TaD: A Plug-and-Play Task-Aware Decoding Method to Better Adapt LLMs on Downstream Tasks

Xinhao Xu[1,2], **Hui Chen**[2*], **Zijia Lin**[1], **Jungong Han**[3], **Lixing Gong**[4], **Guoxin Wang**[4], **Yongjun Bao**[4] and **Guiguang Ding**[1,2]
[1]School of Software, Tsinghua University
[2]Beijing National Research Center for Information Science and Technology (BNRist)
[3]Department of Computer Science, University of Sheffield
[4]JD.com
xxh22@mails.tsinghua.edu.cn, huichen@mail.tsinghua.edu.cn, linzijia07@tsinghua.org.cn, jungonghan77@gmail.com, {gonglixing, wangguoxin14, baoyongjun}@jd.com, dinggg@tsinghua.edu.cn

Abstract

Fine-tuning pre-trained models on downstream tasks is a common practice in leveraging large language models (LLMs) today. A critical issue is how to adapt pre-trained models to downstream tasks better, thereby enhancing their performance. This paper introduces ***T**ask-**a**ware **D**ecoding (TaD)*, a plug-and-play method that exploits the difference in output probability distributions before and after fine-tuning to boost the performance of LLMs on downstream tasks. The proposed *TaD* argues that the difference between the pre-fine-tuning probability distribution and the post-fine-tuning one represents the *direction* from common knowledge towards specific downstream-task knowledge. Aligning the final output probability distribution to that direction can probably result in superior downstream task performance, compared to the original fine-tuned model. Experiments on various datasets across four different task categories well demonstrate *TaD*'s effectiveness on different LLMs, *i.e.*, GPT, BLOOM, and LLaMA, with different fine-tuning methods. Moreover, further experiments reveal that *TaD* better enhances model performance in data-scarce scenarios.

1 Introduction

Large language models, including closed-source models like ChatGPT and GPT-4 [OpenAI, 2023], as well as open-source models such as LLaMA [Touvron *et al.*, 2023], have exhibited remarkable performance in a wide range of tasks [Brown *et al.*, 2020; Ding *et al.*, 2021; Wang *et al.*, 2023; Chen *et al.*, 2020]. Such success is largely due to diverse techniques explored to enhance the pre-trained LLMs in downstream tasks. Among those methods, fine-tuning is a common strategy and has been thoroughly investigated in the literature. It mainly focuses on designing better fine-tuning methods from the algorithmic side [Ding *et al.*, 2023; Hu *et al.*, 2021], or constructing more effective datasets from the data side [Christiano *et al.*, 2017]. For example, the Parameter-Efficient Fine-Tuning (PEFT) methods [Houlsby *et al.*, 2019; Lester *et al.*, 2021; Li and Liang, 2021] stand out among various fine-tuning methods due to their cost-efficient nature and impressive performance. Concurrently, a multitude of high-quality, manually annotated datasets have been constructed, as an increasingly focal technique aimed at aligning the outputs of LLMs with human-like responses [Ouyang *et al.*, 2022].

Despite their remarkable success, existing fine-tuning works rarely investigate the inherent knowledge acquisition of fine-tuned LLMs. Recent works indicate that the outputs of pre-trained LLMs do not always accurately reflect the knowledge they possess. Even if a model generates an incorrect output, it may still possess correct knowledge. For example, [Li *et al.*, 2023b] suggests that the intermediate representations in pre-trained LLMs might be correct, even if the final output is erroneous. [Kadavath *et al.*, 2022] finds that pre-tained LLMs can self-assess their correctness. Therefore, we pose a question: *How can we leverage such inherent knowledge in the fine-tuned LLMs to enhance their performance in downstream tasks?* Intuitively, the inherent knowledge within fine-tuned LLMs should be reflected by their token-predicting behavior alterations during the fine-tuning process. We argue that such token-predicting behavior alterations indicate an adaptive shift from common knowledge gained via pre-training to specific knowledge for downstream tasks. And we can improve the adaptation of LLMs on downstream tasks by manually mining and leveraging such inherent knowledge, regardless of the fine-tuning methods.

In this paper, we propose a novel ***T**ask-**a**ware **D**ecoding **(TaD)*** method, which takes advantage of the differences in knowledge before and after fine-tuning to enhance the adaptation of LLMs on downstream tasks. Firstly, we formulate the knowledge difference as a *knowledge vector* to explicitly denote the direction of *knowledge adaptation* (or *domain adaptation*) learned by a pre-trained LLM during fine-tuning for a downstream task. As illustrated in Figure 1, though the fine-tuned LLM (*i.e.*, after fine-tuning) outputs the

*Corresponding author

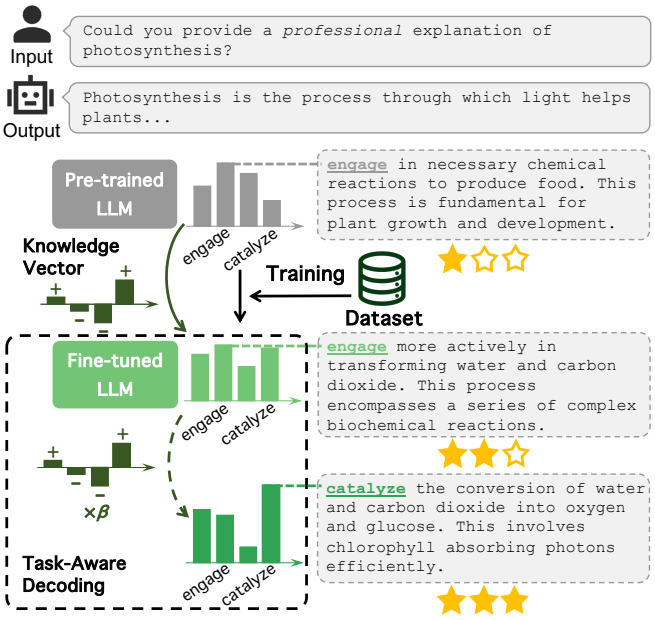

Figure 1: An illustration of the proposed *knowledge vector* and task-aware decoding (*TaD*), where LLMs are asked for a *professional* explanation of `photosynthesis`. The underlined tokens are examples to illustrate the *professional* levels of the outputs of pre-trained, fine-tuned, and *TaD*-enhanced fine-tuned LLMs, and the corresponding probability distributions are displayed on the left. The proposed *TaD* enhances the predicted probability distribution of a token with the *knowledge vector*, and thus amplifies the knowledge learned during fine-tuning and achieves superior performance in downstream tasks.

