Optimal Brain Iterative Merging: Mitigating Interference in LLM Merging

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have demonstrated impressive capabilities, but their high computational costs pose challenges for customization. Model merging offers a cost-effective alternative, yet existing methods suffer from interference among parameters, leading to performance degradation. In this work, we propose **O**ptimal **B**rain **I**terative **M**erging (OBIM), a novel method designed to mitigate both intra-model and inter-model interference. OBIM consists of two key components: (1) A *saliency measurement mechanism* that evaluates parameter importance based on loss changes induced by individual weight alterations, reducing intra-model interference by preserving only high-saliency parameters. (2) A *mutually exclusive iterative merging framework*, which incrementally integrates models using a binary mask to avoid direct parameter averaging, thereby mitigating inter-model interference. We validate OBIM through experiments on both Supervised Fine-Tuned (SFT) models and post-pretrained checkpoints. The results show that OBIM significantly outperforms existing merging techniques. Overall, OBIM provides an effective and practical solution for enhancing LLM merging. We will publicly release our code upon the acceptance of this paper.

1 Introduction

Existing research (Akiba et al., 2025; Dekoninck et al., 2023; Wan et al., 2024a) has demonstrated that a composite LLM can be constructed by merging the parameters of different expert LLMs. Traditional approaches (Wortsman et al., 2022; Matena and Raffel, 2022; Jin et al., 2022) employ matrices to determine task-specific coefficients and perform a weighted average based on these coefficients. Methods grounded in task arithmetic (Ilharco et al., 2023) leverage task vectors, defined as the difference between the parameter values of a fine-tuned model and those of its pre-trained counterpart, to effectively manipulate and integrate the knowledge embedded within the models.

State-of-the-art model merging methods (Yadav et al., 2024; Wang et al., 2024; Yu et al., 2024b) have shown that task performance degradation is primarily caused by interference between parameter values, as aggregation operations, such as averaging, can alter the parameter distribution (Yu et al., 2024a). The interference can be categorized into two types: intra-model interference and inter-model interference.

Intra-model interference arises from redundant parameters within a single model. Due to the over-parameterized nature of neural networks (Choudhary et al., 2020; He and Xiao, 2023), removing a significant portion of the parameters often has little impact on model performance (Sun et al., 2023; Kim et al., 2024). However, these redundant parameters introduce noise during the model merging process, adversely affecting the outcome. To address this, it is crucial to identify parameters that are closely related to the target task. Existing approaches, however, primarily rely on magnitude-based methods, assuming that parameter magnitude directly correlates with saliency. For instance, TIES (Yadav et al., 2024) trims the parameters with the smallest magnitudes, the Model Breadcrumbs (Davari and Belilovsky, 2025) highlight the importance of removing the parameters with the largest weights to further reduce noise. While these methods demonstrate effectiveness, they fall short of fully revealing the true saliency of the parameters.

Interference between models arises due to variations in parameter distributions (Shoemake, 1985; Jang et al., 2024). Directly averaging these parameters can lead to performance degradation. TIES addresses this issue by resolving sign conflicts in parameter values, aligning them based on the direction of the largest total movement across models. Similarly, TALL-Mask (Wang et al., 2024) is de-

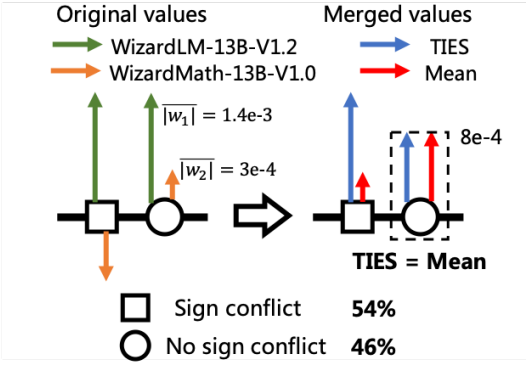

Figure 1: Illustration of inter-model interference. The dotted box highlights cases where TIES fails to resolve interference. Approximately 46% of parameters deviate from the original models due to task vector averaging in the absence of sign conflicts.

signed to exclude parameters that are relevant only to a subset of tasks. While these methods effectively mitigate inter-model interference under certain conditions, their effectiveness diminishes when parameter distributions deviate from expected patterns, causing them to revert to simple averaging. As shown in Figure 1, when there is no sign conflict, TIES yields the same result as simple averaging, deviating from both input models and leading to suboptimal performance.

To address the interference problem in model merging, we propose a novel method for LLMs called Optimal Brain Iterative Merging (OBIM). Our approach comprises two core components: a saliency measurement mechanism to filter intra-model interference and a mutually exclusive iterative merging framework to prevent inter-model interference.

In detail, our approach measures the saliency of parameters within a single model by evaluating the loss change induced by altering each parameter. Inspired by layer-wise model pruning methods (Frantar and Alistarh, 2022; Frantar et al., 2022), we forgo reliance on the overall model loss during training and instead independently apply the Mean Square Error (MSE) to each linear weight. This enables calculation of the output distribution difference between the trained weight and the original weight, providing a more precise and efficient measure of parameter saliency. By retaining parameters with high saliency, we effectively reduce intra-model interference in model merging.

Subsequently, we design an iterative merging framework to integrate models step by step in a mutually exclusive manner, mitigating inter-model interference. Specifically, we employ a binary mask to track the positions that have already been merged. At each step, parameters with the highest saliency, which are not yet recorded in the mask, are selected from a model and integrated into the base model. This ensures that each position is occupied by only one parameter, thereby eliminating the need for averaging operations.

