NOT WHICH PARAMETERS, BUT WHICH UPDATES: FORGETTING-AWARE FINE-TUNING OF MULTIMODAL LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning multimodal large language models (MLLMs) on specialized tasks can substantially degrade their general capabilities. Existing methods largely make adaptation decisions at the parameter level, either regulating parameters during fine-tuning or revising their accumulated changes after adaptation. Yet these formulations do not explicitly evaluate the individual optimizer proposals that form the adaptation trajectory. We observe that comparable target-task adaptation can be reached through trajectories that incur markedly different amounts of forgetting, indicating that forgetting depends not only on the target performance achieved, but also on how optimization reaches it. This motivates controlling adaptation at a finer granularity: the candidate updates proposed throughout optimization. To this end, we introduce **SPUR** (**S**elective **P**er-step **U**pdate **R**econciliation), an update-centric fine-tuning framework that treats each candidate optimizer update as the decision object. At every step, SPUR estimates each update coordinate's plasticity benefit from its contribution to reducing the target loss and its stability cost from its predicted deviation from pre-trained behavior, reconciles the two into an update utility, and applies only the globally highest-utility coordinates. The decision is recomputed at every optimization step, allowing the selected update pattern to evolve with the optimization trajectory. Extensive experiments demonstrate that SPUR consistently improves the stability–plasticity trade-off over strong forgetting-mitigation baselines, without external retention data, additional trainable parameters, or inference-time overhead.

1 INTRODUCTION

Multimodal large language models (MLLMs) acquire broad vision–language capabilities through large-scale pre-training (Li et al., 2023a; Liu et al., 2023; 2025). Adapting these models to specialized downstream tasks, however, can substantially degrade their general capabilities, a phenomenon commonly referred to as catastrophic forgetting (Zhu et al., 2024; Huang et al., 2025b; Zhai et al., 2024; Liang et al., 2026; Huang et al., 2025c;a). Effective adaptation must therefore reconcile two competing objectives: plasticity, the ability to learn the target task, and stability, the preservation of pre-trained behavior.

Existing approaches largely address this stability–plasticity trade-off through parameter-level decisions. One family estimates parameter importance or sensitivity to regulate fine-tuning (Hwang et al., 2026), while another determines after adaptation which components of the resulting parameter displacement should be retained, reverted, or merged toward the pre-trained model (Zhu et al., 2024; Panigrahi et al., 2023; Sun et al., 2024). Although these methods intervene at different stages, they share a common limitation: both reason about adaptation at a granularity coarser than the optimization trajectory itself. Specifically, during fine-tuning, optimizer proposals are regulated through criteria attached to the parameters they modify, whereas after fine-tuning, those proposals have already been aggregated into a final parameter displacement. As a result, in neither case is the individual optimizer proposal at a particular training step treated as the unit of decision. Consequently, the contribution of each update to the evolving optimization trajectory is not explicitly evaluated.

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.

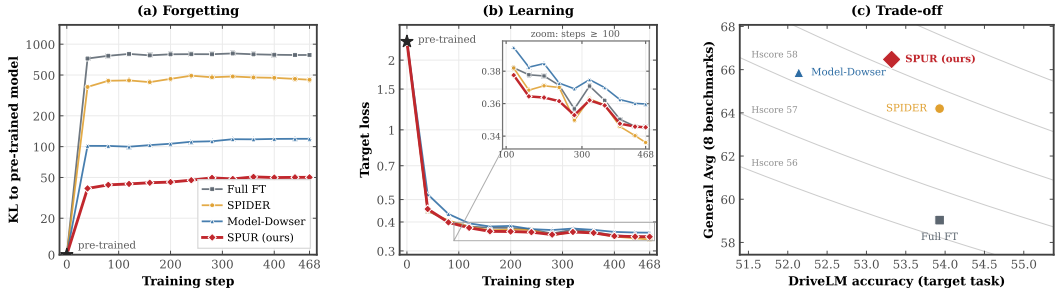


Figure 1: **Comparable target adaptation can be achieved with markedly different forgetting.** Four adaptation runs of LLaVA-1.5-7B on DriveLM with the last 8 language layers fine-tuned, using identical data, learning-rate schedule, and number of optimization steps. **(a)** Behavioral deviation from the pre-trained model, measured by $\mathrm{KL}(p_{\theta_0} \| p_{\theta_t})$, differs by up to $16\times$ at convergence. **(b)** Despite this, the target losses converge within 0.02. **(c)** These trajectory-level differences are reflected in task-level evaluation: runs with larger KL divergence retain lower general capability, while similar target losses correspond to comparable target accuracy. Gray curves denote constant Hscore. Together, the three panels show that comparable target adaptation can incur substantially different amounts of forgetting.

We next examine whether this missing update-level granularity matters in practice. Fig. 1 compares four adaptation runs of LLaVA-1.5-7B on DriveLM. Despite converging to target losses within 0.02, the runs exhibit behavioral deviations from the pre-trained model that differ by up to $16\times$. The gap emerges early in optimization and persists after the target loss has largely converged. The same discrepancy carries over to task-level evaluation, where models with comparable target accuracy differ by as much as 7.4 points in average performance over general multimodal benchmarks. Thus, comparable target-task adaptation can be achieved through trajectories that incur markedly different amounts of forgetting. This indicates that forgetting is not determined by target-task progress alone, but is also shaped by the optimization path used to achieve it. Since this path is formed by a sequence of individual updates, the optimizer proposals that construct it provide a natural granularity at which to control the stability–plasticity trade-off.

Motivated by this observation, we propose **SPUR** (**S**elective **P**er-step **U**pdate **R**econciliation), an update-centric framework for forgetting-aware MLLM fine-tuning. Rather than making adaptation decisions at the parameter level, SPUR treats each candidate optimizer update as the decision object at the moment it is proposed. At every optimization step, SPUR evaluates each update coordinate from two complementary perspectives: its plasticity benefit, which estimates its contribution to reducing the target loss, and its stability cost, which estimates the behavioral deviation it may induce from the pre-trained model. These two signals are then reconciled into an update utility, and only the globally highest-utility coordinates are applied. Because this decision is recomputed at every optimization step, the selected update pattern can evolve with the optimization trajectory. Extensive experiments demonstrate that SPUR consistently improves the stability–plasticity trade-off over strong forgetting-mitigation baselines, without external retention data, additional trainable parameters, or inference-time overhead. Our contributions are as follows:

- We identify a granularity gap in existing approaches: parameter-level regulation and post-hoc parameter revision do not explicitly evaluate the individual optimizer proposals that form the adaptation trajectory. We empirically show that comparable target-task adaptation can follow trajectories with markedly different amounts of forgetting, motivating finer-grained control over the updates that form these trajectories.
- We propose **SPUR**, an update-centric fine-tuning framework that jointly evaluates the plasticity benefit and stability cost of each candidate update and applies only the globally highest-utility updates at every optimization step.
- We conduct extensive experiments across three MLLM backbones, six downstream tasks, and five fine-tuning depths, demonstrating consistent improvements in the stability–plasticity trade-off and showing that the gains critically depend on per-step re-evaluation rather than fixed parameter-level decisions.

2 METHOD

2.1 PRELIMINARIES

We first formalize standard fine-tuning and two representative forgetting-mitigation paradigms: parameter optimization during adaptation (Huang et al., 2025b; Hwang et al., 2026) and post-hoc model merging (Zhu et al., 2024; Yang et al., 2026). Both make decisions at the parameter level, whereas SPUR evaluates candidate updates at each optimization step.

Standard Fine-Tuning. Consider a pre-trained MLLM f_{θ_0} with parameters $\theta_0 \in \mathbb{R}^d$ and a downstream dataset $\mathcal{D}_{\text{task}} = \{(x, y)\}$. Fine-tuning minimizes

$$\mathcal{L}_{\text{task}}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{task}}} \left[\ell(f_\theta(x), y)\right]. \tag{1}$$

At step t, the optimizer combines the task gradient $\mathbf{g}_t = \nabla_\theta \mathcal{L}_{\text{task}}(\theta_t)$ with its internal state to propose a candidate update $\Delta\theta_t^{\text{c}}$, defined as the quantity subtracted from the current parameters:

$$\theta_{t+1} = \theta_t - \Delta\theta_t^{\text{c}}. \tag{2}$$

Standard fine-tuning applies the full proposal without accounting for its effect on pre-trained behavior, allowing forgetting to accumulate alongside target adaptation.

Parameter Optimization. Methods in this family regulate adaptation using parameter-level importance or sensitivity. A representative formulation is

$$\theta_{t+1} = \theta_t - \mathbf{M} \odot \Delta\theta_t^{\text{c}}, \qquad M_i = \phi(I_i^{\text{pre}}), \tag{3}$$

where I_i^{pre} denotes a parameter-level criterion and $\phi(\cdot)$ represents the corresponding regulation rule. The key property is that the decision is attached to the parameter rather than to the particular candidate update proposed at step t. Consequently, different candidate updates to the same parameter are governed by the same parameter-level criterion, even though their effects on target adaptation and pre-trained behavior may vary across optimization steps.

Model Merging. Model merging methods first adapt the model to θ_T and then determine how much of each accumulated parameter change should be retained:

$$\theta_{\text{merge}} = \theta_0 + \mathbf{M} \odot (\theta_T - \theta_0), \qquad M_i = \phi(I_i^{\text{post}}), \quad I_i^{\text{post}} = \mathcal{S}_i(\theta_0, \theta_T). \tag{4}$$

Here, the decision is made after individual updates have already been aggregated into the final displacement $\theta_T - \theta_0$, making the contributions of beneficial and harmful updates difficult to disentangle.

From Parameters to Updates (Our Method). SPUR instead treats each candidate update $\Delta\theta_{t,i}^{\text{c}}$ as the decision object. At every step, it evaluates its plasticity benefit $B_{t,i}$ and stability cost $C_{t,i}$, and reconciles them into an update utility

$$U_{t,i} = \Phi(B_{t,i}, C_{t,i}). \tag{5}$$

The resulting step-dependent mask $\mathbf{M}_t$ determines which candidate updates are applied:

$$\theta_{t+1} = \theta_t - \mathbf{M}_t \odot \Delta\theta_t^{\text{c}}, \qquad M_{t,i} = \mathbb{I}[U_{t,i} \geq \tau_t]. \tag{6}$$

Here, τ_t selects the global top-ρ fraction of candidate updates, where ρ denotes the update budget.

The key distinction from Eqs. 3 and 4 lies in the decision object. Parameter optimization regulates updates through a parameter-level criterion, while model merging acts on the accumulated parameter displacement after adaptation. In contrast, SPUR evaluates the current candidate update and recomputes $\mathbf{M}_t$ at every optimization step. The same parameter may therefore be updated at one step and left unchanged at another, depending on the utility of its current proposal. This allows SPUR to adapt its update pattern throughout fine-tuning without assigning parameters a fixed update status.

2.2 OVERVIEW

Building on the update-centric formulation in Sec. 2.1, SPUR evaluates each candidate update through three stages. First, we estimate its plasticity benefit, measuring its contribution to reducing the downstream task loss. Second, we estimate its stability cost, measuring the behavioral deviation it may induce from the pre-trained model. Finally, we reconcile the two signals into an update utility and globally select the top-ρ candidate updates at each optimization step. The complete training procedure is summarized in Algorithm 1.

Algorithm 1 SPUR

Input: Pre-trained MLLM θ_0, downstream dataset $\mathcal{D}_{\text{task}}$, optimizer $\mathcal{O}$, update budget ρ
Output: Fine-tuned MLLM θ_T

$\theta \leftarrow \theta_0$
for each optimization step t **do**
  Sample mini-batch $\mathcal{B}_t \sim \mathcal{D}_{\text{task}}$
  Construct probe set $\mathcal{P}_t \subseteq \mathcal{B}_t$
  $\mathbf{g}_t \leftarrow \nabla_\theta \mathcal{L}_{\text{task}}(\theta_t; \mathcal{B}_t)$
  $\Delta\theta_t^{\text{c}} \leftarrow \mathcal{O}(\mathbf{g}_t)$ — Candidate update
  /* Plasticity Benefit */
  $\mathbf{B}_t \leftarrow [\mathbf{g}_t \odot \Delta\theta_t^{\text{c}}]_+$ — Eq. 8
  $\hat{\mathbf{B}}_t \leftarrow \text{ModuleNorm}(\mathbf{B}_t)$ — Eq. 9
  /* Stability Cost */
  $\mathbf{r}_t \leftarrow \nabla_\theta \mathcal{D}(\theta_t; \mathcal{P}_t)$ — Eq. 12
  $\hat{\mathbf{C}}_t \leftarrow \text{StabilityCost}\,(\theta_t, \Delta\theta_t^{\text{c}}, \mathbf{r}_t)$ — Eqs. 13–14
  /* Stability--Plasticity Reconciliation */
  $\hat{\mathbf{S}}_t \leftarrow \mathbf{1} - \hat{\mathbf{C}}_t$ — Eq. 15
  $\mathbf{U}_t \leftarrow \dfrac{2\hat{\mathbf{B}}_t \odot \hat{\mathbf{S}}_t}{\hat{\mathbf{B}}_t + \hat{\mathbf{S}}_t + \epsilon}$ — Eq. 16
  /* Global Update Selection */
  $\tau_t \leftarrow Q_{1-\rho}\left(\{U_{t,i}\}_{i \in \mathcal{I}}\right)$
  $\mathbf{M}_t \leftarrow \mathbb{I}\left[\mathbf{U}_t \geq \tau_t\right]$ — Eq. 17
  /* Selective Update */
  $\theta_{t+1} \leftarrow \theta_t - \mathbf{M}_t \odot \Delta\theta_t^{\text{c}}$ — Eq. 6
end for
return θ_T

Plasticity Benefit. The plasticity benefit measures how much a candidate update contributes to improving the downstream objective. Recall that $\Delta\theta_t^{\text{c}}$ denotes the optimizer proposal subtracted from the current parameters. A first-order expansion gives

$$\mathcal{L}_{\text{task}}\left(\theta_t - \Delta\theta_t^{\text{c}}\right) \approx \mathcal{L}_{\text{task}}(\theta_t) - \mathbf{g}_t^\top \Delta\theta_t^{\text{c}}, \qquad \mathbf{g}_t = \nabla_\theta \mathcal{L}_{\text{task}}(\theta_t). \tag{7}$$

We therefore define the coordinate-wise plasticity benefit as

$$B_{t,i} = \left[g_{t,i}\Delta\theta_{t,i}^{\text{c}}\right]_+, \qquad [a]_+ = \max(a, 0). \tag{8}$$

A larger $B_{t,i}$ indicates a greater predicted contribution to reducing the target-task loss, while non-positive contributions receive zero benefit. Importantly, $B_{t,i}$ evaluates the current candidate update, so the same parameter may receive different valuations across optimization steps. Because gradient and update magnitudes vary across modules, we normalize positive benefits independently within each module. Let $\mathcal{I}_m$ denote the trainable coordinates of module m and $\mathcal{I}_{t,m}^+ = \{i \in \mathcal{I}_m : B_{t,i} > 0\}$. For $i \in \mathcal{I}_{t,m}^+$,

$$\hat{B}_{t,i} = \frac{B_{t,i} - B_{t,m}^{\min}}{B_{t,m}^{\max} - B_{t,m}^{\min} + \epsilon}, \qquad B_{t,m}^{\min\,/\,\max} = \min\,/\,\max_{j \in \mathcal{I}_{t,m}^+} B_{t,j}. \tag{9}$$

We set $\hat{B}_{t,i} = 0$ whenever $B_{t,i} = 0$, yielding $\hat{B}_{t,i} \in [0, 1]$ across modules.

Stability Cost. The stability cost measures how much a candidate update is expected to disturb pre-trained behavior. Given a probe set $\mathcal{P}_t$ sampled from the current mini-batch, we quantify behavioral deviation as

$$\mathcal{D}(\theta; \mathcal{P}_t) = \mathbb{E}_{x \sim \mathcal{P}_t}\left[D_{\text{KL}}\left(p_{\theta_0}(\cdot \mid x) \,\|\, p_\theta(\cdot \mid x)\right)\right]. \tag{10}$$

The change induced by the candidate update is

$$\Delta\mathcal{D}_t = \mathcal{D}\left(\theta_t - \Delta\theta_t^{\text{c}}; \mathcal{P}_t\right) - \mathcal{D}\left(\theta_t; \mathcal{P}_t\right). \tag{11}$$

A second-order Taylor expansion around θ_t gives

$$\Delta\mathcal{D}_t \approx -\mathbf{r}_t^\top \Delta\theta_t^{\text{c}} + \frac{1}{2}(\Delta\theta_t^{\text{c}})^\top \mathbf{H}_t \Delta\theta_t^{\text{c}}, \qquad \mathbf{r}_t = \nabla_\theta \mathcal{D}(\theta; \mathcal{P}_t)|_{\theta_t}. \tag{12}$$

Here, $\mathbf{r}_t$ captures the directional effect of the current candidate update, while $\mathbf{H}_t$ characterizes local behavioral sensitivity. Since repeatedly estimating $\mathbf{H}_t$ is expensive, we approximate it with a diagonal Fisher $\mathbf{F}^{\mathrm{sta}}$ computed once at the pre-trained model. This yields

$$C_{t,i}^{\mathrm{dyn}} = \left[-r_{t,i}\Delta\theta_{t,i}^{\mathrm{c}}\right]_+, \qquad C_{t,i}^{\mathrm{sta}} = \frac{1}{2}F_i^{\mathrm{sta}}\left(\Delta\theta_{t,i}^{\mathrm{c}}\right)^2. \tag{13}$$

The dynamic term captures the effect of the current candidate update, while the static term penalizes updates along directions that are sensitive around the pre-trained solution. Negative first-order changes are clipped to zero because they locally reduce behavioral deviation. We normalize both costs independently within each module using the same min–max normalization as in Eq. 9, obtaining $\hat{C}_{t,i}^{\mathrm{dyn}}$ and $\hat{C}_{t,i}^{\mathrm{sta}}$, and define

$$\hat{C}_{t,i} = \max\left(\hat{C}_{t,i}^{\mathrm{dyn}}, \hat{C}_{t,i}^{\mathrm{sta}}\right). \tag{14}$$

An update therefore receives a high stability cost whenever either its current directional effect or its pre-trained sensitivity indicates a risk of behavioral deviation. Details of the Fisher estimation are provided in Appendix E.

