

Progressive LoRA for Multimodal Continual Instruction Tuning

Yahan Yu¹, Duzhen Zhang^{2*}, Yong Ren³, Xuanle Zhao³, Xiuyi Chen³, Chenhui Chu¹

¹Kyoto University, Japan

²Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE

³Institute of Automation, Chinese Academy of Sciences, China

yahan@nlp.ist.i.kyoto-u.ac.jp, duzhen.zhang@mbzuai.ac.ae,

chu@i.kyoto-u.ac.jp

Abstract

Multimodal Continual Instruction Tuning (MCIT) empowers Multimodal Large Language Models (MLLMs) to adapt to ever-evolving requirements without continuous costly retraining. However, MCIT faces challenges in mitigating Catastrophic Forgetting (CF) and enhancing Knowledge Transfer (KT). Existing works combine Mixture-of-Expert (MoE) and LoRA to address these. However, using a fixed number of shared LoRA blocks across tasks can lead to the overwriting of acquired knowledge, making MLLMs harder to handle CF and KT. Therefore, we propose the **Progressive LoRA** framework (ProgLoRA), which contains a progressive LoRA pool and trains a new LoRA block for each incremental task to reduce knowledge interference. Specifically, ProgLoRA has two key mechanisms: task-aware allocation for effectively leveraging acquired knowledge at current task and task recall for realigning the model with learned tasks. Additionally, considering different application scenarios, we design a static ProgLoRA for the more idealized basic setting and a dynamic ProgLoRA for the more realistic challenging setting. Experiments on the latest MCIT benchmark demonstrate that ProgLoRA outperforms existing approaches.¹

1 Introduction

Multimodal Large Language Models (MLLMs) (Zheng et al., 2024b; Dai et al., 2023) have demonstrated remarkable capabilities across a wide range of vision-language tasks (Mishra et al., 2019; Goyal et al., 2017), showcasing their potential to handle diverse scenarios. To support joint training across multiple tasks, instruction tuning (Liu et al., 2023a; Shen et al., 2024) is commonly used. In real-world applications, MLLMs are often required to continuously follow new instructions to keep pace with

*Corresponding author.

¹Our code is available at <https://github.com/ku-nlp/ProgLoRA>.

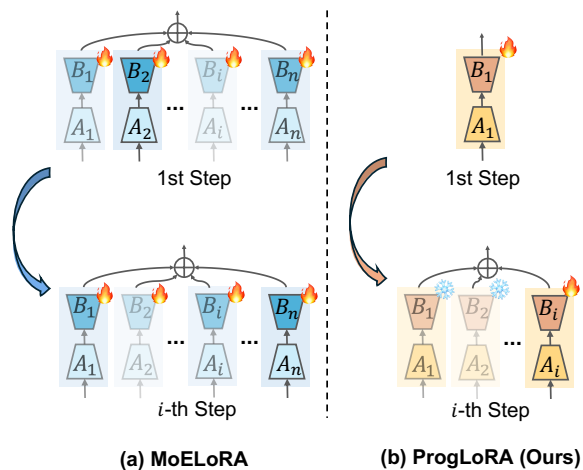


Figure 1: A comparison between MoELoRA and our ProgLoRA. In LoRA blocks, higher transparency represents a lower allocation weight. MoELoRA updates the same set of blocks and allocates them by a simple router. In contrast, ProgLoRA updates only newly added block in current task and assigns blocks by task relevance.

advancing knowledge. Despite this demand, most existing MLLMs are static, restricting their ability to address emerging tasks. Considering the computation cost and efficiency, retraining MLLMs from scratch for every new instruction is impractical. Therefore, existing works (Zheng et al., 2024a) conceptualize this challenge within Multimodal Continual Instruction Tuning (MCIT) (He et al., 2023), which is designed to incrementally instruction-tune MLLMs while maintaining high performance on previously tuned tasks.

However, MCIT faces two challenges: mitigating Catastrophic Forgetting (CF), which involves preserving the model’s previously acquired knowledge after learning new tasks; enhancing knowledge transfer (KT), which encourages models to leverage acquired knowledge to improve the learning of new tasks. To address this challenge, the Mixture-of-Experts LoRA (MoELoRA) (Liu et al., 2023b; Chen et al., 2024) leverages a set of special-

ized LoRA blocks designed to capture task-specific knowledge from sequential tasks. Nevertheless, a limitation is that MoELoRA shares the same set of LoRA blocks across different tasks as shown in Fig. 1 (a). As a result, acquired task-specific knowledge is easily overwritten by the knowledge of subsequent incremental tasks, which is more challenging to mitigate CF and enhance KT.

To further mitigate CF and enhance KT, we propose the **Progressive LoRA** framework (ProgLoRA). To prevent the disruption of acquired knowledge due to model parameter updates, ProgLoRA incorporates a progressive LoRA pool with multiple LoRA blocks. As shown in Fig. 1 (b), in each incremental step, a newly added LoRA block is trained to capture task-specific knowledge, while all previously tuned LoRA blocks are frozen. ProgLoRA isolates task-specific knowledge in independent LoRA blocks, preventing overwriting during updates. Meanwhile, frozen blocks still contribute to the training of other tasks. Specifically, ProgLoRA comprises two main components: **task-aware allocation** and **task recall**. Even if the acquired knowledge within the LoRA pool originates from tasks entirely unrelated to the current one, it may still enhance the learning of the current task. Therefore, we propose task-aware allocation, encouraging the model to effectively select and fuse LoRA blocks. Additionally, since the model contains trainable parameters beyond the LoRA pool, we introduce task recall to align with previous tasks, aiming to mitigate CF.

Considering different application backgrounds, we propose two kinds of implementations: **ProgLoRA (static)** for the ideal basic setting and **ProgLoRA (dynamic)** for a practical, challenging setting. The basic setting can assess task information associated with the input sample during training and testing stages, while challenging setting can only assess during training. Thus, in ProgLoRA (static), we propose static allocation weights in task-aware allocation and simple replay in task recall, which are both based on precomputed task similarity scores. In ProgLoRA (dynamic), as for the task-aware allocation, we assign a key to each LoRA block and calculate dynamic allocation weights with LoRA keys for adaptively fusing LoRA blocks. For the task recall, we apply a Kullback-Leibler (KL) divergence loss to constrain the allocation weights of previous samples in the current step more similar to the ones in previous steps, where the previous samples are the same set

as in the ones of ProgLoRA (static).

Our contributions can be summarized as follows:

1. We propose ProgLoRA, where knowledge from different tasks is stored in separate LoRA blocks, thereby minimizing task interference.
2. We design the task-aware allocation to select and fuse LoRA blocks, aiming to enhance KT by effectively using acquired knowledge. We also propose the task recall to constrain the model updating and further mitigate CF. Considering basic and challenging settings, we design ProgLoRA (static) and ProgLoRA (dynamic), respectively.
3. Comprehensive experiments on LLaVA-1.5 using the latest MCIT benchmark demonstrate that ProgLoRA outperforms MoELoRA.

2 Related Works

2.1 MLLMs

MLLMs extend LLMs' (Chung et al., 2022; Bai et al., 2023; Touvron et al., 2023; Li et al., 2025) capabilities to process visual and textual inputs. By combining the advanced reasoning skills of LLMs with rich visual representations from visual backbones, MLLMs achieve refined multimodal reasoning and deep content understanding abilities (Zhang et al., 2024; Zhao et al., 2025b,a). LLaVA (Liu et al., 2023a), NExT-GPT (Wu et al., 2023), and MiniGPT-v2 (Chen et al., 2023) adopt a linear projection layer to connect a frozen LLM with a visual encoder for multimodal alignment. Meanwhile, models like InstructBLIP (Dai et al., 2023) and BLIP-2 (Li et al., 2023) trained the Q-Former projector to bridge the gap between modalities. Additionally, instruction tuning is leveraged to make models follow human instructions. These developments highlight the rapid evolution of MLLMs in addressing reasoning and comprehension tasks across modalities.

