WHERE, HOW MUCH, AND WHAT TO RETAIN: SPARSE OPTIMIZATION FOR MULTIMODAL LLM FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal Large Language Models (MLLMs) exhibit strong general capabilities across vision-language tasks, yet downstream fine-tuning can degrade pre-trained capabilities due to catastrophic forgetting. Existing approaches mitigate this issue through post-hoc model merging or sparse fine-tuning. While dynamic masking improves upon fixed sparse updates by adapting parameter selection during training, our analysis reveals that step-wise mask reconstruction can incur considerable memory overhead while committing to most of its eventual update region early in optimization, limiting subsequent parameter reallocation. To address these limitations, we propose the *Stability-Plasticity-Aware Sparse Optimizer (SPSO)*, which formulates sparse fine-tuning as an optimizer-level dynamic sparse optimization problem. SPSO introduces a *Safe-Update Score (SUS)* that combines a lightweight pre-trained weight-based preservation prior with online gradient-based task relevance. Guided by SUS, SPSO dynamically reallocates a fixed-size active set, scales parameter-wise updates, and selectively retains sparse Adam states, jointly controlling *where to update*, *how much to update*, and *what to retain*. Experiments across multiple MLLMs and downstream tasks show that SPSO consistently improves the balance between pre-trained capability retention and task adaptation while maintaining sparse updates and reducing optimization memory.

1 INTRODUCTION

Multimodal large language models (MLLMs) Achiam et al. (2023); Bai et al. (2025); Chen et al. (2024b), pretrained on large-scale multimodal corpora, have demonstrated strong zero-shot performance across diverse vision-language tasks, such as visual question answering Tang et al. (2025); Masry et al. (2025) and multimodal reasoning Sun et al. (2025); Lee et al. (2024). However, adapting these models to new domains, data distributions, or task formats often requires downstream fine-tuning. Such fine-tuning can introduce dense and unconstrained parameter updates that interfere with broadly useful pre-trained capabilities Zhai et al. (2024); Zhu et al. (2024), leading to catastrophic forgetting McClelland et al. (1995); McCloskey & Cohen (1989). Therefore, a central challenge in MLLM fine-tuning is not merely to improve target-task performance, but to achieve a balance between *plasticity* for learning new tasks and *stability* for preserving pre-trained capabilities.

Existing approaches to mitigating forgetting in MLLMs mainly fall into two categories: *post-hoc model merging* Zhu et al. (2024); Xie et al. (2025) and *sparse fine-tuning* Liao et al. (2023); Huang et al. (2025b); Liang et al. (2025). Post-hoc methods recover pre-trained capabilities by filtering, rescaling, or recombining parameter updates after fine-tuning. However, their post-hoc nature offers limited control over fine-tuning updates and may compromise target-task adaptation when restoring pre-trained capabilities. By contrast, sparse fine-tuning methods restrict which parameters are updated during training, thereby reducing interference with pre-trained parameters. Broadly, they can be categorized into *static* and *dynamic* masking strategies. Static methods select a fixed update subset before Liao et al. (2023) or early Liang et al. (2025) in fine-tuning and retain it throughout training. While simple and efficient, this design assumes that parameter relevance remains stable throughout optimization. As gradient distributions and parameter relevance evolve, however, a fixed mask cannot reallocate its limited update capacity accordingly. Dynamic methods address this limitation by adapting parameter selection to evolving optimization signals. SPIDER Huang et al. (2025b), for example, combines frozen pre-trained weight magnitudes with accumulated downstream gradient magnitudes to estimate parameter importance to the pre-trained model and target task, respectively.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

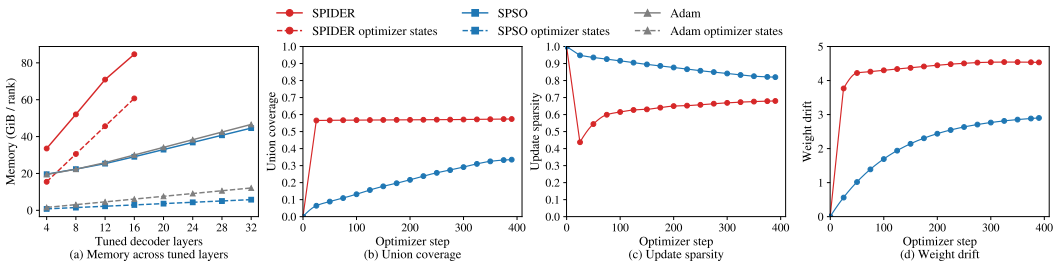


Figure 1: Memory and optimization dynamics across fine-tuning strategies. (a) Total (solid) and optimizer-state (dashed) memory across different numbers of tuned decoder layers. (b) Cumulative fraction of parameters activated during training (union coverage). (c) Fraction of parameters remaining close to their pre-trained values (update sparsity). (d) Weight drift from the pre-trained model.

At each optimization step, it compares these signals to reconstruct the update mask, determining both which parameters are updated and their corresponding mask weights. This provides a representative form of dynamic sparse fine-tuning by repeatedly reconsidering which parameters should be updated as optimization progresses. However, whether step-wise mask reconstruction translates into a memory-efficient and genuinely evolving sparse optimization process remains unclear.

Figure 1 reveals two limitations of step-wise dynamic masking. First, SPIDER introduces additional memory overhead. Beyond the dense first- and second-moment states maintained by Adam, its mask construction retains a frozen copy of the pre-trained weights and accumulated gradient statistics, both scaling with the tuned parameters. Consequently, memory grows with the number of tuned decoder layers; under our default 12-layer setting, SPIDER reaches 70.91 GiB peak memory versus 25.84 GiB for standard fine-tuning (Figure 1(a)). Second, frequent mask reconstruction does not necessarily yield sustained update-space reallocation. When fine-tuning LLaVA-1.5 on IconQA, the first-step active parameters already account for approximately 94% of all parameters ever activated during training (Figure 1(b)), leaving limited subsequent growth in cumulative update coverage. Meanwhile, Figure 1(c) shows that fewer parameters remain close to their pre-trained values under SPIDER, while Figure 1(d) shows that weight drift rises early and changes gradually thereafter. Together, these observations reveal an *early-commitment* behavior: although the mask is reconstructed at every step, SPIDER quickly establishes a broad effective update region, with subsequent optimization largely confined to this parameter set. This suggests that dynamic mask construction alone is insufficient to govern how sparse updates evolve during training. It therefore raises a broader question: *how can a bounded per-step update budget be continually reallocated while avoiding dense optimizer states and unnecessary parameter drift?*

Motivated by these observations, we propose the *Stability-Plasticity-Aware Sparse Optimizer (SPSO)*, an optimizer-level framework that jointly controls parameter selection, optimizer-state retention, and update magnitude under a fixed per-step budget. SPSO introduces a *Safe-Update Score (SUS)*, which combines the *Knowledge-Retention Score (KRS)*, using pre-trained weight magnitude as a lightweight preservation prior, with the *Task-Adaptation Score (TAS)*, capturing evolving downstream demands from online gradients. Rather than maintaining dense optimizer states, SPSO stores Adam moments within a bounded sparse-state budget, reducing memory growth as more layers are tuned (Figure 1(a)). Meanwhile, SPSO periodically reallocates its fixed-size active set using SUS. This decouples *per-step sparsity* from *cumulative parameter exploration*: only a small subset is updated per step, while different subsets become active over training. Figure 1(b) shows that SPSO progressively expands its cumulative update coverage instead of committing to most of its effective update region early in training. It also keeps more parameters close to their pre-trained values (Figure 1(c)). SUS further scales parameter-wise Adam updates, making updates more conservative for parameters with stronger pre-trained saliency or weaker task relevance, thereby reducing weight drift from the pre-trained model (Figure 1(d)). When the active set changes, SPSO retains informative optimizer states within the fixed sparse-state budget, preserving useful optimization history without dense state maintenance. Overall, SPSO extends sparse fine-tuning beyond mask construction to jointly determine *where to update*, *how much to update*, and *what to retain*.

The main contributions of this work are as follows: (1) We revisit dynamic sparse fine-tuning from the perspective of optimization dynamics and identify an *early-commitment* phenomenon: step-wise mask reconstruction can incur considerable memory overhead while activating most of its eventual update region early in training, leaving limited subsequent reallocation across parameters. (2) We

propose the *Stability-Plasticity-Aware Sparse Optimizer (SPSO)*, an optimizer-level framework that uses the Safe-Update Score under a fixed sparsity budget to jointly control active-parameter selection, parameter-wise update scaling, and sparse moment-state retention, addressing *where to update*, *how much to update*, and *what to retain*. (3) We validate SPSO across multiple MLLMs, target tasks, and extended settings, with ablation, training-dynamics, and efficiency analyses further examining its effectiveness in balancing pre-trained capability retention and downstream adaptation.

2 Related Work

Multimodal Large Language Models. The development of multimodal large language models (MLLMs) Li et al. (2023); Dai et al. (2023); Bai et al. (2025); Liu et al. (2023; 2025); Deitke et al. (2025) has evolved from visual-language representation learning toward general-purpose multimodal assistants. Early models such as CLIP Radford et al. (2021) established transferable visual representations through large-scale image–text pre-training. Later approaches connected pre-trained vision encoders with LLMs through lightweight cross-modal interfaces, including projection layers and Q-Former modules Alayrac et al. (2022); Li et al. (2023). Visual instruction tuning subsequently enabled models such as LLaVA Liu et al. (2023) and LLaVA-1.5 Liu et al. (2024a) to perform diverse multimodal understanding and reasoning tasks. More recent MLLMs have further scaled visual resolution, training data, and temporal modeling to support more diverse visual inputs and tasks. NVILA Liu et al. (2025) represents this line of development with improved efficiency and image/video understanding, while Molmo2 Clark et al. (2026) further extends multimodal modeling toward richer multi-image and video-centric capabilities. In this work, we evaluate SPSO on LLaVA-1.5, NVILA-Lite-8B, and Molmo2, spanning multiple generations of MLLMs.

Catastrophic Forgetting In deep learning, catastrophic forgetting refers to the degradation of a model's previously acquired knowledge or capabilities when learning new tasks or data McClelland et al. (1995); Kirkpatrick et al. (2017); Parisi et al. (2019). Classical continual learning mitigates this problem through rehearsal-based Chaudhry et al. (2019); Lopez-Paz & Ranzato (2017); Riemer et al. (2019), regularization-based Kirkpatrick et al. (2017); Schwarz et al. (2018); Liu et al. (2018), and architecture-based methods Yoon et al. (2017); Wang et al. (2023); Razdaibiedina et al. (2023). In LLMs and MLLMs, catastrophic forgetting can manifest as degraded generalization after downstream adaptation Wang et al. (2024). Existing mitigation methods for MLLMs mainly fall into post-hoc model merging and sparse fine-tuning. Post-hoc methods modify parameter updates after fine-tuning: DARE Yu et al. (2024) drops and rescales weight deltas, Grafting Panigrahi et al. (2023) selects task-specific updates, Model Tailor Zhu et al. (2024) constructs sparse model patches, and AdaDARE Xie et al. (2025) adaptively filters task-relevant updates. However, they provide limited control over the fine-tuning trajectory and may compromise downstream performance when recovering pre-trained capabilities. Sparse fine-tuning instead mitigates interference by selectively allocating updates to a subset of parameters during optimization. Among static approaches, FISH Mask Sung et al. (2021) derives its sparse mask from gradient information, whereas PaFi Liao et al. (2023) and LoRA-Sculpt Liang et al. (2025) rely on weight-magnitude information. However, such fixed update patterns cannot reallocate update capacity as parameter relevance evolves. Dynamic masking addresses this limitation by adapting parameter selection to evolving optimization signals. SPIDER Huang et al. (2025b), for example, combines pre-trained weight magnitudes with accumulated downstream gradients to reconstruct the mask at each optimization step. Yet step-wise recomputation adds memory overhead without ensuring effective reallocation of the sparse update budget. More broadly, existing sparse methods mainly treat sparsity as a mask-construction problem, with limited joint control over parameter selection, update magnitude, and optimizer states. SPSO instead formulates sparse fine-tuning as optimizer-level dynamic sparse optimization, jointly controlling these aspects under a fixed sparsity budget through the Safe-Update Score.

3 Preliminary

Adam Optimizer Adam (Kingma & Ba, 2015) and its variants are widely used for supervised fine-tuning (SFT) of MLLMs Liu et al. (2024a); Chen et al. (2024c); Bai et al. (2023); Wu et al. (2024). Adam adapts parameter updates using exponential moving averages of the first- and second-order gradient moments. For parameter w_j, given the gradient $g_j^{(t)}$ at step t, the moment estimates are

updated as

$$u_j^{(t)} = \beta_1 u_j^{(t-1)} + (1-\beta_1) g_j^{(t)}, \tag{1}$$

$$v_j^{(t)} = \beta_2 v_j^{(t-1)} + (1-\beta_2)\big(g_j^{(t)}\big)^2, \tag{2}$$

where $\beta_1, \beta_2 \in [0, 1)$ are decay coefficients. Since both moment estimates are initialized as zeros, Adam applies bias correction to compensate for their initialization bias:

$$\hat{u}_j^{(t)} = \frac{u_j^{(t)}}{1-\beta_1^t}, \qquad \hat{v}_j^{(t)} = \frac{v_j^{(t)}}{1-\beta_2^t}. \tag{3}$$

The parameter is then updated by

$$w_j^{(t+1)} = w_j^{(t)} - \eta \frac{\hat{u}_j^{(t)}}{\sqrt{\hat{v}_j^{(t)}} + \epsilon}, \tag{4}$$

where η is the learning rate and ϵ is a numerical stabilizer.

