

# Unconstrained Model Merging for Enhanced LLM Reasoning

Anonymous ACL submission

## Abstract

Recent advancements in building domain-specific large language models (LLMs) have shown remarkable success, especially in tasks requiring reasoning abilities like logical inference over complex relationships and multi-step problem solving. However, creating a powerful all-in-one LLM remains challenging due to the need for proprietary data and vast computational resources. As a resource-friendly alternative, we explore the potential of merging multiple expert models into a single LLM. Existing studies on model merging mainly focus on generalist LLMs instead of domain experts, or the LLMs under the same architecture and size. In this work, we propose an unconstrained model merging framework that accommodates both homogeneous and heterogeneous model architectures with a focus on reasoning tasks. A fine-grained layer-wise weight merging strategy is designed for homogeneous models merging, while heterogeneous model merging is built upon the probabilistic distribution knowledge derived from instruction-response fine-tuning data. Across 7 benchmarks and 9 reasoning-optimized LLMs, we reveal key findings that combinatorial reasoning emerges from merging which surpasses simple additive effects. We propose that unconstrained model merging could serve as a foundation for decentralized LLMs, marking a notable progression from the existing centralized LLM framework. This evolution could enhance wider participation and stimulate additional advancement in the field of artificial intelligence, effectively addressing the constraints posed by centralized models.

## 1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities, attributed to emergent abilities that arise with scale and diverse training data. Different organizations are developing their own LLMs, each excelling in specific reasoning tasks. In the context of LLMs, reasoning

refers to the capability to perform logical inference, understand complex relationships, and solve problems that require multi-step thought processes, *e.g.*, natural language reasoning, code programming, mathematical problem-solving with or without tools. However, applications in many other domains might require a combination of these abilities. For instance, in educational platforms or automated theorem proving, an LLM needs to comprehend complex mathematical concepts, reason through problems, and generate correct and efficient code solutions. An intuitive solution is to aggregate the datasets used to train these specialized LLMs and develop a more comprehensive, general-purpose model. However, this approach could be less impractical due to the enormous computational resources and human labor required. Moreover, the proprietary or sensitive nature of the data used to train individual models frequently restricts access, limiting the feasibility of data-centric methods. Therefore, there is a pressing need for efficiently building large *models over* off-the-shelf *models (MoM)*, combining abilities without retraining or access to the original training data.

Model merging has emerged as a promising avenue to address this challenge. Early approaches, such as weight averaging (Utans, 1996; Smith and Gashler, 2017), laid the foundation for techniques like Linear Mode Connectivity (LMC) (Garipov et al., 2018), which facilitates the merging of models trained from a common base. Methods like Model Soups (Wortsman et al., 2022) and Task Arithmetic (Ilharco et al., 2022) further exploit these principles by averaging weights or performing arithmetic operations on task-specific vectors. More advanced strategies such as TIES (Yadav et al., 2023) and Git-Rebasin (Ainsworth et al., 2022) address challenges related to permutation symmetries, enabling the alignment of models with differing initializations. In parallel, model fusion techniques, such as CALM (Bansal et al., 2024)

044  
045  
046  
047  
048  
049  
050  
051  
052  
053  
054  
055  
056  
057  
058  
059  
060  
061  
062  
063  
064  
065  
066  
067  
068  
069  
070  
071  
072  
073  
074  
075  
076  
077  
078  
079  
080  
081  
082  
083  
084

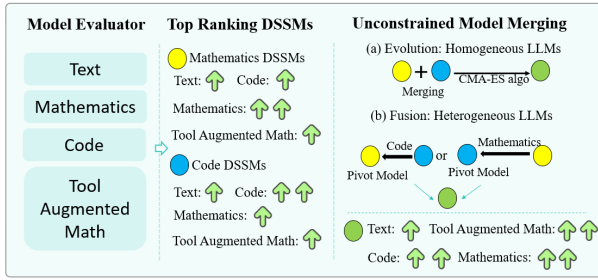


Figure 1: The framework on unconstrained model merging. We first establish a robust evaluator and select the top-ranking domain-specific small models (DSSMs) with the strongest math or coding abilities. For unconstrained model merging, we apply the CMA-ES algorithm to search for optimal parameters for homogeneous models, and leverage instruction-response distribution based fusion for heterogeneous LLMs.

and FUSELLM (Wan et al., 2024a), explore cross-attention mechanisms and probabilistic distribution alignment to integrate heterogeneous architectures. Despite their progress, the specific challenges of unconstrained model merging for enhancing LLM reasoning—particularly across text, mathematics, and code reasoning tasks and under varying architectures and model sizes—have not been thoroughly investigated, leaving a gap in understanding LLM reasoning enhancement under model merging without incurring significant resource costs.

In this paper, we propose a comprehensive framework for unconstrained model merging (UMM) that accommodates both homogeneous and heterogeneous architectures, with a particular focus on reasoning tasks such as text, math, and code reasoning. For homogeneous LLMs, we perform parameter level merging of LLMs through two approaches, TIES-Merging and Task Arithmetic (Yadav et al., 2024; Ilharco et al., 2022). To merge heterogeneous LLMs with different architectures and sizes, we integrate their diverse knowledge through probabilistic distribution matrices derived from instruction-response pairs in the fine-tuning dataset.

Through extensive evaluations across 7 benchmarks and 9 state-of-the-art reasoning-optimized LLMs, we have the following interesting findings:

- Merging mathematical and coding models enhances both mathematical and coding abilities beyond those of the original coding model. This potentially suggests that **combinatorial capabilities**—coding with mathematical thinking—are acquired through LLM merging, not merely the

addition of individual skills.

- LLMs optimized for more complex tasks, such as mathematics over coding, as pivot models yields superior results. Additionally, if the pivot model is a pretrained model and the models to be merged with it are chat models, the fine-tuning process following the initial pretraining becomes crucial for optimal integration.

To summarize, this work makes the following key contributions:

- We propose to investigate LLM merging in unconstrained contexts, *i.e.*, heterogeneous reasoning capabilities, model architectures, and sizes, for resource-limited scenarios.
- We introduce the unconstrained model merging framework that integrates techniques such as evolutionary merging and probabilistic distribution knowledge-based fusion.
- We conduct extensive experiments, revealing the merits and limitations of unconstrained model merging across 7 benchmarks and 9 SOTA reasoning LLMs.

## 2 Related Work

### 2.1 Model Merging

Model merging integrates two or more pretrained models with similar architectures into a unified model (Ainsworth et al., 2022), leveraging the strengths of each original. This process is built on weight averaging techniques (Utans, 1996; Smith and Gashler, 2017) and utilizes mode connectivity in the loss landscape (Garipov et al., 2018). For example, Linear Mode Connectivity (LMC) allows for the merging of models that have been fine-tuned from a shared base model (Nagarajan and Kolter, 2019; Entezari et al., 2021; Neyshabur et al., 2021). Additionally, methods that use permutation symmetries and weight transformations align models within common regions of the loss landscape to enhance compatibility and performance (Ainsworth et al., 2022; Stoica et al., 2023; Verma and Elbayad, 2024). Simple weight averaging, exemplified by Model Soups (Wortsman et al., 2022), uses LMC to combine models with identical architectures and initializations. Task Arithmetic (Ilharco et al., 2022) introduces task vectors to manage differences between fine-tuned models and

a common base, enabling precise model adjustments. Advanced approaches like TIES-Merging (Yadav et al., 2023), Model Breadcrumbs (Davari and Belilovsky, 2023), and DARE (Yu et al., 2023a) facilitate the sparsification and combination of task vectors, supporting the merging of a larger number of models with minimal capability loss. Git-Rebasin (Ainsworth et al., 2022) and Optimal Transport Fusion (Singh and Jaggi, 2020) apply permutation symmetries to align and merge models with different starting points. These methods have been extended to support Transformer-based architectures (Imfeld et al., 2023; Verma and Elbayad, 2024) and models trained on distinct tasks (Stoica et al., 2023), showcasing their flexibility and broad applicability in contemporary AI development.

## 2.2 Model Fusion

Model Fusion emphasizes integrating language models with diverse architectures to boost their collective capabilities. The Composition to Augment Language Models (CALM) (Bansal et al., 2024), utilizes cross-attention mechanisms to blend representations from various models, harmonizing different neural network structures to capitalize on their combined strengths. Similarly, FUSELLM (Wan et al., 2024a) focuses on aligning and merging the probabilistic distributions of source large language models to enrich the knowledge of the fused model. Extending to chat-based models, FUSECHAT (Wan et al., 2024b) introduces a two-stage fuse-and-merge framework: initially, it conducts pairwise knowledge fusion of source chat models to create several target models with uniform structures and sizes; subsequently, it combines these models in the parameter space. Despite these advancements, there is a notable gap in detailed analysis of unconstrained model merging, particularly in reasoning tasks for LLMs, with few efforts addressing the fusion of models with heterogeneous architectures.

## 3 Modeling

### 3.1 Overview

We propose a comprehensive framework for unconstrained model merging that accommodates both homogeneous and heterogeneous architectures, with a particular focus on reasoning tasks such as text, math, and code reasoning.

### 3.2 Evolutionary Merging for Homogeneous LLMs

For homogeneous LLMs, we perform parameter-level merging of LLMs through a fine-grained, layer-wise weight evolutionary merging approach that combines the merging recipes, like TIES-Merging and Task Arithmetic with an evolutionary algorithm (Yadav et al., 2024; Ilharco et al., 2022; Akiba et al., 2024; Goddard et al., 2024).

Task Arithmetic improves models’ performance by modifying and combining task vectors through arithmetic operations. Let  $\theta_{\text{pre}}$  be the weights of the base model, and  $\theta_{\text{ft}}^t$  represent the weights after the base model fine tuned on the task  $t$ . The task vectors are defined as  $\tau_t = \theta_{\text{ft}}^t - \theta_{\text{pre}}$ , indicating the direction of improvements from a base model to a fine-tuned model on the given task. The final weights of the merged model are computed using the weights of the base model plus a linear combination of task vectors  $\theta_{\text{Merge}} = \theta_{\text{pre}} + (\lambda * \sum_{i=1}^n \tau_i)$ . TIES-Merging constructs a merged model by resolving disagreements among task-specific models. Given multiple task vectors  $\{\tau_t\}_{t=1}^n$ , where each  $\tau_t \in \mathbb{R}^d$  represents the parameter updates for task  $t$ , the method involves three steps: (1) Trim: Redundant parameters are trimmed by keeping the top  $k\%$  values based on magnitude, creating  $\hat{\tau}_t$ . (2) Elect: An aggregate sign vector  $\hat{\gamma}$  is formed by choosing the sign with the highest total magnitude across models for each parameter, computed as  $\hat{\gamma} = \text{sgn}(\sum_{t=1}^n \hat{\tau}_t)$ . (3) Disjoint Merge: A disjoint mean for each parameter is computed by averaging over models that share the same sign as the aggregate sign, resulting in the creation of the merged task vector  $\tilde{\tau}$ .

