Lightweight Progressive LoRA for Multimodal Continual Instruction Tuning

Yahan Yu[1] Chenhui Chu[1]
[1]Kyoto University
yahan@nlp.ist.i.kyoto-u.ac.jp, chu@i.kyoto-u.ac.jp

Abstract

Multimodal Continual Instruction Tuning (MCIT) enables Multimodal Large Language Models (MLLMs) to evolve tasks without costly retraining, but it remains challenged by catastrophic forgetting (CF) and negative knowledge transfer (NKT). Existing methods adopted Mixture-of-Experts (MoE), yet sharing a fixed set of LoRA blocks across tasks often causes knowledge interference. To address this, we propose **Prog**ressive **LoRA** (ProgLoRA), which maintains a progressive LoRA pool and allocates a new LoRA block for each incremental task. Experiments on recent MCIT benchmarks show that ProgLoRA consistently outperforms existing methods, while employing lightweight optimization.

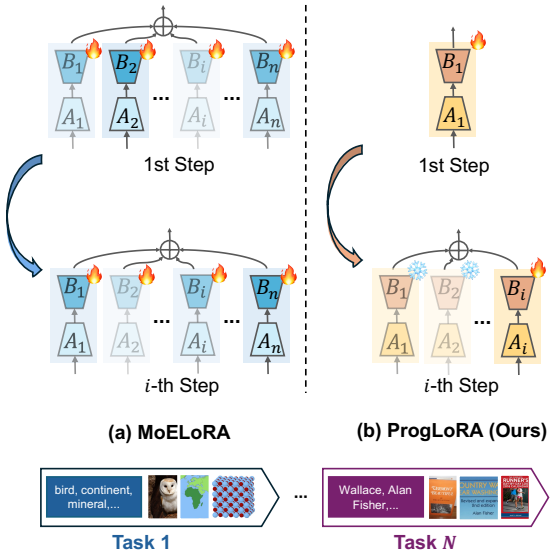

Figure 1 A comparison between MoELoRA and ProgLoRA.

1 Introduction

Multimodal Large Language Models (MLLMs) [1] have shown strong performance across diverse vision–language tasks [2], typically supported by instruction tuning [3] for multi-task learning. However, real-world applications require MLLMs to continuously adapt to new instructions as knowledge evolves. Most existing MLLMs [4] remain static, and retraining from scratch for each new instruction is computationally impractical. To address this limitation, recent work [5] formulates the problem as Multimodal Continual Instruction Tuning (MCIT) [6], which incrementally updates MLLMs while preserving performance on previously learned tasks.

MCIT faces two key challenges: catastrophic forgetting (CF), which represents that performance on the old tasks is dropped after training on new tasks, and negative knowledge transfer (NKT), which represents that performance on the new task is limited by the old tasks in different domains. Mixture-of-Experts LoRA (MoELoRA) [7] addresses these issues by using multiple LoRA blocks to capture task-specific knowledge across sequential tasks. However, it shares a fixed set of LoRA blocks across tasks (Fig. 1(a)), causing previously learned knowledge to be overwritten by subsequent tasks, which limits its effectiveness in handling CF and NKT.

To mitigate CF and NKT, we propose **Prog**ressive **LoRA** (ProgLoRA). ProgLoRA maintains a progressive LoRA pool, where a new LoRA block is trained for each incremental task while previously learned blocks are frozen (Fig. 1(b)). This design isolates task-specific knowledge in independent LoRA blocks, preventing overwriting and thus alleviating CF. Although only LoRA block of the current task is updated, frozen blocks remain active during training, enabling prior knowledge to support new tasks and improving KT. ProgLoRA further introduces two key components: task-aware allocation, which selects and fuses relevant LoRA blocks to leverage acquired knowledge, and task recall, which realigns the model with previously learned

