

Divide-and-Adapt: Fine-Tuning VLMs via Model Division and Routing

Hailing Wang^{*1}, Neil Rodrigues², Xianglong Wang², Wei Li², Lin Chen², Yun Fu¹

¹Northeastern University, ²Wyze Labs, Inc.

Correspondence: wang.haili@northeastern.edu

Abstract

Fine-tuning Vision-Language Models (VLMs) is a common practice to adapt pre-trained models for downstream tasks. Yet existing fine-tuning methods consistently keep the pre-trained architecture intact, leaving open the question of whether maintaining such structural integrity is truly necessary for effective adaptation. To investigate this, we propose Divide-and-Adapt, a new fine-tuning paradigm that departs from preserving the full pre-trained structure. It first divides the pre-trained model into multiple downstream-oriented subnetworks, and then adapts them collaboratively through a routing-based coordination mechanism. This design facilitates the selection of structures that are most beneficial for downstream tasks and enables their collaborative integration. Remarkably, even without any task-specific training, our Divide-and-Adapt framework first divides the pre-trained model into downstream-oriented subnetworks and adaptively activates them for downstream inference, already outperforming the pre-trained model on downstream tasks. With further fine-tuning, Divide-and-Adapt surpasses both full fine-tuning and parameter-efficient baselines across multiple benchmarks, demonstrating its effectiveness. Our code will be made publicly available at <https://github.com/hailing/DivideAndAdapt>.

1 Introduction

Vision Language Models (VLMs) (Chen et al., 2024; Lin et al., 2024; Liu et al., 2024a; Lu et al., 2026a) achieve strong generalization across diverse visual tasks (Wang et al., 2020; Li et al., 2019; Fei et al., 2024) due to large-scale multimodal pretraining. However, in practical applications, VLMs still require further adaptation, as downstream tasks often exhibit domain shifts. This creates a practical dilemma: how to leverage the strong pretrained

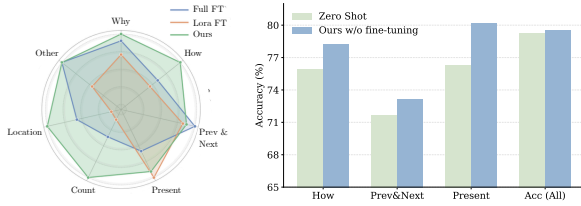
knowledge of VLMs while achieving effective task specialization without incurring prohibitive fine-tuning costs. A straightforward strategy to adapt VLMs is to fully fine-tune all model parameters. However, as model scale grows, this approach becomes impractical due to high computational and memory costs.

To improve efficiency, various parameter-efficient fine-tuning (PEFT) methods (Xu et al., 2023; Han et al., 2024; Wang et al., 2025a) have been developed, which can be broadly categorized into three types. The first category, prompt-based tuning (Cui et al., 2022; Zhu et al., 2023), introduces learnable prompts to steer pretrained models toward downstream tasks while keeping backbone weights frozen. A second category inserts lightweight adapter (He et al., 2021; Chen et al., 2023) modules into transformer layers, enabling task-specific learning without modifying the original architecture. The third category, represented by Low-Rank Adaptation (LoRA) (Hu et al., 2022), applies low-rank updates to selected weight matrices, providing strong adaptation capability with minimal additional parameters. Despite their efficiency, these PEFT methods often fall short of full fine-tuning on complex downstream tasks (Sun et al., 2022; Sundaram et al., 2019).

All these methods, including full fine-tuning and PEFT, operate under the paradigm that the pre-trained model should be kept intact during downstream fine-tuning. However, it remains unclear whether maintaining the full structure is necessary for effective task transfer. *In other words, rather than treating the pretrained model as an inseparable whole, can we divide the pretrained model to locate task-relevant subnetworks and adapt only these subnetworks for downstream tasks?*

Model pruning offers a practical way to divide a pretrained model into subnetworks by selecting a subset of parameters while preserving their functional structure. Prior work (Sreenivas et al., 2024;

^{*}This work was conducted during an internship at Wyze Labs, Inc.



(a) Comparison among Full Fine-tuning, LoRA Fine-tuning and Ours. Our method achieves the best overall result, especially in Location and Count categories. (b) Performance comparison between the zero-shot baseline and our method without fine-tuning. The results show that subnetworks obtained via diversity pruning already outperform the pretrained model across most categories.

Figure 1: Performance comparison across fine-tuning strategies and zero-shot settings.

Jeddi et al., 2025) has shown that a carefully selected sparse subnetwork can retain most of the original model’s performance, indicating that substantial capability can be concentrated within only part of the network. While pruning is motivated by model compression, it offers an important conceptual insight for downstream adaptation: pretrained models already contain subnetworks with strong representational capacity, and uncovering such subnetworks may enable more targeted and efficient fine-tuning than updating the full model. However, existing pruning approaches (Jiang et al., 2025; Federici et al., 2024) typically focus on discovering a single winning subnetwork, which may limit representational diversity and capacity for complex downstream tasks. In contrast, we explore whether multiple complementary subnetworks can be extracted from pretrained VLMs and jointly adapted. To realize this idea, we draw inspiration from the Mixture-of-Experts (MoE) (Zadouri et al., 2023; Puigcerver et al., 2023) paradigm, which provides a way to coordinate multiple subnetworks. Specifically, MoE introduces a lightweight gating network that dynamically routes each input to the most relevant experts, allowing the model to selectively activate different path for different types of inputs.

Building on this idea, we introduce a Divide-and-Adapt fine-tuning paradigm. Instead of preserving the entire dense pretrained architecture, we first divide the model by extracting several task-relevant subnetworks. Each subnetwork inherits useful representations for downstream tasks, and we subsequently adapt them jointly to enhance cooperation and specialization during fine-tuning. To evaluate the effectiveness of our approach, we conduct experiments on both public benchmark and a practical task. As shown in Figure 1a, our method outper-

forms both full fine-tuning and LoRA fine-tuning, demonstrating stronger adaptation capability on downstream tasks. Moreover, even without any fine-tuning, as shown in Figure 1b we observe that directly initializing our model with multiple subnetworks extracted from the pretrained backbone already achieves better downstream performance than the original pretrained model in the zero-shot setting. This finding suggests that dividing a pretrained model into diverse subnetworks inherently provides a stronger starting point for adaptation. To summarize, our main contributions are:

- We propose a novel fine-tuning paradigm that divides a pretrained dense model into multiple task-relevant subnetworks and subsequently adapts them jointly. This design diverges from traditional fine-tuning by explicitly modeling subnetwork cooperation and specialization.
- We introduce a strategy to extract diverse subnetworks from the pre-trained model, promoting complementary representations for cooperative and specialized fine-tuning.
- Extensive experiments on both public benchmark and a practical task demonstrate that DA-MoE achieves superior performance compared to existing fine-tuning methods on downstream tasks.