same token "`engage`" as the pre-trained LLM (*i.e.*, before fine-tuning) for the next token, the shift of the corresponding predicted probability distributions from the latter to the former implicitly reflects how the LLM adapts its knowledge to the downstream task during fine-tuning. Specifically, the predicted probability w.r.t the more *professional* token "`catalyze`" increases, while that w.r.t the less *professional* token "`engage`" actually decreases. Therefore, here we construct the *knowledge vector* based on the difference of output probability distributions w.r.t tokens from the pre-trained and the fine-tuned LLMs, which also reveals the output token preferences for downstream tasks. Given that tokens are inherently semantic, the *knowledge vector* naturally possesses semantic information. We then implement our proposed *TaD* by enhancing the fine-tuned LLM's output probability distribution with the *knowledge vector*, thereby reinforcing the model's knowledge adaptation to downstream tasks for better performance. Figure 1 shows that, with *TaD* to enhance the output probability distribution with the *knowledge vector*, the fine-tuned LLM can then generate a more *professional* next token "`catalyze`", instead of "`engage`". In that manner, the enhanced LLM can finally yield a better answer with more professional words, including "`catalyze`","`oxygen`","`glucose`","`chlorophyll`", etc. The proposed *TaD* is a plug-and-play decoding method, and extensive experiments well demonstrate that it can enhance the performance of various fine-tuned LLMs in many downstream tasks.

In summary, our contributions are as follows:

1. To enhance the adaptation of LLMs to downstream tasks, we propose a concept of *knowledge vector*, which explicitly denotes the knowledge adaptation learned by LLMs during fine-tuning.
2. We further develop *TaD* to enhance fine-tuned LLMs' output probability distribution w.r.t tokens with the *knowledge vector*, and enhance their performance in downstream tasks.
3. We conduct extensive experiments to validate the effectiveness of *TaD* across various tasks, models, and fine-tuning methods. Experimental results well demonstrate its superiority over baselines and promising potential in data-scarce scenarios.

2 Related Work

2.1 Adapting LLMs for Downstream Tasks

Many efforts tackle the challenge of adapting pre-trained LLMs for downstream tasks by focusing on both fine-tuning techniques and the construction of training datasets [Ding *et al.*, 2023; Christiano *et al.*, 2017]. Among various fine-tuning techniques, PEFT methods [Hu *et al.*, 2021; He *et al.*, 2022; Ding *et al.*, 2023; Xiong *et al.*, 2024], due to their capability of freezing most model parameters while yielding impressive performance, have drawn much attention from the community. And thus recently PEFT methods specifically for LLMs, like [Houlsby *et al.*, 2019; Li and Liang, 2021; Li *et al.*, 2022], have been proposed. Concurrently, for aligning LLM's behaviors with human-like responses, fine-tuning LLMs typically requires manually labeled datasets [Ouyang *et al.*, 2022], which incurs significant costs. In contrast, some studies construct high-quality training datasets by maximizing the utilization of existing datasets. For example, [Sanh *et al.*, 2022] enriches the training instances by inverting input-output pairs of existing instances with specially crafted task descriptions. [Wei *et al.*, 2021] augments labeled datasets with human-written task descriptions to instruct LLMs to understand the tasks.

Aside from those efforts, there are initiatives that enhance LLMs by focusing on their intrinsic features. [Li *et al.*, 2023a; Hernandez *et al.*, 2023] employs post-pretraining activation editing to modulate LLM behaviors. [Subramani *et al.*, 2022; Turner *et al.*, 2023] demonstrate that *steering vectors*, whether trained or manually selected, effectively facilitate style transfer in LLMs. ITI [Li *et al.*, 2023b] enhances LLMs' performance in downstream tasks by discovering vectors for activations based on positive and negative instances. Our *TaD* follows a similar direction. But it opts for the differences in output probability distributions w.r.t tokens as the direction of knowledge adaptation, and derives a *knowledge vector* to enhance the performance of fine-tuned LLMs.

2.2 Decoding Strategies for LLMs

Decoding strategies are critical for LLMs, which significantly affect the generation quality. Basic decoding strategies include Greedy Search, Beam Search [Sutskever *et al.*, 2014],

Top-k Sampling [Fan *et al.*, 2018], and Top-p (Nucleus) Sampling [Holtzman *et al.*, 2019]. Greedy Search always selects the most probable next token, one by one. While efficient, it tends to yield repetitive text. In contrast, Beam Search maintains and expands multiple hypotheses for superior sequences. Top-k Sampling, choosing from highly probable tokens, injects diversity but may compromise coherence. Building on that, Top-p Sampling dynamically adjusts the selection pool based on the cumulative probability, targeting at a balance between randomness and coherence for more engaging outputs.

Additionally, there exists a range of incremental works, which can be integrated with and effectively optimize those basic strategies. Within this scope, the contrastive decoding series serves as a prime example. CD [Li *et al.*, 2023c] utilizes the likelihood difference between a large language model and a small one to produce higher-quality texts. Furthermore, ACD [Gera *et al.*, 2023] improves upon CD by maximizing log-probability differences across different layers in a single model. DoLa [Chuang *et al.*, 2023] extends ACD's principles, integrating Jensen-Shannon divergence to dynamically choose layers for contrastive decoding. Those decoding strategies, however, are primarily focused on the performance of pre-trained LLMs, neglecting the changes in models after fine-tuning. Consequently, they fail to utilize the knowledge adaptation learned during fine-tuning, resulting in limited performance improvement or even performance decline in downstream tasks when applied to fine-tuned LLMs. On the contrary, our proposed *TaD* focuses on fine-tuned LLMs, and we define a *knowledge vector* to denote knowledge adaptation for downstream tasks. *TaD* is a plug-and-play method that can be natively integrated with those basic decoding strategies above to enhance the performance of fine-tuned LLMs in downstream tasks.

3 Method

In this section, we first discuss the construction of the *knowledge vector*, then detail the process of the proposed *TaD*, during which we utilize the *knowledge vector* to improve LLMs' outputs for a better adaptation to downstream tasks.

