TWO VECTORS REPLACE IN-CONTEXT DEMOS: STRUCTURED TASK ADAPTATION VIA EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context learning (ICL) adapts frozen large multimodal models (LMMs) to new tasks from a few demonstrations (demos), but re-encodes them at every query, where each demo image adds up to hundreds of visual tokens. Demo-free methods remove this cost with a compact task state. However, they add it at locations searched per task or at every decoder layer, where task parameters grow with depth. Moreover, inserted tokens or keys cannot change how the original prompt divides its attention within a layer. To address these issues, we propose Structured Task Adaptation via Embeddings (**STAVE**), which replaces demos with two task-specific vectors added to existing input embeddings. Specifically, a readout vector updates the answer-producing tokens and a context vector updates the other structural token groups. Both are trained with answer labels on prompts with and without demos. We justify these design choices theoretically using a first-order analysis of the loss and a margin bound. Extensive experiments on six LMMs and five large language models show that STAVE matches or outperforms state-of-the-art methods on multimodal tasks with far fewer task parameters and surpasses 15-shot ICL and prior task vectors on 18 text tasks, all at zero-shot inference cost.

1 INTRODUCTION

In-context learning (ICL) lets a frozen large language model (LLM) perform a new task from a few input–output demonstrations (demos) placed in its prompt (Brown et al., 2020; Dong et al., 2024). Large multimodal models (LMMs) inherit this ability (Alayrac et al., 2022; Laurençon et al., 2024b; Li et al., 2024b), so one frozen model can handle visual question answering (VQA) or image captioning without training. This flexibility is costly, since every query re-encodes its demos (Liu et al., 2022a) and each demo image expands into up to hundreds of visual tokens. Its accuracy also depends on which demos are chosen and in what order they appear (Zhao et al., 2021). We study how to adapt a frozen model to a task at the prompt length and cost of zero-shot inference.

Demo-free methods store the task in a compact state added to the frozen model at inference. Extracted task vectors take this state from demo activations. The task vector (TV) (Hendel et al., 2023) and the function vector (FV) (Todd et al., 2024) add one vector at an intermediate layer, while MTV (Huang et al., 2024) and STV (Ma et al., 2026) place it at attention heads selected for each task. Trained interventions instead optimize the state with supervision. LIVE (Peng et al., 2024) learns a shift vector for every decoder layer, and MimIC (Jiang et al., 2025) and HiFICL (Li et al., 2026) learn per-head shifts or virtual keys and values in every layer to approximate the effect of demos on the query. Prompt tuning (PT) (Lester et al., 2021) inserts learned virtual tokens into the input prompt.

These methods remove the demos at inference, yet two limitations remain. ***First*, the task state enters either at a location searched for each task or at every decoder layer.** A vector added at an intermediate layer cannot act through the blocks before it, and its layer or heads must be chosen for each task. Trained interventions avoid this choice by adding a shift to every layer, so their task parameters and per-query computation grow with model depth. ***Second*, methods that insert tokens or keys, such as PT and HiFICL, cannot change how the original prompt divides its attention within a layer.** For fixed queries and keys of the original tokens, added keys scale all their attention weights by one factor, so the relative attention between the image and the question stays unchanged (Petrov et al., 2024). PT must also choose where to insert its tokens, before the prompt, after it, or before its final token, and each inserted token lengthens the prompt and its computation.

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.

To address these limitations, we ask two questions: where a demo-free update should enter a frozen model, and how it can be shared across tokens without losing answer-relevant directions. For the first, only an update at the input embeddings acts through every decoder block, and to first order an unrestricted input update can reproduce the first-answer effect of shifts in every layer. Added to existing tokens, the update keeps the prompt length and changes their attention queries and keys, so it can reweight the attention between the image and the question. For the second, structural token groups, such as the question text and the query image, recur in every prompt of a task, so one vector per group serves every query. Sharing has a cost: the final prompt token predicts the first answer token, whereas the other tokens reach it only through attention and still hold query-specific evidence, and a vector shared with many weak-gradient tokens has a diluted first-order loss decrease.

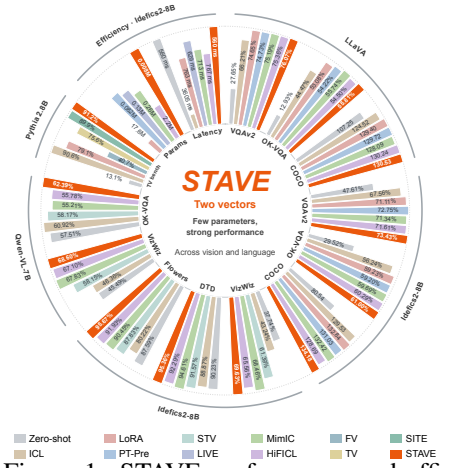


Figure 1: STAVE performance and efficiency. Bars are relative to STAVE, and params use a log scale (Appendix E.3).

Based on this analysis, we propose **S**tructured **T**ask **A**daptation **v**ia **E**mbeddings (STAVE), which adds two task-specific vectors to the input embeddings of a frozen LLM or LMM. A readout vector updates only the final prompt token and the answer cue, and a context vector covers the remaining groups. Both are trained with answer labels instead of an ICL teacher, on queries with and without demos, and added once to the demo-free prompt.

Contributions. ❶ **Framework.** We propose STAVE, which replaces in-context demos with two vectors added to the input embeddings of a frozen model, so its task parameters do not grow with model depth and it runs at zero-shot inference cost. ❷ **Theoretical Explanation.** We show that only an input update reaches every decoder block and that weak-gradient tokens dilute a shared vector. We bound the margin lost by tying structural token groups into two vectors, and prove that a small gradient step on prompts with and without demos lowers the demo-free loss when the two gradients align. ❸ **Extensive Evaluations.** On multimodal and text tasks, STAVE matches or outperforms per-layer interventions with far fewer task parameters and surpasses ICL and task vectors without demos at inference (Figure 1), and ablations support each design choice. ❹ **Insight.** The gain comes from separating the answer-producing tokens from their context, not from a modality split or a second vector. The readout vector sets a shared answer format, the context vector supplies query-specific evidence, and together they mainly help the frozen model output answers it already holds.

2 Motivating Analysis

We answer the two questions of Section 1 for an additive prompt update, which changes a frozen decoder's output without demos. Its *reach* decides where it should enter the model, and the *first-order cost of sharing* decides how to share it across tokens without losing answer-relevant directions.

Proposition 2.1 (Reach of a prompt update). Fix a prompt and teacher-forced answer prefixes in a differentiable causal decoder with N_{layer} blocks and a positionwise final head. Depth zero denotes input embeddings and depth ℓ the output of block ℓ. **(i)** A prompt-only update at depth ℓ can act through later blocks and the head. After the last block, with stored keys and values unchanged, *only the final prompt token can affect answer logits, and only the first prediction*. **(ii)** If all intervening own-token Jacobians are invertible, any infinitesimal change of all depth-ℓ prompt states has a unique unrestricted per-token realization at an earlier depth.

Reach. Thus *input injection can affect every block*. By part (ii), an unrestricted per-prompt input update matches the first-answer effect of simultaneous per-layer shifts to first order (Appendix B.1).

Adding to existing tokens also keeps the sequence length. At a fixed attention layer, added keys scale all weights on the original tokens by one common factor when their queries, keys, and positional scores stay fixed, which preserves the ratios between them (Petrov et al., 2024). An embedding update can change those queries and keys and hence reorder the weights (Appendix B.2). However, a per-prompt update does not transfer to new queries, so reuse requires shared vectors.

Sharing. Fix a differentiable loss and let $g_t \in \mathbb{R}^d$ be its gradient at input embedding t, where d is the hidden dimension. For a nonempty token set $\mathcal{S}$ and budget $\rho > 0$, one vector shared as $w/\sqrt{|\mathcal{S}|}$ has total update norm $\|w\|_2$. Its largest first-order loss decrease is

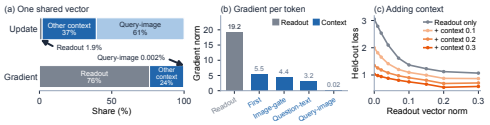


Figure 2: One vector falls short and two do better.

$$\Gamma(\mathcal{S}) = \frac{\rho}{\sqrt{|\mathcal{S}|}} \left\| \sum_{t \in \mathcal{S}} g_t \right\|_2 \le \rho \left(\sum_{t \in \mathcal{S}} \|g_t\|_2^2 \right)^{1/2}. \tag{1}$$

Free per-token updates of equal total norm attain the right side. Equality holds only when all g_t agree. For disjoint nonempty $\mathcal{A}, \mathcal{B}$, two vectors at equal total budget attain $\sqrt{\Gamma(\mathcal{A})^2 + \Gamma(\mathcal{B})^2} \ge \Gamma(\mathcal{A} \cup \mathcal{B})$. If the gradients on $\mathcal{B}$ vanish, one shared vector attains only $\sqrt{|\mathcal{A}|/|\mathcal{A} \cup \mathcal{B}|}\,\Gamma(\mathcal{A})$ (Proposition C.2). Figure 2 measures this in Idefics2 on Flowers before training (Appendix C.4). One vector on all tokens splits its update by token count, so image tokens, with almost no gradient, receive most of it, and the final token and answer cue, with 76% of the gradient, receive under 2% (Figure 2(a)).

Design. STAVE therefore separates a *readout set* $\mathcal{A}$, the final prompt token and answer cue, from the remaining *context tokens* $\mathcal{B}$. The final token supplies the first-token readout (Proposition 2.1(i)), and $\mathcal{A}$ has the largest mean gradient per token (Figure 2(b)), so a vector on $\mathcal{A}$ alone is not diluted by $\mathcal{B}$. Context tokens reach the readout only through attention and have a descent direction nearly orthogonal to that of $\mathcal{A}$ (Appendix C.4), so one shared vector cannot point along both. Along these directions, the held-out loss levels off with the readout vector alone, whereas each added context vector norm lowers the whole curve (Figure 2(c)). Section 3 builds both vectors from reusable structural token groups and bounds the margin cost of tying groups within each vector (Theorem 3.1).

3 METHOD

STAVE adds two task-specific vectors to existing embeddings of a frozen autoregressive model. A *readout vector* updates the final prompt token and answer cue, and a *context vector* updates the other eligible tokens. Both are trained with and without demos and used without them (Figure 3).

3.1 STRUCTURAL EMBEDDING UPDATES

Let f_θ have frozen parameters θ and hidden dimension d. Prompt p has input $E(p) \in \mathbb{R}^{T(p) \times d}$, where $T(p)$ counts tokens after modality expansion. A fixed, label-free map $s_t(p) \in \{0, \dots, R\}$ assigns each token to a *structural token group* or to zero for exclusion. The $R = 7$ groups are LAST (final prompt token), ANSWER-CUE (template tokens introducing the answer before it), FIRST (designated first token), IMAGE-GATE (query-image delimiters), QUERY-IMAGE (expanded query-image embeddings), QUESTION-TEXT (remaining query text), and PREFIX-TEXT (instruction prefix).

Beginning-of-sequence (BOS) tokens, padding, answer tokens, and unmatched tokens are excluded. Demo tokens are also excluded, except FIRST when the template places it in a demo. FIRST is the first attended non-BOS token, or the first query-text token in captioning. Appendix C.1 gives the precedence and template conventions. Remaining tokens are eligible, and groups may be empty.

For $1 \le q \le R$ learned vectors, let $a(k) \in \{1, \dots, q\}$ assign group k to a vector and set $a(0) = 0$. Let $n_k(p)$ count group k and $N_r(p) = \sum_{k:a(k)=r} n_k(p)$ count all tokens assigned to vector r. Define

$$\widetilde{E}(p, W) = E(p) + H(p)W, \qquad H_{tr}(p) = \begin{cases} \mathbf{1}\{a(s_t(p)) = r\}/\sqrt{N_r(p)}, & N_r(p) > 0, \\ 0, & N_r(p) = 0. \end{cases} \tag{2}$$

Here $W \in \mathbb{R}^{q \times d}$ holds one task's vectors as rows, $H(p) \in \mathbb{R}^{T(p) \times q}$ distributes them, and $\mathbf{1}$ is the indicator. STAVE sets $q = 2$: LAST and ANSWER-CUE share the readout vector, and the other five groups share the context vector. Each vector is normalized once by its merged token count $N_r(p)$.

Nonempty columns of $H(p)$ have disjoint supports and unit norm. Hence $\|H(p)W\|_F \le \|W\|_F$, with equality when $N_r(p) > 0$ for every r. Replication thus cannot grow the injected energy. At $W = 0$, *the original computation is recovered exactly*. For a differentiable loss F, the gradient of vector r is $N_r(p)^{-1/2} \sum_{t:a(s_t(p))=r} \nabla_{\widetilde{E}_t} F$, where $\widetilde{E}_t$ is the updated embedding of token t. At $W = 0$, ρ times its norm equals Γ of that vector's token set in Equation 1 (Appendix C.2).

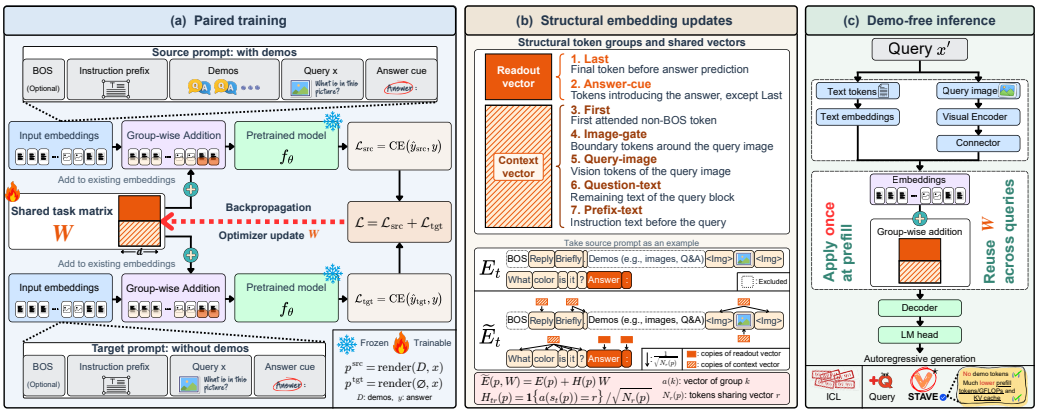


Figure 3: Illustration of the proposed framework STAVE in the multimodal setting.

