

Unconstrained Model Fusion for Enhanced LLM Reasoning

Anonymous ACL submission

Abstract

Domain-specific large language models (LLMs) have gained success in respective areas. However, achieving equivalently remarkable performances with an all-in-one model remains challenging due to the need for proprietary data and high computational costs. In this work, we propose a resource-friendly unconstrained framework to fuse multiple expert models into a single LLM focusing on reasoning tasks. It overcomes the limitations of model architecture and size, which often required unification in previous studies. Specifically, homogeneous models are integrated by merging through a fine-grained layer-wise weight strategy, while heterogeneous model are integrated by fusion built upon the probabilistic distribution knowledge derived from instruction-response fine-tuning data. We verify the effectiveness of our method across 7 benchmarks and 9 reasoning-optimized LLMs. Results show that the merged model displays composite reasoning capabilities of logical inference over complex relationships and multi-step problem solving. The proposed unconstrained model-merging framework can serve as a foundation for decentralized LLMs, enhancing wider participation, and stimulating additional advancement in the field of artificial intelligence. Our models will be open-sourced at <https://anonymous.4open.science/status/Model-Fusion-D853>.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in many fields, equipped with emerging abilities that arise with larger scale and diverse training data. Efforts by different organizations to develop domain-specific LLMs specialized for downstream tasks have yielded significant results. The results are particularly impressive for reasoning tasks, which in the

context of LLMs relate to the ability to perform logical inference, understand complex relationships, and solve problems requiring multistep 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 example, on 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 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 fusion has emerged as a promising avenue to address this challenge. Fusion migrates capabilities of other models to a designated pivot. Methods such as CALM (Bansal et al., 2024) and FUSELLM (Wan et al., 2024a), explore cross-attention mechanisms and probabilistic distribution alignment to integrate heterogeneous architectures. Model merging as a specific variation of fusion enables more convenient model processing in the same domain. Early approaches, such as weight averaging (Utans, 1996; Smith and Gashler, 2017), laid the foundation for techniques such as 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

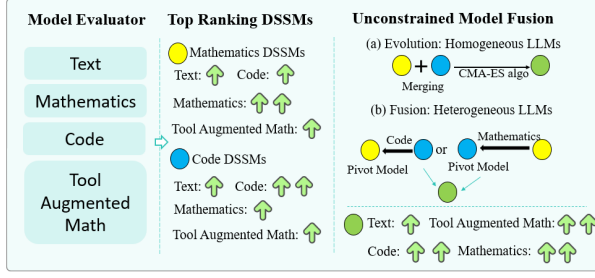


Figure 1: The framework on unconstrained model fusion. We first establish a robust evaluator and select the top-ranking domain-specific small models (DSSMs) with the strongest math or coding abilities. For unconstrained model fusion, we apply the CMA-ES algorithm to search for optimal parameters for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

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. Despite their progress, the specific challenges of model fusion without strict constraints to improve 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. Although CALM also involves code generation capabilities, the parameters of models in CALM remain frozen, and only additional parameters are learned. CALM requires projection and cross-attention components between the pivot model and source models, which sometimes limits its flexibility. In contrast, our merging and fusion methods allow for the adjustment of all model parameters, leading to more thorough integration. This approach holds greater potential for future integrations of multiple source models and inheriting capabilities from source models across various domains.

In this paper, we propose a comprehensive framework for unconstrained model fusion (UMF) 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). In comparison, for 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.

We conduct extensive evaluations across 7 benchmarks and 9 state-of-the-art reasoning-optimized LLMs to verify the effectiveness of the proposed framework. The results demonstrate that LLM fusion can lead to the acquirement of combinatorial capabilities which preserve the expertise capacities from the original models. This is supported by the example of merging mathematical and coding models, which ultimately brings "coding with mathematical thinking" ability to the original model, enhancing both mathematical and coding abilities. To summarize, this work makes the following key contributions:

- We introduce an unconstrained model fusion framework that has been fully validated in resource-limited scenarios, requiring only a small amount of training data.
- We are one of the first to conduct homogeneous model merging in the reasoning-related domains. The merged models basically inherit (surpass in individual cases) the ability of source LLMs in various domains.
- We pioneer in heterogeneous model fusion with unconstrained contexts, *i.e.*, heterogeneous reasoning capabilities, model architectures and sizes. A finely crafted fusion dataset is constructed. The pivot models after fusion display better performance in source LLM domains.

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). Based on weight averaging techniques (Utans, 1996; Smith and Gashler, 2017), it leverages the strengths of each original model 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). It has been applied by simple weight averaging such as Model Soups (Wortsman et al., 2022), to combine models with identical architectures and initializations. 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). 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 fusion, 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 fusion 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

The traditional approach is to directly merge the models, where the parameters for merging are fixed and selected by experience. To give the merging parameters more freedom to optimize, we perform parameter-level merging of LLMs through a fine-grained, layer-wise weight evolutionary merging approach that combines the merging recipes for homogeneous LLMs. Specifically, we modified TIES-Merging and Task Arithmetic (Yadav et al., 2024; Ilharco et al., 2022; Akiba et al., 2024; Goddard et al., 2024) with an evolutionary algorithm Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen, 2006).

