Advancing SMoE for Continuous Domain Adaptation of MLLMs: Adaptive Router and Domain-Specific Loss

Liang Zhang[1,3 *], **Ziyao Lu**[2 *], **Fandong Meng**[2], **Hui Li**[1], **Jie Zhou**[2], **Jinsong Su**[1,3,4 †]
[1]School of Informatics, Xiamen University, China
[2]Pattern Recognition Center, WeChat AI, Tencent Inc, China
[3]Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan (Xiamen University), Ministry of Culture and Tourism, China
[4]Shanghai Artificial Intelligence Laboratory
lzhang@stu.xmu.edu.cn, jssu@xmu.edu.cn, ziyaolu@tencent.com

Abstract

Recent studies have explored Continual Instruction Tuning (CIT) in Multimodal Large Language Models (MLLMs), with a primary focus on Task-incremental CIT, where MLLMs are required to continuously acquire new tasks. However, the more practical and challenging Domain-incremental CIT, focused on the continual adaptation of MLLMs to new domains, remains underexplored. In this paper, we propose a new Sparse Mixture of Expert (SMoE) based method for domain-incremental CIT in MLLMs. During training, we learn a domain-specific SMoE module for each new domain in every FFN sub-layer of MLLMs, preventing catastrophic forgetting caused by inter-domain conflicts. Moreover, we equip the SMoE module with a domain-specific autoregressive loss (DSAL), which is used to identify the most suitable SMoE module for processing each test instruction during inference. To further enhance the SMoE module's ability to learn domain knowledge, we design an adaptive threshold-based router (AT-Router) that allocates computing resources (experts) to instruction tokens based on their importance. Finally, we establish a new benchmark to evaluate the efficacy of our method and advance future research. Extensive experiments show that our method consistently outperforms all competitive baselines.

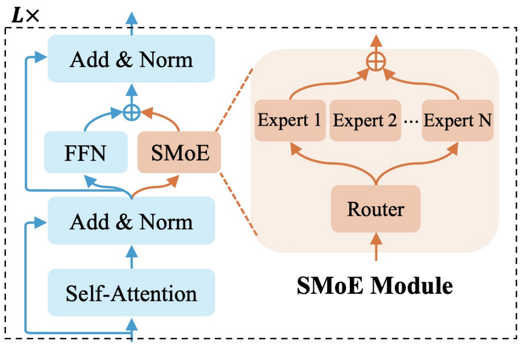


Figure 1: SMoE-based CL method, where an SMoE module is appended to each FFN sub-layer in LLMs.

1 Introduction

Multimodal Large Language Models (MLLMs), as key building blocks for general-purpose assistants, have garnered significant attention in the research community (Chen et al., 2022b; Dai et al., 2023; Zhu et al., 2024; Liu et al., 2024a; Lan et al., 2025). In their success, instruction tuning plays a crucial role, which uses carefully-crafted multi-task instruction data to train MLLMs for generating accurate and human-friendly responses. Furthermore, recent studies (He et al., 2023; Zhai et al., 2023; Chen et al., 2024a) have begun to explore more realistic Continual Instruction Tuning (CIT) in MLLMs, requiring MLLMs to continually learn new knowledge while retaining previously acquired knowledge. However, they mainly focus on *Task-incremental CIT* that requires MLLMs to continually learn new tasks. In contrast, the exploration of *Domain-incremental CIT* in MLLMs, aimed at the continuous adaptation of MLLMs to new domains, remains insufficient.

Indeed, the default multi-task instruction tuning already endows MLLMs with strong task generalization ability, allowing them to tackle unseen tasks in a zero-shot manner (Liu et al., 2023b; Xu et al., 2023). However, the limited domain generalization ability of MLLMs restricts their effectiveness in solving domain-specific problems (Li et al., 2024; Shi et al., 2024). This indicates the greater practicality of domain-incremental CIT over task-incremental CIT. Moreover, domain-incremental CIT is often more challenging due to a higher risk of catastrophic forgetting, resulting from conflicts in tasks and data distributions across domains.

To prevent the forgetting problem of Large Language Models (LLMs) during Continuous Learning

This work is done when Liang Zhang was interning at Pattern Recognition Center, WeChat AI, Tencent Inc, China.
*Equal contribution
†Corresponding author

(CL), several Sparse Mixture of Experts (SMoE)-based methods (Chen et al., 2023; Shen et al., 2023; Dou et al., 2024a) have been proposed, benefiting from the excellent training and inference efficiency inherent in SMoE (Shazeer et al., 2017). As shown in Figure 1, these methods first append an SMoE module, including an expert group and a router, into each FFN sub-layer in LLMs. Then, they sequentially add new experts into the expert group to learn new knowledge while freezing prior parameters, so as to ensure efficient training and avoid catastrophic forgetting. While the number of experts will continue to increase in the CL process, the router only selects the k most relevant experts to process each token, ensuring efficient inference. Nonetheless, these methods have not been applied to domain-incremental CIT in MLLMs and exhibit several notable deficiencies. (1) They allocate the same computational resources (k experts) to each token regardless of its importance, limiting their performance. (2) During training, the router only selects partial tokens to train each expert, causing the issue of insufficient expert training (Gou et al., 2023; Ding et al., 2024). (3) In these methods, the continual updating of the router during CL limits its accuracy in selecting experts for earlier tasks, thus leading to catastrophic forgetting.

In this paper, we propose a new SMoE-based method for domain-incremental CIT in MLLMs, which can continually adapt MLLMs to new domains without catastrophic forgetting. Its overall architecture is illustrated in Figure 2. During CL training, we independently train a domain-specific SMoE module for each new domain in every FFN sub-layer of MLLMs, preaventing catastrophic forgetting caused by inter-domain conflicts. To enhance the SMoE module's ability for learning new domain knowledge, we design a novel adaptive threshold-based router (**AT-Router**) for it. Unlike traditional routers, our AT-Router first dynamically calculates an adaptive threshold for each token to assess its importance, and then allocates experts to process the token based on this importance. Hence, our method can achieve superior performance by activating fewer experts than existing methods.

Moreover, we equip the SMoE module with a domain-specific autoregressive loss (**DSAL**) that is used to accurately identify the most suitable SMoE module for processing each test instruction during inference. Specifically, we first create a virtual expert for each SMoE module by averaging the parameters of all experts in its expert group. Next, we employ the virtual expert and a linear projector to compute an autoregressive loss, specific to the domain of the current SMoE module, for each instruction. In our method, DSAL plays several crucial roles: **First**, during training, the virtual expert allows every expert in the SMoE module to access all instruction tokens in the training set, effectively alleviating the issue of insufficient expert training in traditional SMoE. **Second**, during inference, the SMoE module (trained with DSAL) yields a smaller DSAL for test instructions from its corresponding domain compared to those from other domains. Thus, DSAL intuitively indicates the domain of the test instruction. Notably, since both the virtual expert and the projector are domain-specific, our DSAL remains highly stable without continuous changes during the CL process.

To evaluate the efficacy of our method, we construct a new benchmark for domain-incremental CIT in MLLMs. It covers three common domains, including *Medicine*, *Chart*, and *Math*, each involving instructions from a variety of tasks. Experimental results show that our method consistently outperforms all competitive baselines. Extensive ablation studies further demonstrate the effectiveness of various components in our method.

2 Our Method

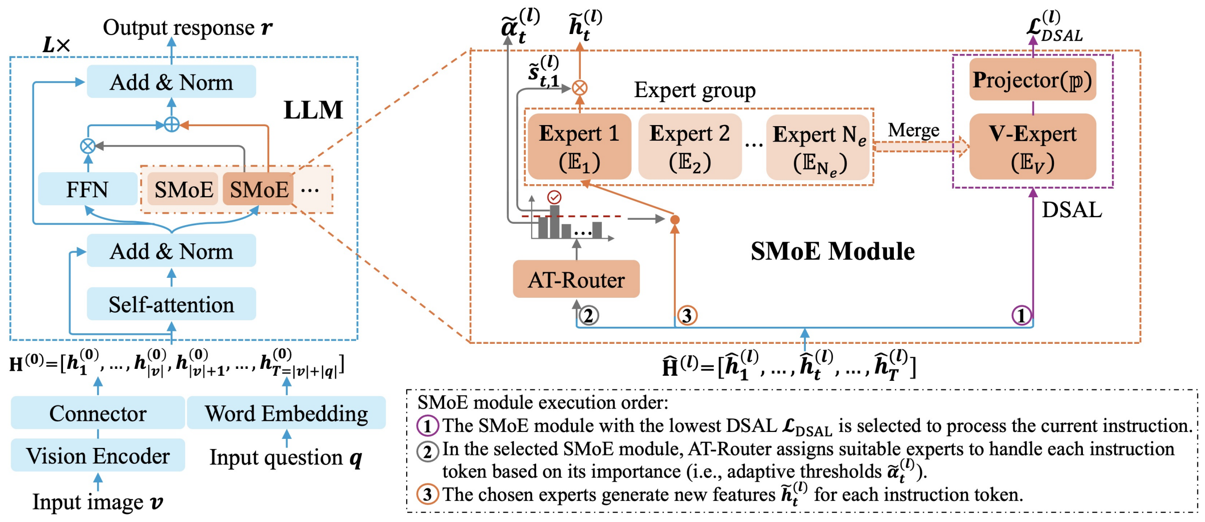
In this section, we provide a detailed description for our proposed method. Specifically, we first present the problem formulation for domain-incremental CIT in MLLMs, and then introduce the overall architecture of our method. Finally, we elaborate on the CL training process involved in our method.

2.1 Problem Formulation

In domain-incremental CIT, MLLMs are required to continuously learn new domains while retaining previously acquired domain knowledge. Thus, MLLMs are successively fine-tuned on a sequence of domain-specific datasets: $[\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_{N_d}]$, where N_d denotes the domain number and $\mathcal{D}_n$ is the multi-task instruction dataset for the n-th domain. In $\mathcal{D}_n$, each instruction instance consists of an instruction $(\boldsymbol{v}, \boldsymbol{q})$ and its response $\boldsymbol{r}$, where $\boldsymbol{v}$ signifies an input image and $\boldsymbol{q}$ is an input question.