In summary, we propose a novel method, OBIM, to mitigate both intra-model and inter-model interference in LLM merging. To validate the effectiveness of our method, we conducted model merging experiments on Supervised Fine-Tuned (SFT) models of Llama2 (Touvron et al., 2023) for multi-task merging and post-pretrained checkpoints of Qwen2 (Yang et al., 2024a) for catastrophic forgetting recovery. The results of both experiments demonstrate that OBIM significantly outperforms existing approaches. In addition, extensive ablation studies and analyses of key factors provide a comprehensive understanding of OBIM.

2 Preliminaries

2.1 Model Merging Problem

In this paper, we focus on merging models that are optimized from the same backbone. Given K models with parameters $\{\theta^1, \theta^2, \ldots, \theta^K\} \in \mathbb{R}^d$, each trained on a distinct task or setting $\{t_1, t_2, \ldots, t_K\}$ from a shared base model $\theta^B \in \mathbb{R}^d$. Model merging aims to fuse these parameters into a single model with parameters $\theta^M \in \mathbb{R}^d$, and enable θ^M to effectively handle all K tasks simultaneously.

2.2 Task Vector

A task vector $\delta^k \in \mathbb{R}^d$ for model θ^k is defined as the delta weights between the trained model's parameters and those of the backbone:

$$\delta^k = \theta^k - \theta^B, \tag{1}$$

which represents both the direction and magnitude of parameter updates during training.

By merging the task vectors $\{\delta^1, \delta^2, \ldots, \delta^K\}$ of K models into a single task vector δ^M, the parameters of the merged model can be expressed as

$$\theta^M = \theta^B + \delta^M. \tag{2}$$

3 Methodology

The proposed merging method comprises two key components: Optimal Brain Merging (OBM),

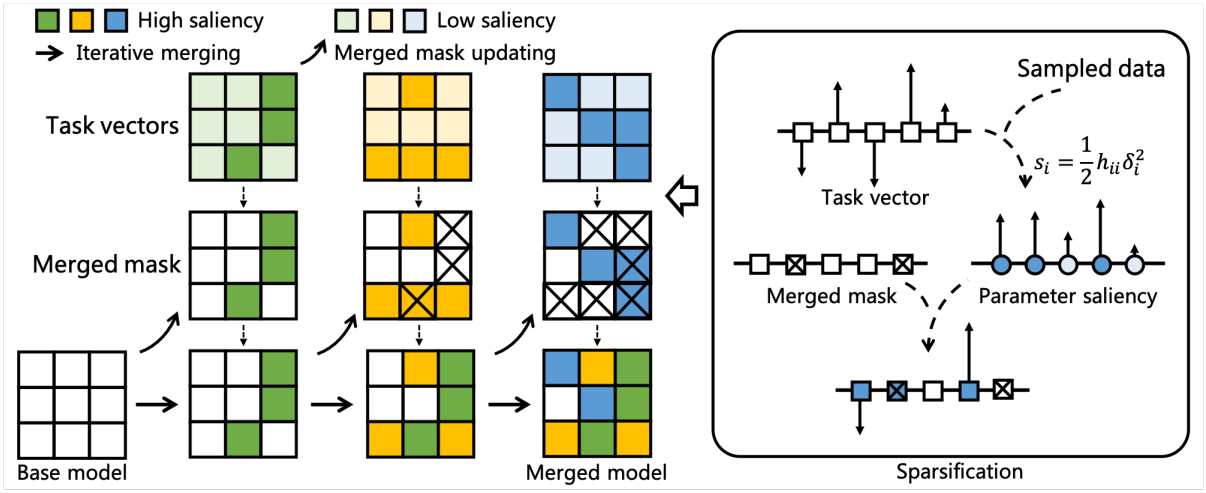

Figure 2: An overview of the proposed method. The left part depicts the iterative merging process, while the right part details how parameters are selected at each iteration step through the cooperation of parameter saliency and the merged mask.

a saliency-based mechanism that selects high-saliency parameters for merging, and Iterative Merging (IM), an iterative framework designed to mitigate interference between models. Figure 2 illustrates the complete merging process of our method.

3.1 Optimal Brain Merging

Optimal Brain Damage (OBD) (LeCun et al., 1989) and its subsequent works (Hassibi et al., 1993; Frantar and Alistarh, 2022) aim to establish effective criteria for pruning or quantizing specific weights while minimizing the impact on model performance. The fundamental idea is to leverage the second derivative of the objective function with respect to the parameters to compute their "saliencies." Building upon the idea, we introduce **O**ptimal **B**rain **M**erging (OBM) to mitigate intra-model interference by identifying and eliminating negligible delta weights in task vectors.

Given a trained LLM with parameters θ and task vector δ, our goal is to identify a subset of parameters in the task vector whose removal results in minimal increase in the objective function $\mathcal{L}$. The change in the objective function is measured using a Taylor series expansion:

$$\Delta\mathcal{L} = \sum_i \frac{\partial \mathcal{L}}{\partial \theta_i}\delta_i + \frac{1}{2}\sum_i \frac{\partial^2 \mathcal{L}}{\partial \theta_i^2}\delta_i^2 + \frac{1}{2}\sum_{i\neq j} \frac{\partial^2 \mathcal{L}}{\partial \theta_i \theta_j}\delta_i\delta_j + O(||\delta||^3). \quad (3)$$

Assuming that $\mathcal{L}$ is at a local minimum and that each parameter contributes to $\Delta\mathcal{L}$ independently, the first derivative, the off-diagonal terms of the second derivative, and the higher-order terms can be discarded. Consequently, $\Delta\mathcal{L}$ can be approximated as:

$$\Delta\mathcal{L} \approx \frac{1}{2}\sum_i \frac{\partial^2 \mathcal{L}}{\partial \theta_i^2}\delta_i^2. \quad (4)$$

The change in $\Delta\mathcal{L}$ when removing the parameter at position i indicates how much it affects performance, thereby representing its saliency:

$$s_i = \frac{1}{2}\frac{\partial^2 \mathcal{L}}{\partial \theta_i^2}\delta_i^2 = \frac{1}{2}h_{ii}\delta_i^2, \quad (5)$$

where h_{ii} denotes the i-th diagonal element of the Hessian matrix of the loss for the given model. Parameters with low saliency, which contribute minimally to $\Delta\mathcal{L}$, should be removed.