2 Related Works

2.1 PEFT For VLM

Vision Language Models (VLMs) (Chen et al., 2024; Bai et al., 2023; Lu et al., 2026b) have shown strong performance across multimodal tasks such as image captioning (Yang et al., 2023; Luu et al., 2024), visual question answering (Antol et al., 2015; Hu et al., 2024), and video understanding (Hong et al., 2024; Weng et al., 2024). However, despite their strong generalization ability, effectively fine-tuning VLMs for downstream applications remains challenging. Fully fine-tuning all model parameters is a straightforward approach, but as model size continues to grow, this strategy becomes increasingly computationally expensive in both time and memory. To address this limitation, parameter-efficient fine-tuning (PEFT) techniques have been introduced. Among them, LoRA (Hu et al., 2022) is widely adopted for efficiently adapting pre-trained models through

low-rank weight updates while keeping the backbone frozen and adding negligible inference overhead. Besides LoRA, several variants have been developed to further improve parameter efficiency. VeRA (Kopiczko et al., 2023) replaces trainable low-rank matrices in LoRA with fixed random matrices and learns only small scaling vectors, further reducing trainable parameters. Tied-LoRA (Renduchintala et al., 2024) further improves parameter efficiency by sharing the same low-rank projection matrices across multiple layers, avoiding separate LoRA parameters per layer while maintaining strong fine-tuning performance. AdaLoRA (Zhang et al., 2023) dynamically adjusts the low-rank capacity by applying SVD to LoRA updates and discarding less informative singular directions, reallocating the rank budget to more important layers for improved parameter efficiency. DoRA (Liu et al., 2024b) decomposes pretrained weights into magnitude and direction, applying low-rank adaptation only to the directional component. This preserves pretrained weight scales, while enabling effective task-specific updates. While these techniques provide substantial parameter and computational savings, they generally fail to surpass, and often lag behind, the performance of fully fine-tuned models especially when tackling complex downstream tasks (He, 2024).

2.2 Model pruning

A large body of work has explored network redundancy (Wang et al., 2026, 2025b) and demonstrated that deep networks can be pruned without sacrificing performance (Gale et al., 2019; Elsen et al., 2020). The lottery ticket hypothesis (Frankle and Carbin, 2018) further revealed that over-parameterized neural networks contain sparse subnetworks capable of training to full accuracy when properly initialized, highlighting that useful subnetworks can exist within dense architectures. In large models, recent techniques have similarly investigated extracting effective subnetworks. Wanda (Sun et al., 2023) identifies subnetworks by ranking connections according to the product of their weight magnitude and activation strength, retaining those that contribute most to the model’s effective computation pathways. Connections with low joint influence are removed, yielding a sparse subnetwork that preserves the model’s forward signaling behavior while discarding minimally relevant structure. SparseGPT (Frantar and Alistarh, 2023) identifies subnetworks by preserving connections

that are most critical for maintaining the pretrained model’s layer-wise behavior. It estimates the impact of removing each connection on local reconstruction error, retaining those that best preserve the model’s functional mapping while discarding weights with minimal contribution. This process yields a sparse subnetwork that faithfully maintains pretrained computational structure while still allowing downstream specialization.

3 Methods

We compare our method with full fine-tuning and LoRA fine-tuning, as illustrated in Figure 2. In full fine-tuning, all parameters of the pre-trained model are updated while the original architecture remains unchanged. In contrast, LoRA fine-tuning freezes the pre-trained weights and trains only a small set of additional low-rank parameters while preserving the original model structure. Both approaches maintain the complete architecture of the pre-trained model during adaptation. In contrast, our method fine-tunes only a set of subnetworks rather than the entire model. These subnetworks are derived from the pre-trained backbone and jointly adapted through a routing mechanism that dynamically selects which subnetworks to activate for each input. We refer to this design as **DA-MoE**.

To clearly illustrate our approach, we introduce its two key blocks: model pruning and MoE. We first present the fine-tuning workflow, describing how it transitions from the pre-trained model, and then provide details on how multiple subnetworks are selected from the dense architecture. Finally, we describe the continued training strategy that adapts these subnetworks to downstream tasks and enables dynamic routing among them.

3.1 Preliminaries

Model Pruning. Model pruning removes redundant parameters from a network while preserving its performance. Given network weights $W = \{w_0, w_1, \dots, w_K\}$, a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, and a desired total number of non-zero weights $k < K$, model pruning seeks a sparse subnetwork that minimizes the loss:

$$\begin{aligned} \min_W \mathcal{L}(W; \mathcal{D}) &= \min_W \frac{1}{N} \sum_{i=1}^N \ell(W; (x_i, y_i)) \\ \text{s.t. } \|W\|_0 &\leq k. \end{aligned} \quad (1)$$

Beyond efficiency, pruning has been observed to identify performant subnetworks inside over-parameterized models, indicating that a large model

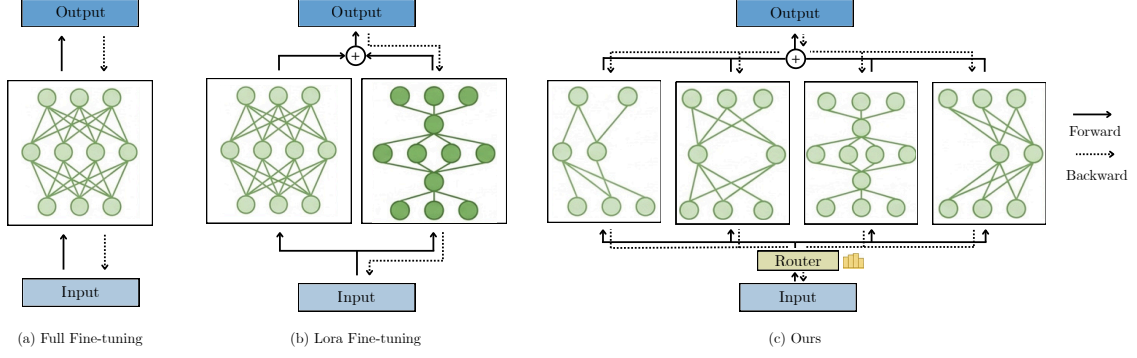


Figure 2: Comparison of different fine-tuning strategies. (a) Full Fine-tuning: all parameters are updated while keeping the original architecture unchanged. (b) LoRA Fine-tuning: The pre-trained structure is preserved; the original weights are frozen, and only a small set of additional low-rank adapters are trained. (c) Ours: instead of updating the entire model, we fine-tune a set of task-relevant subnetworks extracted from the pre-trained model. These subnetworks are jointly adapted through a routing mechanism that dynamically selects which subnetworks to activate for each input.

can implicitly contain multiple useful sparse subnetworks. In this sense, pruning can be regarded as a *divide* operation: it decomposes a large pre-trained model into multiple compact subnetworks that may each capture different structural biases. This motivates us to investigate whether pruning can extract task-relevant subnetworks for downstream task, allowing us to fine-tune a chosen subset rather than updating the entire model.