Problem Definition Let $\mathbf{W}_0$ denote the pre-trained weights, $\mathcal{T}$ the target task, and $\mathcal{P}$ the original tasks. Standard fine-tuning produces an unconstrained dense update $\Delta\mathbf{W}$ for $\mathcal{T}$. Instead, we seek an alternative update $\Delta\mathbf{W}^*$ that preserves target-task performance while limiting degradation on $\mathcal{P}$:

$$\begin{aligned} &\mathcal{L}_{\mathcal{T}}(\mathbf{W}_0 + \Delta\mathbf{W}^*) - \mathcal{L}_{\mathcal{T}}(\mathbf{W}_0 + \Delta\mathbf{W}) \le \epsilon_{\mathcal{T}}, \\ &\mathcal{L}_{\mathcal{P}}(\mathbf{W}_0 + \Delta\mathbf{W}^*) - \mathcal{L}_{\mathcal{P}}(\mathbf{W}_0) \le \epsilon_{\mathcal{P}}. \end{aligned} \tag{5}$$

where $\mathcal{L}_{\mathcal{T}}$ and $\mathcal{L}_{\mathcal{P}}$ denote losses on the target and original tasks, while $\epsilon_{\mathcal{T}}$ and $\epsilon_{\mathcal{P}}$ specify acceptable degradation margins. Our method addresses this objective by constructing $\Delta\mathbf{W}^*$ through stability-plasticity-aware sparse fine-tuning updates.

4 METHOD

To mitigate catastrophic forgetting during fine-tuning, we propose the **Stability-Plasticity-Aware Sparse Optimizer** (SPSO), which addresses the problem from an optimizer-level perspective. Instead of the standard dense update $\Delta\mathbf{W}$, SPSO constructs a sparse counterpart $\Delta\mathbf{W}^*$ over a small, dynamically selected parameter subset (§4.1) selected for task relevance and low interference with pre-trained knowledge. Both parameter selection and update magnitude are guided by a per-parameter Safe-Update Score (SUS, §4.2), where higher scores indicate parameters that are more useful for adaptation and safer to modify. Guided by SUS, SPSO samples active parameters, prunes sparse optimizer states, and scales active updates, thereby jointly controlling update locations, retained states, and update strengths.

4.1 DYNAMIC SPARSE OPTIMIZATION

Standard optimizers Kingma & Ba (2015) apply dense updates to all parameters at every training step, thereby increasing the risk of catastrophic forgetting. In contrast, SPSO adopts a dynamic sparse optimization strategy that confines updates to a small, dynamically selected parameter subset, yielding two complementary advantages. First, this sparsity concentrates changes on a few parameter entries at each step, avoiding indiscriminate modification of pre-trained parameters and thus alleviating catastrophic forgetting. Second, adapting parameter selection to evolving task demands reduces reliance on a fixed parameter group, allowing the optimization to exploit the model's available capacity across training. We now describe how the active parameter set is constructed and how parameters within it are updated.

Dynamic Parameter Set Selection For each trainable parameter, we flatten it into N scalar entries indexed by $j \in \{1, \dots, N\}$. At each optimization step, SPSO updates only an *active set* $\mathcal{A}_t \subseteq \{1, \dots, N\}$ of K indices, where $K = \max\big(1, \lfloor (1-\rho)N \rfloor\big)$ and $\rho \in [0, 1]$ is the *sparsity ratio*, i.e., the fraction of entries in that tensor excluded from updating. To construct $\mathcal{A}_t$, SPSO assigns each parameter entry a sampling weight based on its Safe-Update Score (SUS) $S_j^{(t)} \in [0, 1]$, which

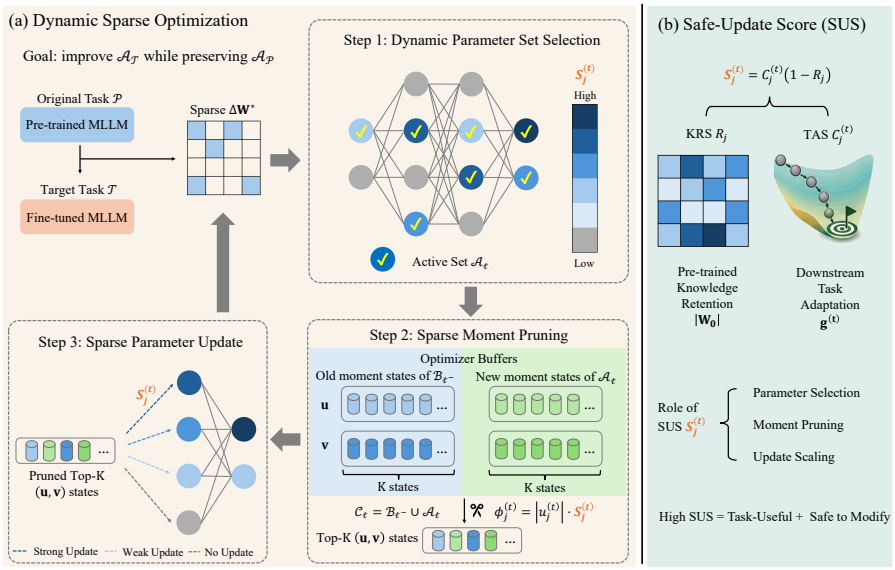


Figure 2: Overview of the proposed Stability-Plasticity-Aware Sparse Optimizer (SPSO). SPSO replaces dense fine-tuning updates with a dynamic sparse optimization pipeline (Figure (a)): It first selects an active parameter set based on Safe-Update Score (SUS), then prunes the sparse optimizer moments to retain task-relevant and safe states, and finally applies SUS-scaled sparse updates on model weights. The SUS combines the Knowledge-Retention Score (KRS), which acts as a pre-trained knowledge preservation prior, and the Task-Adaptation Score (TAS), which identifies parameters useful for the target task (Figure (b)). SUS jointly controls parameter selection, moment pruning, and update scaling to improve task plasticity while preserving pre-trained capabilities.

measures how task-useful and safe to update the entry is, as defined in Section 4.2. When selecting an active set, SPSO samples K entries using the Gumbel top-K trick (Kool et al., 2019):

$$\mathcal{A}_t = \arg\operatorname*{top}_K \left\{ \log S_j^{(t)} + \xi_j \right\}_{j=1}^{N}, \tag{6}$$

where $\xi_j \overset{\text{i.i.d.}}{\sim} \text{Gumbel}(0, 1)$. Consequently, the sampled subset has size K and is biased toward parameters with larger SUS values, while the Gumbel perturbation introduces controlled stochasticity that reduces deterministic reuse of the same high-scoring entries and encourages broader parameter coverage during training.

Sparse Moment Pruning Adam-style updates require first- and second-moment states, but discarding them whenever the active set changes loses useful historical gradient signals, while maintaining dense states for all entries undermines sparsity. SPSO therefore keeps a sparse moment buffer $\mathcal{B}_{t^-}$ with at most K entries retained from previous updates, each storing $\{u_j^{(t)}, v_j^{(t)}\}$. The selected active set $\mathcal{A}_t$ may contain entries outside $\mathcal{B}_{t^-}$, so SPSO forms a candidate state set $\mathcal{C}_t = \mathcal{B}_{t^-} \cup \mathcal{A}_t$, whose size is at most $2K$ since both sets contain at most K entries, with equality when they are disjoint. To restore the K-entry sparse budget, SPSO performs SUS-guided moment pruning by scoring each candidate entry j as

$$\phi_j^{(t)} = \left| u_j^{(t)} \right| \cdot S_j^{(t)}, \tag{7}$$

where $\left| u_j^{(t)} \right|$ captures accumulated task relevance from historical gradients, and $S_j^{(t)}$ reflects update safety under the pretrained-weight preservation prior. SPSO retains the top-K entries ranked by $\phi_j^{(t)}$ together with their first- and second-moment values, thereby preserving momentum states that are both task-relevant and safe to update. Bias correction follows Adam.

Sparse Parameter Update Each parameter j is then updated as:

$$w_j^{(t+1)} = \begin{cases} w_j^{(t)} - \eta \dfrac{\hat{u}_j^{(t)} S_j^{(t)}}{\sqrt{\hat{v}_j^{(t)}} + \epsilon}, & j \in \mathcal{A}_t, \\ w_j^{(t)}, & j \notin \mathcal{A}_t, \end{cases} \tag{8}$$

Here, $S_j^{(t)}$ acts as a parameter-wise adaptive scaling factor: a small $S_j^{(t)}$, indicating lower task relevance or stronger pretrained-weight saliency, yields a conservative update, while a large $S_j^{(t)}$ permits a larger update. This mechanism provides per-parameter control over update magnitudes throughout training, enabling a balance between knowledge preservation and task-specific adaptation.

4.2 Stability-Plasticity-Aware Parameter Scoring

$S_j^{(t)}$ governs sparse parameter selection, moment retention, and update strength, balancing task adaptation with conservative updates to salient pre-trained entries. It consists of two complementary components: the Knowledge-Retention Score (KRS), a pretrained-weight-based preservation prior, and the Task-Adaptation Score (TAS), which measures task-adaptation contribution.

Knowledge-Retention Score KRS does not directly measure semantic knowledge in individual parameters. Instead, it uses pretrained weight magnitude as a lightweight preservation prior. Weight magnitude is commonly viewed as a proxy for a parameter's contribution to model predictions (Han et al., 2015) and is widely used as a saliency signal in pruning and sparse adaptation (Frankle & Carbin, 2019; Sun et al., 2023; Liao et al., 2023). Thus, large-magnitude pretrained entries receive smaller safe-update factors, without requiring pretraining data or extra forward/backward passes. Since KRS depends only on the pretrained weights $\mathbf{W}_0$, it is computed once before training:

$$R_j = \left[\sigma\big(\text{z-score}(|\mathbf{W}_0|)\big)\right]_j, \tag{9}$$

where $\text{z-score}(\cdot)$ is computed over the flattened entries of the parameter tensor, $\sigma(\cdot)$ is the sigmoid function, and $[\cdot]_j$ extracts the j-th component.

Task-Adaptation Score TAS quantifies each parameter entry's contribution to downstream task adaptation, with more contributive entries assigned greater update freedom. We measure this contribution using the current gradient magnitude, a well-established indicator of parameter sensitivity and update utility (Fisher, 1922; Kirkpatrick et al., 2017; Lee et al., 2019; Sanh et al., 2020). Specifically, the TAS at step t is defined as

$$C_j^{(t)} = \left[\sigma\big(\text{z-score}(|\mathbf{g}^{(t)}|)\big)\right]_j. \tag{10}$$

Here, $\text{z-score}(\cdot)$ is computed per tensor over the current gradient magnitudes. A larger $C_j^{(t)}$ indicates that entry j is more task-relevant within the same tensor and warrants a larger update. Unlike static, one-shot pruning scores, TAS is updated online with the evolving gradient signal throughout fine-tuning.

Safe-Update Score We define the Safe-Update Score (SUS) to combine online task-adaptation evidence with the KRS stability prior. Since a desirable update should be both task-useful and safe to modify, SUS follows a conjunctive design rather than an additive one. Specifically, we convert KRS into an update-safety factor $(1 - R_j)$ and define

$$S_j^{(t)} = C_j^{(t)}\big(1 - R_j\big), \tag{11}$$

where $C_j^{(t)}$ promotes task-relevant entries, and $(1 - R_j)$ suppresses entries with stronger pretrained-weight saliency. This multiplicative form prevents one factor from compensating for the other: entries with low task relevance or high retention saliency receive low SUS. Consequently, SUS assigns high scores only to entries that are both important for downstream adaptation and relatively safe to modify. In SPSO, SUS serves as a unified control signal for parameter sampling, moment retention, and update scaling, jointly determining update locations, retained optimizer states, and update strengths.

