RECALL: REpresentation-aligned Catastrophic-forgetting ALLeviation via Hierarchical Model Merging

Bowen Wang[1,§], **Haiyuan Wan**[1,2§], **Liwen Shi**[1], **Chen Yang**[5], **Peng He**[1,2], **Yue Ma**[1,2], **Haochen Han**[2]
Wenhao Li[4], **Tiao Tan**[1], **Yongjian Li**[6], **Fangming Liu**[2,3,†], **Yifan Gong**[2,†], **Sheng Zhang**[1,†]
[1]Shenzhen International Graduate School, Tsinghua University
[2]Peng Cheng Laboratory, [3]Huazhong University of Science and Technology
[4]Xiamen University, [5]The Hong Kong University of Science and Technology, Guangzhou
[6]School of Biomedical Engineering, Tsinghua University
Correspondence: {wangbw23, wanhy24}@mails.tsinghua.edu.cn, fangminghk@gmail.com
gongyf@pcl.ac.cn, zhang_sh@mail.tsinghua.edu.cn

Abstract

We unveil that internal representations in large language models (LLMs) serve as reliable proxies of learned knowledge and propose **RECALL**[i], a novel representation-aware model merging framework for continual learning without access to historical data. RECALL computes inter-model similarity from layer-wise hidden representations over clustered typical samples, and performs adaptive, hierarchical parameter fusion to align knowledge across models. This design enables the preservation of domain-general features in shallow layers while allowing task-specific adaptation in deeper layers. Unlike prior methods that require task labels or incur performance trade-offs, RECALL achieves seamless multi-domain knowledge fusion and strong resistance to catastrophic forgetting. Extensive experiments across five NLP tasks and multiple continual learning scenarios show that RECALL outperforms baselines in both knowledge retention and generalization, providing a scalable and data-free solution for evolving LLMs.

1 Introduction

Large language models (LLMs) have achieved impressive advances across tasks like question answering, text generation, and mathematical reasoning, powering applications such as chatbots, AI business agents, and recommendation systems (Devlin, 2018; Brown et al., 2020; Touvron et al., 2023; Raffel et al., 2020). They are typically trained through unsupervised pre-training on large corpora, followed by supervised fine-tuning (SFT) on task-specific or domain-specific data (Brown et al., 2020; Touvron et al., 2023; Wei et al., 2021; Ouyang et al., 2022). However, LLMs remain susceptible to *catastrophic forgetting* (CF), where distribution shifts during training lead to parameter updates that overwrite prior knowledge (Mccloskey and Cohen, 1989; Kirkpatrick et al., 2016; Li and Hoiem, 2018). As LLMs are increasingly applied in continual and multi-domain settings, mitigating CF is essential to maintain both specialization and generalization (Brown et al., 2020; Wei et al., 2021; Achiam et al., 2023; Doimo et al., 2024).

As illustrated in Figure 1, previous approaches addressing CF generally fall into two categories, each with distinct strengths and limitations:

1) Data-based methods preserve past knowledge by revisiting stored samples from previous tasks during training on new tasks (Lopez-Paz and Ranzato, 2017; Rebuffi et al., 2016; Romanov et al., 2019; Isele and Cosgun, 2018). These methods are effective in retaining task-specific information by directly exposing the model to prior data. However, they require access to historical samples, which may be impractical due to storage constraints or privacy concerns in real-world scenarios.

2) Model-based methods constrain model updates or isolate task-specific knowledge via regularization (Huang et al., 2021; Kirkpatrick et al., 2016; Li and Hoiem, 2018; Wang et al., 2023) or architecture adaptation (Rusu et al., 2016; Fernando et al., 2017; Tian et al., 2024). These approaches enable continual learning without relying on past data, offering better scalability in privacy-sensitive settings. Nonetheless, they often operate within limited optimization spaces and struggle to preserve performance across diverse tasks. Additionally, they may depend on explicit task identifiers and increase model complexity over time.

To overcome the limitations of existing continual learning approaches, we aim to combine the strengths of both data-based and model-based methods: retaining prior knowledge without relying on stored data, while enabling flexible model adaptation across tasks.

However, without access to historical data, it becomes difficult to assess what knowledge should

[§]Equal contribution
[†]Corresponding authors
[i]https://github.com/bw-wang19/RECALL

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 16381–16395
November 4-9, 2025

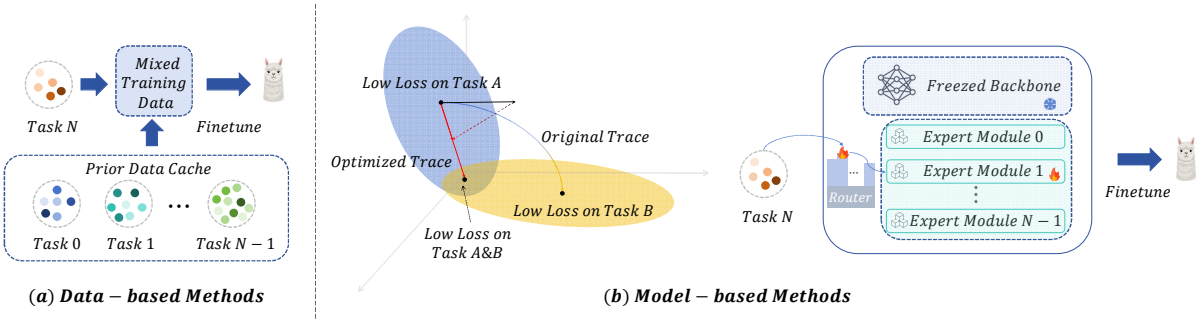

Figure 1: A taxonomy of previous approaches to catastrophic forgetting. Data-based methods (a) rely on stored samples from previous tasks, which are replayed alongside new data during fine-tuning. Model-based methods (b) mitigate forgetting by either constraining parameter updates or isolating task-specific knowledge. In (b), the left side illustrates regularization-based methods that optimize model parameters within the intersection of low-loss regions for both old and new tasks (e.g., Task A and Task B), instead of strictly minimizing the loss on the new task. This encourages a more stable update trajectory that retains previously learned knowledge while adapting to new tasks.

be preserved; and without explicit task boundaries, it is unclear how to guide model updates in a structured and generalizable manner. This raises a core challenge: **how can we identify and preserve useful task knowledge across models in a data-free and task-agnostic way?**

In addressing this question, we observe that internal representations, which reflect how models encode and process inputs, can serve as reliable proxies for their learned knowledge. These representations are inherently shaped by both model architecture and training objectives, making them well-suited for comparing and aligning knowledge across models without requiring access to raw data or task labels.

Motivated by this insight and recent advances in model merging (Xiao et al., 2023; Wortsman et al., 2022; Jiang et al., 2023), we propose a novel representation-aware model merging strategy that addresses both data availability and optimization flexibility. Our method computes inter-model similarities based on intermediate representations and uses them to guide adaptive, layer-wise parameter merging. By avoiding raw data, we circumvent privacy and accessibility concerns, while our fine-grained integration expands the optimization space beyond traditional methods and enables more effective knowledge fusion.