However, computing the Hessian matrix requires a back-propagation process through the LLM if the objective function used during LLM training is considered (Bowen et al., 2024). To avoid the high computational cost comparable to model training, we take inspiration from layer-wise pruning approaches (Frantar and Alistarh, 2022; Frantar et al., 2022) and employ the Mean Squared Error (MSE) as the objective function for each linear layer independently.

Formally, let $\mathbf{X}^l$ be the input to the l-th layer with the weight matrix $\mathbf{W}^l$. The objective is defined as:

$$\Delta\mathcal{L}^l = \left\|\mathbf{W}^l\mathbf{X}^l - \mathbf{W}^B\mathbf{X}^l\right\|_2^2 = \left\|\Delta\mathbf{W}^l\mathbf{X}^l\right\|_2^2, \quad (6)$$

where $\mathbf{W}^B$ is the corresponding layer weight of the base model, and $\Delta\mathbf{W}^l$ is the task vector of the layer. To approximate $\mathbf{X}^l$, we take the mean over a small set of input samples. This function measures the squared distance between the output of the trained weights and the original weights.

The Hessian matrix under the layer-wise MSE loss is computed as $\mathbf{H}^l = 2\mathbf{X}^l\mathbf{X}^{l\top}$. Thus, we only need to perform forward propagation of the LLM to obtain the input for each layer, enabling the computation of parameter saliencies. Moreover, forward propagation does not require any labels or targets, only the input portion of the samples is needed. Beyond its simplicity, layer-wise saliency provides a more precise and accurate measure of parameter importance within each layer. For non-linear layers, such as bias layers, we apply random pruning for parameter sparsification.

3.2 Iterative Merging

To address inter-model interference, we propose a merging framework called **I**terative **M**erging (IM). This method iteratively updates a unique, non-overlapping subset of parameters from each task vector, preventing weight interference among task vectors.

For each task vector δ^k to be merged, a binary mask $\mathcal{M}(P_k)$ is constructed to satisfy the following constraints:

$$\mathcal{M}(P_k)_i = \begin{cases} 1, & \text{if } i \in P_k, \\ 0, & \text{otherwise.} \end{cases}$$
$$\text{subject to} \quad \bigcup_{k=1}^{K} P_k \subseteq \{1, 2, \ldots, d\}, \quad P_k \cap P_j = \emptyset \quad \text{for } k \neq j. \tag{7}$$

Here, P_k represents the set of indices corresponding to the parameters selected for merging from δ^k, and d denotes the total number of parameters in each δ^k. Using the binary masks, the merging process is then formulated as:

$$\theta^M = \theta^B + \sum_{k=1}^{K} \delta^k \cdot \mathcal{M}(P_k). \tag{8}$$

Although the formulation involves summation, no direct addition occurs between different task vectors, as the binary masks ensure that each parameter index is selected at most once.

While there are many ways to construct non-overlapping binary masks for all models, we introduce a simple and easily controllable method by iteratively updating a merged mask to track the positions that have already been merged. Specifically, at the beginning, the merged mask is an empty set. At each step, starting with a task vector δ^k, we first exclude the indices that are already in the merged mask. Then, we sort the remaining parameter indices of δ^k based on their saliencies, selecting the top $n_k\%$[1] of the indices to form P_k. Finally, we update the merged mask with P_k. The procedure is described in Algorithm 1.

Algorithm 1 Iterative Merging

Input: Base model parameters θ^B, total parameter count d, task vectors $\delta^{1:K}$, merging ratios $n_{1:K}\%$, saliency score sets $S_{1:K}$
Output: Merged model θ^M

Initialize: merged mask $M \leftarrow \emptyset$, merging order $O \leftarrow [o_1, o_2, \ldots, o_K]$
for k in O **do**
  $S_k \leftarrow \{s_i \mid i \notin M, s_i \in S_k\}$
  Sort S_k in descending order
  $\hat{S}_k \leftarrow$ Select the top $n_k\%$ elements of S_k
  $P_k \leftarrow \{i \mid s_i \in \hat{S}_k, i \in \{1, \ldots, d\}\}$
  Update merged mask: $M \leftarrow M \cup P_k$
end for
return $\theta^M \leftarrow \theta^B + \sum_{k=1}^{K} \delta^k \cdot \mathcal{M}(P_k)$

In practice, we apply iterative merging to each layer independently, utilizing the task vector of each layer rather than the entire model. This approach not only improves memory efficiency but also enables the integration with OBM by leveraging its layer-wise saliency scores.

