PALM-MERGE: TASK INTEGRATION AND RETENTION IN DATA-FREE CONTINUAL MODEL MERGING

Anonymous authors
Paper under double-blind review

ABSTRACT

Data-free continual model merging can continually extend deployed foundation models without retaining task data or retraining, but it must integrate new checkpoints while preserving earlier capabilities. Existing methods fold successive task deltas into shared weights and control interference by projection or filtering. However, forcing heterogeneous responses into one map creates an approximation floor and can overwrite directions required by earlier tasks. To address this limitation, we propose PALM-Merge (**P**reserve-and-**A**ctivate Low-rank **M**erging), which stores each task residual as an independent low-rank expert and activates experts sparsely from inputs, separating response preservation from access and avoiding cross-task parameter overwriting. Specifically, truncated SVD converts each task update into a frozen low-rank expert whose compression error is controlled by the discarded singular energy, while routing each activation to the most relevant low-rank expert at inference without access to any training data. Theoretically, we show that a historical prediction can change only when a newly added expert changes the routing decision. Forgetting can then occur only if the resulting perturbation is large enough to overturn the original classification margin. This yields a cumulative accuracy-loss bound for task c of $\min\{1, \sum_{t=c+1}^{T} p_{t,c}\}$. Across nine FusionBench CLIP-ViT settings, PALM-Merge achieves 89.8% average accuracy. On a controlled eight-task stream, it outperforms NUFILT by 4.85% and reduces positive final forgetting by 65.9%, while reaching 82.75% on Flan-T5 GLUE-6. These gains require $2.10\times$ storage but only $1.24\times$ active-tensor cost.

1 INTRODUCTION

Foundation models are increasingly deployed as long-lived systems, so they must acquire new capabilities after their initial release. For example, as show in Fig.1 (a), a vision system may first support scene recognition and later receive checkpoints specialized for satellite imagery or traffic signs. Retraining a joint model is often impossible because the data used to produce earlier checkpoints are no longer available. Therefore, data-free continual model merging offers a practical alternative: it combines a stream of fine-tuned models through their parameters alone (Tang et al., 2025b; Yang et al., 2025). However, simply adding the newest task is not enough. Each capability must be integrated well when it arrives and must remain useful after later updates. Therefore, final performance therefore depends on both *integration* and *retention*.

Classic continual merging methods write each arriving task delta into shared weights and use projection or filtering to reduce interference (Tang et al., 2025b; Yang et al., 2025; Qiu et al., 2026). As illustrated in Fig. 1 (a), this design leaves all task responses in one parameter map. When tasks require different responses along overlapping activation directions, even the best shared linear map can retain a nonzero approximation gap (proof in sec.2). Moreover, every update must decide which shared directions can be changed safely, because an inaccurate decision can weaken the new capability or remove directions still needed by earlier tasks. These limitations motivate a different question: *Can a merger preserve task-specific responses while selecting the right one as tasks accumulate?*

We answer this question with PALM-Merge (**P**reserve-and-**A**ctivate **L**ow-rank **M**erging). The method starts from a frozen pre-trained base and represents every arriving checkpoint by compact residual experts. As illustrated in Fig. 1(b) and (c), a truncated SVD preserves each task update in low-rank form, and the stored factors remain fixed after insertion. During inference, checkpoint-

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.

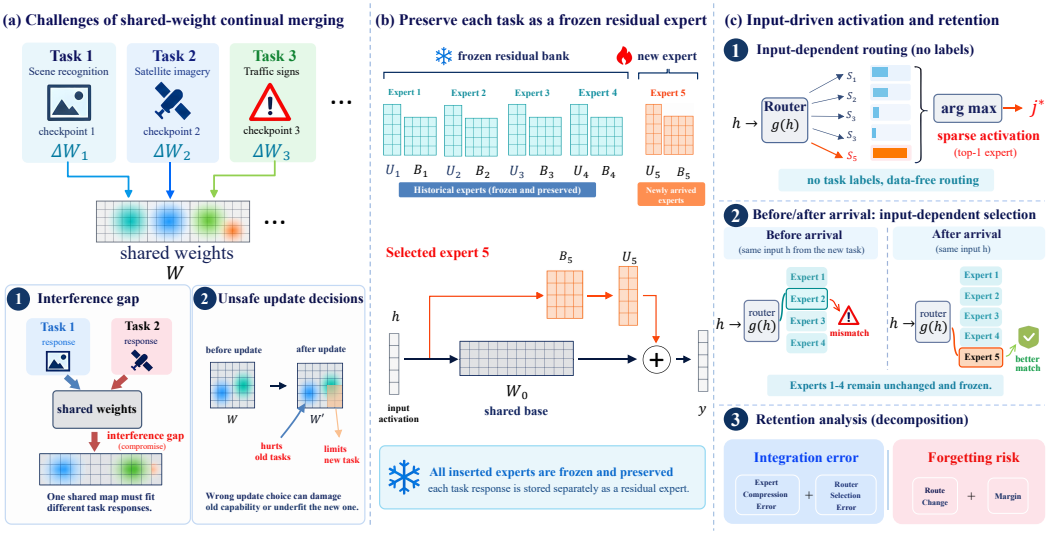

Figure 1: **Challenges and overview of our preserve-and-activate framework. (a)** Shared-weight continual merging can suffer from task interference and unsafe updates, which may compromise both old and new capabilities. **(b)** We preserve each arriving checkpoint as a frozen low-rank residual expert around a shared base and activate only the selected expert. **(c)** The input activation routes to the most relevant expert without task labels; historical experts remain unchanged as new experts arrive, separating integration error from forgetting risk.