2.2 Overall Architecture

As depicted in the left part of Figure 2, MLLMs often consist of three components: a visual encoder, a connector, and a LLM. Given a multimodal instruction $(\boldsymbol{v}, \boldsymbol{q})$, we first feed the input image $\boldsymbol{v}$


Output response $\boldsymbol{r}$
$L\times$
Add & Norm
LLM
FFN
SMoE
SMoE
Add & Norm
Self-attention
$\mathbf{H}^{(0)}=[\boldsymbol{h}_1^{(0)},\ldots,\boldsymbol{h}_{|v|}^{(0)},\boldsymbol{h}_{|v|+1}^{(0)},\ldots,\boldsymbol{h}_{T=|v|+|q|}^{(0)}]$
Connector
Word Embedding
Vision Encoder
Input question $\boldsymbol{q}$
Input image $\boldsymbol{v}$
$\tilde{\boldsymbol{\alpha}}_t^{(l)}$
$\tilde{\boldsymbol{h}}_t^{(l)}$
$\tilde{\boldsymbol{s}}_{t,1}^{(l)}$
Expert group
Expert 1 ($\mathbb{E}_1$)
Expert 2 ($\mathbb{E}_2$)
Expert N_e ($\mathbb{E}_{N_e}$)
Merge
$\mathcal{L}_{DSAL}^{(l)}$
Projector($\mathbb{p}$)
V-Expert ($\mathbb{E}_V$)
DSAL
AT-Router
SMoE Module
$\widehat{\mathbf{H}}^{(l)}=[\widehat{\boldsymbol{h}}_1^{(l)},\ldots,\widehat{\boldsymbol{h}}_t^{(l)},\ldots,\widehat{\boldsymbol{h}}_T^{(l)}]$

SMoE module execution order:
① The SMoE module with the lowest DSAL $\mathcal{L}_{\text{DSAL}}$ is selected to process the current instruction.
② In the selected SMoE module, AT-Router assigns suitable experts to handle each instruction token based on its importance (i.e., adaptive thresholds $\tilde{\boldsymbol{\alpha}}_t^{(l)}$).
③ The chosen experts generate new features $\tilde{\boldsymbol{h}}_t^{(l)}$ for each instruction token.

Figure 2: Illustration of our method. During training, we independently train a domain-specific SMoE module for each new domain in every FFN sub-layer of the LLM. During inference, for each test instruction, we use DSAL to identify the SMoE module corresponding to its domain in each FFN sub-layer, effectively processing the instruction.

into the visual encoder to extract its visual features. Then, the connector maps these visual features into the LLM's embedding space to obtain a feature sequence $\mathbf{H}_v^{(0)}\in\mathbb{R}^{|v|\times d}$, where $|v|$ denotes the number of visual tokens in $\boldsymbol{v}$. Concurrently, we input the question $\boldsymbol{q}$ into the LLM's embedding layer to generate its feature sequence $\mathbf{H}_q^{(0)}\in\mathbb{R}^{|q|\times d}$, where $|q|$ is the number of text tokens in $\boldsymbol{q}$. Lastly, we concatenate these two feature sequences: $\mathbf{H}^{(0)}=[\mathbf{H}_v^{(0)};\mathbf{H}_q^{(0)}]\in\mathbb{R}^{T\times d}$ (with $T=|v|+|q|$), and feed it into the LLM to generate the corresponding response $\boldsymbol{r}$.

Since all layers of the LLM are identical, we use the l-th layer as an example to explain its computation process. Here, we first feed the output $\mathbf{H}^{(l-1)}$ of the previous layer into the multi-head self-attention (MHSA) sub-layer of the current layer, yielding updated instruction features $\widehat{\mathbf{H}}^{(l)}$:

$$\widehat{\mathbf{H}}^{(l)}=\text{Norm}\big(\mathbf{H}^{(l-1)}+\text{MHSA}\left(\mathbf{H}^{(l-1)}\right)\big), \quad (1)$$

where $\text{Norm}(\cdot)$ refers to Root Mean Square Layer Normalization (RMSNorm) (Zhang et al., 2019).

In the FFN sub-layer of the l-th layer, we separately learn a domain-specific **SMoE** module for each new domain to prevent the catastrophic forgetting caused by inter-domain conflicts. The right part of Figure 2 illustrates that our SMoE module consists of an **expert group** and an adaptive threshold-based router (**AT-Router**). Notably, to generate the correct response for the current instruction, it is essential to accurately identify the most suitable SMoE module for processing the instruction. To this end, we equip each SMoE module with a domain-specific autoregressive loss (**DSAL**).

DSAL. For the i-th SMoE module in the FFN sub-layer of the l-th layer, we create a virtual expert $\mathbb{E}_V^{(l,i)}(\cdot)$ by averaging the parameters of all experts in its expert group (see $\xRightarrow{\textbf{Merge}}$ in Figure 2). Then, we leverage the virtual expert with a linear projector $\mathbb{P}^{(l,i)}(\cdot)$ to compute an autoregressive loss (i.e., DSAL) on the feature sequence $\widehat{\mathbf{H}}^{(l)}$. However, the absence of visual tokens in the LLM vocabulary (i.e., no labels) restricts the computation of DSAL. Therefore, we use the LLM embedding matrix $\boldsymbol{W}_{\text{em}}$ to calculate the distribution vectors $\mathbf{P}$ for all instruction tokens, serving as their soft labels: $\mathbf{P}=[p_1,\cdots,p_T]=\text{softmax}(\mathbf{H}^{(0)}\boldsymbol{W}_{\text{em}})$. Next, we utilize the feature $\widehat{\mathbf{h}}_t^{(l)}\in\widehat{\mathbf{H}}^{(l)}$ of each token to predict the distribution $q_{t+1}^{(l,i)}$ of its subsequent token:

$$\begin{aligned} q_{t+1}^{(l,i)} &= \text{softmax}\big(\overline{\mathbf{h}}_t^{(l,i)}\overline{\boldsymbol{W}}_{\text{em}}^{(l,i)}\big), \quad (2)\\ \text{where}\quad \overline{\mathbf{h}}_t^{(l,i)} &= \mathbb{P}^{(l,i)}\big(\mathbb{E}_V^{(l,i)}(\widehat{\mathbf{h}}_t^{(l)})\big),\\ \overline{\boldsymbol{W}}_{\text{em}}^{(l,i)} &= \mathbb{P}^{(l,i)}\big(\boldsymbol{W}_{\text{em}}\big). \end{aligned}$$

Here, the projector $\mathbb{P}^{(l,i)}(\cdot)$ aims to lower the computational cost of DSAL by reducing the feature and embedding dimensions of tokens. Finally, the current SMoE module's DSAL $\mathcal{L}_{\text{DSAL}}^{(l,i)}$ is given by

$$\mathcal{L}_{\text{DSAL}}^{(l,i)}=-\tfrac{1}{T-1}\textstyle\sum_{t=2}^{T}p_t^{\mathsf{T}}\log(q_t^{(l,i)}). \quad (3)$$

Generally, the SMoE module produces a smaller $\mathcal{L}_{\text{DSAL}}$ for instructions from its corresponding domain compared to those from other domains. Thus, following the above steps, we calculate the DSAL for all N_d SMoE modules in the current FFN sub-layer, and select the SMoE module with the smallest DSAL to process the current instruction (see **step ①** in Figure 2):

$$m = \arg\min_i([\mathcal{L}_{\text{DSAL}}^{(l,1)}, \cdots, \mathcal{L}_{\text{DSAL}}^{(l,N_d)}]), \quad (4)$$

Notably, the virtual experts and projector responsible for calculating DSAL are domain-specific and remain unchanged when the model learns other domains, effectively ensuring the stability of DSAL.

AT-Router. In the selected m-th SMoE module, for each instruction token $\widehat{\mathbf{h}}_t^{(l)} \in \widehat{\mathbf{H}}^{(l)}$, we use AT-Router, a multi-layer perceptron (MLP), to compute its adaptive threshold $\alpha_t^{(l)}$ and relevance score $s_{t,n}^{(l)}$ to every expert in the expert group:

$$\boldsymbol{S}_t^{(l)} = [\alpha_t^{(l)}, s_{t,1}^{(l)}, \cdots, s_{t,N_e}^{(l)}] = \text{Route}_{\text{AT}}(\widehat{\mathbf{h}}_t^{(l)}), \quad (5)$$

where N_e is the total number of experts in the expert group of the SMoE module, and $s_{t,n}^{(l)}$ indicates the relevance score of the current token to the n-th expert. As shown in the AT-Router part (**step ②**) in Figure 2, we activate only those experts with scores exceeding $\alpha_t^{(l)}$ to process the token. Accordingly, we employ an activation function $\text{Act}(\cdot)$ to remove the scores of non-activated experts from $\boldsymbol{S}_t^{(l)}$, followed by normalizing the scores of activated experts and the adaptive threshold $\alpha_t^{(l)}$ through a softmax function $\text{softmax}(\cdot)$:

$$\widetilde{\boldsymbol{S}}_t^{(l)} = \text{softmax}(\text{Act}(\boldsymbol{S}_t^{(l)})). \quad (6)$$

Intuitively, $1 - \alpha_t^{(l)}$ reflect the token's importance. The above process ensures that vital tokens (lower $\alpha_t^{(l)}$) are allocated more experts, whereas less significant tokens (higher $\alpha_t^{(l)}$) receive fewer or no expert assignments. Unlike traditional threshold-based methods that manually set a fixed threshold for all tokens, our method computes a dynamic threshold for each instruction token based on its current context, assigning experts more effectively.

Expert group. It consists of N_e identical experts, each is instantiated as a FFN:

$$\mathbb{E}(\widehat{\mathbf{h}}_t^{(l)}) = (\sigma(\widehat{\mathbf{h}}_t^{(l)} \boldsymbol{W}_{\text{gate}}) \circ (\widehat{\mathbf{h}}_t^{(l)} \boldsymbol{W}_{\text{up}})) \boldsymbol{W}_{\text{dow}} \quad (7)$$

where $\sigma(\cdot)$ is the SwiGLU activation function, $\circ$ refers to element-wise multiplication, $\boldsymbol{W}_{\text{gate/up}} \in \mathbf{R}^{d \times d'}$ and $\boldsymbol{W}_{\text{dow}} \in \mathbf{R}^{d' \times d}$ are learnable parameters, d and d' are the hidden state dimensions of the LLM and the expert, respectively. We leverage d' to control the parameter number of each expert.