MoE. After division, we need a mechanism to adapt these subnetworks. The MoE mechanism enables input-adaptive computation by dynamically routing each input to a small subset of experts. A router determines which experts are most relevant to the given input and combines their outputs, allowing the model to handle diverse inputs efficiently while maintaining computational sparsity. Formally, a MoE layer contains a set of experts $\{E_i\}_{i=1}^n$ and a router R that selects which experts to activate for each input. Given a hidden state $\mathbf{h}$, the MoE layer computes

$$\mathbf{h}' = \sum_{i=1}^n R(\mathbf{h})_i E_i(\mathbf{h}), \quad (2)$$

where $R(\mathbf{h})_i$ denotes the routing score assigned to the i -th expert and $E_i(\mathbf{h})$ is the expert output. MoE serves as an *adapt* mechanism: given multiple subnetworks, the router selects which ones to activate for each input, allowing the model to adaptively utilize the most relevant subnetworks.

3.2 Workflow

Building on the divide-and-adapt perspective introduced above, DA-MoE realizes this concept through an architectural design that integrates

model pruning and dynamic routing. We first present the workflow of DA-MoE, which already incorporates the adaptive routing mechanism, and then elaborate on how subnetworks are selected from the pretrained model.

As shown in Figure 3, We replace pre-trained decoder blocks with DA-MoE blocks. Each DA-MoE block consists of a self-attention layer followed by a sparse MoE module, where the original FFN is replaced by a mixture-of-experts structure. The experts in this sparse MoE are derived from the pre-trained FFN through pruning.

Forward pass. Given an input hidden state h , each DA-MoE block first performs self-attention, and then the router dispatches tokens to a small subset of pruned experts. Formally, the output of a DA-MoE block is computed as:

$$h' = \text{LN}\left(h + \text{Attention}(h) + \sum_{i=1}^k R(h)_i M_i E_i(h)\right), \quad (3)$$

where $R(h)_i$ denotes the routing weight for the i -th expert computed by the router, $E_i(h)$ represents the output of the i -th pruned FFN expert, and $M_i \in \{0, 1\}^{d \times d}$ is a binary structural mask generated by pruning the pretrained FFN, enforcing a fixed sparse subnetwork architecture for expert i . Only the top- k experts are activated for each token, enabling efficient sparse computation while preserving diverse capabilities from the dense model.

3.3 Pruning-based Expert Construction

We now describe how experts are constructed within DA-MoE. This stage plays a critical role during model fine-tuning, as it determines how subnetworks are derived and diversified. The

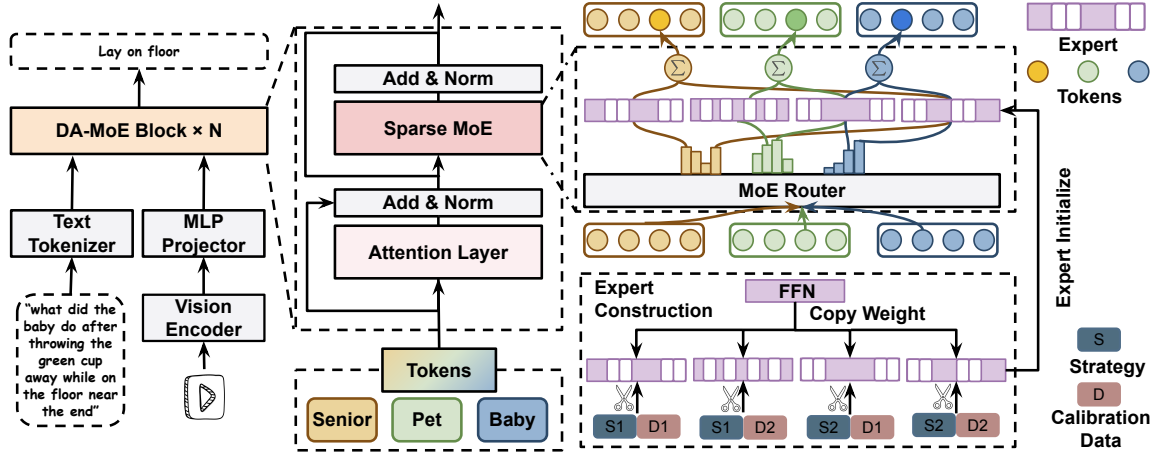


Figure 3: Overall architecture of the proposed DA-MoE framework. Pre-trained decoder blocks in the transformer are replaced with DA-MoE blocks. A DA-MoE block consists of an attention layer followed by the sparse MoE. The sparse MoE is composed of a router and a set of experts. The router dynamically assigns each token to a small subset of experts based on routing scores. These experts are constructed by pruning the pre-trained FFN into multiple sparse experts with distinct masks.

main challenge is to obtain subnetworks that are both effective for the downstream task and sufficiently diverse to promote specialization. To achieve this, we construct experts by pruning the dense FFN using different data and different pruning strategies, which enables each expert to specialize while sharing the pretrained model’s foundational knowledge. To achieve this balance in practice, we prune subnetworks using distinct pruning strategies or different calibration datasets to promote diversity.

Different pruning strategies. To further encourage diversity while ensuring that each expert remains useful for the downstream task, we adopt two pruning strategies when extracting subnetworks from the dense FFN. These two strategies follows different perspectives when selecting subnetworks: identifying weights that strongly interact with activations (Sun et al., 2023), and preserving those that are critical for maintaining the model’s layer-wise behavior (Frantar and Alistarh, 2023). To realize the first perspective, we evaluate each weight based on the product of its magnitude and the corresponding activation norm. Specifically, for a weight W_{ij} and its associated input activation X_j , the pruning score is computed as

$$S_{ij} = |W_{ij}| \cdot \|X_j\|_2, \quad (4)$$

where $|\cdot|$ denotes the absolute value operator and $\|X_j\|_2$ evaluates the ℓ_2 norm of the j -th activation vector aggregated across tokens. We then prune weights with the lowest scores to construct a sparse

subnetwork, ensuring that the retained parameters correspond to both large weights and activations.

To realize the second perspective, we follow the layer-wise reconstruction formulation and evaluate each weight by its estimated impact on the reconstruction error. Specifically, the pruning score for a weight W_{ij} is defined as

$$S_{ij} = \left[\frac{|W_{ij}|^2}{\text{diag}((X^T X + \lambda I)^{-1})} \right]_{ij}, \quad (5)$$

where X denotes the input activations of the layer, $X^T X + \lambda I$ corresponds to the damped Hessian matrix, and λ is a regularization term to stabilize the inverse. We prune weights with the smallest scores to minimize the local reconstruction error, thereby preserving the most critical parameters while inducing sparsity.