derived input subspaces score the current layer activation and select one expert per token and target layer. Thus, PALM-Merge uses the activation itself to access task-specific responses: it needs neither training data nor task labels for routing, and sparse activation avoids evaluating the full expert bank. Together, these choices form a growing mixture that preserves task responses individually while keeping their per-input use selective.

This preserve-and-activate view also provides a clear way to analyze why the method works. As illustrated in Fig. 1(c), for integration, we derive the best local approximation attainable by a shared linear map and separate the error of our expert bank into compression and selection components. The expert rank controls how much of each task update is retained, whereas the projection gate controls which stored response is used. For retention, we analyze how adding a new expert affects earlier tasks. A historical prediction remains unchanged if the new expert does not change its routing decisions. If the routing changes, the prediction is affected only when the resulting response change is large enough relative to its classification margin. This leads to a cumulative forgetting bound that connects later expert arrivals to earlier tasks.

Our experiments support this account at both stages. On a controlled eight-task ViT-B/32 stream, PALM-Merge reaches 88.70% final accuracy, outperforming NUFILT by 4.85%. Saved task trajectories attribute 3.30% of this gap to stronger integration and 1.55% to better subsequent retention. Positive final forgetting also falls from 2.60% to 0.89%, a 65.9% reduction. Across nine CLIP-ViT settings, PALM-Merge averages 89.8% accuracy, and its 82.75% GLUE-6 result extends the evaluation to Flan-T5. These gains use additional storage: at the controlled vision operating point, the model stores $2.10\times$ the base parameters while its active-tensor proxy is $1.24\times$ the base. Accordingly, our contributions are:

- **A preserve-and-activate merger.** PALM-Merge stores task updates as independent, frozen low-rank experts and selects them directly from layer activations, enabling data-free continual integration with sparse residual evaluation.
- **An analysis of integration and retention.** We quantify the local shared-map approximation floor, separate compression from selection error, and connect expert arrival, routing changes, classification margins, and cumulative forgetting.
- **Evidence across tasks and modalities.** We show that the gains arise from both stronger initial integration and better subsequent retention, and we evaluate their accuracy, forgetting, and resource costs across CLIP-ViT and Flan-T5 streams.

2 RETHINKING CONTINUAL MERGING: INTEGRATION AND RETENTION

In this section, we rethink continual merging through the lens of *integration* and *retention*. We first measure the performance when a task arrives, and then examine how this performance changes after new tasks are added. Next, we show that a shared map can limit integration, and explain why retaining separate task responses still requires selective access. These insights motivate our preserve-and-activate design.

Where final performance is gained or lost. Let $a_{t,i}$ denote task i's accuracy after checkpoint t is merged in a stream of $T \geq 2$ tasks. Define $A_{\text{arrival}} = T^{-1}\sum_i a_{i,i}$ and $\text{BWT} = (T-1)^{-1}\sum_{i<T}(a_{T,i} - a_{i,i})$. Then

$$\text{ACC} = A_{\text{arrival}} + \frac{T-1}{T}\text{BWT}, \qquad \Delta\text{ACC} = \Delta A_{\text{arrival}} + \frac{T-1}{T}\Delta\text{BWT}. \tag{1}$$

Here, Δ denotes the difference between two methods on the same stream. Thus, final performance can improve either because tasks are integrated better when they arrive or because they are retained better afterwards. This decomposition shows that performance changes are governed by the algorithm's integration and retention capabilities. (Appendix E).

Why a shared map can limit integration. For task c, consider the desired residual response $\Delta W_c h$. Any input-independent linear combination of stored residuals is itself a single linear map Mh. We therefore study the best possible shared map. Let task probabilities $\pi_c > 0$, $\Sigma_c = \mathbb{E}[hh^\top \mid c]$, $S = \sum_c \pi_c \Sigma_c$, and $H = \sum_c \pi_c \Delta W_c \Sigma_c$. Minimizing $\mathbb{E}\|Mh - \Delta W_c h\|_2^2$ gives

$$\mathcal{J}^\star_{\text{shared}} = \sum_c \pi_c \,\text{Tr}(\Delta W_c \Sigma_c \Delta W_c^\top) - \text{Tr}(HS^\dagger H^\top). \tag{2}$$

Therefore, any fixed linear aggregation of the same residual bank (i.e. $\{\Delta W_1, \Delta W_2, \ldots, \Delta W_T\}$, Fig.1 (b)) has error at least $\mathcal{J}^\star_{\text{shared}}$. For equally weighted tasks with a common activation second moment Σ, this error reduces to $T^{-1}\sum_c \|(\Delta W_c - \overline{\Delta W})\Sigma^{1/2}\|_F^2$, which measures how much the task-specific responses vary around their mean. Thus, when different tasks require different responses along the same activation directions, no single shared map can match them all and a nonzero approximation gap remains. *This motivates preserving separate residual experts for each task rather than relying on a single shared map.* (More proof details are provided in Appendix F).