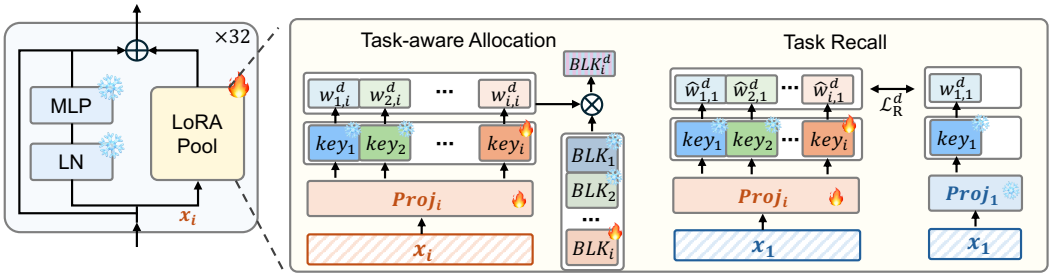

Figure 2 The overall structure of ProgLoRA.

tasks to further reduce forgetting.

Meanwhile, since progressively introducing task-specific LoRA blocks may lead to excessive parameter growth under a large number of incremental tasks, we further explore ProgLoRA (light) for lightweight optimization. Prior works show that lower layers in LLM tend to encode general knowledge while higher layers are more task-specific. Thus, ProgLoRA (light) is applied only to some upper layers in the LLM, while the remaining layers share a single LoRA block during training. This design significantly reduces the overall parameter overhead while preserving the benefits of progressive adaptation.

Our contributions can be summarized as follows:

1. We propose ProgLoRA, where different knowledge is stored in separate LoRA blocks, thereby minimizing task interference.
2. We design the task-aware allocation to select and fuse LoRA blocks, and task recall to constrain the model updates. We design ProgLoRA (light) for lightweight optimization.
3. Experiments on LLaVA-1.5 using the latest MCIT benchmark demonstrate that ProgLoRA outperforms.

2 Related Works

2.1 MLLMs

MLLMs extend LLMs [8] to jointly process visual and textual inputs by combining strong language reasoning with visual representations. Representative models such as LLaVA [1] and MiniGPT-v2 [3] employ projection layers to align frozen LLMs with visual encoders, while InstructBLIP [4] and BLIP-2 adopt Q-Former–based [9] designs to bridge modality gaps. Together with instruction tuning, these approaches have significantly advanced multimodal reasoning and understanding across diverse tasks.

2.2 MCIT

To keep pace with continuously evolving knowledge, MLLMs must be incrementally updated rather than retrained from scratch [10]. MCIT addresses this need by enabling efficient incremental adaptation across tasks. Recent benchmarks and methods [6] focus on mitigating CF and NKT, for example, by aggregating LoRA experts as in MoELoRA [5]. However, sharing experts across tasks often leads to knowledge interference, limiting long-term performance.

3 Problem Definition

Continual learning [11] addresses sequentially evolving tasks while avoiding costly full retraining. MCIT extends this paradigm to MLLMs by incrementally adapting a model $\mathcal{M}$ to a sequence of tasks $\mathcal{T}_1, \ldots, \mathcal{T}_N$ via instruction tuning, while maintaining performance on previously learned tasks. Each task $\mathcal{T}_i$ is associated with a dataset $\mathcal{D}_i = (X_{i,j}^v, X_{i,j}^q, X_{i,j}^a)_{j=1}^{M_i}$, where X^v, X^q, and X^a denote image, instruction, and answer tokens, respectively.

4 Approach

As illustrated in Fig. 2, ProgLoRA addresses CF and NKT through a progressive and task-aware LoRA framework, which maintains a progressive LoRA pool $\{BLK_1, \ldots, BLK_N\}$, a **task-aware allocation** module, and a **task recall** module.

4.1 Task-aware Allocation

At the training stage of $\mathcal{T}_i$, a key vector $\boldsymbol{key}_n$ is assigned to each BLK_n ($n \in [1, i]$), and weights are computed. For better alignment with the key vector spaces, we first fed $\boldsymbol{x}_{i,j}^q$, the embedding of input $X_{i,j}^q$, into a projector $Proj_i$ (we omit the q and j for simplicity):

$$\boldsymbol{h}_i = \mathrm{LN}(\boldsymbol{W}_2\varphi(\boldsymbol{W}_1\boldsymbol{x}_{i[max]} + b_1) + b_2), \quad (1)$$

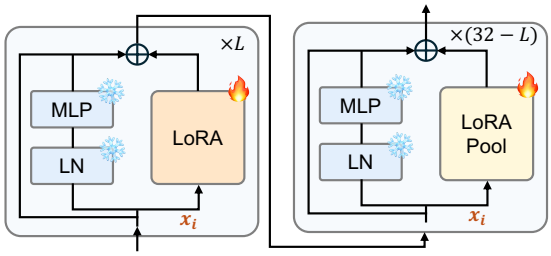

Figure 3 The lightweight variant of ProgLoRA.

