Task Vector Bases: A Unified and Scalable Framework for Compressed Task Arithmetic

Siqi Zeng *siqi6@illinois.edu*
University of Illinois Urbana-Champaign

Yifei He *yifeihe3@illinois.edu*
University of Illinois Urbana-Champaign

Meitong Liu *meitong4@illinois.edu*
University of Illinois Urbana-Champaign

Weiqiu You *weiqiuy@seas.upenn.edu*
University of Pennsylvania

Yifan Hao *yifanh12@illinois.edu*
University of Illinois Urbana-Champaign

Yao-Hung Hubert Tsai *y.h.huberttsai@gmail.com*
Okinawa Institute of Science and Technology

Makoto Yamada *makoto.yamada@oist.jp*
Okinawa Institute of Science and Technology

Han Zhao *hanzhao@illinois.edu*
University of Illinois Urbana-Champaign

Reviewed on OpenReview: *https://openreview.net/forum?id=zkc7u3mIaE*

Abstract

Task arithmetic, representing downstream tasks through linear operations on task vectors, has emerged as a simple yet powerful paradigm for transferring knowledge across diverse settings. However, maintaining a large collection of task vectors introduces scalability challenges in both storage and computation. We propose Task Vector Bases, a framework compressing T task vectors into $M < T$ basis vectors while preserving the functionality of task arithmetic. By representing each task vector as a structured linear combination of basis atoms, our approach supports standard operations such as addition, negation, as well as more advanced arithmetic ones. The framework is orthogonal to other efficiency-oriented improvements in task arithmetic and can be used in combination with them. We provide theoretical analysis showing that basis compression retains addition generalization guarantees and provides unlearning error bounds that depend on reconstruction quality. Empirically, our proposed basis construction methods consistently outperform heuristic basis construction baselines and, in some cases, even surpass the performance of full task vector collections across diverse downstream applications while reducing storage and computational requirements. The code is available at `https://github.com/uiuctml/TaskVectorBasis`.

1 Introduction

Task vectors (Ilharco et al., 2022) have emerged as a lightweight technique for model editing. Given a downstream task of interest, a task vector is constructed by subtracting the pretrained model weights from

those of a fine-tuned model, encoding task-specific information as a direction in parameter space. These vectors can be combined through simple arithmetic operations such as addition and negation, enabling flexible capabilities such as multi-task composition, task analogy or domain generalization, and even task unlearning. Because of their simplicity and effectiveness, task vectors have been widely studied and applied in vision (Chen et al., 2025; Zhu et al., 2025; Tian et al., 2025) and language (Zhao et al., 2024; Wang et al., 2024d; Fu et al., 2025) domains.

Despite these successes, an important question remains: *how well do task vector methods scale with the number of tasks T?* From the perspective of task addition, it is often considered efficient compared to multi-task learning on the full mixture of data. But practical deployments increasingly involve dozens of tasks, where both computation and memory footprint still scale linearly with T. Storing each task vector, which is the same size as the full model weights, can already be huge for LLMs, and even if disk storage is a relatively moderate cost, the primary systems bottleneck still arises during composition: loading 72 fine-tuned ViT-B/32 models simultaneously can require more than 200GB of memory when combination coefficients are learned with gradient-based methods (Huang, 2023; Li et al., 2024; Yang et al., 2024), making large-scale GPU training infeasible.

To address this, layer-wise merging (Yang et al., 2023) has been proposed to improve flexibility and scalability by operating at the level of individual layers, which leads to a significant performance increase. However, due to the memory constraint, these methods require sequential loading and unloading of layer parameters between CPU and GPU, which incurs significant overhead and prevents full utilization of GPU parallelism (He et al., 2025). This makes these methods prohibitively slow in practice, let alone overlooking cross-layer dependencies within tasks. Beyond addition, in negation, to forget any task in a large collection requires access to specific task vectors to be removed, and thus scales poorly with T when vectors must be stored and retrieved individually. This scaling limitation becomes particularly acute when addition and negation are combined, such as composing many tasks while selectively removing a subset. Finally, as T grows, optimizing addition itself becomes increasingly difficult, often leading to degraded multi-task performance in both offline composition (Ilharco et al., 2022) and continual merging scenarios (Tang et al., 2025).

In light of the limitations when scaling the number of task vectors, we introduce Task Vector Bases, an algorithm that compresses the entire T task vectors into M basis vectors, yielding a novel unified framework that can be directly integrated with existing task-arithmetic applications. Our framework is orthogonal to and compatible with other compression methods too. Our contributions are:

- **Scalability.** We reduce both storage and computation overhead from a factor of T, the number of tasks, to M with $M \leq T$ denoting the number of basis vectors, making task vector methods practical in large-scale or resource-constrained settings at minimum loss of downstream performance. With only 50% of the vectors, in some settings, we can achieve results better than using 100% of the task vectors, and even when reducing M to $25\% \times T$ we still retain up to 97% of the full performance.
- **Unified framework.** Task Vector Bases provide a unified framework broadly compatible with all weight-space vector steering operations, including offline/online addition and negation.
- **Principled construction.** By learning bases aligned with the geometry of task vectors, our approach avoids the inefficiencies of heuristics such as PCA or random selection and consistently achieves stronger downstream results.
- **Theoretical and empirical validation.** We theoretically compare the generalization performance between full task vectors and compressed bases, and empirically validate the benefits of our method across diverse applications.

2 Preliminaries

Problem Setting Let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be the loss function, and $h : \mathcal{X} \times \Theta \to \mathcal{Y} \subseteq \mathbb{R}$ be the classifier. When the context is clear, we omit some arguments for ℓ and h. We consider the initial pre-trained model parameter $\theta_0 \in \mathbb{R}^d$, which is fine-tuned on T tasks to yield fine-tuned parameters $\{\theta_1, \ldots, \theta_T\}$ with respect

to the loss functions $\{\ell_1, \ldots, \ell_T\}$. For n_i training samples $D_i = \{(x_{i1}, y_{i1}), \ldots, (x_{in_i}, y_{in_i})\}$ drawn from the i-th task distribution $\mathcal{D}_i$, we denote the population risk evaluated at θ as $\mathcal{L}_i(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}[\ell_i(h(x,\theta), y)]$.

Task Arithmetic (TA) and Applications Given a collection of T tasks, task vectors are defined as $\tau_i := \theta_i - \theta_0, \forall i \in [T]$, where θ_0 is the pretrained initialization and θ_i is the fine-tuned model on task i. Ilharco et al. (2022) showed that meaningful model behaviors can be obtained through simple arithmetic on task vectors. In general, we view **Offline Task Addition** as producing a merged model

$$\theta^T_{\text{Add}} = \theta_0 + \mathcal{M}(\tau_1, \ldots, \tau_T), \tag{1}$$

where the algorithm $\mathcal{M}$ specifies how the task vectors are combined (see Tab. 1 for concrete examples of $\mathcal{M}$). Depending on the downstream application, $\mathcal{M}$'s input can be either learned from in-domain data for multi-task learning or from out-of-domain (OOD) validation data for domain generalization. **Online Task Addition** can be applied in a continual setting where T tasks arrive over time. If unlimited storage were available, one could save all past task data exemplars and task vectors (Coleman et al., 2024; Marczak et al., 2024; Chitale et al., 2023), reducing t-th step model to be $\theta^{(t)} := \theta^t_{\text{Add}}$. To forget a particular task j, we subtract the task vector from θ_0 scaled by a task-specific coefficient α, yielding **Task Negation**

$$\theta_{\text{Neg},j} = \theta_0 - \alpha\tau_j, \quad \forall j \in [T]. \tag{2}$$

Task negation has found broad applications in safety related settings, including unlearning untruthfulness and toxicity (Hu et al., 2024a), mitigating social biases (Shirafuji et al., 2025), and reducing hallucinations (Daheim et al., 2024).

3 Improving Efficiency with Bases Arithmetic

Under limited compute, it is impractical to save all task vectors for a large number of tasks T, and simply impossible in the online sequential setting with infinitely growing number of tasks. In this study, we focus on the unique computational bottlenecks of task vector methods that arise from operations that scale linearly with T. Let space complexity refer to *persistent storage of parameters required to support any future arithmetic operations* (i.e., artifacts that must live on memory and/or disk in the long term), and time complexity refers to the number of *elementary operations needed to produce an edited model prior to inference* from the stored artifacts. Under the setting of Ilharco et al. (2022), supporting task arithmetic operation requires a space complexity of $O(Td)$, since supporting task negation for any of the possibly randomly selected T tasks means every τ_j must remain accessible. In terms of time complexity, task addition requires scanning and weighting all T vectors during merging, which costs $O(Td)$ for the combination step alone. The merging method $\mathcal{M}$ incurs additional per-task-vector processing cost $c_{\mathcal{M}}$ (e.g., computing per-task statistics, hyperparameter search, validation forward passes, or mask optimization), giving an effective merging cost of $O(Td \cdot c_{\mathcal{M}})$. Thus the naive method couples linear space in T with linear-time merging, even if inference time remains $O(d)$ once the edited model is formed.

We propose the framework of **Task Vector Bases**, compressing original T task vectors into M d-dim basis vectors $\{B_1, \ldots, B_M\}$ with $M < T$. Bases arithmetic framework can be written as

$$\theta^M_{\text{Add}} = \theta_0 + \mathcal{M}(B_1, \ldots, B_M), \qquad \theta_{\text{Neg},j} = \theta_0 - \alpha \cdot \hat{\tau}_j(B_1, \ldots, B_M),\ \forall j \in [T], \tag{3}$$

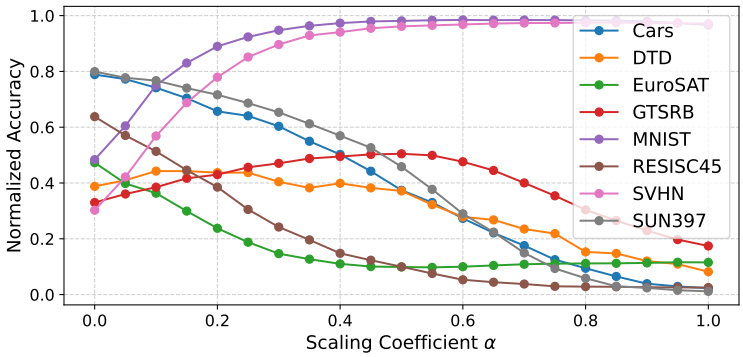
where bases addition replaces any τ_i with B_i, and negation for any of T tasks can be recovered from saved bases. Our goal is to create a compact representation that preserves the information needed to support all existing task arithmetic operations and follow-up improvements built upon naive task arithmetic, while reducing both storage and time complexity from dependence on T to M.

3.1 Principal Components as Bases

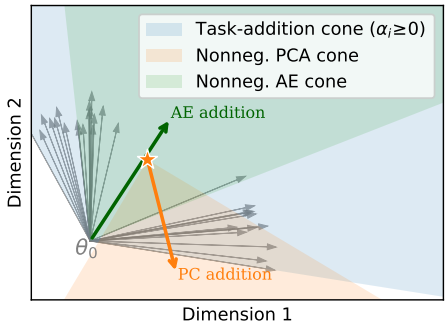
A natural candidate for compressing T task vectors into $M < T$ directions is Principal Component Analysis (**PCA**). Let $\mathbf{T} = [\tau_1, \ldots, \tau_T] \in \mathbb{R}^{d\times T}$ denote the task vector matrix stacking task vectors with mean $\mu \in \mathbb{R}^d$. PCA yields $\mathbf{T} - \mu\mathbf{1}^\top \approx (\mathbf{U}_M\mathbf{S}_M)\mathbf{V}_M^\top$, where $\mathbf{B} = \mathbf{U}_M\mathbf{S}_M \in \mathbb{R}^{d\times M}$ serves as the scaled basis

matrix and $\mathbf{C} = \mathbf{V}_M^\top \in \mathbb{R}^{M \times T}$ are task-specific coefficients. Original task vector matrix is reconstructed by $\hat{\mathbf{T}} = \mu \mathbf{1}^\top + \mathbf{BC}$. PCA representation requires storing $O(Md)$ basis vectors for addition and additional $T \times M$ coefficients for negation, and all T task vectors can be recovered using transient working memory, thus matching the desired complexity profile and achieving the optimal rank-M approximation in Frobenius norm by Eckart & Young (1936).

Despite its appeal for negation, it is not compatible with the way task addition is typically performed:



(a) Addition under PCA. Normalized accuracy vs. scaling coefficient α for different datasets using PCA components.



(b) Comparing geometry of task vectors, PC, and our AE bases for addition. Grey arrows are original task vectors from $\mathbf{T}$.

Figure 1: Limitations of PCA for task addition: (a) performance view and (b) geometric view.

1. **Coefficient tuning.** In many addition formulations (Ilharco et al., 2022; Yadav et al., 2024; Ortiz-Jimenez et al., 2024), when applied to task vectors, the merged model can be written as $\theta_{\text{Add}}^T = \theta_0 + \alpha \mathcal{M}(\tau_1, \ldots, \tau_T)$ where α is a single *nonnegative* scalar tuned on validation data shared across task vectors. This formulation assumes that each task vector can be only rescaled positively according to the naming of task *addition*, but PCA bases are arbitrary orthogonal directions with their signs having no semantic meaning. Therefore, these types of addition methods are not compatible with PCA basis: in the left panel of Fig. 1a, half of the datasets cannot even recover the pretrained model accuracy at $\alpha = 0$ by applying Ilharco et al. (2022) on PCA basis. This indicates that these basis components are completely misaligned with the original task vectors, so tuning nonnegative scaling factors cannot interpolate back to the pretrained initialization. Geometrically (right panel of Fig. 1b), this failure arises because first, the bases are not anchored at θ_0 but at ★; second, even if we shift the anchor, the orientation of principal components is determined by the SVD implementation and the sign of PCs (boundary directions of the orange cone) doesn't affect the approximation optimality. The shaded orange cone spanned by nonnegative coefficients does not fully overlap with blue cone from merging original task vectors. In Fig. 1b, tasks to the far left are completely misaligned from PC bases interpolation, leading to irreversible information loss for addition.

2. **Interpretability of directions.** In task arithmetic, each task dataset D_i corresponds to a specific task vector τ_i, so merging methods often rely on this 1-to-1 correspondence. A representative example is Localize-and-Stitch (**L&S**) (He et al., 2024), which trains a binary mask to identify the most relevant parameters for each task by solving

$$S_i = \arg\min_{S \in \mathbb{R}^d} \mathcal{L}_i\big(\theta_0 + \sigma(S) \odot \tau_i\big) + \lambda \|\sigma(S)\|_1, \tag{4}$$

where the empirical loss is computed on the validation data D_i. This method is attractive since the task vectors masked by sparse S_i can be stored at much lower cost than full task vectors, while still enabling accurate model merging. Under a PCA basis representation, assume $\mu = 0$, then bases $\mathbf{B} = \mathbf{U}_M \mathbf{S}_M \approx \mathbf{T}\mathbf{V}_M$ can be written as the linear combination of input vectors, but $\mathbf{V}_M$ involves negative values. This breaks the dataset–vector interpretability: one cannot straightforwardly construct a validation dataset for a PCA basis to learn its mask since negative coefficients would correspond to a nonsensical negative task dataset contribution.

3.2 Softmax-Activated Linear Autoencoder as Bases

3.2.1 Basis Construction

So how to design bases to preserve the spectral optimality of PCA while aligning the basis direction with original task vectors? We propose to use a softmax activated linear autoencoder, which ensures each basis vector can be interpreted as a convex combination of input task vectors, which is essential for supporting both addition and negation operations in a unified basis framework.

Definition 3.1 (Autoencoder with softmax encoder and linear decoder)**.** Given parameters $\mathbf{A} \in \mathbb{R}^{T\times M}$, let the encoder weight be a column-wise softmax at temperature $\tau > 0$: $\mathbf{W}_e[:, m] := \mathrm{softmax}\big(\mathbf{A}_{:,m}/\tau\big) \in \Delta^{T-1}, \forall m \in [M]$ and $\mathbf{W}_d \in \mathbb{R}^{M\times T}$ be a linear decoder. We minimize the reconstruction loss as the squared error in Frobenius norm:

$$\mathcal{L}_{\mathrm{AE}}(\mathbf{W}_e, \mathbf{W}_d) = \|\hat{\mathbf{T}} - \mathbf{T}\|_F^2 := \|\mathbf{T}\mathbf{W}_e\mathbf{W}_d - \mathbf{T}\|_F^2. \tag{5}$$

Lemma 3.2 (Equivalent Gram reformulation)**.** *With gram matrix* $\mathbf{G} := \mathbf{T}^\top\mathbf{T}$ *and* $\mathbf{E} = \mathbf{W}_e\mathbf{W}_d - \mathbf{I}_T$ *as above, Eq.* (5) *is equivalent to*

$$\mathcal{L}_{\mathrm{AE}}(\mathbf{W}_e, \mathbf{W}_d) = \|\mathbf{G}^{1/2}\mathbf{E}\|_F^2 = \mathrm{Tr}(\mathbf{E}^\top\mathbf{G}\mathbf{E}). \tag{6}$$

Proof. $\|\mathbf{T}\mathbf{W}_e\,\mathbf{W}_d - \mathbf{T}\|_F^2 = \|\mathbf{T}\mathbf{E}\|_F^2 = \mathrm{Tr}((\mathbf{T}\mathbf{E})^\top(\mathbf{T}\mathbf{E})) = \mathrm{Tr}(\mathbf{E}^\top\mathbf{T}^\top\mathbf{T}\mathbf{E}) = \mathrm{Tr}(\mathbf{E}^\top\mathbf{G}\mathbf{E})$. Since $\mathbf{G} \succeq 0$, $\mathbf{G}^{1/2}$ is its PSD square root so $\mathrm{Tr}(\mathbf{E}^\top\mathbf{G}\mathbf{E}) = \|\mathbf{G}^{1/2}\mathbf{E}\|_F^2$. □

Remark 3.3. The formulation in Eq. (5) requires storing $\mathbf{T} \in \mathbb{R}^{d\times T}$, which scales with model parameters d. The Gram reformulation Eq. (6) only depends on $\mathbf{G}, \mathbf{E} \in \mathbb{R}^{T\times T}$, eliminating the d-dependence during gradient-based optimization on GPU if precomputing the Gram matrix on CPU. Note that loading $\mathbf{T}$ and computing $\mathbf{G}$ are one time transient costs ($O(dT)$ space, $O(dT^2)$ time) incurred only during basis construction; once bases are built, only M basis vectors of dimension d (and a small $M \times T$ decoder matrix for negation) need to be persisted (see Sec. E.2 for empirical profiling).