Preservation supplies capacity; selection determines access. Preserving task responses separately avoids forcing them into a single shared map, but introduces two remaining sources of error: compression and selection. Let $e_c^\star(h) = \Delta W_c h + \Delta b_c$ be the desired response, $\hat{e}_c(h)$ its stored approximation, and $\hat{e}_{g(h)}(h)$ the routed response. Define

$$u_c = \hat{e}_c - e_c^\star, \qquad v_c = \hat{e}_{g(h)} - \hat{e}_c.$$

Taking the expectation over tasks and their corresponding activations, we obtain

$$\mathcal{J}_{\text{bank}} \triangleq \mathbb{E}\|\hat{e}_{g(h)}(h) - e_c^\star(h)\|_2^2 = \underbrace{\mathbb{E}\|u_c\|_2^2}_{\mathcal{C}_r} + \underbrace{\mathbb{E}\|v_c\|_2^2}_{\mathcal{S}_g} + 2\mathbb{E}\langle u_c, v_c\rangle \leq (\sqrt{\mathcal{C}_r} + \sqrt{\mathcal{S}_g})^2. \tag{3}$$

Here, the cross term captures whether compression and selection errors reinforce or offset each other. $\mathcal{C}_r$ measures information lost when storing an expert, while $\mathcal{S}_g$ measures the cost of selecting another response. Thus, preserving provides task-specific capacity, whereas routing determines whether that capacity is correctly accessed. For truncated SVD, the rank controls $\mathcal{C}_r$ through the discarded singular components, while $\mathcal{S}_g$ depends on which expert is selected and how different its response is from the task-matched expert. See Appendix F for the proof details.

Based on the above analysis, once an expert is stored and frozen, its approximation at a fixed activation does not change. However, later tasks can still change routing decisions, which may propagate through the network and alter historical predictions. *This shows that preserving historical experts alone is insufficient for retention; routing must also remain stable as new experts arrive.*

3 PALM-MERGE: PRESERVING RESPONSES FOR SELECTIVE USE

Section 2 reveals two requirements for continual merging: task-specific responses should be preserved separately, and preserved responses must remain selectively accessible from the input.

PALM-Merge realizes these two requirements by storing each arriving task residual as a frozen low-rank expert and using checkpoint-derived input subspaces to route each activation to an appropriate expert. Thus, preservation retains task-specific capacity, while routing determines how that capacity is accessed. The following sections detail how these two components are implemented in the continual setting, together with their computational implications.

Continual setting. Let $f_0(\cdot;\theta_0)$ be a pre-trained model and $f_t(\cdot;\theta_t)$ its checkpoint fine-tuned for task t. All checkpoints share the same architecture and initialization. At step t, PALM-Merge accesses only f_0, the arriving f_t, and the state stored from previous steps; no task data or future checkpoints are available. The goal is to maintain a deployed model F_t for the seen tasks $\mathcal{T}_{1:t}$.

We apply task-specific updates only to a set of target linear layers $\ell \in \mathcal{L}$, leaving all other parameters at their pre-trained values. For a target layer with base parameters $W_\ell^{(0)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and $b_\ell^{(0)}$, checkpoint t defines

$$\Delta W_\ell^{(t)} = W_\ell^{(t)} - W_\ell^{(0)}, \qquad \Delta b_\ell^{(t)} = b_\ell^{(t)} - b_\ell^{(0)}. \tag{4}$$

Preserving task responses with low-rank experts. To preserve each task response separately, we approximate its residual by a low-rank factorization. For rank $r_\ell = \min(r, d_{\text{out}}, d_{\text{in}})$,

$$\Delta W_\ell^{(t)} \approx U_\ell^{(t)} \operatorname{diag}(s_\ell^{(t)})(V_\ell^{(t)})^\top. \tag{5}$$

Exact truncated SVD gives the optimal rank-r_ℓ approximation in Frobenius norm (Eckart & Young, 1936); our implementation uses truncated SVD, with approximation details deferred to Appendix G. Absorbing the singular values into $B_\ell^{(t)} = \operatorname{diag}(s_\ell^{(t)})(V_\ell^{(t)})^\top$, we store the residual expert

$$e_\ell^{(t)}(h) = U_\ell^{(t)} B_\ell^{(t)} h + \Delta b_\ell^{(t)}. \tag{6}$$

Once inserted, $U_\ell^{(t)}$, $B_\ell^{(t)}$, and $\Delta b_\ell^{(t)}$ are never updated. If expert c is selected, the layer output is

$$y_\ell(h \mid c) = W_\ell^{(0)} h + b_\ell^{(0)} + e_\ell^{(c)}(h). \tag{7}$$

Fig. 1 (b) illustrates the process described by the above equations.