An important factor that significantly affects the performance of the merged model is the iteration order of the merging process. Since earlier merged models occupy parameter positions, highly salient parameters from later models may have limited opportunities to be incorporated. To address this issue, we utilize a rotation operation that dynamically shifts the merging order across different layers, preventing a few models from dominating the process. Formally, for layer l, we maintain a list to record the merging order of models: $O^l = [o_1, o_2, \ldots, o_K]$ The merging order for layer $l+1$ is then updated by a left rotation operation: $O^{l+1} = \mathcal{LR}(O^l, 1)$. We further conduct an experiment to discuss how the iteration order influences the performance in Section 4.5.

[1] We ensure that $\sum_k n_k \leq 1$.

3.3 Optimal Brain Iterative Merging

OBM and IM can be combined with other existing methods. Taking TIES as an example, when combining TIES with OBM, the magnitude-based parameter pruning is replaced by a saliency-based approach. Similarly, when using TIES together with IM, we utilize global magnitude as saliency scores to construct the merged mask. However, the combination of OBM and IM, referred to as OBIM, yields better results. We conduct an ablation study to demonstrate the effectiveness of each component in Section 4.3.

4 Experiments

We conduct experiments on both SFT models and post-pretrained checkpoints to demonstrate the effectiveness of our method. To validate its robustness, we evaluate OBIM using two popular backbone models, LLaMA2 (Touvron et al., 2023) and Qwen2 (Yang et al., 2024a), in separate experiments. We also perform an ablation study to analyze the contributions of specific components within OBIM. Furthermore, we investigate key factors in our method to assess their influence on the final performance.

4.1 Experimental Setup

Experiment Settings for SFT Models. Following previous works (Yu et al., 2024b; Deep et al., 2024), we use *Llama-2-13b* as the pre-trained backbone and incorporate three fine-tuned models for cross-task merging experiments: *WizardLM-13B-V1.2* (Xu et al., 2023) for instruction following, *WizardMath-13B-V1.0* (Luo et al., 2023) for mathematical reasoning, and *llama-2-13b-code-alpaca* (Chaudhary, 2023) for code generation. To evaluate the capabilities of the merged models, we use AlpacaEval (Li et al., 2023) and MMLU (Hendrycks et al., 2021a) for general understanding; GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b) for mathematical reasoning; and HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for code generation. Performance is measured using the win rate for AlpacaEval[2], zero-shot accuracy for MMLU, GSM8K, and MATH, and pass@1 for HumanEval and MBPP.

Experiment Settings for Post-pretrained Models. We perform post-pretraining on *Qwen2-7B* using a multilingual dataset to enhance its Japanese proficiency. The dataset consists of over 200 billion tokens and includes publicly available pretraining corpora in English, Chinese, and Japanese. Details of the dataset are provided in Appendix A.3. The three best-performing checkpoints on Japanese evaluation are selected and merged with the backbone model. To assess performance across different languages, we employ three benchmarks: C-Eval (Huang et al., 2023) for Chinese, MMLU for English, and the Japanese Language Model Evaluation Harness (JP-LMEH) [3] for Japanese. Five-shot accuracy is used for evaluating C-Eval and MMLU. JP-LMEH encompasses nine distinct NLP tasks[4], with the average score serving as an indicator of overall Japanese language proficiency.

Baselines. We compare our method with the following state-of-the-art model merging approaches applicable to LLMs: **TA** (Ilharco et al., 2023): Task Arithmetic, a simple delta weight merging method that does not explicitly address interference. **TIES** (Yadav et al., 2024): Eliminates redundant parameters based on magnitude and resolves sign conflicts. **DARE** (Yu et al., 2024b): Drop And REscale, Randomly drops a proportion of parameters and rescales the remaining ones to reduce redundancy. **DELLA** (Deep et al., 2024): Assigns dropout probabilities to parameters based on their magnitudes for pruning. **TALL-Mask** (Deep et al., 2024): Uses a masking mechanism to filter out parameters relevant to only a few tasks. **PCB** (DU et al., 2024): Leverages parameter competition to optimize the merging process.

Validation Set. Since OBIM requires a small sample set for parameter saliency computation, we hold out a validation set comprising portions of the training and development sets from each benchmark. Specifically, we compute saliency using data related to the model's training task: AlpacaEval and MMLU for general models, GSM8K and MATH for math models, and MBPP for code models. For post-pretrained models, saliency is computed using a multilingual dataset consisting of C-Eval, MMLU, and JP-LMEH. Details are provided in Appendix A.4.

[2]We calculated the win rate by comparing target model to *text-davinci-003*, using *GPT-4o* as the evaluator.

[3]https://github.com/Stability-AI/lm-evaluation-harness/tree/jp-stable

[4]JSQuAD, JCommonsenseQA, JNLI, MARC-ja, XLSum-ja, JCoLA, MGSM-ja, XWinograd-ja, and JAQKET

Model	Method	General		Math (acc)		Code (pass@1)		Avg.
		AlpacaEval	MMLU	GSM8K	MATH	HumanEval	MBPP	
LM	-	82.72	53.34	45.79	0.14	30.48	31.40	40.65
Math	-	-	-	63.08	11.60	-	-	-
Code	-	-	-	-	-	23.78	27.20	-
LM + Math + Code	TA	78.93	51.04	58.45	9.88	18.29	29.80	41.07
	TIES	80.53	54.30	62.55	9.54	21.95	30.40	43.21
	DARE	75.00	54.12	58.00	9.20	**29.27**	**31.40**	42.83
	DELLA	**83.16**	53.52	61.80	7.88	19.50	**31.40**	42.87
	TALL-Mask	80.31	54.25	62.70	10.62	20.73	30.80	43.23
	PCB	81.98	53.37	63.83	8.24	26.22	26.60	43.37
	OBIM	81.23	**54.39**	**68.23**	**12.50**	25.61	29.40	**45.23**

Table 1: Performance comparison of SFT model merging. The results for each individual model are presented at the top of the table, the lower section displays the results of merging the three models using different methods.

