LoRA-Gen: Specializing Large Language Model via Online LoRA Generation

Yicheng Xiao [1 *] **Lin Song** [2 *] **Rui Yang** [3] **Cheng Cheng** [4] **Yixiao Ge** [2] **Xiu Li** [1 †] **Ying Shan** [2]

Abstract

Recent advances have highlighted the benefits of scaling language models to enhance performance across a wide range of NLP tasks. However, these approaches still face limitations in effectiveness and efficiency when applied to domain-specific tasks, particularly for small edge-side models. We propose the LoRA-Gen framework, which utilizes a large cloud-side model to generate LoRA parameters for edge-side models based on task descriptions. By employing the reparameterization technique, we merge the LoRA parameters into the edge-side model to achieve flexible specialization. Our method facilitates knowledge transfer between models while significantly improving the inference efficiency of the specialized model by reducing the input context length. Without specialized training, LoRA-Gen outperforms conventional LoRA fine-tuning, which achieves competitive accuracy and a 2.1x speedup with TinyLLaMA-1.1B in reasoning tasks. Besides, our method delivers a compression ratio of 10.1x with Gemma-2B on intelligent agent tasks.

1. Introduction

The principle of scaling laws (Kaplan et al., 2020) demonstrates that increasing the size of Large Language Models (LLMs) can significantly improve cross-task generalization. However, due to the constraints of their enormous size, generic LLMs struggle to achieve a good balance between efficiency and effectiveness when addressing domain-specific tasks or preferences. Consequently, research has been shifted towards developing more specialized, compact language models optimized for specific tasks and capable of local deployment on edge devices (Fu et al., 2023; Grangier et al., 2024; Shen et al., 2024). This emerging approach addresses the critical need for more adaptable and resource-efficient AI solutions across academic and industrial domains. Many approaches utilize parameter-efficient fine-tuning techniques (Houlsby et al., 2019; Li & Liang, 2021; Lester et al., 2021; Hu et al., 2021), particularly LoRA (Hu et al., 2021), to train on specific datasets for specialization. However, this method may encounter the issue of catastrophic forgetting, which can result in a decrease in performance on other unseen tasks (Feng et al., 2024; Huang et al., 2023a).

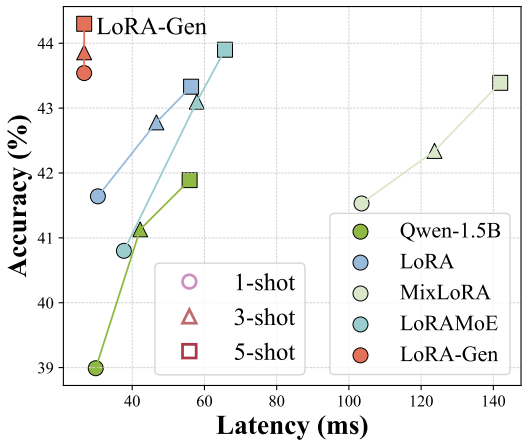
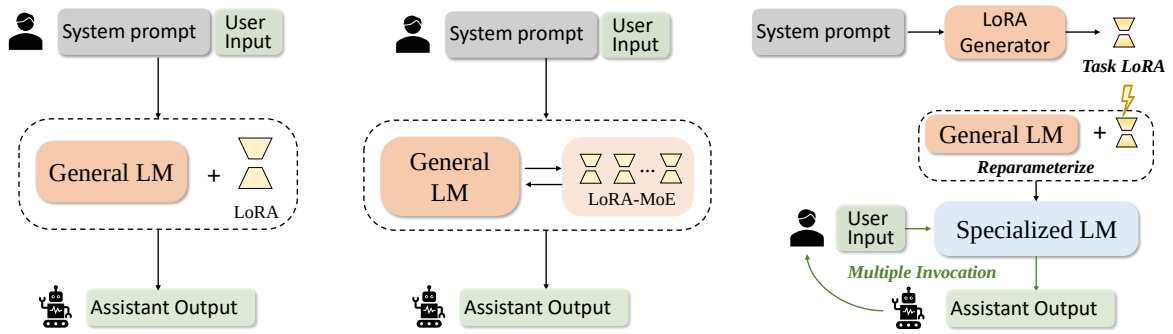


Figure 1. Accuracy-latency curves comparison with various few-shot numbers on ARC-c task. Best view in color. Base model is Qwen-1.5B.

To alleviate knowledge forgetting in specialized training, recent approaches (Dou et al., 2024; Gao et al., 2024a; Yang et al., 2024b; Li et al., 2024a), leverage the flexibility of the Mixture of Experts (MoE) for LoRA training. Specifically, as shown in Figure 2(b), they integrate a group of multiple LoRA components as experts within the language model, allowing the language model to control the selection of LoRA components during token generation. However, these methods introduce additional inference costs due to the extra experts and control units. LoRAHub (Huang et al., 2023b), on the other hand, pretrains a set of task-specific LoRA components and employs a manually designed parameter-free optimization method for selection. Nevertheless, the effectiveness of above mentioned approaches is limited by their model scale, resulting

[*]Equal contribution [1]Tsinghua University [2]ARC Lab, Tencent PCG [3]The University of Hong Kong [4]Xi'an JiaoTong University. Correspondence to: Xiu Li <li.xiu@sz.tsinghua.edu.cn>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.



(a) Vanilla LoRA Paradigm **(b) LoRA-MoE Paradigm** **(c) LoRA-Gen for Specialized LM**

Figure 2. Comparison of different LoRA-based fine-tuning strategies.(a) Vanilla LoRA is fine-tuned on the target task and then merged into the source model. (b) LoRA-MoE introduces additional LoRA experts to improve the generalization performance. (c) Our LoRA-Gen presents a task-specific LoRA generator that customizes a specialized LM for edge-side users.

Method	Context Compression for Unseen Tasks (Fast Inference)	Reparameterized Model (w/o Additional Params.)	Training Free for Unseen Tasks (High Flexibility)	Knowledge Transfer (High Accuracy)
ICL (Dong et al., 2022)	✗	✗	✔	✗
LoRA (Hu et al., 2021)	✗	✔	✗	✗
LoRA-MoE (Dou et al., 2024)	✗	✗	✔	✗
LoraHub (Huang et al., 2023b)	✗	✔	✗	✗
AutoCompressors (Chevalier et al., 2023)	✔	✗	✔	✗
LoRA-Gen	✔	✔	✔	✔

Table 1. Characteristics comparison with other counterparts. ICL indicates the in-context learning.

in constrained performance and generalization capabilities on unseen tasks. Therefore, this paper explores a new perspective: *utilizing a large cloud-side model to generate parameters for a smaller edge-side model to achieve better specialization.*

To achieve it, we propose a new LoRA generation framework, termed LoRA-Gen. As shown in Figure 2(c), our method can be divided into two parts: Online LoRA generation and Specialized LM. The former is used to generate LoRA parameters based on the task-defined system prompt, while the latter facilitates efficient batch inference for user input. Specifically, a fine-tuned large language model and a mixture of LoRA experts are deployed in the cloud. The cloud-side language model generates a set of meta tokens based on the given system prompt. Each meta token corresponds to a transformer layer in the edge-side language model, utilizing these tokens to control the composition of parameters from the LoRA experts. Similarly to vanilla LoRA, the combined parameters are further merged into the edge-side LM through reparameterization, resulting in an efficient specialized model.