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

$$w^d_{n,i} = Softmax(\boldsymbol{h}_i \boldsymbol{key}_n / \alpha_{tem}), \tag{2}$$

where α_{tem} represents temperature factor to improve robustness. Finally, all the existing LoRA blocks are fused to BLK^d_i as follows:

$$\theta_{BLK^d_i} = \sum_{n=1}^{i} w^d_{n,i} \theta_{BLK_n} \tag{3}$$

4.2 Task Recall

As tasks are sequentially trained, the $Proj$ is updated continuously. To ensure that inputs from $\mathcal{T}_p$ still accurately perform the weights and identify the specific fusion of LoRA blocks, we introduce the **KL-divergence-based task recall**. This is achieved by using a small amount of samples from $\mathcal{T}_p$, which are used to generate pseudo 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 $\boldsymbol{w}^d_p = \{w^d_{1,p}, ..., w^d_{p,p}\}$ for samples $\mathcal{T}_p$, each previous sample is fed into layers in Eq. (1) and (2) to get $\hat{\boldsymbol{w}}^d_i = \{\hat{w}^d_{1,i}, ..., \hat{w}^d_{i,i}\}$ at the training of $\mathcal{T}_i$ and minimize the corresponding KL-divergence loss.

4.3 Lightweight Optimization

Progressively introducing task-specific LoRA blocks may result in excessive parameter growth when handling a large number of incremental tasks. To address this issue, we propose ProgLoRA (light) as shown in Fig. 3 that improves scalability while preserving the core benefits of progressive adaptation. Prior works show that different layers of LLMs capture knowledge at different levels of abstraction: lower layers tend to encode more general and transferable knowledge, while higher layers are more task- or instruction-specific. Based on this observation, we reduce the number of incremental LoRA blocks by applying a task-shared LoRA module to the bottom L layers, which is trained jointly across all tasks. The remaining upper layers follow the original ProgLoRA scheme, where task-specific LoRA blocks are incrementally introduced.

5 Experimental Settings

Dataset We follow the CoIN benchmark [5], which includes 8 multimodal datasets: [12], TextVQA [13], ImageNet [14], GQA [15], VizWiz [16], Grounding [17, 18], VQAv2 [19], and OCR-VQA [2].

Metrics We adopt four metrics: **Mean Final Accuracy (MFN)** to average the accuracies of all tasks after the final training, **Mean Average Accuracy (MAA)** to report the average accuracies on each task during the training process, **Backward Transfer (BWT)** to measure the accuracy difference between the final and immediate training of each task, and **Forward Transfer (FWT)** to measure the MFN difference between our method and non-seqential training.

Implementation We follow the CoIN [5] and utilize LLaVA-1.5-7B [1] as the backbone. In ProgLoRA, rank r is set to 16, α is 32, with the replay amount set to 200 per previous task. In ProgLoRA (light), L is set to 3. All experiments are conducted on 4 NVIDIA A6000 GPUs.

Baselines Details are shown in Appendix A.

6 Results and Analysis

6.1 Main Results

Results are summarized in Table 1. Our method significantly outperforms all baselines, which illustrates the designed task-aware allocation and task recall can solve problems of CF and NKT. ProgLoRA both outperforms traditional continual learning methods (i.e., LwF and EWC) and the latest methods (e.g., MoCL and O-LoRA), demonstrating the advantages of our approach in the MCIT task. Specifically, O-LoRA imposes orthogonal regularization to constrain the updates of the newly added LoRA block. However, relying solely on the newly added LoRA block is insufficient to effectively mitigate CF, highlighting the advantage of our method. For ProgLoRA (light), despite reducing parameterization, it still consistently outperforms all baselines, demonstrating that the effectiveness of our ap-

