FEATURE CORE PRESERVATION: PROTECTING TASK-SENSITIVE PARAMETER SUBSPACES FOR MODEL MERGING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model merging has emerged as an efficient parameter-editing paradigm for augmenting pretrained models with fine-tuned downstream weights. A central challenge is that merging task updates can overwrite task-shared knowledge encoded in the pretrained model and degrade downstream performance. Core Subspace Preservation (CSP) mitigates this issue by protecting the dominant singular subspace of pretrained weights, along which the pretrained model is already locally optimal. However, CSP leaves directions highly sensitive to downstream task updates unprotected. In this work, we revisit subspace protection from a sensitivity perspective and show, both analytically and empirically, that directions with high downstream gradient energy have low perturbation tolerance and are therefore vulnerable to merging interference. To make such protection practical, we establish a connection between parameter gradients and layer input features, and propose **Feature Core Preservation (FCP)**, which protects pretrained weights along dominant input-feature subspaces. FCP is a lightweight plug-and-play module that can be applied after arbitrary merging rules without requiring costly backward propagation. Extensive experiments on vision and language benchmarks, across different task scales and fine-tuning regimes, show that FCP consistently improves strong model merging baselines.

1 INTRODUCTION

Model merging (Li et al., 2023) offers a practical route to multi-task deployment by integrating fine-tuned models into the shared pretrained model directly in parameter space. As merging requires neither joint retraining nor access to all downstream datasets, it is attractive in data- or compute-limited scenarios.

Existing model merging methods operate on task vectors, defined as the parameter difference between the fine-tuned and pretrained weights. Task Arithmetic (Ilharco et al., 2023) provides a simple baseline that sums task vectors with a scaling coefficient. Subsequent methods aim to reduce interference among task vectors: TIES Merging (Yadav et al., 2023) and PCB Merging (Du et al., 2024) resolve sign conflicts among task vectors. More recent methods focus on preserving the principal components of task vectors, including WUDI Merging (Cheng et al., 2025), TSV (Gargiulo et al., 2025), and LOT Merging (Sun et al., 2025b).

Once the merged task vector is obtained, it is accumulated element-wise into the pretrained parameters. However, even state-of-the-art merging methods cannot fully avoid knowledge conflicts (Zhang et al., 2023), and merging may therefore overwrite useful knowledge encoded in the pretrained parameters. Core Subspace Preservation (CSP) (Sun et al., 2026) was among the first approaches to explicitly recognize the pretrained model itself as a source of useful knowledge during merging. CSP observes that the pretrained parameters are already near-optimal along their dominant singular directions, and therefore defines a core parameter subspace from the leading singular vectors of the pretrained weights. It then projects the merged task vector away from this subspace, thereby preserving the pretrained solution in these directions.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Although CSP protects the pretrained solution in regions where downstream adaptation is limited (Tang et al., 2026), this protection is inherently conservative. It does not capture the complementary case in which downstream tasks induce substantial parameter changes, even though these directions are precisely the ones most prone to overwriting pretrained knowledge. This raises a more general question: *given an imperfect merged task vector, where should the pretrained solution be preserved, and where should downstream knowledge be incorporated?* We study this question from the perspective of local sensitivity. In particular, we use the magnitude of the downstream gradient as a measure of how the task loss responds to parameter perturbations. As shown in Figure 1, along directions with negligible gradient energy, the pretrained model is already locally effective, leaving limited benefit from further adaptation. In contrast, directions with moderate gradient energy provide a regime in which downstream updates can meaningfully improve the model while remaining relatively tolerant to merging errors. At the other extreme, directions with high gradient energy are highly sensitive to parameter perturbations: although downstream adaptation may be important, even a small error in the merged update can induce a large change in the task loss.

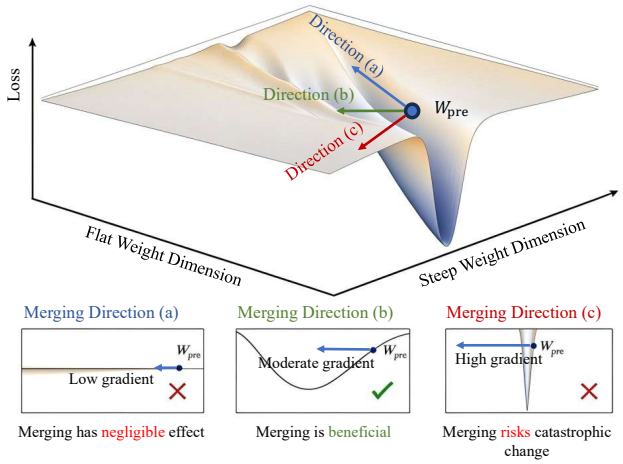


Figure 1: Illustration of parameter update directions with different gradients. Moderate-gradient directions are suitable for merging; directions with small or large gradients either benefit negligibly from merging or pose a high risk.

Based on the above analysis, we hypothesize that in the high-gradient regime, the potential loss caused by errors in the merged task vector outweighs the benefit of incorporating the corresponding downstream updates. To test this hypothesis, we introduce Gradient Core Preservation (GCP), a baseline that constructs a parameter subspace from the leading singular directions of downstream task gradients, and preserves the pretrained parameters along these directions during merging. Experiments show that GCP consistently improves strong merging baselines, including Task Arithmetic and TSV, with gains exceeding those achieved by CSP. These findings support protecting high-gradient directions, where the pretrained solution is more reliable than the merged parameters.

Despite the success of GCP, it requires explicit downstream gradients and incurs substantial computational overhead. Motivated by the structure of linear layers, where the gradient subspace is contained in the span of layer input features, we propose **Feature Core Preservation (FCP)**. FCP leverages the dominant input-feature directions as a practical proxy for the leading gradient directions, and preserves the pretrained parameters along these directions during merging. FCP requires neither expensive backpropagation nor labeled data, making it a lightweight, plug-and-play module that can be readily integrated into existing model merging methods. Several experiments on vision and language benchmarks demonstrate that FCP consistently improves merging baselines across a range of task scales and fine-tuning regimes. Our main contributions are summarized as follows:

- We revisit subspace protection of pretrained weights from a sensitivity perspective and empirically show that preserving high-gradient directions improves merging due to their high sensitivity to merging errors.
- We propose Feature Core Preservation (FCP), a plug-and-play module that identifies gradient-sensitive directions from dominant input-feature subspaces and preserves the pretrained parameters along these directions during merging, requiring neither backpropagation nor labeled data.
- We evaluate FCP on vision and language models. The results demonstrate that FCP consistently improves the performance of existing merging methods across architectures, task scales, and fine-tuning regimes.

2 Related Work

Model merging integrates multiple independently fine-tuned models into a single model without joint training or access to the original training data. A representative approach is Task Arithmetic (Ilharco et al., 2023), which defines task vectors as the parameter differences between fine-tuned models and their pretrained initialization, and combines them through parameter-wise addition. Subsequent methods mainly improve task-vector aggregation by mitigating interference or selecting informative components. TIES (Yadav et al., 2023) and PCB (Du et al., 2024) alleviate conflicts by removing redundant or inconsistent parameter updates, while TSV (Gargiulo et al., 2025) and ESM (Li et al., 2026) exploit the low-rank or singular structure of task vectors to identify more compatible subspaces. AdaRank (Lee et al., 2026) further adaptively selects useful singular components instead of applying a fixed-rank truncation.

Other methods incorporate test-time training to obtain more compatible task updates. RegMean (Jin et al., 2023) and CAT Merging (Sun et al., 2025a) exploit representative features to estimate compatible parameter updates, whereas AdaMerging (Yang et al., 2024b), Surgery (Yang et al., 2024a), and Prodistill (Xu et al., 2025) use unlabeled data or intermediate representations to adapt merging weights or align task models. SCORE (Chaves et al., 2026) constructs a shared singular basis to identify and suppress competing directions across models.

Notably, some methods such as RegMean (Jin et al., 2023) and LOT Merging (Sun et al., 2025b) exploit input features to estimate merging weights. While FCP likewise relies on input features to determine which parameters should not be merged, its underlying principle differs fundamentally: FCP relies on features from the pretrained model, whereas these methods compute features from the fine-tuned models. Consequently, FCP aims to preserve knowledge encoded in the pretrained parameters, while existing feature-based merging methods target conflicts across tasks. FCP is therefore orthogonal to these methods and can be applied to them to further improve performance.

3 Preliminaries

3.1 Model Merging

Let W_{pre} denote the parameters of a model pretrained on a large and diverse corpus, and let $\{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ denote the datasets of K downstream tasks. Starting from W_{pre}, we independently fine-tune the model on each task to obtain $\{W_1, \ldots, W_K\}$. Model merging aims to consolidate these models into a single model W_{mtl} that performs well across all K tasks.

To facilitate model merging, it is essential to isolate task-specific components from the model parameters. This is commonly achieved through the task vector (Ilharco et al., 2023), defined as the parameter difference between the fine-tuned model W_k (take task k as an example) and the pretrained initialization W_{pre}:

$$T_k = W_k - W_{\mathrm{pre}}. \tag{1}$$

The parameter operations are performed layer-wise. For simplicity, we analyze each layer independently throughout this paper and leverage W_k, W_{pre}, and T_k to denote the weight matrices of a given layer.

Existing model-merging methods primarily focus on aggregating these task vectors into a merged task vector T_{merge}. A number of techniques have been proposed for merging task vectors, including masking, projection, and adaptive averaging. The merged model is finally obtained by adding the merged task vector T_{merge} back to the pretrained parameters with a scaling coefficient λ:

$$W_{\mathrm{merge}} = W_{\mathrm{pre}} + \lambda T_{\mathrm{merge}}. \tag{2}$$

3.2 Core Subspace Preservation

Equation (2) captures the common paradigm of model merging, where W_{pre} serves as a common reference, while T_{merge} encodes task-specific adaptations that are combined to construct the merged model. However, even state-of-the-art merging methods cannot fully resolve conflicts among task vectors, potentially overwriting task-relevant knowledge encoded in W_{pre}.