Method	Fine-tune on IconQA									
	GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
Zero-shot	65.29	78.68	63.12	76.72	86.09	81.34	75.21	10.20	42.70	17.96
Fine-tuning	0.09	0.15	0.02	0.51	80.62	76.42	26.30	**96.34**	61.32	41.32
L2-Reg	12.45	21.62	10.29	18.39	63.57	42.94	28.21	92.63	60.42	43.25
DARE Yu et al. (2024)	58.11	59.55	54.43	63.37	81.61	77.47	65.76	93.71	79.73	77.28
Grafting Panigrahi et al. (2023)	57.95	57.98	56.15	61.32	82.14	78.50	65.67	93.47	79.57	77.14
RM Xu & Zhang (2024)	57.17	68.37	55.26	63.78	81.28	77.78	67.27	94.37	80.82	78.55
FISH Mask Sung et al. (2021)	53.15	62.11	52.37	61.85	80.98	75.46	64.32	95.43	79.88	76.85
PaFi Liao et al. (2023)	57.58	72.05	54.99	63.64	83.12	77.50	68.15	94.72	81.43	79.27
Model Tailor Zhu et al. (2024)	58.24	69.25	56.75	65.32	83.59	76.49	68.27	93.32	80.80	78.86
AdaDARE Xie et al. (2025)	57.20	73.44	57.92	68.87	81.61	77.74	69.46	93.89	81.68	79.85
SPIDER Huang et al. (2025b)	56.97	71.64	57.07	72.85	82.35	76.94	69.64	94.85	82.24	80.31
SPSO	**64.82**	**78.48**	**62.15**	**76.53**	**85.69**	**79.30**	**74.50**	96.01	**85.25**	**83.90**
Method	Fine-tune on COCO-Caption									
	GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
Zero-shot	65.29	78.68	63.12	76.72	86.09	81.34	75.21	31.70	53.45	44.60
Fine-tuning	51.45	65.92	43.98	70.62	73.26	71.92	62.86	113.75	88.30	80.97
L2-Reg	54.90	61.99	53.46	58.13	64.85	63.43	59.46	109.83	84.65	77.15
DARE Yu et al. (2024)	59.22	70.33	56.39	70.98	77.15	73.27	67.89	114.06	90.98	85.12
Grafting Panigrahi et al. (2023)	60.47	69.09	58.91	67.49	77.61	75.09	68.11	114.52	91.32	85.42
RM Xu & Zhang (2024)	58.58	68.04	56.20	67.97	81.01	78.12	68.32	116.53	92.43	86.14
FISH Mask Sung et al. (2021)	58.37	65.18	55.94	65.13	81.04	78.38	67.34	122.38	94.86	86.88
PaFi Liao et al. (2023)	61.25	71.72	58.10	70.18	81.66	79.99	70.48	120.32	95.40	88.89
Model Tailor Zhu et al. (2024)	61.79	70.45	60.49	70.22	81.99	79.10	70.67	118.64	94.66	88.58
AdaDARE Xie et al. (2025)	60.45	72.37	59.46	72.15	82.32	80.31	71.18	119.47	95.32	89.21
SPIDER Huang et al. (2025b)	60.11	73.53	57.66	71.52	83.49	80.61	71.15	121.35	96.25	89.71
SPSO	**64.51**	**78.06**	**61.11**	**76.77**	**84.56**	**81.81**	**74.47**	**126.81**	**100.64**	**93.83**

Table 1: Results on NVILA-Lite-8B with a sparsity ratio $\rho = 90\%$. The optimal and sub-optimal results are denoted by boldface and underlining, respectively.

5 Experiments

Baselines and Datasets We conduct experiments on two representative MLLMs, LLaVA-1.5 Liu et al. (2024a) and NVILA-Lite-8B Liu et al. (2025). These models are fine-tuned on downstream Visual Question Answering (VQA) and image-captioning tasks. Specifically, following Huang et al. (2025a); Liang et al. (2025), we use IconQA for VQA and COCO-Caption for image captioning, neither of which was encountered during the pre-training phase of these models. To assess general knowledge retention, we follow Zhu et al. (2024) and evaluate zero-shot performance on pre-trained tasks, including GQA Hudson & Manning (2019), TextVQA Singh et al. (2019), OCRVQA Mishra et al. (2019), OKVQA Marino et al. (2019), MM-Bench Liu et al. (2024b), and MM-Bench-CN Zhang et al. (2023). Additional experiments on Molmo2-8B are reported in Appendix C, while results on ImageNet-R and Flickr30K are provided in Appendix B.

Implementation Details. Following Zhu et al. (2024); Huang et al. (2025a), we freeze the vision encoder and fine-tune the last 12 language-decoder layers and the multimodal projector at learning rates of 2×10^{-5} and 2×10^{-6}, respectively. The number of training epochs is set to 5, with a global batch size of 128. Following Zhou et al. Zhou et al. (2024), we randomly sample 10K training instances from each downstream dataset. We set $\rho = 90\%$ for all sparse fine-tuning methods. For evaluation, $A_{\mathcal{P}}$ denotes the average performance across the pre-trained capability benchmarks, while $A_{\mathcal{T}}$ denotes performance on the target fine-tuning task. We further report their arithmetic mean (Avg) and harmonic mean (H-score). See Appendices G for details.

5.1 Comparison to State-of-the-Arts

Tables 1 and 2 show two consistent findings across both MLLMs and downstream tasks. **(1)** Standard fine-tuning improves target-task performance but degrades pre-trained capabilities. On IconQA, for example, $A_{\mathcal{P}}$ drops from 75.21% to 26.30% for NVILA-Lite-8B and from 61.18 to 31.49 for LLaVA-1.5, with a similar but milder trend on COCO-Caption. **(2)** Existing mitigation methods alleviate forgetting but still exhibit a trade-off between capability retention and downstream adaptation. Both post-merging methods (e.g., Model Tailor and AdaDARE) and sparse fine-tuning methods (e.g., PaFi and SPIDER) improve retention, but remain limited in either retention or adaptation. In comparison, SPSO achieves the highest $A_{\mathcal{P}}$, Avg, and H-score across all four model–task settings, while remaining competitive with standard fine-tuning on IconQA and outperforming it on COCO-Caption. These

Method	Fine-tune on IconQA									
	GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
Zero-shot	61.97	58.23	57.96	66.20	64.60	58.10	61.18	23.18	42.18	33.62
Fine-tuning	17.51	20.10	15.53	25.41	58.18	52.21	31.49	**59.02**	45.26	41.07
L2-Reg	26.87	24.86	24.01	28.74	34.06	33.68	28.70	53.36	41.03	37.33
DARE Yu et al. (2024)	50.36	48.32	51.92	53.46	59.75	57.33	53.52	53.87	53.70	53.70
Grafting Panigrahi et al. (2023)	50.00	48.14	50.93	53.45	57.06	54.34	52.32	52.24	52.28	52.28
RM Xu & Zhang (2024)	52.16	51.46	52.14	57.18	55.69	53.55	53.70	55.16	54.43	54.42
FISH Mask Sung et al. (2021)	50.69	48.67	49.59	56.06	56.49	53.38	52.48	58.46	55.47	55.31
PaFi Liao et al. (2023)	57.93	53.35	53.75	59.48	58.00	55.79	56.38	55.32	55.85	55.85
Model Tailor Zhu et al. (2024)	54.91	52.28	52.87	58.62	58.61	55.89	55.53	55.81	55.67	55.67
AdaDARE Xie et al. (2025)	57.24	53.49	53.39	57.32	58.15	55.75	55.89	56.60	56.25	56.24
SPIDER Huang et al. (2025b)	56.53	53.87	53.33	56.87	59.39	54.38	55.73	57.12	56.42	56.42
SPSO	**61.54**	**58.16**	**58.38**	**65.34**	**59.87**	**57.89**	**60.20**	58.92	**59.56**	**59.55**
Method	Fine-tune on COCO-Caption									
	GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
Zero-shot	61.97	58.23	57.96	66.20	64.60	58.10	61.18	40.03	50.60	48.39
Fine-tuning	54.54	49.56	43.59	55.43	58.07	51.91	52.18	102.40	77.29	69.14
L2-Reg	45.62	42.32	42.34	46.11	51.89	50.72	46.50	93.32	69.91	62.07
DARE Yu et al. (2024)	50.98	51.99	50.61	54.87	61.01	53.26	53.79	98.76	76.27	69.64
Grafting Panigrahi et al. (2023)	51.36	50.23	50.54	56.48	59.36	54.95	53.82	109.13	81.48	72.09
RM Xu & Zhang (2024)	54.79	52.65	54.29	58.31	61.87	53.79	55.95	112.62	84.29	74.76
FISH Mask Sung et al. (2021)	51.52	49.77	51.07	54.56	60.80	53.94	53.61	119.12	86.37	73.94
PaFi Liao et al. (2023)	56.68	53.06	52.72	59.70	62.03	53.97	56.36	116.57	86.47	75.98
Model Tailor Zhu et al. (2024)	54.90	54.08	53.83	58.19	61.87	54.39	56.21	114.32	85.27	75.36
AdaDARE Xie et al. (2025)	56.21	52.84	54.44	57.24	63.66	56.35	56.79	116.31	86.55	76.32
SPIDER Huang et al. (2025b)	55.89	54.28	55.30	57.28	63.85	**57.56**	57.36	118.52	87.94	77.31
SPSO	**61.41**	**57.29**	**57.27**	**64.55**	**64.97**	57.32	**60.47**	**124.42**	**92.44**	**81.38**

Table 2: Results on LLaVA-1.5-7B with a sparsity ratio $\rho = 90\%$. The optimal and sub-optimal results are denoted by boldface and underlining, respectively.

Fine-tune on NVILA-Lite-8B				
Method	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
w/o DS	72.10	93.84	82.97	81.55
w/o US	69.54	95.40	82.47	80.44
w/o MP	73.65	94.91	84.28	82.94
SPSO	**74.50**	**96.01**	**85.25**	**83.90**
Fine-tune on LLaVA-1.5-7B				
Method	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
w/o DS	56.83	56.73	56.78	56.78
w/o US	54.96	58.24	56.60	56.55
w/o MP	58.72	58.12	58.42	58.42
SPSO	**60.20**	**58.92**	**59.56**	**59.55**

Table 3: Component ablation. DS, US, and MP denote dynamic selection, update scaling, and momentum pruning.

Fine-tune on NVILA-Lite-8B				
Score	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
RS	68.94	93.10	81.02	79.22
KRS	73.82	92.74	83.28	82.21
TAS	70.25	95.53	82.89	80.96
SUS	**74.50**	**96.01**	**85.25**	**83.90**
Fine-tune on LLaVA-1.5-7B				
Score	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
RS	53.72	55.96	54.84	54.82
KRS	58.86	55.18	57.02	56.96
TAS	54.72	58.36	56.54	56.48
SUS	**60.20**	**58.92**	**59.56**	**59.55**

Table 4: Scoring ablation. RS, KRS, and TAS denote random scoring, knowledge-retention score, and task-adaptation score.

results show that SPSO provides a more favorable balance between pre-trained capability retention and downstream adaptation.

5.2 Ablation Studies

We present ablation studies and diagnostic analyses for SPSO. Unless otherwise specified, all ablations are conducted by fine-tuning on IconQA with the default sparsity ratio $\rho = 90\%$, corresponding to a 10% active-parameter budget.

Key Component Analysis. Table 3 examines the three core components of SPSO: dynamic selection (DS), update scaling (US), and momentum pruning (MP), which respectively control where to update, how much to update, and what optimization information to retain. Removing DS degrades both pre-trained capability retention and target-task adaptation on NVILA-Lite-8B and LLaVA-1.5-7B, showing that dynamically reallocating the sparse update set is beneficial for using the limited update budget effectively. Removing US leads to the most pronounced reduction in $A_{\mathcal{P}}$ while only mildly affecting $A_{\mathcal{T}}$, supporting its role in suppressing overly aggressive updates to parameters that are less safe to modify. Removing MP causes a smaller but consistent degradation on both models, indicating that selectively retaining informative optimizer states provides complementary benefits beyond parameter selection and update scaling. Together, these results show that the three components play distinct but complementary roles in balancing pre-trained capability preservation and downstream adaptation.

Method	Upd./step (M)	Cum. (M)	Sparsity (%)	Opt. State (GiB)	Peak Mem. (GiB)	H_{IconQA}	$H_{COCO\text{-}Cap}$
Adam	2449.58	2449.58	13.21	18.25	25.84	41.07	69.14
RM	244.96	244.96	90.14	9.13	21.50	54.42	74.76
FISH Mask	244.96	244.96	2.14	9.13	21.50	55.31	73.94
PaFi	244.96	244.96	90.09	9.13	21.50	55.85	75.98
SPIDER	1324.62	1405.80	73.17	182.51	70.91	<u>56.42</u>	<u>77.31</u>
SPSO	263.93	836.08	83.00	11.47	23.75	**59.55**	**81.38**

Table 5: Training efficiency on LLaVA-1.5-7B fine-tuned on IconQA. Upd./step and Cum. denote per-step and cumulative updated parameters. Sparsity is $S_\tau = \frac{1}{N}\sum_i \mathbf{1}[|\theta_i - \theta_i^0| \le \tau]$ with $\tau = 10^{-5}$; higher values indicate less parameter drift. Opt. State and Peak Mem. denote optimizer-state and peak GPU memory. Static-mask methods use a 10% update budget; FISH statistics are reported after calibration.

Parameter Scoring Strategies. Table 4 examines how different scoring signals steer the optimization behavior of SPSO. Random scoring (RS) removes informative guidance for identifying suitable update directions, resulting in the weakest overall stability–plasticity balance. Using only the KRS-derived safety signal favors parameters that are less likely to interfere with pre-trained knowledge, leading to stronger capability retention but more limited downstream adaptation. In contrast, TAS prioritizes parameters that are more responsive to the target task, improving adaptation while providing weaker protection against forgetting. These two variants therefore exhibit complementary optimization behaviors, reflecting the distinct roles of knowledge preservation and task relevance. By combining them, SUS identifies parameters that are both task-relevant and relatively safe to modify, allowing SPSO to allocate its sparse update budget more effectively. Consequently, the complete SUS formulation achieves a better balance between pre-trained capability preservation and downstream adaptation than either signal used in isolation.

