# Efficient Dynamic Ensembling for Multiple LLM Experts

**Jinwu Hu**[1,2*], **Yufeng Wang**[1,2,3*], **Shuhai Zhang**[1,2*], **Kai Zhou**[1], **Guohao Chen**[1,2], **Yu Hu**[4], **Bin Xiao**[5†], **Mingkui Tan**[1,6†]
[1]South China University of Technology
[2]Pazhou Laboratory
[3]Peng Cheng Laboratory
[4]Hong Kong Polytechnic University
[5]Chongqing University of Posts and Telecommunications
[6]Key Laboratory of Big Data and Intelligent Robot, Ministry of Education
fhujinwu@gmail.com, mingkuitan@scut.edu.cn

## Abstract

LLMs have demonstrated impressive performance across various language tasks. However, the strengths of LLMs can vary due to different architectures, model sizes, areas of training data, etc. Therefore, ensemble reasoning for the strengths of different LLM experts is critical to achieving consistent and satisfactory performance on diverse inputs across a wide range of tasks. However, existing LLM ensemble methods are either computationally intensive or incapable of leveraging complementary knowledge among LLM experts for various inputs. In this paper, we propose an efficient **D**ynamic **E**nsemble **R**easoning paradigm, called **DER** to integrate the strengths of multiple LLM experts conditioned on dynamic inputs. Specifically, we model the LLM ensemble reasoning problem as a Markov Decision Process, wherein an agent sequentially takes inputs to request knowledge from an LLM candidate and passes the output to a subsequent LLM candidate. Moreover, we devise a reward function to train a DER-Agent to dynamically select an optimal answering route given the input questions, aiming to achieve the highest performance with as few computational resources as possible. Last, to fully transfer the expert knowledge from the prior LLMs, we develop a Knowledge Transfer Prompt that enables the subsequent LLM candidates to transfer complementary knowledge effectively. Experiments demonstrate that our method uses fewer computational resources to achieve better performance compared to state-of-the-art baselines. Code and appendix are available at https://github.com/Fhujinwu/DER.

## 1 Introduction

Large Language Models (LLMs) such as LLaMA [Touvron *et al.*, 2023] and GPT-3.5 [Achiam *et al.*, 2023] have exhibited remarkable performance across diverse tasks [Stark *et al.*, 2024; Hu *et al.*, 2025; Wang *et al.*, 2025], such as embodied intelligence [Mu *et al.*, 2023]. However, their variations in architectures, model sizes, and training data result in distinct strengths and weaknesses in different tasks [Jiang *et al.*, 2023; Lu *et al.*, 2024]. Although training a larger LLM with more comprehensive data is possible to maintain excellent performance in all tasks, the cost is significant and often impractical. Consequently, it is crucial to assemble LLMs to enhance their generalization while minimizing the consumption of computational resources in practical applications.

*Unfortunately*, assembling knowledge of LLMs with limited computing cost is difficult partly for the following reasons. 1) *Complex knowledge integration*: Each LLM is typically trained on different datasets, which may include varying levels of quality, diversity, and bias. Harmonizing these LLMs requires aligning their understanding with knowledge bases, posing challenges to integration [Jiang *et al.*, 2023; Lu *et al.*, 2024]. 2) *High computational complexity*: LLMs are computationally intensive, requiring significant resources for inference. Combining LLMs increases this complexity, potentially necessitating more efficient algorithms to manage [Sheng *et al.*, 2023; Wan *et al.*, 2024].

Recently, LLM ensemble has emerged as a rapidly progressing research area to leverage the diverse strengths of multiple LLMs. Based on their strategies, most existing methods can be broadly divided into four types. The *Mixture-of-Experts* methods [Jiang *et al.*, 2024; Tang *et al.*, 2024] use a router network to select and activate a subset of experts to aggregate diverse expertise (see Figure 1 (a)), but they often require retraining, cannot integrate non-homologous LLM experts, and fail to leverage complementary knowledge among experts. The *parameter merging* methods [Yu *et al.*, 2024; Matena and Raffel, 2022] merge the parameters of homologous LLMs into a single unified model but cannot assemble non-homologous LLMs. The *rule-based* methods [Dong *et al.*, 2024; Du *et al.*, 2023] assemble the advantages of LLM by manually designing task-specific roles or a fixed order. However, such a static setting makes it difficult for the integration to generalize in various domains. To avoid these issues, *agent-based* methods [Jiang *et al.*, 2023] train an agent to integrate non-homologous LLMs with the various

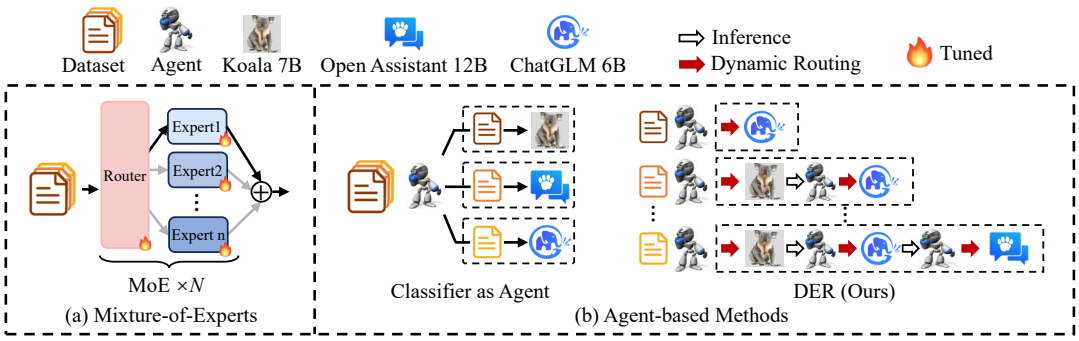


Figure 1: Illustration of different LLM ensemble strategies. **(a)** Ensemble with MoEs. **(b)** Ensemble with the agent.

strengths, making them adaptable to various scenarios.

