TRAINING-FREE TRANSFORMER MERGING VIA SEQUENTIAL LOCAL OPERATOR ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Training-free model merging aims to combine multiple fine-tuned models into a single model without further optimization on labeled data. Yet, in transformers, independently merging individual layers can affect a shared attention computation because the query–key and value–output operators depend on composed matrices, overlooking the *functional* structure. Moreover, when merging earlier components, downstream components receive different activations than they do in the original model, thus, the merged and original execution paths no longer match. In this paper, we introduce *Sequential Local Operator Alignment*, a training-free method that merges transformers along the execution path of the partially merged model. Our method uses calibration data to estimate the local behavior of each functional component, aligns operators sequentially under the intermediate activation of the partially merged model, and subsequently factorizes the merged operators back into valid transformer parameters. We empirically show that this sequential step reduces error accumulation across layers. Furthermore, the proposed operator factorization step enables rank expansion, providing a principled mechanism for increasing multi-task capacity. We demonstrate that our approach generalizes across modalities, model scales, and varying numbers of tasks, from CLIP and RoBERTa to billion-parameter LLMs, and further extends naturally to the merging of LoRA-fine-tuned models. The results indicate improvements over strong merging baselines without requiring rank expansion, while optional expansion provides a further accuracy–inference-cost trade-off.

1 INTRODUCTION

Modern neural networks are often adapted into many task-specific variants from the same pretrained initialization (Bommasani et al., 2021; Devlin et al., 2019; Raffel et al., 2020; Radford et al., 2021a; Ding et al., 2023). However, maintaining a separate model for every task is expensive (Guo et al., 2021; Wortsman et al., 2022; Yadav et al., 2025), while retraining a single multi-task model can require substantial data, compute, and engineering effort (Ruder, 2017; Crawshaw, 2020; Yang et al., 2024b). *Training-free model merging* (Xu et al., 2024; Fu et al., 2025; Matena & Raffel, 2022; Ilharco et al., 2023; Ortiz-Jimenez et al., 2023; Jin et al., 2023) provides a practical alternative: given several independently fine-tuned models, the goal is to combine them into one model without further training. An effective merged model should preserve the task-specific capabilities of the original models while offering the deployment efficiency of a single network (Yang et al., 2026).

Many existing training-free merging methods operate directly on *individual* parameter tensors or *individual* activation spaces. For example, they average corresponding weight matrices (Wortsman et al., 2022), interpolate task vectors (Ilharco et al., 2023), or align intermediate activations layer by layer (Ainsworth et al., 2023; Singh & Jaggi, 2020). While simple and often effective, this layer-wise perspective is parameterization-dependent: a model's function is not uniquely tied to the particular tensors used to represent it (Zhao et al., 2025; Bona-Pellissier et al., 2023; Elhage et al., 2021; Dar et al., 2023). For instance, in a transformer attention head (Vaswani et al., 2017), the matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$, and $\mathbf{W}_O$ are not used as independent functional units. The query and key determine the pre-softmax attention scores through the combined bilinear operator $\boldsymbol{W}_Q\boldsymbol{W}_K^\top$, while the value and output determine the post-attention contribution through the composed operator $\boldsymbol{W}_V\boldsymbol{W}_O$. Thus, the functionally meaningful objects are the composed operators $\boldsymbol{W}_Q\boldsymbol{W}_K^\top$ and $\boldsymbol{W}_V\boldsymbol{W}_O$. A merging

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.

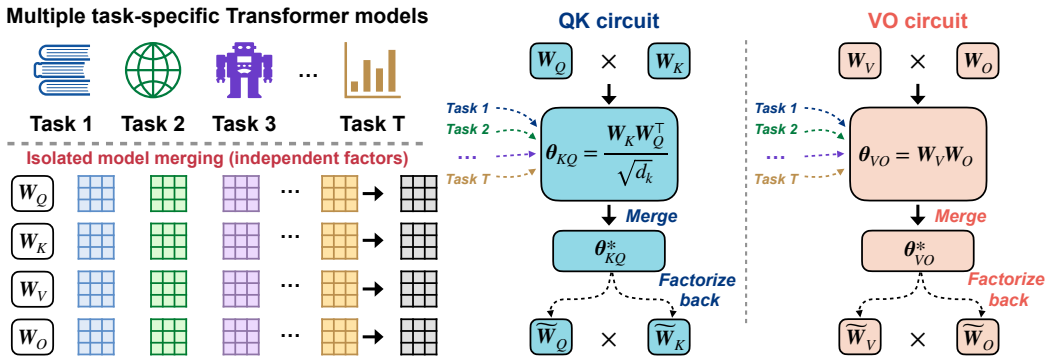


Figure 1: Illustration of *Sequential Local Operator Alignment*. Our method aligns the effective local functional operators, merges them sequentially, and then factorizes the aligned operators. This sequential operator-level view respects coupled attention computation, reduces error accumulation across layers, and enables rank expansion to have additional multi-task capacity.

procedure that independently combines $\mathbf{W}_Q$ and $\mathbf{W}_K$, or $\mathbf{W}_V$ and $\mathbf{W}_O$, may therefore fail to preserve a shared attention computation even when that computation is functionally equivalent.

A natural way is to merge objects at the level at which they enter the model's computation, rather than at the level of an arbitrary parameter decomposition. To this end, we introduce *Sequential Local Operator Alignment* (SLOA), a training-free framework for merging transformer models by aligning *local functional operators* (see Fig. 1 for the illustration). SLOA associates each model component with the local operator. For each component, SLOA constructs a merged operator from the task models. SLOA considers two alignment criteria: *(i) behavior matching*, on calibration inputs to the component, the merged operator should produce outputs close to those of the task-specific operators, and *(ii) data-aware operator regularization*, the merged operator is kept close to the task-specific operators under a metric induced by the calibration activations. This metric emphasizes directions that are active on the data. Importantly, our contribution is to instantiate them as a unified local-operator alignment framework for training-free transformer merging. The same objective applies across LayerNorm modules, linear maps, QK attention operators, and VO attention operators by changing only the behavior map and the induced metric.

A key feature of our framework is that alignment is performed *sequentially*. When earlier operators have already been merged, the hidden activations entering later components are no longer identical to the activations produced by any individual task model. If later components are merged only under the original task-model activations, the local objective assumes that all previous merges were perfect. Our approach instead evaluates the activation-matching term on the activations produced by the partially merged model. This makes each local merge corrective, aligning downstream operators along the actual execution path of the merged network and reducing error accumulation across layers.

Our operator view separates the merged operator from its eventual factorization. For QK and VO components, each individual task head has rank at most the head dimension, but the merged multi-task operator may require a higher-rank representation, as different tasks can use different subspaces. Therefore, we treat rank as a *capacity-control mechanism*. After merging an operator, we can factorize it at the original rank to preserve the architecture, or *expand* the rank to retain additional singular directions that capture task-specific behavior. Empirically, our method achieves state-of-the-art training-free merging performance across vision benchmarks, RoBERTa models, and billion-parameter LLMs. Moreover, SLOA naturally extends to models fine-tuned with LoRA (Hu et al., 2022), further broadening its applicability to parameter-efficient adaptation settings. It consistently outperforms baselines that merge individual weights or rely on isolated activation alignment, supporting the central claim that model merging should be performed at local functional operators.

2 Motivation: Limitations of Isolated Attention-Weight Merging

In a transformer attention head, the query, key, value, and output layers $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, $\boldsymbol{W}_V$, and $\boldsymbol{W}_O$ are often merged independently, or their corresponding intermediate activations are aligned separately. Given token activations $\boldsymbol{X}$, the query and key weights determine the pre-softmax attention

scores only through the composed QK operator $\boldsymbol{M}_{\mathrm{QK}} = \boldsymbol{W}_Q \boldsymbol{W}_K^\top$ and $\mathrm{scores}(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{M}_{\mathrm{QK}}\boldsymbol{X}^\top$. Similarly, for a fixed attention pattern $\boldsymbol{A}$, the value and output weights determine the head contribution only through the composed VO operator $\boldsymbol{M}_{\mathrm{VO}} = \boldsymbol{W}_V \boldsymbol{W}_O$ and $\mathrm{head}(\boldsymbol{X}) = \boldsymbol{A}\boldsymbol{X}\boldsymbol{M}_{\mathrm{VO}}$. Thus, the functional objects are the composed operators $\boldsymbol{M}_{\mathrm{QK}}$ and $\boldsymbol{M}_{\mathrm{VO}}$, whereas the individual matrices $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}_O$ are only one possible factorization of these operators.

Example 1. Consider two QK factorizations that represent the same attention-score operator $\boldsymbol{M}_{\mathrm{QK}}^{(1)} = \boldsymbol{W}_Q \boldsymbol{W}_K^\top$, and $\boldsymbol{M}_{\mathrm{QK}}^{(2)} = (\boldsymbol{W}_Q \boldsymbol{R})(\boldsymbol{W}_K \boldsymbol{R}^{-\top})^\top = \boldsymbol{W}_Q \boldsymbol{W}_K^\top$. Although the two models are functionally identical at the QK level, independently averaging the query and key weights gives $\bar{\boldsymbol{W}}_Q = \frac{1}{2}\boldsymbol{W}_Q(\boldsymbol{I} + \boldsymbol{R})$, and $\bar{\boldsymbol{W}}_K = \frac{1}{2}\boldsymbol{W}_K(\boldsymbol{I} + \boldsymbol{R}^{-\top})$. The resulting merged QK operator is $\bar{\boldsymbol{M}}_{\mathrm{QK}} = \bar{\boldsymbol{W}}_Q \bar{\boldsymbol{W}}_K^\top = \frac{1}{4}\boldsymbol{W}_Q(\boldsymbol{I} + \boldsymbol{R})(\boldsymbol{I} + \boldsymbol{R}^{-1})\boldsymbol{W}_K^\top$, which is not equal to $\boldsymbol{W}_Q \boldsymbol{W}_K^\top$. As a special case, let $\boldsymbol{R} = c\boldsymbol{I}$ with $c > 0$. Then the merged QK operator becomes $\bar{\boldsymbol{M}}_{\mathrm{QK}} = \frac{(1+c)(1+c^{-1})}{4}\boldsymbol{M}_{\mathrm{QK}}$. The multiplicative factor equals 1 only when $c = 1$, and it grows without bound as $c \to \infty$. Thus, for any input $\boldsymbol{X}$ s.t. $\boldsymbol{X}\boldsymbol{M}_{\mathrm{QK}}\boldsymbol{X}^\top \neq 0$, the merged attention scores can differ arbitrarily from the shared scores of the two original models

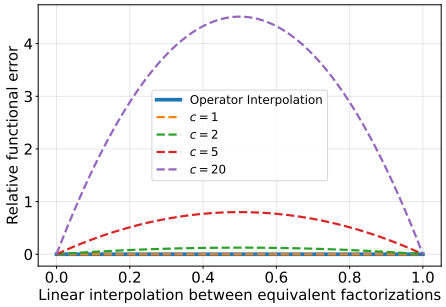


Figure 2: Illustration of the non-invariance of independent merging in attn. operator.

$$\left\|\boldsymbol{X}\bar{\boldsymbol{M}}_{\mathrm{QK}}\boldsymbol{X}^\top - \boldsymbol{X}\boldsymbol{M}_{\mathrm{QK}}\boldsymbol{X}^\top\right\|_F = \left|\frac{(1+c)(1+c^{-1})}{4} - 1\right| \left\|\boldsymbol{X}\boldsymbol{M}_{\mathrm{QK}}\boldsymbol{X}^\top\right\|_F . \tag{1}$$

The same argument applies to VO under $(\boldsymbol{W}_V\boldsymbol{R},\ \boldsymbol{R}^{-1}\boldsymbol{W}_O)$. Our solution is to treat $\boldsymbol{W}_Q\boldsymbol{W}_K^\top$ (and $\boldsymbol{W}_V\boldsymbol{W}_O$) as a single unit for merging (Fig. 2) and re-factorize after merging. This has two advantages: *(1)* we forgo the parameterization ambiguity and any search for a good aligning transformation; *(2)* the rank of the merged operator can be chosen freely at factorization time, allowing us to adjust capacity.

3 Methodology: Sequential Local Operator Alignment (SLOA)

The above example shows that independent merging can produce arbitrary functional errors when computation is performed inside a transformer. To address this, we introduce *Sequential Local Operator Alignment*, a training-free method that merges models by aligning the local functional operators.

3.1 Local functional operators

We define local functional operators as the parameterized object that determines a component's local computation once its input activation and any non-learned part are fixed. For example, the local operator of a linear layer is its weight matrix, while the local operator of an attention head is more naturally expressed through the composed QK and VO operators (see Fig. 1).

We write $\boldsymbol{\theta}$ for a generic local operator and $\mathcal{A}(\boldsymbol{Z};\boldsymbol{\theta})$ for its behavior map, where $\boldsymbol{Z}$ denotes the local input needed to evaluate the operator. For most operators, $\boldsymbol{Z}$ is the activation $\boldsymbol{X}$, and for VO, it includes both the attention matrix and the residual state. Table 1 summarizes the local functional operators used in our transformer merging procedure. The key design criterion is that, for fixed local input $\boldsymbol{Z}$, the behavior map $\mathcal{A}(\boldsymbol{Z};\boldsymbol{\theta})$ is linear in the local operator $\boldsymbol{\theta}$. Specifically, LayerNorm is linear in $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ after the normalization statistics are fixed; a linear layer is linear in $\boldsymbol{W}$; QK is linear in the composed score operator $\boldsymbol{W}_Q\boldsymbol{W}_K^\top/\sqrt{d_k}.$; and VO is linear in the composed operator $\boldsymbol{W}_V\boldsymbol{W}_O$.

The same template can be applied beyond the operators in Table 1 whenever a module admits a local representation that is linear in the chosen operator. For example, in a gated MLP, the down-projection can be treated as a local linear operator after the gated hidden activation has been formed. Operators that remain nonlinear in their parameters after conditioning on the local input would require either a smaller linear suboperator or an additional approximation, which are not the focus of this work.

Table 1: Local functional operators and their data-dependent metrics. Here, $\nu(\boldsymbol{X})$ denotes the normalized input before the learned LayerNorm's scale and shift, and $\boldsymbol{A}$ denotes the attention matrix

Component	Local operator $\boldsymbol{\theta}$	Behavior map $\mathcal{A}(\boldsymbol{Z};\boldsymbol{\theta})$	Metric
LayerNorm	$(\boldsymbol{\gamma},\boldsymbol{\beta})$	$\mathcal{A}(\boldsymbol{X};\boldsymbol{\gamma},\boldsymbol{\beta}) = \nu(\boldsymbol{X})\,\mathrm{diag}(\boldsymbol{\gamma}) + \mathbf{1}\boldsymbol{\beta}^\top$	$\mathrm{Cov}(\nu(\chi))$
Linear map	$\boldsymbol{W}$	$\mathcal{A}(\boldsymbol{X};\boldsymbol{W}) = \boldsymbol{X}\boldsymbol{W}$	$\boldsymbol{\Lambda}$
QK component	$\boldsymbol{W}_Q\boldsymbol{W}_K^\top/\sqrt{d_k}$	$\mathcal{A}(\boldsymbol{X};\boldsymbol{\theta}) = \boldsymbol{X}\boldsymbol{\theta}\boldsymbol{X}^\top$	$\boldsymbol{\Lambda}\otimes\boldsymbol{\Lambda}$
VO component	$\boldsymbol{W}_V\boldsymbol{W}_O$	$\mathcal{A}(\boldsymbol{A}\boldsymbol{X};\boldsymbol{\theta}) = \boldsymbol{A}\boldsymbol{X}\boldsymbol{\theta}$	$\boldsymbol{\Lambda}$

3.2 Activation-based local behavior matching

Suppose we merge T task-specific models. For a fixed local operator, let $\boldsymbol{\theta}_t$ denote the corresponding operator in model t, and let n_t be the number of calibration samples for that model. For calibration example i, let $\boldsymbol{Z}_{t,i}$ be the local input received by the operator when running the original task model t, and let $\widehat{\boldsymbol{Z}}_{t,i}$ be the local input received by the same operator when running the partially merged model. We also define the corresponding local state collections $\chi_t = \{\boldsymbol{Z}_{t,i}\}_{i=1}^{n_t}$ and $\widehat{\chi}_t = \{\widehat{\boldsymbol{Z}}_{t,i}\}_{i=1}^{n_t}$.

The task-model output for this local operator is $\mathcal{A}(\boldsymbol{Z}_{t,i};\boldsymbol{\theta}_t)$, while the merged operator, evaluated on the state it will actually receive during the merged model's forward pass, produces $\mathcal{A}(\widehat{\boldsymbol{Z}}_{t,i};\boldsymbol{\theta})$. Thus, the activation-based local alignment loss is then given by

$$\mathcal{L}_{\mathrm{act}}(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n_t}\sum_{i=1}^{n_t}\left\|\mathcal{A}(\widehat{\boldsymbol{Z}}_{t,i};\boldsymbol{\theta}) - \mathcal{A}(\boldsymbol{Z}_{t,i};\boldsymbol{\theta}_t)\right\|_F^2. \tag{2}$$

Equivalently, Eq. (2) first samples a model uniformly from $\{1,\dots,T\}$ and then samples a calibration example uniformly from that model. This balanced empirical average prevents models with more calibration examples from dominating the local objective. Note that Eq. (2) is functional: the merged operator is asked to reproduce the local outputs of the task-specific operators. However, with a small calibration set, activation matching alone can be underdetermined. Many operators may agree on the observed local activations while behaving differently on unobserved activations. We therefore add a data-dependent weight-space geometry term introduced in the next subsection.

3.3 Data-dependent weight geometry

Pure weight-space merging is stable and sample-efficient, but it treats all parameter directions as equally important. Pure activation matching focuses on functionality, but with only a small calibration set, it can be underdetermined and sensitive to the observed samples. SLOA considers these two views jointly: the activation term matches local behavior, while the geometry term keeps the merged operator close to the task-specific operators in directions that are important under the calibrations. For a standard one-sided linear operator, such as $\boldsymbol{X}\boldsymbol{W}$, we define

$$\|\boldsymbol{\theta}-\boldsymbol{\theta}_t\|_{\widehat{\chi}_t}^2 = \mathrm{Tr}\left[(\boldsymbol{\theta}-\boldsymbol{\theta}_t)^\top\boldsymbol{\Lambda}_t(\boldsymbol{\theta}-\boldsymbol{\theta}_t)\right], \tag{3}$$

where $\boldsymbol{\Lambda}_t = \alpha\,\mathrm{diag}\left(\mathrm{Cov}_{\boldsymbol{u}\in\phi(\widehat{\chi}_t)}(\boldsymbol{u})\right) + \epsilon\boldsymbol{I}$. Here, $\phi(\widehat{\chi}_t)$ denotes the collection of token-level vectors that multiply the operator in the local behavior map. For a linear map, $\phi(\boldsymbol{X}) = \boldsymbol{X}$; for the LayerNorm scale parameter, $\phi(\boldsymbol{X}) = \nu(\boldsymbol{X})$; and for VO, $\phi((\boldsymbol{A},\boldsymbol{X})) = \boldsymbol{A}\boldsymbol{X}$. The coefficient α controls the strength of the data-dependent geometry, and $\epsilon\boldsymbol{I}$ provides a ridge term for numerical stability. Note that Eq. (3) is a Mahalanobis distance in operator space: the merged operator is kept close to $\boldsymbol{\theta}_t$ along directions that are active on the partially merged states, and is regularized less along weakly observed ones. For QK operators, the behavior is bilinear, and thus, we use

$$\|\boldsymbol{\theta}-\boldsymbol{\theta}_t\|_{\widehat{\chi}_t}^2 = \left\|\boldsymbol{\Lambda}_t^{1/2}(\boldsymbol{\theta}-\boldsymbol{\theta}_t)\boldsymbol{\Lambda}_t^{1/2}\right\|_F^2. \tag{4}$$

After vectorization, this corresponds to a Mahalanobis metric with covariance $\boldsymbol{\Lambda}_t\otimes\boldsymbol{\Lambda}_t$. Considering the activation-matching loss and the data-dependent geometry gives the objective

$$\boldsymbol{\theta}_{\mathrm{merge}} = \underset{\boldsymbol{\theta}\in\Theta_{\mathrm{amb}}}{\arg\min}\ \mathcal{L}_{\mathrm{act}}(\boldsymbol{\theta}) + \frac{1}{T}\sum_{t=1}^{T}\|\boldsymbol{\theta}-\boldsymbol{\theta}_t\|_{\widehat{\chi}_t}^2. \tag{5}$$

Here, Θ_{amb} is the ambient operator space. For linear maps and LayerNorm affine parameters, this is the usual parameter space. For QK and VO operators, $\Theta_{\text{amb}} = \mathbb{R}^{d\times d}$ is the residual-space operator space; the rank constraint induced by attention-head width is imposed later during factorization. Because $\mathcal{A}(\widehat{\boldsymbol{Z}};\boldsymbol{\theta})$ is linear in $\boldsymbol{\theta}$ for fixed calibration states, Eq. (5) is a quadratic objective. Thus, each local operator can be merged by solving a closed-form regularized least-squares problem. Further derivations of the local operator alignment optimal solutions are in §D.