Method	Accuracy on Each Task								Overall Results			
	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	Grounding	VQAv2	OCR-VQA	MFN⇑	MAA⇑	BWT⇑	FWT⇑
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				
O-LoRA*	76.27	54.23	96.10	53.66	53.22	30.72	52.69	55.47	43.28	50.28	-15.76	-4.26
	62.92	38.31	56.11	30.57	43.87	14.79	44.23	55.47				
SEMA*	76.27	58.14	96.56	59.14	59.13	35.60	55.49	52.12	47.88	54.78	-17.49	-2.75
	68.63	43.27	58.45	38.38	50.19	23.73	48.34	52.12				
MoCL*	76.25	53.41	97.29	61.03	58.56	38.58	64.32	56.32	49.07	55.21	-14.11	-1.09
	65.25	45.65	59.25	39.87	48.89	24.37	52.98	56.32				
ProgLoRA	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**				
ProgLoRA (light)	74.32	57.54	97.01	60.98	58.90	35.43	64.98	62.46	56.39	61.00	-7.56	-0.36
	71.98	45.37	79.73	48.94	50.89	29.82	61.99	62.46				

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 with the best performance highlighted in **red**, while the second row shows each task's results after tuning the final task 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 [5].

Variant	MFN⇑	MAA⇑	BWT⇑	FWT⇑
ProgLoRA	**59.09**	**62.38**	**-6.59**	**1.37**
w/o task recall	57.52	62.12	-8.07	1.08
ProgLoRA (light)	56.39	61.00	-7.56	-0.36
$L = 6$	51.34	59.83	-10.16	-12.96

Table 2 The ablation study of ProgLoRA on the LLaVA-1.5-7B backbone. **Bold** highlights the best performance.

proach does not depend on excessive model capacity. We observe only an obvious degradation in FWT, mainly due to the lightweight design's limited representational capacity, which restricts its ability to model fine-grained cross-task dependencies. We also discuss the efficiency in the Appendix B and show the detailed main results in the Appendix C.

6.2 Ablation Study

As shown in Table 2, we analyzed each component. For ProgLoRA, removing task recall leads to a significantly greater occurrence of CF, underscoring the critical role of this component in enabling adaptive LoRA block allocation. For ProgLoRA (light), we observe that when the number of blocks is set to $L = 6$, the model achieves competitive performance across almost all evaluation metrics compared to stronger configurations and baselines. In particular, metrics related to overall accuracy and forgetting are largely preserved, indicating that a moderate number of blocks is sufficient to capture task-specific knowledge and maintain stable performance. The only notable exception is FWT, which explicitly measures the effectiveness of knowledge transfer across tasks. We attribute this degradation to the limited diversity and capacity of reusable representations when L is larger, which restricts the generalized knowledge to future tasks. Nevertheless, these results suggest that $L = 6$ offers a strong balance between efficiency and performance, where most benefits of ProgLoRA are retained while incurring only a reduction in KT capability.

7 Conclusion

In this work, we propose a progressive framework to address catastrophic forgetting and knowledge transfer in continual learning, which consistently outperforms strong baselines. Experiments show that our method effectively mitigates forgetting while promoting knowledge reuse, and its lightweight variant remains competitive with substantially reduced parameters. Future work could explore extending the proposed framework to broader task sequences and different backbone models, with the goal of improving its generalization and adaptability across diverse scenarios.

Acknowledgments

This work was supported by JST BOOST Grant Number JPMJBS2407 and JSPS KAKENHI Grant Number JP23K28144.

References

[1] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023.

[2] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In **2019 international conference on document analysis and recognition (ICDAR)**, pp. 947–952. IEEE, 2019.

[3] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. **arXiv preprint arXiv:2310.09478**, 2023.

[4] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven C. H. Hoi. InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023.

[5] Cheng Chen, Junchen Zhu, Xu Luo, Heng Tao Shen, Jingkuan Song, and Lianli Gao. CoIN: A benchmark of continual instruction tuning for multimodel large language models. In **The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track**, 2024.