As shown in Table 1, our LoRA-Gen offers four advantages over previous methods: i) Context compression for unseen tasks: LoRA-Gen dynamically compresses the task-specific system prompt ($e.g.$, task descriptions, few-shot samples, and chat templates) into the LoRA weights, which significantly reduces the context length for the specialized models. ii) Reparameterized model: Unlike LoRA-MoE (Dou et al., 2024), our approach employs reparameterization techniques to merge the generated LoRA weights into the original parameters, thereby avoiding additional inference costs. iii) Training free for unseen tasks: Our method does not require any additional training, including few-shot tuning, when specializing the model for unseen tasks. It only necessitates a single-turn inference on the system prompt to obtain the specialized model parameters, which simplifies model deployment. iv) Knowledge Transfer: LoRA-Gen allows the cloud side and edge side to utilize different models, enabling the injection of knowledge from the large cloud model into the edge model through reparameterization, which enhances performance effectively as shown in Figure 1.

We conduct extensive experiments to validate the effectiveness of LoRA-Gen on various commonsense reasoning tasks as well as an agent benchmark. The results demonstrate that our method balances both performance and efficiency, showing significant advantages across eight language datasets. For the edge-side model of TinyLLaMA-1.1B, LoRA-Gen outperforms vanilla LoRA fine-tuning

by a remarkable margin with only 16% sequence length, $+1.3\%$ on harmonic-mean of accuracy, and 2.1x speedup. Moreover, for the Gemma-2B model, LoRA-Gen demonstrates competitive performance on unseen agent tasks. Additionally, since it does not require the input of agent definitions during inference, it achieves a remarkable 10.1x compression ratio.

2. Related Work

2.1. Parameter-Efficient Fine-Tuning

Given the billions of parameters in LLMs and the limitations of current hardware, fully fine-tuning LLMs in the traditional manner is often impractical. To address this, several parameter-efficient fine-tuning (PEFT) methods have been developed. Adapter-based approaches (Mahabadi et al., 2021; Zhou et al., 2024b; Zhang et al., 2024) involve inserting trainable adapter layers into various blocks of pre-trained models. Soft prompt methods (Li & Liang, 2021; Liu et al., 2022) adjust a small trainable prefix vector to adapt LLMs to new tasks. Unlike these methods, LoRA (Hu et al., 2021) minimizes the number of trainable parameters for downstream tasks by freezing the pre-trained models and tuning only additional rank decomposition layers. This method approximates weight adjustments during fine-tuning without incurring extra costs during inference. Building on this, AdaLoRA (Zhang et al., 2023) dynamically adjusts the parameter budget among weight matrices, while DoRA (Liu et al., 2024c) fine-tunes both the magnitude and directional components decomposed from pre-trained weights. VeRA (Kopiczko et al., 2024) further reduces the number of trainable parameters by utilizing shared low-rank layers and learnable scaling vectors.

2.2. LoRA Meets Mixture of Experts

Leveraging its lightweight nature, LoRA is utilized in Mixture of Experts (MoE) architectures to enhance performance. MoLoRA (Zadouri et al., 2023) incorporates LoRA adapters as experts on top of pre-trained models and uses a router layer to integrate these experts. MOELoRA (Liu et al., 2024b) applies this framework to various medical domain tasks, though it requires task type input for the router. LoRAMoE (Dou et al., 2024) introduces multiple LoRA experts into the feed-forward block to mitigate knowledge forgetting during the instruction-tuning phase. LoraHub (Huang et al., 2023b) allows a dynamic assembling of LoRA modules on various tasks and even unseen tasks by combining adapted LoRA modules. Additionally, MoLA (Gao et al., 2024a) proposes layer-specific experts, allocating a varying number of LoRA experts to different layers to boost performance.

2.3. Context Compression

With the rise of in-context learning (Wei et al., 2022) and agentic pipelines (Yang et al., 2024a), LLMs often need to process thousands of tokens, potentially exceeding their maximum context length. Unlike methods that extend the context window of LLMs, context compression offers an efficient way to reduce the input prompt length. There are two primary methods of context compression: hard prompt and soft prompt. Selective-Context (Li, 2023) and Jiang et al. (2023) exemplify hard prompt methods by removing low-information content at the lexical level (e.g., sentences, words, or tokens) to shorten the prompt. On the other hand, gisting (Mu et al., 2023), AutoCompressors (Chevalier et al., 2023), ICAE (Ge et al., 2024), and 500xCompressor (Li et al., 2024b) represent soft prompt methods that compress input prompts into a small number of special tokens. In contrast to these approaches, we propose compressing the context into rank-decomposition layers using LoRA methods.

3. Methodology

In this section, we first review LoRA-based Mixture of Experts fine-tuning paradigm and then elaborate on our LoRA-Gen, which generates task-specific LoRA weights according to the system prompt for edge-side language models.

3.1. Revisiting Mixture of LoRA Experts

LoRA (Hu et al., 2021) improves the efficiency of fine-tuning by significantly reducing the number of trainable parameters. Formally, it updates the weight matrix $W \in \mathbb{R}^{d' \times d''}$ by using a low-rank approximation via two decomposition matrices $A \in \mathbb{R}^{d' \times r}$ and $B \in \mathbb{R}^{r \times d''}$ with a low rank r ($r \ll min(d', d'')$) as follow:

$$\widetilde{W} = W + AB. \tag{1}$$

Trainable low-rank decomposition matrices can capture the underlying patterns of downstream tasks under the guidance of the task-specific direction (Hu et al., 2021). Moreover, another effective approach, the Mixture of Experts (MoE) (Jacobs et al., 1991; Jordan & Jacobs, 1994), treats multiple networks as experts and seeks to take advantage of their strengths in a hybrid framework. This method aims to combine the advantages of different models, resulting in improved generalization and overall performance. Typically, a MoE layer consists of n experts, denoted as $\{E_i\}_{i=1}^{n}$ with a router R as the gate for expert allocation. Given hidden states $\{h_j\}_{j=1}^{s}$ of a sequence with the length of s, the output of the MoE can be formulated as:

$$h'_j = \sum_{i=1}^{n} R_i(h_j) E_i(h_j) \tag{2}$$

Considering the efficiency of LoRA and the strong performance of MoE, (Li et al., 2024a; Dou et al., 2024; Gao et al., 2024a; Yang et al., 2024b) integrate LoRA into the MoE plugin, boosting the fine-tuning performance by utilizing a mixture of LoRA experts, effectively blending the strengths of both methods.