Despite the recent success of agent-based LLM ensemble methods in outperforming the best one among the LLMs across a wide range of downstream tasks, these methods still face certain limitations. Firstly, a common drawback in most existing methods [Jiang *et al.*, 2023; Liu *et al.*, 2024] is that they integrate LLMs based on the final outputs of all LLMs. These methods require prohibitively high computational costs to run all the LLM candidates and result in inefficient utilization of inference resources. Therefore, some easy samples that could be simply addressed by a single LLM have to run all the LLMs at expensive costs. For example, PairRanker [Jiang *et al.*, 2023] requires significantly more parameters ($> 117B$) than a single LLM in inference, potentially leading to unbearable resource wastage. Secondly, while some agent-based methods implement a classifier to accomplish the integration by selecting only one LLM candidate at each time with very little inference cost (see Figure 1(b), left), *e.g.*, ZOOTER [Lu *et al.*, 2024], their performance is limited by the fact that they do not leverage the complementary knowledge among LLMs.

To address the above limitations, we propose a novel **D**ynamic **E**nsemble **R**easoning paradigm for integrating the strengths of multiple LLM experts, called **DER**. Given that LLMs are always trained on diverse datasets, we hypothesize that they possess complementary knowledge, which can be sequentially assembled. Nevertheless, the exponential growth in possible combinations of routes renders it impractical to address this challenge through classification tasks alone. To overcome this, we view knowledge transfer as a sequential decision process. Specifically, we model the LLM ensemble as a Markov Decision Process (MDP), where a DER-Agent dynamically requests contributions from LLMs and transfers this knowledge to subsequent LLM candidates (see Figure 1(b), right). Moreover, we develop a reward function to train the DER-Agent to select optimal answering routes based on input questions, aiming to maximize performance while minimizing computational resources. Additionally, we introduce a Knowledge Transfer Prompt (KTP) to facilitate effective knowledge transfer among LLMs.

We summarize our main contributions as follows:

- We propose Dynamic Ensemble Reasoning (DER) for the LLM ensemble, modeling it as a Markov Decision Process (MDP) for efficient sequential knowledge transfer. This approach dynamically selects optimal answering routes, integrating complementary knowledge from various LLMs to maximize performance with minimal computational resources. Experiments show DER integrates the strengths of different LLMs, achieving nearly a *7-fold* parameter reduction compared to ensemble methods using the outputs of all LLMs.
- We introduce a Knowledge Transfer Prompt (KTP) to facilitate efficient knowledge sharing among LLMs and develop a reward function to train the DER-Agent. This ensures the DER-Agent leverages expert knowledge from previous LLMs, optimizing task performance while significantly reducing computational costs. Experiments show that more than 9% improvement is achieved on BARTScore using KTP and our reward function.

## 2 Related Works

**Mixture-of-Experts Methods** integrate the knowledge of LLMs by selecting and activating a subset of experts through a router network. Jiang *et al.* [Jiang *et al.*, 2024] propose a Sparse Mixture of Experts language model, which aggregates the knowledge of diverse experts by selecting two experts at each layer of the routing network to process the current state and combining their outputs. Tang *et al.* [Tang *et al.*, 2024] propose the Weight-Ensemble MoE, which achieves assembling different expert knowledge by training a router to select and merge different LLM expert parameters. However, these methods often require re-fine-tuning of the MoE model, and it is difficult for experts to utilize the complementary knowledge of other experts.

**Parameter Merging Methods** assemble the advantages of LLMs by merging the parameters of multiple homologous LLMs into a single model. Matena and Raffel [Matena and Raffel, 2022] propose the "Fisher merging", which merges the parameters of models with the same structure and initialization, achieving the ability to assemble different LLMs. Yu *et al.* [Yu *et al.*, 2024] introduce an operation called DARE to sparsify delta parameters of multiple homologous LLMs, and subsequently merge them into a single model by parameter averaging, realizing the ability to assemble LLMs. However, these methods cannot assemble non-homologous LLMs.

**Rule-based Methods** assemble LLMs by manually setting roles or fixed orders for specific tasks. Du *et al.* [Du *et al.*,

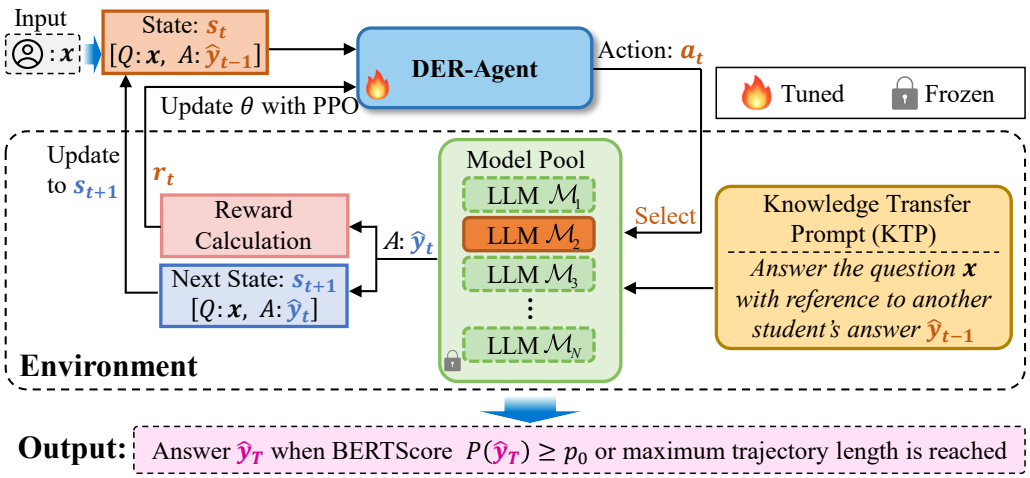


Figure 2: General diagram of DER. We formulate the LLM ensemble as an MDP and train DER-Agent to select an optimal answering route for inputs. At step $t$, the DER-Agent takes $s_t = [Q : x, A : \hat{y}_{t-1}]$ as input and selects an LLM $\mathcal{M}_{a_t}$ to continue answering the question regarding the existing answer, leading to a new answer $\hat{y}_t$. We calculate a reward $r_t$ with $\hat{y}_t$ and update the state to $s_{t+1}$. This process will loop until the answer is evaluated as satisfactory enough or the max trajectory length.