3.2 Margin cost of tying groups

To compare tying with per-group updates, fix finitely many prompts and correct answer prefixes with equal group counts n_k. Scale each group's vector by $1/\sqrt{n_k}$ and stack them into $v \in \mathbb{R}^{Rd}$, where empty groups have no effect. Equation 2 restricts v to a count-weighted subspace $\mathcal{U}_q$ with orthogonal projector P_q. With zero empty-row parameters, the learned-row norm is $\|v\|_2$ (Appendix C.5).

Let $h_j(v)$ be correct-token minus competing-token logit margin j, for a fixed collection of $N \geq 1$ comparisons across these contexts. For radius $\rho > 0$, define

$$m_q(\rho) = \max_{v \in \mathcal{U}_q,\, \|v\|_2 \leq \rho} \min_{1 \leq j \leq N} h_j(v), \qquad \delta_q = \max_j \|(I - P_q)\nabla h_j(0)\|_2. \tag{3}$$

Here m_q is the *best worst-case margin* in the tied class, I is the identity, and δ_q measures sensitivity to discarded group contrasts. Define m_R using the unrestricted group-coordinate ball.

Theorem 3.1 (Cost of tying structural token groups). Under the common-count setup above, suppose each h_j is differentiable on a neighborhood of $\{v : \|v\|_2 \leq \rho\}$ and its gradient is κ-Lipschitz on that ball, with $\kappa \geq 0$. Then

$$0 \leq m_R(\rho) - m_q(\rho) \leq \rho\delta_q + \kappa\rho^2. \tag{4}$$

For affine margins, $\delta_q = 0$ suffices for equal attainable margins.

The proof projects an unrestricted update onto $\mathcal{U}_q$ and bounds the omitted linear response and curvature. For readout/context sharing, δ_q measures count-weighted gradient variation within the two sets, and their mutual contrast is kept. *Small δ_q and κ therefore give a small margin cost.*

Equation 2 and the objective are defined for any group counts. When group proportions vary across prompts, the theorem applies within each common-count collection.

3.3 Paired supervision

For each training query x_i with answer y_i, render $p_i^{\text{src}} = \text{render}(D_i, x_i)$ with demos D_i and $p_i^{\text{tgt}} = \text{render}(\varnothing, x_i)$ without them. Both retain the query modality. For answer $y = (a_1, \ldots, a_L)$ of length $L \geq 1$, use

$$\ell(y \mid p, W) = -\frac{1}{L}\sum_{l=1}^{L} \log p_\theta(a_l \mid \widetilde{E}(p, W), a_{<l}), \tag{5}$$

$$\mathcal{L}(W) = \mathcal{L}_{\text{src}}(W) + \mathcal{L}_{\text{tgt}}(W), \qquad \mathcal{L}_b(W) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i \mid p_i^b, W). \tag{6}$$

Here p_θ is the frozen next-token distribution, $a_{<l}$ is the correct prefix, n counts training query–answer pairs, and $b \in \{\text{src}, \text{tgt}\}$ selects a branch. The two sequence cross-entropy (CE) losses are equally

weighted. Both branches of each paired minibatch contribute before an optimizer step. The vectors start at zero, and backbone parameters and answer-token input embeddings get no direct update.

The *target branch* supervises deployment inputs. The *source branch* fits the same vectors with demos present, using answer labels rather than soft labels or hidden-state targets. This adds a *compatibility requirement*, and Proposition 3.2 bounds its effect on the target loss after one gradient step.

Proposition 3.2 (Target descent under paired supervision). Stack the learned vectors into $w \in \mathbb{R}^{qd}$ and write $g_b = \nabla_w \mathcal{L}_b(w)$ and $g = g_{\mathrm{src}} + g_{\mathrm{tgt}}$. If $\nabla \mathcal{L}_{\mathrm{tgt}}$ is β-Lipschitz on the segment from w to $w - \eta g$, where $\beta \geq 0$ and $\eta > 0$, then

$$\mathcal{L}_{\mathrm{tgt}}(w - \eta g) - \mathcal{L}_{\mathrm{tgt}}(w) \leq -\eta \langle g_{\mathrm{tgt}}, g \rangle + \frac{\beta \eta^2}{2} \|g\|_2^2. \tag{7}$$

Thus $\|g_{\mathrm{tgt}}\|_2^2 + \langle g_{\mathrm{tgt}}, g_{\mathrm{src}} \rangle > 0$ gives target descent for sufficiently small gradient steps within the smoothness region. *Positive source–target alignment* also improves the first-order decrease relative to a target-only step. Appendix D.1 proves both statements and gives finite-step conditions.

For the whole objective, restrict W to a Frobenius ball and define the *source–target compatibility gap* $\mathcal{C}_{\mathrm{src|tgt}}$ as the smallest excess source loss among target minimizers. Any $\widehat{W}$ within $\varepsilon_{\mathrm{opt}} \geq 0$ of $\min \mathcal{L}$ then has excess target loss at most $\mathcal{C}_{\mathrm{src|tgt}} + \varepsilon_{\mathrm{opt}}$, and the gap is zero when the two branches share a minimizer (Appendix D). Under independent target sampling and bounded-loss and Lipschitz assumptions, Appendix D.3 adds a uniform estimation term with covering dimension $qd = 2d$. This separates representation size from compatibility, optimization, and statistical error.

3.4 Demo-free inference and cost

In LLMs, STAVE updates frozen input embeddings. In LMMs, it updates merged text and image embeddings after the frozen visual encoder and connector. At inference, add both vectors once during demo-free prefill. Generated embeddings receive no update. Storage is $2d$ task-specific scalars, independent of query count and demo length. Group counts are built from template metadata in $O(T(p))$ time, and adding the scaled vectors costs $O(T(p)d)$ operations. Prompt length and decoder prefill dimensions *equal those of zero-shot inference*, as does key–value (KV) cache shape at a fixed generated length. Training is a one-time cost: both branches run the frozen model and backpropagate to the two vectors, with no backbone optimizer states or weight updates (Appendix E).

4 Experiments

4.1 Setup

Datasets, backbones, and protocols. We evaluate six frozen LMMs, LLaVA-Interleave-7B (Li et al., 2024b), Idefics2-8B (Laurençon et al., 2024b), Qwen-VL-7B (Bai et al., 2023), InternVL3.5-8B (Wang et al., 2025), Qwen2.5-VL-7B (Bai et al., 2025), and Idefics3-8B (Laurençon et al., 2024a). The tasks are VQA with VQAv2 (Goyal et al., 2017), OK-VQA (Marino et al., 2019), and VizWiz (Gurari et al., 2018), image captioning with the COCO Karpathy split (Lin et al., 2014; Karpathy & Fei-Fei, 2015), and fine-grained classification with DTD (Cimpoi et al., 2014), Flowers (Nilsback & Zisserman, 2008), and CUB (Wah et al., 2011). We also evaluate five frozen LLMs, Pythia 2.8B, 6.9B, and 12B (Biderman et al., 2023), GPT-J 6B (Wang & Komatsuzaki, 2021), and LLaMA 7B (Touvron et al., 2023), on the TV benchmark, the 18 algorithmic, translation, linguistic, and knowledge tasks of Hendel et al. (2023). Each comparison follows the most common protocol for its benchmark. Table 1 follows MimIC (Jiang et al., 2025) and HiFICL (Li et al., 2026), with 1,000 training samples that also form the demo pool and evaluation on the validation split. Table 2 and Table 10 in the appendix follow MTV (Huang et al., 2024) and STV (Ma et al., 2026), with a held-out test split. Table 3 follows Hendel et al. (2023), with at most 256 training, 64 development, and 200 test queries per task. All trained methods answer without demos at test time. COCO is scored by CIDEr (Vedantam et al., 2015) and all other tasks by accuracy.

Compared methods. Besides zero-shot and n-shot ICL, we compare STAVE with three families of methods. Parameter-efficient fine-tuning includes LoRA (Hu et al., 2021), prefix tuning (Li & Liang, 2021), and PT (Lester et al., 2021) in three modes, with its tokens before the prompt (PT-Pre), after it

Table 1: Comparison on VQAv2, OK-VQA, and COCO (left), with qualitative examples (right). # Params: low, medium, high. Best: **bold**, second: underlined.

LLaVA	# Params (M)	VQAv2	OK-VQA	COCO
Zero-shot	–	27.65$_{\pm 0.00}$	12.93$_{\pm 0.00}$	107.25$_{\pm 0.00}$
8-shot ICL	–	68.61$_{\pm 0.20}$	48.43$_{\pm 0.42}$	118.88$_{\pm 0.77}$
16-shot ICL	–	66.21$_{\pm 0.48}$	44.47$_{\pm 0.46}$	124.52$_{\pm 1.05}$
32-shot ICL	–	62.60$_{\pm 1.01}$	39.39$_{\pm 0.52}$	124.56$_{\pm 0.73}$
LoRA	19.7 (×2399.00)	74.65$_{\pm 0.47}$	53.08$_{\pm 0.41}$	129.40$_{\pm 1.01}$
PT-Pre	0.082 (×10.00)	74.73$_{\pm 0.09}$	54.22$_{\pm 0.17}$	129.72$_{\pm 0.08}$
PT-App	0.082 (×10.00)	74.68$_{\pm 0.24}$	53.46$_{\pm 0.33}$	127.98$_{\pm 0.80}$
PT-BL	0.082 (×10.00)	74.47$_{\pm 0.21}$	53.91$_{\pm 0.50}$	128.33$_{\pm 0.47}$
LIVE	0.13 (×16.00)	75.15$_{\pm 0.03}$	53.69$_{\pm 0.19}$	129.75$_{\pm 0.40}$
MimIC	17.0 (×2080.13)	75.19$_{\pm 0.16}$	55.74$_{\pm 0.25}$	128.09$_{\pm 0.25}$
HiFICL	2.2 (×272.00)	75.36$_{\pm 0.30}$	54.50$_{\pm 0.52}$	130.24$_{\pm 0.32}$
STAVE	**0.008** (×**1.00**)	**76.07**$_{\pm 0.10}$	**55.81**$_{\pm 0.10}$	**130.53**$_{\pm 1.17}$
Idefics2-8B	# Params (M)	VQAv2	OK-VQA	COCO
Zero-shot	–	47.61$_{\pm 0.00}$	29.52$_{\pm 0.00}$	80.54$_{\pm 0.00}$
8-shot ICL	–	67.44$_{\pm 0.30}$	57.35$_{\pm 0.20}$	119.25$_{\pm 0.42}$
16-shot ICL	–	67.56$_{\pm 0.74}$	56.24$_{\pm 0.46}$	129.53$_{\pm 0.44}$
32-shot ICL	–	53.77$_{\pm 0.76}$	52.05$_{\pm 1.17}$	122.36$_{\pm 0.32}$
LoRA	17.6 (×2147.00)	71.11$_{\pm 0.87}$	59.23$_{\pm 0.21}$	132.84$_{\pm 0.57}$
PT-Pre	0.082 (×10.00)	72.75$_{\pm 0.14}$	59.20$_{\pm 0.44}$	131.03$_{\pm 1.51}$
PT-App	0.082 (×10.00)	70.75$_{\pm 0.56}$	55.83$_{\pm 1.78}$	131.20$_{\pm 1.59}$
PT-BL	0.082 (×10.00)	71.04$_{\pm 0.38}$	57.82$_{\pm 0.84}$	131.13$_{\pm 1.12}$
LIVE	0.13 (×16.00)	69.73$_{\pm 1.30}$	58.22$_{\pm 0.40}$	128.20$_{\pm 0.84}$
MimIC	0.26 (×32.13)	71.34$_{\pm 0.05}$	59.69$_{\pm 0.37}$	132.42$_{\pm 0.13}$
HiFICL	2.2 (×272.00)	71.61$_{\pm 0.68}$	60.29$_{\pm 0.53}$	128.69$_{\pm 0.14}$
STAVE	**0.008** (×**1.00**)	**73.43**$_{\pm 0.06}$	**61.00**$_{\pm 0.41}$	**134.18**$_{\pm 0.82}$

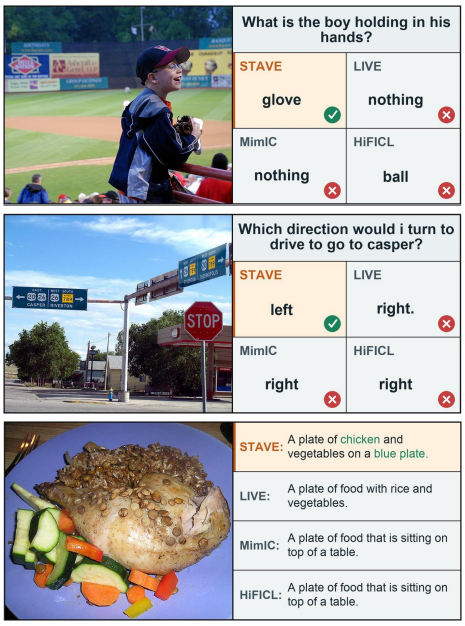


Table 2: Comparison on VQA and classification datasets.

