SWIM: Sparse Weight-Injected Memories

Anonymous authors
Paper under double-blind review

Abstract

Persistent visual memory enables vision-language models to answer multiple queries about an image without repeatedly processing visual tokens. Existing approaches encode such memories as low-rank weight updates. We show that low-rank memory updates are insufficient, particularly for dense-text and OCR-intensive tasks. We introduce a training-free diagnostic that measures the minimum activation-subspace rank required to retain a target level of image-conditioned performance. Guided by this analysis, we propose sparse weight-injected memories (SWIM) which replace low-rank updates with sparse, high-rank updates. Under the same hypernetwork output budget, SWIM provides substantially greater rank than LoRA, improving exactly the rank-hungry benchmarks the diagnostic flagged, by 5.3% and 10.5% normalized dense-text accuracy across two backbones, while preserving general-perception accuracy. These results expose the low-rank bottleneck in persistent visual memory and show that SWIM overcomes it without generating more parameters.

1 Introduction

Vision–language models typically represent images as sequences of visual tokens concatenated with text and processed jointly by a language model (Alayrac et al., 2022; Liu et al., 2023). While effective, this paradigm keeps the visual information in the context throughout inference, increasing sequence length and compute. Hypernetwork conditioning offers an alternative that decouples the two: a generator emits a weight update for the frozen language model from an input, after which the model runs on text alone. Written into the weights, the update is a parametric memory: the input is encoded once, its effect persists in the model's parameters, and it can be queried repeatedly without re-inserting tokens or consuming context. This work is about that memory—what it must contain and how it should be written.

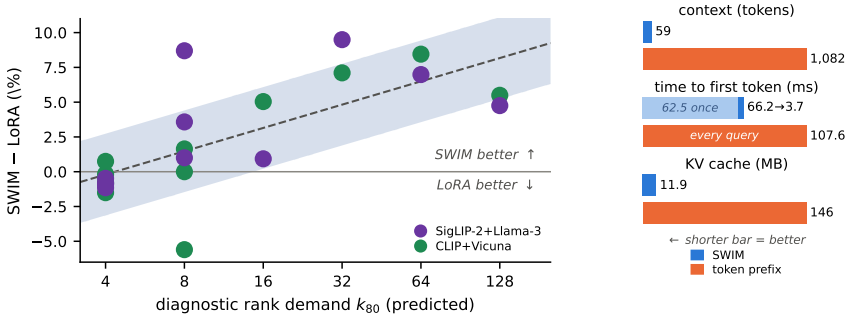

Figure 1: **Teaser. Left:** measured rank demand predicts where SWIM beats LoRA—the positive correlation shows the diagnostic predicts where added rank pays off. **Right:** as a persistent weight memory, SWIM replaces the visual-token prefix at a fraction of its cost: a 62.5 ms write paid on the first query only, a follow-up query starting in 3.7 ms, while the prefix re-pays its 107.6 ms image prefill on every query.

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.

The head that emits the memory is essentially uniform across this line: every weight-modulating input-conditioned hypernetwork we are aware of modulates the base model with LoRA (Hu et al., 2022), generating one or both low-rank factors that compose into the modulated weight $W' = W + AB^\top$ and bound every per-input update by the rank r of the emitted factors (Phang et al., 2023; Charakorn et al., 2025; Xiao et al., 2025; Chen et al., 2025; Liu et al., 2026; Charakorn et al., 2026; Ma et al., 2024; Wu et al., 2026); Doc-to-LoRA's chunking raises that rank with document length by concatenating rank-r chunks, but never leaves the low-rank family—and the extra rank it buys lives on the document-length axis, which a single image does not have.

The appeal of the cap is clear: it modulates every weight of a site while generating only a fraction of that many values, with the built-in caveat that every update is rank r (the prior is supported by the low intrinsic dimension of task updates (Aghajanyan et al., 2021)). A parametric memory, however, raises a demand this budget never addressed: it is written before any query is known, so one update must retain whatever later queries might ask—object identity, chart values, document fields, fine text. This raises a central question: *how much rank does a useful weight-space memory require?*

Our answer is conditional. We introduce a training-free projected-rollout diagnostic that measures how much task accuracy an optimistic rank-k per-image correction can retain, and sweep k across eleven benchmarks (Section 3). We find that on some datasets rank demand saturates at small k, while others keep demanding directions well beyond practical LoRA rank; the highest demands sit in the dense-text and document benchmarks. Each update serves one image and every later question about it, so the rank it must carry grows with the hardest question the image could answer—the demand we measure per benchmark prices exactly that.

Meeting these demands by simply raising r is impractical: setting the rank to satisfy the most demanding benchmark would make the hypernetwork's output layer alone larger than the language network it modulates. The question is therefore not whether information can be stored in weight space, but whether a low-rank parameterization leaves that information enough rank to live in. Either one accepts that hypernetwork-conditioned memories cannot serve these tasks—or the geometry of the emitted update changes. We pursue the latter: we keep the hypernetwork architecture, training recipe, and generated-output budget and replace LoRA's output head with sparse weight-injected memories (SWIM), which write independent values at fixed positions, so the emitted matrix rank is bounded by the target matrix and mask structure rather than by a chosen factor rank—SWIM reaches far higher rank while generating the same number of values. SWIM + LoRA divides the same output budget between LoRA and SWIM.

The gains land where the diagnostic predicted they should (Figure 1, left): the benchmarks whose measured rank demand exceeds the LoRA ceiling are exactly those the high-rank head improves. Across both backbones, pure SWIM improves every high-rank-requiring benchmark, while rank-saturated benchmarks remain near LoRA. The hybrid SWIM + LoRA keeps the dense-text gains on both backbones and, on the headline backbone, lifts the general group above LoRA as well.

Our contributions are:

- a query-agnostic rank perspective for hypernetwork-conditioned weight memories, and a training-free projected-rollout diagnostic that prices each task's rank demand on a frozen model before training;
- SWIM and a low-rank+sparse combination, SWIM + LoRA, that decouple update rank from the generated-output budget, leaving the conditioning front-end unchanged;
- output-budget-matched evaluations on two backbones over eleven benchmarks, in which the measured rank demand predicts which tasks a rank-unbounded head improves; and
- system timings connecting weight-space memory to the context and computation saved relative to visual-token prefixing.

We follow VLoRA's two-stage protocol (Ma et al., 2024) (Appendix A); it is held fixed across LoRA, SWIM, and SWIM + LoRA, so only the generated update geometry changes.

