Fine-Tuning Language Models with Collaborative and Semantic Experts

Jiaxi Yang[1,2,*,†], **Binyuan Hui**[4,†], **Min Yang**[1,3,‡], **Jian Yang**[4], **Lei Zhang**[1,2], **Qiang Qu**[1], **Junyang Lin**[4,‡]

[1] Shenzhen Key Laboratory for High Performance Data Mining, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] Shenzhen University of Advanced Technology
[4] Alibaba Group
{jx.yang, min.yang}@siat.ac.cn, binyuan.hby@alibaba-inc.com

Abstract

Recent advancements in large language models (LLMs) have broadened their application scope but revealed challenges in balancing capabilities across general knowledge, coding, and mathematics. To address this, we introduce a Collaborative and Semantic Experts (CoE) approach for supervised fine-tuning (SFT), which employs a two-phase training strategy. Initially, expert training fine-tunes the feed-forward network on specialized datasets, developing distinct experts in targeted domains. Subsequently, expert leveraging synthesizes these trained experts into a structured model with semantic guidance to activate specific experts, enhancing performance and interpretability. Evaluations on comprehensive benchmarks across MMLU, HumanEval, GSM8K, MT-Bench, and AlpacaEval confirm CoE's efficacy, demonstrating improved performance and expert collaboration in diverse tasks, significantly outperforming traditional SFT methods.

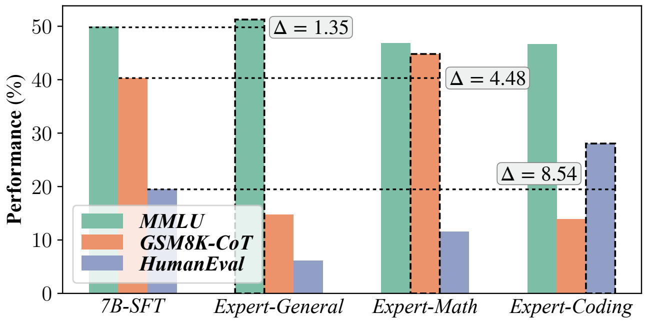


Figure 1: Comparison of performance on MMLU, GSM8K, and HumanEval tasks for different SFT setups. LLaMa2-7B-SFT means mixing all instruction data, whereas expert means training only on specialized data, e.g. Expert-Code only trains on code-related data.

Introduction

Advancements in large language models (LLMs) like GPT-4 (Achiam et al. 2023), PaLM-2 (Anil et al. 2023) and Claude (Anthropic 2023) have demonstrated impressive performance in versatile capabilities, e.g., knowledge understanding (OpenAI 2022; Ouyang et al. 2022), code generation (Roziere et al. 2023; Li et al. 2023), and mathematical reasoning (Shao et al. 2024; Azerbayev et al. 2023). These capabilities emerge from large-scale pre-training (Touvron et al. 2023; Bai et al. 2023) in conjunction with supervised fine-tuning (SFT) (Ouyang et al. 2022). Pre-training compresses data as much as possible by predicting the next token, facilitating the learning of knowledge. In contrast, SFT aligns the model using a limited but diverse set of instruction data pairs, ensuring that LLMs are helpful, honest, and harmless while activating specific capabilities.

Despite the broad range of the model's capabilities, variations in data distribution and training strategies often result in strengths in different domain (Yue et al. 2023; Wei et al. 2023a; Singhal et al. 2023). Notably, achieving a balanced performance in general knowledge, coding, and mathematics remains an unresolved challenge (Xu et al. 2023b) for open models (Touvron et al. 2023; Roziere et al. 2023).

Recent efforts have focused on balancing the diverse capabilities of open LLMs. One approach involves enhancing weaker abilities through continued pre-training (Scialom, Chakrabarty, and Muresan 2022; Azerbayev et al. 2023; Xu et al. 2023b; Sukhbaatar et al. 2024). For example, Lemur (Xu et al. 2023b) aim to balance code and general ability by carefully mixing data. Although effective, the method incurs significant data costs and typically requires collecting a large volume of tokens to achieve the desired capabilities. An alternative strategy, Branch-Train-Mix (Sukhbaatar et al. 2024) builds on continued pre-training by first developing branches skilled in multiple capabilities and then integrating them using a mixture-of-experts (MoE) (Jacobs et al. 1991; Shazeer et al. 2017; Fedus, Zoph, and Shazeer 2022) approach. Some evidence suggests (Jiang et al. 2024) that MoE models trained in the standard way do not demonstrate domain specialization. In other words, it is entirely unclear what the experts are responsible for, which does not meet human expectations for structured and interpretable models.

[*]Work done during an intern at Alibaba Group.
[†]These authors contributed equally.
[‡]Corresponding authors.

Furthermore, we argue that prior works largely overlook necessity for eliciting balancing abilities at the SFT stage. Mixing instruction data from different domains without considering specific capabilities hinders the full utilization of the model's pre-training potential. As shown in Figure 1, if the SFT data is grouped by capability, the direct mixing training approach cannot reach the level of independent training, indicating that the current SFT process indeed suffers from capability loss and conflict.

To this end, we propose a *CoE* (Collaborative and Semantic Experts) based SFT approach, consisting of two distinct phases. The first phase focuses on **expert training**, where only the feed-forward network (FFN) (Vaswani et al. 2017) is fine-tuned on specialized datasets. This phase produces experts in general knowledge, coding, and math. The second phase focuses on **expert leveraging**, where multiple experts are synthesized into a single semantically routed MoE model. In this stage, the experts are frozen, and the remaining parameters are trained. This alternating training approach allows the modules trained during the expert training and expert leveraging phases to be complementary, ensuring a seamless transition between the two phases. Besides, unlike vanilla mixture-of-expert models, *CoE* activates specific experts based on semantically guided data labeling. This makes the model more structured and interpretable, since each expert is trained and utilized for specific capabilities.

