Mixing Inference-time Experts for Enhancing LLM Reasoning

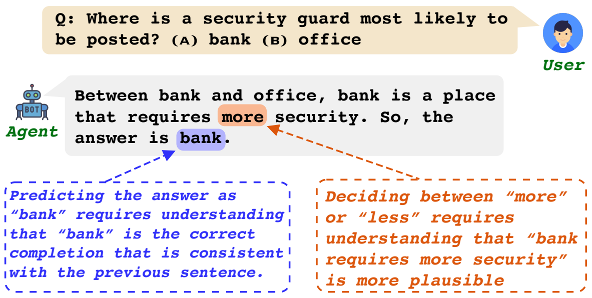
Anonymous ACL submission

Abstract

Large Language Models (LLMs) have demonstrated impressive reasoning abilities, but their generated rationales often suffer from issues such as reasoning inconsistency and factual errors, undermining their reliability. Prior work has explored improving rationale quality via multi-reward fine-tuning or reinforcement learning (RL), where models are optimized for diverse objectives. While effective, these approaches train the model in a fixed manner and do not have any inference-time adaptability, nor can they generalize reasoning requirements for new test-time inputs. Another approach is to train specialized reasoning experts using reward signals and use them to improve generation at inference time. Existing methods in this paradigm are limited to using only a single expert and cannot improve upon multiple reasoning aspects. To address this, we propose MIXIE, a novel inference-time expert-mixing framework that dynamically determines mixing proportions for each expert, enabling contextualized and flexible fusion. We demonstrate the effectiveness of MIXIE on improving chain-of-thought reasoning in LLMs by merging commonsense and entailment reasoning experts finetuned on reward-filtered data. Our approach outperforms existing baselines on three question-answering datasets: StrategyQA, CommonsenseQA, and ARC, highlighting its potential to enhance LLM reasoning with efficient, adaptable expert integration.

1 Introduction

Recent success in NLP is primarily attributed to the progress in scaling the pre-training of large language models (LLMs), enabling them to learn a variety of tasks using a few "in-context" demonstrations as model input (Brown et al., 2020; OpenAI, 2023). Recently, the chain-of-thoughts (CoTs) style of prompting techniques (Wei et al., 2022; Zhou et al., 2023; Yao et al., 2023) has made the



Q: Where is a security guard most likely to be posted? (A) bank (B) office

Between bank and office, bank is a place that requires more security. So, the answer is bank.

Predicting the answer as "bank" requires understanding that "bank" is the correct completion that is consistent with the previous sentence.

Deciding between "more" or "less" requires understanding that "bank requires more security" is more plausible

Figure 1: Motivation for token-level mixing. Predicting different tokens in the CoT rationale requires different reasoning skills. For instance, predicting some tokens need to make common sense while others might need to be consistent with the overall generation.

reasoning process more transparent, but the generated reasoning steps can sometimes be unreliable. Ye and Durrett (2022) attribute this unreliability majorly to two aspects: *factuality*, whether the explanation is correctly grounded in the input and *consistency*, whether the explanation entails the model's prediction. Improving LLMs on these aspects is crucial since using such unreliable reasoning in downstream tasks can further impact user trust and overall performance.

One approach to improve upon these reasoning aspects is to train LLMs on different reward signals using reinforcement learning techniques (Ouyang et al., 2022; Schulman et al., 2017), creating a holistic reasoning expert. However, effectively using multiple rewards corresponding to different reasoning aspects remains challenging. Another way is to train multiple experts using preference data or reward signals corresponding to different aspects, and then merging them into a unified model. Existing techniques (Lu et al., 2023; Ramnath et al., 2024) are either not capable of using multiple rewards or use the reward signals very passively. For instance, as shown in Figure 2(b), IPA can merge only two experts in equal proportions, making it a bit rigid for scenarios where mixing experts in different proportions is required. Lastly, other

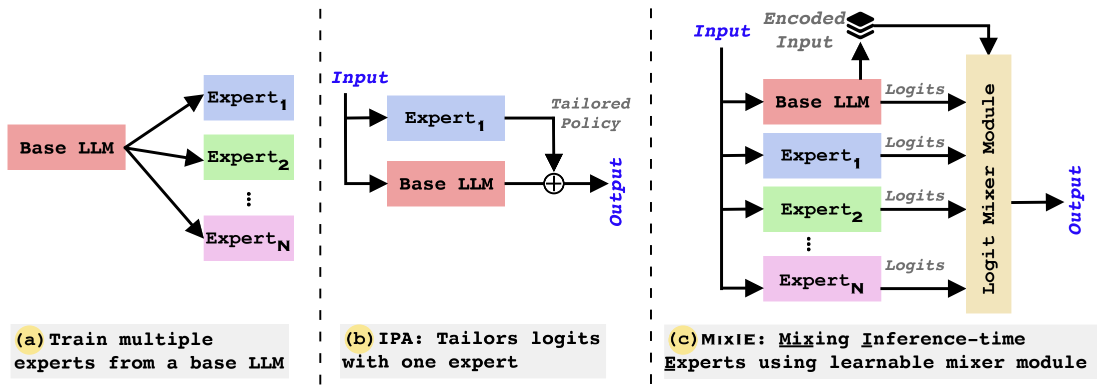


Figure 2: Overview of MIXIE and baselines. **(a)** We create multiple experts from a base LLM via finetuning on different datasets. **(b)** IPA tailors the base LLM's logits using a single expert at inference time. **(c)** In MIXIE, all the base and expert LLMs are kept frozen and only the logit mixer module is trained to mix the final layer logits in an adaptive manner. Please refer to Section 3 for more details.

mixture-of-expert (MoE) approaches (Sukhbaatar et al., 2024; Bansal et al., 2024) modify the transformer's (Vaswani et al., 2017) layer-level details, making it challenging to use out-of-the-box for a new model family.

In this work, we propose MIXIE, a simple approach for mixing multiple experts at inference time. We introduce a mixer module that dynamically decides the mixing proportions of each expert during inference, thus allowing for contextualized weighted merging of the experts. Figure 2(c) depicts an overview of our mixing approach. This approach improves over IPA as it can mix multiple experts in unequal proportions, thus significantly improving the overall flexibility. In contrast to routers in standard MoEs, our inference-time mixer module consists of limited parameters that don't require extensive training on large data, thus making the training process very efficient. Additionally, it is simple to implement for any model family, making it easy to use out of the box. Overall, we introduce a simple technique to mix multiple experts dynamically at inference time, thus addressing the limitations of prior works on merging experts (Lu et al., 2023; Sukhbaatar et al., 2024).

