Mixup Model Merge: Enhancing Model Merging Performance through Randomized Linear Interpolation

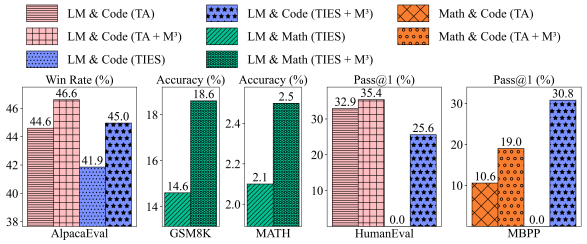
Anonymous ACL submission

Abstract

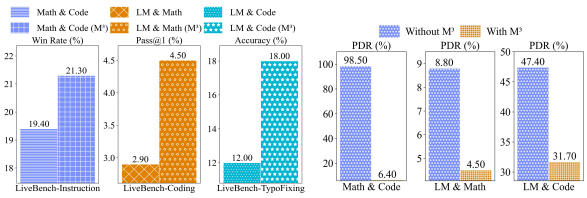
Model merging integrates the parameters of multiple models into a unified model, combining their diverse capabilities. Existing model merging methods are often constrained by fixed parameter merging ratios. In this study, we propose Mixup Model Merge (M^3), an innovative approach inspired by the Mixup data augmentation technique. This method merges the parameters of two large language models (LLMs) by randomly generating linear interpolation ratios, allowing for a more flexible and comprehensive exploration of the parameter space. Extensive experiments demonstrate the superiority of our proposed M^3 method in merging fine-tuned LLMs: (1) it significantly improves performance across multiple tasks, (2) it enhances LLMs' out-of-distribution (OOD) robustness and adversarial robustness, (3) it achieves superior results when combined with sparsification techniques such as DARE, and (4) it offers a simple yet efficient solution that does not require additional computational resources. In conclusion, M^3 is a simple yet effective model merging method that significantly enhances the performance of the merged model by randomly generating contribution ratios for two fine-tuned LLMs.

1 Introduction

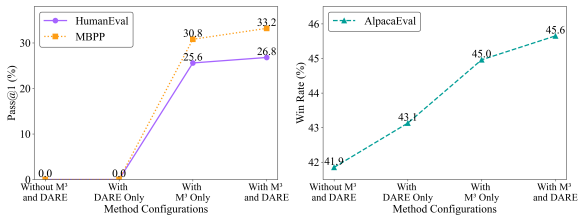
In the field of Natural Language Processing (NLP), the emergence of large language models (LLMs) (Brown et al., 2020; Touvron et al., 2023; OpenAI, 2023; Chowdhery et al., 2023) represents a revolutionary breakthrough. With their remarkable capabilities, these models have demonstrated outstanding performance across various tasks (Jiao et al., 2023; Chang et al., 2024b; Nam et al., 2024; Xing, 2024; Guo et al., 2024), significantly advancing NLP technologies.



(a) Performance of the merged models obtained through Task Arithmetic and TIES-Merging with/without M^3.



(b) (Left) Performance of merged models with/without M^3 on OOD datasets. (Right) Adversarial robustness (PDR) of merged models with/without M^3.



(c) The performance trend of the merged models obtained through the following methods: TIES-Merging without M^3 and DARE, TIES-Merging with DARE, TIES-Merging with M^3, and TIES-Merging with both M^3 and DARE.

Figure 1: (a) M^3 significantly boosts the model merging performance. (b) OOD robustness, and adversarial robustness of the merged models. (c) Combined with DARE, M^3 delivers even better results. LM, MATH, and Code refer to the WizardLM13B, WizardMath-13B, and llama-2-13b-code-alpaca models. TA stands for Task Arithmetic and TIES stands for TIES-Merging.

Supervised Fine-Tuning (SFT) is a crucial technique for adapting LLMs to specific tasks, refining their performance by training on domain-specific data (Hu et al., 2021; Ding et al., 2023; Xia et al., 2024). However, SFT requires substantial computational resources and long training times (Brown et al., 2020; Chang et al., 2024a). To address this

challenge, Model Merging has emerged as an efficient solution, fusing the parameters of multiple fine-tuned LLMs into a unified model with diverse capabilities, without the need for additional training or computational costs (Yang et al., 2024; Akiba et al., 2024). It effectively reduces the resource-intensive demands of SFT while preserving and even enhancing model performance.

A simple analogy for model merging is the Super Mario game, where the protagonist gains special abilities by absorbing power-up items. Similarly, merging model parameters integrates the strengths of different models, enabling more effective multi-task learning (Yu et al., 2024). However, existing model merging methods have some limitations, these methods heavily rely on predefined or heuristic parameter fusion strategies (Wortsman et al., 2022; Ilharco et al., 2022; Matena and Raffel, 2022; Jin et al., 2022; Yadav et al., 2023; Yu et al., 2024) such that they fail to fully explore the parameter space, thereby restricting the potential of the merged model.

To address this issue, we draw inspiration from Mixup (Zhang, 2017) and propose a novel technique called Mixup Model Merge (M^3). This method introduces randomness by dynamically adjusting the contribution ratios between models, making the model merging process more flexible and enabling a thorough exploration of the parameter space. M^3 further unlocks the potential of model merging, significantly enhancing the generalization performance of the merged model while improving its out-of-distribution (OOD) and adversarial robustness (Wang et al., 2023; Zhu et al., 2023). As shown in Figure 2, the implementation of M^3 is similar to the process of proportionally mixing magical potions in *Harry Potter*. Specifically, this method dynamically controls the parameter fusion ratio between two fine-tuned LLMs by randomly generating a linear interpolation ratio λ_m, where $\lambda_m \in (0, 1)$ and $\lambda_m \sim \text{Beta}(\alpha, \alpha)$. By adjusting the parameter α, we can precisely control the distribution of λ_m, allowing M^3 to explore the parameter space of model merging more deeply and thus fully unleash the potential of the models, leading to improved overall performance.

We conducted extensive experiments with three fine-tuned LLMs: WizardLM-13B (Xu et al., 2024), WizardMath-13B (Luo et al., 2023), and llama-2-13b-code-alpaca (Chaudhary, 2023), which specialize in instruction following, mathematical reasoning, and code generation, respectively. Inspired by Mixup's effectiveness in enhancing the robustness of neural networks when handling corrupted labels or adversarial examples (Zhang, 2017), we further performed comprehensive evaluations on LiveBench (White et al., 2024) and PromptBench (Zhu et al., 2024) to validate the potential of M^3 in improving the OOD robustness and adversarial robustness of merged models. The experimental results demonstrate that our proposed M^3 method can significantly improve merged models' performance across various tasks (as shown in Figure 1a), enhance the OOD and adversarial robustness of the merged models (as shown in Figure 1b), and boost model merging when combined with sparsification techniques like DARE (Yu et al., 2024) (as shown in Figure 1c).