2023] propose LLM-debate, in which multiple LLMs get a consensus in multiple rounds of common debate. Dong *et al.* [Dong *et al.*, 2024] use three LLMs, set up as analysts, coders, and testers, to collaboratively develop software in sequential execution. Despite the advantages of these to quickly assemble LLMs, this static setup is hard to generalize.

**Agent-based Methods** dynamically integrate the advantages of different non-homologous LLMs by training an agent and can be used in various scenarios. Jiang *et al.* [Jiang *et al.*, 2023] propose the LLMs ensemble framework LLM-BLENDER in expectation of consistently superior performance, which selects the TOP-K responses by PairRanker and mixes them to generate final outputs using the GEN-FUSER. Liu *et al.* [Liu *et al.*, 2024] propose DyLAN, which enables the LLM ensemble through multiple rounds of interaction and an early stopping mechanism. However, they are assembled with the outputs of all candidate LLMs, which tends to waste significant computational resources. Lu *et al.* [Lu *et al.*, 2024] propose ZOOTER, a reward-guided routing method that can directly and accurately assign each prompt to LLMs with expertise, using a small computational overhead. However, it is limited in answering performance because it does not utilize complementary knowledge between LLMs.

## 3 Proposed Methods

### 3.1 Problem Definition and Motivations

**Problem Definition.** Given an input question set $\{x_1, \ldots, x_K\}$ with $K$ questions, the LLM ensemble aims to aggregate the strengths of different LLMs (experts) to consistently achieve superior performance. Specifically, given an input question $x$ and a model pool $\{\mathcal{M}_1, \ldots, \mathcal{M}_N\}$ with $N$ LLMs, we aim to select $k$ models to answer the question together to improve the quality of the answer.

**Motivations.** Existing methods [Lu *et al.*, 2024; Yao *et al.*, 2023] train an agent to select the most suitable LLM to answer each question individually (see Figure 1 (b), left). Although these methods can improve the answering performance, their performance is limited due to the underutilization of the different knowledge contained in LLMs. To overcome this issue, we aim to develop an effective knowledge aggregation strategy to achieve superior performance. Intuitively, leveraging knowledge transfer from one LLM to another can compensate for individual LLM shortcomings, fostering a collaborative integration of diverse knowledge. Moreover, there exists an optimal knowledge transfer route for each question. *Unfortunately*, the number of possible route combinations is as large as $\sum_{k=1}^{m} N^k$, where $N$ is the number of LLMs, $k$ is the route length, and $m$ is the maximum route length. In this sense, finding an optimal route for each sample poses a severe challenge.

To address the above issue, we naturally model the knowledge transfer as a sequential decision process, given its sequential nature. To this end, we require selecting certain models consecutively from the pool to answer the question, with each model able to refer to the responses provided by its predecessors (see Figure 1 (b), right). In this way, we derive an ultimate response from the terminal model. Our goal lies in identifying an optimal pathway of model execution $\text{Route}^* := [\mathcal{M}_i \Rightarrow \mathcal{M}_j \Rightarrow \cdots \Rightarrow \mathcal{M}_k]$ to enhance the quality of the final answer with modest computational cost. As shown in Figure 2, we formulate the LLM ensemble as a Markov Decision Process (MDP) and train a DER-Agent to select an optimal answering route. At each time step $t$, the DER-Agent determines the next LLM to generate a response based on the input question and the current answer (initially absent). This procedure utilizes the Knowledge Transfer Prompt (KTP), facilitating the LLM to construct an answer that progressively integrates insights from the previous LLM. The newly formed answer serves as the next state, enabling the ongoing knowledge transfer. This process continues looping until the answer is satisfactory or the maximum trajectory length is reached.

### 3.2 Dynamic Ensemble as an MDP

We seek a general DER-Agent to select an optimal sequential execution route of LLMs for the input question $x$, obtaining the highest performance with the least possible computational resources. To this end, we formulate the selection of the sequential execution route of LLMs as Markov Decision Process (MDP) [Van Otterlo and Wiering, 2012]: $< \mathcal{S}, \mathcal{A}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{R}}, \pi >$. The state space of the environment is $\mathcal{S}$ and the action space of the agent is $\mathcal{A}$. At time step $t$, the agent takes the state $s_t \in \mathcal{S}$ as input and performs an action $a_t \in \mathcal{A}$ through the policy network $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. The environment changes to the next state $s_{t+1} = \boldsymbol{\mathcal{T}}(s_t, a_t)$ according to the transition function $\boldsymbol{\mathcal{T}}$ and a reward $r_t = \boldsymbol{\mathcal{R}}(s_t, a_t)$ is received with reward function $\boldsymbol{\mathcal{R}}$. The MDP is detailed as:

**States** $\mathcal{S}$ is a set of states which describe the environment. At time step $t$, the state consists of a question-answer pair: $s_t = [Q : x, A : \hat{y}_{t-1}]$, where $x$ is the input question and $\hat{y}_{t-1}$ is the answer from the selected LLM at time step $t-1$ (the answer is $None$ initially). Thus, the agent judges the quality of the current answer by the question-answer pair and predicts which next LLM to answer.

**Actions** $\mathcal{A}$ is a discrete set of actions the agent takes. The action space $\mathcal{A} = \{1, 2, \ldots, N\}$ is the index of each LLM. At time step $t$, the agent gives the action $a_t \in \mathcal{A}$ based on the state $s_t$ to select a next LLM $\mathcal{M}_{a_t}$ from the model pool $\{\mathcal{M}_1, \ldots, \mathcal{M}_N\}$.

**Transition** $\boldsymbol{\mathcal{T}}(\mathcal{S}, \mathcal{A})$ is a function $\boldsymbol{\mathcal{T}}$: $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$ which maps a state $s_t$ into a new state $s_{t+1}$. When the BERTScore $P(\hat{y}_T)$ for the answer $\hat{y}_T$ reaches a manually-set threshold $p_0$ or the maximum trajectory length $T_{max}$ is reached, this episode will be terminated and $s_{T+1}$ is $None$. Otherwise, the selected LLM at time step $t$ will answer the question $x$ concerning the existing answer $\hat{y}_{t-1}$:

$$\begin{aligned} s_{t+1} &= [Q : x, A : \hat{y}_t], \\ \text{where } \hat{y}_t &= \mathcal{M}_{a_t}(KTP(x, \hat{y}_{t-1})). \end{aligned} \tag{1}$$

$KTP(\cdot)$ is the Knowledge Transfer Prompt (KTP) that we designed to stitch together the question and answer from the previous LLM $\mathcal{M}_{a_{t-1}}$ and to promote the current LLM to answer the question $x$ concerning the previous answer $\hat{y}_{t-1}$. The KTP is detailed in subsection 3.4.