To address this issue, Sun et al. (2026) show that the pretrained parameters are already locally optimal within the parameter subspace spanned by their dominant singular directions. Based on this observation, they propose Core Subspace Preservation (CSP), which protects these pretrained parameters from being overwritten by the merged task update. For each layer, consider the singular value decomposition of the pretrained weight $W_{\text{pre}} = U\Sigma V^\top$. CSP defines a core parameter subspace (CSP Core) spanned by the singular vectors associated with the largest r singular values

$$\Pi_r^{csp}(W_{pre}) \triangleq \{W \in \mathbb{R}^{m\times n} | (I - U_r U_r^\top) W (I - V_r V_r^\top) = 0\}. \tag{3}$$

Then, CSP projects the merged task vector into the orthogonal complement:

$$W_{\text{merge}} = W_{\text{pre}} + \lambda (I - U_r U_r^\top) T_{\text{merge}} (I - V_r V_r^\top). \tag{4}$$

4 Rethinking Subspace Protection in Model Merging

CSP reveals an important design choice in model merging, i.e., pretrained parameters can remain a strong candidate solution rather than being uniformly overwritten by task updates. This raises a fundamental question: *Which parameters in W_{pre} should be preserved during merging, and which should incorporate downstream knowledge through T_{merge}?* This section revisits this question from a gradient perspective and finds that in the subspaces with high gradient, the performance is particularly sensitive to merging errors. Such a high-gradient regime is a natural candidate for protection.

Let $L_k(W)$ denote the loss function of task k, and let $G_k = \nabla_W L_k(W_{\text{pre}})$ denote the gradient at W_{pre}. For a parameter subspace $\mathcal{S}$, let $G_k^{\text{prot}} = \text{Proj}_{\mathcal{S}}(G_k)$ be the projection of the gradient onto $\mathcal{S}$. Consider the component of the merged update in $\mathcal{S}$ and decompose it as

$$T_{\mathcal{S}} = \text{Proj}_{\mathcal{S}}(T_{\text{merge}}) = T_{\mathcal{S}}^\star + \epsilon_{\mathcal{S}}, \tag{5}$$

where $T_{\mathcal{S}}^\star \triangleq \text{Proj}_{\mathcal{S}}(T_k)$ denotes the ideal task-k update restricted to $\mathcal{S}$, and $\epsilon_{\mathcal{S}}$ denotes the merging error induced by aggregating multiple task vectors. A first-order expansion around W_{pre} gives

$$L_k(W_{\text{pre}} + T_{\mathcal{S}}) - L_k(W_{\text{pre}}) \approx \underbrace{\langle G_k^{\text{prot}}, T_{\mathcal{S}}^\star \rangle_F}_{\text{task-update effect}} + \underbrace{\langle G_k^{\text{prot}}, \epsilon_{\mathcal{S}} \rangle_F}_{\text{merging-error effect}}. \tag{6}$$

The first term reflects the useful task adaptation, while the second term captures the effect of merging errors. This trade-off suggests that protecting $\mathcal{S}$ is beneficial when the merging-error term is substantial. To quantify this risk, we bound the merging-error term by its worst-case perturbation:

$$\max_{\substack{\epsilon_{\mathcal{S}} \in \mathcal{S} \\ \|\epsilon_{\mathcal{S}}\|_F \le \rho}} \langle G_k^{\text{prot}}, \epsilon_{\mathcal{S}} \rangle_F = \rho \|G_k^{\text{prot}}\|_F = \rho \left\| \boldsymbol{\sigma}(G_k^{\text{prot}}) \right\|_2, \tag{7}$$

where $\boldsymbol{\sigma}(A)$ denotes the vector of singular values of matrix A. Thus, for a fixed error budget ρ, the worst-case error amplification grows linearly with the projected gradient magnitude.

Equivalently, for a loss tolerance ε, the tolerable perturbation radius satisfies

$$\rho \le \frac{\varepsilon}{\left\| \boldsymbol{\sigma}(G_k^{\text{prot}}) \right\|_2} = \frac{\varepsilon}{\mathcal{E}(G_k^{\text{prot}}) \sqrt{\left\| \boldsymbol{\sigma}(G_k^{\text{prot}}) \right\|_0}}, \text{ where } \mathcal{E}(G_k^{\text{prot}}) \triangleq \frac{\left\| \boldsymbol{\sigma}(G_k^{\text{prot}}) \right\|_2}{\sqrt{\left\| \boldsymbol{\sigma}(G_k^{\text{prot}}) \right\|_0}}. \tag{8}$$

Recall that $\|\boldsymbol{\sigma}(A)\|_0 = \text{rank}(A)$. Normalizing by $\sqrt{\|\boldsymbol{\sigma}\|_0}$ eliminates the dependence on the rank and dimensionality of G_k^{prot}, so that the rank-normalized $\mathcal{E}(G_k^{\text{prot}})$ characterizes the average gradient energy per effective dimension.

Equation (8) shows that, at a fixed projected-gradient rank, a larger $\mathcal{E}(G_k^{\text{prot}})$ implies a smaller perturbation tolerance, making the corresponding subspace more sensitive to merging errors. Although these directions may contain important task-specific information, the potential damage caused by inaccurate task-vector integration can outweigh the benefit of incorporating the corresponding updates. This analysis relies on the common local assumption in model merging that the loss can be approximated in a neighborhood of the pretrained parameters (Zhou et al., 2024; Sommariva et al., 2026). This assumption is reasonable when fine-tuning and update scaling keep the resulting displacement modest (Wei et al., 2026). Beyond this local regime, Appendix B provides an explicit bound: under a local β-Lipschitz gradient assumption, the actual loss increase is at most $\rho\|G_k^{\text{prot}}\|_F + \beta\rho^2/2$.

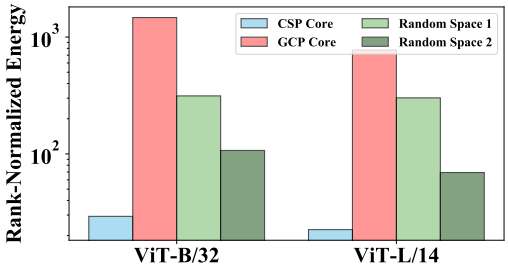


Figure 2: Rank-normalized gradient energy $\mathcal{E}(G_k^{\text{prot}})$ in the subspaces. Random space 1/2 denote randomly sampled, equal-rank subspaces with moderate gradient. Suppressing merging within moderate-gradient space harms performance, whereas suppressing merging along high- or low-gradient directions yields improvement.

Table 1: Average accuracy (%) on eight vision tasks with ViT-B/32 and ViT-L/14.

Method	ViT-B/32	ViT-L/14
Task Arithmetic	69.1	84.5
+ random space 1	68.4	83.7
+ random space 2	68.7	83.5
+ CSP	72.7	86.2
+ GCP	74.0	88.5
+ CSP & GCP	74.1	88.6
TSV	84.1	91.7
+ random space 1	84.0	91.0
+ random space 2	84.3	90.9
+ CSP	85.1	92.5
+ GCP	85.9	93.1
+ CSP & GCP	86.2	93.1

4.1 Protecting High-Gradient Directions Improves Model Merging

The analysis above motivates preserving W_{pre} along high-gradient directions to mitigate merging interference, even though these directions may also contain useful task-specific information. To validate this hypothesis, we introduce a baseline, Gradient Core Preservation (GCP), which protects pretrained weights from being overwritten by the merged task update along high-gradient directions. For each weight matrix, GCP accumulates the left and right gradient Gram matrices:

$$S_{\text{L}} = \sum_{k=1}^{K} G_k G_k^\top, \qquad S_{\text{R}} = \sum_{k=1}^{K} G_k^\top G_k. \tag{9}$$

Let U_r^{grad} and V_r^{grad} denote the leading r orthonormal eigenvectors of S_{L} and S_{R}, respectively. Equivalently, these bases maximize the aggregate gradient energy

$$U_r^{\text{grad}} \in \arg\max_{U^\top U = I_r} \sum_{k=1}^{K} \|U^\top G_k\|_F^2, \qquad V_r^{\text{grad}} \in \arg\max_{V^\top V = I_r} \sum_{k=1}^{K} \|G_k V\|_F^2. \tag{10}$$

We define the GCP core by replacing U_r and V_r in Eq. (3) with U_r^{grad} and V_r^{grad}:

$$\Pi_r^{gcp}(W_{pre}) \triangleq \{W \in \mathbb{R}^{m\times n} | (I - U_r^{\text{grad}} U_r^{\text{grad}\top}) W (I - V_r^{\text{grad}} V_r^{\text{grad}\top}) = 0\}, \tag{11}$$

and project the merged task vector onto the orthogonal complement of this core gives

$$W_{\text{merge}} = W_{\text{pre}} + \lambda\big(I - U_r^{\text{grad}}(U_r^{\text{grad}})^\top\big) T_{\text{merge}} \big(I - V_r^{\text{grad}}(V_r^{\text{grad}})^\top\big). \tag{12}$$

As shown in Figure 2, the rank-normalized gradient energy in the GCP core is consistently one to two orders of magnitude higher than that of the CSP core. The low gradient energy of the CSP core is expected, as CSP preserves directions where the pretrained parameters are already close to local optima, and thus exhibit weak downstream gradient signals. In contrast, GCP targets highly sensitive directions where downstream losses exhibit stronger responses to parameter perturbations.

Interestingly, despite their contrasting gradient characteristics, preserving pretrained parameters in both regions consistently improves model merging. As shown in Table 1, both CSP and GCP improve the performance of two representative merging approaches, Task Arithmetic (Ilharco et al., 2023) and TSV (Gargiulo et al., 2025). For Task Arithmetic, GCP increases the average accuracy from 69.1% to 74.0% on ViT-B/32 and from 84.5% to 88.5% on ViT-L/14. In contrast, preserving randomly selected subspaces with moderate gradient magnitudes degrades performance, suggesting that merging along moderately gradient directions is essential for downstream knowledge. Similar improvements are observed for TSV. Moreover, combining CSP and GCP achieves further gains, suggesting that the two preservation strategies protect complementary regions of the parameter space.

These results support our hypothesis that high-gradient directions are worth protecting during model merging, extending subspace preservation beyond the low-gradient CSP core. However, GCP requires labeled task data and computationally expensive backward passes to estimate the gradient core. In the next section, we propose a lightweight alternative that approximates this subspace using only forward features.