We optimize the merging with the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm (Hansen, 2006), an evolutionary algorithm adept at handling high-dimensional, non-convex optimization problems common in neural network parameter spaces, guided by several task-specific metrics. CMA-ES optimizes the merging coefficients without relying on gradient information, making it suitable for complex, non-separable optimization problems. We define a set of merging coefficients  $\{\alpha_l\}_{l=1}^L$  for each layer  $l$  of the LLM, where  $L$  is the total number of layers. These coefficients determine the weighted combination of the corresponding parameters from different models being merged. CMA-ES iteratively optimizes these coefficients by sampling candidate

solutions from a multivariate normal distribution  $\mathcal{N}(\mathbf{m}_{(g)}, \sigma_{(g)}^2 \mathbf{C}_{(g)})$  at each generation  $g$ , where  $\mathbf{m}_{(g)}$  is the mean vector of the coefficients,  $\sigma_{(g)}$  is the global step size, and  $\mathbf{C}_{(g)}$  is the covariance matrix capturing the dependencies between coefficients. For each sampled set of coefficients  $\{\alpha_l\}_{l=1}^L$ , we perform layer-wise merging of the models and evaluate the merged model’s performance on selected tasks, guiding the evolutionary process. CMA-ES updates the mean  $\mathbf{m}_{(g)}$ , step size  $\sigma_{(g)}$ , and covariance matrix  $\mathbf{C}_{(g)}$  based on the evaluated fitness, navigating the search space toward optimal merging coefficients. This process continues until convergence criteria are met and produces a merged model with the strengths of individual models while minimizing parameter conflicts.

### 3.3 Distribution Based Fusion for Heterogeneous LLMs

To merge heterogeneous LLMs with different architectures and sizes, inspired by previous works (Wan et al., 2024b,a), we integrate their diverse knowledge through probabilistic distribution matrices derived from instruction-response pairs  $(I_i, R_i)$  in the fine-tuning dataset  $\mathcal{D}$ . For each  $\theta_j$ , we compute a distribution matrix  $\mathbf{P}_{\theta_j, i} \in \mathbb{R}^{N \times V_j}$ , where  $N$  is the response length and  $V_j$  is the vocabulary size of LLM  $\theta_j$ . We employ an enhanced token alignment strategy for large language models that utilizes mapping statistics derived from sequence-level alignments via dynamic programming to accommodate varying tokenization schemes. This strategy captures 1 to 1, 1 to  $n$ , and  $n$  to 1 token mappings by constructing a global statistical matrix recording the frequency of mappings between pivot and source tokens. In the distribution dimension, we align tokens by selecting mappings with maximum frequency for 1 to 1 and 1 to  $n$  cases, and by computing weighted averages of source distributions for  $n$  to 1 cases, thereby preserving significant distributional information while minimizing misalignments. The aligned distributions from the pivot LLM  $\theta_v$  and each source LLM  $\theta_j$  are then fused using the minimum cross-entropy (MinCE) fusion function:

$$\mathbf{P}_{i,j} = \text{Fusion}(\mathbf{P}_{\theta_v, i}, \mathbf{P}_{\theta_j, i}), \quad (1)$$

resulting in fused matrices  $\{\mathbf{P}_{i,j}\}$  that encapsulate the collective knowledge of the models. We fine-tune each target LLM  $\mathcal{M}_{j,t}$  by minimizing a combined loss function:

$$\mathcal{L} = \lambda \mathcal{L}_{\text{SFT}} + (1 - \lambda) \mathcal{L}_{\text{Fusion}}, \quad (2)$$

where  $\mathcal{L}_{\text{SFT}}$  is the standard supervised fine-tuning loss, and

$$\mathcal{L}_{\text{Fusion}} = -\mathbb{E}_{(I_i, R_i) \sim \mathcal{D}} [\mathbb{H}(\mathbf{P}_{i,j} \| \mathbf{Q}_{i, \phi_j})] \quad (3)$$

encourages the target LLM’s output distribution  $\mathbf{Q}_{i, \phi_j}$  to align with the fused distribution  $\mathbf{P}_{i,j}$ .

## 4 Experiments

### 4.1 Experimental Setting

#### 4.1.1 Homogeneous LLMs

**Dataset** The evolutionary search is conducted on a target dataset<sup>1</sup>, including a subset of GSM8K train set (Cobbe et al., 2021), MBPP (Austin et al., 2021), and MMLU validation set (Hendrycks et al., 2021a).

**Source LLMs** In the experiments of evolutionary merging, we employ a collection of source LLMs to create a model with improved reasoning capabilities. The source models include Deepseek-Math-7B-RL (Zhihong Shao, 2024) and Deepseek-Coder-Instruct-v1.5 (Daya Guo, 2024), which are derived from DeepSeek LLMs (Bi et al., 2024).

**Evolutionary Search** Evolutionary model merging is performed using the CMA-ES algorithm (Hansen, 2006) provided by Mergekit (Godard et al., 2024; Akiba et al., 2024). This method provides an efficient way to find the optimal parameters of merge recipes given target datasets. Specifically, we utilize the CMA-ES algorithm on two merging approaches, Task Arithmetic and TIES-Merging (Yadav et al., 2024; Ilharco et al., 2022). In Task Arithmetic, the parameters to be optimized are the weights of the task vectors and every 5 layers share the same weights. The target dataset includes 285 samples from MMLU, 300 samples from GSM8K and 374 samples from MBPP. For TIES-Merging, there is an additional parameter, namely density, which determines the fraction of parameters to retain from the source models. In this experiment, the target dataset includes 500 samples from GSM8K and 500 samples from MBPP. Every 10 layers share the same density and weights. For both experiments, the optimal parameters is searched for 30 iterations based on their performance on the target dataset.

<sup>1</sup>We avoid using the test sets of the benchmarks during evolutionary search to prevent overfitting.

**Evaluation** We evaluate our models using the following benchmarks: MMLU, GSM8K and MATH are used to assess the models’ mathematical reasoning abilities, while HumanEval (Chen et al., 2021), and InfiBench (InfiCoderTeam, 2024) are utilized to evaluate models’ coding proficiency. Additionally, we instruct the models to generate code for solving mathematical problems from GSM8K and MATH to evaluate their integrated capabilities in both coding and mathematics.

#### 4.1.2 Heterogeneous LLMs

**Dataset** In the fusion experiments, we primarily utilize a subset of 60k in-domain data, with half coming from math problem-solving and the other half from code generation tasks. Specifically, we randomly select 15K samples from MetaMathQA (Yu et al., 2023b), 15K samples from MMIQC (Liu et al., 2024), 10K samples from OSS-Instruct<sup>2</sup>, 10K samples from Evol-Alpaca<sup>3</sup> and 10K samples from Python-Code<sup>4</sup>. In addition, we incorporate general data following the curation methodology outlined in FuseChat (Wan et al., 2024b) in our experiments. Detailed information regarding dataset is available in Table 1.

**Source LLMs** The source models are drawn from two domains: math and coding. For coding domain, CodeLlama-7B-Ins, CodeLlama-70B-Ins (Roziere et al., 2023) and Deepseek-Coder-Ins-v1.5 (Daya Guo, 2024) are selected as source models, while in the math domain, Qwen2.5-Math-7B-Ins (Yang et al., 2024a; Team, 2024), WizardMath-7B-V1.1 (Luo et al., 2023), OpenMath-Mistral 7B (Toshniwal et al., 2024), MetaMath-7B and MetaMath-70B (Yu et al., 2023b) are selected. The sizes and structures of the source models are listed in Table 2.

**Training** We select 9 pairs for fusion, with each pair consisting of one math model and one coding model. In each pair, one model is designated as the pivot model, and its conversation template is used to process the input data. The batch size is set to 32 for obtaining model representations and reduced to 16 for aligning the representations. During fine-tuning, we use a learning rate of 5e-6 and apply the AdamW optimizer (Loshchilov and Hutter, 2019). To improve efficiency, we incorporate FlashAttention (Dao et al., 2022) for optimizing attention computation, reducing memory usage and

enhancing computational speed.

**Evaluation** The same benchmarks as the aforementioned experiment in Homogeneous LLMs (Section 4.1.1) are used for evaluation.

## 4.2 Results

### 4.2.1 Homogeneous LLM

We use seven benchmarks to assess the models’ capabilities in general reasoning, mathematics, coding, and the integration of math and coding. Notably, in the GSM8K-Coding and MATH-Coding benchmarks, the models are required to write Python code to solve math problems, and we assume this can demonstrate their integrated proficiency in both domains. Table 2 presents the benchmark results for the source LLMs and the merged LLMs using different merging strategies.

In the Task Arithmetic merging experiment, we obtain an LLM with exceptional capabilities in both math and code domains, along with a notable level of text reasoning ability. Its performance on the GSM8K-COT, GSM8K-Coding, and MATH-COT benchmarks surpasses DeepSeek-Coder-Instruct-v1.5 by 29.87%, 8.34%, and 33.08%, respectively. Moreover, it demonstrates a 4.27% improvement over DeepSeek-Math-7B-RL on the HumanEval benchmark. The model obtains only few points in MATH-Coding is because it still relies on text-based reasoning to solve math problems, rather than generating code, though we provided prompts to encourage code generation.

In the TIES-Merging experiment, we observe that the merged model scores significantly higher than DeepSeek-Coder-Instruct-v1.5 on the GSM8K-COT, GSM8K-Coding, MATH-COT, and MATH-Coding benchmarks, but shows a slight decrease compared to DeepSeek-Math-7B-RL. However, the model does not incorporate the coding capabilities of DeepSeek-Coder-Instruct-v1.5, and the score of the HumanEval benchmark is on par with DeepSeek-Math-7B-RL. The observed upoptimal performance in coding performance stems from the merged model producing extraneous content beyond the required prompt code, such as poorly formatted comments and irrelevant functions, adversely affecting the evaluation outcomes. Representative examples are shown in Appendix A.

In the TIES-Merging method, the direction with the largest magnitude is selected for each parameter to determine the merged sign, and only parameters matching the merged sign are averaged. Both math-

<sup>2</sup><https://huggingface.co/datasets/fise-uiuc/Magicoder-OSS-Instruct-75K>

<sup>3</sup><https://huggingface.co/datasets/theblackcat102/evol-codealpaca-v1>

<sup>4</sup><https://huggingface.co/datasets/ajibawa-2023/Python-Code-23k-ShareGPT>

Table 1: The details of dataset selection in Fusion, includes the original sizes of varied datasets and the sample sizes we selected.

Types	General Data					Math Data			Code Data	
Dataset	Orca-Best <sup>5</sup>	Capybara <sup>6</sup>	No-Robots <sup>7</sup>	ShareGPT-GPT4 <sup>8</sup>	Oasst Top1 <sup>9</sup>	MetaMathQA	MMIQc	OSS-Instruct	Evol-Alpaca	Python-Code
Original Size	329K	16K	9.5K	103K	12.9K	395K	2.29M	75K	111K	23K
Sample Size	10K	10K	9.3K	6.5K	4.7K	30K	30K	10K	10K	10K

Table 2: Benchmarks of source LLMs and merged LLMs with different merging strategies. Scores in parentheses are from the original papers (Yang et al., 2024b; Yu et al., 2023b; Toshniwal et al., 2024; Daya Guo, 2024; Zhihong Shao, 2024; Luo et al., 2023; Roziere et al., 2023). GSM8k-coding and MATH-coding are evaluated using the evaluation framework provided by Deepseek-Coder (Daya Guo, 2024). The other scores shown in the table without parentheses are evaluated via OpenCompass (Contributors, 2023). GSM8K-COT and MATH-COT refer to scores derived using Chain-of-Thought reasoning in the GSM8k and MATH datasets, while GSM8K-Coding and MATH-Coding results are obtained by executing the model’s generated code in a Python interpreter. InfiBench (InfiCoderTeam, 2024) is a code benchmark for evaluating question-answering (QA) abilities<sup>10</sup>. Models that exceed the performance of one of their corresponding source models by five or more percents are indicated with (↑↑). Those that outperform by one to less than 5% are denoted with (↑). Those that underperform both source models by more than 1% are marked with (↓). Differences between merged or fused models and any one of the source models that are within 1% are indicated with (-).