**Rewards** $\boldsymbol{\mathcal{R}}(\mathcal{S}, \mathcal{A})$ is the reward function. In the LLM ensemble task, the reward can be considered as an evaluation of the quality of the answer $\hat{y}$ for the selected LLM $M_{a_t}$. The details of the reward function are given in the subsection 3.3.

**Policy** $\pi_\theta(a \mid s) : \mathcal{A} \times \mathcal{S} \to [0, 1]$ describes the behaviors of the agent. During the training process, the agent takes the current state $s_t$ as input and outputs a probability distribution for each possible action $a_t \in \mathcal{A} = \{1, 2, \ldots, N\}$:

$$\pi\left(a_t = i \mid s_t; \theta\right) = \frac{\exp\left\{f_\theta\left(s_t\right)_i\right\}}{\sum_{j=1}^{N} \exp\left\{f_\theta\left(s_t\right)_j\right\}}, \tag{2}$$

where $f_\theta\left(s_t\right)$ is the output vector of the policy network with input $s_t$, and $i$ denotes the index of the action. $\theta$ is the learnable parameters of the policy network.

The general diagram of the proposed DER is shown in Figure 2. Given an input question $x$, the DER is initialized with state $s_0 = [Q : x, A : None]$. The agent takes $s_0$ as input and gives an action $a_0$ so that an LLM $\mathcal{M}_{a_0}$ is selected to answer the question with $\hat{y}_0$. And then the reward $r_0$ is calculated for agent optimization based on the answer's quality and computational resources. The state is updated to $s_1 = [Q : x, A : \hat{y}_0]$ with the answer $\hat{y}_0$. The above process will continue until the BERTScore $P(\hat{y}_t)$ exceeds the threshold $p_0$ or the maximum trajectory length is reached. Finally, the last answer $\hat{y}_T$ is obtained, which is high-quality thanks to the knowledge transfer among LLMs.

### 3.3 Reward Function Design

In our designed MDP, the reward function is defined to reflect three aspects: the quality of the answer provided by the selected LLM, the increment quality of the answer, and the computational resources:

$$\boldsymbol{\mathcal{R}}_t = \begin{cases} P(\hat{y}_t) - \alpha C(M_{a_t}), & t = 0 \\ P(\hat{y}_t) + \beta \Delta P(\hat{y}) - \alpha C(M_{a_t}), & t > 0 \end{cases}, \tag{3}$$

where $P(\cdot)$ is the BERTScore, which is commonly used to evaluate the quality of generated text and its high correlation with human judgment [Zhang *et al.*, 2019]. The $\hat{y}_t$ is the output answer of the selected LLM $\mathcal{M}_{a_t}$, $C(\cdot)$ is the computation cost of $\mathcal{M}_{a_t}$, $\Delta P(\hat{y}) = P(\hat{y}_t) - P(\hat{y}_{t-1})$ is the increment of the BERTScore of the answer from $t-1$ to $t$, and $\alpha, \beta$ is the coefficient to determine the ratio of computation cost and the increment of the score, respectively. In addition, we add additional rewards or penalties to allow the agent to complete the generation of routes in limited steps. Thus, the complete reward follows:

$$\boldsymbol{\mathcal{R}}(s_t, a_t) = \begin{cases} \boldsymbol{\mathcal{R}_t} + \gamma, & t \le T_{max} \text{ and } P(\hat{y}_t) \ge p_0 \\ \boldsymbol{\mathcal{R}}_t - \gamma, & t = T_{max} \text{ and } P(\hat{y}_t) < p_0 \end{cases}, \tag{4}$$

where $p_0$ is the threshold of the BERTScore for which an environment gives an end, $T_{max}$ is the maximum step size, and $\gamma$ is the bias for extra rewards or penalties. Note that in the testing phase $P(\hat{y}_t) \ge p_0$ or $P(\hat{y}_t) < p_0$ is judged by one of our trained Terminator, which is a binary classifier.

### 3.4 Knowledge Transfer Prompt

We develop the Knowledge Transfer Prompt (KTP) to facilitate effective knowledge transfer among LLMs. The proposed KTP expects that the current LLM effectively uses the answer (knowledge) of the previous LLM $\hat{y}_{t-1}$ without being limited by it, to improve the performance of generating a better answer to the input $x$. To ensure that LLM experts follow the knowledge transfer settings, we introduce a role-playing mechanism [Kong *et al.*, 2024] into the KTP as:

**Knowledge Transfer Prompt:**

*[x] \n There is an answer to the question from another student: \n [$\hat{y}_{t-1}$] \n Using another student's answer as additional advice, you need to give a more satisfactory answer directly. DO NOT mention other students.*

First, we treat the previous LLM's answer $\hat{y}_{t-1}$ as the "student's answer", thereby avoiding the overriding influence of