Activating experts from the input. Preservation provides task-specific capacity; routing determines which stored response is used. We construct a routing basis from the retained right-singular directions (Tang et al., 2026). For routing rank $q_\ell = \min(q, r_\ell)$,

$$G_\ell^{(j)} = [v_{\ell,1}^{(j)}, \ldots, v_{\ell,q_\ell}^{(j)}]^\top \in \mathbb{R}^{q_\ell \times d_{\text{in}}}. \tag{8}$$

We cache $G_\ell^{(j)}$ separately and include it in the storage accounting. Given activation h, expert $j \le t$ receives the routing score

$$a_\ell^{(j)}(h) = \|G_\ell^{(j)} h\|_2, \tag{9}$$

which measures how strongly h lies in the residual input subspace of expert j. For top-1 routing,

$$j_\ell^\star(h) = \min \arg\max_{j \le t} a_\ell^{(j)}(h),$$

where ties favor the earlier expert. The layer then returns

$$y_\ell(h) = W_\ell^{(0)} h + b_\ell^{(0)} + e_\ell^{(j_\ell^\star(h))}(h). \tag{10}$$

Thus, routing uses only the current activation and checkpoint-derived subspaces: it requires neither task labels nor training data. All stored bases are scored, but only one residual expert is evaluated for each token and target layer. Fig. 1 (c) illustrates the process described by the above equations. We also provide the general top-k formulation in Appendix H.

Storage and computation. Each expert stores $d_{\text{out}} r_\ell + r_\ell d_{\text{in}} + d_{\text{out}}$ residual parameters and $q_\ell d_{\text{in}}$ routing parameters. After T tasks, the additional storage per target layer is

$$O\big(T\,[r_\ell(d_{\text{in}} + d_{\text{out}}) + d_{\text{out}} + q_\ell d_{\text{in}}]\big)\,.$$

For each token and target layer, routing costs $O(T q_\ell d_{\text{in}})$, while top-1 residual evaluation costs only $O(r_\ell(d_{\text{in}} + d_{\text{out}}))$. Thus, the expert bank grows linearly with the task stream, but the cost of evaluating the selected residual remains independent of T; only routing over the bank grows with its size. We report the resulting storage–computation trade-off in section 5.5.

4 RETAINING HISTORICAL PREDICTIONS AS EXPERTS ARRIVE

Section 2 showed that freezing historical experts alone does not ensure retention, because a new expert may still change which stored response is selected. This section shows that such a routing change is the only way a newly added expert can affect a historical prediction. Even then, forgetting occurs only when the induced response change is large enough relative to the original prediction margin. We formalize this process from local routing changes to prediction changes and finally to the BWT term in eq. (1).

A new expert changes a response only if it wins the route. Consider step $t \geq 2$ and a fixed activation h at one routing site. Let

$$j_\star = \min \arg\max_{j<t} a^{(j)}(h), \qquad m_{t-1}(h) = a^{(j_\star)}(h)$$

denote the previous winner and its score. Because ties favor the earliest expert, the newcomer replaces $j_\star$ only when its score is strictly larger. Since the base model and all historical experts are frozen,

$$y^{[t]}(h) - y^{[t-1]}(h) = \mathbf{1}\Big\{a^{(t)}(h) > m_{t-1}(h)\Big\}\left(e^{(t)}(h) - e^{(j_\star)}(h)\right). \tag{11}$$

Eq. (11) gives a simple conclusion: if the newcomer does not win the route, the layer output is exactly unchanged. If it does win, the local change is precisely the difference between the newcomer and the previous expert. This difference may improve or degrade the prediction; its signed effect on residual error is derived in Appendix I.

Stable routing preserves the full historical prediction. If the newcomer does not change any routing decision along the previous forward path, then the entire computation remains unchanged. Conversely, a routing change at an early site may alter downstream activations and therefore affect later routing decisions. We therefore evaluate the newcomer along the previous model's forward path. For input x, let $h_s^{[t-1]}(x)$ denote the activation at routing site s under F_{t-1}, where s indexes target layers and token positions. Define

$$E_t(x) = \left\{\exists s : a_s^{(t)}\left(h_s^{[t-1]}(x)\right) > \max_{j<t} a_s^{(j)}\left(h_s^{[t-1]}(x)\right)\right\}. \tag{12}$$

The event $E_t(x)$ therefore means that the newcomer overtakes an incumbent somewhere along the historical forward path.

Theorem 1 (Prediction preservation)**.** *Under deterministic top-1 routing with a frozen base, historical experts, and historical routing bases, if $E_t(x)$ is false, then*

$$F_t(x) = F_{t-1}(x).$$

The argument is direct: before the first changed route, both models have identical activations. A first difference can occur only if the newcomer strictly beats the previous winner at some routing site. Hence, if this never happens on the old path, the complete forward computation remains unchanged.

In other words, a historical prediction is guaranteed to remain unchanged as long as the new expert never overtakes the previous winner along the old forward path. *A new expert cannot change a historical prediction unless it first changes the routing path.*

When can the route change? At a given target layer and token position, newcomer t replaces the previously selected expert j exactly when

$$h^\top (P_t - P_j)h > 0, \qquad P_i = G_i^\top G_i.$$

Thus, routing stability depends jointly on the geometry of the two routing subspaces and the location of the historical activation h. In particular, similar subspaces and activations concentrated around the incumbent subspace make a switch less likely. Detailed geometric certificates are given in Appendix I.