We perform comprehensive evaluations on popular benchmarks, including general knowledge on MMLU (Hendrycks et al. 2020), coding on HumanEval (Chen et al. 2021), mathematics on GSM8K (Cobbe et al. 2021) and instruction following on MT-Bench (Zheng et al. 2024) and AlpacaEval (Dubois et al. 2024),. These evaluations demonstrate that *CoE* achieves optimal performance matching each expert, confirming its efficacy. Remarkably, *CoE* exhibits superior collaborative capabilities in PoT (Gao et al. 2023) evaluations, where code experts assist in solving mathematical problems, outperforming traditional SFT methods. This synergy highlights the potential of expert collaboration in enhancing model performance across diverse tasks.

Related Work

Numerous studies have explored the methods to balance diverse capabilities in LLMs. Xu et al. (2023b) attempts to enhance both natural language and coding capabilities, aiming to create versatile language agents with balanced proficiency. Similarly, Xie et al. (2022) proposes UnifiedSKG for structured knowledge grounding, effectively addressing various diverse tasks. FLAN-T5 extends instruction fine-tuning by scaling up the number of tasks and the model size, showcasing significant performance improvements. Dong et al. (2023) study examines how the composition and volume of data during SFT affect performance, finding that total data volume is more influential than data mix ratios. Branch-Train-Mix (BTX)(Sukhbaatar et al. 2024) aims at efficiently training LLMs with multiple domain expertise, which optimizes the accuracy-efficiency trade-off by integrating expert models through MoE layers using continual pre-training strategy. Besides, model merging represents a training-free approach to integrating the capabilities of various models. For example, Yu et al. (2023) proposed DARE, which employs a technique to sparsify delta parameters of multiple homologous models that have undergone supervised fine-tuning. Our work focuses on a nuanced integration of expert knowledge through structured MoE configurations on the SFT phase, ensuring not only balanced but also contextually optimized performance across diverse applications.

Preliminaries

The standard Transformer (Vaswani et al. 2017) architecture comprises L stacked blocks, each containing a self-attention module and a feedforward network (FFN), usually enhanced with Layer Normalization (Ba, Kiros, and Hinton 2016) and residual connections (He et al. 2016), for simplicity, we omit these in the formula. In the context of a Mixture-of-Experts (MoE) (Dai et al. 2024; Fedus, Zoph, and Shazeer 2022) setup within Transformers, these standard FFN layers in each block are replaced by MoE-structured FFN layers (Dai et al. 2024; Jiang et al. 2024). This adaptation involves multiple expert networks and a dynamic routing module that determines experts activation based on input tokens. The computation for the l-th ($l \in [1, N]$) layer of a Transformer with an MoE structure featuring E prepared experts and a TopK routing, the tokens is processed as:

$$\mathbf{z}^l_{1:T} = \text{Attention}(\mathbf{h}^{l-1}_{1:T}) \tag{1}$$

where $\mathbf{h}^{l-1}_{1:T}$ is the outputs from the $(l-1)$-th layer and $\mathbf{z}^l_{1:T}$ represents the outputs from the attention module. For each token $t \in [1, T]$, the following computations are performed:

$$\mathbf{h}^l_t = \sum_{i=1}^{E} \gamma_{t,i} \cdot \text{FFN}_i(\mathbf{z}^l_t) \tag{2}$$

$$\text{logits}_t = \text{Gate}(\mathbf{z}^l_t) \tag{3}$$

$$\gamma_{t,i} = \mathbf{1}_{\text{TopK}}(i) \cdot \text{softmax}_i(\text{logits}_t) \tag{4}$$

where FFN_i denotes the i-th FFN module ($i \in [1, E]$), $\mathbf{h}^l_t$ are the final outputs for token t at that layer, $\text{Gate}(\mathbf{z}^l_t)$ is a linear module that outputs a vector of logits used to compute the routing weights $\gamma_{t,i}$ and $\mathbf{1}_{\text{TopK}}(i)$ is an indicator function as:

$$\mathbf{1}_{\text{TopK}}(i) = \begin{cases} 1 & \text{if } i \in \{\text{indices of the top } K \text{ logits}\} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The softmax function normalizes the gate logits, but only the top K experts, as determined by TopK operation, contribute to the output of the current layer for each token. This results in sparse activation, as only K out of E experts are activated for each token.

In the supervised fine-tuning process of the model, it is common to use a target-only next-token loss, denoted as $\mathcal{L}_{\text{SFT}}$, to optimize the model's performance.

Methodology

Our methodology employs a systematic approach to develop a robust Mixture-of-Experts model, starting with labeling of a large-scale Supervised Fine-Tuning (SFT) dataset into predefined groups. This step is followed by a two-stage training process, *Experts Training* and *Experts Leveraging*.