Our main contributions are summarized as follows:

- We propose a novel representation-aware model merging framework to address catastrophic forgetting, by leveraging intermediate representations to guide parameter fusion without relying on raw data or explicit task boundaries.
- Our method generalizes to the merging of multiple expert models fine-tuned on different domains, enabling effective multi-domain capability fusion through weighted representation alignment.
- We further demonstrate that the proposed framework can be applied to traditional continual learning benchmarks, including sequential fine-tuning scenarios, achieving strong performance without task-specific modifications.
- Extensive experiments across multiple datasets and benchmarks validate the effectiveness and generality of our approach, showing consistent improvements in knowledge retention and transferability.

2 Empirical Observations of Representation Dynamics in LLMs

Prior studies have shown that different layers of large language models encode distinct types of linguistic and semantic information (Tenney et al., 2019; Starace et al., 2023). Building on this, we analyze hidden representations from transformer layers to examine how they evolve within a model and diverge across models fine-tuned on different tasks.

2.1 Layer-wise Representation Shift

We first investigate how internal representations evolve across layers within a single model for a fixed input batch. Specifically, we compute the average RBF kernel similarity between adjacent layers' hidden states. The similarity scores exhibit a non-monotonic pattern, with noticeable drops

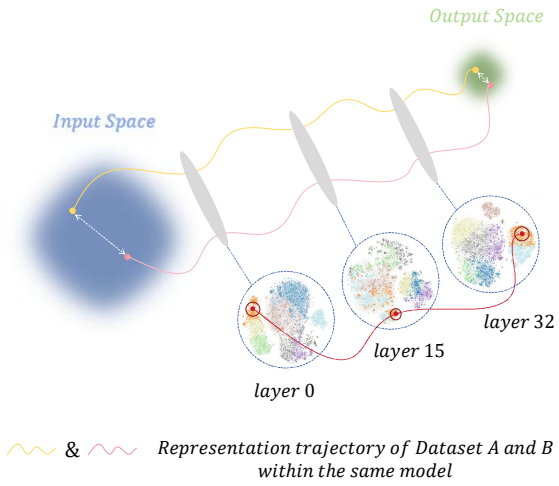

Figure 2: Illustration of representation transformation across layers within a single LLM. The input progresses through a sequence of transformations, and the corresponding hidden states (shown for layers 0, 15, and 32) exhibit distinct structural patterns in the representation space, highlighting the non-uniform nature of internal dynamics.

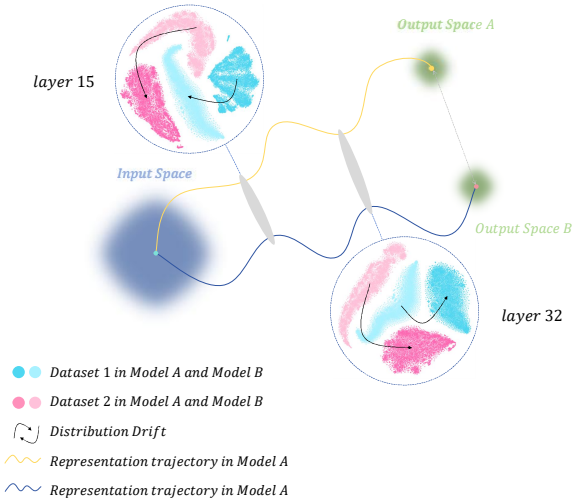

Figure 3: Visualization of representation drift between two models fine-tuned on different tasks (SST2 vs. RACE). Despite sharing the same input, their hidden states evolve along different trajectories and form distinct clustering patterns, especially in deeper layers.

in both early and late layers. This indicates that the transformation of representations varies significantly across the network (see Appendix D for details). In addition, as shown in Figure 2, visualization through clustering and dimensionality reduction techniques shows that hidden states at different layers form distinct structural patterns in the representation space.

This layer-wise variation suggests that each layer contributes differently to the model's behavior. As a result, treating all layers uniformly during model merging—such as through naive parameter averaging—may overlook the unique functional roles of different layers and lead to suboptimal integration.

2.2 Specialization-induced Model Divergence

We next examine how internal representations diverge across models that share the same architecture and initialization but have been fine-tuned on different tasks. Using the same input batch, we extract hidden states from each model and compute the average layer-wise RBF kernel similarity between them.

We observe that lower-layer representations remain relatively consistent, while deeper layers diverge significantly across tasks—a trend highlighted by the model-wise similarity curves in Appendix D.

To further illustrate this phenomenon, Figure 3 visualizes the hidden states from two task-specific models. Despite processing the same inputs, their hidden states evolve along different trajectories and form distinct clustering structures, reinforcing the view that fine-tuning induces semantic specialization in deeper layers.

These results suggest that naive parameter merging, especially in upper layers, may introduce semantic inconsistency or destructive interference if such representational misalignment is ignored.

3 RECALL: REpresentation-aligned Catastrophic-forgetting ALLeviation

In Section 2, we analyze the characteristics of the data representation across models and layers through experimental observations, and illustrate that the knowledge of a model is closely related to its data representation. And previous works (Wortsman et al., 2022; Xiao et al., 2023) have nicely illustrated that knowledge fusion and continual learning do not necessarily require a fine-tuning stage such as knowledge distillation. Model merging can also directly and effectively achieve the goal.

Therefore, inspired by those observations, we propose RECALL in this section, which performs layer-wise model merging by comparing the similarities of data representations between different models, so as to achieve representation alignment. As illustrated in Figure 4, RECALL effectively enhances LLM's abilities in multiple domains and tasks, and mitigates catastrophic forgetting.