2 Related Works

Model Merging Model merging is a technique that integrates the parameters of multiple models to create a unified model with enhanced or diverse capabilities (Wortsman et al., 2022; Ilharco et al., 2022; Matena and Raffel, 2022; Jin et al., 2022; Yadav et al., 2023; Yu et al., 2024; Lin et al., 2024). Task arithmetic (Ilharco et al., 2022) leverages task vectors for model merging through arithmetic operations, incorporating a predefined scaling term to weight the contribution of different models. Fisher Merging (Matena and Raffel, 2022) performs parameter fusion by applying weights derived from the Fisher information matrix (Fisher, 1922), resulting in more precise parameter integration. TIES-Merging (Yadav et al., 2023) addresses task conflicts by removing low-magnitude parameters, resolving sign disagreements, and merging only the parameters that align with the final agreed-upon sign. In (Yu et al., 2024), it is found that LLMs can enhance their capabilities through model merging. Additionally, it introduces DARE, a method for sparsifying the delta parameters of the model (Ilharco et al., 2022), significantly improving the performance of various model merging techniques.

Mixup Mixup is proposed to enhance the generalization ability of deep learning models by surpassing traditional Empirical Risk Minimization (ERM) (Zhang, 2017). It is a simple, data-agnostic augmentation technique that trains models using virtual examples created by linearly interpolating pairs of random examples and their corresponding labels. Rooted in the Vicinal Risk Minimization (VRM) principle (Chapelle et al., 2000), this ap-

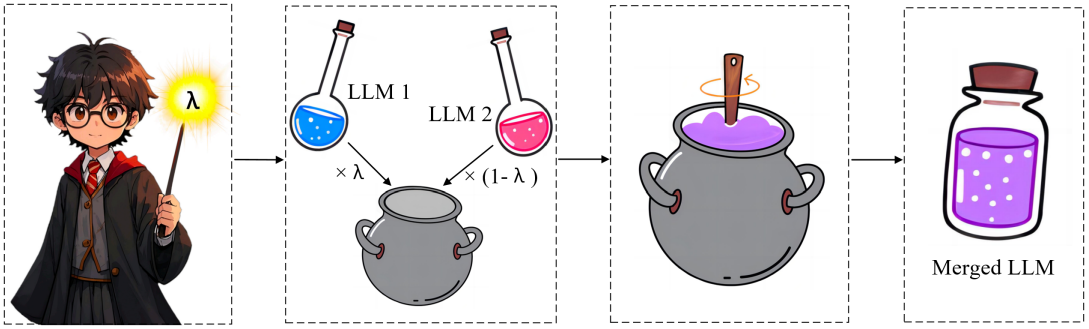


Figure 2: Implementation of m^3: A process analogous to proportionally mixing magical potions in harry potter. The proposed method controls the contribution ratio between two fine-tuned llms by randomly generating a linear interpolation ratio λ_m, where $\lambda_m \in (0, 1)$ and $\lambda_m \sim \text{Beta}(\alpha, \alpha)$. The distribution of λ_m is controlled by adjusting α.

proach improves generalization across a variety of datasets (Russakovsky et al., 2015; Krizhevsky et al., 2009; Warden, 2017; Asuncion et al., 2007), and helps reduce overfitting, sensitivity to adversarial examples, and training instability, all with minimal computational cost. Given two samples (x_i, y_i) and (x_j, y_j), Mixup generates a new sample using the following formulas:

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda) x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda) y_j \end{aligned} \tag{1}$$

$\tilde{x}$ and $\tilde{y}$ represent the generated synthetic sample and its label, respectively, with λ determining their interpolation ratio, typically ranging from 0 to 1. Here, λ is a hyperparameter sampled from a Beta distribution, i.e., $\lambda \sim \text{Beta}(\alpha, \alpha)$, where α controls the strength of the interpolation between feature-target pairs. Inspired by Mixup, we propose a novel model merging method, M^3, which generates the parameters of the merged model by performing linear interpolation between the parameters of two fine-tuned LLMs, with the interpolation ratio being random.

3 Methodology

3.1 Model Merging Problem

Following Yu et al. (2024), we focus on merging fine-tuned LLMs that have been optimized from the same pre-trained backbone (Touvron et al., 2023). Specifically, we aim to fuse the parameters of these LLMs to create a unified model capable of handling multiple tasks. In this context, we restrict our attention to the merging of two models, as the mixup theory, which forms the basis of our approach, is generally applied to two entities.

Given two tasks t_1 and t_2 with the corresponding fine-tuned LLMs having parameters $\theta^{t_1}_{\text{SFT}}$ and $\theta^{t_2}_{\text{SFT}}$, model merging aims to combine the parameters of these two models into a single model with parameters θ_M. The resulting model should be able to effectively perform both tasks simultaneously, leveraging the knowledge learned from each individual model.

3.2 Mixup Model Merge

Inspired by Mixup (Zhang, 2017), we propose a simple yet effective model merging method called Mixup Model Merge (M^3). Unlike existing methods that use fixed merging ratios, M^3 generates the model contribution ratio randomly, harnessing randomness to inject fresh vitality into the model merging process. Specifically, M^3 further explores the parameter space of the merged model to unlock the potential of model merging.

To further elaborate, given two fine-tuned LLMs with parameters $\{\theta^{t_1}_{\text{SFT}}, \theta^{t_2}_{\text{SFT}}\}$, we combine M^3 with established model merging methods to fuse these parameters and obtain a single merged model with parameters θ_M. As illustrative examples, we consider two widely used merging methods: Average Merging (Wortsman et al., 2022) and Task Arithmetic (Ilharco et al., 2022).

The official computation process for Average Merging is described as follows:

$$\theta_M = \frac{1}{2}\left(\theta^{t_1}_{\text{SFT}} + \theta^{t_2}_{\text{SFT}}\right) \tag{2}$$

The official computation process for Task Arith-

metic is:

$$\theta_M = \theta_{\text{PRE}} + \lambda \cdot (\delta^{t_1} + \delta^{t_2}) = \theta_{\text{PRE}} + \lambda \cdot \sum_{i=1}^{2} (\theta_{\text{SFT}}^{t_i} - \theta_{\text{PRE}}), \quad (3)$$

where $\theta_{\text{PRE}} \in \mathbb{R}^d$ represents the parameters of the pre-trained language model (PLM), such as Llama 2 (Touvron et al., 2023). λ is a scaling factor that weights the contribution of each model during the merging process. δ^{t_i} denotes the delta parameter (Ilharco et al., 2022), which is defined as the difference between the parameters of the language models (LMs) before and after SFT, i.e., $\delta^t = \theta_{\text{SFT}}^t - \theta_{\text{PRE}} \in \mathbb{R}^d$, where t refers to task t.