4.2 Main Results

Results on SFT Models. The results are summarized in Table 1. We first present the performance of each individual model, followed by the results of merging the three task-specific experts using different methods. As shown in Table 1, OBIM achieves significant improvements in mathematical reasoning, with a 5.15% gain on GSM8K and a 0.9% gain on MATH compared to the original math model. In contrast, many other methods fail to surpass the source math model. For other benchmarks, OBIM ranks first on MMLU and remains competitive across other tasks. However, its performance on code generation is relatively lower. We suspect this is because the general model, rather than the code model, performs best on code generation, yet its saliency is computed using general data, leading to suboptimal preservation of coding ability. Overall, OBIM achieves the highest average performance, outperforming the second-best method by 1.86%, demonstrating its effectiveness in merging models for task fusion.

Model	Method	C-Eval	MMLU	JP-LMEH	Avg.
Qwen2-7B	-	83.51	69.22	69.19	73.97
ckpt-1	-	77.71	67.01	72.13	72.28
ckpt-2	-	75.48	67.05	71.69	71.41
ckpt-3	-	74.37	66.86	71.61	70.95
Qwen2-7B + ckpt-1 + ckpt-2 + ckpt-3	TA	76.15	68.47	71.90	72.17
	TIES	76.52	67.92	71.93	72.12
	DARE	78.97	68.85	71.21	73.01
	DELLA	77.26	68.51	72.05	72.61
	TALL-Mask	77.72	68.48	71.56	72.59
	PCB	77.71	68.31	72.08	72.70
	OBIM	**80.46**	**69.89**	**72.14**	**74.16**

Table 2: Performance comparison of post-pretrained model merging. *ckpt-1*, *ckpt-2* and *ckpt-3* are the checkpoints that achieve the best JP-LMEH results during post-pretraining but exhibit performance degradation in Chinese and English.

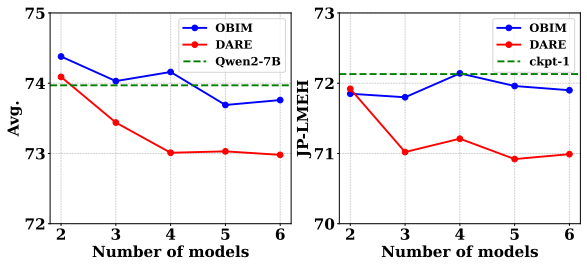

Figure 3: Performance comparison of merging different numbers of models between OBIM and DARE. The left part presents the average performance across three languages, while the right part shows the results for Japanese capability. The green dotted line represents the best performance of models before merging.

Results on Post-pretrained Models. As shown in Table 2, checkpoints trained from *Qwen2-7B* exhibit enhanced Japanese proficiency but experience a significant decline in the Chinese and English capabilities of the base model. The goal of model merging is to restore Chinese and English performance to the base model level while preserving the improved Japanese ability of the continually trained model. Our results show that while no merging method fully restores C-Eval performance to the base model level, OBIM achieves the highest recovery rate, surpassing other methods by 1.49%. For English and Japanese, OBIM is the only method that surpasses the base model on MMLU and JP-LMEH, achieving improvements of 1.04% and 0.04% over other approaches. In terms of overall multilingual performance, OBIM ranks first across all benchmarks, producing a model with the strongest overall capabilities.

To further validate the robustness of our method, we evaluate the performance of merging different numbers of models. Specifically, we select the five

Method	Intra-model	Inter-model	General		Math (acc)		Code (pass@1)		Avg.
			AlpacaEval	MMLU	GSM8K	MATH	HumanEval	MBPP	
TIES	Magnitude	Disjoint Mean	80.53	54.30	62.55	9.54	21.95	30.40	43.21
TIES+OBM	Saliency		79.03	**54.44**	64.29	10.52	**26.83**	**30.80**	44.32
TIES+IM	Magnitude	Iterative merging	80.81	54.31	68.08	**13.08**	21.95	30.40	44.77
OBIM	Saliency		**81.23**	54.39	**68.23**	12.50	25.61	29.40	**45.23**

Table 3: Ablation study on components in OBIM. Each method's strategy for mitigating intra-model and inter-model interference is listed in the columns *Intra-model* and *Inter-model*, respectively.

best checkpoints during post-pretraining and merge them with the base model. We compare our method with DARE, as well as the best-performing individual models before merging, and present the results in Figure 3[5]. The results indicate that performance declines as the number of merged models increases, likely due to increased interference among models. However, our method remains stable and achieves performance competitive with the best individual model in both the average and Japanese capability evaluations. In contrast, DARE underperforms by approximately 1%, demonstrating that our method more effectively mitigates interference when merging multiple models.

4.3 Ablation Study

To assess the contributions of the two key components, OBM and IM, we conduct experiments using SFT model merging settings. Since OBM and IM cannot perform merging independently, we use TIES as the baseline method and integrate OBM and IM with the weight consensus approach for resolving sign conflict, termed "Disjoint Mean," as well as the magnitude-based parameter trimming method in TIES, respectively. Details are provided in Appendix A.1.