| Category | Methods | Param. ↓ Agent | Param. ↓ Infer. | s/sample ↓ | Rouge-S ↑ | BARTScore ↑ | BLEURT ↑ | BERTScore ↑ | GPT-Rank↓ |
|---|---|---|---|---|---|---|---|---|---|
| | Oracle (BARTScore) | – | – | – | 0.33 | -2.87 | -0.38 | 73.23 | - |
| | ChatGPT [Achiam *et al.*, 2023] | – | ≈175B | – | 0.39 | -3.00 | -0.26 | 76.23 | - |
| LLMs | MOSS [Sun *et al.*, 2023] | – | 16B | – | 0.19 | -3.69 | -0.73 | 64.85 | - |
| | Vicuna [Chiang *et al.*, 2023] | – | 13B | – | 0.27 | -3.44 | -0.61 | 69.60 | - |
| | Alpaca [Taori *et al.*, 2023] | – | 13B | – | 0.29 | -3.57 | -0.53 | 71.46 | - |
| | Baize [Xu *et al.*, 2023] | – | 13B | – | 0.20 | -3.53 | -0.66 | 65.57 | - |
| | Open Assistant [LAION-AI, 2023] | – | 12B | – | 0.34 | -3.45 | -0.39 | 74.68 | - |
| | Dolly2 [Conover *et al.*, 2023] | – | 12B | – | 0.16 | -3.83 | -0.87 | 62.26 | - |
| | FLAN-T5 [Chung *et al.*, 2024] | – | 11B | – | 0.13 | -4.57 | -1.23 | 64.92 | - |
| | Koala [Geng *et al.*, 2023] | – | 7B | – | 0.19 | -3.85 | -0.84 | 63.96 | - |
| | Mosaic MPT [Team and others, 2023] | – | 7B | – | 0.14 | -3.72 | -0.82 | 63.21 | - |
| | StableLM [Stability-AI, 2023] | – | 7B | – | 0.17 | -4.12 | -0.98 | 62.47 | - |
| | ChatGLM [Du *et al.*, 2022] | – | 6B | – | 0.27 | -3.52 | -0.62 | 70.38 | - |
| Ensemble | Classifier-OPT | 125M | **13B** | **3.98** | 0.27 | -3.44 | -0.61 | 69.60 | 2.84 |
| | PairRanker [Jiang *et al.*, 2023] | 400M | 117B | 44.41 | 0.32 | -3.14 | -0.38 | 73.03 | 2.25 |
| | LLM-debate [Du *et al.*, 2023] | - | 234B | 56.58 | 0.27 | -3.51 | -0.55 | 71.59 | 3.16 |
| | ReConcile [Chen *et al.*, 2024] | - | 351B | 55.75 | 0.24 | -3.79 | -0.71 | 68.61 | 3.68 |
| | sampling-voting [junyou li *et al.*, 2024] | 110M | 234B | 38.07 | 0.27 | -3.39 | -0.33 | 70.12 | 3.20 |
| | DyLAN [Liu *et al.*, 2024] | - | >234B | 115.68 | 0.28 | -3.69 | -0.64 | 70.90 | 3.85 |
| | **DER (Ours)** | **125M** | 17B | 9.32 | **0.35**(+9.38%) | **-3.14**(+0.00%) | **-0.31**(+6.06%) | **75.00**(+2.70%) | **2.02** |

Table 1: Comparison with LLMs and state-of-the-art baselines on the *MixInstruct*. $(\cdot)$ indicates relative improvement over the second-best.

the answer content of the predecessor. We then ask the current LLM to refer to the "student's answer" to give a more satisfactory answer $\hat{y}_t$ to question $x$ via "*you need to give a more satisfactory answer*". In addition, the proposed KTP avoids LLM outputting role-playing messages and irrelevant information by "*DO NOT mention other students*".

### 3.5 Learning with Proximal Policy Optimization

We use the Proximal Policy Optimization (PPO) [Schulman *et al.*, 2017] to optimize the parameters $\theta$ of the DER-Agent (policy) due to the stability and sample-efficiency as:

**Actor.** The actor (DER-Agent) is trained for LLM selection according to the question-answer pair. To enhance the ability of natural language understanding, we employ a pre-trained OPT-125M [Zhang *et al.*, 2022] with two Linear layers connected to the last hidden layer, where the output dim is the number of Candidate LLMs $N$. In the off-policy algorithm, the old policy $\pi_{\theta_{old}}$ with old parameters $\theta_{old}$ is used to collect trajectories with the environment, while the policy $\pi_\theta$ is updated using trajectories collected by $\pi_{\theta_{old}}$.

**Critic.** The critic $V_\phi(s)$ is used to estimate the expected return $v_t$ of the state $s_t$ and calculate the advantage, which aids the actor in learning more efficiently and stably. The critic is composed of an OPT-125M with two Linear layers. But the output dim is set to one. Besides, the old critic $V_{\phi_{old}}(s)$ is used to collect trajectories, and the new critic $V_\phi(s)$ is updated using the collected trajectories.

**Learning Objectives.** The goal of the learning is to maximize the expected long-term return $\mathcal{J}(\theta)$:

$$\begin{aligned}\mathcal{J}(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[G(\tau)] \\ &= \mathbb{E}_{\tau\sim\pi_{\theta_{old}}(\tau)}[\min(\rho A^{\pi_{\theta_{old}}}(s_t,a_t), \\ &\quad \text{clip}\,(\rho, 1-\epsilon, 1+\epsilon)\, A^{\pi_{\theta_{old}}}(s_t,a_t))], \end{aligned} \tag{5}$$

**Algorithm 1** PPO Training for DER

**Require:** Prompt-Answer dataset $\mathcal{D}$, DER-Agent $\pi_{\boldsymbol{\theta}}$, critic $V_\phi$, replay buffer $\boldsymbol{\mathcal{B}}$, buffer size $M$, training iterations $m$, actor $\theta$ and critic $\phi$.

```
Initialize B, parameters θ and φ.
while Not converged do
  for (question x_i, answer y_i) in D do
    Collect a trajectory τ using old π_{θ_old} and V_{φ_old}, and put it into B.
    if |B| = M then
      for iteration = 1, 2, ..., M do
        Uniformly sample τ ∈ B.
        Calculate J(θ) via Eqn. (5).
        Update θ to maximize J(θ).
        Calculate TD error δ_t via Eqn. (6).
        Update φ to minimize TD error δ_t.
      end for
      Empty the replay buffer B.
      Update θ_old ← θ.
      Update φ_old ← φ.
    end if
  end for
end while
```

where $G(\tau)$ is the total return of the trajectory $\tau = \{s_t, a_t, r_t, v_t, A^{\pi_{\theta_{old}}}(s_t,a_t)\}$ obtained by $\pi_{\theta_{old}}$ and $V_{\phi_{old}}$, $\rho = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the ratio of the probability of action $a_t$ given by $\pi_\theta$ and $\pi_{\theta_{old}}$ for state $s_t$, and $\epsilon$ is a hyperparameter, usually set to 0.2 [Schulman *et al.*, 2017]. The operation $\text{clip}\,(\rho, 1-\epsilon, 1+\epsilon)$ constrains $\rho$ to the range $[1-\epsilon, 1+\epsilon]$, and $A^{\pi_{\theta_{old}}}(s_t,a_t) = r_t - V_{\phi_{old}}(s_t)$ is the advantage at $t$.