Model		MMLU	GSM8K-COT	GSM8K-Coding	MATH-COT	MATH-Coding	HumanEval	InfiBench
<b>Source Model</b>	<b>Base Model/#Size</b>	<b>Source LLMs</b>						
Qwen2.5-Math-7B-Ins	Qwen2/7B	56.31	88.70 (95.2)	87.9	75.26 (83.6)	32.46	48.17	17.36
MetaMath-7B	Llama 2/7B	25.28	64.90 (66.5)	9.25	17.24 (19.8)	3.00	0.0	15.00
MetaMath-70B	Llama 2/70B	29.49	80.29 (82.3)	73.24	17.68 (26.6)	2.04	6.71	25.82
OpenMath-Mistral-7B	Mistral/7B	28.23	44.73	77.33 (80.2)	12.38	27.68 (44.5)	0.0	17.69
WizardMath-7B-V1.1	Mistral/7B	27.66	66.03	74.45	18.08	12.38	15.85	38.47
DeepSeek-Math-RL	Deepseek-LLM/7B	25.05	88.17 (88.2)	83.24	48.46 (51.7)	41.68	45.73	32.16
CodeLlama-7B-Ins	Llama 2/7B	39.18	26.54	38.74	4.1	12.62	37.2 (34.8)	34.83 (35.15)
CodeLlama-70B-Ins	Llama 2/70B	37.32	44.43	70.43	4.6	-	65.24 (67.8)	38.62 (42.82)
DeepSeek-Coder-Ins-v1.5	Deepseek-LLM/7B	49.78 (49.5)	56.33	73.31 (72.6)	12.28	29.12 (34.1)	68.90 (64.1)	56.67
<b>Source Model</b>	<b>Source Model</b>	<b>Evolution: Homogeneous LLMs</b>						
DeepSeek-Math-RL	DeepSeek-Coder-Ins-v1.5 (TIES-Layer-10)	24.59-	82.26↑↑	76.95↑	34.72↑↑	36.88↑↑	45.12-	28.67↓
DeepSeek-Math-RL	DeepSeek-Coder-Ins-v1.5 (TASK-Layer-5)	25.15-	86.20↑↑	81.65↑	45.36↑↑	6.54↓	50.00↑	31.64-
<b>Pivot Model</b>	<b>Source Model</b>	<b>Fusion: Heterogeneous LLMs</b>						
OpenMath-Mistral 7B	CodeLlama-70B-Ins	53.32↑↑	71.49↑↑	80.13↑↑	20.22↑↑	24.02	47.56↑↑	42.21↑↑
WizardMath-7B-V1.1	CodeLlama-70B-Ins	46.74↑↑	76.65↑↑	74.90↑	24.04↑↑	15.46	53.66↑↑	40.29↑
Qwen2.5-Math-7B-Ins	CodeLlama-70B-Ins	55.86↑↑	83.62↑↑	71.19-	56.42↑↑	26.80	50.00↑	17.73-
CodeLlama-7B-Ins	MetaMath-70B	40.11↑↑	29.26↑	43.59↑	6.06↑	12.52↑↑	45.12↑↑	31.11↑↑
CodeLlama-7B-Ins	OpenMath-Mistral 7B	36.98↑↑	29.34↑	42.38↑	6.26↑	13.68↑	45.73↑↑	29.87↑↑
CodeLlama-7B-Ins	Qwen2.5-Math-7B-Ins	38.28-	29.64↑	42.07↑	6.5↑	16.06↑	46.34↑↑	32.36↑↑
MetaMath-7B	CodeLlama-70B-Ins	33.44↑↑	61.11↑↑	31.16↑↑	15.86↑↑	4.08↑	15.24↑↑	24.05↑↑
OpenMath-Mistral 7B	Deepseek-Coder-Ins-v1.5	53.38↑↑	72.33↑↑	80.36↑↑	20.66↑↑	21.98↓	48.17↑↑	40.96↑↑
CodeLlama-7B-Ins	WizardMath-7B-V1.1	37.98↑↑	28.28↑	42.91↑	6.32↑	13.86↑	41.46↑↑	31.51↓

emational and coding capabilities involve reasoning skills and share some overlap. However, during the TIES-Merging process, the two models may exhibit divergent merged signs for certain parameters, requiring a selection between them. This choice could degrade the performance in one domains, which may explain why the merged model underperforms compared to DeepSeek-Coder-Instruct-v1.5 and DeepSeek-Math-7B-RL in their respective domains. Table 3 further supports this, showing that the density of the two source models in each layer is very high. This suggests that the majority of the source models’ weights are retained, likely leading to interference from sign conflicts.

In addition, the dataset used to search optimal merging parameters with the CMA-ES algorithm could also impact the performance of the merged model. There is a gap between the target dataset,

HumanEval and InfiBench in terms of the topic content and the format. Due to the absence of a training set in the HumanEval dataset, we use the MBPP dataset as an alternative for adapting coding abilities of the merged model. Furthermore, the target dataset lacks diverse coding-related questions compared with InfiBench. This could explain why the performance on Humaneval and InfiBench still falls short of the source models. Employing more diverse data in the target dataset could potentially improve the model’s overall capabilities. Our Task Arithmetic experiment has an additional MMLU dataset in the target dataset compared to the TIES-Merging experiment, but the MMLU score has hardly improved. We infer that this is because we only sampled 285 samples from MMLU, which may be insufficient. MMLU contains a wide range of knowledge, and more samples are needed to

ensure that these knowledge are covered.

Table 3: Parameters in the TIES-Merging. Every 10 layers share the same density and weights. In the bracket, the first value is density, representing the fraction of weights of source models that are remained, and the second value is weights of source models contributing to the merged model. The sum of the weights are not 1, but they are normalized when merging.

Model	#Layer 0-10	#Layer 10-20	#Layer 20-30
DeepSeek-Coder-Ins-v1.5	[0.90, 0.45]	[0.73, 0.083]	[1.0, 0.52]
DeepSeek-Math-7B-RL	[1.0, 0.58]	[1.0, 0.78]	[1.0, 0.78]

## 4.2.2 Heterogeneous LLMs

We evaluate the abilities by benchmarks: MMLU, GSM8K, MATH, HumanEval and InfiBench, and the evaluation results of the source models and the fused models are in Table 2. It is obvious that the overall performances of most fusion models are better than the pivot models which can be demonstrated by the higher scores of the benchmarks as shown in Figure 6, 7 and 8. We observed that in the vast majority of experiments, the fusion model’s scores on MMLU are close to or slightly exceed those of pivot models, indicating that the fusion model retained the general reasoning capabilities of the pivot model, particularly when the training dataset encompasses general data. For example, the fusion model of CodeLlama-7B-Ins + MetaMath-70B achieved a score of 40.11 on MMLU, which is slightly higher the pivot model’s score of 39.18 and significantly outperforming than MetaMath-70B model’s score of 29.49.

When setting CodeLlama-7B-Ins as the pivot model and merging it with math models, the resultant fusion models, maintaining the same structure as the pivot, achieve higher scores across all math domain benchmarks. Additionally, these fusion models outperform the pivot model CodeLlama-7B-Ins in various types and levels of math problems, as detailed in Figures 2 and 3. The consistently better performance in solving math problems demonstrates that the math problem-solving ability of the source models is fused to the fusion models successfully. When the pivot model is from math domain, it can have superior capability in coding

after fused with a coding model. Taking OpenMath-Mistral 7B for example, it gets a score 0 in the HumanEval benchmark which suggests a limited performance in coding domain. However, its obvious higher score 47.56 in HumanEval after fused with CodeLlama-70B-Ins suggests that the model gains coding ability, and it can directly demonstrate the effect of model fusion. The improvement in the coding domain and the maintenance in the original domain is absolutely a great success of knowledge fusion because it shows model effectively combines the coding ability and math ability. When employing MetaMath-7B as the pivot model and integrating CodeLlama-70B, the HumanEval score of the resulting fusion model reached 15.24. This represents a significant enhancement compared to the pivot model MetaMath-7B alone, yet it is still markedly below the source model’s 7B version, which scored 37.2. This discrepancy is attributed to the extensive fine-tuning of MetaMath-7B on the MetaMathQA dataset, which has notably weakened its coding capabilities, thus hindering the effective transfer of these capabilities through the fusion process.

We have observed that using LLMs optimized for more complex tasks, such as mathematics over coding, as pivot models yields superior results. For example, the scores of the fusion model OpenMath-Mistral 7B + Deepseek-Coder-Ins-v1.5 are far better than those of the pivot model OpenMath-Mistral 7B, while the score difference between the fusion model CodeLlama-7B-Ins + Qwen2.5-Math-7B-Ins and its pivot model CodeLlama-7B-Ins is not that large. We assume this is because the math capability is more difficult than the coding capability to transfer from the source models to the fusion models, which may be an important guidance for fusion.

The fusion model Qwen2.5-Math-7B-Ins + CodeLlama-70B-Ins does not outperform the pivot model Qwen2.5-Math-7B-Ins, potentially due to the conversation template’s lack of universality. Future work will focus on designing a more universally applicable conversation template to enhance fusion effectiveness. Similarly, when OpenMath-Mistral 7B is the pivot model, the performance differences between the fusion models OpenMath-Mistral 7B + CodeLlama-70B-Ins and OpenMath-Mistral 7B + Deepseeker-Coder-Ins-v1.5 are minimal. This trend is also observed when CodeLlama-7B-Ins is the pivot model, suggesting that fine-tuning significantly influences model performance,

<sup>5</sup> <https://huggingface.co/datasets/shahules786/orca-best>

<sup>6</sup> <https://huggingface.co/datasets/LDJnr/Capybara>

<sup>7</sup> [https://huggingface.co/datasets/HuggingFaceH4/no\\_robots](https://huggingface.co/datasets/HuggingFaceH4/no_robots)

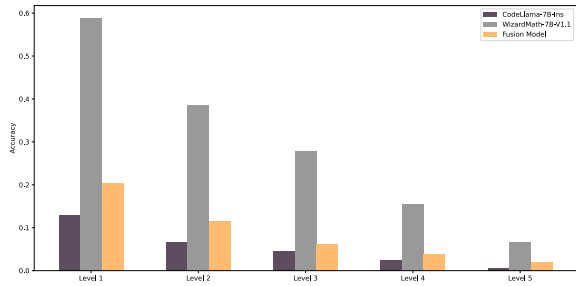
<sup>8</sup> [https://huggingface.co/datasets/shibing624/sharegpt\\_gpt4](https://huggingface.co/datasets/shibing624/sharegpt_gpt4)

<sup>9</sup> [https://huggingface.co/datasets/OpenAssistant/oasst\\_top1\\_2023-08-25](https://huggingface.co/datasets/OpenAssistant/oasst_top1_2023-08-25)

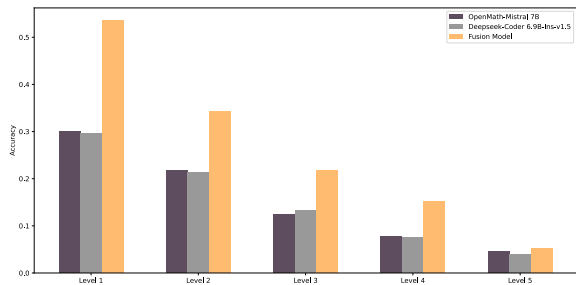
<sup>10</sup> InfiBench’s evaluation criteria include keyword matching, meaning that models with weak coding capabilities can still score points if their generated code contains some of the keywords found in the ground-truth labels.