The results are presented in Table 3, where we also outline each method's strategy for mitigating intra-model and inter-model interference. By comparing TIES with TIES+OBM and TIES+IM with OBIM, where each pair employs the same method for reducing inter-model interference, we observe that methods utilizing saliency-based parameter selection outperform those relying on magnitude-based selection. This finding confirms the superiority of OBM. Furthermore, methods incorporating IM consistently outperform their counterparts using the same intra-model approach. Specifically, TIES+IM surpasses TIES, and OBIM outperforms TIES+OBM, demonstrating the effectiveness of IM. In general, OBIM achieves the highest performance among all methods, proving that the combination of OBM and IM can further enhance the performance.

[5]Detailed results are provided in Appendix B

4.4 Influence of the Validation Set

Samples in the validation set influence saliency scores. To assess this impact, we replace the samples used for computing the saliency of the math model under the experimental settings of SFT model merging and evaluate the merging performance on two mathematical tasks.

As shown in Table 4, we compare the results using task-related data from GSM8K and MATH (*Math*), irrelevant data from MBPP (*Code*), and a mixed dataset (*Math+Code*). All datasets contain the same number of samples. *Math* achieves the best performance, followed by *Math+Code*, while *Code* performs the worst. This suggests that using task-specific data for saliency computation enhances task knowledge retention. We attribute this to our assumption for saliency approximation, which requires the first derivative to be approximately zero. This condition implies that data well learned by the model enables more accurate saliency estimation.

Source	Size	Math (acc)		Avg.
		GSM8K	MATH	
Math	100	**68.23**	**12.50**	**40.37**
Code		67.48	11.84	39.66
Math + Code		<u>68.01</u>	<u>12.42</u>	<u>40.22</u>

Table 4: Comparison of different validation sets used for saliency computation. The winners and runners-up are marked in bold font and underlined, respectively.

We also investigate the impact of sample size by using different numbers of samples from the same source. The results in Table 5 indicate that 100 samples yield the best performance, while using either more or fewer samples leads to a decline. However, the results with only 10 samples remain competitive, suggesting that the method is effective even with a limited number of samples.

Source	Size	Math (acc)		Avg.
		GSM8K	MATH	
Math	10	68.16	12.10	40.13
	50	67.40	11.88	39.64
	100	**68.23**	**12.50**	**40.37**
	200	67.70	12.28	39.99

Table 5: Performance across different sample sizes. 100 samples achieve the highest performance, followed by 10 samples.

4.5 Influence of Iterative Merging Order

We analyze how the order of iterative merging influences performance. We conduct experiments based on the SFT model merging settings with four different merging orders: shifting the order using a rotation operation across model layers (*Rotation*), prioritizing the math model (*Math First*), placing the math model last (*Math Last*), and prioritizing the general language model (*LM First*). We evaluate the results on mathematical tasks.

As shown in Table 6, *Math First* achieves the best performance, whereas *Math Last* performs the worst, and *LM First* also yields poor results. These findings suggest that the earlier a model is merged, the better its knowledge is preserved. However, while *Rotation* performs slightly worse than *Math First*, it still surpasses the original math model, demonstrating its robustness. We attribute this to the redundancy of parameters in LLMs, where core capabilities can be largely retained even when only a subset of layers is preserved.

Iteration Order	Math (acc)		Avg.
	GSM8K	MATH	
Rotation	67.10	11.96	39.53
Math First	**67.48**	**14.38**	**40.93**
Math Last	61.71	2.06	31.89
LM First	63.15	3.14	33.15

Table 6: Performance comparison of different merging orders in iterative merging.

5 Related Work

Model merging has gained popularity in LLM research (Zhou et al., 2024; Yang et al., 2024b). By amalgamating multiple homologous LLMs into a single model, this technique has been applied to address several challenges, such as building multi-task experts (Cai et al., 2023; Wan et al., 2024b), detoxification (Hu et al., 2024; Zhang et al., 2023), and preference alignment (Lin et al., 2024; Rame et al., 2024). Model merging methods are primarily based on two fundamental approaches: weight averaging (Wortsman et al., 2022) and task arithmetic (Ilharco et al., 2023).

Weight-based model merging methods design rules or matrices to determine merging coefficients. For example, RegMean (Jin et al., 2022) optimizes a linear regression problem for linear weights, Fisher-Merging (Matena and Raffel, 2022) uses the Fisher information matrix to assess parameter importance. Some works explore the space of these coefficients using parameter searching algorithms, such as evolutionary algorithms (Akiba et al., 2025) or Bayesian optimization (Liu et al., 2024). Although these methods demonstrate effectiveness, they suffer from inefficiency: parameter search is time-consuming, and solving the objectives requires substantial computation resources.

Subspace-based model merging methods focus on eliminating insignificant parameters and merging sparse models within the parameter subspace to reduce interference. TIES (Yadav et al., 2024) trims individual models based on parameter magnitudes, while Model Breadcrumbs (Davari and Belilovsky, 2025) refines this by removing both low-magnitude and high-magnitude outliers. DARE (Yu et al., 2024b) emphasizes the importance of rescaling after sparsification, and TALL-Mask (Wang et al., 2024) creates task-specific mask matrices based on predefined thresholds to filter out irrelevant parameters. However, these methods are limited to specific patterns, such as sign conflicts or threshold-based filtering, and magnitude-based sparsification remains suboptimal. To better address the interference problem, we propose a solution based on parameter saliency sparsification and a mutually exclusive iterative merging framework.

6 Conclusion

In this work, we propose OBIM, a novel merging method for LLMs that selectively retains representative delta weights based on saliency and iteratively integrates task vectors to reduce both intra-model and inter-model interference. OBIM achieves state-of-the-art performance in merging SFT models and post-pretraining checkpoints, demonstrating its effectiveness and versatility. Extensive ablation studies further validate its key components. Additionally, OBIM is computationally efficient and memory-light, making it well-suited for real-world applications.