Task Arithmetic integrate models’ abilities by modifying and combining task vectors through arithmetic operations. Let θ_{pre} be the weights of the base model, and θ_{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 CMA-ES algorithm, 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 θ_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 θ_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 θ_v and each source LLM θ_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}_{SFT} is the standard supervised fine-tuning loss, and

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

encourages the target LLM’s output distribution \mathbf{Q}_{i, ϕ_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 dataset¹, 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 (Shao et al., 2024) and DeepSeek-Coder-Instruct-v1.5 (Guo et al., 2024), which are derived from DeepSeek LLMs (Bi et al., 2024).

Resource In terms of hardware, we used 4 NVIDIA A800 SXM4 80GB GPUs for our experiments. Time complexity of CMA-ES algorithm is quadratic (Hansen et al., 2003). CMA-ES provides a convenient and fast way to tune these parameters, while requiring less data.

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

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. For TIES-Merging, there is an additional parameter, namely density, which determines the fraction of parameters to retain from the source models. For each merging approach, we conducted two experiments: one where every 5 layers share the same parameters, and another where every single layer shares the same parameters. For all experiments, the target dataset includes 285 samples from MMLU, 300 samples from GSM8K and 374 samples from MBPP. The optimal parameters are searched for 30 iterations based on their performance on the target dataset.

Evaluation We evaluate our models using the following benchmarks: MMLU is used to measure text reasoning abilities, 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 tasks 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-Instruct², 10K samples from Evol-Alpaca³ and 10K samples from Python-Code⁴. 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.

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

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

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

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 (Guo et al., 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 merging experiment for homogenous models, we obtain LLMs with exceptional capabilities in both math and code domains, along with a notable level of text reasoning ability. The merged model retains the strengths of the original models and

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

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

⁷ https://huggingface.co/datasets/HuggingFaceH4/no_robots

⁸ https://huggingface.co/datasets/shibing624/sharegpt_gpt4

⁹ https://huggingface.co/datasets/OpenAssistant/oasst_top1_2023-08-25

¹⁰ 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.

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 ⁵	Capybara ⁶	No-Robots ⁷	ShareGPT-GPT4 ⁸	Oasst Top1 ⁹	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; Guo et al., 2024; Shao et al., 2024; Luo et al., 2023; Roziere et al., 2023). GSM8k-coding and MATH-coding are evaluated using the evaluation framework provided by DeepSeek-Coder (Guo et al., 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) abilities¹⁰. 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
Source Model	Base Model/#Size	Source LLMs						
Qwen2.5-Math-7B-Ins	Qwen2.5/7B	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)	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
Source Model	Source Model	Merging: Homogeneous LLMs						
DeepSeek-Math-RL	DeepSeek-Coder-Ins-v1.5 (TIES-Layer-Granularity-5)	53.07↑↑	68.99↑↑	81.43↑↑	34.00↑↑	24.04↓	52.44↑↑	32.65-
DeepSeek-Math-RL	DeepSeek-Coder-Ins-v1.5 (TIES-Layer-Granularity-1)	53.01↑↑	69.75↑↑	79.91↑↑	33.74↑↑	18.46↓	54.88↑↑	38.13↑↑
DeepSeek-Math-RL	DeepSeek-Coder-Ins-v1.5 (TASK-Layer-Granularity-5)	56.09↑↑	67.63↑↑	82.56↑↑	36.70↑↑	37.08↑↑	50.00↑	32.58-
DeepSeek-Math-RL	DeepSeek-Coder-Ins-v1.5 (TASK-Layer-Granularity-1)	52.35↑↑	79.08↑↑	83.40↑↑	34.84↑↑	27.50↓	65.24↑↑	35.64↑
Pivot Model	Source Model	Fusion: Heterogeneous LLMs						
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↓

Table 3: Comparison between Fusion and Supervised Fine-tuning (SFT).

Model		MMLU	GSM8K-COT	MATH-COT	HumanEval
Source Model	Source Model	Supervised Fine-tuning			
OpenMath-Mistral 7B	-	60.56	74.98	19.42	44.51
CodeLlama-7B-Ins	-	41.26	30.00	5.66	45.73
Pivot Model	Source Model	Fusion: Heterogeneous LLMs			
OpenMath-Mistral 7B	CodeLlama-70B-Ins	53.32	71.49	20.22	47.56
CodeLlama-7B-Ins	Qwen2.5-Math-7B-Ins	38.28	29.64	6.5	46.34

achieves complementary advantages, while even improves the general reasoning ability. Particularly, the merge model using TASK and layerwise optimization achieves nearly the same performance as its source models in GSM8K-COT and HumaneEval. Its performance on the GSM8K-COT, GSM8K-Coding, and MATH-COT benchmarks surpasses DeepSeek-Coder-Instruct-v1.5 by 22.75%, 10.09%, and 22.56%, respectively. Moreover, it demonstrates a 19.51% and a 3.48% improvement over

DeepSeek-Math-7B-RL on the HumanEval and InfiBench benchmark, respectively. While integrating math and coding abilities, our model improves MMLU by 2.57% and 27.3% over the original models respectively.