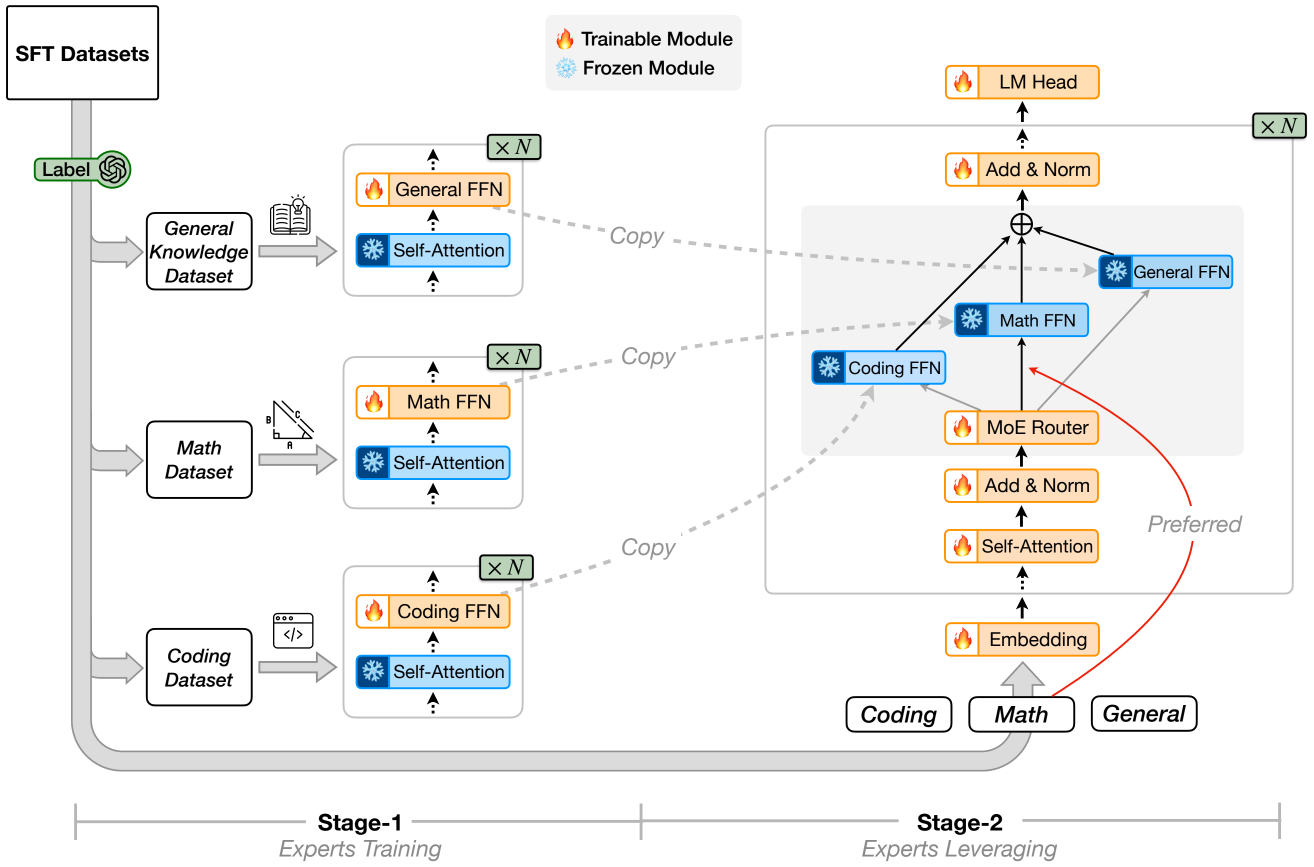


Figure 2: The two-stage training method. The first stage, Expert Training, involves fine-tuning the feedforward network (FFN) parameters for three capability groups: General, Math, and Coding. The second stage, Experts Leveraging, integrates these trained experts into a unified Mixture-of-Experts (MoE) model, utilizing a dynamic routing mechanism to optimize collaboration and performance across diverse tasks.

Instruction Data Labeling

Accurate categorization into predefined capability groups is essential to enable targeted fine-tuning of experts. Inspired by (Lu et al. 2023), the foundation of our methodology involves a process of labeling instruction data crucial for the effective training of specialized models.

We aggregated large-scale SFT data from various source and systematically labeled each example by prompting GPT-4 based on its primary category. This classification resulting in three categories: **General**, for broad knowledge instructions; **Math**, for instructions requiring mathematical reasoning; and **Coding**, for programming-related tasks.

This process led to the creation of three specialized SFT datasets: $\mathcal{D}_{\text{general}}$, $\mathcal{D}_{\text{math}}$, and $\mathcal{D}_{\text{coding}}$, each tailored to the expertise required for specific tasks. These datasets ensure that each expert is fine-tuned effectively, demonstrating high performance in their designated capability group.

Experts Training

The initial phase of our methodology focuses on expert training, where each expert is fine-tuned to enhance its specialization within specific capability groups: General, Math, and Coding.

We denote $\mathcal{M}$ as the entire set of parameters of our prepared base model and define $\mathcal{F}$ as the set of all feedforward network (FFN) (Vaswani et al. 2017) parameters, specifically, $\mathcal{F} = \{\text{FFN}_1, \cdots, \text{FFN}_N\}$. During this expert training phase, only the parameters in $\mathcal{F}$ are updated, while the rest of the parameters, $\mathcal{M} \setminus \mathcal{F}$, remain frozen.

For better trace the updated FFN parameters uniquely associated with each capability group, we denote the $\mathcal{F}_{\text{general}}$, $\mathcal{F}_{\text{math}}$ and $\mathcal{F}_{\text{coding}}$ represent the FFN parameters fine-tuned using $\mathcal{D}_{\text{general}}$, $\mathcal{D}_{\text{math}}$ and $\mathcal{D}_{\text{coding}}$, respectively.

Each set of FFN parameters undergoes SFT on its designated dataset. This ensures that the training is precisely tailored to enhance each expert's abilities within its specific capability group. This specialized training not only prepares each expert for effective collaboration in the subsequent phase but also fosters a deep and robust understanding of its respective capability group.

Experts Leveraging

Following the expert training phase, we move to the experts leveraging phase where the individually trained experts are synthesized into a unified Mixture-of-Experts (MoE)(Jiang et al. 2024; Dai et al. 2024) model, thus deriving our Collaborative and Semantic Experts architecture, i.e. the proposed *CoE* model.