7 Limitations

While our work provides valuable insights into LLM merging, several limitations should be noted: (1) The application of OBIM relies on models with identical architectures and shared initializations, limiting its applicability to diverse model types. (2) Although efficient, OBIM requires an additional validation set and incurs extra computational costs for saliency computation compared to magnitude-based methods. (3) Our analysis primarily focuses on interference from the perspective of parameter aggregation, with limited theoretical exploration, highlighting the need for further research in future work.

References

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. 2025. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, pages 1–10.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Tian Bowen, Lai Songning, Wu Jiemin, Shuai Zhihao, Ge Shiming, and Yue Yutao. 2024. Beyond task vectors: Selective task arithmetic based on importance metrics. *arXiv preprint arXiv:2411.16139*.

Ruisi Cai, Zhenyu Zhang, and Zhangyang Wang. 2023. Robust weight signatures: gaining robustness as easy as patching weights? In *International Conference on Machine Learning*, pages 3495–3506. PMLR.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. 2020. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, 53:5113–5155.

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

MohammadReza Davari and Eugene Belilovsky. 2025. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision*, pages 270–287. Springer.

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

Jasper Dekoninck, Marc Fischer, Luca Beurer-Kellner, and Martin Vechev. 2023. Controlled text generation via language model arithmetic. *arXiv preprint arXiv:2311.14479*.

Guodong DU, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim Kuan Goh, Ho-Kin Tang, Daojing He, and Min Zhang. 2024. Parameter competition balancing for model merging. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Elias Frantar and Dan Alistarh. 2022. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE.

Yang He and Lingao Xiao. 2023. Structured pruning for deep convolutional neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Xinshuo Hu, Dongfang Li, Baotian Hu, Zihao Zheng, Zhenyu Liu, and Min Zhang. 2024. Separate the wheat from the chaff: Model deficiency unlearning via parameter-efficient module operation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18252–18260.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*.

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

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. 2024. Model stock: All we need is just a few fine-tuned models. In *European Conference on Computer Vision*, pages 207–223. Springer.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2022. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 11.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. 2024. Datacomp-lm: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Tzu-Han Lin, Chen-An Li, Hung-yi Lee, and Yun-Nung Chen. 2024. Dogerm: Equipping reward models with domain knowledge through model merging. *arXiv preprint arXiv:2407.01470*.

Deyuan Liu, Zecheng Wang, Bingning Wang, Weipeng Chen, Chunshan Li, Zhiying Tu, Dianhui Chu, Bo Li, and Dianbo Sui. 2024. Checkpoint merging via bayesian optimization in llm pretraining. *arXiv preprint arXiv:2403.19390*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

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

Alexandre Rame, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. 2024. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *Advances in Neural Information Processing Systems*, 36.

Ken Shoemake. 1985. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

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*.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024a. Knowledge fusion of large language models. *arXiv preprint arXiv:2401.10491*.

Fanqi Wan, Longguang Zhong, Ziyi Yang, Ruijun Chen, and Xiaojun Quan. 2024b. Fusechat: Knowledge fusion of chat models. *arXiv preprint arXiv:2408.07990*.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jiménez, François Fleuret, and Pascal Frossard. 2024. Localizing task information for improved model merging and compression. In *International Conference on Machine Learning*.

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.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang,

Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024b. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024a. Extend model merging from fine-tuned to pre-trained large language models via weight disentanglement. *arXiv preprint arXiv:2408.03092*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024b. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Jinghan Zhang, Junteng Liu, Junxian He, et al. 2023. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, 36:12589–12610.

Yuyan Zhou, Liang Song, Bingning Wang, and Weipeng Chen. 2024. Metagpt: Merging large language models using model exclusive task arithmetic. *arXiv preprint arXiv:2406.11385*.

A Experimental Details

A.1 Details of Baselines and Ablation Study

To provide a better understanding of the baselines, we outline the methods used in previous works for addressing intra-model and inter-model interference. We compare these methods with our approach in Table 7, highlighting the innovation of our method. At the bottom of Table 7, we also provide the implementation of the methods used in our ablation study.

Below is a brief introduction to each component.

- **Magnitude Pruning**: Retains parameters with the largest magnitude values.
- **Random Drop and Rescale**: Filters parameters using a Bernoulli distribution and rescales the remaining ones according to the drop rate.
- **Stochastic Magnitude Pruning**: Assigns magnitude values to probabilities and retains parameters according to these probabilities.
- **Disjoint Mean**: Elects parameters at each position based on the direction of summation, then averages the parameters along that direction.
- **Consensus Mask**: Selects parameters using a mask constructed by measuring the l_1 distance to the target task vector.

Method	Intra-Model	Inter-Model
TA	/	/
TIES	Magnitude Pruning	Disjoint Mean
DARE	Random Drop and Rescale	Disjoint Mean
DELLA	Stochastic Magnitude Pruning	Disjoint Mean
TALL-Mask	/	Consensus Mask
OBIM	Saliency-based Pruning	Iterative Merging
TIES+OBM	Saliency-based Pruning	Disjoint Mean
TIES+IM	Magnitude Pruning	Iterative Merging

Table 7: Comparison of methods for addressing intra-model and inter-model interference.

A.2 Hyperparameter Configurations

In the SFT model merging experiments, the hyperparameters that need to be adjusted include the retention ratio of parameters in OBM ($n_k\%$) and the merging order (O). The search ranges and the optimal settings for each hyperparameter are provided in Table 8.