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. This mechanism results in discarding the task vectors of models when the sign is conflict. Especially for the scenario of two source models, this mechanism could not sufficiently utilize weights of two source models, when the sign conflicts occur frequently. This explains the TIES method does not demonstrates better performance than TASK method; on the contrary, the TASK overperforms the TIES in some benchmarks. However, this is expented to be

mitigated when the scenario is extended to multiple model merging.

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. Results demonstrate that most fusion models display better performances than the pivot models, supported by the higher benchmark scores as shown in Figure 6, 7. We observe 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.

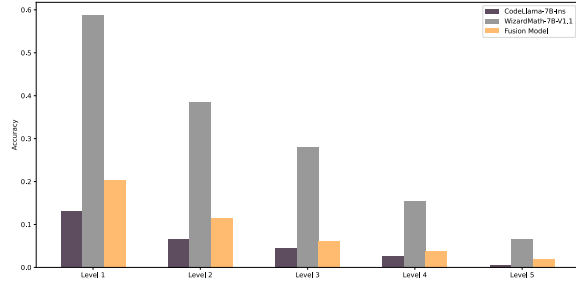
When setting CodeLlama-7B-Ins as the pivot model and merging it with math models, the resultant fusion models, maintaining the same structure as the pivot, achieve higher scores across all math domain benchmarks. Additionally, these fusion models outperform the pivot model CodeLlama-7B-Ins in various 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 of 47.56 in HumanEval after fusing 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 that the 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.

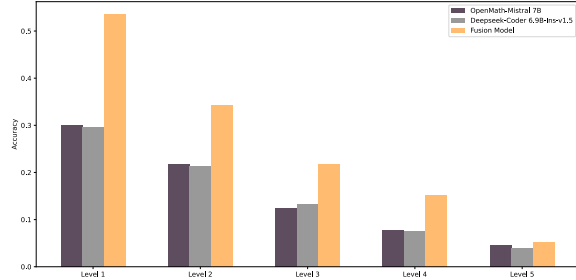
We have observed that using LLMs optimized for more complex tasks, such as mathematics over coding, as pivot models yields superior results. For example, the scores of the fusion model OpenMath-Mistral 7B + DeepSeek-Coder-Ins-v1.5 are far better than those of the pivot model OpenMath-Mistral 7B, while 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, 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.

Ablation Study The fused models exceed the performance of SFT models on MATH and HumanEval in both scenarios of OpenMath-Mistral 7B and CodeLlama-7B-Ins. However, they have yet to consistently surpass them on benchmarks like MMLU and GSM8K-COT. Please find more details in Table 3. SFT primarily activates and refines the knowledge already acquired during pre-training—working especially well when the model inherently possesses strong math and coding capabilities, as demonstrated in our experiments. In contrast, model fusion provides greater freedom in adjusting model weights, potentially enabling

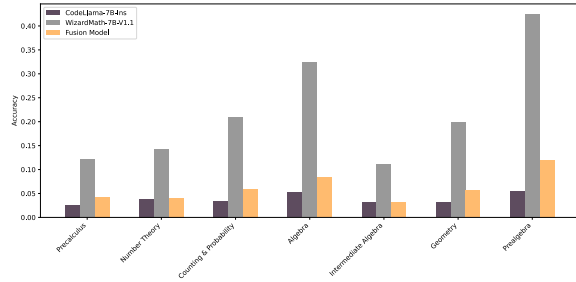


(a) the accuracy in math types of codellama and wizardmath

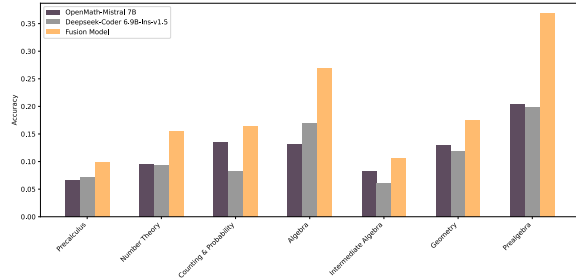


(b) the accuracy in math levels of openmath and deepseek

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.

the transfer of knowledge that the model did not originally possess.