2.2 MCIT

MLLMs are necessary to be continuously updated to equip new abilities and keep pace with the rapidly evolving landscape of human knowledge (Wu et al., 2024b; Zheng et al., 2024b; Shi et al., 2024). To solve this, MCIT has emerged as a key solution, enabling models to integrate new data incrementally without the expensive process of re-training from scratch. Recent efforts have intro-

duced new MCIT benchmarks designed to incrementally fine-tune MLLMs on multiple tasks (He et al., 2023; Chen et al., 2024). Facing the major obstacle of mitigating CF and enhancing KT, where knowledge from earlier tasks interrupts with the new ones, some methods such as MoELoRA have been proposed. These methods aim to preserve instruction alignment across tasks by aggregating LoRA experts (Chen et al., 2024). Despite these efforts, such methods often fall short, as the shared expert design can result in knowledge interference, ultimately degrading model performance over time.

3 Preliminary

3.1 Problem Definition

Continual learning (Rypešć et al., 2024; Zhang et al., 2023; Zheng et al., 2025; Dong et al., 2022, 2024) aims to tackle challenges from sequentially evolving tasks, avoiding the high cost of full re-training. MCIT (Chen et al., 2024) is one of the continual learning approaches that allows MLLMs to adapt to new datasets in a step-by-step sequence through instruction tuning. The goal of MCIT is to sequentially train a model \mathcal{M} on the stream of tasks while ensuring strong performance across all previously encountered tasks. The problem is structured as a sequential stream of tasks, represented as $\{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_N\}$, where N represents the total number of tasks. Each task \mathcal{T}_i is associated with its own dataset \mathcal{D}_i . Notably, datasets from different tasks are diverse and cover knowledge from multiple domains without restriction. In each dataset \mathcal{D}_i , it consists of M_i data pairs $(X_{i,j}^v, X_{i,j}^q, X_{i,j}^a)_{j=1}^{M_i}$, where X^v , X^q and X^a denote image tokens, instruction tokens, and answer tokens, respectively.

3.2 LoRA

LoRA (Hu et al., 2022; Wu et al., 2024a; Wang et al., 2023, 2024a,b) is a parameter-efficient fine-tuning technique designed to adapt pre-trained LLMs to new tasks without modifying full set of parameters. Given an intermediate representation \mathbf{x} from preceding attention layer, the forward pass in LLMs is defined as:

$$\mathbf{h} = \mathbf{W}\mathbf{x} + \frac{\alpha}{r}\mathbf{B}\mathbf{A}\mathbf{x}, \quad (1)$$

where $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ denotes pre-trained weight in LLMs, $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{out}}}$ and $\mathbf{B} \in \mathbb{R}^{d_{\text{in}} \times r}$ denote trainable low-rank matrices with rank r , and α denotes hyperparameter to finetune effect of r .

In MCIT, Chen et al. (2024) adopt the MoELoRA (Dou et al., 2023; Liu et al., 2023b) to mitigate CF, utilizing a set of specialized experts to capture task-specific knowledge with a fixed expert pool $\{E_1(\cdot), E_2(\cdot), \dots, E_K(\cdot)\}$ and a router $R(\cdot)$ (Fedus et al., 2022; Zhu et al., 2024). The MoELoRA layer computes output as follows:

$$\mathbf{h} = \mathbf{W}_F\mathbf{x} + \frac{\alpha}{r} \sum R(\mathbf{x})_{[k]} E_k(\mathbf{x})$$

$$R(\mathbf{x}) = \text{Softmax}(\mathbf{W}_R\mathbf{x}_{[0]}), E_k(\mathbf{x}) = \mathbf{B}_k\mathbf{A}_k\mathbf{x} \quad (2)$$

where $\mathbf{W}_F \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ denotes parameters of the FF layer. The model employs K experts and each is characterized by two trainable low-rank matrices, $\mathbf{A}_k \in \mathbb{R}^{d_{\text{in}} \times \frac{r}{K}}$ and $\mathbf{B}_k \in \mathbb{R}^{\frac{r}{K} \times d_{\text{out}}}$. These matrices are with a reduced rank of $\frac{r}{K}$, ensuring that overall trainable parameters remains equivalent to a single LoRA setup, thereby maintaining computational efficiency. Meanwhile, $R(\cdot)$ takes the first token of \mathbf{x} as input and uses trainable $\mathbf{W}_R \in \mathbb{R}^{d_{\text{in}} \times K}$ to generate a probability distribution of each expert.

4 Approach

4.1 Overview

Our ProgLoRA, as shown in Fig. 2, is a novel framework to address CF and KT in a progressive and task-aware manner. We set a progressive LoRA pool $\{BLK_1, BLK_2, \dots, BLK_N\}$, where $BLK_i(\mathbf{x}) = \mathbf{B}\mathbf{A}\mathbf{x}$ represents i -th LoRA block. ProgLoRA comprises two components:

Task-aware Allocation The LoRA blocks BLK_p trained on previous tasks \mathcal{T}_p ($p \in [1, i)$) are frozen during the training of the current task \mathcal{T}_i to preserve the acquired knowledge. This knowledge, even though it comes from different tasks than the current one, can also be effectively utilized to promote KT. Thus, we design allocation weights w to select and fuse LoRA blocks in LoRA pool, where each block stores task-specific knowledge.

Task Recall Freezing previous BLK_p intuitively mitigates CF. However, MLLMs often include other trainable components (e.g., cross-modal projection layers) that remain active across tasks, which creates a vulnerability of CF. Thus, we introduce task recall.

Specifically, our method has two versions: (1) For the basic setting, where task information (i.e., task ID i) is available during training and testing, we propose **ProgLoRA (static)** in Fig. 2 (a) with fixed allocation weights. (2) For the challenging

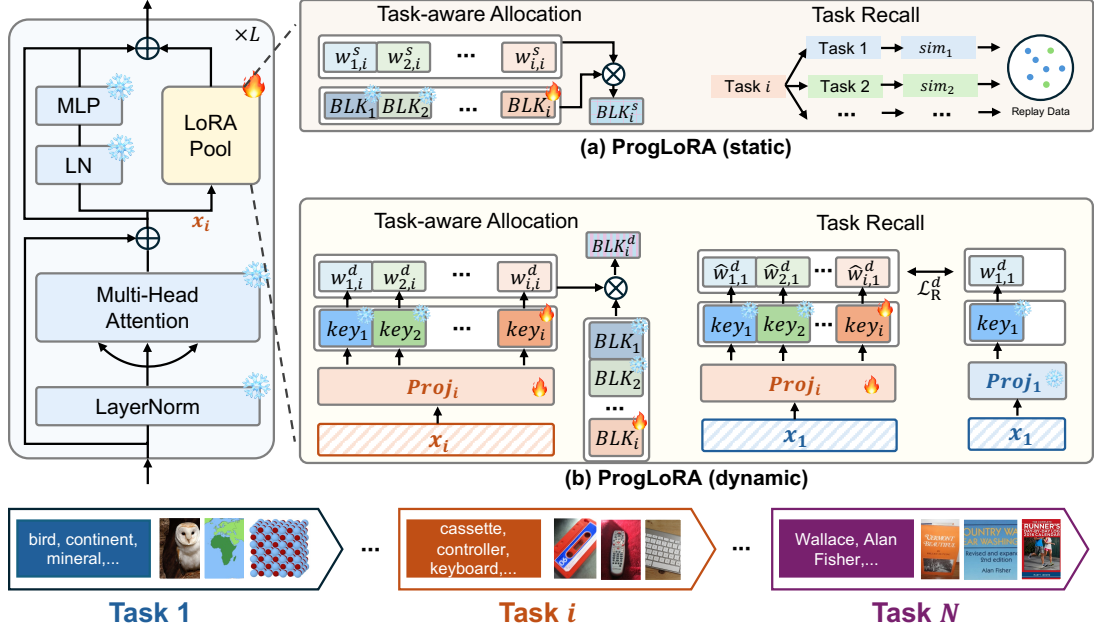


Figure 2: The overall structure of ProgLoRA. Considering different application scenarios, our method has two versions: ProgLoRA (static) in (a) for basic setting and ProgLoRA (dynamic) in (b) for more challenging setting.

setting, where task ID is only available during training, we introduce **ProgLoRA (dynamic)** in Fig. 2 (b), which enables the adaptive allocation of LoRA blocks.

4.2 ProgLoRA (static)

The basic setting represents an idealized scenario. In this context, our goal is to leverage a lightweight model.