3.4 Sequential merging

The objective in Eq. (5) is applied sequentially. Let $\mathcal{O}$ be the set of local operators and write $o_a \prec o_b$ if the output or activation produced by operator o_a can affect the input to operator o_b. We choose a topological order $o_1, o_2, \ldots, o_M$ that is in line with this dependency relation. At step m, the operators $o_1, \ldots, o_{m-1}$ have been merged. For each model t and calibration i, we have two local inputs for o_m. First, $\boldsymbol{Z}_{t,i}^{(m)}$ is the local input produced by the original task model t. Second, $\widehat{\boldsymbol{Z}}_{t,i}^{(m)}$ is the local input produced by the partially merged model, whose predecessors $o_1, \ldots, o_{m-1}$ use the merged operators. The current operator is then merged by solving Eq. (5) with $\boldsymbol{Z}_{t,i} = \boldsymbol{Z}_{t,i}^{(m)}$, $\widehat{\boldsymbol{Z}}_{t,i} = \widehat{\boldsymbol{Z}}_{t,i}^{(m)}$, and $\boldsymbol{\theta}_t = \boldsymbol{\theta}_t^{(m)}$. This sequential construction is important. If the current operator were aligned only on $\boldsymbol{Z}_{t,i}^{(m)}$, the local objective would implicitly assume that all previous operators had been merged perfectly. In practice, earlier merging steps may shift the activations seen by later operators. By evaluating $\widehat{\boldsymbol{Z}}_{t,i}^{(m)}$, SLOA aligns each operator on the activation distribution that it will actually receive inside the partially merged model. The target remains $\mathcal{A}(\boldsymbol{Z}_{t,i}^{(m)};\boldsymbol{\theta}_t^{(m)})$, therefore, the current operator corrects for accumulated upstream error while preserving the task-specific local behavior.

3.5 Rank as capacity control

Our method allows the merged model to have a larger capacity than the task-specific models. For example, let $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d\times d_k}$ and $\boldsymbol{W}_O \in \mathbb{R}^{d_k\times d}$ be the parameters for an attention head from the task-specific models, then the local operators for this head are given by

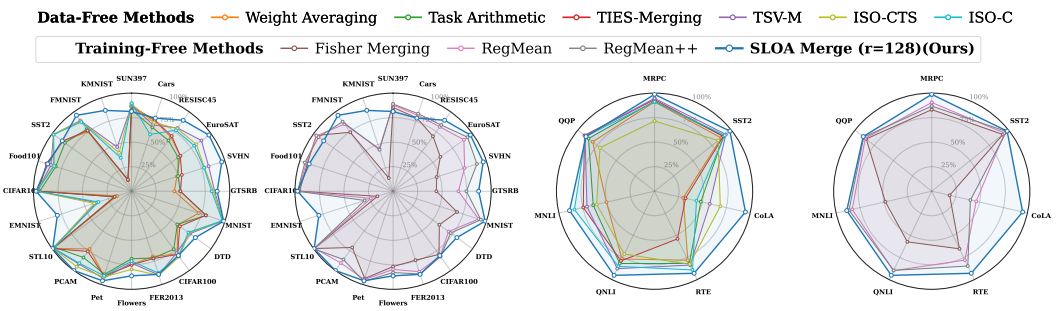
$$\boldsymbol{\theta}_{QK} = \boldsymbol{W}_Q\boldsymbol{W}_K^\top/\sqrt{d_k} \quad \text{and} \quad \boldsymbol{\theta}_{VO} = \boldsymbol{W}_V\boldsymbol{W}_O, \quad \text{s.t.} \quad \boldsymbol{\theta}_{QK}, \boldsymbol{\theta}_{VO} \in \mathbb{R}^{d\times d}. \tag{6}$$

Since these operators are realized through a width-d_k bottleneck, they have rank at most d_k. The merged operator, however, needs to combine T such operators that may occupy different subspaces. Consequently, it is likely that using the same model width for the merged model, i.e., a rank of at most d_k, would result in underfitting. Therefore, it is important to let the merged model be more expressive, which in this case means a rank exceeding d_k. To this end, note that SLOA solves for $\boldsymbol{\theta}_{\text{merge}} \in \mathbb{R}^{d\times d}$ without any constraints, while the factorization rank r for the merged parameters is chosen only afterward. Let $\boldsymbol{\theta}^r_{\text{merge}} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^\top$ be the best rank-r approximation of $\boldsymbol{\theta}_{\text{merge}}$, and

$$\begin{aligned} &\text{QK:} \quad \bar{\boldsymbol{W}}_Q = \boldsymbol{U}_r(\tau\boldsymbol{\Sigma}_r)^{1/2}, \quad \bar{\boldsymbol{W}}_K = \boldsymbol{V}_r(\tau\boldsymbol{\Sigma}_r)^{1/2}, \\ &\text{VO:} \quad \bar{\boldsymbol{W}}_V = \boldsymbol{U}_r\boldsymbol{\Sigma}_r^{1/2}, \quad \bar{\boldsymbol{W}}_O = \boldsymbol{\Sigma}_r^{1/2}\boldsymbol{V}_r^\top, \end{aligned} \quad \text{s.t.} \quad \begin{aligned} &\bar{\boldsymbol{W}}_Q, \bar{\boldsymbol{W}}_K \in \mathbb{R}^{d\times r}, \\ &\bar{\boldsymbol{W}}_V \in \mathbb{R}^{d\times r},\ \bar{\boldsymbol{W}}_O \in \mathbb{R}^{r\times d}, \end{aligned} \tag{7}$$

where $\tau = \sqrt{r}$ is the attention scaling. With this choice of parameters for the merged model, we have $\bar{\boldsymbol{W}}_Q\bar{\boldsymbol{W}}_K^\top/\tau = \boldsymbol{\theta}^r_{\text{merge},QK}$ and $\bar{\boldsymbol{W}}_V\bar{\boldsymbol{W}}_O = \boldsymbol{\theta}^r_{\text{merge},VO}$, where the rank of the merged head operators is r. Setting $r = d_k$ preserves the architecture, whereas $r > d_k$ increases the rank of each head. Naturally, adding more parameters to the model increases the computational cost at inference. Empirically, we find that moderately increasing the rank improves performance while incurring only a small computational overhead. Further discussion about rank expansion in §D.5.

Additional remarks. We conclude this section with two remarks: *(i)* RegMean (Jin et al., 2023) regularizes each linear-layer weight through Gram-matrix shrinkage. However, in attention, similar projection matrices do not necessarily imply similar computations. For example, changes to the query and key matrices can cancel each other and leave the QK operator unchanged, even though a penalty on the two matrices would still be nonzero. We build $\boldsymbol{\Lambda}_t$ from calibration states of the partially merged model. The resulting metric, $\boldsymbol{\Lambda}_t \otimes \boldsymbol{\Lambda}_t$, weights differences in the QK operator along both inputs to the attention-score map. Unlike separate penalties on $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$, it constrains the computation they define together while keeping the objective quadratic in the operator variable. Thus, SLOA keeps the local operators close to their task-specific counterparts without forcing their internal attention factors to use the same coordinates. *(ii)* ProDistill (Xu et al., 2025) progressively



(a) ViT-B/32 vs. data-free (b) ViT-B/32 vs. train-free (c) RoBERTa vs. data-free (d) RoBERTa vs. train-free

Figure 3: Per-task comparison of SLOA (r=128) with data-free and training-free baselines on ViT-B/32 with 20 vision tasks, Panels (a), (b) and RoBERTa with 7 GLUE tasks, Panels (c), (d). SLOA consistently outperforms both data-free and training-free baselines.

matches block outputs on paired partially merged and task-model activations by optimizing element-wise task-vector coefficients. SLOA shares this progressive matching principle, but changes both the merging variables and the level at which behavior is matched. Specifically, it aligns composed QK and VO operators through pre-softmax score matching and attention-weighted output matching, respectively. For fixed local inputs, these behavior maps are linear, yielding quadratic alignment objectives that can be solved with structured linear algebra. Merely replacing projection factors with composed operators in a block-level distillation objective would not yield this property.

Importantly, the main novelty of SLOA over existing methods is that SLOA treats merging as a sequence of operator-level regression problems. At each step, it matches local behavior, regularizes the operators using the data, and reconstructs them at a chosen rank. As it works directly with the composed QK and VO operators, it does not depend on how these operators are factorized. It also lets us choose the rank used by the deployed model, giving the merged model greater capacity.

4 Experiments

In this section, we demonstrate SLOA for model merging across vision, natural language understanding, and large language model settings, using both fully fine-tuned and LoRA-adapted experts. In Sec. 4.1, we evaluate Vision Transformers (Dosovitskiy et al., 2020) on image classification benchmarks comprising 8, 14, and 20 tasks, focusing on the ViT-B/32 architecture. In Sec.4.2, we consider merging RoBERTa models (Liu et al., 2019) for natural language understanding (NLU). Finally, in Sec. 4.3, we evaluate SLOA at a larger model scale by merging LoRA-adapted LLaMA-2-7B experts (Touvron et al., 2023) trained for math, code, and instruction-following tasks. Figure 3 previews the per-task gains of SLOA over all baselines. Unless otherwise stated, all training-free methods that require calibration, including SLOA, use N=256 samples per task drawn exclusively from the corresponding training split; no test data is used for merging. All results are averaged over three random seeds. Additional experiments in §F, and full experimental details are provided in §E.

4.1 Merging Vision Transformers (ViT-B)

We first evaluate SLOA on multi-task merging of ViT-B/32 experts using the task-specific checkpoints provided by Tang et al. (2024). All experts are fine-tuned from a shared CLIP initialization (Radford et al., 2021b), each on a distinct image classification task. We consider three increasingly challenging settings involving 8, 14, and 20 experts. Our data-free baselines are weight averaging (Wortsman et al., 2022), Task Arithmetic (Ilharco et al., 2023), TIES-Merging (Yadav et al., 2023), TSV-M (Gargiulo et al., 2025), and Iso-C and Iso-CTS (Marczak et al., 2025). We also compare against training-free methods that use calibration data: Fisher Merging (Matena & Raffel, 2022), RegMean (Jin et al., 2023), and RegMean++ (Nguyen et al., 2026).

Table 2 summarizes the results. At the original rank, SLOA outperforms all baselines across the three settings. Its advantage over the strongest baseline, RegMean++, increases with the number of merged tasks, from 0.6 percentage points for 8 tasks to 1.9 for 14 tasks and 5.5 for 20 tasks.

These results demonstrate that SLOA handles an increasing number of tasks more effectively than the competing methods on these benchmarks.

Our component ablations suggest that sequential alignment accounts for most of the improvement by compensating for upstream merging errors that subsequent operators would otherwise inherit (See §F.4 and §F.7). Merging the composed QK and VO operators yields a further, consistent gain over sequential alignment alone. The operator formulation also enables rank expansion. Increasing the retained rank from r=64, the original head dimension, to r=128 improves accuracy across all three settings, with larger gains when merging more tasks. This suggests that model capacity increasingly constrains merging performance as the number of tasks grows. We use a fixed calibration budget of N=256 and report retained ranks of r=64 and r=128 to provide a controlled comparison and assess the benefit of rank expansion. These choices do not represent limits of SLOA: larger calibration sets and retained ranks yield further improvements, at the cost of additional merging time and, for rank expansion, additional inference time (Secs.§F.3 and §G). We observe similar trends for ViT-B/16 (§F.1) and LoRA fine-tuned models (§F.2).

Table 2: Performance comparison of model merging methods on ViT-B/32 across 8, 14, and 20 vision classification tasks. For SLOA, r denotes the retained factorization rank used for both QK and VO local operators. The setting $r = 64$ matches the original ViT-B/32 attention-head dimension.

	Method	# Samples	Avg. Accuracy ↑		
			8 Tasks	14 Tasks	20 Tasks
	Fine-tuned	–	90.39	89.26	89.78
Data-Free	Weight Averaging	–	66.29	65.37	61.08
Data-Free	Task Arithmetic	–	69.49	66.47	60.60
Data-Free	TIES-Merging	–	71.70	67.58	62.76
Data-Free	TSV-M	–	83.04	79.11	73.66
Data-Free	Iso-C	–	82.64	77.94	71.16
Data-Free	Iso-CTS	–	82.87	79.24	73.41
Training-Free	Fisher Merging	256	69.12	66.26	61.74
Training-Free	RegMean	256	82.63	77.93	72.20
Training-Free	RegMean++	256	84.37	80.06	74.51
Training-Free	SLOA (r=64)	256	<u>84.89</u>	<u>81.94</u>	<u>80.06</u>
Training-Free	SLOA (r=128)	256	**85.50**	**82.61**	**81.03**

4.2 Merging encoder-based language models

To assess whether operator alignment extends beyond vision, we merge seven RoBERTa-base models fine-tuned on distinct GLUE tasks (Wang et al., 2019), using the checkpoints provided by Yu et al. (2024). These tasks cover linguistic acceptability (CoLA), sentiment classification (SST-2), paraphrase and question-pair matching (MRPC, QQP), and textual entailment (MNLI, QNLI, RTE), requiring the merged model to preserve task-specific computations across diverse input structures and decision boundaries. We compare SLOA against the same baselines used in the ViT experiment.

Table 3: Avg. GLUE metric for merged RoBERTa base models. Full per-task results are in Table 16.

Method	Avg. Metric ↑
Individual (oracle)	0.8483
Data-Free	
Weight Averaging	0.6165
Task Arithmetic	0.6724
TIES-Merging	0.6448
TSV-M	0.7120
Iso-C	0.7115
Iso-CTS	0.6174
Training-Free	
Fisher Merging	0.5680
RegMean	0.7066
RegMean++	0.7139
SLOA (r=64)	<u>0.7906</u>
SLOA (r=128)	**0.7979**

As shown in Table 3, SLOA at the original rank (r=64) improves the average score over the strongest baseline by 7.7 points. Increasing the QK/VO rank to r=128 improves scores on every task and raises this margin to 8.4 points. The gains extend across the benchmark: SLOA achieves the highest scores on six of the seven tasks (CoLA, SST-2, MRPC, MNLI, QNLI, and RTE), trailing only TIES-Merging and TSV-M on QQP (per-task results in Table 16). These findings are consistent with the motivation for aligning composed attention operators. The coupled QK and VO operators govern token interactions and value transformations within each attention head; aligning them can help preserve the distinct attention patterns learned by experts fine-tuned on different language tasks.

4.3 Merging LLaMA-2-7B LoRA experts

SLOA applies directly to LoRA-adapted experts without modifying its local objective. For each task t, we first form the effective fine-tuned weight $\boldsymbol{W}_t = \boldsymbol{W}_0 + \boldsymbol{B}_t\boldsymbol{A}_t$ and then construct the same

Table 4: LLaMA-2-7B LoRA-expert merge results (math, code, instruction-following). Avg. is the mean score over GSM8K (math), HumanEval (code, pass@1), and IFEval (instruction-following). Bold/underline marks the best/second-best merging method per column.

	Method	Benchmark			**Average**
		GSM8K	HumanEval	IFEval	**Avg.** ↑
Experts	Llama2-7B Pretrained	4.32	0.00	20.89	8.40
	Llama2-7B IF-Tulu	12.59	1.22	66.17	26.66
	Llama2-7B Code (Magic-Coder)	9.70	32.93	26.99	23.21
	Llama2-7B Math (Meta-Math)	65.50	0.00	15.53	27.01
Data-Free	Weight Averaging	40.11	18.29	40.67	33.02
	Task Arithmetic ($\alpha = 0.5$)	58.98	15.85	42.70	39.18
	TSV-M ($\alpha = 0.5$)	58.00	18.29	42.14	39.48
	TIES ($\alpha = 0.3$)	48.14	20.73	34.57	34.48
	KnOTS-TIES ($\lambda = 0.85$)	55.27	15.24	40.85	37.12
	Iso-C	38.89	21.34	57.67	39.30
	Iso-CTS	36.09	21.34	55.45	37.63
Training-Free	RegMean	56.03	**29.27**	48.24	44.51
	RegMean++	55.95	<u>29.27</u>	54.16	46.46
	SLOA-VO r_{ov}=128 (Ours)	58.61	28.05	57.86	48.17
	SLOA-VO r_{ov}=256 (Ours)	<u>59.51</u>	27.44	<u>58.96</u>	<u>48.64</u>
	SLOA-VO r_{ov}=384 (Ours)	**59.51**	27.44	**60.63**	**49.19**

functional operators used for fully fine-tuned experts. This approach also avoids the factorization ambiguity of LoRA: since $\boldsymbol{B}_t\boldsymbol{A}_t = (\boldsymbol{B}_t\boldsymbol{R})(\boldsymbol{R}^{-1}\boldsymbol{A}_t)$ for any invertible $\boldsymbol{R}$, merging $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ separately can make the result depend on an arbitrary choice of low-rank coordinates. By operating on the effective weights and their composed operators, SLOA is invariant to this ambiguity.

We evaluate this setting by merging three LoRA-adapted LLaMA-2-7B experts specialized in math, code, and instruction-following (details in Sec. E.2.3). LLaMA-2 is a decoder-only architecture with rotary positional embeddings, whose position-dependent rotations complicate the representation of the QK computation as a single composed operator. We therefore compose only the VO circuit (SLOA-VO), while merging $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$ individually as linear operators. We evaluate both the original VO rank, r_{ov}=128, and expanded ranks, $r_{ov} \in \{256, 384\}$. Our comparisons include the baselines from Sec. 4.1 and the LoRA-specific baseline KnOTS-TIES (Stoica et al., 2025; Yadav et al., 2023). We evaluate math, code, and instruction-following performance on GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), and IFEval (Zhou et al., 2023), respectively.

At the architecture-preserving rank r_{ov}=128, SLOA-VO achieves the highest average score among all merging methods, exceeding RegMean++ by 1.7 points. Increasing r_{ov} to 384 further improves GSM8K and IFEval, with a small reduction in HumanEval performance (27.44 versus 28.05 at r_{ov}=128), indicating that the benefits of additional VO capacity vary across tasks. These results extend sequential operator alignment to LoRA experts on a 7B-parameter decoder-only model. Complementary experiments on LoRA-adapted ViT-B/32 experts, using both composed QK and VO operators, are presented in Sec. F.2.

5 Related Work

Data-free parameter-space merging. A broad class of data-free model merging methods considers task-specific models directly in parameter space. Uniform model soups average weights across models with the same architecture (Wortsman et al., 2022), while Task Arithmetic merges task vectors, defined as differences between fine-tuned and pretrained weights (Ilharco et al., 2023). Later methods reduce task interference: TIES-Merging removes small-magnitude updates and resolves sign conflicts (Yadav et al., 2023), DARE randomly drops and rescales task-vector entries (Yu et al., 2024), and spectral methods such as TSV-M and Iso-C/Iso-CTS manipulate the singular structure of task vectors to denoise or rescale merged updates (Gargiulo et al., 2025; Marczak et al., 2025). While attractive since they require no data or optimization, these methods merge parameter tensors instead of the computation units they implement. This is limiting in transformers, where attention depends on composed QK and OV operators and not isolated query, key, value, and output layers. SLOA instead merges these functional operators, removing dependence on arbitrary parameterizations.

Calibration-based training-free merging. Another line of work uses a small calibration set while still avoiding gradient-based training. Fisher Merging (Matena & Raffel, 2022) weights parameters according to their Fisher information, preserving parameters estimated to be important for the task losses. RegMean (Jin et al., 2023) solves a closed-form regression problem that aligns hidden representations of the merged model with those of the task-specific models. RegMean++ (Nguyen et al., 2026) improves this idea by computing statistics across layers, reducing the covariate shift caused by earlier merging errors. SLOA belongs to this training-free regime, but differs in the object being aligned. Prior methods mainly treat layers as independent units. In contrast, SLOA defines a local operator for each *functional* component: LayerNorm affine maps, linear maps, QK operators, and OV operators. This allows the same closed-form alignment principle to apply to both standard linear layers and coupled attention computations. Moreover, SLOA performs alignment *sequentially* under the activations produced by the partially merged model, so later operators are merged on the activations they will actually receive at inference time.

Test-time adaptation and training for merging. A separate class of methods improves merging by introducing learned coefficients, routers, task-specific components, or distillation objectives. AdaMerging (Yang et al., 2024b) learns layer-wise merging coefficients using unlabeled test samples. Twin-Merging (Lu et al., 2024) decomposes task vectors into shared and private components and uses a router to select fusion weights. EMR-Merging (Huang et al., 2024) maintains both shared and task-specific parameters, requiring task identity at inference time. Representation Surgery (Yang et al., 2024a) trains an auxiliary module to align representations, while ProDistill (Xu et al., 2025) formulates merging as a distillation problem with learned merging coefficients. These methods can achieve strong performance, but they require additional training, test-time data, task identity, routing, or teacher-student optimization. SLOA avoids these requirements: after collecting calibration activations, each local merge is solved in closed form, produces a single deployed model, and requires neither task identity nor test-time adaptation at inference time.

6 Conclusion

We presented *Sequential Local Operator Alignment* (SLOA), a training-free framework for merging transformers by aligning *local functional operators*. By treating the QK and VO products as the effective operators, matching each operator's behavior under the partially merged model's activations with a data-aware operator geometry, and merging sequentially in topological order, SLOA preserves task-specific behavior while reducing the growth of activation error across depth. The operator-level view also decouples the merged attention operator from its final factorization, making head rank an explicit capacity-control parameter. We evaluate SLOA on ViT-B/32 and ViT-B/16 with up to 20 tasks, RoBERTa on GLUE, LoRA-adapted ViT experts, and a VO-only variant for LoRA-adapted LLaMA-2-7B. Across these settings, SLOA outperforms strong merging baselines at the original head rank, showing that its gains do not require widening the deployed model. Its relative advantage becomes more pronounced as the number of merged tasks grows. Increasing the head rank can provide further gains beyond those obtained at the original rank.