2 Related Work

Writing weight-space memories. Model-editing methods demonstrate that knowledge can be written directly into a model's parameters: ROME, MEMIT, MEND, and MALMEN compute targeted weight updates that encode specific factual associations against a pretrained backbone (Meng et al., 2022; 2023; Mitchell et al., 2022; Tan et al., 2024); frozen backbones can also absorb knowledge into attached adapter modules (Wang et al., 2021). This view is supported by evidence that feed-forward layers behave as key–value memories (Geva et al., 2021) and that visual information is integrated at identifiable layers in vision–language models (Basu et al., 2024). These methods, however, optimize each update for a specific known edit. A persistent weight-space memory operates in a different regime: a learned writer must generate the update directly from the image, before any downstream query is known. Prior work establishes that information can be written into weights, but not how much rank a useful memory update must carry.

Hypernetworks as weight-space memory writers. Hypernetworks generate model parameters conditioned on an input (Ha et al., 2017), enabling input-dependent modulation without changing the underlying architecture (Platanios et al., 2018; Jin et al., 2020; Karimi Mahabadi et al., 2021; Üstün et al., 2022; He et al., 2022; Zhang et al., 2023; Ivison & Peters, 2022; Ivison et al., 2023). Recent work has largely converged on a common design: *weight-modulating* input-conditioned generators emit LoRA updates from conditioning signals such as task descriptions, prompts, documents, user profiles, images, or videos (Phang et al., 2023; Xiao et al., 2023; Jin et al., 2024; Ortiz-Barajas et al., 2024; Liao et al., 2025; Lv et al., 2024; Charakorn et al., 2024; 2025; Xiao et al., 2025; Abdalla et al., 2025; Liang et al., 2025; Chen et al., 2025; Liu et al., 2026; Charakorn et al., 2026; Li et al., 2025; Park et al., 2026; Ma et al., 2024; Wu et al., 2026; Tan et al., 2026). The rank of the emitted update is carried as a fixed hyperparameter, chosen before the rank demand of the conditioned content is known; prior work studies how generators emit such updates segment-wise (Liu et al., 2026; Wang et al., 2025) or how their outputs are calibrated (Trojan & Gebala, 2026), but largely assumes the low-rank form itself. Unlike conventional parameter-efficient adaptation, where low rank is motivated by the low intrinsic dimensionality of a specific task update (Aghajanyan et al., 2021), a persistent memory writer must encode arbitrary inputs before future queries are known. In this regime the low-rank prior loses the evidence that justified it, and whether the low-rank form leaves enough rank for persistent weight-space memory has not been asked.

Beyond low-rank memory updates. The choice of update geometry originates from the parameter-efficient adaptation literature rather than the hypernetwork literature: LoRA constrains updates to a low-rank factorization (Hu et al., 2022), an assumption inherited by virtually all weight-modulating hypernetworks. To realize higher-rank updates under the same output budget, we draw on Sparse High Rank Adapters (SHiRA) (Bhardwaj et al., 2024), which replace the low-rank factorization with unconstrained values written to a sparse set of parameter locations whose support is fixed before training, selected by saliency criteria such as weight magnitude, gradient magnitude, or SNIP (Lee et al., 2019). Although developed for per-task adaptation, the fixed-support sparse principle transfers directly to persistent weight-space memory: SWIM builds on it by preserving the conditioning architecture and the generated-output budget and writing sparse, per-input weight values. We select the positions with a diagonal Fisher criterion, taking the importance-selection idea from the saliency-pruning and parameter-importance lineage (LeCun et al., 1989; Hassibi & Stork, 1993; Theis et al., 2018), at the budget priced in advance by the diagnostic of Section 3.

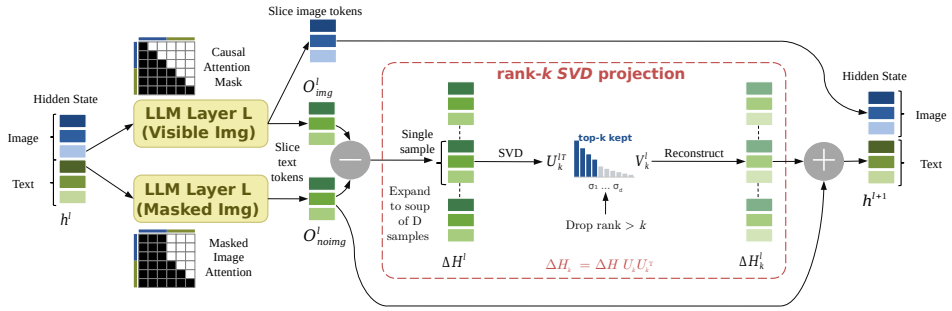

Figure 2: **How the activation rank is controlled.** The hidden state h^ℓ passes through the *same* decoder layer ℓ twice — image tokens *visible*, then *masked* — and the text-token rows of the two outputs subtract to the image correction $\Delta H^\ell = O^\ell_{\text{img}} - O^\ell_{\text{noimg}}$. The dashed red box controls its rank: the corrections of the image's pooled fit queries feed one SVD, only the k leading basis directions $U^\ell_{i,k}$ are kept, and the projected correction $\Delta H^\ell_k = \Delta H^\ell U^\ell_{i,k} U^{\ell\top}_{i,k}$ is added to the no-image state to form $h^{\ell+1}$. Sweeping the single knob k from 0 (text alone) to *full* (native image-conditioned inference) traces how much of the answer-relevant image signal fits in a rank-k subspace (Fig. 3).

3 How Much Rank Does a Persistent Visual Memory Need?

3.1 One update must support many future queries

A persistent weight-space memory must be written before any downstream query is known. Unlike conventional adaptation, which can specialize to a single task objective, the same update must preserve information useful for many possible future questions about an image. This gives rise to *rank demand*: the number of independent directions needed to support the collection of queries an image may eventually answer. We formalize this through the correction the image induces in the text stream. At decoder layer ℓ, let $X^\ell_q \in \mathbb{R}^{T_q \times d}$ be the hidden states of the T_q text positions in query q. Running the same layer with image attention visible and with image columns masked gives text-token outputs $O^\ell_{q,\text{img}}, O^\ell_{q,\text{noimg}} \in \mathbb{R}^{T_q \times d}$. Their difference

$$\Delta H^\ell_q = O^\ell_{q,\text{img}} - O^\ell_{q,\text{noimg}} \tag{1}$$

is the image-induced correction that must reach the query tokens. Because the two passes are identical apart from the image attention, this difference captures the entire effect of the image on the text stream at layer ℓ: as far as the answer is concerned, *all of the image's usable information rests in* ΔH^ℓ_q—once the query tokens have attended to the image, the image tokens themselves are dispensable. For one known query, a module update $\Delta W^\ell \in \mathbb{R}^{d \times d}$ satisfying