Routing changes affect forgetting through response size and margin. A routing change alone does not imply forgetting. The prediction changes only if the resulting logit perturbation is large enough to overcome the previous classification margin. For historical classification task c, let $A_{t,c}$

denote its accuracy after step t, and let $C_{t-1,c}$ be the event that F_{t-1} predicts x correctly. Let $M_{t-1,c}(x)$ denote the corresponding true-class logit margin.

Appendix I constructs a post-arrival bound $B_{t,c}(x)$ on the induced logit change by propagating residual response differences through the affected network branches. A previously correct prediction can change only if the input is exposed to a routing change and its margin is small enough relative to this perturbation. Consequently,

$$[A_{t-1,c} - A_{t,c}]_+ \leq \Pr_c(C_{t-1,c} \cap E_t \cap \{M_{t-1,c} \leq 2B_{t,c}\}) \leq \Pr_c(E_t). \tag{13}$$

Let

$$p_{t,c} = \Pr_c(C_{t-1,c} \cap E_t \cap \{M_{t-1,c} \leq 2B_{t,c}\}).$$

Accumulating these changes from the arrival of task c to the end of the stream gives

$$[A_{c,c} - A_{T,c}]_+ \leq \min\left\{1, \sum_{t=c+1}^{T} p_{t,c}\right\}, \qquad \mathrm{BWT} \geq -\frac{1}{T-1}\sum_{c<T} \min\left\{1, \sum_{t=c+1}^{T} p_{t,c}\right\}. \tag{14}$$

This analysis connects each new expert arrival to the retention term in eq. (1). Since historical experts are frozen, a previous prediction can be affected only when the newly added expert changes its routing decision. A routing change, however, does not necessarily alter the prediction. The prediction remains unchanged when the resulting response perturbation is smaller than its classification margin. Therefore, forgetting is limited to previously correct inputs that both switch routes and have insufficient margins to tolerate the induced perturbation. The same mechanism can also correct previously wrong predictions, so an expanding expert bank may lead to either forgetting or recovery. This margin-based analysis applies directly to fixed-score prediction settings such as CLIP classification, while language-generation metrics require a task-specific treatment.

Takeaway. PALM-Merge separates parameter preservation from prediction retention. Frozen experts protect historical task responses from being overwritten, while stable routing keeps these responses accessible. Even when routing changes, a sufficiently large prediction margin can preserve the original output. Retention is therefore determined jointly by routing stability and prediction robustness as new experts arrive.

5 Experiments

We evaluate PALM-Merge from the two perspectives: integration and retention. We first report the overall performance on vision and language tasks. We then use a controlled ViT-B/32 TA8 setting to examine where the gains come from. In particular, we study integration, retention, expert preservation, routing behavior, and the storage computation trade-off as the expert bank grows.

5.1 Experimental Setup

Vision experiments follow the FusionBench CLIP-ViT protocol (Tang et al., 2025a; Radford et al., 2021) across ViT-B/32, ViT-B/16, and ViT-L/14 with TA8, TA14, and TA20 streams; language experiments evaluate Flan-T5-base on GLUE-6 and GLUE-8 (Wang et al., 2018). We report final average accuracy (ACC) or average task score and backward transfer (BWT). Unless otherwise stated, vision experiments use $r = 64$, $q = 16$, and top-1 routing. Results average three seeded task orders; full configurations and metric definitions appear in Appendix B.

We include C.TA, C.TIES, OPCM, DOP, and NUFILT (Qiu et al., 2026; Tang et al., 2025b; Yang et al., 2025) as baselines and report both predictive performance and storage footprint to characterize their design trade-offs.

5.2 Main Results

Vision results. Across all nine CLIP-ViT settings, PALM-Merge achieves the highest final accuracy among the compared methods (table 1). Averaged across the nine settings, it reaches 89.8% accuracy

Table 1: CLIP-ViT continual merging across nine settings. ACC and BWT are in percent; higher is better for both. Means are shown here, with source uncertainties in Appendix C. Controlled TA8 results are detailed in Appendix K.

