Task Alignment: A Simple Proxy for Practical Model Merging Across Diverse Vision Tasks

Pau de Jorge, César Roberto de Souza, Björn Michele, Mert Bülent Sarıyıldız, Philippe Weinzaepfel, Florent Perronnin, Diane Larlus, and Yannis Kalantidis

NAVER LABS Europe

Abstract. Efficiently merging several models fine-tuned for different tasks, but stemming from the same pretrained base model, is of great practical interest. Despite extensive prior work, most evaluations of model merging in computer vision are restricted to image classification using CLIP, where different classification datasets define different tasks. In this work, our goal is to make model merging more practical and show its relevance on challenging scenarios beyond this specific setting. In most vision scenarios, different tasks rely on *trainable* and usually *heterogeneous* decoders. Differently from previous studies with frozen decoders, where merged models can be evaluated right away, the non-trivial cost of decoder training renders hyperparameter selection based on downstream performance impractical. To address this, we introduce the *task alignment proxy*, and show how it can be used to speed up hyperparameter selection by orders of magnitude while retaining performance. Equipped with the task alignment proxy, we extend the applicability of model merging to multi-task vision models beyond CLIP-based classification.

Keywords: Model Merging · Task Alignment Proxy · Merging LiDAR Models · Heterogeneous Vision Tasks

1 Introduction

Vision foundation models have emerged as generic encoders that support a wide range of computer vision tasks [1, 22, 26–29], typically paired with task-specific decoders. Fine-tuning such models for individual tasks improves performance, but deploying multiple fine-tuned encoders is costly: each additional model increases memory and compute requirements. This is particularly problematic in resource-constrained settings such as robotics, where localization, human detection, and semantic or geometric scene understanding must run simultaneously on limited hardware. This motivates a practical question: Can we efficiently merge multiple task-specialized encoders, fine-tuned from the same base, into a single encoder that retains performance while reducing deployment cost?

Recent works on model merging have brought us closer to answering that question. Several approaches [14, 17, 19, 23, 32] demonstrate that models fine-tuned from a shared initialization can be combined without additional training via simple arithmetic in the weight space. However, in computer vision, empirical evaluations have largely focused on CLIP-based image classification, where

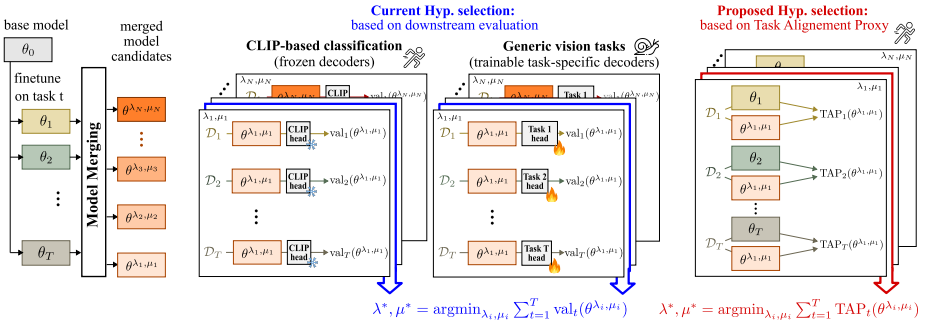

Fig. 1: (Left) Model merging methods produce a large number of candidates which depend on hyperparameters (λ, μ). (Middle) SOTA methods select hyperparameters based on validation performance. Evaluating and ranking all candidates might be feasible when the CLIP's text encoder is used as the frozen head (standard benchmarks) but becomes infeasible for more complex sets of tasks that require fine-tuning multiple task-specific decoders of different sizes, *for each hyperparameter candidate*. (Right) Our proposed Task Alignment Proxy (TAP) can be used to select hyperparameters at a fraction of the cost while maintaining performance (see also Fig. 2).

different label sets define different tasks. Because CLIP's text encoder remains frozen, switching tasks only requires modifying class embeddings. This makes merged models easy and fast to evaluate, as no retraining is required.

Alternatively, some works studied merging on NLP tasks, where models have converged to being "decoder-only", also allowing direct evaluation.

While encouraging, these settings obscure a fundamental challenge: in most vision applications, each task relies on a different trainable decoder. In such scenario, merging can only be performed on part of the model, typically the shared encoder, while task-specific decoders must be fine-tuned after merging. This brings additional challenges and is the main focus of this paper, illustrated in Fig. 1.

Relying on trainable decoders fundamentally changes the evaluation protocol. Unlike CLIP-based classification, where merged models can be assessed immediately, evaluating a merged model in this setting requires decoder training, a process that can

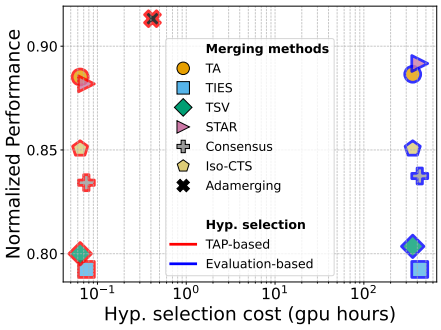

Fig. 2: Task Alignment Proxy (TAP) drastically reduces hyperparameter selection costs when merging models for heterogeneous 2D and 3D vision tasks (see Sec. 4.1). In this case, downstream evaluation is costly due to task-dependent decoder training, and hyperparameter search becomes impractical (blue border). Hyperparameter selection with TAP reduces costs by 3 orders of magnitude (red border).

be costly. As a result, exhaustive hyperparameter search over merging strategies becomes impractical, severely limiting applicability in most vision settings.

To address this bottleneck, we introduce the Task Alignment Proxy (TAP), which efficiently evaluates merging candidates. TAP measures the alignment of feature spaces between models, enabling us to estimate merge compatibility without repeatedly fine-tuning decoders. By decoupling hyperparameter selection from expensive decoder training loops, TAP reduces the cost of model selection by orders of magnitude (models outlined in blue vs in red in Fig. 2).