$$X^\ell_q \Delta W^{\ell\top} = \Delta H^\ell_q \tag{2}$$

can be chosen with rank at most T_q (for example, $\Delta W^{\ell\top} = X^{\ell+}_q \Delta H^\ell_q$ when the system is consistent). Our setting is harder: the image-only hypernetwork writes ΔW^ℓ before q is known. The same update must satisfy the multi-query analogue of Eq. (2)—the stacked constraints for many possible queries,

$$\begin{bmatrix} X^\ell_{q_1} \\ X^\ell_{q_2} \\ \vdots \end{bmatrix} \Delta W^{\ell\top} = \begin{bmatrix} \Delta H^\ell_{q_1} \\ \Delta H^\ell_{q_2} \\ \vdots \end{bmatrix}, \tag{3}$$

so its useful rank can grow with the diversity of questions supported by the image. Equivalently, an image's rank demand is the number of independent directions spanned by the corrections its possible queries require—a quantity we can probe, even though the full collection of future queries cannot be enumerated. These ΔH^ℓ_q are also what a trained hypernetwork ultimately learns: $\Delta W^\ell(I)$ enters the forward pass through exactly the same linear map, and end-to-end training rewards it for reproducing—through the weights—what attention

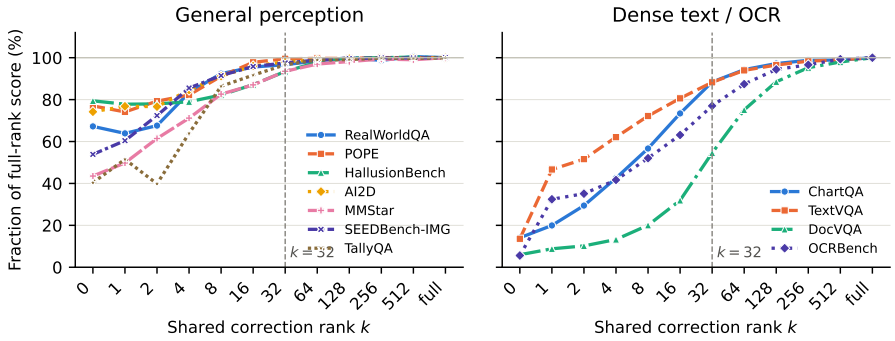

Figure 3: **Training-free rank sweep on a frozen Qwen2.5-VL-7B (Bai et al., 2025):** one projected-rollout diagnostic curve (Section 3) per benchmark, tracing the fraction of the full-rank (image-conditioned) score retained as rank k grows from the no-image floor where each curve starts to 100% at full rank. Reading each curve at a desired performance level picks the k—and hence the r a deployed head must carry.

delivers through ΔH. The rank demanded by Eq. (3) is therefore the rank the module must ultimately supply, and the required rank can be measured before training anything.

3.2 Measuring rank demand without training

An image's rank demand ranges over all the queries the image may receive —a collection no one can enumerate—so we probe it rather than read it off: we measure how much answer quality survives when the image-induced correction is restricted to a low-dimensional subspace. If correct responses survive down to a small rank budget, rank demand is genuinely low and a low-rank head suffices; if they collapse, the corrections require more directions than any low-rank head can supply.

The question we answer per dataset is: what fraction of the full-rank model's answer quality can the *best* rank-k update for an image deliver, and hence what rank does a desired performance require? We answer it by directly controlling—at inference, on the frozen model—how much of ΔH is allowed to reach the text stream. The implementation is a rank-k compression of the corrections: keep their top-k principal components and sweep k from full rank down to zero. The principal directions are read off the frozen model's own activations, so no training is involved.

We fit a separate basis per image. For each image i we ask it a battery of queries: its own native benchmark questions plus 256 pool questions drawn from other images of the same benchmark. For every fit query we extract the per-token corrections ΔH_q^ℓ (Eq. 1) and stack the rows of *all* text tokens of *all* of image i's fit queries into a single pool $D_i^\ell \in \mathbb{R}^{N_i^\ell \times d}$, where N_i^ℓ is the total number of text tokens across those queries and $d' = \min(N_i^\ell, d)$. One SVD of D_i^ℓ per image and layer,

$$D_i^\ell = A_i^\ell \, \Sigma^\ell \, U^{\ell\top}, \qquad \Sigma^\ell = \mathrm{diag}\big(\sigma_1^\ell \geq \cdots \geq \sigma_{d'}^\ell\big), \tag{4}$$

gives the image's *union basis* $U^\ell = [u_1^\ell, \ldots, u_{d'}^\ell]$ (the right singular vectors, columns ordered by singular value; the left factor A_i^ℓ is discarded)—the union of the demands of its fit queries—and its top-k right singular vectors $U_{i,k}^\ell = [u_1^\ell, \ldots, u_k^\ell] \in \mathbb{R}^{d \times k}$ form the optimistic rank-k basis. The rank k is a decision about *directions*, pooled over the fit queries of this image, and the ordering is by correction energy. Each direction u_j^ℓ is a single fixed axis of activation space, shared by every token of every fit query; only the coordinates are per-token. Writing a token's correction in this basis, row t of D_i^ℓ as $\sum_j c_{t,j} u_j^\ell$, the coefficient $c_{t,j}$ is how much of token t's correction lies along u_j^ℓ; summing the squared coordinates over all tokens t of all fit queries gives the correction energy direction u_j^ℓ carries,

$$E_j^\ell = \sum_t c_{t,j}^2, \tag{5}$$

so the direction earns a place in the basis only if many queries of this image demand it. The name *optimistic* is exact: by the Eckart–Young theorem (Eckart & Young, 1936), the rank-k reconstruction of D_i^ℓ with minimum Frobenius residual is the projection onto the span of $U_{i,k}^\ell$,

$$\min_{\mathrm{rank}(B)\le k} \left\|D_i^\ell - B\right\|_F^2 = \left\|D_i^\ell\left(I - U_{i,k}^\ell U_{i,k}^{\ell\top}\right)\right\|_F^2. \tag{6}$$

The selection acts on directions only: every token is projected with the same $U_{i,k}^\ell U_{i,k}^{\ell\top}$ and keeps its own coefficients—what is shared is the basis, not the coordinates. Corrections that pile onto the same few directions travel on shared channels and differ only in their coefficients, so the image's contribution reaches the text stream through few independent directions—which is exactly what a rank-k write can carry.

