How to Merge Multimodal Models Over Time?

Anonymous authors
Paper under double-blind review

Abstract

Model merging combines multiple "expert" models finetuned from a base foundation model on diverse tasks and domains into a single, more capable model. However, most existing model merging approaches assume that all experts are available simultaneously. In reality, new tasks and domains emerge over time, requiring strategies to integrate the knowledge of expert models as they become available: a process we call *temporal model merging*. The temporal dimension introduces unique challenges not addressed in prior work, raising new questions such as: when training for a new task, should the expert model start from the merged past experts or from the original base model? Should we merge all models at each time step? Which merging techniques are best suited for temporal merging? Should different strategies be used to initialize the training and deploy the model? To answer these questions, we propose a unified framework called TIME (Temporal Integration of Model Expertise) which defines temporal model merging across three axes: (1) initialization, (2) deployment, and (3) merging technique. Using TIME, we study temporal model merging across model sizes, compute budgets, and learning horizons on the FoMo-in-Flux benchmark. Our comprehensive suite of experiments across TIME allows us to build a better understanding of current challenges and best practices for effective temporal model merging.

1 Introduction

Foundation models consolidate a wide range of capabilities and knowledge into a single, large model. Consequently, model merging (Regent's et al., 1996; Wortsman et al., 2022a) has emerged as a key technique for unifying multiple task-specific specialist models derived from a shared base into a single, generalist model.

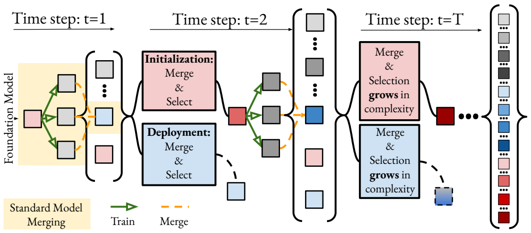
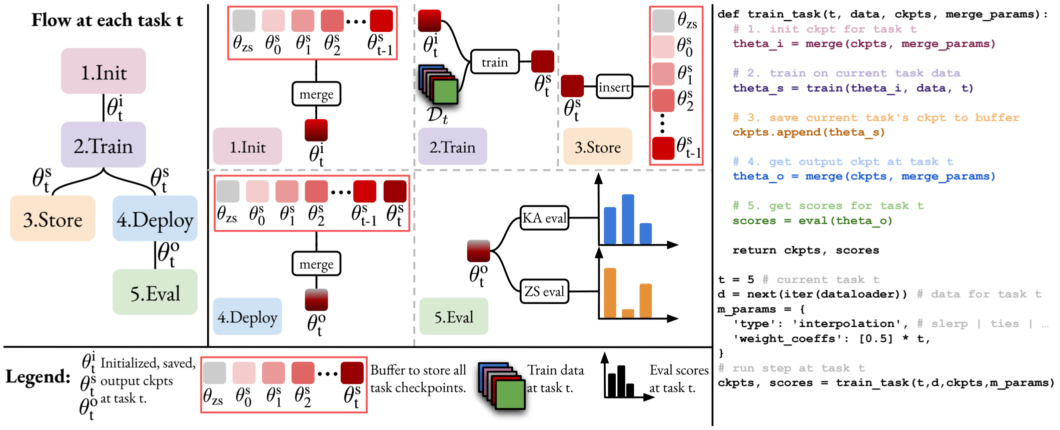


Figure 1: **Temporal Model Merging** generalizes standard model merging (yellow), which merges multiple trained experts once, in a single step. Our study reveals that initialization and deployment strategies are crucial in the temporal setting.

Current model merging approaches typically assume a fixed base model that is fine-tuned independently on k diverse tasks and domains to produce a set of *independent* experts (Garipov et al., 2018; Rofin et al., 2022; Ilharco et al., 2022; Yadav et al., 2023; Li et al., 2022), which are then merged simultaneously. Research in this field has therefore focused on improving merging techniques for larger or structurally different k-sets, exploring the impact of the diversity and scale of finetuning domains, tasks and experts.

However, the world is constantly evolving, leading to continuous shifts over data distributions, domains, and tasks, with new concepts emerging that may have been insufficiently covered during large-scale pretraining (Koh et al., 2021; Hu et al., 2022; Pratt et al., 2023; Menon & Vondrick, 2023; Zhang et al., 2021; Gui et al., 2024). This dynamic nature of real-world applications motivates a hitherto missing systematic exploration into *temporal model merging* (see Fig. 1) to better understand model merging along an additional, understudied axis: *time* (Zhou et al., 2024; Don-Yehiya et al., 2022). Specifically, in this work, we ask: **(1)** What is the best merging strategy to initialize each expert model before training? **(2)** What is the best merging strategy to deploy each expert model after training? **(3)** Which model merging techniques are most suitable for temporal merging? To answer these questions, we propose a unified framework for studying temporal model merging:



```
def train_task(t, data, ckpts, merge_params):
  # 1. init ckpt for task t
  theta_i = merge(ckpts, merge_params)

  # 2. train on current task data
  theta_s = train(theta_i, data, t)

  # 3. save current task's ckpt to buffer
  ckpts.append(theta_s)

  # 4. get output ckpt at task t
  theta_o = merge(ckpts, merge_params)

  # 5. get scores for task t
  scores = eval(theta_o)

  return ckpts, scores

t = 5 # current task t
d = next(iter(dataloader)) # data for task t
m_params = {
  'type': 'interpolation', # slerp | ties | …
  'weight_coeffs': [0.5] * t,
}
# run step at task t
ckpts, scores = train_task(t,d,ckpts,m_params)
```

Figure 2: **Design Space of Temporal Model Merging through TIME.** We showcase our framework for the per-task pipeline of temporal model merging over multiple tasks: At each task t, we first initialize the current checkpoint to start training from, θ_t^i, by using one or more previously stored checkpoints from previous tasks, either directly or by merging them. We train θ_t^i on current task data $\mathcal{D}_t$ to yield the current task checkpoint θ_t^s, which is inserted into the checkpoint buffer. Finally, to produce the output model, θ_t^o, we either merge previously stored checkpoints from the buffer or use them directly. The entire framework is depicted in the pseudo-code on the right panel.

TIME (Temporal Integration of Model Expertise). It is structured around three key axes spanning the design space of temporal merging solutions (as shown in Fig. 2): the choice of past checkpoints to merge before training on the current task (initialization), the choice of past checkpoints to merge after training on the current task (deployment), and the choice of the merging technique. Using our TIME framework, we position existing model merging approaches along each key axis and conduct a systematic study of model merging over time. For this, we leverage the multimodal FoMo-in-Flux by Roth et al. (2024b), a benchmark comprising 63 datasets with well-documented sequential properties, enabling a thorough investigation of temporal model merging under practical compute constraints. Our experiments systematically explore different merging techniques, initialization, and deployment strategies, providing several key insights:

Key Insights for Temporal Model Merging

[A] Accounting for time is crucial. Standard "*offline*" model merging techniques do not generalize well to the temporal merging setting (Sec. 3.1).

[B] Complex merging techniques matter little. Choosing sophisticated merging techniques beyond simple weighted averaging provides at best marginal benefits for temporal model merging, especially for long task sequences (Sec. 3.3).

[C] Initialization and deployment are critical. Choosing how to select and combine available weights before and after each task t is most important for temporal model merging (Sec. 3.2).

[D] Temporal model merging scales well. Larger models or compute budgets allows greater benefits from temporal merging. Scaling enables temporal model merging to even outperform the multitask model, trained on all tasks at once (Sec. 3.4).

2 Design Space of Temporal Model Merging

Notation. Throughout this work, we use t to refer to a given task at time t. Full model parameterization is denoted by θ, with the following key instantiations: θ_t represents model weights at task t, while θ_t^I, θ_t^S, and θ_t^O denote weights used for *initialization*, *saved* weight checkpoints at task t (i.e. the trained expert models), and the *output* deployed model weights, respectively. Note that while standard model merging considers model weights as elements of a fixed set $\{\theta_k\}_{k=1}^K$, temporal model merging organizes them along the time axis θ_t.