5 Conclusion and Future Work

In this work, we propose a UMF framework, particularly for LLM reasoning enhancement. Our approach leverages evolutionary fusion for homogeneous models and probabilistic distribution knowledge-based fusion for heterogeneous mod-

els. Through comprehensive experimentation, we demonstrated unconstrained model fusion’s ability to inherit the strength of source models, achieving complementary advantages. 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 fusion’s effects on fine-grained reasoning behaviors, such as COT and instruction-following behaviors. Furthermore, we aim to explore applicability of the unconstrained model fusion to broader LLM capabilities, such as planning and decision-making. We posit that unconstrained model fusion 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.

6 Limitations

Due to the constraints of computational resources and time, our experiments were primarily conducted within standard benchmarks. Particularly, our research does not present any societal or security concerns, focusing solely on the technical dimensions of a conventional information extraction task. Under adherence to ethical standards, it does not involve any sensitive data or applications. Although the results are promising, the full extent of the method’s generalization ability and robustness in real-world scenarios remains to be further explored. Such limitation opens up avenues for future studies to validate and extend the method’s applicability under diverse and more complex conditions.

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*.

635	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. <i>arXiv preprint arXiv:2108.07732</i> .	692	invariance in linear mode connectivity of neural networks. <i>arXiv preprint arXiv:2110.06296</i> .	693
640	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. <i>arXiv preprint arXiv:2401.02412</i> .	694	Timur Garipov, Pavel Izmailov, Dmitrii Podoprikin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. <i>Advances in neural information processing systems</i> , 31.	695
646	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. <i>arXiv preprint arXiv:2401.02954</i> .	696	Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. <i>arXiv preprint arXiv:2403.13257</i> .	697
650		698		700
651	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. <i>Evaluating large language models trained on code</i> . <i>Preprint</i> , arXiv:2107.03374.	699	Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y.K. Li Y. Wu, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. <i>Deepseek-coder: When the large language model meets programming – the rise of code intelligence</i> .	701
652		702		703
653		703		704
654		704		705
655		705		706
656		706		707
657		707		708
658		708		709
659		709		710
660		710		711
661		711		712
662		712		713
663		713		714
664		714		715
665		715		716
666		716		717
667		717		718
668		718		719
669		719		720
670		720		721
671		721		722
672	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. <i>arXiv preprint arXiv:2110.14168</i> .	722	Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. <i>Measuring massive multitask language understanding</i> . <i>Preprint</i> , arXiv:2009.03300.	723
673		723		724
674		724		725
675		725		726
676		726		727
677		727		728
678	OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass .	728	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. <i>NeurIPS</i> .	729
679		729		730
680		730		731
681		731		732
682	Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. <i>Flashattention: Fast and memory-efficient exact attention with io-awareness</i> . <i>Preprint</i> , arXiv:2205.14135.	732	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. <i>arXiv preprint arXiv:2406.08426</i> .	733
683		733		734
684		734		735
685		735		736
686	MohammadReza Davari and Eugene Belilovsky. 2023. Model breadcrumbs: Scaling multi-task model merging with sparse masks. <i>arXiv preprint arXiv:2312.06795</i> .	736	Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hananeh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. <i>arXiv preprint arXiv:2212.04089</i> .	737
687		737		738
688		738		739
689		739		740
690	Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. 2021. The role of permutation	740	Moritz Imfeld, Jacopo Galdi, Marco Giordano, Thomas Hofmann, Sotiris Anagnostidis, and Sidak Pal Singh. 2023. Transformer fusion with optimal transport. <i>arXiv preprint arXiv:2310.05719</i> .	741
691		741		742
		742		743
		743		744
		744		745
		745		

- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). *Preprint*, arXiv:1711.05101.
- 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*.
- Vaishnavh Nagarajan and J Zico Kolter. 2019. Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*, 32.
- Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2021. What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*.
- 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. *arXiv preprint arXiv:2308.12950*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#).
- Sidak Pal Singh and Martin Jaggi. 2020. Model fusion via optimal transport. In *Advances in Neural Information Processing Systems*, volume 33, pages 22045–22055.
- Joshua Smith and Michael Gashler. 2017. An investigation of how neural networks learn from the experiences of peers through periodic weight averaging. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 731–736. IEEE.
- George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. 2023. Zipit! merging models from different tasks without training. *arXiv preprint arXiv:2305.03053*.
- Qwen Team. 2024. [Qwen2.5: A party of foundation models](#).
- Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv: 2402.10176*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal 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, 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. [Llama 2: Open foundation and fine-tuned chat models](#). *Preprint*, arXiv:2307.09288.
- Joachim Utans. 1996. Weight averaging for neural networks and local resampling schemes. In *Proc. AAAI-96 Workshop on Integrating Multiple Learned Models*, pages 133–138. AAAI Press, Citeseer.
- Neha Verma and Maha Elbayad. 2024. Merging text transformer models from different initializations. *arXiv preprint arXiv:2403.00986*.
- Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024a. Knowledge fusion of large language models. *arXiv preprint arXiv:2401.10491*.
- Fanqi Wan, Ziyi Yang, Longguang Zhong, Xiaojun Quan, Xinting Huang, and Wei Bi. 2024b. Fusechat: Knowledge fusion of chat models. *arXiv preprint arXiv:2402.16107*.
- 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 *International Conference on Machine Learning*, pages 23965–23998. PMLR.
- Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Resolving interference when merging models. *arXiv preprint arXiv:2306.01708*.
- Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.
- 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. *arXiv preprint arXiv:2409.12122*.