During autoregressive generation, we recompute ΔH^ℓ per layer as in Eq. (1), and continue with

$$h^{\ell+1} = O_{\text{noimg}}^\ell + \Delta H^\ell U_{i,k}^\ell U_{i,k}^{\ell\top}. \tag{7}$$

During generation the rows of $\Delta H^\ell \in \mathbb{R}^{T_q\times d}$ are the T_q text positions of the current pass (Eq. (1) with its query subscript dropped), so each per-token correction is one such row. The sweep spans the two ends of this knob: k=0 drops the correction entirely, landing on the no-image floor—identical to a model never shown the image—while full rank restores native image-conditioned inference: removing the image tokens entirely once the text tokens have attended to them leaves accuracy unchanged on each of the ten benchmarks of Table 3 (Appendix), within a margin of error. *Full performance* throughout this section—every sweep curve's 100% point—*is* that native inference, so the rank sweep measures compression against the true image-conditioned model, not an approximation of it.

Every image's basis is the best rank-k subspace available to it and is fitted on its own benchmark. A curve that still collapses at small k therefore reflects genuine rank demand, not a protocol artifact—the measured curve is a strict upper bound on what any rank-k module—not merely a rank-k LoRA head—could deliver toward the native behavior. Each image's native questions are the evaluation set. Its 256 pool questions enter the fit only: as a proxy for the variety of questions the image may eventually receive, they let the basis capture directions the native questions alone would not expose. Figure 2 diagrams the per-layer rollout.

The complete sweep (Figure 3) reveals two regimes. Reading each curve as a fraction of the full-rank (image-conditioned) score, at $k = 32$ the seven general-perception benchmarks retain 93–99% (96.5% average) and the four dense-text/OCR benchmarks retain only 55–88% (77.1% average)—but within that group the split is sharp: chart and scene text already sit at ≈88%, while document/OCR retain only 55–77% (54.6% DocVQA (Mathew et al., 2021), 77.0% OCRBench (Liu et al., 2024)). Thus the same rank ceiling is nearly lossless for general perception on average but removes nearly half of the answer-relevant signal on the hardest dense benchmarks.

3.3 Meeting the Demand: What Rank Supply Costs

The sweep measures rank demand; an output head fixes rank supply. The obvious way to give LoRA more rank is to raise r, but rank is not free: each unit of rank costs $r\times 8192$ generated values at each of the 40 active sites. The demand a deployed head must serve is the *largest* across tasks, set by the dense-text/OCR benchmarks: a rank-r update can only produce a rank-$\le r$ correction, so serving DocVQA at 80% (k_{80}=128, Fig. 3) would alone cost $128\times 8192\times 40 \approx 42$M values per image; read out of a width-512 latent, the output layer would need $42\text{M}\times 512 \approx 21$B parameters just to emit them—an output head three times larger than the 7B LLM it modulates, even on the most favorable per-image subspace. What we want, then, is a high-rank update representation that does not take up more output budget.

The contrast is concrete at the headline budget on a 4096×4096 projection (the $q/o/m$ sites; Table 5, Appendix). Both heads emit the same 262K values, so both have the same

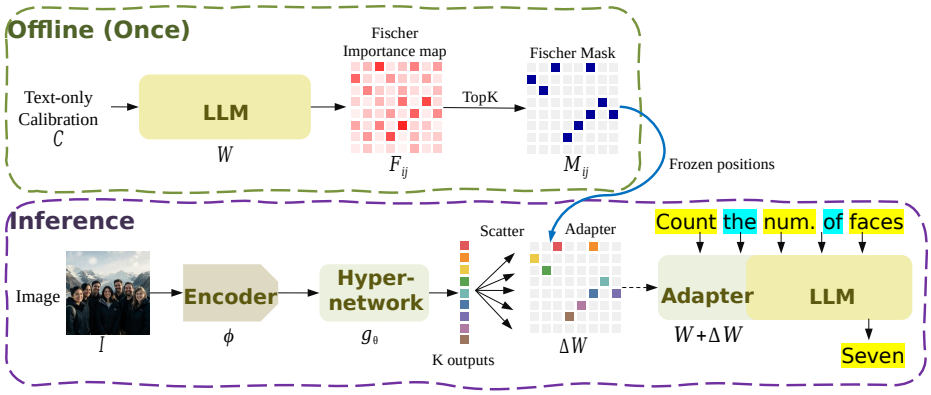

Figure 4: **SWIM spends its sparse budget where the LLM's loss is most sensitive.** *Top (offline, once):* a text-only calibration pass through the frozen LLM scores each weight's importance F_{ij} (diagonal Fisher; darker red indicates higher sensitivity), and the top-k entries define the fixed sparse mask $\hat{\mathcal{M}}_g$. *Bottom (inference):* the image-conditioned hypernetwork g_θ emits K values from the encoder features $\phi(I)$, which are scattered into ΔW at the selected locations and added to W.

parameter count; what differs is the rank of the ΔW those values compose into. A rank-32 head is confined to at most a rank 32 update that touches every weight, while sparse injection places one value per position and reaches full matrix rank on the 1.6% of positions it touches. Note how cheap full rank is under sparse support: a single diagonal of 4096 entries already has full matrix rank, and the headline budget holds 262K values—64× the size of such a diagonal—while the identical budget caps LoRA at rank 32.

4 Sparse and Combined Visual Weight Injection

Section 3 showed that the rank demand of persistent weight-space memory varies substantially across tasks and can far exceed practical LoRA budgets. The challenge is therefore not conditioning itself, but the geometry used to represent the generated update. Our goal is to increase the rank supplied by each per-image update while keeping VLoRA's (Ma et al., 2024) image encoder, hypernetwork, active modules, training recipe, and generated-output budget unchanged.

Let $g_\theta(\phi(I))$ generate an update for a selected LLM weight matrix W, where $\phi(I)$ denotes the image features extracted from image I. The downstream prompt is text-only and processed using the adapted weight

$$W' = W + \Delta W(I). \tag{8}$$

Figure 4 summarizes the complete pipeline, including the offline support-selection stage and the inference-time injection of image-conditioned updates into the language model.

LoRA as the low-rank baseline. *VLoRA's image-conditioned head emits the factor* (Ma et al., 2024) $A \in \mathbb{R}^{d_{\text{out}} \times r}$, and a learned static matrix $B \in \mathbb{R}^{d_{\text{in}} \times r}$ completes $\Delta W_{\text{L}} = AB^\top$. The head emits $r(d_{\text{in}} + d_{\text{out}})$ values, but the resulting update is constrained to rank at most r; as shown in Section 3, this rank constraint becomes a bottleneck when the rank demand of the task exceeds the rank supplied by the update.