In the post-pretraining model merging scenario, the retention ratio $n_k\%$ for merging K checkpoints is set to $\frac{1}{K}$, and the merging order is set to *Rotation*.

A.3 Datasets for Post-Pretraining

We collected and processed a dataset of over 200B tokens comprising Japanese, Chinese, and English texts for post-pretraining. The dataset includes text from websites and publications, all of which are publicly available. Below are the details of the data sources:

- **Japanese**: C4-ja[6], CC100-ja[7], OSCAR-ja[8], CulturaX[9], Wikipedia-ja[10]
- **English**: FineWeb[11], Tiny-Textbooks[12], AutoMathText[13], Wikipedia-en[14]

[6]https://huggingface.co/datasets/systemk/c4-ja
[7]https://huggingface.co/datasets/statmt/cc100
[8]https://huggingface.co/datasets/ohtaman/oscar_ja_clean_filtered
[9]https://huggingface.co/datasets/uonlp/CulturalX
[10]https://huggingface.co/datasets/systemk/wiki-ja
[11]https://huggingface.co/datasets/HuggingFaceFW/fineweb
[12]https://huggingface.co/datasets/nampdn-ai/tiny-textbooks
[13]https://huggingface.co/datasets/math-ai/AutoMathText
[14]https://huggingface.co/datasets/blo05/cleaned_wiki_en

H-Param	Searching Range	Optimal Setting
$n_k\%$	[0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5]	{LM: 0.4, Math: 0.45, Code: 0.1}
O	[Rotation, LM First, Math First, Code First, LM Last, Math last, Code Last]	Code Last

Table 8: Hyperparameter search ranges and optimal settings for the SFT model merging experiment.

- **Chinese**: CLUECorpus[15], SkyPile[16], MAP-CC[17], Wanjuan[18], Wikipedia-zh[19]
- **Parallel Corpus**[20]: CCMatrix, JParaCrawl, WikiMatrix

We follow the pretraining data processing approach of DataComp-LM (Li et al., 2024). The data processing pipeline mainly consists of three steps:

- **Text Extraction**: We extract clean text from raw content using rule-based tools such as HTML parsers and regular expressions.
- **Deduplication**: We apply both locality-sensitive hashing (LSH) deduplication and semantic deduplication to remove redundant data.
- **Quality Filtering**: We employ a `FastText`[21] binary classifier for each language to assess and filter data quality.

After processing, we retain approximately 70B tokens for each language and 1B tokens from the parallel corpus as our post-pretraining dataset.

We then trained *Qwen2-7B* on this dataset using 64 A800 (80G) GPUs, with a training batch size of 4 million tokens per step for two weeks. Checkpoints were saved every 1,000 steps, and the total training duration was approximately 50,000 steps.

We will release the trained model as open-source after the paper is accepted.

A.4 Details of Validation Set

The sample size of the validation set for each benchmark is provided in Table 9.

[15]https://github.com/brightmart/nlp_chinese_corpus
[16]https://huggingface.co/datasets/Skywork/SkyPile-150B
[17]https://huggingface.co/datasets/m-a-p/MAP-CC
[18]https://huggingface.co/datasets/facat/wanjuan
[19]https://huggingface.co/datasets/shaowenchen/wiki_zh
[20]https://opus.nlpl.eu
[21]https://github.com/facebookresearch/fastText

Model	Benchmark	Sample Size	Total
LM	AlpacaEval	50	335
	MMLU	285	
Math	GSM8K	50	100
	MATH	50	
Code	HumanEval	0	50
	MBPP	50	
Post-pretrained Model	MMLU	285	763
	C-Eval	260	
	JP-LMEH	218	

Table 9: Validation Set Sizes. Note that HumanEval is excluded since it only provides a test set.

A.5 Computational Resources

We measured the GPU usage and time cost in OBIM, as shown in Table 11. GPUs used in our experiments is NVIDIA-A800 (80G). The total time is divided into three parts: the computation time for $\mathbf{X}^l$ across all layers, which depends on the size of the validation set; the time required for saliency computation; and the time for iterative merging.

B Results on Merging Multiple Models

Table 10 presents the full results for merging varying numbers of post-pretrained models using OBIM and DARE.

Count	Method	CEVAL	MMLU	JP-LMEH	Avg.
2	OBIM	**81.87**	**69.41**	71.85	**74.38**
	DARE	81.20	69.16	**71.92**	74.09
3	OBIM	**80.98**	69.32	**71.80**	**74.03**
	DARE	79.79	**69.51**	71.02	73.44
4	OBIM	**80.46**	**69.89**	**72.14**	**74.16**
	DARE	78.97	68.85	71.21	73.01
5	OBIM	**79.72**	**69.40**	**71.90**	**73.69**
	DARE	79.19	68.98	70.92	73.03
6	OBIM	**79.72**	**69.66**	**71.90**	**73.76**
	DARE	78.90	69.05	70.99	72.98

Table 10: Detailed performance comparison of OBIM and DARE across different numbers of merged models.

C Ethics Statement

This paper focuses on model merging techniques for Large Language Models (LLMs) to enhance their adaptability. While our work does not directly

Model Count	Model Size	GPUs	Computing Speed of $\mathbf{X}^l$	Saliency Computation Time	Merging Time
3	13B	2	0.40 (s/sample)	50 (s)	2'03
6	7B	1	0.36 (s/sample)	20 (s)	2'01

Table 11: Computational resources for model merging with OBIM.

introduce new risks, it inherits the broader societal concerns associated with LLMs, such as AI safety, reliability, and potential biases in generated content. Beyond these considerations, we do not foresee additional ethical concerns arising from our work.