**Training.** The overview of the optimization process is presented in Algorithm 1. Specifically, given a Prompt-Answer

| Category | Methods | Param. ↓ Agent | Param. ↓ Infer. | Acc (%) ↑ | Rouge-S ↑ | BARTScore ↑ | BLEURT ↑ | BERTScore ↑ | GPT-Rank↓ |
|---|---|---|---|---|---|---|---|---|---|
| LLMs | MOSS [Sun *et al.*, 2023] | – | 16B | 19.41 | 0.29 | -5.27 | -0.58 | 70.14 | - |
| | Vicuna [Chiang *et al.*, 2023] | – | 13B | 33.74 | 0.36 | -4.51 | -0.43 | 74.68 | - |
| | Alpaca [Taori *et al.*, 2023] | – | 13B | 8.11 | 0.17 | -5.86 | -0.95 | 63.07 | - |
| | Baize [Xu *et al.*, 2023] | – | 13B | 31.08 | 0.30 | -5.11 | -0.59 | 69.97 | - |
| | Open Assistant [LAION-AI, 2023] | – | 12B | 16.60 | 0.31 | -5.26 | -0.58 | 70.64 | - |
| | Dolly2 [Conover *et al.*, 2023] | – | 12B | 11.52 | 0.23 | -5.66 | -0.88 | 67.27 | - |
| | FLAN-T5 [Chung *et al.*, 2024] | – | 11B | 24.03 | 0.37 | -4.94 | -0.42 | 74.68 | - |
| | Koala [Geng *et al.*, 2023] | – | 7B | 20.70 | 0.25 | -4.94 | -0.95 | 67.98 | - |
| | Mosaic MPT [Team and others, 2023] | – | 7B | 23.43 | 0.31 | -4.97 | -0.53 | 70.77 | - |
| | StableLM [Stability-AI, 2023] | – | 7B | 19.56 | 0.32 | -4.70 | -0.39 | 74.76 | - |
| | ChatGLM [Du *et al.*, 2022] | – | 6B | 21.08 | 0.33 | -4.90 | -0.57 | 72.64 | - |
| Ensemble | PairRanker [Jiang *et al.*, 2023] | 400M | 117B | 30.17 | 0.36 | -4.74 | **-0.39** | 74.26 | 1.85 |
| | LLM-debate [Du *et al.*, 2023] | - | 234B | 19.71 | 0.27 | -5.35 | -0.74 | 68.39 | 2.43 |
| | sampling-voting [junyou li *et al.*, 2024] | 110M | 234B | 26.08 | 0.36 | -4.91 | -0.34 | 73.21 | 1.68 |
| | DyLAN [Liu *et al.*, 2024] | - | >234B | 19.64 | 0.25 | -5.41 | -0.77 | 68.02 | 2.77 |
| | **DER (Ours)** | **125M** | **26B** | **34.98**(+15.94%) | **0.37** | **-4.44** | -0.41 | **75.14** | **1.27** |

Table 2: Comparison with LLMs and state-of-the-art baselines on the *GSM8K*.

dataset $\mathcal{D}$, we use $\pi_{\theta_{old}}$ and $V_{\phi_{old}}$ to interact with the environment to collect a trajectory $\tau$ and compute the advantage $A^{\pi_{\theta_{old}}}(s_t, a_t)$. We then put $\tau$ into the reply buffer $\boldsymbol{\mathcal{B}}$. When a certain number of trajectories (such as $M$) have been collected, they are used to train the actor and critic. In particular, we first uniformly sample sequences from the reply buffer $\boldsymbol{\mathcal{B}}$, and then calculate the expected long-term return $\mathcal{J}(\theta)$ so that to optimize the parameters of the policy $\pi_\theta$. Besides, the Temporal Difference (TD) error $\delta_t$ is also calculated to optimize the parameters of the critic $V_\phi$:

$$\delta_t = G_t - V_\phi(s_t), \tag{6}$$

where $G_t$ is the total expected return starting from time step $t$. After training a certain number of times using the samples in the existing reply buffer $\boldsymbol{\mathcal{B}}$, we clear the reply buffer and update the parameters of the old policy $\pi_{\theta_{old}}$ and critic $V_{\phi_{old}}$. Then we repeat the above operation until convergence.

## 4 Experiment

**Datasets and LLM experts.** Following the settings of PairRanker [Jiang *et al.*, 2023], we use *MixInstruct* as the benchmark. In addition, we use the *GSM8K* [Cobbe *et al.*, 2021] and *Multidomain* we constructed (see Appendix 2.3) for further evaluation. We select eleven LLM experts for the ensemble task: Open Assistant [LAION-AI, 2023], Vicuna [Chiang *et al.*, 2023], Alpaca [Taori *et al.*, 2023], Baize [Xu *et al.*, 2023], MOSS [Sun *et al.*, 2023], ChatGLM [Du *et al.*, 2022], Koala [Geng *et al.*, 2023], Dolly2 [Conover *et al.*, 2023], Mosaic MPT [Team and others, 2023], StableLM [Stability-AI, 2023] and FLAN-T5 [Chung *et al.*, 2024].

**Baseline Methods.** 1) Classifier-OPT (Appendix 2.5), 2) PairRanker [Jiang *et al.*, 2023], 3) LLM-debate [Du *et al.*, 2023], 4) ReConcile [Chen *et al.*, 2024], 5) sampling-voting [junyou li *et al.*, 2024], and 6) DyLAN [Liu *et al.*, 2024].