In this phase, the feedforward network (FFN) parameters fine-tuned for each capability group during the previous phase are frozen. Additionally, new parameters associated with the MoE router, denoted as $\mathcal{R}$, are introduced to manage the routing of input tokens. We define the whole parameter set of *CoE* as follows:

$$\mathcal{M}_{CoE} = (\mathcal{M} \setminus \mathcal{F}) \cup \mathcal{F}_{\text{general}} \cup \mathcal{F}_{\text{math}} \cup \mathcal{F}_{\text{coding}} \cup \mathcal{R}. \tag{6}$$

The remaining parameters, $\mathcal{M} \setminus \mathcal{F} \cup \mathcal{R}$, are then fine-tuned using the union of the three SFT datasets: $\mathcal{D}_{\text{general}} \cup \mathcal{D}_{\text{math}} \cup \mathcal{D}_{\text{coding}}$. Furthermore, to not only preserve the expertise inherited from each expert but also to enhance the MoE structure's ability to harness collaborative interactions among the experts, we introduced a margin loss specifically designed for flexibility in the router's decision-making process.

Let Logits $\in \mathbb{R}^{B\times T\times E}$ represent the logits output by the router for each token, where B is the batch size, T is the sequence length, and E is the number of experts. Let $L \in \mathbb{N}^{B\times T}$ represent the indices of the semantically targeted experts, corresponding to the capability groups into which the samples have been classified. Since the label applies uniformly across all tokens in a sequence, it remains constant along the T dimension for each example. Let K be the number of top expert logits to consider.

1. First, we extract the targeted expert logits, where b indexes the batch and t indexes the position within the sequence:
$$C_{b,t} = \text{Logits}_{b,t,L_{b,t}} \tag{7}$$
2. then determine the k-th highest logits for each token at each position in the batch:
$$K_{b,t} = \text{TopK}(\text{Logits}_{b,t,:}) \tag{8}$$
3. thus we can compute the margin between the correct logits and the threshold logit:
$$M_{b,t} = C_{b,t} - K_{b,t} \tag{9}$$
4. finally, the margin routing loss can be expressed as the average over all tokens and all examples:
$$\mathcal{L}_{\text{router}} = \frac{1}{B\cdot T}\sum_{b=1}^{B}\sum_{t=1}^{T}\max(0, -M_{b,t}). \tag{10}$$

By strategically freezing and integrating these specialized FFN parameters, our *CoE* model ensures that each capability group's expertise is effectively utilized and preserved, promoting robust performance and facilitating a seamless collaboration between the diverse expert modules.

Experiments

Instruction Tuning Datasets

Our initial methodology involved utilizing a large-scale dataset derived from TULU-v2 (Wang et al. 2023; Ivison et al. 2023), a comprehensive collection of instruction tuning datasets. We extracted samples from ShareGPT (Chiang et al. 2023), WizardLM (Xu et al. 2023a), CoT (Chung et al. 2024), FLAN (Chung et al. 2024), Open-Orca (Mukherjee et al. 2023; Lian et al. 2023), GPT4-Alpaca (Peng et al. 2023), and Open Assistant 1 (Köpf et al. 2024). Each sample was labeled to categorize it into capability groups: **General**, **Coding**, or **Math**. To enhance the coding and math datasets, we incorporated additional samples from Code-Alpaca (Chaudhary 2023) and OSS-Instruct (Wei et al. 2023b) for coding, and the CoT partition from MAmmoTH (Yue et al. 2023) for math.

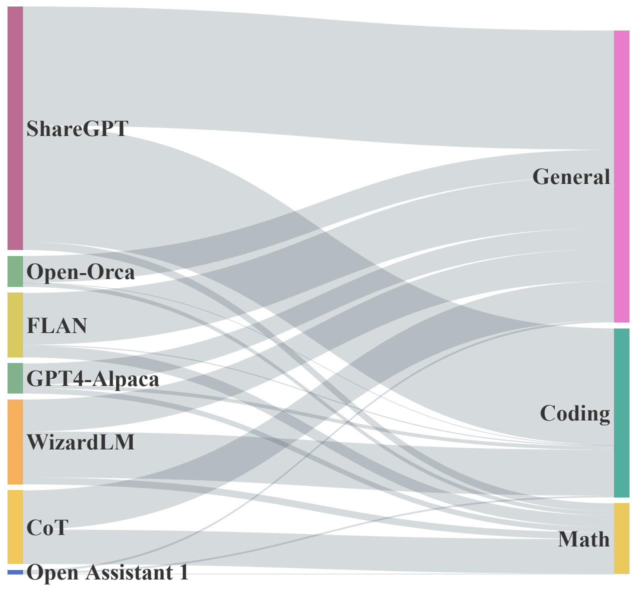


Figure 3: Sankey diagram visualizing the distribution and flow of data from various SFT datasets into the three expertise categories.

Evaluation Protocol

Expertise Evaluation Datasets

General Capability We use the Massive Multitask Language Understanding dataset, **MMLU** (Hendrycks et al. 2020), to measure model's general knowledge capabilities, which consists of a diverse array of topics.

Coding We employed the **HumanEval** (Chen et al. 2021) dataset, which consists of 164 programming problems described in natural language, along with corresponding test cases. The model should generate Python scripts that meets the requirements and passes these tests.

Math For mathematical proficiency, we used the Grade School Math (**GSM8K**)(Cobbe et al. 2021), particularly assessing problem-solving with the Chain-of-Thought (**CoT**) prompting method, which gauges step-by-step reasoning.