3.2. Online LoRA Generation

Overview. The mixture of LoRA experts has showcased reasonable performance in fine-tuning for specific tasks. However, there remains a gap in its effectiveness for multi-task learning and the generalization to unseen tasks. Additionally, most LoRA-MoE (Li et al., 2024a; Dou et al., 2024) methods require calculating the expert routing for each token individually, which significantly increases the computational complexity. To address these challenges, we propose a new framework, termed LoRA-Gen that generates task-aware LoRA via an online large language model with system prompts (including few-shot samples, task description, role specification and the conversation format) as presented in Figure 3. In the following, we elaborate on our LoRA generation method and the reparameterization of the edge-side language model.

Cloud-side LM & Meta Token. In adherence to meta-learning (Hospedales et al., 2021; Finn et al., 2017), we construct a unified representation of the task-related information to achieve generalization capabilities for various tasks, relying on cloud-side LM to facilitate this process. Specifically, given a series of few-shot samples or task-specific system prompts, the cloud-side LM appends L special tokens $\langle meta \rangle$ behind them and transfers the inherent knowledge into these tokens with causal masks in a single forward pass. We define these tokens as meta tokens $\{T_i^{meta}\}_{i=1}^{L}$, where L represents the number of layers of the edge-side language model. Each meta token is associated with a transformer layer in the edge-side LM.

LoRA Expert Pool. Our initial attempt is to generate LoRA parameters directly through a continuous projection on the meta token. However, the expansive parameter space poses optimization challenges, making the model susceptible to overfitting and hindering generalization, whose analysis refers to Table 9. Therefore, similar to the previous works (Dou et al., 2024), we adopt an alternative solution by introducing the discrete MoE mechanism. Specifically, as shown in Figure 3, we construct a LoRA expert pool of n experts, whose weights are defined as $\{E_i\}_{i=1}^{n}$. Each LoRA expert contains three LoRA blocks, corresponding to the gate linear layer, up linear layer and down linear layer in FFN of the edge-side model, respectively. Different from the LoRAHub (Huang et al., 2023b), these experts are trained in an end-to-end manner.

Routing Module. To control the composition of experts, we propose a routing module using meta tokens. Unlike the token-wise LoRA-MoE (Dou et al., 2024), our MoE is layer-wise. We apply an individual MoE for each transformer layer in the edge-side LM, and all tokens in a sequence use the same composition. For simplicity, the routing module consists of two linear projections with a Batch Normalization (BN) layer. Incorporating a BN layer can further increase the diversity of router output, promoting the utilization of a wider range of experts. In formal, the router $R^i \in \mathbb{R}^n$ of i-th layer of edge-side LM can be formulated as:

$$R^i = \text{BN}(f_2 \circ \varsigma \circ f_1(T_i^{meta})), \tag{3}$$

where f_1, f_2 are the linear transform and ς denotes the SiLU (Elfwing et al., 2018) activation function. We attempt to increase selection randomness and balance expert loads, by using Gumbel-Softmax (Jang et al., 2016), which can be formulated as:

$$\text{Gumbel-Softmax}(R_t^i) = \frac{e^{R_t^i+g}}{\sum_{j=1}^{n} e^{R_j^i+g}}, \tag{4}$$

$$\text{where } g \sim \text{Gumbel}(0,1). \tag{5}$$

Nevertheless, the Gumbel-softmax strategy shows a significant reduction in generalization performance, which is reported in experiments of Section 4.4, To this end, following (Li et al., 2024a; Dou et al., 2024), we adopt a KeepTOP-K strategy to select experts in a deterministic manner:

$$G_t^i = \begin{cases} \frac{\widetilde{R_t^i}}{\sum_{j=1}^{K} \overline{R_j^i}} & \widetilde{R_t^i} \in \text{TOP-K}(\widetilde{R^i}) \\ 0 & \text{else} \end{cases}, \tag{6}$$

$$\text{where TOP-K}(\widetilde{R^i}) = \{\overline{R_t^i}\}_{t=1}^{K}, \widetilde{R_t^i} = \frac{e^{R_t^i}}{\sum_{j=1}^{n} e^{R_j^i}}, \tag{7}$$

where G_t^i represents the the gate score of t-th experts for i-th decoder layer of the edge-side language model. Consequently, we generate task-specific LoRA weights as:

$$\theta^i = \sum_{j=1}^{n} G^i E_j. \tag{8}$$

where the θ^i indicates the generated LoRA weights for i-th decoder layer.

Reparametrization. As the same as LoRA, we use the reparameterization strategy to merge the generated LoRA parameters into the FFN layers of the edge-side model. In contrast to the LoRA-MoE, our method is cost-free during inference, which needs no additional components in the specialized edge-side LM.