Stability–Plasticity Reconciliation. We convert the normalized stability cost into a stability score

$$\hat{S}_{t,i} = 1 - \hat{C}_{t,i}, \tag{15}$$

so that larger $\hat{B}_{t,i}$ and $\hat{S}_{t,i}$ both indicate more desirable updates. We reconcile them using their harmonic mean:

$$U_{t,i} = \frac{2\,\hat{B}_{t,i}\hat{S}_{t,i}}{\hat{B}_{t,i} + \hat{S}_{t,i} + \epsilon}, \tag{16}$$

where $U_{t,i} \in [0,1]$ denotes the utility of the current candidate update. The harmonic mean assigns low utility whenever either plasticity or stability is poor, preventing a strong value on one side from compensating for a weak value on the other. Furthermore, SPUR uses this utility for selection rather than rescaling. Let $\mathcal{I} = \bigcup_m \mathcal{I}_m$ denote all trainable coordinates. At each step, we retain the global top-ρ fraction:

$$\tau_t = Q_{1-\rho}\left(\{U_{t,j}\}_{j\in\mathcal{I}}\right), \qquad M_{t,i} = \mathbb{I}\left[U_{t,i} \geq \tau_t\right]. \tag{17}$$

Global selection allocates the update budget across layers and modules according to their current utility. Because both $U_{t,i}$ and τ_t are recomputed at every optimization step, the selected update set evolves throughout fine-tuning.

3 Experiments

3.1 Experimental Setup

Architectures and Datasets. We evaluate SPUR on two representative MLLMs, LLaVA-1.5-7B (Liu et al., 2023) and NVILA-Lite-2B (Liu et al., 2025). Following Curvature-Guided Mixing (CGM) (Yang et al., 2026), we evaluate the preservation of pre-trained capabilities on eight general multimodal benchmarks: VQAv2 (Goyal et al., 2017), GQA (Hudson & Manning, 2019), VizWiz (Gurari et al., 2018), SQA (Lu et al., 2022), TextVQA (Singh et al., 2019), POPE (Li et al., 2023b), and MMBench/MMBench-CN (Liu et al., 2024). For downstream adaptation, we separately fine-tune each backbone on two target tasks from distinct domains: PathVQA (He et al., 2020), an open-ended visual question answering benchmark on pathology images, and DriveLM (Sima et al., 2024), a question answering benchmark over multi-camera driving scenes. Additional dataset and training details are provided in Appendix B.

Baselines. We compare SPUR against the pre-trained model, standard fine-tuning, and a broad set of representative methods for mitigating catastrophic forgetting: DARE (Yu et al., 2024), Model-Tailor (Zhu et al., 2024), Grafting (Panigrahi et al., 2023), CGM and CGM* (Yang et al., 2026), SPIDER (Huang et al., 2025b), and Model-Dowser (Hwang et al., 2026).

Table 1: Comparison of catastrophic-forgetting mitigation methods on **LLaVA-1.5-7B** and **NVILA-Lite-2B** (last 8 layers fine-tuned) under two downstream adaptation tasks.

Method	Pre-trained Tasks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**Target**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
LLaVA-1.5-7B — target task: **PathVQA**													
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.70	27.62	47.17	39.07	0
Fine-tuned	71.44	56.46	26.87	63.15	50.59	87.08	60.27	53.62	58.69	58.36	58.52	58.52	0
DARE	71.12	56.79	26.94	63.10	50.57	87.04	59.93	53.62	58.64	58.07	58.36	58.36	< 0
Model-Tailor	77.72	61.18	52.79	70.05	58.31	85.41	64.62	58.97	66.13	38.00	52.06	48.26	49.51
Grafting	78.13	61.96	55.58	70.43	58.24	86.90	64.45	58.28	66.75	27.67	47.21	39.12	0.29
CGM	76.54	61.30	55.70	69.37	57.33	86.86	62.57	56.47	65.77	37.62	51.70	47.87	47.56
CGM*	72.32	57.13	28.32	65.34	51.60	87.14	60.70	54.40	59.62	58.27	58.94	58.94	20.88
SPIDER	74.58	58.66	40.97	67.58	54.53	86.97	62.66	56.72	62.83	57.49	60.16	60.05	67.52
Model-Dowser	73.63	58.46	46.53	70.01	56.48	86.94	64.88	58.62	64.44	51.38	57.91	57.17	74.44
SPUR	74.22	58.50	47.11	68.71	55.51	87.24	63.26	58.02	64.07	58.13	**61.10**	**60.96**	**80.10**
LLaVA-1.5-7B — target task: **DriveLM**													
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.71	15.64	41.17	25.34	0
Fine-tuned	69.96	56.25	23.84	69.18	50.24	85.23	62.83	54.74	59.03	53.93	56.48	56.37	0
DARE	69.93	56.43	24.38	69.11	50.24	85.15	62.40	54.66	59.04	53.92	56.48	56.36	0.13
Model-Tailor	77.92	61.23	55.17	70.05	57.92	86.94	64.71	58.79	66.59	33.26	49.93	44.36	62.73
Grafting	78.13	61.95	55.55	70.46	58.15	86.81	64.36	58.19	66.70	15.61	41.16	25.30	< 0
CGM	77.31	61.05	55.12	70.08	57.32	85.00	63.34	55.17	65.55	25.32	45.44	36.53	38.96
CGM*	70.91	56.73	27.34	69.32	50.77	85.44	63.09	54.83	59.80	53.75	56.78	56.62	18.24
SPIDER	75.12	58.61	52.77	69.56	52.42	84.09	63.85	57.16	64.20	53.93	59.06	58.62	80.44
Model-Dowser	78.24	60.01	55.43	70.48	56.96	84.35	64.11	57.33	65.86	52.14	59.00	58.20	92.06
SPUR	77.52	61.03	57.88	70.29	57.27	85.57	64.02	58.19	66.47	53.32	**59.90**	**59.17**	**97.67**
NVILA-Lite-2B — target task: **PathVQA**													
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.23	52.82	63.02	61.37	0
Fine-tuned	66.32	50.84	52.96	80.10	59.29	86.61	75.79	56.72	66.08	62.81	64.44	64.40	0
DARE	66.57	51.34	51.75	79.75	57.66	87.82	75.62	57.16	65.96	62.42	64.19	64.14	< 0
Model-Tailor	78.41	60.78	66.47	82.90	71.01	89.72	78.43	59.14	73.36	58.24	65.80	64.93	70.34
Grafting	80.09	62.16	65.56	82.72	70.94	88.91	77.92	58.71	73.38	52.61	62.99	61.28	< 0
CGM	77.88	62.07	63.74	81.02	69.34	87.58	77.75	57.07	72.06	57.36	64.71	63.87	58.90
CGM*	74.07	57.17	61.43	81.73	66.96	89.20	76.90	57.93	70.67	62.05	66.36	66.08	75.82
SPIDER	72.07	54.83	60.17	82.48	66.71	87.54	77.83	58.62	70.03	62.02	66.02	65.78	69.10
Model-Dowser	78.16	60.03	65.28	82.93	70.57	89.86	78.09	58.97	72.99	60.46	66.72	66.13	85.38
SPUR	75.04	57.36	63.02	82.72	70.31	88.01	78.26	58.28	71.62	62.39	**67.01**	**66.69**	**85.74**
NVILA-Lite-2B — target task: **DriveLM**													
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.22	9.68	41.45	17.10	0
Fine-tuned	64.74	49.44	18.55	77.34	35.61	88.66	67.60	52.07	56.75	57.13	56.94	56.94	0
DARE	64.49	47.56	17.91	76.96	33.91	87.01	64.88	48.19	55.11	56.74	55.93	55.92	< 0
Model-Tailor	79.24	61.43	65.57	83.19	70.35	89.25	76.21	58.71	73.00	35.13	54.06	47.43	69.48
Grafting	79.87	61.94	65.42	82.86	70.50	88.09	77.92	57.84	73.05	11.09	42.07	19.26	5.77
CGM	77.22	60.86	64.31	79.01	60.73	88.16	72.89	54.48	69.71	21.29	45.50	32.62	37.32
CGM*	74.41	58.29	53.18	80.66	60.88	88.63	73.32	57.41	68.35	55.63	61.99	61.34	81.53
SPIDER	74.22	58.68	48.44	80.17	59.86	89.08	71.10	52.50	66.76	56.58	61.67	61.25	75.25
Model-Dowser	79.06	60.97	63.39	82.98	69.51	87.82	77.32	59.31	72.55	52.95	62.75	61.22	93.48
SPUR	78.87	60.92	65.65	82.39	69.14	87.63	77.15	58.79	72.57	54.33	**63.45**	**62.14**	**95.04**

Evaluation Metrics. Let S_{P} denote the average accuracy across the eight general benchmarks (Pre-Avg), and let S_{T} denote the target-task accuracy. To measure relative retention and adaptation, we further normalize both scores using the pre-trained model (PT) and standard fine-tuning (FT):

$$\bar{S}_{\mathrm{P}} = \frac{S_{\mathrm{P}} - S_{\mathrm{P}}^{\mathrm{FT}}}{S_{\mathrm{P}}^{\mathrm{PT}} - S_{\mathrm{P}}^{\mathrm{FT}}}, \qquad \bar{S}_{\mathrm{T}} = \frac{S_{\mathrm{T}} - S_{\mathrm{T}}^{\mathrm{PT}}}{S_{\mathrm{T}}^{\mathrm{FT}} - S_{\mathrm{T}}^{\mathrm{PT}}}. \tag{18}$$

We report three aggregate metrics:

$$\mathrm{Avg} = \frac{S_{\mathrm{P}} + S_{\mathrm{T}}}{2}, \qquad \mathrm{Hscore} = \frac{2 S_{\mathrm{P}} S_{\mathrm{T}}}{S_{\mathrm{P}} + S_{\mathrm{T}}}, \qquad \mathrm{Hscore}^{*} = \frac{2 \bar{S}_{\mathrm{P}} \bar{S}_{\mathrm{T}}}{\bar{S}_{\mathrm{P}} + \bar{S}_{\mathrm{T}}}. \tag{19}$$

Avg summarizes overall performance, while Hscore and Hscore* quantify the stability–plasticity trade-off on the absolute and normalized scales, respectively.

Implementation Details. Following Model-Dowser Hwang et al. (2026), we fine-tune the visual projector together with the last $K \in \{4, 8, 12, 16, 20\}$ language layers for both backbones. All experiments use a learning rate of 10^{-4}. On PathVQA, LLaVA-1.5-7B and NVILA-Lite-2B are trained for 1 and 3 epochs, respectively, while on DriveLM both backbones are trained for 1 epoch. We report results up to $K = 20$, since fine-tuning 24 language layers exceeds the memory capacity of a single NVIDIA H200 GPU under our training setup. For each backbone–task configuration, we sweep the SPUR update budget ρ and select the setting achieving the highest Avg. All evaluation metrics for that configuration are reported under the same selected ρ. The sensitivity to ρ is analyzed in Sec. 3.3.

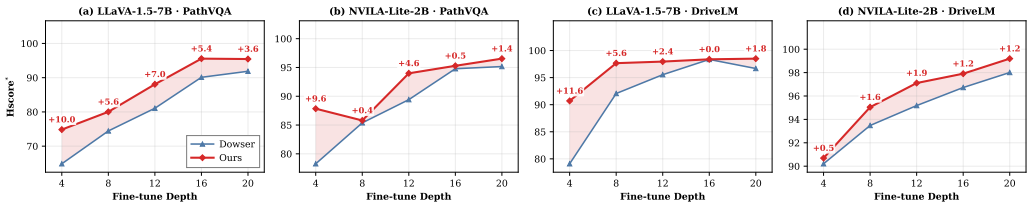


Figure 2: **Hscore* vs. fine-tuning depth.** SPUR vs. Model-Dowser across 4–20 fine-tuned language layers. Shading and annotations show SPUR's improvement at each depth.

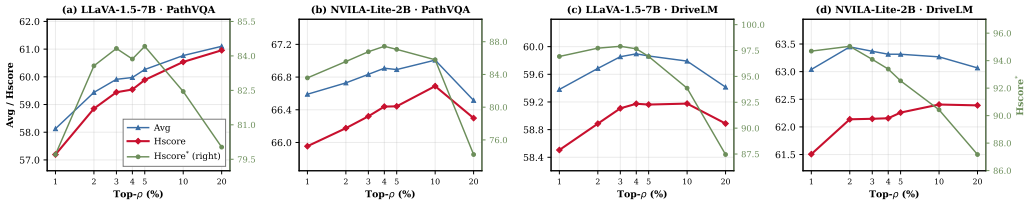


Figure 3: **Avg, Hscore, and Hscore* vs. top-ρ.** Sensitivity to the budget ρ under 8-layer fine-tuning on PathVQA and DriveLM. Avg and Hscore use the left axis, Hscore* the right.

3.2 Main Results

Results Comparisons. Table 1 reports results when fine-tuning the last eight language layers. Existing methods often favor either general-capability retention or target-task adaptation, whereas SPUR achieves the best aggregate stability–plasticity trade-off, yielding the highest Avg, Hscore, and Hscore* across all four settings. The gains are particularly pronounced on LLaVA-1.5-7B, where SPUR improves Hscore* over the strongest baseline by 5.66 points on PathVQA and 5.61 points on DriveLM. Across all settings, SPUR further improves Avg by 0.29–0.94 points and Hscore by 0.55–0.91 points. These results indicate that SPUR maintains strong target-task adaptation while retaining substantially more general capability than standard fine-tuning. This pattern is consistent with its per-step reconciliation of plasticity benefit and stability cost. Additional model and datasets comparisons are provided in Appendix I and Appendix I.1, respectively.

Memory and Update Sparsity. Table 2 compares the memory scaling, update fraction, and overall performance of different methods. SPUR requires $O(3|\mathcal{P}|)$ memory, matching SPIDER, while applying only 2%–20% of candidate update coordinates at each step, compared with the 50% update fraction used by SPIDER. Despite this sparser per-step update budget, SPUR achieves the highest Avg across all four settings. Specifically, it improves over the strongest baseline by 0.29 and 0.94 points on PathVQA, and by 0.70 and 0.84 points on DriveLM for NVILA-Lite-2B and LLaVA-1.5-7B, respectively. Overall, SPUR combines sparse per-step updates with moderate memory overhead while maintaining strong overall performance.

Table 2: Memory scaling and update fraction ρ under 8-layer fine-tuning. For SPUR, ρ is 10%/2% on NVILA and 20%/4% on LLaVA for PathVQA/DriveLM, respectively.