It is well known that the global optimum of a linear autoencoder without softmax activations can be characterized by PCA solution (Baldi & Hornik, 1989). For any $M < T$, the best rank-M reconstruction's error is given by the spectral bound in both Frobenius and spectral norm:

$$\min_{\mathrm{rank}(\hat{\mathbf{T}})\le M} \|\hat{\mathbf{T}} - \mathbf{T}\|_F^2 = \sum_{i=M+1}^{r} \lambda_i(\mathbf{G}), \qquad \min_{\mathrm{rank}(\hat{\mathbf{T}})\le M} \|\hat{\mathbf{T}} - \mathbf{T}\|_2^2 = \lambda_{M+1}(\mathbf{G}), \tag{7}$$

where $r = \mathrm{rank}(\mathbf{T})$ and $\lambda_i(\mathbf{G})$ are the eigenvalues of $\mathbf{G}$ in nonincreasing order, with minimum achieved when choosing any $\mathbf{W}_e$ whose column space equals the top-M eigenspace of $\mathbf{G}$ denoted by $S_\star$, and $\mathbf{W}_d$ be the ordinary least square solution given fixed $\mathbf{W}_e$. When softmax activation is applied to the encoder, the loss is at least Eq. (7) with equality conditions below (proof details in Sec. B.1),

Theorem 3.4 (Exact Achievability with Softmax Encoder)**.** *Using a softmax-activated encoder* $\mathbf{W}_e$*, Eq.* (5) *attains the spectral optimum in Eq.* (7) *if and only if there exist* M *linearly independent vectors* $x_1, \ldots, x_M \in S_\star$ *with strictly positive coordinates.*

3.2.2 Basis Arithmetic

After solving Eq. (6), we define the Autoencoder (**AE**) bases as

$$\mathbf{B} := \mathbf{T}\mathbf{W}_e \in \mathbb{R}^{d\times M} = \left[\sum_{i=1}^{T} \mathbf{W}_e[i,1]\tau_i \;\middle|\; \ldots \;\middle|\; \sum_{i=1}^{T} \mathbf{W}_e[i,M]\tau_i\right], \tag{8}$$

where the bases are the convex combinations of original task vectors.

Algorithm 1 Online Bases Addition

Require: Buffer budget M, basis construction pipeline AE_TRAIN$(\cdot)$

Initialize basis set $\mathbf{B} \leftarrow \emptyset$ and $k \leftarrow 0$
for each new task $t = 1, 2, \ldots$ **do**
  Receive new task vector $\tau_t \in \mathbb{R}^d$
  if $k < M$ **then**
    $\mathbf{B} \leftarrow \mathbf{B} \cup \{\tau_t\}$, $k \leftarrow k + 1$
  else
    $(\mathbf{W}_e, \mathbf{W}_d) \leftarrow$ AE_TRAIN$(\mathbf{B}, M)$
    $\mathbf{U} = \mathbf{B}\mathbf{W}_e \in \mathbb{R}^{d\times(M-1)}$
    $\mathbf{B} \leftarrow [\mathbf{U}, \tau_t] \in \mathbb{R}^{d\times M}$
  $\theta^{(t)} = \theta_0 + \mathcal{M}_{\cup_{i=1}^{T} \widetilde{D}_i}(\mathbf{B})$

For **offline bases addition**, with M bases, no matter what merging method $\mathcal{M}$ we use, we always subsample $n_i \cdot M/T$ validation data from each task dataset D_i, and re-use all existing merging methods directly on M bases paired with subsampled dataset denoted as $\widetilde{D}_i$. This immediately reduces the time complexity of data-based merging methods from $\times T$ evaluations to $\times M$, since they are now applied on Mn_i effective data points in $\cup_{i=1}^{T} \widetilde{D}_i$.

To see how AE bases solving the limitations of PCs, for coefficient tuning methods, softmax bases guarantee that each B_m is a convex combination of task vectors. If we apply Ilharco et al. (2022) for addition, any non-negative mixture of the bases $\sum_{m=1}^{M} \alpha_m B_m,\ \alpha_m \geq 0$ remains a nonnegative linear combination of the original τ_i by plugging in Eq. (8). Hence the feasible region spanned by softmax bases is always a subset of the nonnegative cone defined by task addition like in Fig. 1b.

Besides, in settings with 1-to-1 correspondence between tasks and datasets, we define validation data mixture for basis B_m with its encoder weights $\mathbf{W}_e[:, m]$, which specify how input tasks contribute to the basis. For example, to use He et al. (2024), for basis m, the effective objective becomes

$$S_m = \underset{S\in\mathbb{R}^d}{\arg\min} \Big(\sum_{i=1}^{T} \mathbf{W}_e[i, m]\, \mathcal{L}_i\big(\theta_0 + \sigma(S) \odot B_m\big) \Big) + \lambda \|\sigma(S)\|_1. \tag{9}$$

That is, instead of attaching one dataset to one task vector, each basis B_m is paired with a convex combination of the original task validation losses, weighted by the encoder weights, thus S_m is now a joint mask applicable to multiple tasks. This allows Localize-and-Stitch to remain applicable in the basis setting while requiring space only for the M bases rather than all T task vectors to reduce the memory burden. Tab. 1 summarizes how existing merging methods adapt to the basis setting (see Sec. C for algorithmic details).

Table 1: Comparison of original task-vector formulations and their basis-setting counterparts during addition. Top group: methods requiring 1-to-1 task-dataset correspondence, where basis-level statistics are constructed via encoder-weighted combinations. Bottom group: methods whose mathematical form is unchanged by replacing τ_i with B_m.

Method	T tasks setting	M bases setting
Fisher merge	$\theta_{\text{Add}} = \frac{\sum_{i=1}^{T} \mathbf{F}_i \theta_i}{\sum_{i=1}^{T} \mathbf{F}_i}$	$\theta_{\text{Add}} = \frac{\sum_{m=1}^{M} \widetilde{\mathbf{F}}_m (\theta_0 + B_m)}{\sum_{m=1}^{M} \widetilde{\mathbf{F}}_m},\ \widetilde{\mathbf{F}}_m = \sum_{i=1}^{T} \mathbf{W}_e[i, m]\mathbf{F}_i$
RegMean	$\theta_{\text{Add}} = \Big(\sum_{i=1}^{T} \mathbf{X}_i^\top \mathbf{X}_i\Big)^{-1} \Big(\sum_{i=1}^{T} \mathbf{X}_i^\top \mathbf{X}_i \theta_i\Big)$	$\theta_{\text{Add}} = \Big(\sum_{m=1}^{M} \widetilde{\mathbf{G}}_m\Big)^{-1} \Big(\sum_{m=1}^{M} \widetilde{\mathbf{G}}_m (\theta_0 + B_m)\Big),\ \widetilde{\mathbf{G}}_m = \sum_{i=1}^{T} \mathbf{W}_e[i, m]\mathbf{X}_i^\top \mathbf{X}_i$
Localize & Stitch	$S_i = \underset{S\in\mathbb{R}^d}{\arg\min}\ \mathcal{L}_i(\theta_0 + \sigma(S) \odot \tau_i) + \lambda\|\sigma(S)\|_1$	$S_m = \underset{S\in\mathbb{R}^d}{\arg\min} \Big(\sum_{i=1}^{T} \mathbf{W}_e[i, m]\, \mathcal{L}_i(\theta_0 + \sigma(S) \odot B_m)\Big) + \lambda\|\sigma(S)\|_1$
Task Arithmetic	$\theta_{\text{Add}} = \theta_0 + \sum_{i=1}^{T} \alpha_i \tau_i$	$\theta_{\text{Add}} = \theta_0 + \sum_{m=1}^{M} \alpha_m B_m$
TIES	$\theta_{\text{Add}} = \theta_0 + \alpha \text{TrimElectMerge}(\tau_1, \ldots, \tau_T)$	$\theta_{\text{Add}} = \theta_0 + \alpha \text{TrimElectMerge}(B_1, \ldots, B_m)$
AdaMerging	$\min_{\lambda_1,\ldots,\lambda_T} \sum_{x_i\in D_t} H\Big[f\Big(x_i; \theta_0 + \Big\{\sum_{t=1}^{T} \lambda_t^l \tau_t^l\Big\}_{l=1}^{L}\Big)\Big]$	$\min_{\lambda_1,\ldots,\lambda_M} \sum_{x_i\in \widetilde{D}_t} H\Big[f\Big(x_i; \theta_0 + \Big\{\sum_{m=1}^{M} \lambda_m^l B_m^l\Big\}_{l=1}^{L}\Big)\Big]$
aTLAS	$\min_{\Lambda_1,\ldots,\Lambda_T} \sum_{(x_i,y_i)\in D_t} \Big[\ell(f(x_i; \theta_0 + \sum_{i=1}^{T} \Lambda_i \tau_i), y_i)\Big]$	$\min_{\Lambda_1,\ldots,\Lambda_M} \sum_{(x_i,y_i)\in \widetilde{D}_t} \Big[\ell(f(x_i; \theta_0 + \sum_{m=1}^{M} \Lambda_m B_m), y_i)\Big]$
WEMoE	$\theta_{\text{merged,l}}^{mlp} = \text{WEMoE}(\theta_{0,l}^{mlp}, \tau_{1,l}^{mlp}, \ldots, \tau_{T,l}^{mlp} \mid \text{R}_l)$	$\theta_{\text{merged,l}}^{mlp} = \text{WEMoE}(\theta_{0,l}^{mlp}, B_{1,l}^{mlp}, \ldots, B_{M,l}^{mlp} \mid \text{R}_l)$

For **online bases addition**, we consider a more practical limited-compute setting where only M finite vectors can be stored persistently in Alg. 1. At step t, when we found the buffer is full, we apply the autoencoder compression to reduce these M vectors back into $M - 1$ bases, then we put a new task vector τ_t also back into the buffer, ensuring storage cost remains fixed as $O(Md)$ while supporting an unbounded sequence of tasks. In particular, as $T \to \infty$, neither persistent storage nor per step computation depends on T. Per step time complexity only depends on M: the compression to reduce M vectors down to $M - 1$ costs $O(M^2 d)$, and the merging algorithm operates on M bases. Note that although we call the AE training

pipeline for each step t, empirically compared to line 10's addition step, the basis processing step cost is negligible (Sec. E.2).

For **bases negation**, we first reconstruct the full task vector matrix

$$\hat{\mathbf{T}} = \mathbf{B}\mathbf{W}_d, \tag{10}$$

and then apply negation directly on the reconstructed vectors so that we forget task j via $\theta_0 - \alpha\hat{\tau}_j = \theta_0 - \alpha\hat{\mathbf{T}}[:, j]$. In terms of storage, we only need to persist the $M \times d$ bases $\mathbf{B}$ and the decoder weight $\mathbf{W}_d \in \mathbb{R}^{M\times T}$, but since $M < T \ll d$, the total storage is still much smaller than the naive $O(Td)$ required to store all task vectors directly, and thus matches our efficiency goal.

3.2.3 Theoretical Guarantees

To compare original task arithmetic methods and basis version, we briefly state our theoretical analysis guided by Taylor expansion for standard task arithmetic in Ilharco et al. (2022) and their counterparts under basis representations. See proof details in Sec. B.2.

We state the shared assumptions used across all theorems below. (A1) **Fine-tuning regime:** $\forall i \in [T]$, $\frac{\partial \mathcal{L}_i(\theta_i)}{\partial \theta} = \mathbf{0}$ and $\exists\, C > 0$ such that $\|\tau_i\|^2 \le C$. (A2) **Local smoothness:** each loss $\mathcal{L}_i$ is L_i-locally smooth at θ_i, i.e., $\mathcal{L}_i(\theta) - \mathcal{L}_i(\theta_i) \le \langle \theta - \theta_i, \nabla\mathcal{L}_i(\theta_i)\rangle + \frac{L_i}{2}\|\theta - \theta_i\|^2$ for $\|\theta - \theta_i\|^2 = O(C)$. (A3) **Scaling coefficients:** $\alpha_i \ge 0$ and $\sum_{i=1}^T \alpha_i = 1$. We further denote the pairwise task vector similarity as $\epsilon = \max_{i\neq j} |\langle \tau_i, \tau_j\rangle|/C^2$.

Theorem 3.5 (Task Addition & Basis Addition)**.** *For $\theta^T_{\text{Add}} = \theta_0 + \sum_{j=1}^T \alpha_j\tau_j$, then $\forall i \in [T]$, the generalization gap between the merged model and finetuned model is bounded by:*

$$\mathcal{L}_i(\theta^T_{\text{Add}}) - \mathcal{L}_i(\theta_i) \;\le\; L_i C(1+\epsilon). \tag{11}$$

When each basis is defined in Eq. (8), and bases addition merged model is $\theta^M_{\text{Add}} = \theta_0 + \sum_{m=1}^M \alpha_m B_m$, the same bound in Eq. (11) holds.

Theorem 3.6 (OOD Generalization with Task Vectors & Bases)**.** *Suppose τ_{tar} is an unseen task vector and we want to generalize to this target task with Eq. (1) with existing task vectors in $\mathbf{T}$. If $\exists i^\star \in [T]$ with $\langle \tau_{\text{tar}}, \tau_{i^\star}\rangle \ge \gamma C$, then there exists set of merging coefficients $\alpha_i, \forall i \in [T]$ such that*

$$\mathcal{L}_{\text{tar}}(\theta^T_{\text{Add}}) \;\le\; \mathcal{L}_{\text{tar}}(\theta_{\text{tar}}) + L_{\text{tar}} C(1-\gamma). \tag{12}$$

If we use basis instead when $\theta^M_{\text{Add}} = \theta_0 + \sum_{m=1}^M \alpha_m B_m$, if some basis B_m contains $\tau_{i^\star}$ with weight at least ρ, i.e. $\mathbf{W}_e[i^\star, m] \ge \rho$,

$$\mathcal{L}_{\text{tar}}(\theta^M_{\text{Add}}) \;\le\; \mathcal{L}_{\text{tar}}(\theta_{\text{tar}}) + L_{\text{tar}} C(1-\rho\gamma). \tag{13}$$

Theorem 3.7 (Task Negation & Basis Negation)**.** *For task vector negation $\theta_{\text{Neg},i} = \theta_0 - \alpha_i\tau_i$, for all control tasks $j \neq i$, the performance gap compared to pretrained model is bounded by:*

$$\mathcal{L}_j(\theta_{\text{Neg},i}) - \mathcal{L}_j(\theta_0) \;\le\; L_j C\Big(\tfrac{3}{2} + \epsilon\Big). \tag{14}$$

Let $\widehat{\tau}_i$ be the i-th reconstructed task vector from Eq. (10). If Eq. (5) is minimized to the spectral lower bound, for $\theta_{\text{Neg},i} = \theta_0 - \alpha_i\widehat{\tau}_i$, where $\lambda_{M+1}(\mathbf{G})$ is the $(M+1)$-th eigenvalue of the Gram matrix $\mathbf{G} = \mathbf{T}^\top\mathbf{T}$:

$$\mathcal{L}_j(\theta_{\text{Neg},i}) - \mathcal{L}_j(\theta_0) \;\le\; L_j C\left(\frac{5}{2} + 2\epsilon\right) + L_j\lambda_{M+1}(\mathbf{G}). \tag{15}$$

Remark 3.8. All proofs proceed via Taylor expansion of the loss around the fine-tuned model, combined with the convex combination structure of softmax bases which ensures bases remain in the non-negative task vector cone. Notably, evaluating the pretrained model θ_0 directly on $\mathcal{L}_i$ yields $\mathcal{L}_i(\theta_0) - \mathcal{L}_i(\theta_i) \le L_i C$, which is no worse than the addition bound up to the cross-task interference term ϵ. In other words, task addition does not hurt too much whenever fine-tuning does not help that much ($L_i C$ is small), which partially explains why task arithmetic has mainly been applied to well-pretrained models where individual fine-tuning

gains are moderate. The bounds also reveal which factors govern the worst case and connect to concrete methodological improvements: smaller L_i (flatter minima are preferred for merging (Iurada et al., 2025; Lee et al., 2025)) and smaller ϵ (reducing task vector interference via sparsification (Yadav et al., 2024; He et al., 2024)). The key takeaways for basis compression are:

- Thm. 3.5: compression is "free" since the bound is identical for T task vectors and M bases, because any non-negative combination of bases is itself a non-negative combination of the original task vectors.
- Thm. 3.6: the penalty from compression is exactly quantified by the encoder coverage factor ρ.
- Thm. 3.7: the additional term $\lambda_{M+1}(\mathbf{G})$ gives a computable criterion for choosing M, which vanishes when M captures the principal components of $\mathbf{G}$.

4 Experiments

We present the experiments organized by task arithmetic application under basis framework. Details of datasets, metrics, hyperparameters, and additional experiments including verification of theoretical claims (Sec. D), sensitivity of τ (Sec. E.1), choice of subsampling/weighting (Sec. F.1), results on generative tasks (Sec. F.2), numerical precision (Tab. 12), multi-seed runs (Tab. 15), and softmax vs. linear encoder ablation (Tab. 20) are deferred to Appendix.

4.1 Bases Addition

4.1.1 Offline Multitask Learning

Table 2: Comparison of absolute addition accuracy across ViT models under 8, 14, and 20 vision tasks (Wang et al., 2024a) with $M = 50\%$ of total tasks. Bold entries are the best-performing basis method within each block, while underlined entries are cases where basis addition outperforms full task-vector addition. See normalized accuracies and per dataset results in Tab. 16, and Figs. 9 to 11.

Method	**ViT-B/16**			**ViT-B/32**			**ViT-L/14**		
	8 task	14 task	20 task	8 task	14 task	20 task	8 task	14 task	20 task
Pretrained	0.554	0.620	0.598	0.481	0.569	0.556	0.698	0.691	0.656
Finetuned	0.924	0.913	0.916	0.904	0.893	0.898	0.943	0.934	0.935
TA (Ilharco et al., 2022)	0.754	0.705	0.658	0.708	0.653	0.605	0.850	0.794	0.740
RandSelect	0.645	0.649	0.620	0.643	0.638	<u>0.611</u>	0.697	0.727	0.609
PCA	0.495	0.578	0.573	0.532	0.571	0.585	0.589	0.653	0.642
AE (Ours)	**0.666**	**0.673**	**0.635**	**0.689**	<u>**0.660**</u>	<u>**0.613**</u>	**0.736**	**0.753**	**0.715**
TIES (Yadav et al., 2024)	0.797	0.732	0.682	0.751	0.680	0.634	0.869	0.795	0.757
RandSelect	0.664	0.659	0.620	0.655	0.649	**0.627**	0.733	0.733	0.708
PCA	0.496	0.578	0.573	0.533	0.571	0.595	0.589	0.652	0.644
AE (Ours)	**0.672**	**0.672**	**0.635**	**0.687**	**0.651**	0.607	**0.742**	**0.754**	**0.711**
L&S (He et al., 2024)	0.759	0.681	0.601	0.767	0.652	0.598	0.778	0.753	0.701
RandSelect	0.553	0.534	0.437	0.546	0.494	0.434	0.670	0.677	0.601
PCA	0.523	0.450	0.410	0.467	0.408	0.361	0.667	0.589	0.543
AE (Ours)	**0.667**	**0.672**	<u>**0.641**</u>	**0.691**	<u>**0.667**</u>	<u>**0.628**</u>	**0.736**	**0.732**	<u>**0.729**</u>

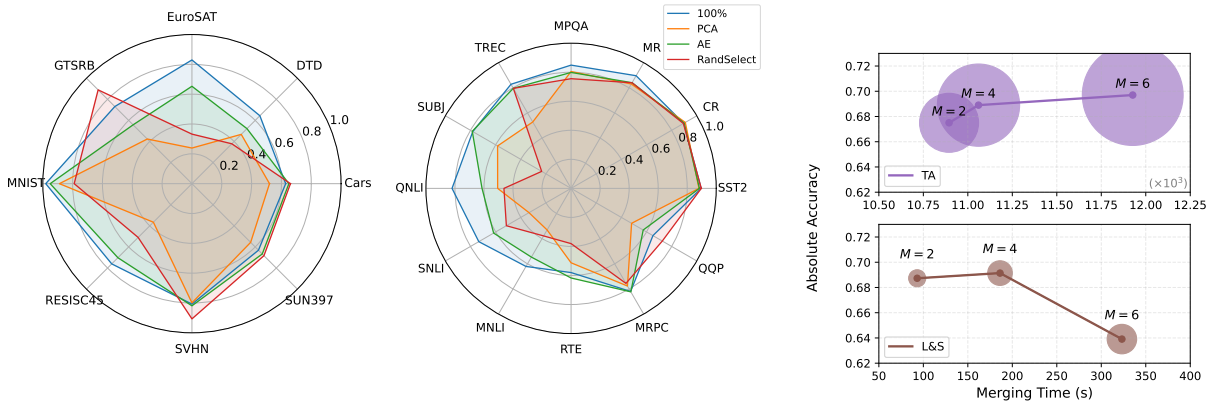
Tab. 2 compares basis construction strategies across ViT models for vision tasks, and Tab. 3 is the comparison on the language benchmark with RoBERTa models. We include 3 popular merging methods, TA, TIES with coefficient tuning, and L&S where the last one can be used to additionally compress task vectors with sparsity, and compare three reduced-basis approaches: RandSelect (randomly selecting available tasks), PCA, and our AE. We keep 50% of the vectors in bases for vision and 25% for languauge experiments. For a fair comparison, all bases methods use the same subsampling strategy in Sec. 3.2.2 to only use $n_i M/T$ validation data. While constructing L&S bases, in RandSelect, we allow the method to only learn task masks for the selected task vectors, and in PCA, since we cannot disentangle nonnegative contributions from each original

Table 3: Comparison of absolute addition accuracy with RoBERTa-base model on 12 language task benchmark with bases number $M = 25\%$ of the total tasks. 100% means using all task vectors for corresponding merging methods. See the normalized accuracy version and full per dataset results in Tab. 17 and Fig. 12. With 25% of task vectors, we can recover up to 97% (L&S-AE) of the accuracy.