Next, we specifically motivate the need for token-level mixing of experts to improve the reasoning of LLMs. For example, in Figure 1, when generating a CoT rationale for the given question, some tokens ("more", in this example) need to be make common sense, some need to be consistent with overall generation ("bank", in this example), while others need to be grammatically coherent. This shows that a commonsense plausibility expert is more suited to predict some tokens, a consistency expert for other tokens, and so on. Thus, this further motivates the MIXIE framework, which enables the mixing of experts dynamically at a token level.

We demonstrate the effectiveness of MIXIE by using it to improve the reasoning process of LLMs. Specifically, inspired by the findings of Ye and Durrett (2022), we aim to improve the plausibility and consistency of CoT rationales (Wei et al., 2022) generated by LLMs. For this, we first collect high-quality CoT data using ChatGPT and filter out the implausible and inconsistent rationales using existing reward models (Liu et al., 2023; Sanyal et al., 2024). These reward-filtered datasets are then used to train expert LLMs by finetuning pretrained models. Then, we train our mixer module to mix these experts under the MIXIE framework. We find that MIXIE improves over existing baselines on three natural language question-answering datasets: StrategyQA (Geva et al., 2021), CommonsenseQA (Talmor et al., 2019), and ARC (Clark et al., 2018), indicating that mixing multiple experts using our framework is an effective way to improve reasoning abilities in LLMs.

2 Background

2.1 Problem Statement

We focus on the text generation task, which generates the output sequence y given the input sequence x. We intend to tailor the generation process to generate a sequence of high quality (e.g., generated text should be self-consistent). Different objectives can quantify such qualities of generative text. We assume that those objectives can be measured using appropriate reward functions $R_1(y), R_2(y), \ldots R_N(y)$. Our goal is to adjust the generation probability $p(y_t|y_{<t})$ so that the generated sequence improves on the various rewards across the desirable aspects.

2.2 Inference-time Policy Adapter

Inference-time Policy Adapter (IPA) (Lu et al., 2023) tailors LLMs using a policy adapter to guide a base model's generation during inference time. For a given base model B and an adapter model A, the tailored policy is as follows:

$$p(y_t|y_{<t}) = \frac{1}{Z} p_B(y_t|y_{<t}) p_A(y_t|y_{<t}), \quad (1)$$

where $p(\cdot)$ is the language modeling probability and Z is a normalizing factor. The IPA workflow is depicted in Figure 2(b). The tailored policy is typically trained to optimize a specific user-defined objective (or reward) using RL algorithms (Schulman et al., 2017; Rafailov et al., 2023; Lu et al., 2022). The key property of IPA is that only the adapter model is trainable during training, while the base model is kept frozen. This can be potentially beneficial in cases where the base model is significantly larger than the adapter.

However, IPA is limited to being able to train a single adapter. This constraint of using only one adapter for mixing is limiting in scenarios where more than one aspect/reward needs to be optimized. Additionally, IPA trains the combined policy (Eq. 1), which still requires loading both the base and adapter models while training the adapter model. This is potentially inefficient both in terms of computing and memory.

3 Overview of MIXIE

Through our proposed method MIXIE, we aim to address the various limitations of IPA that were discussed in Sections 2.2. First, we introduce the ability to use multiple reward signals in MIXIE. This is particularly important in real-world applications such as improving the chain-of-thought (CoT) reasoning, where different aspects of generation need to be tailored using specific reward signals. We achieve this by mixing multiple experts trained on different reward signals instead of using the reward signals directly for training, as done in IPA. Next, we enable a weighted mixing of experts at inference time, ensuring that experts can have dynamic weights depending on the specific context. This is crucial because if all experts contribute with the same strength at all steps, the more relevant expert might get dampened, leading to sub-optimal mixing and subsequent inferior generations. Figure 2(c) depicts an overview of our approach. In the rest of the section, we describe different aspects of our model's design and discuss the training and inference steps for building a MIXIE model.

3.1 Allowing for Multiple Reward Signals

First, we observe that if we use *trained expert* LLMs instead of adapters, we can extend the IPA formulation (Eq. 1) to optimize for different objectives/rewards. Specifically, for a base model B and N trained expert models $E_1, E_2, \dots E_N$, the tailored probability distribution can be defined as:

$$p(y_t|y_{<t}) = \frac{1}{Z} p_B(y_t|y_{<t}) \prod_{i=1}^{N} p_{E_i}(y_t|y_{<t}), \quad (2)$$

where the trained expert model E_i is optimized for a user-defined objective (or reward) using RL algorithms (Schulman et al., 2017; Rafailov et al., 2023; Lu et al., 2022). The key distinction between this and IPA is that IPA initializes an adapter with a pre-trained model. In contrast, here, the trained expert model is already optimized for a specific reward, enabling the stacking of different experts.

3.2 Need for Learnable Mixing Weights

Unlike the trainable adapter network in IPA, expert models in the Eq. 2 are fixed. This can potentially lead to subpar performance, as all the reward signals do not necessarily need to be mixed in equal proportions. To mitigate this, we introduce learnable weights in Eq. 2, which is modified as follows:

$$p(y_t|y_{<t}) = \frac{1}{Z} p_B(y_t|y_{<t})^{w_t^b} \prod_{i=1}^{N} p_{E_i}(y_t|y_{<t})^{w_t^i}, \quad (3)$$

where $p(\cdot)$ is the language modeling probability, w_t^b, w_t^i are the learnable weights for base and expert models at time step t, respectively, and Z is a normalizing factor. These weights are learned at the token level, i.e., for each token being predicted at a decoding step, MIXIE learns the optimal weights for combining the base and expert models.