One limitation of SLOA is that, like other calibration-based training-free methods, it requires a small unlabeled calibration set that represents each task's input distribution. When such data are unavailable, data-free methods may be more suitable. In addition, SLOA follows a greedy sequential procedure. At each step, it optimizes a well-defined local objective, but it does not optimize a single end-to-end objective. Finally, SLOA requires more computation during merging than data-free methods and simpler calibration-based approaches because it propagates activations, solves local regression problems, and factorizes the merged operators. Increasing the rank beyond d_k also widens the attention heads, which increases inference time and memory use. Promising directions include reducing reliance on in-distribution calibration data through synthetic, generated, or proxy inputs, and combining SLOA with data free merging when calibration data provide insufficient coverage of some tasks. Extending the composed QK formulation to attention with relative or rotary position encodings would also broaden its architectural scope. Another direction is automatic per-head rank selection, allowing the merged model's capacity to be allocated according to deployment constraints. More broadly, the proposed method supports a functional view of transformer merging: align local computations along the merged execution path, and choose the attention factorization to control how much of the aligned operators the deployed model retains for multi-task merging. In light of this view, the unit of merging does not need to be the unit of parameterization.

AI USE STATEMENT

We used generative AI tools for writing assistance, rewriting, grammatical correction, improving clarity, and structure formatting. We also used AI as a coding assistant, with all AI-suggested code reviewed and tested for correctness by the authors for figure plotting and formatting. We did not use generative AI for research ideation, for designing or providing feedback on our research methodology or experiments, for conducting our literature survey, for generating synthetic datasets or developing our theoretical framework, or for formulating mathematical claims, proof ingredients, or hypotheses – all such contributions are our own. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI, if any.

REPRODUCIBILITY STATEMENT

To support reproducibility, we rely on publicly released checkpoints wherever possible: the ViT-B/32 and ViT-B/16 task-specific checkpoints are those released by Tang et al. (2024), the RoBERTa checkpoints are those released by Yu et al. (2024), and the ViT-B/32 LoRA experts are those released by Stoica et al. (2025). Full descriptions of the datasets, evaluation metrics, and experimental protocol used across all settings are provided in §E. For the LLM experiments, all three LoRA experts were fine-tuned by us from LLaMA-2-7B; the corresponding training hyperparameters, datasets, and evaluation benchmarks are detailed in §E.2.3. All reported results are averaged over 3 random seeds. Complete derivations and proofs underlying SLOA's local-operator alignment objective are provided in Appendix D. Upon acceptance, we will publicly release our code and the fine-tuned LLM LoRA checkpoints to enable full replication of our experiments.

REFERENCES

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Joachim Bona-Pellissier, François Bachoc, and François Malgouyres. Parameter identifiability of a deep feedforward relu neural network. *Machine learning*, 112(11):4431–4493, 2023.

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.

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

Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16124–16170, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence*, 5(3):220–235, 2023.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017.

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

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Shmuel Friedland and Anatoli Torokhti. Generalized rank-constrained matrix approximations. *SIAM Journal on Matrix Analysis and Applications*, 29(2):656–659, 2007. doi: 10.1137/06065551.

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. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33111–33124, 2025.

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.

Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)*, pp. 4884–4896, 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. In *International Conference on Learning Representations*, 2022.

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. Emr-merging: Tuning-free high-performance model merging. *Advances in Neural Information Processing Systems*, 37:122741–122769, 2024.

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

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. *Advances in Neural Information Processing Systems*, 37:78905–78935, 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.

Behnam Neyshabur, Yuhuai Wu, Russ R Salakhutdinov, and Nati Srebro. Path-normalized optimization of recurrent neural networks with relu activations. *Advances in Neural Information Processing Systems*, 29, 2016.

The-Hai Nguyen, Dang Huu-Tien, Takeshi Suzuki, and Le-Minh Nguyen. Regmean++: Enhancing effectiveness and generalization of regression mean for model merging. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021a.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021b.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. *ICLR*, 2025.

Anke Tang, Li Shen, Yong Luo, Han Hu, Bo Du, and Dacheng Tao. Fusionbench: A comprehensive benchmark of deep model fusion. *arXiv preprint arXiv:2406.03280*, 2024.

Alexander Theus, Alessandro Cabodi, Sotiris Anagnostidis, Antonio Orvieto, Sidak Pal Singh, and Valentina Boeva. Generalized linear mode connectivity for transformers. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bash-lykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wen-Yin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, A. S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian-Xiang Kuan, Puxin Xu, Zheng-Xu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melissa Hall Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnić, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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 *International Conference on Learning Representations*, 2019.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with OSS-instruct. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 52632–52657. PMLR, 21–27 Jul 2024.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pp. 23965–23998. PMLR, 2022.

Jing Xu, Jiazheng Li, and Jingzhao Zhang. Scalable model merging with progressive layer-wise distillation. In *Forty-second International Conference on Machine Learning*, 2025.

Zhengqi Xu, Ke Yuan, Huiqiong Wang, Yong Wang, Mingli Song, and Jie Song. Training-free pretrained model merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5915–5925, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

Prateek Yadav, Tu Vu, Jonathan Lai, Alexandra Chronopoulou, Manaal Faruqui, Mohit Bansal, and Tsendsuren Munkhdalai. What matters for model merging at scale? *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. In *Forty-first International Conference on Machine Learning*, 2024a.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*, 2024b.

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. *ACM Computing Surveys*, 58(8):1–41, 2026.

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.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Binchi Zhang, Zaiyi Zheng, Zhengzhang Chen, and Jundong Li. Beyond the permutation symmetry of transformers: The role of rotation for model fusion. In *Forty-second International Conference on Machine Learning*, 2025.

Bo Zhao, Robin Walters, and Rose Yu. Symmetry in neural network parameter spaces. *arXiv preprint arXiv:2506.13018*, 2025.

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.

Appendix

A Additional background **16**

A.1 Model Merging . . . 16

A.2 Transformer architecture and the attention operation . . . 16

B Extended related work **17**

B.1 Alignment under model symmetries and activation matching . . . 17

B.2 Transformer circuits and operator structure . . . 17

C Technical comparison to RegMean, RegMean++, rotation-symmetry matching, and ProDistill **17**

C.1 Comparison with RegMean . . . 17

C.2 Comparison with RegMean++ . . . 19

C.3 Comparison with rotation-symmetry matching . . . 22

C.4 Comparison to ProDistill . . . 25

D Detailed derivation of local operator alignment **27**

D.1 Optimal QK Operator . . . 28

D.2 Optimal VO Operator . . . 29

D.3 Optimal MLP operator . . . 31

D.4 Optimal LayerNorm Operator . . . 31

D.5 Rank-controlled factorization of merged attention operators . . . 32

E Experimental details **33**

E.1 Handiling bias terms . . . 33

E.2 Dataset description . . . 33

F Additional experiments and ablation analysis of SLOA **35**

F.1 Scaling to a larger vision backbone: ViT-B/16 . . . 35

F.2 Merging LoRA-adapted ViT experts . . . 36

F.3 Scaling with calibration samples and rank . . . 36

F.4 Error accumulation across network depth . . . 37

F.5 Decoupling the QK and OV circuits . . . 38

F.6 Sensitivity to regularization strength . . . 38

F.7 Component ablation. . . . 38

F.8 Sequential correction versus training-free baselines. . . . 39

G Computational Cost and Runtime Analysis **39**

G.1 Merging cost versus baselines . . . 39

G.2 Scaling cost with task count and calibration budget . . . 40

G.3 Inference cost 41

H Task-wise full tables **41**

A ADDITIONAL BACKGROUND

A.1 MODEL MERGING

Let there be a set of T distinct tasks with corresponding training datasets $\mathcal{D} = \{D_1, \ldots, D_T\}$. We assume a homogeneous neural network architecture parameterized by $\mathbf{\Phi} \in \mathbb{R}^d$, where d denotes the total number of parameters. Let $\mathbf{\Phi}_{\text{pre}} \in \mathbb{R}^d$ denote the parameters of a large-scale pretrained foundation model (e.g., CLIP (Radford et al., 2021b)) that serves as a shared initialization. We are given T task-specific models $\{\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_T\}$, where each model $\mathbf{\Phi}_t$ is obtained by independently fine-tuning $\mathbf{\Phi}_{\text{pre}}$ exclusively on its respective dataset D_t.

Model Merging. Model merging is the process of aggregating these distinct parameter sets into a single, unified multi-task model without requiring *computationally expensive joint retraining on the original training datasets*. Mathematically, we seek an aggregation function g such that:

$$\mathbf{\Phi}_{\text{merge}} = g(\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_T). \tag{8}$$

Performance Objectives. The fundamental objective of model merging is to construct a single parameter set $\mathbf{\Phi}_{\text{merge}}$ that encapsulates the combined capabilities of all individual experts. Let $\mathcal{M}_t : \mathbb{R}^d \to \mathbb{R}$ denote a task-specific evaluation metric (e.g., accuracy) for task t, where higher values indicate better performance. A successful merging function ensures that the unified model recovers the performance of the specialized experts:

$$\mathcal{M}_t(\mathbf{\Phi}_{\text{merge}}) \approx \mathcal{M}_t(\mathbf{\Phi}_t), \quad \forall t \in \{1, \ldots, T\}. \tag{9}$$

A.2 TRANSFORMER ARCHITECTURE AND THE ATTENTION OPERATION

The Transformer architecture (Vaswani et al., 2017) has become the foundational backbone for state-of-the-art models across both vision and language domains. Structurally, a Transformer is composed of multiple identical, sequentially stacked layers For a given layer, the input representations are processed through two primary sub-modules: a (i) **Multi-Head Attention (MHA)** block and a (ii) **Multi-Layer Perceptron (MLP)**, accompanied by Layer Normalization and residual connections.

Multi-Head Attention (MHA). The core of the Transformer is the attention mechanism, which routes information across sequence tokens. Let $\boldsymbol{X} \in \mathbb{R}^{L \times d}$ denote the input sequence of L sequence length with model dimension d. The matrices for the layer are defined as $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}_O \in \mathbb{R}^{d \times d}$.

To allow the model to jointly attend to information from different representations subspaces, the MHA operation divides the global matrices into H independent heads. The global parameter matrices $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}_O \in \mathbb{R}^{d \times d}$ are partitioned across H heads, where each head j operates on a reduced subspace of dimension $d_k = d/H$, with head-specific sub-matrices $\boldsymbol{W}_Q^{(j)}, \boldsymbol{W}_K^{(j)}, \boldsymbol{W}_V^{(j)} \in \mathbb{R}^{d \times d_k}$ and $\boldsymbol{W}_O^{(j)} \in \mathbb{R}^{d_k \times d}$ for the j-th head. Crucially, instead of viewing MHA merely as a concatenation operation, it can be mathematically formulated as the sum of independent head contributions added directly to the residual stream. The output $\boldsymbol{Z}_{\text{MHA}} \in \mathbb{R}^{n \times d}$ of the MHA block is given by:

$$\boldsymbol{Z}_{\text{MHA}} = \sum_{j=1}^{H} \text{softmax}\left(\frac{(\boldsymbol{X}\boldsymbol{W}_Q^{(j)})(\boldsymbol{X}\boldsymbol{W}_K^{(j)})^\top}{\sqrt{d_k}} \right) \boldsymbol{X}\boldsymbol{W}_V^{(j)}\boldsymbol{W}_O^{(j)} \tag{10}$$

This formulation highlights two distinct functional circuits within each head j: the bilinear Query-Key (QK) circuit defined by $\boldsymbol{W}_Q^{(j)}{\boldsymbol{W}_K^{(j)}}^\top$, which determines attention scores, and the linear Value-Output (VO) circuit defined by $\boldsymbol{W}_V^{(j)}\boldsymbol{W}_O^{(j)}$, which dictates the information written to the output (Elhage et al., 2021).

Multi-Layer Perceptron (MLP). Following the attention mechanism, the representations pass through a two-layer MLP block. Let $\boldsymbol{W}_1 \in \mathbb{R}^{d \times d_{out}}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{d_{out} \times d}$ be the first and the second MLP layer matrices, respectively. For an input representation to the MLP block, which we denote as $\boldsymbol{Z} \in \mathbb{R}^{L \times d}$ (functionally the layer-normalized sum of $\boldsymbol{X}$ and $\boldsymbol{Z}_{\text{MHA}}$), the block applies a non-linear activation function $\sigma(\cdot)$ (e.g., ReLU or GELU). The transformation sequence maps the inputs to the final block output $\boldsymbol{Z}_{\text{MLP}} \in \mathbb{R}^{L \times d}$: Algebraically, the operation of the entire MLP block is

summarized as:

$$\boldsymbol{Z}_{\mathrm{MLP}} = \sigma(\boldsymbol{Z}\boldsymbol{W}_1)\boldsymbol{W}_2 \tag{11}$$

B Extended related work

B.1 Alignment under model symmetries and activation matching

Neural networks contain many parameter symmetries: different parameterizations can represent the same function (Dinh et al., 2017; Neyshabur et al., 2016; Zhao et al., 2025; Bona-Pellissier et al., 2023). Prior work on model alignment and mode connectivity studies how to account for such symmetries before merging, for example by matching neurons, features, or permutations across independently trained networks (Ainsworth et al., 2023; Singh & Jaggi, 2020; Xu et al., 2024; Zhang et al., 2025). These methods highlight an important principle: successful merging should compare models in a functionally meaningful coordinate system, rather than relying only on raw parameter coordinates.

Our work is motivated by the same non-identifiability issue, but addresses a different symmetry that appears inside transformer attention. For a single attention head, the transformation $(\boldsymbol{W}_Q, \boldsymbol{W}_K) \mapsto (\boldsymbol{W}_Q\boldsymbol{R}, \boldsymbol{W}_K\boldsymbol{R}^{-\top})$ leaves the QK operator $\boldsymbol{W}_Q\boldsymbol{W}_K^{\top}$ unchanged, and analogous changes of the value-head coordinates preserve the VO operator. Thus, aligning or averaging $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, $\boldsymbol{W}_V$, and $\boldsymbol{W}_O$ separately can break a computation that is shared at the operator level. Instead of first searching for a coordinate transformation between factorizations, SLOA removes this ambiguity by merging the invariant local operators themselves and factorizing the merged operators only after alignment.

B.2 Transformer circuits and operator structure

Several analyses of transformers emphasize that attention heads are naturally described through composed circuits rather than isolated projection matrices (Elhage et al., 2021; Dar et al., 2023; Vaswani et al., 2017; Theus et al., 2026). In this view, the QK circuit controls the pre-softmax attention scores, while the VO, or OV, circuit controls how attended token representations are written back to the residual stream. This perspective has been influential for understanding transformer computation, but it has been less directly exploited in training-free model merging. SLOA turns this circuit-level view into a concrete merging algorithm. For each attention head, it constructs and aligns the QK and VO operators as residual-space maps. This also separates the operator being merged from the rank used to implement it. Each task-specific head realizes a low-rank operator because the head dimension is limited, but the merged multi-task operator can have higher effective rank when different tasks use different subspaces. SLOA therefore treats rank as a capacity-control mechanism: after solving for the merged ambient operator, it can either factorize it at the original head rank or retain additional singular directions through rank expansion. This differs from spectral task-vector methods, which apply SVD to task updates in weight space; in SLOA, the SVD is applied to the merged functional attention operator and determines how much of that operator is represented in the final head.

C Technical comparison to RegMean, RegMean++, rotation-symmetry matching, and ProDistill

C.1 Comparison with RegMean

We compare SLOA with Regression Mean (RegMean) (Jin et al., 2023). RegMean is one of the closest predecessors to our method because it also derives a closed-form merging rule from a regression objective. However, RegMean and SLOA differ in the object being merged, the local objective being solved, the treatment of attention, and the role of calibration states.

RegMean as layer-wise linear regression. Consider a linear layer whose local behavior in task model t is

$$\boldsymbol{X}_t\boldsymbol{W}_t, \tag{12}$$

where $\boldsymbol{X}_t$ denotes the input activations to that linear layer and $\boldsymbol{W}_t$ is the corresponding task-specific weight matrix. RegMean seeks a merged weight $\boldsymbol{W}$ whose outputs match the outputs of the individual task layers on their own layer inputs:

$$\min_{\boldsymbol{W}} \sum_{t=1}^{T} \|\boldsymbol{X}_t \boldsymbol{W} - \boldsymbol{X}_t \boldsymbol{W}_t\|_F^2 . \tag{13}$$

Let

$$\boldsymbol{G}_t = \boldsymbol{X}_t^\top \boldsymbol{X}_t \tag{14}$$

be the input Gram matrix for task model t. The closed-form RegMean solution is

$$\boldsymbol{W}_{\text{RegMean}} = \left(\sum_{t=1}^{T} \boldsymbol{G}_t \right)^{-1} \left(\sum_{t=1}^{T} \boldsymbol{G}_t \boldsymbol{W}_t \right), \tag{15}$$

up to the normalization convention used for the empirical average. In transformer models, RegMean applies this rule independently to each linear layer. Other parameters, such as embeddings, biases, and normalization parameters, are typically merged by simple averaging.

Connection to SLOA for ordinary linear maps. SLOA reduces to a RegMean-like objective in a restricted special case. For an ordinary one-sided linear map, SLOA uses the local behavior map

$$\mathcal{A}(\boldsymbol{X}; \boldsymbol{W}) = \boldsymbol{X}\boldsymbol{W}. \tag{16}$$

If we ignore the sequential activation mismatch by setting

$$\widehat{\boldsymbol{X}}_{t,i} = \boldsymbol{X}_{t,i} \tag{17}$$

for all calibration examples, and if we remove the additional operator-geometry regularizer, then the SLOA activation-matching term becomes

$$\min_{\boldsymbol{W}} \sum_{t=1}^{T} \sum_{i=1}^{n_t} \|\boldsymbol{X}_{t,i} \boldsymbol{W} - \boldsymbol{X}_{t,i} \boldsymbol{W}_t\|_F^2 , \tag{18}$$

which is exactly the RegMean regression problem for linear layers. Thus, RegMean can be viewed as a special case of local behavior matching for isolated one-sided linear operators under task-model activations.

The purpose of SLOA is to go beyond this special case. SLOA changes both the local inputs and the operator space:

$$\mathcal{A}\left(\widehat{\boldsymbol{Z}}_{t,i}; \boldsymbol{\theta}\right) \approx \mathcal{A}\left(\boldsymbol{Z}_{t,i}; \boldsymbol{\theta}_t\right), \tag{19}$$

where $\widehat{\boldsymbol{Z}}_{t,i}$ is produced by the partially merged model and $\boldsymbol{Z}_{t,i}$ is produced by the original task model. For linear maps, this gives an objective of the form

$$\min_{\boldsymbol{W}} \sum_{t=1}^{T} \left\| \widehat{\boldsymbol{X}}_t \boldsymbol{W} - \boldsymbol{X}_t \boldsymbol{W}_t \right\|_F^2 + \sum_{t=1}^{T} \|\boldsymbol{W} - \boldsymbol{W}_t\|_{\widehat{\chi}_t}^2 . \tag{20}$$

The corresponding normal equations involve both the partially merged Gram matrix

$$\widehat{\boldsymbol{X}}_t^\top \widehat{\boldsymbol{X}}_t \tag{21}$$

and the cross-state term

$$\widehat{\boldsymbol{X}}_t^\top \boldsymbol{X}_t \boldsymbol{W}_t. \tag{22}$$

This is different from RegMean, whose objective uses the same task-model input $\boldsymbol{X}_t$ on both sides of the regression.

Independent linear-layer merging versus local functional operators. The main limitation of RegMean from our perspective is that it treats each linear layer as the primitive unit of merging. This is appropriate for genuinely one-sided linear maps, but it is not aligned with the computation inside transformer attention. In a transformer head, the query and key weights do not determine attention scores independently. Rather, the pre-softmax attention scores depend on the composed operator

$$\boldsymbol{\theta}_{QK,t}^{(j)} = \frac{\boldsymbol{W}_{Q,t}^{(j)} \left(\boldsymbol{W}_{K,t}^{(j)}\right)^\top}{\sqrt{d_k}}. \tag{23}$$

Similarly, the value and output weights determine the residual write map through

$$\boldsymbol{\theta}_{VO,t}^{(j)} = \boldsymbol{W}_{V,t}^{(j)} \boldsymbol{W}_{O,t}^{(j)}. \tag{24}$$

RegMean applies regression to the individual projection matrices, treating $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, $\boldsymbol{W}_V$, and $\boldsymbol{W}_O$ as separate linear maps. SLOA instead treats $\boldsymbol{\theta}_{QK}$ and $\boldsymbol{\theta}_{VO}$ as the local functional operators to be merged.

This distinction matters because the individual attention factors are not identifiable from the attention computation. For any invertible matrix $\boldsymbol{R}$,

$$\boldsymbol{W}_Q \boldsymbol{W}_K^\top = (\boldsymbol{W}_Q \boldsymbol{R})(\boldsymbol{W}_K \boldsymbol{R}^{-\top})^\top. \tag{25}$$

Thus, two models can have the same QK score operator but different query/key coordinates. A method that merges $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$ independently is sensitive to this arbitrary factorization. SLOA removes this ambiguity by merging the composed QK operator directly. The same argument applies to the VO circuit.

One-sided geometry versus two-sided attention geometry. RegMean uses input Gram matrices for one-sided linear maps. For a linear map $\boldsymbol{XW}$, this is natural because the local discrepancy is measured after left multiplication by $\boldsymbol{X}$. However, the QK behavior map is not one-sided:

$$\mathcal{A}(\boldsymbol{X}; \boldsymbol{\theta}_{QK}) = \boldsymbol{X} \boldsymbol{\theta}_{QK} \boldsymbol{X}^\top. \tag{26}$$

SLOA therefore uses a two-sided metric for QK operators:

$$\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|_{\widehat{\chi}_t}^2 = \left\| \boldsymbol{\Lambda}_t^{1/2} (\boldsymbol{\theta} - \boldsymbol{\theta}_t) \boldsymbol{\Lambda}_t^{1/2} \right\|_F^2, \tag{27}$$

which corresponds after vectorization to a Mahalanobis metric with covariance structure

$$\boldsymbol{\Lambda}_t \otimes \boldsymbol{\Lambda}_t. \tag{28}$$

This two-sided geometry has no analogue in RegMean because RegMean does not formulate QK as a bilinear residual-space operator.