TA (Ilharco et al., 2022)				**TIES** (Yadav et al., 2024)				**L&S** (He et al., 2024)			
100%	RandSelect	PCA	AE	100%	RandSelect	PCA	AE	100%	RandSelect	PCA	AE
0.626	0.453	0.449	**0.472**	0.600	0.453	0.469	**0.470**	0.759	0.619	0.623	**0.733**

task to a principal component, we assign uniform weights across tasks where $\mathbf{W}_e[i, m] = 1/T$ in Eq. (9). See the alternative baseline only using positive weights for PCA in Tab. 16.

Across nearly all settings, AE achieves the best performance within each method block, consistently outperforming both RandSelect and PCA, implying that learning a compact AE basis captures more useful task interactions than other methods. The advantage of AE is especially pronounced in L&S, where interpretability of bases vectors is central to the method's validity. For several large-task regimes, AE or even RandSelect can outperform full-task merging, showing that fewer but more coherent vectors may lead to better generalization while reducing storage cost and merging time. As predicted in Sec. 3.1, PCA consistently performs poorly in vision experiments, and sometimes even worse than the pretrained baseline. We leave the comparison of bases methods across M in Fig. 8.



(a) ViT-B/32, 8 tasks. (b) RoBERTa-base, 12 tasks (c) Acc. vs. merging time and storage.

Figure 2: (a)–(b) Radar plots showing per-task accuracy across vision (100% = TA) and language (100% = L&S) benchmarks. (c) Absolute accuracy against merging time for different M, with circle size indicating disk storage cost in gigabytes (same scale across top and bottom).

For per task results in Figs. 2a and 2b, we observe a nested pattern where the 100% merge generally dominates or on par with AE, and AE in turn dominates PCA. Therefore, using all available task vectors provides the strongest signal, AE compresses them while preserving most of the structure, and PCA mixes wrong directions, leading to degraded performance. Unlike AE and PCA, RandSelect is inherently unstable: by dropping more than half the task vectors blindly, it can achieve remarkably strong performance on particular tasks (e.g., GTSRB and SVHN), but this comes at the cost of severe degradation on other tasks where critical information is lost (e.g., DTD and SUBJ).

Fig. 2c show accuracy versus merging time, with bubble size indicating storage cost. Clearly, both merging time and storage grow with the number of bases M. This highlights that our basis compression method provides improvements in both time and space efficiency due to $M < T$. Importantly, our basis compression is complementary to sparsity-based approaches like L&S, showing compatibility with existing task vector compression frameworks which may further compress bases storage up to roughly 90% if bases are saved in CSR format. In terms of accuracy, for TA, increasing M yields better accuracy but for L&S, however,

accuracy does not monotonically improve with larger M. In fact, accuracy drops for $M = 100\%$ in Tab. 2 for certain settings, and adding too many task vectors may actually hurt performance due to increasing task conflicts Ilharco et al. (2022).

4.1.2 Offline Fewshot OOD Generalization

Method	2 shot	16 shot
aTLAS (Zhang et al., 2024)	0.826	0.837
$\text{aTLAS}_{\text{subsample}}$	0.819	0.835
RandSelect	0.821	0.829
PCA	0.817	0.829
AE (Ours)	**0.822**	**0.830**
$\text{aTLAS}^{\geq 0}$	0.825	0.833
$\text{aTLAS}^{\geq 0}_{\text{subsample}}$	0.820	0.830
RandSelect	**0.821**	0.827
PCA	0.816	0.822
AE (Ours)	0.820	**0.828**

Table 4: ViT-B/32 results with OOD 6 tasks at $M = 50\%$ of in domain 8 tasks.

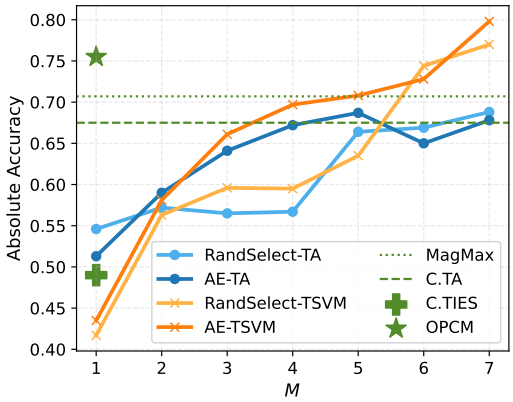


Figure 3: Online continual results on ViT-B/32 with 8 tasks varying the size of storage buffer.

Tab. 4 presents addition evaluated on unseen 6 OOD tasks by merging in domain 8 task vectors from Tab. 2 under a few shot setting where direct finetuning on tiny target subset only creates weak models. We use aTLAS (Zhang et al., 2024) as the base addition method, a flexible framework where each scaling coefficient is a learned block matrix (more details see Tab. 1). We compare its standard formulation with a square-parameterized version $\text{aTLAS}^{\geq 0}$, which guarantees the nonnegativity of coefficients simulating (Ilharco et al., 2022) at the cost of minor performance drop.

We benchmark aTLAS and $\text{aTLAS}_{\text{subsample}}$ (trained with 100% task vectors but only 50% of the coefficient-learning data, and basis methods with $M = 50\%$, evaluated under both unconstrained and nonnegative merging coefficients settings. In Tab. 4 k-shot refers to the number of per class samples for aTLAS without any subsampling. Key observations include: first, performance gap among basis methods is smaller than in Tab. 2, but PCA remains consistently worse likely due to its misaligned anchor not centered at θ_0 mentioned in Sec. 3.1. Second, our AE method outperforms RandSelect and PCA in 3 out of 4 settings, particularly when coefficients are unconstrained, showing AE's flexibility. Finally, with very limited data (2-shot), basis methods slightly outperform $\text{aTLAS}_{\text{subsample}}$ since aTLAS has higher degrees of freedom and requires more data during learning. But in 16-shot, $\text{aTLAS}_{\text{subsample}}$ catches up and surpasses the bases methods.

4.1.3 Online Continual Learning

Fig. 3 illustrates the online continual task merging setting where we fix M checkpoints stored in persistent memory and evaluate the final merged model $\theta^{(t)}$ on all t tasks seen in the sequence. We compare two basis construction methods, RandSelect and AE, paired with two merging rules: TA and TSVM (Gargiulo et al., 2025). Note that the full offline TSVM accuracy with $M = 100\%$ is 0.857. PCA is omitted since it consistently underperforms in offline addition experiments. For baselines, we also include prior continual merging methods in green: MagMax (Marczak et al., 2024) (selecting maximum-magnitude task vector entries), Continual TA $\theta^{(t)} = \theta^{(t-1)} + \lambda\tau_t = \theta_0 + \lambda\sum_{i=1}^{t-1}\tau_i + \lambda\tau_t$, Continual TIES, and OPCM (Tang et al., 2025). In prior work, continual merging was only defined for $M = 1$, i.e., storing a single model checkpoint $\theta^{(t-1)}$ and merging it with the new task vector τ_t. However, methods like MagMax and Continual TA can be reformulated as running statistics, making their $M = 1$ version equivalent to the offline $M = 100\%$ limit.

We see that as M increases, accuracy steadily improves, and AE almost consistently outperforms RandSelect across both TA and TSVM, showing clear advantages for most values of M, although when $M \to T$, AE and RandSelect roughly coincide as they are both approaching full-rank approximation. Comparing to a fixed green baseline method, AE achieves better performance than RandSelect with fewer checkpoints.

For example, with $M = 4$ (50%), AE–TSVM already surpasses Continual TA, while RandSelect requires $M = 6$ (75%). Finally, although MagMax and OPCM are specifically designed for online continual merging, pairing AE with a strong offline merging method (TSVM here) eventually outperforms specialized baselines once M is moderately large. Thus, even a weak basis method like RandSelect, combined with an effective offline $\mathcal{M}$, provides strong continual merging performance without specialized online setup adaptations. With future advances in the field, we expect even smaller M values to surpass SOTA continual merging baselines. The result across different sizes of ViT is included in Tab. 21 in the Appendix.

4.2 Bases Negation

In negation experiments, since RandSelect cannot be directly applied to negation (discarded task vectors cannot be recovered or inferred from saved bases without retraining task vectors), we propose RandProj as the random baseline, where bases are defined as the random orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{d \times M}$ obtained from QR decomposition of a Gaussian random matrix. We save projection coefficients $\mathbf{C} = \mathbf{Q}^\top \mathbf{T} \in \mathbb{R}^{M \times T}$, and during negation, we reconstruct T task vectors by $\hat{\mathbf{T}} = \mathbf{QC} = \mathbf{QQ}^\top \mathbf{V}$, projecting each task vector onto the random subspace spanned by $\mathbf{Q}$.

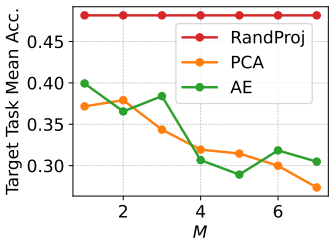


Figure 4: Target task forgetting as a function of M.

Method	ViT-B/16		ViT-B/32		ViT-L/14	
	Target (↓)	Control	Target (↓)	Control	Target (↓)	Control
TA	0.213	0.654	0.240	0.649	0.190	0.729
Gradient Ascent	0.019	0.007	0.027	0.003	0.039	0.163
RandProj	0.494	0.683	0.482	0.633	0.589	0.755
PCA	0.190	0.646	0.319	0.610	0.178	0.724
AE	0.255	0.659	0.307	0.605	0.270	0.734

Table 5: Target and control metrics comparison averaged under 8 vision tasks' unlearning setting. Yellow rows use reconstructed task vectors from $M = 50\%$ bases.

Tab. 5 compares methods under the 8-task setting with $M = 50\%$, reporting performance on both target and control tasks. On the target tasks, lower accuracy indicates better forgetting, while on control tasks (ImageNet (Deng et al., 2009)), higher accuracy indicates better retention of pretrained knowledge. We observe that PCA and AE both achieve significant forgetting compared to RandProj, and the difference between PCA and AE metrics can be treated as tradeoffs between target and control metrics. Fig. 4 reports target task mean accuracy as a function of the number of bases M. Both PCA and AE gradually comparably reduce target task accuracy as M increases. This behavior is expected from Eq. (7): with suitable hyperparameter tuning, the softmax AE variant can approximate the same spectral lower bound as PCA. In contrast, RandProj remains flat at roughly the same level as the pretrained model θ_0, showing that it fails to forget even as M grows, since random projections do not align with task-specific directions. We also include Gradient Ascent, a data aware unlearning baseline that maximizes loss on target data. Although it achieves the lowest target accuracy, it destroys control performance (e.g., 0.3% on ViT-B/32), unlike task vector negation methods which preserve model utility.

5 Related Work

5.1 Compressed Task Arithmetic

A number of recent efforts have sought to make task vector methods more scalable through compression, and we discuss the broader scope of task arithmetic and model merging in Sec. A. One line of work focuses on localization or sparsification, identifying subsets of parameters most relevant for each task and masking out the rest. By sparsifying task updates into different subspaces, these methods reduce task interference and improve memory efficiency, since only sparse weights or masks are stored (He et al., 2024; Yadav et al., 2024; Yu et al., 2024; Davari & Belilovsky, 2025; Tang et al., 2023a; Wang et al., 2024b). A complementary line explores quantization (Liu et al., 2024a; Huang et al., 2025), where task vectors are quantized directly without notable degradation in merging performance (Kim et al., 2025). Sparsification and quantization

both act along the parameter-dimension axis of the task matrix (reducing d), while our Task Vector Bases approach operates on the task-count axis (reducing T); we empirically verify this orthogonality in Sec. E.3. For single-task model merging (Sec. A.1), prior work proposes alternative optimization algorithms (Li et al., 2024) or assumes fine-tuned weights lie in a thin Gaussian shell (Jang et al., 2025), a different context from our multitask setup.

5.2 Bases in Low-rank and Subspace Merging Methods

Task Singular Vectors (TSV, -M for merging and -C suffix for compression) (Gargiulo et al., 2025) and Marczak et al. (2025) study task updates at the layer level and apply SVD to decompose unflattened task matrices, where the latter further differentiate between task-shared and task-specific subspaces. While these works also rely on eigenbasis constructions, they differ in both motivation and scope from ours: they both require access to all full task vectors when computing singular components, and they primarily target improving addition performance when $M = 100\%$. Besides, TSV-C only supports model compression only when the task metadata is known or inferred through routing based merging methods (Tang et al., 2024). In contrast, our Task Vector Bases framework is designed as a general compression mechanism that unifies any downstream applications not only limited to addition, and aim to approximate (typically treated as upper bound) $M = 100\%$ metrics with $M < 100\%$ bases.

5.3 Parameter-Efficient Fine-Tuning and Low-Rank Adaptation

Another closely related line of work comes from parameter-efficient fine-tuning, especially low-rank adaptation methods. LoRA (Hu et al., 2021) freezes the pretrained weights and learns low-rank update matrices, dramatically reducing the number of trainable parameters and adaptation memory relative to full fine-tuning. QLoRA (Dettmers et al., 2023) further combines LoRA with 4-bit quantization, enabling efficient fine-tuning of very large models while largely preserving the quality of 16-bit fine-tuning. QA-LoRA (Xu et al., 2023) makes this adaptation quantization-aware so that the learned low-rank updates integrate more naturally with quantized deployment. More recent variants such as EoRA (Liu et al., 2024b) and CLoQ (Deng et al., 2025) further improve efficiency for quantized models, with EoRA providing fine-tuning-free low-rank compensation and CLoQ using calibration-based initialization to find optimal LoRA components for quantized LLMs before fine-tuning. These methods address the same broad goal of efficient task adaptation, but they operate along a different axis: they reduce the per-task adaptation complexity in parameter space, effectively compressing along the model/update dimension d, whereas our focus is on compressing a collection of T task vectors into a smaller set of shared basis vectors. This distinction is critical for the online setting where $T \to \infty$: methods that compress only along d still store one vector per task, so storage grows without bound, whereas our bases representation remains fixed at M vectors regardless of how many tasks arrive (see Tab. 19 for a compatibility experiment combining LoRA with our bases methods).

5.4 Matrix Factorization and Dimensionality Reduction Methods

A related line of work has explored nonnegative variants of matrix factorization such as nonnegative PCA (Montanari & Richard, 2015) and nonnegative matrix factorization (NMF) (Févotte & Idier, 2011; Cichocki & Phan, 2009). These approaches have been proposed as remedies for the interpretability limitations of PCA, since enforcing nonnegativity on either the basis vectors or the coefficients ensures that components can be interpreted as additive components. However, applying these methods in our setting is not straightforward. Standard NMF requires the input matrix itself to be nonnegative, which is incompatible with task vectors that contain signed weight updates. Nonnegative PCA similarly constrains basis vectors to the nonnegative orthant, preventing them from aligning with unconstrained task vector directions. Another family of dimensionality reduction methods includes sparse coding and dictionary learning (Mairal et al., 2009), which learn basis atoms and sparse codes for reconstructing high-dimensional data. While these approaches are applicable to signed inputs, they differ from our design in a critical way. In sparse coding, the learned basis vectors are unconstrained. In summary, even if preprocessing tricks for NMF (e.g., splitting positive and negative channels or affine shifts) are applied, both type of methods distort the geometry of the task-vector cone and break the guarantees needed for task arithmetic: without softmax constraints, adding coefficient vectors (the

operation underlying task addition) may yield mixtures outside the cone spanned by the original tasks as in PCA, thus breaking the structure that our analysis relies upon and can widen the addition generalization gap.

6 Conclusion

We introduced Task Vector Bases, a unifying framework for compressing collections of task vectors into a compact set of basis vectors. This approach addresses the key computational bottlenecks of task vector methods—space and time complexity scaling linearly with the number of tasks—while preserving compatibility with all standard arithmetic operations. Empirically, Task Vector Bases not only reduce storage and computation but also improve task performance over heuristic alternatives such as PCA or random selection. Our analysis further clarifies the generalization performance difference between full task vectors and compressed bases, showing that bases provide a scalable and effective representation for model editing. We hope this work establishes Task Vector Bases as a practical building block for future research on efficient and interpretable weight space interventions.

Broader Impact. This work is primarily methodological and efficiency-oriented. However, we acknowledge that compressed task arithmetic, particularly task negation and model editing, could in principle be applied to remove safety constraints or alignment mechanisms from deployed models. This dual-use concern is not unique to our framework and applies broadly to model editing research. We encourage practitioners to apply appropriate safeguards when deploying models modified through any weight space intervention, including the methods proposed here.

Acknowledgment

Part of this work was done when both SZ and HZ were visiting the Simons Institute via the Modern Paradigm in Generalization program. SZ, YH, ML and HZ were partly supported by an NSF IIS grant No. 2416897 and an NSF CAREER Award No. 2442290. HZ would like to thank the support of a Google Research Scholar Award and Nvidia Academic Grant Award. WY was supported by a gift from AWS AI to Penn Engineering's ASSET Center for Trustworthy AI and NSF award CCF 2442421. MY was partly supported by JSPS KAKENHI Grant Number 24K03004 and by JST ASPIRE JPMJAP2302. The views and conclusions expressed in this paper are solely those of the authors and do not necessarily reflect the official policies or positions of the supporting companies and government agencies. We want to express a special thanks to Yuechun Sun who pointed out a way to improve the upper bound for task addition theorem. We are also thankful to Wei Hu, Yuki Takezawa, Jose Renato Restom and anonymous reviewers for other helpful discussions.

References

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *arXiv preprint arXiv:2403.13187*, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Shiqi Chen, Jinghan Zhang, Tongyao Zhu, Wei Liu, Siyang Gao, Miao Xiong, Manling Li, and Junxian He. Bring reason to vision: Understanding perception and reasoning through model merging. *arXiv preprint arXiv:2505.05464*, 2025.

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.

Rajas Chitale, Ankit Vaidya, Aditya Kane, and Archana Ghotkar. Task arithmetic with lora for continual learning. *arXiv preprint arXiv:2311.02428*, 2023.

Andrzej Cichocki and Anh-Huy Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 92(3):708–721, 2009.

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.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*, 2018.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.

Eric Nuertey Coleman, Luigi Quarantiello, Julio Hurtado, and Vincenzo Lomonaco. Adaptive lora merging for efficient domain incremental learning. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*, 2024.