### 4.1 Comparison Experiments

We compare our proposed DER, eleven LLM experts, ChatGPT, and state-of-the-art (SOTA) LLM ensemble methods to demonstrate our method superior performance. We conduct experiments on a variety of downstream tasks, including QA task (see Table 1), mathematical reasoning task (see Table 2), and multi-domain QA task (see Appendix 3.2).

| Version | BARTScore ↑ | BLEURT ↑ | BERTScore ↑ |
|---|---|---|---|
| Random (w/o KTP) | -3.48 | -0.45 | 72.69 |
| Random | -3.42 | -0.45 | 72.88 |
| Ours (w/o KTP) | -3.29 | -0.40 | 74.30 |
| **Ours** | **-3.14** | **-0.31** | **75.00** |

Table 3: Experimental results on the effect of KTP.

**DER is consistently better than single LLM.** From Table 1, DER achieves better performance than a single LLM. Crucially, the DER on BARTScore is improved by 8.7% compared to Vicuna. In addition, DER reaches 98% of the ChatGPT performance on the BERTScore, while the inference parameters are only 10% of those of ChatGPT. We conclude that DER achieves better performance than the single LLM due to its ability to aggregate the complementary knowledge of diverse LLMs through knowledge transfer.

**Trade-offs between performance and computational resources.** As shown in Table 1, the proposed DER achieves better performance than the SOTA methods with a little cost. Specifically, the proposed DER reduces the computational overhead of LLMs inference by about 85% (117B $\rightarrow$ 17B) compared to PairRanker, while DER increases the BERTScore by about 2.7% (73.03 $\rightarrow$ 75.00) compared to PairRanker. In addition, our method is also the best performer on the GPT-Rank metric. The main reason is that the reward function of our design requires DER-Agent to aggregate the strengths of diverse LLMs on as few resources as possible while setting the maximum route length.

**Superior performance on mathematical reasoning.** From Table 2, DER outperforms SOTA methods on the mathematical reasoning task. Specifically, compared to PairRanker, DER reduces the inference cost by more than 77%

| $\alpha$ | $\beta$ | $\gamma$ | BARTScore ↑ | BLEURT ↑ | BERTScore ↑ |
|---|---|---|---|---|---|
| ✓ | ✓ | | -3.32 | -0.34 | 74.68 |
| ✓ | | ✓ | -3.22 | -0.34 | 74.52 |
| ✓ | ✓ | ✓ | **-3.14** | **-0.31** | **75.00** |

Table 4: Experimental results for the component of the reward function on *MixInstruct*. Notably, $\alpha = 0.001$.

| Version | Param. ↓ | BARTScore ↑ | BLEURT ↑ | BERTScore ↑ |
|---|---|---|---|---|
| $T_{max} = 3$ | **15B** | -3.15 | -0.32 | 74.97 |
| $T_{max} = 4$ | 17B | **-3.14** | **-0.31** | **75.00** |
| $T_{max}$ (w/o Term.) | 28B | -3.15 | -0.34 | 73.32 |
| $T_{max} = 5$ | 20B | -3.16 | -0.32 | 74.94 |

Table 5: Experimental results with different reachable maximum step ($T_{\text{max}}$) and without Terminator (w/o Term.).

and increases the Accuracy by about 16%. We conclude that DER effectively aggregates the strengths of LLMs to generate better mathematical reasoning results after the knowledge transmission through them.

### 4.2 Ablation Studies

**Effectiveness of Knowledge Transfer Prompt.** We compare DER and DER (w/o KTP) to demonstrate the effectiveness of the KTP. From Table 3, the KTP effectively improves the performance of DER. Specifically, there is a 4.6% increase in BARTScore after using the KTP. This is strong support for the fact that the use of a role-playing prompt allows the LLM to leverage the knowledge of the previous LLM to produce a more satisfactory output [Kong *et al.*, 2024].

**Effectiveness of LLM ensemble using MDP.** As shown in Table 3, compared with the randomly generated route method, the DER outperforms the randomly generated route method by 2.31 in BERTScore. The primary reason is that by modeling the LLM ensemble as an MDP, the trained DER-Agent chooses an appropriate LLM based on the answer of the previous LLM to continue the answering.

**Effects of the reward function $\mathcal{R}$.** We study the effects of $\beta\Delta P(\hat{y})$ and $\gamma$ in reward function on the performance of DER, by conditioning $\beta = 0$ or $\gamma = 0$ in Equation (3) and (4). As shown in Table 4, whenever $\beta = 0$ or $\gamma = 0$ both lead to about 3% performance degradation of DER on the BARTScore metric. This provides strong support for adding additional rewards/ penalties to our reward function (see Section 3.3) for answering the performance increase and whether or not completing the task within a finite step size improves DER's performance on the LLM ensemble task.

**Effects of the Terminator.** We study the effect of Terminator on DER at the maximum reachable step $T_{max} = 4$. As shown in Table 5, DER with Terminator reduces the parameters by 39% while also improving the BERTScore by 1.68. This is because the route reaches the endpoint when the Terminator believes that the output answer is greater than or equal to the threshold $p_0 = 0.73$ on the BERTScore metric (selection of $p_0$ see Appendix 4.2). Therefore, using Terminator reduces the average computational resources required for inference of LLMs and results in better performance.

| Length | $T = 1$ | $T = 2$ | $T = 3$ | $T = 4$ |
|---|---|---|---|---|
| Percentage | 38.1% | 17.6% | 6.1% | 38.2% |

Table 6: Statistics of answer route length generated by DER on *MixInstruct* testset for all samples.

| Version | Agent | Infer. | BARTScore ↑ | BLEURT ↑ | BERTScore ↑ |
|---|---|---|---|---|---|
| Two experts | 125M | 49B | -3.23 | -0.33 | 74.80 |
| One expert | 125M | **17B** | **-3.14** | **-0.31** | **75.00** |

Table 7: Experiments to the number of experts per state.

**Effects of the maximum reachable step $T_{max}$.** As shown in Table 5, as the maximum reachable step increases, the average parameter of the LLMs inference also increases, but it does not make a significant difference to the performance. Therefore, we set the $T_{max}$ to 4. In addition, we count the length of answer routes generated by DER for all samples. As shown in Table 6, a substantial majority, i.e., 55.7% of DER's answer routes on *MixInstruct* have a length $T \leq 2$, validating the practical efficiency of DER.