3.1 Constructing the Knowledge Vector

For a pre-trained language model θ, the conditional probability distribution for tokens is modeled as follows:

$$p_\theta(x_t|x_{<t}),\ x_t \in \mathcal{V}, \tag{1}$$

where $\mathcal{V}$ denotes the vocabulary and t represents the token's position. After applying a fine-tuning method Φ on the pre-trained model θ, we get a fine-tuned model ϕ:

$$\phi = \Phi(\theta, \mathcal{D}), \tag{2}$$

where Φ can be any fine-tuning method like LoRA, AdapterP, etc., and $\mathcal{D}$ denotes the training set for a downstream task. Similarly, for the fine-tuned model ϕ, the conditional probability distribution for tokens is formulated as Eq. 3:

$$p_\phi(x_t|x_{<t}),\ x_t \in \mathcal{V} \tag{3}$$

Given the finite size of vocabulary $\mathcal{V}$, at each position t, both

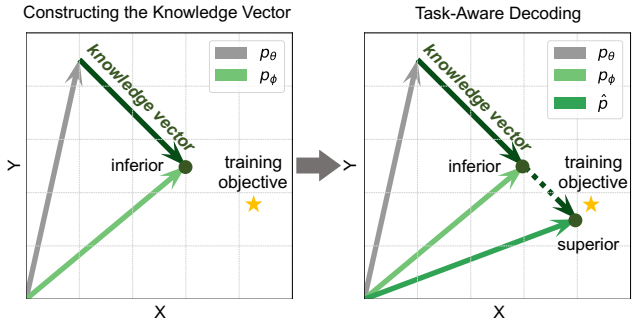

Figure 2: A simplified illustration of our method. The *knowledge vector* is constructed as the vector pointing from p_θ to p_ϕ, which indicates an approximate direction towards the training objective. *TaD* subsequently improves the inferior outputs in this direction to obtain superior outputs.

$p_\theta(x_t|x_{<t})$ and $p_\phi(x_t|x_{<t})$ can be viewed as coordinates in a $|\mathcal{V}|$-dimensional space. Consequently, we can derive a $|\mathcal{V}|$-dimensional vector $\mathcal{V}_K$ on the logit scale, extending from the fine-tuning start point $p_\mathcal{S}$ (*i.e.*, θ) to the fine-tuning endpoint $p_\mathcal{E}$ (*i.e.*, ϕ):

$$\begin{aligned}\mathcal{V}_K &= p_\mathcal{E} - p_\mathcal{S} \\ &= \log p_\phi(x_t|x_{<t}) - \log p_\theta(x_t|x_{<t})\end{aligned} \tag{4}$$

$\mathcal{V}_K$ reflects the shift in conditional probability distributions during fine-tuning, representing a directional move from the common knowledge to the task-specific knowledge. Notably, $\mathcal{V}_K$ directly shows the influence of fine-tuning on the model's output probability distribution, and thus also enhances the interpretability of the fine-tuning process.

It is important to note that $\mathcal{V}_K$ may exhibit false positive cases, as mentioned in CD [Li *et al.*, 2023c]. For example, tokens with extremely low p_ϕ and p_θ (*e.g.*, 10^{-5} and 10^{-10}) might have disproportionately high $\mathcal{V}_K$ values, misleadingly indicating an incorrect *direction* in the corresponding dimension (*i.e.*, token). Therefore, we introduce a constraint function $\mathcal{C}_t$ to ensure the probability values of tokens output by fine-tuned model ϕ are at least α times the maximum token probability:

$$\mathcal{C}_t = \{x_t \in \mathcal{V} : p_\phi(x_t|x_{<t}) \geq \alpha \max_{x_t' \in \mathcal{V}} p_\phi(x_t'|x_{<t})\}, \tag{5}$$

where α ranges from 0 to 1.

Moreover, the indicator function $\mathbb{I}(x_t)$,

$$\mathbb{I}(x_t) = \begin{cases} 1 & \text{if } x_t \in \mathcal{C}_t \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

determines whether a token x_t satisfies the constraint function $\mathcal{C}_t$. We then introduce a penalty coefficient λ to penalize tokens that violate the constraint function, while maintaining the original values of $\mathcal{V}_K$ for all others. This adjustment yields a revised $\hat{\mathcal{V}}_K$, which can effectively prevent false positive cases, as follows:

$$\hat{\mathcal{V}}_K = \mathbb{I}(x_t) \cdot \mathcal{V}_K + (1 - \mathbb{I}(x_t)) \cdot \lambda \tag{7}$$

Then we have the final *knowledge vector* $\hat{\mathcal{V}}_K$.

Model	Method	Multiple Choices			CBQA
		MC1	MC2	MC3	True*Info
GPT-J-6b	LoRA	30.6	51.3	25.6	35.7
	+*TaD*	**33.0**	**52.5**	**27.1**	**37.0**
	AdapterP	34.9	54.3	28.0	51.5
	+*TaD*	**38.2**	**55.5**	**29.5**	**51.7**
	AdapterH	36.4	55.0	28.5	53.0
	+*TaD*	**38.3**	**55.8**	**28.7**	**55.3**
	Parallel	34.3	54.0	27.7	47.2
	+*TaD*	**37.5**	**55.1**	**28.9**	**47.4**
BLOOMz-7b	LoRA	30.8	51.4	25.7	17.4
	+*TaD*	**32.8**	**52.3**	**27.2**	**17.5**
	AdapterP	35.3	53.8	**28.5**	20.6
	+*TaD*	**35.7**	**54.8**	28.4	**20.7**
	AdapterH	36.8	54.5	28.9	50.3
	+*TaD*	**37.9**	**55.2**	**29.2**	**50.8**
	Parallel	34.5	53.6	28.2	21.8
	+*TaD*	**36.5**	**54.4**	**28.5**	**22.7**

Model	Method	Multiple Choices			CBQA
		MC1	MC2	MC3	True*Info
LLaMa-7b	LoRA	32.9	55.0	28.5	49.1
	+*TaD*	**34.2**	**55.7**	**29.0**	**51.2**
	AdapterP	38.1	57.4	30.8	61.4
	+*TaD*	**40.6**	**58.5**	**32.1**	**61.8**
	AdapterH	37.8	57.6	30.3	60.3
	+*TaD*	**39.8**	**59.0**	**32.0**	**61.0**
	Parallel	37.0	56.3	29.5	54.3
	+*TaD*	**39.5**	**57.0**	**30.4**	**55.2**
LLaMa-13b	LoRA	33.4	55.7	29.0	54.1
	+*TaD*	**35.1**	**56.7**	**29.7**	**54.7**
	AdapterP	40.6	58.8	32.4	58.6
	+*TaD*	**42.6**	**60.0**	**33.1**	**60.0**
	AdapterH	38.2	57.0	30.4	61.8
	+*TaD*	**39.5**	**57.8**	**31.2**	**63.3**
	Parallel	39.8	58.2	31.7	60.0
	+*TaD*	**42.0**	**60.2**	**33.8**	**61.6**

Table 1: Results on multiple-choice and closed-book question answering (CBQA) tasks.