Rather than directly learning these weights, we define a logit mixer module, G, that is used to predict the optimal weights at a token level. This mixer module is modeled as a two-layer feed-forward neural network (FFN), which predicts the weights based on the previous token's hidden state representation from the base model, i.e.,

$$G(h_{t-1}) = W_2(\sigma(W_1(h_{t-1}))). \quad (4)$$

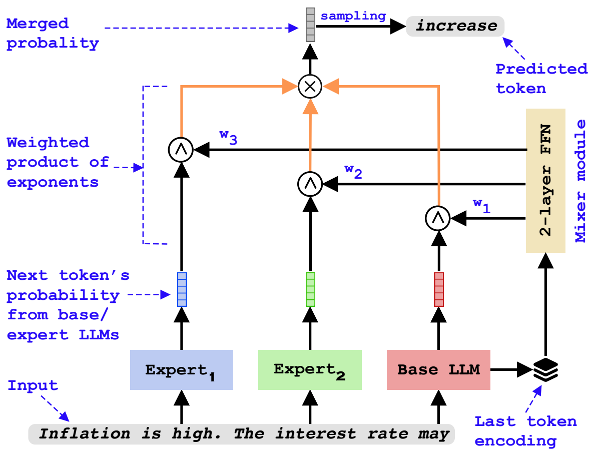


Figure 3: Overview of the logit mixer module in MIXIE. The input is passed through each of the base and expert models to generate the next token's probability. The mixer module uses the last token's representation from the base model to determine the optimal weights for each of the experts. These weights are used to merge the model outputs, according to Equation 3. This merged probability is used to sample the next token. Please refer to Section 3.2 for more details.

Here, $h_{t-1} \in \mathbb{R}^d$ is the hidden state representation of the token y_{t-1} from the base model B, $W_1 \in \mathbb{R}^{d \times d/2}, W_2 \in \mathbb{R}^{d/2 \times (N+1)}$ are learnable weights, and σ is a non-linear activation. Figure 3 depicts the overall workflow of the mixer module for an example scenario with $N = 2$ experts. It essentially models the LLMs' output probabilities according to Equation 3, but uses the learnable 2-layer FFN module to predict the mixing weights, as described in Equation 4.

3.3 Training and Inference

Overall, MIXIE uses a learnable mixer module G to mix a base model B with expert models $E_1, E_2, \dots E_N$. To train the model, we finetune MIXIE using the cross-entropy objective, as is typical for language modeling. During training, we freeze all the LLM parameters and only optimize the mixer module's parameters. More specifically,

$$G^\star = f_{LM}(\text{MIXIE}(B, E_1, E_2, \dots E_N, G)), \quad (5)$$

where f_{LM} is the language modeling objective and $G^\star$ is the trained optimal weights.

Inference follows Equation 3, same as training, i.e., we use the mixing module to mix the base and the expert models' output probabilities to determine the combined language modeling probability, which is used for sampling the generations.

4 Improving Natural Language Reasoning using MIXIE

This section aims to improve LLMs on natural language reasoning (NLR) tasks using MIXIE. We mainly focus on question-answering (QA) datasets to assess the reasoning capability of LLMs. Next, we define the reward models and the experts used for training MIXIE, along with the training procedure for the expert model and our method.

4.1 Reward Models

Ye and Durrett (2022) find that existing LLMs often generate unreliable reasoning chains where the explanation is either inconsistent with the final prediction or not plausible w.r.t. the input or commonsense. To mitigate these, we use the following reward models that are specifically trained to detect such issues in model-generated explanations:

- **Consistency** (CONST): We measure consistency between an explanation and the final prediction by using a Flan-T5-xxl-based entailment verification model (Sanyal et al., 2024), specifically trained to handle multi-sentence premises, which is common in model-generated explanations. It grades consistency on a scale of 0 to 1.
- **Plausibility** (PLAUS): We measure plausibility using VERA (Liu et al., 2023), which is a trained commonsense verification model based on T5-11B that estimates the plausibility of a declarative sentence by scoring between 0 and 1.

4.2 Training Experts

First, we generate high-quality training data from the training set of our QA dataset (described later in Section 5.1) by sampling ChatGPT[1] using chain-of-though (CoT) (Wei et al., 2022) prompting. We sample 40 CoT generations for each training instance. This dataset is referred to as the $\mathcal{D}_{\text{RAW}}$. The data creation process is depicted in Figure 4.

Next, $\mathcal{D}_{\text{RAW}}$ is filtered based on the reward score to create "reward-filtered" training data. For this, we compute the reward score for each CoT instance using the reward models described in Section 4.1 and only keep an instance if the score is above a certain threshold. In this way, we create three different reward-filtered datasets: $\mathcal{D}_{\text{CONST}}$, $\mathcal{D}_{\text{PLAUS}}$, and $\mathcal{D}_{\text{CONST+PLAUS}}$ (filter using both reward scores).

Finally, each dataset is used to finetune the base model B to create a specific expert E_i. For instance, supervised finetuning of B using $\mathcal{D}_{\text{CONST}}$

[1]ChatGPT refers to gpt-3.5-turbo-0125 by OpenAI.

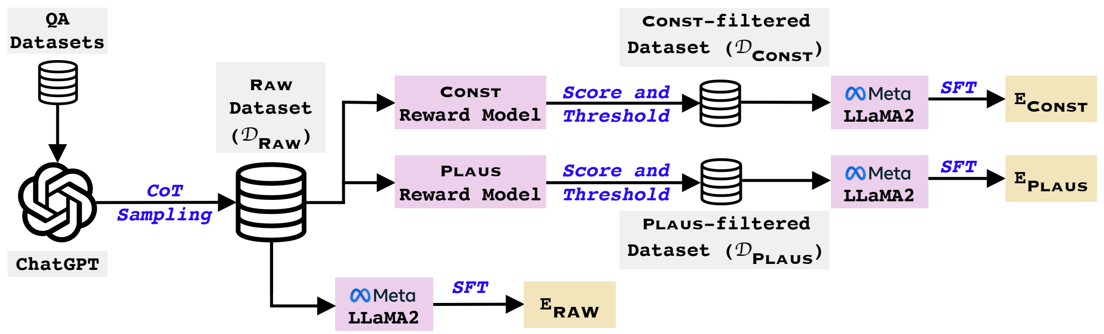


Figure 4: Workflow of training expert models. We use ChatGPT to sample CoTs from existing QA dataset, creating $\mathcal{D}_{\text{RAW}}$. This is used to finetune the E_{RAW} model. Concurrently, $\mathcal{D}_{\text{RAW}}$ is filtered using CONST (or PLAUS) reward model to create $\mathcal{D}_{\text{CONST}}$ (or $\mathcal{D}_{\text{PLAUS}}$). This is then used to finetune the base model to create the corresponding expert E_{CONST} (or E_{PLAUS}). Please refer to Section 4.2 for more details.

creates an expert E_{CONST}. Similarly, we create E_{PLAUS}, $E_{\text{CONST+PLAUS}}$, and E_{RAW} using $\mathcal{D}_{\text{PLAUS}}$, $\mathcal{D}_{\text{CONST+PLAUS}}$, and $\mathcal{D}_{\text{RAW}}$, respectively. Figure 4 depicts the workflow of creating the reward-filtered datasets and how they are used to train the experts. Please refer to Appendix B for more details on the CoT data sampling, reward computation, filtering, and training experts.