Efficiency Analysis. We further evaluate SPSO in terms of update sparsity and memory usage. As shown in Table 5, SPSO updates only 263.93M of 2449.58M trainable parameters per step, while cumulatively covering 836.08M as the active set evolves. Unlike static sparse methods confined to a fixed subset, SPSO maintains a compact per-step update set while exploring a broader parameter space; meanwhile, its final update sparsity remains 83.00%, indicating that most parameters remain close to their pre-trained values. This sparse design also reduces memory usage. Adam maintains optimizer states for all trainable parameters, whereas SPSO maintains only a sparse state budget, reducing optimizer-state memory from 18.25 GiB to 11.47 GiB. The contrast with SPIDER is more pronounced. SPIDER reconstructs its dynamic mask from information retained across steps: it stores the pre-trained parameters, accumulates downstream-gradient statistics, and maintains dense importance and masking signals. These quantities supplement the standard optimizer states, resulting in 182.51 GiB of optimizer-state memory. SPSO instead performs dynamic parameter selection without a full pre-trained parameter copy or accumulated dense gradient statistics, while maintaining sparse optimizer states. As a result, its peak GPU memory is 23.75 GiB, versus 25.84 GiB for Adam and 70.91 GiB for SPIDER. Overall, SPSO keeps each step sparse while allowing the effective update space to evolve, balancing dynamic parameter exploration and memory efficiency.

6 Conclusion

In this paper, we present the *Stability-Plasticity-Aware Sparse Optimizer (SPSO)* for mitigating catastrophic forgetting in MLLM fine-tuning. We show that step-wise dynamic masking does not necessarily yield effective sparse optimization, as it can incur considerable memory overhead while committing to most of its eventual update region early in training. To address these limitations, SPSO formulates sparse fine-tuning from an optimizer-level perspective and introduces the *Safe-Update Score (SUS)*, which combines online task relevance with a lightweight pre-trained preservation prior. Guided by SUS, SPSO dynamically reallocates a fixed-size active set, scales parameter-wise updates, and selectively retains sparse optimizer states, jointly controlling *where to update*, *how much to update*, and *what to retain*. Experiments across multiple MLLMs, downstream tasks, and extended settings show that SPSO improves the balance between pre-trained capability retention and downstream adaptation while maintaining sparse updates and reducing optimization memory.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, presentation refinement and LaTeX formatting. They were not used to generate experimental data or results, execute experiments, or independently establish scientific conclusions. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final content of the submission.

ETHICS STATEMENT

This work studies optimizer-level sparse fine-tuning for multimodal large language models using existing models and benchmark datasets. We do not conduct human-subject studies or collect private user data. The primary potential risks are inherited from the underlying MLLMs and datasets, including biases and hallucinations. Any released artifacts will follow the applicable licenses and access conditions of the original models and datasets.

REPRODUCIBILITY STATEMENT

The main paper specifies the optimization objective, SPSO update rule, model families, trainable scope, datasets, metrics, and key hyperparameters. The appendix further provides dataset details, training settings, the detailed SPSO algorithm, ablation studies, sensitivity analyses, and continual instruction-tuning setup. Code and experimental configurations will be released upon acceptance.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019.

Cheng Chen, Junchen Zhu, Xu Luo, Heng T Shen, Jingkuan Song, and Lianli Gao. Coin: A benchmark of continual instruction tuning for multimodel large language models. *NeurIPS*, 2024a.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024b.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024c.

Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Mohammadreza Salehi, Rohun Tripathi, Sangho Lee, Zhongzheng Ren, Chris Dongjoo Kim, Yinuo Yang, et al. Molmo2: Open weights and data for vision-language models with video understanding and grounding. In *CVPR*, 2026.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. In *NeurIPS*, 2023.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *CVPR*, 2025.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

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

Ronald A Fisher. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 1922.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017.

Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *CVPR*, 2018.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015.

Wenke Huang, Jian Liang, Zekun Shi, Didi Zhu, Guancheng Wan, He Li, Bo Du, Dacheng Tao, and Mang Ye. Learn from downstream and be yourself in multimodal large language model fine-tuning. In *ICML*, 2025a.

Wenke Huang, Jian Liang, Zekun Shi, Didi Zhu, Guancheng Wan, He Li, Bo Du, Dacheng Tao, and Mang Ye. Learn from downstream and be yourself in multimodal large language models fine-tuning. In *ICML*, pp. 25486–25499, 2025b.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, 2019.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

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

Wouter Kool, Herke van Hoof, and Max Welling. Stochastic beams and where to find them: The Gumbel-top-k trick for sampling sequences without replacement. In *ICML*, 2019.

Junlin Lee, Yequan Wang, Jing Li, and Min Zhang. Multimodal reasoning with multimodal knowledge graph. In *ACL*, 2024.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023.

Jian Liang, Wenke Huang, Guancheng Wan, Qu Yang, and Mang Ye. Lorasculpt: Sculpting lora for harmonizing general and specialized knowledge in multimodal large language models. In *CVPR*, 2025.

Baohao Liao, Yan Meng, and Christof Monz. Parameter-efficient fine-tuning without introducing new latency. In *ACL*, 2023.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

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

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

Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. *arXiv preprint arXiv:1802.02950*, 2018.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *ECCV*, 2024b.

Zhijian Liu, Ligeng Zhu, Baifeng Shi, Zhuoyang Zhang, Yuming Lou, Shang Yang, Haocheng Xi, Shiyi Cao, Yuxian Gu, Dacheng Li, Xiuyu Li, Yunhao Fang, Yukang Chen, Cheng-Yu Hsieh, De-An Huang, An-Chieh Cheng, Vishwesh Nath, Jinyi Hu, Sifei Liu, Ranjay Krishna, Daguang Xu, Xiaolong Wang, Pavlo Molchanov, Jan Kautz, Hongxu Yin, Song Han, and Yao Lu. NVILA: Efficient frontier visual language models. In *CVPR*, 2025.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

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

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *NeurIPS*, 2022.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *CVPR*, 2016.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *CVPR*, 2019.

Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, et al. Chartqapro: A more diverse and challenging benchmark for chart question answering. In *ACL*, 2025.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 1995.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. Elsevier, 1989.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In *ICDAR*, 2019.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. In *ICML*, 2023.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 2019.

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

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *ICLR*, 2023.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *NeurIPS*, 2020.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *ICML*, 2018.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *CVPR*, 2019.

Linzhuang Sun, Hao Liang, Jingxuan Wei, Bihui Yu, Tianpeng Li, Fan Yang, Zenan Zhou, and Wentao Zhang. Mm-verify: Enhancing multimodal reasoning with chain-of-thought verification. In *ACL*, 2025.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *ICLR*, 2023.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. In *NeurIPS*, 2021.

Jingqun Tang, Qi Liu, Yongjie Ye, Jinghui Lu, Shu Wei, An-Lan Wang, Chunhui Lin, Hao Feng, Zhen Zhao, Yanjie Wang, et al. Mtvqa: Benchmarking multilingual text-centric visual question answering. In *ACL*, 2025.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE TPAMI*, 2024.

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, et al. Rehearsal-free continual language learning via efficient parameter isolation. In *ACL*, 2023.

Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. NExT-GPT: Any-to-any multimodal LLM. In *ICML*, 2024.

Jingyi Xie, Jintao Yang, Zhunchen Luo, Yunbo Cao, Qiang Gao, Mengyuan Zhang, and Wenpeng Hu. Adadare-gamma: Balancing stability and plasticity in multi-modal llms through efficient adaptation. In *CVPR*, 2025.

Jing Xu and Jingzhao Zhang. Random masking finds winning tickets for parameter efficient fine-tuning. In *ICML*, 2024.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *ICML*, 2024.

Yahan Yu, Duzhen Zhang, Yong Ren, Xuanle Zhao, Xiuyi Chen, and Chenhui Chu. Progressive LoRA for multimodal continual instruction tuning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2779–2796. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.143. URL https://aclanthology.org/2025.findings-acl.143/.

Fanhu Zeng, Fei Zhu, Haiyang Guo, Xu-Yao Zhang, and Cheng-Lin Liu. Modalprompt: Towards efficient multimodal continual instruction tuning with dual-modality guided prompt. In *EMNLP*, 2025.

Yuexiang Zhai, Shengbang Tong, Xiao Li, Mu Cai, Qing Qu, Yong Jae Lee, and Yi Ma. Investigating the catastrophic forgetting in multimodal large language model fine-tuning. In *CPAL*, 2024.

Pan Zhang, Xiaoyi Dong, Bin Wang, Yuhang Cao, Chao Xu, Linke Ouyang, Zhiyuan Zhao, Haodong Duan, Songyang Zhang, Shuangrui Ding, et al. Internlm-xcomposer: A vision-language large model for advanced text-image comprehension and composition. *arXiv preprint arXiv:2309.15112*, 2023.

Xiongtao Zhou, Jie He, Yuhua Ke, Guangyao Zhu, Víctor Gutiérrez-Basulto, and Jeff Pan. An empirical study on parameter-efficient fine-tuning for multimodal large language models. In *ACL*, 2024.

Didi Zhu, Zhongyi Sun, Zexi Li, Tao Shen, Ke Yan, Shouhong Ding, Kun Kuang, and Chao Wu. Model tailor: Mitigating catastrophic forgetting in multi-modal large language models. In *ICML*, 2024.

APPENDIX CONTENTS

A More Discussuion with SPIDER **16**

B Extension to Other Datasets **19**

C Extension to Other MLLMs **21**

D Extension to LoRA-Based Methods **22**

E Extension to Continual Instruction Tuning **23**

F Details of Datasets **24**

G Evaluation Metrics **25**

H Algorithm of SPSO **26**

I Additional Ablation Studies **26**

J Limitations **28**

A MORE DISCUSSUION WITH SPIDER

A.1 REVISITING SPIDER

SPIDER Huang et al. (2025b) is a dynamic masking method that balances downstream adaptation and pre-trained capability preservation by comparing parameter importance under the pre-trained model and the downstream task. For consistency with the notation used in our formulation, let $\mathbf{W}_0$ denote the frozen pre-trained weights, $w_j^{(t)}$ the j-th trainable parameter at optimization step t, and $g_j^{(t)}$ its corresponding downstream gradient.

Pre-trained and Downstream Importance. SPIDER first estimates the importance of each parameter to the pre-trained model using its pre-trained weight magnitude. Rewriting its formulation under our notation, we denote this score as

$$R_j^{\mathrm{SP}} = \left[\sigma\left(\mathrm{zscore}\left(|\mathbf{W}_0|\right)\right)\right]_j, \tag{12}$$

where $\mathrm{zscore}(\cdot)$ is computed within each parameter tensor and $\sigma(\cdot)$ denotes the sigmoid function. Since $\mathbf{W}_0$ is fixed, R_j^{SP} remains unchanged throughout fine-tuning.

To estimate downstream-task importance, SPIDER uses accumulated gradient magnitudes. Let $\bar{g}_j^{(t)}$ denote the accumulated gradient statistic of parameter j up to step t. Its downstream importance can then be expressed as

$$C_j^{\mathrm{SP},(t)} = \left[\sigma\left(\mathrm{zscore}\left(|\bar{\mathbf{g}}^{(t)}|\right)\right)\right]_j. \tag{13}$$

Unlike R_j^{SP}, this score evolves during fine-tuning as downstream gradients are accumulated.

Dynamic Mask Construction. At every optimization step, SPIDER compares the two importance scores element-wise. Parameter j is considered eligible for downstream modification when

$$C_j^{\mathrm{SP},(t)} > R_j^{\mathrm{SP}}. \tag{14}$$

For selected parameters, SPIDER further assigns a soft mask coefficient according to the relative strength of the downstream importance:

$$\widetilde{M}_j^{(t)} = \begin{cases} \dfrac{C_j^{\mathrm{SP},(t)}}{C_j^{\mathrm{SP},(t)} + R_j^{\mathrm{SP}}}, & C_j^{\mathrm{SP},(t)} > R_j^{\mathrm{SP}}, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

The non-zero coefficients are subsequently normalized by their mean and clipped to obtain the final mask:

$$M_j^{(t)} = \begin{cases} \min\left(1, \dfrac{\widetilde{M}_j^{(t)}}{\mathrm{Mean}\left(\widetilde{\mathbf{M}}^{(t)}[\widetilde{\mathbf{M}}^{(t)} > 0]\right)}\right), & \widetilde{M}_j^{(t)} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

After the optimizer produces the tentative fine-tuned parameter $w_j^{(t+1)}$, SPIDER interpolates it with its pre-trained counterpart $w_{0,j}$ according to the mask:

$$w_j^{(t+1)} \leftarrow M_j^{(t)} w_j^{(t+1)} + \left(1 - M_j^{(t)}\right) w_{0,j}. \tag{17}$$

Thus, parameters with $M_j^{(t)} = 0$ are restored to their pre-trained values, while parameters satisfying Eq. equation 14 retain different fractions of their downstream updates.

Overall, SPIDER dynamically reconstructs its mask by repeatedly comparing a fixed pre-trained importance signal R_j^{SP} with an evolving accumulated-gradient signal $C_j^{\mathrm{SP},(t)}$. Importantly, the active set is determined by the element-wise inequality in Eq. equation 14, rather than by an explicit fixed-size update budget. Therefore, recomputing the mask at every step does not by itself guarantee that the effective update set will continue to change substantially throughout optimization.