Different calibration datasets. Since the pruning strategies are input-dependent, different calibration datasets naturally leads to different subnetworks.

3.4 Continued Training

After constructing the experts, we fine-tune the model on the downstream dataset. Following standard autoregressive modeling, the DA-MoE model is optimized to generate the correct output token sequence while simultaneously ensuring balanced expert utilization during routing.

The total training loss consists of an autoregressive loss $\mathcal{L}_{\text{regressive}}$ and an auxiliary balancing loss $\mathcal{L}_{\text{aux}}$, combined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{regressive}} + \alpha \cdot \mathcal{L}_{\text{aux}}, \quad (6)$$

where α is a balancing coefficient controlling contribution of the auxiliary loss.

Autoregressive Loss. We optimize the DA-MoE model using a standard next-token prediction objective. Given input $\mathcal{X}$ and output tokens $\mathcal{Y} = [y_1, \dots, y_K]$, the loss is:

$$\mathcal{L}_{\text{regressive}} = - \sum_{i=1}^K \log p_{\theta}(y_i | \mathcal{X}, y_{<i}), \quad (7)$$

where θ denotes model parameters.

Auxiliary Load-Balancing Loss. To prevent the router from collapsing onto a small subset of experts, we incorporate a differentiable load-balancing loss following (Fedus et al., 2022). For an MoE layer with E experts, the auxiliary loss is:

$$\mathcal{L}_{\text{aux}} = E \cdot \sum_{i=1}^E \mathcal{F}_i \cdot \mathcal{G}_i, \quad (8)$$

where $\mathcal{F}_i$ is the fraction of tokens routed to expert i and $\mathcal{G}_i$ denotes its average routing probability:

$$\mathcal{F}_i = \frac{1}{K} \sum_{j=1}^K \mathbf{1}[\arg \max \mathcal{P}(x_j) = i], \quad \mathcal{G}_i = \frac{1}{K} \sum_{j=1}^K \mathcal{P}(x_j)_i. \quad (9)$$

4 Experiments

4.1 Experimental Setup

Datasets. We fine-tune DA-MoE on two video datasets: NExT-QA (Xiao et al., 2021) and a real-world smart-home dataset (See Section A). NExT-QA is a benchmark consisting of 5,440 videos and 52K question-answer pairs. It focuses on causal, temporal, and descriptive reasoning, requiring models to infer human intentions, temporal dependencies, and causal relations for video understanding.

We evaluate DA-MoE on two benchmarks aligned with the training datasets. For NExT-QA, we adopt the official evaluation metrics and report accuracy across three reasoning categories: causal (Acc_C), temporal (Acc_T), and descriptive (Acc_D), each with fine-grained question subtypes. We use the smart-home benchmark for validation, which consists of 1,143 video-question pairs, and report accuracy for each category as the evaluation metric.

Baseline. We benchmark DA-MoE against standard fine-tuning paradigms using two representative VLMs: InternVL3-8B (Zhu et al., 2025) and SmolVLM-2.2B (Marafioti et al., 2025). For each model, we compare full fine-tuning and LoRA.

Settings. In our experiments, we construct four

sparse experts as experts by pruning 50% of the weights from the pretrained FFN. For each downstream task, the experts are constructed independently using calibration samples drawn exclusively from the corresponding training split, rather than from any separate or external corpus. We randomly sample two calibration subsets, each containing 128 training examples, using fixed random seeds of 42 and 0, resulting in 256 calibration samples in total. The calibration samples follow the standard input format of the corresponding downstream task and are used only to collect activation statistics for pruning. No validation or test samples are involved in the expert construction process.

To enhance diversity, each subnetwork is generated using different downstream calibration datasets and distinct pruning strategies while maintaining the same pruning ratio. These subnetworks are then employed as four experts with top-2 routing in the DA-MoE architecture. For continued training, DA-MoE adopts LoRA-based parameter updates with a rank of 32, preserving the original architecture while efficiently adapting the model through low-rank modules. Additionally, the auxiliary balancing coefficient α is set to 0.01 to control the contribution of the auxiliary loss during optimization.

4.2 Main Results

Quantitative Results. We evaluate DA-MoE against zero-shot inference, full fine-tuning, and LoRA fine-tuning on both NExT-QA and smart-home benchmark. On the NExT-QA benchmark (Table 1), DA-MoE achieves higher overall accuracy than all baselines and outperforms them in the majority of reasoning categories. Our method achieves an overall accuracy of 81.4%, surpassing LoRA (80.2%) and full fine-tuning (80.9%).

On the smart-home benchmark (Table 2), DA-MoE achieves higher overall accuracy and improves performance in most categories compared to baseline strategies. For InternVL3-8B, although DA-MoE does not surpass all baselines in every category, it achieves the best overall performance, reaching a weighted accuracy of 73.35% and outperforming the strongest baseline (71.87%) by a relative gain of 1.48 percentage points. For SmolVLM2, DA-MoE outperforms full fine-tuning in every category, highlighting its effectiveness on smaller-scale models.

Qualitative Results. Figure 4 provides a vi-

Table 1: Performance comparison of different fine-tuning strategies on InternVL3-8B across various question categories in the NExT-QA.

Methods	Acc _C			Acc _T			Acc _D				Acc
	Why	How	All	Prev&Next	Present	All	Count	Location	Other	All	
Zero Shot	80.1	75.9	79.1	71.6	76.3	71.9	68.3	93.5	93.6	86.3	79.1
Full FT	81.2	78.3	80.4	77.1	80.9	78.8	68.3	94.6	89.8	86.5	80.9
LoRA FT	80.7	78.0	80.0	76.3	82.8	78.0	67.7	93.2	88.3	85.3	80.2
Ours	81.5	79.5	81.0	76.5	82.3	79.0	70.2	96.1	89.8	87.5	81.4

Table 2: Comparison of different fine-tuning on InternVL3 and SmolVLM2 across multiple categories in the smart-home benchmark.