Task-aware Allocation Prior works (Nikandrou et al., 2022) have optimized continual learning methods by leveraging similarity between \mathcal{T}_p and \mathcal{T}_i . Higher similarity facilitates the retention of acquired knowledge from \mathcal{T}_p while enabling efficient learning of new knowledge in \mathcal{T}_i . Inspired by these, we propose **static allocation weights** w^s based on task similarity. When the similarity between \mathcal{T}_p and \mathcal{T}_i is higher, the weight $w_{p,i}^s$ assigned to BLK_p increases. The formulas are as follows:

$$\theta_{BLK_i^s} = w_{i,i}^s \theta_{BLK_i} + \sum_{p=1}^{i-1} w_{p,i}^s \theta_{BLK_p}, \quad (3)$$

$$w_{p,i}^s = \gamma(e_p^v, e_i^v) \cdot \gamma(e_p^q, e_i^q) \cdot \gamma(e_p^a, e_i^a)$$

where γ denotes cosine similarity measure, θ denotes the parameters, BLK_i^s denotes the fused LoRA block. e_i^v , e_i^q and e_i^a denote pre-computed average image, instruction and answer embeddings of \mathcal{D}_i . Note that $w_{i,i}^s = 1$. The training loss for

allocation in \mathcal{T}_i is:

$$\mathcal{L}_{A,i}^s = - \sum_j^{M_i^{\text{train}}} \log P(X_{i,j}^a | X_{i,j}^v, X_{i,j}^q; \theta_{BLK_i^s}, \theta_{\mathcal{M}_i}), \quad (4)$$

where $\theta_{\mathcal{M}_i}$ denotes parameters of MLLM backbone. This static method ensures that more relevant acquired knowledge is better leveraged to facilitate the learning of new knowledge.

Task Recall Given that ProgLoRA (static) maintains its simplicity by not introducing any additional trainable parameters beyond the LoRA pool, we propose a **replay-based task recall** to keep this resource-efficient way. This approach leverages a small amount of replayed data from \mathcal{T}_p based on task similarity to align the model. Specifically, we replay $(1 - w_{p,i}^s)\phi$ samples from \mathcal{T}_p when learning \mathcal{T}_i , where ϕ denotes base value of replay samples. \mathcal{T}_p that is less similar to \mathcal{T}_i will have more replay samples, thereby enhancing the recall of acquired knowledge. The training loss $\mathcal{L}_{R,i}^s$ for task recall in \mathcal{T}_i is the same as the calculation process in Eq. (4), while the input data is the replay ones.

The final loss is defined as $\mathcal{L}_i^s = \mathcal{L}_{A,i}^s + \lambda \mathcal{L}_{R,i}^s$, where λ is the loss coefficient. During the inference phase, for a test sample originating from \mathcal{T}_i , we directly execute using the fused LoRA block BLK_i^s .

4.3 ProgLoRA (dynamic)

In challenging settings where task IDs are unavailable during the testing phase, it becomes necessary to incorporate a mechanism within the model that can adaptively allocate and fuse LoRA blocks. Therefore, we propose ProLoRA (dynamic) to extend the static method.

Task-aware Allocation We design **dynamic allocation weights** to fuse LoRA blocks. Specifically, at the training stage of \mathcal{T}_i , a key vector \mathbf{key}_n is assigned to each LoRA block BLK_n ($n \in [1, i]$), and instance-level weights are computed. This ensures that the allocation process is block-agnostic and can be compatible with various adaptation techniques (e.g., LoRA). The dynamic allocation process begins when the $X_{i,j}^q$ is passed through layers preceding the LoRA pool of the LLM backbone and obtains embeddings $\mathbf{x}_{i,j}^q$. For better alignment with the key vector spaces, we first fed $\mathbf{x}_{i,j}^q$ into a projector $Proj_i$ (we omit the q and j for simplicity):

$$\mathbf{h}_i = \text{LN}(\mathbf{W}_2 \varphi(\mathbf{W}_1 \mathbf{x}_{i[\max]} + b_1) + b_2), \quad (5)$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_p \times d_{in}}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_{in} \times d_p}$ denote trainable parameters, b_1 and b_2 denote bias, $\varphi(\cdot)$ and $\text{LN}(\cdot)$ denote SiLU function and Layer Norm to ensure the reliability of the training process. Note that max-pool operation is applied to \mathbf{x}_i along the token length dimension to align it with the dimension of the $\mathbf{key}_n \in \mathbb{R}^{d_{in}}$. Then, dynamic allocation weights $w_{n,i}^d$ is processed with aligned \mathbf{h}_i as:

$$w_{n,i}^d = \text{Softmax}(\mathbf{h}_i \mathbf{key}_n / \alpha_{tem}), \quad (6)$$

where α_{tem} represents a temperature factor to constrain the confidence of $w_{n,i}^d$, which makes the model less sensitive to small differences and leads to more balanced weight distributions to improve robustness. Similar to Eq. (3), the LoRA blocks are fused to BLK_i^d as follows:

$$\theta_{BLK_i^d} = \sum_{n=1}^i w_{n,i}^d \theta_{BLK_n} \quad (7)$$

To update the model, the calculation of training loss $\mathcal{L}_{A,i}^d$ is similar to that of $\mathcal{L}_{A,i}^s$ in Eq. (4), but it incorporates $\theta_{BLK_i^d}$, θ_{key} , θ_{Proj_i} , and $\theta_{\mathcal{M}_i}$ into the optimization process.

Task Recall As tasks are sequentially trained, the $Proj$ in Eq. (5) is updated continuously. To

ensure that inputs from \mathcal{T}_p still accurately perform the dynamic allocation weights and identify the specific fusion of LoRA blocks, we introduce the **KL-divergence-based task recall**. This is achieved by using samples from \mathcal{T}_p ; however, unlike in ProgLoRA (static), we don't perform full LLM replay on these samples. Instead, they are solely used to generate pseudo dynamic allocation weights \hat{w}^d for the recall of $Proj_i$ at the training of \mathcal{T}_i . To help $Proj_i$ accurately recall the correct $\mathbf{w}_p^d = \{w_{1,p}^d, \dots, w_{p,p}^d\}$ for samples \mathcal{T}_p , each previous sample is fed into layers in Eq. (5) and (6) to get $\hat{\mathbf{w}}_i^d = \{\hat{w}_{1,i}^d, \dots, \hat{w}_{i,i}^d\}$ at the training of \mathcal{T}_i . To ensure that $\hat{\mathbf{w}}_i^d$ aligns with its intended value \mathbf{w}_p^d , we minimize the recall loss $\mathcal{L}_{R,i}^d$:

$$\mathcal{L}_{R,i}^d = \sum_{p=1}^{i-1} \sum_{N_p^{recall}} \text{KL}(\hat{\mathbf{w}}_i^d || \mathbf{w}_p^d), \quad (8)$$

where N_p^{recall} denotes the number of samples from \mathcal{T}_p and is kept consistent with the replay sample size used in the ProgLoRA (static). $\text{KL}(\cdot)$ denotes a KL divergence loss. Note that \mathbf{w}_p^d are padded with zeros to match the sequence length of $\hat{\mathbf{w}}_i^d$.

Finally, we also optimize the model in a multi-task learning manner:

$$\mathcal{L}_i^d = \mathcal{L}_{A,i}^d + \lambda \mathcal{L}_{R,i}^d \quad (9)$$

During the inference phase, we dynamically allocate all available LoRA blocks to the testing samples in an adaptive manner by Eq. (5)-(7).

5 Experimental Settings

5.1 Datasets

We follow the setup outlined in the MCIT benchmark, CoIN (Chen et al., 2024), which includes 8 multimodal datasets covering a diverse range of tasks. Datasets in CoIN benchmark are: (Lu et al., 2022), TextVQA (Singh et al., 2019), ImageNet (Deng et al., 2009), GQA (Hudson and Manning, 2019), VizWiz (Gurari et al., 2018), Grounding (Kazemzadeh et al., 2014; Mao et al., 2016), VQAv2 (Goyal et al., 2017), and OCR-VQA (Mishra et al., 2019). The datasets are processed in the same sequential order as experiments in Chen et al. (2024). Detailed statistics for the 8 datasets in the CoIN benchmark (Chen et al., 2024) are presented in Appendix A.