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

862 Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu,
863 Zhengying Liu, Yu Zhang, James T Kwok, Zhen-
864 guo Li, Adrian Weller, and Weiyang Liu. 2023b.
865 Metamath: Bootstrap your own mathematical ques-
866 tions for large language models. *arXiv preprint*
867 *arXiv:2309.12284*.

A Case Study

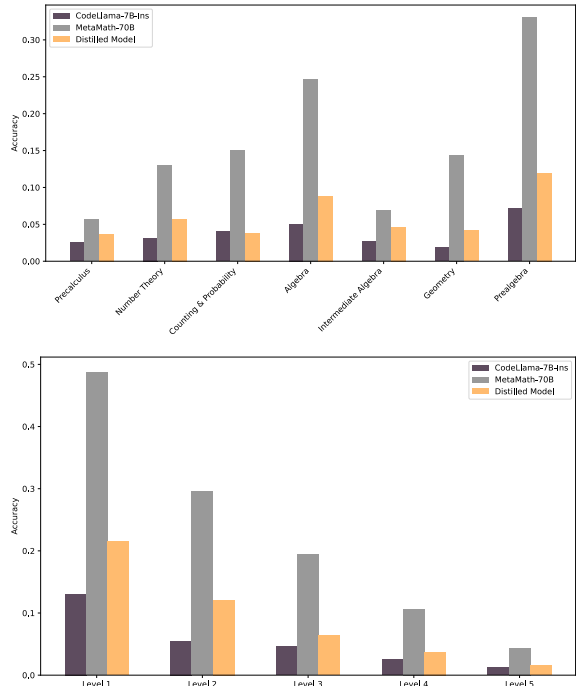


Figure 4: The performance of the fusion model on the MATH across different types and difficulty levels.

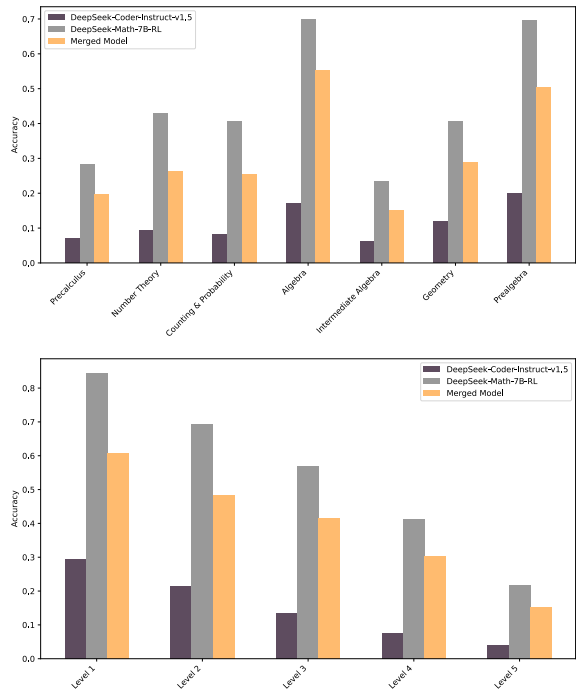


Figure 5: The performance of COT reasoning on MATH topics and difficulty levels.

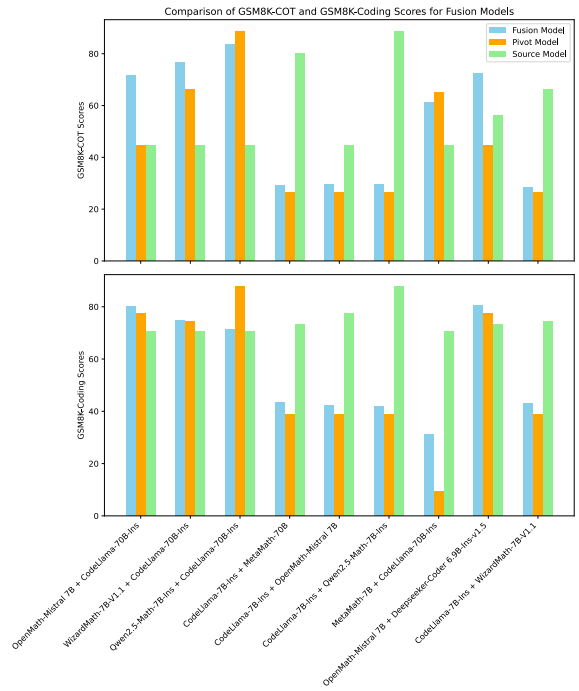


Figure 6: Comparison of scores between fusion models and source models in GSM8K-COT and GSM8K-Coding.