		Idefics2-8B						**Qwen-VL-7B**					
Method	# Params (M)	VizWiz	OK-VQA	DTD	Flowers	CUB	**Avg.**	VizWiz	OK-VQA	DTD	Flowers	CUB	**Avg.**
Zero-shot	–	37.74$_{\pm 0.00}$	49.44$_{\pm 0.00}$	90.23$_{\pm 0.00}$	87.59$_{\pm 0.00}$	87.84$_{\pm 0.00}$	70.57$_{\pm 0.00}$	45.49$_{\pm 0.00}$	57.51$_{\pm 0.00}$	83.77$_{\pm 0.00}$	73.88$_{\pm 0.00}$	91.10$_{\pm 0.00}$	70.35$_{\pm 0.00}$
4-shot ICL	–	43.24$_{\pm 0.11}$	50.38$_{\pm 0.19}$	88.87$_{\pm 0.12}$	80.22$_{\pm 0.40}$	85.91$_{\pm 0.12}$	69.72$_{\pm 0.05}$	46.36$_{\pm 0.32}$	60.92$_{\pm 0.07}$	84.44$_{\pm 0.25}$	87.17$_{\pm 0.05}$	88.73$_{\pm 0.25}$	73.53$_{\pm 0.12}$
PT-Pre	0.082 (×10.00)	66.19$_{\pm 1.26}$	54.49$_{\pm 2.57}$	94.64$_{\pm 0.35}$	87.50$_{\pm 0.41}$	94.25$_{\pm 0.53}$	79.41$_{\pm 0.77}$	63.83$_{\pm 0.82}$	50.56$_{\pm 0.73}$	94.44$_{\pm 0.85}$	93.83$_{\pm 0.26}$	95.65$_{\pm 0.71}$	79.66$_{\pm 0.29}$
PT-App	0.082 (×10.00)	64.56$_{\pm 2.43}$	55.10$_{\pm 0.56}$	94.14$_{\pm 0.07}$	89.60$_{\pm 0.10}$	95.10$_{\pm 0.15}$	79.70$_{\pm 0.63}$	60.19$_{\pm 3.26}$	37.84$_{\pm 4.11}$	93.10$_{\pm 0.96}$	93.56$_{\pm 0.28}$	94.14$_{\pm 0.43}$	75.77$_{\pm 1.03}$
PT-BL	0.082 (×10.00)	66.35$_{\pm 0.21}$	56.58$_{\pm 0.98}$	94.13$_{\pm 0.19}$	88.97$_{\pm 0.52}$	94.55$_{\pm 1.54}$	80.12$_{\pm 0.22}$	60.30$_{\pm 4.64}$	44.35$_{\pm 3.61}$	93.44$_{\pm 0.28}$	93.49$_{\pm 0.29}$	94.73$_{\pm 0.50}$	77.26$_{\pm 0.48}$
TV	**0.004** (×**0.50**)	46.29$_{\pm 6.90}$	50.23$_{\pm 0.90}$	90.58$_{\pm 0.49}$	87.52$_{\pm 0.02}$	87.99$_{\pm 0.26}$	72.52$_{\pm 1.41}$	49.37$_{\pm 2.41}$	59.03$_{\pm 0.68}$	91.00$_{\pm 1.80}$	93.59$_{\pm 0.06}$	92.69$_{\pm 1.36}$	77.14$_{\pm 0.83}$
ICV	0.13 (×16.00)	37.75$_{\pm 0.02}$	49.49$_{\pm 0.03}$	90.30$_{\pm 0.00}$	87.80$_{\pm 0.35}$	87.86$_{\pm 0.02}$	70.64$_{\pm 0.07}$	45.57$_{\pm 0.10}$	57.81$_{\pm 0.39}$	90.07$_{\pm 0.12}$	92.22$_{\pm 0.02}$	91.51$_{\pm 0.01}$	75.44$_{\pm 0.08}$
I2CL	0.26 (×32.02)	42.15$_{\pm 5.87}$	47.41$_{\pm 1.99}$	92.59$_{\pm 0.86}$	89.25$_{\pm 0.32}$	93.40$_{\pm 1.73}$	72.96$_{\pm 1.01}$	53.22$_{\pm 2.04}$	51.55$_{\pm 3.02}$	90.77$_{\pm 1.62}$	93.75$_{\pm 0.25}$	93.36$_{\pm 0.97}$	76.53$_{\pm 0.26}$
MTV	0.13 (×16.00)	50.85$_{\pm 4.94}$	50.30$_{\pm 0.24}$	93.73$_{\pm 0.26}$	89.46$_{\pm 0.17}$	94.15$_{\pm 0.72}$	75.70$_{\pm 1.06}$	52.53$_{\pm 6.36}$	57.68$_{\pm 1.17}$	94.27$_{\pm 0.21}$	93.73$_{\pm 0.30}$	94.66$_{\pm 0.15}$	78.57$_{\pm 1.35}$
STV	**0.008** (×**1.00**)	61.39$_{\pm 0.20}$	49.44$_{\pm 0.00}$	91.57$_{\pm 1.53}$	87.83$_{\pm 0.41}$	93.38$_{\pm 1.01}$	76.72$_{\pm 0.35}$	58.15$_{\pm 0.90}$	58.17$_{\pm 0.75}$	76.32$_{\pm 8.90}$	88.64$_{\pm 5.47}$	91.10$_{\pm 0.00}$	74.48$_{\pm 3.13}$
LIVE	0.13 (×16.00)	59.74$_{\pm 0.45}$	54.12$_{\pm 0.17}$	93.69$_{\pm 0.11}$	87.80$_{\pm 0.98}$	93.36$_{\pm 0.49}$	77.74$_{\pm 0.25}$	62.40$_{\pm 0.44}$	53.85$_{\pm 0.27}$	78.91$_{\pm 0.10}$	86.33$_{\pm 0.22}$	86.87$_{\pm 0.41}$	73.67$_{\pm 0.14}$
MimIC	0.26 (×32.13)	68.46$_{\pm 1.72}$	59.30$_{\pm 0.70}$	94.61$_{\pm 0.60}$	90.45$_{\pm 3.54}$	93.30$_{\pm 1.22}$	81.22$_{\pm 0.35}$	67.83$_{\pm 0.89}$	55.21$_{\pm 0.95}$	95.16$_{\pm 0.52}$	95.79$_{\pm 1.00}$	96.35$_{\pm 0.47}$	82.07$_{\pm 0.54}$
HiFICL	2.2 (×272.00)	65.56$_{\pm 2.05}$	59.24$_{\pm 1.19}$	92.29$_{\pm 0.75}$	91.93$_{\pm 0.73}$	**95.18**$_{\pm 0.39}$	80.84$_{\pm 0.74}$	67.10$_{\pm 1.44}$	55.78$_{\pm 1.27}$	**95.47**$_{\pm 0.27}$	**97.29**$_{\pm 0.56}$	**97.05**$_{\pm 0.34}$	82.54$_{\pm 0.25}$
STAVE	**0.008** (×**1.00**)	**69.63**$_{\pm 0.37}$	**59.59**$_{\pm 0.50}$	**95.36**$_{\pm 0.47}$	**95.67**$_{\pm 0.42}$	93.50$_{\pm 0.72}$	**82.75**$_{\pm 0.19}$	**68.60**$_{\pm 0.55}$	**62.39**$_{\pm 0.12}$	95.20$_{\pm 0.32}$	96.30$_{\pm 0.23}$	95.56$_{\pm 0.26}$	**83.61**$_{\pm 0.14}$

(PT-App), or before the final token (PT-BL). Task vectors include TV (Hendel et al., 2023), FV (Todd et al., 2024), ICV (Liu et al., 2024b), I2CL (Li et al., 2025b), MTV, STV, and SITE (Park et al., 2026), whose values come from demo activations, and the last four learn which heads, values, or scales to use. Trained per-layer interventions include LIVE (Peng et al., 2024), MimIC, and HiFICL.

4.2 Implementation Details

STAVE learns only the two rows of W, which start at zero in single precision, while the backbone stays frozen in half precision. Each source prompt holds one to eight demos. We optimize the vectors with AdamW (Loshchilov & Hutter, 2017) under a cosine learning-rate schedule. Generally, each training step uses $K = 4$ paired examples and a learning rate of 10^{-3} on LMM tasks, and $K = 8$ and 10^{-1} on LLM tasks. We run all compared methods from their released code with the evaluation data, prompts, metrics, and decoding of each protocol. All main experiments are run three times, and we report the mean and standard deviation of the three runs. Additional details, including backbones, protocol settings, hyperparameter sensitivity, and prompt examples, are reported in Appendix F.

4.3 Main Results

Comparison with trained methods on LMMs. Tables 1 and 2 test whether an update at the input embeddings alone can match methods that learn a shift in every decoder layer. STAVE has the best

Table 3: Comparison on the TV benchmark (18 text tasks).

		Vanilla ICL					PT						
Model	Zero-shot	n=1	n=5	n=15	LoRA	Prefix	Pre	App	BL	FV	TV	SITE	**STAVE**
Pythia 2.8B	$13.1_{\pm 0.0}$	$60.4_{\pm 0.6}$	$85.6_{\pm 0.4}$	$\underline{90.6}_{\pm 0.2}$	$79.1_{\pm 5.1}$	$61.8_{\pm 1.6}$	$70.4_{\pm 5.7}$	$79.4_{\pm 6.7}$	$78.4_{\pm 5.2}$	$40.7_{\pm 2.1}$	$75.6_{\pm 0.1}$	$89.9_{\pm 0.2}$	$\mathbf{91.2}_{\pm 0.3}$
Pythia 6.9B	$10.9_{\pm 0.0}$	$65.0_{\pm 0.9}$	$86.6_{\pm 0.5}$	$\underline{90.6}_{\pm 0.1}$	$78.1_{\pm 2.4}$	$65.7_{\pm 2.6}$	$70.5_{\pm 6.2}$	$83.5_{\pm 1.8}$	$76.1_{\pm 8.2}$	$42.4_{\pm 3.0}$	$75.5_{\pm 0.2}$	$90.2_{\pm 0.1}$	$\mathbf{91.4}_{\pm 0.2}$
Pythia 12B	$11.4_{\pm 0.0}$	$64.5_{\pm 0.3}$	$87.0_{\pm 0.4}$	$91.0_{\pm 0.1}$	$81.5_{\pm 4.6}$	$55.2_{\pm 3.8}$	$71.8_{\pm 4.1}$	$82.9_{\pm 1.0}$	$73.9_{\pm 3.9}$	$42.2_{\pm 0.8}$	$75.0_{\pm 0.4}$	$\underline{91.1}_{\pm 0.4}$	$\mathbf{91.2}_{\pm 0.5}$
LLaMA 7B	$17.1_{\pm 0.0}$	$67.8_{\pm 0.5}$	$88.4_{\pm 0.2}$	$90.6_{\pm 0.3}$	$76.1_{\pm 2.7}$	$70.2_{\pm 6.8}$	$59.9_{\pm 3.8}$	$87.5_{\pm 2.5}$	$82.5_{\pm 4.7}$	$59.4_{\pm 0.5}$	$80.4_{\pm 0.9}$	$\mathbf{92.0}_{\pm 0.2}$	$\underline{91.9}_{\pm 0.4}$
GPT-J 6B	$10.2_{\pm 0.0}$	$63.6_{\pm 0.4}$	$86.3_{\pm 0.3}$	$91.0_{\pm 0.2}$	$73.7_{\pm 4.0}$	$66.5_{\pm 6.1}$	$77.8_{\pm 3.6}$	$86.0_{\pm 0.8}$	$84.9_{\pm 3.2}$	$56.3_{\pm 1.7}$	$72.4_{\pm 0.6}$	$\underline{91.8}_{\pm 0.2}$	$\mathbf{91.9}_{\pm 0.4}$
Avg.	$12.5_{\pm 0.0}$	$64.3_{\pm 0.3}$	$86.8_{\pm 0.3}$	$90.8_{\pm 0.1}$	$77.7_{\pm 0.4}$	$63.9_{\pm 0.3}$	$70.1_{\pm 3.4}$	$83.9_{\pm 1.3}$	$79.1_{\pm 2.6}$	$48.2_{\pm 0.9}$	$75.8_{\pm 0.3}$	$\underline{91.0}_{\pm 0.1}$	$\mathbf{91.5}_{\pm 0.3}$

score on every column of Table 1 and the best average on both backbones of Table 2, and it leads on five of six columns with three more recent LMMs (Appendix G.1). It stores 14 to 2,080 times fewer parameters than LIVE, MimIC, and HiFICL and keeps zero-shot latency (Figure 1), since an input update already acts through every block (Proposition 2.1). PT also trains at the input, but it inserts virtual tokens and its best mode varies with the backbone and dataset. STAVE outperforms all three PT modes on average with 10 times fewer parameters and without lengthening the prompt. With over 2,000 times more parameters, LoRA still trails STAVE on every column of Table 1.

Comparison with ICL and task vectors. STAVE outperforms ICL with up to 32 demos on every column of Table 1 while using none, and ICL peaks at 8 or 16 demos on VQA. In Table 2, TV and ICV add demo activations without training and stay near zero-shot on Idefics2. I2CL learns their scales, and MTV and STV learn which heads receive them, yet all trail STAVE on average, even STV at the same size, so fitting the values to the answer matters more than choosing where to add them.

Comparison on LLMs. Table 3 tests whether the design transfers to LLMs. STAVE has the highest average on the TV benchmark and exceeds 15-shot ICL on every LLM without demos at inference. SITE, which edits head activations in every layer, comes closest, yet STAVE is slightly higher with an input update alone. TV and FV remain far below. LoRA, prefix tuning, and PT train more parameters yet stay below 5-shot ICL on every LLM, and in Tables 2 and 3 PT varies far more across runs than STAVE, whose zero-initialized vectors start at the exact zero-shot computation.

4.4 Ablation Studies

Structural token groups. Figure 4(a) changes only how groups are assigned to vectors. The readout/context split is best on COCO and tied for best on VQAv2. One vector is never enough, and one for all tokens is weakest on COCO, as the dilution in Equation 1 predicts. Two vectors also fall short if the answer cue joins the context vector (Last / rest) or they split by modality (Text / image). Both put readout tokens in one vector with context tokens of smaller gradient per token (Figure 2(b)), so the gain comes from separating answer-producing tokens from their context. Three or seven vectors add parameters without a gain, consistent with Theorem 3.1. Appendix H adds equal-norm random splits and count normalization variants.

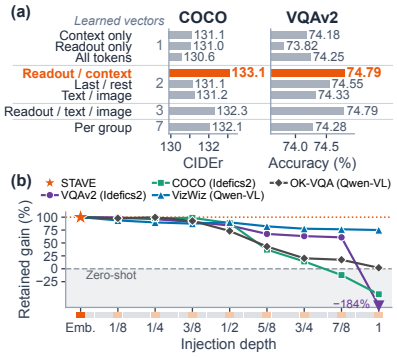


Figure 4: Ablations. (a) Groups split into 1, 2, 3, or 7 vectors, mean of Idefics2 and LLaVA. (b) Injection depth.

