method involves randomly dropping the task updates by a ratio of p and then rescaling the remaining ones by $1/(1-p)$ to recover back the original embeddings. This method is typically used in conjunction with other model-merging baselines like Task-Arithmetic(TA)(Ilharco et al., 2022) and TIES(Yadav et al., 2023). We run DARE in conjunction with TA, TIES, KnOTS-TIES, KnOTS-TA and tune the pruning coefficient p from 0.0 to 1.0 with steps of 0.1 to maximize the average normalized accuracy. After tuning we find that none of the base-lines saw any improvement in performance when using DARE. We suspect this may be due to the presence of fewer redundant parameters when merging LoRA fine-tuned models. Additionally, we observe that the task-updates in our LoRA models tend to have higher magnitudes, often exceeding 5e-3, and hence are less redundant.

F Comparison with a finetuning benchmark

We also compare KnOTS against the popular finetuning benchmark Fisher Weight Averaging (Matena & Raffel, 2021) in the same setting as Table 1. We classify this method as finetuning because it adds learnable parameters to every model that are optimized via gradient descent to obtain the best merged model. We select the best reported Fisher model according to the hyperparameter configuration which achieved the best performance on the same held-out validation data as all methods in this setting. Our hyperparameter search consisted of two ranges. First, the number of examples used to compute the Fisher information matrices: [256, 512, 1024, 2048] (selected from the training data of each dataset). Second, the scaling term to merge model parameters: [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]. The best performing configuration on our held-out validation dataset used a scaling coefficient of 1.0 and required 256 examples to compute the Fisher weights. Table A2 summarizes the results. Overall, we observe that while Fisher matches TIES, it is substantially outperformed by KnOTS—a method that doesn't require any training.

G Joint task full performances

Table A3 shows performances for each merging method across all datasets on the Joint-task. Each dataset column shows results on images *only* from the dataset, but with using the *joint* labels-set of 748 labels.

Finding synonymous labels. After removing duplicates from the joint label space, we are left with 748 total labels. However, a manual search over these labels reveals that some may be synonyms/hy-

Table A3: **Eight models joint-task results.** We merge eight ViT-B/32 models finetuned with LoRA on different datasets. We report the joint-task performances for several merging methods. KnOTS-TIES improves over baselines by significant margins across across nearly every evaluation.

Method	Metric	Joint-Task Performances (%)								
		Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC45	SUN397	SVHN	Union
Ensemble	Hits@1	58.5	41.3	16.9	29.5	35.8	54.2	**62.4**	25.1	40.7
	Hits@3	83.6	61.4	**31.4**	59.5	58.0	78.2	**84.0**	44.3	63.1
	Hits@5	91.0	72.4	**40.0**	73.0	69.1	85.9	**89.5**	55.2	72.6
TA	Hits@1	60.7	40.7	15.3	38.8	31.8	59.7	61.9	29.2	43.2
	Hits@3	84.9	63.7	23.0	66.1	48.4	83.6	83.9	50.1	65.0
	Hits@5	92.0	74.0	31.0	77.9	55.7	90.2	**89.9**	61.6	73.8
TIES	Hits@1	61.3	39.6	12.8	64.1	48.6	50.2	61.2	32.9	43.5
	Hits@3	85.4	62.2	19.4	64.1	48.6	82.9	83.5	54.6	65.7
	Hits@5	92.5	73.4	24.4	75.7	54.5	89.3	89.7	66.2	74.4
KnOTS-TA	Hits@1	61.3	**46.1**	18.5	72.6	45.9	**67.3**	61.4	39.9	51.6
	Hits@3	85.9	69.1	24.0	88.7	63.0	**91.6**	83.5	**60.2**	70.7
	Hits@5	92.4	78.0	31.6	93.3	69.5	**96.0**	89.7	**70.6**	77.6
KnOTS-TIES	Hits@1	**61.4**	42.3	**19.0**	**74.1**	**54.2**	65.4	59.8	**41.4**	**54.6**
	Hits@3	**85.7**	**69.0**	27.8	**90.5**	**68.3**	90.7	81.8	**60.2**	**74.0**
	Hits@5	**92.7**	**78.2**	37.4	**94.5**	**74.3**	95.4	88.5	69.1	**81.0**