Dataless statistics versus sequential calibration states. RegMean was designed for a dataless knowledge-fusion setting: the model provider can release the fine-tuned weights together with low-dimensional inner-product statistics, without sharing the original training data. This is an important privacy and deployment advantage. SLOA instead targets the training-free calibration-data setting. It uses a small calibration set to run both the task models and the partially merged model, collecting the states required for local behavior matching and data-dependent operator regularization. Since SLOA evaluates each local operator on the states produced by the partially merged model, it can correct for upstream merging errors. RegMean computes each layer's merge from the task-model layer inputs and therefore implicitly assumes that previous merged layers will reproduce the task-model activations accurately. SLOA does not make this assumption: the merged operator is evaluated on the activation distribution it will actually receive during the merged model's forward pass.

Rank and capacity. RegMean preserves the original architecture because it directly solves for each original weight matrix. For attention, this means the head dimension remains fixed and the rank of each QK or VO product remains constrained by the original factorization. SLOA decouples the merged operator from its final factorization. It first solves for the merged QK or VO operator in the ambient residual-space operator space

$$\Theta_{\text{amb}} = \mathbb{R}^{d \times d}, \tag{29}$$

and only afterward factorizes the operator at a chosen rank r. This allows either rank-preserving merging with $r = d_k$ or rank-expanded merging with $r > d_k$. RegMean does not expose this rank-control mechanism because it never explicitly constructs the composed attention operator.

C.2 Comparison with RegMean++

We next compare SLOA with RegMean++ (Nguyen et al., 2026). RegMean++ identifies an important limitation of RegMean: independently merging each linear layer using task-model activations ignores the way representation errors propagate through the merged network. RegMean++ addresses this by recomputing statistics progressively using activations produced by previously merged layers.

SLOA shares this sequential motivation, but differs in the local objective, the treatment of attention, the operator geometry, and the final factorization step.

RegMean++ as progressive RegMean. RegMean++ keeps the core RegMean closed-form linear-layer solve. The key modification is how the layer statistics are collected. In RegMean, the input statistics for a layer are obtained from the corresponding task model. In RegMean++, the current layer is processed after earlier layers have already been merged, so the statistics for later layers are computed using activations induced by the partially merged model. This makes the regression statistics more consistent with the actual feature distribution that the merged model will experience.

Schematically, let $\boldsymbol{H}_t^{(\ell-1)}$ denote the activation entering transformer layer ℓ for task t. RegMean uses the task-model state

$$\boldsymbol{H}_t^{(\ell-1)} = f_t^{(<\ell)}(\boldsymbol{X}_t), \tag{30}$$

whereas RegMean++ uses a merged-model state of the form

$$\widehat{\boldsymbol{H}}_t^{(\ell-1)} = f_{\text{merge}}^{(<\ell)}(\boldsymbol{X}_t). \tag{31}$$

The resulting statistics are then used in the same RegMean-style closed-form solve for the linear layers inside the current transformer layer. Thus, RegMean++ is progressive in the sense that earlier merged layers influence the feature statistics used to merge later layers.

Shared motivation with SLOA. SLOA agrees with the core diagnosis of RegMean++: a layer should not be merged under the assumption that all preceding layers have been merged perfectly. If earlier merges shift the hidden states, then later layers should be aligned under the shifted hidden states that the final merged model will actually produce. This is exactly why SLOA introduces the partially merged local input $\widehat{\boldsymbol{Z}}_{t,i}^{(m)}$ for operator o_m.

However, SLOA uses this sequential state in a different regression problem. At operator o_m, SLOA solves

$$\mathcal{A}\left(\widehat{\boldsymbol{Z}}_{t,i}^{(m)}; \boldsymbol{\theta}\right) \approx \mathcal{A}\left(\boldsymbol{Z}_{t,i}^{(m)}; \boldsymbol{\theta}_t^{(m)}\right), \tag{32}$$

where $\widehat{\boldsymbol{Z}}_{t,i}^{(m)}$ is the state seen by the partially merged model and $\boldsymbol{Z}_{t,i}^{(m)}$ is the state seen by the original task model. Thus, the merged operator is trained to act on the actual merged-model input while matching the task-model local behavior (i.e., output). This creates a corrective objective that explicitly bridges the shifted merged-model state and the task-model target behavior.

Progressive statistics versus corrective local behavior matching. RegMean++ modifies RegMean by making the statistics progressive. SLOA modifies the objective itself. For an ordinary linear map, SLOA uses

$$\min_{\boldsymbol{W}} \sum_{t=1}^{T} \left\| \widehat{\boldsymbol{X}}_t \boldsymbol{W} - \boldsymbol{X}_t \boldsymbol{W}_t \right\|_F^2 + \sum_{t=1}^{T} \|\boldsymbol{W} - \boldsymbol{W}_t\|_{\widehat{\chi}_t}^2 . \tag{33}$$

The left side of the behavior-matching term uses the partially merged input $\widehat{\boldsymbol{X}}_t$, while the target output uses the original task-model behavior $\boldsymbol{X}_t \boldsymbol{W}_t$. Consequently, the closed-form solution depends not only on a merged-model Gram matrix

$$\widehat{\boldsymbol{X}}_t^\top \widehat{\boldsymbol{X}}_t, \tag{34}$$

but also on a cross-state term

$$\widehat{\boldsymbol{X}}_t^\top \boldsymbol{X}_t \boldsymbol{W}_t. \tag{35}$$

This cross-state term directly compensates for the mismatch between the state that the merged model receives and the state under which the task-specific operator was originally used.

RegMean++ instead keeps the RegMean form of the linear-layer solve, but obtains the features progressively from the merged computation. This improves over RegMean by incorporating cross-layer dependencies, but it still treats the current component as an isolated linear layer whose input statistics are updated. SLOA uses the progressive state as part of a local functional matching problem between two execution paths.

Layer-wise progression versus operator-wise topological progression. RegMean++ progresses at the transformer-layer level. Within a transformer layer, it computes statistics for linear layers and merges those linear weights using the RegMean rule. Other parameters, including non-linear or normalization parameters, are merged by averaging.

SLOA progresses at the level of local operators in a topological order. The local operators include LayerNorm affine maps, ordinary linear maps, QK attention operators, and VO attention operators. This finer granularity matters inside a transformer block because the internal attention computations have distinct functional roles: QK controls token routing through attention scores, whereas VO controls the residual write map after attention aggregation. A layer-wise regression method can update statistics progressively, but it does not explicitly represent these two coupled attention mechanisms as separate functional operators. SLOA does.

Treatment of attention. The most important difference is the treatment of self-attention. RegMean++ inherits RegMean's parameterization and applies regression to linear projection matrices. Thus, the attention projections are still treated as individual linear maps. This means that the query, key, value, and output matrices are merged as parameter tensors rather than as the composed operators that determine attention behavior.

SLOA instead defines the QK and VO local operators as

$$\boldsymbol{\theta}_{QK,t}^{(j)} = \frac{\boldsymbol{W}_{Q,t}^{(j)} \big(\boldsymbol{W}_{K,t}^{(j)}\big)^\top}{\sqrt{d_k}}, \qquad \boldsymbol{\theta}_{VO,t}^{(j)} = \boldsymbol{W}_{V,t}^{(j)} \boldsymbol{W}_{O,t}^{(j)}. \tag{36}$$

These are the objects that determine the pre-softmax attention scores and the residual write map, respectively. Therefore, SLOA aligns attention at the level of local function rather than at the level of arbitrary attention-coordinate factors.

For any invertible matrix $\boldsymbol{R}$,

$$\boldsymbol{W}_Q \boldsymbol{W}_K^\top = (\boldsymbol{W}_Q \boldsymbol{R})(\boldsymbol{W}_K \boldsymbol{R}^{-\top})^\top. \tag{37}$$

Thus, two QK factorizations can be functionally identical while their individual query and key matrices differ. RegMean++ can reduce covariate shift through progressive statistics, but it does not remove this factorization ambiguity because it still merges the individual factors. SLOA removes the ambiguity by merging the invariant composed operator.

Operator geometry. RegMean++ uses RegMean-style Gram statistics for linear layers. This remains a one-sided geometry appropriate for maps of the form

$$\boldsymbol{X}\boldsymbol{W}. \tag{38}$$

SLOA uses different geometries for different local operators. For one-sided operators, such as linear maps and VO, SLOA uses a Mahalanobis geometry induced by the calibration activations:

$$\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|_{\widehat{\chi}_t}^2 = \mathrm{Tr}\left[(\boldsymbol{\theta} - \boldsymbol{\theta}_t)^\top \boldsymbol{\Lambda}_t (\boldsymbol{\theta} - \boldsymbol{\theta}_t)\right]. \tag{39}$$

For QK, whose behavior is bilinear in the residual state,

$$\mathcal{A}(\boldsymbol{X}; \boldsymbol{\theta}_{QK}) = \boldsymbol{X}\boldsymbol{\theta}_{QK}\boldsymbol{X}^\top, \tag{40}$$

SLOA uses the two-sided metric

$$\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|_{\widehat{\chi}_t}^2 = \left\| \boldsymbol{\Lambda}_t^{1/2} (\boldsymbol{\theta} - \boldsymbol{\theta}_t) \boldsymbol{\Lambda}_t^{1/2} \right\|_F^2. \tag{41}$$

This operator-specific geometry is absent from RegMean++, because RegMean++ does not formulate QK as a two-sided residual-space operator.

LayerNorm and non-linear components. RegMean++ improves the statistics used for linear-layer merging, but it still handles non-linear and normalization-related parameters by simple averaging. SLOA includes LayerNorm affine parameters in the same local-operator alignment framework. After the normalization statistics are fixed for a given input, the LayerNorm affine map is linear in $(\boldsymbol{\gamma}, \boldsymbol{\beta})$:

$$\mathcal{A}(\boldsymbol{X}; \boldsymbol{\gamma}, \boldsymbol{\beta}) = \nu(\boldsymbol{X}) \,\mathrm{diag}(\boldsymbol{\gamma}) + \mathbf{1}\boldsymbol{\beta}^\top. \tag{42}$$

Therefore, SLOA can align LayerNorm behavior with a closed-form local objective instead of simply averaging the affine parameters. This is useful because LayerNorm parameters affect the scale and shift of the states that downstream operators receive.

Rank and capacity control. RegMean++ preserves the original architecture. Since it merges existing attention projection matrices, the attention head dimension remains fixed at d_k. The method therefore does not expose the spectrum of the merged QK or VO operator and cannot choose how much rank to retain after merging.

SLOA explicitly separates the merged operator from its factorization. After solving for an ambient QK or VO operator

$$\bar{\boldsymbol{\theta}}^{(j)} \in \mathbb{R}^{d\times d}, \tag{43}$$

SLOA factorizes it at a chosen rank r_j. When $r_j = d_k$, the original head dimension is preserved. When $r_j > d_k$, the merged model can retain additional singular directions that arise from combining task-specific low-rank operators. This provides a capacity-control mechanism that is not available to RegMean++.

Different notions of "sequential". Both RegMean++ and SLOA are sequential, but the sequentiality serves different purposes.

RegMean++ is sequential because the statistics used for later layers are recomputed from activations influenced by earlier merged layers. Its main goal is to make RegMean's input statistics more faithful to the merged model's internal feature distribution.

SLOA is sequential because each local operator is aligned under the actual execution path of the partially merged model. The target remains the task-specific local behavior, while the input to the merged operator is the partially merged input. Thus, each local solve is not only statistics-aware but also corrective:

$$\text{merged operator on merged state} \quad \approx \quad \text{task operator on task state.} \tag{44}$$

SLOA's cross-state matching explicitly accounts for that shift.

C.3 Comparison with rotation-symmetry matching

We provide a detailed comparison with weight space rotation-symmetry matching (Zhang et al., 2025). This work is closely related to ours because it also recognizes that transformer attention contains nontrivial symmetries beyond permutation symmetry. However, the two methods use this observation in fundamentally different ways. Rotation-symmetry matching searches for functionally equivalent parameterizations of the input models that are closer in weight space before applying a conventional model-fusion rule. SLOA instead changes the object being merged: it constructs, aligns, and merges the local functional operators themselves.

Rotation-symmetry matching. (Zhang et al., 2025) observe that the self-attention factors are not individually identifiable. For a single attention head, if $\boldsymbol{R}$ is an orthogonal matrix, then the transformation

$$\boldsymbol{W}_Q \mapsto \boldsymbol{W}_Q\boldsymbol{R}, \qquad \boldsymbol{W}_K \mapsto \boldsymbol{W}_K\boldsymbol{R} \tag{45}$$

preserves the QK score operator because

$$(\boldsymbol{W}_Q\boldsymbol{R})(\boldsymbol{W}_K\boldsymbol{R})^\top = \boldsymbol{W}_Q\boldsymbol{R}\boldsymbol{R}^\top\boldsymbol{W}_K^\top = \boldsymbol{W}_Q\boldsymbol{W}_K^\top. \tag{46}$$

Similarly, for the value-output path, the transformation

$$\boldsymbol{W}_V \mapsto \boldsymbol{W}_V\boldsymbol{S}, \qquad \boldsymbol{W}_O \mapsto \boldsymbol{S}^\top\boldsymbol{W}_O, \tag{47}$$

with orthogonal $\boldsymbol{S}$, preserves the VO operator:

$$(\boldsymbol{W}_V\boldsymbol{S})(\boldsymbol{S}^\top\boldsymbol{W}_O) = \boldsymbol{W}_V\boldsymbol{W}_O. \tag{48}$$

Based on these invariances, rotation-symmetry matching chooses rotations that make one model's attention parameters close to another model's attention parameters in Euclidean distance. For example, in the QK case, the matching problem can be viewed as an orthogonal Procrustes-type objective of the form

$$\min_{\boldsymbol{R}^\top\boldsymbol{R}=\boldsymbol{I}} \|\boldsymbol{W}_{Q,1}\boldsymbol{R} - \boldsymbol{W}_{Q,2}\|_F^2 + \|\boldsymbol{W}_{K,1}\boldsymbol{R} - \boldsymbol{W}_{K,2}\|_F^2, \tag{49}$$

with an analogous objective for the VO factors. After this matching step, the transformed models are merged using an existing fusion method, such as weight averaging, Fisher merging, or RegMean. Thus, rotation-symmetry matching is a *preprocessing* or *re-basing* step for parameter-space fusion.

SLOA merges operators, not matched parameter factors. SLOA starts from the same structural fact that QK and VO factors are not individually meaningful. However, rather than searching for rotations that make the factors close in parameter space, SLOA eliminates the factorization ambiguity by directly forming the composed operators

$$\boldsymbol{\theta}_{QK,t}^{(j)} = \frac{\boldsymbol{W}_{Q,t}^{(j)} \big(\boldsymbol{W}_{K,t}^{(j)}\big)^\top}{\sqrt{d_k}}, \qquad \boldsymbol{\theta}_{VO,t}^{(j)} = \boldsymbol{W}_{V,t}^{(j)} \boldsymbol{W}_{O,t}^{(j)}. \tag{50}$$

The optimization variable in SLOA is the operator $\boldsymbol{\theta}$ itself, not a rotation matrix and not the individual factors. For each local operator, SLOA solves

$$\boldsymbol{\theta}_{\mathrm{merge}} = \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \Theta_{\mathrm{amb}}} \mathcal{L}_{\mathrm{act}}(\boldsymbol{\theta}) + \frac{1}{T} \sum_{t=1}^{T} \|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|_{\widehat{\chi}_t}^2 . \tag{51}$$

The merged operator is then factorized back into attention-compatible weights. Thus, the order of operations is different and this difference is central. Rotation matching preserves the function of each individual model before fusion, but the final fused model is still produced by merging the individual parameter factors. SLOA instead performs the merge in the space of the functional objects that determine the local computation.

Why factor matching does not remove factor-level interference. Even after rotation matching, factor-wise merging can introduce cross terms that are not determined by the desired QK or VO operators. Consider two QK heads, and suppose model 1 has been rotation-matched to model 2 using an orthogonal matrix $\boldsymbol{R}$. If we then average the matched factors,

$$\bar{\boldsymbol{W}}_Q = \frac{1}{2} \left(\boldsymbol{W}_{Q,1} \boldsymbol{R} + \boldsymbol{W}_{Q,2} \right), \qquad \bar{\boldsymbol{W}}_K = \frac{1}{2} \left(\boldsymbol{W}_{K,1} \boldsymbol{R} + \boldsymbol{W}_{K,2} \right), \tag{52}$$

then the resulting QK operator is

$$\bar{\boldsymbol{W}}_Q \bar{\boldsymbol{W}}_K^\top = \frac{1}{4} \Big(\boldsymbol{W}_{Q,1} \boldsymbol{W}_{K,1}^\top + \boldsymbol{W}_{Q,2} \boldsymbol{W}_{K,2}^\top + \boldsymbol{W}_{Q,1} \boldsymbol{R} \boldsymbol{W}_{K,2}^\top + \boldsymbol{W}_{Q,2} \boldsymbol{R}^\top \boldsymbol{W}_{K,1}^\top \Big).$$

The first two terms are the task-specific QK operators. The last two terms are cross terms that depend on the particular factorization and the chosen rotation. They are not functions only of the task operators $\boldsymbol{W}_{Q,1} \boldsymbol{W}_{K,1}^\top$ and $\boldsymbol{W}_{Q,2} \boldsymbol{W}_{K,2}^\top$. Thus, even if the rotation makes the factors closer, the final factor-wise average is not the same as an operator-level merge.

The same issue appears for VO. After matching model 1 with an orthogonal matrix $\boldsymbol{S}$, factor averaging gives

$$\bar{\boldsymbol{W}}_V = \frac{1}{2} \left(\boldsymbol{W}_{V,1} \boldsymbol{S} + \boldsymbol{W}_{V,2} \right), \qquad \bar{\boldsymbol{W}}_O = \frac{1}{2} \left(\boldsymbol{S}^\top \boldsymbol{W}_{O,1} + \boldsymbol{W}_{O,2} \right), \tag{53}$$

and therefore

$$\bar{\boldsymbol{W}}_V \bar{\boldsymbol{W}}_O = \frac{1}{4} \Big(\boldsymbol{W}_{V,1} \boldsymbol{W}_{O,1} + \boldsymbol{W}_{V,2} \boldsymbol{W}_{O,2} + \boldsymbol{W}_{V,1} \boldsymbol{S} \boldsymbol{W}_{O,2} + \boldsymbol{W}_{V,2} \boldsymbol{S}^\top \boldsymbol{W}_{O,1} \Big).$$

Again, the cross terms are artifacts of merging the factors rather than the composed VO operators. SLOA avoids this issue by solving directly for the merged QK and VO operators.

Different invariance classes. Rotation-symmetry matching works within an orthogonal symmetry class, optionally combined with scalar rescaling. This is natural for parameter matching because orthogonal transformations preserve Frobenius norms and lead to efficient Procrustes-style solutions. However, the functional non-identifiability of QK and VO operators is larger. For any invertible matrix $\boldsymbol{R}$, the QK product is preserved by

$$\boldsymbol{W}_Q \mapsto \boldsymbol{W}_Q \boldsymbol{R}, \qquad \boldsymbol{W}_K \mapsto \boldsymbol{W}_K \boldsymbol{R}^{-\top}, \tag{54}$$

since

$$(\boldsymbol{W}_Q \boldsymbol{R})(\boldsymbol{W}_K \boldsymbol{R}^{-\top})^\top = \boldsymbol{W}_Q \boldsymbol{R} \boldsymbol{R}^{-1} \boldsymbol{W}_K^\top = \boldsymbol{W}_Q \boldsymbol{W}_K^\top. \tag{55}$$

Similarly, the VO product is invariant under

$$\boldsymbol{W}_V \mapsto \boldsymbol{W}_V \boldsymbol{R}, \qquad \boldsymbol{W}_O \mapsto \boldsymbol{R}^{-1} \boldsymbol{W}_O. \tag{56}$$

SLOA is invariant to this full product-level ambiguity because it only depends on the composed operator. If two factorizations realize the same QK or VO operator, they become exactly the same object for SLOA. By contrast, rotation matching selects a representative within a restricted symmetry family and then still applies a parameter-space fusion rule.

Different objectives. The optimality guarantees of the two methods are for different problems. Rotation-symmetry matching is optimal for choosing an equivalent parameterization that minimizes Euclidean distance to an anchor model under orthogonal rotation constraints. Its objective is parameter proximity, where it chooses an equivalent representative that is close in weight space. This is useful because many model-fusion methods perform better when the input models are close in parameter space.

SLOA's closed-form solution is optimal for a local data-aware operator-alignment objective. Its objective is local functional agreement:

$$\mathcal{A}\left(\widehat{\boldsymbol{Z}}_{t,i};\boldsymbol{\theta}\right) \approx \mathcal{A}\left(\boldsymbol{Z}_{t,i};\boldsymbol{\theta}_t\right), \tag{57}$$

together with a data-dependent operator-space regularizer. Thus, SLOA does not ask for the closest equivalent factorization of a task model. It asks for the merged local operator that best reproduces the task-specific local behaviors under the activations that the partially merged model will actually produce.

Sequential correction versus independent parameter matching. Rotation-symmetry matching is applied as a matching step between model parameters, typically module by module or layer by layer. The matching transformation preserves the function of the source model before fusion, but it does not explicitly account for the activation distribution induced by earlier merged components.