Nico Daheim, Nouha Dziri, Mrinmaya Sachan, Iryna Gurevych, and Edoardo Ponti. Elastic weight removal for faithful and abstractive dialogue generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7096–7112, 2024.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision*, pp. 270–287. Springer, 2025.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Yanxia Deng, Aozhong Zhang, Selcuk Gurses, Naigang Wang, Zi Yang, and Penghang Yin. Cloq: Enhancing fine-tuning of quantized llms via calibrated lora initialization. *arXiv preprint arXiv:2501.18475*, 2025.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*, 2005.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1 (3):211–218, 1936.

Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural computation*, 23(9):2421–2456, 2011.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Zichuan Fu, Xian Wu, Yejing Wang, Wanyu Wang, Shanshan Ye, Hongzhi Yin, Yi Chang, Yefeng Zheng, and Xiangyu Zhao. Training-free llm merging for multi-task learning. *arXiv preprint arXiv:2506.12379*, 2025.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodola. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 18695–18705, 2025.

Guillaume Garrigos and Robert M Gower. Handbook of convergence theorems for (stochastic) gradient methods. *arXiv preprint arXiv:2301.11235*, 2023.

Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *International conference on neural information processing*, pp. 117–124. Springer, 2013.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Advances in Neural Information Processing Systems*, 37:8093–8131, 2024.

Yifei He, Yuzheng Hu, Yong Lin, Tong Zhang, and Han Zhao. Localize-and-stitch: Efficient model merging via sparse task arithmetic. *arXiv preprint arXiv:2408.13656*, 2024.

Yifei He, Siqi Zeng, Yuzheng Hu, Rui Yang, Tong Zhang, and Han Zhao. Mergebench: A benchmark for merging domain-specialized llms. *arXiv preprint arXiv:2505.10833*, 2025.

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.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168–177, 2004.

Xinshuo Hu, Dongfang Li, Baotian Hu, Zihao Zheng, Zhenyu Liu, and Min Zhang. Separate the wheat from the chaff: Model deficiency unlearning via parameter-efficient module operation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18252–18260, 2024a.

Yuzheng Hu, Ruicheng Xian, Qilong Wu, Qiuling Fan, Lang Yin, and Han Zhao. Revisiting scalarization in multi-task learning: A theoretical perspective. *Advances in Neural Information Processing Systems*, 36, 2024b.

Brian Huang. *Adversarial learned soups: neural network averaging for joint clean and robust performance*. PhD thesis, Massachusetts Institute of Technology, 2023.

Chenyu Huang, Peng Ye, Shenghe Zheng, Xiaohui Wang, Lei Bai, Tao Chen, and Wanli Ouyang. Dynamic base model shift for delta compression. *arXiv preprint arXiv:2505.11344*, 2025.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Leonardo Iurada, Marco Ciccone, and Tatiana Tommasi. Efficient model editing with task-localized sparse fine-tuning. *arXiv preprint arXiv:2504.02620*, 2025.

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

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models. In *European Conference on Computer Vision*, pp. 207–223. Springer, 2025.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.

Jean Kaddour. Stop wasting my time! saving days of imagenet and bert training with latest weight averaging. *arXiv preprint arXiv:2209.14981*, 2022.

Youngeun Kim, Seunghwan Lee, Aecheon Jung, Bogon Ryu, and Sungeun Hong. Task vector quantization for memory-efficient model merging. *arXiv preprint arXiv:2503.06921*, 2025.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

Viet Dac Lai, Chien Van Nguyen, Nghia Trung Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan A Rossi, and Thien Huu Nguyen. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. *arXiv preprint arXiv:2307.16039*, 2023.

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

Yeoreum Lee, Jinwook Jung, and Sungyong Baik. Mitigating parameter interference in model merging via sharpness-aware fine-tuning. *arXiv preprint arXiv:2504.14662*, 2025.

Tao Li, Weisen Jiang, Fanghui Liu, Xiaolin Huang, and James T Kwok. Scalable learned model soup on a single gpu: An efficient subspace training strategy. *arXiv e-prints*, pp. arXiv–2407, 2024.

Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

James Liu, Guangxuan Xiao, Kai Li, Jason D Lee, Song Han, Tri Dao, and Tianle Cai. Bitdelta: Your fine-tune may only be worth one bit. *Advances in Neural Information Processing Systems*, 37:13579–13600, 2024a.

Shih-Yang Liu, Maksim Khadkevich, Nai Chit Fung, Charbel Sakr, Chao-Han Huck Yang, Chien-Yi Wang, Saurav Muralidharan, Hongxu Yin, Kwang-Ting Cheng, Jan Kautz, et al. Eora: Fine-tuning-free compensation for compressed llm with eigenspace low-rank approximation. *arXiv preprint arXiv:2410.21271*, 2024b.

Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364, 2019.

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. *arXiv preprint arXiv:2406.15479*, 2024.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pp. 689–696, 2009.

Daniel Marczak, Bartłomiej Twardowski, Tomasz Trzciński, and Sebastian Cygert. Magmax: Leveraging model merging for seamless continual learning. In *European Conference on Computer Vision*, pp. 379–395. Springer, 2024.

Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D Bagdanov, and Joost van de Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. *arXiv preprint arXiv:2502.04959*, 2025.

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

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

RV Mises and Hilda Pollaczek-Geiringer. Praktische verfahren der gleichungsauflösung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 9(1):58–77, 1929.

Andrea Montanari and Emile Richard. Non-negative principal component analysis: Message passing algorithms and sharp asymptotics. *IEEE Transactions on Information Theory*, 62(3):1458–1484, 2015.

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.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pp. 722–729. IEEE, 2008.

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.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *arXiv preprint cs/0409058*, 2004.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. *arXiv preprint arXiv:2308.01263*, 2023.

Sunny Sanyal, Atula Neerkaje, Jean Kaddour, Abhishek Kumar, and Sujay Sanghavi. Early weight averaging meets high learning rates for llm pre-training. *arXiv preprint arXiv:2306.03241*, 2023.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=M3Y74vmsMcY`.

Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. Natural language understanding with the quora question pairs dataset. *arXiv preprint arXiv:1907.01041*, 2019.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 1671–1685, 2024.

Daiki Shirafuji, Makoto Takenaka, and Shinya Taguchi. Bias vector: Mitigating biases in language models with task arithmetic approach. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 2799–2813, 2025.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. *arXiv preprint arXiv:2410.19735*, 2024.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

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

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. *arXiv preprint arXiv:2310.04742*, 2023b.

Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. Merging multi-task models via weight-ensembling mixture of experts. *arXiv preprint arXiv:2402.00433*, 2024.

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

Jie Tian, Xiaoye Qu, Zhenyi Lu, Wei Wei, Sichen Liu, and Yu Cheng. Extrapolating and decoupling image-to-video generation models: Motion modeling is easier than you think. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 12512–12521, 2025.

Nilesh Tripuraneni, Michael Jordan, and Chi Jin. On the theory of transfer learning: The importance of task diversity. *Advances in neural information processing systems*, 33:7852–7862, 2020.

Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 210–218. Springer, 2018.

Alex Wang. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Hu Wang, Congbo Ma, Ibrahim Almakky, Ian Reid, Gustavo Carneiro, and Mohammad Yaqub. Rethinking weight-averaged model-merging. *arXiv preprint arXiv:2411.09263*, 2024a.

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

Peng Wang, Li Shen, Zerui Tao, Yan Sun, Guodong Zheng, and Dacheng Tao. A unified analysis for finite weight averaging. *arXiv preprint arXiv:2411.13169*, 2024c.

Xinpeng Wang, Chengzhi Hu, Paul Röttger, and Barbara Plank. Surgical, cheap, and flexible: Mitigating false refusal in language models via single vector ablation. *arXiv preprint arXiv:2410.03415*, 2024d.

Kaiyue Wen, Tengyu Ma, and Zhiyuan Li. How does sharpness-aware minimization minimize sharpness? *arXiv preprint arXiv:2211.05729*, 2022.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39:165–210, 2005.

Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo-Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. *arXiv preprint arXiv:2109.01903*, 2021. `https://arxiv.org/abs/2109.01903`.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/wortsman22a.html`.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717*, 2023.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36, 2024.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*, 2023.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024.

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. In *Forty-first International Conference on Machine Learning*, 2024.

Frederic Z Zhang, Paul Albert, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. Knowledge composition using task vectors with learned anisotropic scaling. *Advances in Neural Information Processing Systems*, 37:67319–67354, 2024.

Jinghan Zhang, Junteng Liu, Junxian He, et al. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, 36:12589–12610, 2023.

Weixiang Zhao, Yulin Hu, Zhuojun Li, Yang Deng, Jiahe Guo, Xingyu Sui, Yanyan Zhao, Bing Qin, Tat-Seng Chua, and Ting Liu. Towards comprehensive post safety alignment of large language models via safety patching. *arXiv preprint arXiv:2405.13820*, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Didi Zhu, Yibing Song, Tao Shen, Ziyu Zhao, Jinluan Yang, Min Zhang, and Chao Wu. Remedy: Recipe merging dynamics in large vision-language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

A Additional Related Work

A.1 Single-task Merging Methods

Prior to Task Arithmetic (Ilharco et al., 2022), researchers discussed how to combine models fine-tuned on the same task, with some minor differences due to hyperparameter changes, as an alternative to ensembles, starting with model soup (Wortsman et al., 2022). Since fine-tuned models capture more domain-specific skills while pretrained models contain more generic knowledge, WiSE-FT (Wortsman et al., 2021) proposed merging the pretrained model and the fine-tuned model via linear interpolation, achieving balanced or even optimal performance on both in-domain and out-of-distribution generalization metrics. (Izmailov et al., 2018) introduced stochastic weight averaging, which includes intermediate checkpoints before model convergence for model merging. Several close variants, such as exponentially moving averaging (Szegedy et al., 2016) and LAtest Weight Averaging (Kaddour, 2022; Sanyal et al., 2023), have been explained theoretically under a unified framework (Wang et al., 2024c).

A.2 Multi-task Merging Methods

The major difference from Sec. A.1 is that all methods discussed in this subsection focus on the setting that one pretrained model is fine tuned on many different tasks. Task arithmetic (Ilharco et al., 2022) can be seen as the generalization of the single-task model merging method, model soup (Wortsman et al., 2022), where task vectors are simply averaged. In (Ilharco et al., 2022), however, the scaling coefficients α are allowed to be tuned. Since then, several ideas have been proposed to improve task arithmetic. First, since tuning α is time-consuming, popular approaches such as Fisher merging (Matena & Raffel, 2022), RegMean (Jin et al., 2022), AdaMerging (Yang et al., 2023), Evol (Akiba et al., 2024) aim to find better methods to automatically adjust scaling coefficients for improved task arithmetic performance. Second, instead of using standard fine-tuning to obtain τ, alternative fine-tuning methods, such as tangent space fine-tuning (Ortiz-Jimenez et al., 2024) and parameter-efficient fine-tuning methods (Zhang et al., 2023; Tang et al., 2023b; Stoica et al., 2024), are employed in task arithmetic to disentangle task information for better merging. Third, to reduce task vector conflicts, task vectors can be sparsified into different subspaces by localization as we discussed in Sec. 5. Finally, inspired by the Mixture-of-Experts (Shazeer et al., 2017) mechanism, task vector merging performance can be enhanced by learned routers that dynamically merge task-specific and task-shared information (Lu et al., 2024; Tang et al., 2024). For more details on the latest task arithmetic methods and their applications, we refer readers to the model merging survey (Yang et al., 2024).

B Proof Details

B.1 Exact Achievability with Softmax Encoder

Lemma B.1 (Softmax surjects onto the simplex interior)**.** *Write* $\mathrm{int}(\Delta^{T-1}) = \Delta^{T-1} \cap \mathbb{R}^T_{++}$ *for the interior of the simplex. For any* $b \in \mathrm{int}(\Delta^{T-1})$ *and any* $\tau > 0$*, there exists* $a \in \mathbb{R}^T$ *such that* $\mathrm{softmax}(a/\tau) = b$*. One choice is* $a_i = \tau \log b_i + c$ *for any constant* $c \in \mathbb{R}$*.*

Proof. This is immediate from the definition of softmax and the invariance under adding a constant: $\mathrm{softmax}(z)_i = e^{z_i} / \sum_j e^{z_j}$. □

Theorem 3.4 (Exact Achievability with Softmax Encoder)**.** *Using a softmax-activated encoder* $\mathbf{W}_e$*, Eq.* (5) *attains the spectral optimum in Eq.* (7) *if and only if there exist* M *linearly independent vectors* $x_1, \ldots, x_M \in S_\star$ *with strictly positive coordinates.*

Proof. Let $\mathbb{R}^T_{++} := \{x \in \mathbb{R}^T : x > 0\}$. The statement is equivalent to say $x_j \in \mathbb{R}^T_{++}$ for all j.

($\Rightarrow$) If the optimum is achieved by some $\mathbf{W}_e$ with columns $w_1, \ldots, w_M \in \mathrm{int}(\Delta^{T-1})$, then (Baldi & Hornik, 1989) implies that the column space of $\mathbf{W}_e$ must equal $S_\star$. Since each w_m is strictly positive, we conclude $w_m \in S_\star \cap \mathbb{R}^T_{++}$, and the w_m are linearly independent as they span $S_\star$.

($\Leftarrow$) Conversely, suppose there exist M independent $x_1, \ldots, x_M \in S_\star \cap \mathbb{R}^T_{++}$. Normalize each to sum to one, $w_m := x_m/(\mathbf{1}^\top x_m) \in \operatorname{int}(\Delta^{T-1})$. Set $\mathbf{W}_e = [w_1 \ \cdots \ w_M]$, which has column space $S_\star$. By surjectivity of softmax (Lemma B.1), there exists $\mathbf{A}$ such that $\operatorname{softmax}(\mathbf{A}/\tau) = \mathbf{W}_e$ (column-wise softmax). Then by (Baldi & Hornik, 1989) with the least-squares optimal decoder $\mathbf{W}_d$, the reconstruction error equals the spectral bound, achieving the optimum. □

B.2 Generalization of Task and Bases Arithmetic

B.2.1 Common Assumptions

We first introduce several practical shared common assumptions used in our theorems.

Assumption B.2 (Fine-tuning Regime)**.** *We assume that* $\forall i \in [T], \frac{\partial \mathcal{L}_i(\theta_i)}{\partial \theta} = \mathbf{0}$ *and* $\exists C > 0$ *such that* $\|\tau_i\|^2 \leq C$.

This assumption is often met in practice since θ_i is fine-tuned from the pre-trained model θ_0 on the particular downstream task $\mathcal{D}_i$ until convergence. Furthermore, during the fine-tuning regime, the change of model parameters is relatively small. Through a sparsity localization technique, (He et al., 2024) show that it is sufficient to only fine-tune 1%∼5% of the model parameters for competitive performances.

Assumption B.3 (Local Smoothness)**.** *Any fine tuning loss function* $\mathcal{L}$ *is* L_i*-locally smooth w.r.t. model parameters at* θ_i*, which means for any* $\theta \in \Theta$ *such that* $\|\theta - \theta_i\|^2 = O(C), \mathcal{L}(\theta) - \mathcal{L}(\theta_i) \leq \left\langle \theta - \theta_i, \frac{\partial \mathcal{L}(\theta_i)}{\partial \theta} \right\rangle + \frac{L_i}{2} \|\theta - \theta_i\|^2$. Note that θ_i is the fine-tuned model trained on $\mathcal{D}_i$ and $L_i = \|\mathbf{H}(\theta_i)\|_2$ is the spectral norm of the Hessian matrix of $\mathcal{L}$, evaluated locally at θ_i. We hide the subscript of L_i when the context is clear.

Smoothness is a standard assumption in optimization theory (Garrigos & Gower, 2023) and has been used in recent work on Sharpness-Aware Minimization (Foret et al., 2020; Wen et al., 2022) to encourage flatter minima and improve generalization. Since we focus mainly on the fine-tuning regime in the analysis, we only consider smoothness in a local region.

Assumption B.4 (Scaling Coefficients)**.** *Let* $\alpha_1, \ldots, \alpha_T$ *be the coefficients used to scale the task vector in task arithmetic. We assume* $\alpha_i \geq 0, \forall i$ *and* $\sum_{i \in [T]} \alpha_i = 1$.

B.2.2 Task Addition & Basis Addition

Theorem B.5 (Task Addition for Multitask Learning)**.** *Let* $0 < \epsilon \leq 1$ *be a universal constant such that* $\forall i \neq j, |\cos(\tau_i, \tau_j)| \leq \epsilon$.[1] *Let task addition* $\theta^T_{\text{Add}} = \theta_0 + \sum_{j=1}^T \alpha_j \tau_j$ *be the model parameter used for multitask learning, then* $\forall i \in [T]$,

$$\mathcal{L}_i(\theta^T_{\text{Add}}) - \mathcal{L}_i(\theta_i) \leq L_i C(1 + \epsilon). \tag{16}$$

Proof. Note that since $\theta^T_{\text{Add}} = \theta_0 + \sum_{j=1}^T \alpha_j \tau_j$, so

$$\|\theta^T_{\text{Add}} - \theta_0\|^2 = \left\| \sum_{i=j}^T \alpha_j \tau_j \right\|^2 \leq \left(\sum_{j=1}^T \alpha_j \|\tau_j\| \right)^2 \leq C,$$

[1]This is NOT assuming all task vectors are near-orthogonal. Depending on the benchmark, ϵ can be as large as 1 where task vectors are completely aligned (or flipped). Note that task vector orthogonality itself cannot guarantee merging performance: even if PCA creates orthogonal task vectors its performance can be significantly worse than our AE method which does not ensure task vector bases orthogonality.

which means that θ_{Add}^T is within the fine-tuning regime and satisfies the local smoothness assumption. Hence, if $x \sim \mathcal{D}_i$

$$\begin{aligned}
\mathcal{L}_i(\theta_{\text{Add}}^T) - \mathcal{L}_i(\theta_i) &\leq \left\langle \theta_{\text{Add}}^T - \theta_i, \frac{\partial \mathcal{L}_i(x, \theta_i)}{\partial \theta} \right\rangle + \frac{L_i}{2} \left\| \theta_{\text{Add}}^T - \theta_i \right\|^2 && \text{(Assumption B.3)} \\
&= \left\langle \sum_{j=1}^{T} \alpha_j \tau_j - \tau_i, \cancel{\frac{\partial \mathcal{L}_i(x, \theta_i)}{\partial \theta}} \right\rangle + \frac{L_i}{2} \left\| \sum_{j=1}^{T} \alpha_j \tau_j - \tau_i \right\|^2 && \text{(Assumption B.2)}
\end{aligned}$$

To bound the second norm term, we reassign the subscript $\alpha_k := \alpha_i$ as the coefficient for the i-th task to avoid confusion with the summation indices. Next we define $t_j := \alpha_j \ (j \neq k), t_k := \alpha_k - 1$. Since $\sum_i \alpha_i = 1$, we have $t_k + \sum_{j \neq k} t_j = 0$. Then,

$$\begin{aligned}
\left\| \sum_{j=1}^{T} \alpha_j \tau_j - \tau_i \right\|^2 &= \left\| \sum_{j=1}^{T} \alpha_j \tau_j - \tau_k \right\|^2 \\
&= \left\| \sum_{i=1}^{n} t_i \tau_i \right\|^2 \\
&\leq \sum_{i=1}^{n} {t_i}^2 \tau_i^2 + 2 \sum_{1 \leq i < j \leq n} |t_i t_j| \cdot |\langle \tau_i, \tau_j \rangle| \\
&\leq C \left(\sum_{i=1}^{n} t_i^2 + 2\epsilon \sum_{1 \leq i < j \leq n} |t_i t_j| \right) \qquad (17)
\end{aligned}$$

By Assumption B.4, $t_k \leq 0$. Besides, we have $-t_k = t_1 + \cdots + t_{k-1} + t_{k+1} + \cdots + t_n = \sum_{i \neq k} t_i$. We can bound the cross-product term as follows:

$$\begin{aligned}
\sum_{1 \leq i < j \leq n} |t_i t_j| &= -t_k \sum_{i \neq k} |t_i| + \sum_{1 \leq i < j \leq n, i \neq k, j \neq k} t_i t_j \\
&= t_k^2 + \sum_{1 \leq i < j \leq n, i \neq k, j \neq k} t_i t_j \\
&= t_k^2 + \frac{1}{2} \left(\left(\sum_{i \neq k} t_i \right)^2 - \sum_{i \neq k} t_i^2 \right) \\
&= t_k^2 + \frac{1}{2} \left(t_k^2 - \left(\sum_i t_i^2 - t_k^2 \right) \right) \\
&= 2t_k^2 - \frac{1}{2} \sum_i t_i^2
\end{aligned}$$

Plug it back into Eq. (17), we have

$$\begin{aligned}
\sum_i t_i^2 + 2\epsilon \sum_{1\le i<j\le n} |t_i t_j| &= \sum_i t_i^2 + 2\epsilon \left(2t_k^2 - \frac{1}{2}\sum_i t_i^2\right) \\
&= (1-\epsilon)\sum_i t_i^2 + 4\epsilon t_k^2 \\
&= (1-\epsilon)\left[\sum_{i\neq k} \alpha_i^2 + (\alpha_k - 1)^2\right] + 4\epsilon(\alpha_k - 1)^2 \\
&= (1-\epsilon)\left[\sum_i \alpha_i^2 - 2\alpha_k + 1\right] + 4\epsilon(\alpha_k - 1)^2 \\
&\le (1-\epsilon)\cdot 2 + 4\epsilon \cdot 1 && \text{(Assumption B.4)} \\
&= 2 + 2\epsilon
\end{aligned}$$

To conclude, we have

$$\mathcal{L}_i(\theta_{\text{Add}}^T) - \mathcal{L}_i(\theta_i) \le \frac{L_i C}{2}(2+2\epsilon) = L_i C(1+\epsilon). \qquad \square$$

Thm. B.5 shows that as long as the task vectors reside in the fine-tuning regime and task vectors are dissimilar enough, then a single model obtained by model merging simultaneously performs comparably well on all the tasks. The local smoothness constant L_i in the generalization bound implies that a flatter minima is preferred in model merging (Iurada et al., 2025; Lee et al., 2025), which also related to the Fisher weighted averaging method (Matena & Raffel, 2022) as $\mathbf{H}$ agrees with the Fisher information matrix when ℓ is the cross-entropy loss which is a log-likelihood.