Sparse weight injection. SWIM allocates the same output budget differently. Rather than generating low-rank factors, it writes independent values into a fixed sparse support $\mathcal{M} \subseteq [d_{\text{out}}] \times [d_{\text{in}}]$:

$$[\Delta W_{\text{S}}(I)]_{ij} = \begin{cases} [g_\theta(\phi(I))]_{ij}, & (i,j) \in \mathcal{M}, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Unlike LoRA, the rank of the resulting update is no longer tied to a factorization rank: it can reach $\min(d_{\text{out}}, d_{\text{in}})$ as long as the support spreads entries over distinct rows and

columns—already a single diagonal of $\min(d_{\text{out}}, d_{\text{in}})$ entries has full matrix rank—and both the uniform-random and Fisher-selected supports satisfy this at the budgets we train. Consequently, LoRA and SWIM can emit the same number of values while supplying radically different matrix rank.

Spending the sparse budget. If rank is no longer the limiting resource, the next question is where the sparse writes should be placed. A uniform-random support is the clean control that removes the rank constraint without introducing additional selection criteria, but not all parameters are equally useful locations. We score each candidate using the diagonal Fisher information (LeCun et al., 1989; Hassibi & Stork, 1993; Theis et al., 2018) on a 1,000-example calibration set,

$$F_{ij} = \mathbb{E}[(\partial \mathcal{L}/\partial W_{ij})^2],$$

and select

$$\mathcal{M} = \underset{S:\,|S|=k}{\arg\max} \sum_{(i,j)\in S} F_{ij}, \tag{10}$$

where k is fixed by the generated-output budget. The support is computed once on the frozen LLM and remains fixed throughout training and inference. Fisher selection is one possible importance criterion rather than a required component of sparse injection; the random-versus-Fisher ablation isolates its contribution.

SWIM + LoRA injection. Sparse and low-rank updates provide complementary structure: low-rank updates efficiently capture coordinated global changes, whereas sparse updates supply high rank under a fixed output budget. The sum can retain coordinated global directions while adding a high-rank sparse term:

$$\Delta W_{\text{H}}(I) = A(I)B(I)^{\top} + \Delta W_{\text{S}}(I). \tag{11}$$

At the headline rank-32-equivalent output budget, SWIM + LoRA assigns half to rank-16 LoRA and half to Fisher-sparse values.

Training . All methods share the Perceiver-style hypernetwork (Jaegle et al., 2021) with 40 active sites (every fourth decoder layer, $q/k/v/o$/MLP) and VLoRA's two-stage protocol on the Cambrian corpora (Tong et al., 2024) (LLM frozen in alignment, trainable in instruction tuning; Appendix A), so LoRA, SWIM, and SWIM + LoRA differ only in the geometry of the generated update.

5 Experiments

Benchmarks and evaluation. We evaluate on eleven vision–language benchmarks spanning the two regimes identified by the rank-demand analysis of Section 3. General perception and reasoning: RealWorldQA (RWQA) (xAI, 2024), POPE (Li et al., 2023), HallusionBench (Guan et al., 2024), AI2D (Kembhavi et al., 2016), MMStar (Chen et al., 2024), SEEDBench-IMG (Li et al., 2024), and TallyQA (Acharya et al., 2019); dense-text/OCR: ChartQA (Masry et al., 2022), TextVQA (Singh et al., 2019), DocVQA (Mathew et al., 2021), and OCRBench (Liu et al., 2024). Raw cells use each benchmark's native metric. To enable comparison across heterogeneous metrics, each reported group average first normalizes each column to the same-backbone LoRA result and then averages within the group; LoRA is therefore 100.0 in both groups.

5.1 Matched-budget results on two backbones

If added rank supply helps, it should show up on the high-demand tasks first. We test exactly that: LoRA, SWIM, and SWIM + LoRA under an exactly matched generation budget.

Within each backbone, LoRA, SWIM, and SWIM + LoRA use the same image encoder, LLM, data, active sites, hypernetwork, schedule, and generated-output budget. LoRA emits rank-32 factors; SWIM emits the same number of values through a Fisher-selected sparse

Table 1: **Results:** At exactly matched output budget, SWIM's gains match the diagnostic's prediction (dense norm. 105.3%/110.5% of LoRA) with general perception near parity. Columns are group averages of per-column LoRA-normalized accuracy (%), dense text/OCR and general perception; bold marks the best cell within each backbone. Full metrics: Table 4.

Backbone	Method	Dense norm.	General norm.
SigLIP-2+Llama-3	LoRA, $r = 32$	100.0	100.0
	SWIM	105.3	97.9
	SWIM + LoRA	**105.8**	**101.5**
CLIP+Vicuna	LoRA, $r = 32$	100.0	100.0
	SWIM	**110.5**	99.2
	SWIM + LoRA	106.8	99.2

support; SWIM + LoRA divides the budget equally between rank-16 LoRA and a SWIM head at the matching budget.

The gains land where the rank-demand analysis predicts they should. On SigLIP-2+Llama-3 (Tschannen et al., 2025; Grattafiori et al., 2024), *SWIM improves all four dense tasks with general perception near parity.* SWIM + LoRA preserves the dense gain (105.8% of LoRA) while recovering the general group (101.5%). On CLIP+Vicuna (Radford et al., 2021; Chiang et al., 2023), SWIM gives the largest dense gain (110.5%) with general perception at 99.2%, while SWIM + LoRA is intermediate. SWIM + LoRA is therefore the only method that improves on LoRA in both groups on the headline backbone. The added rank pays off exactly where the diagnostic flagged LoRA's constraint binding.

5.2 Where the sparse budget goes: mask-selection ablation

The matched-budget results combine two ingredients: removing the rank ceiling, and choosing informative positions. To isolate the contribution of support selection, the CLIP+Vicuna random and Fisher rows place the same sparse output budget in different locations. Normalizing each benchmark column to random SWIM (Table 6, Appendix), Fisher selection changes little on the general group but improves the dense group by 4.2%: removing the rank ceiling and selecting useful positions are separate contributions.

Table 4 reports the full per-benchmark raw metrics for Table 1; full raw CLIP+Vicuna rows and the capacity sweep are reported in Appendix F; their capacity summaries normalize each column to the same-capacity LoRA row.

6 Limitations and Future Work

Weight-only conditioning keeps sequence length and KV-cache constant as images pile up (Appendix B): one stored update per image, the backbone frozen, never re-encoding, never growing the context (Aljundi et al., 2017; Mallya & Lazebnik, 2018). Applying the diagnostic and trained updates to other persistent conditioning settings—multi-image streams and document conditioning (Snell et al., 2022; Charakorn et al., 2026)—is the natural next step.