A.2 Differences between SPSO and SPIDER

Although both SPIDER and SPSO dynamically determine which parameters participate in fine-tuning, they differ substantially in how sparsity is formulated and incorporated into optimization. SPIDER primarily treats sparsity as a *mask-construction problem*: at every optimization step, it recomputes a parameter-wise mask by comparing a pre-trained importance signal derived from the frozen initial weights with a downstream importance signal derived from accumulated gradients. The resulting mask directly determines the parameters to be updated and their corresponding mask weights. In contrast, SPSO formulates sparsity as an *optimizer-level dynamic sparse optimization problem*. Rather than reconstructing a dense mask at every step, SPSO periodically selects an active parameter subset under a fixed update budget and integrates this selection into parameter updates, update scaling, and optimizer-state maintenance.

Table A1 summarizes the main differences between the two approaches. First, their parameter-selection objectives are different. SPIDER determines parameter-wise update behavior through the relative relationship between pre-trained weight importance and accumulated downstream gradient importance, whereas SPSO uses the Safe-Update Score (SUS) to identify parameters that provide useful downstream plasticity while controlling deviation from the pre-trained model. Second, SPIDER recomputes its mask at every optimization step, while SPSO maintains an active set for multiple optimization steps before reconstructing it, allowing parameter selection to evolve at a different timescale from individual gradient updates. Third, SPIDER does not explicitly impose a fixed active-parameter budget; by contrast, SPSO directly controls the number of active parameters through a predefined sparsity budget. Fourth, the two approaches handle optimizer states differently. SPIDER applies masking on top of the conventional optimizer, while SPSO couples parameter activation with sparse moment-state maintenance, such that optimizer states are retained only for parameters participating in sparse optimization. Finally, SPSO further applies parameter-wise update scaling within the selected active set, separating the decisions of *where to update* and *how strongly to update*, whereas these two effects are jointly encoded by the mask values in SPIDER.

Table A1: Comparison between SPIDER and SPSO from the perspective of sparse optimization.

Aspect	**SPIDER**	**SPSO**
Formulation	Dynamic mask construction	Optimizer-level dynamic sparse optimization
Selection signal	Pre-trained weight importance and accumulated downstream gradient importance	Safe-Update Score combining downstream relevance and a preservation prior
Mask / active-set update	Recomputed at every optimization step	Active set reconstructed periodically
Sparsity control	No explicit fixed active-parameter budget	Explicit fixed sparsity budget
Parameter dynamics	Selection is strongly influenced by persistent pre-trained weights and cumulative gradient statistics	Active parameters are reselected according to the evolving safe-update distribution
Update magnitude	Encoded together with parameter selection through mask values	Separates active-parameter selection from parameter-wise update scaling
Optimizer state	Conventional dense optimizer states are maintained independently of mask construction	Optimizer states are coupled with the active set and maintained sparsely
Optimization scope	Primarily determines which parameter updates are retained or suppressed	Jointly controls where to update, how much to update, and what optimization history to retain

A.3 Details of the Introductory Diagnostic Experiment

To better understand the optimization behavior of SPIDER and motivate the design of SPSO, we conduct a diagnostic experiment using LLaVA-1.5 fine-tuned on IconQA. Unless otherwise specified, we follow the same training configuration as in the main experiments, where the vision encoder is frozen and the multimodal projector together with the last 12 language-decoder layers are optimized.

We compare standard Adam fine-tuning, SPIDER, and SPSO under the same trainable parameter scope and record their optimization statistics throughout training.

For the memory analysis, we additionally vary the number of trainable language-decoder layers while keeping the remaining training configuration unchanged. This allows us to examine how the memory consumption of different methods scales as the number of trainable parameters increases. For the optimization-dynamics analysis, we use the default 12-layer setting and record the sparse update behavior at every optimization step.

Let $\theta^{(0)}$ denote the pre-trained parameters, $\theta^{(t)}$ the parameters after optimization step t, and N the total number of trainable parameter coordinates considered in the analysis. We use four complementary measurements.

Memory Consumption. We first measure the peak training memory as the number of trainable decoder layers increases. This experiment is intended to quantify the additional memory introduced by the sparse optimization mechanism itself, rather than its predictive performance. Standard Adam mainly maintains the trainable parameters together with the first- and second-order moment states. SPIDER additionally requires persistent statistics for constructing its parameter-wise mask, including the pre-trained parameter reference and accumulated downstream gradient information. Consequently, its memory consumption increases substantially as the trainable parameter space grows. In contrast, SPSO couples sparse parameter activation with sparse optimizer-state maintenance, so that the optimizer states need not be densely maintained for the entire trainable parameter space.

Cumulative Active-Set Coverage. To examine whether a dynamically reconstructed mask actually explores new parameters throughout training, we define the active parameter set at step t as

$$\mathcal{V}_t = \{i \mid \text{parameter } i \text{ is selected for update at step } t\}. \tag{18}$$

We then measure the cumulative union coverage

$$U(t) = \frac{\left|\bigcup_{k=1}^{t} \mathcal{V}_k\right|}{N}. \tag{19}$$

Here, $U(t)$ measures the fraction of trainable parameters that have been activated *at least once* up to step t. Importantly, this metric is different from the instantaneous active ratio $|\mathcal{V}_t|/N$. The latter only describes how many parameters are active at a particular step, whereas $U(t)$ measures how broadly the optimization process explores the parameter space over time.

A rapidly saturating $U(t)$ indicates that most parameters that will ever be selected are already encountered during the early stage of fine-tuning. Conversely, a continuously increasing $U(t)$ indicates that the optimization process continues to introduce previously inactive parameters into the update space. To further quantify early commitment, we can consider

$$R_{\text{early}} = \frac{U(1)}{U(T)}, \tag{20}$$

where T denotes the final optimization step. A value close to one means that the first-step active set already covers most of the parameters that will ever participate in optimization. In our IconQA experiment, the parameters activated by SPIDER at the first optimization step already account for approximately 94% of its final cumulative active set, revealing that most of its effective update support is established extremely early.

Pre-trained Weight Drift. Union coverage describes *where* updates occur, but does not indicate how far the parameters move from their pre-trained values. We therefore additionally measure the parameter drift from the initialization. Let $\mathcal{W}$ denote the collection of trainable parameter tensors. At step t, we compute the normalized weight drift as

$$D(t) = \sqrt{\frac{\sum_{W \in \mathcal{W}} \left\|W^{(t)} - W^{(0)}\right\|_F^2}{\sum_{W \in \mathcal{W}} \left\|W^{(0)}\right\|_F^2}}. \tag{21}$$

A larger $D(t)$ indicates a larger departure from the pre-trained model, while a smaller value indicates that the learned parameters remain closer to their initialization. This metric therefore characterizes the overall magnitude of parameter modification and provides a complementary view of preservation behavior beyond the binary notion of whether a parameter has been selected.

Effective Update Sparsity. Finally, we measure how many parameters remain effectively unchanged relative to their pre-trained values. Specifically, given a small threshold τ, we define

$$S_\tau(t) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(\left|\theta_i^{(t)} - \theta_i^{(0)}\right| \le \tau\right), \tag{22}$$

where we use $\tau = 10^{-5}$ in the main analysis. A larger $S_\tau(t)$ means that a larger fraction of parameters remain close to their pre-trained values and therefore corresponds to a sparser *effective* update. Conversely, a smaller value indicates that a broader portion of the parameter space has undergone non-negligible modification.

We emphasize that $S_\tau(t)$ is not simply the sparsity of the algorithmic mask. A parameter may be selected by a mask but receive only a very small accumulated update, while another parameter may repeatedly receive updates and move substantially away from its initialization. Measuring parameter displacement therefore captures the *effective update sparsity* induced by the complete optimization trajectory rather than merely the nominal sparsity of an instantaneous mask.

Taken together, these measurements characterize different aspects of sparse optimization. Memory consumption evaluates the computational cost of maintaining the sparse mechanism; $U(t)$ measures how broadly the optimizer explores the parameter space; $D(t)$ measures the overall magnitude of deviation from the pre-trained model; and $S_\tau(t)$ measures how much of the parameter space remains effectively unchanged. In particular, cumulative union coverage and effective update sparsity are complementary: the former records whether a parameter has ever entered the update set, whereas the latter determines whether that parameter has ultimately accumulated a non-negligible change. These diagnostics allow us to distinguish between a mask that is nominally recomputed at every step and an update space that genuinely evolves throughout optimization. “‘

B Extension to Other Datasets

Method	NVILA-Lite-8B: Fine-tune on ImageNet-R									
	GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_\mathcal{P}$	$A_\mathcal{T}$	Avg	H-score
Zero-shot	65.29	78.68	63.12	76.72	86.09	81.34	75.21	16.70	45.95	27.33
Fine-tuning	23.09	31.65	17.32	46.73	77.30	74.34	45.07	**93.67**	69.37	60.86
L2-Reg	32.62	38.99	31.63	39.85	44.35	43.20	38.44	80.09	59.27	51.95
DARE Yu et al. (2024)	55.17	69.43	34.61	75.84	78.21	75.12	64.73	86.72	75.72	74.13
Grafting Panigrahi et al. (2023)	52.37	60.18	43.69	75.91	78.66	75.89	64.45	86.41	75.43	73.83
RM Xu & Zhang (2024)	62.17	66.38	35.57	74.31	85.77	77.14	66.89	87.77	77.33	75.92
FISH Mask Sung et al. (2021)	58.47	70.16	19.83	66.97	84.96	79.65	63.34	89.21	76.28	74.08
PaFi Liao et al. (2023)	48.27	55.36	**64.91**	75.05	85.25	77.66	67.75	88.63	78.19	76.80
Model Tailor Zhu et al. (2024)	50.08	72.93	48.76	75.93	81.53	78.11	67.89	88.03	77.96	76.66
AdaDARE Xie et al. (2025)	**63.94**	58.07	50.11	74.02	**86.84**	79.14	68.69	88.88	78.78	77.49
SPIDER Huang et al. (2025b)	56.08	71.93	40.17	**76.12**	86.60	**81.06**	68.66	89.96	79.31	77.88
SPSO	62.28	**73.66**	58.57	75.96	85.94	80.29	**72.78**	91.93	**82.36**	**81.24**

Method	NVILA-Lite-8B: Fine-tune on Flickr30K									
	GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_\mathcal{P}$	$A_\mathcal{T}$	Avg	H-score
Zero-shot	65.29	78.68	63.12	76.72	86.09	81.34	75.21	30.20	52.70	43.09
Fine-tuning	50.03	61.79	47.30	67.14	75.04	75.81	62.85	82.89	72.87	71.49
L2-Reg	54.89	61.82	51.56	62.65	65.40	63.74	60.01	74.45	67.23	66.45
DARE Yu et al. (2024)	53.17	70.46	51.12	56.72	81.97	79.14	65.43	83.32	74.38	73.30
Grafting Panigrahi et al. (2023)	55.27	51.13	52.09	72.23	**82.34**	79.04	65.35	84.52	74.94	73.71
RM Xu & Zhang (2024)	52.11	68.93	50.47	72.15	81.08	78.82	67.26	88.13	77.69	76.29
FISH Mask Sung et al. (2021)	55.83	70.37	52.14	46.97	81.19	78.82	64.22	90.02	77.12	74.96
PaFi Liao et al. (2023)	52.83	72.36	51.24	75.00	81.17	78.82	68.57	88.38	78.47	77.22
Model Tailor Zhu et al. (2024)	54.47	67.18	50.42	73.97	81.16	78.82	67.67	87.85	77.76	76.45
AdaDARE Xie et al. (2025)	61.47	67.09	50.57	72.69	81.85	79.37	68.84	88.12	78.48	77.30
SPIDER Huang et al. (2025b)	54.83	68.47	50.86	73.95	82.09	**79.66**	68.31	88.75	78.53	77.20
SPSO	**64.65**	**78.37**	**61.33**	**76.93**	81.19	78.75	**73.54**	**92.07**	**82.80**	**81.77**

Table A2: Results on NVILA-Lite-8B for ImageNet-R and Flickr30K with a sparsity ratio $\rho = 90\%$. The optimal and sub-optimal results are denoted by boldface and underlining, respectively.