3.2 Task-Aware Decoding

With the constructed *knowledge vector*, *i.e.*, $\hat{\mathcal{V}}_K$ in Eq. 7, we advance the probability outputs of the fine-tuned model in the direction indicated by the *knowledge vector*, effectively amplifying the downstream task knowledge adaptation learned during fine-tuning. It is important to note that $\hat{\mathcal{V}}_K$ is not a probability distribution. Hence, we apply the `softmax` function to convert it into a probability distribution:

$$p_K(x_t|x_{<t}) = \text{softmax}(\hat{\mathcal{V}}_K) \tag{8}$$

Then we obtain the probability distribution $\hat{p}$ w.r.t x_t output by *TaD*, via merging $p_\phi(x_t|x_{<t})$ and $p_K(x_t|x_{<t})$ with a weighting parameter μ.

$$\hat{p}(x_t|x_{<t}) = (1-\mu)\cdot p_\phi(x_t|x_{<t}) + \mu \cdot p_K(x_t|x_{<t}) \tag{9}$$

With the improved output probability distribution w.r.t x_t, *i.e.*, $\hat{p}(x_t|x_{<t})$, we can simply apply various basic decoding methods like Greedy Search for text generation. A simplified illustration of the proposed *TaD* is shown in Figure 2.

Deriving from the *knowledge vector*, *i.e.*, $\hat{\mathcal{V}}_K$, p_K indicates the impact of fine-tuning on the model's output. Eq. 9 enables the modulation of this impact by adjusting μ. Therefore, we can make LLM's implicit knowledge learned during fine-tuning explicit by simply increasing the value of μ to amplify this impact. Additionally, we preserve p_ϕ in Eq. 9 to prevent false negative cases caused by using p_K alone. The false negative case occurs when the next token is easily predictable. In this circumstance, both p_ϕ and p_θ yield a probability near 1, but the value of p_K is close to 0. Preserving p_ϕ and choosing a proper μ guarantee that the model outputs tokens with high probability in p_ϕ. We believe this preservation addresses the limitations of using p_K alone, proving essential for preserving the baseline fine-tuned model's efficacy and ensuring the effectiveness of *TaD*. Moreover, it also can be seen that when $\mu = 0$, the proposed *TaD* would degrade to the original fine-tuned model, *i.e.*, p_ϕ, and thus the performance of *TaD* would at least match that of the original fine-tuned model.

4 Experiments

4.1 Setup

Models and Datasets

For our experiments, we select three remarkable LLMs: LLaMA (including LLaMA-7b and LLaMA-13b) [Touvron *et al.*, 2023], GPT-J (6b) [Wang and Komatsuzaki, 2021] and BLOOMz (7b) [Muennighoff *et al.*, 2022], and focus on two main categories of downstream tasks: multiple-choice tasks and open-ended generation tasks.

For multiple-choice tasks, we employ the commonly used TruthfulQA [Lin *et al.*, 2022] benchmark, following previous works [Li *et al.*, 2023b; Chuang *et al.*, 2023]. In this benchmark, questions are concatenated with correct or incorrect answers and fed into the model. Then the model's performance is evaluated with MC (multiple-choice accuracy) based on the predicted probabilities for each question-answer pair. Specifically, TruthfulQA involves three specific MC scores, *i.e.*, MC1, MC2, and MC3, for different experimental setups, and higher MC1/2/3 scores indicate better model performance.

For open-ended generation tasks, we focus on three subtasks: closed-book question answering, mathematical reasoning, and commonsense reasoning. For closed-book question answering, we use TruthfulQA, where the model's input and output are, respectively, given questions and generated answers. The answers are evaluated for Truthfulness and Informativeness using GPT-3 [Brown *et al.*, 2020] fine-tuned with the official dataset. For mathematical reasoning, we utilize

Model	Method	Math Reasoning		CS Reasoning	
		GSM8K	MultiArith	BoolQ	PIQA
GPT-J-6b	LoRA	21.9	92.5	61.8	63.4
	+*TaD*	**22.8**	**94.2**	**62.7**	**64.6**
	AdapterP	19.0	92.2	63.9	71.0
	+*TaD*	**19.5**	**92.5**	**64.2**	**71.2**
BLOOMz-7b	LoRA	18.9	91.7	66.8	73.6
	+*TaD*	**19.3**	**94.2**	**66.9**	**73.9**
	AdapterP	16.3	90.7	66.2	74.4
	+*TaD*	**17.1**	**93.0**	66.2	**75.0**
LLaMa-7b	LoRA	26.6	90.5	68.7	78.9
	+*TaD*	**27.7**	**91.0**	**69.3**	**79.5**
	AdapterP	31.5	93.5	65.4	76.3
	+*TaD*	**32.0**	**93.7**	**66.3**	76.3
LLaMa-13b	LoRA	35.9	91.5	70.1	82.5
	+*TaD*	**38.1**	**92.0**	**70.8**	**83.1**
	AdapterP	36.8	91.5	69.4	78.1
	+*TaD*	**37.5**	**94.0**	69.4	**79.2**

Table 2: Results on mathematical reasoning and commonsense reasoning tasks.

GSM8K [Cobbe *et al.*, 2021] and MultiArith [Roy and Roth, 2016]. GSM8K offers a variety of high-quality, grade-school math word problems crafted by human writers. MultiArith involves math problems requiring extensive reasoning and computational steps. For commonsense reasoning, we employ BoolQ [Clark *et al.*, 2019] and PIQA [Bisk *et al.*, 2020]. BoolQ comprises a dataset of 15,942 yes/no questions. PIQA offers a collection of questions demanding physical commonsense reasoning.