Method	Accuracy on Each Task								Overall Results			
	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	Grounding	VQAv2	OCR-VQA	ACC \uparrow	MAA \uparrow	BWT \uparrow	FWT \uparrow
Zero-shot	49.91	2.88	0.33	2.08	0.90	0.00	0.68	0.17	–	7.12	–	–
Multi-task	56.77	49.35	95.55	56.65	53.90	30.09	59.50	55.65	–	57.18	–	–
LoRA	82.45	49.99	96.05	56.40	55.45	31.27	62.20	57.08	28.74	32.97	-32.62	-4.82
	21.26	28.74	10.25	36.78	32.45	0.83	42.50	57.08				
LwF*	81.36	50.59	96.84	51.98	48.19	25.13	41.30	64.12	30.41	34.95	-27.03	-8.75
	26.78	37.52	12.64	35.18	25.24	2.87	38.92	64.12				
EWC*	82.81	51.76	96.80	46.19	48.68	26.82	66.37	63.46	32.90	36.93	-27.46	-5.81
	30.33	36.08	11.62	35.75	37.50	3.48	44.98	63.46				
MoELoRA	75.78	51.73	96.70	59.42	58.88	37.50	64.22	60.08	37.13	42.76	-25.91	-4.01
	63.09	38.63	10.50	37.38	43.62	0.59	43.15	60.08				
ProgLoRA (dynamic)	76.27	60.78	97.32	61.27	60.16	39.35	65.83	64.44	59.09	62.38	-6.59	1.37
	74.84	51.83	83.90	49.93	53.87	31.19	62.71	64.44				

Table 1: Main results on the LLaVA-1.5-7B model. For accuracy of each task of MCIT methods, the first row denotes the results for each task evaluated after its tuning (*i.e.*, $P_{i,i}$ ($i \in [1, N]$)) with the best performance highlighted in **red**, while the second row shows each task’s results after tuning the final task (*i.e.*, $P_{N,i}$) with the best ones in **blue**. For overall results, the **bold** highlights the best performance. * represents results from our re-implementation, while others are cited from CoIN (Chen et al., 2024).

5.2 Compared Methods

We compare ProgLoRA with zero-shot, multi-task and MCIT methods (LoRA (Hu et al., 2022), MoELoRA (Chen et al., 2024), LwF (Li and Hoiem, 2017), and EWC (Kirkpatrick et al., 2017)). Details are shown in Appendix B.

5.3 Evaluation Metrics

For metrics, we use Average Accuracy (ACC), Mean Average Accuracy (MAA), Backward Transfer (BWT), and Forward Transfer (FWT). $ACC = \frac{1}{N} \sum_{i=1}^N P_{N,i}$, where $P_{N,i}$ is the performance on i -th task after training the final task \mathcal{T}_N . $MAA = \frac{1}{N} \sum_{i=1}^N (\frac{1}{i} \sum_{k=1}^i P_{i,k})$, where $P_{i,k}$ is the performance on \mathcal{T}_k after training \mathcal{T}_i . $BWT = \frac{1}{N} \sum_{i=1}^N (P_{N,i} - P_{i,i})$, where $P_{i,i}$ is the performance on \mathcal{T}_i after training on \mathcal{T}_i . $FWT = \frac{1}{N} \sum_{i=1}^N (P_{i,i} - P_{0,i})$, where $P_{0,i}$ represents the performance of training \mathcal{T}_i in isolation. Among these metrics, ACC reflects the model’s ability to promote KT, MAA measures accuracy across all tasks to focus on stability, BWT assesses the model’s capacity to mitigate CF, and FWT measures the model’s capacity to enhance KT. More details are shown in Appendices C.

5.4 Implementation

We also follow the same implementation settings as Chen et al. (2024) and utilize **LLaVA-1.5-7B** (Liu et al., 2023a) as the backbone. For all the experiments, LoRA (Hu et al., 2022) is integrated into the LLMs. The vision encoder and LLM remain frozen,

while only the projection layer and LoRA are updated. For a fair comparison, for a single LoRA block in ProgLoRA, rank r is set to 16, α is kept consistent with MoELoRA, with the base value of replay amount ϕ set to 200 and loss coefficient λ set to 2.0. For the calculation of task similarity, e_i^v , e_i^q and e_i^a denote pre-computed average image, instruction and answer embeddings of \mathcal{D}_i , respectively. These embeddings are obtained using BERT (Kenton and Toutanova, 2019) to encode texts and ViT (Dosovitskiy et al., 2020) to encode images. All experiments are conducted on 4 NVIDIA A100 GPUs, each with 80GB of memory.

6 Results and Analysis

6.1 Main Results

Results are summarized in Table 1. Considering that MCIT baselines don’t require task information during testing, we compare only ProgLoRA (dynamic) with them. For the overall results, **our method significantly outperforms all baselines**, which illustrates the designed task-aware allocation and task recall can solve problems of CF and KT. For accuracy on each task, our method consistently achieves the best performance on both $P_{i,i}$ and $P_{N,i}$, where $P_{i,i}$ denotes performance on \mathcal{T}_i after its tuning and $P_{N,i}$ denotes performance on \mathcal{T}_i after tuning on the last task \mathcal{T}_N . Meanwhile, it’s worth noting that while baselines may perform better on $P_{i,i}$, this doesn’t indicate a weakness in ProgLoRA. This is because the overall r of ProgLoRA is always less than the baselines before training the last

Method	Accuracy on Each Task								Overall Results			
	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	Grounding	VQAv2	OCR-VQA	ACC \uparrow	MAA \uparrow	BWT \uparrow	FWT \uparrow
ProgLoRA (static)	76.27	60.07	92.65	58.45	57.33	17.68	64.45	63.54	58.73	66.44	-2.57	1.06
	72.70	57.08	91.20	53.12	56.04	12.54	63.64	63.54				
ProgLoRA (dynamic)	76.27	60.78	97.32	61.27	60.16	39.35	65.83	64.44	59.09	62.38	-6.59	1.37
	74.84	51.83	83.90	49.93	53.87	31.19	62.71	64.44				

Table 2: Comparison of ProgLoRA (static) and ProgLoRA (dynamic) on LLaVA-1.5-7B. Meaning of the highlight marker is the same as in Table 1.

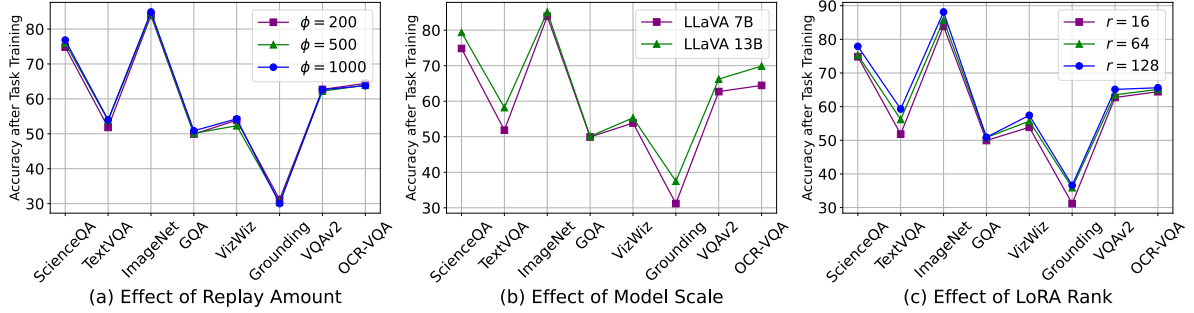


Figure 3: Step-wise comparison of ProgLoRA (dynamic) with different (a) replay amount, (b) model scale, and (c) LoRA rank.

Variant	ACC \uparrow	MAA \uparrow	BWT \uparrow	FWT \uparrow
ProgLoRA (static)	58.73	66.44	-2.57	1.06
w/ allocation weight $w^s = 1$	54.60	63.43	-2.62	0.83
w/o task recall	56.06	65.06	-4.84	0.90
w/ random task recall	57.62	66.00	-2.76	0.97
ProgLoRA (dynamic)	59.09	62.38	-6.59	1.37
w/o task recall	57.52	62.12	-8.07	1.08

Table 3: The ablation study of ProgLoRA on the LLaVA-1.5-7B backbone. **Bold** highlights the best performance.

task, making the slightly lower results reasonable. We also observed that ProgLoRA (dynamic) outperformed the multi-task setting in terms of MAA, demonstrating that our method effectively facilitates knowledge transfer.

We also compare the ProgLoRA (static) results separately with the dynamic ones to demonstrate the impact of different allocation and recall strategies within the same progressive LoRA pool, which is shown in Table 2. **ProgLoRA (static) achieves higher MAA, BWT and most $P_{N,i}$** , demonstrating its advantage in mitigating CF. We think that it operates under idealized setting, where predetermined allocation weights are applied in the testing stage based on known task details. As a result, its performance after tuning different tasks remains stable. Additionally, for each task, the higher $P_{i,i}$ and FWT only occur with ProgLoRA (dynamic), for the reason that **ProgLoRA (static) is lack of flexibility**, highlighting the advantages

of dynamic task allocation in improving KT.