ponyms of each other. To get a better idea of how many there are, we encode each label using a pretrained "distilbert-base-nli-mean-tokens" model taken from the SentenceTransformers library (Reimers & Gurevych, 2019), and compute the pairwise cosine-similarities between each label. We then take all label pairs with cosine-similarity ≥ 0.8 to be "synonyms," and reproduce them at the bottom of this section in dictionary form where keys and values are tuples containing "(label, dataset origin)." In total, we find 111 synonyms, leading to an average of 0.15 synonyms *per-label* and 1.95 synonyms *per-labels that have synonyms*. Thus, measuring performance on the joint-task using "Hits@k" with $k = \{1, 3, 5\}$ appears to be a suitable choice.

Below we print all synonyms found by our automatic search.

```
found_synonyms = {
('lake natural', 'sun397'): [('lake', 'resisc45')],
('forest', 'resisc45'): [
    ('tree house', 'sun397'), ('rainforest', 'sun397'),
    ('forest broadleaf', 'sun397'), ('forest road', 'sun397'),
    ('forest needleleaf', 'sun397'), ('forest path', 'sun397')
],
('athletic field outdoor', 'sun397'): [('ground track field', 'resisc45')],
('lake or sea', 'eurosat'): [('lake', 'resisc45')],
('rainforest', 'sun397'): [('forest', 'resisc45')],
('iceberg', 'sun397'): [('snowberg', 'resisc45')],
('meadow', 'resisc45'): [
    ('field wild', 'sun397'), ('park', 'sun397'), ('field cultivated', 'sun397'),
    ('pasture land', 'eurosat'), ('yard', 'sun397'), ('pasture', 'sun397')
],
('pond', 'sun397'): [('lake', 'resisc45')],
('ground track field', 'resisc45'): [
    ('athletic field outdoor', 'sun397'), ('track outdoor', 'sun397')
],
('desert vegetation', 'sun397'): [('desert', 'resisc45')],
('marsh', 'sun397'): [('wetland', 'resisc45')],
('freeway', 'resisc45'): [('highway', 'sun397')],
('islet', 'sun397'): [('island', 'resisc45')],
('permanent crop land', 'eurosat'): [
    ('rectangular farmland', 'resisc45'), ('field cultivated', 'sun397')
],
('track outdoor', 'sun397'): [('ground track field', 'resisc45')],
('swamp', 'sun397'): [('wetland', 'resisc45')],
('sea ice', 'resisc45'): [('ice shelf', 'sun397'), ('ice floe', 'sun397')],
```

```
('desert', 'resisc45'): [
    ('desert vegetation', 'sun397'), ('desert sand', 'sun397')
],
('snowberg', 'resisc45'): [
    ('iceberg', 'sun397'), ('mountain snowy', 'sun397'), ('snowfield', 'sun397')
],
('railway station', 'resisc45'): [
('train railway', 'sun397'), ('railroad track', 'sun397'), (
'train station platform', 'sun397')
],
('street', 'sun397'): [
    ('commercial area', 'resisc45'), ('intersection', 'resisc45')
],
('rectangular farmland', 'resisc45'): [('permanent crop land', 'eurosat')],
('field wild', 'sun397'): [('meadow', 'resisc45')],
('thermal power station', 'resisc45'): [('electrical substation', 'sun397')],
('lake', 'resisc45'): [
    ('lake natural', 'sun397'), ('lake or sea', 'eurosat'), ('pond', 'sun397')
],
('runway', 'sun397'): [('airplane', 'resisc45'), ('airport', 'resisc45')],
('baseball field', 'sun397'): [('baseball diamond', 'resisc45')],
('industrial area', 'sun397'): [
    ('industrial buildings or commercial buildings', 'eurosat')
],
('ice shelf', 'sun397'): [('sea ice', 'resisc45')],
('airplane', 'resisc45'): [('runway', 'sun397'), ('airplane cabin', 'sun397')],
('park', 'sun397'): [('forest', 'resisc45'), ('meadow', 'resisc45')],
('thermal power station', 'resisc45'): [('electrical substation', 'sun397')],
('electrical substation', 'sun397'): [('thermal power station', 'resisc45')],
('field cultivated', 'sun397'): [
    ('meadow', 'resisc45'), ('permanent crop land', 'eurosat'),
    ('circular farmland', 'resisc45'), ('pasture land', 'eurosat')
],
('patio', 'sun397'): [('terrace', 'resisc45')],
('shopfront', 'sun397'): [('commercial area', 'resisc45')],
('highway', 'sun397'): [('freeway', 'resisc45'), ('highway or road', 'eurosat')],