Following previous works [Chuang *et al.*, 2023; Hu *et al.*, 2023], we employ the same prompt as the official one proposed in [Lin *et al.*, 2022], and formulate specific instructions for both mathematical and commonsense reasoning tasks.

Fine-Tuning Methods and Implementation Details

PEFT methods are widely adopted for LLM fine-tuning, offering computational efficiency and comparable performance to full-parameter fine-tuning. In our experiments, we employ four PEFT methods to incorporate the proposed *TaD*, *i.e.*, LoRA [Hu *et al.*, 2021], AdapterP [Pfeiffer *et al.*, 2020], AdapterH [Houlsby *et al.*, 2019] and Parallel Adapter [He *et al.*, 2022]. Specifically, our fine-tuning settings are based on LLM-Adapters [Hu *et al.*, 2023], which provides comprehensive insights on applying PEFT to LLMs. We align our fine-tuning hyper-parameters with the study.

For TruthfulQA, we adopt a 2-fold cross-validation strategy following ITI [Li *et al.*, 2023b], ensuring no test data leakage, and we perform fine-tuning with only <Question, Best Answer> pairs. In mathematical and commonsense reasoning, we utilize Math10K and Commonsense170K datasets from LLM-Adapters for fine-tuning, followed by evaluations on GSM8K, MultiArith, BoolQ, and PIQA.

Model	Method	Multiple Choices			Math Reasoning	
		MC1	MC2	MC3	GSM8K	MultiArith
LLaMa-7b	LoRA	32.9	55.0	28.5	26.6	90.5
	+*DoLa*	31.6	48.6	22.7	26.6	89.7
	+*TaD*	**34.2**	**55.7**	**29.0**	**27.7**	**91.0**
	AdapterP	38.1	57.4	30.8	31.5	93.5
	+*DoLa*	39.7	54.9	25.5	31.5	93.3
	+*TaD*	**40.6**	**58.5**	**32.1**	**32.0**	**93.7**
LLaMa-13b	LoRA	33.4	55.7	29.0	35.9	91.5
	+*CD*	**36.2**	55.4	26.5	19.0	70.3
	+*DoLa*	34.9	51.2	24.8	38.0	**94.2**
	+*TaD*	35.1	**56.7**	**29.7**	**38.1**	92.0
	AdapterP	40.6	58.8	32.4	36.8	91.5
	+*CD*	41.1	56.0	26.2	17.8	72.5
	+*DoLa*	41.3	56.5	27.5	35.9	93.5
	+*TaD*	**42.6**	**60.0**	**33.1**	**37.5**	**94.0**

Table 3: The comparison results with other contrastive decoding strategies on multiple-choice and mathematical reasoning tasks.

Moreover, we set α in Eq. 5 and λ in Eq. 7 by default to 0.1 and $-\infty$, respectively, without additional tuning. For TruthfulQA, we set the weighting parameter μ in Eq. 9 to 0.8 empirically for all setups. For mathematical and commonsense reasoning tasks, we select the optimal μ based on the model's performance on the training sets of GSM8K and BoolQ, respectively, because we utilize Math10K and Commonsense170K rather than them for fine-tuning. Then the selected μ on GSM8K is applied to MultiArith, and similarly, the selected μ on BoolQ is applied to PIQA, so as to see the transferability of μ from one dataset to another in the same task category. Unless explicitly stated, all experiments employ Greedy Search, which is a commonly used and time-efficient decoding strategy adopted in many previous works [Li *et al.*, 2023b; Chuang *et al.*, 2023].

4.2 Main Results

Results on Multiple-Choice and Closed-Book Question Answering Tasks

As shown in Table 1, *TaD* consistently improves fine-tuned LLMs' performance across different models and PEFT methods. The experimental results well demonstrate its general applicability. Moreover, given that the metric MC1 focuses on only Best Answer while MC2/MC3 also considers other Correct Answers, we can see that, despite our training data comprising only <Question, Best Answer> pairs, the alignment to the *knowledge vector* introduced by *TaD* also improves both MC2 and MC3.

Results on Mathematical Reasoning and Commonsense Reasoning Tasks

Table 2 illustrates the performance of our method on more challenging reasoning tasks. Without loss of generality, we just present LoRA and AdapterP's results here. The results show that *TaD* exhibits substantial improvement across various mathematical reasoning datasets and models, and exceeds

Model	Method	G / M	Model	Method	G / M
LLaMa-7b	Greedy	26.6/90.5	LLaMa-13b	Greedy	35.9/91.5
	+*TaD*	**27.7/91.0**		+*TaD*	**38.1/92.0**
	Beam-4	30.5/91.3		Beam-4	43.6/93.3
	+*TaD*	**30.9/91.8**		+*TaD*	**43.7/94.3**
	Top-p	26.7/90.7		Top-p	36.7/91.7
	+*TaD*	**27.4/91.3**		+*TaD*	**37.1/93.0**
	Top-k	27.0/90.3		Top-k	36.8/91.7
	+*TaD*	**27.7/91.6**		+*TaD*	**37.2/93.0**

Table 4: Integrating *TaD* with four basic decoding strategies, Greedy Search, Beam Search, Top-p sampling, and Top-k sampling, where Beam-4 represents the number of beams is 4. G and M represent GSM8K and MultiArith, respectively.

the performance of fine-tuned models in commonsense reasoning datasets.

Compared to commonsense reasoning tasks, *TaD* gets larger performance gains in mathematical reasoning tasks. It can be attributed to the characteristics of the latter's outputs, which are usually longer and semantically dense. That enhances *TaD*'s benefits in iterative generation and reasoning. In contrast, the commonsense reasoning benchmarks typically require outputs formatted like "*the correct answer is solution*[N]", which generally have limited semantic information. And *TaD*'s effectiveness primarily lies on the single token [N], which is limited and leads to smaller performance gains. Nonetheless, *TaD* still yields considerable improvements for commonsense reasoning tasks in nearly all cases, as shown in Table 2.