Experts Collaboration Evaluation We evaluated the model's ability to integrate mathematical and coding skills using the Program-of-Thought (**PoT**)(Gao et al. 2023) accompany the **GSM8K** dataset. This method tests the model's

Model	***MMLU*** *5-shot*	***GSM8K*** *CoT, 8-shot*	***HumanEval*** *Greedy, 0-shot*	***GSM8K*** *PoT, 3-shot*	***Average*** *All*	***Average*** *w/o PoT*
LLaMA2-7B-Base	45.67	14.40	12.80	17.89	22.69	24.29
LLaMA2-7B-SFT	49.91	40.33	19.51	31.69	35.36	36.58
Expert-General	**51.26**	14.71	6.10	19.79	22.97	24.02
Expert-Math	46.86	**44.81**	11.59	17.44	30.18	34.42
Expert-Coding	46.61	13.87	**28.05**	21.08	27.40	29.51
CoE-3E2A	50.47	44.73	26.22	**44.81**	**41.56**	**40.47**

Table 1: Performance of the base model (*LLaMA2-7B-Base*), the supervised fine-tuned model (*LLaMA2-7B-SFT*), individual expert models (*Expert-General*, *Expert-Math*, *Expert-Coding*), and *CoE-3E2A* across variouis datasets

Model	***MT-Bench*** *T1*	*T2*	*Overall*	***Alpaca Eval***
LLaMA2-7B-SFT	5.91	5.28	5.59	62.69
Expert-General	6.11	5.43	5.76	64.88
Expert-Math	5.81	5.04	5.43	57.87
Expert-Coding	6.15	5.53	5.84	61.64
CoE-3E2A	**6.63**	**6.09**	**6.37**	**73.01**

Table 2: Performance of *CoE-3E2A* and other models on MT-Bench and AlpacaEval.

proficiency in solving math problems and generating executable Python scripts. The scripts are executed to assess accuracy and efficiency against expected outcomes. The PoT evaluation, absent from our training data, critically assesses the model's combinatorial generalization and its capability to apply integrated skills to new tasks.

Instruction Following Evaluations We evaluated the models' ability to understand and respond to user inputs using two benchmarks. **MT-Bench**(Zheng et al. 2024) consists of 80 multi-turn questions, each involving a two-turn interaction, with GPT-4 (Achiam et al. 2023) scoring responses on a 0-10 scale to quantitatively assess conversational abilities. **AlpacaEval**(Dubois et al. 2024) measures the instruction-following ability by comparing their responses to those from *text-davinci-003*, using GPT-4 to determine a "win rate".

Implementation Details

We utilized *LLaMA2-7B-Base* (Touvron et al. 2023) for our experiments on 8 NVIDIA A100 GPUs, with training sequences limited to 2048 tokens using the ChatML formatting template(OpenAI 2022). Batch sizes were standardized at 8 per device to maintain consistency. Optimization was handled with the AdamW optimizer, starting with a learning rate warmup to 1×10^{-5}, and then adjusted down to 10% of its maximum via a cosine scheduler.

Expert Training Phase In the expert training phase, fine-tuning was concentrated on the feedforward network (FFN) components of the Transformer blocks, with other parameters frozen. The specialized models produced are termed Expert-General, Expert-Math, and Expert-Coding.

Expert Leveraging Phase During the expert leveraging phase, we integrated the trained experts using a Mixture-of-Experts (MoE) architecture. Each MoE block was equipped with a TopK routing module, initialized randomly, and set to Top-2 routing for token distribution among three experts. The synthesized model is named CoE-3E2A, with "3E" representing the three experts and "2A" indicating Top-2 expert activation.

Main Results

We conducted a comparative analysis to assess our expert training and leveraging processes. Models evaluated include the base model *LLaMA2-7B-Base*, a supervised fine-tuned model *LLaMA2-7B-SFT* across all capability groups, expert models *Expert-General*, *Expert-Math*, *Expert-Coding*, and our integrated MoE model, *CoE-3E2A*. Results are shown in Table 1 and Table 2, respectively.

Results on Expertise Evaluation Datasets Expert models, tailored to specific capability groups, consistently surpassed the SFT model, demonstrating an 11% improvement in GSM8K-CoT and a 43% in HumanEval, reinforcing the efficacy of specialized over mixed training.

Results on Collaboration Dataset Evaluation *CoE-3E2A* achieved a 44.81 execution accuracy in GSM8K-PoT evaluations, highlighting its proficiency in integrating and generalizing across different expertise areas. This performance validates the MoE architecture's capability to create a dynamic and strong model.

Results on MT-Bench and AlpacaEval In the GPT-4 based evaluations, *CoE-3E2A* notably outperforms other models, achieving a 6.37 overall score in MT-Bench and a 73.01% win rate in AlpacaEval. In MT-Bench, *CoE-3E2A* starts with a score of 6.63 and maintains a strong score of 6.09 in the second turn, demonstrating consistent contextual coherence. In AlpacaEval, it leads significantly, outperforming the SFT model by over 10%. These results highlight *CoE-3E2A*'s adeptness at responding to complex user instructions. The robust performance across these benchmarks underscores the *CoE* architecture's effectiveness in applying

Model	*MMLU* 5-shot	*GSM8K* CoT, 8-shot	*HumanEval* Greedy, 0-shot	*GSM8K* PoT, 3-shot	*Average*	*MT-Bench*	*AlpacaEval*
CoE-3E2A	**50.47**	**44.73**	26.22	44.81	**41.56**	**6.37**	**73.01**
CoE-3E2A-full	50.40	**44.73**	25.00	**44.88**	41.25	5.95	70.20
CoE-3E2A-router	49.77	40.71	**27.44**	33.43	37.84	5.94	71.51

Table 3: Performance comparison of *CoE-3E2A* with FFN-excluded fine-tuning, *CoE-3E2A-full* with all parameters updated, and *CoE-3E2A-router* with only router modules updated across various benchmarks.

expert knowledge, proving its suitability for handling complex, real-world scenarios.