When introducing M^3, the process for Average Merging is reformulated as:

$$\theta_M = \lambda_m \theta_{\text{SFT}}^{t_1} + (1 - \lambda_m)\theta_{\text{SFT}}^{t_2}, \quad (4)$$

while the process for Task Arithmetic is reformulated as:

$$\theta_M = \theta_{\text{PRE}} + \lambda_m \delta^{t_1} + (1 - \lambda_m)\delta^{t_2}, \quad (5)$$

where λ_m determines the linear interpolation ratio between the two fine-tuned LLMs, and is generally a value between 0 and 1. λ_m is sampled from a Beta distribution, typically $\lambda_m \sim \text{Beta}(\alpha, \alpha)$, where α controls the shape of the Beta distribution.

The hyperparameter α for M^3 is selected from the range $[0.2, 0.4, 0.5, 1, 2, 3, 5]$. As shown in Figure 3: (1) When $\alpha = 1$, λ follows a uniform distribution, meaning all values within the range $(0, 1)$ are equally likely to be sampled. (2) When $\alpha < 1$, the distribution of λ exhibits a bimodal shape, with higher probabilities near the extremes (0 and 1). This indicates that the merged model is more likely to be dominated by one of the two models. (3) When $\alpha > 1$, the distribution of λ becomes concentrated around the middle (e.g., 0.5), resulting in more balanced contributions from both models.

3.3 Theoretical Analysis

Mixup performs linear interpolations between data samples and their labels in the data space (Zhang, 2017). Similarly, M^3 can be viewed as applying random linear interpolation in the parameter space, where the interpolation occurs between the parameters of two fine-tuned models trained on different tasks. M^3 represents a natural extension of the Vicinal Risk Minimization (VRM) principle (Chapelle et al., 2000) into the model parameter

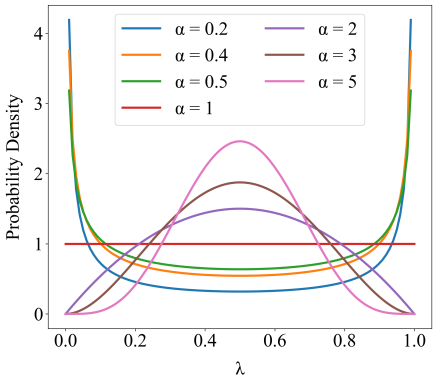


Figure 3: The Beta distribution visualization for different α values.

space. In data space, VRM introduces a vicinal distribution to simulate the true data distribution, thereby increasing the diversity of training data and enhancing the model's generalization ability. Similarly, M^3 extends this principle by constructing a virtual neighborhood in the model parameter space between two task-specific models, combining their knowledge in a way that is more natural and balanced.

In the context of model merging, interpolating between two sets of model parameters, $\theta_{\text{SFT}}^{t_1}$and $\theta_{\text{SFT}}^{t_2}$, defines a new neighborhood distribution in the parameter space, which can be expressed as:

$$P_\nu(\theta_M) = \int \nu(\theta_M \mid \theta_{\text{SFT}}^{t_1}, \theta_{\text{SFT}}^{t_2})\, d\theta_M \times P(\theta_{\text{SFT}}^{t_1}) P(\theta_{\text{SFT}}^{t_2}), \quad (6)$$

where $P_\nu(\theta_M)$ represents the probability distribution of the new model parameters θ_M generated through interpolation. The function $\nu(\theta_M \mid \theta_{\text{SFT}}^{t_1}, \theta_{\text{SFT}}^{t_2})$ defines the range and behavior of θ_M, depending on the interpolation strategy (such as linear interpolation). Additionally, $P(\theta_{\text{SFT}}^{t_1})$ and $P(\theta_{\text{SFT}}^{t_2})$ represent the prior distributions of the parameters of the two models, respectively.

Under the linear interpolation strategy, the function $\nu(\theta_M \mid \theta_{\text{SFT}}^{t_1}, \theta_{\text{SFT}}^{t_2})$ is defined as:

$$\theta_M = \lambda_m \cdot \theta_{\text{SFT}}^{t_1} + (1 - \lambda_m) \cdot \theta_{\text{SFT}}^{t_2}, \quad (7)$$

where $\lambda_m \in (0, 1)$ is the interpolation ratio. The corresponding distribution $P_\nu(\theta_M)$ is expressed as:

$$P_\nu(\theta_M) = \int \delta\left(\theta_M - \left(\lambda_m \cdot \theta_{\text{SFT}}^{t_1} + (1 - \lambda_m) \cdot \theta_{\text{SFT}}^{t_2}\right)\right) \cdot P(\theta_{\text{SFT}}^{t_1}) P(\theta_{\text{SFT}}^{t_2})\, d\theta_{\text{SFT}}^{t_1} d\theta_{\text{SFT}}^{t_2}, \quad (8)$$

where $\delta(\cdot)$ is the Dirac delta function, ensuring that θ_M satisfies the linear interpolation rule. By using this linear interpolation method, the parameters $\theta_{\text{SFT}}^{t_1}$ and $\theta_{\text{SFT}}^{t_2}$ are combined in the parameter space, forming a new neighborhood distribution $P_\nu(\theta_M)$. This process effectively merges the knowledge of both models into a unified parameter space, thus achieving robust performance across different tasks while maintaining the original strengths of the models.

By interpolating the parameters, M^3 encourages the model to learn smoother decision boundaries. In this context, smoother decision boundaries can be understood as the boundaries between different tasks, such as instruction following, mathematical reasoning, and code generation, where the model must understand and adapt to each task differently. In tasks t_1 and t_2, M^3 creates a virtual neighborhood that seamlessly integrates knowledge from both tasks. This approach prevents the merged model from overfitting task-specific details, ensuring a balanced and effective performance across all tasks.

Secondly, M^3 introduces a linear inductive bias in the parameter space, encouraging the merged model parameters to lie on a linear manifold between the two source models. This linear structure offers significant advantages in terms of simplicity and generalization. According to Occam's Razor, simpler solutions tend to generalize better. By performing linear interpolation between two sets of parameters, M^3 avoids unnecessary complexity in the model merging process, leading to a more straightforward and efficient solution.

Thirdly, M^3 can improve the performance of the merged model across multiple tasks by mitigating task conflicts. Different tasks may require conflicting parameter values, which can lead to performance degradation on one task while optimizing for another. Linear interpolation helps balance these conflicts, resulting in a model that performs well among all tasks.