	Method	ViT-B/32			ViT-B/16			ViT-L/14			Avg
		8 tasks	14 tasks	20 tasks	8 tasks	14 tasks	20 tasks	8 tasks	14 tasks	20 tasks	
	Pre-Trained	48.1	56.9	55.6	55.4	62.0	59.8	64.9	69.1	65.6	59.7
	Fine-Tuned	90.4	89.3	89.8	92.4	91.3	91.6	94.3	93.4	93.5	91.8
	C. Fine-Tuned	79.8	67.4	62.6	82.9	72.2	68.2	90.0	70.9	77.7	74.6
ACC↑	Average (SWA)	66.3	65.4	61.1	72.3	69.7	64.8	80.0	77.5	71.1	69.8
ACC↑	C. TA	67.5	66.5	60.6	77.1	70.9	64.2	82.1	77.9	70.3	70.8
ACC↑	C. TIES	49.0	66.2	59.9	66.8	70.5	63.0	64.3	78.0	68.3	65.1
ACC↑	OPCM	75.5	71.9	65.7	81.8	77.1	70.3	87.0	83.5	76.0	76.5
ACC↑	DOP	78.3	73.0	68.5	84.5	76.2	72.2	88.3	85.5	80.7	78.6
ACC↑	NUFILT	83.6	78.0	71.0	87.3	83.1	78.1	91.6	89.2	84.7	83.0
ACC↑	PALM-Merge (Ours)	88.7	86.6	85.9	91.3	89.5	88.8	93.2	92.4	91.8	89.8
BWT↑	Average (SWA)	-11.5	-8.0	-7.1	-9.7	-7.1	-7.3	-7.3	-5.8	-6.4	-7.8
BWT↑	C. TA	-9.6	-1.3	-3.4	-4.2	-1.3	-3.6	-7.1	-1.8	-3.3	-4.0
BWT↑	C. TIES	-15.3	1.9	-1.5	-5.5	1.4	-1.5	-13.0	1.1	-2.9	-3.9
BWT↑	OPCM	-6.3	-6.0	-7.8	-4.8	-5.1	-6.3	-2.6	-4.3	-6.5	-5.5
BWT↑	DOP	-6.1	-8.8	-12.5	-3.9	-7.5	-11.4	-3.2	-4.6	-8.4	-7.4
BWT↑	NUFILT	-2.7	-5.7	-8.9	-1.6	-3.5	-7.1	-1.1	-2.0	-4.6	-4.1
BWT↑	PALM-Merge (Ours)	-0.8	-0.9	-1.7	-0.4	-0.7	-1.1	-0.5	-0.4	-1.2	-0.8

Table 2: Flan-T5-base results on the GLUE-6 continual-merging stream. Task scores, average performance (AVG), and BWT are in percent; higher is better. Single FT is a task-specific reference and therefore has no BWT. Bold denotes the highest value in each column.

Method	CoLA	MRPC	QNLI	QQP	RTE	SST-2	AVG ↑	BWT ↑
Single FT	69.1	85.5	90.9	84.0	84.4	92.9	84.6±0.0	–
SWA	62.0	80.2	86.1	79.2	78.0	86.8	78.7±0.0	-2.3±0.9
C. TA	57.4	80.6	87.1	79.5	76.9	88.0	78.3±0.0	-2.0±1.2
C. TIES	54.5	81.1	87.3	79.3	76.9	86.8	77.7±0.7	-1.4±1.1
OPCM	68.7	77.0	85.1	75.7	76.9	86.9	78.7±0.3	-2.3±0.6
DOP	67.9	82.4	87.9	79.9	77.9	89.2	80.8±0.1	-1.4±0.1
NUFILT	69.2	82.1	88.2	80.8	79.7	90.7	81.8±0.2	-1.0±0.4
PALM-Merge (Ours)	**71.0**	**83.3**	**89.8**	**81.3**	**80.1**	**90.9**	82.8±0.0	**-0.7**±0.0

and -0.8% BWT, compared with 83.0% and -4.1% for NUFILT. BWT stays within -1.7% to -0.4%, showing that the retention gain holds across architectures and stream lengths.

Language results. On Flan-T5-base GLUE-6, PALM-Merge is the strongest continual-merging method on all six tasks (Table 2). Relative to NUFILT, it improves CoLA, MRPC, QNLI, QQP, RTE, and SST-2 by 1.8%, 1.2%, 1.6%, 0.5%, 0.4%, and 0.2%, respectively. These consistent task-level gains raise the average score from 81.8% to 82.8%, while BWT improves from -1.0% to -0.7%. The resulting model is within 1.8% of the task-specific Single FT reference while retaining one model for the full stream.

The advantage also persists when the stream is extended to GLUE-8: PALM-Merge achieves an average task score of 80.5 with -0.9% BWT. This language result complements the nine-setting vision evaluation and shows that the same preserve-and-activate construction transfers across architectures and modalities. Configuration details, GLUE-8 task scores, and metric definitions are provided in Appendix B.1.

5.3 Where Does the Gain Come From? Integration and Retention

To understand where the improvement originates, we use the controlled ViT-B/32 TA8 setting and decompose final performance according to Eq.1. PALM-Merge reaches 88.70% ACC and -0.82% BWT, compared with 83.85% and -2.60% for NUFILT. The full four-method comparison is reported in Table 11 in Appendix K.

PALM-Merge achieves $A_{\text{arrival}} = 89.42\%$, compared with 86.12% for NUFILT. Of the 4.85% final accuracy gap, 3.30% arises from stronger integration and 1.55% from improved subsequent

retention. Thus, PALM-Merge both integrates new tasks more accurately and preserves more of their performance as the stream grows.

Fine-grained diagnostics in fig. 2 show that PALM-Merge reduces mean positive update drops from 0.66% for NUFILT to 0.25%, consistent with its improved BWT. Metric definitions and task-level analyses are provided in Appendix K.