In addition to the main results on IconQA and COCO-Caption, we further evaluate SPSO on two additional target tasks, ImageNet-R and Flickr30K, covering image classification and image captioning, respectively. As shown in Tables A2 and A3, SPSO consistently improves the stability–plasticity balance across both NVILA-Lite-8B and LLaVA-1.5-7B. On NVILA-Lite-8B, SPSO improves the H-score over standard fine-tuning by 20.38 points on ImageNet-R and 10.28 points on Flickr30K, while also outperforming the strongest competing baselines. Similar gains are observed

Method	LLaVA-1.5-7B: Fine-tune on ImageNet-R									
	GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
Zero-shot	61.97	58.23	57.96	66.20	64.60	58.10	61.18	–	61.18	61.18
Fine-tuning	28.51	25.49	9.70	27.40	50.39	48.21	31.62	**89.37**	60.49	46.71
L2-Reg	25.01	24.40	23.32	27.07	32.71	31.59	27.35	81.47	54.41	40.95
DARE Yu et al. (2024)	49.14	45.63	47.42	54.45	59.69	**58.71**	52.51	84.44	68.47	64.75
Grafting Panigrahi et al. (2023)	51.32	48.77	48.47	52.96	57.89	54.41	52.30	84.13	68.22	64.50
RM Xu & Zhang (2024)	53.29	51.39	50.29	58.28	61.33	53.38	54.66	87.05	70.86	67.15
FISH Mask Sung et al. (2021)	51.19	49.97	47.61	51.68	58.33	52.34	51.85	89.12	70.49	65.56
PaFi Liao et al. (2023)	54.08	52.12	50.62	59.02	62.42	53.24	55.25	87.73	71.49	67.80
Model Tailor Zhu et al. (2024)	50.25	50.26	47.05	56.86	62.09	53.65	53.36	87.14	70.25	66.19
AdaDARE Xie et al. (2025)	52.48	52.71	48.79	56.64	64.23	54.51	54.89	87.76	71.33	67.54
SPIDER Huang et al. (2025b)	51.47	51.68	48.67	57.73	63.50	56.41	54.91	88.02	71.47	67.63
SPSO	**58.32**	**57.02**	**53.46**	**65.00**	**64.69**	56.36	**59.14**	88.27	**73.71**	**70.83**
Method	LLaVA-1.5-7B: Fine-tune on Flickr30K									
	GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
Zero-shot	61.97	58.23	57.96	66.20	64.60	58.10	61.18	–	61.18	61.18
Fine-tuning	53.55	50.01	42.90	52.14	57.71	50.07	51.06	68.54	59.80	58.52
L2-Reg	43.18	40.12	40.33	44.38	49.61	47.70	44.22	63.47	53.84	52.12
DARE Yu et al. (2024)	51.42	49.65	50.48	54.80	62.03	**60.72**	54.85	76.32	65.58	63.83
Grafting Panigrahi et al. (2023)	53.72	51.18	50.68	54.08	58.94	54.78	53.90	73.41	63.65	62.16
RM Xu & Zhang (2024)	56.63	53.76	55.51	58.90	59.82	53.36	56.33	77.37	66.85	65.19
FISH Mask Sung et al. (2021)	53.16	50.80	49.49	53.51	59.47	53.19	53.27	82.88	68.08	64.86
PaFi Liao et al. (2023)	58.14	53.45	54.89	59.97	59.76	55.13	56.89	77.85	67.37	65.74
Model Tailor Zhu et al. (2024)	55.60	53.97	54.42	59.43	60.27	54.53	56.37	78.42	67.39	65.59
AdaDARE Xie et al. (2025)	57.96	55.57	55.08	60.78	63.75	55.70	58.14	79.35	68.75	67.11
SPIDER Huang et al. (2025b)	55.48	53.85	53.36	57.17	**63.96**	58.18	57.00	80.02	68.51	66.58
SPSO	**61.73**	**57.48**	**58.56**	**64.38**	63.11	57.59	**60.48**	**85.46**	**72.97**	**70.83**

Table A3: Results on LLaVA-1.5-7B for ImageNet-R and Flickr30K with a sparsity ratio $\rho = 90\%$. The optimal and sub-optimal results are denoted by boldface and underlining, respectively.

on LLaVA-1.5-7B, where SPSO improves the H-score over fine-tuning by 24.12 and 12.31 points, respectively. These results show that the effectiveness of SPSO extends beyond the two main target datasets and remains consistent across different downstream task types.

C EXTENSION TO OTHER MLLMS

We further evaluate SPSO on Molmo2-8B to examine its generalization beyond the MLLMs used in the main experiments. As shown in Table A4, SPSO consistently improves the stability–plasticity balance on both IconQA and COCO-Caption. On IconQA, it preserves substantially more pre-trained capability while maintaining target-task performance close to standard fine-tuning; on COCO-Caption, it improves both $A_{\mathcal{P}}$ and $A_{\mathcal{T}}$. These consistent gains over both fine-tuning and competing sparse adaptation methods indicate that SPSO is not tied to a particular MLLM architecture.

Method	Fine-tune on IconQA									
	GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
Zero-shot	58.36	83.86	57.78	47.75	84.19	81.87	68.97	23.18	46.07	34.70
Fine-tuning	54.01	48.54	45.77	28.81	68.51	61.85	51.25	**74.48**	62.86	60.72
L2-Reg	45.95	50.13	39.84	25.67	61.92	59.27	47.13	68.17	57.65	55.73
DARE Yu et al. (2024)	55.85	60.24	49.07	40.66	72.77	70.43	58.17	70.73	64.45	63.84
Grafting Panigrahi et al. (2023)	56.50	61.97	50.23	41.24	74.03	71.47	59.24	70.13	64.69	64.23
RM Xu & Zhang (2024)	53.55	63.08	51.04	45.85	75.96	73.54	60.50	70.08	65.29	64.94
FISH Mask Sung et al. (2021)	52.60	65.16	47.08	42.74	78.37	72.03	59.66	72.27	65.97	65.36
PaFi Liao et al. (2023)	54.81	71.47	52.06	44.24	79.19	72.81	62.43	71.76	67.10	66.77
Model Tailor Zhu et al. (2024)	54.93	70.83	50.47	45.29	78.82	72.48	62.14	70.38	66.26	66.00
AdaDARE Xie et al. (2025)	56.50	69.07	51.94	44.31	78.96	75.84	62.77	70.84	66.81	66.56
SPIDER Huang et al. (2025b)	54.97	72.04	54.06	43.19	80.37	75.03	63.28	71.69	67.48	67.22
SPSO	**56.63**	**79.78**	**56.14**	**46.46**	**81.72**	**77.42**	**66.36**	73.75	**70.05**	**69.86**
Method	Fine-tune on COCO-Caption									
	GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
Zero-shot	58.36	83.86	57.78	47.75	84.19	81.87	68.97	40.40	54.68	50.95
Fine-tuning	50.03	68.62	38.87	42.11	63.68	63.39	54.45	86.32	70.39	66.78
L2-Reg	47.93	60.21	34.82	38.16	54.73	52.93	48.13	80.17	64.15	60.15
DARE Yu et al. (2024)	51.31	73.03	48.02	44.38	70.04	67.46	59.04	83.47	71.26	69.16
Grafting Panigrahi et al. (2023)	53.59	75.04	49.08	43.71	71.53	69.47	60.40	83.33	71.87	70.04
RM Xu & Zhang (2024)	51.32	71.93	51.47	44.02	73.96	71.54	60.71	85.37	73.04	70.96
FISH Mask Sung et al. (2021)	50.35	73.46	47.93	42.04	73.38	70.82	59.66	89.64	74.65	71.64
PaFi Liao et al. (2023)	54.65	76.24	51.18	45.21	76.98	73.52	62.96	85.37	74.17	72.47
Model Tailor Zhu et al. (2024)	54.52	72.63	50.47	45.02	73.76	73.24	61.61	85.22	73.41	71.51
AdaDARE Xie et al. (2025)	55.71	74.53	43.06	45.44	80.08	74.62	62.24	85.93	74.09	72.19
SPIDER Huang et al. (2025b)	54.71	72.08	50.53	45.36	**80.58**	**75.72**	63.16	87.13	75.15	73.24
SPSO	**56.78**	**81.52**	**54.37**	**46.37**	79.71	75.25	**65.67**	**91.20**	**78.43**	**76.36**

Table A4: Results on Molmo2-8B with a sparsity ratio $\rho = 90\%$. The optimal and sub-optimal results are denoted by boldface and underlining, respectively.

D Extension to LoRA-Based Methods

Rank	Method	Fine-tune on IconQA									
		GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
–	Zero-shot	61.97	58.23	57.96	66.20	64.60	58.10	61.18	23.18	42.18	33.62
16	LoRA	55.36	48.73	50.95	56.20	59.52	**53.94**	54.12	84.47	69.29	65.97
	LoRASculpt	57.26	52.84	53.72	57.35	58.47	51.00	55.11	84.61	69.86	66.74
	SPSO	**61.40**	**55.22**	**57.61**	**61.79**	**61.18**	53.76	**58.49**	**85.39**	**71.94**	**69.43**
32	LoRA	50.67	39.92	47.36	47.30	58.66	50.19	49.02	82.78	65.90	61.57
	LoRASculpt	57.83	54.96	53.67	59.15	59.22	52.18	56.17	84.81	70.49	67.58
	SPSO	**60.24**	**57.69**	**57.79**	**64.23**	**62.03**	**56.01**	**59.66**	**85.67**	**72.67**	**70.34**
64	LoRA	12.82	2.68	2.84	2.20	59.21	49.44	21.53	37.62	29.58	27.39
	LoRASculpt	58.05	51.44	54.81	56.25	57.13	52.04	54.95	84.45	69.70	66.58
	SPSO	**60.13**	**55.24**	**56.98**	**60.72**	**62.99**	**55.68**	**58.62**	**85.42**	**72.02**	**69.53**
Rank	Method	Fine-tune on COCO-Caption									
		GQA	TextVQA	OKVQA	OCRVQA	MM-Bench	MM-Bench-CN	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
–	Zero-shot	61.97	58.23	57.96	66.20	64.60	58.10	61.18	–	61.18	61.18
16	LoRA	59.36	42.91	50.44	57.90	**61.99**	55.19	54.63	112.57	83.60	73.56
	LoRASculpt	56.44	52.35	50.71	59.40	60.35	54.49	55.62	121.84	88.73	76.38
	SPSO	**60.95**	**56.53**	**55.14**	**64.90**	61.27	**56.14**	**59.16**	**129.05**	**94.10**	**81.12**
32	LoRA	55.48	40.35	44.57	58.25	60.72	54.31	52.28	110.55	81.42	70.99
	LoRASculpt	57.08	51.18	49.29	58.55	60.48	55.37	55.33	122.88	89.10	76.30
	SPSO	**61.25**	**56.11**	**55.13**	**65.01**	**61.16**	**56.22**	**59.15**	**129.42**	**94.28**	**81.19**
64	LoRA	44.55	24.33	35.66	52.45	58.14	52.63	44.63	108.21	76.42	63.19
	LoRASculpt	57.16	52.34	49.95	59.00	62.15	53.79	55.73	123.00	89.37	76.71
	SPSO	**60.65**	**56.48**	**55.08**	**65.45**	**63.04**	**54.43**	**59.19**	**127.93**	**93.56**	**80.93**

Table A5: LoRA-based adaptation results on LLaVA-1.5-7B with a sparsity ratio $\rho = 90\%$. The optimal and sub-optimal results within each rank are denoted by boldface and underlining, respectively.

We further extend SPSO to LoRA-based adaptation to examine whether its optimizer-level design remains effective under parameter-efficient fine-tuning. Since LoRA optimizes newly introduced low-rank parameters rather than the pre-trained weights directly, the weight-based prior used in the original SPSO formulation cannot be applied to LoRA parameters in the same form. We therefore follow LoRASculpt Liang et al. (2025) to construct a LoRA-compatible weight prior, which replaces the original weight-related score in SPSO, while retaining the remaining optimization mechanisms unchanged. Table A5 compares LoRA, LoRASculpt, and SPSO across ranks 16, 32, and 64 on IconQA and COCO-Caption. Across all ranks and both target tasks, SPSO consistently achieves

a better stability–plasticity balance than the LoRA-based baselines. In particular, compared with LoRASculpt, SPSO improves both $A_{\mathcal{P}}$ and $A_{\mathcal{T}}$ across all settings, indicating that its gains do not rely on sacrificing downstream adaptation for stronger capability retention. Moreover, SPSO remains stable as the adapter rank increases, with only minor variation in H-score across ranks on both tasks. This robustness contrasts with vanilla LoRA, which exhibits a marked degradation at rank 64 on IconQA. These results show that SPSO can be naturally extended to parameter-efficient adaptation and remains effective across different adapter capacities.

E Extension to Continual Instruction Tuning

Experimental Setup We further evaluate SPSO under multimodal continual instruction tuning (MCIT), where a model is trained on a sequence of vision-language instruction tasks and evaluated after each continual stage. Following the CoIN benchmark Chen et al. (2024a), the task order is ScienceQA Lu et al. (2022), TextVQA Singh et al. (2019), ImageNet Deng et al. (2009), GQA Hudson & Manning (2019), VizWiz Gurari et al. (2018), REC Mao et al. (2016); Kazemzadeh et al. (2014), VQAV2 Goyal et al. (2017), and OCRVQA Mishra et al. (2019). These tasks cover different multimodal capabilities, including visual question answering, scientific reasoning, image classification, and referring expression comprehension. We use MoELoRA Dou et al. (2024) and ProgLoRA Yu et al. (2025) as LoRA-based continual learning baselines and compare them with ModalPrompt Zeng et al. (2025), a prompt-based MCIT method reported in the benchmark material. Since SPSO operates at the optimizer level, it can be applied on top of continual instruction tuning pipelines without changing the task sequence, model architecture, or training objective.

Evaluation Metrics Let T be the number of tasks and $A_{t,i}$ denote the performance on task i after training on task t, where $i \leq t$. We report two final-performance metrics. The first is Last, which measures the final performance on each task after the model has finished all sequential training stages:

$$\mathrm{Last}_i = A_{T,i}, \qquad i = 1, \ldots, T. \tag{23}$$

Last directly reflects how well each earlier task is retained at the end of continual instruction tuning.