4 Experiments

4.1 Experimental Setup

Task-Specific Fine-Tuned LLMs and Datasets Following the experimental setup given in Yu et al. (2024), we select three task-specific fine-tuned LLMs: WizardLM-13B (Xu et al., 2024), WizardMath-13B (Luo et al., 2023), and llama-2-13b-code-alpaca (Chaudhary, 2023), all of which use Llama-2-13b (Touvron et al., 2023) as the pre-trained backbone. These models are respectively designed for instruction-following, mathematical reasoning, and code generation tasks. To evaluate the instruction-following task we use AlpacaEval (Li et al., 2023). For testing mathematical reasoning task, we employ GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). For estimating the performance of code-generating task, we use HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). More details of these LLMs and datasets can be found in Appendix A.1.

The Benchmarks for evaluating Out-of-Distribution and Adversarial Robustness To assess OOD robustness, we evaluate math & code, LM & math, and LM & code models using instruction following (LiveBench-Instruction), coding (LiveBench-Coding), and language comprehension (LiveBench-TypoFixing) category in LiveBench (White et al., 2024), respectively. More details on OOD benchmarks are given in Appendix A.3.

We utilize the Adversarial Prompt Attacks module in PromptBench (Zhu et al., 2024) to assess the robustness of LLMs against adversarial prompts. Specifically, we employ three attack methods: DeepWordBug (character-level) (Gao et al., 2018), BERTAttack (word-level) (Li et al., 2020), and StressTest (sentence-level) (Naik et al., 2018). The evaluation is conducted on two datasets supported by PromptBench: SST2 (sentiment analysis) (Socher et al., 2013) and CoLA (grammatical correctness) (Warstadt, 2019). For more details on PromptBench and attack methods, please refer to Appendix A.4.

Evaluation Metrics We calculate win rate for AlpacaEval and LiveBench-Instruction, zero-shot accuracy for GSM8K and MATH, pass@1 for HumanEval, MBPP and LiveBench-Coding, Matthews correlation coefficient (MCC) for CoLA, accuracy for SST2, and zero-shot accuracy for LiveBench-TypoFixing.

Implementation Details Unless otherwise specified, the details of the model merging experiments are consistent with Yu et al. (2024). The hyperparameter α for M^3 is chosen from the range $[0.2, 0.4, 0.5, 1, 2, 3, 5]$. For a detailed description of the hyperparameter settings in model merging methods, please refer to Appendix A.2. Additionally, all experiments are conducted on NVIDIA GeForce RTX 4090 GPUs.

4.2 Merging Task-Specific Fine-Tuned LLMs

We integrate M^3 into three prominent model merging techniques: Average Merging, Task Arithmetic, and TIES-Merging. The performance of merging task-specific fine-tuned LLMs is presented in Table 1.

From Table 1, we obtain the following observations: 1) M^3 generally enhances Average Merging, Task Arithmetic, and TIES-Merging when merging fine-tuned LLMs. For example, the improvements achieved by Average Merging with M^3 for Math & Code are 7.43% on GSM8K, 3.74% on Math, and 11.0% on MBPP. For LM & Code, Average Merging with M^3 shows improvements of 7.31% on AlpacaEval, 7.32% on HumanEval, and 2.4% on MBPP. Task Arithmetic with M^3 results in improvements of 2.0% on AlpacaEval and 2.44% on HumanEval for LM & Code, and 10.4% on MBPP for Math & Code. TIES-Merging with M^3 achieves an improvement of 4.01% for LM & Math on GSM8K. For LM & Code, TIES-Merging with M^3 shows significant improvements of 3.11% on AlpacaEval, 25.61% on HumanEval, and 30.8% on MBPP. 2) Compared to Task Arithmetic, Average Merging and TIES-Merging tend to benefit more from M^3. This is because both Average Merging and TIES-Merging use a fixed merging ratio of 1/2, whereas Task Arithmetic allows the merging ratio to vary within the range $[0.5, 1.0]$. Consequently, the randomness introduced by M^3 in the merging ratio has a more pronounced impact on Average Merging. This further highlights the critical role of an effective merging ratio in determining the performance of the merged model. 3) Yu et al. (2024) has indicated that the suboptimal results of merging WizardMath-13B with llama-2-13b-code-alpaca are due to llama-2-13b-code-alpaca not being well fine-tuned for code generation. In this context, the proposed M^3 approach improves the pass@1 score on MBPP by 10.4% for the merged model of WizardMath-13B and llama-2-13b-code-alpaca. The improvement demonstrates that when one of the fine-tuned models to be merged is not well fine-tuned for the specific task, M^3 can effectively unlock the potential of both models, maximizing the performance of the merged model. The M^3 approach helps mitigate the impact of suboptimal fine-tuning on model merging performance.

4.3 Model Robustness

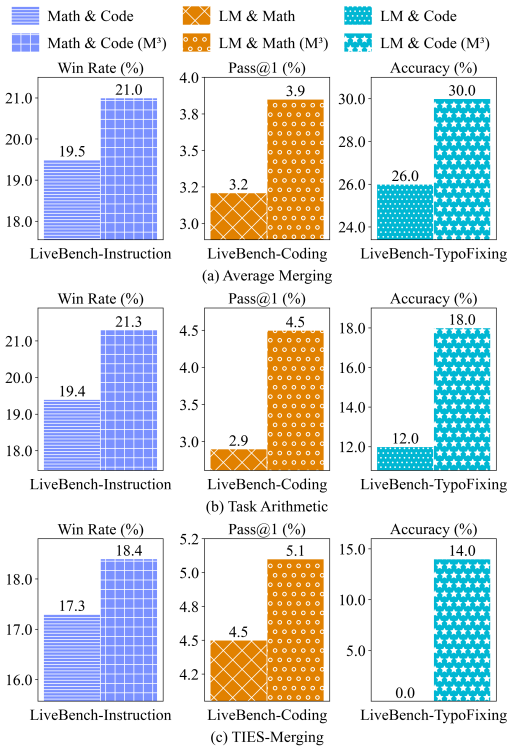


Figure 4: Performance of merged models (Math & Code, LM & Math, and LM & Code) using three model merging methods (Average Merging, Task Arithmetic, and TIES-Merging) on OOD datasets.