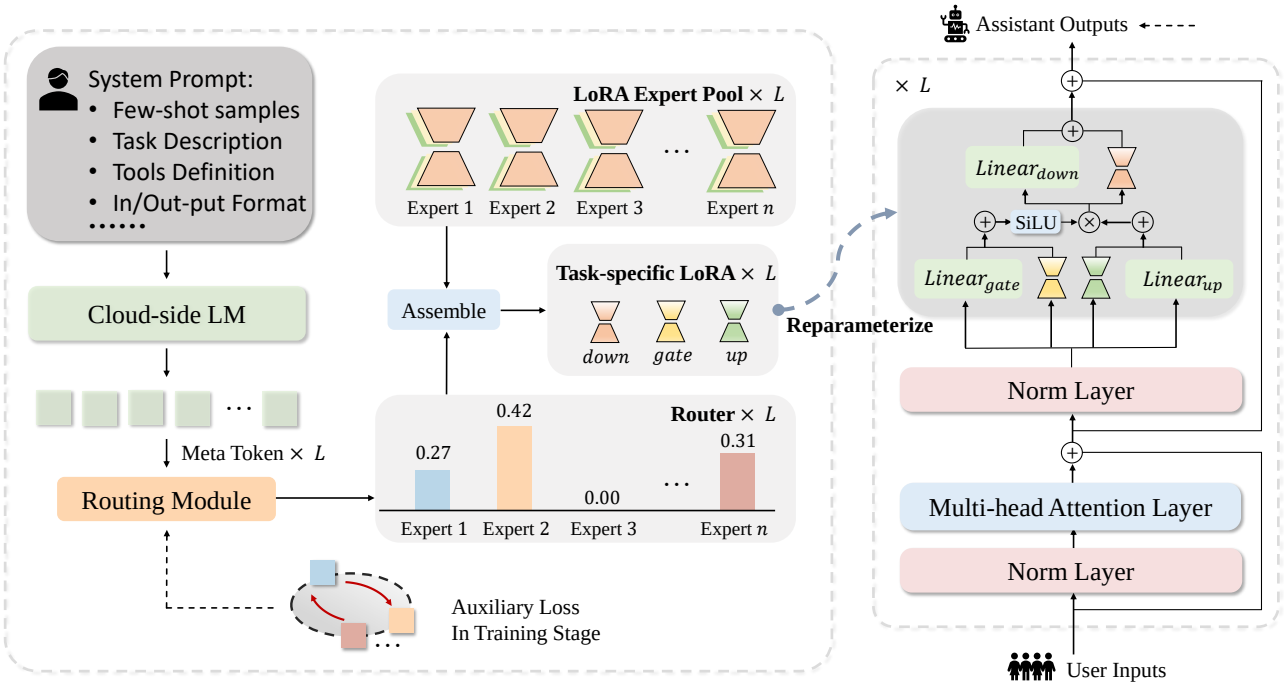


Figure 3. Overview of our proposed LoRA-Gen. Given the system prompts by users, a large language model first generates meta tokens autoregressively. With a routing module, we obtain the gates of all experts in the online LoRA pool. After assembling, we produce the specialized LoRA in the cloud side and deploy it to the edge-side language model by merging the LoRA weights.

3.3. Training Target

Auxiliary Loss. Balanced load of MoE structure is essential for capability of generalization and stability (Jacobs et al., 1991). Without constraints, the routing module tends to select a fixed small set of experts, leaving other experts unused and causing load imbalance. To mitigate this issue, we introduce a soft constraint with the coefficient of variation as the auxiliary loss, encouraging a more balanced usage of the available experts. Formally, the constraint can be formulated as:

$$\mathcal{L}_{cv} = \alpha(\frac{\sigma(G)}{\mu(G)})^2, \tag{9}$$

where σ and μ represent the standard deviation and mean of the gates assigned to each expert within a batch, separately. The coefficient α is to balance the auxiliary objective and the main objective.

Total Loss. The total loss is consist of the language modeling loss and auxiliary loss as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{cv} + \mathcal{L}_{LM}, \tag{10}$$

where $\mathcal{L}_{LM}$ is the Cross Entropy loss of language modeling in causal LMs.

4. Experiments

We conduct extensive experiments to evaluate the effectiveness of our LoRA-Gen and compare it to the widely adopted LoRA-based fine-tuning method on commonsense reasoning tasks in a fair experimental setting. Furthermore, we assess the generalization capacity and system prompt compression performance of LoRA-Gen on an agent dataset, GPT4Tools (Yang et al., 2024a).

4.1. Datasets and Metrics.

Reasoning Tasks. Following (Dou et al., 2024; Li et al., 2024a), we select eight widely-used benchmarks to assess the reasoning ability of LoRA-Gen across various knowledge domains ranging from natural science to daily life. One classification task: BoolQ (Clark et al., 2019). Five question-answering tasks: ARC-c (Clark et al., 2018), ARC-e (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020) and SocialQA (Sap et al., 2019). One science completion task: Hellaswag (Zellers et al., 2019) and a fill-in-the-blank task: Winogrande (Sakaguchi et al., 2020).

Agent Dataset. We utilize the GPT4Tools (Yang et al., 2024a) which provides a benchmark to evaluate the ability

of LLM to use tools, to assess the effectiveness of LoRA-Gen in the deployment of intelligent agents. GPT4Tools constructs a tool-related instructional dataset, including positive samples, negative samples, and context samples. It consists of 71k instruction-response pairs with 21 tools in the training set and 652 items in the test set with 8 novel tools absent from the training set.

Metrics. The performance of all commonsense reasoning benchmarks is measured with the accuracy metric in all datasets. To further evaluate the performance in multi-task learning, we utilize two metrics: the average accuracy (AVE.) and the harmonic mean (HAR.) of all results. For GPT4Tools, we measure the performance of method from five aspect: successful rate of thought (SR_t), successful rate of action (SR_{act}), successful rate of arguments (SR_{args}), successful Rate (SR) and IoU according to (Yang et al., 2024a).

4.2. Implementation Details

We deploy LLaMA3-8B (Grattafiori et al., 2024) as the cloud-side LM during online task-specific LoRA parameters generation. We finetune the q and v projection layers of the LLM with a LoRA adapter. The number of experts is 8 and we set K in the routing function TOP-K to 2 by default. The coefficient α for auxiliary loss $\mathcal{L}_{cv}$ is set 0.01. All the latencies are measured on the same GPU with 40GB of memory. More details can be viewed in the Appendix.

4.3. Main results

Reasoning Tasks. We first evaluate the performance of LoRA-Gen in the reasoning scenario as shown in Table 2. We divide eight commonly used datasets into two parts, one as the multi-task learning set, including ARC-c, ARC-e, OpenBookQA, BoolQ, SocialQA and the other as an unseen test set, including Hellaswag, Winogrande and PIQA. We randomly sample to construct multi-shot training data. As shown in Table 2, LoRA-Gen consistently achieves comparable performance while exhibiting lower latency compared to other fine-tuning methods across various backbone models. Additionally, As shown in Table 4, based on the same LLM, our method achieves absolute gains of 1.5% over AutoCompressors (Chevalier et al., 2023), while maintaining much higher efficiency. The results underscore the advantage of using LoRA-Gen, which balances effectiveness and efficiency across both seen and unseen tasks.

Intelligent Agent Scenario. We evaluate the performance of LoRA-Gen with edge-side model Gemma-2B on the GPT4Tools benchmark (Yang et al., 2024a). The results in Table 3 present a comparison of successful rates, intersection-over-union (IoU), average performance, and compression ratio (speedup). One key advantage of LoRA-Gen is to compress the tools definition within the system prompt into the generated LoRA parameters via a single-turn inference. It significantly reduces the context length with a compression ratio of 10.1x, which maintains comparable performance of 91.5% average score. On the other hand, our method without training on GPT4Tools boosts original Gemma-2B by 4.9% in average score, which shows the effective generalization of our method. In contrast, removing the tool definitions in the vanilla LoRA setting leads to a marked reduction in performance (SR: -26.1%, IoU: -9.7%). Furthermore, benefiting from knowledge injection from the cloud-side language model, it surpasses the baseline by 3.1 points while maintaining a 10.1x compression ratio. The results highlight the strengths of LoRA-Gen in effectiveness and efficiency, attributed to its inference-time specialization and generalization ability to unseen tools, making it well-suited for tasks with extensive prefix descriptions.