4.3 Training MIXIE

To train MIXIE, we chose the base model as E_{RAW}, which is trained using $\mathcal{D}_{\text{RAW}}$ and the expert models as E_{CONST} and E_{PLAUS}, trained using $\mathcal{D}_{\text{CONST}}$ and $\mathcal{D}_{\text{PLAUS}}$, respectively. Overall, these three models are mixed according to Equation 3. As discussed in Section 3.3, training MIXIE only involves optimizing the mixer module's parameters. We finetune the composite model using the $\mathcal{D}_{\text{CONST+PLAUS}}$.

5 Experimental Setup

Here, we discuss the QA datasets we use for our reasoning experiments, the different baselines we compare with MIXIE, and details on the evaluation setup. Please refer to Appendix F for the prompts used for each QA dataset.

5.1 Datasets

We mainly focus on three question-answering (QA) datasets: CSQA (Talmor et al., 2019), StrategyQA (Geva et al., 2021), and ARC (Clark et al., 2018). Researchers widely use these to assess the reasoning capability of LLMs.

CommonsenseQA (CSQA) (Talmor et al., 2019) This is a multiple-choice QA dataset that requires different types of commonsense knowledge to predict the correct answers.

StrategyQA (Geva et al., 2021) In this dataset, the required reasoning steps are implicit in the question and should be inferred using a strategy. Questions in StrategyQA are short, topic-diverse, and cover various strategies.

ARC (Clark et al., 2018) This is a multiple-choice QA dataset containing questions from science exams from grade 3 to grade 9. The dataset is split into two partitions: Easy and Challenge, where the latter partition includes the more difficult questions that require reasoning.

We merge all the train splits from each dataset into a combined train split, which is further processed and used in training the experts (Section 4.2) and our method (Section 4.3). Please refer to Appendix A for dataset statistics.

5.2 Baselines

We compare our MIXIE with the following baselines described below.

FLMs Foundation language models are off-the-shelf LLMs trained for a wide variety of natural language auto-regressive tasks. We compare with LLAMA2-7B and LLAMA3-8B-INSTRUCT.

SFT In this, the FLM is further finetuned on either the $\mathcal{D}_{\text{RAW}}$ or the $\mathcal{D}_{\text{CONST+PLAUS}}$ to create different SFT baselines. SFT with $\mathcal{D}_{\text{RAW}}$ evaluates the effect of distilling ChatGPT-sampled knowledge into an FLM. Likewise, SFT using $\mathcal{D}_{\text{CONST+PLAUS}}$ evaluates the effectiveness of having high-quality reward-filtered training data for distillation.

Model Merging Here, we directly combine the parameters of the expert models and the base model into a single model, giving it the combined abilities of each model without any additional training. We

Model	Task Accuracy					Reward Score		ARG
	StrategyQA	CSQA	ARC-e	ARC-c	**Avg**	CONST	PLAUS	
LLAMA2-7B	0.555	0.518	0.613	0.406	0.523	0.737	0.545	0
SFT w/								
- $\mathcal{D}_{\text{RAW}}$	0.696	0.643	0.756	0.500	0.649	0.840	0.570	14.15
- $\mathcal{D}_{\text{CONST+PLAUS}}$	0.687	0.635	0.746	0.535	0.651	0.850	0.582	15.48
TIES	0.692	0.654	0.749	0.512	0.652	0.844	0.574	14.76
DARE	0.640	0.661	0.740	0.553	0.649	0.847	0.572	14.62
IPA w/ $\mathcal{D}_{\text{CONST+PLAUS}}$	0.695	0.656	0.746	0.531	0.657	0.848	0.578	15.54
MARIO	0.592	0.641	0.646	0.468	0.587	0.811	0.561	8.34
MIXIE	**0.730**	**0.674**	**0.789**	**0.600**	**0.698**	**0.871**	**0.589**	**19.87**

Table 1: Comparisons between MIXIE and baselines using LLAMA2-7B. Under task accuracy, we report the accuracy for the test set of each QA dataset and the average accuracy. Additionally, we report the CONST and PLAUS reward scores, and the average relative gain score. We find that MIXIE outperforms all baselines consistently, demonstrating the superiority of our approach in mixing experts. Please refer to Section 6.1 for more details.

consider two prominent model-merging techniques, TIES-Merging (Yadav et al., 2023) and DARE (Yu et al., 2024), as our baselines.

IPA As introduced in Section 2.2, IPA trains an adapter to tailor the model generations to improve a specific reward. Here, we finetune the adapter model using $\mathcal{D}_{\text{CONST+PLAUS}}$ to include the effect of both the reward models into the single adapter.

MARIO (Ramnath et al., 2024) This is a controlled-generation framework that uses separate control tokens for different reward signals, training the model to produce generations that score highly on all targeted properties.

5.3 Evaluation

We evaluate the baselines and MIXIE on the test sets of the QA datasets described in Section 5.1 by generating CoT completions (Wei et al., 2022) and computing accuracy by checking if the model-generated response correctly predicts the correct answer choice. Additionally, we compute the CONST and PLAUS reward scores for the generated CoT response. Please refer to Appendix B.2 for further details on reward computations. Lastly, we compute the Average Relative Gain (**ARG**) (Ye et al., 2021) metric to quantify the average improvement across all metrics relative to a baseline. This helps quantify the overall performance of both task accuracy and reward scores under a single metric. Please refer to Appendix C for further details on how the ARG metric is computed.

6 Results

In this section, we evaluate MIXIE on the three reasoning datasets described in Section 5.1 and compare them with the baselines mentioned in Section 5.2. We also perform various ablation studies of MIXIE to evaluate the design choices.

6.1 Performance of MIXIE

In Tables 1 and 2, we evaluate MIXIE and the baselines on the three QA datasets. Specifically for the ARC dataset, we report separate numbers for the ARC-easy and ARC-challenge subsets. For each dataset, we report the accuracy metric on the test set. Additionally, we report the average accuracy (column **Acc**) across all datasets, the average CONST and PLAUS reward scores, and the average relative gain score (column **ARG**) that computes the average gains across all metrics relative to the base model, as discussed in Section 5.3.