Method	Memory	ρ	Avg_{Path} (N/L)	$\text{Avg}_{\text{Drive}}$ (N/L)
Full FT	$\mathcal{O}(\lvert\mathcal{P}\rvert)$	100%	64.44 / 58.52	56.94 / 56.48
Grafting	$\mathcal{O}(2\lvert\mathcal{P}\rvert)$	100%	62.99 / 47.21	42.07 / 41.16
DARE	$\mathcal{O}(\lvert\mathcal{P}\rvert)$	10%	64.19 / 58.36	55.93 / 56.48
Model-Tailor	$\mathcal{O}(\lvert\mathcal{P}\rvert)$	10%	65.80 / 52.06	54.06 / 49.93
CGM	$\mathcal{O}(4\lvert\mathcal{P}\rvert)$	100%	64.71 / 51.70	45.50 / 45.44
SPIDER	$\mathcal{O}(3\lvert\mathcal{P}\rvert)$	50%	66.02 / 60.16	61.67 / 59.06
Model-Dowser	$\mathcal{O}(\lvert\mathcal{P}\rvert)$	10%	66.72 / 57.91	62.75 / 59.00
SPUR	$\mathcal{O}(3\lvert\mathcal{P}\rvert)$	$2\sim20\%$	**67.01 / 61.10**	**63.45 / 59.90**

3.3 Ablation Studies

Effect of Fine-Tuning Depth. Fig. 2 compares SPUR with Model-Dowser across $K \in \{4, 8, 12, 16, 20\}$ fine-tuned language layers. SPUR matches or outperforms Model-Dowser at every depth across both backbones and both target tasks. The advantage is most pronounced under

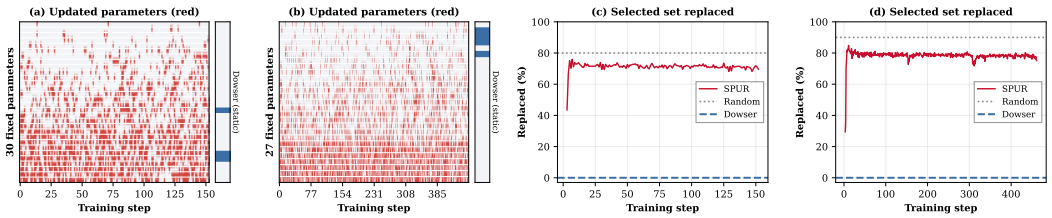


Figure 4: **SPUR decides per update, not per parameter.** Results on PathVQA with 8 fine-tuned language layers: (a)(c) LLaVA-1.5-7B with $\rho = 20\%$, and (b)(d) NVILA-Lite-2B with $\rho = 10\%$. (a)(b) Selection status of a fixed set of parameters across training (red: selected in the top-ρ updates); the right strip shows the corresponding fixed trainable set of Model-Dowser. (c)(d) Fraction of the selected set replaced between consecutive steps. The dashed line denotes a fixed set (0% turnover), while the dotted line denotes random top-ρ selection, whose expected turnover is $1 - \rho$.

shallow fine-tuning. At $K = 4$, SPUR improves Hscore* by 10.0 and 9.6 points on PathVQA for LLaVA-1.5-7B and NVILA-Lite-2B, respectively, and by 11.6 points on LLaVA-1.5-7B/DriveLM. On NVILA-Lite-2B/DriveLM, where Model-Dowser already achieves over 90 Hscore* at $K = 4$, SPUR still provides consistent gains of 0.5–1.9 points across depths. As more layers are fine-tuned, the performance gap generally narrows, with the two methods reaching similarly strong stability–plasticity trade-offs at larger depths. One possible explanation is that a larger trainable parameter space offers greater flexibility for adaptation, reducing the disadvantage of parameter-level constraints. We report results up to $K = 20$, as fine-tuning 24 language layers exceeds the memory capacity of a single NVIDIA H200 GPU under our setup. More results are provided in Appendix F.

Effect of the Top-ρ Budget. Fig. 3 examines the sensitivity of SPUR to the update budget ρ over the predefined candidate range used in the main experiments. Increasing ρ allows more candidate updates to be applied, which generally strengthens target-task adaptation but can reduce the preservation of pre-trained capabilities. Consequently, performance typically improves at first and then saturates or declines, reflecting the stability–plasticity trade-off. Across both backbones and target tasks, Hscore tends to peak at relatively larger budgets, around $\rho = 10$–20%, whereas Hscore* favors smaller budgets, around $\rho = 2$–5%.. Additional results across fine-tuning depths are provided in Appendix G.

Components Ablation. Table 3 ablates the main components of SPUR on NVILA-Lite-2B/PathVQA with 8 fine-tuned layers and $\rho = 10\%$. Removing the stability cost decreases Hscore* by 16.5 points, indicating that plasticity benefit alone is insufficient to identify desirable updates. Among the two stability terms, the dynamic component contributes more than the static curvature prior. Replacing the harmonic mean with a ratio reduces Avg by 0.54 points, suggesting that balancing plasticity and stability is more effective than favoring either side. Selection is also important: fixing the initial mask decreases Hscore* by 28.8 points, while random selection and utility scaling both underperform dynamic top-ρ selection, highlighting the value of re-evaluating candidate updates throughout optimization. Finally, different Fisher estimators and anchor sizes yield only minor variations, suggesting that SPUR is relatively insensitive to the exact curvature approximation. Additional results and implementation details are provided in Appendix H.

Table 3: **Ablation** on NVILA-Lite-2B / PathVQA with 8 fine-tuned layers and $\rho = 10\%$.

Variant	Avg	Hscore	Hscore*
SPUR (full)	**67.01**	**66.69**	**85.74**
Stability Cost C			
benefit only (no C)	66.14	65.91	69.29
w/o static curvature F^{sta}	66.82	66.52	82.82
w/o dynamic term $\mathbf{r}_t$	66.29	66.03	74.24
Utility			
ratio $\hat{B}/(\hat{C} + \epsilon)$	66.47	65.86	82.39
Selection			
per-tensor top-ρ (not global)	66.91	66.59	84.39
mask fixed after step 1	65.34	65.12	57.04
random set, same budget	66.37	65.89	81.05
utility-scaled, no top-ρ	65.60	65.46	52.34
Curvature			
dynamic Fisher (every 10 steps)	66.93	66.60	85.33
static Fisher, 32 anchors	66.98	66.66	85.42
static Fisher, 64 anchors	66.95	66.63	85.03

3.4 VISUALIZATION

SPUR's Decisions Follow the Updates, Not the Parameters. Fig. 4 visualizes how SPUR's selected update coordinates evolve throughout fine-tuning. Fig. 4(a)(b) track a fixed sample of parameter coordinates and show that their associated candidate updates are selected at some steps but not others. Within the sampled coordinates, no update coordinate is selected throughout the entire trajectory, nor is any permanently excluded, whereas Model-Dowser uses a fixed trainable set. Fig. 4(c)(d) further quantify this temporal variation: between consecutive steps, SPUR replaces 71% of the selected set on LLaVA-1.5-7B and 78% on NVILA-Lite-2B, compared with 0% for a fixed mask and 80%/90% expected under random selection with the same budget. Across the full trajectory, updates associated with 98.6% and 99.1% of the sampled trainable parameters are selected at least once, while each sampled parameter's update is selected in only 19% and 11% of the steps on average, respectively. These results show that SPUR re-evaluates candidate updates throughout optimization rather than assigning parameters a fixed update status (Appendix C).

Where Fine-Tuning Moves the Model. Fig. 5 visualizes the optimization trajectories together with their target loss and behavioral deviation. In Fig. 5 (a)(b), standard fine-tuning (FT) moves substantially farther from the pre-trained checkpoint, reaching a parameter-space distance of 33.0 and a KL divergence of 785. In contrast, SPUR remains much closer, with a distance of 8.7 and a KL divergence of 50, while achieving the same target loss. Fig. 5 (c) compares the final checkpoints across methods. Despite similar target losses, their KL divergences differ substantially: 785 for FT, 450 for SPIDER, 119 for Model-Dowser, and 50 for SPUR. Overall, these results show that comparable target-task optimization can be reached with markedly different parameter displacement and behavioral deviation from the pre-trained model (Appendix D).

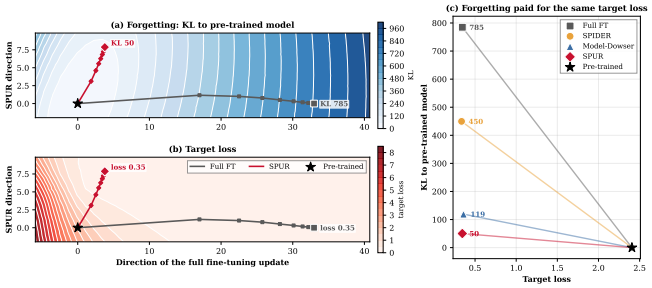


Figure 5: **Fine-tuning loss landscape** (LLaVA-1.5-7B, 8 layers, DriveLM). (a,b) KL divergence and target loss on the plane spanned by the pre-trained model and the endpoints of standard FT and SPUR; markers every 40 steps. (c) Target loss versus KL divergence at each method's checkpoint.

4 DISCUSSION

Our results support an update-centric view of catastrophic forgetting, motivating per-step rather than fixed parameter-level decisions. Several limitations remain. The text-only anchors used for static Fisher estimation may not fully reflect multimodal sensitivity, and the diagonal curvature approximation ignores interactions among jointly applied updates. SPUR also incurs additional training-time computation for estimating stability, despite adding no trainable parameters or inference overhead. Future work could explore multimodal anchors, structured curvature approximations, and more efficient stability estimators.

5 CONCLUSION

We present SPUR, an update-centric framework for mitigating catastrophic forgetting during MLLM adaptation. Rather than assigning persistent importance to parameters, SPUR evaluates each candidate update at the moment it is proposed, jointly accounting for its contribution to downstream learning and its potential disruption of pre-trained behavior. This per-step perspective allows the update pattern to evolve throughout optimization as the model changes. Extensive experiments across multiple MLLMs, downstream tasks, and fine-tuning depths show that SPUR consistently achieves stronger stability–plasticity trade-offs than the compared state-of-the-art baselines. Our results suggest that controlling which updates are applied during optimization provides a useful alternative to deciding which parameters should be protected.

AI USE STATEMENT

In this work, we used generative AI tools for language editing, including improving grammar, clarity, and presentation of the manuscript. All technical claims, mathematical formulations, experimental results, and references were manually checked by the authors. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or other significant ethical concerns.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed method, experimental setup, and evaluation protocol in the main paper and appendices. Implementation details, additional ablations, hyperparameter settings, and analysis procedures are provided in the appendix to facilitate reproducibility.

REFERENCES

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*, 37:57817–57840, 2024.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *AISTATS*, pp. 3762–3773, 2020.

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*, pp. 6904–6913, 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*, pp. 3608–3617, 2018.

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

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, pp. 8320–8329. IEEE, 2021.

Wenke Huang, Jian Liang, Xianda Guo, Yiyang Fang, Guancheng Wan, Xuankun Rong, Chi Wen, Zekun Shi, Qingyun Li, Didi Zhu, et al. Keeping yourself is important in downstream tuning multimodal large language model. *arXiv preprint arXiv:2503.04543*, 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 model fine-tuning. In *ICML*, pp. 25486–25499, 2025b.

Wenke Huang, Jian Liang, Guancheng Wan, Didi Zhu, He Li, Jiawei Shao, Mang Ye, Bo Du, and Dacheng Tao. Be confident: Uncovering overfitting in mllm multi-task tuning. In *ICML*, 2025c.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, pp. 6693–6702, 2019.

Hyeontaek Hwang, Nguyen Dinh Son, and Daeyoung Kim. Model-dowser: data-free importance probing to mitigate catastrophic forgetting in multimodal large language models. 2026.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

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*, pp. 19730–19742, 2023a.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *EMNLP*, pp. 292–305, 2023b.

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*, pp. 26170–26180, 2025.

Jian Liang, Wenke Huang, Xianda Guo, Guancheng Wan, Bo Du, and Mang Ye. Lorasculpt: Harmonious low-rank adaptation for multimodal large language models. *TPAMI*, 2026.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, pp. 34892–34916, 2023.

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*, pp. 216–233. Springer, 2024.

Zhijian Liu, Ligeng Zhu, Baifeng Shi, Zhuoyang Zhang, Yuming Lou, Shang Yang, Haocheng Xi, Shiyi Cao, Yuxian Gu, Dacheng Li, et al. Nvila: Efficient frontier visual language models. In *CVPR*, pp. 4122–4134, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 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. *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*, 35:2507–2521, 2022.

Nikhil Shivakumar Nayak, Krishnateja Killamsetty, Ligong Han, Abhishek Bhandwaldar, Prateek Chanda, Kai Xu, Oleg Silkin, Mustafa Eyceoz, Hao Wang, Aldo Pareja, and Akash Srivastava. Sculpting subspaces: Constrained full fine-tuning in LLMs for continual learning. In *ICLR*, 2026.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. In *ICML*, pp. 27011–27033, 2023.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. Drivelm: Driving with graph visual question answering. In *ECCV*, pp. 256–274. Springer, 2024.

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*, pp. 8309–8318, 2019.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. Lamol: Language modeling for lifelong language learning. In *ICLR*, 2020.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *ICLR*, volume 2024, pp. 4942–4964, 2024.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, pp. 4566–4575, 2015.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Orthogonal subspace learning for language model continual learning. In *EMNLP*, pp. 10658–10671, 2023.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*, pp. 631–648. Springer, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *CVPR*, pp. 139–149. IEEE, 2022b.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025a.

Junda Wu, Yuxin Xiong, Xintong Li, Yu Xia, Ruoyu Wang, Yu Wang, Tong Yu, Sungchul Kim, Ryan A Rossi, Lina Yao, et al. Mitigating visual knowledge forgetting in mllm instruction-tuning via modality-decoupled gradient descent. In *EMNLP*, 2025b.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *NeurIPS*, 36:7093–7115, 2023.

Jinglong Yang, Jiaxuan He, Wenjian Huang, Zhan Zhuang, and Jianguo Zhang. Curvature-guided mixing for mllm adaptation. In *ECCV*, pp. 342–358. Springer, 2026.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*, 2: 67–78, 2014.

Linyan 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*, volume 2, pp. 21, 2024.

Zeping Yu and Sophia Ananiadou. Locate-then-merge: Neuron-level parameter fusion for mitigating catastrophic forgetting in multimodal LLMs. In *EMNLP*, pp. 7065–7078, 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*, pp. 202–227, 2024.

Hongbo Zhao, Fei Zhu, Haiyang Guo, Meng Wang, Rundong Wang, Gaofeng Meng, and Zhaoxiang Zhang. Mllm-cl: Continual learning for multimodal large language models. *arXiv preprint arXiv:2506.05453*, 2025.

Didi Zhu, Zhongyisun Sun, Zexi Li, Tao Shen, Ke Yan, Shouhong Ding, Chao Wu, and Kun Kuang. Model tailor: Mitigating catastrophic forgetting in multi-modal large language models. In *ICML*, pp. 62581–62598, 2024.

In this appendix, we provide additional discussion, implementation details, analyses, and results supporting the main paper.

- **Appendix A: More Discussions of Related Work.**
- **Appendix B: More Details of Experiments Protocols.**
- **Appendix C: More Details of Selection and Update Visualization.**
- **Appendix D: More Details of Loss-Landscape Visualization.**
- **Appendix E: More Details of Static Curvature.**
- **Appendix F: More Details of Fine-tuning Depth.**
- **Appendix G: More Details of Top-ρ Budget.**
- **Appendix H: More Details of Full Ablation Results.**
- **Appendix I: More Results from Qwen2.5-VL-3B.**
- **Appendix I.1: More Results from More Target Datasets.**

A More Discussions of Related Work

Table 4: Comparison of major paradigms for mitigating catastrophic forgetting. "Extra Arch.": introduces task-specific modules or trainable parameters. "Replay": relies on previous or generated data. "Merging": operates on the fine-tuned model after adaptation. "Train-time": regulates adaptation during optimization. "Param. Imp.": relies on parameter-level importance or sensitivity. "Update Val.": evaluates the candidate update proposed at the current step.

Paradigm	Extra Arch.	Replay	Merging	Train-time	Param. Imp.	Update Val.
Parameter optimization	✗	✗	✗	✓	✓	✗
Model merging	✗	✗	✓	✗	✓	✗
Replay	✗	✓	✗	✓	✗	✗
Regularization	✗	✗	✗	✓	✓	✗
Architecture / PEFT	✓	✗	✗	✓	✗	✗
Gradient / subspace regulation	✗	✗	✗	✓	✗	✗
SPUR	✗	✗	✗	✓	✗	✓

The main paper focuses on the two families most closely related to SPUR: parameter optimization during adaptation and post-hoc model merging. Table 4 places these approaches within a broader taxonomy of forgetting-mitigation strategies. Replay and regularization control the information preserved during adaptation; architecture-based methods alter the parameterization through which new knowledge is acquired; and gradient or subspace methods constrain the directions along which optimization proceeds. SPUR differs primarily in its decision object: rather than deciding which data, parameters, modules, or directions should be protected or modified, it evaluates the concrete candidate update proposed at the current optimization step.

Parameter Optimization. This family regulates adaptation through parameter-level importance or sensitivity. SPIDER (Huang et al., 2025b) allocates updates using importance derived from the pre-trained model and downstream gradients, while Model-Dowser (Hwang et al., 2026) identifies parameters that are less critical to pre-trained generalization and selectively adapts them. Although their scoring mechanisms differ, both attach the adaptation decision to the parameter rather than to the particular optimizer proposal at the current step. Consequently, different candidate updates to the same parameter are governed by the same parameter-level criterion.

Model Merging. Model-merging methods intervene after downstream adaptation. Model-Tailor (Zhu et al., 2024) constructs a sparse patch of the fine-tuned model using salience and sensitivity; TIES (Yadav et al., 2023) resolves conflicts among parameter differences; Locate-then-Merge (Yu & Ananiadou, 2025) performs fine-grained neuron-level fusion; and CGM (Yang et al.,

Table 5: Evaluation protocols for the eight general-capability benchmarks and six downstream target tasks, including task type, evaluation split and sample size, answer format, and official resource. The last four target tasks use the 10k training subsets and the test files released by Model-Dowser (Hwang et al., 2026).