As a prerequisite condition, we have the source

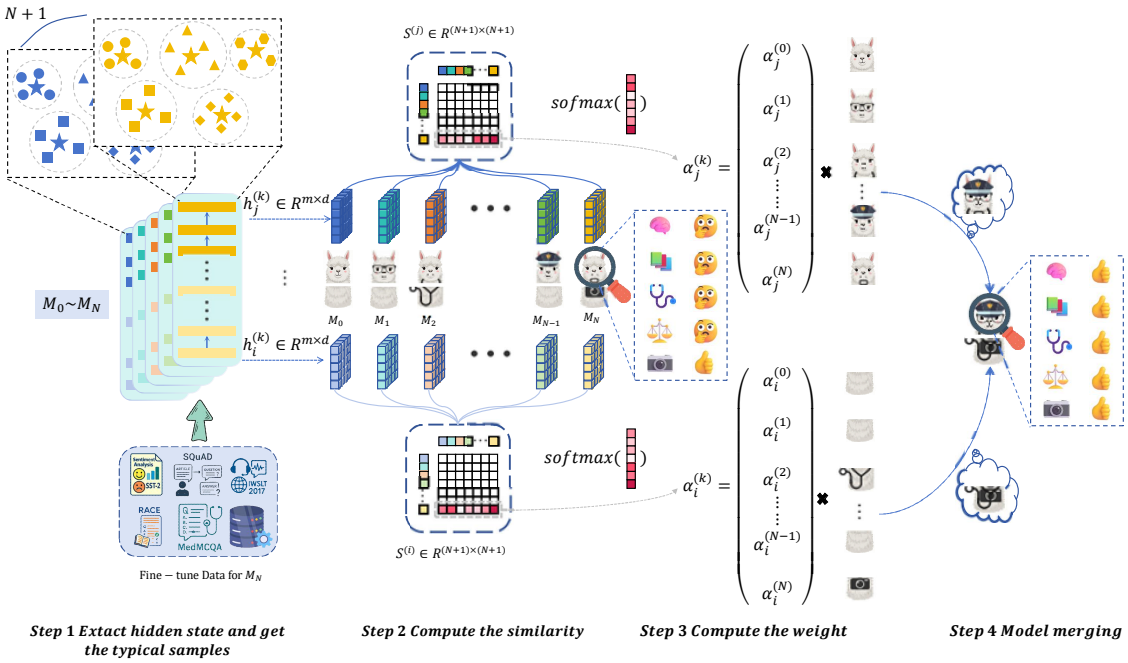

Figure 4: Illustration of **RECALL**, our proposed representation-aware model merging framework. The pipeline consists of four stages: (1) extract hidden states from typical samples using the newly fine-tuned model M_N, (2) compute the pairwise representational similarities across all models (including M_N), (3) derive layer-wise adaptive weights based on similarity scores via softmax, and (4) perform hierarchical parameter merging guided by the computed weights. This process enables effective knowledge fusion across models while preserving task-specific features.

model M_0 and multiple homologous expert models $M_1, M_2, \cdots, M_{N-1}$, which have the same architecture but different parameters with M_0. On the new task T_N, we obtain the new model M_N by fine-tuning M_0 from the dataset D_N. Then, we select m typical samples $D_{type} = \{d_1, d_2, \cdots, d_m\} \subset D_N$ through a clustering algorithm. For each $d_k \in D_{type}$, we extract its representations on models $M_0 \sim M_N$, analyze the differences between M_N and other models in semantic and syntactic knowledge through the similarities between data representations. Finally, we perform hierarchical model merging for knowledge fusion.

3.1 Data Representation

Representation Extraction. We extract the hidden states of layer n of M_p, which is formulated as: $\mathbf{R}_n = (\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_L) \in \mathbb{R}^{L \times E}$, where L is the number of input tokens, E is the dimension of embedding vectors, and $\mathbf{r}_i$ indicates the embedding of the ith token. Referring to the practice of most embedding models (Reimers and Gurevych, 2019; Xiao et al., 2023), we average the hidden states by token to obtain the representation vector: $\mathbf{r} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{r}_i \in \mathbb{R}^E$.

Typical Dataset Selection. Our approach does not place restrictions on the composition of datasets, which means that samples from multiple domains and tasks may be included in D_N. Therefore, we cluster all data representations of D_N, and select m samples which are nearest to the m cluster centers $C = \{c_1, c_2, \cdots, c_m\}$ to form the typical dataset $D_{type} = \{d_{t_1}, d_{t_2}, \cdots, d_{t_m}\} \subset D_N$. For $k \in [1, m]$: $d_{t_k} = \arg\min_{d_i \in D_N} ||d_i - c_k||_2$, in which c_k is the kth cluster center clustered by Kmeans. In order to reduce the number of samples needed to perform forward inference for data representation analysis, we use D_{type} as the representative of D_N to analyze the knowledge difference of models.

3.2 Similarity Calculation

For each sample d_k in the typical dataset D_{type} and each model M_p, its data representation at layer i is $\mathbf{r}_i^{p,k} \in \mathbb{R}^E$. As mentioned above, we measure the difference in knowledge between models by data representations. Specifically, as we select typical samples by Kmeans which is closely related to the norm distance, RBF kernel function is adopted to measure similarity between representation vectors,

and we calculate the algebraic average of similarities of all m samples in D_{type} as the overall similarity. We also carefully discuss the differences using different similarity measures and do experiments to compare them; results and discussions are detailed in Appendix F.

The similarity between M_p and M_q on layer i is formulated as:

$$S_i^{p,q} = \frac{1}{m}\sum_{k=1}^{m}\exp\left(-\frac{||\boldsymbol{r}_i^{p,k} - \boldsymbol{r}_i^{q,k}||_2^2}{2\sigma^2}\right), \quad (1)$$

in which σ is a scaling factor.

3.3 Hierarchical Merging

For later narration, we summarize here the paradigm approach to model merging. Current model merging is essentially a linear interpolation of the model parameters, which means for each parameter θ in the model and merging weight w, we have:

$$\theta^* = \sum_{i=1}^{N} w_i\theta_i = \boldsymbol{w}^T\boldsymbol{\theta}, \quad (2)$$

in which $\boldsymbol{\theta}$ is the vector concatenated by parameter θ of different models, and $\boldsymbol{w}$ is the vector of corresponding weights.

Furthermore, Eq 2 can be easily extended to the case that a group of parameters correspond to the same weights. We can compute linear interpolations of multiple parameters at once via the inner product operation like the following equation,

$$\boldsymbol{\theta} = \begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} = \begin{bmatrix}\boldsymbol{w}^T\boldsymbol{\theta}_1\\ \boldsymbol{w}^T\boldsymbol{\theta}_2\end{bmatrix} = \begin{bmatrix}\boldsymbol{\theta}_1^T\\ \boldsymbol{\theta}_2^T\end{bmatrix}\boldsymbol{w}. \quad (3)$$

In Section 3.2, we present the similarity metric to measure the similarities of data representation between models. To align their representations, we use Softmax to normalize representation similarities as merging weights. Then the weight of M_q in layer i is as follows:

$$w_i^q = \frac{\exp S_i^{n,q}}{\sum_{p=0}^{N}\exp S_i^{n,p}}. \quad (4)$$

Therefore, we provide representation-aligned merging method for one layer:

$$\boldsymbol{\theta}_i^* = \sum_{q=0}^{N} w_i^q\boldsymbol{\theta}_i^q = \left[\boldsymbol{\theta}_i^1,\ \boldsymbol{\theta}_i^2,\ \cdots,\ \boldsymbol{\theta}_i^N\right]\boldsymbol{w}_i = \boldsymbol{\Theta}_i^T\boldsymbol{w}_i, \quad (5)$$

where $\boldsymbol{\theta}_i$ denote the model's parameters of layer i.

According to Eq 3, 5, we perform hierarchical model merging layer by layer:

$$\boldsymbol{\theta}^* = \begin{bmatrix}\boldsymbol{\theta}_1^*\\ \boldsymbol{\theta}_2^*\\ \vdots\\ \boldsymbol{\theta}_L^*\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Theta}_1^T\boldsymbol{w}_1\\ \boldsymbol{\Theta}_2^T\boldsymbol{w}_2\\ \vdots\\ \boldsymbol{\Theta}_L^T\boldsymbol{w}_L\end{bmatrix} = diag(\boldsymbol{\Theta}^T\boldsymbol{w}), \quad (6)$$

in which $\boldsymbol{\theta}^*$ is the parameter vector of the final merging model. $\boldsymbol{\Theta} = [\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \cdots, \boldsymbol{\Theta}_L]$ is the parameter matrix of $M_{0\sim N}$, and $\boldsymbol{w}^T = [\boldsymbol{w}_1^T, \boldsymbol{w}_2^T, \cdots, \boldsymbol{w}_N^T]$ is the corresponding weight matrix.