Overall Results From Tables 1 and 2, we observe that MIXIE outperforms all the baselines in terms of the mean score. We find that our method outperforms the existing baselines on all metrics, except for LLAMA3-8B-INSTRUCT-based DARE, which beats MIXIE on CSQA dataset. Notably, there is a 7.5% improvement in accuracy of LLAMA2-7B-based MIXIE over the SFT baseline. Thus, this demonstrates the effectiveness of our model mixing strategy in improving both task accuracy and reward scores, compared to current baselines such as model merging, IPA, and MARIO.

Model	Task Accuracy					Reward Score		ARG
	StrategyQA	CSQA	ARC-e	ARC-c	**Avg**	CONST	PLAUS	
LLAMA3-8B-INSTRUCT	0.711	0.732	0.892	0.741	0.769	0.859	0.598	0
SFT w/								
- $\mathcal{D}_{\text{RAW}}$	0.730	0.755	0.889	0.738	0.778	0.892	0.610	2.37
- $\mathcal{D}_{\text{CONST+PLAUS}}$	0.734	0.737	0.882	0.734	0.772	0.893	0.612	2.21
TIES	0.713	0.729	0.892	0.726	0.765	0.891	0.607	1.63
DARE	0.692	**0.766**	0.894	0.756	0.777	0.890	0.608	2.14
IPA w/ $\mathcal{D}_{\text{CONST+PLAUS}}$	0.715	0.735	0.890	0.740	0.770	0.893	0.610	2.05
MARIO	0.739	0.740	0.897	0.729	0.776	0.897	0.607	2.32
MIXIE	**0.768**	0.760	**0.902**	**0.785**	**0.804**	**0.903**	**0.615**	**4.14**

Table 2: Comparisons between MIXIE and baselines using LLAMA3-8B-INSTRUCT. Under the task accuracy column, we report the accuracy for the test set of each QA dataset and the average accuracy. We also report the CONST and PLAUS reward scores, and the average relative gain score. Please refer to Section 6.1 for more details.

Model	Avg Acc	CONST	PLAUS	ARG
LLAMA2-7B	0.523	0.737	0.545	0
SFT w/				
- $\mathcal{D}_{\text{RAW}}$	0.649	0.840	0.570	14.15
- $\mathcal{D}_{\text{CONST}}$	0.624	**0.852**	0.572	13.24
- $\mathcal{D}_{\text{PLAUS}}$	**0.654**	0.843	0.580	15.26
- $\mathcal{D}_{\text{CONST+PLAUS}}$	0.651	0.850	**0.582**	**15.48**

Table 3: Comparisons between SFT runs using different reward-filtered datasets. We find that reward filtering helps improve the corresponding reward score. Please refer to Sections 4.2 and 6.2 for details.

Comparisons within SFT First, we observe that using $\mathcal{D}_{\text{RAW}}$ for supervised finetuning of the base model is helpful. The gains are more prominent for LLAMA2-7B since it is a weaker model and thus benefits more from the ChatGPT-sampled data. Next, we find that using reward-filtered data for SFT leads to further performance improvements, demonstrating the effectiveness of our reward-filtering strategy. Here, the main improvements are in the reward scores, which is expected since the filtering essentially creates a reward-rich dataset.

6.2 Ablation Studies

Here, we ablate the following three design choices of MIXIE: the effect of using different reward-filtered data for creating experts, the choice of expert models for mixing, and the effectiveness of the mixing module for mixing the experts.

Effect of Reward Filtering In Table 3, we evaluate the effect of using different reward-filtered data for supervised finetuning the LLAMA2-7B model. Compared to the pre-trained model, we observe that finetuning using the ChatGPT-sampled $\mathcal{D}_{\text{RAW}}$ significantly improves performance on all the metrics. Next, we find that using the training data filtered by the CONST reward model leads to improvements in the CONST reward score, as shown in bold. We see a similar trend when using $\mathcal{D}_{\text{PLAUS}}$. This demonstrates that reward filtering improves the respective reward scores, thus justifying using these models as the expert models in MIXIE. Lastly, we find that using the $\mathcal{D}_{\text{CONST+PLAUS}}$ leads to a further improvement in performance, indicated by $\sim$1.3% improvement in the average relative gain score over using $\mathcal{D}_{\text{RAW}}$.

Model	Avg Acc	CONST	PLAUS	ARG
LLAMA2-7B	0.523	0.737	0.545	0
MIXIE w/o mixer	0.675	0.865	0.581	17.68
MIXIE				
$B + E_{\text{CONST}}$	0.683	0.869	0.587	18.69
$B + E_{\text{PLAUS}}$	0.691	0.867	**0.589**	19.20
$B + E_{\text{CONST}} + E_{\text{PLAUS}}$	**0.698**	**0.871**	**0.589**	**19.87**

Table 4: Ablation of MIXIE without logit mixer module and using different expert models. Please refer to Section 6.2 for further details.

Effect of Expert Models In Table 4, we ablate the choice of the expert models used in MIXIE. As our method can mix one or more experts, we choose to test variants of our model with a single expert to test the impact of individual experts. We find that using the expert E_{PLAUS} leads to stronger accuracy results than using the E_{CONST} expert. This is likely because in Table 3, we find that using $\mathcal{D}_{\text{PLAUS}}$ for SFT creates a stronger expert with higher average accuracy than using $\mathcal{D}_{\text{CONST}}$. Additionally, we hypothesize that model-generated explanations tend to be more improbable than inconsistent, thus benefiting more from E_{PLAUS}. This is also consistent with the findings of Ye and Durrett (2022). Lastly, we find that combining both experts leads to the best performance, demonstrating the complementary nature of both experts.

Effect of Mixer Module Next, we evaluate the benefits of having a learnable mixer module in MIXIE. For this, we ablate versions of our model where the mixer module is replaced by constant mixing weights, i.e., no "learnable" weights. These results are shown in Table 4. We observe that "MIXIE w/o mixer module" underperforms compared to our method. Notably, MIXIE has an improvement of $\sim$2.2% on ARG compared to the version with fixed mixing weights. This demonstrates the effectiveness of having a learnable mixing module in our MIXIE framework.