Benchmark	Task	Evaluation Split / # Samples	Answer Format	URL
VQAv2	Open-ended VQA	Val subset / 3,000	Short free-form answer	Link
GQA	Compositional reasoning VQA	Testdev-balanced / 12,578	Short answer	Link
VizWiz	VQA on images from blind users	Val / 4,319	Short answer / "unanswerable"	Link
ScienceQA	Science multiple choice	Test / 4,241	Option letter	Link
TextVQA	Scene-text VQA	Val / 5,000	Short answer	Link
POPE	Object hallucination detection	Random / Popular / Adversarial / 8,910	Yes / No	Link
MMBench	Multidimensional multiple choice (English)	Dev / 4,377 questions (1,173 groups)	Option letter	Link
MMBench-CN	Multidimensional multiple choice (Chinese)	Dev / 4,329 questions (1,160 groups)	Option letter	Link
Downstream target tasks				
PathVQA	Pathology VQA (open + closed)	Train 19,654 / Test 6,719 (858 images)	Short free-form answer; exact match	Link
DriveLM	Driving-scene QA (six-camera stitched images)	Train 60,000 / Test 10,000	Free-form answer; containment accuracy	Link
COCO-Caption	Image captioning	Train 10,000 / Test 5,000	One-sentence caption; CIDEr	Link
Flickr30k	Image captioning	Train 10,000 / Test 1,000	One-sentence caption; CIDEr	Link
IconQA	Abstract-diagram multiple choice	Train 10,000 / Test 6,316	`Answer: X`; exact match	Link
ImageNet-R	Rendition classification as VQA (200 classes)	Train 10,000 / Test 3,000	Single class word; exact match	Link

2026) combines the pre-trained and fine-tuned models using curvature information. These approaches can restore part of the pre-trained capability without altering the original fine-tuning procedure, but their decisions are made after individual updates have already accumulated into the final parameter displacement.

Replay and Regularization. Replay-based methods preserve previous knowledge by revisiting stored or generated examples, as in LAMOL (Sun et al., 2020). Regularization-based methods instead constrain adaptation toward a previous solution, for example by penalizing changes to parameters considered important for earlier knowledge (Kirkpatrick et al., 2017). Both provide established mechanisms for balancing stability and plasticity, but their control signals are defined through retained data or parameter-level constraints rather than the utility of the candidate update currently proposed by the optimizer.

Architecture-Based and Parameter-Efficient Adaptation. Another family changes the parameterization through which new knowledge is stored. Prompt-based methods allocate learnable prompts to different tasks (Wang et al., 2022b;a), while parameter-efficient fine-tuning (PEFT) approaches, such as adapters and LoRA, introduce lightweight trainable parameters instead of modifying the entire backbone. For language and multimodal models, O-LoRA, MoELoRA, and LoRASculpt further orthogonalize, route, or sparsify low-rank adaptation subspaces to reduce interference with existing capabilities (Wang et al., 2023; Chen et al., 2024; Liang et al., 2025). Their primary mechanism therefore lies in structuring where and how adaptation is represented, rather than evaluating individual optimizer proposals.

Gradient and Subspace Regulation. A related line of work controls the direction of adaptation. Orthogonal-gradient and subspace methods project gradients away from directions associated with interference (Farajtabar et al., 2020; Nayak et al., 2026), while modality-decoupled gradient descent modifies gradient directions to alleviate visual forgetting in MLLMs (Wu et al., 2025b). These methods regulate gradients or optimization subspaces, whereas SPUR evaluates the utility of the concrete candidate update produced by the optimizer at the current training state.

From Parameters to Updates (Our Method). SPUR shifts the unit of decision to the candidate update. At each optimization step, it evaluates each proposed update according to its plasticity benefit and stability cost, and selectively applies only high-utility coordinates. The same parameter may therefore be updated at one step and left unchanged at another. Conceptually, SPUR changes the question from where adaptation should occur to whether the current candidate update is worth applying. This update-centric view is complementary to many existing strategies and could, in principle, be combined with replay-, architecture-, or gradient-based mechanisms.

B More Details of Experiments Protocols

General-Capability Evaluation. To measure the preservation of general multimodal capabilities, we evaluate on eight benchmarks: VQAv2 (Goyal et al., 2017), GQA (Hudson & Manning, 2019), VizWiz (Gurari et al., 2018), ScienceQA (Lu et al., 2022), TextVQA (Singh et al., 2019), POPE (Li et al., 2023b), and MMBench/MMBench-CN (Liu et al., 2024). Following the official implementation of CGM (Yang et al., 2026), we use the same preprocessing and evaluation splits for all methods: a 3,000-example subset of VQAv2 validation, the balanced GQA test-dev split (12,578 examples), VizWiz validation (4,319), the full ScienceQA test set (4,241), TextVQA validation (5,000), the random, popular, and adversarial POPE subsets (8,910 in total), and the MMBench/MMBench-CN development sets (4,377 questions in 1,173 groups and 4,329 questions in 1,160 groups, respectively).

These benchmarks cover open-ended VQA, compositional and scientific reasoning, scene-text understanding, object hallucination detection, and multidimensional multimodal reasoning. Their expected output formats also differ: VQAv2, GQA, VizWiz, and TextVQA use short free-form answers; ScienceQA, MMBench, and MMBench-CN use option letters; and POPE uses *yes/no* responses. Table 5 summarizes the complete evaluation protocol.

Downstream Target Tasks. We consider six target tasks from distinct domains: two are used throughout the main paper, and four further tasks are taken from the setting of ModelDowser (Hwang et al., 2026) so that our results can be read against the setting that method was designed for (Appendix I.1).

PathVQA (He et al., 2020) evaluates visual question answering over pathology images. We fine-tune on its training split containing 19,654 question–answer pairs over 2,599 images and report exact-match accuracy on the test split containing 6,719 pairs over 858 images, including both open-ended and closed *yes/no* questions.

DriveLM (Sima et al., 2024) evaluates question answering over multi-camera driving scenes. We follow the data formulation of MLLM-CL (Zhao et al., 2025), where the six camera views of each frame are stitched into a single image while the questions and object coordinates are kept unchanged. The resulting dataset contains 60,000 training and 10,000 test questions. Following MLLM-CL, we evaluate with containment accuracy, where a prediction is considered correct if it contains the reference answer, using a maximum decoding length of 128 tokens.

COCO-Caption (Chen et al., 2015) and **Flickr30k** (Young et al., 2014) evaluate image captioning. We use the $10{,}000$-example training subsets released by (Hwang et al., 2026) and their test files with $5{,}000$ and $1{,}000$ images respectively, and report CIDEr (Vedantam et al., 2015).

IconQA (Lu et al., 2021) evaluates multiple-choice reasoning over abstract diagrams. We use the released $10{,}000$-example training subset and its $6{,}316$-question test file. The reference answers are of the form `Answer: A`, and the released evaluation code compares the whole generated string to the reference, case-insensitively and without option extraction; we keep this rule unchanged.

ImageNet-R (Hendrycks et al., 2021) evaluates object recognition on renditions (art, cartoons, sketches, and similar) posed as visual question answering over 200 classes. We use the released $10{,}000$-example training subset and its $3{,}000$-image test file, and report case-insensitive exact-match accuracy of the predicted class word.

All six tasks are fine-tuned with the same recipe within a backbone: the last N decoder blocks and the projector, AdamW at learning rate $1\mathrm{e}{-4}$ and global batch size 128. PathVQA and DriveLM follow the schedules of the main paper; for the four 10k subsets we train LLaVA-1.5-7B for 2 epochs (156 steps) and NVILA-Lite-2B for 3 epochs (234 steps).

Baseline Implementations. For reproducibility, Table 6 reports the venue and public implementation used for each comparison method. We use the official implementation of each baseline.

Table 6: Public implementations of the comparison methods used in our experiments.

Assets		Venue	URL
Compared Methods	DARE	ICML 2024	https://github.com/yule-BUAA/MergeLM
	Grafting	ICML 2023	https://github.com/abhishekpanigrahi1996/Skill-Localization-by-grafting
	Model-Tailor	ICML 2024	https://github.com/didizhu-judy/Model-Tailor
	SPIDER	ICML 2025	https://github.com/WenkeHuang/SPIDER
	Model-Dowser	ICML 2026	https://github.com/nguyendinhson-kaist/Model-Dowser
	CGM / CGM*	ECCV 2026	https://github.com/zzsyjl/CGM-ECCV-2026

C More Details of Selection and Update Visualization

Logging Protocol. During fine-tuning, we record the binary top-ρ selection mask at every optimization step for a fixed random sample of 20,000 parameter coordinates from each trainable tensor, totaling approximately 2M coordinates. The masks are packed as bits and flushed every 50 steps. The sample is drawn once before training, so the same parameter coordinates are tracked throughout the entire optimization trajectory. These logged coordinates are used for both the visualizations and the aggregate selection statistics reported below. For Model-Dowser, we record its trainable parameter set only once, as the set remains fixed throughout fine-tuning.

Fig. 4 (a)(b). Each row represents a sampled parameter coordinate and each column an optimization step; a red cell indicates that the candidate update associated with that coordinate belongs to the top-ρ set at that step. For visualization, we randomly select 30 rows for LLaVA-1.5-7B and 27 rows for NVILA-Lite-2B from the full log and sort them by selection frequency. The strip on the right indicates whether the corresponding parameter coordinate belongs to Model-Dowser's fixed trainable set. The resulting patterns illustrate that SPUR does not assign a persistent update status to individual coordinates: the candidate update associated with the same parameter coordinate can be selected at some steps and excluded at others.

Fig. 4 (c)(d). We define the replacement rate at step t as

$$R_t = \frac{|S_t \setminus S_{t-1}|}{|S_t|}, \tag{20}$$

where S_t denotes the set of candidate update coordinates selected at step t. The replacement rate is zero for a fixed set. For a set independently sampled uniformly at every step with budget ρ, its expected value is $1 - \rho$, corresponding to 0.80 for $\rho = 20\%$ and 0.90 for $\rho = 10\%$.

SPUR exhibits high but non-random turnover. Its utility scores have only 0.03 step-to-step autocorrelation, while an update coordinate selected at step t remains more likely than chance to be selected again at step $t + 1$. Thus, the selected update set changes substantially as the candidate updates evolve, while still retaining temporal structure across adjacent optimization steps. The lower replacement rate during the first few steps is consistent with the early evolution of the optimizer state.

Across the full trajectory, candidate updates associated with a large fraction of the sampled trainable coordinates are selected at least once, even under small update budgets (Table 7). At the same time, the update associated with each coordinate is selected only during a subset of optimization steps. For example, 98.6% and 99.1% of the sampled coordinates are selected at least once on PathVQA for LLaVA-1.5-7B and NVILA-Lite-2B, respectively, despite using per-step budgets of only 20% and 10%. On DriveLM, the corresponding coverage remains 87.1% and 75.4% under substantially smaller budgets of 4% and 2%. These results further show that SPUR does not partition parameters into permanently active and inactive subsets. Instead, the candidate update associated with each parameter coordinate is re-evaluated throughout fine-tuning according to its current utility.

D More Details of Loss-Landscape Visualization

Settings. We fine-tune LLaVA-1.5-7B on DriveLM for one epoch (468 optimization steps), with the visual projector and the last 8 decoder layers trainable. We compare four methods: standard

Table 7: Selection statistics over the full fine-tuning trajectory with 8 trainable language layers. "Replaced" denotes the fraction of the selected update set replaced between consecutive steps; "Random" gives the expected value $1 - \rho$ under independent random top-ρ selection; "Ever selected" denotes the fraction of sampled trainable parameter coordinates whose associated candidate updates are selected at least once.

Setting	ρ	Replaced	Random	Ever selected
LLaVA-1.5-7B · PathVQA	20%	71%	80%	98.6%
NVILA-Lite-2B · PathVQA	10%	78%	90%	99.1%
LLaVA-1.5-7B · DriveLM	4%	85%	96%	87.1%
NVILA-Lite-2B · DriveLM	2%	89%	98%	75.4%

fine-tuning, SPIDER, Model-Dowser, and SPUR ($\rho = 4\%$). All runs use the same data order, learning-rate schedule, and number of optimization steps. We save the trainable parameters every 40 steps, yielding 11 intermediate snapshots together with the final checkpoint for each run.

Visualization Plane. Let θ_0 denote the pre-trained values of the trainable parameters, and let Δ_{FT} and Δ_{SPUR} denote the final parameter displacements produced by standard fine-tuning and SPUR, respectively. We construct an orthonormal basis

$$\mathbf{u}_1 = \frac{\Delta_{\mathrm{FT}}}{\|\Delta_{\mathrm{FT}}\|}, \qquad \mathbf{u}_2 = \frac{\tilde{\mathbf{u}}_2}{\|\tilde{\mathbf{u}}_2\|}, \tag{21}$$

where

$$\tilde{\mathbf{u}}_2 = \Delta_{\mathrm{SPUR}} - \left(\Delta_{\mathrm{SPUR}}^{\top}\mathbf{u}_1\right)\mathbf{u}_1. \tag{22}$$

A coordinate (a, b) therefore represents

$$\theta(a, b) = \theta_0 + a\mathbf{u}_1 + b\mathbf{u}_2, \tag{23}$$

so that the final checkpoints of standard fine-tuning and SPUR lie exactly in this plane. Distances are Euclidean norms over the trainable parameters, and the two axes therefore share the same scale. Intermediate checkpoints are shown through their orthogonal projections onto this plane.

SPIDER and Model-Dowser are not visualized as trajectories in the plane, because only 31% and 4%, respectively, of their final displacement is captured by this two-dimensional subspace. We therefore compare them using quantities measured directly from their original checkpoints rather than from their projections.

Grid Evaluation. We evaluate a 41×41 grid covering the standard fine-tuning and SPUR trajectories with an additional 25% margin. At each grid point, we instantiate the corresponding model and measure two quantities: (i) behavioral deviation from the pre-trained model using $D_{\mathrm{KL}}(p_{\theta_0} \| p_\theta)$ on the general-capability evaluation data; and (ii) the DriveLM target loss. Both quantities are computed with teacher forcing on the reference answers. The KL divergence is summed over answer tokens and then averaged over examples.

Trajectory Measurements. Table 8 reports, for every saved checkpoint, the parameter displacement $\|\theta_t - \theta_0\|$, KL divergence from the pre-trained model, and target loss. Unlike the two-dimensional visualization, all quantities in the table are measured directly on the original checkpoints, without projection. The final row corresponds to the points shown in Fig. 5(c).

Several patterns are evident. First, all four methods reduce the target loss rapidly. By step 40, it falls from 2.41 to 0.45–0.53, and from approximately step 280 onward all four runs remain within 0.34–0.37. In contrast, their behavioral deviation differs substantially from the early stages of training. At step 40, standard fine-tuning already reaches a KL divergence of 725, compared with 383 for SPIDER, 102 for Model-Dowser, and 39 for SPUR.

Second, similar target losses correspond to markedly different parameter displacements and behavioral deviations. At the final checkpoint, standard fine-tuning, SPIDER, Model-Dowser, and SPUR reach target losses of 0.35, 0.34, 0.36, and 0.35, respectively, while their KL divergences are 785,

Table 8: **Trajectory measurements underlying Fig. 5** (LLaVA-1.5-7B, last 8 decoder layers, DriveLM). For each checkpoint, we report the displacement from the pre-trained model $\|\theta_t - \theta_0\|$, KL divergence from the pre-trained model on the general-capability evaluation data, and the DriveLM target loss. All quantities are measured directly on the original checkpoints rather than on their two-dimensional projections. The pre-trained model has KL divergence 0 and target loss 2.41.

Step	Std. FT			SPIDER			Model-Dowser			SPUR		
	$\|\Delta\|$	KL	Loss	$\|\Delta\|$	KL	Loss	$\|\Delta\|$	KL	Loss	$\|\Delta\|$	KL	Loss
40	21.9	725	0.45	14.7	383	0.45	15.4	102	0.53	5.2	39	0.46
80	26.1	770	0.40	17.2	439	0.41	19.1	102	0.43	6.4	42	0.40
120	28.2	804	0.38	18.9	443	0.38	20.9	100	0.39	7.1	43	0.38
160	29.9	778	0.38	20.2	426	0.37	22.1	103	0.38	7.7	44	0.36
200	31.1	797	0.38	21.1	459	0.37	22.9	107	0.38	8.1	45	0.36
240	32.0	799	0.37	21.7	493	0.37	23.5	112	0.37	8.3	47	0.36
280	32.5	798	0.36	22.1	475	0.35	23.9	113	0.37	8.5	50	0.35
320	32.8	812	0.37	22.2	484	0.36	24.1	118	0.37	8.6	48	0.36
360	32.9	797	0.36	22.2	474	0.36	24.3	118	0.37	8.7	51	0.36
400	33.0	789	0.35	22.1	470	0.35	24.3	119	0.36	8.7	50	0.35
440	33.0	784	0.35	21.9	458	0.34	24.3	119	0.36	8.7	50	0.35
Final (468)	33.0	785	0.35	21.7	450	0.34	24.3	119	0.36	8.7	50	0.35

450, 119, and 50. Their final parameter displacements likewise differ substantially, reaching 33.0, 21.7, 24.3, and 8.7, respectively.