We demonstrate that TAP enables model merging well beyond classification tasks. First, we show that in standard classification benchmarks, TAP accurately selects optimal merging configurations using only a few dozen unlabeled images (Sec. 4.3). Second, we present, to the best of our knowledge, the first successful application of model merging to LiDAR. We focus on LiDAR point cloud semantic segmentation, with a single encoder being applied to diverse scenarios, spanning different LiDAR sensors, environments, and annotated with different label sets. This demonstrates that merging generalizes beyond RGB (Sec. 4.4). Third, we apply TAP to the DUNE benchmark [28] that considers a challenging multi-task scenario with heterogeneous (*i.e.* very diverse) tasks, namely semantic segmentation, depth estimation, 3D pose relocalization, and human mesh recovery. In Sec. 4.5 we show TAP-guided merging can be applied even in this scenario to further boost the performance of a model obtained with large-scale multi-teacher distillation. Notably, since computing TAP does not require task-specific decoders, the same hyperparameter selection criterion (TAP) can be applied universally to merge models across CLIP classification, LiDAR segmentation, and heterogeneous 2D/3D vision settings; regardless of the set of downstream tasks.

2 Background and related work

Preliminaries and notation. Let $\theta_0 \in \mathbb{R}^d$ denote the parameters of a pre-trained visual encoder, referred to as the *base model*. We assume that this base model is fine-tuned end-to-end independently for each task t in a set of T downstream tasks. Let θ_t represent the encoder parameters after fine-tuning for task t, defined as $\theta_t = \arg\min_\theta \mathcal{L}_t(\theta)$ where $\mathcal{L}_t(\theta)$ is the training loss for task t.

Following [14], we define the *task vector* for task t as $\tau_t = \theta_t - \theta_0$. The goal of model merging is to find a single set of encoder parameters that jointly minimizes the losses across all tasks, *i.e.* $\theta^* = \text{argmin}_\theta \sum_{t=1}^{T} \mathcal{L}_t(\theta)$. *Arithmetic-based merging* methods aim to efficiently approximate θ^* by combining task vectors in the parameter space. Most merging methods can be framed as a variation of the following weighted sum:

$$\theta_{\text{merged}}(\lambda, \mu) = \theta_0 + \sum_{t=1}^{T} \lambda_t \odot \phi(\tau_t; \mu), \tag{1}$$

where $\lambda_t \in \mathbb{R}^d$ is a vector of task-specific weights, $\phi : \mathbb{R}^d \to \mathbb{R}^d$ is a transformation parametrized by hyperparameters μ and $\odot$ is the Hadamard product. (App.

B discusses how Eq. (1) applies to different merging methods). Let λ denote the set of all λ_t parameters. When there is no ambiguity, we denote $\theta_{\text{merged}}(\lambda, \mu)$ as $\theta(\lambda, \mu)$ or just θ.

Model merging methods. The simplest approach is **model averaging** [7, 16, 20, 34], where $\lambda_t = [1/T, ..., 1/T]$ and ϕ is the identity function. **Task Arithmetic** [14] extends weight averaging by linearly interpolating between the average of fine-tuned models and the base model. In **TIES** [36], ϕ applies weight pruning to task vectors, retaining the values with highest magnitudes and consistent signs, in order to reduce task interference. **Breadcrumbs** [8] also applies pruning, but removes both high- and low-magnitude weights at the layer level to eliminate outliers. **Consensus** [33] prunes weights per task, then constructs a consensus mask by retaining weights that appear in more than one task-specific mask. **Lines** [32] uses per-layer weights where coefficients increase linearly across layers, based on the intuition that early layers capture general features (thus closer to the base model) whereas later layers are more task-specific.

More recently, several methods apply SVD to per-layer weights, when weights are matrix-shaped, defaulting to Task Arithmetic otherwise. **STAR** [17] performs SVD truncation on each weight matrix of a task vector, retaining the top singular values to preserve a specified fraction of the energy, and rescales the remaining singular values to match the initial energy. **TSV** [12] applies SVD twice: first independently to each task vector (as in STAR), and then again after concatenating the decomposed matrices across tasks to enforce orthogonality. **Iso** [19] applies a similar SVD-based strategy but balances per-task SVD with the SVD of the sum of task vectors with a tunable hyperparameter. Moreover, it imposes "isotropy" by fixing all singular values to the average.

Finding optimal merging hyperparameters (lambdas, pruning and SVD truncation thresholds), although efficient for decoder-only NLP models or CLIP-based classification with frozen text embeddings, becomes prohibitively expensive when performing merging tasks that require fine-tuning the decoders for evaluation, as is common in computer vision (Fig. 1).

Inspired by test-time adaptation, **AdaMerging** [37] directly optimizes merging coefficients via entropy minimization. However, it is only applicable to CLIP-like settings with frozen classification heads. In Sec. 3.2, we extend it with our proposed TAP to *any* set of tasks, but the optimization requires storing all fine-tuned models in memory, which is very demanding as the number of tasks increases. **AdaMMS** [9] was designed to merge LLMs and selects merging weights based on generation consistency between models obtained with adjacent hyperparameter values. Such requirements restrict it to methods with a single scalar hyperparameter, rendering it unsuitable for our setting.

In this work, we introduce the task alignment proxy and show it can *accelerate hyperparameter search by several orders of magnitude* for any set of tasks.

Alternative model merging applications. Recently, some studies have explored merging in the context of continual learning. Dziadzio *et al.* [10] use CLIP-based tasks in a continual learning setting where training data and inference objectives evolve over time. Sokar *et al.* [31] investigate model merging

to mitigate catastrophic forgetting when fine-tuning Vision Language models on a continual VQA benchmark. These studies find that simple methods (such as task arithmetic) perform surprisingly well, often matching or outperforming more complex approaches. Although our setup differs significantly, this is consistent with our observations in Sec. 4.4 and Sec. 4.5.

3 Task alignment

As discussed before, most model merging evaluations in vision have focused on CLIP-based zero-shot image classification. In this section, we ask: *Can we apply model merging to any set of vision tasks?* We first discuss the new challenges when merging models with trainable decoders (see Sec. 3.1). This motivates our generic and efficient performance proxy to guide hyperparameter selection in more complex scenarios (Sec. 3.2).