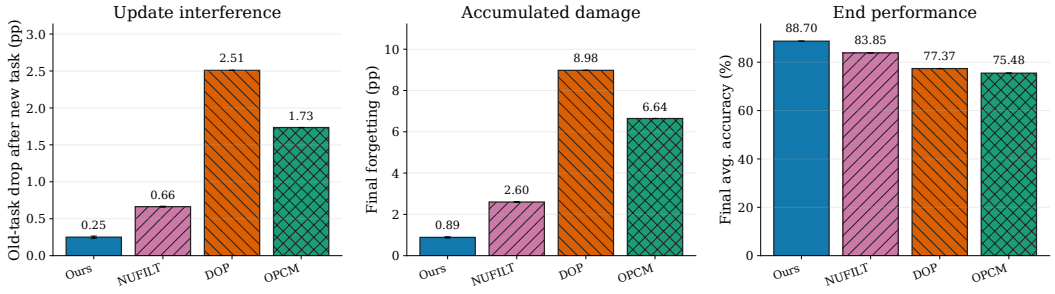

Figure 2: **Controlled ViT-B/32 TA8 retention.** Mean positive accuracy drop after each arrival (left), positive final forgetting (middle), and final average accuracy (right). All methods use the same three seeded task orders and evaluation protocol. Lower drops coincide with higher final accuracy for PALM-Merge.

5.4 Ablation: Preservation and Routing

Table 3: Expert rank on ViT-B/32 TA8 (three seeds, $q = 16$, top-$k = 1$). Storage and active-cost overhead are relative to the backbone; rank 64 is the default.

Rank r	ACC (%) ↑	Added storage (%)	Active overhead (%)
32	87.95	61.46	18.30
64	88.70	110.02	24.37
128	89.15	207.13	36.51
256	89.40	401.37	60.79

Preservation capacity. Increasing the expert rank consistently improves final accuracy, from 87.95% at $r = 32$ to 89.40% at $r = 256$ (table 3), consistent with the reduction in residual compression error predicted by our decomposition in **??**. The improvement, however, exhibits clear diminishing returns: increasing the rank from 32 to 64 yields a 0.75-point gain, whereas further increasing it from 128 to 256 improves accuracy by only 0.25 points. Meanwhile, the corresponding storage overhead rises from 61.46% to 401.37%, and the active-cost overhead increases from 18.30% to 60.79%. This trend suggests that preserving more singular directions improves the fidelity of each frozen residual expert, but progressively larger ranks retain increasingly less critical residual components while incurring substantially higher resource costs. We therefore use $r = 64$ as the default vision operating point, which captures most of the accuracy gain while maintaining a considerably smaller storage and active-computation footprint.

Table 4: Sparse activation on ViT-B/32 TA8 ($r = 64$, $q = 16$). Full-bank rows are analytical cost references for the same stored experts; dashes denote unmeasured accuracy. Overhead uses the original backbone as the denominator.

Evaluation	Experts / token / layer	Active overhead (%)	ACC (%)
Full bank (cost reference)	T	97.88	—
Full bank + gate (cost reference)	T	110.02	—
PALM-Merge top-1	1	24.37	88.70 ± 0.01

Selective access. Top-1 routing evaluates only one residual expert per token while retaining access to the full bank. It reaches 88.70% accuracy with 24.37% active-cost overhead, reducing total active cost by 40.8% relative to dense full-bank evaluation with routing scores (table 4).

These ablations probe the two components identified in eq. (3): rank controls the fidelity of preserved responses, whereas routing controls how that capacity is accessed. Because the methods use different storage budgets, these experiments should be interpreted as operating-point comparisons rather than a memory-matched causal decomposition.

5.5 Efficiency and Scaling

Efficiency and scaling. As the expert bank grows from TA8 to TA20 on ViT-B/32, storage overhead increases from 110.0% to 275.0%, corresponding to $2.10\times$ and $3.75\times$ total model storage. Active-cost overhead grows much more slowly, from 24.4% to 42.6%. This gap reflects the sparse inference design: every task adds a stored residual expert and routing basis, but each token only scores the growing routing bank and evaluates one selected residual. Hence, only routing scales with the number of stored tasks, while residual evaluation remains constant with respect to T. As a result, the reduction in total active cost relative to dense full-bank evaluation increases from 40.8% at TA8 to 62.0% at TA20. PALM-Merge therefore trades growing persistent storage for a substantially smaller increase in per-input computation as the task stream expands. Full scaling results and active-tensor accounting are provided in Appendix J.

6 Related Work

Model merging. Model merging combines independently adapted models without joint retraining. Model soups average compatible solutions (Wortsman et al., 2022), while task arithmetic represents adaptation through task vectors (Ilharco et al., 2023). TIES reduces conflicts through sign-aware merging (Yadav et al., 2023). Fisher merging and RegMean incorporate parameter importance or activation statistics (Matena & Raffel, 2022; Jin et al., 2023), while AdaMerging learns data-dependent coefficients from unlabeled examples (Yang et al., 2024). These methods consider a fixed model collection, whereas continual merging must update the model as checkpoints arrive sequentially.