5 METHOD

For linear operations, the gradient subspace is structurally coupled with the input feature subspace. Let $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ denotes a weight matrix, and let $X \in \mathbb{R}^{d_{\text{in}} \times N_k}$ collects the input features as columns, with the corresponding outputs given by $Z = WX$. By the chain rule, the gradient of W can be computed as

$$G = \frac{\partial L}{\partial W} = \frac{\partial L}{\partial Z} X^\top. \tag{13}$$

Each row of G is therefore a linear combination of the transposed input features, implying that $\text{range}(G^\top) \subseteq \text{range}(X)$. Thus, the feature span contains the right singular subspace of the gradient, providing a natural basis for estimating gradient-sensitive directions from forward features.

The leading feature and gradient subspaces may not coincide, since the loss derivatives $\frac{\partial L}{\partial Z}$ may reweigh the contributions of different feature directions and can consequently alter their dominant components. Nevertheless, our experiments suggest that this discrepancy has limited practical impact. As shown by the gradient-energy comparison in Figure 3, the dominant feature subspaces (FCP Core) capture substantial gradient energy, with their rank-normalized gradient energy reaching levels comparable to those of GCP. Although our construction does not guarantee an accurate estimate of the gradient subspace, it provides an efficient proxy for identifying parameter subspaces with high downstream gradient energy.

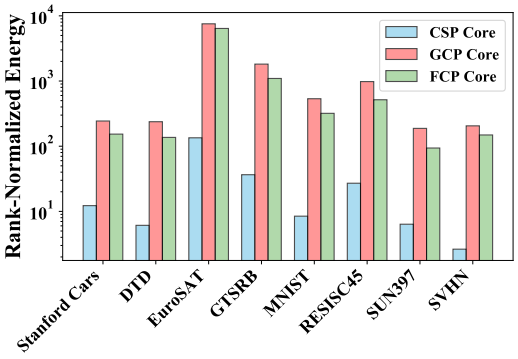


Figure 3: Rank-normalized gradient energy $\mathcal{E}(G_k^{\text{prot}})$ in the core subspaces.

5.1 FEATURE CORE PRESERVATION

In this section, we propose Feature Core Preservation (FCP), which identifies protection directions from pretrained features. For each task k, we sample a small calibration set $\mathcal{C}_k$ and collect layer inputs through forward passes of W_{pre}. Each example or token contributes a feature column to X_k, and we concatenate these matrices across tasks as $X = [X_1, \ldots, X_K]$.

We compute the singular value decomposition of X and retain its leading r orthonormal singular vectors $U_{X,r}$, and define FCP Core with $\Pi_r^{fcp}(W_{pre}) = \{W | W(I - U_{X,r}U_{X,r}^\top) = 0\}$, and constructs

$$W_{\text{merge}} = W_{\text{pre}} + \lambda T_{\text{merge}}(I - U_{X,r}U_{X,r}^\top). \tag{14}$$

Consequently, the merged update does not alter the pretrained linear mapping along the selected input-feature subspace.

We additionally consider output-side protection using the pretrained layer outputs $Z = W_{\text{pre}}X$. Let $U_{Z,r}$ contain the leading r singular vectors of Z. The bilateral variant becomes

$$W_{\text{merge}} = W_{\text{pre}} + \lambda(I - U_{Z,r}U_{Z,r}^\top)T_{\text{merge}}(I - U_{X,r}U_{X,r}^\top). \tag{15}$$

Notably, output-side protection is not directly motivated by the gradient–feature relationship established above. Our ablation results show that input-side protection accounts for most of the performance improvement, while output-side protection provides modest additional gains. We therefore retain it as a complementary component.

Table 2: Multi-task performance when merging ViT-B/32 models on eight vision tasks. The checkpoints are obtained from Ilharco et al. (2023).

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg Acc
Pretrained	62.3	59.7	60.7	45.5	31.4	32.6	48.5	43.8	48.0
Individual	75.3	77.7	96.1	99.7	97.5	98.7	99.7	79.4	90.5
Traditional MTL	73.9	74.4	93.9	98.2	95.8	98.9	99.5	77.9	89.1
Task Arithmetic	55.2	54.9	66.7	78.9	80.2	69.7	**97.3**	50.4	69.1
Task Arithmetic w/ CSP	62.9	61.5	71.8	83.4	79.5	70.5	96.8	55.1	72.7
Task Arithmetic w/ FCP	**66.3**	**65.5**	**76.8**	**85.8**	**82.6**	**72.3**	97.2	**55.9**	**75.3**
LOT Merging	67.7	**67.5**	85.7	94.9	93.4	89.8	98.7	63.6	82.7
LOT Merging w/ CSP	67.7	66.5	87.0	95.5	94.6	91.6	98.9	**67.0**	83.6
LOT Merging w/ FCP	**68.4**	66.7	**88.4**	**96.9**	**96.0**	**92.0**	**99.3**	66.5	**84.3**
ISO-C	71.7	70.5	87.0	95.0	**90.0**	90.8	**99.2**	68.6	84.1
ISO-C w/ CSP	74.2	**73.2**	87.4	**95.4**	88.8	**91.0**	99.1	70.1	84.9
ISO-C w/ FCP	**74.7**	72.1	**88.8**	95.3	88.5	90.8	99.1	**71.3**	**85.1**
ISO-CTS	74.5	**74.1**	88.9	93.6	84.7	90.1	98.8	69.8	84.3
ISO-CTS w/ CSP	73.7	73.8	88.7	**94.7**	**86.2**	**90.8**	**99.0**	69.6	84.6
ISO-CTS w/ FCP	**74.9**	73.7	**89.1**	**94.7**	84.5	90.5	**99.0**	**71.4**	**84.7**
TSV	70.1	72.1	85.9	94.3	90.9	91.2	99.2	68.8	84.1
TSV w/ CSP	71.1	73.4	86.3	95.4	**92.3**	92.6	99.3	70.7	85.1
TSV w/ FCP	**72.7**	**75.2**	**88.7**	**96.9**	92.0	**93.1**	**99.4**	**71.8**	**86.2**

Analogously to CSP, FCP serves as a plug-and-play module for existing merging methods, including Task Arithmetic, TSV, ISO, and LOT Merging. Algorithm 1 summarizes the procedure, which requires neither calibration labels nor backward passes.

6 Experiments

6.1 Experimental Setup

Benchmarks and checkpoints. Our primary vision benchmark contains eight tasks: SUN397, Stanford Cars, RESISC45, EuroSAT, SVHN, GTSRB, MNIST, and DTD. We use ViT-B/32 and ViT-L/14 checkpoints from Ilharco et al. (2023). To study larger task collections, we additionally evaluate the 8-, 14-, and 20-task vision settings provided by FusionBench (Tang et al., 2025) with ViT-B/32, ViT-B/16, and ViT-L/14. We report this checkpoint collection separately from the primary vision benchmark. For language experiments, we use Flan-T5-base checkpoints from FusionBench, obtained by full fine-tuning or LoRA fine-tuning with rank 16 on eight GLUE tasks (Wang et al., 2018): CoLA, MNLI, MRPC, QNLI, QQP, RTE, SST-2, and STS-B. We follow the benchmark's text-to-text formulation, in which the model generates a class label or a similarity score.

Baselines. In the experiments, we select the following five representative model merging methods as the basic mergers: Task Arithmetic (Ilharco et al., 2023), LOT Merging (Sun et al., 2025b), ISO-C (Marczak et al., 2025), ISO-CTS (Marczak et al., 2025), and TSV (Gargiulo et al., 2025). For each merger, we compare the original performance, parameter-based CSP, and our activation-based FCP. All variants within a comparison use the same task checkpoints, so protection changes the merged update without retraining the constituent task models. We also report the performance with the pretrained model, denoted as Pretrained. Individual denotes separately fine-tuned models evaluated on their respective tasks, and Traditional MTL denotes joint multi-task training.

Implementation. Each task contributes 64 unlabeled calibration examples drawn from its training set. The rank r is selected from $\{1, 2\}$ using a validation set. For vision tasks, we report test accuracy (%) and the unweighted average across tasks. For language tasks, following the FusionBench Flan-T5 evaluation protocol, the seven GLUE classification tasks are evaluated by exact-match accuracy between generated and target labels, while STS-B is evaluated by Spearman's rank correlation. Both metrics are reported on a 0–100 scale, and we report their unweighted average score.

Table 3: Multi-task performance when merging ViT-L/14 models on eight vision tasks. The checkpoints are obtained from Ilharco et al. (2023).

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg Acc
Pretrained	66.8	77.7	71.0	59.9	58.4	50.5	76.3	55.3	64.5
Individual	82.3	92.4	97.4	100.0	98.1	99.2	99.7	84.1	94.2
Traditional MTL	80.8	90.6	96.3	96.3	97.6	99.1	99.6	84.4	93.1
Task Arithmetic	73.9	82.1	86.6	94.1	87.9	86.7	98.9	65.6	84.5
Task Arithmetic w/ CSP	75.1	83.6	88.0	95.6	90.2	89.9	99.1	67.8	86.2
Task Arithmetic w/ FCP	**77.2**	**88.2**	**92.2**	**97.3**	**90.4**	**93.2**	**99.2**	**72.7**	**88.8**
LOT Merging	76.7	88.6	91.7	98.7	97.1	95.7	99.5	76.4	90.5
LOT Merging w/ CSP	76.3	89.1	93.1	98.9	**97.2**	96.5	**99.6**	**77.6**	91.0
LOT Merging w/ FCP	**77.1**	**89.7**	**94.2**	**99.2**	**97.2**	**97.3**	99.5	77.5	**91.5**
ISO-C	82.9	90.6	95.4	99.0	**94.2**	97.0	99.4	81.0	92.5
ISO-C w/ CSP	83.1	91.4	95.7	**99.2**	94.0	97.2	99.4	81.5	92.7
ISO-C w/ FCP	**83.3**	**91.6**	**95.8**	99.0	93.6	**97.3**	**99.5**	**82.7**	**92.9**
ISO-CTS	83.1	91.9	96.0	99.0	93.8	97.9	99.5	82.4	93.0
ISO-CTS w/ CSP	83.3	91.8	96.1	**99.5**	**95.1**	**98.5**	99.5	83.0	93.4
ISO-CTS w/ FCP	**83.6**	**92.2**	**96.6**	99.2	94.7	98.3	**99.9**	**83.7**	**93.5**
TSV	79.2	89.8	93.8	98.9	95.6	96.5	99.5	80.0	91.7
TSV w/ CSP	80.5	90.2	94.2	99.4	**96.5**	97.8	**99.7**	82.1	92.5
TSV w/ FCP	**82.4**	**91.4**	**95.8**	**99.5**	96.0	**98.2**	99.6	**83.5**	**93.3**

Table 4: Average accuracy (%) when merging vision models with more tasks. The checkpoints are from Tang et al. (2025).