6.2 Ablation Study

As shown in Table 3, we analyzed each component. For ProgLoRA (static), KT is hindered when setting $w^s = 1$, indicating the necessity of leveraging task similarity for LoRA block allocation when the allocation weight is fixed. Additionally, removing task recall or replacing it with random sampling of ϕ samples from each previous task increases CF, highlighting the importance of task recall. For ProgLoRA (dynamic), removing task recall leads to a significantly greater occurrence of CF, underscoring the critical role of this component in enabling adaptive LoRA block allocation.

6.3 More Explorations

Effect of Replay Amount We show the step-wise results on ProgLoRA (dynamic) with different replay amount ϕ after training on the final task (OCR-VQA) in Fig. 3 (a), where the backbone is LLaVA-1.5-7B. More results are shown in Appendix F. The results showed a very slight improvement with the increase in ϕ , but it can be observed that tasks later in the task sequence (such as OCR-VQA) are slightly affected as the ϕ increases. Meanwhile, it’s preferable to use as little replay data as possible, because storing and replaying large amounts of data can be computationally

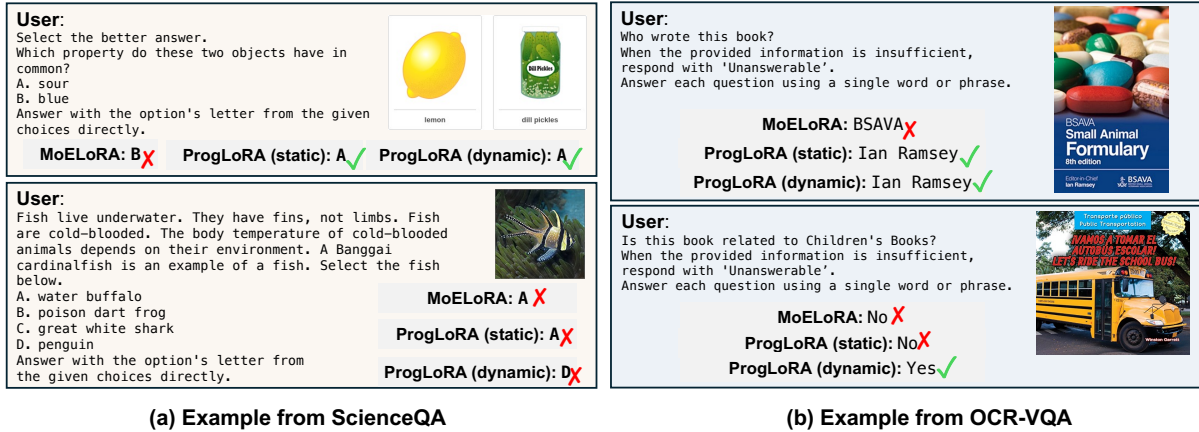


Figure 4: Comparison between MoELoRA and ProgLoRA on cases after training on the final task.

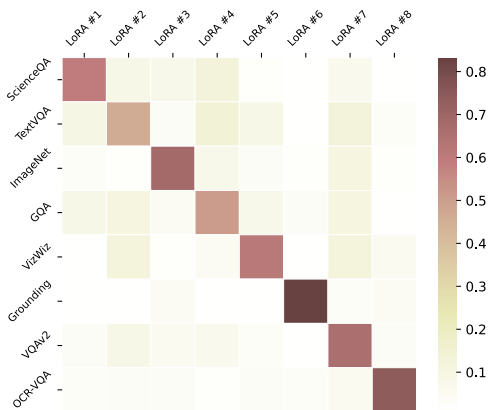


Figure 5: Visualization of allocation weights in ProgLoRA (dynamic).

expensive and time-consuming in real-world scenarios. Therefore, $\phi = 200$ is appropriate.

Effect of Model Scale We choose LLaVA-1.5-13B as the new backbone to evaluate ProgLoRA (dynamic). We show the step-wise results after training on the final task in Fig. 3 (b). We observed that larger models tend to offer better capabilities for mitigating CF and promoting KT. We analyse the reason that, with an increase in model size, the model retains a larger optimization space for learning more new knowledge.

Effect of LoRA Settings We investigate the influence of LoRA rank r . Step-wise results on ProgLoRA (dynamic) after training on the final task are provided in Fig. 3 (c), where MLLM backbone is LLaVA-1.5-7B. More results are shown in Appendix G. We observed that performance improves as r increases, demonstrating that more trainable parameters enhances model’s ability to acquire new multimodal knowledge. Additionally,

CF also decreases, because the additional parameters provide the model with sufficient optimization space to learn more multimodal knowledge.

We also conduct further explorations, including results on **Other MLLM Backbone** (Appendix D), analysing the effect of **Task Order** (Appendix H), effect of **Instruction Diversity** (Appendix I), and discussion on **Efficiency** (Appendix J).

6.4 Visualization of Allocation Weights

We show the allocation weights on ProgLoRA (dynamic) after completing the final task in Fig. 5, where MLLM backbone is LLaVA-1.5-7B. We observe that task-specific LoRA block associated with the input samples occupies the largest proportion. This demonstrates that our designed Task-aware Allocation can automatically identify the most relevant task-specific knowledge. Compared to static weights provided in Appendix K, the partial consistency between dynamic weights and their corresponding parts proves that task similarity is effective in improving the performance of MCIT, but it cannot be fully relied upon. Designing dynamic allocation weights is necessary.

6.5 Case Study

We provide some case studies in Fig. 4. For the top case of Fig. 4 (a), after fine-tuning the final task, ProgLoRA still provides the correct answer for the first task, while MoELoRA doesn’t. This highlights ProgLoRA’s advantage to mitigate CF. At the same time, we also demonstrate potential weaknesses in ProgLoRA’s reasoning capability. For the bottom case of Fig. 4 (a), we present a failed case where ProgLoRA provides an incorrect answer when handling scenarios with extensive contextual informa-

tion and complex options. Similarly, for the top case of Fig. 4 (b), ProgLoRA successfully generates the correct answer for the final task, whereas MoELoRA fails to do so, demonstrating the edge of ProgLoRA in leveraging acquired knowledge to promote KT. The bottom one shows a failed case of ProgLoRA (static), illustrating limitations of static method when dealing with obscure information. More case studies are shown in Appendix L.

7 Conclusion

In this paper, we proposed ProgLoRA with a progressive LoRA pool that mitigates task interference by isolating knowledge in separate LoRA blocks. We design task-aware allocation and task recall to mitigate CF and promote KT. Experiments show that ProgLoRA outperforms MoELoRA and LoRA across two MLLM backbones on CoIN benchmark, offering a more efficient and stable MCIT approach. Future work could explore how to achieve a progressive LoRA pool with fewer parameters.

Acknowledgments

We express our gratitude to the anonymous reviewers for their valuable and insightful comments. This work was supported by JST BOOST, Grant Number JPMJBS2407.

Limitations

While ProgLoRA offers significant improvements in mitigating catastrophic forgetting and enhancing knowledge transfer, it does have certain limitations. First, our work was confined to the CoIN benchmark. To further validate the effectiveness of MCIT methods, future work could focus on creating more comprehensive and challenging benchmarks that encompass a wider array of tasks and domains.

Secondly, our method is influenced by the task order to some extent. We believe that maintaining robustness across different task orders is a valuable research direction. For instance, further enhancing the ability to transfer knowledge could allow the model to not only learn new knowledge but also revisit and fill gaps in previously learned knowledge.

Finally, We have considered that the total parameters of the model will increase as new tasks are added. Since the additional parameters introduced by our method at each task come from one LoRA block, which is small compared to the total number of model parameters, the impact of the increasing number of tasks on the model’s parameter size can

be neglected in our work. Thus, currently, we are primarily focused on how to best mitigate forgetting and promote knowledge transfer within a task sequence of a certain length for MLLM. For future work, this is an important and interesting direction to explore how to achieve a more efficient MCIT model in the context of an infinite number of new tasks. Some methods, like the fusion of similar block fusion and block pruning, can be used to address this problem.