Continual model merging. Continual model merging extends model merging to a stream of arriving checkpoints without access to previous task data. OPCM updates the merged model sequentially, while DOP and NUFILT mitigate interference through projection- and filtering-based mechanisms (Tang et al., 2025b; Yang et al., 2025; Qiu et al., 2026). These methods mainly control successive updates to shared parameters. In contrast, PALM-Merge preserves each arriving residual as a frozen low-rank expert and selects the required response from the input at inference time, separating response preservation from selective access.

Compact experts and selective access. Parameter-efficient adaptation provides compact task-specific updates (Hu et al., 2022; Houlsby et al., 2019), while modular continual learning preserves task-specific capacity in separate components (Rusu et al., 2016). Sparse mixture-of-experts models further enable selective access to specialized components (Shazeer et al., 2017). PALM-Merge brings these ideas into data-free continual merging: residual experts are derived from arriving checkpoints, frozen after insertion, and selected from current activations without task labels or retained data. Unlike standard sparse-expert training, neither the expert bank nor router is jointly trained; both are derived directly from the continual checkpoint stream.

Appendix A provides a broader comparison with parameter alignment, task-vector editing, parameter-efficient composition, sparse experts, and continual learning.

7 Conclusion and Limitations

We introduced PALM-Merge, a preserve-and-activate framework for data-free continual model merging that separates task-specific response preservation from selective access. Frozen low-rank experts preserve task-specific capacity, while input-dependent routing retrieves the appropriate response. Our analysis characterizes the limitation of shared mappings and shows how expert arrival, routing changes, and prediction margins jointly affect retention. On the controlled ViT-B/32 TA8 setting, PALM-Merge improves over NUFILT by 4.85%, including 3.30% from better integration and 1.55% from improved retention. The main limitation is the growing storage and routing cost of the expert bank. Future work should consider matched-memory comparisons, direct runtime measurements, and expert-bank compression for longer task streams.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.04836.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7120–7129, 2017. doi: 10.1109/CVPR.2017.753. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Aljundi_Expert_Gate_Lifelong_CVPR_2017_paper.html.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022. doi: 10.1109/TPAMI.2021.3057446. URL https://arxiv.org/abs/1909.08383.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3): 211–218, 1936. doi: 10.1007/BF02288367.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2110.06296.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://jmlr.org/papers/v23/21-0998.html.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3259–3269. PMLR, 2020. URL https://proceedings.mlr.press/v119/frankle20a.html.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://arxiv.org/abs/1802.10026.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

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. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. LoraHub: Efficient cross-task generalization via dynamic LoRA composition. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=lyRpY2bpBh.

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

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

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

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2006.16668.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pp. 614–629. Springer, 2016. doi: 10.1007/978-3-319-46493-0_37. URL https://arxiv.org/abs/1606.09282.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/f87522788a2be2d171666752f97ddebb-Abstract.html.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018. doi: 10.1109/CVPR.2018.00810. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Mallya_PackNet_Adding_Multiple_CVPR_2018_paper.html.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17703–17716, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/70c26937fbf3d4600b69a129031b66ec-Abstract-Conference.html.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2305.12827.

Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. MAD-X: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 7654–7673. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.617. URL https://aclanthology.org/2020.emnlp-main.617/.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rueckle, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 487–503. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.39. URL https://aclanthology.org/2021.eacl-main.39/.

Zihuan Qiu, Lei Wang, Yang Cao, Runtong Zhang, Bing Su, Yi Xu, Fanman Meng, Linfeng Xu, Qingbo Wu, and Hongliang Li. Null-space filtering for data-free continual model merging: Preserving stability, promoting plasticity. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=HDIf3fYqPP.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Rebuffi_iCaRL_Incremental_Classifier_CVPR_2017_paper.html.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. URL https://arxiv.org/abs/1606.04671.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22045–22055, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fb2697869f56484404c8ceee2985b01d-Abstract.html.

George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. ZipIt! merging models from different tasks without training. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.03053.

Anke Tang, Li Shen, Yong Luo, Enneng Yang, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. FusionBench: A unified library and comprehensive benchmark for deep model fusion. *Journal of Machine Learning Research*, 26(307):1–38, 2025a. URL https://jmlr.org/papers/v26/25-1243.html.

Anke Tang, Enneng Yang, Li Shen, Yong Luo, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. Merging on the fly without retraining: A sequential approach to scalable continual model merging. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. URL https://openreview.net/forum?id=rdGMyTPhui.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*, pp. 353–355, 2018. doi: 10.18653/v1/W18-5446.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1644c9af28ab7916874f6fd6228a9bcf-Abstract-Conference.html.

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

Enneng Yang, Anke Tang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, and Jie M. Zhang. Continual model merging without data: Dual projections for balancing stability and plasticity. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.neurips.cc/paper_files/paper/2025/hash/37d9f19150fce07bced2a81fc87d47a6-Abstract-Conference.html.

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57755–57775. PMLR, 2024. URL https://proceedings.mlr.press/v235/yu24p.html.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 2017. URL https://proceedings.mlr.press/v70/zenke17a.html.