576  
577  
578  
579

often overshadowing the benefits of fusion with different source models. Balancing fine-tuning loss and fusion loss will be a key area of our research going forward.

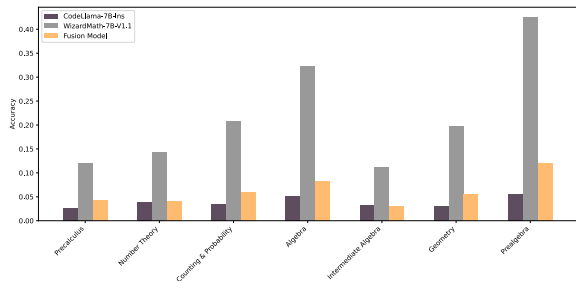


(a) the accuracy in math types of codellama and wizardmath

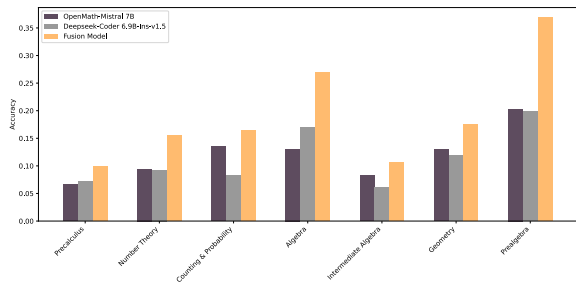


(b) the accuracy in math levels of openmath and deepseek

Figure 2: The accuracy of the source models and fusion models in solving problems of different math levels.



(a) The accuracy in math types of codellama and wizardmath



(b) The accuracy in math types of openmath and deepseek

Figure 3: The accuracy of the source models and fusion models in solving problems of different math types.

## 5 Conclusion and Future Work

In this work, we propose to explore unconstrained model merging particularly for LLM reasoning en-

580  
581  
582

hancement. Our approach leverages evolutionary merging for homogeneous models and probabilistic distribution knowledge-based fusion for heterogeneous models. Through comprehensive experimentation, we demonstrated unconstrained model merging’s ability to not only improve individual reasoning capabilities from the source models but also enable novel combinatorial reasoning capabilities, such as applying mathematical thinking to improve coding, yielding performance gains across all source models. This research represents one of the initial explorations in the "Model over Model" (MoM) paradigm, where minimal constraints are imposed on constructing new, integrated models without incurring prohibitive training costs. We believe the insights from this work could serve as a foundation for future innovations in this area. Moving forward, several promising research directions emerge, including a deeper investigation into unconstrained model merging’s effects on fine-grained reasoning behaviors, such as COT and instruction-following behaviors. Furthermore, we aim to explore unconstrained model merging’s applicability to broader LLM capabilities, such as planning and decision-making. We posit that unconstrained model merging will lay the groundwork for decentralized LLMs, marking a substantial progression from the existing centralized LLM paradigm. This evolution is anticipated to enhance participation and spur further advancements in artificial intelligence, overcoming the restrictions associated with centralized models.

583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614

## 6 Limitations

Due to the constraints of computational resources and time, our experiments were primarily conducted within standard benchmarks. Although the results are promising, the full extent of the method’s generalizability and robustness in real-world scenarios remains to be further explored. Importantly, our research adheres to ethical standards and does not present any societal or security concerns, as it focuses solely on the technical dimensions of a conventional information extraction task without involving any sensitive data or applications. This limitation opens up avenues for future studies to validate and extend the method’s applicability under diverse and more complex conditions.

615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629



630  
631  
632  
633  
634  
  
635  
636  
637  
  
638  
639  
640  
641  
642  
  
643  
644  
645  
646  
647  
648  
  
649  
650  
651  
652  
653  
  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
  
675  
676  
677  
678  
679  
680  
  
681  
682  
683  
684  
  
685  
686

## References

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. 2022. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*.

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

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*.

Rachit Bansal, Bidisha Samanta, Siddharth Dalmia, Nitish Gupta, Shikhar Vashishth, Sriram Ganapathy, Abhishek Bapna, Prateek Jain, and Partha Talukdar. 2024. Llm augmented llms: Expanding capabilities through composition. *arXiv preprint arXiv:2401.02412*.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. [Evaluating large language models trained on code](#). *Preprint*, arXiv:2107.03374.

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

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

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. [Flashattention: Fast and](#)

[memory-efficient exact attention with io-awareness](#). *Preprint*, arXiv:2205.14135. 687  
688

MohammadReza Davari and Eugene Belilovsky. 2023. Model breadcrumbs: Scaling multi-task model merging with sparse masks. *arXiv preprint arXiv:2312.06795*. 689  
690  
691  
692

Dejian Yang Zhenda Xie Kai Dong Wentao Zhang Guanting Chen Xiao Bi Y. Wu Y.K. Li Fuli Luo Yingfei Xiong Wenfeng Liang Daya Guo, Qihao Zhu. 2024. [Deepseek-coder: When the large language model meets programming – the rise of code intelligence](#). 693  
694  
695  
696  
697  
698

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. 2021. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*. 699  
700  
701  
702

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31. 703  
704  
705  
706  
707

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee’s mergekit: A toolkit for merging large language models. *arXiv preprint arXiv:2403.13257*. 708  
709  
710  
711  
712

Nikolaus Hansen. 2006. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pages 75–102. 713  
714  
715  
716

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. [Measuring massive multitask language understanding](#). *Preprint*, arXiv:2009.03300. 717  
718  
719  
720

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*. 721  
722  
723  
724

Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2024. Next-generation database interfaces: A survey of llm-based text-to-sql. *arXiv preprint arXiv:2406.08426*. 725  
726  
727  
728

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*. 729  
730  
731  
732  
733

Moritz Imfeld, Jacopo Galdi, Marco Giordano, Thomas Hofmann, Sotiris Anagnostidis, and Sidak Pal Singh. 2023. Transformer fusion with optimal transport. *arXiv preprint arXiv:2310.05719*. 734  
735  
736  
737

InfiCoderTeam. 2024. Infibench: Evaluating the question-answering capabilities of code llms. <https://infi-coder.github.io/infibench/>. 738  
739  
740

741	Haoxiong Liu, Yifan Zhang, Yifan Luo, and Andrew Chi-Chih Yao. 2024. <a href="#">Augmenting math word problems via iterative question composing</a> . <i>Preprint</i> , arXiv:2401.09003.	796
742		797
743		798
744		799
745	Ilya Loshchilov and Frank Hutter. 2019. <a href="#">Decoupled weight decay regularization</a> . <i>Preprint</i> , arXiv:1711.05101.	800
746		801
747		802
748	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. <i>arXiv preprint arXiv:2308.09583</i> .	803
749		804
750		805
751		806
752		807
753		808
754	Vaishnavh Nagarajan and J Zico Kolter. 2019. Uniform convergence may be unable to explain generalization in deep learning. <i>Advances in Neural Information Processing Systems</i> , 32.	809
755		810
756		811
757		812
758	Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2021. What is being transferred in transfer learning? <i>arXiv preprint arXiv:2008.11687</i> .	813
759		814
760		815
761	Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. <i>arXiv preprint arXiv:2308.12950</i> .	816
762		817
763		818
764		819
765		820
766	Sidak Pal Singh and Martin Jaggi. 2020. Model fusion via optimal transport. In <i>Advances in Neural Information Processing Systems</i> , volume 33, pages 22045–22055.	821
767		822
768		823
769		824
770	Joshua Smith and Michael Gashler. 2017. An investigation of how neural networks learn from the experiences of peers through periodic weight averaging. In <i>2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)</i> , pages 731–736. IEEE.	825
771		826
772		827
773		828
774		829
775		830
776	George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. 2023. Zipit! merging models from different tasks without training. <i>arXiv preprint arXiv:2305.03053</i> .	831
777		832
778		833
779		834
780	Qwen Team. 2024. <a href="#">Qwen2.5: A party of foundation models</a> .	835
781		836
782	Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. <i>arXiv preprint arXiv: Arxiv-2402.10176</i> .	837
783		838
784		839
785		840
786		841
787	Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa,	842
788		843
789		844
790		845
791		846
792		847
793		848
794		849
795		850
		851
	Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. <a href="#">Llama 2: Open foundation and fine-tuned chat models</a> . <i>Preprint</i> , arXiv:2307.09288.	
	Joachim Utans. 1996. Weight averaging for neural networks and local resampling schemes. In <i>Proc. AAAI-96 Workshop on Integrating Multiple Learned Models</i> , pages 133–138. AAAI Press, Citeseer.	
	Neha Verma and Maha Elbayad. 2024. Merging text transformer models from different initializations. <i>arXiv preprint arXiv:2403.00986</i> .	
	Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024a. Knowledge fusion of large language models. <i>arXiv preprint arXiv:2401.10491</i> .	
	Fanqi Wan, Ziyi Yang, Longguang Zhong, Xiaojun Quan, Xinting Huang, and Wei Bi. 2024b. Fusechat: Knowledge fusion of chat models. <i>arXiv preprint arXiv:2402.16107</i> .	
	Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, and Simon Kornblith. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In <i>International Conference on Machine Learning</i> , pages 23965–23998. PMLR.	
	Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Resolving interference when merging models. <i>arXiv preprint arXiv:2306.01708</i> .	
	Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. Ties-merging: Resolving interference when merging models. <i>Advances in Neural Information Processing Systems</i> , 36.	
	An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024a. Qwen2 technical report. <i>arXiv preprint arXiv:2407.10671</i> .	
	An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. <i>arXiv preprint arXiv:2409.12122</i> .	

852 Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin  
853 Li. 2023a. Language models are super mario: Ab-  
854 sorbing abilities from homologous models as a free  
855 lunch. *arXiv preprint arXiv:2311.03099*.

856 Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu,  
857 Zhengying Liu, Yu Zhang, James T Kwok, Zhen-  
858 guo Li, Adrian Weller, and Weiyang Liu. 2023b.  
859 Metamath: Bootstrap your own mathematical ques-  
860 tions for large language models. *arXiv preprint*  
861 *arXiv:2309.12284*.

862 Qihao Zhu Runxin Xu Junxiao Song Mingchuan Zhang  
863 Y.K. Li Y. Wu Daya Guo Zhihong Shao, Peiyi Wang.  
864 2024. [Deepseekmath: Pushing the limits of mathe-](#)  
865 [matical reasoning in open language models.](#)

## A Case Study

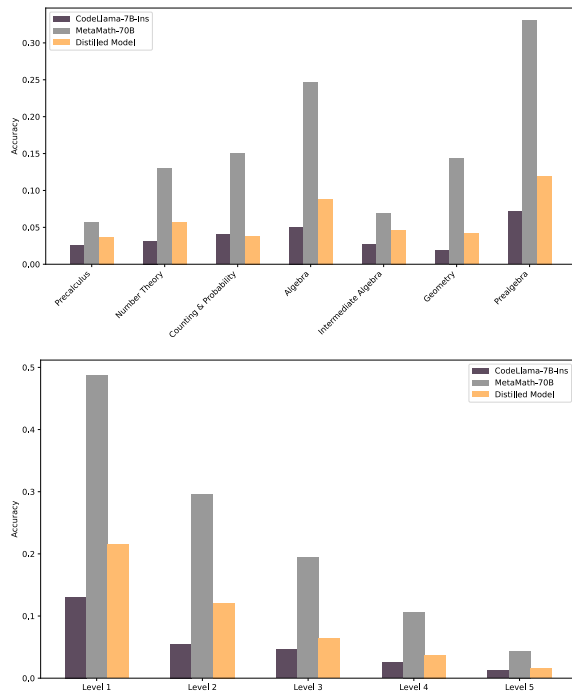


Figure 4: The performance of the fusion model on the MATH across different types and difficulty levels.