Backbone	ViT-B/32			ViT-B/16			ViT-L/14		
Number of Tasks	8	14	20	8	14	20	8	14	20
Task Arithmetic	69.50	65.76	60.63	77.17	70.41	64.46	82.08	77.53	70.98
Task Arithmetic w/ CSP	71.50	67.37	61.84	78.28	71.49	65.14	83.60	78.71	71.68
Task Arithmetic w/ FCP	**76.60**	**71.21**	**65.36**	**80.86**	**75.98**	**69.29**	**87.34**	**83.15**	**75.26**
ISO-C	80.39	77.38	69.94	85.06	82.00	75.71	90.63	89.22	83.18
ISO-C w/ CSP	80.38	78.16	71.95	86.71	82.73	76.91	91.28	89.91	84.06
ISO-C w/ FCP	**80.63**	**78.69**	**72.84**	**87.20**	**83.46**	**77.47**	**91.75**	**90.25**	**84.83**
ISO-CTS	78.84	78.12	73.17	83.50	83.61	77.86	90.28	89.60	84.75
ISO-CTS w/ CSP	78.91	78.41	73.75	85.23	84.45	**78.70**	91.43	90.36	84.92
ISO-CTS w/ FCP	**79.64**	**79.11**	**74.84**	**86.23**	**84.48**	**78.70**	**91.45**	**90.41**	**85.02**
TSV	83.08	78.38	73.20	87.22	81.77	77.96	90.57	87.84	82.81
TSV w/ CSP	84.43	79.74	75.06	87.93	82.60	79.54	91.64	88.91	85.48
TSV w/ FCP	**85.56**	**82.00**	**78.33**	**89.15**	**84.73**	**82.56**	**92.68**	**90.20**	**88.52**

6.2 Comparison Results

Merging eight ViT-B/32 models. As shown in Table 2, FCP consistently improves all five merging methods. The largest improvement is observed for Task Arithmetic, increasing the average accuracy from 69.1% to 75.3%, FCP also improves stronger merging methods such as TSV, which achieves the highest accuracy at 86.2%. Compared with CSP, FCP yields larger improvements on most model merging methods.

Merging eight ViT-L/14 models. The improvements remain consistent on the larger ViT-L/14 backbone (Table 3). FCP improves all five merging methods and consistently outperforms CSP. The best overall result is achieved by ISO-CTS with FCP at 93.5%, closely followed by TSV at 93.3%. The experiments across both backbones demonstrate that calibration features provide robust improvements across model architectures.

Scaling to More Tasks. Table 4 extends the evaluation to merging a larger number of tasks. As shown, FCP consistently improves over the corresponding unprotected merger across all configurations. The gains remain substantial as the number of merged tasks increases. These results demonstrate that FCP remains effective as the number of merged tasks scales.

Table 5: Multi-task performance of model merging with fully fine-tuned and LoRA fine-tuned checkpoints on eight language tasks. The checkpoints are from Tang et al. (2025).

Fine-tuning	Method	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Avg Score
Pretrained		69.1	56.5	76.2	88.4	82.1	80.1	91.2	62.2	75.7
Full fine-tuned	Individual	75.0	83.4	87.5	91.5	85.4	85.9	93.6	88.7	86.4
	Simple Average	69.1	62.6	79.4	89.8	83.9	81.2	91.7	73.2	78.9
	Task Arithmetic	**70.5**	57.8	78.4	90.2	83.6	80.5	**92.3**	**77.8**	78.9
	Task Arithmetic w/ CSP	69.8	64.0	78.9	**90.3**	**83.9**	81.9	92.2	77.0	79.8
	Task Arithmetic w/ FCP	69.9	**72.2**	**79.2**	**90.3**	83.8	**83.0**	**92.3**	76.6	**80.9**
	TSV	70.8	79.7	78.7	**90.7**	84.3	83.4	**93.1**	81.4	82.8
	TSV w/ CSP	70.9	79.9	79.4	**90.7**	**84.4**	**83.8**	**93.1**	83.6	83.2
	TSV w/ FCP	**72.1**	**81.2**	**79.9**	**90.7**	84.3	83.0	**93.1**	**85.8**	**83.8**
LoRA fine-tuned (LoRA rank = 16)	Individual	69.1	82.7	85.5	90.9	84.0	84.4	92.9	87.4	84.6
	Simple Average	69.7	59.7	78.9	90.1	83.8	90.5	91.2	72.0	79.5
	Task Arithmetic	68.8	55.2	78.7	89.8	83.7	79.1	91.5	72.4	77.4
	Task Arithmetic w/ CSP	**69.3**	58.5	78.2	**89.9**	**83.9**	79.1	**91.7**	**72.7**	77.9
	Task Arithmetic w/ FCP	**69.3**	**68.3**	**79.9**	89.4	83.6	**80.5**	**91.7**	71.3	**79.3**
	TSV	**69.3**	77.1	80.4	90.0	83.6	79.1	92.5	82.6	81.8
	TSV w/ CSP	69.2	77.8	**80.6**	90.4	83.9	78.7	**92.7**	82.1	81.9
	TSV w/ FCP	69.1	**78.7**	**80.6**	**90.8**	**84.0**	**80.9**	**92.7**	**83.5**	**82.5**

Table 6: Ablation studies when merging ViT-B/32 models on eight vision tasks.

Base Method	Input-Side	Output-Side	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg Acc
Task Arithmetic			55.2	54.9	66.7	78.9	80.2	69.7	97.3	50.4	69.1
	✓		66.0	65.2	76.5	85.7	**82.8**	**72.6**	97.0	55.5	75.2
		✓	60.4	58.8	69.3	78.9	79.6	69.6	96.6	52.1	70.7
	✓	✓	**66.3**	**65.5**	**76.8**	**85.8**	82.6	72.3	**97.2**	**55.9**	**75.3**
TSV			70.1	72.1	85.9	94.3	90.9	91.2	99.2	68.8	84.1
	✓		72.6	75.0	**88.8**	**97.0**	91.7	**93.1**	99.3	71.3	86.1
		✓	69.5	71.2	85.2	94.3	**92.4**	92.1	99.3	69.3	84.2
	✓	✓	**72.7**	**75.2**	88.7	96.9	92.0	**93.1**	**99.4**	**71.8**	**86.2**

Merging eight Flan-T5-base models. Table 5 reports the evaluation on language tasks. FCP consistently improves Task Arithmetic under both full and LoRA fine-tuning. Under full fine-tuning, FCP yields a clear gain over Task Arithmetic, raising the average performance from 78.9 to 80.9 and further lifting TSV to 83.8. Under LoRA fine-tuning, FCP still brings consistent benefits, improving Task Arithmetic from 77.4 to 79.3 and TSV from 81.8 to 82.5 in average performance.

6.3 Ablation of Input- and Output-Side Protection

Table 6 ablates the input-side and output-side protection of FCP. Input-side protection alone accounts for most of the improvement, raising the average accuracy of Task Arithmetic from 69.1% to 75.3%, whereas output-side protection alone is considerably weaker. Combining both sides yields only marginal additional gains over the input-side variant. This asymmetry is consistent with our analysis: the input-feature subspace directly contains the right gradient subspace of each linear layer, and thus serves as a faithful proxy for the gradient-sensitive directions that should be protected, while the output-side subspace lacks this theoretical connection.

7 Conclusion

This paper focuses on protecting pretrained parameters in model merging and studies this problem through the lens of downstream gradients. We show that subspaces with high downstream gradient energy are particularly sensitive to merging perturbations. Based on this observation, we propose Feature Core Preservation (FCP), which uses dominant input-feature directions as a lightweight proxy for sensitive parameter subspaces and can be readily applied to existing model merging methods. Extensive experiments across vision and language tasks demonstrate consistent improvements from FCP, validating the effectiveness of this gradient-guided protection strategy. A limitation is that FCP uses input features as a practical proxy for gradient-sensitive subspaces; although effective in our experiments, this approximation may not capture all sensitive directions.

Reproducibility Statement

We have provided an anonymous code package in the supplementary materials to facilitate the regeneration of the experimental results (see README.md).

AI Use Statement

In this work, we used an AI tool (ChatGPT) to assist with a limited part of the research process, mainly for language polishing and minor code refactoring. It was not involved in designing experiments, collecting or generating data, performing analyses, or interpreting results. All experiments, data, and results reported in this paper were produced manually by the authors.

References

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In Proceedings of the 11th international workshop on semantic evaluation, pp. 1–14, 2017.

Levy Chaves, Chao Zhou, Rebekka Burkholz, Eduardo Valle, and Sandra Avila. Bridging domains through subspace-aware model merging. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 7990–7999, June 2026.

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. doi: 10.1109/JPROC.2017.2675998.

Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. Whoever started the interference should end it: Guiding data-free model merging via task vectors. In International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=xR9msNaREW.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In IEEE Conference on Computer Vision and Pattern Recognition, June 2014.

William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In Proceedings of the third international workshop on paraphrasing (IWP2005), 2005.

Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim Kuan Goh, Ho-Kin Tang, Daojing He, and Min Zhang. Parameter competition balancing for model merging. In Advances in Neural Information Processing Systems, 2024.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task singular vectors: Reducing task interference in model merging. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 18695–18705, June 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. doi: 10.1109/JSTARS.2019.2918242.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=FCnohuR6AnM.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In IEEE International Conference on Computer Vision Workshops, June 2013.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Chanhyuk Lee, Jiho Choi, Chanryeol Lee, Donggyun Kim, and Seunghoon Hong. Adarank: Adaptive rank pruning for enhanced model merging. In International Conference on Learning Representations, volume 2026, pp. 6484–6510, 2026.

Longhua Li, Lei Qi, Qi Tian, and Xin Geng. Model merging in the essential subspace. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 31145–31154, June 2026.

Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. Deep model fusion: A survey. arXiv preprint arXiv:2309.15698, 2023.

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. In International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=RBZpAa27ls.

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, 2011.

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 conference on empirical methods in natural language processing, pp. 1631–1642, 2013.

Thomas Sommariva, Francesca Morandi, Simone Calderara, and Angelo Porrello. Distilling linearized behavior into non-linear fine-tuning for effective task arithmetic. In International Conference on Machine Learning, 2026. URL https://openreview.net/forum?id=2lMqcGV81Z.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In International Joint Conference on Neural Networks, pp. 1453–1460, 2011.

Wenju Sun, Qingyong Li, Yangli-ao Geng, and Boyang Li. Cat merging: A training-free approach for resolving conflicts in model merging. International Conference on Machine Learning, 2025a.

Wenju Sun, Qingyong Li, Wen Wang, Yang Liu, Yangli-ao Geng, and Boyang Li. Towards minimizing feature drift in model merging: Layer-wise task vector fusion for adaptive knowledge integration. In Advances in Neural Information Processing Systems, 2025b.

Wenju Sun, Qingyong Li, Tiancheng Li, Yangliao Geng, and Boyang Li. Revisiting the role of pre-trained weights in model merging: On near-optimality within the core subspace. In International Conference on Machine Learning, 2026. URL https://openreview.net/forum?id=NrGCh79ANM.

Anke Tang, Li Shen, Yong Luo, Enneng Yang, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. Fusionbench: A comprehensive benchmark of deep model fusion. Journal of Machine Learning Research, 2025.

Anke Tang, Li Shen, Yong Luo, Shuai Xie, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. Zero-shot sparse mixture of low-rank experts construction from pre-trained foundation models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 48(2):1145–1157, 2026. doi: 10.1109/TPAMI.2025.3612480.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In EMNLP Workshop, pp. 353–355, November 2018.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. Transactions of the Association for Computational Linguistics, 7:625–641, 09 2019. ISSN 2307-387X. doi: 10.1162/tacl_a_00290. URL https://doi.org/10.1162/tacl_a_00290.

Yongxian Wei, Runxi Cheng, Weike Jin, Enneng Yang, Li Shen, LU HOU, SiNan Du, Chun Yuan, Xiaochun Cao, and Dacheng Tao. Optmerge: Unifying multimodal llm capabilities and modalities via model merging. In International Conference on Learning Representations, volume 2026, pp. 61571–61595, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/64741c1e80cc3bbed205b1c2b040dfa8-Paper-Conference.pdf.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In North American Chapter of the Association for Computational Linguistics, pp. 1112–1122, June 2018. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101/.

Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. International Journal of Computer Vision, 119: 3–22, 2016.

Jing Xu, Jiazheng Li, and Jingzhao Zhang. Scalable model merging with progressive layer-wise distillation. In International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=xX8NJShgny.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In Advances in Neural Information Processing Systems, 2023.

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. In International Conference on Machine Learning, 2024a. URL https://openreview.net/forum?id=Sbl2keQEML.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In International Conference on Learning Representations, 2024b. URL https://openreview.net/forum?id=nZP6NgD3QY.

Wen Zhang, Lingfei Deng, Lei Zhang, and Dongrui Wu. A survey on negative transfer. IEEE/CAA Journal of Automatica Sinica, 10(2):305–329, 2023. doi: 10.1109/JAS.2022.106004.

Zhanpeng Zhou, Zijun Chen, Yilan Chen, Bo Zhang, and Junchi Yan. On the emergence of cross-task linearity in pretraining-finetuning paradigm. In International Conference on Machine Learning, 2024. URL https://openreview.net/forum?id=qg6AlnpEQH.

Algorithm 1: Feature Core Preservation (FCP)

Input: Pretrained model W_{pre}; merged task vector T_{merge}; calibration sets $\{\mathcal{C}_k\}_{k=1}^{K}$
Output: Merged model W_{merge}

```
// Collect activations without using labels
```

Run W_{pre} on $\bigcup_{k=1}^{K} \mathcal{C}_k$ to collect layer inputs and outputs;
for $l = 1$ **to** L **do**

```
// Arrange feature vectors as columns
```

$X^{(l)} \leftarrow [X_1^{(l)}, \ldots, X_K^{(l)}]$;
$Z^{(l)} \leftarrow [Z_1^{(l)}, \ldots, Z_K^{(l)}]$;

```
// Construct the feature protection subspaces
```

$U_{X,r} \leftarrow \text{TopSingularvectors}(X^{(l)})$;
$U_{Z,r} \leftarrow \text{TopSingularvectors}(Z^{(l)})$;

```
// Filter the merged task vector and update the layer
```

$\widehat{T}^{(l)} \leftarrow (I - U_{Z,r}U_{Z,r}^{\top})T_{\text{merge}}^{(l)}(I - U_{X,r}U_{X,r}^{\top})$;
$W_{\text{merge}}^{(l)} \leftarrow W_{\text{pre}}^{(l)} + \lambda\widehat{T}^{(l)}$;

return W_{merge};

A Algorithm Details

Algorithm 1 summarizes the complete procedure of Feature Core Preservation (FCP). Given a pretrained model W_{pre}, a merged task vector T_{merge} produced by any existing merging rule, and a small set of unlabeled calibration examples for each task, FCP first performs forward passes through the pretrained model to collect layer-wise input and output features. For each layer, it arranges the collected features as columns of matrices $X^{(l)}$ and $Z^{(l)}$, and computes the leading r singular vectors $U_{X,r}$ and $U_{Z,r}$. These subspaces define the feature core. The merged task vector is then projected onto the orthogonal complements of these subspaces, yielding a filtered update $\widehat{T}^{(l)}$. Finally, the filtered update is added to the pretrained weights to obtain the merged layer. The entire procedure requires only forward passes and no labeled data, making it lightweight and compatible with arbitrary merging rules.

B Local Loss Bounds and Perturbation Tolerance

We formalize the sensitivity analysis in Section 4. Recall that we decompose the merged update in $\mathcal{S}$ as

$$T_{\mathcal{S}} = \text{Proj}_{\mathcal{S}}(T_{\text{merge}}) = T_{\mathcal{S}}^{\star} + \epsilon_{\mathcal{S}},$$

where $T_{\mathcal{S}}^{\star}$ is the ideal task update in $\mathcal{S}$ and $\epsilon_{\mathcal{S}}$ is the merging error. We are interested in the additional loss induced by the error $\epsilon_{\mathcal{S}}$ on top of the ideal update $T_{\mathcal{S}}^{\star}$. Define the error-induced loss increase as

$$\Delta L_k(\epsilon_{\mathcal{S}}) = L_k(W_{\text{pre}} + T_{\mathcal{S}}^{\star} + \epsilon_{\mathcal{S}}) - L_k(W_{\text{pre}} + T_{\mathcal{S}}^{\star}).$$

Assume that L_k is continuously differentiable on a neighborhood of the segment between $W_{\text{pre}} + T_{\mathcal{S}}^{\star}$ and $W_{\text{pre}} + T_{\mathcal{S}}^{\star} + \epsilon_{\mathcal{S}}$, and that its gradient is β-Lipschitz on this set:

$$\|\nabla L_k(A) - \nabla L_k(B)\|_F \leq \beta\|A - B\|_F.$$

For a perturbation budget ρ, the worst-case error-induced loss increase is

$$\mathcal{R}_k(\rho; \mathcal{S}) = \max_{\substack{\epsilon_{\mathcal{S}} \in \mathcal{S} \\ \|\epsilon_{\mathcal{S}}\|_F \leq \rho}} \Delta L_k(\epsilon_{\mathcal{S}}).$$

By the fundamental theorem of calculus,

$$\begin{aligned}\Delta L_k(\epsilon_{\mathcal{S}}) = {} & \langle \nabla L_k(W_{\text{pre}} + T_{\mathcal{S}}^{\star}), \epsilon_{\mathcal{S}} \rangle_F \\ & + \int_0^1 \langle \nabla L_k(W_{\text{pre}} + T_{\mathcal{S}}^{\star} + t\epsilon_{\mathcal{S}}) - \nabla L_k(W_{\text{pre}} + T_{\mathcal{S}}^{\star}), \epsilon_{\mathcal{S}} \rangle_F \, dt. \end{aligned} \tag{16}$$

Using Cauchy–Schwarz and the Lipschitz assumption, the integral is bounded by

$$\int_0^1 \beta t \|\epsilon_{\mathcal{S}}\|_F^2 \, dt = \frac{\beta}{2} \|\epsilon_{\mathcal{S}}\|_F^2.$$

If we approximate the gradient at the ideal update by the pretrained gradient, i.e.,

$$\nabla L_k(W_{\text{pre}} + T_{\mathcal{S}}^\star) \approx G_k,$$

then the first-order term becomes

$$\langle G_k, \epsilon_{\mathcal{S}} \rangle_F = \langle G_k^{\text{prot}}, \epsilon_{\mathcal{S}} \rangle_F.$$

Therefore,

$$\mathcal{R}_k(\rho; \mathcal{S}) \leq \rho \|G_k^{\text{prot}}\|_F + \frac{\beta}{2} \rho^2.$$

This bound quantifies the worst-case additional loss caused by the merging error in $\mathcal{S}$ under a local smoothness assumption.

C Experiment Details

C.1 Baselines

We benchmark five representative model merging approaches. For each approach, we report the original merger as well as its CSP- and FCP-augmented versions.

- **Task Arithmetic** (Ilharco et al., 2023) encodes each task as a task vector, defined as the difference between the fine-tuned and pretrained parameters, and merges them by adding a scaled sum of these vectors to the pretrained model.
- **LOT Merging** (Sun et al., 2025b) aligns the intermediate feature representations of the merged model with those of the individual task experts, thereby encouraging feature-level consistency during merging.
- **ISO-C** (Marczak et al., 2025) obtains a shared merging subspace by applying SVD to the summed task matrix, and then equalizes the singular values to prevent a few directions from dominating the merged update and to improve cross-task balance.
- **ISO-CTS** (Marczak et al., 2025) extends ISO-C by keeping the leading common directions and supplementing them with task-specific residual directions extracted after projecting out the common subspace. The combined basis is orthogonalized and isotropically scaled, which enhances robustness when merging many diverse tasks.
- **TSV** (Gargiulo et al., 2025) builds the merged task vector along the principal components of per-task updates, prioritizing high-variance directions that are expected to capture informative shared structure.