7 Related Work

7.1 Multi Reward Training of Single Model

To exploit the potential of a single model and make it suitable for multiple fields, researchers infuse more than one reward objective into model training (Yang et al., 2024; He et al.; Shi et al., 2024). As an extension of Quark (Lu et al., 2022), Ramnath et al. (2024) propose to employ multiple reward tokens that indicate the quality of the generation in different aspects. The model learns to associate those reward tokens with the quality of the data to which it is assigned in training and yields better generations by explicitly providing reward tokens with the highest quality. Lu et al. (2023) propose Inference-time Policy Adapters (IPA), which use reinforcement learning to train a lightweight policy adapter and steer a model's output toward user-defined objectives.

7.2 Fusion of Reward-Specific Experts

Instead of adapting one model to several objectives, merging models trained with different objectives is another approach to improve performance.

Mixture of Experts Mixture of Experts (MoE) is one of the strategies to train multiple experts simultaneously. It is composed of separate neural networks, each of which is an expert in handling one subtask. With the rise of large language models (LLM), MoE has experienced a revival (Abdin et al., 2024; Sun et al., 2024; Jiang et al., 2024; Dai et al., 2024). It distributes the model's capacity in multiple specialized experts by adopting a learned gating mechanism that selectively activates only the relevant experts for each input (Fedus et al., 2022).

Branch-Train Mix Proposed by (Sukhbaatar et al., 2024), it first trains multiple domain-specific LLMs from a base LLM separately. Then, it stacks feed-forward networks on top of these dense experts to instantiate a sparse MoEs module, followed by further fine-tuning to learn token-level routing.

CALM CALM (Bansal et al., 2024), on the other hand, uses cross-attention layers to blend representations from different models and leverage their combined strengths across varied neural network structures.

Model Merging Given that expert models are essentially sets of parameters, merging these into one model is another approach. Branch-Train-Merge (Li et al., 2022) independently trains expert models tailored to a specific domain within the training corpus. These experts are then ensembled at inference time to coalesce into a single model. TIES (Yadav et al., 2023) addresses the parameter's direction conflicts and merges only the parameters that align with the final agreed-upon sign. DARE (Yu et al., 2024) sparsifies by parameter magnitude and highlights the importance of further performing rescaling on sparse models.

8 Conclusion

In this paper, we introduced MIXIE, a novel inference-time expert-mixing framework. It trains experts from one base LLM specializing in different aspects by finetuning on appropriate datasets. Then, it trains a logit mixer module to dynamically merge the logits from different expert models to yield a better token prediction. We conducted experiments on merging commonsense and entailment reasoning experts to demonstrate the effectiveness of our method in improving the chain-of-thought reasoning abilities of LLMs. Our merged model surpasses SFT, model merging, and IPA baselines on three QA datasets.

Limitations

Although our implementation of MIXIE performs quite well and significantly saves computation costs, there is still potential that has not been fully exploited in this paper, and imperfections that can be improved. Our current implementation forces all experts to be loaded into one single GPU during inference, making our method relatively memory inefficient. In addition, it also restricts the scale of our expert models and potentially compromises performance. In a more optimized implementation, expert models can generate probabilities across different GPUs, and the logit mixing module could mix all logits in the master GPU. This parallel computation can significantly reduce the inference time and also allow larger expert models to be mixed efficiently.

In contrast to traditional MoE methods, we only experiment with dense mixing, i.e., the weight of each expert model during inference is nonzero by design. However, taking inspiration from MoE setups like sparse routing (Fedus et al., 2022) can potentially benefit our mixer module. A sparse mixer module can selectively activate some experts during decoding instead of aggregating the logits from every expert. This can further make the inference more efficient. We leave this to future work.

A key assumption of MIXIE is that we have access to the logits of the LLMs during token generation. Hence, any serverless API models that do not provide logits (Brown et al., 2020; OpenAI, 2023) cannot be easily implemented within this framework. A possible way to circumvent this is to distill knowledge from these serverless models, but this does not fully address the concern.

Lastly, our framework can work with any expert model. Thus, users can potentially steer generation to malicious content by using "bad" experts built using harmful reward signals. Currently, we do not safeguard against such scenarios. Nevertheless, we highly recommend avoiding such negative applications of our framework.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, and 1 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, and 1 others. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

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

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*.

Qiang He, Yucheng Yang, Tianyi Zhou, Meng Fang, and Setareh Maghsudi. One model for all: Multi-objective controllable language models.

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

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. 2022. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*.

Jiacheng Liu, Wenya Wang, Dianzhuo Wang, Noah A. Smith, Yejin Choi, and Hanna Hajishirzi. 2023. Vera: A general-purpose plausibility estimation model for commonsense statements. *ArXiv*, abs/2305.03695.

Ximing Lu, Faeze Brahman, Peter West, Jaehun Jung, Khyathi Chandu, Abhilasha Ravichander, Prithviraj Ammanabrolu, Liwei Jiang, Sahana Ramnath, Nouha Dziri, Jillian Fisher, Bill Lin, Skyler Hallinan, Lianhui Qin, Xiang Ren, Sean Welleck, and Yejin Choi. 2023. Inference-time policy adapters (IPA): Tailoring extreme-scale LMs without fine-tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6863–6883, Singapore. Association for Computational Linguistics.

Ximing Lu, Sean Welleck, Jack Hessel, Liwei Jiang, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. 2022. QUARK: Controllable text generation with reinforced unlearning. In *Advances in Neural Information Processing Systems*.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Sahana Ramnath, Brihi Joshi, Skyler Hallinan, Ximing Lu, Liunian Harold Li, Aaron Chan, Jack Hessel, Yejin Choi, and Xiang Ren. 2024. Tailoring self-rationalizers with multi-reward distillation. In *The Twelfth International Conference on Learning Representations*.

Soumya Sanyal, Tianyi Xiao, Jiacheng Liu, Wenya Wang, and Xiang Ren. 2024. Are machines better at complex reasoning? unveiling human-machine inference gaps in entailment verification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10361–10386, Bangkok, Thailand. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Ruizhe Shi, Yifang Chen, Yushi Hu, Alisa Liu, Hanna Hajishirzi, Noah A Smith, and Simon S Du. 2024. Decoding-time language model alignment with multiple objectives. *Advances in Neural Information Processing Systems*, 37:48875–48920.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen tau Yih, Jason Weston, and Xian Li. 2024. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm. *Preprint*, arXiv:2403.07816.