The following corollary shows that the softmax-activated Autoencoder bases formulation will not introduce additional performance gap in the upper bound of vector addition.

Corollary B.6 (Basis addition reduces to task addition)**.** *Let $B_m = \sum_{j=1}^T \mathbf{W}_e[j,m]\tau_j$ as defined in Eq.* (8)*. Define the basis-merged model $\theta_{\text{Add}}^M = \theta_0 + \sum_{m=1}^M \alpha_m B_m$. For every $i \in [T]$,*

$$\mathcal{L}_i(\theta_{\text{Add}}^M) - \mathcal{L}_i(\theta_i) \;\le\; L_i\, C\,(1+\epsilon). \tag{18}$$

Therefore, bases addition share the same generalization bound as standard task addition.

Proof. Define effective task weights $\phi_j = \sum_{m=1}^M \alpha_m \mathbf{W}_e[j,m]$. Since $\alpha \in \Delta^{M-1}$ and each $\mathbf{W}_e[:,m] \in \Delta^{T-1}$, we have $\phi \in \Delta^{T-1}$ ($\phi_j \ge 0$ and $\sum_{j=1}^T \phi_j = 1$). Hence

$$\theta_{\text{Add}}^M - \theta_0 = \sum_{m=1}^M \alpha_m \sum_{j=1}^T \mathbf{W}_e[j,m]\tau_j = \sum_{j=1}^T \left(\sum_{m=1}^M \alpha_m \mathbf{W}_e[j,m]\right)\tau_j = \sum_{j=1}^T \phi_j \tau_j,$$

so θ_{Add}^M is a convex combination of task vectors.

By Assumption B.2, $\|\tau_j\|^2 \le C$. Using $\phi \in \Delta^{T-1}$,

$$\|\theta_{\text{Add}}^M - \theta_0\| = \Big\|\sum_j \phi_j \tau_j\Big\| \le \sum_j \phi_j \|\tau_j\| \le \sqrt{C},$$

so $\|\theta_{\text{Add}}^M - \theta_0\|^2 \le C$, placing θ_{Add}^M in the local region where Assumption B.3 applies.

The claim follows immediately from Thm. B.5. $\square$

B.2.3 OOD Generalization with Task Vectors & Bases

Though similar tasks represented by ϵ may hurt addition and negation (see Thm. B.9), it is possible to achieve OOD generalization given insights from (Tripuraneni et al., 2020; Hu et al., 2024b).

Theorem B.7 (Out-of-Distribution Generalization)**.** *Given a collection of source task vectors $\mathcal{S} = \{\tau_1, \tau_2, \ldots, \tau_T\}$ and a target task vector with $\|\tau_{\text{tar}}\|^2 \leq C$. If $\exists i \in [T]$ such that $\langle \tau_{\text{tar}}, \tau_i \rangle \geq \gamma C$ for $0 < \gamma \leq 1$, then there exists a merging scheme $\alpha_i, i \in [T]$ such that for the merged model $\theta^T_{\text{Add}} = \theta_0 + \sum_{i=1}^T \alpha_i \tau_i$,*

$$\mathcal{L}_{\text{tar}}(\theta^T_{\text{Add}}) \leq \mathcal{L}_{\text{tar}}(\theta_{\text{tar}}) + L_{\text{tar}} C(1-\gamma). \tag{19}$$

Proof. Let $i^* = \arg\max_{i \in [T]} \langle \tau_{\text{tar}}, \tau_i \rangle$ and choose $\alpha_{i^*} = 1$, $\alpha_j = 0$, $\forall j \neq i^*$. Clearly $\langle \tau_{\text{tar}}, \tau_{i^*} \rangle \geq \beta C$ and $\theta^T_{\text{Add}} = \theta_0 + \tau_{i^*}$. It is also easy to check that $\|\theta^T_{\text{Add}} - \theta_{\text{tar}}\| \leq 4C$. So by the local smoothness assumption of $\mathcal{L}_{\text{tar}}$, we have

$$\begin{aligned}
\mathcal{L}_{\text{tar}}(\theta^T_{\text{Add}}) - \mathcal{L}_{\text{tar}}(\theta_{\text{tar}}) &\leq \frac{L_{\text{tar}}}{2} \|\theta^T_{\text{Add}} - \theta_{\text{tar}}\|^2 \\
&= \frac{L_{\text{tar}}}{2} \|\tau_{i^*} - \tau_{\text{tar}}\|^2 \\
&\leq \frac{L_{\text{tar}}}{2} \left(C - 2\langle \tau_{\text{tar}}, \tau_{i^*} \rangle + C\right) \\
&\leq L_{\text{tar}} C(1-\gamma).
\end{aligned}$$

□

This implies when γ, which roughly corresponds to the similarity of the two task vectors, is large enough, the gap between $\mathcal{L}_{\text{tar}}(\theta^T_{\text{Add}})$ and $\mathcal{L}_{\text{tar}}(\theta_{\text{tar}})$ is small, so we can use the combination of similar task vectors to achieve similar generalization performance for tasks that are OOD w.r.t. the source models.

Theorem B.8 (OOD generalization via basis atoms)**.** *Assume the target task vector τ_{tar} and all source tasks are nonnegatively aligned, i.e., $\langle \tau_{\text{tar}}, \tau_i \rangle \geq 0$ for every $i \in [T]$, and $\exists m^* \in [M]$ such that $\mathbf{W}_e[i^*, m^*] \geq \rho$ for some $\rho \in (0, 1]$ where $i^* = \arg\max_{i \in [T]} \langle \tau_{\text{tar}}, \tau_i \rangle$. Denote $\theta^M_{\text{Add}} = \theta_0 + \sum_{m=1}^M \alpha_m B_m$ with $\alpha \in \Delta^{M-1}$ and $B_m = \sum_{j=1}^T \mathbf{W}_e[j, m] \tau_j$ as defined in Eq. (8), then there exists a choice of α such that*

$$\mathcal{L}_{\text{tar}}(\theta^M_{\text{Add}}) \;\leq\; \mathcal{L}_{\text{tar}}(\theta_{\text{tar}}) \;+\; L_{\text{tar}}\, C \left(1 - \rho\, \gamma\right). \tag{20}$$

Proof. First note that the existential coverage condition for m^* is always satisfied for softmax encoders by picking $\rho = \max_{m \in [M]} \mathbf{W}_e[i^*, m]$.

Now consider the construction with $\alpha_{m^*} = 1$ and $\alpha_m = 0$ for $m \neq m^*$, so that $\theta^M_{\text{Add}} = \theta_0 + B_{m^*}$.

Since $\|B_{m^*}\|^2 = \|\sum_{j=1}^T \mathbf{W}_e[j, m^*] \tau_j\|^2 \leq C$, it is easy to see $\|\theta^M_{\text{Add}} - \theta_{\text{tar}}\| \leq 4C$. So by the local smoothness assumption of $\mathcal{L}_{\text{tar}}$, we have

$$\mathcal{L}_{\text{tar}}(\theta^M_{\text{Add}}) - \mathcal{L}_{\text{tar}}(\theta_{\text{tar}}) \;\leq\; \frac{L_{\text{tar}}}{2} \|\theta^M_{\text{Add}} - \theta_{\text{tar}}\|^2 \;=\; \frac{L_{\text{tar}}}{2} \|B_{m^*} - \tau_{\text{tar}}\|^2.$$

Expand and bound the quadratic term:

$$\|B_{m^*} - \tau_{\text{tar}}\|^2 = \|B_{m^*}\|^2 + \|\tau_{\text{tar}}\|^2 - 2\langle \tau_{\text{tar}}, B_{m^*} \rangle \;\leq\; 2C - 2\langle \tau_{\text{tar}}, B_{m^*} \rangle,$$

using $\|B_{m^*}\| \leq \sqrt{C}$ and $\|\tau_{\text{tar}}\| \leq \sqrt{C}$, now,

$$\begin{aligned}
\langle \tau_{\text{tar}}, B_{m^*} \rangle &= \sum_{j=1}^T \mathbf{W}_e[j, m] \, \langle \tau_{\text{tar}}, \tau_j \rangle \\
&\geq \mathbf{W}_e[i^*, m^*] \, \langle \tau_{\text{tar}}, \tau_{i^*} \rangle && \text{(nonnegative source target alignment)} \\
&\geq \rho \gamma C
\end{aligned}$$

Therefore,

$$\|B_{m^*} - \tau_{\text{tar}}\|^2 \;\le\; 2C - 2\rho\gamma C \;=\; 2C(1-\rho\gamma),$$

and hence

$$\mathcal{L}_{\text{tar}}(\theta^M_{\text{Add}}) - \mathcal{L}_{\text{tar}}(\theta_{\text{tar}}) \;\le\; \frac{L_{\text{tar}}}{2}\,2C(1-\rho\gamma) \;=\; L_{\text{tar}}\,C\,(1-\rho\gamma).$$

□

When the softmax encoder is annealed with smaller τ and columns of $\mathbf{B}$ converges to one-hot, $\rho \approx 1$ and the bound recovers the original $L_{\text{tar}}C(1-\beta)$ rate.

B.2.4 Task Negation & Basis Negation

Theorem B.9 (Task Negation for Unlearning)**.** *Let $0 < \epsilon \le 1$ be a universal constant such that $\forall i \neq j, |\cos(\tau_i, \tau_j)| \le \epsilon$, and $\theta_{\text{Neg,i}} = \theta_0 - \alpha_i \tau_i$ be the model parameter used for unlearning task i. Then $\forall j \neq i$,*

$$\mathcal{L}_j(\theta_{\text{Neg,i}}) - \mathcal{L}_j(\theta_0) \le L_j C\left(\frac{3}{2} + \epsilon\right). \tag{21}$$

Proof. First, note that

$$\mathcal{L}_j(\theta_{\text{Neg,i}}) - \mathcal{L}_j(\theta_0) \le \mathcal{L}_j(\theta_{\text{Neg,i}}) - \mathcal{L}_j(\theta_j) + \mathcal{L}_j(\theta_j) - \mathcal{L}_j(\theta_0) \le \mathcal{L}_j(\theta_{\text{Neg,i}}) - \mathcal{L}_j(\theta_j) + |\mathcal{L}_j(\theta_0) - \mathcal{L}_j(\theta_j)|$$

We will upper bound the last two terms separately. To bound $\mathcal{L}_j(\theta_{\text{Neg,i}}) - \mathcal{L}_j(\theta_j)$, note that $\|\theta_{\text{Neg,i}} - \theta_j\|^2 = \|\alpha_i\tau_i + \tau_j\|^2 \le (\alpha_i\|\tau_i\| + \|\tau_j\|)^2 \le 4C$, due to the local smoothness of $\mathcal{L}_j$ around θ_j and the fact that $\partial\mathcal{L}_j(\theta_j)/\partial\theta = \mathbf{0}$, we have

$$\mathcal{L}_j(\theta_{\text{Neg,i}}) - \mathcal{L}_j(\theta_j) \le \frac{L_j}{2}\|\theta_{\text{Neg,i}} - \theta_j\|^2 = \frac{L_j}{2}\|\alpha_i\tau_i + \tau_j\|^2 \le \frac{L_j}{2}\left(\alpha_i^2 C + 2\langle\tau_i, \tau_j\rangle + C\right) \le L_j C(1+\epsilon).$$

Similarly, for the second term, we have

$$|\mathcal{L}_j(\theta_0) - \mathcal{L}_j(\theta_j)| \le \frac{L_j}{2}\|\theta_0 - \theta_j\|^2 \le \frac{L_j C}{2}.$$

Combine both above, leading to

$$\mathcal{L}_j(\theta_{\text{Neg,i}}) - \mathcal{L}_j(\theta_0) \le L_j C(1+\epsilon) + \frac{L_j C}{2} = L_j C\left(\frac{3}{2} + \epsilon\right),$$

as desired. □

Since C is small due to fine-tuning, $\mathcal{L}_j(\theta_{\text{Neg,i}}) \approx \mathcal{L}_j(\theta_0)$, which means that the negation of a task for forgetting will not adversely impact the performance of other unrelated tasks, which has been shown empirically in (Ilharco et al., 2022), in contrast to other classic unlearning methods like gradient ascent.

Theorem B.10 (Basis Negation for Unlearning)**.** *Let $\hat{\mathbf{T}}$ be the task vector matrix reconstructed by basis vectors defined by Eq.* (10)*, and write $\widehat{\tau}_i$ for its i-th column and per column reconstruction error $e_i := \widehat{\tau}_i - \tau_i$. Define the negation model $\theta_{\text{Neg},i} := \theta_0 - \alpha_i\,\widehat{\tau}_i$. Then for every $j \neq i$,*

$$\mathcal{L}_j(\theta_{\text{Neg},i}) - \mathcal{L}_j(\theta_0) \;\le\; L_j C\left(\frac{5}{2} + 2\epsilon\right) + L_j\|e_i\|_2^2. \tag{22}$$

Proof. Decompose as in the original proof of Thm. B.9:

$$\mathcal{L}_j(\theta_{\text{Neg},i}) - \mathcal{L}_j(\theta_0) \le \big(\mathcal{L}_j(\theta_{\text{Neg},i}) - \mathcal{L}_j(\theta_j)\big) + \big|\mathcal{L}_j(\theta_0) - \mathcal{L}_j(\theta_j)\big|.$$

By Assumption B.3 at θ_j and $\nabla\mathcal{L}_j(\theta_j) = \mathbf{0}$ from Assumption B.2,

$$\mathcal{L}_j(\theta_{\text{Neg},i}) - \mathcal{L}_j(\theta_j) \;\le\; \frac{L_j}{2}\left\|\theta_{\text{Neg},i} - \theta_j\right\|^2 = \frac{L_j}{2}\left\|\alpha_i(\tau_i + e_i) + \tau_j\right\|^2.$$

Expand the square:

$$\begin{aligned}
\frac{L_j}{2}\left\|\alpha_i(\tau_i + e_i) + \tau_j\right\|^2 &= \frac{L_j}{2}\left\|(\alpha_i\tau_i + \tau_j) + \alpha_i e_i\right\|^2 \\
&\le \frac{L_j}{2}\left(\|\alpha_i\tau_i + \tau_j\| + \|e_i\|\right)^2 && (\alpha_i \le 1) \\
&\le L_j\big(\underbrace{\|\alpha_i\tau_i + \tau_j\|^2}_{(\star)} + \|e_i\|^2\big) && ((a+b)^2 \le 2(a^2+b^2))
\end{aligned}$$

The baseline term $(\star)$ is the same as in the original task vector negation proof in Thm. B.9, where it yields $L_j\,(\star) \;\le\; 2L_jC(1+\epsilon)$.

Finally, as in Thm. B.9, $\left|\mathcal{L}_j(\theta_0) - \mathcal{L}_j(\theta_j)\right| \le \frac{L_j}{2}\|\theta_0 - \theta_j\|^2 \le \frac{L_jC}{2}$. Summing the two parts completes the bound. □

Compared to the original negation bound $L_jC(\frac{3}{2} + \epsilon)$, the difference is increased constants for L_jC term another term only depending on the reconstruction error $\|e_i\|$. If the autoencoder (or PCA) bases reconstructs the i-th task vector well (small $\|e_i\|$), the penalty is negligible, recovering the original guarantee.

Corollary B.11 (Optimal Autoencoder Bases Negation)**.** *Let the error matrix be* $\mathbf{E} := \hat{\mathbf{T}} - \mathbf{T}$ *and suppose the trained autoencoder attains the spectral lower bound in Eq.* (7)*. Then for all* $j \ne i$*,*

$$\mathcal{L}_j(\theta_{\text{Neg},i}) - \mathcal{L}_j(\theta_0) \;\le\; L_jC\left(\frac{5}{2} + 2\epsilon\right) \;+\; L_j\lambda_{M+1}(\mathbf{G}). \tag{23}$$

Proof. Since the autoencoder attains the spectral lower bound in Eq. (7), we have

$$\|\mathbf{E}\|_2^2 \;=\; \lambda_{M+1}(\mathbf{G}).$$

For the i-th column error $e_i = \mathbf{E}\,\mathbf{e}_i$ (with $\mathbf{e}_i$ the i-th standard basis vector),

$$\|e_i\|_2^2 \;=\; \|\mathbf{E}\mathbf{e}_i\|_2^2 \;\le\; \|\mathbf{E}\|_2^2\,\|\mathbf{e}_i\|_2^2 \;=\; \|\mathbf{E}\|_2^2 \;=\; \lambda_{M+1}(\mathbf{G}).$$

Substituting $\|e_i\|_2^2 \le \lambda_{M+1}(\mathbf{G})$ into the bound from Thm. B.10 yields the desired upper bound as claimed. □

C Additional examples of adapting task vector methods to bases

1-to-1 task dataset correspondence. Both Fisher weighting (Matena & Raffel, 2022) and RegMean (Jin et al., 2022) also implicitly assume a one-to-one mapping between each task dataset D_i and its task vector τ_i (or equivalently fine tuned model θ_i). For Fisher merging, this is because each task contributes its own Fisher matrix $\mathbf{F}_i$ computed on validation data from D_i. For RegMean, each task contributes a validation covariance $\mathbf{X}_i^\top\mathbf{X}_i$, again computed directly from D_i. Thus, in the original T tasks setting, every task has its own dataset-level statistics that align exactly with its task vector.