7 Conclusion

Fine-tuning to inject knowledge is expensive (Mecklenburg et al., 2024); a hypernetwork produces its update in a single forward pass, and the update then persists as a memory that costs no context to keep. Persistence makes update rank load-bearing: one write, made before any query is known, must preserve every direction later queries may require. The projected-rollout diagnostic prices that demand before training, and SWIM meets it at LoRA's output budget—the measured demand predicts exactly where the added rank pays off. Persistent weight memories, written in a geometry that meets their measured rank demand, are a practical path toward vision–language models that learn continuously.

Ethics statement

This work studies model architecture using public vision–language benchmarks and training corpora. It does not collect human-subject data or introduce a new dataset. The resulting systems inherit the biases, privacy risks, and failure modes of their base language and vision models; weight-space conditioning does not remove those risks.

Reproducibility statement

Appendix A reports both training stages, generated-update budgets, active modules, compute, and evaluation details. Evaluation is deterministic given a checkpoint. Each trained configuration is represented by one training seed; we therefore treat small differences as directional rather than statistically established. Our code will be released with the paper.

AI use statement

Generative-AI tools assisted with code development, debugging, and editing the manuscript. The authors reviewed the resulting code, verified reported values against experiment outputs, and take responsibility for all claims and artifacts in this submission.

References

M. H. I. Abdalla, Zhipin Wang, Christian Frey, Steffen Eger, and Josif Grabocka. Zhyper: Factorized hypernetworks for conditioned LLM fine-tuning. arXiv preprint arXiv:2510.19733, 2025.

Manoj Acharya, Kushal Kafle, and Christopher Kanan. TallyQA: Answering complex counting questions. In *Proc. AAAI Conf. Artificial Intelligence (AAAI)*, 2019.

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert Gate: Lifelong learning with a network of experts. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2017.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. arXiv preprint arXiv:2502.13923, 2025.

Samyadeep Basu, Martin Grayson, Cecily Morrison, Besmira Nushi, Soheil Feizi, and Daniela Massiceti. Understanding information storage and transfer in multi-modal large language models. arXiv preprint arXiv:2406.04236, 2024.

Kartikeya Bhardwaj, Nilesh Prasad Pandey, Sweta Priyadarshi, Viswanath Ganapathy, Shreya Kadambi, Rafael Esteves, Shubhankar Borse, Paul Whatmough, Risheek Garrepalli, Mart van Baalen, Harris Teague, and Markus Nagel. Sparse high rank adapters. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Instant transformer adaption via hyperLoRA. In *NeurIPS 2024 Workshop on Adaptive Foundation Models*, 2024.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-LoRA: Instant transformer adaption. In *International Conference on Machine Learning (ICML)*, 2025.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-LoRA: Learning to instantly internalize contexts. arXiv preprint arXiv:2602.15902, 2026.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Tong Chen, Hao Fang, Patrick Xia, Xiaodong Liu, Benjamin Van Durme, Luke Zettlemoyer, Jianfeng Gao, and Hao Cheng. Generative adapter: Contextualizing language models in parameters with a single forward pass. In *International Conference on Learning Representations (ICLR)*, 2025.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality. Blog post, 2023.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, Dahua Lin, and Kai Chen. VLMEvalKit: An open-source toolkit for evaluating large multi-modality models. In *ACM Int. Conf. Multimedia (MM)*, 2024.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Aaron Grattafiori, Abhimanyu Dubey, et al. The Llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. HallusionBench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2024.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2017.

Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1993.

Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, Yaguang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng, and Ed H. Chi. HyperPrompt: Prompt-based task-conditioning of transformers. In *International Conference on Machine Learning (ICML)*, 2022.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023.

Hamish Ivison and Matthew E. Peters. HyperDecoders: Instance-specific decoders for multi-task NLP. In *Findings of the Association for Computational Linguistics: EMNLP*, 2022.

Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew Peters. HINT: Hypernetwork instruction tuning for efficient zero- and few-shot generalisation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning (ICML)*, 2021.

Tian Jin, Zhun Liu, Shengjia Yan, Alexandre Eichenberger, and Louis-Philippe Morency. Language to network: Conditional parameter adaptation with natural language descriptions. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Xiaolong Jin, Kai Wang, Dongwen Tang, Wangbo Zhao, Yukun Zhou, Junshu Tang, and Yang You. Conditional LoRA parameter generation. arXiv preprint arXiv:2408.01415, 2024.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *Eur. Conf. Comput. Vis. (ECCV)*, 2016.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1989.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR)*, 2019.

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. SEED-Bench: Benchmarking multimodal large language models. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2024.

Mengtian Li, Jinshu Chen, Wanquan Feng, Bingchuan Li, Fei Dai, Songtao Zhao, and Qian He. HyperLoRA: Parameter-efficient adaptive generation for portrait synthesis. arXiv preprint arXiv:2503.16944, 2025.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Zhiyuan Liang, Dongwen Tang, Yuhao Zhou, Xuanlei Zhao, Mingjia Shi, Wangbo Zhao, Zekai Li, Peihao Wang, Konstantin Schürholt, Damian Borth, Michael M. Bronstein, Yang You, Zhangyang Wang, and Kai Wang. Drag-and-drop LLMs: Zero-shot prompt-to-weights. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Huanxuan Liao, Shizhu He, Yao Xu, Yuanzhe Zhang, Shengping Liu, Kang Liu, and Jun Zhao. Awakening augmented generation: Learning to awaken internal knowledge of large language models for question answering. In *Int. Conf. Computational Linguistics (COLING)*, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yewei Liu, Xiyuan Wang, Yansheng Mao, Yoav Gelbery, Haggai Maron, and Muhan Zhang. SHINE: A scalable in-context hypernetwork for mapping context to LoRA in a single pass. arXiv preprint arXiv:2602.06358, 2026.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: On the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 67(12), 2024.

Chuancheng Lv, Lei Li, Shitou Zhang, Gang Chen, Fanchao Qi, Ningyu Zhang, and Hai-Tao Zheng. HyperLoRA: Efficient cross-task generalization via constrained low-rank adapters generation. In *Findings of the Association for Computational Linguistics: EMNLP*, 2024.

Feipeng Ma, Hongwei Xue, Guangting Wang, Yizhou Zhou, Fengyun Rao, Shilin Yan, Yueyi Zhang, Siying Wu, Mike Zheng Shou, and Xiaoyan Sun. Visual perception by large language model's weights. arXiv preprint arXiv:2405.20339, 2024.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018.

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

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, 2021.

Nick Mecklenburg, Yiyou Lin, Xiaoxiao Li, Daniel Holstein, Leonardo Nunes, Sara Malvar, Bruno Silva, Ranveer Chandra, Vijay Aski, Pavan Kumar Reddy Yannam, Tolga Aktas, and Todd Hendry. Injecting new knowledge into large language models via supervised fine-tuning. arXiv preprint arXiv:2404.00213, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *International Conference on Learning Representations (ICLR)*, 2023.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*, 2022.