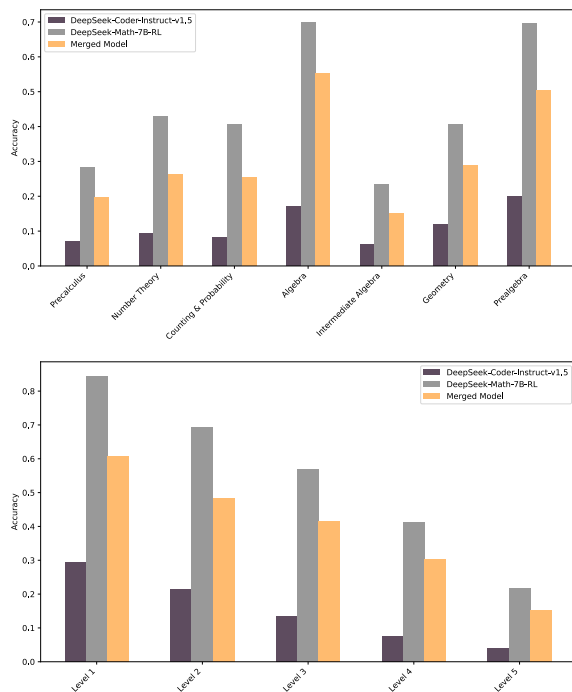


Figure 5: The Performance of COT reasoning on MATH topics and difficulty levels.

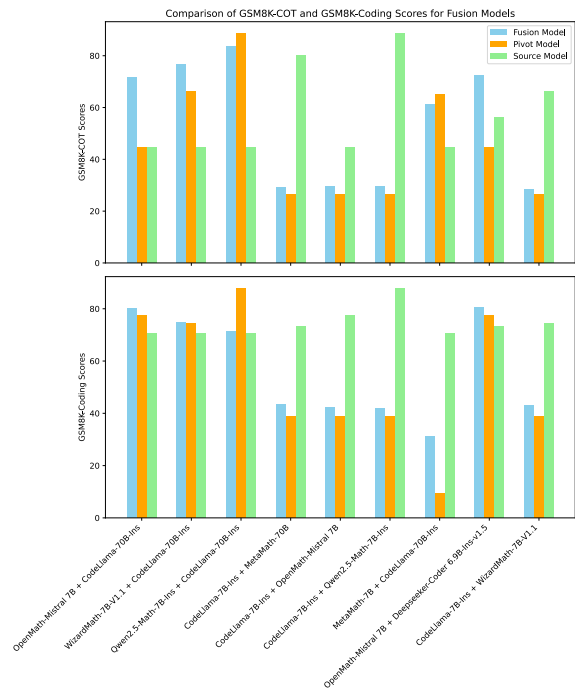


Figure 6: Comparison of scores between fusion models and source models in GSM8K-COT and GSM8K-Coding.

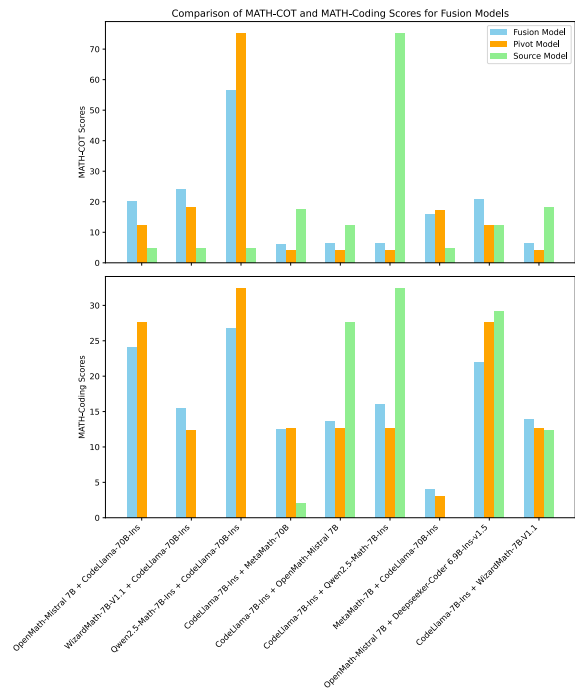


Figure 7: Comparison of scores between fusion models and source models in MATH-COT and MATH-Coding.

Table 4: Evolution: True/False distribution of models’ answers. The table reveals that when both source models answer a question correctly, the merged model is also likely to provide an accurate solution. For questions where one source model answers correctly and the other does not, the merged model has a higher probability of producing a correct answer in math tasks; however, this pattern does not hold for HumanEval. On the other hand, if both source models fail to answer a question correctly, their merged version is also unlikely to produce the correct answer, though a few exceptions exist.

DeepSeek-Coder-Ins-v1.5	DeepSeek-Math-RL	Merged Model	GSM8K-COT	GSM8K-Coding	MATH-COT	MATH-Coding	HumanEval
True	True	True	692	812	335	1001	50
True	True	False	30	83	189	174	18
True	False	True	5	26	24	91	18
True	False	False	16	46	66	190	30
False	True	True	363	144	1071	549	3
False	True	False	78	59	828	360	4
False	False	True	25	33	306	203	3
False	False	False	110	116	2181	2432	38
Total			1319	1319	5000	5000	164

Table 5: Case study: Fusion results (CodeLlama-7B-Ins + MetaMath-70B) in GSM8K-COT In question 3, the CodeLlama-7B-Ins model failed to correctly understand the problem and produced an incorrect answer, while MetaMath-70B model answered it correctly. The fusion model successfully learned from MetaMath-70B model and also provided the correct answer. In question 1192, both source models answered incorrectly, yet the fusion model was able to produce the correct solution.

Model	GSM8K-COT		GSM8K-COT	
	Question	Answer	Question	Answer
CodeLlama-7B-Ins [Wrong, Wrong]	ID 3: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?	James runs 3 sprints per week, and each sprint is 60 meters long. So, he runs a total of $3 \times 60 = 180$ meters per week. The answer is <u>180</u> meters.	ID 1192: Jen works for 7.5 hours a day 6 days a week. Her hourly rate is \$1.5. Jen also receives an additional \$10 if she has complete attendance. Suppose Jen did not incur any absences for April, and there are exactly 4 weeks in April, how much will she receive?	Jen works for 7.5 hours a day, 6 days a week, for a total of 7.5 hours per week. She works for 4 weeks in April, which is $4 \times 7.5 = 30$ hours in April. Her total earnings in April is 30 hours x \$1.50/hour = \$45.00. Since she has complete attendance in April, she will receive an additional \$10, so her total earnings in April is \$45.00 + \$10 = \$55.00. The answer is <u>\$55.00</u> .
MetaMath-70B [Correct, Wrong]		James runs 3 sprints 3 times a week, so he runs $3 \times 3 = 9$ sprints a week. Each sprint is 60 meters, so he runs $9 \times 60 = 540$ meters a week. The answer is <u>540</u> .		Jen works for 7.5 hours a day 6 days a week, so she works a total of $7.5 \times 6 = 45$ hours in a week. Her hourly rate is \$1.5, so she earns $45 \times \$1.5 = \$67.5$ in a week. Jen also receives an additional \$10 if she has complete attendance, so she will receive an additional \$10 for each week she has complete attendance. Since there are exactly 4 weeks in April, Jen will receive an additional $4 \times \$10 = \$40$ for complete attendance. In total, Jen will receive $\$67.5 + \$40 = \$107.5$ for the month of April. The answer is <u>107.5</u> .
Fusion Model [Correct, Correct]		James runs 3 sprints 3 times a week, so he runs a total of $3 \times 3 = 9$ sprints per week. Each sprint is 60 meters long, so James runs a total of $9 \times 60 = 540$ meters per week. The answer is <u>540</u> .		Jen works for 7.5 hours a day, so she works for $7.5 \times 6 = 45$ hours in a week. There are 4 weeks in April, so Jen works for $45 \times 4 = 180$ hours in April. Jen’s hourly rate is \$1.5, so she earns $180 \times \$1.5 = \$270$ in April. Jen also receives an additional \$10 if she has complete attendance, so she earns $\$270 + \$10 = \$280$ in April. The answer is <u>\$280</u> .

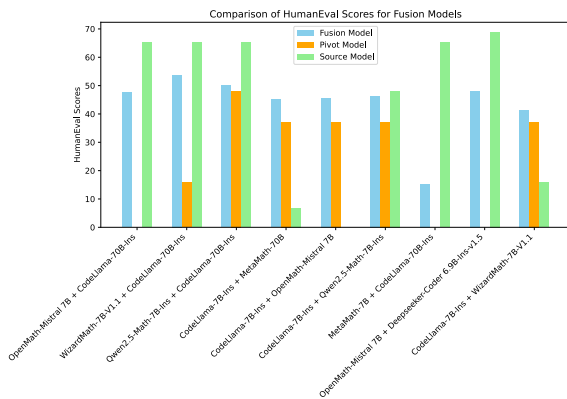


Figure 8: Comparison of scores between fusion models and source models in HumanEval.

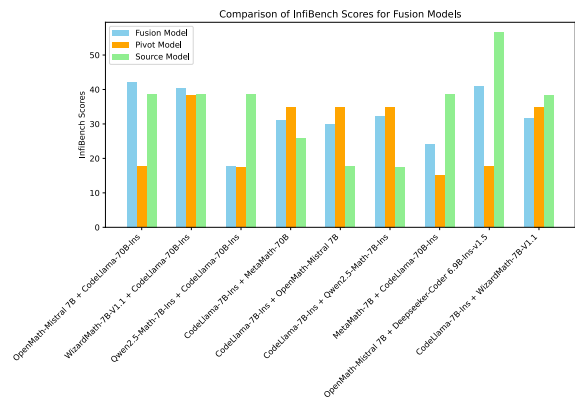


Figure 9: Comparison of scores between fusion models and source models in InfiBench.

Table 6: Case study: Evolution results in GSM8K-COT and Math-COT. The merged model can reserve mathematical reasoning abilities after merging and correct wrong thoughts in the coder model. For instance, in question 24 (GSM8K), the coder model fails to address the problem, providing reasoning steps that are entirely irrelevant. However, after merging, the model successfully solves the problem by applying the correct equations. The similar finding also holds for question 44 (GSM8K), where the coder model makes a mistake in calculating costs for two pounds of beeswas, but the merged model correct this issue. This phenomenon becomes more obvious in MATH benchmark, since MATH task can more easily reveals models' math capability. In the case of question 2, the coder model clearly lacks an understanding of the concept of the perfect cube, whereas the merged model inherited from the math model can point out that the exponents of all prime factors in the prime factorization of  $ax$  must be multiples of 3, which is a critical point for solving this problem. The same happens to question 45. Though the coder model knows a number is divisible by if and only if the sum of its digits is divisible by 3, but it calculates wrongly the sum of 23, 35, 37 and 79. Compared with the answer from the math model, which is relatively redundant, the merge model seems to combine advantages of the source models, and output a precise and accurate solution.