After deriving the normalized score $\tilde{s}_{t,n}^{(l)} \in \widetilde{\boldsymbol{S}}_t^{(l)}$ for each activated expert via Eq. 6, we compute the weighted sum of the output features from these activated experts to form the final output of the selected m-th SMoE module (see **step ③** in Figure 2):

$$\widetilde{\mathbf{h}}_t^{(l)} = \sum_{\tilde{s}_{t,n}^{(l)} \in \widetilde{\boldsymbol{S}}_t^{(l)}} \left(\tilde{s}_{t,n}^{(l)} \times \mathbb{E}_n^{(l,m)}\left(\widehat{\mathbf{h}}_t^{(l)}\right)\right), \quad (8)$$

where $\mathbb{E}_n^{(l,m)}$ is the n-th expert in the expert group. Finally, we combine the output of the original FFN sub-layer with that of the selected SMoE module using the normalized adaptive threshold $\tilde{\alpha}_t^{(l)} \in \widetilde{\boldsymbol{S}}_t^{(l)}$ to generate the l-th layer output $\mathbf{H}^{(l)}$ of the LLM:

$$\mathbf{H}^{(l)} = \text{Norm}\left(\check{\mathbf{H}}^{(l)} + \widehat{\mathbf{H}}^{(l)}\right), \quad (9)$$

$$\check{\mathbf{H}}^{(l)} = [\check{\mathbf{h}}_1^{(l)}, \cdots, \check{\mathbf{h}}_T^{(l)}], \quad (10)$$

$$\check{\mathbf{h}}_t^{(l)} = \text{len}(\widetilde{\boldsymbol{S}}_t^{(l)}) \times \left(\widetilde{\mathbf{h}}_t^{(l)} + \tilde{\alpha}_t^{(l)} \times \text{FFN}(\widehat{\mathbf{h}}_t^{(l)})\right), \quad (11)$$

where $\text{len}(\widetilde{\boldsymbol{S}}_t^{(l)})$ represents the number of elements in $\widetilde{\boldsymbol{S}}_t^{(l)}$, used to scale the features of important tokens (with more activated experts). It enhances the effect of key instruction tokens in response generation and accelerates model training.

After obtaining the final feature $\mathbf{H}^{(L)}$ for the current instruction, the LLM autoregressively generates its response $\boldsymbol{r} = [r_1, \cdots, r_{|\boldsymbol{r}|}]$:

$$r_t = \arg\max_r P(r | \mathbf{H}^{(L)}; r_{1:t-1}). \quad (12)$$

where L indicates the number of layers in the LLM, $|\boldsymbol{r}|$ refers to the number of text tokens in $\boldsymbol{r}$.

2.3 Model Training

During CL training, upon the arrival of a new domain, we first add a new SMoE module into each FFN sub-layer in the LLM. Then, we freeze the prior model parameters and use three loss terms to jointly train the newly added SMoE modules in the new domain: $\mathcal{L} = \mathcal{L}_1 + \beta\mathcal{L}_2 + \eta\mathcal{L}_3$, where the hyper-parameters β and η balance their contributions.

The first loss term $\mathcal{L}_1$ is an autoregressive loss derived on the token sequence of the response $\boldsymbol{r}$:

$$\mathcal{L}_1 = -\tfrac{1}{|\boldsymbol{r}|} \sum_{t=1}^{|\boldsymbol{r}|} \log P(r_{t+1} | \mathbf{H}^{(L)}, r_{1:t}). \quad (13)$$

It is our primary loss, aiming at training the model to generate accurate responses for instructions.

The second loss term $\mathcal{L}_2$ is defined as the sum of the DSAL for all newly added SMoE modules: $\mathcal{L}_2 = \sum_{l=1}^{L} \mathcal{L}_{\text{DSAL}}^{(l)}$, allowing these modules to effectively capture the data distribution of the new domain. Through this loss and the virtual expert for calculating DSAL, we can utilize all instruction tokens to train each expert in these SMoE modules, mitigating the issue of insufficient expert training.

The third loss term $\mathcal{L}_3$ is designed to balance the activation rates of experts in each SMoE module. To achieve this, we employ the adaptive threshold as a reference point, decreasing the scores of over-activated experts and increasing those of under-activated experts. Specifically, for the newly added SMoE module in the l-th layer of the LLM, we first compute the average score $\bar{s}_n^{(l)}$ for each expert and the average adaptive threshold $\bar{\alpha}^{(l)}$ based on Eq. 5: $[\bar{\alpha}^{(l)}, \bar{s}_1^{(l)}, \cdots, \bar{s}_{N_e}^{(l)}] = \frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|} \boldsymbol{S}_t^{(l)}$, where $|\mathcal{B}|$ denotes the number of instruction tokens in the current training batch $\mathcal{B}$. Next, we derive the probability score $\widehat{p}_n^{(l)} = e^{\bar{s}_n^{(l)}}/(e^{\bar{s}_n^{(l)}} + e^{\bar{\alpha}^{(l)}})$ of each expert relative to $\bar{\alpha}^{(l)}$. To identify over-activated and under-activated experts, we also count the activation frequency $\boldsymbol{f}^{(l)} = [f_1^{(l)}, \cdots, f_{N_e}^{(l)}]$ of each expert from $\widetilde{\boldsymbol{S}}_t^{(l)}$ in Eq. 6. Finally, we formalize the expert balance loss $\mathcal{L}_{\mathrm{Bal}}^{(l)}$ of the current SMoE module as a binary cross-entropy:

$$\mathcal{L}_{\mathrm{Bal}}^{(l)} = -\frac{1}{N_e}\sum_{n=1}^{N_e}\Big[\mathbf{I}\big(f_n^{(l)} < \tfrac{1}{N_e}\big)\log\big(\widehat{p}_n^{(l)}\big) + \mathbf{I}\big(f_n^{(l)} \geq \tfrac{1}{N_e}\big)\log\big(1 - \widehat{p}_n^{(l)}\big)\Big], \tag{14}$$

where $\mathbf{I}(\cdot)$ is the indicator function, $\mathbf{I}(f_n^{(l)} < \frac{1}{N_e})$ and $\mathbf{I}(f_n^{(l)} \geq \frac{1}{N_e})$ indicate the under-activated and over-activated experts, respectively. To balance the experts in all newly added SMoE modules, our third loss term is defined as $\mathcal{L}_3 = \sum_{l=1}^{L} \mathcal{L}_{\mathrm{Bal}}^{(l)}$.

In this way, our method can learn each domain independently and without mutual interference, offering several advantages: (1) It minimizes conflicts among domains in the CL process, preventing catastrophic forgetting. (2) This allows parallel adaptation of our method to multiple domains, thus accelerating the development of multi-domain MLLMs. (3) It also ensures that our method is unaffected by the order of domain learning.

3 Benchmark

Although recent studies (Zhai et al., 2023; Chen et al., 2024a) have explored task-incremental CIT in MLLMs and introduced relevant benchmarks, domain-incremental CIT remains underexplored due to the lack of benchmarks. Thus, we construct a new benchmark to comprehensively investigate the behavior of MLLMs in domain-incremental CIT. Table 1 presents the statistics for our benchmark.

Domain	Task	Dataset	Train	Test
Medicine	VQA	VQA-RAD	1,793	451
		PathVQA	25,913	6,719
		SLAKE	11,934	2,094
		LLaVA-Med	56,649	–
Chart	VQA	ChartQA	28,138	2,490
	Chart-to-Table	ChartQA	19,373	1,509
	Chart-to-Text	Pew	7,892	1,393
		Statista	24,368	–
Math	MWP	IconQA	20,771	1,741
		CLEVR-Math	4,840	445
		TabMWP	20,639	1,743
	GPS	Geometry3K	8,854	835
		GeoQA+	15,660	1,433
		UniGeo	11,014	901
		GEOS	483	24

Table 1: The statistic of datasets in our benchmark, where VQA=Visual Question Answering, MWP=Math Word Problem and GPS=Geometry Problem Solving.

Given the time-intensive nature of data collection, our benchmark, as the first of its kind, initially focuses on three common domains: *Medicine*, *Chart*, and *Math*. Meanwhile, we collect various publicly available datasets from each domain and convert their instances into a unified instruction format. To better reflect real-world scenarios, our benchmark covers as many task types as possible in each domains. In total, the benchmark includes 3 domains, 5 task and 14 datasets, with each domain involving about 90K instructions. In the future, we will incorporate more domains into our benchmark to better facilitate subsequent studies.

4 Experiment

4.1 Datasets & Evaluation Metrics

We conduct experiments on our benchmark. In **Appendix A**, we provide detailed descriptions of each task and dataset in our benchmark, along with their commonly used metrics and relevant cases. Notably, since the performance of our method remains unaffected by the order of domain learning, we randomly adopt a continual learning sequence: *Medicine→Chart→Match.*

4.2 Settings

Following prior studies (Han et al., 2023a; Li et al., 2024; Shi et al., 2024; Chen et al., 2024a; Cao et al., 2024), we adopt the widely used LLaVA-1.5 (Liu et al., 2024a) as our base MLLM. Notably, the default instruction tuning dataset of LLaVA-1.5 does not overlap with our benchmark, ensuring more accurate evaluations of various CL methods. We fine-tune our model for 2 epochs on each domain, using a learning rate of 2e-4 and a batch size of 16. For our SMoE module, we set the expert dimension d' to 128 and the expert number N_e to 4. Moreover, both hyper-parameters β and η in our training objective $\mathcal{L}$ are set to 0.1. As shown in **Appendices B and C**, the optimization of these hyper-parameters can further enhance our method's performance.

4.3 Baselines

We compare our method with the following three categories of competitive CL methods:

Replay-based Methods. Following prior studies (Chaudhry et al., 2019a; Huang et al., 2021; He et al., 2021), after learning each domain, we store 1% of representative instruction instances chosen via the K-means algorithm from the domain. Then, we replay these stored instances while learning a new domain. We refer to this method as *Replay*.

Regularization-based Methods. Here, we consider two classical methods: (1) *EWC* (Kirkpatrick et al., 2017b). This method uses the Laplace approximation with the Fisher information matrix to identify key parameters for previously seen tasks. Next, it slows down learning on these parameters when learning new tasks. (2) *LWF* (Li et al., 2017b). It adopts knowledge distillation to prevent the forgetting of prior knowledge.

Architecture-based Methods. These methods use independent parameters to learn distinct tasks, avoiding catastrophic forgetting. In this aspect, the typical methods mainly include: (1) *CODA* (Smith et al., 2023). It learns a set of task-specific soft prompts for each task while introducing an orthogonal constraint to prevent conflicts between tasks. (2) *M-LoRA*. This method learns a task-specific LoRA for each task, all of which are loaded and integrated during inference. (3) *O-LoRA* (Wang et al., 2023). Here, an orthogonal constraint is introduced on top of M-LoRA to further avoid catastrophic forgetting. (4) *MoELoRA* (Dou et al., 2024b; Liu et al., 2024b). Unlike the above methods, this method attempts to utilize different experts in the Dense MoE (DMoE) framework to learn various tasks, reducing the risk of conflicts between tasks. (5) *SMoELoRA*. Based on MOELoRA, this method uses a sparse activation router to further reduce conflicts between tasks. (6) *L-SMoE* (Qin et al., 2022) and (7) *MoLM* (Shen et al., 2023). Both methods are designed to incrementally add new experts into the SMoE module to accommodate new tasks.