Model	*#Params*	*#Trainable*	*GPU Mem*
LLaMA2-7B-Base	6.74 B	-	-
LLaMA2-7B-SFT	6.74 B	6.74 B	39.88 GB
Expert	6.74 B	4.59 B	35.90 GB
CoE-3E2A	15.40 B	2.41 B	45.01 GB
CoE-3E2A-full	15.40 B	15.40 B	56.30 GB
CoE-3E2A-router	15.40 B	0.0004 B	40.71 GB

Table 4: Computational resource usage summary for various settings, presenting the total number of parameters (***#Params***), trainable parameters (***#Trainable***), and GPU memory consumption (***GPU Mem***) for each model, illustrating the impact of different parameter update strategies on resource efficiency.

Ablations of Parameter Update Strategies

This section assesses the impact of different parameter update strategies on our MoE model's performance during the expert leveraging phase. We conducted ablation studies to determine the optimal update strategy that enhances performance while maintaining the specialized knowledge from the expert training phase. We compared our FFN-excluded SFT model, *CoE-3E2A*, with two variants:

Full Parameter SFT Updates all parameters uniformly within the MoE model, termed *CoE-3E2A-full*.

Router-Only SFT Focuses updates solely on the MoE routing modules, creating *CoE-3E2A-router*.

Results, shown in Table 3, indicate that *CoE-3E2A* generally outperforms both variants by preserving FFN knowledge while dynamically updating other components. *CoE-3E2A-full* shows a slight performance dip, suggesting that complete updates may dilute specialized capabilities. Meanwhile, *CoE-3E2A-router*, although performing similarly to *CoE-3E2A-full*, falls behind in tasks like GSM8K-PoT, highlighting challenges in fostering expert collaboration.

Table 4 outlines the computational resources for each model configuration, detailing total parameters, trainable parameters, and GPU memory consumption.[1] *LLaMA2-7B-Base* and *LLaMA2-7B-SFT* maintain 6.74 B parameters. The *Expert* models use 4.58 B trainable parameters due to selective FFN-only updates. *CoE-3E2A* balances performance and resource use with 2.41 B trainable parameters and 45.01 GB of GPU memory. In contrast, *CoE-3E2A-full* updates all parameters, while *CoE-3E2A-router*, focusing solely on the routing mechanism, uses the least resources at 40.71 GB of GPU memory. These variations illustrate the efficiency gains from targeted updates, with *CoE-3E2A* and *CoE-3E2A-router* showing how focused enhancements can optimize performance and reduce operational costs.

[1]We standardized the batch size at 8 per device across all experiments for consistency.

Ablations of Model Size

In this section, we investigate whether the *CoE-3E2A* model's performance enhancements are due to expanded model scale or the strategic integration of expert knowledge. We conduct ablation studies comparing *CoE-3E2A* with variants that increase model scale but lack fine-tuned expertise:

Replicating MoE Structure To test the impact of the MoE architecture's complexity, we developed *MoE-3E2A*, expanding parameters by using three duplicates of the *LLaMA2-7B-Base* to initialize the experts, without specialized training. This model assesses whether increases in structure and size can contribute to performance gains.

Matching Activated Parameters We derived *MoE-2E2A* model which matching *CoE-3E2A*'s activated parameters to evaluates whether merely expanding parameters can achieve the performance improvements observed.

As shown in Table 5, *CoE-3E2A* consistently outperforms both *MoE-3E2A* and *MoE-2E2A* on almost every evaluations, especially in the GSM8K-PoT and AlpacaEval, affirming the value of integrating specialized expert knowledge into the MoE architecture, which also suggests that simply increasing the number of model experts or parameters without targeted expertise does not consistently translate into better outcomes. Interestingly, the *MoE-2E2A* demonstrates a slight advantage over the *MoE-3E2A* in several benchmarks. This subtle performance discrepancy might be attributed to the more streamlined and efficient parameter usage in *MoE-2E2A*. With fewer experts to manage, *MoE-2E2A* potentially benefits from reduced complexity in its routing processes, allowing for more effective utilization of its computational resources.

Ablations of Routing Loss

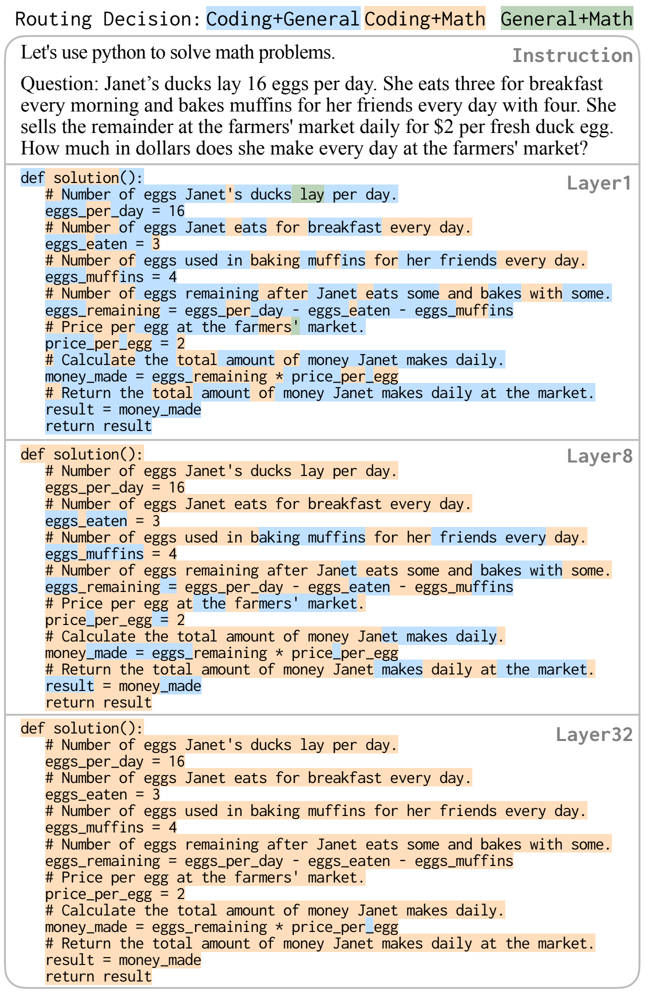
In this section, we discuss the effectiveness of our routing loss, which is designed to enhance flexibility in the router's