As mentioned above, our method enhances the abilities of LLM in multi-domains and resists catastrophic forgetting by performing independent weight calculation between layers and hierarchical merging operations. The detailed procedure of RECALL is presented in Algorithm 1 in the Appendix G, and meanwhile we provide an analysis of runtime, memory usage, and scalability.

4 Experiments

In this section, we will provide a detailed introduction to our implementation and the results of experiments, which are mainly composed of three main parts: Experimental Setup, Different Merging Scenarios, and Sequential Fine-tuning Scenario. Furthermore, we summarize and analyze the results of these experiments, which strongly prove the superiority of our method.

4.1 Experimental Setup

Datasets. Considering a challenging experimental setup in knowledge fusion and continual learning, we selected 5 datasets as targets from multiple domains and tasks, including text classification, single-choice questions, and text generation, which are SST-2(Socher et al., 2013), SQuAD2.0(Rajpurkar et al., 2016, 2018), MedMCQA(Pal et al., 2022), RACE(Lai et al., 2017) and IWSLT2017(Cettolo et al., 2017). Since these datasets come from different tasks and have different formats, in order to adapt our method, we unify them into **QA** format by constructing prompts. Examples of the prompts are accessible in Appendix C.

Baseline.(1) **SFT only**: directly fine-tunes the base model on a single downstream task without considering any cross-task interactions or parameter sharing. (2) **Avg.**(Wortsman et al., 2022): averaging their parameters without any alignment or adjustment. (3) **DARE**(Yu et al., 2023): flexible strategy to combine with other baselines(Average or Task Vector method) and random dropout parameters. (4) **LM-Cocktail**(Xiao et al., 2023): merges models by comparing loss on validation set. (5) **Task Vector**(Ilharco et al., 2022): computes the difference between the base model and each fine-

tuned model to perform add, subtraction, or interpolation to construct new task behaviors. (6) **EWC**(Kirkpatrick et al., 2016): introduces a regularization term based on the Fisher Information Matrix to prevent forgetting.

We selected the Llama-2-7B-chat[i](Touvron et al., 2023) as the base model for fine-tuning and weight merging on 8 NVIDIA V100 GPUs, and LoRA(Hu et al., 2022) is deployed for the fine-tuning pipeline. The implementation details of the fine-tuning and evaluation pipeline are provided in Appendix B. All implementation details are supplied in Appendix A.

In experiments of comparing with other baselines, our method always uses the same setting: we select 20 typical samples for each layer by the clustering algorithm, and those samples are concatenated to form the typical dataset. Same as Eq 1, we adopt the RBF kernel function as the similarity, of which the scale factor σ is set to 1.0. Then we segment and calculate weights for each layer of the model to merge them independently(taking Llama2-7b-chat as an example, the model will have 33 different groups of merging weights).

4.2 Performance of RECALL in Different Merging Scenarios

Firstly, we fine-tune the base model on the above 5 datasets to obtain five corresponding expert models. We then set up two different scenarios depending on the number of models used in the merging, which will be illustrated in the next two subsections.

4.2.1 Single Fine-tuned Model Merging

In this study, we consider the case of merging using a single fine-tuned model and its base model. With access to the training datasets for both models, we conduct comprehensive experiments to evaluate the proposed approach across different datasets. Our experiments compare the performance of several baselines using different datasets, and the results are presented in Table 1.

We draw the following observations from Table 1: Our method RECALL consistently outperforms all baselines across diverse settings, achieving the highest average performance (**45.00**) and the best generalization to unseen tasks (**38.92**, **+7.86%** over the best baseline). It maintains top-tier results across all fine-tuning sources and excels in challenging domains such as MEDMCQA and IWSLT2017-EN-FR, demonstrating both robustness and transferability. These results underscore the effectiveness of leveraging representational similarity for model merging and motivate the extension to more complex multi-source integration scenarios.

4.2.2 Multiple Fine-tuned Models Merging

To simulate a more complex knowledge fusion setting, we simultaneously merge five task-specific expert models. As shown in Table 2, we consider two configurations: merging with and without the inclusion of the base model.

From Table 2, we observe: RECALL achieves the best overall performance in both settings, with or without the base model, reaching averages of **56.93** and **62.83**, respectively. Notably, it outperforms all other methods even without relying on the base model, demonstrating a strong capability in fusing knowledge from multiple fine-tuned experts. These results highlight the advantage of representation-aware merging over both parameter averaging and task-vector-based baselines, and demonstrate that RECALL is not only effective for single-expert scenarios but also scalable to multi-expert merging, showing robust performance in both knowledge preservation and generalization without requiring access to training data.

To more effectively demonstrate the effectiveness of RECALL across multiple models, we conducted supplementary experiments using Qwen2-7B-Instruct[ii]. As shown in Table 3, RECALL also demonstrated a strong knowledge fusion ability and the ability to resist catastrophic forgetting in this test.

4.3 Sequential Fine-tuning Scenario

To further assess the effectiveness of RECALL in realistic continual learning settings, we conduct sequential fine-tuning experiments across five tasks introduced in a fixed order. After training on each new task, the current model is merged with the previously accumulated one using different strategies. We compare RECALL against two baselines: standard LoRA-based fine-tuning (LoRA SFT) and Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2016).

Figure 5 illustrates the forward forgetting curves over the task sequence, where the y-axis indicates model performance on the current task immediately