Like our method, these baselines are developed on LLaVA-1.5 with parameter-efficient fine-tuning techniques, including LoRA (Hu et al., 2022), DMoE (Ma et al., 2018), and SMoE (Shazeer et al., 2017). To ensure a fair comparison, all methods use a comparable average number of parameters to learn each domain (see **Line 2** in Table 3). For further details on the implementation of baselines, please refer to **Appendix E**. Besides, to validate our SMoE module's ability to learn domain knowledge, we compare it with traditional LoRA , DMoE, and SMoE in both single-domain and multi-domain learning scenarios. Here, SMoE selects a single most relevant expert to process each token, while DMoE uses all experts for every token.

4.4 Main Results

The experimental results on our benchmark are shown in Table 2. After carefully analyzing these results, we draw several conclusions:

First, the original LLaVA-1.5 exhibits significantly poor performance across various domains in our benchmark, indicating that its pre-training and instruction fine-tuning do not cover these domains.

Second, in the single-domain learning scenario where the model is only trained and tested on a single domain, our method consistently outperforms all baselines. This demonstrates the powerful capability of our SMoE module, equipped with AT-Router and DSAL, for learning domain knowledge. Particularly, our method exceeds its theoretical upper bound, DMoE, across all domains. The main reason is that our AT-Router can more effectively alleviate the conflict between various tasks in the same domain, and such conflict has been found by prior studies (Gou et al., 2023; Chen et al., 2024b).

Third, in the multi-domain scenarios, we assume that instruction data of all domains can be obtained at once, which can be regarded as a special case (upper bound) of replay-based approaches. As shown in the second part of Table 2, our method exceeds other baselines across all domains in this scenario. We also note that most methods perform worse in the multi-domain scenarios than in single-domain one. This is mainly attributed to the conflicts in task and data distribution across domains.

Lastly, we compare our method to baselines in the continual learning scenario, as shown in the third part of Table 2. Our method exhibits a signif-

	Model	Medicine			Chart			Math
		VQA-RAD	PathVQA	SLAKE	ChartQA	Chart-to-Table	Chart-to-Text	MWP&GPS
		Closed(**Acc.**) / Open(**Recall**)			**Relax_Acc.**	**RMS$_{F1}$**	**BLEU4**	**Acc.**
	LLaVA-1.5	2.10/13.27	4.62/8.87	7.33/28.92	9.27	9.97	4.20	8.93
Single–Domain Learning	LoRA	68.40/35.37	90.09/33.07	82.65/82.25	54.13	56.08	9.28	37.23
	DMoE	72.40/36.48	90.24/34.82	82.77/83.38	55.18	57.32	9.94	38.62
	SMoE	65.20/32.66	87.59/30.43	80.09/80.05	50.72	54.51	7.37	36.08
	Ours	74.10/36.87	91.35/35.51	83.96/84.61	55.58	57.94	10.27	39.14
Multi–Domain Learning	LoRA	66.10/33.56	88.32/31.73	80.15/80.66	52.21	56.57	8.39	32.74
	DMoE	69.20/35.12	89.02/33.16	81.21/82.36	54.31	56.64	9.03	34.05
	SMoE	65.00/32.60	87.60/30.06	79.76/80.00	50.37	54.04	7.19	33.63
	Ours	72.70/36.98	89.89/34.55	82.85/83.98	55.02	57.69	9.84	37.69
Continual Learning	Replay	59.60/21.66	64.77/12.31	48.25/48.63	20.51	22.73	4.45	36.79
	EWC	60.70/23.35	65.81/14.55	51.46/52.87	23.92	25.77	4.89	35.55
	LWF	61.30/24.83	65.59/15.74	55.38/57.16	26.12	29.45	6.07	35.87
	CODA	62.20/25.66	66.62/16.02	58.46/62.31	30.51	32.73	6.45	34.19
	M-LoRA	64.80/29.75	72.82/21.68	65.15/66.24	40.42	42.62	6.54	30.97
	O-LoRA	66.50/30.71	76.35/22.68	77.15/67.24	45.42	49.62	7.31	36.52
	MoELoRA	60.90/23.66	66.43/16.11	56.38/57.52	25.73	28.94	4.51	38.06
	SMoELoRA	59.50/23.13	66.09/17.77	55.16/58.61	23.82	27.76	5.04	35.10
	L-SMoE	65.20/28.69	70.14/20.25	62.50/63.01	38.53	41.05	6.18	38.19
	MoLM	63.10/25.37	66.09/18.07	60.35/61.26	36.77	38.21	5.79	37.64
	Ours	**74.10/36.95**	**91.12/35.33**	**84.06/84.61**	**55.58**	**57.94**	**10.31**	**39.10**

Table 2: The results of our method and baselines on our benchmark.

icant advantage over all baselines, particularly in early-learned domains, such as Medicine. A comparison of our method's performance in the single-domain learning and continual learning reveals that it suffers almost no catastrophic forgetting. This implies that our DSAL can accurately identify the SMoE module corresponding to the test instruction's domain in each layer of the LLM. We further assess its accuracy, achieving a score of **97.2%**.

In addition, Table 3 reports the average number of experts activated by the relevant methods to handle each instruction token, along with the corresponding number of activated parameters. We observe that our method activates an average of only 0.81 experts to process each token, resulting in a significantly lower number of activated parameters than other methods. Further analysis of our AT-Router is presented in **Appendices C and G**.

Model	LoRA	DMoE	SMoE	SMoE (Ours)
Learnable	221M	208M	208M	208M
Activated	221M	208M	52M	**42M**
Top-K	–	top-4	top-1	**top-0.81**

Table 3: Statistics on the average learnable/activated parameters per domain. Here, "Top-K" means the average number of experts activated to process each token.

4.5 Ablation Study

We further conduct extensive ablation studies by removing different components from our method to investigate their different impacts. We compare our method with the following variants in Table 4.

(1) *w/o AT-Router*. In this variant, we replace our AT-Router with the router from the traditional SMoE, which activates the most relevant one expert for each token. While this variant activates more experts for each token, its performance remains inferior to our method (See **Line (1)**). This confirms that allocating experts based on token importance can indeed achieve better model performance with fewer expert activations. Moreover, this variant is equivalent to the traditional SMoE equipped with our DSAL, yet its performance in the continuous learning scenario still exceeds that of the SMoE in the single-domain learning scenario (See Table 2). It affirms our DSAL's ability to alleviate the issue of insufficient expert training in SMoE.

(2) *DSAL→EDI*. Following prior studies (Jang et al., 2023; Bohao et al., 2024), we replace DSAL in our method with an External Domain Identifier (EDI) to ascertain the domain of each instruction. Specifically, this variant first uses the image and text encoders of the CLIP model (Radford et al., 2021) to generate representations of all instructions in the training set. Then, the instruction representations in the same domain are averaged to form a prototype vector for each domain. During infer-

Model	Medicine			Chart			Math
	VQA-RAD	PathVQA	SLAKE	ChartQA	Chart-to-Table	Chart-to-Text	MWP&GPS
Ours	**74.10/36.95**	**91.12/35.33**	**84.06/84.61**	**55.58**	**57.94**	**10.31**	**39.10**
(1) *w/o* AT-Router	67.00/34.41	89.11/32.28	81.80/81.49	52.65	55.04	8.16	36.75
(2) DSAL→EDI	66.70/32.52	84.47/29.78	77.97/77.06	50.36	52.94	7.60	37.48
(3) *w/o* DSAL	64.50/30.18	72.97/22.14	64.73/65.80	40.17	42.83	6.96	32.07
(4) *w/o* $\mathcal{L}_3$	70.30/34.13	87.49/32.55	80.90/81.77	52.53	55.44	8.14	37.11
(5) *w/o* $\text{len}(\widetilde{\boldsymbol{S}}_t^{(l)})$	71.69/35.34	89.13/33.26	81.77/82.89	53.93	56.25	8.87	38.25

Table 4: Ablation results of our method on our benchmark. EDI=External Domain Identifier.

ence, the test instruction's domain is identified by its nearest prototype vector. As shown in **Line (2)**, this variant causes a consistent drop in performance across all domains. This can be attributed to two factors: 1) The accuracy of EDI is significantly lower than that of our DSAL (**78.4%** vs. **97.2%**). 2) The absence of DSAL leads to insufficient expert training, limiting the model performance.

(3) *w/o DSAL*. This variant removes DSAL from our method, relying only on AT-Router to indirectly achieve domain identification for test instructions. As shown in **Line (3)**, this variant also shows a significant performance decline across all domains. These results clearly illustrate the significant contribution of DSAL to the effectiveness of our method.

(4) *w/o* $\mathcal{L}_3$. As described in Section 2.3, based on our adaptive threshold, we design $\mathcal{L}_3$ to balance the expert activations in the SMoE module. To verify its efficacy, this variant eliminates it from our training objective $\mathcal{L}$. The results in **Line (4)** reveal that balancing expert activation is essential for improving the performance of our method.

(5) *w/o* $\text{len}(\widetilde{\boldsymbol{S}}_t^{(l)})$. In Eq. 11, we use $\text{len}(\widetilde{\boldsymbol{S}}_t^{(l)})$ to scale the representations of important tokens while accelerating model training. As shown in **Line (5)**, the removal of $\text{len}(\widetilde{\boldsymbol{S}}_t^{(l)})$ consistently degrades the performance of our method across all domains, thereby confirming its effectiveness.

Notably, we further explore the generalization ability of our method to unseen relevant domains in **Appendix D**. In **Appendix F**, we also investigate the efficacy of our method in task-incremental CIT.

5 Related Work

Continual Learning. In this regard, models are required to continually learn new knowledge without forgetting previously acquired knowledge. To do this, three categories of CL methods have been proposed: (1) *replay-based* methods (Chaudhry et al., 2019b; Scialom et al., 2022; Zhang et al., 2023; Huang et al., 2024). By retraining on historical instances while learning new knowledge, these methods prevent models from forgetting prior knowledge. However, historical examples are not always available and require additional training and storage costs. (2) *regularization-based* methods (Li et al., 2017a; Kirkpatrick et al., 2017a; Feng et al., 2022; Zheng et al., 2023; Wang et al., 2024b). These methods alleviate catastrophic forgetting by constraining updates on crucial parameters that store prior knowledge during the learning of new knowledge, which, however, may impair the model's ability to acquire new knowledge. (3) *architecture-based* methods (Mallya et al., 2018; Gururangan et al., 2021; Qin et al., 2022; Yu et al., 2024). They use untapped or newly added parameters to learn new knowledge, minimizing inter-task conflicts and mitigating forgetting. Notably, these method will continually increase the model's parameters during CL, limiting their inference efficiency.