Finally, once the target loss has largely stabilized, these differences persist throughout the remainder of training. Standard fine-tuning remains far from the pre-trained checkpoint in both parameter distance and behavioral deviation, whereas SPUR remains substantially closer and changes only modestly during the later stages. These measurements support the visualization in Fig. 5: comparable target optimization can be reached through trajectories that differ substantially in both parameter displacement and deviation from pre-trained behavior.

E More Details of Static Curvature

The stability cost in Sec. 2.2 retains the second-order form of the KL expansion, but replaces the curvature $\mathbf{H}_t$ at the current model with a static diagonal Fisher $\mathbf{F}^{\mathrm{sta}}$ computed once at the pre-trained model. In this section, we describe its construction and motivate this approximation.

Construction. Computing $\mathbf{F}^{\mathrm{sta}}$ requires neither downstream nor pre-training data; it uses only the pre-trained model and a fixed random seed.

Self-generated anchors. We construct $A = 8$ text-only anchor sequences. Each sequence begins with the beginning-of-sequence token followed by five token IDs drawn uniformly from the vocabulary, excluding the first 100 indices reserved for special and rarely used tokens. Using a fixed random seed, the frozen pre-trained language model then greedily continues each prefix for 64 tokens, yielding anchors of length $L = 70$. The random prefix keeps the anchors task-independent, while greedy continuation moves them toward contexts favored by the model's own predictive distribution. The resulting anchors are therefore model-induced rather than sampled from an external corpus. They are fed directly to the language model, bypassing the vision tower and projector. Because the prefixes are generated deterministically and the continuations are greedy, every rank reconstructs the same anchor set without communication, and the resulting set remains fixed throughout fine-tuning.

Fisher at the pre-trained point. Given the anchor set $\mathcal{S}_{\mathrm{self}} = \{s^{(1)}, \ldots, s^{(A)}\}$, we estimate the diagonal Fisher of the pre-trained predictive distribution. For each of $M = 16$ Monte Carlo draws, we sample outputs from the pre-trained model, $\hat{y}^{(m)}_{a,t} \sim p_{\theta_0}(\cdot \mid s^{(a)}_{<t})$, and back-propagate the mean negative log-likelihood of the sampled tokens:

$$\mathcal{L}^{(m)}(\theta) = -\frac{1}{A}\sum_{a=1}^{A}\sum_{t=1}^{L-1}\log p_\theta\left(\hat{y}^{(m)}_{a,t} \mid s^{(a)}_{<t}\right), \qquad F^{\mathrm{sta}}_i = \frac{1}{M}\sum_{m=1}^{M}\left(\frac{\partial \mathcal{L}^{(m)}}{\partial \theta_i}\bigg|_{\theta_0}\right)^2. \tag{24}$$

Table 9: **Static versus dynamic curvature** on NVILA-Lite-2B / PathVQA with the last 8 layers fine-tuned and $\rho = 10\%$. "Dynamic" re-estimates the diagonal Fisher at the current model every 10 steps using the 8-example probe batch; "Static" uses $\mathbf{F}^{\mathrm{sta}}$ estimated once at θ_0 from self-generated anchors with 16 Monte Carlo draws. "Extra passes" counts the per-sample backward passes used for curvature, incurred during training for the dynamic variant and once before training for the static variants. "Time" reports the wall-clock time of the complete fine-tuning run in GPU-hours. Parentheses indicate the change relative to the full method.

Curvature	Avg	Hscore	Hscore*	Extra passes	Time (h)
None (first-order only)	66.82 (-0.19)	66.52 (-0.17)	82.82 (-2.92)	–	1.49
Dynamic, every 10 steps	66.93 (-0.08)	66.60 (-0.09)	85.33 (-0.41)	46×8	1.52
Static, 8 anchors (ours)	67.01	66.69	85.74	8×16	1.52
Static, 32 anchors	66.98 (-0.03)	66.66 (-0.03)	85.42 (-0.32)	32×16	1.53
Static, 64 anchors	66.95 (-0.06)	66.63 (-0.06)	85.03 (-0.71)	64×16	1.51

Because the targets are sampled from the model's own predictive distribution rather than taken directly from the anchor sequences, Eq. 24 estimates the model Fisher rather than the empirical Fisher. Squared gradients are accumulated in fp32, and the resulting estimate is computed once before fine-tuning and reused unchanged at every step. It covers the language-model parameters within the fine-tuned depth; the visual projector receives only the dynamic first-order stability term.

Why Static Curvature? The primary motivation is efficiency. Re-estimating curvature at the current model requires an additional sampled forward–backward pass beyond the probe used to compute $\mathbf{r}_t$, and would therefore introduce substantial per-step overhead if performed at every update. Table 9 evaluates a more economical alternative that re-estimates the diagonal Fisher every ten steps using the probe batch. Despite requiring more curvature-related backward passes during training than the static estimate requires once before training, the dynamic variant does not improve any of the three performance metrics. Moreover, mini-batch curvature estimates are inherently noisy and reflect only directions activated by the current batch. In contrast, $\mathbf{F}^{\mathrm{sta}}$ is computed once before fine-tuning and reused throughout training, avoiding repeated curvature estimation. Increasing the number of anchors from 8 to 32 or 64 also yields no further improvement, indicating that eight anchors are sufficient in our setting.

The approximation also admits a natural interpretation. A second-order expansion of the retention objective around the pre-trained weights gives

$$\mathrm{KL}(p_{\theta_0} \| p_{\theta_0+\Delta}) = \tfrac{1}{2}\,\Delta^\top \mathbf{F}(\theta_0)\Delta + \mathcal{O}\big(\|\Delta\|^3\big)\,, \qquad \nabla_\theta \,\mathrm{KL}\big|_{\theta_0} = \mathbf{0}, \quad \nabla^2_\theta \,\mathrm{KL}\big|_{\theta_0} = \mathbf{F}(\theta_0), \tag{25}$$

so $\mathbf{F}^{\mathrm{sta}}$ captures the local curvature of the retention objective around the pre-trained solution. We therefore keep this sensitivity prior fixed, while recomputing the first-order term $\mathbf{r}_t$ at every step to capture the state-dependent effect of the current candidate update. The two terms play complementary roles: the static Fisher captures pre-trained sensitivity, whereas $\mathbf{r}_t$ provides dynamic, update-specific information.

F More Details of Fine-tuning Depth

Tables 10–13 report the complete results for LLaVA-1.5-7B and NVILA-Lite-2B on both target tasks across all fine-tuning depths (last 4, 8, 12, 16, and 20 language layers). We report all eight general benchmarks, target-task performance, and the three aggregate metrics. The main paper reports the 8-layer setting in Table 1 and summarizes the depth sweep using Hscore* in Fig. 2.

Three trends are consistent across the four backbone–task combinations. First, standard fine-tuning exhibits increasingly severe forgetting as more layers are unfrozen. From 4 to 20 trainable language layers, its Pre-Avg decreases by 16.6–37.2 points across the four settings, whereas target-task performance changes by at most 1.8 points. Increasing the fine-tuning depth therefore provides little additional target adaptation while substantially reducing general-capability retention.

Second, the relative behavior of the baselines changes with depth. SPIDER is competitive under shallow fine-tuning but degrades as more layers are unfrozen, whereas Model-Dowser becomes

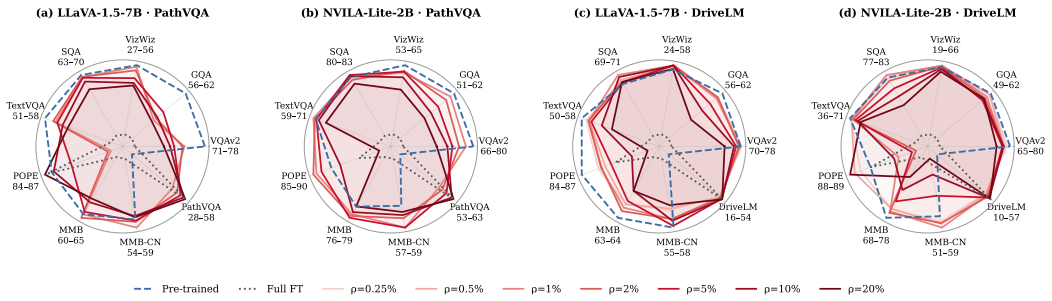


Figure 6: **Per-benchmark effect of the update budget** ρ with 8 fine-tuned layers. Each axis corresponds to one benchmark (eight general benchmarks and the target task) and is scaled to the range of the plotted values, shown below the axis label. Polygons from light to dark red denote SPUR with $\rho = 0.25, 0.5, 1, 2, 5, 10, 20\%$ where available; the PathVQA sweeps start at $\rho = 1\%$. The dashed blue polygon denotes the pre-trained model and the dotted gray polygon standard fine-tuning.

the strongest baseline in most 16–20-layer settings. This suggests that the relative effectiveness of different forgetting-mitigation strategies depends on the size of the trainable parameter space.

Third, SPUR achieves the highest Avg, Hscore, and Hscore* at every fine-tuning depth on both backbones and both target tasks, covering all 20 settings. Its advantage is generally larger under shallow fine-tuning: at 4 layers, its Hscore margin over Model-Dowser ranges from 1.1 to 5.5 points across the four backbone–task combinations. The margin narrows as the fine-tuning depth increases, ranging from 0.02 to 1.29 points at 20 layers. The closest Hscore* comparison occurs at 16 layers on LLaVA-1.5-7B/DriveLM, where SPUR scores 98.38 compared with 98.35 for Model-Dowser.

The preferred update budget also generally decreases with depth. Selecting ρ by Avg, with Hscore used only to break ties, gives 20/20/4/2/1% for LLaVA-1.5-7B/PathVQA, 10/4/4/2/0.25% for LLaVA-1.5-7B/DriveLM, 10/10/4/1/1% for NVILA-Lite-2B/PathVQA, and 10/2/4/2/1% for NVILA-Lite-2B/DriveLM at 4/8/12/16/20 layers, respectively. Thus, as the trainable parameter space grows, strong target adaptation can generally be maintained with a smaller fraction of the candidate update applied at each step. We report results up to 20 layers because fine-tuning 24 language layers exceeds the memory capacity of a single NVIDIA H200 under our configuration.

G More Details of the Top-ρ Budget

Fig. 3 in the main paper reports the aggregate metrics across different values of ρ, while Fig. 6 provides the corresponding per-benchmark view. Three observations emerge.

First, forgetting is highly non-uniform across benchmarks. Under standard fine-tuning, the largest drops occur on VizWiz and VQAv2 for LLaVA-1.5-7B/PathVQA (28.6 and 6.7 points), and on TextVQA and MMBench for NVILA-Lite-2B/DriveLM (35.3 and 10.4 points). As ρ increases, the degradation under SPUR remains concentrated on a subset of sensitive benchmarks. For example, at $\rho = 20\%$ on LLaVA-1.5-7B/PathVQA, the largest drops relative to the pre-trained model occur on VizWiz (8.4 points) and VQAv2 (3.9 points), followed by GQA (3.4 points) and TextVQA (2.7 points), while POPE and MMBench-CN remain nearly unchanged. Thus, increasing ρ does not reduce general capability uniformly across benchmarks.

Second, target-task performance saturates relatively early. On LLaVA-1.5-7B/PathVQA, target accuracy increases from 50.8 at $\rho = 1\%$ to 58.1 at $\rho = 20\%$, close to standard fine-tuning at 58.4. On DriveLM, $\rho = 1\%$ already reaches 52.2 compared with 53.9 for standard fine-tuning on LLaVA-1.5-7B, and 53.2 compared with 57.1 on NVILA-Lite-2B. Very small budgets, such as 0.25% and 0.5%, primarily sacrifice target-task performance while keeping most general benchmarks close to their pre-trained levels. Increasing the budget therefore provides additional target adaptation, but progressively exposes more of the general-capability degradation associated with fine-tuning.

Third, small budgets preserve most of the pre-trained general capability. At $\rho = 1\%$, Pre-Avg remains within approximately 1.2 points of the pre-trained model across all four settings. Increas-

ing ρ therefore traces a clear stability–plasticity trade-off: more candidate update coordinates are admitted, improving target adaptation while reducing retention on the most sensitive general benchmarks. The budgets selected in the main 8-layer experiments (20%, 10%, 4%, and 2%) lie in the region where target performance has largely saturated while general-capability degradation remains limited.

H More Details of Full Ablation Results

Settings. All variants in Table 3 and Table 14 modify exactly one component of SPUR, with all other settings held fixed. We use NVILA-Lite-2B fine-tuned on PathVQA with the last 8 language layers trainable, AdamW with a learning rate of 1×10^{-4}, a batch size of 128, and an update budget of $\rho = 10\%$.

SPUR Configuration. SPUR evaluates each coordinate of the candidate update using the plasticity benefit $[\mathbf{g}_t \odot \Delta\theta_t^c]_+$ and two stability terms, $[-\mathbf{r}_t \odot \Delta\theta_t^c]_+$ and $\frac{1}{2}\mathbf{F}^{\mathrm{sta}} \odot (\Delta\theta_t^c)^2$. The retention gradient $\mathbf{r}_t$ is recomputed at every optimization step using an 8-example probe batch, whereas $\mathbf{F}^{\mathrm{sta}}$ is the static anchor Fisher described in Appendix E, estimated once before fine-tuning from 8 anchors and 16 Monte Carlo draws.

The two stability terms are normalized independently within each module using the same min–max normalization as the plasticity benefit, and their maximum defines the final stability cost. Plasticity and stability are then reconciled using the harmonic utility. Selection is hard and global: all trainable coordinates are ranked jointly, only the top-ρ candidate update coordinates are applied, and the ranking is recomputed at every step. No coordinate is assigned a fixed update status in advance.

Variants.

- *Benefit only*: removes the stability cost and ranks coordinates using normalized plasticity benefit alone.
- *W/o static curvature*: retains only the dynamic first-order stability term $[-\mathbf{r}_t \odot \Delta\theta_t^c]_+$.
- *W/o dynamic term*: retains only the static curvature term $\frac{1}{2}\mathbf{F}^{\mathrm{sta}} \odot (\Delta\theta_t^c)^2$; no retention probe is required.
- *Ratio*: replaces the harmonic utility with $\hat{\mathbf{B}}/(\hat{\mathbf{C}} + \epsilon)$.
- *Per-tensor top-ρ*: applies the update budget independently within each tensor instead of globally, so that exactly a ρ fraction of the coordinates in each tensor receive their candidate updates.
- *Mask fixed after step 1*: computes the selection mask at the first optimization step and keeps it fixed for the remainder of training.
- *Random set, same budget*: samples a fresh uniformly random subset of the same size at every step, preserving the update budget without utility-based selection.
- *Utility-scaled, no top-ρ*: removes hard top-ρ selection and instead scales every candidate update by its normalized utility, with per-tensor renormalization to match the average AdamW step magnitude.
- *Dynamic Fisher* and *static Fisher with 32/64 anchors*: alternative curvature estimators discussed in Appendix E and Table 9.

Per-Benchmark Behavior. Table 14 reports the per-benchmark results underlying Table 3. Removing the stability cost primarily degrades benchmarks that are sensitive to fine-tuning: relative to full SPUR, VQAv2, GQA, VizWiz, and TextVQA decrease by 2.4, 2.5, 2.9, and 3.2 points, respectively, whereas ScienceQA, MMBench, and MMBench-CN change by at most 0.5 points.

The selection variants show a related pattern. Fixing the mask after the first step reduces all eight general-benchmark scores and lowers Hscore* from 85.74 to 57.04, highlighting the importance of recomputing the selected update set throughout training. The utility-scaled variant also degrades seven of the eight general benchmarks, with the largest drop on VizWiz (6.3 points); MMBench-CN is the only exception, improving slightly from 58.28 to 58.88. Random selection with the same

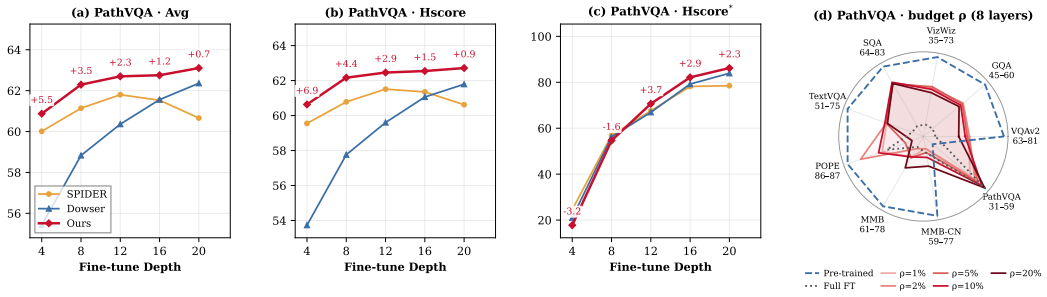


Figure 7: **Qwen2.5-VL-3B on PathVQA.** (a–c) Avg, Hscore, and Hscore* as a function of fine-tuning depth for SPIDER, Model-Dowser (official setting), and SPUR with the best ρ selected at each depth. Numbers above the SPUR curve indicate its margin over Model-Dowser. (d) Per-benchmark effect of the budget ρ with the last 8 layers fine-tuned. Each axis corresponds to one benchmark and is scaled to the range shown below its label. Polygons from light to dark red denote SPUR with $\rho = 1, 2, 5, 10, 20\%$; the dashed blue polygon denotes the pre-trained model and the dotted gray polygon standard fine-tuning.

budget performs better than the fixed and utility-scaled variants, but remains below full SPUR on Avg, Hscore, and Hscore*.