Ethics Statement

In addressing the ethical aspects of our work, we would like to provide the following clarifications: (1) This research does not make use of any sensitive or private data, and no tasks were performed that involve sensitive information. (2) All experiments were conducted using publicly accessible datasets, which have been sourced from reputable and established scientific research. (3) We have provided comprehensive details on the dataset statistics and the hyperparameter configurations employed in our methodology, ensuring full transparency and alignment with the reported results. (4) In the interest of fostering reproducibility and encouraging further research, we plan to make our code publicly available on GitHub upon the publication of this work.

References

- Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.
- Cheng Chen, Junchen Zhu, Xu Luo, Heng Tao Shen, Jingkuan Song, and Lianli Gao. 2024. [CoIN: A benchmark of continual instruction tuning for multimodal large language models](#). In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. 2023. Minigt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

- Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven C. H. Hoi. 2023. InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- Jiahua Dong, Hongliu Li, Yang Cong, Gan Sun, Yulun Zhang, and Luc Van Gool. 2024. [No one left behind: Real-world federated class-incremental learning](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(4):2054–2070.
- Jiahua Dong, Lixu Wang, Zhen Fang, Gan Sun, Shichao Xu, Xiao Wang, and Qi Zhu. 2022. Federated class-incremental learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.
- Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. 2023. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *arXiv preprint arXiv:2312.09979*, 4(7).
- William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.
- Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.
- Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. 2018. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3608–3617.
- Jinghan He, Haiyun Guo, Ming Tang, and Jinqiao Wang. 2023. Continual instruction tuning for large multimodal models. *arXiv preprint arXiv:2311.16206*.
- Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.
- Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709.
- Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798.
- Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.
- Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, pages 19730–19742.
- Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.
- Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, et al. 2025. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual Instruction Tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2023b. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *arXiv preprint arXiv:2310.18339*.
- Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

- Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20.
- Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. 2019. Ocr-vqa: Visual question answering by reading text in images. In *2019 international conference on document analysis and recognition (ICDAR)*, pages 947–952. IEEE.
- Mavina Nikandrou, Lu Yu, Alessandro Suglia, Ioannis Konstas, and Verena Rieser. 2022. Task formulation matters when learning continually: A case study in visual question answering. *arXiv preprint arXiv:2210.00044*.
- Grzegorz Rypeś, Sebastian Cygert, Valeriya Khan, Tomasz Trzcinski, Bartosz Zieliński, and Bartłomiej Twardowski. 2024. Divide and not forget: Ensemble of selectively trained experts in continual learning. *arXiv preprint arXiv:2401.10191*.
- Ying Shen, Zhiyang Xu, Qifan Wang, Yu Cheng, Wenpeng Yin, and Lifu Huang. 2024. Multimodal instruction tuning with conditional mixture of lora. *arXiv preprint arXiv:2402.15896*.
- Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. 2024. Continual learning of large language models: A comprehensive survey. *arXiv preprint arXiv:2404.16789*.
- Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. 2024a. Self-expansion of pre-trained models with mixture of adapters for continual learning. *arXiv preprint arXiv:2403.18886*.
- Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schütze. 2024b. Rehearsal-free modular and compositional continual learning for language models. *arXiv preprint arXiv:2404.00790*.
- Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. 2023. Orthogonal subspace learning for language model continual learning. *arXiv preprint arXiv:2310.14152*.
- Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. 2023. Next-gpt: Any-to-any multimodal llm. *arXiv preprint arXiv:2309.05519*.
- Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. 2024a. Mixture-of-subspaces in low-rank adaptation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7880–7899, Miami, Florida, USA. Association for Computational Linguistics.
- Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. 2024b. Continual Learning for Large Language Models: A Survey. *arXiv preprint arXiv:2402.01364*.
- Duzhen Zhang, Wei Cong, Jiahua Dong, Yahan Yu, Xiuyi Chen, Yonggang Zhang, and Zhen Fang. 2023. Continual named entity recognition without catastrophic forgetting. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8186–8197.
- Duzhen Zhang, Yahan Yu, Chenxing Li, Jiahua Dong, Dan Su, Chenhui Chu, and Dong Yu. 2024. Mm-llms: Recent advances in multimodal large language models. In *Findings of the Association for Computational Linguistics ACL 2024*.
- Xuanle Zhao, Xuexin Liu, Haoyue Yang, Xianzhen Luo, Fanhu Zeng, Jianling Li, Qi Shi, and Chi Chen. 2025a. Chartedit: How far are mllms from automating chart analysis? evaluating mllms’ capability via chart editing. *arXiv preprint arXiv:2505.11935*.
- Xuanle Zhao, Xianzhen Luo, Qi Shi, Chi Chen, Shuo Wang, Wanxiang Che, Zhiyuan Liu, and Maosong Sun. 2025b. Chartcoder: Advancing multimodal large language model for chart-to-code generation. *arXiv preprint arXiv:2501.06598*.
- Junhao Zheng, Qianli Ma, Zhen Liu, Binqun Wu, and Huawen Feng. 2024a. Beyond Anti-Forgetting: Multimodal Continual Instruction Tuning with Positive Forward Transfer. *arXiv preprint arXiv:2401.09181*.
- Junhao Zheng, Shengjie Qiu, Chengming Shi, and Qianli Ma. 2024b. Towards Lifelong Learning of Large Language Models: A Survey. *arXiv preprint arXiv:2406.06391*.
- Junhao Zheng, Chengming Shi, Xidi Cai, Qiuke Li, Duzhen Zhang, Chenxing Li, Dong Yu, and Qianli Ma. 2025. Lifelong learning of large language model based agents: A roadmap. *arXiv preprint arXiv:2501.07278*.
- Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024. Llama-moe: Building mixture-of-experts from llama with continual pre-training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15913–15923.

Task	Dataset	Instruction	Train Number	Test Number
Grounding	RefCOCO	Please provide the bounding box coordinate of the region	55k	31k
	RefCOCO+ RefCOCOg	this sentence describes: <description>		
Classification	ImageNet	What is the object in the image?	129k	5k
		Answer the question using a single word or phrase		
Image Question Answering (IQA)	VQAv2	Answer the question using a single word or phrase	82k	107k
Knowledge Grounded IQA	ScienceQA	Answer with the option's letter from the given choices directly	12k	4k
Reading Comprehension IQA	TextVQA	Answer the question using a single word or phrase	34k	5k
Visual Reasoning IQA	GQA	Answer the question using a single word or phrase	72k	1k
Blind People IQA	VizWiz	Answer the question using a single word or phrase	20k	8k
OCR IQA	OCR-VQA	Answer the question using a single word or phrase	165k	100k

Table 4: The statistic of multimodal datasets in CoIN (Chen et al., 2024).

Method	Accuracy on Each Task								Overall Results			
	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	Grounding	VQAv2	OCR-VQA	ACC \uparrow	MAA \uparrow	BWT \uparrow	FWT \uparrow
Zero-shot	64.56	48.15	11.82	44.50	9.57	0.00	64.10	27.50	33.78	-	-	-
LoRA	67.69	66.36	53.70	59.30	36.38	63.10	71.00	47.80	41.23	43.35	-16.94	-2.87
	31.05	42.45	29.57	55.57	15.30	40.33	67.75	47.80				
MoELoRA*	73.99	50.91	29.84	60.03	39.44	3.93	73.44	55.10	42.08	46.83	-6.26	-1.15
	54.66	48.82	29.15	56.31	35.97	1.30	55.33	55.10				
ProgLoRA (dynamic)	68.33	56.17	31.98	57.38	40.63	8.46	68.78	57.13	47.09	52.07	-1.51	0.23
	68.07	53.08	30.22	53.16	45.22	1.26	68.58	57.13				

Table 5: Main results on Qwen-VL model. For accuracy of each task of MCIT methods, the first row denotes the results for each task evaluated after its tuning (*i.e.*, $P_{i,i}$ ($i \in [1, N]$)) with the best performance highlighted in **red**, while the second row shows each task's results after tuning the final task (*i.e.*, $P_{N,i}$) with the best ones in **blue**. For overall results, the **bold** highlights the best performance. * represents results from our re-implementation, while others are cited from CoIN (Chen et al., 2024).

A Datasets

Detailed statistics for the 8 datasets in the CoIN benchmark (Chen et al., 2024) are presented in Table 4.