Table 1: **Comparison of model merging techniques.**

Method	**Sparsification**	**Consensus**	**Scaling**
Weight averaging Regent's et al. (1996); Wortsman et al. (2022a)	✗	Linear Int.	Weight coeff.
SLERP Ramé et al. (2024)	✗	Spherical Int.	Weight coeff.
Task Arithmetic Ilharco et al. (2023)	✗	Linear Int.	Scaling factor
MagMax Marczak et al. (2024)	✗	Max. Magnitude	Scaling factor
TIES Yadav et al. (2023)	Top-k	Sign Agreement	Scaling factor
DARE-TIES Yu et al. (2024)	Random	Sign Agreement	Scaling factor
Breadcrumbs-TIES Davari & Belilovsky (2025)	Top/Bottom-k	Sign Agreement	Scaling factor
Model Stock Jang et al. (2024)	✗	Geometric	Adaptive ratio
LiNeS Wang et al. (2024a)	✗	✗	Layer weights

2.1 Temporal Model Merging through TIME

Standard model merging is typically performed offline, after all experts have been trained to convergence. In contrast, model merging in continual pretraining is generally done sequentially, using past checkpoints. Both approaches are specific instances of our more general temporal merging framework, TIME, which defines temporal merging along *three key axes*: initialization of each expert, merging for deployment at step t, and merging techniques f_{merge} applied over time:

Axis 1: Initialization

As expert models are created continuously over time, initialization becomes a crucial choice. Unlike model merging at a single point in time, the number of potential starting points grows exponentially over time as new experts are created. This raises the question: should starting points for each time step be derived from the base weights (as in traditional merging), from a merged combination of previous experts, or from most recent weights? In this work, we study the following initialization protocols at time step t for TIME: **(1)** $\texttt{init}_{\text{ZS}}$, which consistently initializes with the base zero-shot model weights θ_0 at each timestep t; **(2)** $\texttt{init}_{\text{FT}}$, which at step t always initializes with the latest available finetuned model weights θ_{t-1}^{S}; and **(3)** $\texttt{init}_{\text{EMA}}$, which computes an unrolled exponential moving average merge over all previously seen expert models $\{\theta_{t'}^{S}\}_{1,\ldots,t-1}$ following the equation:

$$\theta_{t'}^{\text{EMA}} = f_{\text{merge}}\left(\theta_{t'-1}^{\text{EMA}}, \theta_{t'-1}^{S}, \mathcal{F}\right) \tag{1}$$

with merging hyperparameters $\mathcal{F}$. Consequently, the initialization weights are given as $\theta_t^I = \theta_t^{\text{EMA}}$.

Axis 2: Deployment

With each update iteration and expert training phase t, a decision must be made on the final model to deploy, determining which weights to present for downstream use. In continual pretraining, the trained model θ_t^S is deployed directly. In contrast, standard model merging applies a merging technique f_{merge} to a fixed set of k expert models. Temporal model merging, however, must account for both previously deployed models and the growing number of expert models available over time. Unlike standard merging, where k remains constant, the number of experts to merge increases with each step. As a result, temporal merging introduces the idea of weighted combinations, balancing recent updates with retained past knowledge to achieve adaptability and stability—both critical for effective continual learning (Kirkpatrick et al., 2017; Zenke et al., 2017). In this work, we study three strategies for model deployment: **(1)** $\texttt{deploy}_{\text{FT}}$, which always deploys the latest finetuned expert model at step t, i.e., $\theta_s^O = \theta_t^S$; **(2)** $\texttt{deploy}_{\text{EMA}}$, which computes an unrolled exponential moving average merge over all expert models, i.e., $\theta_t^O = \theta_{t+1}^{\text{EMA}}$ following Eq. (1); and **(3)** $\texttt{deploy}_{\text{ALL}}$, which applies a merging technique f_{merge} over all previously computed expert models $\{\theta_{t'}^S\}_1^{t-1}$.

Axis 3: Merging Techniques

At each point in time, for both initialization and deployment, merging technique f_{merge} defines *how* to combine the available expert models and checkpoints. In this work, we study nine different variants in total, shown in Tab. 1. Please refer to Appendix B for details.

2.2 Complete Temporal Model Merging Pipeline

Incorporating all three axes of temporal model merging, we define a five-stage update pipeline for each task t (see Fig. 2), consisting of the following steps: **(1) Init:** choose one of the aforementioned initialization protocols—$\texttt{init}_{\text{ZS}}$, $\texttt{init}_{\text{FT}}$, or $\texttt{init}_{\text{EMA}}$. This produces initialization weights θ_t^I at task t; **(2) Train:** given θ_t^I, train on current task data $\mathcal{D}_t$ within a set compute budget to produce the expert model θ_t^S; **(3) Store:** append θ_t^S to storage of expert model weights $\mathcal{S}$; **(4) Deploy:** choose a deployment protocol: $\texttt{deploy}_{\text{FT}}$, $\texttt{deploy}_{\text{EMA}}$, or $\texttt{deploy}_{\text{ALL}}$, and produce the output weights $\theta_t^O = \texttt{deploy}(\mathcal{S})$; and **(5) Eval:** the deployed θ_t^O is used for downstream applications and, in our case, extensive evaluation. Note that particular choices of $\texttt{init}$, $\texttt{deploy}$ and f_{merge} correspond to existing approaches, for example $(\texttt{init}_{\text{ZS}}, \texttt{deploy}_{\text{ALL}}, f_{\text{merge}}^{\text{WA}})$ simply recovers offline merging through weight averaging over experts models derived from original base weights θ_0 for each task t. Similarly, $(\texttt{init}_{\text{FT}}, \texttt{deploy}_{\text{EMA}}, f_{\text{merge}}^{\text{WA}})$ recovers exponential moving average approaches as done in Stojanovski et al. (2022); Roth et al. (2024b).

3 Experiments

Training at task *t*. We continuously finetune and merge a ViT-B/16 CLIP Radford et al. (2021); Cherti et al. (2022) model using the standard CLIP objective. The model is pretrained on LAION-2B (Schuhmann et al., 2022). We fix the training steps for each task based on the `DataComp-Small` budget of 1.8×10^9 GFLOPS, split equally across 20 tasks. At each step, we allow unrestricted access to a pretraining data pool $\mathcal{P}$, using the same 2M random subset of `LAION-400M` as in Roth et al. (2024b). We use a cosine-decay LR schedule with a linear warmup of 10%, AdamW optimizer Loshchilov & Hutter (2017), a batch size of 512 and gradient norm clipping to 1. All experiments use PyTorch Paszke et al. (2019), and are run on a compute cluster using NVIDIA A100/H100s.

Evaluation and Metrics. We use the continual pretraining benchmark Fomo-in-Flux by Roth et al. (2024b). It includes 41 adaptation datasets and 22 separate evaluation datasets, covering visual and semantic distribution shifts. We focus on two aspects: the level of *adaptation*, reflecting performance improvement with each merging step, and *retention*, capturing the preservation of prior knowledge. Specifically, we report two metrics following Roth et al. (2024b): Knowledge Accumulation ($\mathcal{A}_{KA}$), the average accuracy (or recall@5 for retrieval) across all 41 adaptation datasets, and Zero-Shot Retention ($\mathcal{A}_{ZS}$), the zero-shot accuracy or recall@5 on all 22 held-out evaluation datasets. Additional details can be found in the supplementary.

3.1 Do We Need Model Merging Across Time?

The simplest approach to temporal merging is to disregard the time axis and follow the standard *offline* merging paradigm. In TIME terms, this corresponds to a configuration of $(\texttt{init}_{\text{ZS}}, \texttt{deploy}_{\text{ALL}}, f_{\text{merge}})$, which always fine-tunes the initial base weights θ_0. To study the effectiveness of this strategy, we test it with various choices of f_{merge} in Fig. 6, including averaging, task-arithmetic, magmax, ties, dare-ties, breadcrumbs-ties, and lines-ties. For context, we include **(1)** a simple continual fine-tuning baseline (replay), which replays on both pretraining and previous task data, **(2)** initial zero-shot (θ_0) performance lower bound, and **(3)** multitask training upper bound. We visualize trajectories over time for knowledge accumulation $\mathcal{A}_{\text{KA}}$, zero-shot retention $\mathcal{A}_{\text{ZS}}$, and the geometric mean. Our results show that there are marginal differences between merging techniques when deployed in an offline manner for a temporal problem, and they all trace similar trajectories in the $\mathcal{A}_{\text{KA}}$ and $\mathcal{A}_{\text{ZS}}$ space and achieve similar final performance. Overall, however, unlike straightforward continual fine-tuning (replay), offline merging with *any* technique fails to address the temporal aspects of the problem, particularly struggling to consistently acquire new knowledge over time (as shown in Fig. 6, left).