The second is Avg, which averages each task's performance over the subsequent continual stages:

$$\mathrm{Avg}_i = \frac{1}{T-i} \sum_{t=i+1}^{T} A_{t,i}, \qquad i = 1, \ldots, T-1. \tag{24}$$

Avg measures the stability of each task throughout the MCIT procedure rather than only at the final checkpoint. The benchmark also considers time-dependent continual metrics such as backward transfer and mean accuracy, but Table A6 focuses on Last and Avg because they summarize the final and trajectory-level task performance.

Experimental Results Table A6 evaluates SPSO on CoIN under multimodal continual instruction tuning. SPSO reaches 61.66 Last and achieves the best Last performance on ScienceQA, ImageNet, VizWiz, OCRVQA, and the average score, improving the average Last score over ProgLoRA Yu et al. (2025) by 2.57 points. Among methods with available stage-wise trajectories, SPSO achieves the best Avg score of 58.11, improving over ProgLoRA by 2.64 points. The gain in Avg indicates better stability throughout the continual learning trajectory. Compared with MoELoRA Dou et al. (2024), which drops to 37.48 Last and 28.53 Avg, SPSO substantially alleviates forgetting across the task stream. These results suggest that SUS-guided sparse optimization provides an effective stability-plasticity trade-off compared with mixture-based LoRA adaptation, progressive LoRA adaptation, and prompt-based continual tuning, because it directly regulates update locations, update magnitudes, and retained optimizer states during sequential training.

	Method	ScienceQA	TextVQA	ImageNet	GQA	VizWiz	REC	VQAV2	OCRVQA	*Average*
	Multi-task	81.43	61.36	90.05	60.67	52.39	66.14	63.54	61.28	67.10
	Zero-shot	67.92	57.71	47.37	61.28	45.17	6.12	52.03	53.58	48.89
Last	Finetune	26.00	25.38	28.51	33.07	26.52	0.10	40.00	52.92	29.06
	CODA-Prompt	58.15	50.16	24.04	54.33	48.94	17.83	55.86	54.42	45.46
	DualPrompt	56.40	47.12	34.96	42.03	44.14	12.01	54.43	53.36	43.05
	L2P	54.42	46.04	30.36	<u>57.09</u>	42.19	9.38	50.45	54.03	42.99
	MoELoRA	47.34	32.91	38.73	37.15	42.48	0.97	42.77	57.50	37.48
	ModalPrompt	68.42	**56.40**	41.13	**61.11**	50.13	**36.69**	**66.90**	59.68	55.06
	ProgLoRA	<u>74.84</u>	51.83	<u>83.90</u>	49.93	<u>53.87</u>	31.19	62.71	<u>64.44</u>	<u>59.09</u>
	SPSO	**77.42**	<u>54.46</u>	**86.34**	52.74	**56.31**	<u>33.82</u>	<u>65.35</u>	**66.88**	**61.66**
Avg	Finetune	13.79	15.74	9.08	28.84	15.20	0.06	40.00	–	17.53
	CODA-Prompt	48.84	47.17	18.74	50.77	42.68	15.43	55.86	–	39.93
	DualPrompt	42.81	43.41	24.12	40.52	40.39	12.76	54.43	–	36.92
	L2P	43.76	41.35	18.28	50.03	38.78	8.77	50.45	–	35.91
	MoELoRA	39.12	27.10	20.01	40.65	28.72	1.36	42.77	–	28.53
	ModalPrompt	**68.36**	**56.30**	39.66	**61.45**	50.02	**36.66**	**66.90**	–	54.19
	ProgLoRA	62.71	51.64	79.28	50.79	47.60	33.58	62.71	–	55.47
	SPSO	65.34	54.29	**81.75**	53.59	**50.26**	36.23	65.35	–	**58.11**

Table A6: Results on CoIN; SPSO uses a sparsity ratio $\rho = 90\%$. Boldface and underlining denote the optimal and sub-optimal Last results, respectively. For Avg, boldface denotes the optimal result among methods with available stage-wise trajectories.

F Details of Datasets

We utilize multimodal datasets for both target-task fine-tuning and general capability evaluation. IconQA Lu et al. (2021) and COCO-Caption Lin et al. (2014) are used for the main experiments, while ImageNet-R and Flickr30K provide additional target-task evaluations. To measure whether the model preserves its general multimodal ability after fine-tuning, we evaluate on six held-out benchmarks, including GQA Hudson & Manning (2019), TextVQA Singh et al. (2019), OKVQA Marino et al. (2019), OCRVQA Mishra et al. (2019), MM-Bench Liu et al. (2024b), and MM-Bench-CN Zhang et al. (2023). Detailed information for each dataset, including task type, split, metric, and description, is provided in Table A7. We further introduce these datasets below.

GQA: GQA Hudson & Manning (2019) represents a visual question answering benchmark specifically developed to assess real-world visual inference and compositional comprehension. It incorporates structured queries necessitating multi-stage reasoning concerning objects, characteristics, and relationships within images, thereby enabling granular examination of a model's visual reasoning competencies.

TextVQA: TextVQA Singh et al. (2019) serves as a visual question answering benchmark that necessitates models to recognize and comprehend textual information embedded within images for responding to natural language queries. This benchmark assesses the capability of models to simultaneously conduct scene text comprehension and visual–linguistic inference, thereby exposing the shortcomings of conventional VQA models that are deficient in text recognition functionality.

OKVQA: OKVQA Marino et al. (2019) evaluates knowledge-based visual question answering. It requires models to combine visual perception with external commonsense or factual knowledge that is not directly observable from the image. This dataset therefore assesses whether a model can synthesize visual evidence with knowledge-driven inference.

OCRVQA: OCRVQA Mishra et al. (2019) concentrates on formulating answers through the recognition and comprehension of textual content present within images. It assesses the model's proficiency in conducting optical character recognition and correlating identified text with visual context to facilitate precise question answering.

MM-Bench: MM-Bench Liu et al. (2024b) constitutes a holistic multimodal benchmark developed to assess the comprehensive capabilities of multimodal models spanning various vision–language

Dataset	Task	Split	Metric	Description	# Train	# Test
ImageNet-R	Image classification	Test	Accuracy	Distribution-shifted image classification used as an additional target fine-tuning task.	10,000	3,000
Flickr30K	Image captioning	Train/Test	CIDEr	Caption generation on natural images used as an additional target fine-tuning task.	10,000	1,000
IconQA	Abstract visual reasoning	Train/Test	Accuracy	Question answering over abstract icon diagrams, requiring visual reasoning over symbolic scenes and spatial relations.	10,000	6,316
COCO-Caption	Image captioning	Train/Test	CIDEr	Caption generation on natural images from MS-COCO, used to evaluate task adaptation under image-to-text supervision.	10,000	5,000
GQA	Compositional VQA	Train/Test-Dev	Accuracy	Visual question answering benchmark emphasizing object relations, compositional reasoning, and scene understanding.	–	12,578
TextVQA	Text-centric VQA	Train/Test	VQA score	Visual question answering that requires reading and grounding scene text in images.	–	5,000
OKVQA	Knowledge-based VQA	Train/Test	VQA score	Open-ended VQA benchmark requiring both visual perception and external commonsense or factual knowledge.	–	5,046
OCRVQA	OCR-based VQA	Train/Test	Accuracy	Question answering over images containing text, focusing on OCR ability and text-image reasoning.	–	100,424
MMBench	Multimodal benchmark	Dev/Test	Accuracy	Comprehensive multiple-choice benchmark covering perception, reasoning, and knowledge-oriented multimodal abilities.	–	6,666 / 7,299
MMBench-CN	Chinese multimodal benchmark	Dev/Test	Accuracy	Chinese version of MMBench for evaluating multilingual multimodal understanding and reasoning.	–	6,666 / 7,299

Table A7: Datasets used for fine-tuning and evaluation.

competencies, encompassing perception, inference, and knowledge comprehension. It offers standardized multiple-choice problems partitioned into English and Chinese versions, facilitating systematic assessment of zero-shot and cross-lingual performance in multimodal large language models.

MM-Bench-CN: MM-Bench-CN Zhang et al. (2023) constitutes the Chinese variant of the MM-Bench benchmark, designed to evaluate multimodal models' performance in Chinese language contexts. It maintains the same comprehensive evaluation framework as its English counterpart, covering perception, reasoning, and knowledge understanding tasks, while specifically assessing models' capabilities in processing Chinese visual-language tasks. This benchmark enables systematic comparison of multimodal models' cross-lingual generalization abilities.

IconQA: IconQA Lu et al. (2021) represents a visual question answering benchmark specifically developed for abstract diagram comprehension and visual–linguistic inference. In our work, we concentrate on the text-based multiple-choice subset.

COCO-Caption: COCO-Caption Lin et al. (2014) constitutes a large-scale vision dataset comprising images depicting complex everyday scenarios with numerous objects annotated for object identification, segmentation, and caption generation. This dataset is extensively employed to assess visual comprehension and linguistic generation capabilities, particularly within image captioning and multimodal learning applications.

G Evaluation Metrics

We denote the average accuracy across pre-trained tasks as $A_{\mathcal{P}}$ to quantify retained zero-shot capability, and the performance on target tasks as $A_{\mathcal{T}}$ to assess downstream adaptation capability. Following established protocols Zhu et al. (2024), we adopt two complementary metrics to evaluate MLLM performance. The first is the arithmetic mean (Avg), yielding a balanced measure of overall performance:

$$\text{Avg} = \frac{A_{\mathcal{P}} + A_{\mathcal{T}}}{2}. \tag{25}$$

The second is the H-score, defined as the harmonic mean of the pre-trained and target performances:

$$\text{H-score} = \frac{2 \cdot A_{\mathcal{P}} \cdot A_{\mathcal{T}}}{A_{\mathcal{P}} + A_{\mathcal{T}}}. \tag{26}$$

The H-score is particularly sensitive to performance imbalances, providing a rigorous measure of the trade-off between downstream adaptation and pre-trained knowledge preservation.

H ALGORITHM OF SPSO

The algorithm is outlined in Algorithm 1.

Algorithm 1 Stability-Plasticity-Aware Sparse Optimizer (SPSO)

Require: Total training steps N_{step}, training data D_{tr}, pre-trained weights $\mathbf{W}_0$, sparsity ratio ρ, learning rate η, optimizer coefficients β_1, β_2, and numerical constant ϵ
Ensure: Fine-tuned weights $\mathbf{W}^{(N_{\text{step}})}$

Initialize $\mathbf{W}^{(0)} \leftarrow \mathbf{W}_0$, sparse moment buffers $\mathbf{u}, \mathbf{v} \leftarrow \emptyset$, and $\mathbf{g}^{(0)} \leftarrow \mathbf{0}$
Compute the Knowledge-Retention Score $R_j \leftarrow [\sigma(\text{z-score}(|\mathbf{W}_0|))]_j$
for $t = 1, 2, \ldots, N_{\text{step}}$ **do**
  Sample a mini-batch (x, y) from D_{tr}
  Compute $\mathcal{L}^{(t)} \leftarrow \mathcal{L}_{\text{task}}(\Phi(x; \mathbf{W}^{(t-1)}), y)$ and $\mathbf{g}^{(t)} \leftarrow \nabla_{\mathbf{W}} \mathcal{L}^{(t)}$
  Compute the Task-Adaptation Score $C_j^{(t)} \leftarrow [\sigma(\text{z-score}(\mathbf{g}^{(t)}))]_j$
  Compute the Safe-Update Score $S_j^{(t)} \leftarrow C_j^{(t)}(1 - R_j)$
  $K \leftarrow \max(1, \lfloor (1-\rho)N \rfloor)$
  Sample $\xi_j \sim \text{Gumbel}(0, 1)$ for each parameter
  $\mathcal{A}_t \leftarrow \arg\text{top}_K \{\log S_j^{(t)} + \xi_j\}_{j=1}^{N}$
  for each $j \in \mathcal{A}_t$ **do**
    $u_j^{(t)} \leftarrow \beta_1 u_j^{(t-1)} + (1-\beta_1) g_j^{(t)}$
    $v_j^{(t)} \leftarrow \beta_2 v_j^{(t-1)} + (1-\beta_2)(g_j^{(t)})^2$
  end for
  Prune sparse moment buffers by retaining the top-K entries ranked by $\phi_j^{(t)} \leftarrow |u_j^{(t)}| S_j^{(t)}$
  Apply bias correction to obtain $\hat{u}_j^{(t)}$ and $\hat{v}_j^{(t)}$
  for each parameter j **do**
    if $j \in \mathcal{A}_t$ **then**
      $$w_j^{(t)} \leftarrow w_j^{(t-1)} - \eta \frac{\hat{u}_j^{(t)} S_j^{(t)}}{\sqrt{\hat{v}_j^{(t)}} + \epsilon}$$
    else
      $w_j^{(t)} \leftarrow w_j^{(t-1)}$
    end if
  end for
end for
return $\mathbf{W}^{(N_{\text{step}})}$

I ADDITIONAL ABLATION STUDIES

This section provides detailed ablation studies and diagnostic analyses for SPSO. All default SPSO results use the same sparsity ratio $\rho = 90\%$ as the main experiments and are aligned with the corresponding IconQA entries in Tables 1 and 2. Unless otherwise specified, the ablations are conducted on IconQA with NVILA-Lite-8B and LLaVA-1.5-7B.