Comparison With Other Contrastive Decoding Strategies

We compare *TaD* with other baselines that do not consider the knowledge adaptation for downstream tasks during fine-tuning. Our experiments are conducted on TruthfulQA for multi-choice tasks and on mathematical reasoning datasets for open-ended generation tasks. Specifically, we compare *TaD* with CD [Li *et al.*, 2023c] and DoLa [Chuang *et al.*, 2023].

Following DoLa, we conduct comparisons on LLaMA-7b and LLaMA-13b. And for CD, we treat the fine-tuned LLaMA-7b as an amateur and the fine-tuned LLaMA-13b as an expert. For fairness, we apply both CD and DoLa on fine-tuned models, and we carefully tune their hyper-parameters to yield the best results for comparisons. Specifically, for TruthfulQA, considering the training data comprises only <Question, Best Answer> pairs, we select the optimal DoLa interval and hyper-parameters based on MC1 scores using a 2-fold cross-validation strategy.

The comparison results, as shown in Table 3, indicate that *TaD* outperforms the baselines in most cases. Specifically, *TaD* exhibits a significant advantage over CD and DoLa in maintaining or improving fine-tuned LLMs' performance in various tasks. Actually, CD or DoLa can even degrade the performance of the original fine-tuned LLMs. For example, on TruthfulQA, though CD's performance surpasses that of *TaD* in terms of MC1 under the "LLaMA-13b + LoRA" setup,

$\mathcal{M}$	$p_\mathcal{S} \to p_\mathcal{E}$	G / M
7b	/	10.8/37.5
7b*	/	26.6/90.5
13b	/	16.7/53.2
13b*	/	35.9/91.5

(a) Comparison results on pre-trained and fine-tuned models.

$\mathcal{M}$	$p_\mathcal{S} \to p_\mathcal{E}$	G / M
7b*	/	26.6/90.5
	7b* → 7b	23.7/79.0

(c) The effect of the opposite direction of the proposed *knowledge vector* (from the fine-tuned to the pre-trained model).

$\mathcal{M}$	$p_\mathcal{S} \to p_\mathcal{E}$	G / M
7b	/	10.8/37.5
	7b → 7b*	11.9/38.1
	7b → 13b	11.2/37.6

(e) Comparison results on the direction of the knowledge and model size difference.

$\mathcal{M}$	$p_\mathcal{S} \to p_\mathcal{E}$	G / M
7b*	/	26.6/90.5
	7b → 7b*	27.7/91.0
13b*	/	35.9/91.5
	13b → 13b*	38.1/92.0

(b) *TaD*'s effectiveness on the fine-tuned models.

$\mathcal{M}$	$p_\mathcal{S} \to p_\mathcal{E}$	G / M
13b	/	16.7/53.2
	7b → 13b	17.2/51.8
13b*	/	35.9/91.5
	7b* → 13b*	36.2/91.8

(d) The effect of the direction of the model size difference (from the smaller to the larger model).

$\mathcal{M}$	$p_\mathcal{S} \to p_\mathcal{E}$	G / M
13b*	/	35.9/91.5
	7b* → 13b*	36.2/91.8
	13b → 13b*	38.1/92.0
	7b → 13b*	38.2/92.0

(f) The cumulative effect of the direction of the knowledge and model size difference.

Table 5: Ablation study results of the *knowledge vector* on LLaMa models. The column $\mathcal{M}$ and $p_\mathcal{E} \to p_\mathcal{S}$ represent the models used for calculating p_ϕ in Eq. 5, Eq. 9 and the logarithm of the distribution in Eq. 4, respectively. A / in column $p_\mathcal{E} \to p_\mathcal{S}$ indicates the independent performance of model $\mathcal{M}$ without applying *TaD*. Models marked with an * signify those fine-tuned with LoRA on Math10K. G and M represent GSM8K and MultiArith, respectively. We highlight our method with a gray background.

it leads to a significant decline in terms of MC2 and MC3. In contrast, *TaD* consistently maintains and mostly improves the fine-tuned LLMs' performance in all MC1/2/3 metrics. On GSM8K and MultiArith, the decrease in CD's performance is particularly pronounced, which well aligns with the results reported in DoLa. In contrast, *TaD* still gains performance improvement upon fine-tuned LLMs. Overall, *TaD* generates better results in various downstream tasks by considering the knowledge adaptation learned during fine-tuning and consistently maintains and improves the fine-tuned LLMs' performance.

4.3 Analysis

Integrated With Different Basic Decoding Strategies

As previously mentioned, our derived $\hat{p}$ in Eq. 9 is compatible with various basic decoding strategies. Table 4 shows the experimental results for the more challenging mathematical reasoning task, with *TaD* incorporated in Greedy Search, Beam Search, Top-p sampling, and Top-k sampling, respectively. It is observed that *TaD* consistently improves upon the

Model	Method	10%	30%	60%	100%
LLaMa-7b	LoRA	58.8	80.5	86.2	90.5
	+*TaD*	**62.2**	**83.2**	**88.0**	**91.0**
	Δ	**+3.4**	**+2.7**	**+1.8**	**+0.5**
LLaMa-13b	LoRA	70.8	86.5	90.2	91.5
	+*TaD*	**79.8**	**89.3**	**91.5**	**92.0**
	Δ	**+9.0**	**+3.2**	**+1.3**	**+0.5**

Table 6: *TaD*'s performance with different ratios of Math10K as training dataset. Δ represents the improvement after applying *TaD*.

fine-tuned LLMs' performance across different basic decoding strategies on both mathematical reasoning datasets.

Ablation Study of the Knowledge Vector

Table 5 displays the ablation study results of the *knowledge vector* on the more challenging mathematical reasoning task. We conduct extensive experiments by defining different directions of the *knowledge vector* to demonstrate the effectiveness and reasonableness of the proposed one.

Firstly, we present the pre-trained and fine-tuned LLMs' performance and showcase the effectiveness of the proposed *TaD*. Table 5 (a) shows that the performance of the fine-tuned LLaMA-7b* and LLaMA-13b* is significantly improved compared to the original pre-trained LLaMA-7b and LLaMA-13b. Table 5 (b) further shows that the proposed *TaD* consistently enhances the performance of the fine-tuned LLaMA-7b* and LLaMA-13b* by setting the direction of the *knowledge vector* as 7b→7b* or 13b→13b*. To validate our motivation in setting the *knowledge vector*'s direction, we further investigate the effect of reverting the original *knowledge vector*, *i.e.*, from the fine-tuned LLM to the pre-trained LLM, as shown in Table 5 (c). We can see that reverting the direction of the proposed *knowledge vector* significantly degrades the fine-tuned LLaMA-7b*'s performance, which is consistent with our illustration in Figure 2.