Sparse Mixture of Experts. By replacing the FFN sub-layer in the standard Transformer with the SMoE module, SMoE (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Jiang et al., 2024) aims to scale the model's parameters while maintaining training and inference efficiency. Generally, each SMoE module consists of a group of FFN-based experts and a router. The router only selects the k most relevant experts to process each token. Given the improved inference efficiency of SMoE, several SMoE-based methods (Chen et al., 2023; Shen et al., 2023; Dou et al., 2024a) have been proposed for CL in LLMs. They first append a SMoE module to each FFN sub-layer in LLMs, and then sequentially add new experts to the SMoE module for acquiring new knowledge. Since their routers are typically continually updated in the CL process, it is difficult for them to accurately identify suitable experts for instructions of earlier tasks during inference. Moreover, they suffer from the inherent issue of insufficient expert training in SMoE

(Gou et al., 2023; Ding et al., 2024).

Benchmark for CIT in MLLMs. Some recent studies (He et al., 2023; Zhai et al., 2023; Chen et al., 2024a) have explored task-incremental CIT in MLLMs and introduced relevant benchmarks. However, domain-incremental CIT remains unexplored in MLLMs and lacks relevant benchmarks. Thus, we construct a new benchmark for domain-incremental CIT in MLLM, comprising instruction data for various tasks in three different domains.

6 Conclusion

In this paper, we propose a SMoE-based method for domain-incremental CIT in MLLMs, which can continually adapt MLLMs to new domains without catastrophic forgetting. During CL training, it learns a domain-specific SMoE module for each new domain in every FFN sub-layer of the LLM, minimizing inter-domain conflicts and avoiding catastrophic forgetting. To enhance the SMoE module's ability for learning domain knowledge, we design a novel router AT-Router that allocates experts to process each tokens based on its importance. We also equip each SMoE module with a DSAL, which is used to identify the most suitable SMoE module for processing each test instruction during inference. Moreover, we construct a new benchmark for domain-incremental CIT in MLLMs to promote future research. Experimental results on our benchmark validate the efficacy of our method.

Acknowledgments

The project was supported by National Key R&D Program of China (No. 2022ZD0160501), Natural Science Foundation of Fujian Province of China (No. 2024J011001), and the Public Technology Service Platform Project of Xiamen (No.3502Z20231043). We also thank the reviewers for their insightful comments.

Limitations

The limitations of our method mainly include following two aspects: 1) Although our benchmark covers three commonly used multimodal domains, more domains can be incorporated in the future. 2) While our method has demonstrated improved domain generalization, further progress is required.

References

PENG Bohao, Zhuotao Tian, Shu Liu, Ming-Chang Yang, and Jiaya Jia. 2024. Scalable Language Model with Generalized Continual Learning. In *Proceedings of ICLR*.

Jie Cao, Jing Xiao, Jie Cao, and Jing Xiao. 2022. An augmented benchmark dataset for geometric question answering through dual parallel text encoding. In *Proceedings of COLING*.

Meng Cao, Yuyang Liu, Yingfei Liu, Tiancai Wang, Jiahua Dong, Henghui Ding, Xiangyu Zhang, Ian Reid, and Xiaodan Liang. 2024. Continual LLaVA: Continual Instruction Tuning in Large Vision-Language Models. *arXiv:2411.02564*.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. 2019a. On tiny episodic memories in continual learning. *arXiv:1902.10486*.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. 2019b. On tiny episodic memories in continual learning. *arXiv:1902.10486*.

Cheng Chen, Junchen Zhu, Xu Luo, Hengtao Shen, Lianli Gao, and Jingkuan Song. 2024a. CoIN: A Benchmark of Continual Instruction tuNing for Multimodel Large Language Model. In *Proceedings of NeurIPS*.

Jiaqi Chen, Tong Li, Jinghui Qin, Pan Lu, Liang Lin, Chongyu Chen, and Xiaodan Liang. 2022a. Unigeo: Unifying geometry logical reasoning via reformulating mathematical expression. In *Proceedings of EMNLP*.

Jun Chen, Han Guo, Kai Yi, Boyang Li, and Mohamed Elhoseiny. 2022b. Visualgpt: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of CVPR*.

Shaoxiang Chen, Zequn Jie, and Lin Ma. 2024b. Llavamole: Sparse mixture of lora experts for mitigating data conflicts in instruction finetuning mllms. *arXiv:2401.16160*.

Wuyang Chen, Yanqi Zhou, Nan Du, Yanping Huang, James Laudon, Zhifeng Chen, and Claire Cui. 2023. Lifelong language pretraining with distribution-specialized experts. In *Proceedings of ICML*.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning. In *Proceedings of NeurIPS*.

Yifeng Ding, Jiawei Liu, Yuxiang Wei, Terry Yue Zhuo, and Lingming Zhang. 2024. XFT: Unlocking the Power of Code Instruction Tuning by Simply Merging Upcycled Mixture-of-Experts. In *Proceedings of ACL*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, et al. 2024a. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of ACL*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, et al. 2024b. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of Findings of ACL*.

William Fedus, Barret Zoph, Noam Shazeer, William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *JMLR*.

Tao Feng, Mang Wang, Hangjie Yuan, Tao Feng, Mang Wang, and Hangjie Yuan. 2022. Overcoming catastrophic forgetting in incremental object detection via elastic response distillation. In *Proceedings of CVPR*.

Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T Kwok, and Yu Zhang. 2023. Mixture of cluster-conditional lora experts for vision-language instruction tuning. *arXiv:2312.12379*.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A Smith, and Luke Zettlemoyer. 2021. Demix layers: Disentangling domains for modular language modeling. In *Proceedings of NAACL*.

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023a. Chartllama: A multimodal llm for chart understanding and generation. *arXiv:2311.16483*.

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023b. Chartllama: A multimodal llm for chart understanding and generation. *arXiv:2311.16483*.

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

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James Glass, and Fuchun Peng. 2021. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *Proceedings of EACL*.

Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. 2020. Pathvqa: 30000+ questions for medical visual question answering. *arXiv:2003.10286*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of ICLR*.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. 2024. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In *Proceedings of ACL*.

Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. 2021. Continual learning for text classification with information disentanglement based regularization. In *Proceedings of NAACL*.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2023. Exploring the benefits of training expert language models over instruction tuning. In *Proceedings of ICML*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv:2401.04088*.

Shankar Kantharaj, Rixie Tiffany Ko Leong, Xiang Lin, Ahmed Masry, Megh Thakkar, Enamul Hoque, and Shafiq Joty. 2022. Chart-to-text: A large-scale benchmark for chart summarization. In *Proceedings of ACL*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017a. Overcoming catastrophic forgetting in neural networks. *PNAS*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017b. Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences*.

Zhibin Lan, Liqiang Niu, Fandong Meng, Wenbo Li, Jie Zhou, and Jinsong Su. 2025. AVG-LLaVA: A Large Multimodal Model with Adaptive Visual Granularity. In *Proceedings of Findings of ACL*.

Jason J Lau, Soumya Gayen, Asma Ben Abacha, and Dina Demner-Fushman. 2018. A dataset of clinically generated visual questions and answers about radiology images. *Scientific data*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *Proceedings of ICLR*.

Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. 2024. Llavamed: Training a large language-and-vision assistant for biomedicine in one day. In *Proceedings of NeurIPS*.

Zhizhong Li, Derek Hoiem, Zhizhong Li, and Derek Hoiem. 2017a. Learning without forgetting. *IEEE TPAMI*.

Zhizhong Li, Derek Hoiem, Zhizhong Li, and Derek Hoiem. 2017b. Learning without forgetting. *IEEE TPAMI*.

Adam Dahlgren Lindström, Savitha Sam Abraham, Adam Dahlgren Lindström, and Savitha Sam Abraham. 2022. Clevr-math: A dataset for compositional language, visual and mathematical reasoning. In *Proceedings of NeSy*.

Bo Liu, Li-Ming Zhan, Li Xu, Lin Ma, Yan Yang, and Xiao-Ming Wu. 2021. Slake: A semantically-labeled knowledge-enhanced dataset for medical visual question answering. In *Proceedings of ISBI*.

Fangyu Liu, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. 2023a. Deplot: One-shot visual language reasoning by plot-to-table translation. In *Proceedings of Findings of ACL*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of CVPR*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023b. Visual instruction tuning. In *Proceedings of NeurIPS*.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2024b. When moe meets llms: Parameter efficient fine-tuning for multi-task medical applications. In *Proceedings of SIGIR*.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 2021a. Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of ACL*.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *Proceedings of ICLR*.

Pan Lu, Liang Qiu, Jiaqi Chen, Tony Xia, Yizhou Zhao, Wei Zhang, Zhou Yu, Xiaodan Liang, and Song-Chun Zhu. 2021b. Iconqa: A new benchmark for abstract diagram understanding and visual language reasoning. In *Proceedings of NeurIPS*.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of SIGKDD*.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of ECCV*.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Proceedings of Findings of ACL*.

Fanqing Meng, Wenqi Shao, Quanfeng Lu, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. Chartassisstant: A universal chart multimodal language model via chart-to-table pre-training and multitask instruction tunin. In *Proceedings of Findings of ACL*.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. ELLE: Efficient lifelong pre-training for emerging data. In *Proceedings of Findings of ACL*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of ICML*.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. In *Proceedings of EMNLP*.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram understanding in geometry questions. In *Proceedings of AAAI*.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving Geometry Problems: Combining Text and Diagram Interpretation. In *Proceedings of EMNLP*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of ICLR*.

Yikang Shen, Zheyu Zhang, Tianyou Cao, Shawn Tan, Zhenfang Chen, and Chuang Gan. 2023. ModuleFormer: Modularity Emerges from Mixture-of-Experts. *arXiv:2306.04640*.

Wenhao Shi, Zhiqiang Hu, Yi Bin, Junhua Liu, Yang Yang, See-Kiong Ng, Lidong Bing, and Roy Ka-Wei Lee. 2024. Math-llava: Bootstrapping mathematical reasoning for multimodal large language models. In *Proceedings of Findings of EMNLP*.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. 2023. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of CVPR*.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. 2024a. Measuring multimodal mathematical reasoning with math-vision dataset. In *Proceedings of NeurIPS*.