B Compared Methods

We evaluate the ProgLoRA by comparing with these methods: (1) **Zero-shot**: Evaluating each task directly using pre-trained MLLMs without any fine-tuning; (2) **LoRA** (Hu et al., 2022): Sequentially updating knowledge through two low-rank matrices while keeping the pre-trained MLLM parameters intact; (3) **MoELoRA** (Chen et al., 2024): Leveraging multiple independent yet identical LoRA blocks to capture task-specific knowledge across

sequential tasks, achieving state-of-the-art performance on the CoIN benchmark; (4) **LwF** (Li and Hoiem, 2017): Restricting the shared representation layer to remain close to its original state before acquiring the new task; (5) **EWC** (Kirkpatrick et al., 2017): Finetuning the entire model with a regularization loss that restricts parameter updates to avoid disrupting previously learned tasks.

Note that zero-shot approach primarily reflects the reasoning capability of the MLLM backbone on the CoIN benchmark without task-specific adaptation, serving as a lower bound for MCIT methods. we aim for ProgLoRA's overall results to surpass zero-shot.

Task	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	Grounding	VQAv2	OCR-VQA
ScienceQA	75.78							
TextVQA	31.40	51.73						
ImageNet	40.98	13.15	96.70					
GQA	37.42	35.53	6.62	59.42				
VizWiz	47.15	41.69	5.53	45.27	58.88			
Grounding	59.06	13.97	0.25	32.62	30.40	37.50		
VQAv2	33.27	50.48	8.15	51.75	28.52	0.97	64.22	
OCR-VQA	63.09	38.63	10.50	37.38	43.62	0.59	43.15	60.08

Table 6: Detailed results of LLaVA-1.5-7B MoELoRA in CoIN, cited from (Chen et al., 2024).

Task	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	Grounding	VQAv2	OCR-VQA
ScienceQA	76.27							
TextVQA	75.72	60.07						
ImageNet	74.95	58.56	92.65					
GQA	74.60	56.08	88.86	58.45				
VizWiz	75.58	57.38	89.67	54.66	57.33			
Grounding	74.53	56.66	90.20	54.67	56.01	17.68		
VQAv2	73.58	57.02	92.24	52.45	56.43	13.28	64.45	
OCR-VQA	72.70	57.08	91.20	53.12	56.04	12.54	63.64	63.54

Table 7: Detailed results of LLaVA-1.5-7B ProgLoRA (static) in CoIN.

C Evaluation Metrics

We use four metrics: Average Accuracy (ACC) to evaluate after training on the final task, Mean Average Accuracy (MAA) to evaluate throughout training process, Backward Transfer (BWT) to quantify the extent of CF, and Forward Transfer (FWT) to evaluate the extent of KT.

We evaluate outputs of MLLMs by performing word-by-word comparisons with ground truths. Given the diverse output formats across tasks, we employ task-specific evaluation metrics.

In line with the CoIN benchmark (Chen et al., 2024), we evaluate performance on the Image Question Answering task, which includes VQAv2, ScienceQA, TextVQA, GQA, VizWiz, and OCR-VQA, by measuring the accuracy of predicted answers against the ground truth, following the method used in LLaVA (Liu et al., 2023a). For classification tasks, we compare the predicted labels with the actual labels. In the referring expression comprehension (grounding) task, we use the Intersection over Union (IoU) metric, which is commonly used for this purpose, to evaluate prediction accuracy. A prediction is considered correct

if the IoU between the predicted and ground-truth bounding boxes is greater than 0.5.

D Results on Other MLLM Backbone

We also evaluate our ProgLoRA on the **Qwen-VL-7B** (Bai et al., 2023) model. Results are shown in Table 5. For the overall results, **our method significantly outperforms all baselines**, which illustrates the designed task-aware allocation and task recall can solve problems of CF and KT. For accuracy on each task, while the $P_{i,i}$ for some tasks is not as good as the baselines, it still maintains a leading position in $P_{N,i}$ for most tasks. This reflects that although Qwen-VL may be more sensitive to the allocation of LoRA blocks as both MoELoRA and ProgLoRA exhibit lower $P_{i,i}$, **our method can still effectively reduce interference and forgetting among tasks**.

E Detailed Main Results

Detailed results of MoELoRA and ProgLoRA on LLaVA-1.5-7B are shown in Tables 6, 7, and 8.

Task	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	Grounding	VQAv2	OCR-VQA
ScienceQA	76.27							
TextVQA	59.06	60.78						
ImageNet	70.92	52.57	97.32					
GQA	51.56	50.33	79.68	61.27				
VizWiz	65.62	50.79	81.17	48.99	60.16			
Grounding	40.08	47.53	77.75	50.61	48.30	39.35		
VQAv2	76.90	56.79	73.90	53.64	40.63	35.96	65.83	
OCR-VQA	74.84	51.83	83.90	49.93	53.87	31.19	62.71	64.44

Table 8: Detailed results of LLaVA-1.5-7B ProgLoRA (dynamic) in CoIN.

Method	ACC \uparrow	MAA \uparrow	BWT \uparrow	FWT \uparrow
ProgLoRA (dynamic) w/ $\phi = 200$	59.09	62.38	-6.59	1.37
ProgLoRA (dynamic) w/ $\phi = 500$	59.14	62.74	-6.47	1.40
ProgLoRA (dynamic) w/ $\phi = 1000$	59.66	63.23	-6.21	1.38

Table 9: Overall Results of ProgLoRA (dynamic) with different replay amount. **Bold** highlights the best performance.

Method	ACC \uparrow	MAA \uparrow	BWT \uparrow	FWT \uparrow
ProgLoRA (dynamic) w/ $r = 16$	59.09	62.38	-6.59	1.37
ProgLoRA (dynamic) w/ $r = 64$	61.04	64.88	-6.23	1.39
ProgLoRA (dynamic) w/ $r = 128$	62.63	65.01	-6.09	1.43

Table 10: Overall Results of ProgLoRA (dynamic) with different LoRA setting. **Bold** highlights the best performance.

F Effect of Replay Amount

We provide comparison of ProgLoRA (dynamic) with different replay amount ϕ as shown in Table 9.

G Effect of LoRA Settings

We provide comparison of ProgLoRA (dynamic) with different LoRA Rank r as shown in Table 10.

H Effect of Task Order

We compared the alphabet order of tasks as shown in Table 11, which proves that changing the order significantly impacts performance of ProgLoRA. This is because the knowledge acquired from earlier tasks can either facilitate or hinder the learning of subsequent tasks in our proposed LoRA pool. The overall performance in the alphabet order drops, which we attribute to the decline in performance on the GQA and Grounding. In contrast, with the original sequential order, since ScienceQA has few training samples, even a single LoRA block is sufficient for effective learning, providing a solid foundation for subsequent tasks. Therefore, the knowledge acquired early in sequential order, due

to its higher quality, enables better KT. However, since GQA and Grounding have larger datasets and there is no prior knowledge available for assistance at the start of the alphabet order, their performance is poorer. These results demonstrate that the performance improvement of ProgLoRA is attributed to the effective utilization of acquired knowledge. It’s worth noting that GQA shows significant improvement after completing all tasks in alphabet order, further validating that the proposed dynamic allocation can efficiently allocate LoRA blocks.

Moreover, the task order with the worst performance (Alphabet order) still outperforms other comparison methods (such as MoELoRA) in terms of ACC, MAA, BWT, and FWT. This indicates that, while our method is influenced by task order, its performance does not degrade to a level where it is no longer SOTA. It provides evidence from another perspective of the superiority of our method in mitigating CF and promoting KT.

I Effect of Instruction Diversity

We investigated the sensitivity of ProgLoRA to different instruction templates because tasks in CoIN share overlapping instructions as shown in Table 12. We followed Diverse and 10Type instruction templates in Chen et al. (2024). The list of instruction templates for each task is shown in Table 14. **Original:** Certain tasks in the CoIN benchmark (Chen et al., 2024) use similar instructions. **Diverse:** Assigning distinct and specific instruction templates to each task. **10Type:** Randomly selecting an instruction template from a pool of 10 unique templates for each task.