**Effects of the number of experts selected for each state.** We conduct experiments where two experts are selected in each state to demonstrate that selecting one expert per state is optimal. From Table 7, DER produces better-generated answers by selecting an expert per state, with a 65% reduction in inference parameters. The reason is that the knowledge gained from the non-optimal experts selected in each state affects the quality of the expert's answer in the next state.

### 4.3 Eaxmple of DER

From Appendix 5, the DER-Agent selects a suitable LLM to continue the answering task as the sequence decision process proceeds, and the answering performance ends up being the best. Notably, the next LLM expert tends to leverage the knowledge of the previous LLM expert to improve the output answer. Thus, DER assembles complementary knowledge among LLMs to obtain better output answers.

## 5 Conclusion

In this paper, we propose a novel dynamic ensemble reasoning method, called DER, to integrate the strengths of LLM experts dynamically conditioned on dynamic inputs. Specifically, we model the LLM ensemble as an MDP, where a DER-Agent takes dynamic inputs, sequentially asks an LLM candidate to provide knowledge, and passes the knowledge to subsequent LLM candidates. We devise a reward function to train a DER-Agent to select an optimal answering route given the input questions, aiming to achieve the highest performance with as little computational cost as possible. We develop a KTP that enables the subsequent LLM to utilize the expert knowledge of the prior LLMs. Experiments demonstrate that DER integrates the advantages of LLMs with only 15% of the inference parameters compared to the LLM ensemble methods based on the output of all LLMs.

## Acknowledgments

This work was partially supported by the Joint Funds of the National Natural Science Foundation of China (Grant No.U24A20327), Key-Area Research and Development Program of Guangdong Province (2018B010107001), and TCL Science and Technology Innovation Fund.

## Contribution Statement

This work was a collaborative effort by all contributing authors. Jinwu Hu, Yufeng Wang, and Shuhai Zhang made equal contributions to this study and are designated as co-first authors. Mingkui Tan and Bin Xiao, serving as the corresponding authors, are responsible for all communications related to this manuscript.

## References

[Achiam *et al.*, 2023] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[Chen *et al.*, 2024] Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *The 62st Annual Meeting Of The Association For Computational Linguistics*, 2024.

[Chiang *et al.*, 2023] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2023.

[Chung *et al.*, 2024] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[Cobbe *et al.*, 2021] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[Conover *et al.*, 2023] Mike Conover, Matt Hayes, Ankit Mathur, Xiangrui Meng, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, et al. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

[Dong *et al.*, 2024] Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. Self-collaboration code generation via chatgpt. *ACM Transactions on Software Engineering and Methodology*, 33(7):1–38, 2024.

[Du *et al.*, 2022] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 320–335, 2022.

[Du *et al.*, 2023] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*, 2023.

[Geng *et al.*, 2023] Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. Koala: A dialogue model for academic research. Blog Post, April 2023.

[Hu *et al.*, 2025] Jinwu Hu, Wei Zhang, Yufeng Wang, Yu Hu, Bin Xiao, Mingkui Tan, and Qing Du. Dynamic compressing prompts for efficient inference of large language models. *arXiv preprint arXiv:2504.11004*, 2025.

[Jiang *et al.*, 2023] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise comparison and generative fusion. *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics*, 1:14165–14178, 2023.

[Jiang *et al.*, 2024] 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, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[junyou li *et al.*, 2024] junyou li, Qin Zhang, Yangbin Yu, QIANG FU, and Deheng Ye. More agents is all you need. *Transactions on Machine Learning Research*, 2024.

[Kong *et al.*, 2024] Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. Better zero-shot reasoning with role-play prompting. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4099–4113, 2024.

[LAION-AI, 2023] LAION-AI. Open assistant. 2023.

[Liu *et al.*, 2024] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *Conference On Language Modeling*, 2024.

[Lu *et al.*, 2024] Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1964–1974, 2024.

[Matena and Raffel, 2022] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

[Mu *et al.*, 2023] Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng

Dai, Yu Qiao, and Ping Luo. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *Advances in Neural Information Processing Systems*, 36:25081–25094, 2023.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Sheng *et al.*, 2023] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR, 2023.

[Stability-AI, 2023] Stability-AI. Stablelm: Stability ai language models. 2023.

[Stark *et al.*, 2024] Carson Stark, Bohkyung Chun, Casey Charleston, Varsha Ravi, Luis Pabon, Surya Sunkari, Tarun Mohan, Peter Stone, and Justin Hart. Dobby: a conversational service robot driven by gpt-4. In *2024 33rd IEEE International Conference on Robot and Human Interactive Communication (ROMAN)*, pages 1362–1369. IEEE, 2024.

[Sun *et al.*, 2023] Tianxiang Sun, Xiaotian Zhang, Zhengfu He, Peng Li, Qinyuan Cheng, Hang Yan, Xiangyang Liu, Yunfan Shao, Qiong Tang, Xingjian Zhao, et al. Moss: Training conversational language models from synthetic data. *arXiv preprint arXiv:2307.15020*, 7(3), 2023.

[Tang *et al.*, 2024] Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. Merging multi-task models via weight-ensembling mixture of experts. In *International Conference on Machine Learning*, pages 47778–47799. PMLR, 2024.

[Taori *et al.*, 2023] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

[Team and others, 2023] MosaicML NLP Team et al. Introducing mpt-7b: A new standard for open-source, ly usable llms, 2023.

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[Van Otterlo and Wiering, 2012] Martijn Van Otterlo and Marco Wiering. Reinforcement learning and markov decision processes. In *Reinforcement Learning: State-of-the-Art*, pages 3–42. Springer, 2012.

[Wan *et al.*, 2024] Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. Knowledge fusion of large language models. *International Conference on Learning Representations*, 2024.

[Wang *et al.*, 2025] Qianyue Wang, Jinwu Hu, Zhengping Li, Yufeng Wang, Yu Hu, Mingkui Tan, et al. Generating long-form story using dynamic hierarchical outlining with memory-enhancement. *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics*, 2025.

[Xu *et al.*, 2023] Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6268–6278, 2023.

[Yao *et al.*, 2023] Jie Yao, Zihao Zhou, and Qiufeng Wang. Solving math word problem with problem type classification. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 123–134. Springer, 2023.

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

[Zhang *et al.*, 2019] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[Zhang *et al.*, 2022] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.