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. In *Proceedings of Findings of EMNLP*.

Zhihao Wang, Shiyu Liu, Jianheng Huang, Zheng Wang, Yixuan Liao, Xiaoxin Chen, Junfeng Yao, and Jinsong Su. 2024b. A Learning Rate Path Switching Training Paradigm for Version Updates of Large Language Models. In *Proceedings of EMNLP*.

Zhiyang Xu, Ying Shen, Lifu Huang, Zhiyang Xu, Ying Shen, and Lifu Huang. 2023. Multiinstruct: Improving multi-modal zero-shot learning via instruction tuning. In *Proceedings of ACL*.

Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. 2024. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *Proceedings of CVPR*.

Tongtian Yue, Longteng Guo, Jie Cheng, Xuange Gao, and Jing Liu. 2025. Ada-K Routing: Boosting the Efficiency of MoE-based LLMs. In *Proceedings of ICLR*.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. 2024. AdaMoE: Token-Adaptive Routing with Null Experts for Mixture-of-Experts Language Models. In *Proceedings of Findings of EMNLP*.

Yuexiang Zhai, Shengbang Tong, Xiao Li, Mu Cai, Qing Qu, Yong Jae Lee, and Yi Ma. 2023. Investigating the catastrophic forgetting in multimodal large language models. *arXiv:2309.10313*.

Biao Zhang, Rico Sennrich, Biao Zhang, and Rico Sennrich. 2019. Root mean square layer normalization. In *Proceedings of NeurIPS*.

Liang Zhang, Anwen Hu, Haiyang Xu, Ming Yan, Yichen Xu, Qin Jin, Ji Zhang, and Fei Huang. 2024. Tinychart: Efficient chart understanding with visual token merging and program-of-thoughts learning. In *Proceedings of EMNLP*.

Xi Zhang, Feifei Zhang, and Changsheng Xu. 2023. Vqacl: A novel visual question answering continual learning setting. In *Proceedings of CVPR*.

Zangwei Zheng, Mingyuan Ma, Kai Wang, Ziheng Qin, Xiangyu Yue, and Yang You. 2023. Preventing zero-shot transfer degradation in continual learning of vision-language models. In *Proceedings of CVPR*.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2024. Minigpt-4: Enhancing vision-language understanding with advanced large language models. In *Proceedings of ICLR*.

A Benchmark

As shown in Table 1, our benchmark covers 3 domains, 5 tasks, and 14 datasets. In Figure 3, we illustrate intuitive examples of our benchmark. In this section, we will provide a detailed description for each dataset in our benchmark, along with the relevant evaluation metrics.

Medicine. Following Li et al. (2024), we primarily focus on visual question answering task (VQA) in the biomedical domain, while considering four commonly used datasets in this domain:

- **VQA-RAD** (Lau et al., 2018) contains 3,515 QA pairs generated by clinicians and 315 radiology images that are evenly distributed over the head, chest, and abdomen. Here, questions are classified into 11 categories: abnormality, attribute, modality, organ system, color, counting, object/condition presence, size, plane, positional reasoning, and other. Half of the answers are closed-ended (i.e., yes/no type), while the rest are open-ended with either one-word or short phrase answers.

- **PathVQA** (He et al., 2020) is a dataset comprising pathology images. It contains a total of 4,998 pathology images with 32,799 QA pairs. Every image has several questions that relate to multiple aspects such as location, appearance, shape, color, etc. The questions are categorized into two types: closed-ended and open-ended questions.

- **SLAKE** (Liu et al., 2021) includes of 642 radiology images and over 7,000 diverse QA pairs annotated by experienced physicians. Here, the questions may involve external medical knowledge, and the images are associated with rich visual annotations, including semantic segmentation masks and object detection bounding boxes. Besides, SLAKE includes richer modalities and covers more human body parts than the currently available dataset, including brain, neck, chest, abdomen, and pelvic cavity.

- **LLaVA-Med** (Li et al., 2024) is a biomedical dataset developed using language-only GPT-4. With the caption and additional context of the medical image, the authors use carefully designed prompts to ask GPT-4 to generate multi-round conversations about the image. It includes about 60k instructions from the five common imaging modalities: CXR (chest X-ray), CT (computed tomography), MRI (magnetic resonance imaging), histopathology, and gross (i.e., macroscopic) pathology.

In the medical domain, we follow Li et al. (2024) to report the accuracy (**ACC.**) for the closed-set questions. Meanwhile, for open-set questions, we use **Recall** to evaluate the ratio that ground-truth tokens appear in the generated sequences.

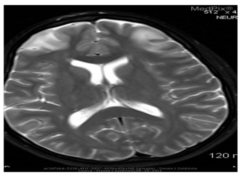
Chart. Charts play a crucial role in presenting and elucidating complex data relationships. To effectively address problems in this domain, several domain-specific MLLMs have recently been proposed, such as ChartAssistant (Meng et al., 2024) and TinyChart (Zhang et al., 2024). In the domain, we consider three types of tasks:

- **VQA** requires the model to provide a concise answer for each question based on the chart content. Here, we primarily focuses on the ChartQA (Masry et al., 2022) dataset, which includes a lot of questions requiring numerical calculation. This dataset comprises 9,600 human-written questions and 23,100 questions generated from human-written chart summaries. Meanwhile, we follow most prior studies (Masry et al., 2022; Han et al., 2023b; Zhang et al., 2024) to adopt the relaxed accuracy (**Relax_Acc.**) as the evaluation metric on this dataset, which allows numerical error within 5%.

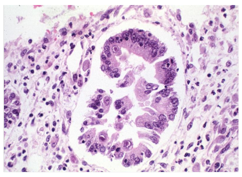
- **Chart-to-Table** aims to extract the underlying data table represented by a chart image. Given that the ChartQA dataset provides corresponding data table annotation for each chart image, we also employ this dataset for the Chart-to-Table task. In addition, we use the commonly used **RMS**$_{F_1}$ (Liu et al., 2023a) metric to evaluate the model's performance on this task.

- **Chart-to-Text** focuses on generating textual summaries based on chart content. In this task, we utilize the Statista and Pew datasets developed by Kantharaj et al. (2022). These two datasets consist of 44,096 charts covering a broad range of topics and a variety of chart types. Following Zhang et al. (2024), we evaluate the model solely on the Pew dataset and report **BLEU4** as the metric.

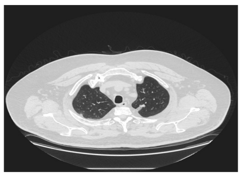
Medicin



Q: where are the lesions?
A: bilateral frontal lobes.
Task: VQA
Dataset: VQA-RAD

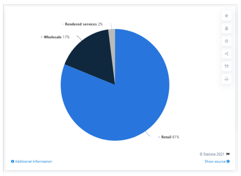


Q: is acute lymphocytic leukemia present?
A: no.
Task: VQA
Dataset: PathVQA

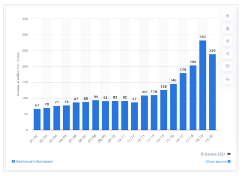


Q: What is the main organ in the image?
A: Lung, Spinal Cord.
Task: VQA
Dataset: SLAKE

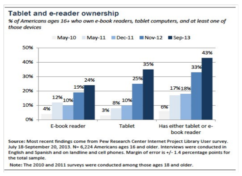
Chart



Q: By how much percentage wholesale is less than retail?
A: 64.
Task: VQA
Dataset: ChartQA

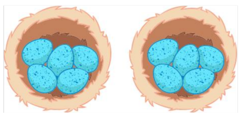


Q: Generate underlying data table of the chart?
A: Characteristic | Revenue in million U.S. dollars\n19/20 | 239\n18/19 | 283\n17/18 | 204\n16/17 | 179\n15/16 | 146\n14/15 | 126\n13/14 | 110\n12/13 | 109\n11/12 | 87\n10/11 | 92\n09/10 | 92\n08/09 | 91\n07/08 | 94\n06/07 | 88\n05/06 | 87\n04/05 | 78\n03/04 | 77\n02/03 | 70\n01/02 | 67
Task: Chart-to-Table
Dataset: ChartQA

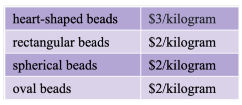


Q: Write a summary for this chart.?
A: The number of Americans ages 16 and older who own tablet computers has grown to 35%, and the share who have e-reading devices like Kindles and Nooks has grown to 24%. Overall, the number of people who have a tablet or an e-book reader among those 16 and older now stands at 43%.
Task: Chart-to-Text
Dataset: Pew

Math

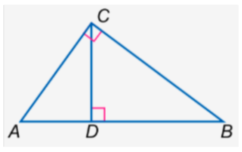


Q: How many eggs are there?
A: 10.
Task: MWP
Dataset: IconQA



heart-shaped beads	$3/kilogram
rectangular beads	$2/kilogram
spherical beads	$2/kilogram
oval beads	$2/kilogram

Q: Rebecca bought 2.5 kilograms of oval beads. How much did she spend? (Unit: $)
A: 5.
Task: MWP
Dataset: TABMWP



Q: In triangle ABC, AD = 3 and BD= 14. Find CD. \nChoices:\n(A) 6.0\n(B) 6.5\n(C) 7.0\n(D) 8.5
A: B.
Task: GPS
Dataset: Geometry3K

Figure 3: Examples of our benchmark for domain-incremental CIT in MLLMs.

Math. Recently, many studies (Wang et al., 2024a; Shi et al., 2024) has focused on improving the mathematical reasoning capabilities of MLLMs. To prevent overlap with the default instruction tuning datasets of widely used MLLMs, we collected seven multimodal mathematical reasoning datasets. Meanwhile, we categorized these datasets into two common task types, math word problem (MWP) and geometry problem solving (GPS):

- **MWP** aims to solve geometrical problems with diagrams and figures. In this task, we considered three datasets: (1) IconQA (Lu et al., 2021b) focuses on answering a question in an icon image context. It consists of 107,439 questions and three sub-tasks: *multi-image-choice*, *multi-text-choice*, and *filling-in-the-blank*. (2) CLEVR-Math (Lindström et al., 2022) consists of simple math word problems involving addition/subtraction, represented partly by a textual description and partly by an image illustrating the scenario. Solving these problems requires a combination of language, visual and mathematical reasoning. (3) TabMWP (Lu et al., 2023) contains 38,431 open-domain grade-level problems that require mathematical reasoning on both textual and tabular data. Each question in TABMWP is aligned with a tabular context, which is presented as an image, semi-structured text, and a structured table.