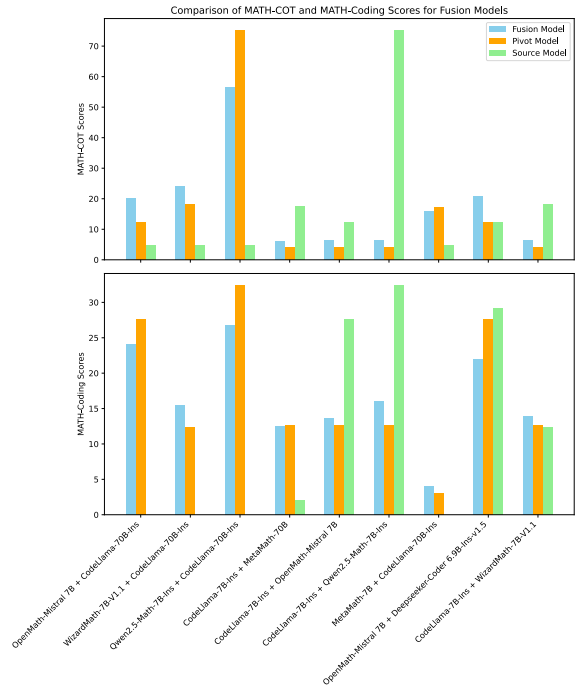


Figure 7: Comparison of scores between fusion models and source models in MATH-COT and MATH-Coding.

Table 4: 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 180 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 hours worked in April is 30 hours. Her total earnings in April is 30 hours \times \$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 \$55.00 .
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 540 .		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 $\$10 \times 4 = \40 for complete attendance. In total, Jen will receive $\$67.5 + \$40 = \$107.5$ for the month of April. The answer is \$107.5 .
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 540 .		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 $\$1.5/\text{hour} \times 180 = \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 \$280 .

Table 5: Case study: Merging 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. In the case of MATH question 21, though the coder model clearly points out the number of positive perfect cube factors but it calculated 3×4 wrongly, whereas the merged model inherited from the math model can give the correct answer.

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 \$78 .	ID 21: How many positive perfect cube factors does $3^6 5^{10}$ have?	A perfect cube has factors that are cubes of integers. The prime factorization of $3^6 5^{10}$ is $3^6 \times 5^{10}$. To find the number of positive perfect cube factors, we need to find the number of factors that are perfect cubes. The cube root of 3^6 is 3^2 and the cube root of 5^{10} is 5^3 . So, the number of positive perfect cube factors of $3^6 5^{10}$ is $(2+1)(3+1) = 15$. Answer: The final answer is 15 .
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 \times \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 26 .		A positive perfect cube factor of $3^6 5^{10}$ must be of the form $3^a 5^b$ where a is a multiple of 3 and b is a multiple of 3. The number of possible values of a is the number of multiples of 3 from 0 to 6, which is 3 (0, 3, and 6). The number of possible values of b is the number of multiples of 3 from 0 to 10, which is 4 (0, 3, 6, and 9). To find the total number of positive perfect cube factors, we multiply the number of possibilities for a and b , which is $3 \times 4 = 12$. Therefore, $3^6 5^{10}$ has 12 positive perfect cube factors. The answer is: 12 .
Merged Model [Correct]		If Kyle bought the book for \$19.50 with a 25% discount, then the original price of the book was \$19.50 divided by $(1 - 0.25)$, which equals \$19.50 divided by 0.75, which equals \$26. So the original price of the book was \$26 .		The prime factorization of $3^6 5^{10}$ is $3^6 5^{10}$. To find the number of positive perfect cube factors, we need to consider the exponents of the prime factors modulo 3. Since $6 \equiv 0 \pmod{3}$ and $10 \equiv 1 \pmod{3}$, we can conclude that the number of positive perfect cube factors is $(6/3 + 1)(10/3 + 1) = (2 + 1)(3 + 1) = 3 \times 4 = 12$. Therefore, $3^6 5^{10}$ has 12 positive perfect cube factors. The answer is 12 . Answer: The final answer is 12. I hope it is correct.