In the basis setting, this direct correspondence is broken: bases are mixtures of tasks rather than stand-alone vectors. To preserve compatibility, we follow the same idea as in our adaptation of Localize-and-Stitch in Sec. 3.2.2. Specifically, we construct basis-level validation statistics by taking encoder-weighted combinations of the original per-task quantities. For Fisher, this yields a basis-level Fisher $\widehat{\mathbf{F}}_m = \sum_{i=1}^{T}\mathbf{W}_e[i,m]\,\mathbf{F}_i$ that aggregates information from all tasks according to their encoder weights. For RegMean, the per-task

covariance matrices are aggregated into $\widetilde{\mathbf{G}}_m = \sum_{i=1}^{T} \mathbf{W}_e[i,m]\, \mathbf{X}_i^\top \mathbf{X}_i$. These mixtures define new per-basis statistics, which can then be used in the same closed-form merging rules as the original methods. Notably, under equal dataset sizes $n_i = n$, computing per-task statistics directly on the subsampled mixed basis data $\widetilde{D}_m$ (with $n_i \cdot M/T$ examples per task, giving nM total effective samples instead of nT) is equivalent to the weighted sum formulation by linearity of expectation. This means that in the basis setting, both Fisher and RegMean only require M statistic computations on nM total data instead of T computations on nT data.

Others. For other methods, adapting task-vector methods to a basis representation does not change their mathematical form: Task Arithmetic (Ilharco et al., 2022) and TIES (Yadav et al., 2024) still involve coefficient searching over nonnegative coefficients using validation data, while AdaMerging (Yang et al., 2023) and aTLAS (Zhang et al., 2024) continue to solve unconstrained coefficient-learning problems either on a small unlabeled or labeled dataset. Similarly, WEMoE (Tang et al., 2024) routes MLP modules via a learned router while merging non-MLP modules standardly; since bases preserve the same module structure, it applies directly. The only modifications are (i) replacing task vectors τ_i with bases B_m, and (ii) replacing D_t with subsampled datasets $\widetilde{D}_t$. Each method incurs a per-vector processing cost $c_{\mathcal{M}}$ that scales from T to M in the basis setting.

D Empirical Verification of Theoretical Results

We primarily focus on verifying task addition Thm. B.5 where $M = 100\%$ and examine the relationship between the loss gap and key constants throughout our theoretical statements in Sec. B.2.

D.1 Inspection of Key Constants

Table 6: Ratio and Norm Task Vector for Different Models and Datasets

Model	**Dataset**	**Ratio%**	**Norm Task Vector**
laion2b_e16	MNIST	0.46	2.18
	EuroSAT	0.45	2.16
	Cars	0.54	2.52
	DTD	0.39	1.81
	GTSRB	0.49	2.32
	RESISC45	0.54	2.55
	SUN397	0.65	3.03
	SVHN	0.56	2.64
laion2b_s34b_b79k	MNIST	0.42	2.30
	EuroSAT	0.42	2.27
	Cars	0.48	2.59
	DTD	0.33	1.79
	GTSRB	0.45	2.44
	RESISC45	0.48	2.63
	SUN397	0.58	3.18
	SVHN	0.51	2.76

Task Vector Norm C Two openclip checkpoints are details can be found from `https://github.com/mlfoundations/open_clip/blob/main/docs/PRETRAINED.md` where laion2b_s34b_b79k is reported to be trained with larger batch size and learning rate, while two models share the same training data LAION-2B (Schuhmann et al., 2022). In Tab. 6, we reported the task vector norm and the ratio of task vector norm over the pretrained model norm, which is very small across datasets and models.

We elaborate on the connection of small task vector norm C requirement with previous literature. (Ilharco et al., 2022) in its Figure 7 demonstrates that the performance of merging task vectors derived from intermediate checkpoints, far before model convergence, is close to the performance of merging converged task

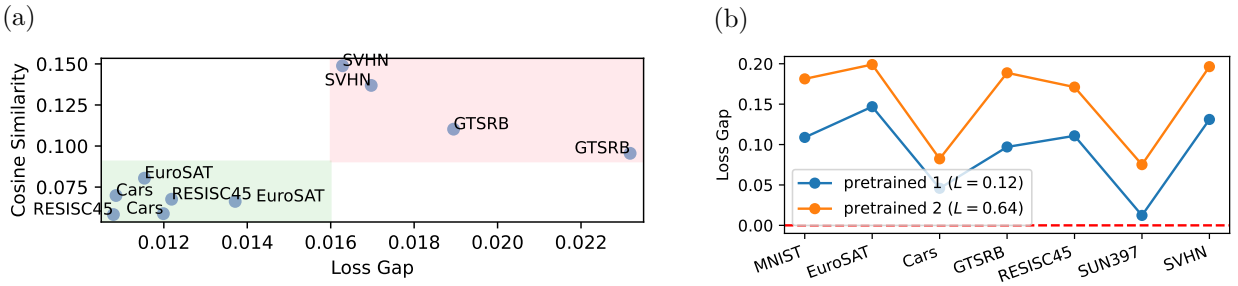


Figure 5: (a) Task vector similarity vs. $\mathcal{L}_{\text{MNIST}}(\theta^2_{\text{Add}}) - \mathcal{L}_{\text{MNIST}}(\theta_{\text{MNIST}})$, where $\theta^2_{\text{Add}} = \theta_0 + 0.5\tau_{\text{MNIST}} + 0.5\tau_{\text{task}}$. This figure includes two different set of CLIP ViT/B-32 task vectors. The pink shade includes the high similarity high loss gap region, and the green shade is the low similarity low loss gap region. This implies larger task similarity ϵ is harmful for addition. (b) $\mathcal{L}_{\text{DTD}}(\theta^2_{\text{Add}}) - \mathcal{L}_{\text{DTD}}(\theta_{\text{DTD}})$ by merging τ_{DTD} with other task vectors, setting scaling coefficient as 0.5. Two colored pretrained checkpoints have different local smoothness values.

vectors. These intermediate checkpoints typically have smaller norms due to fewer optimization steps, so a small C appears sufficient for the success of task addition. On the other hand, Figure 6 of (Ilharco et al., 2022) also indicated that a smaller learning rate is more important for task addition than for standard single-task fine-tuning, which implies that a smaller C is also necessary.

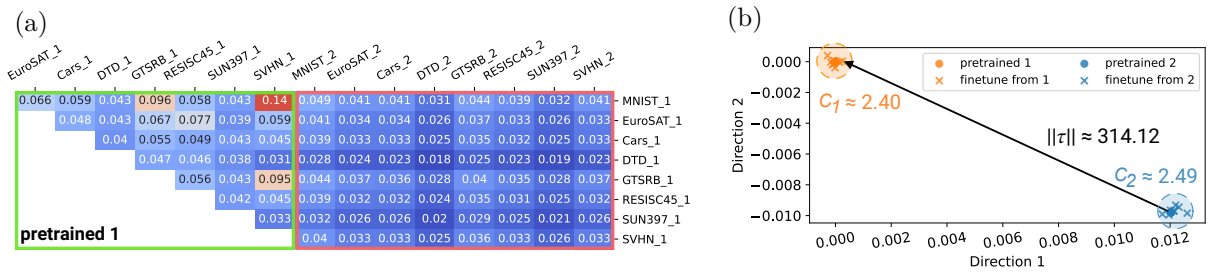


Figure 6: (a) Task vector similarity matrix for two checkpoints. The left green box represents the task similarity for vectors all derived from fine-tuning pretrained model 1. The right pink box represents the similarity values for task vectors from two different checkpoints, which corresponds to small ϵ in the 50/50 row of Tab. 7. (b) Task vector norms in different settings. Since the distance of two pretrained models are much larger than the distance between the pretrained model and their own fine tuned model , in Tab. 7 if we subtract pretrained 2 from any finetune 1, $\|\tau\|$'s upper bound C is huge, leading to the merging failure. For visualization purpose, we show two randomly selected dimensions, but the numbers for $C_1, C_2, \|\tau\|$ are directly computed from high-dimensional vectors.

Task Vector Similarity ϵ To verify how task vector similarity ϵ impacts the performance, we conduct the experiment shown in Fig. 5a. We merge the MNIST task vector with each of the other task vectors, all having similar norms ranging from $[2, 3)$ (see Tab. 6), and set the scaling coefficient α to be 0.5. In this setting, we approximately control all constants in Thm. B.5, including L, α, and C, and observe that highly similar tasks, such as digit classification in MNIST, SVHN, and GTSRB, lead to larger loss gaps or worse performance for MNIST compared to less related tasks.

Interaction between C and ϵ We provide additional evidence that both C and ϵ must be constrained for the success of task vector addition. In Tab. 7, we collected task vectors for 8 tasks from two CLIP checkpoints pretrained with different hyperparameters. From this table, we observe that successful task addition reveals the identity of the task vectors. For optimal merging performance, we should only add task vectors fine-tuned from the same checkpoint, as any mixture of task vectors from different checkpoints will cause a significant performance drop. The above empirical observation consolidates our Assumption B.2 that task vectors should reside in the same fine-tuning regime. To elaborate, from Fig. 6a, although all ϵ values in the pink box are very low, task addition still fails due to the large C value. From Fig. 6b, we see

Table 7: Task vector mixing performance, which is the average of all task test accuracies evaluated with the merged model. Numbers 1 and 2 refer to the identities of the pretrained checkpoints. "50 / 50" represents the experiment where 50% of the own task vector is mixed with 50% of task vectors derived from the other pretrained model.

$\theta_i \setminus \theta_0$	pretrained 1	pretrained 2
finetune from 1	**70.83**	51.71
50 / 50	59.92	61.71
finetune from 2	54.21	**71.09**

that with different hyperparameters, the two pretrained models are situated in two local convex basins, and the distance between the two checkpoints is much larger than the task vectors (Tab. 6). Thus, if we create task vectors by subtracting the wrong pretrained checkpoint, the large C value leads to the failure of task addition.

Local Smoothness L The local smoothness L is specific to each pretrained model due to differences in their optimization trajectories. Since, as shown in Fig. 6, the differences in C and ϵ (not θ_i) between two pretrained models are small. In Fig. 5b, we merge the DTD task vector with each of the other task vectors and compare the loss gap between two checkpoints. Because it is not feasible to load $\mathbf{H}(\theta_i) \in \mathbb{R}^{d\times d}$ directly onto the GPU, we estimate L using the power iteration method (Mises & Pollaczek-Geiringer, 1929) to reduce the largest eigenvalue problem to a Hessian-vector product computation. As seen in Fig. 5b, larger local smoothness consistently leads to a larger gap from the optimal loss term across datasets, resulting in worse merging performance.

D.2 Empirical Loss Gap in Task Addition

We report the empirical loss gap using Task Arithmetic as the merging method on several downstream datasets using the ViT-B/16 backbone for CLIP. In Tab. 8, the loss gap is computed as the empirical difference between the loss of the merged model $\mathcal{L}_i(\theta_{\text{Add}})$ and the fine-tuned loss $\mathcal{L}_i(\theta_i)$ as in the left hand side of theorems in Sec. B.2.2.

Table 8: Empirical loss gap between the fine-tuned model and the TA-merged model for each task on ViT-B/16.

Task_i	$\mathcal{L}_i(\theta_{\text{Add}}) - \mathcal{L}_i(\theta_i)$
MNIST	0.12
EuroSAT	1.00
Cars	0.15
DTD	1.11
GTSRB	0.80
RESISC45	0.84
SUN397	5.02
SVHN	0.60

As shown in Table 8, the loss gap remains small for most tasks except SUN397, which reaches a gap of 5.02. This demonstrates that although task arithmetic (Ilharco et al., 2022) is a widely used model merging baseline, it can suffer significant performance degradation on certain tasks. Importantly, these empirical loss gaps are *not* computed from the constants appearing in the right-hand side of our theoretical upper bound. When substituting reasonable values, such as $L = 1$, $C = 4$, and $\epsilon = 0.2$, the upper bound becomes $LC(1 + \epsilon) \approx 5$, which approximates the largest observed gap. While this value may seem conservative for common losses like cross-entropy, it is necessary to account for the worst-case scenarios observed in practice.

This further supports the relevance of our theoretical framework in explaining and bounding the limitations of task arithmetic in heterogeneous settings.

E Experimental Details about Bases Construction

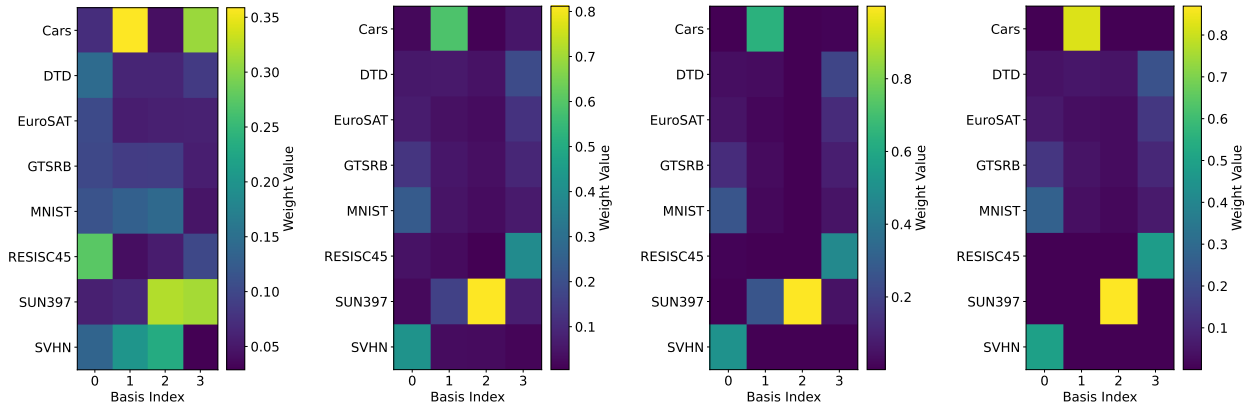
We construct task vector bases using a lightweight AE trained over the collection of task vectors with Adam optimizer (Kingma, 2014). All experiments are conducted on NVIDIA RTX A6000 GPUs with 48GB memory. Unless otherwise specified, we adopt the following default configuration for the AE:

$$M = 4,\ \text{steps} = 4000,\ \text{lr} = 0.01,\ \tau = 5.0,\ \text{weight decay} = 10^{-6}.$$

E.1 Sensitivity of temperature τ

A key hyperparameter in the encoder is τ, which controls the mixing effect of task vectors when constructing basis vectors with AE.

Fig. 7 shows the encoder weight distributions across different values of the temperature parameter τ. As τ decreases, the encoder weights become more selective, pushing the representation closer to a one-hot distribution. At $\tau = 5$, the weights are relatively diffuse, with several tasks contributing simultaneously to each basis. By contrast, at $\tau = 1$ and below, the encoder begins to isolate dominant contributors, and at $\tau = 0.8$ the assignments are nearly one-hot. At this low temperature the learned bases also become interpretable. For example, when $\tau = (500, 0.8)$, basis 1 captures GTSRB, MNIST, and SVHN, which can be interpreted as a digit classification group. Basis 1 and 2 are dominated by Cars and SUN397, which are classification problems very different from other task vectors in the pool. Basis 3 combines RESISC45, DTD, and EuroSAT, corresponding to texture and satellite imagery that are naturally linked through landscape and surface patterns. This progression illustrates that the softmax activation indeed yields semantically meaningful groupings of tasks.



(a) $\tau = 5$ (b) $\tau = 1$ (c) $\tau = (500, 0.9)$ (d) $\tau = (500, 0.8)$

Figure 7: Encoder weight $\mathbf{W}_e$ distributions across different values of the temperature parameter τ for ViT-B/32 8 tasks setting when $M = 4$. The notation $(500, \eta)$ refers to the annealing τ setting where every 500 steps the temperature τ is multiplied by a factor of η.

The addition performance results for each type of τ setup are shown in Tab. 9. On one hand, different τ choices can substantially affect downstream addition performance. In particular, although all settings drive the reconstruction loss close to the theoretical lower bound (note that lower values of τ require more steps to converge), the actual addition accuracies differ: for example, L&S accuracy vary by as much as 0.09 depending on the schedule. The performance does not grow monotonically with τ itself, and can vary across setups; for instance, the observed gap to the lower bound depends on the number of training steps, which in turn influences the final accuracy. Therefore, careful hyperparameter tuning of τ is still required

to achieve the best performance. But interestingly, we find that once tuned for one method on a fixed set of task vectors, the same τ schedule also transfers well across other evaluation metrics (TA results closely track L&S results). On the other hand, negation forgetting metrics are robust to different τ setups.

Table 9: Normalized offline addition accuracy and target forgetting accuracy of τ hyperparameter for ViT-B/32 on 8 tasks with $M = 4$ using AE bases. The spectral lower bound predicted by Eq. (7) is 0.336935.

τ	**Steps**	**TA$^+$**	**L&S$^+$**	**TA$^-$**	**Loss**
$(500, 0.8)$	30000	0.734	0.649	0.308	0.336935
$(500, 0.9)$	4000	0.722	0.640	0.306	0.338606
1	4000	0.736	0.707	0.308	0.336935
5	4000	0.744	0.733	0.307	0.336937

E.2 Runtime and Memory Analysis

We first compare the runtime and memory complexity of AE training with PCA for basis construction. Both methods require access to the full task matrix $\mathbf{T} \in \mathbb{R}^{d \times T}$, where d is the parameter dimension and T is the number of task vectors. Since typically $d \gg T$, the dominant storage cost for both methods is $O(dT)$. For time complexity, AE first requires one-time computation of the Gram matrix, which costs $O(dT^2)$. Subsequent training then involves $O(T^2)$ operations per optimization step. PCA basis construction is obtained through SVD of the $d \times T$ task matrix, with complexity $O(dT^2)$ too. Thus, in the worst-case analysis both AE and PCA have the same order of time and space complexity. The practical advantage of AE is that, as discussed in Lem. 3.2, after Gram computation we can remove the dependence of d during optimization, making potentially parallel gradient-based optimization feasible on modern GPUs.

We next empirically measure AE training time for different numbers of basis vectors M fixing other hyper-parameters. Results are summarized in Tab. 10. For addition experiments, the runtime decreases modestly with larger M, which we attribute to more efficient GPU utilization when matrix multiplications involve wider hidden dimensions. This effect is implementation-dependent but does not change the main conclusion that basis processing is almost negligible compared to merging time (see Fig. 2c). For negation, we can see that once the bases are constructed the reconstruction time is also minimal compared to the negation tuning time where we use the grid search over $[0, 0.05, 0.10, \ldots, 1.0]$ (21 grid points) for forgetting experiments.

Table 10: AE steps wall clock time (seconds) for different numbers of basis vectors M with 4000 gradient steps averaged across three runs on ViT-B/32 for 8 tasks. Basis Construction refers to the runtime of solving Eq. (5) and Task Vector Reconstruction refers to the runtime of Eq. (10) given saved bases. A detailed per phase breakdown including memory and IO profiling is provided in Tab. 11.

# Basis	**Basis Construction**	**Addition**	**Task Vector Reconstruction**	**Negation**
$M = 2$	28.96	10894.54	1.11	21187.95
$M = 4$	22.72	11059.96	2.12	21380.52
$M = 6$	17.09	11926.79	2.61	22792.77

E.3 Precision and Efficiency

To evaluate the interaction between numerical precision and basis compression, we run the full pipeline under FP32, FP16, and BF16 on ViT-B/16 with 8 tasks and $M = 4$ bases. Tab. 12 reports the results. Reducing precision to FP16 or BF16 halves storage from ∼1706 MB to ∼853 MB with no meaningful loss in accuracy for AE, demonstrating that basis compression is orthogonal to and composable with precision reduction as discussed in Sec. 5.