3.1 Merging with trainable decoders

The dominant evaluation setup for model merging in vision builds on CLIP's zero-shot classification capabilities (see Fig. 1, left): the visual encoder is fine-tuned separately for each task, while the frozen text encoder serves as a shared classification head (*i.e.* a very light-weight decoder) by embedding class names or descriptions into a compatible feature space [15, 26]. As a result, the visual encoder effectively constitutes the *entire model* used to solve all tasks. Hence, in this setting, evaluation of merged models is cheap and requires only forward inference, making exhaustive search over merging hyperparameters feasible.

However, many practical multi-task vision systems produce outputs in different spaces (*e.g.* semantic labels, depth maps, 3D poses, meshes) and therefore rely on a task-specific decoder attached to an encoder. In this setting, only encoder parameters can be merged, while decoders remain task-specific and must be fine-tuned after merging. Unlike CLIP-based classification, where merged encoders can be evaluated immediately by swapping label embeddings, here, each candidate encoder to merge $\theta(\lambda, \mu)$ must be paired with a newly fine-tuned decoder to assess performance. Fine-tuning decoders often requires hours, if not days, of compute per configuration, rendering exhaustive hyperparameter search intractable. This evaluation bottleneck severely limits the practical applicability of model merging across multi-task vision settings.

One way to bring this setup closer to the CLIP-based formulation would be to keep decoders frozen during hyperparameter selection. However, this still requires the use of multiple, potentially expensive and complex decoders and oftentimes leads to suboptimal performance (see App. F).

Another possible solution is to use *proxy* signals to efficiently estimate downstream performance. Proxy-based approaches AdaMerging [37] and AdaMMS [9] follow this direction. However, AdaMerging relies on entropy minimization, which is only applicable to categorical tasks such as classification, while AdaMMS is

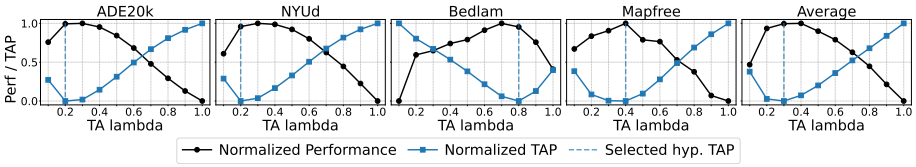

Fig. 3: Normalized TAP *vs.* performance for different values of the merging coefficient λ when using Task Arithmetic (TA) on the heterogeneous task setting, see Sec. 4.1. The selected value is indicated by a dashed line.

specifically designed for generative models and assumes one can compare generated outputs. These limitations motivate us to introduce a generic proxy measure, applicable across *any* set of tasks, enabling both efficient hyperparameter selection and learning-based merging.

3.2 Task Alignment Proxy

Motivated by the prohibitive cost of hyperparameter tuning when tasks require decoder training, we propose a generic proxy to approximate the performance drop incurred by swapping, for a given task t, the fine-tuned parameters θ_t with the merged model θ_{merged}. We refer to this as the **Task Alignment Proxy** (**TAP**). The intuition behind TAP is the following: if the features of the fine-tuned encoder θ_t and the merged encoder θ_{merged} are similar for images of task t, then downstream performance should also be similar for this task. Computing TAP only requires access to *unlabeled* data from each downstream task and is both task-agnostic and computationally efficient, as it only involves inference.

Formally, let $D_1, \ldots, D_T$ denote unlabeled training image sets for each task. Let $f(x;\theta) : \mathcal{X} \to \mathcal{F}$ denote the image encoder parameterized by θ, where $\mathcal{X}$ is the input space and $\mathcal{F}$ the output feature space, and let $x \in \mathcal{X}$. Given a dissimilarity function $d : \mathcal{F} \times \mathcal{F} \to \mathbb{R}^+$, we define the *task alignment proxy* for task t as the average dissimilarity between features produced by the merged model and the task-specific fine-tuned model over N samples $x_i \sim D_t$:

$$\text{TAP}_t(\theta) = \frac{1}{N} \sum_{i=1}^{N} d\big(f(x_i;\theta), f(x_i;\theta_t)\big), \quad x_i \sim D_t \tag{2}$$

To obtain a single score across all tasks, we average over the tasks and compute $\text{TAP}(\theta) = \sum_{t=1}^{T} \text{TAP}_t(\theta)/T$.

Default settings. We use the ℓ_2 distance for d and 128 samples for N. Yet, in Sec. 5.1 we show that TAP is robust to the choice of dissimilarity function d and the number of samples used N.

Combining TAP with existing merging methods. In the following sections, we combine TAP with several merging methods to select hyperparameters. We denote such combinations with an additional "$_{\text{w/TAP}}$" (*e.g.* $\text{TSV}_{\text{w/TAP}}$). This is done by selecting the λ (and whenever relevant μ) parameters that minimize

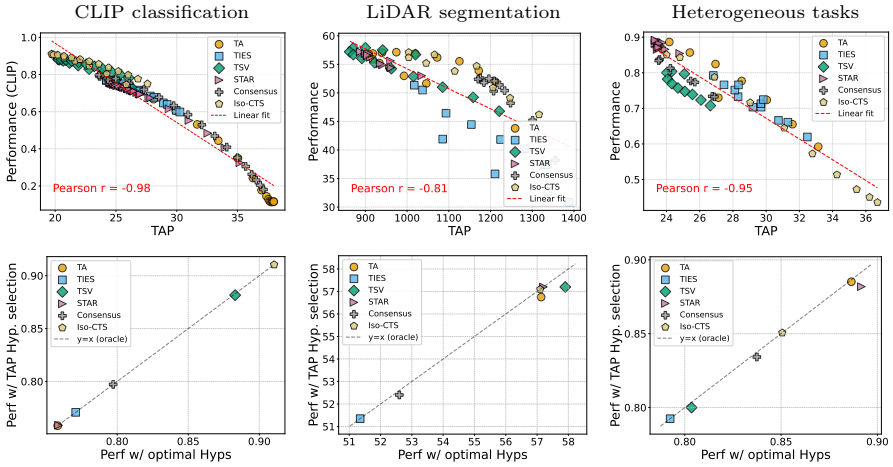

Fig. 4: Top row: Correlation between Task Alignment Proxy (TAP) and Performance on the validation sets when merging models for CLIP classification with a ViT-L: 20 task setting (left), 3D segmentation with different LiDAR sensors (middle) and heterogeneous vision tasks (right). **Bottom row:** Performance with TAP-selected hyperparameters *vs*. optimal hyperparameters (with costly downstream evaluation) for the same settings. We observe a strong correlation between TAP and performance which leads to almost optimal hyperparameters *without any downstream task evaluation*.