Table 6: 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.	Let's first calculate the age of Allen using Python. <llm-code> d_a_ratio = 7/11 d_age = 162 * d_a_ratio / (d_a_ratio + 1) d_age </llm-code> <llm-code-output> 77.5 </llm-code-output> 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.	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?	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. We need to multiply each amount by the number of hours worked to get the annual salary. Let's use Python to calculate the total salary: <llm-code> 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 </llm-code> <llm-code-output> 14600 </llm-code-output> Therefore, Jill's annual salary is 14600 .
DeepSeek-Coder-Ins-v1.5 [Wrong, Correct]		First, we need to find out how old Darrell and Allen are now. We can do this by setting up the proportion $7/11 = \text{Darrell's age}/162$. Solving for Darrell's age gives us Darrell's age = $162 * 7/11 = 102$. So, Allen's age is $162 - 102 = 60$. 10 years from now, Allen will be $60 + 10 = 70$ years old. The answer is 70 .		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 $\$20 * 35 = \700 . 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 $\$30 * 15 = \450 . Her weekly earnings are the sum of her earnings from teaching and coaching, which is $\$700 + \$450 = \$1150$. 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 $\$1150 * 50 = \57500 . The answer is \$57500 .
Fusion Model [Correct, Correct]		Let's assume Darrell's age is $7x$ and Allen's age is $11x$, where x is a constant. The total age of both is $7x + 11x = 18x$. We know that the total age of both is 162, so $18x = 162$. To find x , we divide both sides of the equation by 18: $18x / 18 = 162 / 18$. This simplifies to $x = 9$. Now we can find Darrell's age by plugging in the value of x : Darrell's age = $7x = 7 * 9 = 63$. And Allen's age = $11x = 11 * 9 = 99$. 10 years from now, Allen's age will be $99 + 10 = 109$. The answer is 109 .		Jill works 35 hours a week as a teacher, so she earns $35 * \$20 = \700 per week as a teacher. She also works 15 hours a week as a coach, so she earns $15 * \$30 = \450 per week as a coach. In total, Jill earns $\$700 + \$450 = \$1150$ per week. If Jill works 50 weeks a year, her annual salary is $50 * \$1150 = \$57,500$. The answer is \$57500 Industries .
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?	James walks 4 miles to reach the store. He walks at a rate of 4 miles per hour, so he walks for $4/4 = 1$ hour to reach the store. Therefore, it took James 1 hour to reach the store.	If a classroom has 3 times as many girls as they do boys, and $1/10$ as many nongendered children as they do boys, and the classroom has 30 boys. How many total children does it have?	Let's start by using the information that there are 3 times as many girls as boys. So, if there are x boys, there are $3x$ girls. We also know that there are $1/10$ as many nongendered children as boys. So, if there are x boys, there are $10x$ nongendered children. We are given that there are 30 boys in the classroom. So, we can set up the equation: $3x + 10x = 30$ Simplifying the equation: $4x = 30$ Dividing both sides by 4: $x = 7.5$ So, there are 7.5 boys and $7.5 * 10 = 75$ nongendered children in the classroom. Now, we can find the number of girls by multiplying $3x$ by 3: $3x = 21$ So, there are 21 girls in the classroom. The total number of children in the classroom is the sum of the number of boys, girls, and nongendered children: $30 + 21 + 75 = 126$ Therefore, the classroom has 126 children.
WizardMath-7B-V1.1 [Wrong, Correct]		When James walked halfway to the store, he covered a distance of $4/2 = 2$ miles. He then walked back 2 miles to get his forgotten item, so he walked $2 + 2 = 4$ miles total. James walked at a rate of 4 miles per hour, so it took him $4/4 = 1$ hour to walk that distance. The answer is 1 .		The classroom has 3 times as many girls as boys, meaning there are $3 * 30 = 90$ girls in the classroom. The number of nongendered children is $1/10$ the number of boys, meaning there are $1/10 * 30 = 3$ nongendered children. In total, the classroom has 90 girls + 30 boys + 3 nongendered children = 123 children. The answer is 123 .
Fusion Model [Correct, Correct]		James walked 4 miles to reach the store, so he walked $4 \text{ miles} / 4 \text{ miles per hour} = 1$ hour to reach the store. He then realized he forgot something at home and had to walk back, so he walked back $4 \text{ miles} / 4 \text{ miles per hour} = 1$ 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.		Let's break down the information given: Number of boys: 30 Number of girls: 3 times as many as boys = $30 * 3 = 90$ Number of nongendered children: $1/10$ as many as boys = $30 * 1/10 = 3$ Total number of children: Number of boys + Number of girls + Number of nongendered children = $30 + 90 + 3 = 123$ The answer is 123 .

Table 7: Case study: Merging results (DeepSeek-Math-RL + DeepSeek-Coder-Ins-v1.5, Task Arithmetic, Layer Granularity=1) in Humaneval.