3.2 TIME Travel

Since offline merging is ill suited to the temporal setting, we systematically explore the design space for temporal merging methods by testing all valid combinations of three initialization protocols and three deployment protocols described in Sec. 2.2. After discarding incompatible pairs, such as

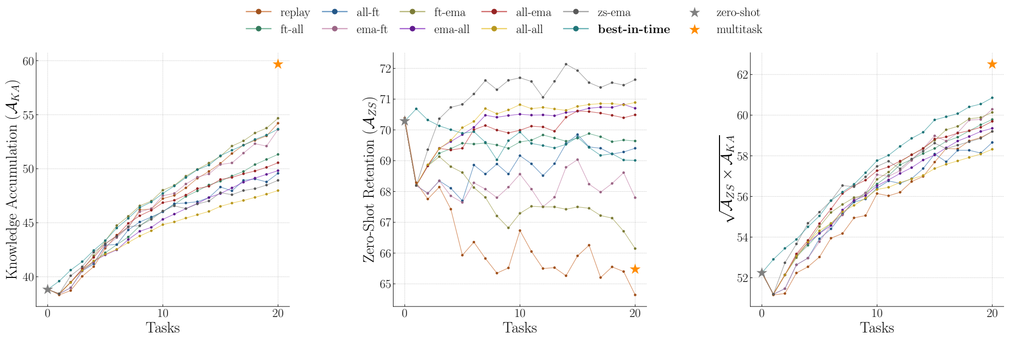


Figure 3: **A journey through TIME.** We explore various initialization and deployment protocols, finding that the EMA initialization-deployment strikes the best balance between knowledge accumulation and zero-shot retention. We refer to this strategy as *Best-in-*TIME.

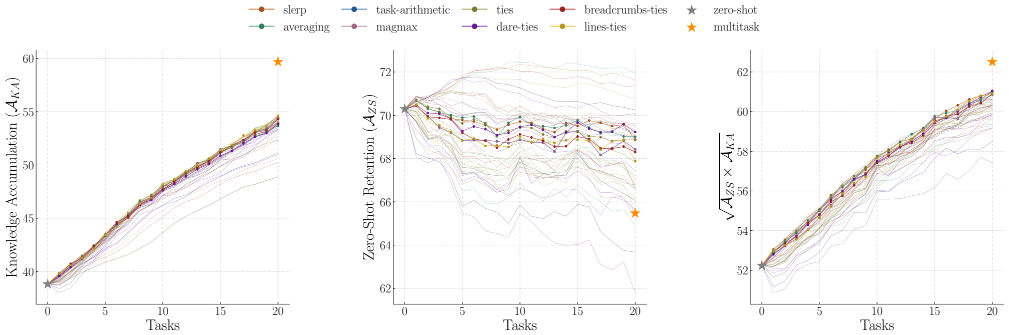


Figure 4: **Sweeping *Best-in-*TIME.** All merging techniques perform well with the *Best-in-*TIME strategy. Indeed, there are no significant differences between techniques, indicating that initialization and deployment matter more for temporal merging.

$\texttt{init}_{\text{ZS}}$ with $\texttt{deploy}_{\text{FT}}$, we evaluated the remaining eight variants using weight averaging as the merging technique. As shown in Fig. 3, the choice of initialization and deployment strategy largely determines performance, significantly affecting both knowledge accumulation and retention. One combination that stands out consistently is $\texttt{init}_{\text{EMA}}$ with $\texttt{deploy}_{\text{EMA}}$. This supports the findings of Stojanovski et al. (2022); Roth et al. (2024b) on small-scale continual learning and pretraining.

As the application of EMA model merging achieves a notably better balance between knowledge accumulation and retention than other methods, we call this approach *Best-in-*TIME. In the next section, we will explore the robustness of this strategy across different merging techniques. Please refer to Appendix E for additional EMA experiments.

3.3 What is the best merge for Best-in-TIME?

Having identified the optimal initialization and deployment merging strategy, we now investigate the robustness of our finding by sweeping over merging techniques. In particular, we test 7 different merging techniques while keeping the *Best-in-*TIME initialization and deployment strategy. From Fig. 4, it is clear that all merging techniques perform very similarly. This indicates that, over a sufficiently long time horizon, all techniques converge to a similar behavior, echoing our results in Sec. 3.1. However, we do notice higher variance in the retention metric ($\mathcal{A}_{ZS}$).

3.4 Scaling Up Temporal Model Merging

We next scale temporal model merging up across three-axes: *model size*, *compute budget*, and *number of tasks* (results in Fig. 5 and Appendix E.2). All our experiments use the *Best-in-*TIME setup described previously, conducting hyperparameter-optimal EMA at each task.

Scaling the Model. As we increase the model scale from $S/16$ (62.3M parameters) to $B/16$ (149.6M), $L/14$ (427.6M), and finally $g/14$ (1.37B) in Fig. 5 (left), we study the tradeoff between knowledge accumulation and retention over time. We compare sequential fine-tuning (circles) and

*Best-in-*TIME (squares). *Best-in-*TIME scales well with model size, with larger models exhibiting increased affinity to merges over time. This extends and further corroborates offline merging insights by Yadav et al. (2023), who showed that model scale facilitates merging. Moreover, while Roth et al. (2024b) and Ibrahim et al. (2024) highlight better continual fine-tuning with scale, we show temporal model merging to be substantially more effective across scale. For larger models all the way to the largest ViT-g/14, *Best-in-*TIME vastly outperforms or matches sequential fine-tuning and the multitask target in knowledge retention and positive backward transfer. Furthermore, scale facilitates equivalent degrees of knowledge accumulation between sequential fine-tuning and temporal model merging. Therefore, our model scaling results strongly suggest the use of temporal model merging solutions over standard continual fine-tuning methods.

Scaling the Compute. Keeping the underlying base model fixed to ViT-B/16, we next change the available compute budget by increasing the number of update steps per task. We compare a multitask model, trained on all tasks simultaneously, to a budget-optimal *Best-in-*TIME. The only hyperparameter for *Best-in-*TIME is the interpolation weight w. For each compute budget, there is a clear optimal choice of that hyperparameter (suboptimal runs shown as gray dots in Fig. 5 (right)). Higher values of w put greater emphasis on accumulation, allowing optimal accumulation-retention trade-offs to be reached at lower compute budgets. However, for a larger compute budget, less aggressive temporal model merging can achieve higher absolute trade-offs. Note that in Fig. 5 (right), we report the geometric mean between accumulation and retention, corresponding to the right-most panel in previous plots. *Best-in-*TIME scales very well across compute budgets, *clearly approaching the multitask upper bound* in accumulation-retention balance at larger compute budgets.

Scaling the Number of Tasks. Given that all our results until now have been with T=20, we next study how *Best-in-*TIME performs as we increase the number of merging time-steps to much longer time-sequences: T=50 and T=100. *Best-in-*TIME remains the optimal method of choice across different initialization and deployment strategies. Please refer to Appendix E.2 for details.

4 Conclusion

In this work, we study *temporal model merging*, addressing the challenge of continually merging multimodal models as new tasks and data arrive, and new expert models are trained in succession. To formalize this setting, we propose TIME, a novel unifying framework breaking down temporal model merging into three key axes: (1) initialization phase defining starting weights before each task, (2) deployment phase denoting post-training expert model aggregation, and (3) the choice of merging technique. Using TIME, we conduct a large-scale systematic study uncovering crucial practical guidelines for temporal model merging. Our experiments on the FoMo-in-Flux benchmark spanning 63 datasets, showcase that accounting for the temporal aspect is crucial, with standard offline merging techniques falling short in this dynamic setting. Moreover, we find the particular choice of merging technique matters far less than the merging strategy for initialization and deployment. Finally, we introduce *Best-in-*TIME, which scales favorably with model size and outperforms existing methods for continual multimodal pretraining. Our work provides a systematic entry point into temporal model merging and establishes best practices for this emerging field.

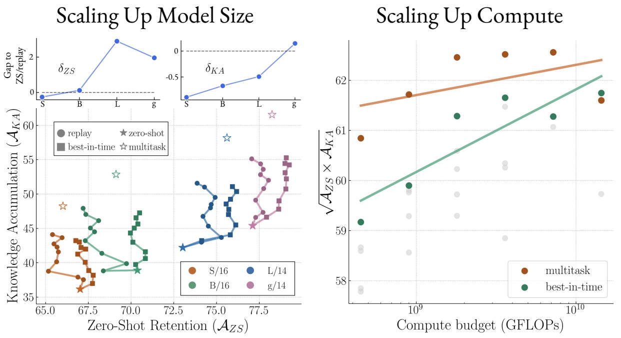


Figure 5: **Scaling up model merging.** *(left)* With scale, we observe continued improvements of model merging compared to the standard replay baseline. *(right)* Our *Best-in-*TIME method continues to improve with scaled total compute budget moving close to the multitask upper-bound. Gray points in the plot visualize suboptimal *Best-in-*TIME hyperparameter-instantiations.

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. URL https://openreview.net/forum?id=CQsmMYmlP5T.

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

Anton Alexandrov, Veselin Raychev, Mark Niklas Müller, Ce Zhang, Martin Vechev, and Kristina Toutanova. Mitigating catastrophic forgetting in language transfer via model merging. *arXiv preprint arXiv:2407.08699*, 2024.

Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. Knowledge distillation: A good teacher is patient and consistent. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10925–10934, 2022.

Jorg Bornschein, Alexandre Galashov, Ross Hemsley, Amal Rannen-Triki, Yutian Chen, Arslan Chaudhry, Xu Owen He, Arthur Douillard, Massimo Caccia, Qixuan Feng, et al. Nevis' 22: A stream of 100 tasks sampled from 30 years of computer vision research. *JMLR*, 2023.

Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *ICCV*, 2021.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. *arXiv preprint arXiv:2212.07143*, 2022.

Leshem Choshen, Elad Venezian, Shachar Don-Yehia, Noam Slonim, and Yoav Katz. Where to start? analyzing the potential value of intermediate models. *arXiv preprint arXiv:2211.00107*, 2022.

Geoffrey Cideron, Andrea Agostinelli, Johan Ferret, Sertan Girgin, Romuald Elie, Olivier Bachem, Sarah Perrin, and Alexandre Ramé. Diversity-rewarded cfg distillation. *arXiv preprint arXiv:2410.06084*, 2024.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 270–287, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73226-3.

Pala Tej Deep, Rishabh Bhardwaj, and Soujanya Poria. Della-merging: Reducing interference in model merging through magnitude-based sampling. *arXiv preprint arXiv:2406.11617*, 2024.

Shachar Don-Yehiya, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Cold fusion: Collaborative descent for distributed multitask finetuning. *arXiv preprint arXiv:2212.01378*, 2022.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3259–3269. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/frankle20a.html.

Saurabh Garg, Mehrdad Farajtabar, Hadi Pouransari, Raviteja Vemulapalli, Sachin Mehta, Oncel Tuzel, Vaishaal Shankar, and Fartash Faghri. Tic-clip: Continual training of clip models. In *ICLR*, 2024.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/be3087e74e9100d4bc4c6268cdbe8456-Paper.pdf.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's mergekit: A toolkit for merging large language models. *arXiv preprint arXiv:2403.13257*, 2024.

Alexey Gorbatovski, Boris Shaposhnikov, Alexey Malakhov, Nikita Surnachev, Yaroslav Aksenov, Ian Maksimov, Nikita Balagansky, and Daniil Gavrilov. Learn your reference model for real good alignment. *arXiv preprint arXiv:2404.09656*, 2024.

Zhongrui Gui, Shuyang Sun, Runjia Li, Jianhao Yuan, Zhaochong An, Karsten Roth, Ameya Prabhu, and Philip Torr. knn-clip: Retrieval enables training-free segmentation on continually expanding large vocabularies. *arXiv preprint arXiv:2404.09447*, 2024.

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

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`.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and scalable strategies to continually pre-train large language models. *arXiv preprint arXiv:2403.08763*, 2024.

Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights. In *NeurIPS*, 2022.

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. URL `https://openreview.net/forum?id=6t0Kwf8-jrj`.

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

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XLIV*, pp. 207–223, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-72783-2. doi: 10.1007/978-3-031-72784-9_12. URL `https://doi.org/10.1007/978-3-031-72784-9_12`.

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. URL `https://openreview.net/forum?id=FCnohuR6AnM`.

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

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. *arXiv preprint arXiv:2402.07865*, 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, pp. 5637–5664. PMLR, 2021.

Jeffrey Li, Mohammadreza Armandpour, Seyed Iman Mirzadeh, Sachin Mehta, Vaishaal Shankar, Raviteja Vemulapalli, Oncel Tuzel, Mehrdad Farajtabar, Hadi Pouransari, and Fartash Faghri. Tic-lm: A multi-year benchmark for continual pretraining of language models. In *NeurIPS 2024 Workshop on Scalable Continual Learning for Lifelong Foundation Models*.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*, 2022.

Tao Li, Weisen Jiang, Fanghui Liu, Xiaolin Huang, and James T Kwok. Learning scalable model soup on a single gpu: An efficient subspace training strategy. *arXiv preprint arXiv:2407.03641*, 2024.

Yong Lin, Hangyu Lin, Wei Xiong, Shizhe Diao, Jianmeng Liu, Jipeng Zhang, Rui Pan, Haoxiang Wang, Wenbin Hu, Hanning Zhang, Hanze Dong, Renjie Pi, Han Zhao, Nan Jiang, Heng Ji, Yuan Yao, and Tong Zhang. Mitigating the alignment tax of RLHF. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 580–606, Miami, Florida, USA, November 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.emnlp-main.35.

Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The clear benchmark: Continual learning on real-world imagery. In *NeurIPS*, 2021.

Adam Liska, Tomas Kocisky, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, D'Autume Cyprien De Masson, Tim Scholtes, Manzil Zaheer, Susannah Young, et al. Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models. In *International Conference on Machine Learning*, pp. 13604–13622. PMLR, 2022.

Tianlin Liu, Shangmin Guo, Leonardo Bianco, Daniele Calandriello, Quentin Berthet, Felipe Llinares-López, Jessica Hoffmann, Lucas Dixon, Michal Valko, and Mathieu Blondel. Decoding-time realignment of language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 31015–31031. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu24r.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Jinliang Lu, Ziliang Pang, Min Xiao, Yaochen Zhu, Rui Xia, and Jiajun Zhang. Merge, ensemble, and cooperate! a survey on collaborative strategies in the era of large language models. *arXiv preprint arXiv:2407.06089*, 2024.

Daniel Marczak, Bartłomiej Twardowski, Tomasz Trzciński, and Sebastian Cygert. Magmax: Leveraging model merging for seamless continual learning. *arXiv preprint arXiv:2407.06322*, 2024.

Michael S Matena and Colin Raffel. Merging models with fisher-weighted averaging. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=LSKlp_aceOC.

Matías Mendieta, Boran Han, Xingjian Shi, Yi Zhu, and Chen Chen. Towards geospatial foundation models via continual pretraining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16806–16816, 2023.

Sachit Menon and Carl Vondrick. Visual classification via description from large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=jlAjNL8z5cs.

Remi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Zhaohan Daniel Guo, Yunhao Tang, Matthieu Geist, Thomas Mesnard, Côme Fiegel, Andrea Michi, Marco Selvi, Sertan Girgin, Nikola Momchev, Olivier Bachem, Daniel J Mankowitz, Doina Precup, and Bilal Piot. Nash learning from human feedback. In Ruslan Salakhutdinov, Zico

Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 36743–36768. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/munos24a.html`.

Anshul Nasery, Jonathan Hayase, Pang Wei Koh, and Sewoong Oh. Pleas–merging models with permutations and least squares. *arXiv preprint arXiv:2407.02447*, 2024.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 512–523. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/0607f4c705595b911a4f3e7a127b44e0-Paper.pdf`.

Kai Nylund, Suchin Gururangan, and Noah A Smith. Time is encoded in the weights of finetuned language models. *arXiv preprint arXiv:2312.13401*, 2023.

Changdae Oh, Yixuan Li, Kyungwoo Song, Sangdoo Yun, and Dongyoon Han. Dawin: Training-free dynamic weight interpolation for robust adaptation. *arXiv preprint arXiv:2410.03782*, 2024.

Oleksiy Ostapenko, Timothee Lesort, Pau Rodríguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2022.

Jyothish Pari, Samy Jelassi, and Pulkit Agrawal. Collective model intelligence requires compatible specialization. *arXiv preprint arXiv:2411.02207*, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Ameya Prabhu, Zhipeng Cai, Puneet Dokania, Philip Torr, Vladlen Koltun, and Ozan Sener. Online continual learning without the storage constraint. *arXiv preprint arXiv:2305.09253*, 2023a.

Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet Dokania, Philip HS Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. Computationally budgeted continual learning: What does matter? In *CVPR*, 2023b.

Ameya Prabhu, Hasan Abed Al Kader Hammoud, Ser-Nam Lim, Bernard Ghanem, Philip HS Torr, and Adel Bibi. From categories to classifier: Name-only continual learning by exploring the web. *arXiv preprint arXiv:2311.11293*, 2023c.

Sarah Pratt, Ian Covert, Rosanne Liu, and Ali Farhadi. What does a platypus look like? generating customized prompts for zero-shot image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15691–15701, October 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, 2021.

Alexandre Rame, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Leon Bottou, and David Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28656–28679. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/rame23a.html`.

Alexandre Ramé, Johan Ferret, Nino Vieillard, Robert Dadashi, Léonard Hussenot, Pierre-Louis Cedoz, Pier Giuseppe Sessa, Sertan Girgin, Arthur Douillard, and Olivier Bachem. Warp: On the benefits of weight averaged rewarded policies. *arXiv preprint arXiv:2406.16768*, 2024.

Alexandre Ramé, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, 2024. ISBN 9781713871088.

Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187*, 2024.

Regent's, ParkLondon, Ukj, and . Utans. Weight averaging for neural networksand local resampling. 1996. URL `https://api.semanticscholar.org/CorpusID:475398`.

Mark Rofin, Nikita Balagansky, and Daniil Gavrilov. Linear interpolation in parameter space is good enough for fine-tuned language models. *arXiv preprint arXiv:2211.12092*, 2022.

Karsten Roth, Lukas Thede, A. Sophia Koepke, Oriol Vinyals, Olivier J Henaff, and Zeynep Akata. Fantastic gains and where to find them: On the existence and prospect of general knowledge transfer between any pretrained model. In *The Twelfth International Conference on Learning Representations*, 2024a. URL `https://openreview.net/forum?id=m50eKHCttz`.

Karsten Roth, Vishaal Udandarao, Sebastian Dziadzio, Ameya Prabhu, Mehdi Cherti, Oriol Vinyals, Olivier Hénaff, Samuel Albanie, Matthias Bethge, and Zeynep Akata. A practitioner's guide to continual multimodal pretraining. *arXiv preprint arXiv:2408.14471*, 2024b.

Sunny Sanyal, Atula Tejaswi Neerkaje, Jean Kaddour, Abhishek Kumar, and sujay sanghavi. Early weight averaging meets high learning rates for LLM pre-training. In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=IA8CWtNkUr`.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

Ekansh Sharma, Daniel M Roy, and Gintare Karolina Dziugaite. The non-local model merging problem: Permutation symmetries and variance collapse. *arXiv preprint arXiv:2410.12766*, 2024.

Li Shen, Anke Tang, Enneng Yang, Guibing Guo, Yong Luo, Lefei Zhang, Xiaochun Cao, Bo Du, and Dacheng Tao. Efficient and effective weight-ensembling mixture of experts for multi-task model merging. *arXiv preprint arXiv:2410.21804*, 2024.

Ken Shoemake. Animating rotation with quaternion curves. *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 1985. URL `https://api.semanticscholar.org/CorpusID:11290566`.

Shikhar Srivastava, Md Yousuf Harun, Robik Shrestha, and Christopher Kanan. Improving multimodal large language models using continual learning. *arXiv preprint arXiv:2410.19925*, 2024.

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

Zafir Stojanovski, Karsten Roth, and Zeynep Akata. Momentum-based weight interpolation of strong zero-shot models for continual learning. *arXiv preprint arXiv:2211.03186*, 2022.

Yi-Lin Sung, Linjie Li, Kevin Lin, Zhe Gan, Mohit Bansal, and Lijuan Wang. An empirical study of multimodal model merging. *arXiv preprint arXiv:2304.14933*, 2023.

Derek Tam, Mohit Bansal, and Colin Raffel. Merging by matching models in task parameter subspaces. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL `https://openreview.net/forum?id=qNGo6ghWFB`.

Derek Tam, Yash Kant, Brian Lester, Igor Gilitschenski, and Colin Raffel. Realistic evaluation of model merging for compositional generalization. *arXiv preprint arXiv:2409.18314*, 2024b.

Lukas Thede, Karsten Roth, Olivier J Hénaff, Matthias Bethge, and Zeynep Akata. Reflecting on the state of rehearsal-free continual learning with pretrained models. *arXiv preprint arXiv:2406.09384*, 2024.

Ke Wang, Nikolaos Dimitriadis, Alessandro Favero, Guillermo Ortiz-Jimenez, Francois Fleuret, and Pascal Frossard. Lines: Post-training layer scaling prevents forgetting and enhances model merging. *arXiv preprint arXiv:2410.17146*, 2024a.

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

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

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7959–7971, June 2022b.

Feng Xiong, Runxi Cheng, Wang Chen, Zhanqiu Zhang, Yiwen Guo, Chun Yuan, and Ruifeng Xu. Multi-task model merging via adaptive weight disentanglement. *arXiv preprint arXiv:2411.18729*, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xtaX3WyCj1.

Prateek Yadav, Colin Raffel, Mohammed Muqeeth, Lucas Caccia, Haokun Liu, Tianlong Chen, Mohit Bansal, Leshem Choshen, and Alessandro Sordoni. A survey on model moerging: Recycling and routing among specialized experts for collaborative learning. *arXiv preprint arXiv:2408.07057*, 2024a.

Prateek Yadav, Tu Vu, Jonathan Lai, Alexandra Chronopoulou, Manaal Faruqui, Mohit Bansal, and Tsendsuren Munkhdalai. What matters for model merging at scale? *arXiv preprint arXiv:2410.03617*, 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. *arXiv preprint arXiv:2408.07666*, 2024a.

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. *arXiv preprint arXiv:2402.02705*, 2024b.

Çağatay Yıldız, Nishaanth Kanna Ravichandran, Prishruit Punia, Matthias Bethge, and Beyza Ermis. Investigating continual pretraining in large language models: Insights and implications. *arXiv preprint arXiv:2402.17400*, 2024.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57755–57775. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/yu24p.html.

Kerem Zaman, Leshem Choshen, and Shashank Srivastava. Fuse to forget: Bias reduction and selective memorization through model fusion. *arXiv preprint arXiv:2311.07682*, 2023.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

Renrui Zhang, Rongyao Fang, Peng Gao, Wei Zhang, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. *arXiv preprint arXiv:2111.03930*, 2021.

Shenghe Zheng and Hongzhi Wang. Free-merging: Fourier transform for model merging with lightweight experts. *arXiv preprint arXiv:2411.16815*, 2024.

Luca Zhou, Daniele Solombrino, Donato Crisostomi, Maria Sofia Bucarelli, Fabrizio Silvestri, and Emanuele Rodolà. Atm: Improving model merging by alternating tuning and merging. *arXiv preprint arXiv:2411.03055*, 2024.

Acknowledgements. The authors would like to thank (in random order) Shyamgopal Karthik, Shashwat Goel, Ankit Sonthalia, Olivier Hénaff, Alexandre Ramé, and Daniel Marczak for helpful feedback. The plotting style in our work is inspired by figures from Beyer et al. (2022). The style of Fig. 2 is inspired by Figure 1 of Karamcheti et al. (2024). VU, KR, and SD thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS). VU, KR, and SD also thank the European Laboratory for Learning and Intelligent Systems (ELLIS) PhD program for support. VU is supported by a Google PhD Fellowship in Machine Intelligence. SA is supported by a Newton Trust Grant. AP is supported by the Federal Ministry of Education and Research (BMBF), FKZ: 011524085B. MB acknowledges financial support via the Open Philanthropy Foundation funded by the Good Ventures Foundation. MB is a member of the Machine Learning Cluster of Excellence, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC number 2064/1 – Project number 390727645. ZA acknowledges the support from the German Research Foundation (DFG): SFB 1233, Robust Vision: Inference Principles and Neural Mechanisms, project number: 276693517 and ERC Grant DEXIM, project number: 853489. This research utilized compute resources at the Tübingen Machine Learning Cloud, DFG FKZ INST 37/1057-1 FUGG.

A Related Works

Model Merging. We provide a short overview of the model merging literature, detailed in these excellent surveys (Yadav et al., 2024a; Yang et al., 2024a). While both model aggregation through distillation Roth et al. (2024a); Cideron et al. (2024) and averaging checkpoints during training (Kaddour, 2022; Sanyal et al., 2024; Li et al., 2024) have shown success, the requirement of additional compute limits practicability of these methods (Prabhu et al., 2023b). Instead, recent work (Wortsman et al., 2022b;a; Ilharco et al., 2022; 2023; Rame et al., 2023; Sanyal et al., 2024; Sung et al., 2023; Pari et al., 2024; Nylund et al., 2023; Zaman et al., 2023; Stoica et al., 2024; Wang et al., 2024b; He et al., 2024; Oh et al., 2024; Shen et al., 2024; Sharma et al., 2024; Goddard et al., 2024; Yadav et al., 2024a; Xiong et al., 2024; Yang et al., 2024b; Lu et al., 2024; Zheng & Wang, 2024; Nasery et al., 2024) has shown the effectiveness of training-free weight averaging and interpolation of fine-tuned expert models to produce an improved base model, benefiting from (linear) mode connectivity in models fine-tuned from a single pre-trained checkpoint Izmailov et al. (2018); Ramé et al. (2024); Neyshabur et al. (2020); Frankle et al. (2020); Ainsworth et al. (2023). These insights have been extended into weight-averaged reward models Ramé et al. (2024), policy models Ramé et al. (2024) with spherical interpolation, and KL-constrained RLHF Lin et al. (2024); Liu et al. (2024); Munos et al. (2024); Gorbatovski et al. (2024). Works such as Fisher-Merge Matena & Raffel (2022), TIES Yadav et al. (2023), RegMean Jin et al. (2023), MATS Tam et al. (2024a), DELLA Deep et al. (2024), DARE Yu et al. (2024), Breadcrumbs Davari & Belilovsky (2025), evolutionary merging Akiba et al. (2024) and MagMax Marczak et al. (2024) have explored merging strategies beyond simple interpolation to determine which weights should be merged across expert models. These methods have different benefits for in- and out-of-distribution generalization across domains Tam et al. (2024b), though recently they have been shown to perform similarly at scale Yadav et al. (2024b). Additionally, some works have explored the initialization dimension for effectively merging models (Choshen et al., 2022; Don-Yehiya et al., 2022; Zhou et al., 2024; Marczak et al., 2024). In this work, we propose a unifying framework for temporal merging and conduct the most comprehensive study of this topic to date.