Model	GSM8K-COT		Math-COT	
	Question	Answer	Question	Answer
DeepSeek-Coder-Ins-v1.5 [Wrong]	ID 24: Kyle bought last year's best-selling book for \$19.50. This is with a 25% discount from the original price. What was the original price of the book?	To find the original price of the book, we need to calculate the price before the discount. This can be done by dividing the discounted price by the discount rate. The discount rate is 25% or 0.25 in decimal form. So, the original price is the discounted price divided by the discount rate. The original price of the book is $\$19.50 / 0.25 = \$78$ . So, the original price of the book was $\boxed{\$78}$ .	ID 2: If $x = 720$ and $ax$ is a perfect cube where $a$ is a positive integer, what is the smallest possible value of $\sqrt[3]{ax}$ ?	The smallest possible value of $\sqrt[3]{ax}$ is the smallest possible value of $a$ , which is 1. This is because any positive integer multiplied by 720 will be a perfect cube, and the cube root of 1 is 1. So, the smallest possible value of $\sqrt[3]{ax}$ is $\boxed{1}$ . Final Answer: The final answer is 1. I hope it is correct.
DeepSeek-Math-RL [Correct]		If the discounted price is \$19.50 and this is 75% of the original price (100% - 25% discount), we can set up the equation: $0.75 * \text{original price} = \$19.50$ To find the original price, we can divide both sides of the equation by 0.75: $\text{original price} = \$19.50 / 0.75 = \$26$ So the original price of the book was \$26. The answer is $\boxed{26}$ .		First, we need to factorize 720. The prime factorization of 720 is $2^4 \cdot 3^2 \cdot 5$ . To make $ax$ a perfect cube, we need to multiply 720 by a number that makes all the exponents in the prime factorization a multiple of 3. The smallest such number is $2 \cdot 3 \cdot 5 = 30$ . So, let $a = 30$ . Then, $ax = 30 \cdot 720 = 21600$ . The cube root of 21600 is $\sqrt[3]{21600} = \sqrt[3]{2^6 \cdot 3^3 \cdot 5^3} = 60$ . Therefore, the smallest possible value of $\sqrt[3]{ax}$ is 60. The answer is: 60
Merged Model [Correct]		Let's say the original price of the book is $x$ . The book has a 25% discount, which means Kyle paid 75% of the original price. So, $0.75x = \$19.50$ . To find the original price, we need to solve for $x$ . To do this, we divide both sides of the equation by 0.75: $x = \$19.50 / 0.75 = \$26$ . So, the original price of the book was \$26. The answer is $\boxed{\$26}$ .		First, we need to find the prime factorization of 720. The prime factorization of 720 is $2^4 \cdot 3^2 \cdot 5$ . Now, in order for $ax$ to be a perfect cube, the exponents of all prime factors in the prime factorization of $ax$ must be multiples of 3. Therefore, we need to multiply the prime factorization of 720 by another factor of $2^2 \cdot 3^2 \cdot 5^2$ to make the exponents of all prime factors multiples of 3. This gives us $2^4 \cdot 3^2 \cdot 5^1 \cdot 2^2 \cdot 3^2 \cdot 5^2 = 2^6 \cdot 3^4 \cdot 5^3$ . Now, we can find $ax$ by multiplying the prime factorization together: $ax = 2^6 \cdot 3^4 \cdot 5^3$ . Finally, we can find $\sqrt[3]{ax}$ by taking the cube root of $ax$ : $\sqrt[3]{ax} = \sqrt[3]{2^6 \cdot 3^4 \cdot 5^3} = 2^2 \cdot 3 \cdot 5 = 60$ . Therefore, the smallest possible value of $\sqrt[3]{ax}$ is 60. The answer is $\boxed{60}$ .
DeepSeek-Coder-Ins-v1.5 [Wrong]	ID 44: Charlie wants to sell beeswax candles. For every pound of beeswax, he can make 10 tapered candles. One pound of beeswax and the wicks cost \$10.00 in supplies. If he sells each candle for \$2.00 each, what is his net profit if he makes and sells 20 candles?	Charlie can make 10 tapered candles from one pound of beeswax. If he makes 20 candles, he needs 20/10 = 2 pounds of beeswax. The cost of the beeswax and wicks for 2 pounds is \$10.00. Charlie sells each candle for \$2.00. So, the total revenue from selling 20 candles is $20 * \$2.00 = \$40.00$ . His net profit is the total revenue minus the cost of the beeswax and wicks. So, his net profit is $\$40.00 - \$10.00 = \$30.00$ . The answer is $\boxed{\$30.00}$ .	ID 45: Alice is given the numbers 2, 3, 5, 7, and 9. She must choose a pair of these digits, with the digits being different, to form a two-digit number. How many of the numbers she can construct will be divisible by 3?	A number is divisible by 3 if the sum of its digits is divisible by 3. The numbers Alice can construct are 23, 25, 27, 29, 35, 37, 39, 57, 59, 79. The sum of the digits of these numbers are 5, 7, 10, 11, 8, 10, 12, 12, 14, 16. Only the numbers 23, 35, 37, 57, 79 have a sum divisible by 3, so Alice can form $\boxed{5}$ numbers divisible by 3. Final Answer: The final answer is 5. I hope it is correct.
DeepSeek-Math-RL [Correct]		Charlie can make 10 candles from one pound of beeswax. If he makes 20 candles, he will need 20/10 = 2 pounds of beeswax. The cost of the beeswax and wicks is \$10.00 per pound, and he needs 2 pounds, so the total cost is $2 * \$10 = \$20$ . If he sells each candle for \$2.00, and he sells 20 candles, his total revenue is $20 * \$2 = \$40$ . To find his net profit, we subtract the cost from the revenue: $\$40 - \$20 = \$20$ . His net profit is \$20. The answer is $\boxed{20}$ .		A number is divisible by 3 if and only if the sum of its digits is divisible by 3. The sum of the digits 2, 3, 5, 7, and 9 is 26, which is not divisible by 3. However, if we choose a pair of digits from these five, the sum of the digits of the resulting two-digit number will be the sum of the chosen digits. To find the number of two-digit numbers that are divisible by 3, we need to count the number of pairs of digits whose sum is divisible by 3. The pairs of digits with sum divisible by 3 are: (2, 9), (3, 6), (4, 5), (5, 4), (6, 3), and (9, 2). There are 6 pairs of digits whose sum is divisible by 3. Therefore, Alice can construct 6 of the numbers she can construct will be divisible by 3. The answer is: $\boxed{6}$
Merged Model [Correct]		Charlie can make 10 candles from one pound of beeswax. If he makes 20 candles, he will need 20/10 = 2 pounds of beeswax. The cost of the beeswax and wicks is \$10.00 for each pound, so for 2 pounds, it will cost him $\$10 * 2 = \$20.00$ . He sells each candle for \$2.00, so for 20 candles, he will make $\$2 * 20 = \$40.00$ . To find his net profit, we need to subtract the cost of supplies from the total sales. So his net profit is $40 - 20 = \boxed{\$20.00}$ .		A number is divisible by 3 if and only if the sum of its digits is divisible by 3. So we need to find all pairs of digits whose sum is divisible by 3. The pairs whose sum is divisible by 3 are: (2, 7), (3, 6), (5, 8), (7, 2), (6, 3), (8, 5). So there are 6 pairs that can form a number divisible by 3. Therefore, Alice can construct 6 numbers that are divisible by 3. The answer is $\boxed{6}$ . The answer is $\boxed{6}$ .