Furthermore, we find that the direction from the smaller LLMs to the larger LLMs can also improve the pre-trained or fine-tuned LLMs' performance, as can be seen in Table 5 (d). Therefore, we further conduct experiments to compare the *knowledge vector* derived from knowledge adaptation (*i.e.*, from the pre-trained LLMs to the fine-tuned LLMs) and that derived from model size increase (*i.e.*, from the smaller LLMs to the larger LLMs). The results are reported in Table 5 (e), and we can see that the former outperforms the latter. It suggests that the direction indicated by the knowledge adaptation learned during fine-tuning is more essential in deriving the proposed *knowledge vector*. Moreover, we investigate the cumulative effect of combining these two kinds of directions, by setting the direction of *knowledge vector* from the pre-trained LLaMA-7b to the fine-tuned LLaMA-13b*. Table 5 (f) shows that combining both directions can gain slight or even negligible further improvement over the default direction from LLaMA-13b to LLaMA-13b*, which further demonstrates the reasonableness of how we define the direction of the proposed *knowledge vector*.

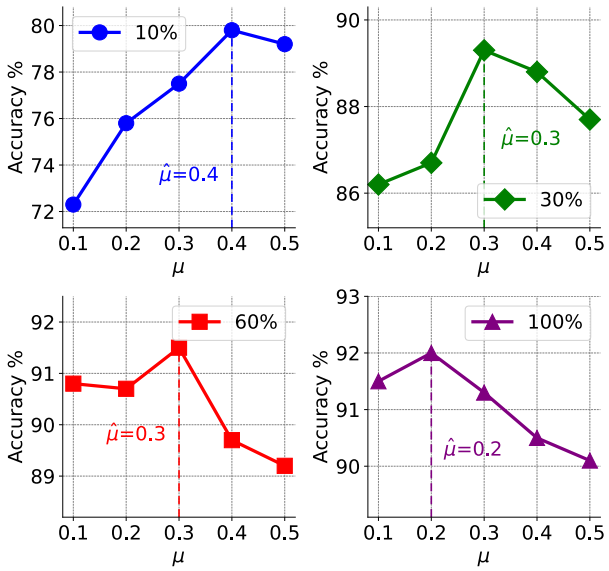

Figure 3: Ablation study results on the weighting parameter μ in the LLaMA-13b model. The numbers in the legend represent the training data ratios and $\hat{\mu}$ marks the optimal value in each ratio.

Different Ratios of Training Datasets and the Selection of μ

Table 6 displays the performance of models fine-tuned with LoRA using different ratios of training data and highlights *TaD*'s improvement over the original fine-tuned models. For each setup, we select the optimal hyper-parameters and train the LLMs for sufficient epochs to ensure convergence. As shown in Table 6, we can see that, for smaller ratios of the training data, the proposed *TaD* yields larger performance gains.

Furthermore, we conduct an ablation study on the selection of μ, and present the results in Figure 3. It can be observed that the optimal $\hat{\mu}$ incrementally increases as the ratio of training data decreases, meaning that the fine-tuned LLMs prefer stronger alignment with the *knowledge vector* for smaller ratios of training data.

Intuitively, with less training data, the LLMs' output probability distribution w.r.t tokens is far from reaching the final training objective in Figure 2. Therefore, starting from a more distant point, moving in the direction of the *knowledge vector* can strengthen the knowledge adaptation for the downstream tasks and lead to more significant improvements. Interestingly, the results above also indicate that even with limited training data, the model can be guided in the correct direction. Such a finding well highlights our method's effectiveness in data-limited scenarios.

5 Conclusion

In this paper, we introduce *TaD*, a plug-and-play method to enhance the performance of fine-tuned LLMs in downstream tasks. *TaD* leverages the differences in output probability distributions w.r.t tokens before and after fine-tuning, to construct the *knowledge vector* for downstream tasks. Then *TaD* refines the output probability distribution of the fine-tuned

Model	Method	MC Average
LLaMa-7b	FPFT	44.4
	+*TaD*	**46.1**
LLaMa-13b	FPFT	45.7
	+*TaD*	**47.3**

Table 7: Full-parameter fine-tuning results. 'MC Average' indicates the average values of metrics MC1, MC2, and MC3 on the multiple-choice tasks.

Model	Method	MC Average	
		50%	100%
LLaMa-7b	LoRA	37.9	38.8
	+*TaD*	**39.2**	**39.6**
	Δ	**+1.3**	**+0.8**
LLaMa-13b	LoRA	38.2	39.4
	+*TaD*	**39.7**	**40.5**
	Δ	**+1.5**	**+1.1**

Table 8: *TaD*'s performance with different ratios of training dataset on the multiple-choice tasks. Δ represents the improvement after applying *TaD*. 'MC Average' indicates the average values of metrics MC1, MC2, and MC3 on the multiple-choice tasks.

LLMs with the derived *knowledge vector*, enhancing it with knowledge adaptation learned during fine-tuning. Extensive experiments well demonstrate that *TaD* can consistently gain performance improvement across different LLMs, different fine-tuning methods, and different downstream tasks. And it exhibits superiority over existing approaches. Moreover, our experiments also show that *TaD* has a distinct advantage in data-limited scenarios.

A Experimental Results with Full-Parameter Fine-Tuning

We conduct experiments with full-parameter fine-tuning (FPFT) on the multiple-choice tasks. As shown in Table 7, the experimental results demonstrate that *TaD* achieves comparable and even more significant improvement compared to being applied to PEFT models. This further demonstrates that *TaD* can achieve performance that cannot be obtained with the distribution of training data.

B The Difference in Improvements on the Multiple-Choice Tasks

To investigate the difference in improvements observed in Table 6 across other tasks, we conduct experiments using the LLaMa series models on the multiple-choice tasks with 50% and 100% of the training data, respectively. As shown in Table 8, we observe a consistent pattern with that in Table 6. The performance of *TaD*'s performance is notably enhanced when the training data is reduced.