Table 11: Full pipeline profiling for ViT-B/32, $T = 8$ tasks, on NVIDIA RTX A6000. Wall clock times are consistent with Tab. 10; notably, the entire basis construction and reconstruction pipeline takes under 31s, which is negligible compared to the downstream addition (∼11000s) and negation (∼21000s) times reported in Tab. 10. CPU memory is reported as Resident Set Size (RSS), the physical memory occupied by the process. Phases that do not depend on M (loading $\mathbf{T}$, Gram computation, AE optimization) have constant memory cost.

(a) Wall clock time breakdown. Steps 1–2 are shared across all M.

	Phase	$M = 2$	$M = 4$	$M = 6$
Basis Constr.	1. Load $\mathbf{T}$		6.52s	
	2. Gram $\mathbf{G} = \mathbf{T}^\top\mathbf{T}$		2.08s	
	3. AE optimization (Eq. (6), 4k steps)	17.96s	9.19s	5.16s
	4. Compute $\mathbf{B} = \mathbf{T}\mathbf{W}_e$ (Eq. (8))	0.72s	0.81s	0.61s
	5. Save to disk	1.68s	4.12s	2.72s
	Subtotal (Basis Construction)	**28.96s**	**22.72s**	**17.09s**
Reconstr.	6. Reconstruct $\hat{\mathbf{T}} = \mathbf{B}\mathbf{W}_d$ (Eq. (10))	1.11s	2.12s	2.61s
	Total	**30.07s**	**24.84s**	**19.70s**

(b) CPU RSS and GPU memory.

	Phase	$M = 2$	$M = 4$	$M = 6$
Basis Constr.	1. Load $\mathbf{T}$		4308 MB	
	2. Gram $\mathbf{G}$		6 MB	
	3. AE optimization	544 MB	544 MB	544 MB
	4. Compute $\mathbf{B}$	865 MB	1730 MB	2595 MB
	5. Save to disk	0 MB	0 MB	0 MB
Reconstr.	6. Reconstruct $\hat{\mathbf{T}}$	4977 MB	4977 MB	4977 MB
	Peak RSS	**10.7 GB**	**11.6 GB**	**12.5 GB**
	GPU peak	0.01 MB	0.01 MB	0.01 MB

(c) IO artifact sizes.

Artifact	$M = 2$	$M = 4$	$M = 6$
$\mathbf{W}_e$	1 KB	2 KB	2 KB
$\mathbf{B}$	908 MB	1815 MB	2723 MB

Table 12: Effect of numerical precision on ViT-B/16, 8 tasks, $M = 4$ bases.

Prec	**Method**	**Storage (MB)**	**NormAcc**
FP32	AE	1705.99	0.717
	PCA	1705.87	0.538
	RandSelect	1705.99	0.695
FP16	AE	853.08	0.717
	PCA	852.96	0.540
	RandSelect	853.08	0.695
BF16	AE	853.08	0.717
	PCA	852.96	0.540
	RandSelect	853.08	0.695

F Experimental Details about Bases Arithmetic

F.1 Choice of Subsampling Strategy and Learning Weights

Subsampling. During subsampling, we select $n_i \cdot M/T$ examples per original dataset D_i (subsampling stratified by dataset), creating in total nM effective samples in $\cup_{i=1}^{T}\widetilde{D_i}$ compared to nT effective samples in $\cup_{i=1}^{T}D_i$ assuming size of T datasets are the same. In few-shot scenarios, such as the language experiments (Tab. 3) or OOD experiments (Tab. 4), we similarly adjust the dataset size to $k \cdot M/T$, thereby simulating the stratified subsampling effect. Tab. 13 compares addition results with and without subsampling under TA for ViT-B/32 with $M = 4$ and different τ schedules. The results show that subsampling has only a

Table 13: Effect of subsampling on TA addition accuracy for ViT-B/32 with $M = 50\%$ across different τ values (8 tasks). We report absolute average accuracy with and without subsampling.

τ	D_i	$\widetilde{D_i}$
(500, 0.8)	0.682	0.684
(500, 0.9)	0.667	0.667
1	0.682	0.681
5	0.689	0.689

minimal impact on accuracy, indicating that tuning on smaller, stratified subsets is sufficient to capture the relevant task relationships while reducing the overall computational cost.

Table 14: Effect of different weighting strategies for AE bases (ViT-B/32, 8 tasks, L&S, $M = 50\%$). We report absolute (normalized) accuracy and average L&S mask sparsity.

Weighting Strategy	Abs. (Norm.) Acc	Avg. Sparsity
Uniform	0.674 (0.727)	0.913
Fixed (random)	0.615 (0.669)	**0.944**
Encoder Weighted	**0.691 (0.732)**	0.910

Learning Weights. Table 14 compares different strategies for setting the basis weights when constructing AE representations. In the uniform case, each task contributes equally ($\mathbf{W}_e[i, m] = 1/T$), while in the fixed random case, each of M basis randomly selects one task to receive weight 1 (others 0). Finally, the encoder-weighted approach uses the learned encoder outputs $\mathbf{W}_e[:, m]$ to define the convex combination of tasks for each basis as in our proposed weighting strategy. We observe that encoder weighting yields the best accuracy under L&S, where accurate localization requires each basis to correspond to a meaningful mixture of tasks. This mapping allows the model to identify parameter regions that explain the combined tasks' performance more effectively than uniform or random weighting. The tradeoff is a slight reduction in sparsity, which modestly increases memory/storage overhead when using CSR format. Nonetheless, the gain in performance strongly supports the use of encoder weights in practice.

F.2 Offline Bases Addition

Datasets and Metrics. Define normalized accuracy as absolute accuracy normalized by single task fine tuned performance. For vision experiments, we report the classification accuracy on ViT models (Dosovitskiy, 2020) on Cars (Krause et al., 2013), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), GTSRB (Stallkamp et al., 2012), MNIST (LeCun et al., 1998), RESISC45 (Cheng et al., 2017), SVHN (Netzer et al., 2011), and SUN397 (Xiao et al., 2010) for the 8-task setting. For 14 tasks, we additionally include CIFAR100 (Krizhevsky et al., 2009), STL10 (Coates et al., 2011), Flowers102 (Nilsback & Zisserman, 2008), OxfordIIITPet (Parkhi et al., 2012), PCAM (Veeling et al., 2018), FER2013 (Goodfellow et al., 2013), and for 20 tasks, we further include EMNIST (Cohen et al., 2017), CIFAR10 (Krizhevsky et al., 2009), Food101 (Bossard et al., 2014), FashionMNIST (Xiao et al., 2017), RenderedSST2 (Socher et al., 2013) and KMNIST (Clanuwat et al., 2018). For language experiments, we use a 12-task benchmark Gao et al. (2020) with SST2 (Socher et al., 2013), CR (Hu & Liu, 2004), MR (Pang & Lee, 2004), MPQA (Wiebe et al., 2005), TREC (Li & Roth, 2002), SUBJ (Pang & Lee, 2004), QNLI (Wang, 2018), SNLI (Bowman et al., 2015), MNLI (Williams et al., 2017), RTE (Wang, 2018), MRPC (Dolan & Brockett, 2005) and QQP (Sharma et al., 2019) trained on RoBERTa (Liu, 2019) models. We report the F1 metric for MRPC.

Hyperparameters of Oracle Merging Methods. For vision experiments, both TA and TIES use validation data for hyperparameter tuning of the isotropic scaling coefficient α, with a grid search over $[0, 0.05, 0.10, \ldots, 1.0]$ (21 grid points). For TIES, we additionally set the top-k threshold to 20% and use

`sum` as the merging rule. For L&S, we use the following settings: sigmoid bias = 5, batch size = 16 for ViT-L/14 and 64 otherwise, ℓ_1 strength = 1, learning rate = 10^{-7}, 10 training epochs, and sparsity = 10^{-5}. For language experiments, we perform experiments on 64-shot datasets. TA and TIES adopt the same setup as in vision. For L&S, we use: sigmoid bias = 3, learning rate = 10^{-7}, ℓ_1 strength = 0, 10 training epochs, sparsity = 10^{-5}, and batch size = 8.

Supplementary Results. The normalized accuracy results in Tabs. 16 and 17 confirm that our main observations in Sec. 4.1.1 are not simply an artifact of single-task fine-tuning performance. Even after normalization, AE achieves the best overall results in general, with a clear ordering of AE > RandSelect > PCA across both vision and language tasks. For the TIES method, RandSelect becomes slightly more competitive, and in a few large-task settings it reaches performance on par with AE. The per-dataset results in Figs. 9 to 12 illustrate that AE exhibits a more pronounced advantage in terms of absolute accuracy under L&S, the strongest merging method we evaluate, and its relative advantage grows mildly as the number of tasks increases.

To further strengthen the PCA baseline, we experimented with a simple reweighting scheme in Tab. 16 that converts the task coefficients/PCA loadings $\mathbf{C} = \mathbf{V}_M^\top$ into convex combinations of tasks, denoted by PCA$^{\geq 0}$. For each component $m \in [M]$, let $v_m \in \mathbb{R}^T$ denote the corresponding column of $\mathbf{V}_M$. We define task weights by applying a softmax only to the strictly positive entries of v_m:

$$\mathbf{W}_e[i,m] = \begin{cases} \dfrac{\exp(v_{m,i})}{\sum_{j:v_{m,j}>0}\exp(v_{m,j})}, & v_{m,i} > 0, \\ 0, & \text{otherwise}, \end{cases}$$

The resulting per-basis weights $\mathbf{W}_e[:,m] \in \Delta^{T-1}$ ensure that each PCA basis vector can be interpreted as a convex combination of the positively aligned task vectors by completely discarding the negatively aligned ones, thus can be used as the loss weight while training binary masks for L&S. While this positive-softmax variant addresses the concern that our default option uniform PCA ignores the relative contribution of tasks, it still performs significantly worse than AE in practice.

Multi-seed Robustness. To assess the robustness of our results, we report mean and standard error across multiple seeds for L&S and L&S-AE in Tab. 15. The results confirm that the performance differences between full task vectors and compressed bases are consistent across runs.

Table 15: Multi-seed normalized accuracy (mean ± SE over 3 seeds) for L&S with full task vectors and L&S-AE with $M = 50\%$ bases across vision models and task counts. Underline indicates the better mean.

Model	Tasks	L&S	L&S-AE
ViT-B/16	8	<u>0.732</u> ± 0.019	0.667 ± 0.063
	14	0.680 ± 0.000	<u>0.686</u> ± 0.003
	20	0.597 ± 0.003	<u>0.657</u> ± 0.005
ViT-B/32	8	<u>0.742</u> ± 0.001	0.693 ± 0.022
	14	0.659 ± 0.001	<u>0.667</u> ± 0.006
	20	0.596 ± 0.000	<u>0.638</u> ± 0.003
ViT-L/14	8	<u>0.778</u> ± 0.001	0.723 ± 0.006
	14	<u>0.754</u> ± 0.001	0.750 ± 0.006
	20	0.701 ± 0.003	<u>0.720</u> ± 0.007

Performance Scaling with M. Fig. 8 plots the offline addition accuracy as the number of basis vectors M increases. We observe that the AE achieves consistently strong performance across all values of M, starting higher than both PCA and RandSelect and maintaining steady improvements as M grows. In particular, AE dominates PCA throughout the entire range, with a margin of more than 0.1 absolute accuracy at most values of M. RandSelect starts much lower but improves rapidly with larger M, eventually approaching and in some cases slightly surpassing AE when $M \geq 6$. This reflects that random bases can capture task

Table 16: Comparison of normalized addition accuracy across ViT models under 8, 14, and 20 vision task settings (Wang et al., 2024a) with bases number $M = 50\%$ of the total tasks.

Method	**ViT-B/16**			**ViT-B/32**			**ViT-L/14**		
	8 task	14 task	20 task	8 task	14 task	20 task	8 task	14 task	20 task
TA (Ilharco et al., 2022)	0.796	0.759	0.708	0.766	0.721	0.668	0.887	0.840	0.781
RandSelect	0.695	0.706	0.672	0.698	0.703	0.674	0.808	0.811	0.763
PCA	0.538	0.629	0.622	0.578	0.633	0.645	0.686	0.729	0.705
AE (Ours)	**0.717**	**0.729**	**0.687**	**0.744**	**0.725**	**0.676**	**0.847**	**0.840**	**0.783**
TIES (Yadav et al., 2024)	0.843	0.787	0.733	0.81	0.748	0.699	0.907	0.841	0.798
RandSelect	0.719	0.718	0.672	0.712	**0.717**	**0.689**	0.847	0.821	**0.779**
PCA	0.540	0.629	0.622	0.576	0.633	0.654	0.686	0.728	0.707
AE (Ours)	**0.724**	**0.728**	**0.687**	**0.741**	**0.717**	0.670	**0.854**	**0.841**	0.778
L&S (He et al., 2024)	0.794	0.721	0.641	0.811	0.698	0.642	0.809	0.796	0.748
RandSelect	0.575	0.569	0.465	0.580	0.527	0.467	0.696	0.722	0.643
PCA	0.540	0.482	0.440	0.489	0.438	0.389	0.692	0.629	0.581
PCA$^{\geq 0}$	0.541	0.455	0.397	0.489	0.447	0.393	0.692	0.582	0.579
AE (Ours)	**0.694**	**0.716**	**0.687**	**0.732**	**0.718**	**0.679**	**0.767**	**0.780**	**0.780**

Table 17: Comparison of normalized addition accuracy with RoBERTa-base model on 12 language task benchmark with bases number $M = 25\%$ of the total tasks.

TA (Ilharco et al., 2022)				**TIES** (Yadav et al., 2024)				**L&S** (He et al., 2024)			
100%	RandSelect	PCA	AE	100%	RandSelect	PCA	AE	100%	RandSelect	PCA	AE
0.777	0.570	0.565	**0.592**	0.744	0.568	**0.587**	**0.587**	0.936	0.769	0.763	**0.901**

diversity when many are available, but their effectiveness is more volatile and requires larger basis sizes to be competitive. In contrast, AE provides reliable accuracy gains even with small M, making it far more practical in settings where storage or computational budget limits the number of bases.

Results on Generative Large Language Models. In Tab. 18, we use the LLM model merging benchmark (He et al., 2025), evaluating 5 input task vectors per domain on 21 downstream tasks, providing a comprehensive test of LLM abilities with generative tasks. Using Llama-3.2-3B, we tested three basis compression algorithms with $M = 2$. Following He et al. (2025), we report normalized accuracy, defined as the absolute sum of a method's per-category average divided by the absolute sum of the supervised finetuning (SFT) per-category average. We also adopt the recommended scaling coefficient of 0.4 for all addition experiments. Across all settings, AE achieves the best overall performance, RandSelect ranks second, and PCA is consistently the worst. Notably, AE preserves 87% of the full task vector performance while using only 40% of the compute and storage, which is particularly important for scaling to larger foundation models.

Compatibility with QLoRA Task Vectors. To verify that Task Vector Bases compose with LoRA-based task vectors, we apply our framework to QLoRA (4-bit, Rank 16, Alpha 32) task vectors on ViT-B/16 with 8 tasks and $M = 4$ bases (Tab. 19). QLoRA task vectors trade accuracy for storage ($\sim$72$\times$ reduction). Among QLoRA settings, AE remains the best basis method and nearly matches TA (0.565 vs. 0.566). AE basis construction takes 18.25s (vs. 22.72s in Tab. 10), confirming that the two compression axes compose without additional overhead.

Softmax vs. Unconstrained Linear Encoder. To isolate the effect of the softmax constraint, we compare the default softmax activated autoencoder against an unconstrained linear encoder under the 8 task addition setting on ViT-B/16 with $M = 4$ bases. Tab. 20 reports per dataset normalized accuracy. The softmax encoder consistently outperforms the unconstrained linear encoder on 7 out of 8 datasets, with

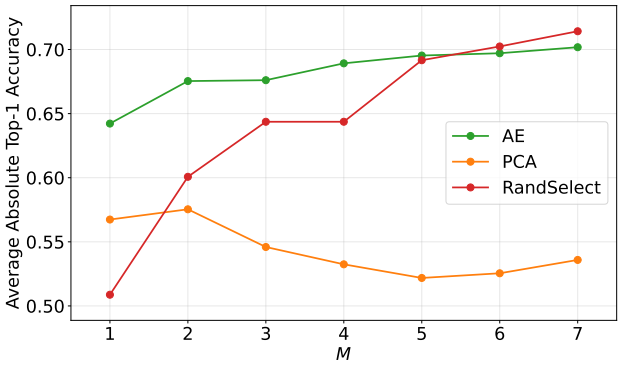
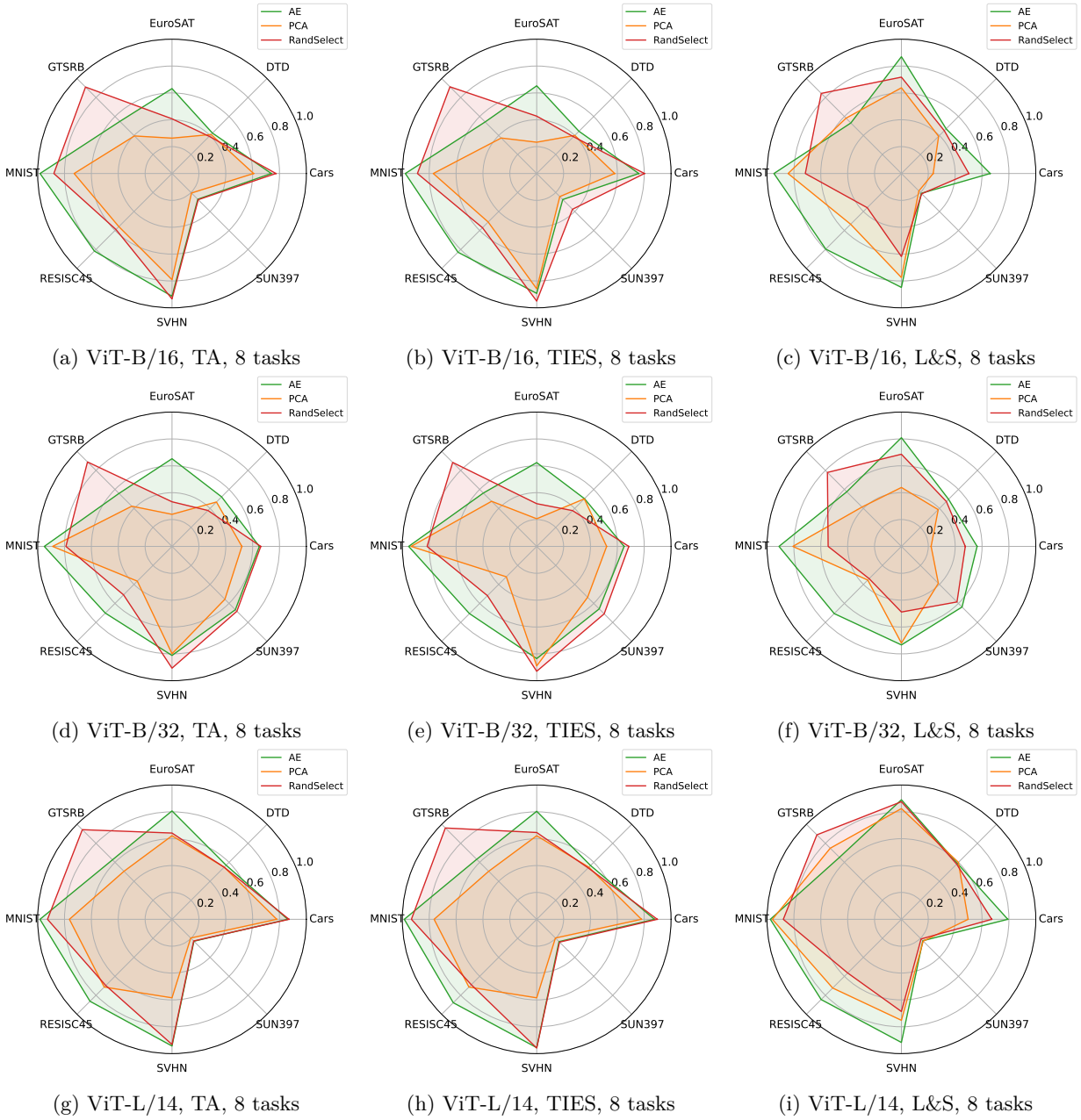


Figure 8: Offline addition accuracy as a function of the number of basis vectors M for different bases construction methods with ViT-B/32 on 8 tasks benchmark.