Robustness Across Random Seeds. We evaluate each method using three random seeds (21, 42, and 63) and report the mean and standard deviation in Table A8. SPSO achieves the highest Avg and H-score on both NVILA-Lite-8B and LLaVA-1.5-7B across the three-seed evaluation. Its performance margins over the strongest baseline are larger than the corresponding standard deviations. Despite the stochasticity introduced by Gumbel top-K sampling, SPSO also exhibits variance comparable to existing sparse fine-tuning methods, showing that the sampling procedure does not increase sensitivity to random seeds.

Effect of Training Data Size. We fine-tune NVILA-Lite-8B and LLaVA-1.5-7B on IconQA using 1K, 5K, 10K, and 15K training samples, and report the mean and standard deviation over three random seeds in Table A9. For standard fine-tuning, increasing the amount of training data consistently improves $A_{\mathcal{T}}$ but is accompanied by a decrease in $A_{\mathcal{P}}$ on both models. For example, on NVILA-Lite-8B, $A_{\mathcal{T}}$ increases from 68.24 to 98.12, while $A_{\mathcal{P}}$ decreases from 58.10 to 22.18 as the training set grows from 1K to 15K samples. In comparison, SPSO improves $A_{\mathcal{T}}$ with increasing

Method	NVILA-Lite-8B				LLaVA-1.5-7B			
	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
Fine-tuning	$26.30_{\pm 0.44}$	$96.34_{\pm 0.21}$	$61.32_{\pm 0.25}$	$41.32_{\pm 0.42}$	$31.49_{\pm 0.38}$	$59.02_{\pm 0.25}$	$45.26_{\pm 0.24}$	$41.07_{\pm 0.30}$
RM	$67.27_{\pm 0.31}$	$94.37_{\pm 0.18}$	$80.82_{\pm 0.20}$	$78.55_{\pm 0.23}$	$53.70_{\pm 0.29}$	$55.16_{\pm 0.21}$	$54.43_{\pm 0.20}$	$54.42_{\pm 0.20}$
FISH Mask	$64.32_{\pm 0.34}$	$95.43_{\pm 0.16}$	$79.88_{\pm 0.21}$	$76.85_{\pm 0.25}$	$52.48_{\pm 0.31}$	$58.46_{\pm 0.18}$	$55.47_{\pm 0.18}$	$55.31_{\pm 0.20}$
PaFi	$68.15_{\pm 0.28}$	$94.72_{\pm 0.17}$	$81.43_{\pm 0.19}$	$79.27_{\pm 0.22}$	$56.38_{\pm 0.27}$	$55.32_{\pm 0.19}$	$55.85_{\pm 0.17}$	$55.85_{\pm 0.18}$
SPSO	$74.50_{\pm 0.24}$	$96.01_{\pm 0.15}$	$85.25_{\pm 0.18}$	$83.90_{\pm 0.20}$	$60.20_{\pm 0.25}$	$58.92_{\pm 0.16}$	$59.56_{\pm 0.16}$	$59.55_{\pm 0.17}$

Table A8: Results over three random seeds on IconQA with a sparsity ratio $\rho = 90\%$. Boldface denotes the optimal result.

Training Data	Method	NVILA-Lite-8B				LLaVA-1.5-7B			
		$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
1K	FT	$58.10_{\pm 0.48}$	$68.24_{\pm 0.34}$	$63.17_{\pm 0.31}$	$62.76_{\pm 0.36}$	$45.20_{\pm 0.46}$	$47.80_{\pm 0.32}$	$46.50_{\pm 0.30}$	$46.46_{\pm 0.34}$
	SPSO	$74.95_{\pm 0.32}$	$69.10_{\pm 0.25}$	$72.03_{\pm 0.24}$	$71.91_{\pm 0.27}$	$60.55_{\pm 0.31}$	$49.10_{\pm 0.24}$	$54.83_{\pm 0.23}$	$54.23_{\pm 0.26}$
5K	FT	$42.85_{\pm 0.45}$	$84.73_{\pm 0.31}$	$63.79_{\pm 0.29}$	$56.92_{\pm 0.33}$	$38.74_{\pm 0.43}$	$54.36_{\pm 0.29}$	$46.55_{\pm 0.27}$	$45.24_{\pm 0.31}$
	SPSO	$74.72_{\pm 0.29}$	$86.88_{\pm 0.22}$	$80.80_{\pm 0.23}$	$80.34_{\pm 0.25}$	$60.38_{\pm 0.28}$	$55.42_{\pm 0.21}$	$57.90_{\pm 0.22}$	$57.79_{\pm 0.24}$
10K	FT	$26.30_{\pm 0.44}$	$96.34_{\pm 0.21}$	$61.32_{\pm 0.25}$	$41.32_{\pm 0.42}$	$31.49_{\pm 0.38}$	$59.02_{\pm 0.25}$	$45.26_{\pm 0.24}$	$41.07_{\pm 0.30}$
	SPSO	$74.50_{\pm 0.24}$	$96.01_{\pm 0.15}$	$85.25_{\pm 0.18}$	$83.90_{\pm 0.20}$	$60.20_{\pm 0.25}$	$58.92_{\pm 0.16}$	$59.56_{\pm 0.16}$	$59.55_{\pm 0.17}$
15K	FT	$22.18_{\pm 0.44}$	$98.12_{\pm 0.33}$	$60.15_{\pm 0.30}$	$36.18_{\pm 0.34}$	$28.62_{\pm 0.42}$	$61.35_{\pm 0.30}$	$44.98_{\pm 0.28}$	$39.03_{\pm 0.32}$
	SPSO	$73.86_{\pm 0.30}$	$98.04_{\pm 0.21}$	$85.95_{\pm 0.22}$	$84.25_{\pm 0.24}$	$59.78_{\pm 0.27}$	$61.12_{\pm 0.19}$	$60.45_{\pm 0.20}$	$60.44_{\pm 0.22}$

Table A9: Results at different IconQA training-data scales with a sparsity ratio $\rho = 90\%$. Boldface denotes the optimal result at each scale.

data scale while maintaining relatively stable $A_{\mathcal{P}}$, resulting in higher Avg and H-score across all four training-data scales. These results show that SPSO maintains a favorable stability–plasticity balance across different amounts of downstream training data.

Sensitivity to Learning Rate. Figure A1 evaluates the sensitivity of FT and SPSO to different learning rates on IconQA. For FT, increasing the learning rate initially improves $A_{\mathcal{T}}$, while $A_{\mathcal{P}}$ decreases continuously; at larger learning rates, $A_{\mathcal{T}}$ also declines. In comparison, SPSO maintains relatively stable $A_{\mathcal{P}}$ and H-score across the evaluated learning rates, while preserving competitive $A_{\mathcal{T}}$. These results indicate that SPSO is less sensitive to the choice of learning rate and maintains a more consistent stability–plasticity balance across different learning-rate settings.

Sensitivity to Sparsity Ratio and Updated Layers. Figure A2 examines the sensitivity of SPSO to the number of updated decoder layers and the sparsity ratio ρ on IconQA. For the number of updated layers, updating too few layers limits target-task adaptation, as reflected by the lower $A_{\mathcal{T}}$ on both models. Increasing the trainable depth improves $A_{\mathcal{T}}$, while its effect on $A_{\mathcal{P}}$ differs across models. NVILA-Lite-8B reaches a favorable H-score with the last 12 decoder layers updated, whereas LLaVA-1.5-7B continues to benefit from updating additional layers. We therefore adopt the last 12 decoder layers as a unified configuration in the main experiments.

For the sparsity ratio, increasing ρ progressively reduces the active update budget and improves pre-trained-task retention on both models. Meanwhile, $A_{\mathcal{T}}$ remains relatively stable over a broad range of sparsity ratios, indicating that a large fraction of parameters can be excluded from optimization without immediately limiting downstream adaptation. Consequently, the H-score increases as ρ grows and reaches its highest value around $\rho = 90\%$. Beyond this point, $A_{\mathcal{P}}$ is already close to its zero-shot level, while the increasingly limited update budget begins to reduce $A_{\mathcal{T}}$. The H-score therefore decreases at $\rho = 95\%$ and $\rho = 99\%$, and approaches the zero-shot endpoint at $\rho = 100\%$. This trend reflects the stability–plasticity trade-off controlled by the sparsity ratio: increasing sparsity improves capability retention up to a certain range, whereas excessive sparsity restricts the capacity available for downstream adaptation. Accordingly, we use $\rho = 90\%$ as the default sparsity setting.

Diagnostic Analysis of Safe-Update Score. To examine whether SUS provides informative guidance for parameter selection, we rank trainable parameters by their average SUS during SPSO training and construct fixed sparse-update variants using the bottom, middle, and top 10% groups, together with a random 10% baseline under the same update budget. As shown in Table A10, the results exhibit a clear ordering with respect to SUS. Updating the bottom 10% group yields $A_{\mathcal{P}} = 33.20$, close to standard fine-tuning, whereas the middle and top 10% groups progressively improve pre-trained-task retention. The top 10% group also achieves the highest $A_{\mathcal{T}}$, Avg, and H-score among

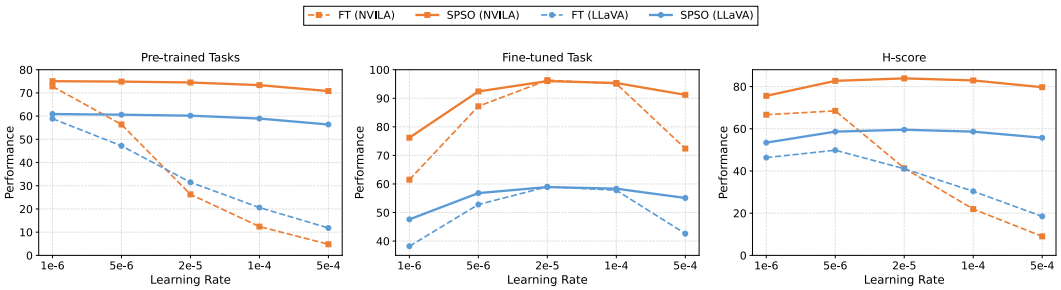


Figure A1: Learning-rate sensitivity of FT and SPSO on IconQA using NVILA-Lite-8B and LLaVA-1.5-7B. The markers at the default decoder learning rate 2×10^{-5} correspond to the IconQA entries in the main experiments, where SPSO preserves substantially stronger pre-trained-task performance and H-score than FT.

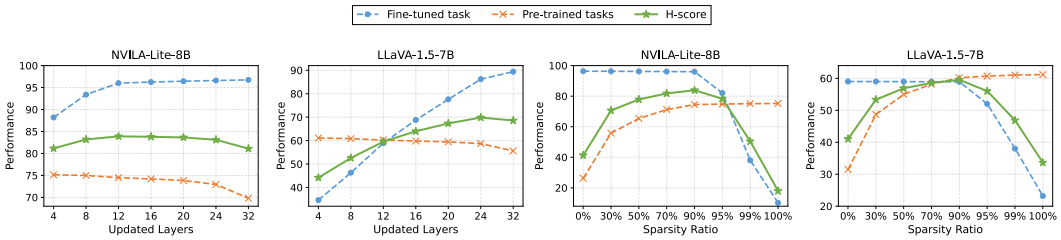


Figure A2: Sensitivity of SPSO to the sparsity ratio ρ and the number of updated decoder layers on IconQA using NVILA-Lite-8B and LLaVA-1.5-7B. The default configuration used in the main experiments is $\rho = 90\%$ with the last 12 decoder layers updated.

the fixed sparse-update variants, indicating that higher-SUS parameters provide a more favorable stability–plasticity trade-off. SPSO further improves both $A_{\mathcal{P}}$ and $A_{\mathcal{T}}$ over the fixed top-10% variant, suggesting that dynamically using SUS during optimization provides additional benefits beyond fixed SUS-based parameter selection.

Group	Norm. SUS	$A_{\mathcal{P}}$	$A_{\mathcal{T}}$	Avg	H-score
Zero-shot	–	61.18	23.18	42.18	33.62
FT	–	31.49	59.02	45.26	41.07
Bottom 10%	0.09	33.20	56.40	44.80	41.80
Middle 10%	0.44	48.10	56.72	52.41	52.06
Top 10%	0.86	55.84	57.88	56.86	56.84
Random 10%	0.49	47.65	56.35	52.00	51.64
SPSO	**0.79**	**60.20**	**58.92**	**59.56**	**59.55**

Table A10: SUS diagnostic results on LLaVA-1.5-7B. SUS denotes the safe-update score. Boldface denotes the optimal result.

J LIMITATIONS

This work leaves several directions for future exploration. First, SPSO focuses on optimizer-level control of sparse updates, making it complementary to rehearsal-based, routing-based, or memory-augmented continual learning methods. Combining SPSO with these strategies may further improve performance in continual adaptation scenarios. Second, SPSO uses a fixed sparsity ratio throughout fine-tuning. While our sensitivity analysis shows consistent performance across a range of sparsity levels, dynamically adjusting the active update budget according to training dynamics may provide a more flexible balance between pre-trained capability retention and downstream adaptation. Third, while this paper focuses on vision-language instruction tuning, SPSO is formulated at the optimizer level and is not tied to a specific input modality. Extending SPSO to other multimodal settings, such as audio-language or video-language adaptation, remains an interesting direction for future work.