('circular farmland', 'resisc45'): [
    ('field cultivated', 'sun397'), ('pasture land', 'eurosat'),
    ('pasture', 'sun397')
],
('residential buildings or homes or apartments', 'eurosat'): [
    ('residential neighborhood', 'sun397')
],
('pasture land', 'eurosat'): [
    ('meadow', 'resisc45'), ('field cultivated', 'sun397'),
    ('circular farmland', 'resisc45'), ('yard', 'sun397'),
    ('pasture', 'sun397')
],
('desert sand', 'sun397'): [('desert', 'resisc45')],
('train railway', 'sun397'): [('railway', 'resisc45')],
('wetland', 'resisc45'): [('marsh', 'sun397'), ('swamp', 'sun397')],
('commercial area', 'resisc45'): [
    ('street', 'sun397'), ('shopfront', 'sun397'),
    ('industrial buildings or commercial buildings', 'eurosat')
],
('industrial buildings or commercial buildings', 'eurosat'): [
    ('industrial area', 'sun397'), ('commercial area', 'resisc45')
],
('airport', 'resisc45'): [('airport terminal', 'sun397'), ('runway', 'sun397')],
```

```
('tennis court outdoor', 'sun397'): [('tennis court', 'resisc45')],
('baseball diamond', 'resisc45'): [
    ('baseball field', 'sun397'), ('stadium baseball', 'sun397'),
    ('batters box', 'sun397')
],
('railway', 'resisc45'): [
    ('train railway', 'sun397'), ('railroad track', 'sun397'),
    ('train station platform', 'sun397')
],
('highway or road', 'eurosat'): [('highway', 'sun397')],
('yard', 'sun397'): [('meadow', 'resisc45'), ('pasture land', 'eurosat')],
('railroad track', 'sun397'): [
('railway station', 'resisc45'), ('railway', 'resisc45')
],
('tennis court', 'resisc45'): [
    ('tennis court outdoor', 'sun397'), ('tennis court indoor', 'sun397')
],
('residential neighborhood', 'sun397'): [
    ('residential buildings or homes or apartments', 'eurosat')
],
('island', 'resisc45'): [('islet', 'sun397')],
('train station platform', 'sun397'): [('railway station', 'resisc45')],
('pasture', 'sun397'): [
    ('meadow', 'resisc45'), ('circular farmland', 'resisc45'),
    ('pasture land', 'eurosat')
],
('terrace', 'resisc45'): [
    ('courtyard', 'sun397'), ('promenade deck', 'sun397'),
    ('pavilion', 'sun397'), ('patio', 'sun397'),
    ('carrousel', 'sun397'), ('balcony exterior', 'sun397'),
    ('veranda', 'sun397')
],
}
```

H LIMITATIONS AND POTENTIAL IMPACT

KnOTS contributes to more efficient model merging which is data-efficient and training-free, creating multi-task models by accumulating skills from multiple trained/finetuned models. To the best of our knowledge, KnOTS is the first to address the performance gap that previous model merging methods face when applied on LoRA finetuned models.

The possible limitations of our method are as follows. We have made two main observations which motivate our method, but they do not necessarily imply a causal relationship to the success of the merge. While these observations are interesting by themselves and give us some intuition about how models behave, it is possible that the reason for the performance drop of previous methods or our increase in performance does not come from solving those issues, but from a third confounding factor. Even though we have tested our method across different settings and benchmarks, different modalities (e.g. language) and scenarios (e.g. regression, segmentation...etc.) are to be verified.

On the other hand, a notable advantage of our method is its *efficiency* that allows users who do not have access to sufficient compute to train or fully finetune large models themselves or would like to extend the capabilities of their already trained models but face data sharing constraints or other challenges in adapting and deploying large foundation models. This approach is useful across various scenarios such as recycling trained models, patching model weaknesses, and supporting collaborative efforts to improve existing models.