We additionally include the pretrained model, individually fine-tuned models, and traditional multi-task training as reference points where applicable. All variants within a comparison use identical task checkpoints, so the protection mechanism modifies only the merged update without retraining the constituent task models.

C.2 Datasets

Vision tasks. Our vision evaluation uses eight image classification datasets:

- **SUN397** (Xiao et al., 2016): 108,754 images covering 397 scene categories, with at least 100 images per class.
- **Stanford Cars (Cars)** (Krause et al., 2013): 16,185 images from 196 car classes, with an approximately balanced train/test split.
- **RESISC45** (Cheng et al., 2017): 31,500 remote-sensing images spanning 45 scene types, roughly 700 images per class.

- **EuroSAT** (Helber et al., 2019): 27,000 geo-referenced satellite images labeled into 10 classes.
- **SVHN** (Netzer et al., 2011): 10-way digit recognition from street-view imagery, containing 73,257 training and 26,032 test images, plus 531,131 additional training samples.
- **GTSRB** (Stallkamp et al., 2011): 43-category traffic sign recognition with over 50,000 images.
- **MNIST** (LeCun & Cortes, 2010): handwritten digit classification with 60,000 training and 10,000 test images across 10 classes.
- **DTD** (Cimpoi et al., 2014): 5,640 texture images from 47 categories, approximately 120 images per class.

For larger task collections, we further evaluate the 8-, 14-, and 20-task vision settings provided by FusionBench (Tang et al., 2025) with ViT-B/32, ViT-B/16, and ViT-L/14 backbones.

Language tasks. Our language evaluation uses eight GLUE tasks (Wang et al., 2018):

- **CoLA** (Warstadt et al., 2019): The Corpus of Linguistic Acceptability, a binary classification task for judging grammatical acceptability.
- **MNLI** (Williams et al., 2018): Multi-Genre Natural Language Inference, a three-way entailment classification task.
- **MRPC** (Dolan & Brockett, 2005): Microsoft Research Paraphrase Corpus, a binary paraphrase detection task.
- **QNLI**: Question Natural Language Inference, a binary classification task derived from SQuAD.
- **QQP**: Quora Question Pairs, a binary duplicate question detection task.
- **RTE**: Recognizing Textual Entailment, a binary entailment classification task.
- **SST-2** (Socher et al., 2013): Stanford Sentiment Treebank, a binary sentiment classification task.
- **STS-B** (Cer et al., 2017): Semantic Textual Similarity Benchmark, a regression task for sentence similarity.

C.3 Environment

All experiments were performed on a single Ubuntu 16.04 workstation equipped with 2 Intel Xeon CPUs (2.60 GHz), 512 GB RAM, and 8 NVIDIA RTX 3090 GPUs. The implementation used Python 3.8.

D Additional Experiments

D.1 More Ablation Study

This section extends the component-wise ablation to ViT-L/14, with ViT-B/32 results included for comparison, as shown in Table 7. The conclusion is consistent with that in the main paper, where the input-side preservation plays the dominant role, output-side preservation alone brings limited benefit, and combining both sides yields a further, though modest, improvement. The dominance of input-side preservation is expected, since the input-side subspace is connected with the gradient space and thus more directly captures the directions that are sensitive to merging interference. The same trend holds on both architectures, indicating that this behavior generalizes across backbones and merging methods.

D.2 Analysis of the Number of Exemplars

We evaluate the sensitivity of FCP to the size of the calibration set by varying the number of exemplars per task from 1 to 128. Table 8 reports the results for Task Arithmetic on ViT-B/32 and

Table 7: Ablation studies when merging ViT-B/32 and ViT-L/14 models on eight vision tasks.

Base Method	Input-Side	Output-Side	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg Acc
ViT-B/32											
Task Arithmetic			55.2	54.9	66.7	78.9	80.2	69.7	97.3	50.4	69.1
	✓		66.0	65.2	76.5	85.7	**82.8**	**72.6**	97.0	55.5	75.2
		✓	60.4	58.8	69.3	78.9	79.6	69.6	96.6	52.1	70.7
	✓	✓	**66.3**	**65.5**	**76.8**	**85.8**	82.6	72.3	**97.2**	**55.9**	**75.3**
TSV			70.1	72.1	85.9	94.3	90.9	91.2	99.2	68.8	84.1
	✓		72.6	75.0	**88.8**	**97.0**	91.7	**93.1**	99.3	71.3	86.1
		✓	69.5	71.2	85.2	94.3	**92.4**	92.1	99.3	69.3	84.2
	✓	✓	**72.7**	**75.2**	88.7	96.9	92.0	**93.1**	**99.4**	**71.8**	**86.2**
ViT-L/14											
Task Arithmetic			73.9	82.1	86.6	94.1	87.9	86.7	98.9	65.6	84.5
	✓		**77.2**	**88.2**	92.0	**97.3**	**90.4**	**93.2**	**99.2**	72.5	88.7
		✓	73.2	79.9	84.7	90.2	89.6	87.6	99.1	65.0	83.6
	✓	✓	**77.2**	**88.2**	**92.2**	**97.3**	**90.4**	**93.2**	**99.2**	**72.7**	**88.8**
TSV			79.2	89.8	93.8	98.9	95.6	96.5	99.5	80.0	91.7
	✓		82.0	90.8	94.9	99.4	**96.8**	98.4	**99.7**	83.2	93.1
		✓	78.1	87.8	92.1	98.6	95.8	96.9	99.5	77.7	90.8
	✓	✓	**82.4**	**91.4**	**95.8**	**99.5**	96.0	98.2	99.6	**83.5**	**93.4**

Table 8: Multi-task performance when merging eight vision tasks. The checkpoints are obtained from Ilharco et al. (2023).

Method	No. of Exemplars	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg Acc
ViT-B/32										
Task Arithmetic	-	55.2	54.9	66.7	78.9	80.2	69.7	97.3	50.4	69.1
Task Arithmetic w/ FCP	1	66.2	65.5	76.2	85.8	82.8	72.7	97.3	55.5	75.3
Task Arithmetic w/ FCP	2	66.3	65.4	76.6	86.1	82.7	72.6	97.2	55.8	75.3
Task Arithmetic w/ FCP	4	66.2	65.3	76.5	85.3	82.8	72.5	97.2	55.7	75.2
Task Arithmetic w/ FCP	8	66.3	65.5	76.9	85.9	82.6	72.3	97.3	56.0	75.3
Task Arithmetic w/ FCP	16	66.2	65.3	76.8	86.0	82.6	72.3	97.2	56.0	75.3
Task Arithmetic w/ FCP	32	66.3	65.5	76.8	85.7	82.5	72.3	97.2	56.0	75.3
Task Arithmetic w/ FCP	64	66.3	65.5	76.8	85.8	82.6	72.3	97.2	55.9	75.3
Task Arithmetic w/ FCP	128	66.3	65.5	76.8	85.7	82.6	72.2	97.2	56.0	75.3
ViT-L/14										
Task Arithmetic	-	73.9	82.1	86.6	94.1	87.9	86.7	98.9	65.6	84.5
Task Arithmetic w/ FCP	1	77.2	88.1	92.1	97.4	90.9	93.2	99.2	72.6	88.8
Task Arithmetic w/ FCP	2	77.3	88.1	92.0	97.3	90.3	93.2	99.2	72.5	88.7
Task Arithmetic w/ FCP	4	77.2	88.2	92.2	97.4	90.4	93.2	99.2	72.5	88.8
Task Arithmetic w/ FCP	8	77.2	88.2	92.2	97.4	90.4	93.2	99.2	72.7	88.8
Task Arithmetic w/ FCP	16	77.3	88.3	92.2	97.3	90.4	93.2	99.2	72.6	88.8
Task Arithmetic w/ FCP	32	77.2	88.2	92.3	97.4	90.4	93.1	99.2	72.6	88.8
Task Arithmetic w/ FCP	64	77.2	88.2	92.2	97.3	90.4	93.2	99.2	72.7	88.8
Task Arithmetic w/ FCP	128	77.2	88.2	92.2	97.3	90.4	93.1	99.2	72.6	88.8

ViT-L/14. The performance of FCP is remarkably stable across the entire range. On ViT-B/32, Task Arithmetic improves from 69.1% to 75.3% with a single exemplar per task, and remains at 75.2–75.3% as the number of exemplars increases to 128. On ViT-L/14, the average accuracy increases from 84.5% to 88.8% with one exemplar and stays within 88.7–88.8% for all larger calibration sets.

Notably, using one exemplar per task means that the calibration data contains only a single class for each task, resulting in an extremely class-imbalanced and label-agnostic setting. A plausible explanation is that each image in a ViT is tokenized into many patch tokens, so a single exemplar still provides a large number of feature vectors for estimating the input-feature covariance. Moreover, FCP only estimates the dominant principal subspace of layer inputs and does not use calibration labels, making it insensitive to label distribution. The same trend is observed for other merging rules, suggesting that the effectiveness of FCP stems from the stable, task-agnostic structure of the input-feature space rather than from the specific composition of the calibration set.

D.3 Analysis of Sample Robustness

Table 9 evaluates the sensitivity of FCP to the random batch used for merging. Across five random batches, the standard deviation of the average accuracy is at most 0.02 in all four settings, and the standard deviation on any individual task never exceeds 0.09. For example, TSV w/ FCP varies only within 93.34–93.36 on ViT-L/14 and 86.24–86.30 on ViT-B/32, while Task Arithmetic w/ FCP exhibits similarly small fluctuations. This stability is not specific to a particular backbone or merging

Table 9: Multi-task performance when merging eight vision tasks using five different random batches for FCP. The checkpoints are obtained from Ilharco et al. (2023). All accuracies are reported in percentages. Mean and STD summarize the five runs, with STD computed using a divisor of five.