SLOA is sequential in a different sense. At operator o_m, it evaluates the merged operator on the local input $\widehat{\boldsymbol{Z}}_{t,i}^{(m)}$ produced by the partially merged model, while the target is the output of the task-specific operator on the task-model local input $\boldsymbol{Z}_{t,i}^{(m)}$:

$$\mathcal{A}\left(\widehat{\boldsymbol{Z}}_{t,i}^{(m)};\boldsymbol{\theta}\right) \approx \mathcal{A}\left(\boldsymbol{Z}_{t,i}^{(m)};\boldsymbol{\theta}_t^{(m)}\right). \tag{58}$$

This makes each local merge corrective with respect to upstream merging errors. The method therefore aligns operators under the actual execution path of the merged network, rather than only reducing parameter distance between the original task models.

Anchor-based matching versus symmetric multi-model alignment. The core rotation-matching formulation aligns one source model to an anchor model. For multi-model fusion, this requires choosing an anchor or performing matching relative to some reference representation. This anchor choice is natural for parameter matching, but it introduces an asymmetry among the input models.

SLOA treats all task models symmetrically in the local objective:

$$\frac{1}{T}\sum_{t=1}^{T}\frac{1}{n_t}\sum_{i=1}^{n_t}\left\|\mathcal{A}\left(\widehat{\boldsymbol{Z}}_{t,i};\boldsymbol{\theta}\right)-\mathcal{A}\left(\boldsymbol{Z}_{t,i};\boldsymbol{\theta}_t\right)\right\|_F^2. \tag{59}$$

No single task model serves as the canonical parameter reference. The merged operator is estimated from a balanced objective over all task-specific operators and calibration states.

Rank and capacity control. Rotation-symmetry matching preserves the original architecture during matching and subsequent fusion. In particular, the attention head dimension remains fixed, and the matched QK and VO products remain represented through the original head bottleneck.

SLOA decouples the merged operator from its factorization rank. The aligned QK and VO operators are first solved in the ambient residual-space operator space $\mathbb{R}^{d\times d}$. Only after the operator is obtained do we choose a factorization rank r. When $r = d_k$, the original head dimension is preserved. When $r > d_k$, SLOA can retain additional singular directions of the merged operator that arise from combining multiple task-specific subspaces. This rank-expansion mechanism has no analogue in rotation-symmetry matching, because rotation matching modifies existing factors but does not expose or refactorize the merged operator spectrum.

Data usage and scope. Rotation-symmetry matching is primarily data-free: the matching transformation is computed from model parameters and can be plugged into several downstream fusion rules. This is an advantage when calibration data are unavailable. SLOA uses calibration data and therefore targets the training-free, data-aware setting. The calibration data serve two roles: they define the local behavior-matching targets and they induce the data-dependent operator metric used for

regularization. Thus, SLOA trades the data-free nature of rotation matching for a more functional, data-aware merge.

Complementarity. The two methods should be viewed as complementary rather than redundant. Rotation-symmetry matching shows that transformer attention has useful continuous parameter-space symmetries and that exploiting these symmetries can improve conventional model fusion. SLOA takes the next step from symmetry-aware factor matching to symmetry-invariant operator merging. Instead of choosing better representatives within the equivalence class and then merging factors, SLOA quotients out the factorization ambiguity by working directly with the composed QK and VO operators.

C.4 Comparison to ProDistill

We provide a detailed comparison between SLOA and ProDistill (Xu et al., 2025), as both methods use calibration data and proceed progressively through the network. Despite this superficial similarity, the two methods address different merging problems and optimize fundamentally different objects.

Overview of ProDistill. ProDistill views model merging as a multi-teacher distillation problem. Let $\boldsymbol{\theta}_0$ be the pretrained initialization and let $\boldsymbol{\theta}_t$ be the fine-tuned model for task t. Writing the task vector as $\boldsymbol{\tau}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_0$, ProDistill parameterizes the merged model as

$$\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \boldsymbol{\lambda}_t \circ \boldsymbol{\tau}_t, \tag{60}$$

where $\boldsymbol{\lambda}_t$ are learned merging coefficients and $\circ$ denotes element-wise multiplication. The method optimizes these coefficients so that the intermediate representations of the merged model match those of the fine-tuned teacher models on a small unlabeled validation set.

To reduce memory cost, ProDistill decomposes the objective layer by layer. At layer ℓ, it solves an optimization problem of the form

$$\min_{\{\boldsymbol{\lambda}_t^{(\ell)}\}_{t=1}^{T}} \sum_{t=1}^{T} \frac{1}{|D_t|} \sum_{(z_1,z_2)\in D_t^{(\ell-1)}} \left\| \phi^{(\ell)}\left(\boldsymbol{\theta}_0^{(\ell)} + \sum_{s=1}^{T} \boldsymbol{\lambda}_s^{(\ell)} \circ \boldsymbol{\tau}_s^{(\ell)}, z_1\right) - \phi^{(\ell)}\left(\boldsymbol{\theta}_t^{(\ell)}, z_2\right) \right\|_2^2, \tag{61}$$

where z_1 is the input activation produced by the partially merged model and z_2 is the corresponding input activation produced by the fine-tuned model. After optimizing the coefficients for layer ℓ, ProDistill updates the paired activations and proceeds to the next layer. Thus, ProDistill is a progressive activation-distillation method over task-vector coefficients.

SLOA is training-free, whereas ProDistill is training-based. The most direct difference is the optimization procedure. ProDistill trains the merging coefficients $\boldsymbol{\lambda}_t^{(\ell)}$ using gradient descent. Although the optimization is restricted to one layer at a time, it still requires iterative updates, gradients, learning-rate choices, optimizer states, and stopping criteria.

SLOA, in contrast, is training-free. For each local operator, SLOA solves a regularized least-squares problem in closed form:

$$\boldsymbol{\theta}_{\text{merge}} = \operatorname*{arg\,min}_{\boldsymbol{\theta}\in\Theta_{\text{amb}}} \mathcal{L}_{\text{act}}(\boldsymbol{\theta}) + \frac{1}{T}\sum_{t=1}^{T} \|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|_{\widehat{\chi}_t}^2 . \tag{62}$$

The closed-form solution is possible because SLOA chooses *local functional operators* whose behavior maps are linear with respect to $\boldsymbol{\theta}$. Thus, SLOA does not learn merging coefficients by gradient descent; it directly solves for the merged local operator.

SLOA merges local functional operators, whereas ProDistill merges task-vector parameter coordinates. ProDistill retains the standard task-vector parameterization. For each layer, the merged layer remains in the affine span

$$\boldsymbol{\theta}_0^{(\ell)} + \operatorname{span}_{\circ}\left\{\boldsymbol{\tau}_1^{(\ell)}, \ldots, \boldsymbol{\tau}_T^{(\ell)}\right\}, \tag{63}$$

where the coefficients act element-wise on the task-vector coordinates. This gives ProDistill a flexible parameter-space interpolation mechanism, but the primitive objects being combined are still the original parameter tensors.

SLOA instead changes the unit of merging. In transformer attention, the individual matrices $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}_O$ are not the local functional units. The attention scores depend on the composed QK operator

$$\boldsymbol{\theta}_{\mathrm{QK}} = \frac{1}{\sqrt{d_k}} \boldsymbol{W}_Q \boldsymbol{W}_K^\top, \tag{64}$$

and the logits depends on the composed VO operator

$$\boldsymbol{\theta}_{\mathrm{VO}} = \boldsymbol{W}_V \boldsymbol{W}_O. \tag{65}$$

SLOA merges these composed operators directly. This distinction is central: ProDistill optimizes a block-level representation-matching objective over task-vector coefficients, while SLOA constructs a *local* operator-level objective over the functional objects that determine the computation.

SLOA removes attention-factorization ambiguity by construction. Attention heads have internal reparameterization symmetries. For any invertible matrix $\boldsymbol{R}$,

$$\boldsymbol{W}_Q \boldsymbol{W}_K^\top = (\boldsymbol{W}_Q \boldsymbol{R})(\boldsymbol{W}_K \boldsymbol{R}^{-\top})^\top. \tag{66}$$

Thus, two attention heads can produce identical QK score operators while using different query/key coordinates. An analogous ambiguity exists for the value/output factors. Any method that operates directly on the individual matrices $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}_O$ must handle this coordinate ambiguity implicitly.

ProDistill uses an activation loss, so its objective can partially compensate for poor parameter-space interpolation. However, its parameterization still acts on the individual task-vector coordinates of the original projection matrices. It does not explicitly identify the invariant QK and VO operators, nor does it merge in the quotient space induced by attention-head reparameterizations. Consequently, the factorization ambiguity remains inside the optimization problem.

SLOA removes this ambiguity before solving the merge. Since the optimization variable is the composed operator itself, two different factorizations that realize the same QK or VO operator correspond to the same object in SLOA. The merged operator is therefore invariant to internal changes of head coordinates, and factorization is postponed until after the operator has been aligned.

SLOA is more fine-grained than layer-wise distillation inside transformer blocks. ProDistill progresses layer by layer. The layer is treated as the unit whose output representation should match the teacher representation. This is a general design and can be applied to nonlinear modules because the optimization is performed by gradient descent.

SLOA instead uses a topological order over local operators inside the network. For a transformer block, SLOA can separately align LayerNorm affine maps, standard linear maps, QK attention operators, and VO attention operators. This allows the method to respect the internal computational structure of attention rather than treating the whole attention layer or block as a single black-box feature map. In particular, SLOA distinguishes the two roles of self-attention:

- the QK operator determines token-token routing through the pre-softmax attention scores;
- the VO operator determines what information is written back to the logit after attention weights are applied.

This decomposition is absent from ProDistill. ProDistill aligns block outputs, whereas SLOA aligns the local functional mechanisms that produce those outputs.

Both methods are progressive, but the progressive state has a different meaning. Both ProDistill and SLOA use a progressive procedure in which later modules are merged under activations affected by earlier merges. However, the role of these activations differs.

In ProDistill, the progressive state consists of paired features (z_1, z_2), where z_1 is produced by the partially merged student and z_2 is produced by the corresponding teacher. The layer objective trains task-vector coefficients so that the student layer output matches the teacher layer output on these paired inputs.

In SLOA, the progressive state defines the local inputs $\widehat{\boldsymbol{Z}}_{t,i}^{(m)}$ received by the partially merged model and $\boldsymbol{Z}_{t,i}^{(m)}$ received by the task model. The merged operator is evaluated on the former, while the target output is generated by the task-specific local operator on the latter:

$$\mathcal{A}\left(\widehat{\boldsymbol{Z}}_{t,i}^{(m)};\boldsymbol{\theta}\right) \approx \mathcal{A}\left(\boldsymbol{Z}_{t,i}^{(m)};\boldsymbol{\theta}_t^{(m)}\right). \tag{67}$$

Thus, SLOA is progressive at the level of local operator alignment, not at the level of learned task-vector coefficients. Each merge is corrective under the actual execution path of the partially merged network, while still solving a closed-form local regression problem.

SLOA decouples operator alignment from factorization rank. A further distinction is architectural capacity. ProDistill keeps the original model architecture and learns coefficients for the existing task-vector parameters. In particular, an attention head retains the original head dimension. Thus, ProDistill does not create a mechanism for increasing the rank of the QK or VO operators after merging.

SLOA explicitly separates the ambient operator from its final factorization. For a head with original dimension d_k, each task-specific QK or VO operator has rank at most d_k. However, the merged multi-task operator may have larger effective rank because different tasks can use different singular directions. SLOA first solves for the merged operator in the ambient space $\Theta_{\text{amb}} = \mathbb{R}^{d\times d}$ and only afterward factorizes it at a chosen rank r. When $r = d_k$, SLOA preserves the original head width. When $r > d_k$, SLOA retains additional singular directions supported by the aligned operator. This rank-expansion mechanism is not available in ProDistill, because ProDistill does not decouple the merged attention operator from its original factorization.

The two methods make different trade-offs. ProDistill is a general distillation-based merging framework. Its strength is that it can optimize feature matching for arbitrary modules without requiring the module to be linear in a chosen operator. This generality comes at the cost of gradient-based optimization and a parameter-coordinate task-vector representation.

SLOA makes the opposite trade-off. It focuses on modules that admit a local behavior map linear in an appropriate operator. This restriction enables closed-form merging, data-dependent operator geometry, attention-operator invariance, and post-merge rank control. Therefore, SLOA is not simply a training-free variant of ProDistill. It changes the mathematical object being merged: from task-vector coordinates trained to match layer features, to local functional operators solved by sequential regularized regression.

D Detailed derivation of local operator alignment

As described in §3, the Sequential Local Operator Alignment (SLOA) method merges a single operator at a time. By explicitly writing the term $\mathcal{L}_{\text{act}}(\boldsymbol{\theta})$, the unified loss objective for merging is defined as:

$$L(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\left\|\mathcal{A}(\widehat{\boldsymbol{Z}}_{t,i};\boldsymbol{\theta}) - \mathcal{A}(\boldsymbol{Z}_{t,i};\boldsymbol{\theta}_t)\right\|_F^2 + \|\boldsymbol{\theta}-\boldsymbol{\theta}_t\|_{\widehat{\chi}_t}^2\right) \tag{68}$$

$$\boldsymbol{\theta}_{\text{merge}} = \arg\min_{\boldsymbol{\theta}\in\Theta_{\text{amb}}} L(\boldsymbol{\theta}) \tag{69}$$

Here, $\widehat{\chi}_t = \{\widehat{\boldsymbol{Z}}_{t,i}\}_{i=1}^{n_t}$ is the collection of local inputs produced by the partially merged model. For standard linear operators and QK operator that taking only the activation as input, $\widehat{\boldsymbol{Z}}_{t,i} = \widehat{\boldsymbol{X}}_{t,i} \in \mathbb{R}^{L\times d}$, while for components like the VO circuit, it also includes the composed attention matrix.

Now, for each operator, we will show how this unified loss objective can be reduced to a structured regression problem with a closed-form solution. We will also detail the explicit form of the data-dependent regularization term and the ambient operator space Θ_{amb} for each component, applying constraints (such as rank) during the subsequent factorization step for each operators mentioned in §3.1.

D.1 Optimal QK Operator

For the Query-Key circuit, for each attention head j, let $\boldsymbol{W}_Q^{(j)}, \boldsymbol{W}_K^{(j)} \in \mathbb{R}^{d\times d_k}$ denote the query and key matrices, respectively. The consolidated score operator is defined as $\boldsymbol{\theta}^{(j)} = \frac{1}{\sqrt{d_k}} \boldsymbol{W}_Q^{(j)} \boldsymbol{W}_K^{(j)\top} \in \mathbb{R}^{d\times d}$. Since the operation acts bilinearly on the input representations, when evaluated on a single calibration example i from task t, the behavior map and the customized task-level regularization term are defined as:

$$\mathcal{A}(\widehat{\boldsymbol{Z}}_{t,i}; \boldsymbol{\theta}^{(j)}) = \widehat{\boldsymbol{Z}}_{t,i} \boldsymbol{\theta}^{(j)} \widehat{\boldsymbol{Z}}_{t,i}^\top, \quad \left\| \boldsymbol{\theta}^{(j)} - \boldsymbol{\theta}_t^{(j)} \right\|_{\widehat{\chi}_t}^2 = \left\| \boldsymbol{\Lambda}_t^{1/2} (\boldsymbol{\theta}^{(j)} - \boldsymbol{\theta}_t^{(j)}) \boldsymbol{\Lambda}_t^{1/2} \right\|_F^2 \tag{70}$$

where $\boldsymbol{\Lambda}_t \in \mathbb{R}^{d\times d}$ is a diagonal precision matrix derived from the partially merged state collection $\widehat{\chi}_t$, and $\boldsymbol{\Lambda}_t^{1/2}$ is its principal square root. Hereafter, we drop the head index j for notational brevity, as the alignment is solved independently for each head.

To find the optimal operator, we first relax the rank-d_k structural constraint Θ_{amb} to solve for an unconstrained consolidated operator (the rank constraint is subsequently enforced via Singular Value Decomposition). Substituting the behavior map into the unified loss objective (68) and writing $\mathcal{A}(\boldsymbol{Z}_{t,i}; \boldsymbol{\theta}_t) = \boldsymbol{Y}_{t,i}$ as target, we have:

$$L(\boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^{T} \left(\frac{1}{n_t} \sum_{i=1}^{n_t} \left\| \widehat{\boldsymbol{Z}}_{t,i} \boldsymbol{\theta} \widehat{\boldsymbol{Z}}_{t,i}^\top - \boldsymbol{Y}_{t,i} \right\|_F^2 + \left\| \boldsymbol{\Lambda}_t^{1/2} (\boldsymbol{\theta} - \boldsymbol{\theta}_t) \boldsymbol{\Lambda}_t^{1/2} \right\|_F^2 \right) \tag{71}$$

where n_t is the number of calibration samples for task t.

Because the relaxed objective function $L(\boldsymbol{\theta})$ is strictly convex, the global minimum is found by setting the gradient with respect to $\boldsymbol{\theta}$ to zero. Utilizing the matrix derivative identity $\nabla_{\boldsymbol{\theta}} \|\boldsymbol{A}\boldsymbol{\theta}\boldsymbol{A}^\top - \boldsymbol{Y}\|_F^2 = 2\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{\theta}\boldsymbol{A}^\top - \boldsymbol{Y})\boldsymbol{A}$ and the linearity of the gradient over the finite sum, we obtain:

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \frac{2}{T} \sum_{t=1}^{T} \left(\frac{1}{n_t} \sum_{i=1}^{n_t} \widehat{\boldsymbol{Z}}_{t,i}^\top \left(\widehat{\boldsymbol{Z}}_{t,i} \boldsymbol{\theta} \widehat{\boldsymbol{Z}}_{t,i}^\top - \boldsymbol{Y}_{t,i} \right) \widehat{\boldsymbol{Z}}_{t,i} + \boldsymbol{\Lambda}_t (\boldsymbol{\theta} - \boldsymbol{\theta}_t) \boldsymbol{\Lambda}_t \right) = 0 \tag{72}$$

Dividing by the scalar $\frac{2}{T}$ and distributing the terms inside the summations gives:

$$\sum_{t=1}^{T} \left(\frac{1}{n_t} \sum_{i=1}^{n_t} \left(\widehat{\boldsymbol{Z}}_{t,i}^\top \widehat{\boldsymbol{Z}}_{t,i} \boldsymbol{\theta} \widehat{\boldsymbol{Z}}_{t,i}^\top \widehat{\boldsymbol{Z}}_{t,i} - \widehat{\boldsymbol{Z}}_{t,i}^\top \boldsymbol{Y}_{t,i} \widehat{\boldsymbol{Z}}_{t,i} \right) + \boldsymbol{\Lambda}_t \boldsymbol{\theta} \boldsymbol{\Lambda}_t - \boldsymbol{\Lambda}_t \boldsymbol{\theta}_t \boldsymbol{\Lambda}_t \right) = 0 \tag{73}$$

We group the terms dependent on the merged parameter $\boldsymbol{\theta}$ on the left-hand side, and move the constant target terms to the right-hand side:

$$\sum_{t=1}^{T} \left(\frac{1}{n_t} \sum_{i=1}^{n_t} \left(\widehat{\boldsymbol{Z}}_{t,i}^\top \widehat{\boldsymbol{Z}}_{t,i} \boldsymbol{\theta} \widehat{\boldsymbol{Z}}_{t,i}^\top \widehat{\boldsymbol{Z}}_{t,i} \right) + \boldsymbol{\Lambda}_t \boldsymbol{\theta} \boldsymbol{\Lambda}_t \right) = \sum_{t=1}^{T} \left(\frac{1}{n_t} \sum_{i=1}^{n_t} \left(\widehat{\boldsymbol{Z}}_{t,i}^\top \boldsymbol{Y}_{t,i} \widehat{\boldsymbol{Z}}_{t,i} \right) + \boldsymbol{\Lambda}_t \boldsymbol{\theta}_t \boldsymbol{\Lambda}_t \right) \tag{74}$$

To solve Equation 74 analytically, we define $\boldsymbol{R}_{t,i} = \widehat{\boldsymbol{Z}}_{t,i}^\top \widehat{\boldsymbol{Z}}_{t,i} \in \mathbb{R}^{d\times d}$, and let $\boldsymbol{C} \in \mathbb{R}^{d\times d}$ denote the right-hand side *constant* matrix. The equation simplifies to:

$$\sum_{t=1}^{T} \left(\left[\frac{1}{n_t} \sum_{i=1}^{n_t} \boldsymbol{R}_{t,i} \boldsymbol{\theta} \boldsymbol{R}_{t,i} \right] + \boldsymbol{\Lambda}_t \boldsymbol{\theta} \boldsymbol{\Lambda}_t \right) = \boldsymbol{C} \tag{75}$$

To isolate $\boldsymbol{\theta}$, we apply the vectorization operator $\text{vec}(\cdot)$. Using the Kronecker product identity $\text{vec}(\boldsymbol{A}\boldsymbol{\theta}\boldsymbol{B}) = (\boldsymbol{B}^\top \otimes \boldsymbol{A}) \text{vec}(\boldsymbol{\theta})$ and noting that both $\boldsymbol{R}_{t,i}$ and $\boldsymbol{\Lambda}_t$ are symmetric, we obtain:

$$\sum_{t=1}^{T} \left(\left[\frac{1}{n_t} \sum_{i=1}^{n_t} (\boldsymbol{R}_{t,i} \otimes \boldsymbol{R}_{t,i}) \right] + (\boldsymbol{\Lambda}_t \otimes \boldsymbol{\Lambda}_t) \right) \text{vec}(\boldsymbol{\theta}) = \text{vec}(\boldsymbol{C}) \tag{76}$$

By defining the aggregated precision matrix $\boldsymbol{H} \in \mathbb{R}^{d^2\times d^2}$ as the operator on the left-hand side, the system reduces to the linear form:

$$\boldsymbol{H} \, \text{vec}(\boldsymbol{\theta}) = \text{vec}(\boldsymbol{C}) \tag{77}$$

Assuming $\boldsymbol{H}$ is invertible (or utilizing the Moore-Penrose pseudo-inverse $\boldsymbol{H}^{\dagger}$ in rank-deficient scenarios), we isolate the vectorized parameters and apply the inverse vectorization operator to recover the optimal unconstrained operator $\boldsymbol{\theta}_{\text{merge}} \in \mathbb{R}^{d\times d}$:

$$\boldsymbol{\theta}_{\text{merge}} = \text{vec}^{-1}\left(\boldsymbol{H}^{-1}\,\text{vec}(\boldsymbol{C})\right) \quad \square \tag{78}$$

The above derivation yields an exact closed-form analytical solution for the optimal QK operator, which is subsequently factorized into $\widehat{\boldsymbol{W}}_Q$ and $\widehat{\boldsymbol{W}}_K$ via SVD to reintroduce the rank constraint.