Injection depth. Figure 4(b) adds the same vectors after a decoder block instead of at the input (Appendix H.7). The embedding injection is best or tied for best on all four curves, and the retained gain shrinks with depth, falling below zero-shot after the last block on two of them. This matches Proposition 2.1: a later update reaches fewer blocks, and after the last block it can change only the first answer token, and only through the final token.

Paired supervision. Table 4 drops one branch of Equation 6. Paired training scores highest in every row, so demos in training also help the demo-free prompt, as Proposition 3.2 allows for aligned branch gradients. Source-only vectors, which never see a demo-free prompt, are lowest for every LMM and dataset. On the TV benchmark, a single branch holds only in its training format, whereas paired vectors hold in both. Unlike target-only vectors, paired vectors keep almost all of their demo-free COCO score when demos are added, and their captions hallucinate less (Appendices H.1 and H.2).

Table 4: Training-objective ablation.

Backbone / Dataset	Paired	Target-only	Source-only
Idefics2 VQAv2	**73.46%**	73.32%	72.41%
LLaVA VQAv2	**76.12%**	76.01%	74.98%
Idefics2 COCO	**134.38**	134.01	131.34
LLaVA COCO	**131.78**	129.80	129.11
TV bench	**92.07%**	91.53%	79.67%
w/ 1 demo	**92.23%**	88.90%	92.10%

Table 5: Hallucination on COCO.

Idefics2 COCO	CIDEr ↑	CHAIRs ↓	CHAIRi ↓	Recall ↑
Zero-shot	80.54	3.44	2.84	35.73
8-shot ICL	119.07	3.06	2.26	41.76
LoRA	132.37	2.46	1.68	43.64
LIVE	127.62	2.56	1.83	43.15
MimIC	132.38	3.48	2.28	45.62
HiFICL	128.62	3.30	2.10	**45.98**
STAVE	**133.27**	**2.24**	**1.53**	44.20

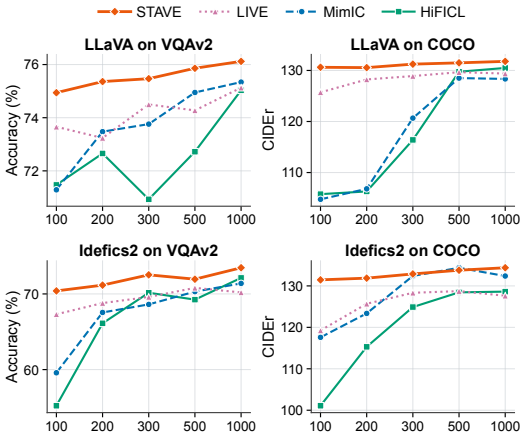


Figure 7: Data efficiency across training set sizes.

4.5 Analysis

Query-specific answers from one update. The same two vectors are added to every prompt, yet the answer changes with the query (Figure 5). Without the update, LLaVA predicts a template token for almost every VQAv2 question. STAVE moves the states of each answer type in a different direction, and within object questions it separates the categories they ask about, because the frozen blocks process the vectors together with each query. Table 1 shows examples.

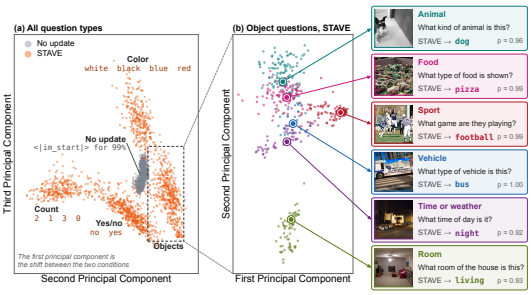


Figure 5: Final-token states of LLaVA on VQAv2 validation questions, without and with STAVE.

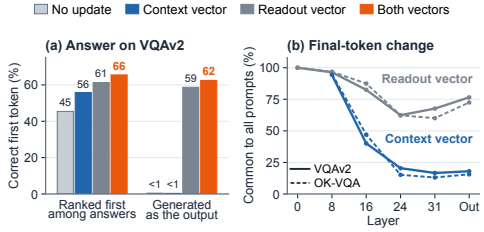


Figure 6: Effect of each learned vector.

What each vector contributes. Figure 6(a) adds the Idefics2 VQAv2 vectors one at a time to held-out prompts without retraining and reads the reference first token. Without the update, this token often already ranks first among the task's answers, yet the model almost never generates it. The context vector alone ranks it first more often but rarely changes the output, whereas the readout vector alone makes the model answer. Both vectors score highest. Figure 6(b) shows why the vectors play different roles. On VQAv2 and OK-VQA, most of the change that the readout vector makes to the final-token state is common to all prompts, whereas the change that the context vector adds becomes specific to each prompt from the middle layers on. The readout vector thus sets a shared answer format, and the context vector supplies query-specific evidence for choosing the answer. Appendix I adds more tests of each vector.

Why a small update is sufficient. STAVE's $2d$ parameters draw on what the frozen backbone already holds (Figure 6(a)) and leave it intact. On Qwen-VL OK-VQA, PT and all per-layer interventions in Table 2 fall below zero-shot, while STAVE exceeds it (Appendix F.3). The small update makes STAVE data efficient (Figure 7). With a tenth of the training samples it loses a few points, whereas MimIC and HiFICL lose far more on Idefics2, consistent with the risk bound of Section 3.3, whose estimation term depends on the covering dimension $2d$, not on the number of decoder layers.

Fine-grained classification. STAVE leads on Idefics2 DTD and Flowers, by 3.7 points on Flowers, and stays within 1.7 points of the best method on the other four columns, all led by HiFICL. The test classes of these two-choice tasks never appear as training queries, so correcting a zero-shot error means separating similar classes that no training query contains. HiFICL can add query-specific visual evidence in every layer with 272 times more parameters, whereas the two vectors of STAVE act only through the frozen blocks and draw on the classes the backbone already recognizes.

Object hallucination. Table 5 checks what drives the CIDEr gain on Idefics2 COCO. STAVE has the highest CIDEr and lowest CHAIRs and CHAIRi (Rohrbach et al., 2018), the shares of captions and of mentions that name absent objects. MimIC and HiFICL recall more objects, but their CHAIRs stays near the zero-shot level, while STAVE lowers it by a third, so its gain comes from more accurate mentions. In the COCO example of Table 1, STAVE names the chicken, whereas MimIC and HiFICL describe only a plate of food.

Table 6: Inference cost on Idefics2 VQAv2.

vs. ICL	Tokens	TTFT (ms)	Mem. (GiB)
Zero-shot	**128**	**98**	**16.5**
8-shot ICL	1,026	1,430	29.8
16-shot ICL	1,922	2,854	43.0
32-shot ICL	3,717	5,877	68.7
STAVE	**128**	**98**	**16.5**
vs. trained	**State (KiB)**	**TTFT (ms)**	**Decode (ms/tok)**
LoRA	68,704	116	33.6
LIVE	512	101	27.4
MimIC	1,028	106	31.6
HiFICL	8,704	108	34.3
STAVE	**32**	**98**	**23.9**

Inference cost. Table 6 measures the per-query cost of the Idefics2 VQAv2 checkpoints of Table 1 on one NVIDIA H200. STAVE adds its vectors before the first decoder block, so its decoder computation is that of zero-shot inference (Appendix E.1). ICL re-encodes its demos at every query, and with 32 demos its time to first token (TTFT) is 60 times that of STAVE, and its memory grows with the demo count. The trained methods keep the prompt length but add computation inside the decoder layers, which slows every decoded token. STAVE has the lowest cost in every column of both panels.

5 Related Work

ICL puts demos in every query's prompt (Brown et al., 2020), and in LMMs each demo image adds many visual tokens (Laurençon et al., 2024b). Extracted task vectors reuse demo activations as one vector at one layer (Hendel et al., 2023; Todd et al., 2024), at heads selected per task (Huang et al., 2024; Ma et al., 2026; Park et al., 2026), or in every layer with fixed or learned scales (Liu et al., 2024b; Li et al., 2025b), and STAVE beats all seven on average, even STV at equal size. Learned task vectors instead train one vector from the answer at a chosen layer and token (Yang et al., 2026). STAVE also learns from the answer, but on tokens set by prompt structure, so it needs no layer, head, or token search. Trained interventions add per-layer vectors (Peng et al., 2024), per-head shifts (Jiang et al., 2025), attention-logit biases (Talemi et al., 2026), or virtual keys and values (Li et al., 2026) to every decoder layer, and all four train only on demo-free prompts, three against an ICL teacher. STAVE stores $2d$ input scalars regardless of depth, adds no per-layer computation, and scores higher on average. Its teacher-free training with and without demos beats demo-free training (Table 4). PT and prefix tuning insert new tokens or attention memory (Lester et al., 2021; Li & Liang, 2021; Gupta et al., 2026), and within a layer added keys scale a token's original attention weights by one factor, so its relative attention over image and text stays fixed (Petrov et al., 2024). STAVE writes onto existing tokens, so it can change this relative attention, keeps the zero-shot prompt length, and outperforms 20-token PT with 10 times fewer parameters. Steering vectors and ReFT also edit existing hidden states, but at chosen layers and tokens (Turner et al., 2023; Wu et al., 2024). Adapters and LoRA modify decoder modules per task (Houlsby et al., 2019; Hu et al., 2021; Liu et al., 2022a), whereas STAVE swaps two input vectors, shares every module across tasks, and outperforms LoRA with orders of magnitude fewer parameters. Appendix A extends this discussion.

6 Conclusion

We presented STAVE, which replaces in-context demos with two task-specific vectors added to the input embeddings of a frozen LLM or LMM. This is the only depth that reaches every decoder block, and the vectors are added to existing tokens, one for the answer-producing tokens and one for the other five groups. Trained with and without demos, they match or outperform per-layer interventions on six LMMs with 14 to 2,080 times fewer task parameters and exceed 15-shot ICL on five LLMs, while keeping the prompt length, latency, and memory of zero-shot inference. The ablations show that the gain comes from separating the answer-producing tokens from their context, not from more vectors or a modality split, and that training with demos also helps the demo-free prompt. Our analysis suggests that the vectors mainly make the frozen model output answers it already represents, and their captions name fewer absent objects. Training is a one-time cost but backpropagates through the frozen decoder, so its memory grows with the backbone, and it requires access to the weights. Since each task adds only two vectors and no prompt tokens, one deployed backbone can serve many tasks. Extending structural token groups to multi-image and video prompts is a natural next step.

AI USE STATEMENT

An LLM was used for language polishing and grammar checking to improve the clarity and readability of the manuscript, and as a coding assistant for writing and debugging experiment code. The authors reviewed and tested all LLM-assisted code before using it. All research ideas, methodology, experimental design, analyses, results, and conclusions were developed by the authors. The authors take full responsibility for the final manuscript.

ETHICS STATEMENT

This work adapts publicly available pretrained LLMs and LMMs with public benchmarks, VQAv2, OK-VQA, COCO, VizWiz, DTD, Flowers, CUB, and the TV benchmark. We recruited no participants and collected no new data. STAVE keeps the backbone frozen, so it inherits the biases and failure modes of the pretrained models, including hallucinated content in generated captions.

REPRODUCIBILITY STATEMENT

Section 3 defines the structural token groups, the two learned vectors, and the training objective, and Section 4.2 gives the training settings. Appendix F lists the backbone checkpoints, data splits, random seeds, and hyperparameters of STAVE and the baselines, together with hyperparameter sensitivity and prompt examples. Appendices B, C, and D state the assumptions and give the proofs of the theoretical results, and Appendix E describes how inference cost is measured.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023. URL https://arxiv.org/abs/2308.12966.

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, 2025. URL https://arxiv.org/abs/2502.13923.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International conference on machine learning*, pp. 2397–2430. PMLR, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Wang Cai, Hsiu-Yuan Huang, Zhixiang Wang, and Yunfang Wu. Beyond demonstrations: Dynamic vector construction from latent representations. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5853–5868, 2025.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Xi Ding and Lei Wang. Do language models understand time? In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 1855–1868, 2025.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 1107–1128, 2024.

Yuxin Dong, Jiachen Jiang, Zhihui Zhu, and Xia Ning. Understanding task vectors in in-context learning: Emergence, functionality, and limitations. In *International Conference on Learning Representations*, volume 2026, pp. 19393–19428, 2026.

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6904–6913, 2017.

Akash Gupta, Amos Storkey, and Mirella Lapata. Meta-adaptive prompt distillation for few-shot visual question answering. In *International Conference on Learning Representations*, volume 2026, pp. 120800–120837, 2026.

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3608–3617, 2018.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, 2023.

Alberto Hojel, Yutong Bai, Trevor Darrell, Amir Globerson, and Amir Bar. Finding visual task vectors. In *European Conference on Computer Vision*, pp. 257–273. Springer, 2024.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Brandon Huang, Chancharik Mitra, Assaf Arbelle, Leonid Karlinsky, Trevor Darrell, and Roei Herzig. Multimodal task vectors enable many-shot multimodal in-context learning. *Advances in Neural Information Processing Systems*, 37:22124–22153, 2024.

Dongfu Jiang, Xuan He, Huaye Zeng, Cong Wei, Max Ku, Qian Liu, and Wenhu Chen. Mantis: Interleaved multi-image instruction tuning. *arXiv preprint arXiv:2405.01483*, 2024.

Yuchu Jiang, Jiale Fu, Chenduo Hao, Xinting Hu, Yingzhe Peng, Xin Geng, and Xu Yang. Mimic in-context learning for multimodal tasks. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 29825–29835, 2025.

Joonseong Kang, Soojeong Lee, Subeen Park, Sumin Park, Taero Kim, Jihee Kim, Ryunyi Lee, and Kyungwoo Song. Adaptive task vectors for large language models. *arXiv preprint arXiv:2506.03426*, 2025.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

Hugo Laurençon, Andrés Marafioti, Victor Sanh, and Léo Tronchon. Building and better understanding vision-language models: insights and future directions. *arXiv preprint arXiv:2408.12637*, 2024a.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What matters when building vision-language models? *Advances in Neural Information Processing Systems*, 37:87874–87907, 2024b.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Dongfang Li, Zhenyu Liu, Xinshuo Hu, Zetian Sun, Baotian Hu, and Min Zhang. In-context learning state vector with inner and momentum optimization. *Advances in Neural Information Processing Systems*, 37:7797–7820, 2024a.

Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *arXiv preprint arXiv:2407.07895*, 2024b.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Xiaoyu Li, Yuhang Liu, Xuanshuo Kang, Zheng Luo, Fangqi Lou, Xiaohua Wu, and Zihan Xiong. Hificl: High-fidelity in-context learning for multimodal tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3069–3078, 2026.

Yanshu Li, Yi Cao, Hongyang He, Qisen Cheng, Xiang Fu, Xi Xiao, Tianyang Wang, and Ruixiang Tang. M^2IV: Towards efficient and fine-grained multimodal in-context learning via representation engineering. *arXiv preprint arXiv:2504.04633*, 2025a.

Zhuowei Li, Zihao Xu, Ligong Han, Yunhe Gao, Song Wen, Di Liu, Hao Wang, and Dimitris Metaxas. Implicit in-context learning. In *International Conference on Learning Representations*, volume 2025, pp. 101644–101666, 2025b.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022a.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023a.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 26296–26306, 2024a.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35, 2023b.

Sheng Liu, Haotian Ye, Lei Xing, and James Y. Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32287–32307. PMLR, 21–27 Jul 2024b. URL `https://proceedings.mlr.press/v235/liu24bx.html`.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, 2022b.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *AI open*, 5:208–215, 2024c.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Grace Luo, Trevor Darrell, and Amir Bar. Vision-language models create cross-modal task representations. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41157–41176. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/luo25c.html.

Ziyu Ma, Chenhui Gou, Yiming Hu, Yong Wang, Bohan Zhuang, and Jianfei Cai. Where and what matters: Sensitivity-aware task vectors for many-shot multimodal in-context learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 7892–7900, 2026.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pp. 3195–3204, 2019.

Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352, 2023.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pp. 722–729. IEEE, 2008.

Samet Oymak, Ankit Singh Rawat, Mahdi Soltanolkotabi, and Christos Thrampoulidis. On the role of attention in prompt-tuning. In *International Conference on Machine Learning*, pp. 26724–26768. PMLR, 2023.

Jungwon Park, Jimyeong Kim, Changin Choi, and Wonjong Rhee. Soft head selection for injecting icl-derived task embeddings. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 27161–27214, 2026.

Yingzhe Peng, Chenduo Hao, Xinting Hu, Jiawei Peng, Xin Geng, and Xu Yang. Live: Learnable in-context vector for visual question answering. *Advances in Neural Information Processing Systems*, 37:9773–9800, 2024.

Aleksandar Petrov, Philip Torr, and Adel Bibi. When do prompting and prefix-tuning work? a theory of capabilities and limitations. In *International Conference on Learning Representations*, volume 2024, pp. 6031–6054, 2024.

Guanghui Qin and Jason Eisner. Learning how to ask: Querying lms with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5203–5212, 2021.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, 2024.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. Object hallucination in image captioning. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 4035–4045, 2018.

Baturay Saglam, Xinyang Hu, Zhuoran Yang, Dionysis Kalogerias, and Amin Karbasi. Learning task representations from in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6634–6663, 2025.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations*, volume 2024, pp. 25055–25083, 2024.

Nishant Subramani, Nivedita Suresh, and Matthew E Peters. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 566–581, 2022.

Niloufar Alipour Talemi, Hossein Kashiani, and Fatemeh Afghah. Hyper-icl: Attention calibration with hyperbolic anchor distillation for multimodal in-context learning. *arXiv preprint arXiv:2606.04434*, 2026.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron Wallace, and David Bau. Function vectors in large language models. In *International conference on learning representations*, volume 2024, pp. 17282–17333, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4566–4575, 2015.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. Understanding reasoning in thinking language models via steering vectors. *arXiv preprint arXiv:2506.18167*, 2025.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Ben Wang and Aran Komatsuzaki. Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency, 2025. URL https://arxiv.org/abs/2508.18265.

Yihan Wang, Jatin Chauhan, Wei Wang, and Cho-Jui Hsieh. Universality and limitations of prompt tuning. *Advances in Neural Information Processing Systems*, 36:75623–75643, 2023.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962, 2024.

Haolin Yang, Hakaze Cho, Kaize Ding, and Naoya Inoue. Task vectors, learned not extracted: Performance gains and mechanistic insights. In *International Conference on Learning Representations*, volume 2026, pp. 74846–74903, 2026.

Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023.

Yiyuan Zhang, Handong Li, Jing Liu, and Xiangyu Yue. Learning beyond still frames: Scaling vision-language models with video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22425–22435, 2025.

Haozhe Zhao, Zefan Cai, Shuzheng Si, Xiaojian Ma, Kaikai An, Liang Chen, Zixuan Liu, Sheng Wang, Wenjuan Han, and Baobao Chang. Mmicl: Empowering vision-language model with multi-modal in-context learning. In *International Conference on Learning Representations*, volume 2024, pp. 14942–14980, 2024.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pp. 12697–12706. Pmlr, 2021.

Yucheng Zhou, Xiang Li, Qianning Wang, and Jianbing Shen. Visual in-context learning for large vision-language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 15890–15902, 2024.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

APPENDIX CONTENTS

A Extended related work **17**

B Injection depth and inserted tokens **19**
B.1 Proof of Proposition 2.1 19
B.2 Attention ratios at a fixed layer 19

C Structural token groups and vector sharing **20**
C.1 Partition conventions and modality handling 20
C.2 Normalized geometry and gradients 20
C.3 First-order cost of sharing 21
C.4 Sharing cost before training 22
C.5 Proof of Theorem 3.1 22

D Paired supervision and demo-free risk **23**
D.1 Proof of Proposition 3.2 23
D.2 Compatibility of the empirical objectives 24
D.3 Demo-free risk guarantees 24

E Inference cost **25**
E.1 Measured inference cost 26
E.2 Stored task state 27
E.3 Performance overview settings 29

F Implementation details **29**
F.1 Backbones 29
F.2 VQAv2, OK-VQA, and COCO 30
F.3 VizWiz, OK-VQA, and fine-grained classification 31
F.4 TV benchmark 31
F.5 Hyperparameter sensitivity 32
F.6 Dataset and prompt examples 33

G Additional results **35**
G.1 More recent LMMs 35
G.2 VQAv2 answer types 35

H Ablation details **36**
H.1 Demos in the prompt on COCO Captioning 36
H.2 Object hallucination by training objective 36
H.3 Training objective on the TV benchmark 36
H.4 Structural token groups 37
H.5 Allocation of a shared update 38
H.6 Count normalization 40
H.7 Injection depth 41

I Analysis of the learned vectors **41**
I.1 Promoted tokens 41
I.2 States and decoded tokens 42
I.3 What each learned vector changes 45
I.4 Answer-prefix probabilities 46
I.5 Qualitative examples 46

J Limitations and future work **50**

K Impact statement **50**

A Extended related work

Multimodal ICL and its cost. Visual instruction tuning connects image understanding with language generation (Liu et al., 2023a; 2024a), and interleaved image–text training together with multi-image instruction tuning supports richer multimodal contexts (Laurençon et al., 2024b; Jiang et al., 2024; Zhao et al., 2024). These models inherit demo-based adaptation from language models (Brown et al., 2020), but each demo image adds visual processing and many visual tokens to every query. Visual In-Context Learning retrieves demos, summarizes their images, and composes textual demos (Zhou et al., 2024), yet the demos remain in every prompt. Demo-free methods remove them and store the task in a compact state. They differ in where this state enters the frozen model: as extracted or trained decoder states, as added input tokens or attention memory, or, in STAVE, as updates to existing input embeddings. STAVE keeps only the query image and text at inference, so it never encodes a demo image for a new query and runs with the prompt length and cache of zero-shot inference.

Extracted visual and multimodal task vectors. Visual task vectors in MAE-VQGAN show that averaged attention-head activations can specify visual tasks without input–output examples (Hojel et al., 2024), and cross-modal activation patching shows that LMMs encode task information shared across text and image inputs (Luo et al., 2025). MTV averages demo-conditioned head activations and uses reinforcement learning to select the heads that receive them (Huang et al., 2024). STV selects context-sensitive heads from activation differences and then uses reinforcement learning to pick values from clustered activation banks (Ma et al., 2026). In both, the values are restricted to demo activations, and every task needs a search over head locations. STAVE learns its values directly with answer supervision on tokens fixed by prompt structure, which removes the head search and the activation bank and lets the readout and context vectors be optimized jointly for the same answer. Because the update enters before the first decoder layer, one pair of vectors reaches every layer's attention, whereas a head-level vector acts only where it is inserted. STAVE outperforms MTV and STV in every comparison, including STV at an equal parameter count.

Distillation through decoder interventions. LIVE learns one vector and one scale per decoder layer by matching the output distribution of a demo-conditioned model, with an added answer loss (Peng et al., 2024). M^{2}IV assigns separate vectors to the attention and feed-forward branches of every layer and combines output distillation, answer supervision, and a branch-correlation objective (Li et al., 2025a). MimIC learns per-head shifts after attention with query-dependent scaling and aligns hidden states with the demo-conditioned model layer by layer (Jiang et al., 2025). Hyper-ICL adds low-rank, query-modulated biases to attention logits and aligns intermediate features with a demo-conditioned teacher in hyperbolic space (Talemi et al., 2026). These parameters sit in every decoder layer, so their storage grows with depth, their objectives match teacher outputs or intermediate features, and the student sees only demo-free prompts. STAVE stores $2d$ scalars regardless of depth, 14 to 2,080 times fewer than LIVE and MimIC, adds no per-layer computation at inference, and scores higher on average. Its two vectors are trained with answer CE on prompts with and without demos, with no teacher target, and this paired objective improves demo-free accuracy over training on demo-free prompts alone.

Teacher-free virtual attention context. HiFICL places low-rank virtual KV pairs in every attention head, whose interaction with each query mixes learned context values into the original attention output, and trains them with the task loss alone, as STAVE does (Li et al., 2026). STAVE instead changes the embeddings of existing readout and context tokens before the decoder. It stores $2d$ scalars, over 200 times fewer than HiFICL, adds no attention-memory entries, and scores higher on average on every LMM we evaluate. Its readout vector also changes the attention query of the token that predicts the answer, so it can re-weight attention over the query's own image and text tokens, whereas virtual keys and values can only mix a learned context into the original attention output.

Extracted and adaptive task representations in language models. TV takes the prompted residual state at one layer (Hendel et al., 2023), FV sums the outputs of task-relevant attention heads (Todd et al., 2024), and in-context vectors derive steering directions from latent input–output differences (Liu et al., 2024b). Later methods optimize injection coefficients or weighted combinations of head outputs (Li et al., 2025b; Saglam et al., 2025), generate a vector for each query (Kang et al., 2025), or refine it with inner and momentum optimization (Li et al., 2024a). SITE learns soft head-selection weights over ICL-derived embeddings (Park et al., 2026), and dynamic vector construction segments extracted representations and learns their injection locations (Cai et al., 2025). Learned task vectors

replace extraction with answer-supervised training of a vector at a chosen layer and token (Yang et al., 2026), and analyses of extracted task vectors identify limits on representing high-rank task mappings (Dong et al., 2026). STAVE also learns its vectors from the answer, but places them at the input over tokens defined by prompt structure, so its tokens need no per-task search and one structural partition transfers across tasks and backbones. Its normalization keeps the update energy from growing with the number of tokens a vector updates, which matters in LMMs, where one image expands into hundreds of tokens. On the TV benchmark, STAVE exceeds TV and FV on every model and reaches the highest average accuracy, above SITE and 15-shot ICL, without any demo at inference. The vector-sharing tradeoffs of this parameterization are analyzed in Appendix C.5.

Task adaptation at added input tokens. PT prepends learned vectors to the input embeddings of a frozen model and approaches full fine-tuning as model scale grows (Lester et al., 2021). P-tuning generates continuous prompts at template-defined locations with a small prompt encoder (Liu et al., 2024c), deep prompt tuning and prefix tuning supply prompt vectors or KV pairs at every decoder layer (Liu et al., 2022b; Li & Liang, 2021), and gist tokens compress an instruction into the activations of a few added tokens (Mu et al., 2023). In LMMs, MAPD meta-learns soft prompts together with an attention mapper over visual features and adapts them from a few labeled examples at test time (Gupta et al., 2026). Surveys of prompting collect further variants (Liu et al., 2023b). All of these methods add tokens or attention-memory entries whose placement is a design choice, and language models are sensitive to prompt formatting (Sclar et al., 2024). Theory shows that added prefix tokens only bias the attention output of the original tokens without changing their relative attention (Petrov et al., 2024), characterizes the expressive limits of bounded-length prompts (Wang et al., 2023), and traces how a tuned prompt acts through a single attention layer (Oymak et al., 2023). STAVE also keeps the decoder frozen and stores the task at the input, but differs in three respects. It writes onto existing tokens selected by a structural partition shared across tasks and backbones, so no insertion point is chosen and the prompt keeps its zero-shot length and cache. Its readout vector changes the attention query of the token that predicts the answer, so it can change the relative attention over the original image and text tokens, which added keys cannot do at a fixed layer. A zero update recovers the original computation, so training starts from the frozen model's own predictions, whereas an added token with a zero embedding still enters the attention normalizer, and LLaMA-Adapter obtains this property only through a separately learned zero-initialized gate (Zhang et al., 2023). A two-token soft prompt stores as many parameters as STAVE, so these differences come from where the parameters enter rather than from capacity. STAVE outperforms 20-token PT on LMMs with 10 times fewer parameters and outperforms PT and prefix tuning on the TV benchmark. Appendix B.2 analyzes the attention-level consequence.

Updates to existing representations. Steering vectors add a fixed direction to the residual stream at chosen layers (Subramani et al., 2022; Turner et al., 2023; Rimsky et al., 2024; Zou et al., 2023), inference-time intervention shifts selected attention-head outputs (Li et al., 2023), and ReFT learns low-rank interventions on the hidden states of a fixed set of prefix and suffix tokens in selected layers (Wu et al., 2024). Mixtures of soft prompts initialize each prompt vector from a template word and tune it in place (Qin & Eisner, 2021). These methods choose a layer, a direction, or a set of tokens per task, and the steering approaches take their directions from activation contrasts rather than from the task answer. STAVE also updates existing tokens but acts at the input embeddings, so one update reaches every layer's attention and no layer is chosen. Moving the same update to the output of a decoder layer does not improve it in our depth ablation. Its tokens follow prompt structure, including the expanded image embeddings and the final prompt token, and its two shared vectors are scaled by the number of tokens each updates and learned from the answer on paired prompts.