[i]https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

[ii]https://huggingface.co/Qwen/Qwen2-7B-Instruct

Fine-tuned on	Method	Datasets					Avg from all tasks	Avg from unseen tasks
		SST-2	SQuAD2.0	IWSLT2017-en-fr	RACE	MedMCQA		
SST-2	SFT only	95.76	31.68	13.28	44.71	32.32	43.55	30.50
	Avg	94.95	5.21	12.32	32.75	34.28	35.90	21.14
	DARE+Avg	95.07	8.75	11.68	50.44	35.14	40.22	26.50
	LM-Cocktail	95.76	25.54	11.88	32.55	34.26	40.00	26.06
	RECALL(Our)	94.50	30.72	12.08	47.44	34.90	**43.93**	**31.29**
SQuAD2.0	SFT only	86.81	85.46	21.28	48.78	32.61	**55.00**	47.37
	Avg	89.11	80.92	18.28	31.58	34.23	50.82	43.3
	DARE+Avg	89.11	78.67	19.59	50.52	34.69	54.52	**48.48**
	LM-Cocktail	79.36	84.46	18.38	42.24	32.66	51.42	43.16
	RECALL(Our)	86.19	84.87	18.20	49.50	34.34	54.62	47.06
IWSLT2017-en-fr	SFT only	82.68	10.72	45.33	29.39	33.21	40.27	39.00
	Avg	89.91	4.35	42.01	32.85	35.45	40.91	40.64
	DARE+Avg	89.33	10.45	41.63	44.6	35.84	44.37	45.06
	LM-Cocktail	89.91	5.23	43.13	30.08	35.07	40.68	40.07
	RECALL(Our)	89.56	10.55	43.09	48.73	34.43	**45.27**	**45.82**
RACE	SFT only	18.23	50.64	19.06	85.71	39.68	42.66	31.90
	Avg	47.36	14.80	22.58	73.47	34.66	38.57	29.85
	DARE+Avg	29.13	50.05	19.55	78.68	34.97	42.48	33.42
	LM-Cocktail	30.39	51.24	23.02	82.31	37.29	**44.85**	**35.49**
	RECALL(Our)	34.93	40.27	23.12	79.31	36.96	42.92	33.82
MedMCQA	SFT only	9.91	6.58	18.22	31.76	45.54	22.40	16.62
	Avg	0.11	5.97	15.34	23.17	43.25	17.57	11.15
	DARE+Avg	24.36	11.99	14.04	57.46	42.86	30.14	26.96
	LM-Cocktail	17.58	12.61	14.07	24.89	44.18	22.67	17.29
	RECALL(Our)	70.32	18.58	13.86	43.77	44.82	**38.27**	**36.63**
All Average	SFT only	58.68	37.02	23.43	48.07	36.67	40.77	33.08
	Avg	64.29	22.25	22.11	38.76	36.37	36.76	29.22
	DARE+Avg	65.40	31.98	21.30	56.34	36.7	42.34	36.08
	LM-Cocktail	62.60	35.82	22.10	42.40	36.69	39.92	32.41
	RECALL(Our)	75.10	37.00	22.07	53.75	37.09	**45.00**(+6.28%)	**38.92**(+7.86%)

Table 1: Performance of merging the base model(Llama-2-7B-chat) and the model fine-tuned on one specific dataset. We compared our method with 4 baselines and marked the best two results in **bold** and underlined fonts. The average performance on 5 datasets and 4 datasets(except the fine-tuning dataset) is also labeled in the last two columns.

after learning it, and the x-axis denotes the task index.

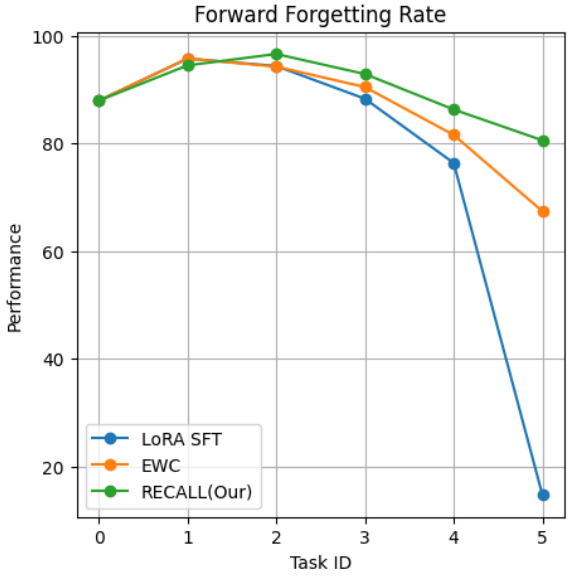

Figure 5: Performance curves on SST-2 during sequential fine-tuning with other two baselines.

As illustrated in Figure 5, LoRA SFT suffers from a dramatic performance decline on the original SST-2 task as training progresses on new tasks, indicating a severe forward forgetting phenomenon. EWC alleviates this to some extent, but still shows a noticeable downward trend. In contrast, our proposed RECALL method maintains relatively stable performance throughout the sequential fine-tuning process, with only a moderate decline toward the final tasks. This suggests that RECALL is more effective at preserving prior task knowledge compared to the other two baselines.

These results confirm that RECALL is well-suited for deployment in dynamic learning environments, offering resilience to forgetting while ensuring consistent learning progress. Detailed per-task results are available in Appendix E.

5 Related works

Catastrophic forgetting (CF) is particularly severe in realistic deployment settings, where training data from previous tasks may be inaccessible due to privacy concerns, and task boundaries or identifiers are typically unavailable. To address CF, existing continual learning (CL) approaches can be broadly categorized into two classes: **data-based methods** and **model-based methods**. Data-based methods leverage stored or generated samples from earlier tasks (Lopez-Paz and Ranzato, 2017; Re-

	Method	Datasets					Average
		SST-2	SQuAD2.0	IWSLT2017-en-fr	RACE	MedMCQA	
Llama2-7B-chat(base model)		87.96	0.94	9.64	50.14	35.91	36.918
With base model	Avg.	86.47	54.85	27.25	58.65	35.84	52.612
	DARE+Avg.	86.35	63.9	34.24	61.63	36.82	<u>56.588</u>
	LM-Cocktail	51.38	66.74	29.31	68.89	36.07	50.478
	RECALL(Our)	85.44	78.4	28.26	57.9	34.66	**56.932**
Without base model	Avg.	91.28	67.85	35.87	66.94	37.2	59.828
	DARE+Avg.	89.6	68.01	36.85	69.08	40.96	<u>60.9</u>
	Task Vector	11.82	29	9.98	49.64	7.36	21.56
	DARE+Task Vector	16.86	29.34	11.11	50.34	9.25	23.38
	RECALL(Our)	89.11	77.66	33.12	74.39	39.86	**62.828**

Table 2: Performance of merging multiple models. **With base model**: Merging the five fine-tuned models and the base model(Llama-2-7B-chat). **Without base model**: Merging the five fine-tuned models. We compared our method with several baselines and marked the best two results in **bold** and <u>underlined</u> fonts.

Fine-tuned on	Datasets					Average
	SST-2	SQuAD2.0	IWSLT2017-en-fr	RACE	MedMCQA	
Without SFT	93	37.02	41.85	87.66	52.45	62.396
SST-2	96.79	35	41.43	76.92	46.78	59.384
SQuAD2.0	92.55	98.51	42.33	50.63	24.38	61.68
IWSLT2017-en-fr	92.32	31.49	52.62	86.34	50.9	62.734
RACE	28.9	25.89	35.28	99.15	54.35	48.714
MedMCQA	0.57	13.43	35.5	88.71	97.7	47.182
RECALL_6merges	94.15	81.45	45.03	91.93	59.14	**74.34**

Table 3: **Supplementary Experiments**: Performance of Qwen2-7B-Instruct and models fine-tuned on one specific dataset, compared with the model merged with the above six models using RECALL. The best result is marked in **bold** font.

buffi et al., 2016), while model-based methods impose constraints on parameter updates or isolate task-specific modules (Kirkpatrick et al., 2016; Fernando et al., 2017). Some recent work adapts these paradigms to LLMs using parameter-efficient tuning modules (Wei et al., 2025; Tian et al., 2024).

5.1 Model Merging

Model merging has emerged as an alternative to traditional CL methods, enabling knowledge integration without access to historical training data. Most methods perform parameter-level fusion, typically via uniform averaging, without accounting for layer-wise functional differences.