TAP, *i.e.* $\lambda^*, \mu^* = \arg\min_{\lambda,\mu} \text{TAP}(\theta_{\text{merged}}(\lambda, \mu))$, where $\theta_{\text{merged}}(\lambda, \mu)$ is given by Eq. (1). We specify the standard hyperparameter selection based on downstream performance evaluation with "$_{\text{w/Eval}}$"; see Fig. 1 for an illustration.

While most methods rely on hyperparameter search for method selection, AdaMerging finds the optimal λ via entropy minimization of the merged model on the different task datasets. As discussed in Sec. 2, this cannot be directly applied to non-categorical tasks, *i.e.* beyond classification. To extend AdaMerging to any set of downstream tasks, we replace the entropy loss with TAP, which can also be treated as a loss, and we name this version AdaMerging$_{\text{w/TAP}}$.

4 Experiments

4.1 Experimental protocol

We perform model merging in three main settings: CLIP-based classification on multiple datasets, LiDAR-based segmentation with different sensors and data domains, and a combination of heterogeneous vision tasks introduced in [28].

CLIP-based classification. Initially introduced in [14] and later re-used and extended in many works, [12, 19, 32, 33, 36, 37] to name a few, this has been the de-facto evaluation protocol for model merging in vision. We follow the evaluation code and checkpoints from [12] and evaluate merging across three different architectures (ViT-B with patch size 32, ViT-B with patch size 16 and

ViT-L with patch size 14) and incremental sets of tasks with 8, 14 and 20 datasets respectively. For more details we refer the reader to [12].

We consider four different 3D LiDAR semantic segmentation datasets: nuScenes [6, 11], SemanticKITTI [4, 13], Panda64 [35] and PandaGT [35]; captured with different LiDAR sensors (*e.g.* varying number of beams, resolution or field of view) and different label taxonomy. We use the WaffleIron-768 (WI) architecture [24] as an encoder, pretrained with ScaLR [25], a multi-modal image-to-LiDAR distillation, and use linear decoders.

Regarding per-task fine-tuning, we use the models from [25] as they are the highest performing models publicly available at the time of writing. Once fine-tuned encoders are merged, an independent decoder is trained for each task on top of the frozen encoder. After decoder training, we evaluate the performance on each task following the protocol in [25].

Heterogeneous vision tasks. We adopt the evaluation protocol introduced in DUNE [28] to assess model merging across a diverse set of 2D and 3D vision tasks; we refer to it as the **DUNE benchmark**. Specifically, we evaluate on semantic segmentation using ADE20k [39], depth estimation on NYUd [30], 3D human mesh recovery on Bedlam [5], and pose relocalization on Niantic's MapFree dataset [2]. As a base model we use DUNE (ViT-Base/14), given its demonstrated performance on such a broad range of tasks, precisely the setting we want to study. We also start from DINOv2 as a base model in that particular setting and report results in Sec. 5.2.

We fine-tune the base model end-to-end for each task, resulting in four task-specific encoders. After merging, we freeze the encoder and fine-tune only the corresponding task-specific decoders. For the decoders, we use a linear head for semantic segmentation (ADE20k) and a DPT model for depth estimation (NYUd), following [22]. For human mesh recovery on Bedlam, we adopt the two-headed architecture from MultiHMR [3]. For visual relocalization on MapFree, we use a ViT-Large decoder as proposed in MASt3R [18].

We follow the evaluation of DUNE, so results reported in Tab. 3 and Tab. 4 are directly comparable to those in [28]. However, due to the high cost of model evaluation, we introduce an *ablation* protocol which is used when evaluating different hyperparameters per model in Fig. 4. In this protocol, we reduce the number of fine-tuning iterations for task-specific decoders, and, in the case of MapFree, we also limit the number of training samples. Moreover, we use a linear head (instead of DPT) for depth estimation. To avoid bias towards the original validation sets, evaluations are conducted on custom held-out sets. Further details on our experimental settings can be found in App. A.

4.2 Is TAP a good proxy for performance?

In this section we show that TAP strongly correlates with downstream performance and that TAP selection is comparable to that of using an oracle (*i.e.* selecting parameters based on full downstream evaluation).

In Fig. 3 we merge the different models in the heterogeneous task setting with Task Arithmetic (TA). We consider different hyperparameter values and

Table 1: Comparing hyp. selection strategies on CLIP benchmark. Test performance of models selected via standard hyp. selection ("w/Eval") or using TAP on a subset of 128 unlabeled images per task ("w/TAP"). Both lead to comparable performance across merging methods and settings. We report ViT-B-16 in App. C. *Fine-tuned* refers to an upper-bound for which separate expert models are trained per task.