Continual Pretraining extends beyond standard Continual Learning (Prabhu et al., 2023a; Roth et al., 2024b), focusing on large-scale model updates starting from pretrained foundation models Ibrahim et al. (2024); Garg et al. (2024); Roth et al. (2024b); Gui et al. (2024); Prabhu et al. (2023c) and addressing more complex and substantial update tasks Lin et al. (2021); Cai et al. (2021); Liska et al. (2022); Garg et al. (2024); Bornschein et al. (2023); Roth et al. (2024b). There has been limited exploration into using model merging for continual pretraining (Marczak et al., 2024; Alexandrov et al., 2024; Stojanovski et al., 2022; Roth et al., 2024b), as most prior works focus on training strategies including regularization objectives and learning-rate schedules (Roth et al., 2024b; Prabhu et al., 2023b; Garg et al., 2024; Ibrahim et al., 2024; Srivastava et al., 2024; Li et al.; Yıldız et al., 2024; Thede et al., 2024; Ostapenko et al., 2022; Mendieta et al., 2023). We keep the training strategy fixed, and provide an in-depth exploration beyond simple merging techniques.

B Details of the merging methods

Denoting the number of models to merge at timestep t as M_t (with $t = 0$ and $M_t = M$ for standard model merging), we can define these methods as follows:

Weight Averaging Regent's et al. (1996); Wortsman et al. (2022b); Ilharco et al. (2022); Stojanovski et al. (2022); Roth et al. (2024b) simply employs a uniformly weighted, element-wise average over all models $\theta_{t,i}$, resulting in a merge function $f_{\text{merge}}^{\text{WA}}$:

$$\theta_t = \frac{1}{M_t} \sum_i \theta_{t,i}. \tag{2}$$

SLERP Shoemake (1985); Ramé et al. (2024) assumes weights to live on a hypersphere, and consequently conducts interpolation along a curved path connecting weight entries. In particular, for two models $\theta_{t,1}$ and $\theta_{t,2}$ deriving from some base weight θ_{t-1} and the corresponding task vectors $\delta_{t,i} = \theta_{t,i} - \theta_{t-1}$, SLERP with interrpolation weight λ is defined as

$$\theta_t = \theta_{t-1} + \frac{\sin(1-\lambda)\Omega_{1,2}}{\sin \Omega_{1,2}} \cdot \delta_{t,1} + \frac{\sin \lambda \Omega_{1,2}}{\sin \Omega_{1,2}} \cdot \delta_{t,2} \tag{3}$$

with $\Omega_{1,2}$ being the angle between task vectors $\delta_{t,1}$ and $\delta_{t,2}$. We denote the corresponding merge function $f_{\text{merge}}^{\text{SLERP}}$.

Task Arithmetic Ilharco et al. (2023) defines the merge as a function over task vectors $\delta_{t,i} = \theta_{t,i} - \theta_{t-1}$ for each weight $\theta_{t,i}$ fine-tuned from θ_{t-1}. This introduces a simple merge formalism $f_{\text{merge}}^{\text{TA}}$ for weighted parameter averaging with a scale λ:

$$\theta_t = \theta_{t-1} + \lambda \frac{1}{M_t} \sum_i \delta_{t,i} \tag{4}$$

TIES Yadav et al. (2023) builds on the task arithmetic formalism through controlled pruning of task vector entries with low magnitude. Moreover, the sign for each final merged parameter is set based on the sign of the highest total magnitude across the merge candidates. The final update follows basic task arithmetic, only for entries with matching signs. We refer to the respective merge function as $f_{\text{merge}}^{\text{TIES}}$.

DARE Yu et al. (2024) is a similar extension of task arithmetic, but instead of targetted pruning, it randomly zeroes out task vector entries using a random mask $Z_i \sim \text{Bernoulli}(p)$ and masking probability p. Final task vector values for $f_{\text{merge}}^{\text{DARE}}$ are then rescaled based on p:

$$\delta_{t,i}^{\text{DARE}} = \frac{(1 - Z_i)\delta_{t,i}}{1 - p}. \tag{5}$$

Model Stock Jang et al. (2024) provides a geometric extension of simple weight averaging as done in Model Soup Wortsman et al. (2022a) by incorporating base weights θ_{t-1} into the merging process. Given fine-tuned weights $\theta_{t,1}$ and $\theta_{t,2}$, the Model Stock merge $f_{\text{merge}}^{\text{Stock}}$ is defined as follows:

$$\theta_t = \frac{2 \cdot \cos \Omega_{1,2}}{1 + \Omega_{1,2}} \cdot (\theta_{t,2} - \theta_{t,1}) + \left(1 - \frac{2 \cdot \cos \Omega_{1,2}}{1 + \cos \Omega_{1,2}}\right), \tag{6}$$

utilizing angle $\Omega_{1,2}$ between task vectors $\delta_{t,1}$ and $\delta_{t,2}$.

Breadcrumbs Davari & Belilovsky (2025) deploys another variation on task arithmetic for model merging. In particular, for a given task vector $\delta_{t,i}$, extreme left and right tails of the absolute magnitude distribution in $\delta_{t,i}$ are zeroed out with left and right thresholds β and γ. The modified task vectors $\delta_{t,i}^{\text{Bread}}$ are then applied on base weights θ_{t-1} following the task arithmetic setup, and giving $f_{\text{merge}}^{\text{Bread}}$.

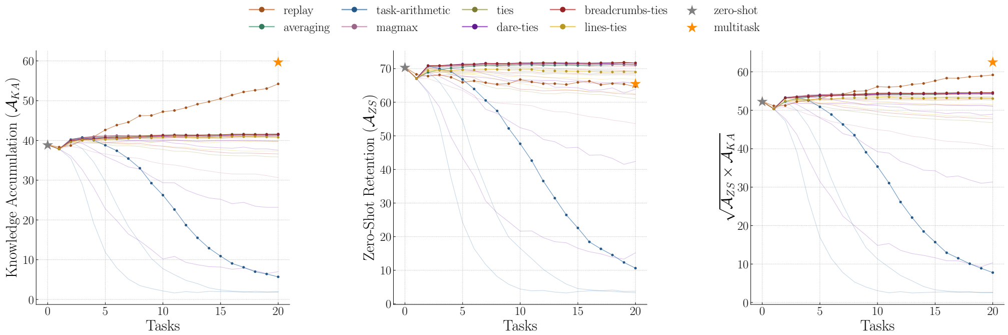


Figure 6: **Offline merging methods struggle with TIME.** All tested merging techniques perform extremely poorly, and are unable to adapt to the temporal setting, underperforming even a simple replay baseline that sequentially trains the base model on task-replayed data.

MagMax Marczak et al. (2024) also uses task vectors—given multiple task vectors $\delta_{t,i}$ (with increments possible along both time t and count axis i), the final task vector δ_t is yielded through maximum magnitude entry selection; copying the largest magnitude entries across all $\{\delta_{t,i}\}$ into δ_t, giving $f^{\text{Max}}_{\text{merge}}$.

LiNeS Wang et al. (2024a), for Layer-increasing Network Scaling, scales weight updates based on their respective layer depth enabling early layers to remain close to original pretraining weights (cf. Neyshabur et al. (2020)). Given task vectors $\delta_{t,i}$, now broken down across model layers $\delta^l_{t,i}$ with $l \in [1, ..., L]$ and L the number of layers, LiNeS follows the base task arithmetic merging formalism, but updates task vectors as

$$\delta^{\text{LiNeS}}_{t,i} = \texttt{concat}\left(\lambda^{l=1}\delta^{l=1}_{t,i}, ..., \lambda^{l=L}\delta^{l=L}_{t,i}\right) \tag{7}$$

with layer-scaled interpolation weights $\lambda^l = \alpha + \beta\frac{l-1}{L-1}$ and hyperparameters α, β, giving $f^{\text{Lines}}_{\text{merge}}$.