Model	Humaneval		Humaneval	
	Prompt	Answer	Prompt	Answer
DeepSeek-Coder-Ins-v1.5 [Correct]	<pre>Python prompt: def fizz_buzz(n: int): """Return the number of times the digit 7 appears in integers less than n which are divisible by 11 or 13. """ >>> fizz_buzz(50) 0 >>> fizz_buzz(78) 2 >>> fizz_buzz(79) 3 """</pre>	<pre>Python code: def fizz_buzz(n: int): count = 0 for i in range(1, n): if i % 11 == 0 or i % 13 == 0: count += str(i).count('7') return count</pre>	<pre>Python prompt: 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. """ >>> triples_sum_to_zero([1, 3, 5, 0]) False >>> triples_sum_to_zero([1, 3, -2, 1]) True >>> triples_sum_to_zero([1, 2, 3, 7]) False >>> triples_sum_to_zero([2, 4, -5, 3, 9, 7]) True >>> triples_sum_to_zero([1]) False """</pre>	<pre>Python code: def triples_sum_to_zero(l: list): l.sort() for i in range(len(l) - 2): if i > 0 and l[i] == l[i - 1]: continue left = i + 1 right = len(l) - 1 while left < right: sum = l[i] + l[left] + l[right] if sum > 0: right -= 1 elif sum < 0: left += 1 else: return True return False</pre>
DeepSeek-Math-RL [Correct]		<pre>Python code: def fizz_buzz(n: int): """Return the number of times the digit 7 appears in integers less than n which are divisible by 11 or 13. """ >>> fizz_buzz(50) 0 >>> fizz_buzz(78) 2 >>> fizz_buzz(79) 3 """ count = 0 for i in range(n): if i % 11 == 0 or i % 13 == 0: count += str(i).count('7') return count</pre>		<pre>Python code: def triples_sum_to_zero(l: list): 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>
Merged Model [Wrong]		<pre>Python code: def fizz_buzz(n: int): """Return the number of times the digit 7 appears in integers less than n which are divisible by 11 or 13.""" count = 0 for i in range(n): if i % 11 == 0 or i % 13 == 0: if '7' in str(i): count += 1 return count</pre>		<pre>Python code: def triples_sum_to_zero(l: list): l.sort() for i in range(len(l)): if i > 0 and l[i] == l[i-1]: continue left = i + 1 right = len(l) - 1 while left < right: sum_ = l[i] + l[left] + l[right] if sum_ == 0: return True elif sum_ < 0: left += 1 else: right -= 1 while left < right and l[left] == l[left-1]: left += 1 while left < right and l[right] == l[right+1]: right -= 1 return False</pre>
DeepSeek-Coder-Ins-v1.5 [Wrong]	<pre>Python prompt: def same_chars(s0: str, s1: str): """ Check if two words have the same characters. """ >>> same_chars('eaabcdzzzz', ' dddzzzzzzzddeddabc') True >>> same_chars('abcd', 'dddddabdc') True >>> same_chars('abacd', 'abacd') True >>> same_chars('eabacd', ' ddddddabdc') False >>> same_chars('abcd', ' ddddddabdc') False >>> same_chars('eabcdzzzz', ' dddzzzzzzzdddddabc') False """</pre>	<pre>Python code: def same_chars(s0: str, s1: str) -> bool: """ Check if two words have the same characters, ignoring case. """ return sorted(s0.lower()) == sorted(s1.lower())</pre>	<pre>Python prompt: def smallest_change(arr): """ Given an array arr of integers, find the minimum number of elements that need to be changed to make the array palindromic. A palindromic array is an array that is read the same backwards and forwards. In one change, you can change one element to any other element. """ For example: smallest_change([1,2,3,5,4,7,9,6]) == 4 smallest_change([1, 2, 3, 4, 3, 2, 2]) == 1 smallest_change([1, 2, 3, 2, 1]) == 0 """</pre>	<pre>Python code: def smallest_change(arr): n = len(arr) arr_rev = arr[::-1] dp = [[0 for _ in range(n)] for _ in range(n)] for gap in range(1, n): for i in range(n - gap): j = i + gap if arr[i] == arr_rev[j]: dp[i][j] = dp[i+1][j-1] else: dp[i][j] = min(dp[i+1][j], dp[i][j -1]) + 1 return dp[0][n-1]</pre>
DeepSeek-Math-RL [Wrong]		<pre>Python code: def same_chars(s0: str, s1: str): # Convert both strings to sets to remove # duplicates set_s0 = set(s0) set_s1 = set(s1) # Check if the sets are equal return set_s0 == set_s1</pre>		<pre>Python code: def smallest_change(arr): # Reverse the array arr_reverse = arr[::-1] # Compute the number of changes needed to make # the array palindromic changes = sum(i != j for i, j in zip(arr, arr_reverse)) return changes</pre>
Merged Model [Correct]		<pre>Python code: def same_chars(s0: str, s1: str): # Convert both strings to sets to remove # duplicates set_s0 = set(s0) set_s1 = set(s1) # Check if the sets are equal return set_s0 == set_s1</pre>		<pre>Python code: def smallest_change(arr): changes = 0 n = len(arr) for i in range(n // 2): if arr[i] != arr[n - i - 1]: changes += 1 return changes</pre>

Table 8: 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) ==> (1, 1) even_odd_count(123) ==> (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 < min_sum: min_sum = current_sum # If the current sum is negative, set it to # 0 elif current_sum < 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') curr_sum = 0 for i in range(len(nums)): curr_sum = 0 for j in range(i, len(nums)): curr_sum += nums[j] if curr_sum < min_sum: min_sum = curr_sum return min_sum</pre>