[6] Jinghan He, Haiyun Guo, Ming Tang, and Jinqiao Wang. Continual instruction tuning for large multimodal models. **arXiv preprint arXiv:2311.16206**, 2023.

[7] Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. **arXiv preprint arXiv:2310.18339**, 2023.

[8] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. **arXiv preprint arXiv:2210.11416**, 2022.

[9] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. 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**, pp. 19730–19742, 2023.

[10] Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual Learning for Large Language Models: A Survey. **arXiv preprint arXiv:2402.01364**, 2024.

[11] Grzegorz Rypeść, Sebastian Cygert, Valeriya Khan, Tomasz Trzciński, Bartosz Zieliński, and Bart lomiej Twardowski. Divide and not forget: Ensemble of selectively trained experts in continual learning. **arXiv preprint arXiv:2401.10191**, 2024.

[12] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. **Advances in Neural Information Processing Systems**, Vol. 35, pp. 2507–2521, 2022.

[13] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**, pp. 8317–8326, 2019.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In **2009 IEEE conference on computer vision and pattern recognition**, pp. 248–255. Ieee, 2009.

[15] Drew A Hudson and Christopher D Manning. 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**, pp. 6700–6709, 2019.

[16] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In **Proceedings of the IEEE conference on computer vision and pattern recognition**, pp. 3608–3617, 2018.

[17] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**, pp. 787–798, 2014.

[18] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In **Proceedings of the IEEE conference on computer vision and pattern recognition**, pp. 11–20, 2016.

[19] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 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**, pp. 6904–6913, 2017.

[20] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-Rank Adaptation of Large Language Models. In **International Conference on Learning Representations**, 2022.

[21] Zhizhong Li and Derek Hoiem. Learning without forgetting. **IEEE transactions on pattern analysis and machine intelligence**, Vol. 40, No. 12, pp. 2935–2947, 2017.

[22] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. **Proceedings of the national academy of sciences**, Vol. 114, No. 13, pp. 3521–3526, 2017.

[23] Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. **arXiv preprint arXiv:2310.14152**, 2023.

[24] Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. Self-expansion of pre-trained models with mixture of adapters for continual learning. **arXiv preprint arXiv:2403.18886**, 2024.

[25] Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schütze. Rehearsal-free modular and compositional continual learning for language models. **arXiv preprint arXiv:2404.00790**, 2024.

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 3 Detailed results of ProgLoRA.

Task	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	Grounding	VQAv2	OCR-VQA
ScienceQA	74.32							
TextVQA	58.48	57.34						
ImageNet	69.13	53.09	97.01					
GQA	50.11	50.02	80.37	60.98				
VizWiz	67.63	50.11	80.09	47.96	58.90			
Grounding	40.12	45.98	75.37	49.25	49.36	35.43		
VQAv2	72.90	54.89	72.91	51.37	38.98	33.69	64.98	
OCR-VQA	71.98	45.37	79.73	48.94	50.89	29.82	61.99	62.46

Table 4 Detailed results of ProgLoRA (light) with $L = 3$.

A 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** [20]: Sequentially updating knowledge through two low-rank matrices while keeping the pre-trained MLLM parameters intact; (3) **MoELoRA** [5]: 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** [21]: Restricting the shared representation layer to remain close to its original state before acquiring the new task; (5) **EWC** [22]: Finetuning the entire model with a regularization loss that restricts parameter updates to avoid disrupting previously learned tasks; (6) **O-LoRA** [23]: Employing orthogonal subspace learning to facilitate continual learning in language models; (7) **SEMA** [24]: Learning automatically to reuse and expand modules without relying on memory replay; (8) **MoCL** [25]: A modular and compositional framework for continual learning.

B Efficiency Discussion

We compare the amount of trainable parameters of ProgLoRA with MoELoRA on LLaVA-1.5-7B. 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). For the lightweight variant with $L = 3$, trainable parameters amount to 38M (0.22% of total parameters). The light variant achieves a performance–efficiency trade-off, making it well-suited for resource-constrained scenarios. 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 too much due to the LoRA architecture.

C Detailed Main Results

Detailed results of ProgLoRA and ProgLoRA (light) are shown in Tables 3 and 4.