Computational Remark. While Equation 78 provides an exact analytical equality, directly materializing and inverting the Kronecker matrix $\boldsymbol{H}$ incurs an intractable time complexity of $\mathcal{O}(d^6)$ and space complexity of $\mathcal{O}(d^4)$. For standard Transformer dimensions (e.g., $d \geq 768$), this is computationally prohibitive. Therefore, in practice, we compute the optimal $\boldsymbol{\theta}_{\text{merge}}$ iteratively without ever materializing $\boldsymbol{H}$ by employing the Conjugate Gradient (CG) algorithm [1].

D.1.1 Rank-Constrained Factorization of the QK Operator

Following the unconstrained optimization derived in §D.1, the optimal consolidated operator $\boldsymbol{\theta}_{\text{merge}} \in \mathbb{R}^{d\times d}$ must be projected back onto the structural constraints of the attention mechanism. By the Eckart-Young-Mirsky theorem, the optimal rank-r approximation in the Frobenius norm is obtained via the truncated Singular Value Decomposition (SVD):

$$\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top} \quad \Longrightarrow \quad \boldsymbol{\theta}^{(r)}_{\text{merge}} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^{\top} \tag{79}$$

where $\boldsymbol{U}_r, \boldsymbol{V}_r \in \mathbb{R}^{d\times r}$ contain the top r left and right singular vectors, respectively, and $\boldsymbol{\Sigma}_r \in \mathbb{R}^{r\times r}$ is the diagonal matrix of the top r singular values. The standard attention mechanism computes the QK logit matrix as $\frac{1}{c}\widehat{\boldsymbol{W}}_Q\widehat{\boldsymbol{W}}_K^{\top}$, where c is the attention scaling denominator (e.g., $c = \sqrt{r}$ or $c = \sqrt{d_k}$). To satisfy the equality $\frac{1}{c}\widehat{\boldsymbol{W}}_Q\widehat{\boldsymbol{W}}_K^{\top} = \boldsymbol{\theta}^{(r)}_{\text{merge}}$, we distribute the singular values and the scalar c symmetrically across the QK weight matrices:

$$\widehat{\boldsymbol{W}}_Q = \boldsymbol{U}_r(c\boldsymbol{\Sigma}_r)^{1/2} \in \mathbb{R}^{d\times r} \tag{80}$$

$$\widehat{\boldsymbol{W}}_K = \boldsymbol{V}_r(c\boldsymbol{\Sigma}_r)^{1/2} \in \mathbb{R}^{d\times r} \tag{81}$$

D.2 Optimal VO Operator

For value-output (VO) circuits, for each head j, we have $\boldsymbol{W}_V^{(j)} \in \mathbb{R}^{d\times d_k}$ as the value matrix, and $\boldsymbol{W}_O^{(j)} \in \mathbb{R}^{d_k\times d}$ as the output matrix. The consolidated operator is defined as $\boldsymbol{\theta}^{(j)} = \boldsymbol{W}_V^{(j)}\boldsymbol{W}_O^{(j)} \in \mathbb{R}^{d\times d}$. In accordance with the sequential topological ordering described in §3, the VO operator takes as input the activation $\widehat{\boldsymbol{X}}_{t,i}$ as well as the attention matrix $\widehat{\boldsymbol{A}}_{t,i} \in \mathbb{R}^{L\times L}$, which is computed using the previously aligned QK operator. We explicitly define the partially merged local input state as $\widehat{\boldsymbol{Z}}_{t,i} = \widehat{\boldsymbol{A}}_{t,i}\widehat{\boldsymbol{X}}_{t,i}$. The operation of the VO circuit is linear with respect to the input representations, and thus when evaluated on a single calibration example i from task t, the behavior map and the customized task-level regularization term are defined as:

$$\mathcal{A}(\widehat{\boldsymbol{Z}}_{t,i};\boldsymbol{\theta}) = \widehat{\boldsymbol{Z}}_{t,i}\boldsymbol{\theta}, \quad \|\boldsymbol{\theta}-\boldsymbol{\theta}_t\|^2_{\widehat{\chi}_t} = \left\|\boldsymbol{\Lambda}_t^{1/2}(\boldsymbol{\theta}-\boldsymbol{\theta}_t)\right\|_F^2 \tag{82}$$

where $\widehat{\chi}_t = \{\widehat{\boldsymbol{Z}}_{t,i}\}_{i=1}^{n_t}$ representing attention-weighted input representations. Similar to the QK case, we droped j for ease of notaion, since the same operation is opertated for each attention head. $\boldsymbol{\Lambda}_t$ is a diagonal matrix derived from the attention score weighted partially merged state collection $\widehat{\chi}_t$, and $\boldsymbol{\Lambda}_t^{1/2}$ is its square root.

For the VO operator, we solve for $L(\boldsymbol{\theta})$ with the structural constraint $\Theta_{\text{amb}} = \{rank(\boldsymbol{\theta}) \leq r\}$. Substituting these into the unified objective function (5) and writing $\mathcal{A}(\boldsymbol{Z}_{t,i};\boldsymbol{\theta}_t) = \boldsymbol{Y}_{t,i}$ as target yields:

$$L(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\left\|\widehat{\boldsymbol{Z}}_{t,i}\boldsymbol{\theta}-\boldsymbol{Y}_{t,i}\right\|_F^2 + \left\|\boldsymbol{\Lambda}_t^{1/2}(\boldsymbol{\theta}-\boldsymbol{\theta}_t)\right\|_F^2\right) \tag{83}$$

$$\boldsymbol{\theta}_{\text{merge}} = \underset{\boldsymbol{\theta}\in\{rank(\boldsymbol{\theta})\leq r\}}{\arg\min} L(\boldsymbol{\theta}) \tag{84}$$

[1] https://en.wikipedia.org/wiki/Conjugate_gradient_method

To solve this rank-constrained minimization problem, we first eliminate the summations by stacking the task-specific matrices vertically. Because scaling the objective by a constant T does not alter the minimization, minimizing $L(\boldsymbol{\theta})$ is strictly equivalent to minimizing a single, consolidated Frobenius norm:

$$\min_{\text{rank}(\boldsymbol{\theta})\leq r} \left\| \widehat{\boldsymbol{Z}}\boldsymbol{\theta} - \widehat{\boldsymbol{Y}} \right\|_F^2 \tag{85}$$

where augmented matrices $\widehat{\boldsymbol{Z}}$ and $\widehat{\boldsymbol{Y}}$ are defined as:

$$\widehat{\boldsymbol{Z}} = \begin{bmatrix} \widehat{\boldsymbol{Z}}_{1,1} \\ \vdots \\ \widehat{\boldsymbol{Z}}_{1,n_1} \\ \vdots \\ \widehat{\boldsymbol{Z}}_{T,1} \\ \vdots \\ \widehat{\boldsymbol{Z}}_{T,n_T} \\ \boldsymbol{\Lambda}_1^{1/2} \\ \vdots \\ \boldsymbol{\Lambda}_T^{1/2} \end{bmatrix} \quad \widehat{\boldsymbol{Y}} = \begin{bmatrix} \boldsymbol{Y}_{1,1} \\ \vdots \\ \boldsymbol{Y}_{1,n_1} \\ \vdots \\ \boldsymbol{Y}_{T,1} \\ \vdots \\ \boldsymbol{Y}_{T,n_T} \\ \boldsymbol{\Lambda}_1^{1/2}\boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\Lambda}_T^{1/2}\boldsymbol{\theta}_T \end{bmatrix}$$

Let $\text{trunc}_r(\cdot)$ denote the truncated Singular Value Decomposition (SVD) operator that retains only the top r singular components of a matrix. The exact closed-form solution for this rank-constrained objective is given by Friedland & Torokhti (2007):

$$\boldsymbol{\theta}_{\text{merge}} = (\widehat{\boldsymbol{Z}}^\top \widehat{\boldsymbol{Z}})^{-1/2} \,\text{trunc}_r\left[(\widehat{\boldsymbol{Z}}^\top \widehat{\boldsymbol{Z}})^{-1/2} \widehat{\boldsymbol{Z}}^\top \widehat{\boldsymbol{Y}}\right] \tag{86}$$

Crucially, we do not need to explicitly materialize the massive stacked matrices $\widehat{\boldsymbol{Z}}$ and $\widehat{\boldsymbol{Y}}$. By computing the inner products of the block matrices, they elegantly collapse back into the $\widehat{\boldsymbol{G}}_t$ and $\widehat{\boldsymbol{C}}_t$ matrices:

$$\widehat{\boldsymbol{G}}_t = \sum_{i=1}^{n_t} \frac{1}{n_t} \widehat{\boldsymbol{Z}}_{t,i}^\top \widehat{\boldsymbol{Z}}_{t,i} + \boldsymbol{\Lambda}_t \tag{87}$$

$$\widehat{\boldsymbol{C}}_t = \sum_{i=1}^{n_t} \frac{1}{n_t} \widehat{\boldsymbol{Z}}_{t,i}^\top \boldsymbol{Y}_{t,i} + \boldsymbol{\Lambda}_t\boldsymbol{\theta}_t \tag{88}$$

Substituting these sums back into the solution, we obtain the final closed-form equation for the rank-constrained VO operator:

$$\boldsymbol{\theta}_{\text{merge}} = \left(\sum_{t=1}^{T} \widehat{\boldsymbol{G}}_t\right)^{-1/2} \text{trunc}_r\left[\left(\sum_{t=1}^{T} \widehat{\boldsymbol{G}}_t\right)^{-1/2} \left(\sum_{t=1}^{T} \widehat{\boldsymbol{C}}_t\right)\right] \square \tag{89}$$

D.2.1 Factorization of the VO operator

After solving our rank constraint solution for VO operator, we need to factorize the optimal merged parameter back to $\widehat{\boldsymbol{W}}_V$ and $\widehat{\boldsymbol{W}}_O$.

$$\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \quad \Longrightarrow \quad \boldsymbol{\theta}_{\text{merge}}^{(r)} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^\top \tag{90}$$

where $\boldsymbol{U}_r, \boldsymbol{V}_r \in \mathbb{R}^{d\times r}$ contain the top r left and right singular vectors, respectively, and $\boldsymbol{\Sigma}_r \in \mathbb{R}^{r\times r}$ is the diagonal matrix of the top r singular values. Unlike QK circuit, there is no scaling factor in OV circuit of standard attention Therefore the factorization follows:

$$\widehat{\boldsymbol{W}}_V = \boldsymbol{U}_r\boldsymbol{\Sigma}_r^{1/2} \in \mathbb{R}^{d\times r} \tag{91}$$

$$\widehat{\boldsymbol{W}}_O = \boldsymbol{V}_r\boldsymbol{\Sigma}_r^{1/2} \in \mathbb{R}^{d\times r} \tag{92}$$

D.3 Optimal MLP operator

For MLP layers, the behavior map is defined as and the regularization term is defined as:

$$\mathcal{A}(\widehat{\boldsymbol{Z}}_{t,i};\boldsymbol{\theta}) = \widehat{\boldsymbol{Z}}_{t,i}\boldsymbol{\theta}, \quad \|\boldsymbol{\theta}-\boldsymbol{\theta}_t\|^2_{\widehat{\chi}_t} = \left\|\boldsymbol{\Lambda}_t^{1/2}(\boldsymbol{\theta}-\boldsymbol{\theta}_t)\right\|_F^2 \tag{93}$$

Unlike the VO circuit, the MLP layer does not have a low-rank bottleneck constraint. Therefore, we can directly solve for the optimal operator $\boldsymbol{\theta}$ without imposing any structural constraints. Substituting these into the unified objective function Eq. (68) and and writing $\mathcal{A}(\boldsymbol{Z}_{t,i};\boldsymbol{\theta}_t) = \boldsymbol{Y}_{t,i}$ as target yields:

$$L(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\left\|\widehat{\boldsymbol{Z}}_{t,i}\boldsymbol{\theta}-\boldsymbol{Y}_{t,i}\right\|_F^2 + \left\|\boldsymbol{\Lambda}_t^{1/2}(\boldsymbol{\theta}-\boldsymbol{\theta}_t)\right\|_F^2\right) \tag{94}$$

The objective reeduced to standard regularized multivariate least squares problem (i.e., ridge regression under a Mahalanobis prior), and the closed-form solution can be derived by setting the gradient to zero. The optimal operator $\boldsymbol{\theta}$ is given by:

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \frac{2}{T}\sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\widehat{\boldsymbol{Z}}_{t,i}^{\top}(\widehat{\boldsymbol{Z}}_{t,i}\boldsymbol{\theta}-\boldsymbol{Y}_{t,i}) + \boldsymbol{\Lambda}_t(\boldsymbol{\theta}-\boldsymbol{\theta}_t)\right) = 0 \tag{95}$$

Rearranging the terms, we obtain:

$$\left(\sum_{t=1}^{T}\frac{1}{n_t}\sum_{i=1}^{n_t}\widehat{\boldsymbol{Z}}_{t,i}^{\top}\widehat{\boldsymbol{Z}}_{t,i} + \boldsymbol{\Lambda}_t\right)\boldsymbol{\theta} = \sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\widehat{\boldsymbol{Z}}_{t,i}^{\top}\boldsymbol{Y}_{t,i} + \boldsymbol{\Lambda}_t\boldsymbol{\theta}_t\right) \tag{96}$$

Let us define Gram matrix $\widehat{\boldsymbol{G}}_t$ and cross-covariance matrix $\widehat{\boldsymbol{C}}_t$ as:

$$\widehat{\boldsymbol{G}}_t = \frac{1}{n_t}\sum_{i=1}^{n_t}\widehat{\boldsymbol{Z}}_{t,i}^{\top}\widehat{\boldsymbol{Z}}_{t,i} + \boldsymbol{\Lambda}_t, \quad \widehat{\boldsymbol{C}}_t = \frac{1}{n_t}\sum_{i=1}^{n_t}\widehat{\boldsymbol{Z}}_{t,i}^{\top}\boldsymbol{Y}_{t,i} + \boldsymbol{\Lambda}_t\boldsymbol{\theta}_t \tag{97}$$

The closed-form solution for the optimal MLP operator is then given by:

$$\boldsymbol{\theta} = \left(\sum_{t=1}^{T}\widehat{\boldsymbol{G}}_t\right)^{-1}\left(\sum_{t=1}^{T}\widehat{\boldsymbol{C}}_t\right)\square \tag{98}$$

D.4 Optimal LayerNorm Operator

For the Layer Normalization module, the operator consists of a scale vector $\boldsymbol{\gamma} \in \mathbb{R}^d$ and a bias vector $\boldsymbol{\beta} \in \mathbb{R}^d$. Evaluated on a single calibration example i from task t, the behavior map and the data-dependent regularization term are defined as:

$$\mathcal{A}(\widehat{\boldsymbol{X}}_{t,i};\boldsymbol{\gamma},\boldsymbol{\beta}) = \nu(\widehat{\boldsymbol{X}}_{t,i})\,\mathrm{diag}(\boldsymbol{\gamma}) + \mathbf{1}\boldsymbol{\beta}^{\top} \tag{99}$$

$$\|\boldsymbol{\theta}-\boldsymbol{\theta}_t\|^2_{\widehat{\chi}_t} = \left\|\boldsymbol{\Lambda}_{\gamma,t}^{1/2}(\boldsymbol{\gamma}-\boldsymbol{\gamma}_t)\right\|_2^2 + \left\|\boldsymbol{\Lambda}_{\beta,t}^{1/2}(\boldsymbol{\beta}-\boldsymbol{\beta}_t)\right\|_2^2 \tag{100}$$

where $\nu(\cdot)$ denotes the normalization that zeroes the mean and scales to unit variance across the feature dimension, and $\mathbf{1} \in \mathbb{R}^L$ is a vector of ones. The diagonal precision matrices $\boldsymbol{\Lambda}_{\gamma,t}$ and $\boldsymbol{\Lambda}_{\beta,t}$ govern the task-dependent geometry for the scale and bias, respectively.

Because the affine transformation acts independently on each feature dimension $j \in \{1,\ldots,d\}$, the Frobenius norm cleanly decouples. We can therefore break the global objective into d independent optimization problems.

Let $\hat{\boldsymbol{x}}_{t,i} \in \mathbb{R}^L$ denote the j-th column of the normalized input $\nu(\widehat{\boldsymbol{X}}_{t,i})$, and let $\boldsymbol{y}_{t,i} \in \mathbb{R}^L$ denote the j-th column of the target output $\boldsymbol{Y}_{t,i}$. Correspondingly, let the scalars $\gamma, \beta \in \mathbb{R}$ denote the j-th elements of the vectors $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, and let $\gamma_t, \beta_t \in \mathbb{R}$ denote the j-th elements of the task-specific parameters $\boldsymbol{\gamma}_t$ and $\boldsymbol{\beta}_t$. Furthermore, let the scalars $\lambda_{\gamma,t}, \lambda_{\beta,t} \in \mathbb{R}$ represent the j-th diagonal entries of the precision matrices $\boldsymbol{\Lambda}_{\gamma,t}$ and $\boldsymbol{\Lambda}_{\beta,t}$. Dropping the feature dimension index j for notational brevity, the objective for a single feature dimension over the empirical calibration mixture is:

$$L(\gamma,\beta) = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\|\hat{\boldsymbol{x}}_{t,i}\gamma + \mathbf{1}\beta - \boldsymbol{y}_{t,i}\|_2^2 + \lambda_{\gamma,t}(\gamma-\gamma_t)^2 + \lambda_{\beta,t}(\beta-\beta_t)^2\right) \tag{101}$$

Because this dimension-wise objective is strictly convex, we find the global minimum by setting the partial derivatives with respect to γ and β to zero.

For the scale parameter γ, we obtain:

$$\frac{\partial L}{\partial \gamma} = 0 \implies \frac{2}{T}\sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\hat{\boldsymbol{x}}_{t,i}^{\top}(\hat{\boldsymbol{x}}_{t,i}\gamma + \mathbf{1}\beta - \boldsymbol{y}_{t,i}) + \lambda_{\gamma,t}(\gamma - \gamma_t)\right) = 0 \tag{102}$$

Rearranging terms, we get:

$$\sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\hat{\boldsymbol{x}}_{t,i}^{\top}\hat{\boldsymbol{x}}_{t,i} + \lambda_{\gamma,t}\right)\gamma + \sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\hat{\boldsymbol{x}}_{t,i}^{\top}\mathbf{1}\right)\beta = \sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\hat{\boldsymbol{x}}_{t,i}^{\top}\boldsymbol{y}_{t,i} + \lambda_{\gamma,t}\gamma_t\right) \tag{103}$$

For the bias parameter β, noting that $\mathbf{1}^{\top}\mathbf{1} = L$ (the sequence length), we obtain:

$$\begin{aligned}\frac{\partial L}{\partial \beta} = 0 &\implies \frac{2}{T}\sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\mathbf{1}^{\top}(\hat{\boldsymbol{x}}_{t,i}\gamma + \mathbf{1}\beta - \boldsymbol{y}_{t,i}) + \lambda_{\beta,t}(\beta - \beta_t)\right) = 0 \\ &\implies \sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\mathbf{1}^{\top}\hat{\boldsymbol{x}}_{t,i}\right)\gamma + \sum_{t=1}^{T}\left(L + \lambda_{\beta,t}\right)\beta = \sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\mathbf{1}^{\top}\boldsymbol{y}_{t,i} + \lambda_{\beta,t}\beta_t\right)\end{aligned} \tag{104}$$

Eq. (103) and (104) form a coupled system of two linear equations. To express the closed-form solution elegantly, we define the following scalar accumulators:

$$\begin{aligned} S_{xx} &= \sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\hat{\boldsymbol{x}}_{t,i}^{\top}\hat{\boldsymbol{x}}_{t,i} + \lambda_{\gamma,t}\right), & S_{xy} &= \sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\hat{\boldsymbol{x}}_{t,i}^{\top}\boldsymbol{y}_{t,i} + \lambda_{\gamma,t}\gamma_t\right) \\ S_1 &= \sum_{t=1}^{T}\left(L + \lambda_{\beta,t}\right), & S_y &= \sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\mathbf{1}^{\top}\boldsymbol{y}_{t,i} + \lambda_{\beta,t}\beta_t\right) \\ S_x &= \sum_{t=1}^{T}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\mathbf{1}^{\top}\hat{\boldsymbol{x}}_{t,i}\right) \end{aligned}$$

This coupled system cleanly reduces to a symmetric 2×2 matrix equation:

$$\begin{bmatrix} S_{xx} & S_x \\ S_x & S_1 \end{bmatrix}\begin{bmatrix}\gamma \\ \beta\end{bmatrix} = \begin{bmatrix} S_{xy} \\ S_y \end{bmatrix} \tag{105}$$

By inverting this 2×2 matrix (e.g., via Cramer's rule), we obtain the exact closed-form solution for the optimal merged scale and bias parameters for this feature dimension:

$$\gamma = \frac{S_1 S_{xy} - S_x S_y}{S_{xx} S_1 - S_x^2} \quad \beta = \frac{S_{xx} S_y - S_x S_{xy}}{S_{xx} S_1 - S_x^2} \tag{106}$$

We can perform the same operation for each dimension of vectors, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, in parallel to obtain the final solution.