The ratio utility favors retention more strongly: it obtains the highest VQAv2 and GQA scores among the ablation variants (79.48 and 61.99), but reduces PathVQA accuracy to 60.11, 2.28 points below full SPUR. This leads to lower Avg and Hscore despite its stronger scores on several general benchmarks.

Finally, the alternative curvature estimators produce only small changes: each individual benchmark differs from the full method by less than 0.5 points, and Avg changes by at most 0.08. In contrast, removing the dynamic first-order term causes a substantially larger reduction in Hscore*, from 85.74 to 74.24. These results indicate that the dynamic term provides the dominant state-dependent stability signal, while the static curvature acts as a complementary sensitivity prior.

I More Results on Qwen2.5-VL-3B

To further evaluate the generality of SPUR, we repeat the comparison on a third backbone, Qwen2.5-VL-3B (Wu et al., 2025a), using the same PathVQA protocol. The results are shown in Fig. 7.

On Avg and Hscore, SPUR performs best at every fine-tuning depth. Its margin over Model-Dowser is largest at 4 layers, reaching 5.5 points in Avg and 6.9 points in Hscore, and gradually narrows as the trainable depth increases, to 1.2/1.5 points at 16 layers and 0.7/0.9 points at 20 layers. SPIDER is more competitive at shallow and intermediate depths, but falls 2.5 Avg points below SPUR at 20 layers.

The Hscore* results are more nuanced. SPUR leads from 12 layers onward by 3.7, 2.9, and 2.3 points, but trails SPIDER by 3.2 and 1.6 points at 4 and 8 layers, respectively. At shallow depths, Hscore* is especially sensitive to small differences in general-capability retention. For example, at 4 layers, standard fine-tuning reduces Pre-Avg from 76.7 to 63.4, while SPIDER and SPUR recover only part of this drop. SPIDER retains 0.6 more Pre-Avg points than SPUR (65.3 vs. 64.7), which gives it a 3.2-point advantage in Hscore* despite SPUR achieving 2.3 points higher PathVQA accuracy (57.1 vs. 54.8) and higher Avg and Hscore. A similar effect occurs at 8 layers. Using $\rho = 10\%$ instead of the Avg-selected $\rho = 40\%$ raises SPUR's Hscore* to 59.6, above both baselines, while reducing Avg by only 0.3 points. From 12 layers onward, SPUR leads on all three aggregate metrics.

Fig. 7(d) provides the corresponding per-benchmark view. Across the displayed values of ρ, the adapted models remain substantially below the pre-trained model on MMBench, MMBench-CN, and VQAv2 (approximately $78 \to 64$, $77 \to 59$, and $81 \to 71$, respectively). These gaps vary only modestly across the budget sweep, whereas PathVQA accuracy increases from 50.7 to 58.4. Thus,

varying ρ controls the stability–plasticity trade-off within the adapted regime, while a substantial portion of the general-capability drop on this backbone persists across the tested budgets.

I.1 More Results on Target Datasets

Why these datasets. We additionally evaluate on the four downstream tasks used by Model-Dowser (Hwang et al., 2026): COCO-Caption (Chen et al., 2015), Flickr30k (Young et al., 2014), IconQA (Lu et al., 2021), and ImageNet-R (Hendrycks et al., 2021). These tasks complement the medical and driving domains used in the main paper: COCO-Caption and Flickr30k are captioning tasks evaluated with CIDEr (Vedantam et al., 2015), whereas IconQA and ImageNet-R are evaluated using exact-match accuracy. This additional evaluation allows us to compare SPUR under the task setting used by Model-Dowser.

Training. All runs fine-tune the last eight decoder blocks of LLaVA-1.5-7B (Liu et al., 2023) and NVILA-Lite-2B (Liu et al., 2025) together with the projector, using AdamW (Loshchilov & Hutter, 2017) with a learning rate of 10^{-4} and a global batch size of 128. Following the released recipes for these 10k subsets, LLaVA-1.5-7B is trained for 2 epochs (156 steps) and NVILA-Lite-2B for 3 epochs (234 steps). The training data and optimization schedule are kept fixed across methods within each backbone–task configuration. We compare unconstrained fine-tuning, SPIDER (Huang et al., 2025b), Model-Dowser (Hwang et al., 2026) in its official configuration, and SPUR with $\rho \in \{1, 2, 4, 10, 20\}\%$. For each backbone–task configuration, we select ρ according to Avg, using Hscore only to break ties, and report all metrics under the same selected ρ.

Results. Tables 15–18 report the per-benchmark results for both backbones and all tested SPUR budgets. Under the Avg-selected budget, SPUR achieves the highest Avg and Hscore in all eight backbone–task combinations.

On the two captioning tasks, SPUR improves both target adaptation and general-capability retention relative to unconstrained fine-tuning. For example, on COCO-Caption with LLaVA-1.5-7B, SPUR reaches a CIDEr score of 128.29 compared with 114.53 for standard fine-tuning, while increasing Pre-Avg from 63.36 to 66.55. On Flickr30k with NVILA-Lite-2B, SPUR reaches 96.55 CIDEr compared with 75.08 for standard fine-tuning, while its Pre-Avg of 73.25 remains essentially unchanged from the pre-trained value of 73.22. These results indicate that selectively applying candidate updates can improve target learning without requiring the general-capability degradation observed under unconstrained fine-tuning.

IconQA produces a substantially more severe form of forgetting. For NVILA-Lite-2B, unconstrained fine-tuning reduces VQAv2, GQA, VizWiz, ScienceQA, and TextVQA to 0.00, while POPE also drops from 88.66 to 50.52. MMBench and MMBench-CN are more resilient, decreasing from 78.01 to 73.06 and from 58.02 to 55.69, respectively. SPUR substantially mitigates this collapse. On LLaVA-1.5-7B, for example, Model-Dowser reaches a Pre-Avg of 60.92 and an IconQA accuracy of 48.91, whereas SPUR at $\rho = 4\%$ reaches 65.25 Pre-Avg and 72.88 target accuracy. Its Avg therefore exceeds the strongest mitigation baseline by 14.15 points on LLaVA-1.5-7B and 5.94 points on NVILA-Lite-2B.

Finally, the selected budgets are consistently small across these additional tasks: the Avg-selected ρ lies between 1% and 4% in all eight backbone–task combinations. On the captioning tasks, larger budgets can reduce both retention and CIDEr after the optimum, whereas on IconQA and ImageNet-R the target score continues to improve or remain high as general capability declines. These results further illustrate that ρ controls the stability–plasticity trade-off, with the preferred operating point depending on how quickly target performance saturates relative to general capability degradation.

Table 10: **Full fine-tuning-depth results on LLaVA-1.5-7B / PathVQA.** Results are reported when fine-tuning the last 4, 8, 12, 16, and 20 language layers. SPUR uses the Avg-selected update budget at each depth: 20%, 20%, 4%, 2%, and 1%, respectively.

Last 4 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**PathVQA**	**Avg** ↑	**Hscore** ↑	**Hscore*** ↑
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.70	27.62	47.17	39.07	0
Fine-tuned	70.81	56.67	32.23	66.00	50.75	86.95	60.61	54.22	59.78	57.76	58.77	58.75	0
DARE	70.72	56.83	32.79	66.45	51.10	86.82	60.87	54.40	60.00	57.54	58.77	58.74	6.02
Model-Tailor	78.01	61.25	54.26	70.15	57.69	84.55	64.45	58.62	66.12	32.36	49.24	43.45	26.81
Grafting	78.21	61.99	55.56	70.41	58.28	86.88	64.45	58.36	66.77	27.61	47.19	39.06	< 0
CGM	76.33	61.29	54.67	69.16	57.04	86.89	62.83	56.81	65.63	38.53	52.08	48.56	50.67
CGM*	71.32	57.31	33.62	66.35	51.36	86.93	61.13	55.26	60.41	57.73	59.07	59.04	16.65
SPIDER	73.64	57.91	45.23	68.24	54.89	87.41	63.09	57.41	63.48	57.43	60.46	60.31	69.35
Model-Dowser	72.68	55.94	47.73	70.05	56.77	85.99	64.88	58.28	64.04	48.30	56.17	55.06	64.85
SPUR	73.28	57.80	47.88	68.47	55.76	87.60	62.92	57.84	63.94	57.60	**60.77**	**60.60**	**74.94**

Last 8 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**PathVQA**	**Avg** ↑	**Hscore** ↑	**Hscore*** ↑
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.70	27.62	47.17	39.07	0
Fine-tuned	71.44	56.46	26.87	63.15	50.59	87.08	60.27	53.62	58.68	58.36	58.52	58.52	0
DARE	71.12	56.79	26.94	63.10	50.57	87.04	59.93	53.62	58.64	58.07	58.36	58.36	< 0
Model-Tailor	77.72	61.18	52.79	70.05	58.31	85.41	64.62	58.97	66.13	38.00	52.06	48.26	49.51
Grafting	78.13	61.96	55.58	70.43	58.24	86.90	64.45	58.28	66.75	27.67	47.21	39.12	0.29
CGM	76.54	61.30	55.70	69.37	57.33	86.86	62.57	56.47	65.77	37.62	51.70	47.87	47.56
CGM*	72.32	57.13	28.32	65.34	51.60	87.14	60.70	54.40	59.62	58.27	58.94	58.94	20.88
SPIDER	74.58	58.66	40.97	67.58	54.53	86.97	62.66	56.72	62.83	57.49	60.16	60.05	67.52
Model-Dowser	73.63	58.46	46.53	70.01	56.48	86.94	64.88	58.62	64.44	51.38	57.91	57.17	74.44
SPUR	74.22	58.50	47.11	68.71	55.51	87.24	63.26	58.02	64.07	58.13	**61.10**	**60.96**	**80.10**

Last 12 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**PathVQA**	**Avg** ↑	**Hscore** ↑	**Hscore*** ↑
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.70	27.62	47.17	39.07	0
Fine-tuned	68.33	52.43	24.34	51.57	47.83	87.39	57.54	51.47	55.11	58.65	56.88	56.83	0
DARE	68.56	52.35	24.47	54.66	48.11	86.91	58.23	52.07	55.67	58.61	57.14	57.10	9.11
Model-Tailor	76.97	60.75	49.13	70.08	58.55	87.10	64.45	58.88	65.74	41.81	53.77	51.11	60.99
Grafting	78.21	62.01	55.54	70.43	58.25	86.87	64.36	58.19	66.73	27.64	47.19	39.09	0.10
CGM	76.47	61.50	54.08	69.06	57.54	87.12	63.00	56.72	65.69	36.78	51.23	47.15	44.58
CGM*	69.60	53.94	26.23	57.84	49.15	87.48	59.08	52.84	57.02	58.62	57.82	57.81	28.25
SPIDER	74.28	58.73	42.87	67.60	55.55	87.25	63.94	57.84	63.51	58.22	60.87	60.75	83.50
Model-Dowser	74.47	59.99	45.96	69.94	56.66	86.79	64.79	58.53	64.64	52.45	58.54	57.91	81.07
SPUR	74.13	58.75	50.79	69.28	55.69	87.17	62.40	55.34	64.20	58.86	**61.53**	**61.41**	**88.13**

Last 16 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**PathVQA**	**Avg** ↑	**Hscore** ↑	**Hscore*** ↑
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.70	27.62	47.17	39.07	0
Fine-tuned	62.12	42.39	18.55	21.24	43.24	85.20	20.63	14.91	38.54	58.77	48.66	46.55	0
DARE	61.90	42.60	18.61	24.05	42.97	84.95	20.38	16.38	38.98	58.79	48.88	46.88	3.10
Model-Tailor	76.03	60.39	47.04	69.91	58.53	87.38	64.62	59.05	65.37	47.27	56.32	54.87	75.89
Grafting	78.20	61.91	55.56	70.38	58.20	86.88	64.36	58.28	66.72	27.59	47.16	39.04	< 0
CGM	72.29	51.30	47.72	69.25	53.49	87.70	64.45	57.84	63.00	54.06	58.53	58.19	85.84
CGM*	62.74	43.43	19.63	28.08	44.54	85.57	27.96	18.36	41.29	58.82	50.05	48.52	17.80
SPIDER	72.48	56.81	37.23	67.77	54.70	87.78	62.32	55.78	61.86	58.49	60.17	60.13	90.20
Model-Dowser	74.48	60.18	45.10	69.42	56.76	86.77	64.36	58.79	64.48	55.10	59.79	59.42	90.11
SPUR	74.09	59.43	49.85	69.25	56.31	86.98	62.40	56.90	64.40	58.67	**61.54**	**61.40**	**95.59**

Last 20 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**PathVQA**	**Avg** ↑	**Hscore** ↑	**Hscore*** ↑
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.70	27.62	47.17	39.07	0
Fine-tuned	50.66	29.94	13.00	3.02	30.28	84.86	3.15	0.00	26.86	59.55	43.21	37.02	0
DARE	50.82	30.02	13.07	2.48	29.78	85.73	3.67	0.09	26.96	59.40	43.18	37.08	0.47
Model-Tailor	71.81	54.34	42.97	68.00	56.06	87.82	63.51	57.67	62.77	51.33	57.05	56.48	81.43
Grafting	78.17	61.98	55.58	70.41	58.21	86.85	64.36	58.28	66.73	27.62	47.18	39.07	0
CGM	59.11	33.93	31.63	58.10	44.18	87.57	55.92	48.28	52.34	56.20	54.27	54.20	74.60
CGM*	51.24	30.18	13.39	4.34	31.71	85.10	3.92	0.17	27.51	59.65	43.58	37.65	3.17
SPIDER	65.22	45.33	23.75	61.28	51.08	88.03	58.31	53.02	55.75	59.38	57.57	57.51	83.88
Model-Dowser	72.66	57.31	40.77	68.66	55.42	87.55	63.68	58.02	63.01	57.33	60.17	60.04	91.87
SPUR	73.64	58.32	47.71	69.23	55.57	86.92	61.98	56.72	63.76	59.07	**61.42**	**61.33**	**95.47**

Table 11: **Full fine-tuning-depth results on LLaVA-1.5-7B / DriveLM.** Results are reported when fine-tuning the last 4, 8, 12, 16, and 20 language layers. SPUR uses the Avg-selected update budget at each depth: 10%, 4%, 4%, 2%, and 0.25%, respectively.