- **GPS** involves arithmetic calculations within the context of images. Here, the following four datasets are considered: (1) GEOS (Seo et al., 2015) is a dataset of SAT plane geometry questions where every question has a textual description in English accompanied by a diagram and multiple choices. Questions and answers are compiled from previous official SAT exams and practice exams offered by the College Board. In addition, we use a portion of the publicly available high-

β	Medicine			Chart			Math
	VQA-RAD	PathVQA	SLAKE	ChartQA	Chart-to-Table	Chart-to-Text	MWP&GPS
0.00	64.50/30.18	72.97/22.14	64.73/65.80	40.17	42.83	6.96	32.07
0.05	73.30/36.00	90.62/34.79	83.11/83.73	55.03	57.15	9.83	38.35
0.10	74.10/36.95	91.12/35.33	84.06/84.61	55.58	57.94	10.31	39.10
0.15	**74.60/37.20**	**91.52/35.97**	**84.45/85.19**	**55.71**	**58.37**	**10.42**	**39.75**
0.20	74.20/37.01	91.16/35.63	84.11/84.89	55.35	58.07	10.16	39.39
0.25	73.80/36.55	90.33/34.97	83.57/84.15	55.22	57.06	9.87	38.84

Table 5: The results of our method with varying values of β on our benchmark. η maintains its default value of 0.1.

η	Medicine		
	VQA-RAD	PathVQA	SLAKE
0.00	70.30/34.13	87.49/32.55	80.90/81.77
0.05	73.10/36.05	90.36/34.10	83.30/83.38
0.10	**74.10/36.95**	**91.12/35.33**	**84.06/84.61**
0.15	73.80/36.73	90.80/34.75	83.64/84.23
0.20	73.50/36.75	90.62/34.82	83.39/84.15
0.25	73.00/36.28	90.46/34.25	83.37/84.08

Table 6: The results of our method with different values of η in the Medicine domain. Here, β retains its default value of 0.1.

N_e	Medicine			Top-K
	VQA-RAD	PathVQA	SLAKE	
2	70.30/32.46	88.56/31.30	81.78/79.65	0.87
4	74.10/36.95	91.12/35.33	84.06/84.61	0.81
6	74.10/37.01	91.40/35.31	84.36/84.95	0.85
8	74.50/37.28	91.77/35.56	84.85/85.13	0.80

Table 7: The results of our method with different values of N_e in the Medicine domain. d' retains its default value of 128. "Top-K" means the average number of experts activated by our AT-Router to process each token.

school plane geometry questions (Seo et al., 2014) as our training set. (2) Geometry3K (Lu et al., 2021a) consisting of 3,002 geometry problems with dense annotation in formal language. It also provides a geometry solving approach with formal language and symbolic reasoning, called *Interpretable Geometry Problem Solver* (Inter-GPS). (3) GeoQA+ (Cao et al., 2022) is obtained by extending the training set of the original GeoQA dataset with 2,518 newly annotated geometric questions. The original GeoQA dataset contains 5,010 geometric problems, 3,509 for training, 746 for validation and 755 for test. (4) UniGeo (Chen et al., 2022a) involves 4,998 calculation problems with corresponding annotated program sequence, which illustrates the calculating process of the given problems and is considered as training and testing target. Besides, the GeoQA also provides a proving dataset with 9,543 problems.

In this domain, we follow Shi et al. (2024) to focus primarily on the closed-set and computational problems, while employing accuracy (**ACC.**) as the evaluation metric.

B Analysis of Hyper-parameter

Our training objective consists of three loss terms: $\mathcal{L}$=$\mathcal{L}_1$+$\beta\mathcal{L}_2$+$\eta\mathcal{L}_3$, where hyper-parameters β and η are used to balance their contributions. In this section, we will examine the impact of these two hyper-parameters on the performance of our method.

Since the loss term $\mathcal{L}_3$ aims to balance the expert activation in the SMoE module, it mainly affects our method's capacity to learn domain knowledge in the single-domain learning scenario. Therefore, we primarily examine the impact of its coefficient η on the performance of our method in the Medicine domain. As shown in Table 6, our method exhibits robust performance across various values of η and achieves best performance when η is set to 0.1.

Instead, our DSAL ($\mathcal{L}_2$) is mainly utilized to identify the most suitable SMoE module for processing each test instruction during inference, directly influencing the performance of our method in the continual learning scenario. Hence, we primarily investigate the impact of β on the performance of our method in continual learning scenarios. From Table 5, we observe that our method is insensitive to β and achieves optimal performance when β is set to 0.15. Notably, our method achieves superior results when β is set to 0.15 and η to 0.1, surpassing the results presented in the Experiment section (See Lines 5 and 6 of Table 5)

C Analysis of Our SMoE Module

In our method, we apply separate SMoE modules to learn domain-specific knowledge for each domain. This suggests that the configuration of our

Model	Medicine			Chart			Math
	VQA-RAD	PathVQA	SLAKE	ChartQA	Chart-to-Table	Chart-to-Text	MWP&GPS
LLaVA-1.5	2.10/13.27	4.62/8.87	7.33/28.92	9.27	9.97	4.20	8.93
Ours	**74.10/36.95**	**91.12/35.33**	**84.06/84.61**	**55.58**	**57.94**	**10.31**	**39.10**
Ours$^+$	73.60/36.70	90.52/34.97	83.45/84.19	55.79	57.82	10.50	39.42
Ours (Chart+Math)$^+$	2.10/12.55	4.70/8.36	7.18/28.29	–	–	–	–
Ours (Med+Math)$^+$	–	–	–	–	–	–	16.70
Ours (Med+Chart)$^+$	–	–	–	13.91	11.16	5.07	–

Table 8: The results of our method with different inference strategies and training domains. $+$ indicates that our method uses a new inference strategy for domain generalization. The domain enclosed in parentheses refers to the training domain. By default, our method utilizes all domains in our benchmark for model training.

d'	Medicine				
	VQA-RAD	PathVQA	SLAKE	Top-k	Norm
32	67.60/26.25	83.37/23.41	74.33/70.24	0.42	0.31
64	70.10/30.60	88.03/30.18	81.30/77.65	0.64	0.37
128	74.10/36.95	91.12/35.33	84.06/84.61	0.81	0.50
256	74.70/37.31	91.94/35.73	84.72/85.20	1.35	0.58
512	75.00/37.93	92.16/36.38	85.10/86.01	2.15	0.66

Table 9: The results of our method with different values of d' in the Medicine domain. Here, N_e retains its default value of 4. "Top-K" means the average number of experts assigned to process each instruction token by our AT-router. "Norm" represents the average L2-norm of the output features from the experts.

SMoE module mainly affects the performance of our method in the single-domain learning scenario, rather than in the continual learning scenario. Thus, in this section, we aim to analyze the effects of the expert number of N_e and the expert dimension d' within the SMoE module on our model's performance in the Medicine domain. We present the experimental results in Table 7 and Table 9. Further analysis of these results reveals several significant conclusions:

Firstly, increasing the number of experts N_e and the expert dimension d' can indeed enhance the performance of our method. The primary reason is that increasing N_e and d' directly leads to a growth in the trainable parameters involved in our method. Nevertheless, when N_e exceeds 4 and d' surpasses 128, the performance gains from further increasing the trainable parameters become limited. Thus, we adopt $N = 4$ and $d' = 128$ as the default settings in the Experiment section. Certainly, when there is enough training data, increasing both the number and dimension of experts is still an effective strategy for improving the performance of our method.

Secondly, we observe that the average number of experts assigned to process each instruction token by our AT-router increases with the expert dimension d' (see the "Top-k" column in Table 9). In contrast, this phenomenon does not seem to occur when increasing the number of experts N_e (see the "Top-k" column in Table 7). To explore the underlying reasons, we calculated the average L2-norm of the output features from experts with different dimensions d'. To facilitate better comparison, we divide the L2-norms obtained from the experts by the L2-norm produced from the original FFN sublayer in the LLM. As shown in the "Norm" column of Table 9, the L2-norms from the experts is positively correlated with their dimension d'. Thus, we conjecture that the expert dimension d' influences the average number of experts assigned to process each token via the L2-norm of the output features.

To enhance the practicality of our AT-Router, we designed a new loss function $\mathcal{L}_{\text{top}_\text{k}}$ to control the average number of experts assigned to process each instruction token. Specifically, we first compute the average score for each expert and the average adaptive threshold based on Eq. 5: $[\bar{\alpha}, \bar{s}_1, \cdots, \bar{s}_N] = \frac{1}{|\mathcal{B}|}\sum_{b=1}^{|\mathcal{B}|} \boldsymbol{S}_b$, where $|\mathcal{B}|$ denotes the number of instruction tokens in the current training batch $\mathcal{B}$. Next, we combine the average scores of all experts: $\bar{s} = \frac{1}{N_e}\sum_{n=1}^{N_e} \bar{s}_n$. Then, we calculate the average number of experts k assigned to process each token in the current training batch based on Eq. 6. Finally, $\mathcal{L}_{\text{top}_\text{k}}$ is defined as a binary cross-entropy:

$$\mathcal{L}_{\text{top}_\text{k}} = -\left(\mathbf{I}(k<K)\log(\widehat{p}) + \mathbf{I}(k>K)\log(1-\widehat{p})\right),$$
$$\widehat{p} = e^{\bar{s}}/(e^{\bar{s}}+e^{\bar{\alpha}}),$$

where $\mathbf{I}(\cdot)$ represents the indicator function, K is a hyper-parameter that denotes the desired average number of experts assigned to handle each token. In the equation, we omit the indexes of the SMoE module and the layers in LLM. Finally, we sum the loss $\mathcal{L}_{\text{top}_\text{k}}$ of the SMoE modules in training to derive our fourth loss term $\mathcal{L}_4$.

K	Medicine		
	VQA-RAD	PathVQA	SLAKE
1	74.20/36.72	91.36/35.18	84.20/84.48
2	74.60/37.10	91.98/35.92	84.54/85.04
3	74.90/37.95	92.13/36.30	84.88/85.55
4	74.97/38.01	92.20/36.47	85.02/85.71

Table 10: The results of our method with different values of K in the Medicine domain. Here, N_e and d' retain their default values of 4 and 128, respectively. Meanwhile, we employ the objective function $\mathcal{L}{=}\mathcal{L}_1{+}0.1{*}(\mathcal{L}_2{+}\mathcal{L}_3{+}\mathcal{L}_4)$ for model training.