Method	Random Batch	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg Acc
					ViT-B/32					
Task Arithmetic w/ FCP	1	66.37	65.43	76.87	86.09	82.58	72.26	97.23	55.85	75.33
Task Arithmetic w/ FCP	2	66.32	65.43	76.76	85.94	82.70	72.25	97.20	55.90	75.31
Task Arithmetic w/ FCP	3	66.34	65.45	76.78	85.98	82.57	72.24	97.20	55.96	75.31
Task Arithmetic w/ FCP	4	66.34	65.46	76.78	85.87	82.71	72.22	97.20	55.96	75.32
Task Arithmetic w/ FCP	5	66.37	65.44	76.78	85.83	82.63	72.22	97.20	55.85	75.29
Mean	-	66.35	65.44	76.79	85.94	82.64	72.24	97.21	55.90	75.31
Std.	-	0.02	0.01	0.04	0.09	0.06	0.01	0.01	0.05	0.01
TSV w/ FCP	1	72.68	75.28	88.62	97.04	92.10	93.05	99.34	72.13	86.28
TSV w/ FCP	2	72.67	75.25	88.56	96.96	92.13	93.07	99.35	72.18	86.27
TSV w/ FCP	3	72.67	75.28	88.65	97.04	92.08	93.08	99.35	72.23	86.30
TSV w/ FCP	4	72.67	75.25	88.62	97.04	92.04	92.98	99.35	72.13	86.26
TSV w/ FCP	5	72.64	75.23	88.67	97.00	92.05	93.02	99.35	71.97	86.24
Mean	-	72.67	75.26	88.62	97.01	92.08	93.04	99.35	72.13	86.27
Std.	-	0.01	0.02	0.04	0.03	0.03	0.04	0.00	0.09	0.02
					ViT-L/14					
Task Arithmetic w/ FCP	1	77.28	88.38	92.00	97.85	90.38	93.10	99.22	72.61	88.85
Task Arithmetic w/ FCP	2	77.28	88.38	92.03	97.81	90.42	93.13	99.22	72.50	88.85
Task Arithmetic w/ FCP	3	77.30	88.35	92.03	97.78	90.48	93.16	99.22	72.61	88.86
Task Arithmetic w/ FCP	4	77.25	88.35	92.10	97.78	90.38	93.13	99.22	72.61	88.85
Task Arithmetic w/ FCP	5	77.26	88.42	92.08	97.78	90.40	93.15	99.23	72.61	88.87
Mean	-	77.27	88.38	92.05	97.80	90.41	93.13	99.22	72.59	88.86
Std.	-	0.02	0.03	0.03	0.03	0.04	0.02	0.00	0.04	0.01
TSV w/ FCP	1	82.35	91.42	95.76	99.50	96.42	98.15	99.58	83.54	93.34
TSV w/ FCP	2	82.43	91.42	95.78	99.56	96.43	98.17	99.59	83.49	93.36
TSV w/ FCP	3	82.40	91.40	95.78	99.56	96.41	98.17	99.59	83.49	93.35
TSV w/ FCP	4	82.41	91.41	95.79	99.50	96.39	98.16	99.58	83.59	93.35
TSV w/ FCP	5	82.38	91.42	95.79	99.50	96.38	98.17	99.59	83.59	93.35
Mean	-	82.39	91.41	95.78	99.52	96.41	98.16	99.59	83.54	93.35
Std.	-	0.03	0.01	0.01	0.03	0.02	0.01	0.01	0.05	0.01

method. FCP exhibits similarly low variance with Task Arithmetic and TSV, on both ViT-B/32 and ViT-L/14.

D.4 FCP Core Approximates the GCP Core across Models and Layer Types

We further examine whether the feature-derived FCP core provides a faithful proxy for the gradient-sensitive subspace. Figure 4 compares the rank-normalized gradient energy of the CSP, GCP, and FCP cores on all 20 tasks for ViT-B/32 and ViT-L/14. Across all downstream tasks and both backbones, the FCP core consistently captures substantially higher gradient energy than the CSP core, typically by one to two orders of magnitude. More importantly, its energy is close to that of the GCP core, indicating that the dominant input-feature subspace effectively recovers the main gradient directions. We also note that the overall gradient energy is lower for ViT-L/14 than for ViT-B/32. While multiple factors may contribute, a plausible explanation is that the larger and better-pretrained ViT-L/14 model already attains lower task loss at the pretrained initialization, leading to smaller gradients.

Figure 5 breaks down the comparison by parameter type for ViT-B/32, covering the attention projections (Q, K, V, Out) and the MLP layers (FC1, FC2). The same pattern holds across all six layer types, where the FCP core exhibits gradient energy comparable to GCP and far exceeding CSP. This consistency suggests that the feature–gradient relationship is not specific to a particular module but holds broadly across the architecture.

D.5 Sensitivity Analysis of the Protection Rank r

We study the effect of the protection rank r on both the gradient energy captured by the FCP core and the final merging performance. Figure 6 reports the rank-normalized gradient energy of the FCP core as r increases. The energy decreases monotonically with r, indicating that the leading singular directions of the input-feature matrix X correspond to the highest-gradient directions, while subsequent directions contribute progressively lower gradient energy. This trend provides empirical support for our assumption that the dominant input-feature subspace is aligned with the gradient-

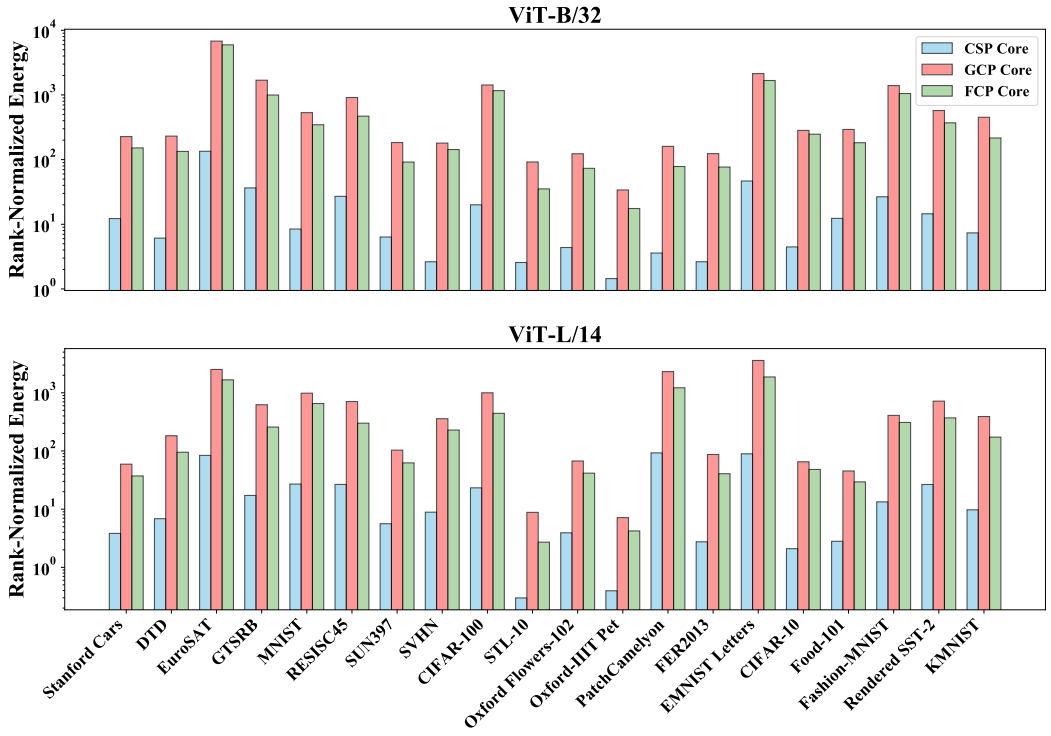


Figure 4: Rank-normalized gradient energy in the core subspaces on all 20 tasks, checkpoints are from Tang et al. (2025).

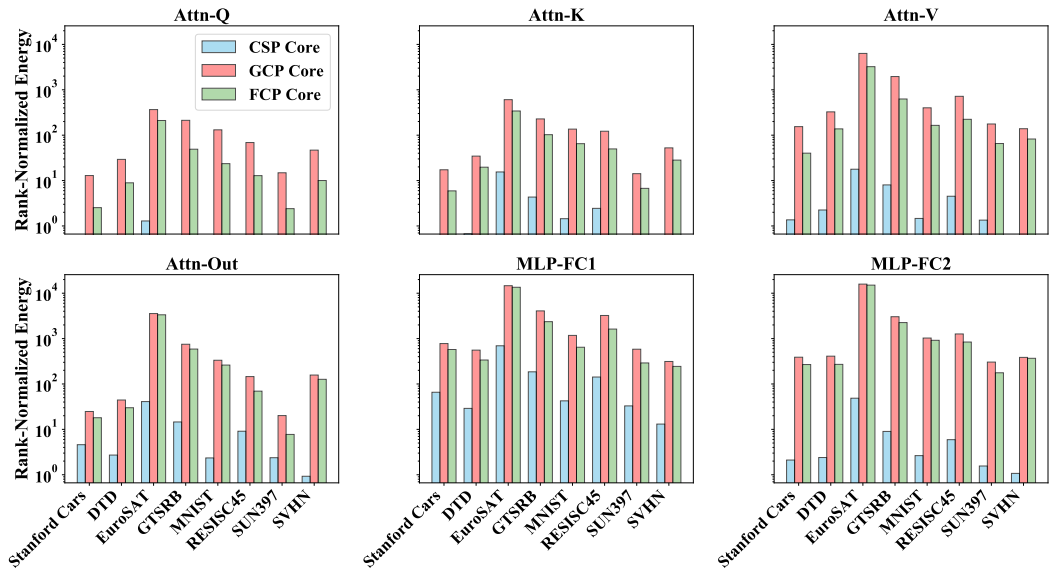


Figure 5: Rank-normalized gradient energy in the core subspaces across layer types when merging eight ViT/B-32 models, checkpoints are from Tang et al. (2025).

sensitive parameter subspace. We also observe that the gradient energy is consistently lower for ViT-L/14 than for ViT-B/32, which is consistent with the earlier comparison and may reflect the stronger pretrained initialization of the larger model.