Xingwu Sun, Yanfeng Chen, Yiqing Huang, Ruobing Xie, Jiaqi Zhu, Kai Zhang, Shuaipeng Li, Zhen Yang, Jonny Han, Xiaobo Shu, and 1 others. 2024. Hunyuan-large: An open-source moe model with 52 billion activated parameters by tencent. *arXiv preprint arXiv:2411.02265*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Resolving interference when merging models. *arXiv preprint arXiv:2306.01708*, 1(3).

Rui Yang, Xiaoman Pan, Feng Luo, Shuang Qiu, Han Zhong, Dong Yu, and Jianshu Chen. 2024. Rewards-in-context: Multi-objective alignment of foundation models with dynamic preference adjustment. *arXiv preprint arXiv:2402.10207*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7163–7189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning. In *Advances in Neural Information Processing Systems*.

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

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A QA Data Collection

This section introduces the process of collating the QA dataset we use to train and evaluate MIXIE. As discussed in Section 5.1, we mainly focus on CSQA, StrategyQA, and ARC datasets. We create the training dataset by integrating the train splits of CSQA and StrategyQA and the train and validation splits of the ARC dataset. By doing so, our final training split contains $15,583$ data points. Similarly, we combine the test split of ARC and StrategyQA and the validation split of CSQA into our overall test set. Through this, we collect $5,456$ data points for our test set, which is used in all evaluations. The dataset statistics of each QA dataset we consider are presented in Table 5.

Dataset Statistics	StrategyQA	CSQA	ARC-e	ARC-c
Train	1,603	9,741	2,251	1,119
Validation	-	1,221	570	299
Test	687	-	2,376	1,172

Table 5: Statistics of train/validation/test split of different QA datasets. Please refer to Appendix A for more details.

B Reward Data Collection and Expert Training

In this section, we briefly introduce the reward data collection pipeline. We sample generations from ChatGPT first, then use reward models to compute the reward scores for those generations. Finally, we filter out bad generations based on some thresholds to create the "reward-filtered" training data.

B.1 Creating $\mathcal{D}_{\text{RAW}}$ dataset using ChatGPT

As discussed in Section 4.2, we collect chain-of-thought (CoT) generations by sampling outputs from ChatGPT[2] using prompts mentioned in Appendix F. To encourage diversified generations, we set the *temperature* at 0.5 and the *top_p* at 0.8. In total, 40 rationales are sampled for every instance.

After eliciting responses for each question, we conduct a data cleaning process to de-duplicate redundant responses. This results in $297,553$ instances in the final $\mathcal{D}_{\text{RAW}}$ training data.

[2]ChatGPT refers to gpt-3.5-turbo-0125 in the OpenAI API.

B.2 Computing Reward Scores using Reward Models

The $\mathcal{D}_{\text{RAW}}$ data is fed to reward models and gets numerical scores for each instance. For PLAUS, we directly provide the generation to the VERA commonsense model (Liu et al., 2023), and the model provides a score based on the plausibility of the sentence. However, we first convert the question into a hypothesis for the consistency score. Then, the question-modified hypothesis and the generated CoT (premise) are fed to the entailment verification model (Sanyal et al., 2024) to get the consistency score in a format shown in Box 1. The PLAUS and CONST reward scores estimate the plausibility of the explanation in the generated CoT and the consistency between the CoT and the model's prediction, respectively.

Premise: {*question*}
Hypothesis: {*generation*}
Question: Given the premise, is the hypothesis correct?
Answer:

Box 1: Converted input data format for Flan-T5-xxl-based entailment verification model. Please refer to Appendix B.2 for more details.

B.3 Filtering and Creating Reward Data

After computing the reward scores, we manually set some reward thresholds to filter out bad generations and get high-quality data. For the VERA model, any generation receiving a score higher than 0.85 is regarded as a good generation, while others are treated as bad and discarded. This results in $\mathcal{D}_{\text{PLAUS}}$ dataset. Similarly, we set a threshold of 0.99 for the CONST reward scores since consistency is crucial for a good generation. Filtering based on this threshold creates $\mathcal{D}_{\text{CONST}}$. This workflow is also depicted in Figure 4. Further, we also filter using both the reward scores to create $\mathcal{D}_{\text{CONST+PLAUS}}$. For different filtering strategies, the exact number of examples in the training set after filtering is listed in Table 6.

B.4 Hyperparameter in Training Expert Model

As shown in Figure 4, we finetune the expert models using "reward-filtered" data. During finetuning, we explicitly use the parameter efficient finetuning algorithm LoRA (Hu et al., 2021). Specifically, the *load_in_4bit* is activated. The *lora_r* is set to be 16, and the *lora_alpha* is set to be 8. We also set the *lora_dropout* value to be 0.05. We set the learning rate to $1e^{-5}$, the gradient accumulation step to 4, and the per-device batch size to 3. Training an expert on a Quadro RTX 8000 GPU takes approximately 24 hours on average.

Data Statistics	Remaining	Percentage
$\mathcal{D}_{\text{RAW}}$	297,553	100%
$\mathcal{D}_{\text{CONST}}$	184,957	62%
$\mathcal{D}_{\text{PLAUS}}$	62,845	21%
$\mathcal{D}_{\text{CONST+PLAUS}}$	32,937	11%

Table 6: Data statistics of different reward-filtered datasets and their corresponding percentage share of original raw data $\mathcal{D}_{\text{RAW}}$. Please refer to Appendix B.3 for more details.

C Computing the Average Relative Gain metric

The Average Relative Gain (**ARG**) (Ye et al., 2021) metric is used to quantify the average improvement across all metrics relative to a baseline. This helps quantify the overall performance of both task accuracy and reward scores under a single metric. We use the FLMs as the baselines for this metric. The metric essentially computes how much a specific method improves on average across the three metrics: accuracy (Acc), CONST reward score (CONST), and PLAUS reward score (PLAUS). Concretely, for a method $\mathcal{M}$ based on an FLM $\mathcal{F}$, the ARG metric is defined as follows:

$$ARG = \frac{1}{3}\Bigg[\frac{Acc_{\mathcal{M}} - Acc_{\mathcal{F}}}{Acc_{\mathcal{F}}} + \frac{\text{CONST}_{\mathcal{M}} - \text{CONST}_{\mathcal{F}}}{\text{CONST}_{\mathcal{F}}} + \frac{\text{PLAUS}_{\mathcal{M}} - \text{PLAUS}_{\mathcal{F}}}{\text{PLAUS}_{\mathcal{F}}}\Bigg] * 100.$$

D Hyperparameters for Training MIXIE

To achieve better performance while training the mixer module, we choose $7e^{-5}$ as the *learning_rate* and 0.1 as the *weight_decay*. Besides, the per-device batch size is set at 8, and we use a 2-layer FFN as the router network during training.