Weight and module adaptation. Adapters and LoRA learn task-specific modules or low-rank weight updates (Houlsby et al., 2019; Hu et al., 2021), and few-shot parameter-efficient fine-tuning has been proposed as an alternative to supplying demos repeatedly (Liu et al., 2022a). These methods also keep the pretrained weights frozen, but their task parameters are spread across decoder layers and grow with depth and rank. STAVE stores two embedding vectors, shares every decoder module across tasks, switches tasks by replacing two vectors, and outperforms LoRA with orders of magnitude fewer parameters.

B INJECTION DEPTH AND INSERTED TOKENS

The derivative statements concern the real-valued decoder map at fixed tokenization, attention masks, positional indices, and correct answer prefixes. Modality features are frozen, and any stochastic computation is fixed or disabled. These statements do not differentiate through token selection or numerical quantization.

B.1 PROOF OF PROPOSITION 2.1

Fix a nonempty prompt with T attended tokens. Let $H_\ell \in \mathbb{R}^{T\times d}$ contain all its states after block ℓ, with H_0 the input embeddings. A prompt-only intervention at depth ℓ changes these states before later blocks are evaluated. It does not alter answer-token states at that depth or retroactively modify keys and values already formed in blocks $1,\dots,\ell$. After the last block, the output computation consists only of positionwise normalization, if present, and a positionwise vocabulary head.

Proof. For part (i), the prompt keys and values used by block k are computed from H_{k-1}. An addition to H_ℓ therefore leaves all prompt keys and values in blocks $k \le \ell$ unchanged and can affect later blocks only. The first answer token is predicted from the final prompt token. Every subsequent teacher-forced answer token is predicted from an answer-prefix token. At $\ell = N_{\mathrm{layer}}$, no later block can transmit a prompt-state change to these answer tokens. The positionwise head can therefore depend on the intervention only through the final prompt state, and only for the first prediction. Its derivative there may also vanish. Since the context vector excludes the final token, it has zero sequence-loss gradient at this depth.

For part (ii), take depths $0 \le \ell' < \ell \le N_{\mathrm{layer}}$. Causality makes the Jacobian of each intervening block, restricted to all prompt states, block lower triangular in token order. Its diagonal $d \times d$ blocks are the own-token derivatives. By assumption, these blocks are invertible. Their product, the Jacobian $\mathsf{T}_{\ell,\ell'}$ from the row-stacked entries of $H_{\ell'}$ to those of H_ℓ, is thus invertible. For any desired infinitesimal change $\Delta \in \mathbb{R}^{Td}$ at depth ℓ, the unique prompt-state perturbation producing it to first order is $\mathsf{T}_{\ell,\ell'}^{-1}\Delta$. The first-answer logits factor through H_ℓ and the remaining prompt computation, so the two interventions have the same first-order effect on those logits. □

Several depths. For simultaneous prompt-state shifts Δ_ℓ at multiple depths, all evaluated about the same unmodified forward pass, let $\mathsf{T}_\ell = \mathsf{T}_{\ell,0}$ and $\mathsf{T}_0 = I$. The input perturbation $\sum_\ell \mathsf{T}_\ell^{-1}\Delta_\ell$ reproduces their combined first-order first-answer effect by the chain rule and linearity, given nonsingularity along every involved path. Part (ii) concerns unrestricted perturbations over all attended prompt tokens and the first answer token. It characterizes what the input interface can reach, while the two-vector parameterization is analyzed in Appendix C.

B.2 ATTENTION RATIOS AT A FIXED LAYER

Fix one attention head and an attention query $\xi \in \mathbb{R}^{d_h}$, where d_h is the head dimension. For visible token i, let k_i and u_i be its key and value, and let its finite attention score be $s_i = \xi^\top k_i/\sqrt{d_h} + c_i$, with fixed positional term c_i. Partition the visible tokens into nonempty original and added sets Q and D. Writing $Z_Q = \sum_{i\in Q} e^{s_i}$ and $Z_D = \sum_{i\in D} e^{s_i}$ gives

$$o = (1-\lambda)o^Q + \lambda o^D, \qquad \lambda = \frac{Z_D}{Z_Q + Z_D}, \qquad o^Q = \frac{\sum_{i\in Q} e^{s_i}u_i}{Z_Q}, \quad o^D = \frac{\sum_{i\in D} e^{s_i}u_i}{Z_D}. \tag{B.1}$$

This is the single-head decomposition used to motivate shift-based adaptation (Jiang et al., 2025). All quantities here belong to the same forward computation. In a multilayer decoder, o^Q need not equal the output on a separately rendered demo-free prompt, because the original states and positional scores can also change.

Added keys and changed queries. If the original scores remain fixed when keys are added, every original weight $\alpha_i = e^{s_i}/Z_Q$ becomes $\alpha_i' = Z_Q\alpha_i/(Z_Q + Z_D)$. Hence $\alpha_i'/\alpha_j' = \alpha_i/\alpha_j$ for $i, j \in Q$ (Petrov et al., 2024). In contrast, holding keys and positional terms fixed while changing the query to $\xi + \epsilon b$ gives

$$\frac{d}{d\epsilon}\log\frac{\alpha_i(\epsilon)}{\alpha_j(\epsilon)}\bigg|_{\epsilon=0} = \frac{(k_i - k_j)^\top b}{\sqrt{d_h}}, \tag{B.2}$$

where $b \in \mathbb{R}^{d_h}$ is the query-change direction. This follows by subtracting the two scores, since their common normalizer cancels. An update at an existing token can change queries, keys, and values without inserting tokens, and Equation B.2 isolates the query response that added keys lack at a fixed layer.

C STRUCTURAL TOKEN GROUPS AND VECTOR SHARING

Write $S \in \{0,1\}^{R\times q}$ for the assignment matrix, with $S_{kr} = \mathbf{1}\{a(k) = r\}$ and exactly one nonzero entry per row. The transpose of learned row r for task τ is denoted by $v_{\tau,r} \in \mathbb{R}^d$, and the task subscript is suppressed on W. The independent-group configuration learns R vectors, while STAVE ties the groups into the readout and context vectors as in Equation 2.

C.1 PARTITION CONVENTIONS AND MODALITY HANDLING

The partition uses the serialized prompt and template metadata, not the answer label, model predictions, or learned attention scores. Counts refer to decoder-input tokens after image expansion. Demo spans include their inputs, answers, and demo-specific formatting. They are excluded before any span is assigned, with the FIRST token as the only exception. BOS tokens, padding, supervised answer tokens, and generated answer tokens are also excluded.

Endpoints and precedence. LAST is the final prompt token, the last attended token before the answer. FIRST is the first attended non-BOS token of the rendered prompt. When the template has an instruction prefix, as in VizWiz and VQAv2, this token lies in the shared prefix in both renderings and no demo token is updated. When the template has no prefix, it opens the query in the demo-free rendering and the first demo in the source rendering, and that single token receives the context vector while every other demo token is unchanged. On prompts that begin with an image, it is the opening delimiter of that image. The caption experiments instead use the first query-text token in both renderings. Tokens are assigned in the order LAST, FIRST, ANSWER-CUE, IMAGE-GATE, QUERY-IMAGE, QUESTION-TEXT, and PREFIX-TEXT, and the first matching label is kept. The same precedence is used for both renderings.

Spans. ANSWER-CUE consists of the template tokens that introduce the answer, such as "Answer:", except the final prompt token, which forms LAST. IMAGE-GATE holds explicit delimiters around the query image, not its expanded visual embeddings, which form QUERY-IMAGE. QUESTION-TEXT is the remaining text of the query, such as the question in VQA. PREFIX-TEXT is the template's instruction span outside demos. Tokens not identified by the fixed metadata remain unchanged. A group with no eligible tokens has count zero and contributes no update.

Modality handling. Text prompts use the frozen input-embedding layer. In LMMs, the frozen visual encoder and connector supply image embeddings at the hidden dimension of the decoder. STAVE operates on the merged decoder-input sequence, not on image pixels, visual encoder activations, or unexpanded placeholder tokens. Image groups are absent in text-only prompts, and IMAGE-GATE can also be absent in a multimodal template. Both paired renderings retain the query and its modality. These conventions specify an unambiguous operator.

C.2 NORMALIZED GEOMETRY AND GRADIENTS

For prompt p, let $\mathcal{I}(p) = \{k : n_k(p) > 0\}$ and $\mathcal{J}(p) = \{r : N_r(p) > 0\}$ denote the active groups and rows. Define $G_{tk}(p) = \mathbf{1}\{s_t(p) = k\}/\sqrt{n_k(p)}$ for $k \in \mathcal{I}(p)$, with zero columns otherwise. Let $\Lambda(p)$ be diagonal, with

$$\Lambda_{kk}(p) = \begin{cases} \sqrt{n_k(p)/N_{a(k)}(p)}, & n_k(p) > 0, \\ 0, & n_k(p) = 0. \end{cases} \tag{C.1}$$

An active group always has $N_{a(k)}(p) > 0$, so every division is well-defined.

Proposition C.1 (Normalized geometry under vector sharing). The injection operator satisfies

$$H(p) = G(p)\Lambda(p)S, \qquad G(p)^\top G(p) = \operatorname{diag}\big(\mathbf{1}\{k \in \mathcal{I}(p)\}\big),$$
$$H(p)^\top H(p) = \operatorname{diag}\big(\mathbf{1}\{r \in \mathcal{J}(p)\}\big), \qquad \|H(p)W\|_F^2 = \sum_{r \in \mathcal{J}(p)} \|v_{\tau,r}\|_2^2 \le \|W\|_F^2. \tag{C.2}$$

Moreover, $H(p)H(p)^\top$ is the orthogonal projector onto corrections constant on each active row's token set and zero elsewhere.

Proof. For an active group k assigned to row r, the corresponding product of scales is $n_k^{-1/2}\sqrt{n_k/N_r} = N_r^{-1/2}$. Excluded tokens and inactive groups contribute zero. This proves $H = G\Lambda S$. Active columns of G have disjoint supports and squared norm $n_k/n_k = 1$, and the same argument gives squared norm $N_r/N_r = 1$ for active columns of H. Taking the trace of $W^\top H^\top HW$ proves the norm identity. The orthonormal active columns of H span exactly the stated correction space, so their outer products sum to its orthogonal projector. □

Row gradients. For any differentiable loss F on $\widetilde{E} = E + HW$, the chain rule gives

$$\nabla_W F = H^\top \nabla_{\widetilde{E}} F, \qquad \nabla_{v_{\tau,r}} F = \frac{1}{\sqrt{N_r}} \sum_{t:a(s_t)=r} \nabla_{\widetilde{E}_t} F \quad (N_r > 0). \tag{C.3}$$

The row gradient is zero when $N_r = 0$, and $\|\nabla_W F\|_F \le \|\nabla_{\widetilde{E}} F\|_F$. Normalization controls replication-induced energy, not gradient invariance: if all token gradients in a row equal g, its gradient is $\sqrt{N_r}g$.

C.3 First-order cost of sharing

Fix one prompt and a differentiable loss F, evaluated at zero update. Let $g_t = \nabla_{\widetilde{E}_t} F \in \mathbb{R}^d$, and fix $\rho > 0$. For a nonempty eligible set $\mathcal{S}$, define $\Gamma(\mathcal{S})$ as in Equation 1.

Proposition C.2 (First-order cost of sharing). (i) The largest first-order decrease from an update $w/\sqrt{|\mathcal{S}|}$ at every token in $\mathcal{S}$, with $\|w\|_2 \le \rho$, is $\Gamma(\mathcal{S})$. The right side of Equation 1 is the optimum for free per-token updates of total Frobenius norm at most ρ. Equality holds exactly when all g_t in $\mathcal{S}$ are equal.

(ii) For disjoint nonempty $\mathcal{A}, \mathcal{B}$, two separately normalized vectors with combined squared norm at most ρ^2 attain

$$\Gamma_{\text{split}} = \sqrt{\Gamma(\mathcal{A})^2 + \Gamma(\mathcal{B})^2} \ge \max\{\Gamma(\mathcal{A}\cup\mathcal{B}), \Gamma(\mathcal{A}), \Gamma(\mathcal{B})\}.$$

If $g_t = 0$ on $\mathcal{B}$, then $\Gamma(\mathcal{A}\cup\mathcal{B}) = \sqrt{|\mathcal{A}|/|\mathcal{A}\cup\mathcal{B}|}\,\Gamma(\mathcal{A})$.

Proof. The shared update has directional derivative $\langle w, \sum_{t\in\mathcal{S}} g_t\rangle/\sqrt{|\mathcal{S}|}$. Minimizing this linear functional on the radius-ρ ball yields $-\Gamma(\mathcal{S})$. Applying the same argument to the stacked free token updates gives the right side of Equation 1. With $\overline{g} = |\mathcal{S}|^{-1}\sum_{t\in\mathcal{S}} g_t$, the identity

$$\sum_{t\in\mathcal{S}} \|g_t\|_2^2 - \frac{1}{|\mathcal{S}|}\left\|\sum_{t\in\mathcal{S}} g_t\right\|_2^2 = \sum_{t\in\mathcal{S}} \|g_t - \overline{g}\|_2^2$$

proves the inequality and its equality condition.

For two vectors, the stacked gradient has norm $\sqrt{\Gamma(\mathcal{A})^2 + \Gamma(\mathcal{B})^2}/\rho$, giving the stated optimum. A one-vector update w on $\mathcal{S} = \mathcal{A}\cup\mathcal{B}$ is realized by $w_\mathcal{A} = \sqrt{|\mathcal{A}|/|\mathcal{S}|}\,w$ and $w_\mathcal{B} = \sqrt{|\mathcal{B}|/|\mathcal{S}|}\,w$, with the same total norm. Thus splitting cannot reduce the optimum for this fixed prompt. Setting either vector to zero yields the other two lower bounds. The zero-gradient statement follows because $\mathcal{B}$ changes only the normalization factor, not the gradient sum. □

The dilution statement is exact under its zero-gradient hypothesis for a fixed prompt. Appendix H.5 measures it across image resolutions.