4.4. Ablation study

Number of Experts in Online Expert Pool. As shown in Table 5, we present the performance of different numbers of experts in the cloud-side LoRA pool. Performance generally improves with an increasing number of experts. With 4 experts, the AVE. is 56.4%, and the HAR. is 52.3 %. Increasing the number of experts to 12 yields slight improvements, with the AVE. rising to 57.3% and the HAR. to 53.1%. However, the best performance is achieved with 8 experts, where both AVE. (58.7%) and HAR. (53.6%) reach their peak values. This may indicate that 8 experts strike the best balance between multi-task learning and unseen generalization.

Effectiveness of Balanced Load Strategy. Ensuring a balanced load of experts can significantly improve the robustness and stability of the model. We initially conduct an ablation study to assess the impact of the absence of auxiliary losses on model performance. Without the auxiliary loss, the AVE. decreases by 1.2 points. Subsequently, we summarize the impact of different values of the coefficient for auxiliary loss as shown in Table 6. As the auxiliary loss coefficient decreases, a significant improvement in both performance metrics is observed. Reducing the coefficient from 0.1 to 0.01 yields further gains, resulting in an average (AVE) of 58.7% and a harmonic mean (HAR) of 53.6%, thereby achieving an optimal balance between the auxiliary strategy and the primary objective function. In addition, we investigate the strategy of the router function. As illustrated in Table 8, we compare two routing strategies employed for online experts within the cloud-side LoRA pool. Compared to Gumbel-softmax, KeepTOP-K strategy exhibits a notable improvement, attaining an AVE of

Method	Seen Tasks					Unseen Tasks			AVE. ↑	HAR. ↑	Latency (ms) ↓
	ARC-c	ARC-e	OBQA	BoolQ	SIQA	HellaS	WinoG	PIQA			
TinyLlaMA-1.1B	34.2	66.9	27.4	58.8	46.0	**45.8**	60.7	73.9	51.7	46.7	44.5
+LoRA	33.6	67.6	28.6	71.9	51.5	44.5	61.9	**75.1**	54.3	48.5	44.5
+LoRAMoE	35.2	68.8	28.6	73.2	**52.1**	45.4	62.0	74.1	54.9	49.3	55.9
+MixLoRA	33.5	67.7	28.4	73.3	51.4	44.9	62.3	74.6	54.5	48.6	100.1
+LoRA-Gen	**35.8**	**69.1**	**30.4**	**73.6**	49.6	45.5	**62.6**	74.1	**55.1**	**49.8**	**21.2**
Qwen-1.5B	41.9	73.1	29.0	73.3	50.6	49.0	65.3	76.2	57.3	51.9	56.3
+LoRA	43.3	73.9	31.2	77.6	**54.9**	48.8	66.5	76.9	59.1	53.9	56.3
+LoRAMoE	43.9	73.7	29.8	77.3	53.4	48.7	66.3	76.9	58.8	53.2	65.7
+MixLoRA	43.4	73.8	31.8	78.2	54.6	48.9	66.4	76.5	59.2	54.2	141.9
+LoRA-Gen	**44.3**	**74.3**	**33.4**	**79.6**	53.6	**49.1**	**67.4**	**76.9**	**59.8**	**55.0**	**26.7**
Gemma-2B	50.3	81.5	33.8	73.4	49.3	55.6	71.5	78.7	61.8	57.0	87.3
+LoRA	49.9	78.2	36.0	**80.9**	56.8	55.4	71.7	79.2	63.5	59.2	87.3
+LoRAMoE	50.9	**82.0**	38.8	78.4	55.2	54.0	**72.9**	79.3	63.9	60.0	101.8
+MixLoRA	**52.3**	79.4	38.6	75.6	**59.1**	54.1	72.7	78.2	63.8	60.2	177.7
+LoRA-Gen	51.2	81.9	**39.0**	76.2	55.6	**56.0**	71.6	**79.5**	**63.9**	**60.2**	**36.1**

Table 2. Comparison of the performance with 5-shot samples on various commonsense reasoning benchmarks. Seen tasks indicate that the datasets are part of the training set, while unseen tasks are not. AVE denotes the average accuracy of 8 tasks while HAR is the harmonic mean. The latency scores of various methods are all calculated on ARC-c. Latency is measured on a Nvidia A100 GPU.

Method	W/ Training	W/ Tools Definiton	SR_t	SR_{act}	SR_{args}	SR	IoU	Average Score ↑	Compress Ratio ↑
Gemma-2B	✗	✔	86.3	77.6	77.7	65.0	89.7	79.3	1x
+LoRA	✔	✔	**99.4**	79.6	**93.8**	78.2	91.0	88.4	
+LoRA	✔	✗	98.0	60.9	83.2	52.1	81.3	75.1	
+LoRA-Gen	✗	✗	94.1	86.8	79.7	73.3	86.9	84.2	10.1x
+LoRA-Gen	✔	✗	98.6	**88.0**	93.4	**84.0**	**93.6**	**91.5**	

Table 3. Performance of different fine-tuning strategies with Gemma-2B (Team et al., 2024) on test set of GPT4Tools (Yang et al., 2024a). W/ Training denotes Gemma-2B is fine-tuning on the training set of GPT4Tools with vanilla LoRA or our LoRA-Gen. Gray rows indicate scenarios where the system prompt does not contain tools definitions, typically constituting 91% of the input context.

Method	HellaS	WinoG	PIQA	AVE. ↑	Latency (ms)↓
AutoCompressors	44.7	62.4	73.3	60.1	11.4ms
LoRA-Gen	**46.3**	**63.7**	**74.9**	**61.6**	**7.54ms**

Table 4. Comparison with AutoCompressors (Chevalier et al., 2023) in unseen tasks based on OPT-2.7B.

58. 7% and a HAR of 53. 6%. We consider that an overabundance of randomness may affect expert ability to learn specific tasks during the optimization process.