C Plotting Style

Across TIME, we utilize a common plotting style to visualize our results—with three base subplots (see for *e.g.*, Fig. 5):

- Knowledge Accumulation ($\mathcal{A}_{KA}$) versus number of tasks over time. In this plot, a gray star indicates the base-weight zero-shot performance on adaptation datasets. An orange star indicates an upper bound achieved through jointly training on all the data at once, with no separation over time.
- Zero-Shot Retention ($\mathcal{A}_{ZS}$) versus number of tasks over time. Similar to $\mathcal{A}_{KA}$ versus tasks, this plot visualizes merging results for TIME-variants, but measuring performance on withheld evaluation datasets. Again, gray and orange star indicate base and joint training lower and upper bounds, respectively.
- Finally, we also aggregate both previous plots into one showcasing the progression of merged performance geometric mean $\sqrt{\mathcal{A}_{ZS} \times \mathcal{A}_{KA}}$ over time; utilizing the same star indication as in the previous subplots.

The only deviation from this plotting style is Fig. 5. The left panel visualizes the trajectory across tasks in the $\mathcal{A}_{KA}$ - $\mathcal{A}_{ZS}$ space. Here, full-colored stars reference base model performance and hollow stars the corresponding joint training upper bounds. The right panel shows the geometric mean of $\mathcal{A}_{KA}$ and $\mathcal{A}_{ZS}$ at the end of the last task for different compute budgets.

Finally, several plots such as Figs. 4, 6 and 7 show the extensive scale of our experiments through background visualizations of sub-optimal hyperparameter choices in lighter colors (as opposed to the optimal choices using darker coloring). This plotting style is loosely inspired by Beyer et al. (2022).

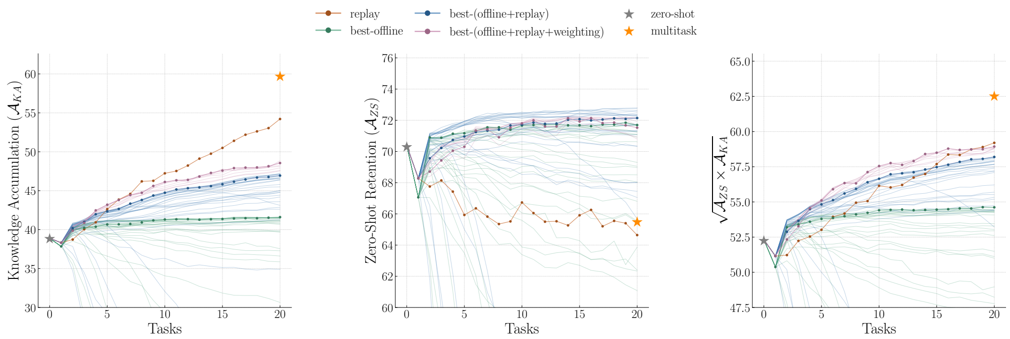


Figure 7: **Improving *offline* merging.** We identify two simple methods for adapting offline-merging methods to the temporal setting: (1) replaying data from previous tasks (best-(offline+replay)) and (2) recency-biased weighting of task checkpoints (best-(offline+replay+weighting)). With these method improvements, offline merging methods can match the replay baseline.

D Additional Offline Merging Experiments

D.1 Replay and Time-Weighting

In this section, we analyze extensions to offline methods that can help close the gap to the replay baseline. As the continual fine-tuning baseline *replays* on past data from all previous tasks while training at the current task t, can this task data-mixing also help offline merging methods?

Data replaying improves offline merging. Since offline methods operate entirely under a task-independent assumption, they fail to capture any temporal dependencies. Fig. 7 shows that simply applying data-replay on top of standard offline merging leads to significant boosts in the overall performance. For instance, best-(offline+replay) achieves 58.2% compared to best-offline at 54.6%, bringing it closer to the replay baseline. However, a notable performance gap remains, with best-(offline+replay) at 58.2% falling short of replay at 59.1%.

Recency-biased weighting helps. Next, unlike in standard *offline* averaging, where all task checkpoints are weighted uniformly, we impose temporal ordering via non-uniform weighting for offline merging. We explore several recency-biased, non-uniform weighting schemes, assigning higher weights to more recent tasks to account for the temporal nature of the setting.

We explore various discounting schemes: logarithmic, quadratic, exponential, and cubic, applied to the best offline merge replay method from the previous experiment (please refer to the supplementary for details). As shown in Fig. 7, these schemes improve performance, with best-(offline+replay+weighting) reaching 58.9%, yet still falling slightly short of the replay baseline at 59.1%. These results provide strong evidence that accounting for the new temporal axis is crucial for effective temporal model merging, even when implemented as an extension of offline merging. **Key takeaway:** accounting for the time aspect is crucial for effective temporal model merging, even as an extension on top of standard offline merging. Still, a small gap to the simple replay baseline remains.

D.2 Reversed Non-Uniform Weighting Schemes

In Fig. 7, we found that a simple yet effective method for boosting the performance of offline merging methods is recency-biased non-uniform weighting, i.e. giving larger weights to more recent checkpoints while merging. Here, we ask the question—what if we reversed the weighting schemes such that we give larger weights to older task checkpoints? From Fig. 8, we indeed observe that such a reverse strategy performs worse than the best recency-biased weighting schemes, since the knowledge accumulation ability is hampered by giving more emphasis to older tasks. However, note that such a sub-optimal reverse weighting strategy is still better than the pure offline merging strategy with *no replay*. This helps further ablate the exact importance of *replay* and *non-uniform weighting* for improving pure offline-merging techniques in the presence of the time axis.

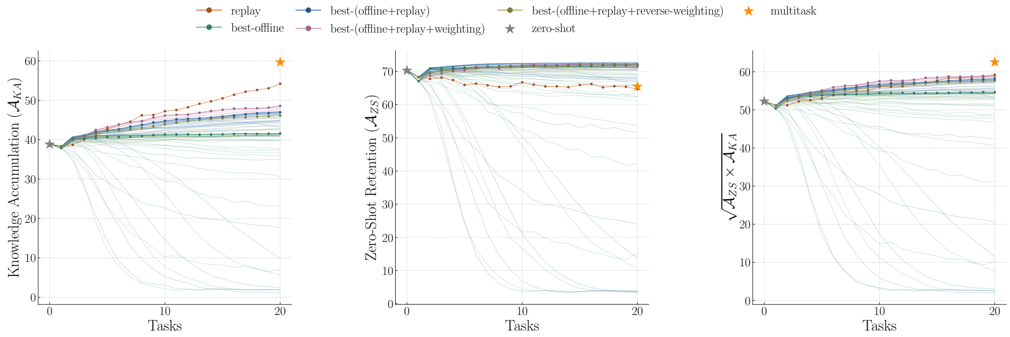


Figure 8: **Effect of reverse-weighting for offline merging techniques.** We find that reversing the weighting scheme that yielded consistent boosts from Fig. 7 is sub-optimal—indeed, it performs worse than the offline merging with replay methods.

E Additional EMA Experiments

E.1 Tasks as Datasets

In the main text, we presented all results using a data stream that randomly mixes concepts from different datasets into a coherent set of tasks—following the *random* data-stream in Roth et al. (2024b). Here, we relax this constraint and re-run our experiments using individual datasets as tasks, consistent with the standard model merging literature (Ilharco et al., 2022; 2023; Yadav et al., 2023). Specifically, we use the *dataset-incremental* stream from Roth et al. (2024b). Even in this setup, we reproduce our main findings. In Fig. 9, we confirm the results from Fig. 6, showing that all offline merging techniques perform poorly when exposed to the axis of time, failing to even match the performance of a simple continual fine-tuning *replay* baseline. Additionally, in Fig. 10, we corroborate the results from Fig. 3, demonstrating that the *best-in*-TIME method remains the most effective temporal model merging approach. We also confirm that the choice of model merging technique is far less critical for temporal model merging than the initialization and deployment strategies.

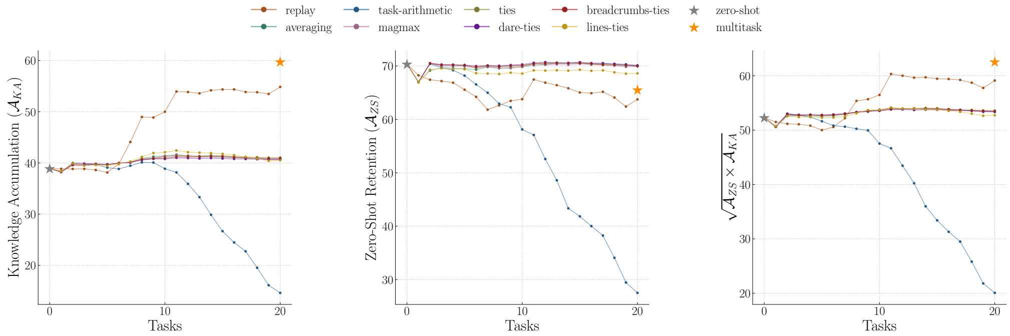


Figure 9: **Offline merging techniques still struggle in the tasks-as-datasets setting.** Switching from the *random* data-stream (Fig. 6 in the main paper) to the *dataset-incremental* stream, which aligns more closely with the standard multi-task merging literature setups, reveals that offline merging techniques still severely underperform compared to the simple *replay* baseline.