C.4 SHARING COST BEFORE TRAINING

Figure 2 measures the quantities of Proposition C.2 in the frozen Idefics2 backbone of Table 2 on Flowers. No vector is trained and no optimizer is involved. The measurement uses 512 demo-free samples from the training pool of that table, 256 to fit the update directions and 256 others to measure the loss. For each sample, g_t is the zero-update gradient of the answer CE of Equation 5 with respect to input embedding t. A prompt holds one token in each of LAST, ANSWER-CUE, FIRST, and IMAGE-GATE, 64 QUERY-IMAGE tokens, and 37.5 QUESTION-TEXT tokens on average, and no PREFIX-TEXT. The readout set $\mathcal{A}$ is LAST and ANSWER-CUE, and the context set $\mathcal{B}$ holds the other eligible tokens.

Update and gradient shares. One vector shared by all eligible tokens adds $w/\sqrt{|\mathcal{S}|}$ to each of them, so the share of $\|w\|_2^2$ that a group receives equals its share of the tokens. For the gradient, let $\hat{g}_k$ average $n_k^{-1/2}\sum_{t:s_t=k} g_t$ over all 512 samples. By Equation C.3, $\hat{g}_k$ is the mean gradient of an independent vector for group k, and $\rho\|\hat{g}_k\|_2$ is its value of Γ. Panel (a) reports each group's share of $\sum_k \|\hat{g}_k\|_2^2$. The readout set receives 1.9% of the shared update and holds 76% of the gradient. QUERY-IMAGE receives 61% of the update and holds 0.002% of the gradient, so it comes close to the zero-gradient case of Proposition C.2(ii): it enlarges the normalization count without adding to the gradient sum. On the held-out samples, the first-order loss decrease per unit update norm is 12.5 along the direction of one shared vector and 27.1 along that of the readout vector alone.

Gradient per token. Panel (b) reports, for each token set $\mathcal{S}$, the norm of $|\mathcal{S}|^{-1}\sum_{t\in\mathcal{S}} g_t$ after this vector is averaged over the 512 samples. The readout set reaches 19.2, 3.5 times FIRST, the largest context group at 5.5, and QUERY-IMAGE reaches 0.02. The readout value is below that of ANSWER-CUE (30.2) and of LAST (22.4) alone, because the mean gradients of these two tokens are nearly orthogonal, so their average is shorter than either of them.

Adding context. Panel (c) uses the unit descent directions $u_{\mathcal{A}}$ and $u_{\mathcal{B}}$ of the readout and context vectors, each the negative mean zero-update gradient of its vector over the fitting samples, scaled to unit norm. On the held-out samples, the readout vector is set to $a\,u_{\mathcal{A}}$ and the context vector to $c\,u_{\mathcal{B}}$, and the panel reports the mean answer CE in nats. The zero update gives 3.05. With $c = 0$, the loss falls to 1.12 at $a = 0.2$ and 1.07 at $a = 0.3$. Each context norm $c \in \{0.1, 0.2, 0.3\}$ lowers the whole curve, and $c = 0.3$ reaches 0.55 at $a = 0.2$. The two directions have cosine similarity 0.03, so one vector on $\mathcal{A} \cup \mathcal{B}$ cannot point along both. All directions come from zero-update gradients and all norms stay at most 0.3, so the figure describes the neighborhood of the zero update.

C.5 PROOF OF THEOREM 3.1

Tied coordinates. Fix the common group counts used in the theorem, and let $w \in \mathbb{R}^{qd}$ stack the transposed rows of W. The normalized independent-group coordinates of its tied update are

$$v = Q_q w, \qquad Q_q = (\Lambda S) \otimes I_d, \qquad \mathcal{U}_q = \mathrm{col}(Q_q), \qquad P_q = Q_q Q_q^\top. \tag{C.4}$$

Here $\otimes$ is the Kronecker product and I_d is the identity on $\mathbb{R}^d$. The active columns of ΛS are orthonormal, by summing n_k/N_r over groups assigned to row r, and inactive columns are zero. Hence P_q is an orthogonal projector, even with empty rows. For each $v \in \mathcal{U}_q$, $w = Q_q^\top v$ realizes v with inactive coordinates zero and $\|w\|_2 = \|v\|_2$. Inactive full-group coordinates have no effect on the margins and can also be set to zero.

Interpreting δ_q. To interpret δ_q, let $g_{jk} = \nabla_{v_k} h_j(0)/\sqrt{n_k}$ for active group k, where $v_k \in \mathbb{R}^d$ is its independent vector. This is the mean gradient with respect to that group's input embeddings. For active row r, put $\overline{g}_{jr} = \sum_{k:a(k)=r,\, n_k>0}(n_k/N_r) g_{jk}$. Projection onto $\mathcal{U}_q$ replaces each gradient block by $\sqrt{n_k}\,\overline{g}_{jr}$, so

$$\delta_q^2 = \max_j \sum_{r:N_r>0} \sum_{k:a(k)=r,\, n_k>0} n_k \|g_{jk} - \overline{g}_{jr}\|_2^2. \tag{C.5}$$

Thus readout/context sharing discards only within-row group contrasts. This identity does not assume that their gradients agree.

Proof of Theorem 3.1. The tied feasible set is contained in the unrestricted group-coordinate ball, so $m_q(\rho) \le m_R(\rho)$. Continuity and compactness give an unrestricted maximizer $v^\star$. Its projection $\overline{v} = P_q v^\star$ is tied-feasible and has norm at most ρ.

Gradient Lipschitzness along every segment from zero to a point in the ball gives

$$h_j(v) = h_j(0) + \nabla h_j(0)^\top v + r_j(v), \qquad |r_j(v)| \le \frac{\kappa}{2}\|v\|_2^2. \tag{C.6}$$

Indeed, the remainder is the integral of $\langle \nabla h_j(tv) - \nabla h_j(0), v\rangle$ over $t \in [0,1]$. Since $v^\star - \overline{v}$ is orthogonal to $\mathcal{U}_q$,

$$\begin{aligned} h_j(\overline{v}) &\ge h_j(v^\star) - \|(I - P_q)\nabla h_j(0)\|_2 \|v^\star - \overline{v}\|_2 - \frac{\kappa}{2}\big(\|v^\star\|_2^2 + \|\overline{v}\|_2^2\big) \\ &\ge h_j(v^\star) - \rho\delta_q - \kappa\rho^2. \end{aligned}$$

Taking the minimum over the finite margin collection and using feasibility of $\overline{v}$ proves the upper bound. Affine margins admit $\kappa = 0$, which gives equality of attainable margins when $\delta_q = 0$. □

From margins to cross-entropy. For any desired margin $\gamma \in \mathbb{R}$, an unrestricted update attaining every margin at least $\gamma + \rho\delta_q + \kappa\rho^2$ implies the existence of a tied update attaining every margin at least γ on these contexts. If the collection includes every competing token for each supervised prefix in a vocabulary of size $C \ge 2$, the tied logits $z \in \mathbb{R}^C$ at each prefix satisfy $z_a - z_c \ge \gamma$ for every $c \ne a$, where a is the correct token. Thus

$$-\log \mathrm{softmax}(z)_a = \log\left(1 + \sum_{c \ne a} e^{z_c - z_a}\right) \le c_\gamma := \log\big(1 + (C-1)e^{-\gamma}\big). \tag{C.7}$$

Averaging preserves this CE bound.

Varying counts. The theorem requires a single Q_q for all its contexts. Equal group counts suffice, and so do common normalized within-row proportions. With varying proportions, the shared row w induces different full-group vectors $Q_q(p)w$. For example, with two groups, one scalar row, and count profiles $(1,1)$ and $(1,4)$, these vectors are $(w/\sqrt{2}, w/\sqrt{2})$ and $(w/\sqrt{5}, 2w/\sqrt{5})$. No single full-group vector realizes both corrections for nonzero w. Conversely, the full-group vector $(1,0)$ is not tied on the first profile. Thus the two shared embedding-correction families need not be nested across varying-count prompts, and the theorem applies to each count profile separately.

D Paired supervision and demo-free risk

Equation 6 is a sum of answer-supervised source and target CE losses. It contains no hidden-state matching or soft-label objective. Throughout the whole-sample results below, n denotes the number of training query–answer pairs, not the optimizer's minibatch size. The source renderings are fixed for these comparisons. The same deterministic arguments apply to a source loss averaged over a prescribed demo-sampling rule whenever that averaged loss satisfies the stated continuity or differentiability assumptions.

D.1 Proof of Proposition 3.2

Proof. Let $w^+ = w - \eta g$. Integrating the target gradient along the segment gives

$$\begin{aligned} \mathcal{L}_{\mathrm{tgt}}(w^+) - \mathcal{L}_{\mathrm{tgt}}(w) &= -\eta \int_0^1 \langle \nabla \mathcal{L}_{\mathrm{tgt}}(w - t\eta g), g\rangle \, dt \\ &\le -\eta\langle g_{\mathrm{tgt}}, g\rangle + \frac{\beta\eta^2}{2}\|g\|_2^2, \end{aligned}$$

which proves Equation 7. □

Finite-step conditions. Write $a = \langle g_{\mathrm{tgt}}, g\rangle = \|g_{\mathrm{tgt}}\|_2^2 + \langle g_{\mathrm{tgt}}, g_{\mathrm{src}}\rangle$. If $a > 0$, then $g \ne 0$. For $\beta > 0$, a step satisfying $0 < \eta < 2a/(\beta\|g\|_2^2)$ and the segment smoothness assumption strictly lowers target loss. For $\beta = 0$, every positive step satisfying that assumption does so.

Comparison with a target-only step. For comparison with a target-only step, additionally assume the same gradient Lipschitz bound on the segment from w to $w - \eta g_{\mathrm{tgt}}$. Applying the integral remainder bound in the other direction yields

$$\mathcal{L}_{\mathrm{tgt}}(w - \eta g_{\mathrm{tgt}}) - \mathcal{L}_{\mathrm{tgt}}(w) \geq -\eta \|g_{\mathrm{tgt}}\|_2^2 - \frac{\beta\eta^2}{2}\|g_{\mathrm{tgt}}\|_2^2.$$

Subtracting this from the paired-step upper bound gives

$$\begin{aligned}\mathcal{L}_{\mathrm{tgt}}(w - \eta g) - \mathcal{L}_{\mathrm{tgt}}(w - \eta g_{\mathrm{tgt}}) \leq\ & -\eta\langle g_{\mathrm{tgt}}, g_{\mathrm{src}}\rangle \\ & + \frac{\beta\eta^2}{2}\big(\|g\|_2^2 + \|g_{\mathrm{tgt}}\|_2^2\big).\end{aligned} \tag{D.1}$$

If $c = \langle g_{\mathrm{tgt}}, g_{\mathrm{src}}\rangle > 0$ and $\beta > 0$, the paired step is strictly better at the same step size whenever $0 < \eta < 2c/[\beta(\|g\|_2^2 + \|g_{\mathrm{tgt}}\|_2^2)]$ and both segment assumptions hold. For $\beta = 0$, positivity of c suffices on those segments. These comparisons concern plain gradient steps from the same parameter value.

D.2 Compatibility of the empirical objectives

Fix $\rho > 0$ and the comparison class $\mathcal{W}_{q,\rho} = \{W \in \mathbb{R}^{q\times d} : \|W\|_F \leq \rho\}$. Assume both whole-sample branch losses are continuous on it. Define

$$\mathcal{L}_b^\star = \min_{W\in\mathcal{W}_{q,\rho}} \mathcal{L}_b(W), \qquad \mathcal{C}_{\mathrm{src|tgt}} = \min_{W\in\arg\min_{U\in\mathcal{W}_{q,\rho}} \mathcal{L}_{\mathrm{tgt}}(U)} \mathcal{L}_{\mathrm{src}}(W) - \mathcal{L}_{\mathrm{src}}^\star. \tag{D.2}$$

The nonnegative gap $\mathcal{C}_{\mathrm{src|tgt}}$ is the source-loss sacrifice required to remain target-optimal. The ball and the target minimizer set are nonempty compact sets, so all displayed minima exist.

Proposition D.1 (Target cost of incompatible supervision). If $\widehat{W} \in \mathcal{W}_{q,\rho}$ satisfies $\mathcal{L}(\widehat{W}) \leq \min_{W\in\mathcal{W}_{q,\rho}} \mathcal{L}(W) + \varepsilon_{\mathrm{opt}}$ for $\varepsilon_{\mathrm{opt}} \geq 0$, then

$$0 \leq \mathcal{L}_{\mathrm{tgt}}(\widehat{W}) - \mathcal{L}_{\mathrm{tgt}}^\star \leq \mathcal{C}_{\mathrm{src|tgt}} + \varepsilon_{\mathrm{opt}}. \tag{D.3}$$

Proof. Choose a target minimizer $W^\dagger$ attaining the source minimum in Equation D.2. Approximate paired optimality implies

$$\mathcal{L}_{\mathrm{tgt}}(\widehat{W}) + \mathcal{L}_{\mathrm{src}}(\widehat{W}) \leq \mathcal{L}_{\mathrm{tgt}}^\star + \mathcal{L}_{\mathrm{src}}(W^\dagger) + \varepsilon_{\mathrm{opt}}.$$

Use $\mathcal{L}_{\mathrm{src}}(\widehat{W}) \geq \mathcal{L}_{\mathrm{src}}^\star$ and subtract $\mathcal{L}_{\mathrm{tgt}}^\star$ to obtain the upper bound. The lower bound follows from the definition of the target minimum. □

Shared minimizers. The symmetric source bound follows by exchanging branches. If the two losses share a minimizer, both compatibility gaps vanish and an approximate paired minimizer is within $\varepsilon_{\mathrm{opt}}$ of each branch optimum on the stated ball.

A counterexample. Shared answer labels do not ensure compatible responses. For a scalar binary-logit example, margins $h_{\mathrm{src}}(w) = w$ and $h_{\mathrm{tgt}}(w) = -w$ give losses $\log(1 + e^{-w})$ and $\log(1 + e^{w})$. Their gradients cancel at zero, and no w gives both margins a positive value. Replacing the source margin by $2w$ makes a paired gradient step at zero increase target loss. The compatibility gap $\mathcal{C}_{\mathrm{src|tgt}}$ is the quantity that separates this case from compatible supervision.