Last 4 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**DriveLM**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.71	15.64	41.17	25.34	0
Fine-tuned	72.69	58.67	44.72	70.08	51.00	85.64	63.68	56.55	62.88	53.87	58.37	58.03	0
DARE	72.46	58.54	44.42	69.98	50.75	85.64	63.17	55.86	62.60	53.74	58.17	57.83	< 0
Model-Tailor	78.69	61.58	55.83	70.15	58.18	86.80	64.28	58.79	66.79	26.00	46.39	37.43	42.64
Grafting	78.18	61.93	55.56	70.43	58.24	86.84	64.36	58.19	66.72	15.60	41.16	25.29	< 0
CGM	77.29	61.00	54.85	70.03	57.18	85.29	63.43	56.64	65.71	25.30	45.51	36.53	37.68
CGM*	73.00	59.05	46.75	70.10	51.63	85.82	63.85	56.72	63.36	53.64	58.50	58.10	22.51
SPIDER	76.12	59.71	54.73	70.50	54.83	83.90	63.94	57.59	65.16	53.58	59.37	58.81	74.56
Model-Dowser	78.12	59.72	54.77	70.43	56.63	84.10	63.94	56.12	65.48	51.81	58.64	57.85	79.09
SPUR	77.24	60.38	57.58	70.43	56.96	84.85	64.28	56.98	66.09	53.42	**59.75**	**59.08**	**90.70**

Last 8 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**DriveLM**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.71	15.64	41.17	25.34	0
Fine-tuned	69.96	56.25	23.84	69.18	50.24	85.23	62.83	54.74	59.03	53.93	56.48	56.37	0
DARE	69.93	56.43	24.38	69.11	50.24	85.15	62.40	54.66	59.04	53.92	56.48	56.36	0.13
Model-Tailor	77.92	61.23	55.17	70.05	57.92	86.94	64.71	58.79	66.59	33.26	49.93	44.36	62.73
Grafting	78.13	61.95	55.55	70.46	58.15	86.81	64.36	58.19	66.70	15.61	41.16	25.30	< 0
CGM	77.31	61.05	55.12	70.08	57.32	85.00	63.34	55.17	65.55	25.32	45.44	36.53	38.96
CGM*	70.91	56.73	27.34	69.32	50.77	85.44	63.09	54.83	59.80	53.75	56.78	56.62	18.24
SPIDER	75.12	58.61	52.77	69.56	52.42	84.09	63.85	57.16	64.20	53.93	59.06	58.62	80.44
Model-Dowser	78.24	60.01	55.43	70.48	56.96	84.35	64.11	57.33	65.86	52.14	59.00	58.20	92.06
SPUR	77.52	61.03	57.88	70.29	57.27	85.57	64.02	58.19	66.47	53.32	**59.90**	**59.17**	**97.67**

Last 12 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**DriveLM**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.71	15.64	41.17	25.34	0
Fine-tuned	63.47	52.42	16.53	69.87	46.25	85.23	62.49	55.34	56.45	54.06	55.26	55.23	0
DARE	64.21	52.53	16.27	69.61	46.57	85.03	61.47	54.14	56.23	54.13	55.18	55.16	< 0
Model-Tailor	77.87	60.81	54.50	70.55	57.82	86.82	64.36	58.79	66.44	41.61	54.03	51.17	79.80
Grafting	78.14	61.97	55.57	70.41	58.17	86.84	64.36	58.19	66.71	15.60	41.15	25.29	< 0
CGM	77.20	60.76	54.78	70.05	57.48	85.24	63.51	55.78	65.60	24.37	44.98	35.54	36.22
CGM*	65.03	53.38	17.43	69.72	48.03	85.34	62.23	54.83	57.00	54.07	55.53	55.50	10.17
SPIDER	73.69	57.49	49.58	69.94	52.64	84.05	63.85	56.98	63.53	53.99	58.76	58.37	81.59
Model-Dowser	77.92	60.43	55.78	70.43	57.20	85.53	63.68	57.84	66.10	52.91	59.51	58.78	95.54
SPUR	77.66	60.87	57.46	70.62	56.68	85.69	64.28	57.76	66.38	53.74	**60.06**	**59.39**	**97.96**

Last 16 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**DriveLM**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.71	15.64	41.17	25.34	0
Fine-tuned	56.18	44.70	14.91	70.36	38.35	84.86	61.13	55.60	53.26	53.80	53.53	53.53	0
DARE	56.11	44.72	14.91	70.34	37.73	84.87	61.30	55.43	53.18	53.76	53.47	53.47	< 0
Model-Tailor	77.38	60.80	40.66	70.97	57.24	86.94	64.19	58.97	64.64	47.03	55.84	54.45	83.44
Grafting	78.23	61.96	55.51	70.41	58.21	86.87	64.36	58.28	66.73	15.60	41.16	25.29	< 0
CGM	74.96	59.58	49.67	70.03	56.54	85.92	62.40	55.26	64.30	47.08	55.69	54.36	82.22
CGM*	57.73	46.30	15.40	70.41	40.38	84.99	61.55	55.86	54.08	53.81	53.94	53.94	11.48
SPIDER	71.48	56.73	45.71	70.24	50.92	85.17	63.85	57.24	62.67	54.20	58.43	58.13	82.32
Model-Dowser	78.07	60.92	55.30	71.28	57.20	86.53	64.45	57.76	66.44	53.31	59.87	59.15	98.35
SPUR	77.73	61.29	56.32	71.07	57.32	86.59	63.77	56.29	66.30	53.74	**60.02**	**59.36**	**98.38**

Last 20 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**DriveLM**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	78.14	61.93	55.51	70.41	58.21	86.89	64.36	58.19	66.71	15.64	41.17	25.34	0
Fine-tuned	45.81	31.10	11.27	62.37	16.57	84.07	55.41	46.64	44.15	53.97	49.06	48.57	0
DARE	46.34	30.48	10.58	63.99	15.02	84.85	50.21	43.02	43.06	53.83	48.45	47.85	< 0
Model-Tailor	75.59	60.09	26.86	69.70	55.37	86.80	61.81	55.34	61.44	50.92	56.18	55.69	83.65
Grafting	78.28	61.97	55.53	70.43	58.22	86.88	64.36	58.36	66.76	15.59	41.17	25.28	< 0
CGM	67.94	52.35	33.29	63.24	48.66	85.39	53.11	46.12	56.26	48.41	52.34	52.04	65.96
CGM*	46.86	31.49	11.13	63.99	17.31	85.04	51.75	44.22	43.97	53.95	48.96	48.45	< 0
SPIDER	64.08	48.43	32.48	65.81	41.41	81.60	52.26	44.48	53.82	54.18	54.00	54.00	59.99
Model-Dowser	77.37	59.96	53.03	70.95	55.67	85.36	63.68	56.03	65.26	53.97	59.61	59.08	96.68
SPUR	78.48	61.53	55.99	71.35	57.09	86.75	64.11	56.64	66.49	53.19	**59.84**	**59.10**	**98.50**

Table 12: **Full fine-tuning-depth results on NVILA-Lite-2B / PathVQA.** Results are reported when fine-tuning the last 4, 8, 12, 16, and 20 language layers. SPUR uses the Avg-selected update budget at each depth: 10%, 10%, 4%, 1%, and 1%, respectively.

Last 4 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**PathVQA**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.23	52.82	63.02	61.37	0
Fine-tuned	67.21	52.50	52.01	80.24	61.78	87.09	75.70	57.59	66.77	61.56	64.16	64.06	0
DARE	67.46	52.26	50.44	80.22	61.45	87.24	75.62	56.55	66.41	61.05	63.73	63.61	< 0
Model-Tailor	78.66	60.80	66.74	83.07	70.82	89.65	78.18	59.05	73.37	56.24	64.81	63.68	56.30
Grafting	79.94	62.07	65.64	82.67	71.05	89.17	78.01	58.19	73.34	52.51	62.92	61.20	< 0
CGM	77.84	61.60	61.73	80.92	68.91	87.89	76.56	56.81	71.53	59.00	65.27	64.66	72.24
CGM*	76.12	59.37	60.42	81.63	68.62	89.34	76.47	58.10	71.26	60.75	66.01	65.59	78.80
SPIDER	72.30	56.35	58.59	82.36	66.13	89.35	77.15	58.53	70.09	60.98	65.54	65.22	66.45
Model-Dowser	78.88	60.81	65.67	82.95	70.67	89.66	78.01	58.19	73.10	58.51	65.81	65.00	78.27
SPUR	76.40	59.31	62.24	82.29	70.08	90.04	78.09	58.36	72.10	61.01	**66.55**	**66.09**	**87.79**

Last 8 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**PathVQA**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.23	52.82	63.02	61.37	0
Fine-tuned	66.32	50.84	52.96	80.10	59.29	86.61	75.79	56.72	66.08	62.81	64.44	64.40	0
DARE	66.57	51.34	51.75	79.75	57.66	87.82	75.62	57.16	65.96	62.42	64.19	64.14	< 0
Model-Tailor	78.41	60.78	66.47	82.90	71.01	89.72	78.43	59.14	73.36	58.24	65.80	64.93	70.34
Grafting	80.09	62.16	65.56	82.72	70.94	88.91	77.92	58.71	73.38	52.61	62.99	61.28	< 0
CGM	77.88	62.07	63.74	81.02	69.34	87.58	77.75	57.07	72.06	57.36	64.71	63.87	58.90
CGM*	74.07	57.17	61.43	81.73	66.96	89.20	76.90	57.93	70.67	62.05	66.36	66.08	75.82
SPIDER	72.07	54.83	60.17	82.48	66.71	87.54	77.83	58.62	70.03	62.02	66.02	65.78	69.10
Model-Dowser	78.16	60.03	65.28	82.93	70.57	89.86	78.09	58.97	72.99	60.46	66.72	66.13	85.38
SPUR	75.04	57.36	63.02	82.72	70.31	88.01	78.26	58.28	71.62	62.39	**67.01**	**66.69**	**85.74**

Last 12 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**PathVQA**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.23	52.82	63.02	61.37	0
Fine-tuned	64.39	39.27	39.43	75.05	54.99	89.30	70.16	49.14	60.22	63.05	61.63	61.60	0
DARE	63.47	38.62	37.18	73.94	52.57	89.63	69.22	48.28	59.11	62.60	60.86	60.81	< 0
Model-Tailor	77.21	58.48	63.60	82.98	70.51	89.87	77.24	57.33	72.15	59.44	65.80	65.18	75.94
Grafting	80.18	62.27	65.41	82.62	71.05	88.74	77.66	58.62	73.32	52.69	63.00	61.31	< 0
CGM	78.47	62.28	65.42	81.42	70.20	87.51	76.64	56.38	72.29	54.56	63.43	62.19	28.78
CGM*	72.77	51.45	51.78	80.31	65.10	89.83	74.85	54.05	67.52	62.49	65.01	64.91	70.46
SPIDER	73.29	55.63	50.95	81.09	66.52	90.07	75.62	56.03	68.65	62.72	65.68	65.55	77.66
Model-Dowser	76.67	58.29	62.34	83.09	69.42	89.87	77.75	58.62	72.01	61.84	66.92	66.54	89.41
SPUR	75.87	58.00	62.90	82.95	69.90	90.19	78.01	58.97	72.10	62.72	**67.41**	**67.08**	**93.96**

Last 16 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**PathVQA**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.23	52.82	63.02	61.37	0
Fine-tuned	58.52	29.71	20.61	62.41	41.06	86.28	54.65	36.55	48.72	63.33	56.03	55.07	0
DARE	57.83	29.35	18.49	57.51	38.69	85.81	46.89	29.05	45.45	63.03	54.24	52.82	< 0
Model-Tailor	75.70	53.83	53.48	81.84	69.05	88.58	76.56	58.45	69.69	60.81	65.25	64.95	80.53
Grafting	80.27	62.08	65.34	82.90	71.24	88.25	77.75	57.76	73.20	52.60	62.90	61.21	< 0
CGM	78.63	62.34	65.16	81.23	69.56	86.89	76.47	56.81	72.14	53.88	63.01	61.68	18.20
CGM*	66.69	38.03	34.07	75.01	59.03	87.48	68.97	47.93	59.65	63.24	61.44	61.39	61.52
SPIDER	70.70	49.01	39.63	79.11	63.80	88.50	75.79	56.38	65.37	63.18	64.27	64.25	80.43
Model-Dowser	77.12	58.40	59.51	83.14	69.22	89.96	77.66	59.74	71.84	62.82	67.33	67.03	94.78
SPUR	77.48	59.91	64.14	82.88	71.06	89.66	77.41	58.62	72.64	62.60	**67.62**	**67.25**	**95.28**

Last 20 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**PathVQA**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.23	52.82	63.02	61.37	0
Fine-tuned	50.81	25.54	14.36	11.18	29.05	86.61	8.95	9.83	29.54	63.31	46.43	40.28	0
DARE	49.43	25.10	13.11	10.28	27.33	86.33	5.12	3.53	27.53	63.15	45.34	38.34	< 0
Model-Tailor	72.30	47.59	41.31	81.02	67.42	90.01	75.36	58.79	66.72	60.43	63.58	63.42	78.30
Grafting	80.29	62.08	65.09	83.16	71.27	88.36	77.32	58.28	73.23	52.42	62.83	61.10	< 0
CGM	78.69	61.41	65.30	81.56	69.31	87.20	77.07	56.90	72.18	53.00	62.59	61.12	3.35
CGM*	62.28	32.32	25.93	47.58	54.03	87.15	52.09	44.40	50.72	63.76	57.24	56.50	65.31
SPIDER	68.68	45.56	29.71	68.10	60.50	88.17	67.43	53.53	60.21	63.55	61.88	61.84	82.50
Model-Dowser	76.01	56.79	54.36	82.32	68.33	89.33	77.41	59.91	70.56	62.94	66.75	66.53	95.16
SPUR	77.14	58.60	62.09	82.55	69.43	89.37	77.32	59.83	72.04	62.87	**67.45**	**67.14**	**96.53**

Table 13: **Full fine-tuning-depth results on NVILA-Lite-2B / DriveLM.** Results are reported when fine-tuning the last 4, 8, 12, 16, and 20 language layers. SPUR uses the Avg-selected update budget at each depth: 10%, 2%, 4%, 2%, and 1%, respectively.

Last 4 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**DriveLM**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.22	9.68	41.45	17.10	0
Fine-tuned	67.07	50.71	27.21	77.20	41.57	85.63	67.95	51.72	58.63	56.63	57.63	57.61	0
DARE	65.80	49.28	26.92	76.61	40.13	84.20	64.79	48.79	57.07	55.95	56.51	56.50	< 0
Model-Tailor	79.33	61.61	65.66	83.07	70.46	88.57	77.41	58.36	73.06	27.10	50.08	39.54	53.96
Grafting	79.92	61.93	65.33	82.83	70.63	88.17	77.83	58.02	73.08	10.28	41.68	18.02	2.52
CGM	75.94	59.06	63.41	77.51	56.32	87.12	71.01	52.93	67.91	24.83	46.37	36.36	42.82
CGM*	75.60	57.65	61.21	80.05	61.55	87.17	73.66	55.86	69.09	54.86	61.98	61.16	82.17
SPIDER	73.80	55.80	52.99	80.15	59.69	85.78	72.72	54.83	66.97	55.74	61.35	60.84	72.22
Model-Dowser	78.58	60.44	63.71	82.53	69.21	86.78	76.64	58.97	72.11	51.07	61.59	59.79	90.20
SPUR	77.76	59.68	64.57	81.47	65.83	87.26	75.28	57.33	71.15	54.86	**63.00**	**61.95**	**90.69**

Last 8 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**DriveLM**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.22	9.68	41.45	17.10	0
Fine-tuned	64.74	49.44	18.55	77.34	35.61	88.66	67.60	52.07	56.75	57.13	56.94	56.94	0
DARE	64.49	47.56	17.91	76.96	33.91	87.01	64.88	48.19	55.11	56.74	55.93	55.92	< 0
Model-Tailor	79.24	61.43	65.57	83.19	70.35	89.25	76.21	58.71	73.00	35.13	54.06	47.43	69.48
Grafting	79.87	61.94	65.42	82.86	70.50	88.09	77.92	57.84	73.05	11.09	42.07	19.26	5.77
CGM	77.22	60.86	64.31	79.01	60.73	88.16	72.89	54.48	69.71	21.29	45.50	32.62	37.32
CGM*	74.41	58.29	53.18	80.66	60.88	88.63	73.32	57.41	68.35	55.63	61.99	61.34	81.53
SPIDER	74.22	58.68	48.44	80.17	59.86	89.08	71.10	52.50	66.76	56.58	61.67	61.25	75.25
Model-Dowser	79.06	60.97	63.39	82.98	69.51	87.82	77.32	59.31	72.55	52.95	62.75	61.22	93.48
SPUR	78.87	60.92	65.65	82.39	69.14	87.63	77.15	58.79	72.57	54.33	**63.45**	**62.14**	**95.04**

Last 12 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**DriveLM**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.22	9.68	41.45	17.10	0
Fine-tuned	57.79	36.21	13.74	78.47	23.63	86.66	70.42	53.62	52.57	57.80	55.18	55.06	0
DARE	55.77	33.50	13.37	77.10	21.19	84.97	68.37	48.45	50.34	57.50	53.92	53.68	< 0
Model-Tailor	79.04	60.53	62.47	83.19	69.47	89.10	77.41	59.40	72.58	50.08	61.33	59.27	89.95
Grafting	79.76	61.86	65.42	82.93	70.62	87.83	77.83	58.02	73.03	12.76	42.90	21.72	12.02
CGM	77.61	61.38	65.63	79.53	65.17	87.42	75.02	56.12	70.99	18.76	44.87	29.68	31.15
CGM*	71.06	51.07	40.06	81.21	54.96	87.52	74.17	56.29	64.54	57.18	60.86	60.64	73.04
SPIDER	72.80	54.21	32.42	80.71	57.44	88.32	74.77	56.90	64.70	57.91	61.30	61.12	73.99
Model-Dowser	78.79	60.99	62.95	83.38	69.64	88.27	77.32	58.10	72.43	55.02	63.73	62.54	95.18
SPUR	79.22	61.08	64.36	82.46	68.55	88.92	76.98	58.28	72.48	56.75	**64.62**	**63.66**	**97.11**

Last 16 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**DriveLM**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.22	9.68	41.45	17.10	0
Fine-tuned	49.68	28.10	11.25	73.10	10.71	86.91	62.15	45.95	45.98	58.04	52.01	51.31	0
DARE	48.16	25.71	10.64	69.94	7.91	84.32	55.33	38.02	42.50	57.70	50.10	48.95	< 0
Model-Tailor	77.68	59.33	55.61	82.36	68.99	89.00	75.36	55.78	70.51	53.13	61.82	60.60	89.95
Grafting	79.83	61.83	65.84	82.74	70.60	87.88	78.18	58.36	73.16	15.15	44.15	25.10	20.32
CGM	78.43	61.24	65.80	80.36	65.55	87.52	75.53	56.98	71.43	18.03	44.73	28.79	29.15
CGM*	61.97	37.87	18.52	77.81	39.73	88.24	69.48	51.12	55.59	57.23	56.41	56.40	51.93
SPIDER	67.68	44.37	21.99	79.98	46.64	87.10	72.38	53.36	59.19	58.32	58.75	58.75	65.30
Model-Dowser	79.31	61.17	61.64	82.72	69.03	89.24	76.90	59.48	72.44	56.27	64.35	63.34	96.72
SPUR	79.40	60.92	64.22	82.72	68.98	89.25	77.66	57.67	72.60	57.12	**64.86**	**63.94**	**97.91**

Last 20 layers fine-tuned

Method	General Benchmarks									Target Task	Overall Metrics		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	**Pre-Avg**	**DriveLM**	**Avg ↑**	**Hscore ↑**	**Hscore* ↑**
Pre-trained	80.16	62.27	65.15	82.67	70.87	88.66	78.01	58.02	73.22	9.68	41.45	17.10	0
Fine-tuned	44.81	24.28	8.37	65.76	6.89	86.65	57.63	41.72	42.02	57.66	49.84	48.61	0
DARE	42.36	23.41	8.03	64.65	6.73	86.00	56.10	40.00	40.91	57.62	49.26	47.85	< 0
Model-Tailor	76.62	57.43	45.44	80.90	67.88	89.74	75.11	56.55	68.71	54.19	61.45	60.59	89.00
Grafting	79.98	61.76	66.09	83.14	70.98	87.85	78.26	58.62	73.34	18.17	45.75	29.12	30.07
CGM	78.03	60.92	65.02	81.07	66.15	87.66	75.62	56.55	71.38	17.58	44.48	28.21	28.03
CGM*	56.18	32.07	11.38	73.52	30.43	89.37	68.20	51.64	51.60	56.24	53.92	53.82	46.65
SPIDER	63.73	38.25	16.38	77.25	41.47	88.00	70.84	50.69	55.83	58.66	57.24	57.21	61.36
Model-Dowser	78.73	60.50	60.10	82.95	68.65	88.83	77.49	58.97	72.03	57.62	64.82	64.02	98.01
SPUR	79.40	61.33	65.65	82.76	69.68	88.85	77.83	59.22	73.09	57.09	**65.09**	**64.11**	**99.19**

Table 14: **Per-benchmark ablation results** on NVILA-Lite-2B / PathVQA with the last 8 language layers fine-tuned and $\rho = 10\%$ (full version of Table 3). General benchmarks are evaluated under the CGM protocol.