Jesús-Germán Ortiz-Barajas, Helena Gomez-Adorno, and Thamar Solorio. HyperLoader: Integrating hypernetwork-based LoRA and adapter layers into multi-task transformers for sequence labelling. arXiv preprint arXiv:2407.01411, 2024.

Junho Park, Dohoon Kim, and Taesup Moon. PRISP: Privacy-safe few-shot personalization via lightweight adaptation. arXiv preprint arXiv:2601.06471, 2026.

Jason Phang, Yi Mao, Pengcheng He, and Weizhu Chen. HyperTuning: Toward adapting large language models without back-propagation. In *International Conference on Machine Learning (ICML)*, 2023.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. Contextual parameter generation for universal neural machine translation. In *Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pp. 8748–8763, 2021.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2019.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. arXiv preprint arXiv:2209.15189, 2022.

Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhaoxuan Tan, Zixuan Zhang, Haoyang Wen, Zheng Li, Rongzhi Zhang, Pei Chen, Fengran Mo, Zheyuan Liu, Qingkai Zeng, Qingyu Yin, and Meng Jiang. Instant personalized large language model adaptation via hypernetwork. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026.

Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. Faster gaze prediction with dense networks and fisher pruning. arXiv preprint arXiv:1801.05787, 2018.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Austin Wang, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Bartosz Trojan and Filip Gebala. HypeLoRA: Hyper-network-generated LoRA adapters for calibrated language model fine-tuning. arXiv preprint arXiv:2603.19278, 2026.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. arXiv preprint arXiv:2502.14786, 2025.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, Gertjan van Noord, and Sebastian Ruder. Hyper-X: A unified hypernetwork for multi-task multilingual transfer. In *Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Kai Wang, Dongwen Tang, Wangbo Zhao, Konstantin Schürholt, Zhangyang Wang, and Yang You. Recurrent diffusion for large-scale parameter generation. arXiv preprint arXiv:2501.11587, 2025.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. K-Adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of the Association for Computational Linguistics: ACL*, 2021.

Zexi Wu, Baolu Li, Jing Dai, Yiming Zhang, Yue Ma, Qinghe Wang, Xu Jia, and Hongming Xu. Video2LoRA: Unified semantic-controlled video generation via per-reference-video LoRA. arXiv preprint arXiv:2603.08210, 2026.

xAI. RealWorldQA: A benchmark for real-world spatial understanding. https://x.ai/blog/grok-1.5v, 2024.

Yicheng Xiao, Lin Song, Rui Yang, Cheng Cheng, Yixiao Ge, Xiu Li, and Ying Shan. LoRA-Gen: Specializing large language model via online LoRA generation. In *International Conference on Machine Learning (ICML)*, 2025.

Zedian Xiao, William Barr Held, Yanchen Liu, and Diyi Yang. Task-agnostic low-rank adapters for unseen english dialects. In *Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

Zhengkun Zhang, Wenya Guo, Xiaojun Meng, Yasheng Wang, Yadao Wang, Xin Jiang, Qun Liu, and Zhenglu Yang. HyperPELT: Unified parameter-efficient language model tuning for both language and vision-and-language tasks. In *Findings of the Association for Computational Linguistics: ACL*, 2023.

A Reproducibility Details

A.1 Training recipe

We follow VLoRA's Perceiver-style (Jaegle et al., 2021) hypernetwork and two-stage recipe (Ma et al., 2024), except that SigLIP-2+Llama-3 uses a lower Stage-1 learning rate for stable alignment. The hypernetwork has latent width 512 and depth 8. Active sites are every fourth decoder layer times $\{q, k, v, o, m\}$ (40 total): $q/k/v/o$ augment their attention projections, while m is VLoRA's parallel $d \times d$ residual branch around the MLP. The vision tower is frozen throughout; the LLM is frozen during alignment and trainable during SFT.

Table 2: **Two-stage training recipe** (identical across methods). Global batch is per-device batch × eight GPUs × gradient accumulation.

	Stage 1: alignment	Stage 2: instruction tuning
Data	Cambrian-Alignment (released with Cambrian-1), 2.5M	image subset of Cambrian-7M, ∼5.5M
LR (CLIP+Vicuna)	5×10^{-5}	2×10^{-5}
LR (SigLIP-2+Llama-3)	1×10^{-5}	2×10^{-5}
Schedule	cosine, 3% warmup	cosine, 3% warmup
Global batch / epochs	768 / 1	256 / 1
Max sequence / precision	2048 / bf16	2048 / bf16
DeepSpeed / weight decay	ZeRO-2 / 0	ZeRO-2 / 0
LLM	frozen	trainable
Trainable components	hypernetwork + projector	hypernetwork + projector + LLM

SigLIP-2 uses 1,024 visual positions and per-position input centering estimated from 1,000 Cambrian examples to reduce domination by position-dependent sink features. CLIP-L/336 uses 576 positions without centering. Headline rows generate 262K values for $q/o/m$ and 164K for GQA k/v, totaling 8,912,896 values across active sites per image. LoRA spends this budget on rank-32 factors, SWIM on Fisher-selected entries, and SWIM + LoRA half on rank-16 factors and half on sparse values.

A representative chain takes 9h30m for alignment and 46h for SFT on eight H100s, about 445 H100 GPU-hours per trained row. Ten primary and ablation chains total approximately 4,500 H100 GPU-hours.

A.2 Evaluation and diagnostic checks

Evaluation uses VLMEvalKit (Duan et al., 2024), greedy model decoding, and a temperature-zero Qwen3-32B-AWQ judge (Yang et al., 2025) where required. TextVQA, DocVQA, and OCRBench use their standard heuristic scorers. Evaluation is deterministic given a checkpoint; uncertainty comes from training runs. Removing the image tokens after the text tokens have attended to them leaves performance essentially unchanged relative to the vanilla forward pass, within a margin of error on every benchmark checked (Table 3).

Table 3: **Removing the image tokens once the text tokens have attended to them is lossless**: accuracy of the vanilla pass versus the same run with image tokens dropped after attention, scored on the same examples (Qwen2.5-VL-7B), is within a margin of error on all ten benchmarks—so the rank sweep's full-rank endpoint *is* the vanilla model.