Task Arithmetic (Ilharco et al., 2022) and ModelSoup (Wortsman et al., 2022) showed that simple weight averaging can yield multi-task models. Fisher Merging (Matena and Raffel, 2022) incorporates importance weights based on Fisher information to preserve task-relevant parameters. RegMean (Jin et al., 2023) formulates merging as a regression problem over model outputs, aligning them via low-rank projection.

Other works attempt to mitigate interference through more selective merging. TIES-Merging (Yadav et al., 2023) trims parameter deltas and aligns signs, while DARE (Yu et al., 2023) sparsifies task-specific shifts to preserve key differences. LM-Cocktail (Xiao et al., 2023) and LLM-Blender (Jiang et al., 2023) perform weighted merging or output blending using learned domain signals or generation-based rankers.

5.2 Probing Representations

Probing techniques analyze how LLMs internally organize linguistic and task knowledge. Prior work has shown that lower layers tend to encode syntactic information, while upper layers capture semantics and abstract features (Tenney et al., 2019; Starace et al., 2023).

Starace et al. (2023) demonstrate that linguistic features are unevenly distributed across layers and can shift during adaptation. Tighidet et al. (2024) find that past knowledge may remain latent but inaccessible, while Kotha et al. (2023) show that representation-level forgetting is limited, with performance loss arising from usage changes rather than loss of internal content.

These findings highlight the importance of analyzing internal representations when studying model behavior under adaptation and support representation-driven approaches to knowledge retention and integration.

6 Conclusions

In this work, we first conduct exploratory experiments to explore the phenomenon that data representations drift between layers and models, and relate this phenomenon to knowledge differences and catastrophic forgetting of models. Based on these findings, we propose a method to achieve knowledge fusion and resist catastrophic forgetting by aligning the representations of different layers of the model, called RECALL. RECALL does not require past data and only requires hierarchical model aggregation by exploiting the similarity of model representations to achieve the goal effectively. We verify the effectiveness of the method in multiple scenarios, and analyze the details of the method in more depth through ablation experiments and other tests.

7 Limitations

While RECALL provides an effective and data-free solution to continual learning in large language models, several limitations remain. First, our method assumes access to multiple fine-tuned models on related tasks, which may not always be available in real-world deployment scenarios. Second, the current implementation relies on clustering and similarity computations over a small set of representative samples; while efficient, the selection quality of these typical samples can influence the final merging outcome. Moreover, RECALL is tailored to models with identical architectures and aligned tokenizers—extending to heterogeneous model families or multilingual settings poses additional challenges. Finally, although we empirically validate RECALL across diverse NLP tasks, further investigation is needed on scaling to dozens of tasks or integrating with training-time regularization techniques for tighter lifelong learning integration.

Acknowledgments

This work was supported in part by the Major Key Project of PCL under Grant PCL2024A06 and PCL2025AS10, and in part by the Shenzhen Science and Technology Program under Grant RCJC20231211085918010.

We would like to thank our colleagues and collaborators for their valuable feedback and insightful discussions throughout the course of this work. We are also grateful to the open-source community for providing access to pretrained language models and toolkits that significantly accelerated our research.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pages 2–14, Tokyo, Japan. International Workshop on Spoken Language Translation.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

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

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Diego Doimo, Alessandro Serra, Alessio Ansuini, and Alberto Cazzaniga. 2024. The representation landscape of few-shot learning and fine-tuning in large language models. *ArXiv*, abs/2409.03662.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large

language models. In *International Conference on Learning Representations*.

Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. 2021. Continual learning for text classification with information disentanglement based regularization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

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

David Isele and Akansel Cosgun. 2018. Selective experience replay for lifelong learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*.

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

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, and Agnieszka Grabska-Barwinska. 2016. Overcoming catastrophic forgetting in neural networks. *Proc Natl Acad Sci U S A*, 114(13):3521–3526.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. 2023. Understanding catastrophic forgetting in language models via implicit inference. *arXiv preprint arXiv:2309.10105*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Zhizhong Li and Derek Hoiem. 2018. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947.

David Lopez-Paz and Marc' Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Michael Mccloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Sylvestre Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2016. icarl: Incremental classifier and representation learning. *IEEE Computer Society*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*

and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Alexey Romanov, Anna Rumshisky, Anna Rogers, and David Donahue. 2019. Adversarial decomposition of text representation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 2. long and short papers: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2019), 2-7 June 2019, Minneapolis, Minnesota, USA*.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Giulio Starace, Konstantinos Papakostas, Rochelle Choenni, Apostolos Panagiotopoulos, Matteo Rosati, Alina Leidinger, and Ekaterina Shutova. 2023. Probing llms for joint encoding of linguistic categories. *arXiv preprint arXiv:2310.18696*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *Annual Meeting of the Association for Computational Linguistics*.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. 2024. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *arXiv preprint arXiv:2404.19245*.

Zineddine Tighidet, Andrea Mogini, Jiali Mei, Benjamin Piwowarski, and Patrick Gallinari. 2024. Probing language models on their knowledge source. *arXiv preprint arXiv:2410.05817*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. 2023. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10658–10671, Singapore. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Xiwen Wei, Guihong Li, and Radu Marculescu. 2025. Online-lora: Task-free online continual learning via low rank adaptation. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 6634–6645. IEEE.

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

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2023. C-Pack: Packed Resources For General Chinese Embeddings. *arXiv e-prints*, arXiv:2309.07597.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Xingrun Xing. 2023. Lm-cocktail: Resilient tuning of language models via model merging. *arXiv preprint arXiv:2311.13534*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Le Yu, Yu Bowen, Haiyang Yu, Fei Huang, and Yongbin Li. 2023. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *ArXiv*, abs/2311.03099.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A Datasets and Fine-Tuning Settings

We fine-tune the LLaMA-2-7B model on 5 different datasets from diverse domains and tasks, including sentiment classification, question answering, medical QA, reading comprehension, and machine translation. Detailed statistics and supervised fine-tuning (SFT) hyperparameters are presented below.

Dataset	# train	# test	Metric
SST-2	60,000	872	Accuracy
SQuAD2.0	130,000	11,873	Exact Match
MedMCQA	100,000	4,183	Accuracy
RACE	80,000	4,934	Accuracy
IWSLT2017-en-fr	100,000	8,597	Exact Match

Table 4: Statistics for the datasets used to fine-tune LLaMA-2-7B.

A.1 Dataset Statistics and Prompt Format

Dataset Descriptions.

- **SST-2** (Socher et al., 2013): Binary sentiment classification dataset with movie reviews labeled as positive or negative.
- **SQuAD2.0** (Rajpurkar et al., 2016, 2018): Reading comprehension dataset with both answerable and unanswerable questions.
- **MedMCQA** (Pal et al., 2022): Multiple-choice QA dataset from Indian medical entrance exams.
- **RACE** (Lai et al., 2017): Reading comprehension dataset from English exams for Chinese middle and high school students.
- **IWSLT2017-en-fr** (Cettolo et al., 2017): English-to-French translation dataset from TED talks.

A.2 Fine-Tuning Hyperparameters

We fine-tune five task-specific models based on LLaMA-2-7B using LoRA (Hu et al., 2022) on 8 NVIDIA V100 GPUs. Each model is trained with distinct hyperparameters tailored to its dataset. The LoRA config is reported as $r/\alpha/$dropout ((see Table 5 for details)).