E.2 Longer Task Sequences

To test the robustness of our findings in Sec. 3.2, we repeat the experiment shown in Fig. 3 on a longer sequence with the number of tasks $T = 50$ (Fig. 11). For 50 tasks, *Best-in*-TIME still strikes the optimal balance between knowledge accumulation and zero-shot retention. One notable difference with respect to Fig. 3 is the large initial advantage of the zero-shot initialization strategy combined with the EMA deployment strategy. When the learning horizon is further extended to 100

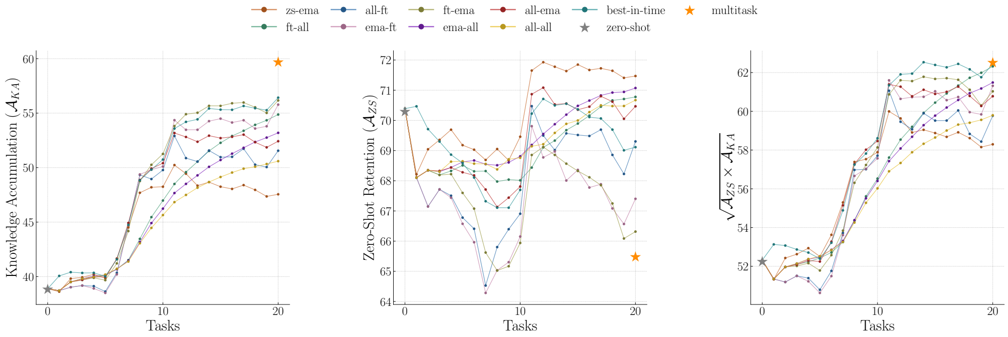


Figure 10: **Dataset-Incremental TIME Exploration.** We replicate the results from Fig. 3 using the dataset-incremental stream instead of the random stream. The main takeaways remain unchanged: initialization and deployment strategies primarily determine temporal merging performance, and the EMA-averaging initialization and deployment strategy utilized in *Best-in-*TIME is the best approach.

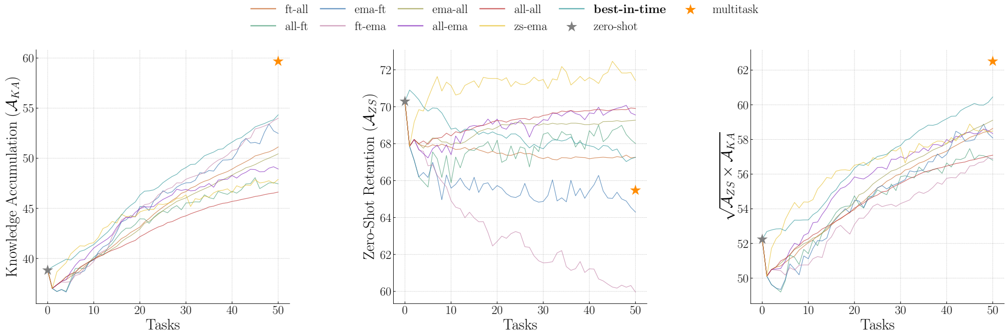


Figure 11: **A long journey through TIME.** We compare all valid combinations of initialization and deployment protocols on a longer sequence of 50 tasks. *Best-in-*TIME remains the best in balancing knowledge accumulation and zero-shot retention.

tasks, this initial advantage is maintained, establishing the zero-shot initialization approach as the best-performing method, as shown in Fig. 12. Although the double EMA variant surpasses zero-shot initialization in knowledge accumulation, its poor retention relegates it to third place on the combined metric. In this exploration we re-use the optimal interpolation weight from the 20 task scenario, which may no longer be ideal for longer horizons, as it directly influences the balance between knowledge accumulation and zero-shot retention.

E.3 Variance Analysis across Runs

To put our results from Sec. 3.3 in perspective, we quantify the variance across runs for a single merging method. Specifically, we run *Best-in-*TIME three times and show the mean and standard deviation across runs in Fig. 13. Comparing this to Fig. 4 reveals that the best results for different methods fall within the standard deviation of multiple runs of the same method. In particular, for the last task, the standard deviation of the geometric mean of knowledge accumulation and zero-shot retention is 0.96.

F Hyperparameter Details

In an effort to remove any confounding factors, we conduct an extensive hyperparameter sweep, to the best of our abilities, for each individual merging technique for Figs. 4, 6 and 7. We list the hyperparameter ranges swept over for each technique below:

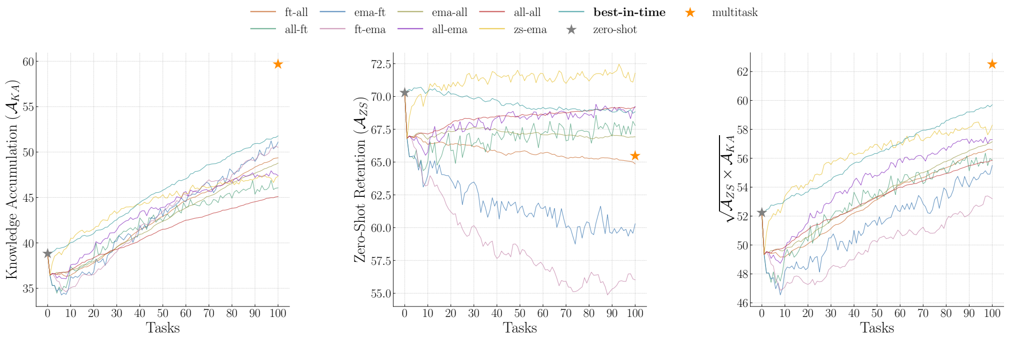


Figure 12: **An even longer journey through TIME.** We compare all valid combinations of initialization and deployment protocols on a longer sequence of 100 tasks. *Best-in*-TIME still remains the best approach balancing knowledge accumulation and retention, measured as the geometric mean of the two metrics in the right-most figure.

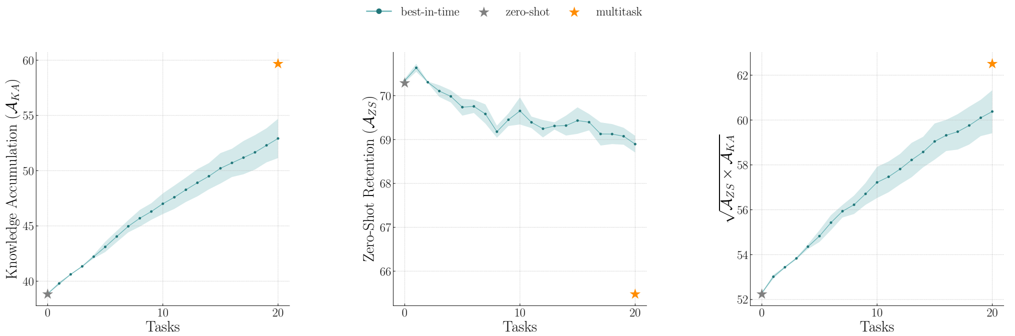


Figure 13: The mean and standard deviation across three runs of *Best-in*-TIME.

- **Weight Averaging.** For the offline merging, we use a standard merging coefficient of $\frac{1}{N}$, where N is the number of task checkpoints to merge.
- **SLERP.** In SLERP, as we can only merge two checkpoints at a time, we sweep over the following weight-coefficients: {0.1,0.3,0.5,0.7,0.9}.
- **Task-Arithmetic.** We sweep over the scaling factor: {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0}
- **TIES.** We sweep over the scaling factor: {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0} and the pruning-fraction: {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0}.
- **DARE-TIES.** We sweep over the scaling factor: {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0} and the pruning-fraction: {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0}.
- **Breadcrumbs-TIES.** We sweep over the scaling factor: {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0} and the pruning-fraction: {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0}.
- **MagMax.** We sweep over the scaling factor: {0.2,0.4,0.8,1.0}.
- **LiNeS-TIES.** We keep α fixed to 0.5, and sweep β: {0.2,0.5,0.8} and prune-fraction: {0.2,0.5,0.8} as recommended in the original paper (Wang et al., 2024a).