	General perception						Dense text / OCR			
Benchmark	RealWorld	POPE	Halluc.	AI2D	MMStar	SEED	ChartQA	TextVQA	DocVQA	OCRBench
Vanilla	68.63	91.15	72.03	84.40	61.60	76.90	86.32	92.55	92.05	85.90
Image tokens removed	68.76	91.15	72.03	84.45	61.73	76.95	87.00	92.10	92.20	85.70

Table 4: **Full results behind Table 1:** leading columns are per-column LoRA-normalized group averages (%); raw native metrics follow, dense text/OCR before general perception; SWIM + LoRA = rank-16 LoRA + a Fisher-masked SWIM head at the matching budget; bold marks the best cell within each backbone.

		Norm. (%)		Dense text / OCR				General perception						
Backbone	Method	Dense	Gen.	ChartQA	TextVQA	DocVQA	OCRBench	RWQA	POPE	Hallus.	AI2D	MMStar	SEED	TallyQA
SigLIP-2+Llama-3	LoRA, $r = 32$	100.0	100.0	30.1	42.9	26.7	30.4	**52.4**	**83.8**	49.6	**67.1**	38.5	**60.4**	67.8
	SWIM	105.3	97.9	**33.8**	43.0	27.5	32.1	50.1	82.1	47.4	66.2	38.3	59.0	68.0
	SWIM + LoRA	**105.8**	**101.5**	33.1	**43.3**	**28.0**	**32.6**	51.8	83.4	**50.1**	66.5	**42.0**	59.9	**70.2**
CLIP+Vicuna	LoRA, $r = 32$	100.0	100.0	19.0	23.2	17.7	20.4	**46.8**	80.3	**46.9**	**63.9**	**36.7**	**56.4**	64.2
	SWIM	**110.5**	99.2	**21.8**	**25.5**	18.5	**23.0**	46.6	79.9	45.9	62.9	35.8	56.1	**65.6**
	SWIM + LoRA	106.8	99.2	20.4	24.4	**18.7**	22.2	46.1	**80.9**	**46.9**	63.4	34.7	56.3	65.2

B Efficiency Ledger

A persistent weight memory adds no visual tokens to the LLM sequence: per-image updates are summed and merged once (Ilharco et al., 2023), after which the LLM sees a fixed text-only sequence however many images it remembers, while a visual-token prefix adds roughly a thousand visual tokens per image. The ledger is measured on an H100 using the SigLIP-2+Llama-3 SWIM + LoRA checkpoint, and backs the single-image systems comparison of the teaser (Figure 1); Figure 5 reports the measurements, and the point is structural rather than numerical. The two paths differ in *what must be re-paid per query*: weight memory pays its image cost once, as an image-to-weights write plus merge, and every query—including follow-ups about the same images—then runs on text-only prefill, while the prefix re-runs encode-plus-prefill over every remembered image's tokens on every query. The consequences follow: sequence length and KV-cache size stay constant along the weight-memory path but grow linearly with the image count along the prefix, and time to first token rises only with the linear write cost versus steeply with the re-paid per-image prefill, leaving the two paths an order of magnitude apart by eight images. These are systems measurements, not a multi-image accuracy claim: composition-aware accuracy training remains future work (Section 6).

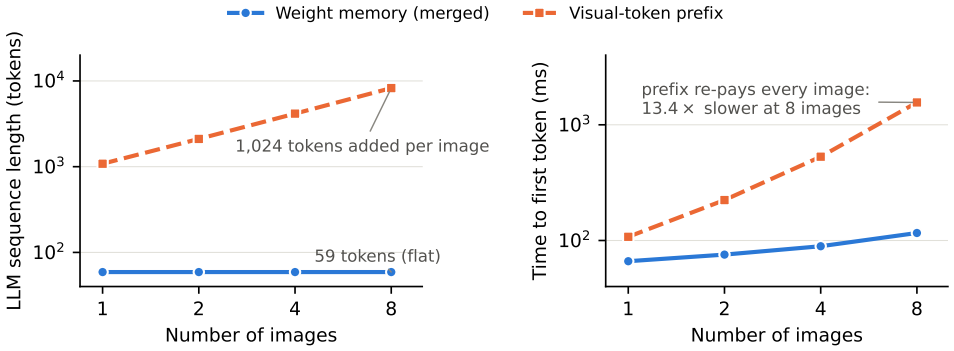

Figure 5: **Multi-image scaling:** weight memory stays flat in the number of images (left: 59 tokens always); a visual-token prefix grows linearly. *Right:* time to first token — the write plus text prefill for weight memory (encoder, hypernetwork, one-time merge, 59-token prefill; paid once per image set, since follow-up queries re-run only the 3.7 ms text prefill), versus encode-plus-1024-tokens-per-image prefill for the prefix, re-paid on every query; both exclude the first decode step, which costs the same on the two paths. The prefix baseline uses the same encoder and LLM.

Table 5: **The same output budget buys wildly different rank**: on a 4096×4096 target weight, the low-rank head is confined to rank ≤ 32 while sparse injection can reach rank 4096; SWIM + LoRA splits the budget and inherits the sparse head's rank.

Head	Output values	Rank of ΔW	Weights modulated
LoRA	$2 \times 4096 \times 32 = 262\text{K}$	≤ 32	all 16.8M (100%)
SWIM	262K	≤ 4096	262K (1.6%)
SWIM + LoRA	$2 \times 4096 \times 16 + 131\text{K} = 262\text{K}$	≤ 4096	all 16.8M (100%)

C Full Matched-Budget Results

D Output Budget vs Rank

E Mask-Selection Ablation

Table 6: **Removing the rank ceiling and choosing good positions are separate wins**: Fisher selection adds +4.2% on dense text/OCR over a random mask at the same 262K budget. Columns normalized per benchmark to random SWIM (CLIP+Vicuna).

Mask	General norm.	Dense norm.
Uniform random	100.0	100.0
Fisher top-k	100.4	**104.2**

F CLIP+Vicuna Capacity Sweep

Raw per-benchmark values for the matched 262K comparison appear in Table 4. Table 7 summarizes the wider capacity sweep after normalizing every benchmark to same-capacity LoRA before averaging. The dense-text advantage is consistent across budgets.

Table 7: **The dense-text advantage holds at every budget from 16K to 262K generated values** (CLIP+Vicuna capacity sweep, LoRA-normalized per benchmark (%); each LoRA cell is 100 by construction; bold marks the best value within each budget).

Budget	Method	General norm.	Dense norm.
16K	LoRA $r = 2$	100.0	100.0
	SWIM random	**100.5**	108.2
	SWIM Fisher	99.7	**109.8**
65K	LoRA $r = 8$	**100.0**	100.0
	SWIM random	99.3	105.5
	SWIM Fisher	99.6	**110.7**
262K	LoRA $r = 32$	**100.0**	100.0
	SWIM random	98.4	106.1
	SWIM Fisher	98.7	**110.5**