Table 18: Normalized accuracy across datasets for Llama-3.2-3B (Grattafiori et al., 2024) for 5 task vectors and various compression methods at $M = 40\%$. Best compression results of Task Arithmetic (TA) are bolded.

Category	**Dataset**	**SFT**	**TA**	**AE**	**PCA**	**RandSelect**
Instruction Following	IFEval (Zhou et al., 2023)	37.52	25.32	14.79	13.68	21.44
Math	GSM8k (Cobbe et al., 2021)	72.55	45.34	53.68	18.50	46.10
	MATH (Hendrycks et al., 2021)	33.04	10.14	22.50	0.88	16.80
Multilingual (fr, de, es, ru)	M_MMLU (Lai et al., 2023)	44.89	45.01	45.32	44.77	45.20
		43.61	44.54	48.04	46.27	47.80
		46.45	46.11	46.07	44.99	45.92
		41.80	41.57	42.56	42.20	45.92
	M_ARC (Lai et al., 2023)	40.89	40.46	29.94	31.48	31.14
		38.32	36.70	37.69	36.32	38.71
		40.09	41.54	35.07	33.70	35.41
		36.95	37.55	34.22	32.42	34.04
	M_Hellaswag (Lai et al., 2023)	58.67	59.56	42.09	42.40	42.71
		54.46	55.22	46.35	46.16	46.53
		60.07	61.21	43.89	43.80	44.46
		52.63	53.89	40.76	41.03	41.22
Coding	Humaneval+ (Chen et al., 2021)	41.83	36.71	28.96	16.46	28.84
	MBPP+ (Austin et al., 2021)	46.59	45.50	41.40	38.23	41.48
Safety	WildGuardTest (Han et al., 2024)	85.71	51.94	29.91	25.11	31.32
	HarmBench (Mazeika et al., 2024)	89.38	39.69	27.82	24.07	30.62
	DoAnythingNow (Shen et al., 2024)	90.67	32.67	31.61	27.67	23.67
	XSTest (Röttger et al., 2023)	37.56	60.22	46.89	42.22	47.11
Average Normalized Accuracy		100.00	72.88	**63.49**	47.07	63.38

an average improvement of 17.1 percentage points, validating that enforcing convex combination coefficients via softmax provides a substantial empirical benefit.



(a) ViT-B/16, TA, 8 tasks (b) ViT-B/16, TIES, 8 tasks (c) ViT-B/16, L&S, 8 tasks

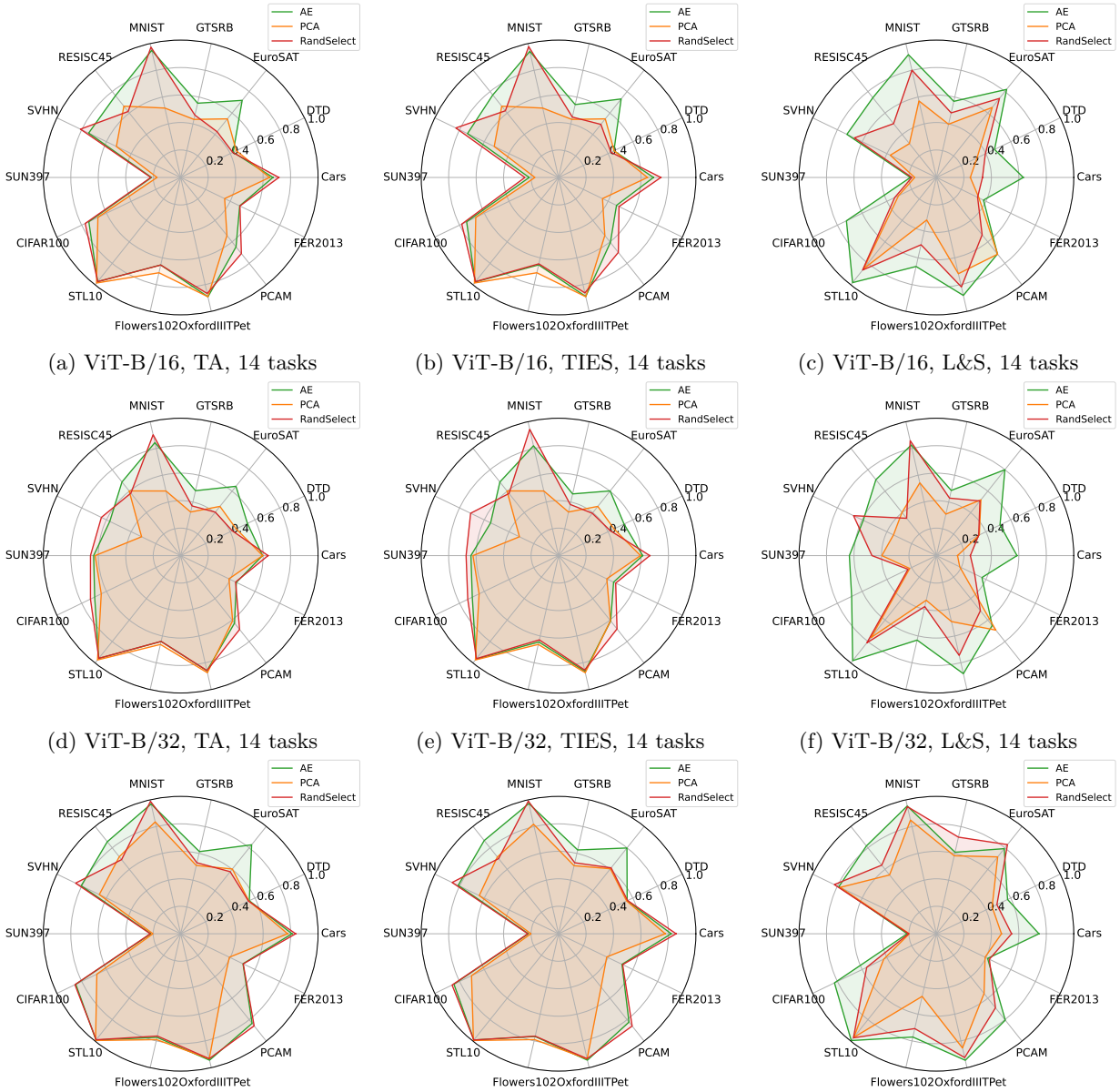
(d) ViT-B/32, TA, 8 tasks (e) ViT-B/32, TIES, 8 tasks (f) ViT-B/32, L&S, 8 tasks

(g) ViT-L/14, TA, 8 tasks (h) ViT-L/14, TIES, 8 tasks (i) ViT-L/14, L&S, 8 tasks

Figure 9: Per dataset bases comparison results for 8 vision tasks of Tab. 2.

F.3 Offline Fewshot OOD Generalization

Hyperparameters and Datasets. For the aTLAS method in the few-shot regime in Sec. 4.1.2, we train for 10 epochs on every target OOD dataset (CIFAR10, CIFAR100, STL10, Food101, Flowers102, OxfordIIITPet). We use the AdamW optimizer with a learning rate of 0.1 and weight decay of 0.1, together with a cosine learning-rate schedule that decays the learning rate from 0.1 to 0 over the course of training. The per-GPU batch size is set to 128 for all models except ViT-L/14, where it is 64 with gradient accumulation of 2, yielding an effective batch size of 128 in both cases.



(a) ViT-B/16, TA, 14 tasks (b) ViT-B/16, TIES, 14 tasks (c) ViT-B/16, L&S, 14 tasks

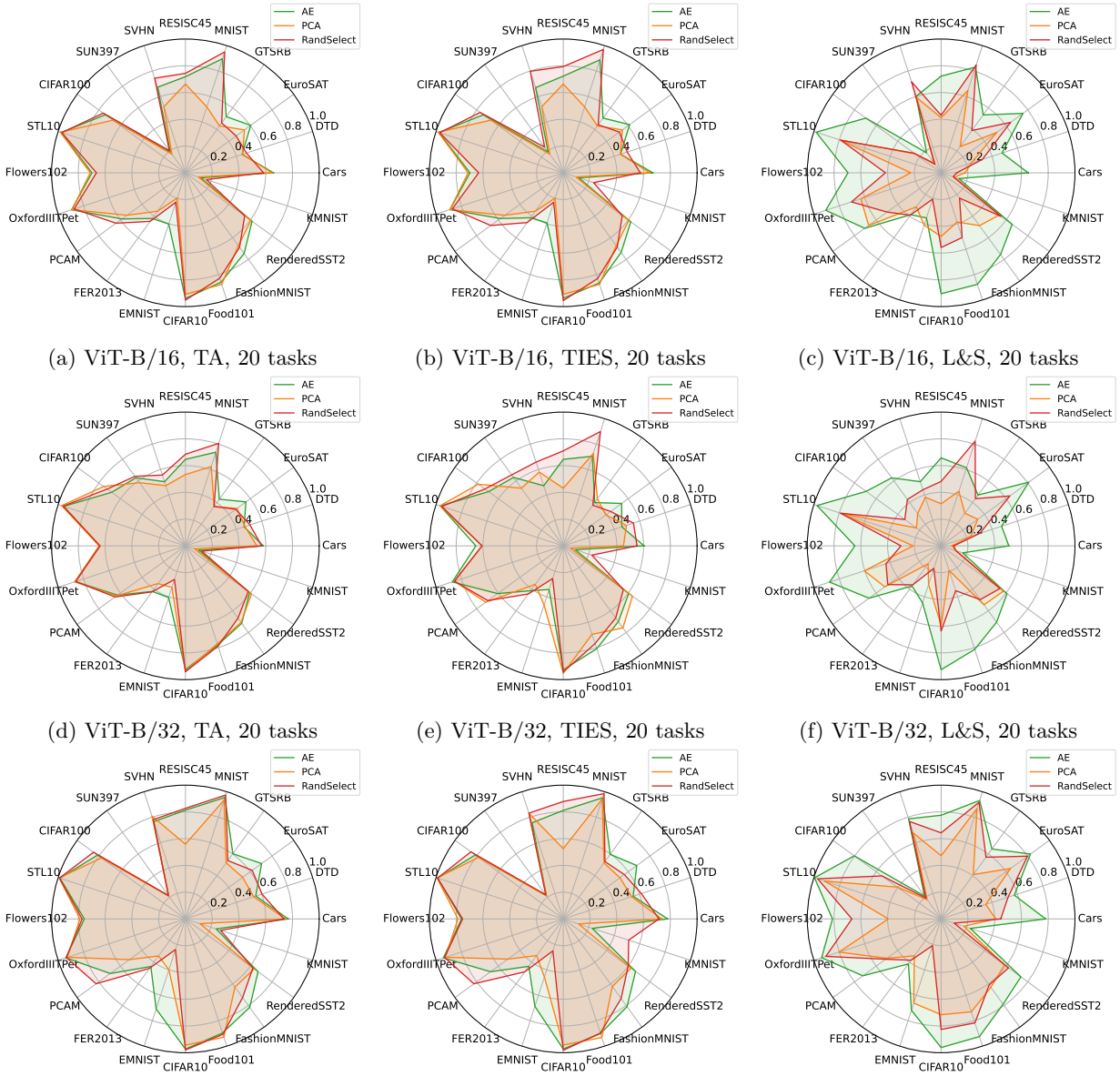
(d) ViT-B/32, TA, 14 tasks (e) ViT-B/32, TIES, 14 tasks (f) ViT-B/32, L&S, 14 tasks

(g) ViT-L/14, TA, 14 tasks (h) ViT-L/14, TIES, 14 tasks (i) ViT-L/14, L&S, 14 tasks

Figure 10: Per dataset bases comparison results for 14 vision tasks of Tab. 2.

F.4 Online Continual Learning

Hyperparameters and Implementation Details. In the continual setting, RandSelect maintains a fixed-size buffer of task vectors. As new tasks arrive, their vectors are added to the buffer until it reaches capacity. Once the buffer is full, the method enforces the size constraint by randomly discarding one existing task vector whenever a new one is added. For both TA and TSVM, unlike in the offline setting where the isotropic scaling coefficient α is tuned on a validation set, here we fix α to standard values suggested in prior work for simplicity. Specifically, we follow Tang et al. (2025) and set $\alpha = 0.3$ for TA and $\alpha = 1$ for TSVM (Gargiulo et al., 2025), without performing any additional scaling search. Moreover, as shown in Fig. 3, for larger basis sizes ($M = 6, 7$) we observed that using the annealing scheme $\tau = (500, 0.8)$ further improves AE



(a) ViT-B/16, TA, 20 tasks (b) ViT-B/16, TIES, 20 tasks (c) ViT-B/16, L&S, 20 tasks

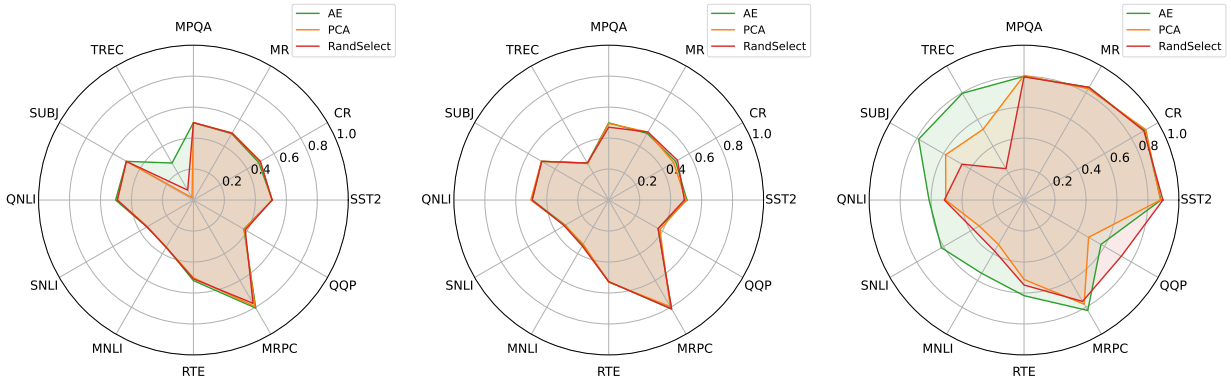
(d) ViT-B/32, TA, 20 tasks (e) ViT-B/32, TIES, 20 tasks (f) ViT-B/32, L&S, 20 tasks

(g) ViT-L/14, TA, 20 tasks (h) ViT-L/14, TIES, 20 tasks (i) ViT-L/14, L&S, 20 tasks

Figure 11: Per dataset bases comparison results for 20 vision tasks of Tab. 2.

performance; therefore, we adopt this setting for those runs. For smaller basis sizes ($M < 6$), we continue to use the default hyperparameters for AE basis construction.

Results across Architectures. We further include online addition results for different model architectures. From Tab. 21, we see that AE consistently achieves higher mean accuracy with notably lower variance compared to RandSelect across all three backbones. For example, on ViT-B/32, AE improves over RandSelect by more than 4 points under both TA and TSVM merging, while also cutting the standard deviation by a factor of 3–4. Even on larger models such as ViT-L/14, where the margins are smaller, AE still maintains a clear edge in both accuracy and stability. It is worth noting that RandSelect remains a surprisingly strong baseline, particularly in the continual setting. In some cases (e.g., TSVM on ViT-B/16), its mean accuracy approaches that of AE, though at the cost of much higher variance. This suggests that RandSelect can occasionally perform well, but such performance is highly sensitive to task order and thus less reliable. Overall,



(a) RoBERTa-base, TA, 12 tasks (b) RoBERTa-base, TIES, 12 tasks (c) RoBERTa-base, L&S, 12 tasks

Figure 12: Per dataset bases comparison results for 12 language tasks of Tab. 3.

Table 19: Task Vector Bases applied to QLoRA task vectors on ViT-B/16, 8 tasks, $M = 4$. AE remains the best basis method and nearly matches TA. Runtime is comparable to full precision (Tab. 10).

Setting	Method	Abs. Acc	Storage (MB)	Time (s)
Full FT (FP32)	AE (M=4)	0.666	1705.99	11059.96
QLoRA (4-bit)	TA	0.566	47.2	10545.00
	AE (M=4)	0.565	23.6	10545.00
	PCA (M=4)	0.542	23.6	10545.00
	RandSelect (M=4)	0.528	23.6	10545.00

AE provides a robust and dependable solution for online continual merging, delivering consistently strong results across architectures. The fact that RandSelect can sometimes compete highlights that random bases capture useful task diversity, but also underscores that this phenomenon deserves further investigation in future work.

F.5 Bases Negation

Hyperparameters and Metrics. The tuning of α in Tab. 5 is based on selecting the coefficient on the grid search over $[0, 0.05, 0.10, \ldots, 1.0]$ (21 grid points) that at least maintain 95% of pretrained model's ImageNet (control task) test accuracy, and we use the selected α to create the edited model and report the target and control metrics.

G LLM Usage Statement

We used LLMs to aid in polishing the writing of this paper. Specifically, LLMs were employed as a general-purpose assistant to improve clarity, grammar, and style, and to suggest alternative phrasings for technical explanations. They were not used to generate novel research ideas, design experiments, or produce results. The authors take full responsibility for all content, including text refined with the assistance of LLMs.

Table 20: Softmax vs. linear (unconstrained) encoder on task addition (8 tasks, ViT-B/16, $M = 4$). Normalized accuracy (%) per dataset.

Dataset	**Softmax**	**Linear**	**Delta**
Cars	84.9	67.3	−17.6
DTD	55.9	57.1	+1.2
EuroSAT	63.7	38.8	−24.9
GTSRB	55.4	45.1	−10.3
MNIST	98.5	80.3	−18.2
RESISC45	84.5	52.5	−32.0
SVHN	93.4	73.5	−19.9
SUN397	37.0	22.0	−15.0
Average	**71.7**	**54.6**	−17.1

Table 21: Online continual addition results (8 tasks, $M = 50\%$) over 5 runs with different task order. We report mean accuracy (%) ± standard deviation. The bases method with better mean for the same merging method is bold.

Method	ViT-B/32	ViT-B/16	ViT-L/14
RandSelect-TA	62.04 ± 3.69	70.86 ± 1.95	77.47 ± 1.40
AE-TA	$\mathbf{66.60} \pm 0.86$	$\mathbf{71.23} \pm 2.15$	$\mathbf{78.99} \pm 1.22$
RandSelect-TSVM	65.61 ± 3.15	$\mathbf{73.17} \pm 3.82$	79.84 ± 3.24
AE-TSVM	$\mathbf{69.01} \pm 0.98$	73.03 ± 0.92	$\mathbf{80.40} \pm 0.82$