	ViT-B-32			ViT-L-14			Average
	T=8	T=14	T=20	T=8	T=14	T=20	
Zero-shot	48.3	57.2	56.1	64.7	68.2	65.2	60.0
Fine-tuned	92.8	90.9	91.3	95.8	94.3	94.7	93.3
$\text{TA}_{\text{w/Eval}}$	71.1	65.2	62.6	84.9	79.4	76.0	72.7
$\text{TA}_{\text{w/TAP}}$	71.1	65.0	62.1	84.9	79.7	76.0	72.5
$\text{STAR}_{\text{w/Eval}}$	71.4	65.4	62.9	84.9	79.7	76.1	72.9
$\text{STAR}_{\text{w/TAP}}$	70.9	65.3	63.0	84.9	79.7	76.1	72.8
$\text{TIES}_{\text{w/Eval}}$	74.8	67.8	64.8	86.9	80.2	77.1	74.9
$\text{TIES}_{\text{w/TAP}}$	74.5	67.8	64.8	87.0	80.2	77.1	74.9
$\text{Consensus}_{\text{w/Eval}}$	74.9	70.3	66.9	86.3	82.3	79.6	76.2
$\text{Consensus}_{\text{w/TAP}}$	74.7	70.3	66.9	86.3	82.3	79.6	76.2
$\text{TSV}_{\text{w/Eval}}$	85.8	80.0	76.9	93.0	89.2	87.5	85.2
$\text{TSV}_{\text{w/TAP}}$	85.5	80.0	76.9	93.0	89.1	87.6	85.1
$\text{Iso-CTS}_{\text{w/Eval}}$	86.5	81.6	78.2	94.8	91.0	90.2	86.9
$\text{Iso-CTS}_{\text{w/TAP}}$	85.8	81.6	78.1	94.7	91.0	90.2	86.8
$\text{AdaMerging}_{\text{w/TAP}}$	84.2	75.0	71.5	91.4	87.0	84.3	81.1
$\text{AdaMerging}_{\text{w/Entropy}}$	80.7	75.4	72.4	91.7	87.7	87.1	81.5

show TAP (in blue) *vs*. the actual performance after the costly decoder fine-tuning per-task (in black), for each task (first four plots) and averaged across tasks (right-most plot). In both cases, *i.e*. per-task TAP and average TAP, we observe a strong correlation between performance and TAP.

Fig. 4 (top) shows the average normalized performance *vs*. the average TAP for different merging methods and hyperparameter settings. Fig. 4 (bottom) shows the performance of the models selected with TAP *vs*. the actual best model for each method found via exhaustive evaluation of the hyperparameter configurations. We report these results for the three different settings described in Sec. 4.1. As already noted in Fig. 3, we observe a strong negative correlation between TAP and performance. Moreover, on the bottom plots we observe that all methods are close to the $y = x$ line; that corresponds to always selecting hyperparameters leading to the best performance. Hence, TAP is indeed a good proxy that can be used for hyperparameter selection across a variety of settings in computer vision. In the following sections we present more detailed results for each merging setting.

Table 2: Merging LiDAR models. Results after merging different 3D segmentation models with TAP-based hyperparameter selection. We successfully apply several methods, significantly improving over the pretrained baseline. We report the mIoU and the normalized performance w.r.t. the fine-tuned models.

	nuScenes mIoU (↑)	Sem.KITTI mIoU (↑)	Panda64 mIoU (↑)	PandaGT mIoU (↑)	Normalized Performance
Pretrained	68.4	55.6	37.0	35.4	0.841
$\text{TA}_{\text{w/TAP}}$	73.7	63.5	47.0	42.9	0.972
$\text{STAR}_{\text{w/TAP}}$	74.1	63.7	**47.5**	43.3	0.979
$\text{TIES}_{\text{w/TAP}}$	69.0	56.0	41.9	38.5	0.879
$\text{Consensus}_{\text{w/TAP}}$	70.1	59.5	41.3	38.8	0.898
$\text{TSV}_{\text{w/TAP}}$	74.7	63.5	47.2	43.4	0.979
$\text{Iso-CTS}_{\text{w/TAP}}$	**74.8**	62.4	47.0	44.6	0.979
$\text{AdaMerging}_{\text{w/TAP}}$	73.1	**64.4**	46.6	**46.1**	**0.985**

4.3 CLIP-based image classification

In this section we compare the performance of evaluation *vs*. TAP-based hyperparameter selection in the standard CLIP-based image classification benchmark. In Tab. 1 we report the test performance for different merging methods when selecting hyperparameters following one of the two strategies, across different architectures and task sets. We observe that TAP-based selection leads to comparable performance to exhaustive hyperparameter selection. We even see TAP selection achieving higher test performance (by a small margin) in some cases. We attribute these small variations to validation-test set noise.

For AdaMerging, there seems to be a trend where optimizing TAP leads to better performance than with the original entropy-based formulation as we increase the number of tasks and architecture size.

It is also worth noting, that, although exhaustive hyperparameter evaluation in this setting is feasible thanks to the use of frozen CLIP decoders, computing TAP does not require labels. Moreover, in Fig. 5 we show that TAP is very stable even when using very few images. This shows that TAP-based selection would still be beneficial in cases where labels are not available, or only a few samples per-task are kept, *e.g.* due to privacy or memory constraints.

4.4 Merging LiDAR models

To illustrate the broad applicability of model merging in computer vision, we merge different models fine-tuned to perform LiDAR-based semantic segmentation. When using LiDAR inputs, besides potential domain gaps due to geolocation or weather conditions (as one would encounter in RGB-based vision), changes in the LiDAR sensors used to acquire the data often lead to stark drops in performance [21, 38]. Hence, it is quite common to start from a strong pretrained model and fine-tune it for the relevant sensor [25]. However, if inputs

Table 3: Improving heterogeneous vision models via merging. Several models merged with TAP-based hyperparameter selection outperform the pretrained model DUNE. † denotes our reproduction of the public DUNE model results under the exact same evaluation setting.

	ADE20k mIoU (↑)	NYUd rmse (↓)	Bedlam pa-pve (↓)	MapFree AUC (↑)	Normalized Performance
DUNE† (Pretrained)	45.6	0.330	62.5	94.7	0.929
$TA_{w/TAP}$	46.6	0.315	59.7	95.2	0.956
$STAR_{w/TAP}$	46.6	0.312	**59.3**	**95.7**	0.962
$TIES_{w/TAP}$	46.3	0.324	63.3	94.7	0.934
$Consensus_{w/TAP}$	46.7	0.324	65.7	94.7	0.928
$TSV_{w/TAP}$	45.6	0.321	62.9	95.5	0.936
$Iso\text{-}CTS_{w/TAP}$	48.2	0.326	65.5	94.4	0.934
$AdaMerging_{w/TAP}$	**49.3**	**0.300**	62.9	95.2	**0.971**

from different sensors need to be processed simultaneously, *e.g.* on a robot with multiple LiDAR sensors, one would need to deploy several models at once.