Model	*MMLU* 5-shot	*GSM8K* CoT, 8-shot	*HumanEval* Greedy, 0-shot	*GSM8K* PoT, 3-shot	*Average*	*MT-Bench*	*AlpacaEval*
CoE-3E2A	50.47	**44.73**	**26.22**	**44.81**	**41.56**	**6.37**	**73.01**
MoE-3E2A	50.30	41.70	25.61	32.37	37.50	6.07	67.60
MoE-2E2A	**50.76**	44.28	25.00	33.81	38.46	5.96	69.50

Table 5: Performance comparison of *CoE-3E2A* against *MoE-3E2A* and *MoE-2E2A* across several benchmarks, examining the influence of expanded model scale versus targeted expert integration.

	MMLU	*GSM8K*	*HumanEval*	*Avg*
CoE-3E2A	**50.47**	**44.73**	**26.22**	**40.47**
w/o $\mathcal{L}_{\text{router}}$	50.06	41.17	24.39	38.54

Table 6: Performance of *CoE-3E2A* with and without the routing loss.


Routing Decision: Coding+General Coding+Math General+Math
Instruction
Layer1
Layer8
Layer32

Let's use python to solve math problems.

Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

```
def solution():
    # Number of eggs Janet's ducks lay per day.
    eggs_per_day = 16
    # Number of eggs Janet eats for breakfast every day.
    eggs_eaten = 3
    # Number of eggs used in baking muffins for her friends every day.
    eggs_muffins = 4
    # Number of eggs remaining after Janet eats some and bakes with some.
    eggs_remaining = eggs_per_day - eggs_eaten - eggs_muffins
    # Price per egg at the farmers' market.
    price_per_egg = 2
    # Calculate the total amount of money Janet makes daily.
    money_made = eggs_remaining * price_per_egg
    # Return the total amount of money Janet makes daily at the market.
    result = money_made
    return result
```

```
def solution():
    # Number of eggs Janet's ducks lay per day.
    eggs_per_day = 16
    # Number of eggs Janet eats for breakfast every day.
    eggs_eaten = 3
    # Number of eggs used in baking muffins for her friends every day.
    eggs_muffins = 4
    # Number of eggs remaining after Janet eats some and bakes with some.
    eggs_remaining = eggs_per_day - eggs_eaten - eggs_muffins
    # Price per egg at the farmers' market.
    price_per_egg = 2
    # Calculate the total amount of money Janet makes daily.
    money_made = eggs_remaining * price_per_egg
    # Return the total amount of money Janet makes daily at the market.
    result = money_made
    return result
```

```
def solution():
    # Number of eggs Janet's ducks lay per day.
    eggs_per_day = 16
    # Number of eggs Janet eats for breakfast every day.
    eggs_eaten = 3
    # Number of eggs used in baking muffins for her friends every day.
    eggs_muffins = 4
    # Number of eggs remaining after Janet eats some and bakes with some.
    eggs_remaining = eggs_per_day - eggs_eaten - eggs_muffins
    # Price per egg at the farmers' market.
    price_per_egg = 2
    # Calculate the total amount of money Janet makes daily.
    money_made = eggs_remaining * price_per_egg
    # Return the total amount of money Janet makes daily at the market.
    result = money_made
    return result
```

Figure 4: Visualization of routing decisions across different layers of the *CoE-3E2A* model while processing a PoT question. Different background colors represent various expert combinations activated at each output token.

decision-making process. To validate its effectiveness, we compare our *CoE-3E2A* model against its variation trained without an additional routing loss function. The results for this compared model are presented as w/o $\mathcal{L}_{\text{router}}$.

As shown in Table 6, the configurations employing margin loss surpass those without it, indicating that a flexible approach to expert selection optimizes the use of diverse expertise. These results highlight that a well-crafted routing loss is crucial for maximizing the potential of MoE, particularly in tasks requiring collaboration among experts.

Routing Analysis

In this section, we examine the routing decisions made by the *CoE-3E2A* model across different layers when processing a GSM8K-PoT question. We visualizing the routing decisions for each output token at layer 1, 8 and 32 by giving different background color for different activated experts combinations.

As shown in Figure 4, the routing decisions demonstrate a "convergence"-style behavior across various layers. Initially, at layer 1, there is frequent "routing switching" among different expert sets, indicating early exploration of expertise for optimal input interpretation. By layer 8, this behavior stabilizes, with longer sequences of tokens consistently routed to the same experts. This trend intensifies by layer 32, where almost all tokens are directed to a single expert combination, suggesting that as the model processes deeper, it identifies and commits to the most effective expert configuration for the task, enhancing both processing efficiency and output coherence.

Conclusion

In this study, we introduced the *CoE* (Collaborative and Semantic Experts) framework, a novel approach that optimizes large language models through a two-phase supervised fine-tuning strategy, focusing on expert training and leveraging within a well-crafted Mixture-of-Experts (MoE) structure. This strategy includes efficiently performed selective parameter updates, making the two phases integrated seamlessly. Our evaluations across diverse benchmarks have demonstrated *CoE*'s effectiveness, showcasing superior problem-solving capabilities through experts collaboration. This framework not only enhances the interpretability and resource efficiency of large models but also sets a new standard for deploying sophisticated MoE model to address complex tasks with enhanced adaptability and performance.

Acknowledgments

This work was supported by National Key Research and Development Program of China (2022YFF0902100), National Natural Science Foundation of China (Grant No. 62376262), the Natural Science Foundation of Guangdong Province of China (2024A1515030166).