Model	Method	baby	light	pet	senior	temporal	vehicle	wildlife	AVG.
InternVL3-8B	Zero Shot	70	75.4	71.5	88.1	68.8	62.7	72.5	71.07
	Full FT	70	66.9	73.8	92.9	68.1	64.2	75.4	71.73
	Lora FT	68	75.7	72.7	88.1	70.9	63.7	73.2	71.87
	Ours	72	74.6	76.2	90.5	66.3	66.3	75.9	73.35
SmolVLM2-2.2B	Zero Shot	54	45.1	44.2	85.7	48.2	36.6	52.3	48.35
	Full FT	54	43.0	45.1	81.0	49.4	37.3	55.8	49.56
	Lora FT	52	42.9	45.9	80.9	51.8	35.1	47.7	46.61
	Ours	56	45.6	45.4	87.7	49.4	39.8	57.8	51.42

sual comparison between DA-MoE and other fine-tuning methods. In this example, the question asks: “How many items did the boy put on the shelf in this video?” All baseline models (Zero-Shot, Full Fine-tuning, and LoRA Fine-tuning) incorrectly predict the number as three, whereas DA-MoE produces the correct answer, two. This example primarily evaluates the model’s counting ability. In the video, the boy places both his own toy and the man’s toy from the scene back onto the shelf, corresponding to two distinct items. However, in the first few frames, the toy in the boy’s hands appears as two separate objects, which misleads baseline models (Zero-Shot, Full Fine-tuning, and LoRA Fine-tuning) into predicting three. In contrast, DA-MoE correctly recognizes that both hands are holding the same toy and accurately counts two placement actions. This example aligns with the quantitative results in Table 1, where DA-MoE shows a improvement in the Count category.

4.3 Model Efficiency

To evaluate efficiency, we compare our method with other fine-tuning strategies in terms of trainable rate and activated parameters. As shown in Table 7, the trainable rate measures the proportion of weights updated during fine-tuning, while activated parameters denote the number of parameters involved during inference. Compared to full fine-tuning, which updates all model weights, LoRA re-

Table 3: Comparison of parameter utilization across different fine-tuning strategies on InternVL3-8B and SmolVLM2-2.2B. “Tra.” shows the ratio of trainable weights during fine-tuning, while “Act.” denote the number of parameters actively involved during inference. “Tot.” denote the total parameter number

Model	Method	Tra.	Act.	Tot.
InternVL3-8B	Full FT	96.2%	8B	8B
	LoRA FT	1.05%	8B	8B
	Ours	1.05%	6.6B	10.8B
SmolVLM2-2.2B	Full FT	100%	2.2B	2.2B
	LoRA FT	1.0%	2.2B	2.2B
	Ours	1.0%	1.9B	3.4B

duces the training cost to about 1% by introducing low-rank adapters. Our method maintains the same low trainable ratio while reducing activated parameters during inference (e.g., from 8B to 6.6B on InternVL3-8B). These results demonstrate a better trade-off between training efficiency and inference computation while preserving model capacity.

4.4 Ablation Study

Component Analysis. To understand the contribution of each component in DA-MoE, we conduct ablation studies on NExT-QA (Table 4). We begin by examining single-pruning settings. Initializing the model with a single pruned subnetwork (SP Init) leads to a slight performance drop compared to the zero shot model (77.4% vs. 79.2%). This



Figure 4: Visual comparison on a video counting question. The baselines (Zero-Shot, Full Fine-tuning, LoRA Fine-tuning) all miscount three items, while our DA-MoE correctly recognizes two placements.

Table 4: Ablation study on DA-MoE configurations. Starting from pruned subnetworks, progressively adding adaptation and MoE routing leads to steady performance gains. Notably, even without fine-tuning, the model initialized with diversity pruning already outperforms the zero-shot baseline.

Methods	Acc_C			Acc_T			Acc_D				Acc
	Why	How	All	Prev&Next	Present	All	Count	Location	Other	All	
Zero Shot	80.1	75.9	79.1	71.6	76.3	71.9	68.3	93.5	93.6	86.3	79.1
SP Init	77.5	74.4	76.7	69.4	79.5	73.8	68.6	94.2	88.8	86.0	77.4
SP FT	79.2	76.6	78.5	74.6	81.3	77.5	65.2	93.8	90.0	85.6	79.4
SP MoE	79.6	78.4	79.3	76.4	80.6	78.3	67.7	94.8	89.2	86.2	80.1
DP Init	79.4	78.2	79.1	73.1	80.2	76.2	68.9	93.2	89.5	86.0	79.5

Table 5: Ablation study on the placement of DA-MoE layers within the model architecture. We compare four variants: applying DA-MoE to the first half, the second half, all layers, and our interval-based design.

Architecture	Acc_C			Acc_T			Acc_D				Acc
	Why	How	All	Prev&Next	Present	All	Count	Location	Other	All	
First-Half	80.7	79.2	80.4	76.9	80.8	78.6	64.9	94.6	88.7	85.2	80.6
Second-Half	80.8	78.3	80.1	76.1	79.7	77.7	66.5	93.0	87.5	84.6	80.0
All	81.2	78.2	80.5	76.8	81.1	78.6	72.0	96.9	89.5	88.1	81.2
Ours	81.5	79.5	81.0	76.5	82.3	79.0	70.2	96.1	89.8	87.5	81.4

behavior is expected, as pruning removes a portion of model capacity. Nevertheless, the performance degradation is marginal, suggesting that the pruned subnetwork still preserves most of the task-relevant knowledge from the pretrained model. Next, we fine-tune the single pruned model (SP FT), which increases accuracy to 79.4%. This behavior aligns with our expectation that adaptation helps recover capacity lost during pruning and further tailor the subnetwork to the downstream task. We then evaluate a variant that applies MoE over a single pruning configuration (SP MoE), achieving 80.1%, thereby further enhancing fine-tuning performance. Notably, even without fine-tuning, the model initialized with diversity-pruned subnetworks (DP Init) surpasses the zero-shot baseline, suggesting that decomposing a pre-trained model into task-relevant subnetworks provides a stronger initialization than

the original pre-trained model itself.

Effect of the Architectures. In our experiments, not all Transformer decoder blocks are replaced with DA-MoE layers. The remaining blocks that are not converted to DA-MoE use a sparse FFN as a lightweight alternative without applying MoE routing. As shown in Table 5, we explore four architectural variants of the DA-MoE design. Specifically, First-Half applies DA-MoE layers only to the first half of the decoder, while the second half uses sparse FFN layers. Second-Half reverses this configuration, applying DA-MoE layers to the latter half of the model. All replaces every decoder block with DA-MoE layers. Ours alternates DA-MoE and sparse FFN layers throughout the network. Empirically, ours proves to be an effective choice, achieving a better performance.

5 Conclusion

In this work, we introduce DA-MoE, a new fine-tuning paradigm for VLMs that challenges the conventional assumption of preserving the full pre-trained structure. DA-MoE divides the pre-trained model into multiple downstream-oriented subnetworks and adapts them collaboratively through a routing-based mechanism. This strategy enables efficient specialization while reusing rich pre-trained knowledge for downstream tasks. Extensive experiments on public benchmark and a practical task demonstrate that DA-MoE consistently outperforms both full fine-tuning and parameter-efficient baselines, highlighting its potential as an efficient and scalable fine-tuning framework for large-scale multimodal models.