Out-of-distribution robustness To ensure that the evaluation datasets are as representative as possible of OOD data, we select datasets with sufficiently recent release dates and ensure they cover domains that fine-tuned LLMs have not been specifically trained on. Consequently, Math & Code is evaluated on LiveBench-Instruction, LM & Math on LiveBench-Coding, and LM & Code on LiveBench-TypoFixing. The performance of the merged LLMs is shown in Figure 4. As illustrated in Figure 4, M^3 consistently enhances the performance of merged models—Math & Code, LM & Math, and LM & Code—on OOD datasets. Specifically, when Task Arithmetic is combined with M^3, the Math & Code model demonstrates a 1.9% improvement in win rate on LiveBench-Instruction, the LM & Math model achieves a 1.6% increase in pass@1 on LiveBench-Coding, and the LM & Code model shows a significant 6% boost in accuracy on LiveBench-TypoFixing. Similarly, when Average Merging is combined with M^3, the Math & Code model attains a 1.5% improvement in win rate on LiveBench-Instruction, the LM & Math model achieves a 0.7% increase

Merging Methods	Models	Use Model Mixup	Use DARE	Instruction Following	Mathematical Reasoning		Code Generating	
				AlpacaEval	GSM8K	MATH	HumanEval	MBPP
/	LM	No	No	45.14	2.20	0.04	36.59	34.00
	Math	No	No	/	64.22	14.02	/	/
	Code	No	No	/	/	/	23.78	27.60
Average Merging	LM & Math	No	No	**45.28**	66.34	13.40	/	/
		Yes	No	44.40	66.26	13.80	/	/
		No	Yes	44.22	66.57	12.96	/	/
		Yes	Yes	43.53	**66.57**	**14.12**	/	/
	LM & Code	No	No	36.60	/	/	29.88	32.00
		Yes	No	**43.91**	/	/	**37.20**	34.40
		No	Yes	38.81	/	/	31.71	32.40
		Yes	Yes	40.31	/	/	36.59	**37.00**
	Math & Code	No	No	/	56.17	10.28	8.53	8.20
		Yes	No	/	63.61	**14.02**	8.54	19.20
		No	Yes	/	56.18	10.28	6.10	7.80
		Yes	Yes	/	**64.97**	13.54	**9.76**	**21.20**
Task Arithmetic	LM & Math	No	No	45.78	66.34	13.40	/	/
		Yes	No	41.65	66.34	**13.74**	/	/
		No	Yes	**49.00**	66.64	13.02	/	/
		Yes	Yes	44.90	**67.32**	13.74	/	/
	LM & Code	No	No	44.64	/	/	32.93	33.60
		Yes	No	**46.64**	/	/	35.37	33.80
		No	Yes	41.47	/	/	35.98	33.00
		Yes	Yes	45.20	/	/	**35.98**	**35.20**
	Math & Code	No	No	/	64.67	13.98	8.54	8.60
		Yes	No	/	63.53	13.94	7.93	**19.00**
		No	Yes	/	65.05	13.96	**10.37**	9.80
		Yes	Yes	/	**65.13**	**14.32**	8.54	18.00
TIES-Merging	LM & Math	No	No	38.63	14.56	2.12	/	/
		Yes	No	38.73	18.57	2.48	/	/
		No	Yes	37.92	18.04	2.34	/	/
		Yes	Yes	**39.93**	**19.26**	**2.82**	/	/
	LM & Code	No	No	41.85	/	/	0.0	0.0
		Yes	No	44.96	/	/	25.61	30.80
		No	Yes	43.13	/	/	0.0	0.0
		Yes	Yes	**45.65**	/	/	**26.83**	**33.20**
	Math & Code	No	No	/	64.67	13.68	9.15	22.60
		Yes	No	/	64.75	14.16	9.76	21.4
		No	Yes	/	**64.82**	13.88	**10.37**	**23.60**
		Yes	Yes	/	64.75	**14.78**	9.15	19.60

Table 1: Performance of merging task-specific LLMs WizardLM-13B (LM), WizardMath-13B (Math), and llama-2-13b-codealpaca (Code) on all the datasets. The best and second-best results are marked in bold and underlined fonts.

in pass@1 on LiveBench-Coding, and the LM & Code model exhibits a 4% enhancement in accuracy on LiveBench-TypoFixing. Finally, when TIES-Merging is applied alongside M^3, the Math & Code model achieves a 1.1% improvement in win rate on LiveBench-Instruction, the LM & Math model records a 0.6% increase in pass@1 on LiveBench-Coding, and the LM & Code model demonstrates a remarkable 14% improvement in accuracy on LiveBench-TypoFixing. These results underscore the robustness and versatility of M^3 in enhancing model performance across diverse merging strategies and OOD tasks.

Adversarial robustness We employ three Prompt Attack Methods supported by the promptbench codebase (DeepWordBug, BERTAttack, and StressTest) (Zhu et al., 2024) to evaluate the adversarial robustness of three merged models (Math & Code, LM & Math, and LM & Code) obtained through the task arithmetic method. To balance experimental effectiveness with computational efficiency, we randomly selected the positions of the three attacked words in the prompts when executing the DeepWordBug and BERTAttack attacks. Adversarial robustness is assessed using the Performance Drop Rate (PDR) (Zhu et al., 2023), where a lower PDR indicates stronger robustness. Further details on PDR can be found in Appendix D. The performance of the merged LLMs is shown in Table 2.

As shown in Table 2, M^3 improves the adversarial robustness of the merged models in most cases with the StressTest Prompt Attack Method. For example, with M^3, the PDR of Math & Code decreased by 3.2% on the SST2 dataset and by 92.12% on the CoLA dataset, while the PDR of LM & Code decreased by 30.36% on SST2 and

Model	Dataset	Use Mixup	Use Attack	Metric (%)	PDR (%)
Math & Code	SST2	No	No	57.68	38.97
			Yes	35.21	
		Yes	No	86.24	**35.77**
			Yes	55.39	
	CoLA	No	No	45.54	98.53
			Yes	0.67	
		Yes	No	71.72	**6.42**
			Yes	67.11	
LM & Math	SST2	No	No	92.78	**29.05**
			Yes	65.83	
		Yes	No	91.28	34.55
			Yes	59.75	
	CoLA	No	No	79.19	8.84
			Yes	72.20	
		Yes	No	80.54	**4.52**
			Yes	76.89	
LM & Code	SST2	No	No	10.55	38.04
			Yes	6.54	
		Yes	No	73.17	**7.68**
			Yes	67.55	
	CoLA	No	No	74.21	47.42
			Yes	39.02	
		Yes	No	74.78	**31.67**
			Yes	51.10	

Table 2: Adversarial Robustness of Merged Models on the SST2 and CoLA Datasets when Executing StressTest prompt attack method. The best and second-best results are marked in bold and underlined fonts.

by 15.75% on CoLA. Furthermore, in most cases, M^3 not only improves the adversarial robustness of the merged models but also enhances their performance metrics (accuracy and MCC) on the SST2 and CoLA datasets. Specifically, with M^3, Math & Code demonstrates a 28.56% increase in accuracy on SST2 and a 26.18% increase in MCC on CoLA, while LM & Code achieves a 62.62% increase in accuracy on SST2. These results show that M^3 effectively enhances both the adversarial robustness and the performance of the merged models in sentiment analysis and grammar correctness tasks. Detailed experimental results for the remaining Prompt Attack Methods (DeepWordBug and BERTAttack) are presented in Appendix B.