Effectiveness of Meta Token. We attempt to utilize the cloud-side large language model to generate LoRA parameters in a single forward pass directly instead of meta tokens. Specifically, we directly transform the output tokens of LLM to the LoRA weights space with a feedforward neural network and get the i-th layer generated LoRA weights $\in \mathbb{R}^{3\times2\times d\times r}$, where d is the hidden dimension and r denotes the low rank of LoRA. As indicated by the experimental results in Table 9, this approach exhibits comparable performance to that achieved through meta tokens on the seen tasks, while the results on the unseen tasks are significantly lower than those obtained with meta tokens, trailing by 11.1%. Generating LoRA parameters directly leads to pronounced overfitting to the training domain, caused by the large parameter space, thereby limiting its ability to generalize to unseen tasks.

Effectiveness of Knowledge Transfer. As depicted in Table 7, we compare the performance of the baseline model and our LoRA-Gen across different few-shot samples. Remarkably, LoRA-Gen with just a 1-shot sample surpasses the baseline with 5-shot samples by 3.5% on HAR. We attribute this to the use of LLaMA3-8B (Grattafiori et al., 2024) as the cloud model, which transfers a portion of its knowledge to the edge-side language model via reparameterization.

4.5. Qualitative Study in Agent Scenario

We deploy LoRA-Gen within Gemma-2B and conduct case studies and visualizations. As illustrated in Figure 4, LoRA-Gen removes the 26 tools description from input

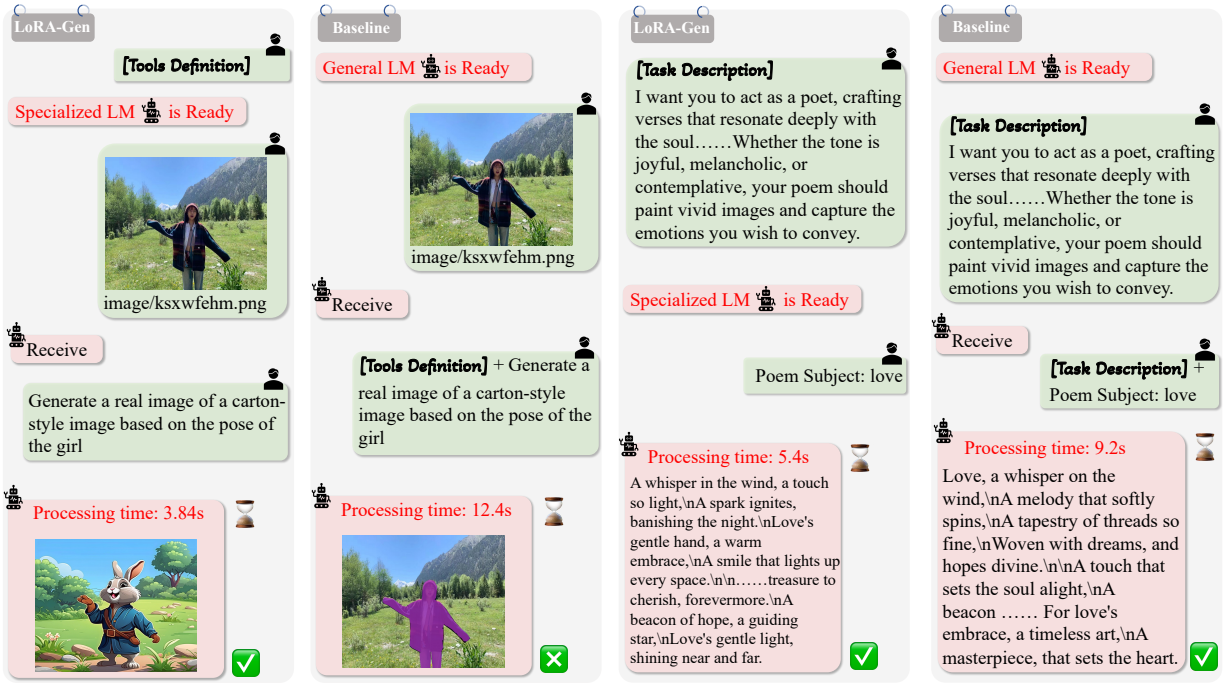


Figure 4. Visualization comparison between LoRA-Gen and baseline, Gemma-2B (Team et al., 2024). LoRA-Gen compresses the tools definition and task description into the generated LoRA parameters, effectively specializing the language model to reduce processing times while maintaining comparable performance. The detailed LM outputs and system prompt can be accessed in the Appendix.

Number	AVE.	HAR.
4	56.4	52.3
12	57.3	53.1
8	**58.7**	**53.6**

Table 5. Number of Experts in LoRA pool.

Coefficient	AVE.	HAR.
0.1	57.1	52.0
0.005	56.8	50.5
0.01	**58.7**	**53.6**

Table 6. Coefficient of auxiliary loss.

Few-shot	AVE.	HAR.
3-shot†	55.5	49.3
5-shot†	56.0	49.9
1-shot‡	**58.7**	**53.6**

Table 7. † is baseline and ‡ indicates LoRA-Gen.

Strategy	AVE.	HAR.
GumbleTOP-K	56.4	52.3
KeepTOP-K	**58.7**	**53.6**

Table 8. Routing strategy for online experts.

LoRA Generation	Seen AVE.	Unseen AVE.
Direct	52.4	61.0
Meta-Token	52.0	72.1

Table 9. Different LoRA generation manner.

of the model, significantly reducing inference time and achieving a 3.2x speedup compared to the baseline. The limited generalization of the baseline model results in incorrect tool selection, thereby highlighting the effectiveness of our method. Additionally, in the open text generation scenario, LoRA-Gen accelerates reasoning time by compressing the task definition while achieving comparable results. The corresponding generation results are detailed in the appendix.

5. Conclusion

In this paper, we propose an online LoRA generation framework, called LoRA-Gen, which utilizes a cloud-side language model to generate task-specific LoRA parameters for edge-side models. Our strategy offers four advantages over previous methods: context compression for unseen tasks, a reparameterized language model, inference-time specialization, and knowledge transfer. Extensive experiments show that LoRA-Gen achieves competitive results and an impressive speedup on common-sense reasoning tasks. Additionally, our method achieves a compression ratio of 10.1x on zero-shot agent tasks, indicating its potential applicability to more scenarios. We believe our methodological approach can inspire future LLM-based research.

Acknowledgement

This work was partly supported by Shenzhen Key Laboratory of next generation interactive media innovative technology (No:ZDSYS20210623092001004) and National Natural Science Foundation of China (No.62293544, 62425117).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bai, J., Bai, S., Yang, S., Wang, S., Tan, S., Wang, P., Lin, J., Zhou, C., and Zhou, J. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 2023.

Bisk, Y., Zellers, R., Le bras, R., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7432–7439, Jun 2020.