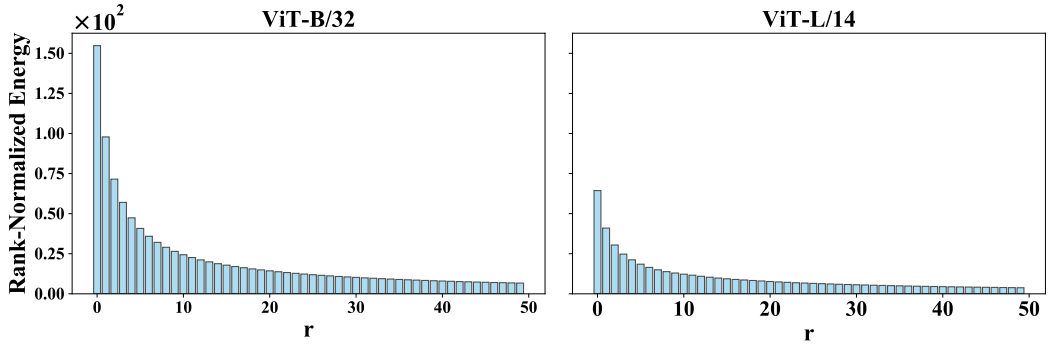


Figure 6: Rank-normalized gradient energy in the core subspaces across different r when merging eight ViT/B-32 models, checkpoints are from Tang et al. (2025).

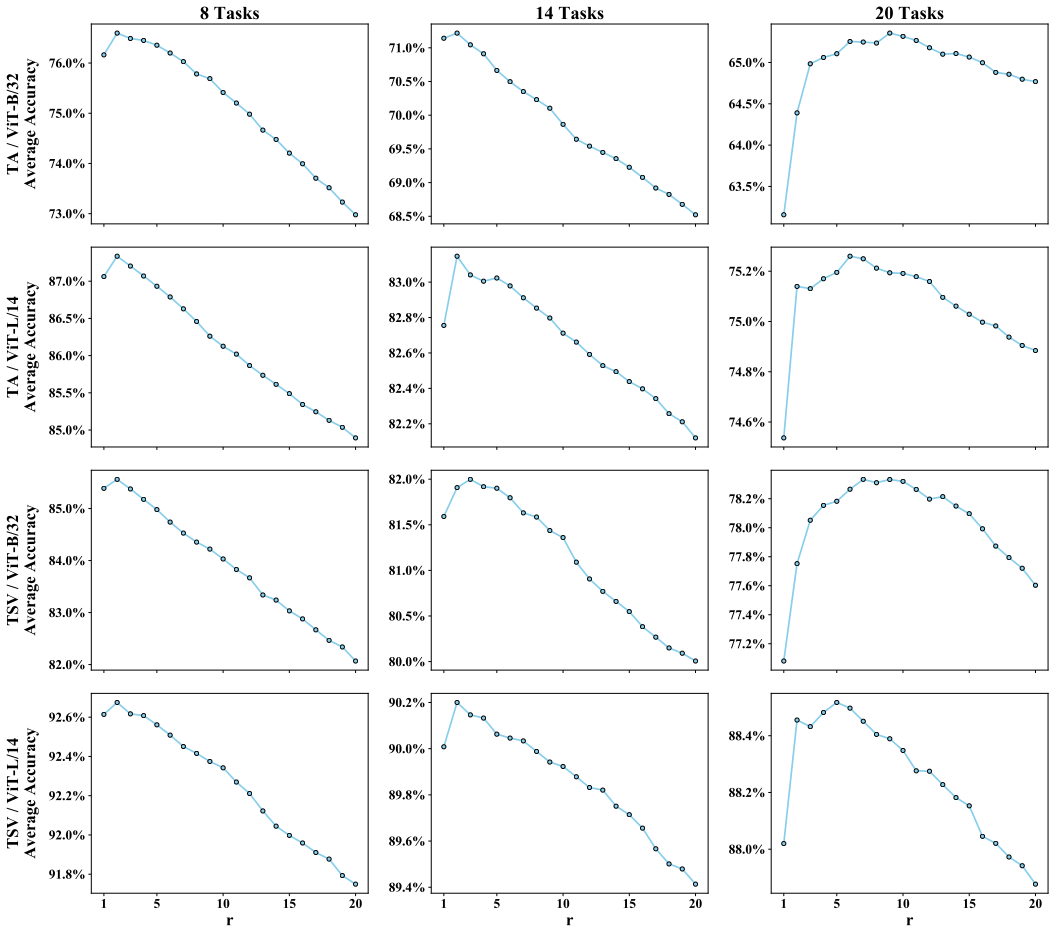


Figure 7: Average accuracy across different architectures with various r, checkpoints are from Tang et al. (2025).

Figure 7 shows the average accuracy as a function of r for Task Arithmetic and TSV on ViT-B/32 and ViT-L/14, with 8, 14, and 20 tasks. When the number of tasks is relatively small, $r = 1$ or $r = 2$ achieves the best performance. As the number of tasks increases to 20, the optimal rank shifts to around $r = 5$. A plausible explanation is that merging more tasks introduces stronger interference among task vectors, making it more beneficial to protect a larger portion of the pretrained parameter space. Across all evaluated settings, $r = 2$ consistently delivers competitive performance and is

Table 10: Multi-task performance when merging ViT-B/32 models on eight vision tasks. The checkpoints are obtained from Ilharco et al. (2023).

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg Acc
ViT-B/32									
Pretrained	62.3	59.7	60.7	45.5	31.4	32.6	48.5	43.8	48.0
Individual	75.3	77.7	96.1	99.7	97.5	98.7	99.7	79.4	90.5
Traditional MTL	73.9	74.4	93.9	98.2	95.8	98.9	99.5	77.9	89.1
Task Arithmetic	55.2	54.9	66.7	78.9	80.2	69.7	**97.3**	50.4	69.1
Task Arithmetic w/ CSP	62.9	61.5	71.8	83.4	79.5	70.5	96.8	55.1	72.7
Task Arithmetic w/ FCP	66.3	65.5	**76.8**	85.8	**82.6**	**72.3**	97.2	55.9	75.3
Task Arithmetic w/ CSP + FCP	**66.6**	**65.7**	**76.8**	**86.7**	82.0	72.0	97.2	**56.4**	**75.4**
TSV	70.1	72.1	85.9	94.3	90.9	91.2	99.2	68.8	84.1
TSV w/ CSP	71.1	73.4	86.3	95.4	**92.3**	92.6	99.3	70.7	85.1
TSV w/ FCP	72.7	75.2	88.7	96.9	92.0	**93.1**	**99.4**	71.8	86.2
TSV w/ CSP + FCP	**72.8**	**75.3**	**88.8**	**97.2**	91.9	93.0	99.3	**72.3**	**86.3**
ViT-L/14									
Task Arithmetic	73.9	82.1	86.6	94.1	87.9	86.7	98.9	65.6	84.5
Task Arithmetic w/ CSP	75.1	83.6	88.0	95.6	90.2	89.9	99.1	67.8	86.2
Task Arithmetic w/ FCP	**77.2**	88.2	92.2	97.3	**90.4**	**93.2**	**99.2**	72.7	88.8
Task Arithmetic w/ CSP + FCP	**77.2**	**88.3**	**92.3**	**97.9**	90.3	93.1	**99.2**	**72.9**	**88.9**
TSV	79.2	89.8	93.8	98.9	95.6	96.5	99.5	80.0	91.7
TSV w/ CSP	80.5	90.2	94.2	99.4	**96.5**	97.8	**99.7**	82.1	92.5
TSV w/ FCP	82.4	91.4	**95.8**	**99.5**	96.0	**98.2**	99.6	**83.5**	93.3
TSV w/ CSP + FCP	**82.6**	**91.8**	95.7	**99.5**	96.4	**98.2**	**99.7**	**83.5**	**93.4**

therefore adopted as our default choice. These results suggest that the optimal rank depends mildly on the task scale, while a small fixed rank remains a robust and practical setting.

D.6 Compatibility and Complementarity of FCP and CSP

We further examine whether FCP is compatible with CSP, which preserves the dominant singular subspace of the pretrained weights. Although both methods protect pretrained parameters, they target different regions of the parameter space. CSP focuses on directions where the pretrained model is already locally optimal, and downstream gradients are weak, whereas FCP protects input-feature directions that are more sensitive to downstream updates. The two mechanisms are therefore expected to be complementary.

Table 10 reports the results of combining CSP and FCP on eight vision tasks with ViT-B/32 and ViT-L/14. Across all four evaluated settings, CSP + FCP achieves the best average accuracy, although the additional gain over FCP alone is modest. On ViT-B/32, CSP + FCP improves Task Arithmetic from 69.1% to 75.4% and TSV from 84.1% to 86.3%. On ViT-L/14, it raises Task Arithmetic from 84.5% to 88.9% and TSV from 91.7% to 93.4%. In most cases, the combination also improves over FCP on several individual tasks, such as EuroSAT and DTD for Task Arithmetic, and SUN397 and Cars for TSV. We observe only minor drops on a few tasks, e.g., SVHN for TSV on ViT-B/32 and RESISC45 for TSV on ViT-L/14, which is expected because the two protection subspaces are not perfectly orthogonal in practice.

Overall, these results support the view that CSP and FCP protect complementary aspects of the pretrained parameter space. CSP preserves the low-gradient principal subspace of the pretrained weights, while FCP preserves the high-gradient input-feature subspace. Combining them provides a simple way to extend subspace protection without additional training or backward propagation. Since the gains over FCP alone are small but consistent on average, we view FCP as the primary protection module and CSP + FCP as an optional stronger variant when an additional validation-based selection is affordable.

D.7 Analysis of Computational Complexity

Table 11 compares the computational complexity in merging time on a single RTX 3090 GPU. Training-based methods such as Surgery and AdaMerging are the most expensive, requiring thousands of seconds on both backbones. In contrast, FCP takes only 37s on ViT-B/32 and 126s on ViT-L/14, comparable to PCB Merging and LOT Merging, and over two orders of magnitude faster

Table 11: Computational complexity comparison (in seconds) for merging ViT-B/32 and ViT-L/14 models across eight vision tasks, measured on a single RTX 3090 GPU.

Method	Surgery	AdaMerging	TATR	PCB Merging	LOT Merging	CSP	FCP (ours)
ViT-B/32	12621	8276	176	43	44	5	37
ViT-L/14	36826	16299	283	131	161	10	126

than TA w/ Surgery and AdaMerging. Although FCP is slightly slower than CSP due to forward feature collection and SVD, its overhead remains small and scales gracefully with model size. Since FCP requires no labels or backpropagation, it offers a practical efficiency–effectiveness trade-off for existing merging pipelines.