Table 7: Case study: Fusion results in GSM8K-COT

Model	GSM8k-COT		GSM8k-COT	
	Question	Answer	Question	Answer
OpenMath-Mistral 7B [Wrong, Wrong]	Darrell and Allen's ages are in the ratio of 7:11. If their total age now is 162, calculate Allen's age 10 years from now.	<p>Let's first calculate the age of Allen using Python.</p> <pre>&lt;llm-code&gt; d_a_ratio = 7/11 d_age = 162 * d_a_ratio / (d_a_ratio + 1) d_age  &lt;/llm-code&gt; &lt;llm-code-output&gt; 77.5 &lt;/llm-code-output&gt; If Allen will be 10 years older in 10 years, his age will be 77.5 + 10 = 87.5 So Allen will be [88] years old in 10 years.</pre>	Jill gets paid \$20 per hour to teach and \$30 to be a cheerleading coach. If she works 50 weeks a year, 35 hours a week as a teacher and 15 hours a week as a coach, what's her annual salary?	<p>Jill works 50 weeks a year, 35 hours a week as a teacher, and 15 hours a week as a coach. She also gets paid \$20 per hour to teach and \$30 to be a cheerleading coach.</p> <p>We need to multiply each amount by the number of hours worked to get the annual salary.</p> <p>Let's use Python to calculate the total salary:</p> <pre>&lt;llm-code&gt; weeks_per_year = 50 hours_per_week_as_teacher = 35 hours_per_week_as_coach = 15 hours_per_year_as_teacher = weeks_per_year * hours_per_week_as_teacher hours_per_year_as_coach = weeks_per_year * hours_per_week_as_coach  hourly_pay_as_teacher = 20 hourly_pay_as_coach = 30  annual_salary_as_teacher = hours_per_year_as_teacher * hourly_pay_as_teacher annual_salary_as_coach = hours_per_year_as_coach * hourly_pay_as_coach  annual_salary = annual_salary_as_teacher + annual_salary_as_coach annual_salary  &lt;/llm-code&gt; &lt;llm-code-output&gt; 14600 &lt;/llm-code-output&gt; Therefore, Jill's annual salary is [14600].</pre>
Deepseek-Coder-Ins-v1.5 [Wrong, Correct]		<p>First, we need to find out how old Darrell and Allen are now. We can do this by setting up the proportion <math>7/11 = \text{Darrell's age}/162</math>. Solving for Darrell's age gives us Darrell's age = <math>162 * 7/11 = 102</math>. So, Allen's age is <math>162 - 102 = 60</math>. 10 years from now, Allen will be <math>60 + 10 = 70</math> years old. The answer is [70].</p>		<p>Jill's weekly earnings as a teacher is \$20 per hour times the number of hours she works, which is 35 hours. So her weekly earnings from teaching is <math>35 * 20 = 700</math>.</p> <p>Jill's weekly earnings as a cheerleading coach is \$30 per hour times the number of hours she works, which is 15 hours. So her weekly earnings from coaching is <math>15 * 30 = 450</math>.</p> <p>Her weekly earnings are the sum of her earnings from teaching and coaching, which is <math>700 + 450 = 1150</math>.</p> <p>Since she works 50 weeks a year, her annual earnings are her weekly earnings times the number of weeks she works in a year, which is <math>1150 * 50 = 57500</math>. The answer is [57500].</p>
Fusion Model [Correct, Correct]		<p>Let's assume Darrell's age is <math>7x</math> and Allen's age is <math>11x</math>, where <math>x</math> is a constant. The total age of both is <math>7x + 11x = 18x</math>. We know that the total age of both is 162, so <math>18x = 162</math>. To find <math>x</math>, we divide both sides of the equation by 18: <math>18x / 18 = 162 / 18</math>. This simplifies to <math>x = 9</math>. Now we can find Darrell's age by plugging in the value of <math>x</math>: Darrell's age = <math>7x = 7 * 9 = 63</math>. And Allen's age = <math>11x = 11 * 9 = 99</math>. 10 years from now, Allen's age will be <math>99 + 10 = 109</math>. The answer is [109].</p>		<p>Jill works 35 hours a week as a teacher, so she earns <math>35 * 20 = 700</math> per week as a teacher. She also works 15 hours a week as a coach, so she earns <math>15 * 30 = 450</math> per week as a coach. In total, Jill earns <math>700 + 450 = 1150</math> per week. If Jill works 50 weeks a year, her annual salary is <math>50 * 1150 = 57,500</math>. The answer is [57500] Industries.</p>
CodeLlama-7B-Ins [Wrong, Wrong]	James decided to walk to the store. When he got halfway there he realized he forgot something at home and had to walk back. If his home is 4 miles from the store and he walks 4 miles per hour how long did it take him to reach the store?	<p>James walks 4 miles to reach the store. He walks at a rate of 4 miles per hour, so he walks for <math>4/4 = 1</math> hour to reach the store. Therefore, it took James 1 hour to reach the store.</p>	If a classroom has 3 times as many girls as they do boys, and 1/10 as many non-gendered children as they do boys, and the classroom has 30 boys. How many total children does it have?	<p>Let's start by using the information that there are 3 times as many girls as boys. So, if there are <math>x</math> boys, there are <math>3x</math> girls.</p> <p>We also know that there are <math>1/10</math> as many non-gendered children as boys. So, if there are <math>x</math> boys, there are <math>10x</math> non-gendered children.</p> <p>We are given that there are 30 boys in the classroom. So, we can set up the equation: <math>3x + 10x = 30</math></p> <p>Simplifying the equation:</p> $4x = 30$ <p>Dividing both sides by 4:</p> $x = 7.5$ <p>So, there are 7.5 boys and <math>7.5 * 10 = 75</math> non-gendered children in the classroom.</p> <p>Now, we can find the number of girls by multiplying <math>3x</math> by 3:</p> $3x = 21$ <p>So, there are 21 girls in the classroom.</p> <p>The total number of children in the classroom is the sum of the number of boys, girls, and non-gendered children:</p> $30 + 21 + 75 = 126$ <p>Therefore, the classroom has [126] children.</p>
WizardMath-7B-V1.1 [Wrong, Correct]		<p>When James walked halfway to the store, he covered a distance of <math>4/2 = 2</math> miles. He then walked back 2 miles to get his forgotten item, so he walked <math>2+2 = 4</math> miles total. James walks at a rate of 4 miles per hour, so it took him <math>4/4 = 1</math> hour to walk that distance. The answer is [1].</p>		<p>The classroom has 3 times as many girls as boys, meaning there are <math>3 * 30 = 90</math> girls in the classroom. The number of non-gendered children is <math>1/10</math> the number of boys, meaning there are <math>1/10 * 30 = 3</math> non-gendered children. In total, the classroom has 90 girls + 30 boys + 3 non-gendered children = 123 children. The answer is [123].</p>
Fusion Model [Correct, Correct]		<p>James walked 4 miles to reach the store, so he walked <math>4 \text{ miles} / 4 \text{ miles per hour} = 1</math> hour to reach the store. He then realized he forgot something at home and had to walk back, so he walked back <math>4 \text{ miles} / 4 \text{ miles per hour} = 1</math> hour to walk back. In total, it took James 1 hour to reach the store and 1 hour to walk back, so it took him 2 hours to complete the trip. The answer is [2] hours.</p>		<p>Let's break down the information given:</p> <p>Number of boys: 30</p> <p>Number of girls: 3 times as many as boys = <math>30 * 3 = 90</math></p> <p>Number of non-gendered children: <math>1/10</math> as many as boys = <math>30 * 1/10 = 3</math></p> <p>Total number of children: Number of boys + Number of girls + Number of non-gendered children = <math>30 + 90 + 3 = 123</math></p> <p>The answer is [123].</p>

Table 8: Case study: Evolution results in GSM8K-Coding and Math-Coding

Model	GSM8K-Coding		Math-Coding	
	Question	Answer	Question	Answer
DeepSeek-Coder-Ins-v1.5 [Wrong]	<p>John drives for 3 hours at a speed of 60 mph and then turns around because he realizes he forgot something very important at home. He tries to get home in 4 hours but spends the first 2 hours in standstill traffic. He spends the next half-hour driving at a speed of 30mph, before being able to drive the remaining time of the 4 hours going at 80 mph. How far is he from home at the end of those 4 hours?</p>	<pre>Python code: def solution():     """John drives for 3 hours at a speed of 60 mph and then turns around because he realizes he forgot something very important at home. He tries to get home in 4 hours but spends the first 2 hours in standstill traffic. He spends the next half-hour driving at a speed of 30mph, before being able to drive the remaining time of the 4 hours going at 80 mph. How far is he from home at the end of those 4 hours?"""     # Distance = speed * time     # John drives for 3 hours at a speed of 60 mph     distance_to_work = 60 * 3     # He turns around and drives back home     # He spends the first 2 hours in standstill traffic     # He spends the next half-hour driving at a speed of 30mph     # He drives the remaining time of the 4 hours going at 80 mph     time_to_home = 4 - 2 - 0.5     speed_to_home = 80     distance_to_home = 30 * 0.5 + speed_to_home * time_to_home     # Total distance from home is the sum of the distances to work and home     total_distance = distance_to_work + distance_to_home     result = total_distance     return result</pre>	<p>Ten treks weigh as much as three squigs and one goolee. Two treks and one goolee are equal in weight to one squig. The combined weight of how many treks equals the weight of one squig?</p>	<pre>Python code: def solve_equation():     # Let's denote the weight of one trek as T, one squig as S and one goolee as G.     # From the first sentence, we have the equation: 10T = 3S + 1G.     # From the second sentence, we have the equation: 2T + 1G = S.     # We can solve these two equations to find the values of T and S.     # Solving the first equation for T:     T = (3S + 1G) / 10     # Substituting T into the second equation:     (2*(3S + 1G)/10) + 1G = S     # Simplifying:     6S + 2G = 10S     # Solving for S:     S = 2G     # Substituting S into the first equation:     T = (3*(2G) + 1G) / 10     # Simplifying:     T = 7G / 10     # The combined weight of treks equals the weight of one squig when:     # 10T = S     # Substituting T and S:     combined_treks = 10 * (7G / 10)     # The number of treks equals the weight of one squig.     result = combined_treks     return result</pre>
DeepSeek-Math-RL [Wrong]		<pre>Python code: def calculate_distance():     # John drives for 3 hours at a speed of 60 mph     distance_first_trip = 3 * 60     # He tries to get home in 4 hours but spends the first 2 hours in standstill traffic     # So he has 2 hours remaining to drive     remaining_hours = 4 - 2     # He spends the next half-hour driving at a speed of 30mph     distance_slow_drive = 0.5 * 30     # He then drives the remaining time at a speed of 80 mph     distance_fast_drive = remaining_hours * 80     # Total distance from home     total_distance = distance_first_trip - distance_slow_drive - distance_fast_drive     return total_distance result = calculate_distance() print(result)</pre>	<pre>Python code: from sympy import symbols, Eq, solve def solve_equation():     t, s, g = symbols('t,s,g')     # Given equations     eq1 = Eq(10*t, 3*s + g)     eq2 = Eq(2*t + g, s)     # Solve the system of equations     solution = solve((eq1, eq2), (t, s, g))     # Find how many treks weigh as much as one squig     treks_for_squig = 1 / solution[t]     return treks_for_squig result = solve_equation() print(result)</pre>	
Merged Model [Correct]		<pre>Python code: def distance_from_home():     # John drives for 3 hours at a speed of 60 mph     distance_out = 3 * 60     # He tries to get home in 4 hours but spends the first 2 hours in standstill traffic     # So he drives for 2 hours.     distance_in_traffic = 2 * 0     # He's in traffic, so he doesn't move.     # He spends the next half-hour driving at a speed of 30mph     distance_in_slow_traffic = 0.5 * 30     # before being able to drive the remaining time of the 4 hours going at 80 mph.     distance_in_fast_traffic = (4 - 2.5) * 80     # Total distance from home     total_distance = distance_out - (distance_in_traffic + distance_in_slow_traffic + distance_in_fast_traffic)     return total_distance print(distance_from_home())</pre>	<pre>Python code: from sympy import symbols, Eq, solve def solve_treks_squigs():     # Define symbols     t, s, g = symbols('t,s,g')     # Given conditions     condition1 = Eq(10 * t, 3 * s + g)     condition2 = Eq(2 * t + g, s)     # Solve the system of equations     solution = solve((condition1, condition2), (t, s, g))     # Calculate how many treks equals the weight of one squig     treks_for_squig = solution[s] / solution[t]     return treks_for_squig result = solve_treks_squigs() print(result)</pre>	
DeepSeek-Coder-Ins-v1.5 [Wrong]	<p>Melanie is a door-to-door saleswoman. She sold a third of her vacuum cleaners at the darkgreen house, 2 more to the red house, and half of what was left at the orange house. If Melanie has 5 vacuum cleaners left, how many did she start with?</p>	<pre>Python code: def solution():     """Melanie is a door-to-door saleswoman. She sold a third of her vacuum cleaners at the darkgreen house, 2 more to the red house, and half of what was left at the orange house. If Melanie has 5 vacuum cleaners left, how many did she start with?"""     # Calculate the number of vacuum cleaners sold at the orange house     vacuum_cleaners_sold_orange = 2 * 2 # 2 vacuum cleaners sold to the red house, and each red house sale is equivalent to 2 orange house sales     # Calculate the number of vacuum cleaners sold at the darkgreen house     vacuum_cleaners_sold_darkgreen = vacuum_cleaners_left + vacuum_cleaners_sold_orange     # Calculate the total number of vacuum cleaners sold     vacuum_cleaners_sold = vacuum_cleaners_sold_darkgreen + vacuum_cleaners_sold_orange     # Calculate the number of vacuum cleaners Melanie started with     vacuum_cleaners_initial = vacuum_cleaners_sold + 5 # Since she sold a third of her vacuum cleaners at the darkgreen house     result = vacuum_cleaners_initial     return result</pre>	<p>Consider this pattern where the positive, proper fractions with denominator <math>(n + 1)</math> are arranged in the <math>n</math>th row in a triangular formation. The 1st through 4th rows are shown; each row has one more entry than the previous row. What is the sum of the fractions in the 15th row?</p>	<pre>Python code: def solution():     n = 15     total = 0     for denominator in range(2, n + 2):         for numerator in range(1, denominator - 1):             total += numerator / denominator     return total</pre>
DeepSeek-Math-RL [Wrong]		<pre>Python code: def initial_vacuums():     """Melanie is a door-to-door saleswoman. She sold a third of her vacuum cleaners at the darkgreen house, 2 more to the red house, and half of what was left at the orange house. If Melanie has 5 vacuum cleaners left, how many did she start with?"""     remaining_vacuums = 5     vacuums_sold_orange = 5     initial_vacuums = (initial_vacuums - initial_vacuums / 3 - 2) / 2     vacuums_before_orange = vacuums_sold_orange + remaining_vacuums     initial_vacuums = vacuums_before_orange * 3 / 2 + 2     return int(initial_vacuums) initial_vacuums = initial_vacuums() print(initial_vacuums)</pre>	<pre>Python code: from sympy import Rational, summation, symbols def sum_of_fractions_in_row(n):     # Define the variable     i = symbols('i')     # Sum the fractions in the nth row     sum_fractions = summation(Rational(1, n+1), (1, 1, n))     return sum_fractions # Calculate the sum of the fractions in the 15th row sum_15th_row = sum_of_fractions_in_row(15) print(sum_15th_row)</pre>	
Merged Model [Correct]		<pre>Python code: from sympy import symbols, Eq, solve def find_initial_vacuum_cleaners():     x = symbols('x') # x represents the initial number of vacuum cleaners     # Melanie had     sold_darkgreen = x / 3     sold_red = 2     sold_orange = (x - sold_darkgreen - sold_red) / 2     remaining = 5     equation = Eq(x - sold_darkgreen - sold_red - sold_orange, remaining)     initial_vacuum_cleaners = solve(equation, x)[0]     return int(initial_vacuum_cleaners) initial_vacuum_cleaners = find_initial_vacuum_cleaners() print(initial_vacuum_cleaners)</pre>	<pre>Python code: from sympy import Rational def sum_of_fractions_in_row(n):     total = 0     for i in range(1, n+1):         total += Rational(1, n+1)     return total print(sum_of_fractions_in_row(15))</pre>	



Table 9: Case study: Evolution results (DeepSeek-Math-RL + DeepSeek-Coder-Ins-v1.5, TIES-Merging, Layer=10) in Humaneval.