Brooks, T., Holynski, A., and Efros, A. A. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18392–18402, 2023.

Chevalier, A., Wettig, A., Ajith, A., and Chen, D. Adapting language models to compress contexts. In *EMNLP*, pp. 3829–3846. Association for Computational Linguistics, 2023.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North*, Jan 2019.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., and Sui, Z. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

Dou, S., Zhou, E., Liu, Y., Gao, S., Shen, W., Xiong, L., Zhou, Y., Wang, X., Xi, Z., Fan, X., et al. Loramoe: Alleviating world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1932–1945, 2024.

Elfwing, S., Uchibe, E., and Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

Feng, W., Hao, C., Zhang, Y., Han, Y., and Wang, H. Mixture-of-loras: An efficient multitask tuning for large language models. *arXiv preprint arXiv:2403.03432*, 2024.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Fu, Y., Peng, H., Ou, L., Sabharwal, A., and Khot, T. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pp. 10421–10430. PMLR, 2023.

Gao, C., Chen, K., Rao, J., Sun, B., Liu, R., Peng, D., Zhang, Y., Guo, X., Yang, J., and Subrahmanian, V. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562*, 2024a.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 07 2024b. URL `https://zenodo.org/records/12608602`.

Ge, T., Hu, J., Wang, L., Wang, X., Chen, S., and Wei, F. In-context autoencoder for context compression in a large language model. In *ICLR*. OpenReview.net, 2024.

Grangier, D., Katharopoulos, A., Ablin, P., and Hannun, A. Specialized language models with cheap inference from limited domain data. *arXiv preprint arXiv:2402.01093*, 2024.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44 (9):5149–5169, 2021.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Huang, C., Liu, Q., Lin, B. Y., Pang, T., Du, C., and Lin, M. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*, 2023a.

Huang, C., Liu, Q., Lin, B. Y., Pang, T., Du, C., and Lin, M. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*, 2023b.

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

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*, 2023.

Jordan, M. I. and Jacobs, R. A. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2): 181–214, 1994.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kopiczko, D. J., Blankevoort, T., and Asano, Y. M. Vera: Vector-based random matrix adaptation. In *ICLR*. OpenReview.net, 2024.

Labs, B. F. Flux. `https://github.com/black-forest-labs/flux`, 2023.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Li, D., Ma, Y., Wang, N., Cheng, Z., Duan, L., Zuo, J., Yang, C., and Tang, M. Mixlora: Enhancing large language models fine-tuning with lora based mixture of experts. *arXiv preprint arXiv:2404.15159*, 2024a.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Li, Y. Unlocking context constraints of llms: Enhancing context efficiency of llms with self-information-based content filtering. *arXiv preprint arXiv:2304.12102*, 2023.

Li, Z., Su, Y., and Collier, N. 500xcompressor: Generalized prompt compression for large language models. *arXiv preprint arXiv:2408.03094*, 2024b.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *NeurIPS*, 2022.

Liu, H., Yuan, Y., Liu, X., Mei, X., Kong, Q., Tian, Q., Wang, Y., Wang, W., Wang, Y., and Plumbley, M. D. Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024a.

Liu, Q., Wu, X., Zhao, X., Zhu, Y., Xu, D., Tian, F., and Zheng, Y. When MOE meets llms: Parameter efficient fine-tuning for multi-task medical applications. In *SIGIR*, pp. 1104–1114. ACM, 2024b.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024c.

Mahabadi, R. K., Henderson, J., and Ruder, S. Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*, pp. 1022–1035, 2021.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Jan 2018.

Mu, J., Li, X., and Goodman, N. D. Learning to compress prompts with gist tokens. In *NeurIPS*, 2023.

Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., and Rombach, R. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

Sakaguchi, K., Le Bras, R., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 8732–8740, Jun 2020.

Sap, M., Rashkin, H., Chen, D., LeBras, R., and Choi, Y. Socialiqa: Commonsense reasoning about social interactions. *Cornell University - arXiv,Cornell University - arXiv*, Apr 2019.

Shen, J., Tenenholtz, N., Hall, J. B., Alvarez-Melis, D., and Fusi, N. Tag-llm: Repurposing general-purpose llms for specialized domains. *arXiv preprint arXiv:2402.05140*, 2024.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Wang, J., Pu, J., Qi, Z., Guo, J., Ma, Y., Huang, N., Chen, Y., Li, X., and Shan, Y. Taming rectified flow for inversion and editing. *arXiv preprint arXiv:2411.04746*, 2024.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Xiao, Y., Luo, Z., Liu, Y., Ma, Y., Bian, H., Ji, Y., Yang, Y., and Li, X. Bridging the gap: A unified video comprehension framework for moment retrieval and highlight detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18709–18719, 2024a.

Xiao, Y., Song, L., Wang, J., Song, S., Ge, Y., Li, X., Shan, Y., et al. Mambatree: Tree topology is all you need in state space model. *Advances in Neural Information Processing Systems*, 37:75329–75354, 2024b.

Xiao, Y., Song, L., Chen, Y., Luo, Y., Chen, Y., Gan, Y., Huang, W., Li, X., Qi, X., and Shan, Y. Mindomni: Unleashing reasoning generation in vision language models with rgpo. *arXiv preprint arXiv:2505.13031*, 2025a.

Xiao, Y., Song, L., Yang, R., Cheng, C., Xu, Z., Zhang, Z., Ge, Y., Li, X., and Shan, Y. Haploomni: Unified single transformer for multimodal video understanding and generation. *arXiv preprint arXiv:2506.02975*, 2025b.

Yang, R., Song, L., Li, Y., Zhao, S., Ge, Y., Li, X., and Shan, Y. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36, 2024a.

Yang, R., Song, L., Xiao, Y., Huang, R., Ge, Y., Shan, Y., and Zhao, H. Haplovl: A single-transformer baseline for multi-modal understanding. *arXiv preprint arXiv:2503.14694*, 2025.

Yang, S., Ali, M. A., Wang, C.-L., Hu, L., and Wang, D. Moral: Moe augmented lora for llms' lifelong learning. *arXiv preprint arXiv:2402.11260*, 2024b.

Zadouri, T., Üstün, A., Ahmadian, A., Ermiş, B., Locatelli, A., and Hooker, S. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Jan 2019.

Zhang, Q., Chen, M., Bukharin, A., Karampatziakis, N., He, P., Cheng, Y., Chen, W., and Zhao, T. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

Zhang, R., Han, J., Liu, C., Zhou, A., Lu, P., Qiao, Y., Li, H., and Gao, P. Llama-adapter: Efficient fine-tuning of large language models with zero-initialized attention. In *ICLR*. OpenReview.net, 2024.