The LLaMA-2-7B-chat (Touvron et al., 2023) is used as the base model. We intentionally chose diverse datasets to simulate a challenging setup for continual learning and knowledge fusion.

B Experimental Framework

We adopt `llama-factory` (Zheng et al., 2024) for instruction tuning. It supports various parameter-efficient fine-tuning methods such as LoRA (Hu et al., 2022) and QLoRA (Dettmers et al., 2023), enabling flexible configuration and easy adaptation to various data formats.

Model performance is evaluated using `OpenCompass` (Contributors, 2023), which integrates a broad range of standardized benchmarks to ensure consistency and reproducibility. For efficient inference, we deploy models with `vLLM` (Kwon et al., 2023), providing high throughput and low latency.

C Instruction and Clustering Sample Details

To illustrate the data used in our experiments, we present two sets of representative samples. Table 6 shows instruction samples used during supervised fine-tuning (SFT).

D Supplementary Similarity Curves

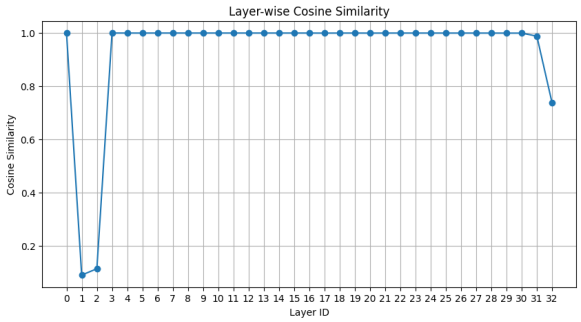

Figure 6: Cosine similarity between adjacent hidden layers within a single LLM. The similarity drops in both early and late layers, suggesting non-uniform transformation of representations across the network.

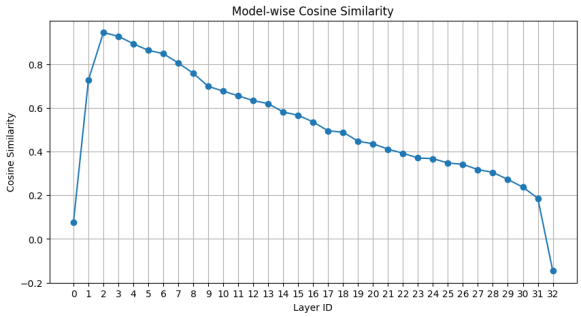

Figure 7: Cosine similarity between representations at the same layer across two LLMs fine-tuned on different tasks. Similarity remains high in early layers but decreases in deeper layers, indicating increasing task-specific divergence.

As shown in Figure 6, 7, representational similarity varies across layers and tasks. Models trained on similar tasks (e.g., SST-2 and RACE) show higher alignment in middle and upper layers, while

Dataset	LoRA (r/α/dropout)	Max Len	LR	Batch	Epochs	Deepspeed
SST-2	8 / 32 / 0.1	2048	5e-5	64	3	ZeRO-3
SQuAD2.0	8 / 32 / 0.1	2048	5e-5	32	4	ZeRO-3
MedMCQA	8 / 32 / 0.1	2048	5e-5	64	3	ZeRO-3
RACE	8 / 32 / 0.1	2048	5e-5	128	5	ZeRO-3
IWSLT2017-en-fr	8 / 32 / 0.1	2048	5e-5	64	5	ZeRO-3

Table 5: SFT hyperparameters for each dataset.

Task	Example
SST-2	**Instruction:** Statement: the characters in swimfan seem motivated by nothing short of dull, brain-deadening hangover. What's sentiment should the above sentence be? OPTIONS:- negative.- positive. Answer: **Output:** negative
SQuAD2.0	**Instruction:** Unpopulated boards are usually bare-board tested for ... the appropriate contact points and only on these. According to the above passage, answer the following question. If it is impossible to answer according to the passage, answer 'impossible to answer': Question:Whatś an absent connection that needs to be linked up on an unpopulated board called? **Output:** An open
MedMCQA	**Instruction:** Question: Which of the following metabolic reactions require vitamin B12 but not folate? Options: A: Conversion of malonic acid to succinic acid B: Conversion of homocysteine to methionine C: Conversion of serine to glycine D: Thymidylate synthesis Choose an correct answer from A/B/C/D.Answer: **Output:** A
RACE	**Instruction:** Read the article, and answer the question by replying A, B, C or D. Article: Tired of all the pushing in supermarkets? Angry at wasting ... claim it to be. Q:The author agrees with the fact that ... **Output:** D

Table 6: Instruction samples used for supervised fine-tuning.

those from different domains (e.g., MedMCQA vs. IWSLT) diverge significantly, especially in deeper layers. These patterns are consistent with our main findings and further support the use of representation-aware merging strategies.

E Sequential Fine-Tuning Results

To provide a strong baseline for comparison, we conduct sequential fine-tuning (SeqFT) experiments, where a single model is trained on multiple datasets in a fixed order without revisiting previous ones. This setting simulates a continual learning scenario and serves to quantify the extent of catastrophic forgetting.

We sequentially fine-tune the LLaMA-2-7B model across five diverse tasks, including sentiment classification, question answering, medical QA, reading comprehension, and machine translation. All models are trained under the same LoRA configuration for consistency. After completing each step in the sequence, we evaluate the model on all previously seen datasets to track performance drop.

As shown in Table 7, performance on earlier tasks gradually deteriorates as the model is updated on subsequent ones. The trend clearly reflects catastrophic forgetting and reinforces the need for continual learning strategies such as our proposed representation-aware model merging, which avoids overwriting previous knowledge by aligning and preserving internal representations.

F Comparison of Similarity Metrics

To determine the most effective similarity metric for guiding our representation-aware model merging, we conduct a comparative study across five widely-used similarity measures. These metrics are used to compute the alignment between hidden representations of models, which in turn inform the layer-wise merging weights.

The five similarity metrics evaluated are:

- **Cosine similarity**: x, y are vectors.

$$Sim = \frac{x^T y}{||x||_2 \times ||y||_2} \tag{7}$$

Method	Task Sequence		Datasets					Average
			SST-2	SQuAD2.0	MedMCQA	IWSLT2017	RACE	
LoRA SFT	Task 1	SST-2	95.76	31.68	32.32	13.28	44.71	43.55
	Task 2	SQuAD2.0	94.38	87.42	16.88	25.06	58.15	56.378
	Task 3	MedMCQA	88.3	74.89	42.62	19.29	68.38	58.696
	Task 4	IWSLT2017	76.38	75.71	42.39	45.29	58.73	59.7
	Task 5	RACE	14.79	68.05	39.8	34.85	86.24	48.746
EWC	Task 1	SST-2	95.76	31.68	32.32	13.28	44.71	43.55
	Task 2	SQuAD2.0	94.27	88.32	25.77	20.94	51.64	56.188
	Task 3	MedMCQA	90.47	72.12	42.53	12.44	57.75	55.062
	Task 4	IWSLT2017	81.59	65.31	41.05	47.86	55.6	58.282
	Task 5	RACE	67.42	64.81	39.68	33.54	87.34	58.558
RECALL(Our)	Task 1	SST-2	94.5	30.72	34.9	12.08	47.44	43.928
	Task 2	SQuAD2.0	96.61	86.34	30.79	19.83	57.52	58.218
	Task 3	MedMCQA	92.89	71.66	40.65	18.48	69.06	58.548
	Task 4	IWSLT2017	86.31	67.09	38.16	45.73	67.55	60.968
	Task 5	RACE	80.59	62.38	36.22	43.14	88.97	**62.26**

Table 7: Detailed performance of sequence training scenario.