Although ACC is similar to those of the original type, the use of Diverse and 10Type templates both increase MAA by exposing it to a wider variety of instructions. Additionally, both

Order	Accuracy on Each Task								Overall Results			
	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	Grounding	VQAv2	OCR-VQA	ACC↑	MAA↑	BWT↑	FWT↑
Sequential	76.27	60.78	97.32	61.27	60.16	39.35	65.83	64.44	59.09	62.38	-6.59	1.37
	74.84	51.83	83.90	49.93	53.87	31.19	62.71	64.44				
Alphabet	GQA	Grounding	ImageNet	OCR-VQA	ScienceQA	TextVQA	VizWiz	VQAv2	ACC↑	MAA↑	BWT↑	FWT↑
	39.23	8.11	90.75	56.98	69.37	56.52	60.96	69.95	49.51	42.85	-6.96	0.04
	57.03	4.65	66.32	42.42	68.96	54.62	32.18	69.95				

Table 11: The results of LLaVA-1.5-7B ProgLoRA (dynamic) about different task orders. **Bold** highlights the best performance in overall results.

Instruction Type	Accuracy on Each Task								Overall Results			
	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	Grounding	VQAv2	OCR-VQA	ACC↑	MAA↑	BWT↑	FWT↑
Original	76.27	60.78	97.32	61.27	60.16	39.35	65.83	64.44	59.09	62.38	-6.59	1.37
	74.84	51.83	83.90	49.93	53.87	31.19	62.71	64.44				
Diverse	76.27	60.95	97.69	59.88	58.87	37.03	67.34	66.49	59.76	62.72	-5.80	1.32
	75.50	55.44	82.37	53.57	52.38	29.34	63.01	66.49				
10Type	76.85	60.11	97.95	60.32	57.21	38.15	68.93	65.23	59.97	63.74	-5.62	1.41
	75.35	56.26	85.62	52.36	49.14	31.70	64.10	65.23				

Table 12: The results of LLaVA-1.5-7B ProgLoRA (dynamic) about different instruction templates. Meaning of the highlight marker is the same as in Table 1.

Method	Trainable Param.↓	ACC↑	MAA↑	BWT↑	FWT↑
MoELoRA	320M	37.13	42.76	-25.91	-4.01
ProgLoRA (dynamic)	42M	59.09	62.38	-6.59	1.37

Table 13: Efficiency comparison of ProgLoRA and MoELoRA. **Bold** highlights the best performance.

Diverse and 10Type templates lead to an increase in BWT because instruction diversity helps mitigate instruction-following degradation by enhancing adaptability. Although changing the template helps improve performance, the variation in results within ProgLoRA remains below 1.5%, and it consistently outperforms the baselines. This shows that ProgLoRA has low sensitivity to different instructions and can reliably extract task-specific knowledge from them.

J Discussion on Efficiency

We compare the amount of trainable parameters of ProgLoRA (dynamic) with MoELoRA on LLaVA-1.5-7B, with results presented in Table 13. In ProgLoRA, trainable parameters in LoRA pool amount to 42M (0.25% of total parameters), while MoELoRA has 320M trainable parameters (2.06% of total parameters). Therefore, our method significantly reduces the number of trainable parameters, and with the arrival of new tasks, the number of trainable parameters also doesn't increase.

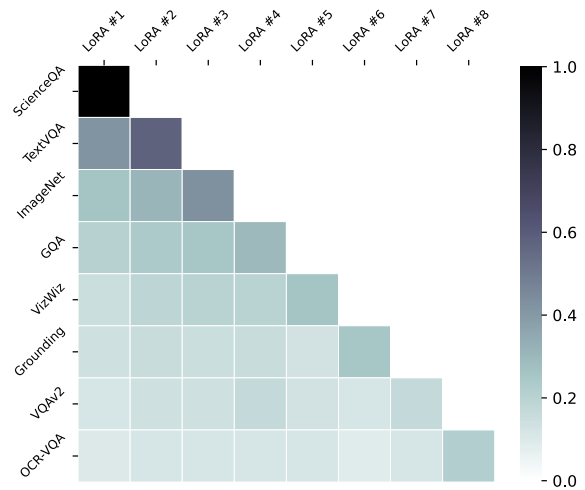


Figure 6: Visualization of allocation weights in ProgLoRA (static).

K Visualization of Allocation Weights

We show the allocation weights after completing the final task (OCR-VQA) in Fig. 6, where the MLLM backbone is LLaVA-1.5-7B and the method is ProgLoRA (static).

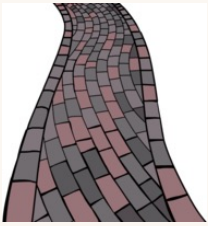
L More Cases

Fig. 7 and 8 show more cases from ScienceQA and OCR-VQA after training on the final task.

Task	Original	Diverse	10Type
ScienceQA	Answer with the option's letter from the given choices directly	Answer with the option's letter from the given choices directly	Answer with the option's letter from the given choices directly
			Select the correct answer from the given choices and respond with the letter of the chosen option
Grounding	Please provide the bounding box coordinate of the region this sentence describes	Please provide the bounding box coordinate of the region this sentence describes	Determine the correct option from the provided choices and reply with its corresponding letter
			Pick the correct answer from the listed options and provide the letter of the selected option
GQA	Answer the question using a single word or phrase	Respond to the question briefly, using only one word or a phrase	Identify the correct choice from the options below and respond with the letter of the correct option
			From the given choices, choose the correct answer and respond with the letter of that choice
ImageNet	Answer the question using a single word or phrase	Express your answer in a single word or a short, descriptive phrase	Choose the right answer from the options and respond with its letter
			Select the correct answer from the provided options and reply with the letter associated with it
OCR-VQA	Answer the question using a single word or phrase	Condense your answer for each question into a single word or concise phrase	From the given choices, select the correct answer and reply with the letter of the chosen option
			Identify the correct option from the choices provided and respond with the letter of the correct option
TextVQA	Answer the question using a single word or phrase	Capture the essence of your response in a single word or a concise phrase	From the given choices, pick the correct answer and respond by indicating the letter of the correct option
			Identify and provide the bounding box coordinates that match the description given in this sentence
VizWiz	Answer the question using a single word or phrase	Provide a succinct response with a single word or phrase	Extract and provide the bounding box coordinates based on the region described in the sentence
			Please provide the bounding box coordinate of the region this sentence describes
VQAv2	Answer the question using a single word or phrase	Answer the question using a single word or phrase	Find and provide the bounding box coordinates for the region mentioned in the sentence
			Provide the coordinates of the bounding box that correspond to the region described in the sentence


Table 14: The list of different instructions template for each task (Chen et al., 2024).

User:
Which material is this path made of?
A. plastic
B. brick
Answer with the option's letter from the given choices directly.




MoELoRA: plastic ProgLoRA (static): A ProgLoRA (dynamic): B

User:
Are the bubbles in soda a solid, a liquid, or a gas?
A. a gas
B. a solid
C. a liquid
Answer with the option's letter from the given choices directly.



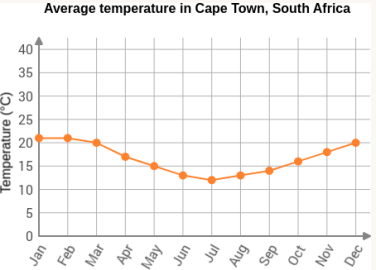
MoELoRA: C ProgLoRA (static): A ProgLoRA (dynamic): A

User:
This organism is *Asimina triloba*. It is a member of the plant kingdom. *Asimina triloba* is commonly called the pawpaw. Pawpaw trees grow in the southeastern part of the United States. They have large, sweet fruit. The fruit is sometimes called a prairie banana. Does *Asimina triloba* have cells that have a nucleus?
A. Yes
B. no
Answer with the option's letter from the given choices directly.



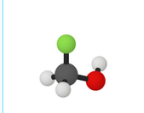
MoELoRA: no ProgLoRA (static): A ProgLoRA (dynamic): A

User:
Use the graph to answer the question below. Which month is the hottest on average in Cape Town?
A. April, May, and November
B. December, January, February, and March
C. August, September, October, and November
Answer with the option's letter from the given choices directly.




MoELoRA: B ProgLoRA (static): A ProgLoRA (dynamic): C


User:
Look at the models of molecules below. Select the elementary substance.
A. Fluoromethanol
B. tetraphosphorus
C. methane
Answer with the option's letter from the given choices directly.




fluoromethanol



tetraphosphorus



methane



MoELoRA: C ProgLoRA (static): A ProgLoRA (dynamic): C

Figure 7: Comparison between MoELoRA and ProgLoRA on cases from ScienceQA after training on the final task.



Figure 8: Comparison between MoELoRA and ProgLoRA on cases from OCR-VQA after training on the final task.