Zhou, C., Yu, L., Babu, A., Tirumala, K., Yasunaga, M., Shamis, L., Kahn, J., Ma, X., Zettlemoyer, L., and Levy, O. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039*, 2024a.

Zhou, H., Tang, L., Yang, R., Qin, G., Zhang, Y., Hu, R., and Li, X. Uniqa: Unified vision-language pre-training for image quality and aesthetic assessment. *arXiv preprint arXiv:2406.01069*, 2024b.

6. Appendix

6.1. Training details

The models are trained with eight NPUs (64GB memory per device) by default. We set betas and momentum of the AdamW optimizer with (0.9, 0.999) and 0.9, respectively. During training, we utilize a Cosine Scheduler with an initial learning rate of 2×10^{-5} and weight decay of 0.1. The details are shown in Table 10

Hyper-parameters	LoRA-Gen
optimizer	AdamW
learning rate	2e-5
warm steps	50
weight decay	0.1
optimizer momentum	β_1, β_2=0.9, 0.999
batch size	64
epoch	4
max length	2048
LoRA attention dimension (r)	16
LoRA scaling alpha (α)	16
LoRA drop out	0.05

Table 10. Fine-tuning configuration.

6.2. Detailed Assistant Output

The definition of tools follows GPT4Tools (Yang et al., 2024a), encompassing vision foundation models (Xiao et al., 2024b), generative models (Podell et al., 2023), and application-specific models (Brooks et al., 2023; Xiao et al., 2024a). Task description for the role play in the qualitative study of the main text can be seen in Table 14. To strengthen LoRA-Gen's ability to compress and process instructions in the system prompt, we modify the Alpaca dataset, using GPT-4 to generalize specific problems into instruction sets, which are subsequently used as training data.

6.3. Statistical Significance

The standard errors of different tasks are shown in Table 11, all statistics are calculated with the open-sourced lm-evaluation-harness project (Gao et al., 2024b). Additionally, we have re-evaluated our method 4 times on GPT4Tools with a variation of about 0.65% in average score.

6.4. Training Data.

Table 12 outlines the data scale for each reasoning task. Moreover, we process the Alpaca dataset through GPT-4, resulting in a filtered and abstracted set of 37,658 training samples.

6.5. Efficiency Comparison

Table 13 presents the efficiency Comparison among different approaches. MixLoRA† indicates the method without specific optimization. All metrics are measured on an Nvidia GPU. FLOPs are measured using an input of 100 tokens and an instruction of 200 tokens, while memory and latency are evaluated in training mode with a batch size of 8 per GPU.

6.6. More Potential Application

Our current validation focuses on LLMs, and future work will explore its applicability to multimodal large models (Yang et al., 2025; Bai et al., 2023) and unified systems (Xiao et al., 2025a; Zhou et al., 2024a; Xiao et al., 2025b) for understanding and generation. In addition, we plan to enrich more models as tools by incorporating advanced image generation and editing models (Labs, 2023; Wang et al., 2024), as well as speech recognition and synthesis models (Liu et al., 2024a).

Method	ARC-c	ARC-e	OBQA	BoolQ	SIQA	HellaS	WinoG	PIQA
TinyLLaMA	0.0146	0.0089	0.0219	0.0076	0.0112	0.0050	0.0134	0.0100
Qwen	0.0145	0.0089	0.0229	0.0071	0.0113	0.0050	0.0132	0.0098
Gemma	0.0146	0.0089	0.0218	0.0075	0.0112	0.0050	0.0135	0.0096

Table 11. **Standard error on language model benchmarks.**.

Method	ARC-c	ARC-e	OBQA	BoolQ	SIQA	HellaS	WinoG	PIQA
Train	1120	2250	4957	9427	33410	39905	9248	16100
Test	1171	2380	500	3270	1954	10042	1267	1838

Table 12. **The data size of tasks used in our experiments.**

Method	Training Mode			Inference Mode		
	FLOPs	Memory	Latency	FLOPs	Memory	Latency
+LoRA	4.736E+11	37096MiB	0.85s	4.708E+11	11208MiB	0.19s
+LoRAMoE	4.742E+11	26326MiB	1.19s	4.742E+11	11286MiB	0.22s
+MixLoRA†	5.061E+11	30844MiB	2.17s	5.048E+11	11828MiB	1.08s
+LoRA-Gen	1.667E+12	39603MiB	2.84s	1.552E+11	10932MiB	0.11s

Table 13. **Efficiency Comparison.**

Task description of role play:
I want you to act as a poet, crafting verses that resonate deeply with the soul. Your task is to create a poem that not only evokes emotions but also possesses the power to move and inspire those who read it. The subject matter will be provided by the user, and your words should be carefully chosen to express the essence of that theme in a way that touches hearts and stirs deep feelings. Aim for beauty in your language—let your choice of words carry both meaning and emotion, so that each line speaks to the reader on a profound level. Whether the tone is joyful, melancholic, or contemplative, your poem should paint vivid images and capture the emotions you wish to convey.

Edge-side model with LoRA-Gen answer:
A whisper in the wind, a touch so light,\A spark ignites, banishing the night.\Love's gentle hand, a warm embrace,\A smile that lights up every space.\\Two souls entwined, a tapestry of dreams,\Where laughter dances and love softly gleams.\A bond unbreakable, a trust so deep,\In each other's eyes, secrets they keep.\\Through storms and sunshine, hand in hand they roam,\Building castles of hope, a love that calls them home.\A silent language, spoken in the heart,\A symphony of feelings, a work of art.\\For love is a journey, a path to explore,\A treasure to cherish, forevermore.\A beacon of hope, a guiding star,\Love's gentle light, shining near and far.

The baseline answer:
Love, a whisper on the wind,\A melody that softly spins,\A tapestry of threads so fine,\Woven with dreams, and hopes divine.\\A touch that sets the soul alight,\A beacon in the darkest night,\A warmth that melts the coldest fear,\A solace, ever drawing near.\\It blooms in laughter, shared and bright,\In silent moments, bathed in light,\A gentle hand, a knowing glance,\A shared embrace, a sacred dance.\\Love's strength endures, a steadfast guide,\Through storms and calm, it cannot hide,\A constant echo, soft and deep,\A promise whispered, secrets to keep.\\So let us cherish every breath,\This precious gift, beyond all death,\For love's embrace, a timeless art,\A masterpiece, that sets the heart. \\\Let me know if you'd like me to explore a specific aspect of love, or if you have a particular style or tone in mind.

Table 14. Detailed supplement to the visualization results in the main text.