Limitations

In this paper, we incorporate the MoE mechanism into the model architecture. Although MoE reduces the number of activated parameters during inference compared to dense models, it does not reduce the total number of model parameters. As a result, the practical inference latency and deployment cost are not significantly improved. A promising future direction is to combine MoE with structural pruning techniques to derive compact subnetworks from the original model. Such subnetworks could reduce actual inference latency, leading to more efficient deployment in real-world scenarios.

References

- Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.
- Yuyan Chen, Qiang Fu, Ge Fan, Lun Du, Jian-Guang Lou, Shi Han, Dongmei Zhang, Zhixu Li, and Yanghua Xiao. 2023. Hadamard adapter: An extreme parameter-efficient adapter tuning method for pre-trained language models. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 276–285.
- Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, and 1 others. 2024. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24185–24198.
- Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. 2022. Prototypical verbalizer for prompt-based few-shot tuning. *arXiv preprint arXiv:2203.09770*.
- Erich Elsen, Marat Dukhan, Trevor Gale, and Karen Simonyan. 2020. Fast sparse convnets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14629–14638.
- Marco Federici, Davide Belli, Mart Van Baalen, Amir Jalalirad, Andrii Skliar, Bence Major, Markus Nagel, and Paul Whatmough. 2024. Efficient llm inference using dynamic input pruning and cache-aware masking. *arXiv preprint arXiv:2412.01380*.
- William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.
- Hao Fei, Shengqiong Wu, Wei Ji, Hanwang Zhang, Meishan Zhang, Mong-Li Lee, and Wynne Hsu. 2024. Video-of-thought: Step-by-step video reasoning from perception to cognition. *arXiv preprint arXiv:2501.03230*.
- Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.
- Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pages 10323–10337. PMLR.
- Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.
- Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.
- Pengfei He. 2024. Parameter efficient instruction tuning: An empirical study. *arXiv preprint arXiv:2411.16775*.
- Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jiawei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 2208–2222.
- Wenyi Hong, Weihang Wang, Ming Ding, Wenmeng Yu, Qingsong Lv, Yan Wang, Yean Cheng, Shiyu Huang, Junhui Ji, Zhao Xue, and 1 others. 2024. Cogvlm2: Visual language models for image and video understanding. *arXiv preprint arXiv:2408.16500*.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.
- Wenbo Hu, Yifan Xu, Yi Li, Weiyue Li, Zeyuan Chen, and Zhuowen Tu. 2024. Bliva: A simple multimodal llm for better handling of text-rich visual questions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 2256–2264.
- Shashank Mohan Jain. 2022. Hugging face. In *Introduction to transformers for NLP: With the hugging face library and models to solve problems*. Springer.
- Ahmadreza Jeddi, Negin Baghbanzadeh, Elham Dolatabadi, and Babak Taati. 2025. Similarity-aware token pruning: Your vlm but faster. *arXiv preprint arXiv:2503.11549*.
- Titong Jiang, Xuefeng Jiang, Yuan Ma, Xin Wen, Bailin Li, Kun Zhan, Peng Jia, Yahui Liu, Sheng Sun, and Xianpeng Lang. 2025. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. *arXiv preprint arXiv:2509.12594*.
- Dawid J Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2023. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*.
- Kunpeng Li, Yulun Zhang, Kai Li, Yuanyuan Li, and Yun Fu. 2019. Visual semantic reasoning for image-text matching. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4654–4662.

- Bin Lin, Yang Ye, Bin Zhu, Jiayi Cui, Munan Ning, Peng Jin, and Li Yuan. 2024. Video-llava: Learning united visual representation by alignment before projection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5971–5984.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.
- Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Jianglin Lu, Hailing Wang, Xu Ma, Qihua Dong, Mingyuan Zhang, Yizhou Wang, and Yun Fu. 2026a. Muse: A unified agentic harness for mllms. *arXiv preprint arXiv:2606.03005*.
- Jianglin Lu, Hailing Wang, Kuo Yang, Yitian Zhang, Simon Jenni, and Yun Fu. 2026b. The indra representation hypothesis for multimodal alignment. *Advances in Neural Information Processing Systems*, 38:156980–157000.
- Duc-Tuan Luu, Viet-Tuan Le, and Duc Minh Vo. 2024. Questioning, answering, and captioning for zero-shot detailed image caption. In *Proceedings of the Asian Conference on Computer Vision*, pages 242–259.
- Ratajczak Waldemar Maćkiewicz, Andrzej. 1993. Principal components analysis (pca). *Computers & Geosciences*.
- Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, and 1 others. 2025. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Neurips*.
- Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. 2023. From sparse to soft mixtures of experts. *arXiv preprint arXiv:2308.00951*.
- Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *ICML*. PMLR.
- Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. 2024. Tied-lora: Enhancing parameter efficiency of lora with weight tying. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8694–8705.
- Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, and 1 others. 2024. Llm pruning and distillation in practice: The minitron approach. *arXiv preprint arXiv:2408.11796*.
- Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.
- Zehua Sun, Huanqi Yang, Kai Liu, Zhimeng Yin, Zhenjiang Li, and Weitao Xu. 2022. Recent advances in lora: A comprehensive survey. *ACM Transactions on Sensor Networks*, 18(4):1–44.
- Jothi Prasanna Shanmuga Sundaram, Wan Du, and Zhiwei Zhao. 2019. A survey on lora networking: Research problems, current solutions, and open issues. *IEEE Communications Surveys & Tutorials*, 22(1):371–388.
- Chaozheng Wang, Jia Feng, Shuzheng Gao, Cuiyun Gao, Zongjie Li, Ting Peng, Hailiang Huang, Yuetang Deng, and Michael Lyu. 2025a. Beyond peft: Layer-wise optimization for more effective and efficient large code model tuning. *Proceedings of the ACM on Software Engineering*, 2(FSE):1567–1590.
- Hailing Wang, Jianglin Lu, Yitian Zhang, and Yun Fu. 2025b. Outlier-aware post-training quantization for image super-resolution. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16175–16184. IEEE.
- Hailing Wang, Jianglin Lu, Yitian Zhang, Huimin Zeng, and Yun Fu. 2026. Acqbq: Adaptive cross-block quantization of large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 42578–42592.
- Haoran Wang, Yue Zhang, and Xiaosheng Yu. 2020. An overview of image caption generation methods. *Computational intelligence and neuroscience*, 2020(1):3062706.
- Yuetian Weng, Mingfei Han, Haoyu He, Xiaojun Chang, and Bohan Zhuang. 2024. Longvlm: Efficient long video understanding via large language models. In *European Conference on Computer Vision*, pages 453–470. Springer.
- Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. 2021. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9777–9786.

- Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.
- Xu Yang, Yongliang Wu, Mingzhuo Yang, Haokun Chen, and Xin Geng. 2023. Exploring diverse in-context configurations for image captioning. *Advances in Neural Information Processing Systems*, 36:40924–40943.
- Ted Zadori, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. 2023. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*.
- Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.
- Beier Zhu, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang. 2023. Prompt-aligned gradient for prompt tuning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15659–15669.
- Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, and 1 others. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*.

Appendix

A A Practical Vertical-Domain Task

In practical applications, the zero-shot capability of pre-trained models is often insufficient to achieve optimal performance. Domain shifts between pre-training data and downstream tasks usually require additional fine-tuning to adapt the model effectively. This is particularly evident in the smart-home domain, where the data distribution differs from that of large-scale public datasets. To facilitate domain-specific fine-tuning, we collect a large-scale smart-home video dataset comprising 102,040 daily-life videos captured in residential environments using egocentric cameras. All data were collected with informed consent from the participants and anonymized prior to research use to protect personal privacy. The dataset is a proprietary in-house dataset provided by our industry collaborator and is used in accordance with the collaborator’s internal data-use and legal requirements. To further safeguard participant privacy, the underlying video data are not publicly released.

To ensure high-quality and consistent annotations, we employ a semi-automated annotation pipeline that combines model-assisted generation with human verification. Specifically, Gemini-2.5-Pro first produces context-aware video summaries and drafts of multiple-choice questions grounded in observable visual content. Human annotators then perform validation to ensure that each question and its answer are visually grounded, unambiguous, and independent of any non-visible or external information. Our smart-home benchmark includes 1,143 videos covering diverse residential environments, lighting conditions, subject scales, and camera viewpoints. A total of 1,154 multiple-choice questions are annotated across seven functional categories to evaluate visual reasoning capabilities. Dataset statistics, including video counts and average durations per category are provided in Table 6.

B Experimental Setup Details

We implement DA-MoE using PyTorch (Paszke et al., 2019), leveraging the Hugging Face (Jain, 2022) Transformers and Datasets libraries for model and data management. The MoE functionality is built upon the DeepSpeed-MoE (Rajbhandari et al., 2022) framework. All experiments are conducted on a server with four NVIDIA A100 GPUs,

Table 6: Summary of smart-house benchmark, including video counts and mean video duration for each category.

	Baby	Senior	Pet	Wildlife	Vehicle	Temporal	Lighting
Video Count	50	32	172	408	195	141	145
Mean Duration (s)	27.1	27.8	25.8	25.0	27.3	29.1	23.1
Total	1,143 videos, mean duration 25.9 s						

each with 80 GB of memory. We use AdamW (Loshchilov and Hutter, 2017) for all models, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, a weight decay of 0.02, and a warm-up ratio of 0.02. Given the video processing setting, each input sample contains up to 16 frames, which are processed together with the corresponding text sequence. The maximum sequence length is fixed at 8192 tokens to accommodate long multimodal contexts. During full fine-tuning, the parameters of the vision encoder are kept fixed, while both the LLM and MLP projector are fully updated to adapt to the downstream task. In the LoRA fine-tuning setting, both the vision encoder and MLP projector are frozen, while LoRA adapters with a rank of 32 are inserted into the LLM to enable efficient parameter updates. For our DA-MoE method, we begin by pruning 50% of the parameters in the LLM’s FFNs using two pruning strategies, namely Wanda (Sun et al., 2023) and SparseGPT (Frantar and Alistarh, 2023), together with two calibration datasets sampled from different random seeds, producing four structurally diverse subnetworks. We then replace every other FFN layer in the LLM with an DA-MoE, where these four pruned FFNs serve as the experts for initialization. The constructed DA-MoE can subsequently be optimized using either full or LoRA fine-tuning. For computational efficiency, we employ LoRA fine-tuning with a rank of 32 in all experiments.

C Hyperparameter Analysis

In our experiments, we analyze two hyperparameters: the LoRA rank that determines the fine-tuning capacity and the coefficient α that balances the auxiliary loss. As shown in Figure 5, we set the LoRA rank to 32, which achieves the best overall performance. Testing smaller (16) and larger (64) ranks results in inferior performance, indicating that excessively low or high adaptation capacity can both hinder effective fine-tuning. For the balancing coefficient α , we experiment with three values: 0.001, 0.01, and 0.1. As shown in Figure 6, the model achieves the best performance when $\alpha = 0.01$,

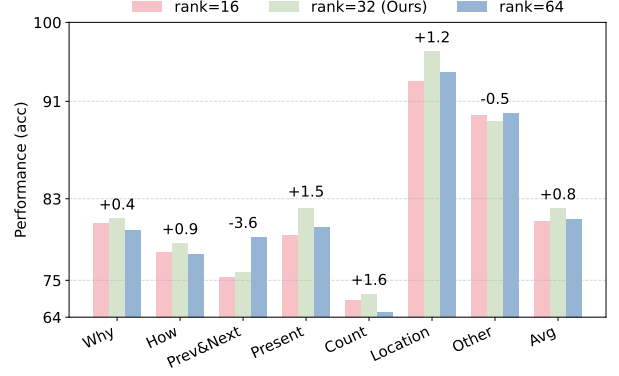


Figure 5: Sensitivity analysis of the LoRA rank. We compare ranks of 16, 32, and 64 across different question categories. The configuration with rank 32 (ours) achieves the best overall accuracy. Both smaller and larger ranks lead to slight performance drops, indicating that an intermediate rank provides a balanced fine-tuning capacity.

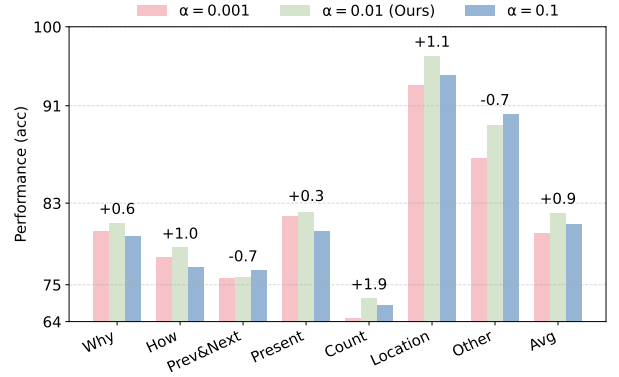


Figure 6: Sensitivity analysis of the balancing coefficient α . We evaluate $\alpha = 0.001$, 0.01, and 0.1. The best is achieved at $\alpha = 0.01$, while both smaller and larger values degrade results, indicating that a moderate auxiliary loss weight provides the best trade-off.

while too small or too large values lead to suboptimal results due to under- or over-emphasis of the auxiliary loss.