E Effect of CoT data source

In Table 7, we evaluate the effect of using a different model as the source for sampling $\mathcal{D}_{\text{RAW}}$. Here we use gpt-4o-mini as another data source

Model	Avg Acc	CONST	PLAUS	ARG
LLAMA2-7B	0.523	0.737	0.545	0
MIXIE using $\mathcal{D}_{\text{RAW}}$ from				
- gpt-3.5-turbo-0125	0.698	0.871	0.589	19.87
- gpt-4o-mini-2024-07-18	0.696	0.840	0.588	18.26

Table 7: Ablation of the source of CoT data. Here we compare between using ChatGPT and GPT-4o-mini to sample the Chain-of-Thought data used for training MIXIE. Please refer to Appendix E for further details.

and rerun the data generation, expert training, and MIXIE training pipeline. We observe that data sampled from gpt-3.5-turbo leads to better performance than gpt-4o-mini. However, we note that MIXIE performance is relatively robust and the performance gains are consistent even with a different data source.

F Full Prompts

In Tables 8, 9, and 10 we show the CoT prompts used for creating $\mathcal{D}_{\text{RAW}}$ for CSQA, StrategyQA, and ARC datasets, respectively. We use the same CoT prompt for both easy and challenge sets of the ARC dataset.

PROMPT FOR COMMONSENSE QA

Q: What do people use to absorb extra ink from a fountain pen? (a) shirt pocket (b) calligrapher's hand (c) inkwell (d) desk drawer (e) blotter
A: The answer must be an item that can absorb ink. Of the above choices, only blotters are used to absorb ink. So the answer is (e).

Q: What home entertainment equipment requires cable? (a) radio shack (b) substation (c) television (d) cabinet
A: The answer must require cable. Of the above choices, only television requires cable. So the answer is (c).

Q: The fox walked from the city into the forest, what was it looking for? (a) pretty flowers (b) hen house (c) natural habitat (d) storybook
A: The answer must be something in the forest. Of the above choices, only natural habitat is in the forest. So the answer is (b).

Q: Sammy wanted to go to where the people were. Where might he go? (a) populated areas (b) race track (c) desert (d) apartment (e) roadblock
A: The answer must be a place with a lot of people. Of the above choices, only populated areas have a lot of people. So the answer is (a).

Q: Where do you put your grapes just before checking out? (a) mouth (b) grocery cart (c)super market (d) fruit basket (e) fruit market
A: The answer should be the place where grocery items are placed before checking out. Of the above choices, grocery cart makes the most sense for holding grocery items. So the answer is (b).

Q: Google Maps and other highway and street GPS services have replaced what? (a) united states (b) mexico (c) countryside (d) atlas
A: The answer must be something that used to do what Google Maps and GPS services do, which is to give directions. Of the above choices, only atlases are used to give directions. So the answer is (d).

Q: Before getting a divorce, what did the wife feel who was doing all the work? (a) harder (b) anguish (c) bitterness (d) tears (e) sadness
A: The answer should be the feeling of someone getting divorced who was doing all the work. Of the above choices, the closest feeling is bitterness. So the answer is (c).

Table 8: Few-shot exemplars for full chain of thought prompt for CommonsenseQA.

PROMPT FOR STRATEGY QA

Q: Do hamsters provide food for any animals? (a) true (b) false
A: Hamsters are prey animals. Prey are food for predators. Thus, hamsters provide food for some animals. So the answer is (a).

Q: Could Brooke Shields succeed at University of Pennsylvania? (a) true (b) false
A: Brooke Shields went to Princeton University. Princeton University is about as academically rigorous as the University of Pennsylvania. Thus, Brooke Shields could also succeed at the University of Pennsylvania. So the answer is (a).

Q: Hydrogen's atomic number squared exceeds number of Spice Girls? (a) true (b) false
A: Hydrogen has an atomic number of 1.1 squared is 1. There are 5 Spice Girls. Thus, Hydrogen's atomic number squared is less than 5. So the answer is (b).

Q: Is it common to see frost during some college commencements? (a) true (b) false
A: College commencement ceremonies can happen in December, May, and June. December is in the winter, so there can be frost. Thus, there could be frost at some commencements. So the answer is (a).

Q: Could a llama birth twice during War in Vietnam (1945-46)? (a) true (b) false
A: The War in Vietnam was 6 months. The gestation period for a llama is 11 months, which is more than 6 months. Thus, a llama could not give birth twice during the War in Vietnam. So the answer is (b).

Q: Would a pear sink in water? (a) true (b) false
A: The density of a pear is about \$$0.6g/cm^3$, which is less than water. Objects less dense than water float. Thus, a pear would float. So the answer is (b).

Table 9: Few-shot exemplars for full chain of thought prompt for StrategyQA.

PROMPT FOR ARC

Q: George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? (a) dry palms (b) wet palms (c) palms covered with oil (d) palms covered with lotion
A: Dry surfaces will more likely cause more friction via rubbing than other smoother surfaces, hence dry palms will produce the most heat. So the answer is (a).

Q: Which factor will most likely cause a person to develop a fever? (a) a leg muscle relaxing after exercise (b) a bacterial population in the bloodstream (c) several viral particles on the skin (d) carbohydrates being digested in the stomach
A: Bacterial population is the most likely cause for a person developing fever. So the answer is (b).

Q: Which change in the state of water particles causes the particles to become arranged in a fixed position? (a) boiling (b) melting (c) freezing (d) evaporating
A: When water is freezed, the particles are arranged in a fixed position; the particles are still moving for all other options. So the answer is (c).

Q: When a switch is used in an electrical circuit, the switch can (a) cause the charge to build (b) increase and decrease the voltage (c) cause the current to change direction (d) stop and start the flow of current
A: The function of a switch is to start and stop the flow of a current. So the answer is (d).

Table 10: Few-shot exemplars for full chain of thought prompt for ARC dataset. Both ARC-easy and ARC-challenge share the same prompt in the generation process.