Variant	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MMB	MMB-CN	PathVQA	Avg	Hscore	Hscore*
SPUR (full)	75.04	57.36	63.02	82.72	70.31	88.01	78.26	58.28	62.39	**67.01**	**66.69**	**85.74**
Stability Cost C												
benefit only (no C)	72.68	54.90	60.17	82.48	67.11	86.67	77.75	58.02	62.30	66.14	65.91	69.29
w/o static curvature F^{sta}	74.62	56.86	62.75	82.50	69.95	87.63	77.92	58.19	62.35	66.82	66.52	82.82
w/o dynamic term $\mathbf{r}_t$	73.68	56.08	60.76	82.48	68.46	87.06	77.75	57.67	62.09	66.29	66.03	74.24
Utility												
ratio $\hat{B}/(\hat{C}+\epsilon)$ (not harmonic)	79.48	61.99	63.06	82.10	71.43	88.81	77.58	58.19	60.11	66.47	65.86	82.39
Selection												
per-tensor top-ρ (not global)	74.90	57.20	62.94	82.55	70.10	87.82	78.09	58.36	62.32	66.91	66.59	84.39
mask fixed after step 1	71.01	52.87	59.94	82.50	67.04	85.04	76.81	57.50	61.59	65.34	65.12	57.04
random set, same budget	76.12	58.32	62.16	82.65	69.97	89.89	78.43	58.45	60.74	66.37	65.89	81.05
utility-scaled, no top-ρ	69.88	53.25	56.73	81.87	65.05	85.95	77.41	58.88	62.57	65.60	65.46	52.34
Curvature												
dynamic Fisher (every 10 steps)	75.30	57.63	62.97	82.62	70.16	88.17	78.43	58.02	62.20	66.93	66.60	85.33
static Fisher, 32 anchors	74.99	57.38	62.82	82.69	70.21	88.07	78.26	58.28	62.38	66.98	66.66	85.42
static Fisher, 64 anchors	74.92	57.31	62.90	82.65	69.99	87.97	78.43	58.28	62.35	66.95	66.63	85.03

Table 15: **Per-benchmark results on COCO-Caption** with the last 8 language layers fine-tuned. Pre-Avg denotes the average over the eight general benchmarks, CIDEr is the target-task score, Avg is their arithmetic mean, and Hscore is their harmonic mean. Hscore* measures normalized retention and adaptation relative to the pre-trained and standard fine-tuning references. Bold indicates the best result among the compared mitigation methods and SPUR budgets.

					SPUR (ours), ρ				
Benchmark	Pre.	FT	SPIDER	Dowser	1%	2%	4%	10%	20%
LLaVA-1.5-7B									
VQAv2	78.14	74.76	76.63	78.10	77.87	77.62	77.42	76.87	75.94
GQA	61.93	58.86	60.96	61.92	62.02	61.96	61.69	61.04	60.53
VizWiz	55.51	55.82	56.32	55.24	56.46	57.50	58.15	57.51	55.92
SQA	70.41	68.69	69.25	69.87	69.58	69.94	69.96	69.96	69.51
TextVQA	58.21	55.42	55.95	57.83	57.82	57.78	57.20	56.83	56.02
POPE	86.89	84.92	85.75	86.96	86.69	86.74	86.58	86.56	86.10
MM-Bench	64.36	58.31	60.02	63.94	64.71	64.36	63.43	62.40	61.30
MM-Bench-CN	58.19	50.09	51.29	57.67	57.24	56.38	54.66	53.19	52.59
Pre-Avg	66.71	63.36	64.52	66.44	66.55	66.53	66.14	65.54	64.74
CIDEr	43.67	114.53	122.69	125.20	128.29	127.99	126.04	121.79	118.72
Avg	55.19	88.95	93.61	95.82	**97.42**	97.26	96.09	93.67	91.73
Hscore	52.78	81.59	84.57	86.81	**87.64**	87.55	86.75	85.22	83.79
Hscore*	–	0.00	52.96	102.26	**105.97**	105.54	96.80	81.98	59.29
NVILA-Lite-2B									
VQAv2	80.16	72.63	76.60	79.79	79.52	79.43	78.77	77.42	76.72
GQA	62.27	57.10	59.63	62.00	61.17	61.12	60.56	60.07	59.52
VizWiz	65.15	51.76	59.51	63.94	65.17	64.77	64.31	62.98	60.70
SQA	82.67	79.70	80.78	82.72	82.62	82.50	82.24	81.87	81.56
TextVQA	70.87	58.35	63.95	69.76	69.14	69.01	68.44	66.53	65.09
POPE	88.66	86.03	86.97	88.84	88.01	87.90	87.71	87.58	87.43
MM-Bench	78.01	68.97	72.46	76.98	76.81	76.73	77.07	76.13	75.45
MM-Bench-CN	58.02	50.86	55.69	59.05	57.24	57.84	58.45	58.36	57.67
Pre-Avg	73.22	65.67	69.45	72.88	72.46	72.41	72.19	71.37	70.52
CIDEr	26.41	108.41	117.32	129.98	135.45	133.36	130.59	120.93	114.33
Avg	49.82	87.04	93.39	101.43	**103.96**	102.89	101.39	96.15	92.42
Hscore	38.82	81.80	87.25	93.40	**94.41**	93.86	92.98	89.76	87.23
Hscore*	–	0.00	68.93	**108.77**	107.27	105.99	102.81	91.16	80.27

Table 16: **Per-benchmark results on Flickr30k** with the last 8 language layers fine-tuned. Pre-Avg denotes the average over the eight general benchmarks, CIDEr is the target-task score, Avg is their arithmetic mean, and Hscore is their harmonic mean. Hscore* measures normalized retention and adaptation relative to the pre-trained and standard fine-tuning references. Bold indicates the best result among the compared mitigation methods and SPUR budgets.

					SPUR (ours), ρ				
Benchmark	Pre.	FT	SPIDER	Dowser	1%	2%	4%	10%	20%
LLaVA-1.5-7B									
VQAv2	78.14	75.32	76.82	77.79	77.81	77.56	77.42	76.93	76.62
GQA	61.93	57.92	60.70	61.96	61.93	61.70	61.40	60.78	60.30
VizWiz	55.51	58.15	59.61	55.09	55.70	56.82	57.51	57.80	59.02
SQA	70.41	66.02	68.47	70.24	69.89	69.89	69.84	69.87	69.79
TextVQA	58.21	55.98	57.29	57.77	57.88	57.92	57.68	57.48	57.18
POPE	86.89	82.94	85.30	86.92	86.26	86.15	86.08	85.91	85.45
MM-Bench	64.36	54.56	59.51	64.45	65.30	65.05	64.96	63.43	61.81
MM-Bench-CN	58.19	46.38	50.09	58.10	57.59	57.24	56.90	55.86	53.97
Pre-Avg	66.71	62.16	64.72	66.54	66.54	66.54	66.47	66.01	65.52
CIDEr	31.14	77.21	86.33	83.83	87.94	88.10	87.16	83.82	81.22
Avg	48.92	69.69	75.53	75.19	77.24	**77.32**	76.82	74.91	73.37
Hscore	42.46	68.87	73.98	74.19	75.76	**75.82**	75.42	73.85	72.53
Hscore*	–	0.00	76.68	104.56	108.21	**108.28**	106.58	97.26	87.93
NVILA-Lite-2B									
VQAv2	80.16	74.60	77.02	79.43	80.04	79.94	79.44	78.34	77.71
GQA	62.27	56.38	58.67	61.77	61.97	61.83	61.31	60.04	58.89
VizWiz	65.15	56.07	60.05	64.61	65.80	65.41	65.02	63.60	61.59
SQA	82.67	79.44	81.58	82.79	82.95	82.81	82.57	82.62	82.39
TextVQA	70.87	64.38	66.51	69.97	70.28	70.49	70.26	69.15	68.09
POPE	88.66	85.74	86.94	88.51	88.06	87.86	87.76	87.72	87.57
MM-Bench	78.01	68.54	74.17	77.15	77.41	77.32	77.58	76.98	76.39
MM-Bench-CN	58.02	52.50	58.45	59.74	59.48	59.74	59.57	59.66	58.97
Pre-Avg	73.22	67.21	70.43	72.99	73.25	73.18	72.94	72.26	71.45
CIDEr	28.34	75.08	80.46	91.34	96.55	94.47	89.93	81.91	78.40
Avg	50.78	71.14	75.44	82.17	**84.90**	83.83	81.44	77.09	74.92
Hscore	40.87	70.92	75.11	81.14	**83.30**	82.47	80.55	76.78	74.76
Hscore*	–	0.00	72.31	112.27	**118.99**	116.64	110.58	96.98	85.03

Table 17: **Per-benchmark results on IconQA** with the last 8 language layers fine-tuned. Pre-Avg denotes the average over the eight general benchmarks, Acc. is the target-task score, Avg is their arithmetic mean, and Hscore is their harmonic mean. Hscore* measures normalized retention and adaptation relative to the pre-trained and standard fine-tuning references. Since the pre-trained model obtains zero target accuracy under the official exact-match evaluation, Hscore* uses a degenerate target-task reference in this setting and should therefore be interpreted with caution. Bold indicates the best result among the compared mitigation methods and SPUR budgets.

					SPUR (ours), ρ				
Benchmark	Pre.	FT	SPIDER	Dowser	1%	2%	4%	10%	20%
LLaVA-1.5-7B									
VQAv2	78.14	0.47	1.43	77.96	78.34	78.28	78.03	74.27	65.14
GQA	61.93	0.02	0.06	61.27	61.74	61.67	61.41	57.36	47.69
VizWiz	55.51	0.04	0.01	45.52	55.90	55.02	53.00	22.74	6.87
SQA	70.41	69.84	54.19	37.68	33.60	66.00	63.29	69.91	66.99
TextVQA	58.21	3.25	0.73	57.52	58.18	58.28	57.59	50.62	41.06
POPE	86.89	85.26	83.73	87.09	86.57	86.16	85.79	85.77	85.52
MM-Bench	64.36	65.30	65.73	62.83	62.92	63.68	64.79	65.22	65.73
MM-Bench-CN	58.19	58.71	58.19	57.50	57.50	57.59	58.10	57.59	57.93
Pre-Avg	66.71	35.36	33.01	60.92	61.84	65.83	65.25	60.43	54.62
Acc.	0.00	78.55	76.00	48.91	56.93	64.66	72.88	76.38	77.77
Avg	33.35	56.95	54.50	54.91	59.39	65.25	**69.06**	68.41	66.19
Hscore	0.00	48.77	46.03	54.26	59.29	65.24	**68.85**	67.48	64.17
Hscore*	–	0.00	$\leq$0.00	70.61	78.03	89.15	**94.05**	87.77	75.82
NVILA-Lite-2B									
VQAv2	80.16	0.00	0.00	78.62	79.76	79.70	79.31	78.06	68.62
GQA	62.27	0.00	0.00	61.10	62.14	62.23	61.85	57.69	36.78
VizWiz	65.15	0.00	0.00	28.56	63.63	59.31	50.83	25.33	2.64
SQA	82.67	0.00	0.00	0.00	21.36	7.36	3.54	0.92	0.00
TextVQA	70.87	0.00	0.00	65.12	71.51	71.36	70.78	65.48	31.83
POPE	88.66	50.52	85.17	89.50	89.29	89.23	89.17	89.39	89.44
MM-Bench	78.01	73.06	74.17	75.62	75.62	75.02	74.51	73.49	74.00
MM-Bench-CN	58.02	55.69	57.07	57.67	56.72	56.90	57.16	54.91	55.52
Pre-Avg	73.22	22.41	27.05	57.02	65.00	62.64	60.89	55.66	44.85
Acc.	0.00	94.05	93.56	87.63	91.53	92.15	92.75	93.35	93.84
Avg	36.61	58.23	60.30	72.33	**78.27**	77.39	76.82	74.50	69.35
Hscore	0.00	36.19	41.97	69.09	**76.02**	74.58	73.52	69.74	60.69
Hscore*	–	0.00	16.73	78.70	**90.07**	87.57	85.68	78.87	61.23

Table 18: **Per-benchmark results on ImageNet-R** with the last 8 language layers fine-tuned. Pre-Avg denotes the average over the eight general benchmarks, Acc. is the target-task score, Avg is their arithmetic mean, and Hscore is their harmonic mean. Hscore* measures normalized retention and adaptation relative to the pre-trained and standard fine-tuning references. Bold indicates the best result among the compared mitigation methods and SPUR budgets.

					SPUR (ours), ρ				
Benchmark	Pre.	FT	SPIDER	Dowser	1%	2%	4%	10%	20%
LLaVA-1.5-7B									
VQAv2	78.14	58.47	66.63	72.74	74.14	73.21	71.71	69.73	67.98
GQA	61.93	46.09	51.61	57.04	57.93	57.25	55.84	54.03	52.51
VizWiz	55.51	39.21	44.52	51.10	52.70	52.12	50.41	47.88	46.37
SQA	70.41	69.39	69.37	70.01	70.08	69.82	69.94	69.91	70.01
TextVQA	58.21	45.24	52.95	56.38	57.08	56.63	56.02	55.25	54.02
POPE	86.89	86.27	86.41	87.26	87.12	87.15	86.86	86.56	86.26
MM-Bench	64.36	62.40	62.15	64.19	64.19	64.36	64.28	63.60	63.51
MM-Bench-CN	58.19	51.81	53.28	57.67	58.28	57.93	57.76	56.81	55.78
Pre-Avg	66.71	57.36	60.87	64.55	65.19	64.81	64.10	62.97	62.06
Acc.	16.33	90.47	89.57	87.07	87.37	88.77	89.07	89.93	90.07
Avg	41.52	73.91	75.22	75.81	76.28	**76.79**	76.58	76.45	76.06
Hscore	26.24	70.21	72.48	74.14	74.67	**74.92**	74.55	74.08	73.48
Hscore*	–	0.00	54.36	85.17	**89.38**	87.79	83.14	74.82	66.74
NVILA-Lite-2B									
VQAv2	80.16	36.01	60.92	74.70	76.34	74.76	72.78	69.48	65.63
GQA	62.27	24.25	43.58	57.12	58.60	57.46	55.58	50.89	47.34
VizWiz	65.15	16.59	37.83	60.68	63.29	62.64	60.57	54.99	48.51
SQA	82.67	72.93	75.41	81.25	82.57	82.43	81.54	80.59	80.24
TextVQA	70.87	21.33	48.94	65.12	67.46	66.11	64.36	61.53	56.45
POPE	88.66	83.08	89.08	89.04	88.98	89.07	88.91	88.65	88.83
MM-Bench	78.01	57.72	63.34	73.40	75.70	75.28	73.49	72.12	70.84
MM-Bench-CN	58.02	31.72	42.50	50.86	55.34	54.14	51.72	49.22	47.41
Pre-Avg	73.22	42.95	57.70	69.02	71.04	70.23	68.62	65.94	63.16
Acc.	22.13	93.53	93.17	91.93	91.77	92.17	92.93	93.40	93.40
Avg	47.68	68.24	75.43	80.48	**81.40**	81.20	80.78	79.67	78.28
Hscore	33.99	58.87	71.27	78.85	**80.08**	79.72	78.95	77.30	75.36
Hscore*	–	0.00	65.41	91.57	**95.09**	93.94	91.41	86.24	80.00