In Tab. 2, we show that model merging (with TAP-based hyperparameter selection) can be used to effectively merge different encoders fine-tuned on different datasets/sensors, significantly improving the performance over the pretrained model. To the best of our knowledge, this is the first work to show that model merging can be successfully applied to LiDAR models. To avoid redundancy, we only report numbers with TAP-based hyperparameter selection, but a table with evaluation-based selection can be found in App. D.

Similar to the findings in Fig. 2 for heterogeneous tasks, TAP accelerates hyperparameter selection in this setting significantly. While decoder training requires more than 30 gpu hours for each evaluated hyperparameter, computing TAP requires approximately only 7 minutes, *accelerating hyperparameter selection by two orders of magnitude*. For further details see App. D.

4.5 Merging models for heterogeneous vision tasks

In this section we study the effectiveness of model merging when downstream tasks (and decoders) are significantly different. We would like to note that, given the task diversity, it is not obvious that such models could be merged effectively, even with expensive merging techniques such as distillation, as discussed in [28]. In Tab. 3 we apply different model merging methods with TAP-based hyperparameter selection and show they can indeed improve the performance of a state-of-the-art multi-task vision model, *i.e.* DUNE [28], via further fine-tuning (on top of DUNE) and merging. To the best of our knowledge, no prior work has evaluated model merging on a set of heterogeneous vision tasks spanning both 2D and 3D. This is the setting where TAP speeds up hyperparameter selection the most, and this is mainly due to the expensive decoder training protocol of the MASt3R decoder [18] (used for MapFree). *As illustrated in Fig. 2, we*

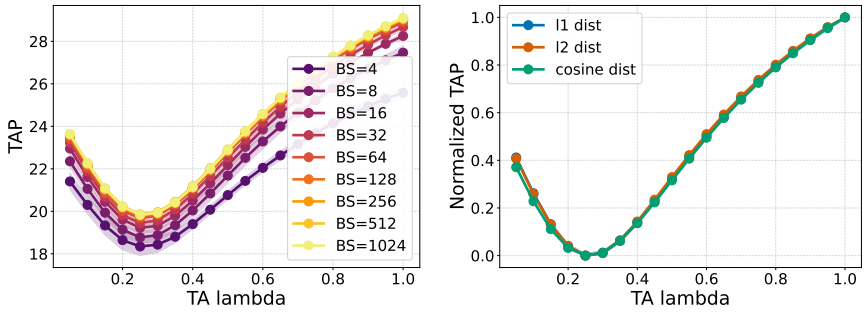

Fig. 5: Left: TAP *vs*. Batch size. When increasing the number of samples per task to compute TAP, the score becomes more stable. However, the selected model remains the same even when computing TAP on very few images. **Right: TAP with different distance metrics.** TAP is very robust to the distance metric used to compare fine-tuned and merged model features.

can reduce hyperparameter search by three orders of magnitude. Due to its prohibitive cost, we do not perform evaluation-based hyperparameter selection in this setting using the original decoder training protocol from [28].

5 Analyses

5.1 Task alignment proxy ablations

We ablate the design choices of TAP, in particular, the distance to compare features and the number of samples used to compute TAP. In Fig. 5 (left) we plot the TAP across different hyperparameter values when computed with different number of images. We observe that TAP is very robust to the images being used (shaded area corresponding to standard deviation across seeds is small). Moreover, it is highly data efficient, resulting in a significant reduction in computation. This also makes TAP especially suitable when per-task data is scarce. In Fig. 5 (right) we plot TAP across different hyperparameter values when measuring feature distance with ℓ_1, ℓ_2 or cosine distance (*i.e.* 1 − cosine similarity). We see TAP is very robust to the choice of distance between features.

5.2 Can merging replace distillation?

In Sec. 4.5 we show that, in combination with TAP, model merging can be efficiently applied to a set of heterogeneous tasks. However, DUNE was obtained by distilling already strong teachers in the respective downstream tasks. If such teachers were not available, can model merging be competitive with distillation? In this section we introduce an analysis experiment to address this question by

Table 4: Teacher distillation *vs*. fine-tuning + merging. We report results when merging models, initialized from DINOv2 (one of DUNE teachers), and fine-tuned on the DUNE benchmark. We also compare with the DUNE distilled model as a baseline.

	ADE20k mIoU (↑)	NYUd rmse (↓)	Bedlam pa-pve (↓)	MapFree AUC (↑)	Normalized Performance
DUNE†	45.6	0.330	62.5	94.7	0.847
DINOv2 (Pretrained)	48.3	0.303	63.3	92.1	0.871
$\text{TA}_{\text{w/TAP}}$	50.2	0.303	62.0	94.5	0.889
$\text{STAR}_{\text{w/TAP}}$	46.8	0.310	65.0	**95.5**	0.863
$\text{TIES}_{\text{w/TAP}}$	49.3	0.298	62.2	94.1	0.887
$\text{Consensus}_{\text{w/TAP}}$	49.9	0.309	**57.4**	93.1	0.892
$\text{TSV}_{\text{w/TAP}}$	50.3	0.296	63.4	94.8	0.893
$\text{Iso-CTS}_{\text{w/TAP}}$	50.9	**0.276**	61.5	93.3	**0.915**
$\text{AdaMerging}_{\text{w/TAP}}$	**52.8**	0.294	60.9	93.4	0.909

starting from a strong generalist model, *i.e*. DINOv2, and merging the corresponding fine-tuned models on the target tasks. The fine-tuning, merging methods and evaluation protocols are the same as used in Tab. 3. However, instead of having DUNE as the base model, in Tab. 4 we use DINOv2.

Since all fine-tuned models are initialized from DINOv2, most merged models perform well on ADE20k and NYUd, *i.e*. tasks where DINOv2 is already strong. Several methods significantly outperform pretrained DINOv2 and even the distilled DUNE model. However, while some merged models surpass DUNE in normalized (average) performance, a trade-off exists between tasks (especially Bedlam and MapFree), and no single model surpasses DUNE on all of them. We hypothesize that human mesh recovery and 3D scene reconstruction are further out-of-domain for DINOv2 than for DUNE, which was distilled on expert teachers for those tasks. These results are highly encouraging for future model merging using generalist backbones like DINOv2.