4.4 Mixup Model Merge with DARE

DARE is a model sparsification method proposed by (Yu et al., 2024), with a more detailed introduction provided in Appendix E. We combine M^3 and DARE with three model merging techniques, including Average Merging, Task Arithmetic, and TIES-Merging, to compare the effects of M^3 and DARE individually and explore their combined impact. The experimental results are presented in Table 1. Additionally, in the DARE method, the drop rate hyperparameter is set to 0.2.

From Table 1, we conclude that: 1) In most cases, M^3 outperforms DARE, with a particularly significant advantage on certain datasets. For instance, the Math & Code model achieves a pass@1 score of 9.8% on the MBPP dataset when combined with DARE, while this score increases to 19% when combined with M^3. This demonstrates that M^3 unlocks new potential in model merging by randomly generating merging ratios, leading to performance improvements that surpass those achieved by DARE. 2) Combining DARE and M^3 generally results in better model merging performance. For example, the LM & Math and LM & Code models, enhanced by TIES-Merging with M^3 and DARE, achieve the best performance on the test datasets. While only incorporating TIES-Merging with M^3 to these models, the enhanced models achieve the second best performance. This suggests that M^3 and DARE can complement each other. Moreover, in some cases, M^3 alone can deliver the best results, while DARE alone only achieves the best performance in very few cases, further demonstrating the superiority of M^3.

5 Conclusion

Inspired by the mixup method and the Vicinal Risk Minimization (VRM) principle, we propose Mixup Model Merge (M^3), a novel approach for merging fine-tuned LLMs by introducing randomness into the parameters linear interpolation process. Unlike traditional methods such as average merging and task arithmetic, M^3 leverages a Beta distribution to dynamically adjust the merging ratio, enabling more flexible exploration of the parameter space. Experimental results demonstrate that M^3 not only significantly enhances the performance of the merged model across various tasks but also improves its OOD and adversarial robustness. Furthermore, when combined with sparsification techniques such as DARE, our approach achieves even more favorable model merging outcomes. In summary, M^3 is a simple yet powerful technique that requires minimal computational resources. By merely adjusting the merging ratio, it produces a merged model with enhanced task-specific capabilities and robustness. This exciting discovery paves the way for further research into optimizing merging ratio selection in model merging processes.

6 Limitations

There are several limitations of the M^3 method: While it performs well for merging two models,

(1) its scalability when merging a larger number of models, especially those with significant differences, remains uncertain. Additionally, (2) due to the inherent randomness in the merging process, multiple attempts may be required to achieve a merged model that meets expectations. This unpredictability can lead to increased computational costs, particularly in large-scale applications, resulting in a significant rise in resource consumption. Finally, (3) Our method may also be extended to a wider range of applications, such as merging fine-tuned models with RLHF models to reduce the alignment tax (FINE-TUNING).

References

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

Arthur Asuncion, David Newman, et al. 2007. Uci machine learning repository.

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*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard. *Hugging Face*.

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.

Yupeng Chang, Yi Chang, and Yuan Wu. 2024a. Balora: Bias-alleviating low-rank adaptation to mitigate catastrophic inheritance in large language models. *arXiv preprint arXiv:2408.04556*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024b. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. 2000. Vicinal risk minimization. *Advances in neural information processing systems*, 13.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. *GitHub repository*.

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*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

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*.

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

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

TIVE LLM FINE-TUNING. Paft: Aparallel training paradigm for effec-tive llm fine-tuning.

Ronald A Fisher. 1922. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 222(594-604):309–368.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Chenlu Guo, Nuo Xu, Yi Chang, and Yuan Wu. 2024. Chbench: A chinese dataset for evaluating health in large language models. *arXiv preprint arXiv:2409.15766*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

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*.

Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Xing Wang, and Zhaopeng Tu. 2023. Is chatgpt a good translator? a preliminary study. *arXiv preprint arXiv:2301.08745*, 1(10).

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*.

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*.

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.

Yong Lin, Hangyu Lin, Wei Xiong, Shizhe Diao, Jianmeng Liu, Jipeng Zhang, Rui Pan, Haoxiang Wang, Wenbin Hu, Hanning Zhang, et al. 2024. Mitigating the alignment tax of rlhf. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 580–606.

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.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. *arXiv preprint arXiv:1806.00692*.

Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13.

R OpenAI. 2023. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5).

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

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*.

Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, et al. 2023. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. *arXiv preprint arXiv:2302.12095*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Pete Warden. 2017. Launching the speech commands dataset. *Google Research Blog*.

A Warstadt. 2019. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. 2024. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*.

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.

Tingyu Xia, Bowen Yu, Kai Dang, An Yang, Yuan Wu, Yuan Tian, Yi Chang, and Junyang Lin. 2024. Rethinking data selection at scale: Random selection is almost all you need. *arXiv preprint arXiv:2410.09335*.

Frank Xing. 2024. Designing heterogeneous llm agents for financial sentiment analysis. *ACM Transactions on Management Information Systems*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Resolving interference when merging models. *arXiv preprint arXiv:2306.01708*, 1.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024. 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. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Hongyi Zhang. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Gong, et al. 2023. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. In *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis*, pages 57–68.

Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. 2024. Promptbench: A unified library for evaluation of large language models. *Journal of Machine Learning Research*, 25(254):1–22.

A Detailed Experimental Settings

A.1 Task-Specific Fine-Tuned LLMs and Datasets Details

We conduct model merging experiments using three task-specific LLMs fine-tuned from Llama-2-13b:

- **WizardLM-13B** is an instruction-following model based on Llama-2-13b, designed to improve open-domain instruction-following. Using the Evol-Instruct method (Xu et al., 2024), it generates high-complexity instruction data to reduce human annotation and enhance generalization. The model undergoes supervised fine-tuning with AI-generated data, followed by refinement via RLHF. Evaluation results show that Evol-Instruct-generated instructions outperform human-written ones, and WizardLM-13B surpasses ChatGPT in high-complexity tasks. In GPT-4 automated evaluation, it achieves over 90% of ChatGPT's performance in 17 out of 29 tasks, demonstrating the effectiveness of AI-evolved instruction fine-tuning (Xu et al., 2024).

- **WizardMath-13B**, optimized from Llama-2-13b, is designed for mathematical reasoning and enhances Chain-of-Thought (CoT) (Wei et al., 2022) capabilities. It uses Reinforcement Learning from Evol-Instruct Feedback to evolve math tasks and improve reasoning. Trained on GSM8K and MATH datasets, it excels in both basic and advanced math problems. In evaluations, WizardMath-Mistral 7B outperforms all open-source models with fewer training data, while WizardMath 70B surpasses GPT-3.5-Turbo, Claude 2, and even early GPT-4 versions in mathematical reasoning tasks.