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anil, R.; Dai, A. M.; Firat, O.; Johnson, M.; Lepikhin, D.; Passos, A.; Shakeri, S.; Taropa, E.; Bailey, P.; Chen, Z.; et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Anthropic. 2023. Claude 2. Technical report, Anthropic.

Azerbayev, Z.; Schoelkopf, H.; Paster, K.; Santos, M. D.; McAleer, S.; Jiang, A. Q.; Deng, J.; Biderman, S.; and Welleck, S. 2023. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Chaudhary, S. 2023. Code Alpaca: An Instruction-following LLaMA model for code generation. https://github.com/sahil280114/codealpaca.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H. P. d. O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Chiang, W.-L.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J. E.; et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3): 6.

Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, Y.; Wang, X.; Dehghani, M.; Brahma, S.; et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70): 1–53.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Dai, D.; Deng, C.; Zhao, C.; Xu, R.; Gao, H.; Chen, D.; Li, J.; Zeng, W.; Yu, X.; Wu, Y.; et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Dong, G.; Yuan, H.; Lu, K.; Li, C.; Xue, M.; Liu, D.; Wang, W.; Yuan, Z.; Zhou, C.; and Zhou, J. 2023. How abilities in large language models are affected by supervised fine-tuning data composition. *arXiv preprint arXiv:2310.05492*.

Dubois, Y.; Galambosi, B.; Liang, P.; and Hashimoto, T. B. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Fedus, W.; Zoph, B.; and Shazeer, N. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120): 1–39.

Gao, L.; Madaan, A.; Zhou, S.; Alon, U.; Liu, P.; Yang, Y.; Callan, J.; and Neubig, G. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, 10764–10799. PMLR.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Ivison, H.; Wang, Y.; Pyatkin, V.; Lambert, N.; Peters, M.; Dasigi, P.; Jang, J.; Wadden, D.; Smith, N. A.; Beltagy, I.; et al. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1): 79–87.

Jiang, A. Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Hanna, E. B.; Bressand, F.; et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Köpf, A.; Kilcher, Y.; von Rütte, D.; Anagnostidis, S.; Tam, Z. R.; Stevens, K.; Barhoum, A.; Nguyen, D.; Stanley, O.; Nagyfi, R.; et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

Li, R.; Allal, L. B.; Zi, Y.; Muennighoff, N.; Kocetkov, D.; Mou, C.; Marone, M.; Akiki, C.; Li, J.; Chim, J.; et al. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Lian, W.; Goodson, B.; Pentland, E.; Cook, A.; Vong, C.; and "Teknium". 2023. OpenOrca: An Open Dataset of GPT Augmented FLAN Reasoning Traces. https://https://huggingface.co/Open-Orca/OpenOrca.

Lu, K.; Yuan, H.; Yuan, Z.; Lin, R.; Lin, J.; Tan, C.; Zhou, C.; and Zhou, J. 2023. # InsTag: Instruction Tagging for Analyzing Supervised Fine-tuning of Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Mukherjee, S.; Mitra, A.; Jawahar, G.; Agarwal, S.; Palangi, H.; and Awadallah, A. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

OpenAI. 2022. Introducing ChatGPT.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

Peng, B.; Li, C.; He, P.; Galley, M.; and Gao, J. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Roziere, B.; Gehring, J.; Gloeckle, F.; Sootla, S.; Gat, I.; Tan, X. E.; Adi, Y.; Liu, J.; Remez, T.; Rapin, J.; et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Scialom, T.; Chakrabarty, T.; and Muresan, S. 2022. Fine-tuned language models are continual learners. *arXiv preprint arXiv:2205.12393*.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Zhang, M.; Li, Y.; Wu, Y.; and Guo, D. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Singhal, K.; Azizi, S.; Tu, T.; Mahdavi, S. S.; Wei, J.; Chung, H. W.; Scales, N.; Tanwani, A.; Cole-Lewis, H.; Pfohl, S.; et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972): 172–180.

Sukhbaatar, S.; Golovneva, O.; Sharma, V.; Xu, H.; Lin, X. V.; Rozière, B.; Kahn, J.; Li, D.; Yih, W.-t.; Weston, J.; et al. 2024. Branch-Train-MiX: Mixing Expert LLMs into a Mixture-of-Experts LLM. *arXiv preprint arXiv:2403.07816*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Proc. of NeurIPS*.

Wang, Y.; Ivison, H.; Dasigi, P.; Hessel, J.; Khot, T.; Chandu, K.; Wadden, D.; MacMillan, K.; Smith, N. A.; Beltagy, I.; et al. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 36: 74764–74786.

Wei, Y.; Wang, Z.; Liu, J.; Ding, Y.; and Zhang, L. 2023a. Magicoder: Source Code Is All You Need. *arXiv preprint arXiv:2312.02120*.

Wei, Y.; Wang, Z.; Liu, J.; Ding, Y.; and Zhang, L. 2023b. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120*.

Xie, T.; Wu, C. H.; Shi, P.; Zhong, R.; Scholak, T.; Yasunaga, M.; Wu, C.-S.; Zhong, M.; Yin, P.; Wang, S. I.; et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

Xu, C.; Sun, Q.; Zheng, K.; Geng, X.; Zhao, P.; Feng, J.; Tao, C.; and Jiang, D. 2023a. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Xu, Y.; Su, H.; Xing, C.; Mi, B.; Liu, Q.; Shi, W.; Hui, B.; Zhou, F.; Liu, Y.; Xie, T.; et al. 2023b. Lemur: Harmonizing natural language and code for language agents. *arXiv preprint arXiv:2310.06830*.

Yu, L.; Yu, B.; Yu, H.; Huang, F.; and Li, Y. 2023. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Yue, X.; Qu, X.; Zhang, G.; Fu, Y.; Huang, W.; Sun, H.; Su, Y.; and Chen, W. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.