5.3 Task vector norm analysis

To understand the differences between the standard CLIP benchmark and other studied settings (*i.e*. LiDAR and DUNE), we analyze the task vectors directly.

Task vector imbalance. Fig. 6 (top) shows highly imbalanced task vector norms in the LiDAR and heterogeneous settings (DUNE/DINOv2 encoders), likely due to diverse tasks and fine-tuning protocols. Conversely, CLIP benchmark task vectors, which use a fixed fine-tuning protocol, have comparable magnitudes and significantly lower norms (see App. G). Fig. 6 (bottom) shows that this norm imbalance biases the average task vector toward higher-norm tasks, creating an asymmetry which we hypothesize may hinder model merging.

Task subset experiments. To further explore the impact of task vector norms, we merge task subsets from the DUNE benchmark using four combinations:

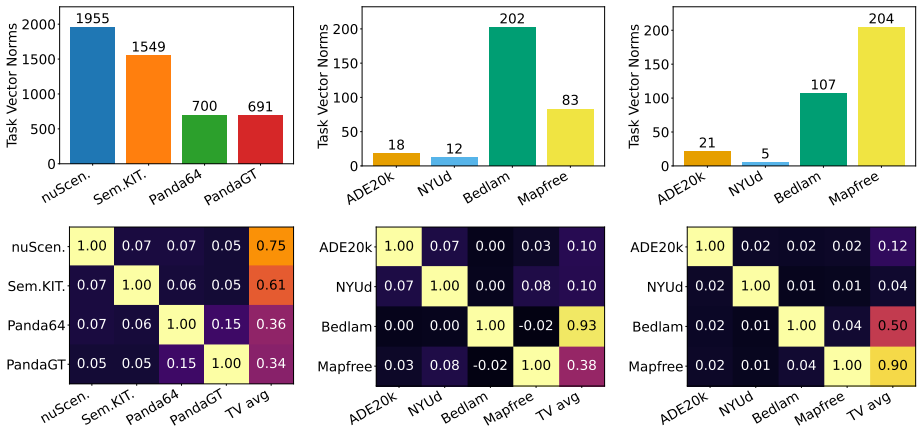

Fig. 6: Top row: Task Vector Norms for the LiDAR benchmark (left) and for the DUNE benchmark when finetuning models from DUNE (middle) or DINOv2 (right). **Bottom row:** Cosine similarity between task vectors and the average task vector. We observe that for all settings, task vector norms are rather large and unbalanced, strongly biasing the average task vector towards the task with high norm.

{ADE20k, NYUd}, {ADE20k, Bedlam}, {NYUd, Bedlam}, and {ADE20k, NYUd, Bedlam}, excluding MapFree due to its high evaluation cost. Results are reported in Fig. 7. All methods achieve their best normalized performance when merging {ADE20k, NYUd}, the tasks with the smallest and most similar norms. Performance consistently degrades when Bedlam is included, signaling increased "merging difficulty". Adding Bedlam also completely reverses the relative ranking of methods, highlighting that some approaches may be more sensitive to large and unbalanced norms.

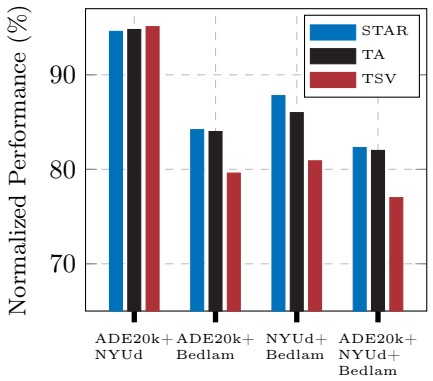

Fig. 7: Performance *vs.* task subsets. When task vector norms are small and balanced (left-most), merging methods perform significantly better than when some task vectors are much larger than others.

Normalizing task vectors. If all tasks are to be weighted equally, a natural baseline is to normalize all task vectors before merging. We call this baseline **NormAvg**, where each task vector is rescaled to the smallest norm before averaging. This will naturally produce a more balanced norm and cosine similarity across tasks. However, in App. A.4 we show this does not lead to a promising performance. Thus, although a baseline, naively scaling task vectors does not seem to be an effective strategy.

6 Conclusion

In this paper, we explore model merging in vision beyond the widespread CLIP-based classification benchmark. Because most vision tasks rely on task-specific trainable decoders, downstream performance-based hyperparameter selection becomes impractical. To address this, we propose *Task Alignment*, a simple proxy of downstream performance that selects merging hyperparameters at a fraction of the cost. On top of being very cost efficient, it requires no access to task-decoders or labels, making it widely applicable: *any* merging method can use it for *any* set of tasks, and it can even serve as a loss function to generalize some learning-based methods like AdaMerging. Our extensive experiments show that TAP-based hyperparameter selection leads to near-optimal model selection on a diverse set of model merging benchmarks, including CLIP classification. Moreover, task vectors in our newly introduced merging settings are significantly larger and more unbalanced than in the CLIP setting, testing the limits of existing merging methods. Explicitly addressing large, unbalanced task vectors in model merging is a promising direction for future work, and our preliminary findings provide initial guidance toward this goal.