As shown in Table 10, we also explore the impact of the hyper-parameter K on the performance of our method. We find that increasing the average activations of experts can enhance the performance of our method. Nevertheless, when K is relatively large, the performance enhancement from further increasing K is very limited. The primary reason is that allocating more experts to unimportant tokens does not yield obvious performance gains. In contrast, a larger K will result in lower model inference efficiency.

D Analysis of Domain Generalization

In this section, we evaluate the generalization capability of our method to unseen relevant domains. However, our method only selects one domain-specific SMoE module in every layer of LLM to process each test instruction, which may limit the inter-domain knowledge transfer and the domain generalization of our method. To address this challenge, we further optimize the inference strategy utilized in our method. Specifically, we first count and record the average DSAL of each SMoE module on the training instructions in the corresponding domain. Then, during inference, we select the SMoE module with the minimum DSAL as the primary module for processing the current test instruction (**See Eq. 4**). Meanwhile, we select the SMoE module, whose average DSAL exceeds its DSAL on the current test instruction, as auxiliary modules to provide supplementary knowledge in related domains. In this way, our method can be adapted to open-domain scenarios without requiring model retraining. From the experimental results presented in Table 8, we derive several conclusions:

First, we notice that directly applying the above inference strategy to our method improves its performance improvement in the Math and Chart domains, while causing a slight drop in the Medicine domain (see the line for Ours$^+$ in Table 8). The main reason for this is the inherent correlation between the Chart and Math domains, which allows them to mutually enhance each other. However, the significant gap between the Medicine domain and the other two domains prevents our method from leveraging knowledge from the other two domains to improve its performance in the Medicine domain. Moreover, we also observe that LLaVA-1.5 has a slight performance decline in the Medicine domain after learning the Chart and Math domains via our method (see LLaVA-1.5 vs. Ours (Chart+Math)$^+$). This further validates the gap between the Medicine domain and these two domains.

Second, we find that our method shows superior domain generalization in the Chart and Math domains (see LLaVA-1.5 vs. Ours (Med+Math)$^+$ and LLaVA-1.5 vs. Ours (Med+Chart)$^+$). This clearly confirms that our method can indeed achieve domain generalization by transferring knowledge from *relevant* (already learned) domains. In addition, we attribute the excellent domain generalization of our method to the following two aspects: **(1)** Our method can effectively fuse knowledge from multiple domains to process test instructions in new domains by selecting SMoE modules corresponding to different domains at various layers of the LLM. **(2)** The above inference strategy enhances our method's capacity to integrate knowledge from various learned domains, thereby further improving its domain generalization.

E Implementation Details of Baselines

In the paper, our method and all baselines are developed on LLaVA-1.5 using the following parameter-efficient fine-tuning techniques:

- **LoRA** (Hu et al., 2022): Replay, EWC, LWF, M-LoRA, O-LoRA.
- **DMoE** (Ma et al., 2018): MoELoRA.
- **SMoE** (Shazeer et al., 2017): SMoELoRA, L-SMoE, MoLM, **Ours**.

To ensure a fair comparison, all methods add a comparable average number of learnable parameters to the FFN sub-layers in the LLM for learning each domain (see **Line 2** in Table 3). Unlike architecture-based methods that dynamically and sequentially add the same amount of parameters for each domain to the LLM, both replay-based and regularization-based methods add the learnable parameters for all domains to the LLM at once. By doing so, we further guarantee that the total number of parameters for all methods remains comparable. In the multi-domain learning, where all domains can be accessed simultaneously, all methods add the learnable parameters for all domains to the LLM at once.

Furthermore, in methods based on DMoE and SMoE, experts are often implemented in two ways:

- **FFN-based Expert**: It directly utilizes the FFN sublayer of the LLM as the expert. L-SMoE, MoLM, and our method employ this type of expert. Meanwhile, we leverage the dimensionality d' of the intermediate hidden layer in the FFN sublayer to control the number of parameters for each expert (**See Eq. 7**).

- **LoRA-based Expert**: It applies LoRA to implement the FFN sub-layer of the LLM as the expert, and then uses the rank r in LoRA to control the parameter count for each expert. Here, both MoELoRA and SMoELoRA adopt this kind of expert.

For a fair comparison, we utilize d' and r to ensure that the size and number of experts in our method are consistent with those in the related baselines.

F Analysis on Task-incremental CIT

Here, we aim to validate the efficacy of our method in task-incremental CIT. Therefore, we further conduct experiments on the recent task-incremental CIT benchmark, CoIN (Chen et al., 2024a). As shown in Table 11, our method consistently outperforms all relevant baselines in the task-incremental CIT. This further confirms that our method can effectively prevent catastrophic forgetting during continual learning. The underlying reason is that the output formats of some tasks in CoIN are significantly different, causing baselines to forget the response formats of earlier tasks after learning new ones. In contrast, our method employs independent SMoE modules to learn different tasks and leverages DSAL to identify the corresponding task for each test instruction, effectively avoiding this issue.

By analyzing the results in **the first line** of various methods, we find that our approach demonstrates stronger task learning capabilities, achieving outstanding performance on each task. Moreover, comparing the results in the two rows for our method indicates a slight forgetting issue in task-incremental CIT, which was not observed in domain-incremental CIT. The possible reason is that some tasks in CoIN contain similar instruction instances, which restricts the ability of our DSAL to distinguish them.

G Analysis of AT-Router

Similar to our AT-Router, several recent studies also focus on reducing expert activation in SMoE. In this section, we first provide a detailed discussion of the differences between their method and our AT-Router:

- **AdaMoE** (Zeng et al., 2024): It first adds many *null experts* to the SMoE module, and then reduces the activation of true experts by assigning some null experts to tokens. However, AdaMoE only employs the load balancing loss, without the objective loss of the target task, to train the routing scores of null experts, which restricts its performance on the target task.

- **AdaK** (Yue et al., 2025): It introduces an extra *allocator* into the already trained SMoE module for predicting the number k of experts to be activated for each token. However, the allocator requires training using complex reinforcement learning (RL) strategies, restricting the practicality and generalization of this method. Meanwhile, AdaMoE's performance depends on the quality of its base SMoE model.

- **AT-Router**: Unlike the above two methods, our AT-Router calculates an adaptive threshold for each token to assess its importance, and then assigns experts based on this importance. **Notably**, the adaptive threshold is used as the coefficient for the output feature of the original FFN in the LLM, providing several benefits to our AT-Router: (1) We can directly use the objective loss of the target task to optimize the adaptive threshold, thus reducing expert activations while improving model performance. (2) With the help of adaptive thresholds, AT-Router can determine how much additional experts are needed to further process the current token based on how well the original FFN processes it.

Furthermore, to validate the superiority of our AT-Router, we further conduct comparative experiments on our benchmark. Here, considering that

Model	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	Grounding	VQAV2	OCR-VQA
	Accuracy on Each Task							
LoRA	75.33	47.06	94.95	52.95	50.77	10.25	56.73	55.33
	49.96	23.60	7.22	36.12	33.05	0.09	39.20	55.33
LwF	75.33	48.18	96.90	48.58	44.12	6.60	38.58	62.35
	63.14	39.60	8.90	34.83	14.53	2.48	40.67	62.35
EWC	75.28	48.37	96.83	42.77	44.25	8.65	60.27	61.02
	67.14	40.41	8.18	35.05	37.88	2.67	41.27	61.02
MoELoRA	75.85	49.05	93.95	56.53	48.70	25.57	61.90	55.35
	58.92	38.59	8.85	37.10	44.25	2.45	41.40	55.35
SMoELoRA	74.67	47.15	94.36	54.38	47.51	23.46	60.30	55.26
	60.53	39.84	9.73	38.35	45.02	3.61	41.88	55.26
Ours	**76.36**	**49.64**	**96.96**	**57.05**	**50.83**	**25.79**	**62.45**	**62.76**
	75.04	**48.51**	**95.12**	**56.10**	**50.68**	**24.19**	**61.71**	**62.76**

Table 11: The results of our method and baselines on CoIN. **The first line** of each method presents the results for each task evaluated when just tuned on the corresponding task, and **the second line** displays the final results of each task after fine-tuning on the last task.

Model	Medicine			Chart			Math
	VQA-RAD	PathVQA	SLAKE	ChartQA	Chart-to-Table	Chart-to-Text	MWP&GPS
LLaVA-1.5	2.10/13.27	4.62/8.87	7.33/28.92	9.27	9.97	4.20	8.93
DMoE	72.40/36.48	90.24/34.82	82.77/83.38	55.18	57.32	9.94	38.62
SMoE	65.20/32.66	87.59/30.43	80.09/80.05	50.72	54.51	7.37	36.08
AdaMoE	66.50/33.45	88.42/31.50	81.22/81.14	51.60	55.77	8.29	36.68
AdaK (+DMoE)	70.00/35.04	88.86/32.37	81.85/.82.11	52.84	56.19	8.72	36.95
Ours	**74.10/36.95**	**91.12/35.33**	**84.06/84.61**	**55.58**	**57.94**	**10.31**	**39.10**
Ours (AT-Router$\rightarrow$AdaMoE)	67.20/34.82	89.16/32.44	81.93/81.75	52.65	56.04	8.96	37.23
Ours (AT-Router$\rightarrow$AdaK)	72.00/35.98	89.82/33.55	82.65/82.81	53.47	56.89	9.46	37.76

Table 12: The results of our method and baselines on our benchmark in **single-domain learning** scenario. Here, AdaK is implemented on the basis of DMoE. Meanwhile, we align the expert activation levels of AdaMoE and AdaK with those of our AT-Router, ensuring a fair comparison.

different routing strategies only affect our method's ability to learn domain knowledge, we compared our AT-Router with AdaMoE and AdaK in the single-domain learning scenario. From the experimental results presented in Table 12, we can draw several observations:

First, AdaMoE and AdaK consistently outperform traditional SMoE, thus confirming their effectiveness. Additionally, although AdaK reduces the expert activation of DMoE, it also leads to performance degradation across all domains. It implies that the primary objective of AdaK is to reduce the expert activation of its base model rather than to improve its performance.

Second, our method significantly outperforms both AdaMoE and AdaK across all domains, further validating the strong domain learning capability of our method. To conduct a fine-grained comparison, we replace the AT-Router in our method with AdaMoE and AdaK, respectively (see **the bottom rows** of Table 12). We note that both replacements lead to a performance drop in our method, effectively demonstrating the superiority of our AT-Router and its compatibility with other components of our method.