D.5 Rank-controlled factorization of merged attention operators

This section details the factorization step summarized in §3.5.

Low-rank structure of attention heads. Let d be the residual-stream dimension and d_k the head dimension. For a single head, $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d \times d_k}$ and $\boldsymbol{W}_O \in \mathbb{R}^{d_k \times d}$, so the local operators

$$\boldsymbol{\theta}_{QK} = \boldsymbol{W}_Q \boldsymbol{W}_K^{\top}/\sqrt{d_k}, \qquad \boldsymbol{\theta}_{VO} = \boldsymbol{W}_V \boldsymbol{W}_O \tag{107}$$

are $d \times d$ matrices of rank at most d_k: each head implements a $d \times d$ operator through a width-d_k bottleneck. SLOA merges these operators in the ambient space $\Theta_{\text{amb}} = \mathbb{R}^{d \times d}$, where no rank constraint is imposed, and restores a factorized form only afterwards. All steps below are applied independently to every head, and separately to its QK and VO operators.

Truncated SVD. Given a merged operator $\bar{\boldsymbol{\theta}} \in \Theta_{\text{amb}}$ with SVD $\bar{\boldsymbol{\theta}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, let $\boldsymbol{U}_r$, $\boldsymbol{\Sigma}_r$, and $\boldsymbol{V}_r$ retain the top r singular directions. By the Eckart–Young theorem, $\bar{\boldsymbol{\theta}}_r = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^\top$ is the best rank-r approximation of $\bar{\boldsymbol{\theta}}$ in Frobenius norm.

QK factorization. Let τ denote the attention scaling used after factorization: $\tau = \sqrt{d_k}$ if the default implementation is kept, or $\tau = \sqrt{r}$ if the expanded head uses the standard scaling for width r. Setting

$$\boldsymbol{W}_Q = \boldsymbol{U}_r(\tau\boldsymbol{\Sigma}_r)^{1/2}, \qquad \boldsymbol{W}_K = \boldsymbol{V}_r(\tau\boldsymbol{\Sigma}_r)^{1/2} \tag{108}$$

gives $\boldsymbol{W}_Q\boldsymbol{W}_K^\top/\tau = \bar{\boldsymbol{\theta}}_{QK,r}$ for either choice of τ.

VO factorization. The VO behavior map $\boldsymbol{A}\boldsymbol{X}\boldsymbol{\theta}_{VO}$ has no scaling, so we set

$$\boldsymbol{W}_V = \boldsymbol{U}_r\boldsymbol{\Sigma}_r^{1/2}, \qquad \boldsymbol{W}_O = \boldsymbol{\Sigma}_r^{1/2}\boldsymbol{V}_r^\top, \tag{109}$$

which gives $\boldsymbol{W}_V\boldsymbol{W}_O = \bar{\boldsymbol{\theta}}_{VO,r}$. In both cases the head width becomes r while d is unchanged. The QK and VO ranks may differ (r_{qk}, r_{ov}; Fig. 7); unless stated otherwise, $r = r_{\text{qk}} = r_{\text{ov}}$.

Why the merged operator may need rank above d_k. The rank r controls how much of the merged operator's spectrum is retained; $r = d_k$ preserves the original head width. Different task models, however, may use different low-dimensional subspaces, and a combination of low-rank operators need not share their rank: the sum of two rank-one operators with different singular directions has rank two, and in general $\operatorname{rank}\big(\sum_t \boldsymbol{\theta}_t\big)$ can reach $\min(d, Td_k)$. Truncating the merged operator back to d_k can therefore discard directions that are useful for multi-task behavior; $r > d_k$ retains them.

Does rank expansion overfit? Increasing r beyond d_k adds capacity without gradient-based training, which might be expected to overfit the calibration set. It does not, however, introduce free parameters: the factorization only retains singular directions already present in the aligned operator $\bar{\boldsymbol{\theta}}$, which is anchored to the task-specific operators by the regularizer in Eq. (5). Consistent with this, rank expansion did not reduce average accuracy in any of our settings and typically improved it (§4, §F.3). The cost is a wider head at inference, quantified in §G.3.

E Experimental details

E.1 Handiling bias terms

Many prior works treat weight matrices and bias terms differently during the merging operation, opting to simply average the bias terms across models. In contrast, we handle bias terms correctly with respect to our objective via an augmentation trick that applies uniformly to all linear layers, including MLPs and attention projections. For instance with weight matrix $\boldsymbol{W} \in \mathbb{R}^{d\times d_{\text{out}}}$ and bias $\boldsymbol{b} \in \mathbb{R}^{d_{\text{out}}}$, the standard forward pass for a batch of L sequence length is:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W} + \boldsymbol{1}_L\boldsymbol{b}^\top, \qquad \boldsymbol{X} \in \mathbb{R}^{l\times d}, \quad \boldsymbol{Y} \in \mathbb{R}^{L\times d_{\text{out}}}. \tag{110}$$

We rewrite this compactly by augmenting the input and weight matrix:

$$\boldsymbol{Y} = \tilde{\boldsymbol{X}}\,\tilde{\boldsymbol{W}}, \qquad \tilde{\boldsymbol{X}} = \big[\boldsymbol{X} \mid \boldsymbol{1}_n\big] \in \mathbb{R}^{l\times(d+1)}, \qquad \tilde{\boldsymbol{W}} = \begin{bmatrix}\boldsymbol{W}\\ \boldsymbol{b}^\top\end{bmatrix} \in \mathbb{R}^{(d+1)\times d_{\text{out}}}. \tag{111}$$

Our merging procedure operates directly on the augmented matrices $\{\tilde{\boldsymbol{W}}_t\}_{t=1}^T$, yielding $\tilde{\boldsymbol{W}}_{\text{merge}} \in \mathbb{R}^{(d+1)\times d_{\text{out}}}$. The merged weight matrix and bias are then recovered by partitioning. The same augmentation applies to the QK and VO operators, where the bias terms are absorbed into the augmented weight matrices.

E.2 Dataset description

E.2.1 Dataset for vision task.

In our vision model merging experiments, we utilize the datasets and task-specific fine-tuned checkpoints provided by the FusionBench framework (Tang et al., 2024). We employ the Vision Transformer (ViT) variant of CLIP (Radford et al., 2021b) as our base architecture. Rather than appending

Table 5: Data statistics of the 20 image classification datasets used for model merging, grouped by task set: the first 8 form 8-tasks the next 6 extend it to 14-tasks, and the final 6 extend it to 20-tasks.

Task	Data Source	# Classes	Train	Test	Calib.	Calib. (%)
		8-Tasks				
SUN397	tanganke/sun397	397	76,128	10,875	256	0.34
GTSRB	tanganke/gtsrb	43	39,209	12,630	256	0.65
Cars	tanganke/stanford_cars	196	8,144	8,041	256	3.14
EuroSAT	tanganke/eurosat	10	21,600	5,400	256	1.19
RESISC45	tanganke/resisc45	45	25,200	6,300	256	1.02
DTD	tanganke/dtd	47	3,760	1,880	256	6.81
MNIST	ylecun/mnist	10	60,000	10,000	256	0.43
SVHN	ufldl-stanford/svhn	10	73,257	26,032	256	0.35
		14 Tasks (+ 6 datasets)				
Flowers	dpdl-benchmark/oxford_flowers102	102	1,020	6,149	256	25.10
PCAM	1aurent/PatchCamelyon	2	262,144	32,768	256	0.10
FER2013	clip-benchmark/wds_fer2013	7	28,709	3,589	256	0.89
Pet	timm/oxford-iiit-pet	37	3,680	3,669	256	6.96
STL10	tanganke/stl10	10	5,000	8,000	256	5.12
CIFAR100	tanganke/cifar100	100	50,000	10,000	256	0.51
		20 Tasks (+ 6 datasets)				
CIFAR10	tanganke/cifar10	10	50,000	10,000	256	0.51
Food101	ethz/food101	101	75,750	25,250	256	0.34
FashionMNIST	zalando-datasets/fashion_mnist	10	60,000	10,000	256	0.43
EMNIST	tanganke/emnist_letters	37	88,800	14,800	256	0.29
KMNIST	tanganke/kmnist	10	60,000	10,000	256	0.43
RenderedSST2	nateraw/rendered-sst2	2	6,920	1,821	256	3.70

a randomly initialized linear classification head during fine-tuning, these models retain CLIP's native dual-encoder structure. Specifically, classification is formulated as a multi-modal contrastive task: the model computes the cosine similarity between the visual embedding of the input image and the textual embeddings generated by passing class-specific text prompts (e.g., "a photo of a $\{class\}$") through the frozen text encoder. To maintain strict consistency with this paradigm, our evaluation reports the standard zero-shot classification accuracy utilizing these text-driven prototypes. The datasets used for model merging are listed in Table 5. We also report the calibration set size and the percentage of the training set used for calibration. The calibration set size is fixed at 256 samples across all datasets, which represents a small fraction of the training data.

E.2.2 Dataset for NLU task.

For language model experiments, we use model checkpoints from the DARE (Yu et al., 2024) work. We evaluate the performance of these models on the 7 GLUE benchmark (Wang et al., 2019),[2] , which covers a range of NLU tasks, including grammatical acceptability (CoLA), natural language classification (MNLI, RTE, QNLI), semantic equivalence (MRPC, QQP), and sentiment analysis (SST-2). The dataset statistics and evaluation metrics for these tasks are summarized in Table 6. We use standard training and validation splits for each task, and we set aside a small calibration set of 256 examples from the training set to perform model merging. The percentage of the calibration set relative to the training set is also reported in the table.

E.2.3 Dataset for LLM experiments

For the LLM merging experiments, we did not use an off-the-shelf pool of LoRA experts for LLaMA-2-7B (Touvron et al., 2023); instead, we constructed our own task pool. We train three LoRA experts (Hu et al., 2022) independently from the LLaMA-2-7B base checkpoint, each targeting all linear projections at rank 32(α=64), for 3 epochs with a cosine learning-rate schedule (peak learning rate 2×10^{-4}) and an effective batch size of 32: a *math* expert on MetaMathQA (Yu

[2] https://huggingface.co/datasets/nyu-mll/glue

Table 6: Summary of GLUE tasks, Metric used, and dataset statistics

Task	# Labels	Primary Metric	Train	Val.	Calib.	Calib. (%)
CoLA	2	Matthews correlation	8,551	1,043	256	2.99
MNLI	3	Accuracy	392,702	9,815	256	0.07
MRPC	2	Avg. F1 & Acc.	3,668	408	256	6.98
QNLI	2	Accuracy	104,743	5,463	256	0.24
QQP	2	Avg. F1 & Acc.	363,849	40,430	256	0.07
RTE	2	Accuracy	2,490	277	256	10.28
SST-2	2	Accuracy	67,349	872	256	0.38

Table 7: Summary of the LLM fine-tuning corpora, evaluation benchmarks, and dataset statistics. Each row is one LoRA expert (rank 32, all-linear modules) fine-tuned from LLaMA-2-7B.

Task	Train Corpus	Benchmark	Primary Metric	Train	Eval	Calib.	Calib. (%)
Math	MetaMathQA	GSM8K	Exact match (flexible extract)	395,000	1,319	256	0.065
Code	Magicoder-Evol-Instruct-110K	HumanEval	pass@1	111,183	164	256	0.230
Instruction-following	Tulu-3 Persona-IF	IFEval	Prompt-level strict accuracy	29,980	541	256	0.854

et al., 2023), a *code* expert on Magicoder-Evol-Instruct-110K (Wei et al., 2024), and an *instruction-following* expert on the persona-conditioned instruction-following split of Tulu-3-SFT (Lambert et al., 2025). Each expert is trained with a single, fixed prompt template applied to every example (the code expert uses Magicoder's own instruction/response wrapper; the math and instruction-following experts share an Alpaca-style instruction/response template) see 4, and the same template is reused uniformly at both calibration and evaluation time. We evaluate each task on its corresponding standard benchmark rather than a held-out split of its own training corpus: GSM8K (Cobbe et al., 2021) for math (exact-match accuracy under flexible answer extraction), HumanEval (Chen et al., 2021) for code (pass@1), and IFEval (Zhou et al., 2023) for instruction-following (prompt-level strict accuracy).

All three are evaluated zero-shot through `lm-evaluation-harness` [3], with each task's training template baked into its eval prompt. Dataset statistics, calibration-set sizes, and calibration coverage (a fixed 256 examples per task, drawn from the same training split and formatted with the same template used for fine-tuning) are summarized in Table 7.

F Additional experiments and ablation analysis of SLOA

We study how SLOA scales with calibration data, rank expansion, and attention-circuit capacity. We also investigate the behavior of SLOA on LoRA finetuned models.

F.1 Scaling to a larger vision backbone: ViT-B/16

To test whether SLOA's advantage persists as model capacity grows, we repeat the ViT-B/32 multi-task merging experiments (§4.1) on the larger ViT-B/16 backbone across the same 8-, 14-, and 20-task benchmarks (Table 8); as a complementary scaling direction, we also evaluate SLOA on a much larger, 7B-parameter decoder-only LLM in the main text (§4.3, Table 4). The trend mirrors ViT-B/32: SLOA's margin over the best baseline widens as the number of merged tasks grows. On 8 tasks, the data-free baseline Iso-CTS is narrowly the strongest method overall (89.01 vs. 88.04 for SLOA ($r = 128$)); by 14 tasks, SLOA (85.16) has overtaken it (84.54); and by 20 tasks, SLOA leads by nearly 5 points (83.43 vs. 78.50). As on ViT-B/32, increasing r from 64 to 128 improves performance at every task count, with the gain growing alongside the number of merged tasks. Ortiz-Jimenez et al. (2023) show that larger, better-pretrained vision backbones tend to be more weight-disentangled than smaller ones, so task-specific fine-tuning directions interfere less under simple weight-space combination. This may explain why data-free baselines remain competitive with SLOA on ViT-B/16 when only a few tasks are merged. However, as the number of merged

[3] https://github.com/eleutherai/lm-evaluation-harness

```
### Instruction:
{instruction}

### Response:
```

(a) Alpaca-style template (math, instruction-following).

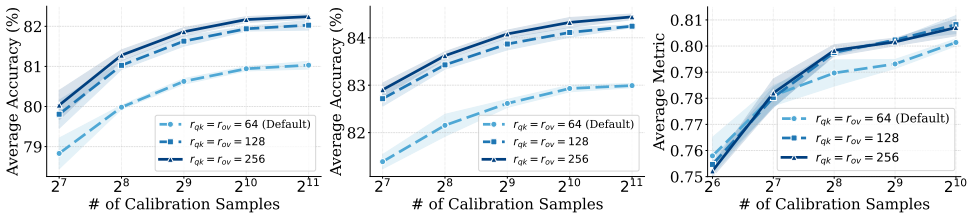
```
You are an exceptionally
intelligent coding
assistant that consistently
delivers accurate and
reliable responses to user
instructions.

@@ Instruction
{instruction}

@@ Response
```

(b) Magicoder template (code).

Figure 4: Prompt templates applied uniformly at training, calibration, and evaluation time for each LoRA expert. The math and instruction-following experts share the Alpaca-style template (a); the code expert uses Magicoder's own template (b).



(a) ViT-B/32 (b) ViT-B/16 (c) RoBERTa

Figure 5: Average performance of SLOA as the number of calibration data increases for (a) ViT-B/32 and (b) ViT-B/16 on 20 vision tasks and (c) RoBERTa on 7 NLU tasks.

tasks grows and interference increases, SLOA, which aligns functional operators rather than raw weights, wins by an increasingly clear margin.

F.2 Merging LoRA-adapted ViT experts

SLOA extends naturally to LoRA-adapted experts: there is no additional difficulty caused by the LoRA parameterization itself. For a LoRA-adapted task t-weight $W_t = W_0 + \Delta W_t$, with $\Delta W_t = B_t A_t$, we first form the effective fine-tuned weight W_t and then construct the same functional QK, VO, linear, and normalization operators used for fully fine-tuned experts; no change to the SLOA local objective is required. This distinction is important because LoRA's factors are themselves non-identifiable: $B_t A_t = (B_t R)(R^{-1} A_t)$ for any invertible R, so methods that operate separately on A_t and B_t can depend on an arbitrary low-rank coordinate system. SLOA does not merge the individual LoRA factors; it merges the effective functional operator and is therefore invariant to this factorization ambiguity. We evaluate this on ViT-B/32 LoRA experts (KnOTS checkpoints) (Stoica et al., 2025) across the same 8-task vision benchmark. Table 9 shows that SLOA achieves the highest normalized accuracy among all methods: SLOA ($r = 128$) reaches 88.07%, 7.0 points above the strongest baseline. Notably, even SLOA ($r = 64$) already exceeds all baselines at 82.66%, despite each task-specific LoRA update being rank 16 – reinforcing the view that the subspace spanned by the merged operator can exceed the rank of any individual task expert.

F.3 Scaling with calibration samples and rank

Figure 5 reports average accuracy as a function of calibration samples for vision (ViT-B/32 and ViT-B/16, 20 tasks) and language (RoBERTa, 7 tasks). Across both modalities and all rank configurations, SLOA improves steadily as calibration data increase. This is consistent with our unified objective in Eq. (5): larger calibration sets provide a better-conditioned system for estimating the merged weights. Notably, even with only 128 samples, our SLOA outperforms all baselines (see Tables 2 -16), demonstrating strong data efficiency. Increasing the rank from the default $r = r_{\mathrm{qk}} = r_{\mathrm{ov}} = 64$ to 128 and 256 further improves performance. On vision tasks, higher rank consistently helps across

Table 8: Performance comparison of model merging methods on ViT-B/16 across 8, 14, and 20 vision classification tasks. For SLOA, r denotes the retained factorization rank used for both QK and VO local operators. The setting $r = 64$ matches the original ViT-B/16 attention-head dimension.

	Method	# Samples	Avg. Accuracy ↑		
			8 Tasks	14 Tasks	20 Tasks
	Fine-tuned	–	92.34	91.31	91.61
Data-Free	Weight Averaging	–	72.28	69.72	64.81
Data-Free	Task Arithmetic	–	77.17	70.92	64.23
Data-Free	TIES-Merging	–	80.21	71.48	66.31
Data-Free	TSV-M	–	86.70	82.09	78.31
Data-Free	Iso-C	–	88.36	82.39	75.82
Data-Free	Iso-CTS	–	**89.01**	84.54	78.50
Training-Free	Fisher Merging	256	73.67	71.00	66.50
Training-Free	RegMean	256	86.13	81.43	75.53
Training-Free	RegMean++	256	87.25	82.23	76.64
Training-Free	SLOA (r=64)	256	87.16	84.19	82.12
Training-Free	SLOA (r=128)	256	88.04	**85.16**	**83.43**

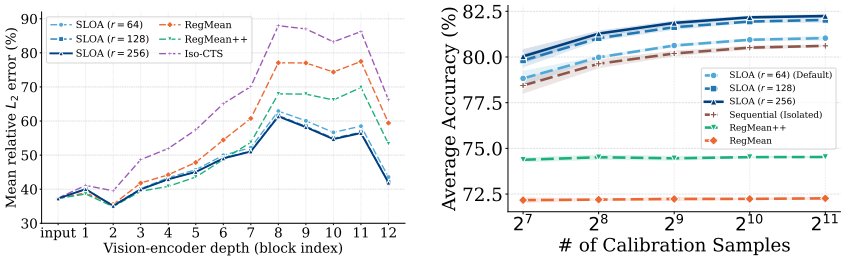
Table 9: Normalized per-task accuracy of model merging methods on ViT-B/32 LoRA (KnOTS, r=16 (Stoica et al., 2025)) experts across 8 vision tasks. Normalized accuracy is the mean over tasks of (merged/fine-tuned) accuracy $\times$ 100; the fine-tuned LoRA oracle's raw average accuracy is 84.28%.

	Method	# Samples	Norm. Acc. (%) ↑
	Fine-tuned (LoRA, oracle)	–	100.00
Data-Free	Simple Avg	–	63.76
Data-Free	TA	–	63.70
Data-Free	TIES	–	63.70
Data-Free	DARE-TIES	–	63.70
Data-Free	KnOTS-TIES	–	68.00
Data-Free	KnOTS-DARE-TIES	–	63.90
Data-Free	Iso-C	–	73.60
Data-Free	Iso-CTS	–	73.70
Training-Free	Fisher Merging	256	62.81
Training-Free	RegMean	256	72.72
Training-Free	RegMean++	256	81.11
Training-Free	SLOA (r=64)	256	82.66
Training-Free	SLOA (r=128)	256	**88.07**

sample sizes, reflecting the greater diversity of the 20-task setting. On NLU tasks, the improvement from 64 to 128 is clear, while 256 gives diminishing returns, suggesting that the effective rank required for merging grows with task number and diversity.

F.4 Error accumulation across network depth

To directly assess whether and how merging error compounds across the network's topological ordering of operators, we measure, at each vision-encoder block, the mean relative L_2 error between the merged model's activation and the corresponding task-specific expert's activation, averaged across all 20 tasks. We report this on ViT-B/16 in the 20-task setting, the most challenging regime in our vision benchmarks, comparing SLOA ($r \in \{64, 128, 256\}$) against RegMean, RegMean++, and Iso-CTS (best performing data-free method). Figure 6(a) shows that error compounds with depth for every method, but at sharply different rates. Through the first six blocks, SLOA and RegMean++ track closely; the two only begin to diverge from block 7 onward, once upstream discrepancies have had several blocks to accumulate. SLOA's error also grows more slowly relative to its own input-level error, peaking at 1.65–1.69$\times$ its starting value versus 1.87$\times$ for RegMean++ and



(a) Layer-wise merging error on ViT-B/16. (b) Baseline data scaling on ViT-B/32.