- **llama-2-13b-code-alpaca** is a code generation model fine-tuned from Llama-2-13b, designed to enhance code understanding and generation. It follows the same training approach as Stanford Alpaca (Taori et al., 2023) but focuses on code-related tasks. The model is fine-tuned with 20K instruction-following code samples generated using the Self-Instruct method (Wang et al., 2022). However, as it has not undergone safety fine-tuning, caution is required when using it in production environments.

We use one dataset to evaluate the instruction-following task:

- **AlpacaEval** (Li et al., 2023) is an LLM-based automated evaluation metric that assesses model performance by testing on a fixed set of 805 instructions and computing the win rate of the evaluated model against a baseline. The evaluation process involves an LLM-based evaluator that compares the responses and determines the probability of

preferring the evaluated model. In this paper, we use AlpacaEval 2.0 (Dubois et al., 2024). To reduce costs, we use chatgpt_fn for evaluation.

We use two dataset to evaluate the mathematical reasoning task:

- **GSM8K** is a dataset of 8.5K high-quality, linguistically diverse grade school math word problems, designed to evaluate the multi-step mathematical reasoning abilities of large language models. It consists of 7.5K training problems and 1K test problems. In this paper, we use the 1K test set for evaluation (Cobbe et al., 2021).
- **MATH** is a dataset containing 12,500 competition-level math problems, designed to evaluate and enhance the problem-solving abilities of machine learning models. It consists of 7,500 training problems and 5,000 test problems. We use the 5,000 test set for evaluation (Hendrycks et al., 2021).

We used two dataset to evaluate the code generation task:

- **HumanEval** is a dataset consisting of 164 hand-written programming problems, designed to evaluate the functional correctness of code generation models. Each problem includes a function signature, docstring, function body, and unit tests. The dataset tests models' language comprehension, reasoning, and algorithmic abilities (Chen et al., 2021).
- **MBPP** is a dataset containing 974 programming problems designed to evaluate a model's ability to synthesize Python programs from natural language descriptions. The problems range from basic numerical operations to more complex tasks involving list and string processing. The test set consists of 500 problems, which are used for evaluation in this paper (Austin et al., 2021).

A.2 Hyperparameter Setting Details in Model Merging Methods

Table 3 presents the hyperparameter search ranges for the model merging methods. For Task Arithmetic and TIES-Merging, the scaling terms are selected from $[0.5, 1.0]$, while in TIES-Merging, the retain ratio for the largest-magnitude parameters is chosen from $[0.5, 0.7, 0.9]$. In contrast, the Average Merging method does not require any hyperparameters.

Merging Methods	Search Ranges of Hyperparameters
Task Arithmetic	Scaling term for merging model parameters: [0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
TIES-Merging	Scaling term for merging model parameters: [0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
	Ratio for retaining parameters with the largest-magnitude values: [0.5, 0.7, 0.9]

Table 3: Hyperparameter search ranges for model merging methods.

Table 4 presents the optimal hyperparameter settings for the TIES-Merging model merging method obtained through searching. These settings are further applied to model merging experiments involving M^3 and DARE.

Merging Method	Model	Hyperparameter Values
TIES-Merging	LM & Math	scaling_term=0.5, retain_ratio=0.5
	LM & Code	scaling_term=1.0, retain_ratio=0.7
	Math & Code	scaling_term=1.0, retain_ratio=0.5

Table 4: Hyperparameter settings in TIES-Merging.

A.3 Out-of-Distribution Dataset Selection Details

LiveBench (White et al., 2024) is a dynamic benchmark for large language models, featuring frequently updated questions and diverse tasks. To assess OOD robustness, we evaluate math & code, LM & math, and LM & code models using instruction following (LiveBench-Instruction), coding (LiveBench-Coding), and language comprehension (LiveBench-TypoFixing) category in LiveBench, respectively, deliberately avoiding the fine-tuning domains of the merged fine-tuned models. These tasks were released after November 2023, whereas WizardLM-13B, WizardMath-13B, and llama-2-13b-code-alpaca were all introduced earlier. Furthermore, their shared Llama-2-13b backbone was trained on data only up to July 2023. Consequently, these factors collectively ensure that the evaluation remains OOD in the temporal dimension.

When assessing the OOD robustness of LM & Code using the Language Comprehension category in LiveBench, only the typo-fixing task is considered. This decision is based on the fact that LiveBench is highly challenging, and the merged model performs poorly on other tasks in this category, with accuracy close to zero, rendering the evaluation results inconclusive and uninformative.

Finally, we acknowledge the limitations of these datasets. For large models like Llama-2-13b, identifying truly OOD datasets is difficult, as their training data likely covers similar distributions. These datasets are better described as "out-of-example", representing instances not explicitly seen during training. As discussed in (Wang et al., 2023), distribution shifts can occur across domains and time. While Llama-2-13b may have been trained on datasets for tasks like instruction-following, coding, and language comprehension, the datasets we selected remain valuable for OOD evaluation by capturing temporal shifts, providing insights into robustness over time.

A.4 Adversarial Robustness Evaluation Experiments Setting Details

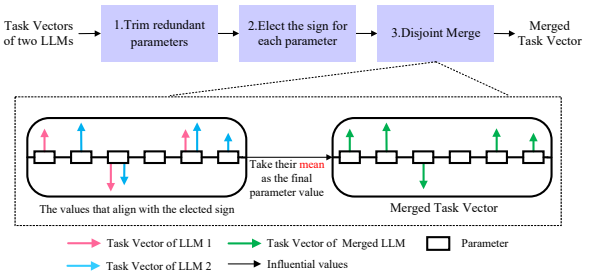
PromptBench (Zhu et al., 2024) is a unified library designed for evaluating LLMs, providing a standardized and extensible framework. It includes several key components such as prompt construction, prompt engineering, dataset and model loading, adversarial prompt attacks, dynamic evaluation protocols, and analysis tools.

We use the Adversarial Prompt Attacks module in PromptBench aims to evaluate the robustness of LLMs against adversarial prompts. We employ three methods to perform adversarial attacks on prompts to evaluate the adversarial robustness of the merged models: DeepWordBug (Gao et al., 2018), BERTAttack (Li et al., 2020), and StressTest (Naik et al., 2018), representing Character-level, Word-level, and Sentence-level attacks, respectively.

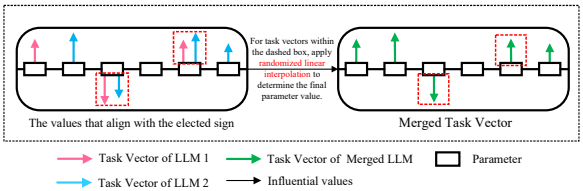
- **DeepWordBug** introduces subtle character-level perturbations (e.g., adding, deleting, or replacing characters) to words in text to deceive language models. It aims to evaluate a model's robustness against small typographical errors that may alter the model's performance without being easily detected.