D Trainable and Activated Parameters

As shown in Table 7, trainable rates indicate the proportion of model weights updated during fine-tuning, while activated parameters represent the

Table 7: Comparison of parameter utilization across different strategies on InternVL3-8B and SmolVLM2-2.2B. “Activated Parameters” denote the activated parameters during inference, while “Trainable Parameters” show the ratio of trainable weights.

Model	Method	Experts	Top-k	MoE Layers	FFN Layers	Trainable Rate	Activated Parameters	Total Parameters
InternVL3-8B	Full Fine-Tuning	-	-	-	28	96.2%	8B	8B
	LoRA Fine-Tuning	-	-	-	28	1.05%	8B	8B
	Ours	4	2	14	14 (Sparse)	1.05%	6.6B	10.8B
SmolVLM2-2.2B	Full Fine-Tuning	-	-	-	24	100%	2.2B	2.2B
	LoRA Fine-Tuning	-	-	-	24	1.0%	2.2B	2.2B
	Ours	4	2	12	12 (Sparse)	1.0%	1.9B	3.4B

Table 8: Comparison of different expert construction strategies without fine-tuning.

Expert Construction	Acc _C			Acc _T			Acc _D			Acc
	Why	How	All	Prev&Next	Present	All	Count	Location	Other	All
Homogeneous Expert Construction	80.2	75.9	79.0	73.0	79.2	75.8	70.5	92.3	87.8	79.1
PCA-Guided Expert Construction	80.3	77.5	79.6	73.0	79.1	75.7	71.4	94.4	88.3	79.5
Strategy-Guided Expert Construction	79.4	78.2	79.1	73.1	80.2	76.2	68.9	93.2	89.5	79.5

number of parameters actively involved during inference. Compared to full fine-tuning, which requires updating almost all model parameters, LoRA fine-tuning drastically reduces the training cost to about 1% of the total parameters by introducing low-rank adapters into selected layers. Our method maintains the same low trainable ratio, effectively reducing the activated parameters during inference (e.g., from 8B to 6.6B on InternVL3-8B and 2.2B to 1.9B on SmolVLM2-2.2B). This reduction arises from our design: aside from the DA-MoE layers, all remaining FFN layers are sparse (50% pruned), resulting in fewer active parameters during inference. As a result, our approach attains a more efficient trade-off between training cost and inference computation while preserving the overall representational capacity of the model.

Efficiency Analysis. We evaluate the computational efficiency of DA-MoE in terms of activated parameters and inference FLOPs as shown in 9. Although DA-MoE increases the total parameter by introducing multiple sparse experts, only a subset of experts is activated for each token through top- k routing, thereby reducing the effective computation during inference. Under the same input settings, the dense baseline activates 8B parameters and requires 13.05 GFLOPs per token, whereas DA-MoE activates 6.6B parameters and requires 10.20 GFLOPs per token. This corresponds to an around 21.8% reduction in per-token computation.

Table 9: Computational efficiency comparison under the same input setting.

Method	Activated Params.	FLOPs per Token
Dense Model	8B	13.05 GFLOPs
DA-MoE (Ours)	6.6B	10.20 GFLOPs

E Distribution of Expert Loadings

To further analyze the internal routing behavior of DA-MoE, we visualize the layer-wise expert activation ratios on two benchmarks, as shown in Figure 7 and Figure 8. Each stacked bar represents the relative activation frequency of four experts within each DA-MoE layer. To begin with, the expert loads across all DA-MoE layers are generally well balanced. We further observe that different layers rely on different combinations of experts, implying that DA-MoE has learned an implicit collaboration pattern in which experts specialize and cooperate to process inputs. Although the detailed distributions vary slightly across different benchmarks, the overall activation patterns remain consistent. For example, the activation rate of Expert 2 in Layer 0 and that of Expert 3 in Layer 2 are both relatively low in the two benchmarks.

F Effect of Expert Diversity

Table 8 compares the performance of different expert construction strategies without fine-tuning. In the PCA-Guided Expert Construction, we first ap-

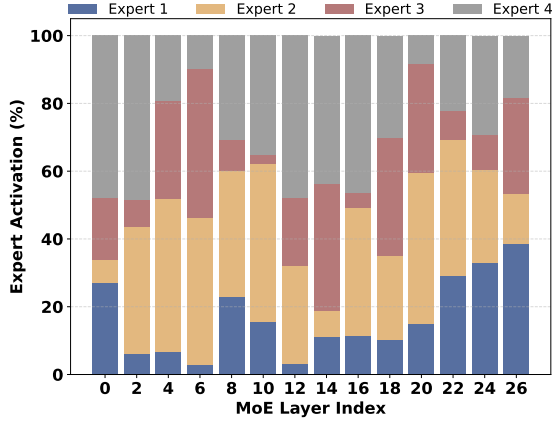


Figure 7: Layer-wise expert activation ratios in the proposed DA-MoE on NExT-QA. Different experts are selectively activated at varying frequencies across layers.

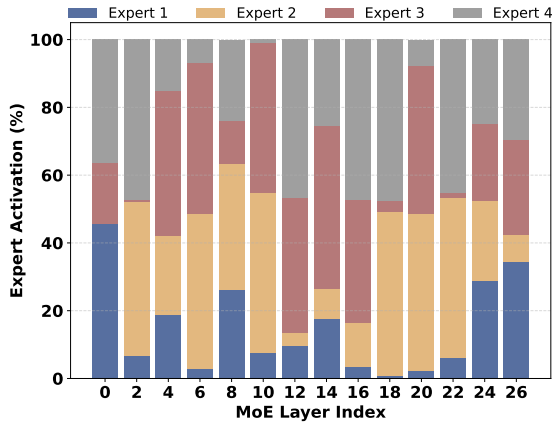


Figure 8: Layer-wise expert activation ratios in the proposed DA-MoE on smart-home benchmark. Different experts are activated at varying frequencies across layers.

ply PCA (Maćkiewicz, 1993) to the training data and cluster it into four groups. Samples are then drawn from each cluster, and four distinct FFN subnetworks are obtained by pruning them with the Wanda method (Sun et al., 2023). In contrast, the Strategy-Guided Expert Construction uses different random seeds to select varying calibration subsets and applies two pruning strategies (Wanda (Sun et al., 2023) and SparseGPT (Frantar and Alistarh, 2023)) to each subset. The results show that introducing expert diversity through different initialization methods can effectively improve model specialization and overall performance. Specifically, both PCA-Guided and Strategy-Guided Expert Construction outperform the Homogeneous Expert Construction baseline across most categories, indicating that diversity among experts en-

hances the model’s adaptability to different question types. These results suggest that diverse pruning strategies or calibration criteria help each expert specialize in complementary aspects of the task, enabling more effective cooperation within the DA-MoE framework.

G LLM Usage

Large language models (LLMs) were used only as writing assistance tools for improving grammar, wording, and readability of the manuscript.