- **Euclidean distance (converted to similarity)**: x, y are vectors.

$$Sim = \frac{||x - y||_2}{\max_{\mathcal{X},\mathcal{Y}} ||\mathcal{X} - \mathcal{Y}||_2} \tag{8}$$

- **Centered Kernel Alignment (CKA)** (Kornblith et al., 2019): X, Y are two distributions.

$$\text{CKA}(X, Y) = \frac{\|X^\top Y\|_F^2}{\|X^\top X\|_F \cdot \|Y^\top Y\|_F} \tag{9}$$

- **Maximum Mean Discrepancy (MMD)**: X, Y are two distributions.

$$\begin{aligned} \text{MMD}^2(X, Y) = &\frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} k(x_i, x_j) \\ &+ \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} k(y_i, y_j) \\ &- \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} k(x_i, y_j) \end{aligned} \tag{10}$$

- **RBF kernel**: See Eq 1.

For each metric, we compute layer-wise alignment scores between expert models, normalize the weights, and perform hierarchical model merging using the same fusion strategy. The final merged models are evaluated on multiple tasks to assess performance consistency.

As shown in Table 8, cosine similarity yields the highest performance across all evaluation datasets. While CKA and dot product also perform competitively, metrics like Euclidean distance and MMD are less stable. These results support our choice of cosine similarity as the default alignment metric in our model merging framework.

G RECALL Algorithm Details

The Analysis of Runtime, Memory Usage and Scalability

Compared with other model merging methods, our method mainly adds the following steps: (1) Feature extraction; (2) Extracting typical samples; (3) Extracting the representation of typical samples in all models; (4) Similarity calculation; (5) Hierarchical merging.

Mathematical notation convention: t: number of iterations; k: number of clusters; n: number of samples; E: dimension of features/hidden layers; l: number of model layers; b: mini-batch size; s: number of GPUs; m: number of typical samples; N: number of models to merge.

- **Feature extraction**: Feature extraction forwards all samples and saves all hidden layer states. In order to save memory and speed up, we use distributed inference and pass features back to rank0 on each batch, where they are offloaded to CPU memory to reduce GPU memory usage. So the GPU memory complexity is $O(bEl)$, and the CPU memory and subsequent storage complexity is $O(nEl)$. And the time complexity is: $O(\alpha l \frac{n}{bs} + \beta nEl \frac{s-1}{s} + \gamma nEl)$, where the first term is the time required to forward samples across GPUs in parallel, The second term is the time it takes for sub nodes to send a batch of features to rank 0, the third term is the time it takes to offload all the features from GPU to CPU, and $\alpha, \beta, \gamma$are scaling constants. **If this distributed offload-**

Metric	SST-2	SQuAD2.0	IWSLT2017-en-fr	RACE	MedMCQA	Avg.
Cosine	83.83	67.99	33.24	65.2	37.03	57.458
Euclidean	88.65	26.72	**43.64**	38.93	34.71	46.53
CKA	83.94	68.04	33.25	65.16	36.91	57.46
MMD	65.83	28.93	41.26	50.87	36.58	44.694
RBF	**89.11**	**77.66**	33.12	**74.39**	**39.86**	**62.828**

Table 8: Performance of merged multiple models(without base model) using different similarity metrics. RBF Kernel similarity consistently achieves the best average performance across tasks.

Algorithm 1 RECALL

Require: Task dataset D_N, source model M_0, fine-tuned models $M_1, M_2, \ldots, M_{N-1}$ with parameters $\boldsymbol{\theta}^q$ ($q \in [0, N-1]$)

Ensure: Merged model parameters $\boldsymbol{\theta}^*$

1: $M_N \leftarrow$ Fine-tune M_0 on D_N

2: $D_{type} \leftarrow Kmeans(\mathbf{R}_N)$, which are representations extracted from D_N using M_N

3: **for** each expert model $M_j,\ j \in [1, N-1]$ **do**

4: $\quad \mathbf{R}_j \leftarrow$ Extract representations from D_{type} using M_j

5: **end for**

6: **for** each layer $i \in [1, L]$ **do**

7: $\quad$ Compute similarity $S_i^{p,q}$ between models M_p and M_q for layer i

$$S_i^{p,q} = \frac{1}{||D_{type}||} \sum_{D_{type}} \mathbf{RBF}(R_p^i, R_q^i)$$

8: $\quad$ Normalize similarities to obtain merging weights:

$$w_i^q = \frac{\exp(S_i^{N,q})}{\sum_{q=0}^{N} \exp(S_i^{N,q})}$$

9: $\quad$ Merge model parameters at layer i:

$$\boldsymbol{\theta}_i^* = \sum_{q=0}^{N} w_i^q \boldsymbol{\theta}_i^q$$

10: **end for**

11: **return** $\boldsymbol{\theta}^*$

ing strategy isn't adopted, the time complexity and space complexity will greatly increase: Space complexity: $O(nEl)$; Time complexity: $O(\alpha nl)$.

- **Extracting typical samples**: We perform Kmeans clustering for the features of each layer in step (1), and the space complexity and time complexity of the one-pass Kmeans algorithm are $O(E(n+k))$and $O(tknE)$, respectively. The overall complexity is: Space complexity: $O(E(n+k)l)$(on CPU); Time complexity: $O(tknEl)$.

- **Extracting the representation of typical samples in all models**: We use m typical samples to perform forward inference on the N models to be merged, and save the hidden layer information of all layers. We adopt the same strategy as step (1), so the space complexity and time complexity are: Space complexity: $O(b'ElN)$(b' will be smaller than b because a typical sample set generally does not occupy all GPU memory); Time complexity: $O((\alpha l\frac{m}{bs} + \beta mEl\frac{s-1}{s} + \gamma mEl)N)$.

- **Similarity calculation**: We compute the similarity between the representations of the main model and all other models at each layer, using **rbf kernel** as the similarity metric, and averaging the similarity across all typical samples, so: Space complexity: $O(mlN)$; Time Complexity: $O(mlEN)$.

- Obviously, when $n \gg m$, utilizing typical samples will **greatly reduce the memory and time consumption of extracting features and calculating similarity** (step (3/4)), which is one of the important reasons for our choice of sampling.

- **Hierarchical merging**: Compared with other model merging algorithms, we increase the number of weights in model merging (each model has an independent floating-point weight in each layer), but still take a single parameter as the unit for merging. So: Space complexity: $O(lN)$; Time complexity: $O(N)$.

- **Scalability**: When the scale of the model increases, time and space consumption will grow at a **linear** rate.