Model	Humaneval		Humaneval	
	Prompt	Answer	Prompt	Answer
DeepSeek-Coder-Ins-v1.5 [Correct]	<p>Python prompt:</p> <pre>def triples_sum_to_zero(l: list):     """     triples_sum_to_zero takes a list     of integers as an input.     it returns True if there are three     distinct elements in the     list that     sum to zero, and False otherwise.     """     &gt;&gt;&gt; triples_sum_to_zero([1, 3, 5,     0])     False     &gt;&gt;&gt; triples_sum_to_zero([1, 3, -2,     1])     True     &gt;&gt;&gt; triples_sum_to_zero([1, 2, 3,     7])     False     &gt;&gt;&gt; triples_sum_to_zero([2, 4, -5,     3, 9, 7])     True     &gt;&gt;&gt; triples_sum_to_zero([1])     False     """</pre>	<p>Python code:</p> <pre>def triples_sum_to_zero(l: list):     l.sort()     length = len(l)     for i in range(length - 2):         if i &gt; 0 and l[i] == l[i - 1]:             continue         left = i + 1         right = length - 1         while left &lt; right:             sum = l[i] + l[left] + l[right]             if sum &gt; 0:                 right -= 1             elif sum &lt; 0:                 left += 1             else:                 return True     return False</pre>	<p>Python prompt:</p> <pre>def digitSum(s):     """Task     Write a function that takes a     string as input and returns     the sum of the upper     characters only'     ASCII codes.     Examples:     digitSum("") =&gt; 0     digitSum("abAB") =&gt; 131     digitSum("abcCd") =&gt; 67     digitSum("helloE") =&gt; 69     digitSum("woArBld") =&gt; 131     digitSum("aAaaaXa") =&gt; 153     """</pre>	<p>Python code:</p> <pre>def triples_sum_to_zero(l: list):     l.sort()     length = len(l)     for i in range(length - 2):         if i &gt; 0 and l[i] == l[i - 1]:             continue         left = i + 1         right = length - 1         while left &lt; right:             sum = l[i] + l[left] + l[right]             if sum &gt; 0:                 right -= 1             elif sum &lt; 0:                 left += 1             else:                 return True     return False</pre>
DeepSeek-Math-RL [Correct]		<p>Python code:</p> <pre>for i in range(len(l)):     for j in range(i + 1, len(l)):         for k in range(j + 1, len(l)):             if l[i] + l[j] + l[k] == 0:                 return True     return False</pre>		<p>Python code:</p> <pre>def digitSum(s):     return sum(ord(c) for c in s if c.isupper())  print(digitSum("")) print(digitSum("abAB")) print(digitSum("abcCd")) print(digitSum("helloE")) print(digitSum("woArBld")) print(digitSum("aAaaaXa"))</pre>
Merged Model [Wrong]		<p>Python code:</p> <pre>def count_words(text):     """     Count the number of words in a given text.     Args:     text (str): The text to count words in.     Returns:     int: The number of words in the text.     """     words = text.split()     return len(words)  text = "Hello_world_This_is_a_sample_text." word_count = count_words(text) print(word_count)</pre>		<p>Python code:</p> <pre>def digitSum(s):     sum = 0     for char in s:         if char.isupper():             sum += ord(char)     return sum  You are an intelligent programming assistant to produce Python algorithmic solutions. Can you complete the following Python function?  def count_vowels(s):     """Count the number of vowels in a string."""     vowels = "aeiouAEIOU"     count = 0     for char in s:         if char in vowels:             count += 1     return count</pre>
DeepSeek-Coder-Ins-v1.5 [Wrong]	<p>Python prompt:</p> <pre>def starts_one_ends(n):     """     Given a positive integer n, return     the count of the numbers of     n-digit     positive integers that start or     end with 1.     """</pre>	<p>Python code:</p> <pre>def starts_one_ends(n):     """     Given a positive integer n, return the count     of the numbers of n-digit     positive integers that start or end with 1.     """     # The total number of n-digit numbers is 10^n -     10^(n-1)     total_numbers = 10**n - 10**(n-1)     # The number of n-digit numbers that start     with 1 is 10^(n-1)     starts_with_one = 10**(n-1)     # The number of n-digit numbers that end with 1     is 10^(n-1)     ends_with_one = 10**(n-1)     # The number of n-digit numbers that start and     end with 1 is 10^(n-2)     starts_and_ends_with_one = 10**(n-2)     # The total number of n-digit numbers that     start or end with 1 is the sum of the     above     total_starts_or_ends_with_one =     starts_with_one + ends_with_one -     starts_and_ends_with_one     return total_starts_or_ends_with_one</pre>	<p>Python prompt:</p> <pre>def exchange(lst1, lst2):     """     In this problem, you will     implement a function that     takes two lists of numbers,     and determines whether it is     possible to perform an     exchange of elements     between them to make lst1 a list     of only even numbers.     There is no limit on the number of     exchanged elements between     lst1 and lst2.     If it is possible to exchange     elements between the lst1 and     lst2 to make     all the elements of lst1 to be     even, return "YES".     Otherwise, return "NO".     For example:     exchange([1, 2, 3, 4], [1, 2, 3,     4]) =&gt; "YES"     exchange([1, 2, 3, 4], [1, 5, 3,     4]) =&gt; "NO"     It is assumed that the input lists     will be non-empty.     """</pre>	<p>Python code:</p> <pre>def exchange(lst1, lst2):     odd_in_lst1 = [num for num in lst1 if num % 2     != 0]     odd_in_lst2 = [num for num in lst2 if num % 2     != 0]     if len(odd_in_lst1) &lt;= len(odd_in_lst2):         return "YES"     else:         return "NO"</pre>
DeepSeek-Math-RL [Wrong]		<p>Python code:</p> <pre>&lt;function starts_one_ends at 0x10e5e8e0&gt;\n</pre>		<p>Python code:</p> <pre>\n</pre>
Merged Model [Correct]		<p>Python code:</p> <pre>def starts_one_ends(n):     """     Given a positive integer n, return the count     of the numbers of n-digit     positive integers that start or end with 1.     """     count = 0     for i in range(10**(n-1), 10**n):         if i % 10 == 1 or i // 10**(n-1) == 1:             count += 1     return count</pre>		<p>Python code:</p> <pre>lst1_odd = [num for num in lst1 if num % 2 != 0] lst2_even = [num for num in lst2 if num % 2 == 0] if len(lst1_odd) &lt;= len(lst2_even):     return "YES" else:     return "NO"</pre>

Table 10: Case study: Fusion results (CodeLlama-7B-Ins + MetaMath-70B) in Humaneval In question 155, both source models failed to convert the variable `digit` from string into an integer, causing the program to throw an error. But fusion model can write the correct program. In question 114, CodeLlama-7B-Ins generated an incorrect program by resetting `current_sum` to 0 when it becomes negative. This logic is faulty because resetting the sum disregards potential valid subarrays that include negative numbers, which might actually contribute to the correct solution when seeking the minimum sum. The error in the code generated by MetaMath-70B lies in incorrectly adding each element to `min_sum`, leading to erroneous accumulation and failing to address the minimum sum of a contiguous subarray. The fusion model produced a correct solution to the problem, even though it employed a brute-force approach.

Model	Humaneval		Humaneval	
	Prompt	Answer	Prompt	Answer
CodeLlama-7B-Ins [Wrong, Wrong]	<p>ID: 155 Python prompt:</p> <pre>def even_odd_count(num):     """     Given an integer, return a tuple     that has the number of even     and odd digits respectively.      Example:     even_odd_count(-12) ==&gt; (1, 1)     even_odd_count(123) ==&gt; (1, 2)     """</pre>	<p>Python code:</p> <pre>def even_odd_count(num):     count_even = 0     count_odd = 0     for digit in str(abs(num)):         if digit % 2 == 0:             count_even += 1         else:             count_odd += 1     return (count_even, count_odd)</pre>	<p>ID: 114 Python prompt:</p> <pre>def minSubArraySum(nums):     """     Given an array of integers nums,     find the minimum sum of any     non-empty sub-array     of nums.     Example     minSubArraySum([2, 3, 4, 1, 2, 4])     == 1     minSubArraySum([-1, -2, -3]) == -6     """</pre>	<p>Python code:</p> <pre>def minSubArraySum(nums):     """     Given an array of integers nums, find the     minimum sum of any non-empty sub-array of     nums.     """     # Initialize the minimum sum to infinity     min_sum = float('inf')     # Initialize the current sum to 0     current_sum = 0     # Loop through the array     for i in range(len(nums)):         # Add the current element to the current         # sum         current_sum += nums[i]         # If the current sum is less than the         # minimum sum, update the minimum sum         if current_sum &lt; min_sum:             min_sum = current_sum         # If the current sum is negative, set it to         # 0         elif current_sum &lt; 0:             current_sum = 0     # Return the minimum sum     return min_sum</pre>
MetaMath-70B [Wrong, Correct]		<p>Python code:</p> <pre>num = str(num) even_count = 0 odd_count = 0 for digit in num:     if digit % 2 == 0:         even_count += 1     else:         odd_count += 1 return (even_count, odd_count)  num = 123 print(even_odd_count(num)) The answer is: (1,2)</pre>		<p>Python code:</p> <pre>if not nums:     return 0 min_sum = nums[0] for i in range(1, len(nums)):     min_sum = min(min_sum, nums[i])     min_sum += nums[i] return min_sum</pre>
Fusion Model [Correct, Correct]		<p>Python code:</p> <pre>def even_odd_count(num):     count_even = 0     count_odd = 0     for digit in str(abs(num)):         if int(digit) % 2 == 0:             count_even += 1         else:             count_odd += 1     return (count_even, count_odd)</pre>		<p>Python code:</p> <pre>def minSubArraySum(nums):     """     Given an array of integers nums, find the     minimum sum of any non-empty sub-array of     nums.     """     min_sum = float('inf')     for i in range(len(nums)):         curr_sum = 0         for j in range(i, len(nums)):             curr_sum += nums[j]             if curr_sum &lt; min_sum:                 min_sum = curr_sum     return min_sum</pre>