Model	Method	Multiple Choices			CBQA
		MC1	MC2	MC3	True*Info (%)
GPT-J-6b	LoRA	0.982	1.008	0.996	0.485
	+*TaD*	0.989	0.987	0.813	0.401
	Δ	0.614	0.493	0.477	0.274
LLaMa-7b	LoRA	1.030	1.140	1.057	0.482
	+*TaD*	0.978	0.986	0.746	0.388
	Δ	0.554	0.385	0.382	0.283

Table 9: Standard deviations of GPT-J-6b and LLaMa-7b experiments for multiple-choice and CBQA tasks over 5 runs. Δ represents standard deviations of the improvement after applying *TaD*.

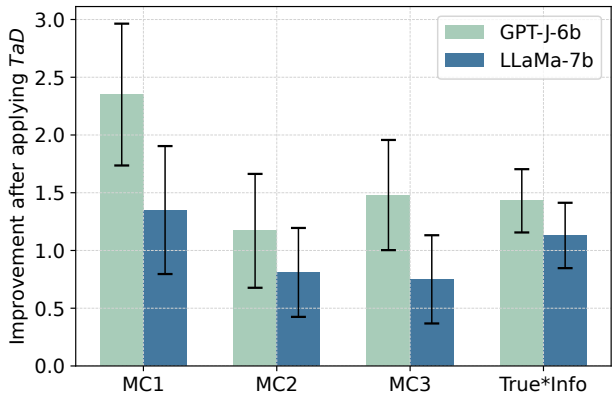

Figure 4: A bar graph of the mean value of the improvement after applying *TaD* to GPT-J-6b and LLaMa-7b with standard deviation error bars.

C Standard Deviations of Experiments

Table 9 shows the standard deviations of the performance of fine-tuned models, *TaD*-enhanced fine-tuned models, and the improvement after applying *TaD* over 5 runs, respectively. Figure 4 correspondingly displays a bar graph of the mean value of the improvement with standard deviation error bars. It is noted that the standard deviation of the improvement is smaller than that of the single model's performance. Concurrently, we observe that although the performance of fine-tuned models and *TaD*-enhanced models varies across different runs, *TaD* guarantees the improvement upon the fine-tuned models' performance. The experimental results above well demonstrate the robustness of our proposed *TaD*.

Acknowledgments

This work was supported by National Natural Science Foundation of China (Nos. 62271281, 61925107, 62021002). It was also sponsored by CAAI-CANN Open Fund, developed on OpenI Community.

References

[Bisk *et al.*, 2020] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reason-

ing about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Chen *et al.*, 2020] Hui Chen, Guiguang Ding, Xudong Liu, Zijia Lin, Ji Liu, and Jungong Han. Imram: Iterative matching with recurrent attention memory for cross-modal image-text retrieval. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12655–12663, 2020.

[Christiano *et al.*, 2017] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[Chuang *et al.*, 2023] Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*, 2023.

[Clark *et al.*, 2019] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[Cobbe *et al.*, 2021] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[Ding *et al.*, 2021] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021.

[Ding *et al.*, 2023] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Yang Zonghan, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5:1–16, 03 2023.

[Fan *et al.*, 2018] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018.

[Gera *et al.*, 2023] Ariel Gera, Roni Friedman, Ofir Arviv, Chulaka Gunasekara, Benjamin Sznajder, Noam Slonim, and Eyal Shnarch. The benefits of bad advice: Autocontrastive decoding across model layers. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10406–10420, Toronto, Canada, July 2023. Association for Computational Linguistics.

[He *et al.*, 2022] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022.

[Hernandez *et al.*, 2023] Evan Hernandez, Belinda Z Li, and Jacob Andreas. Measuring and manipulating knowledge representations in language models. *arXiv preprint arXiv:2304.00740*, 2023.

[Holtzman *et al.*, 2019] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2019.

[Houlsby *et al.*, 2019] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, 2019.

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

[Hu *et al.*, 2023] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276, Singapore, December 2023. Association for Computational Linguistics.

[Kadavath *et al.*, 2022] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[Lester *et al.*, 2021] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November,*

2021, pages 3045–3059. Association for Computational Linguistics, 2021.

[Li and Liang, 2021] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics, 2021.

[Li *et al.*, 2022] Yuchao Li, Fuli Luo, Chuanqi Tan, Mengdi Wang, Songfang Huang, Shen Li, and Junjie Bai. Parameter-effcient sparsity for large language models fine-tuning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 4223–4229. ijcai.org, 2022.

[Li *et al.*, 2023a] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023.

[Li *et al.*, 2023b] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341*, 2023.

[Li *et al.*, 2023c] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada, July 2023. Association for Computational Linguistics.

[Lin *et al.*, 2022] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, 2022.

[Muennighoff *et al.*, 2022] Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*, 2022.

[OpenAI, 2023] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[Ouyang *et al.*, 2022] 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 F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155, 2022.

[Pfeiffer *et al.*, 2020] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online, November 2020. Association for Computational Linguistics.

[Roy and Roth, 2016] Subhro Roy and Dan Roth. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*, 2016.

[Sanh *et al.*, 2022] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

[Subramani *et al.*, 2022] Nishant Subramani, Nivedita Suresh, and Matthew E Peters. Extracting latent steering vectors from pretrained language models. *arXiv preprint arXiv:2205.05124*, 2022.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[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.

[Turner *et al.*, 2023] Alex Turner, Monte M, David Udell, Lisa Theirgart, and Ulisse Mini. Steering gpt-2-xl by adding an activation vector, May 2023.

[Wang and Komatsuzaki, 2021] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

[Wang *et al.*, 2023] Ao Wang, Hui Chen, Zijia Lin, Hengjun Pu, and Guiguang Ding. Repvit: Revisiting mobile cnn from vit perspective. *arXiv preprint arXiv:2307.09283*, 2023.

[Wei *et al.*, 2021] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021.

[Xiong *et al.*, 2024] Yizhe Xiong, Hui Chen, Tianxiang Hao, Zijia Lin, Jungong Han, Yuesong Zhang, Guoxin Wang, Yongjun Bao, and Guiguang Ding. Pyra: Parallel yielding re-activation for training-inference efficient task adaptation. *arXiv preprint arXiv:2403.09192*, 2024.