Figure 6: Diagnosing the source of SLOA's gains : (a) error accumulation across network depth on ViT-B/16, and (b) merging accuracy versus the number of calibration samples on ViT-B/32, comparing SLOA at increasing rank ($r = 64, 128, 256$) against sequential correction in isolation (no operator composition), RegMean, and RegMean++.

2.08× for RegMean. This depth-dependent divergence, rather than a uniform improvement at every block, localizes SLOA's advantage to the later stages of the network, where uncorrected merging error would otherwise compound through the topological ordering of blocks. Expanding the rank from $r = 64$ to $r = 256$ narrows the error gap further.

F.5 Decoupling the QK and OV circuits

To test whether the QK and OV circuits should always use the same rank, we vary r_{qk} and r_{ov} independently on ViT-B/32 with 20 tasks and 1024 calibration samples, while fixing the other rank at 64. Figure 7 shows three main trends. First, expanding either circuit alone improves performance only modestly and saturates after roughly $r = 256$. Second, QK expansion is more effective than OV expansion, indicating that query-key interactions are a stronger bottleneck for task-specific representation. Third, jointly expanding both circuits gives the best and most sustained gains. Thus, we conclude that the two circuits contribute complementary information, and therefore in practice, scaling r_{qk} and r_{ov} together is the most effective design.

F.6 Sensitivity to regularization strength

The regularization term in Eq. (5) is controlled by α and stabilizes the solution when calibration data is limited. Figure 9 evaluates this effect on ViT-B/32 with 8 tasks. With few samples, especially 64 or 128, larger α improves performance by anchoring the solution toward the task-specific weights. With more samples, 256 or 512, the activation-alignment term is already sufficiently constrained, so strong regularization becomes unnecessary and can reduce performance. The optimal α therefore decreases as the calibration set grows, approaching $\alpha \approx 0.05$ at larger sample sizes. These results suggest that SLOA is robust to α when enough data is available, while the regularizer is most useful in the low-data regime.

F.7 Component ablation.

SLOA extends standard linear-layer merging with three components: sequential operator alignment, the QK operator, and the OV operator (§3.1). Figure 8 compares four configurations on ViT-B/32 with 20 tasks: sequential alignment with isolated weight merging, sequential alignment plus QK only, sequential alignment plus OV only, and the full SLOA. Sequential alignment alone is already strong, confirming that the sequential operator structure is a major source of improvement. However, adding only QK or only OV degrades performance, as the two circuits are tightly coupled inside self-attention; modifying one without the other introduces a functional error that propagates through later layers. In contrast, combining both operators recovers this loss and improves over sequential alignment alone, showing that QK and OV should be merged jointly.

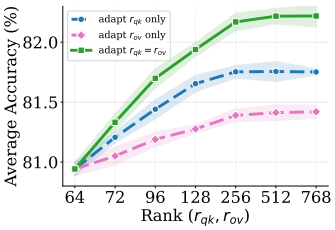


Figure 7: Decoupling QK and OV circuit ranks. Avg. accuracy of SLOA on ViT-B/32 merging 20 vision tasks.

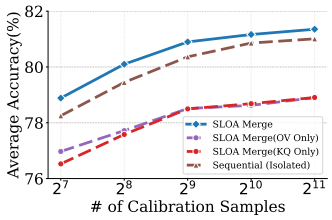


Figure 8: Component ablation of SLOA. Avg. accuracy on ViT-B/32 merging 20 tasks with different calib. samples.

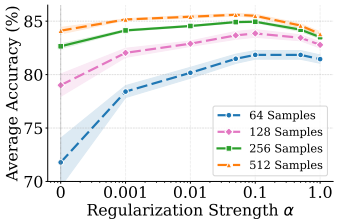


Figure 9: Sensitivity of SLOA to α. Avg. accuracy on ViT-B/32 merging 8 tasks with different regularization strength

Table 10: Wall-clock merging cost and accuracy when merging 20 task-specific ViT-B/16 checkpoints. Calibration-based methods use $N = 256$ samples per task; data-free methods use none.

Method	Data requirement	Merge time (s)	Accuracy (%)
Task Arithmetic	Data-free	14.63	64.23
TIES-Merging	Data-free	39.64	66.31
TSV-M	Data-free	153.34	78.31
Iso-C	Data-free	17.09	75.82
Iso-CTS	Data-free	167.24	78.50
Fisher Merging	Calibration-based ($N = 256$)	112.70	66.50
RegMean	Calibration-based ($N = 256$)	84.11	75.53
RegMean++	Calibration-based ($N = 256$)	115.08	76.64
SLOA (Ours), $r = 64$	Calibration-based ($N = 256$)	175.95	82.12
SLOA (Ours), $r = 128$	Calibration-based ($N = 256$)	182.44	83.43
SLOA (Ours), $r = 256$	Calibration-based ($N = 256$)	186.74	83.74

F.8 Sequential correction versus training-free baselines.

Figure 6(b) examines how these gains behave as the amount of calibration data changes. RegMean and RegMean++ estimate Gram statistics; once these estimates stabilize, additional calibration samples make little further difference, and both curves plateau early. SLOA's cross-state behavior-matching objective, by contrast, continues to benefit from additional samples, since more calibration data improves alignment across a broader range of inputs. The isolated sequential-correction baseline again outperforms RegMean and RegMean++ throughout, consistent with the component ablation above. Composing the QK and OV operators, however, is what allows their retained rank to be increased beyond what the original per-matrix factorization permits, and doing so yields a further, consistent gain over the isolated sequential baseline at every calibration budget. Sequential correction and operator composition are therefore complementary rather than redundant: the former limits the accumulation of merging error across depth, while the latter supplies the additional capacity needed to represent the union of task-specific directions once that error has been corrected.

G Computational Cost and Runtime Analysis

G.1 Merging cost versus baselines

Table 10 compares wall-clock merging time and accuracy across all methods when merging 20 task-specific ViT-B/16 checkpoints, using $N = 256$ calibration samples per task for the calibration-based methods. Being data-free is not synonymous with being computationally inexpensive: for example, TSV-M and Iso-CTS take comparable time due to concatenating weight matrices across different tasks and performing singular value decomposition. SLOA's dominant offline costs are activation propagation through the merged network, solving the local operator objective at each layer, and defactorizing the composed operators back into weight updates. Critically, this cost is incurred once, when constructing the merged checkpoint: we pay it a single time, and the resulting accuracy gain

Table 11: Accuracy and merge time versus the number of merged tasks, at $N = 256$ calibration samples per task and $r = 128$.

	ViT-B/32		ViT-B/16	
# Tasks	Accuracy (%)	Merge time (s)	Accuracy (%)	Merge time (s)
8	85.50 ± 0.05	74.2 ± 0.5	88.04 ± 0.24	107.50 ± 1.76
14	82.61 ± 0.04	103.3 ± 3.1	85.16 ± 0.09	136.50 ± 0.49
20	81.03 ± 0.12	130.6 ± 1.3	83.43 ± 0.09	180.49 ± 1.88

Table 12: Accuracy and merge time versus calibration samples per task, at 20 tasks and $r = 128$.

	ViT-B/32		ViT-B/16	
N	Accuracy (%)	Merge time (s)	Accuracy (%)	Merge time (s)
128	79.77 ± 0.38	99.31 ± 2.13	82.67 ± 0.12	124.96 ± 0.92
256	81.03 ± 0.12	130.64 ± 1.26	83.43 ± 0.09	180.49 ± 1.88
512	81.63 ± 0.13	147.59 ± 3.28	83.86 ± 0.08	245.69 ± 0.58
1024	81.92 ± 0.08	177.78 ± 1.89	84.12 ± 0.11	376.18 ± 0.57
2048	82.01 ± 0.07	252.40 ± 2.37	84.24 ± 0.03	635.90 ± 2.24

persists over the model's entire deployment lifetime. After merging, SLOA requires no task identity, routing, test-time calibration, or online adaptation – the merged checkpoint is used exactly like any single fine-tuned model. Increasing the retained rank r increases merging time only modestly (from 175.95s to 186.74s, from $r = 64$ to $r = 256$), since it enlarges the local operators being solved and defactorized rather than introducing new computation; we quantify the resulting effects on memory usage and inference cost separately below.

G.2 Scaling cost with task count and calibration budget

Tables 11 and 12 characterize how merging cost scales along the two axes: the number of tasks being merged, and the calibration budget. As the number of tasks grows from 8 to 20, accuracy decreases on both backbones, reflecting the increasing interference among a larger and more heterogeneous task pool, while merge time grows roughly in line with the number of task-specific checkpoints that must be processed. Accuracy improves monotonically with more calibration data on both backbones. Peak GPU memory usage is reported in Tables 13 and 14. GPU usage is a function of the number of parameters, the amount of data processed for operator alignment, and the number of tasks we need to merge. ViT-B/32 stays under 6GB across every setting tested, while ViT-B/16 grows to 20.2GB at $N = 2048$, still within a single modern GPU budget.

Table 14: Peak GPU memory (GB) versus calib. samples per task, at 8 tasks.

Backbone	$N = 128$	$N = 256$	$N = 512$	$N = 1024$	$N = 2048$
ViT-B/32	1.4	2.1	2.4	3.2	5.5
ViT-B/16	3.8	7.0	8.1	10.9	20.2

Table 13: Peak GPU memory (GB) vs. the number of merged tasks, at $N = 256$ calib. samples.

Backbone	8 tasks	14 tasks	20 tasks
ViT-B/32	2.1	2.5	2.9
ViT-B/16	7.0	8.0	9.0

Table 16: Multi-task performance on seven GLUE tasks with merged RoBERTa models. Training-free methods use 256 calibration samples. SLOA denotes rank-preserving operator alignment, while SLOA ($r = 128$) expands the factorization rank of the merged QK and VO operators.

	Method	# Samples	Accuracy							**Average**
			CoLA	SST2	MRPC	QQP	MNLI	QNLI	RTE	**Avg.** ↑
	Individual	–	0.6018	0.9404	0.8922	0.9141	0.8720	0.9271	0.7906	0.8483
Data-Free	Weight Averaging	–	0.1808	0.8188	0.8116	0.7381	0.4383	0.7106	0.6173	0.6165
Data-Free	Task Arithmetic	–	0.2909	0.8463	0.8188	0.7869	0.5591	0.7584	0.6462	0.6724
Data-Free	TIES-Merging	–	0.1951	0.8429	0.8333	**0.8351**	0.6501	0.7309	0.4260	0.6448
Data-Free	TSV-M	–	0.3477	0.8739	0.8446	0.8286	0.6163	0.8124	0.6606	0.7120
Data-Free	Iso-C	–	0.2639	0.8601	0.8130	0.8211	0.7303	0.7882	0.7040	0.7115
Data-Free	Iso-CTS	–	0.4136	0.7798	0.6385	0.6500	0.5277	0.6590	0.6534	0.6174
Training-Free	Fisher Merging	256	0.1126	0.8719	0.7419	0.7808	0.4227	0.5311	0.5150	0.5680
Training-Free	RegMean	256	0.2797	0.8933	0.8069	0.7915	0.7251	0.8336	0.6161	0.7066
Training-Free	RegMean++	256	0.2433	0.9170	0.7740	0.8099	0.7543	0.8299	0.6691	0.7139
Training-Free	SLOA (r=64)	256	0.5608	0.9232	**0.8814**	0.8152	0.7678	0.8755	0.7100	0.7906
Training-Free	SLOA (r=128)	256	**0.5721**	**0.9239**	0.8810	0.8166	**0.7753**	**0.8824**	**0.7341**	**0.7979**

Table 15: Inference cost by deployed rank on ViT-B/16 (20-task merge). Inference measured with batch size 64, FP16, on a single A100 GPU, averaged over 100 iterations.

Deployed model	QK/VO rank	Accuracy (%)	Throughput (img/s) ↑	Latency (ms/img) ↓	Peak memory (GB) ↓
All baseline merges (shared original architecture)	64 / 64	64.23–78.50	≈3257	0.307	0.60
SLOA (original rank, $r = 64$)	64 / 64	82.37	3261.1	0.307	0.60
SLOA (expanded rank, $r = 128$)	128 / 128	83.53	2711.5	0.369	0.65
SLOA (expanded rank, $r = 256$)	256 / 256	83.74	1999.1	0.500	0.83

G.3 Inference cost

Table 15 isolates the cost of deploying the merged model itself, separate from the one-time merging cost reported above. Since every baseline in Table 10 merges by combining weights within the original 64/64 attention-head architecture, all of them, share an identical inference profile: throughput of approximately 3257 images/s, 0.307 ms/image latency, and 0.60GB peak memory. At the default rank ($r = 64$, no expansion), SLOA's merged model is indistinguishable in inference cost from every baseline, while still improving accuracy to 82.37%. Rank expansion to $r = 128$ or $r = 256$ increases inference time and GPU memory, both direct consequences of the wider attention head, in exchange for further accuracy gains (83.53% and 83.74%, respectively). This is a deliberate, user-controlled trade-off between capacity and test-time cost.

Computational resources. We performed our SLOA merging method and conducted all the ablation studies, including benchmarking on a single NVIDIA DGX A100 with 48 GB memory.

H Task-wise full tables

This section reports the per-task results behind the averages in the main paper. Tables 17, 18, and 19 extend Table 2 with task-wise accuracies for ViT-B/32 merged over 8, 14, and 20 vision tasks, respectively. Table 16 likewise extends Table 3 with the per-task scores for RoBERTa on the seven GLUE tasks. All methods share the setup of the main experiments: calibration-based methods use N=256 samples per task, and SLOA is reported at the original rank (r=64) and with rank expansion (r=128). SLOA's average gains are broad rather than driven by a few tasks. Data-free methods use no calibration data, so for fixed checkpoints and hyperparameters, their merged model, and therefore their accuracy, is the same in every run; we report standard deviations only for calibration-based methods, where the variation comes from the random draw of calibration samples.

Table 17: Per-task accuracy (%) of merged ViT-B/32 models on 8 vision tasks. Training-free methods use $N{=}256$ calibration samples per task; their averages are mean$\pm$std over three seeds. For SLOA, r is the retained rank of the QK and VO operators ($r{=}64$ preserves the original architecture). Bold/underline marks the best/second-best merging method per column. Data-free methods are deterministic given the checkpoints, so standard deviations are reported only for calibration-based methods, computed over three random draws of the calibration set.

Method	SUN397	GTSRB	Cars	EuroSAT	RESISC45	DTD	MNIST	SVHN	Avg.
Fine-tuned (individual)	74.90	98.91	78.47	99.07	95.14	79.79	99.58	97.27	90.39
Simple Avg.	65.39	54.92	62.42	75.67	70.63	50.53	86.28	64.49	66.29
Task Arithmetic	64.30	62.71	61.42	78.30	70.57	51.76	93.01	73.88	69.49
TIES-Merging	61.40	78.68	60.15	69.04	69.62	51.06	97.72	85.92	71.70
TSV-M	68.59	91.27	72.01	93.41	85.14	64.36	98.76	90.79	83.04
Iso-C	67.73	93.51	71.61	90.00	84.98	65.05	98.82	89.39	82.64
Iso-CTS	68.30	92.75	**72.39**	88.81	85.97	67.87	98.76	88.09	82.87
Fisher Merging	65.84	61.71	65.74	79.74	76.54	54.47	85.28	63.67	69.12$\pm$0.13
RegMean	69.03	83.56	70.61	95.64	85.50	65.87	98.41	92.41	82.63$\pm$0.11
RegMean++	**69.34**	90.29	70.75	95.64	86.89	68.60	98.98	94.45	84.37$\pm$0.12
SLOA (r=64)	65.15	95.05	66.92	97.62	89.29	69.80	99.20	96.07	84.89$\pm$0.08
SLOA (r=128)	66.11	**95.70**	68.35	**97.65**	**90.22**	**70.35**	**99.28**	**96.31**	**85.50**$\pm$0.05

Table 18: Per-task accuracy (%) of merged ViT-B/32 models on 14 vision tasks. Setup and notation as in Table 17.

Method	SUN397	GTSRB	Cars	EuroSAT	RESISC45	DTD	MNIST	SVHN	Flowers	PCAM	FER2013	Pets	STL10	CIFAR100	Avg.
Fine-tuned (individual)	74.90	98.91	78.47	99.07	95.14	79.79	99.58	97.27	88.62	87.96	71.62	92.37	97.55	88.38	89.26
Simple Avg.	64.73	45.63	60.40	66.96	67.06	47.13	76.57	50.66	67.25	65.27	51.62	84.25	97.20	70.39	65.37
Task Arithmetic	64.41	50.00	59.63	67.74	67.29	47.77	80.75	53.93	66.17	69.80	53.05	84.25	96.62	69.14	66.47
TIES-Merging	62.10	63.95	54.58	62.93	65.29	50.21	92.56	65.70	58.22	77.10	54.95	81.30	94.84	62.44	67.58
TSV-M	67.30	87.80	63.85	91.04	81.22	61.17	98.35	85.23	68.01	82.03	65.53	87.82	97.18	71.06	79.11
Iso-C	70.02	81.45	63.66	83.85	80.10	60.96	96.36	75.98	70.34	83.29	65.30	87.87	97.08	**74.99**	77.94
Iso-CTS	**70.31**	83.76	**68.42**	85.30	81.54	62.18	97.03	76.04	74.58	83.16	**67.02**	87.87	97.29	74.86	79.24
Fisher Merging	64.51	47.80	62.96	66.09	69.47	48.62	73.94	48.50	69.42	64.12	56.02	87.68	96.96	71.58	66.26$\pm$0.21
RegMean	66.88	72.91	66.17	91.63	79.68	58.94	96.65	81.92	72.92	80.23	63.80	88.67	97.02	73.65	77.93$\pm$0.04
RegMean++	67.54	80.60	67.04	94.38	81.78	60.73	98.03	91.25	75.96	77.41	64.67	**90.08**	**97.47**	73.95	80.06$\pm$0.03
SLOA (r=64)	62.68	88.83	63.19	96.96	85.97	66.03	98.95	94.99	77.62	87.95	64.13	89.47	97.13	73.26	81.94$\pm$0.01
SLOA (r=128)	63.18	**90.49**	64.62	**97.17**	**87.25**	**66.77**	**99.03**	**95.32**	**78.07**	**88.24**	64.84	89.79	97.21	74.49	**82.61**$\pm$0.04

Table 19: Per-task accuracy (%) of merged ViT-B/32 models on 20 vision tasks. Setup and notation as in Table 17.

Method	SUN397	GTSRB	Cars	EuroSAT	RESISC45	DTD	MNIST	SVHN	Flowers	PCAM	FER2013	Pets	STL10	CIFAR100	CIFAR10	Food101	FMNIST	EMNIST	KMNIST	SST2	Avg.
Fine-tuned (individual)	74.90	98.91	78.47	99.07	95.14	79.79	99.58	97.27	88.62	87.96	71.62	92.37	97.55	88.38	97.60	88.40	94.75	95.62	98.23	71.28	89.78
Simple Avg.	64.22	43.14	59.56	60.85	64.76	46.44	71.79	47.22	66.29	63.95	50.14	83.92	96.99	69.80	92.66	80.38	71.31	14.98	11.41	61.78	61.08
Task Arithmetic	62.03	48.95	53.74	58.15	60.87	45.96	79.45	48.43	60.95	73.45	51.32	82.23	94.89	64.64	91.38	71.88	73.86	17.81	12.14	59.86	60.60
TIES-Merging	65.02	49.22	59.77	60.78	66.22	48.51	79.27	52.37	66.55	66.75	51.66	84.06	97.05	70.40	93.33	80.45	72.43	17.32	12.16	61.94	62.76
TSV-M	65.50	83.79	55.80	87.11	75.76	58.62	97.61	80.28	65.00	80.57	64.29	86.29	96.55	68.92	93.89	76.37	84.27	35.91	46.48	70.18	73.66
Iso-C	**67.13**	80.72	47.76	78.19	73.16	57.07	96.82	75.47	63.16	79.79	62.75	83.73	96.05	71.96	94.36	73.38	82.60	33.99	35.11	70.02	71.16
Iso-CTS	66.61	82.52	57.67	80.74	75.13	57.87	97.10	75.94	70.56	83.69	**64.54**	84.96	96.58	**72.95**	94.62	74.06	83.58	38.29	40.48	**70.35**	73.41
Fisher Merging	63.94	43.62	61.63	59.00	65.82	46.49	68.27	45.50	68.09	62.36	52.94	86.39	96.95	70.35	93.07	81.16	70.36	16.08	13.84	68.95	61.74$\pm$0.06
RegMean	65.74	65.87	63.88	88.40	77.14	55.85	91.32	74.54	70.66	79.60	61.76	87.48	96.53	72.14	94.58	82.21	80.06	22.75	44.82	68.63	72.20$\pm$0.03
RegMean++	66.55	73.95	**65.21**	92.90	79.34	57.87	93.65	87.43	73.48	75.69	63.55	**89.26**	**97.42**	72.83	94.92	**83.57**	82.97	29.06	43.70	66.94	74.51$\pm$0.12
SLOA (r=64)	60.23	84.31	59.91	96.26	83.66	63.67	98.01	93.48	75.94	87.21	63.12	88.03	96.95	70.68	94.66	78.30	90.06	72.88	82.88	61.01	80.06$\pm$0.18
SLOA (r=128)	60.85	**86.24**	61.50	**96.54**	**85.05**	**64.11**	**98.24**	**94.01**	**76.45**	**87.92**	63.82	88.65	97.00	71.64	**95.01**	79.50	**90.60**	**76.03**	**85.11**	62.27	**81.03**$\pm$0.12