- **BERTAttack** manipulates text at the word level by replacing words with contextually similar synonyms to mislead large language models. This method tests the model's ability to maintain accuracy despite small lexical changes that might alter the meaning of the input.

- **StressTest** appends irrelevant or redundant sentences to the end of a prompt to distract and confuse language models. It assesses the model's ability to handle extraneous information and maintain accuracy when faced with unnecessary distractions.



(a) The operational steps of TIES-Merging.



(b) After introducing M^3, the Disjoint Merge step in the TIES-Merging procedure.

Figure 5: The difference between M^3 and the original TIES-Merging is that, in the Disjoint Merge step, when two task vectors are retained for a given parameter, the mean of the task vectors is replaced by a random linear interpolation, while the other operations remain unchanged.

The evaluation is conducted on the Sentiment Analysis dataset (SST2 (Socher et al., 2013)) and the Grammar Correctness dataset (CoLA (Warstadt, 2019)):

- **SST2** (Socher et al., 2013): A sentiment analysis dataset designed to assess whether a given sentence conveys a positive or negative sentiment.

- **CoLA** (Warstadt, 2019): A dataset for grammar correctness, where the model must determine whether a sentence is grammatically acceptable.

B Additional Experimental Results on Adversarial Robustness

All the merged models are obtained using the Task Arithmetic method. Table 5 presents the detailed experimental results of the adversarial robustness of merged models on the SST2 and CoLA datasets applying the DeepWordBug prompt attack method. Table 6 presents the detailed experimental results of the adversarial robustness of merged models on the

Model	Dataset	Use Mixup	Use Attack	Metric (%)	PDR (%)
Math & Code	SST2	No	No	57.68	11.73
			Yes	50.92	
		Yes	No	78.21	37.10
			Yes	49.20	
	CoLA	No	No	72.87	56.97
			Yes	31.35	
		Yes	No	74.02	58.94
			Yes	30.39	
LM & Math	SST2	No	No	92.78	2.60
			Yes	90.37	
		Yes	No	91.28	3.77
			Yes	87.84	
	CoLA	No	No	79.19	4.96
			Yes	75.26	
		Yes	No	80.54	1.07
			Yes	79.67	
LM & Code	SST2	No	No	10.55	98.91
			Yes	0.11	
		Yes	No	73.17	97.65
			Yes	1.72	
	CoLA	No	No	74.21	8.79
			Yes	67.69	
		Yes	No	73.922	11.15
			Yes	65.68	

Table 5: Adversarial robustness of merged models on the SST2 and CoLA datasets when executing the DeepWordBug prompt attack method.

Model	Dataset	Use Mixup	Use Attack	Metric (%)	PDR (%)
Math & Code	SST2	No	No	57.68	13.92
			Yes	49.66	
		Yes	No	78.21	4.11
			Yes	75.00	
	CoLA	No	No	45.54	13.47
			Yes	39.41	
		Yes	No	71.72	17.25
			Yes	59.35	
LM & Math	SST2	No	No	92.78	2.22
			Yes	90.71	
		Yes	No	91.28	0.0
			Yes	91.28	
	CoLA	No	No	79.19	12.00
			Yes	69.70	
		Yes	No	80.54	5.83
			Yes	75.84	
LM & Code	SST2	No	No	10.55	95.65
			Yes	0.46	
		Yes	No	73.17	55.02
			Yes	32.91	
	CoLA	No	No	74.21	7.24
			Yes	68.84	
		Yes	No	73.92	7.52
			Yes	68.36	

Table 6: Adversarial robustness of merged models on the SST2 and CoLA datasets when executing the Bertattack prompt attack method.

SST2 and CoLA datasets applying the BERTAttack prompt attack method.

C Integrating M^3 into the TIES-Merging Model Merging Method

Figure 5 shows the specific implementation approach to incorporating M^3 into TIES-Merging. After the steps of trimming parameters with lower magnitudes and resolving sign disagreements, the

two models to be merged are denoted as M_1 and M_2. During the M^3 process, only the parameters that are preserved in both M_1 and M_2 are interpolated according to the model merging hyperparameter λ_m to obtain the merged parameters. For parameters that are preserved in only one of the models, no interpolation is performed, and the original value from the preserved model is retained in the merged model.

D Performance Drop Rate (PDR) for Adversarial Robustness

The adversarial robustness is evaluated using the Performance Drop Rate (PDR) (Zhu et al., 2023), which is defined as follows:

$$\text{PDR} = \frac{\text{Metric}_{\text{no attack}} - \text{Metric}_{\text{attack}}}{\text{Metric}_{\text{no attack}}}, \quad (9)$$

where $\text{Metric}_{\text{no attack}}$ denotes the performance metric without any prompt attack, and $\text{Metric}_{\text{attack}}$ represents the performance metric under the prompt attack. A smaller PDR indicates stronger adversarial defense against prompt attacks, implying better adversarial robustness.

E Detailed Introduction to DARE

DARE (Drop And REscale) (Yu et al., 2024) is a model sparsification method designed to reduce the redundancy of delta parameters in fine-tuned models while preserving their task-specific capabilities. In SFT, model parameters are optimized to unlock abilities for specific tasks, with the difference between fine-tuned and pre-trained parameters referred to as delta parameters.

However, studies have shown that delta parameters are often highly redundant. DARE addresses this redundancy by randomly dropping a proportion p of delta parameters (referred to as the drop rate) and rescaling the remaining ones by a factor of $1/(1-p)$. This simple yet effective approach enables DARE to eliminate up to 99% of delta parameters with minimal impact on model performance, particularly in large-scale models, and it can be applied using only CPUs.

Beyond sparsification, DARE serves as a versatile plug-in for merging multiple homologous fine-tuned models (fine-tuned from the same base model) by reducing parameter interference. When combined with existing model merging techniques such as Average Merging, Task Arithmetic, and TIES-Merging, DARE facilitates the fusion of models while retaining or even enhancing task performance across multiple benchmarks. This effect is especially pronounced in decoder-based LMs, where DARE boosts task generalization.

Experiments on AlpacaEval, GSM8K, and MBPP reveal that the merged LM has the potential to outperform any individual source LM, presenting a significant new discovery. Notably, the 7B model obtained through DARE merging, Super-Mario v2, ranks first among models of the same scale on the Open LLM Leaderboard (Beeching et al., 2023). These improvements were achieved without the need for retraining, positioning DARE as an efficient and resource-friendly solution for model merging.