References

1. Alayrac, J.B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al.: Flamingo: a visual language model for few-shot learning. In: Proc. NeurIPS (2022)
2. Arnold, E., Wynn, J., Vicente, S., Garcia-Hernando, G., Monszpart, Á., Prisacariu, V.A., Turmukhambetov, D., Brachmann, E.: Map-free visual relocalization: Metric pose relative to a single image. In: Proc. ECCV (2022)
3. Baradel, F., Armando, M., Galaaoui, S., Brégier, R., Weinzaepfel, P., Rogez, G., Lucas, T.: Multi-HMR: Multi-person whole-body human mesh recovery in a single shot. In: Proc. ECCV (2024)
4. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In: Proc. ICCV (2019)
5. Black, M.J., Patel, P., Tesch, J., Yang, J.: BEDLAM: A synthetic dataset of bodies exhibiting detailed lifelike animated motion. In: Proc. CVPR (2023)
6. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving. In: Proc. CVPR (2020)
7. Choi, J., Kim, D., Lee, C., Hong, S.: Revisiting weight averaging for model merging. arXiv preprint arXiv:2412.12153 (2024)
8. Davari, M., Belilovsky, E.: Model breadcrumbs: Scaling multi-task model merging with sparse masks. In: Proc. ECCV (2024)
9. Du, Y., Wang, X., Chen, C., Ye, J., Wang, Y., Li, P., Yan, M., Zhang, J., Huang, F., Sui, Z., et al.: Adamms: Model merging for heterogeneous multimodal large language models with unsupervised coefficient optimization. In: Proc. CVPR (2025)
10. Dziadzio, S., Udandarao, V., Roth, K., Prabhu, A., Akata, Z., Albanie, S., Bethge, M.: How to merge your multimodal models over time? In: Proc. CVPR (2025)

11. Fong, W.K., Mohan, R., Hurtado, J.V., Zhou, L., Caesar, H., Beijbom, O., Valada, A.: Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking. RA-L (2021)
12. Gargiulo, A.A., Crisostomi, D., Bucarelli, M.S., Scardapane, S., Silvestri, F., Rodola, E.: Task singular vectors: Reducing task interference in model merging. In: Proc. CVPR (2025)
13. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: Proc. CVPR (2012)
14. Ilharco, G., Ribeiro, M.T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., Farhadi, A.: Editing models with task arithmetic. In: Proc. ICLR (2023)
15. Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., Schmidt, L.: OpenCLIP. `https://github.com/mlfoundations/open_clip` (2021), version 0.1
16. Jin, X., Ren, X., Preotiuc-Pietro, D., Cheng, P.: Dataless knowledge fusion by merging weights of language models. In: Proc. ICLR (2023)
17. Lee, Y.A., Ko, C.Y., Pedapati, T., Chung, I.H., Yeh, M.Y., Chen, P.Y.: Star: Spectral truncation and rescale for model merging. In: Proc. HLT-NAACL (2025)
18. Leroy, V., Cabon, Y., Revaud, J.: Grounding image matching in 3D with MASt3R. In: Proc. ECCV (2024)
19. Marczak, D., Magistri, S., Cygert, S., Twardowski, B., Bagdanov, A.D., van de Weijer, J.: No Task Left Behind: Isotropic Model Merging with Common and Task-Specific Subspaces. In: Proc. ICML (2025)
20. Matena, M.S., Raffel, C.A.: Merging models with fisher-weighted averaging. In: Proc. NeurIPS (2022)
21. Michele, B., Boulch, A., Vu, T.H., Puy, G., Marlet, R., Courty, N.: Train till you drop: Towards stable and robust source-free unsupervised 3d domain adaptation. In: Proc. ECCV (2024)
22. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.Y., Xu, H., Sharma, V., Li, S.W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: DINOv2: Learning robust visual features without supervision. TMLR (2024)
23. Ortiz-Jiménez, G., Favero, A., Frossard, P.: Task arithmetic in the tangent space: Improved editing of pre-trained models. In: Proc. NeurIPS (2023)
24. Puy, G., Boulch, A., Marlet, R.: Using a waffle iron for automotive point cloud semantic segmentation. In: Proc. ICCV (2023)
25. Puy, G., Gidaris, S., Boulch, A., Siméoni, O., Sautier, C., Pérez, P., Bursuc, A., Marlet, R.: Three pillars improving vision foundation model distillation for lidar. In: Proc. CVPR (2024)
26. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: Proc. ICML (2021)
27. Ranzinger, M., Heinrich, G., Kautz, J., Molchanov, P.: AM-RADIO: Agglomerative vision foundation model reduce all domains into one. In: Proc. CVPR (2024)
28. Sarıyıldız, M.B., Weinzaepfel, P., Lucas, T., de Jorge, P., Larlus, D., Kalantidis, Y.: Dune: Distilling a universal encoder from heterogeneous 2d and 3d teachers. In: Proc. CVPR (2025)
29. Sarıyıldız, M.B., Weinzaepfel, P., Lucas, T., Larlus, D., Kalantidis, Y.: UNIC: Universal classification models via multi-teacher distillation. In: Proc. ECCV (2024)
30. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: Proc. ECCV (2012)

31. Sokar, G., Dziugaite, G.K., Arnab, A., Iscen, A., Castro, P.S., Schmid, C.: Continual learning in vision-language models via aligned model merging. arXiv preprint arXiv:2506.03189 (2025)
32. Wang, K., Dimitriadis, N., Favero, A., Ortiz-Jiménez, G., Fleuret, F., Frossard, P.: Lines: Post-training layer scaling prevents forgetting and enhances model merging. In: Proc. ICLR (2025)
33. Wang, K., Dimitriadis, N., Ortiz-Jiménez, G., Fleuret, F., Frossard, P.: Localizing task information for improved model merging and compression. In: Proc. ICML (2024)
34. Wortsman, M., Ilharco, G., Gadre, S.Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A.S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al.: Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In: Proc. ICML (2022)
35. Xiao, P., Shao, Z., Hao, S., Zhang, Z., Chai, X., Jiao, J., Li, Z., Wu, J., Sun, K., Jiang, K., et al.: Pandaset: Advanced sensor suite dataset for autonomous driving. In: ITSC (2021)
36. Yadav, P., Tam, D., Choshen, L., Raffel, C., Bansal, M.: TIES-merging: Resolving interference when merging models. In: Proc. NeurIPS (2023)
37. Yang, E., Wang, Z., Shen, L., Liu, S., Guo, G., Wang, X., Tao, D.: Adamerging: Adaptive model merging for multi-task learning. Proc. ICLR (2024)
38. Yi, L., Gong, B., Funkhouser, T.: Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds. In: Proc. CVPR (2021)
39. Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ADE20k dataset. IJCV (2019)