D.3 Demo-free risk guarantees

Let $n \geq 1$ and let $(x_i, y_i)_{i=1}^n$ be independent, identically distributed query–answer pairs from the deployment distribution of one task. The frozen model, target rendering, structural partition, row assignment, and radius $\rho > 0$ are fixed independently of this sample. Assume the losses are measurable in the example, and define

$$\mathcal{R}_{\mathrm{tgt}}(W) = \mathbb{E}_{(x,y)}\ell(y \mid \mathrm{render}(\varnothing, x), W). \tag{D.4}$$

The empirical mean is $\mathcal{L}_{\mathrm{tgt}}$ in Equation 6. Source demos may be reused or chosen using the training sample: the concentration argument uses only the n independent target examples. Paired renderings, answer tokens, and repeated minibatches do not increase this sample count.

Assumption D.2 (Bounded and Lipschitz target losses). There are constants $0 < M_\ell < \infty$ and $0 \le G_\ell < \infty$ such that, for every target example in the distribution's support and all $W, W' \in \mathcal{W}_{q,\rho}$, its loss lies in $[0, M_\ell]$ and

$$|\ell(y \mid p, W) - \ell(y \mid p, W')| \le G_\ell \|H(p)(W - W')\|_F, \qquad p = \mathrm{render}(\varnothing, x). \tag{D.5}$$

These conditions are uniform over examples and the comparison class. Uniform bounds on logits and input derivatives over the ball supply such constants. The radius is an analysis device, not an extra penalty, clipping rule, or projection in STAVE.

Theorem D.3 (Uniform target-risk deviation). Under Assumption D.2, for fixed $\epsilon \in (0, \rho]$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$\sup_{W \in \mathcal{W}_{q,\rho}} |\mathcal{R}_{\mathrm{tgt}}(W) - \mathcal{L}_{\mathrm{tgt}}(W)| \le \mathfrak{B}_n(\epsilon, \delta) := 2G_\ell \epsilon + M_\ell \sqrt{\frac{qd \log(1 + 2\rho/\epsilon) + \log(2/\delta)}{2n}}. \tag{D.6}$$

Here ϵ is the covering radius, δ the failure probability, and $\mathfrak{B}_n$ the deviation bound.

Proof. Proposition C.1 makes every target loss G_ℓ-Lipschitz in $\|W - W'\|_F$. The comparison class is a Euclidean ball of dimension qd. A maximal set separated by more than ϵ is an ϵ-net. Its disjoint open balls of radius $\epsilon/2$ lie inside a ball of radius $\rho + \epsilon/2$, so a volume comparison bounds the net size by $(1 + 2\rho/\epsilon)^{qd}$.

For completeness, if $X \in [0, M_\ell]$, the log moment-generating function $\psi(s) = \log \mathbb{E} e^{s(X - \mathbb{E}X)}$ has $\psi(0) = \psi'(0) = 0$ and $\psi''(s) \le M_\ell^2/4$. The second derivative is a variance under an exponentially tilted distribution, still supported on $[0, M_\ell]$, and any such variance is at most its second moment about $M_\ell/2$, hence at most $M_\ell^2/4$. Integrating twice gives $\psi(s) \le s^2 M_\ell^2/8$. Independence and the exponential Markov inequality, optimized over s for each tail, therefore give at every fixed net point

$$\Pr\left(|\mathcal{L}_{\mathrm{tgt}} - \mathcal{R}_{\mathrm{tgt}}| > t\right) \le 2 \exp(-2nt^2/M_\ell^2).$$

A union bound over the net yields the square-root term in Equation D.6. Every point of the ball is within ϵ of a net point. The empirical and population losses each change by at most $G_\ell \epsilon$, proving the result. Continuity in W permits the supremum to be taken over a countable dense subset, so the event is measurable. □

Corollary D.4 (Conditional target risk of paired training). On the event of Theorem D.3, suppose the empirical source loss is continuous on $\mathcal{W}_{q,\rho}$ and $\widehat{W}$ satisfies the radius and paired optimization condition of Proposition D.1. Then

$$\mathcal{R}_{\mathrm{tgt}}(\widehat{W}) - \inf_{W \in \mathcal{W}_{q,\rho}} \mathcal{R}_{\mathrm{tgt}}(W) \le \mathcal{C}_{\mathrm{src|tgt}} + \varepsilon_{\mathrm{opt}} + 2\mathfrak{B}_n(\epsilon, \delta). \tag{D.7}$$

Proof. Uniform deviation and Proposition D.1 give

$$\mathcal{R}_{\mathrm{tgt}}(\widehat{W}) \le \mathcal{L}^\star_{\mathrm{tgt}} + \mathcal{C}_{\mathrm{src|tgt}} + \varepsilon_{\mathrm{opt}} + \mathfrak{B}_n.$$

The same event implies $\mathcal{L}^\star_{\mathrm{tgt}} \le \inf_W \mathcal{R}_{\mathrm{tgt}}(W) + \mathfrak{B}_n$. Combining the two inequalities proves the claim. □

Capacity term. The uniform event covers any sample-dependent choice of $\widehat{W}$ in the fixed ball, including one fitted using source prompts. The capacity term uses $2d$ coordinates for STAVE and $7d$ for independent group vectors, a comparison made at common radius and loss constants. No common-count assumption is needed for this statistical result. When the class is chosen from M_{cfg} predetermined configurations, a union bound replaces $\log(2/\delta)$ by $\log(2M_{\mathrm{cfg}}/\delta)$.

E Inference cost

Prompt tokens and storage. Let T_{src} and T_{tgt} be decoder-input lengths with and without demos for the same query, including expanded visual tokens. STAVE inference uses exactly T_{tgt} prompt tokens. Counts and assignments can be constructed in $O(T_{\mathrm{tgt}})$ time from fixed structural metadata.

Adding the scaled vectors costs $O(T_{\text{tgt}}d)$ operations and can be performed in place. Storage is $2d$ task-specific scalars, independent of query count and demo length. No independent full-group matrix needs to be stored for inference.

KV cache. For a decoder with N_{layer} layers, h_{kv} KV heads per layer, and head dimension d_h, a conventional uncompressed prompt cache stores

$$\text{KVElements}(T) = 2N_{\text{layer}}Th_{\text{kv}}d_h. \tag{E.1}$$

Here T is the number of cached tokens and the factor two counts keys and values. For equal precision, batch size, and output length, STAVE and zero-shot inference have the same cache shape. The values in that cache differ. Relative to keeping demos, the saved prompt-cache elements are $2N_{\text{layer}}(T_{\text{src}} - T_{\text{tgt}})h_{\text{kv}}d_h$ when the source prompt is longer. Both caches grow identically per subsequently generated token, although the number of generated tokens can differ.

Prefill computation. Under conventional dense attention, the prefill attention term is $O(N_{\text{layer}}T^2d)$ and the positionwise decoder terms are linear in T for a fixed architecture. STAVE preserves the target prompt's decoder dimensions and adds only the embedding-update pass. A soft prompt of length P instead changes the attention-size term from T^2 to $(T + P)^2$. A KV prefix adds attention-memory entries even when it computes no new attention queries.

Visual preprocessing and latency. The visual encoder and connector process the query image as in zero-shot inference. Demo removal avoids the demo prompt tokens and, when demos would otherwise be encoded at inference, their visual preprocessing. Measured latency also depends on kernels, precision, batching, and stopping behavior, so Appendix E.1 runs every method under one setting.

Training cost. Both paired branches run the frozen model, and gradients propagate through the decoder to the input vectors. Backbone optimizer states and weight updates are absent, while activation storage and backward computation remain, and the source branch includes demos. This one-time training cost is separate from the per-query inference cost above.

E.1 Measured inference cost

Table 7 reports measured per-query cost for the Idefics2-8B VQAv2 entries of Table 1. Every method loads the checkpoint that produces the score in that table and runs on the same H200 inside one job, with batch size 1, fp16 weights, three beams, and the same 100 validation samples after warmup. TTFT is one prefill call with a single new token and is the like-for-like measure of prompt cost. Because the methods emit answers of different lengths, the Fixed-20 column forces exactly 20 generated tokens for every method. TTFT and the decode column time the generation call alone, with decode taken from the 20-token generation time, while Fixed-20 is end to end and also includes the processor pass over the prompt text and images. State is the stored task state. STAVE only adds its vectors to the input embeddings before the first decoder block, so its decoder computation during prefill and decoding is that of zero-shot inference. The zero-shot and STAVE rows therefore share one value in the TTFT, decode, and Fixed-20 columns, the mean of their two measurements, which differ by less than 1%.

STAVE keeps the prompt length, prefill FLOPs, and KV cache of zero-shot inference, at 128 tokens. MimIC, HiFICL, and LIVE run the same prompt through an attention wrapper, which multiplies TTFT by 1.03 to 1.10 relative to STAVE. LoRA keeps the prompt but adds low-rank branches in the decoder, the visual encoder, and the connector, which multiplies TTFT by 1.18 and the per-token decode cost by 1.41. ICL with 32 demos needs 60 times the TTFT, 31 times the prefill FLOPs, and 52 GiB more peak memory than STAVE.

STAVE has the lowest stored state of any trained method here and matches the TTFT, per-token decode cost, and fixed-length latency of zero-shot inference, below every other row, and Table 1 gives it the highest accuracy on this model and dataset, so nothing in this comparison trades accuracy for speed against it.

Figures 8 and 9 report six cost dimensions against the demo count on Idefics2-8B VizWiz and OK-VQA and Qwen-VL-7B OK-VQA, measured on one A100 40GB with batch size 1, fp16 weights, greedy decoding, at most 10 new tokens, and a shared stop rule, as medians over 500 test queries. They use the full group set. Prompt tokens, prefill FLOPs, KV cache, TTFT, and peak activation

memory depend only on the prompt and are identical for the two-vector configuration. Answer latency also depends on the generated answer length and includes the CPU pass that builds the group assignment (4 to 14 ms). Qwen-VL with 32 demos exceeds the 40 GB memory and is omitted. On all three model and dataset pairs, STAVE keeps the prompt tokens, prefill FLOPs, KV cache, and TTFT of zero-shot inference, while ICL cost grows with every added demo. At 16 demos, ICL multiplies the TTFT of STAVE by 8 to 13 and its peak activation memory by 14 to 17.

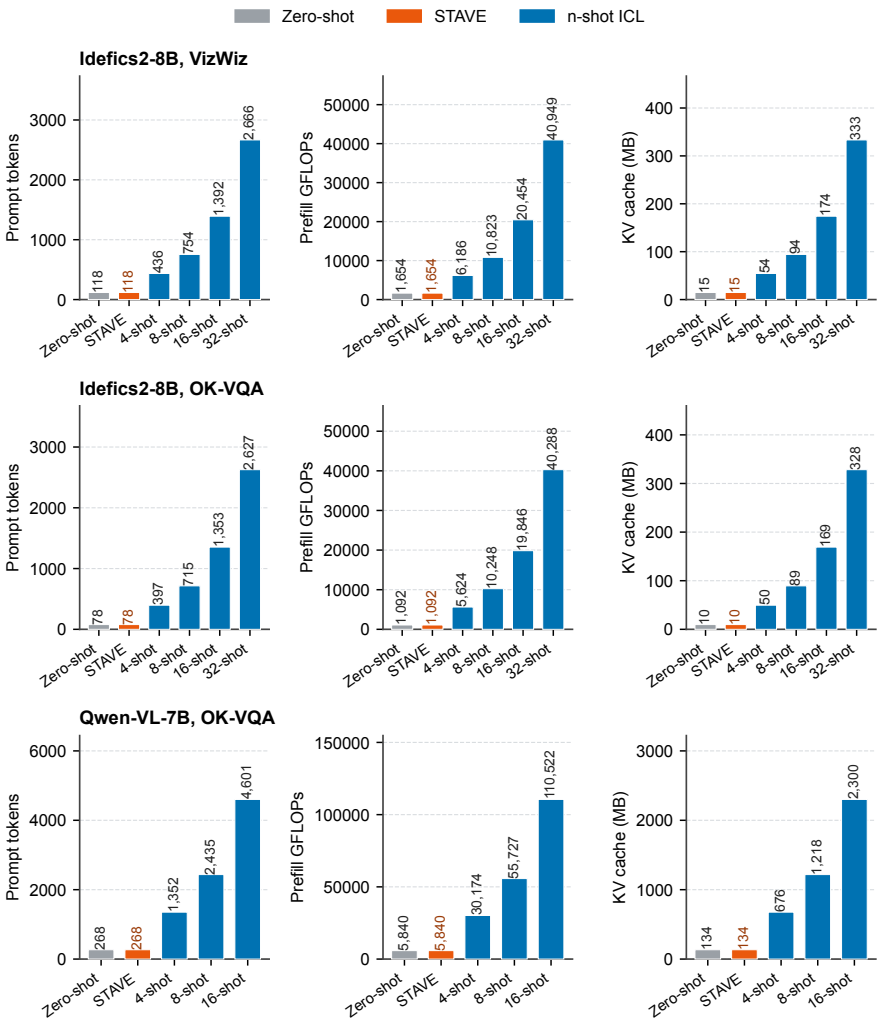


Figure 8: Prompt tokens, prefill FLOPs, and KV cache against the demo count.

E.2 Stored task state

Table 8 lists the task-specific state each method stores per task and backbone at hidden dimension $d = 4096$. STAVE stores two vectors. The full group set stores one vector per structural token group. ICV stores one vector per layer, added after the MLP of each block. MTV stores one vector per attention head across layers, and LIVE stores one vector and one scale per layer. I2CL stores one attention-output vector and one MLP-output vector per layer, with two scales for each. MimIC and HiFICL store trained attention modules, and LoRA stores low-rank weight updates. Parameter counts are read back from the loaded module of each method and match the main tables. Sizes assume fp32 storage. The Idefics2 MimIC entry is the original encoder, which adds one shift vector per layer rather than learning a projection of the query states. The largest state here, LoRA on LLaVA, is 2,399 times that of STAVE.

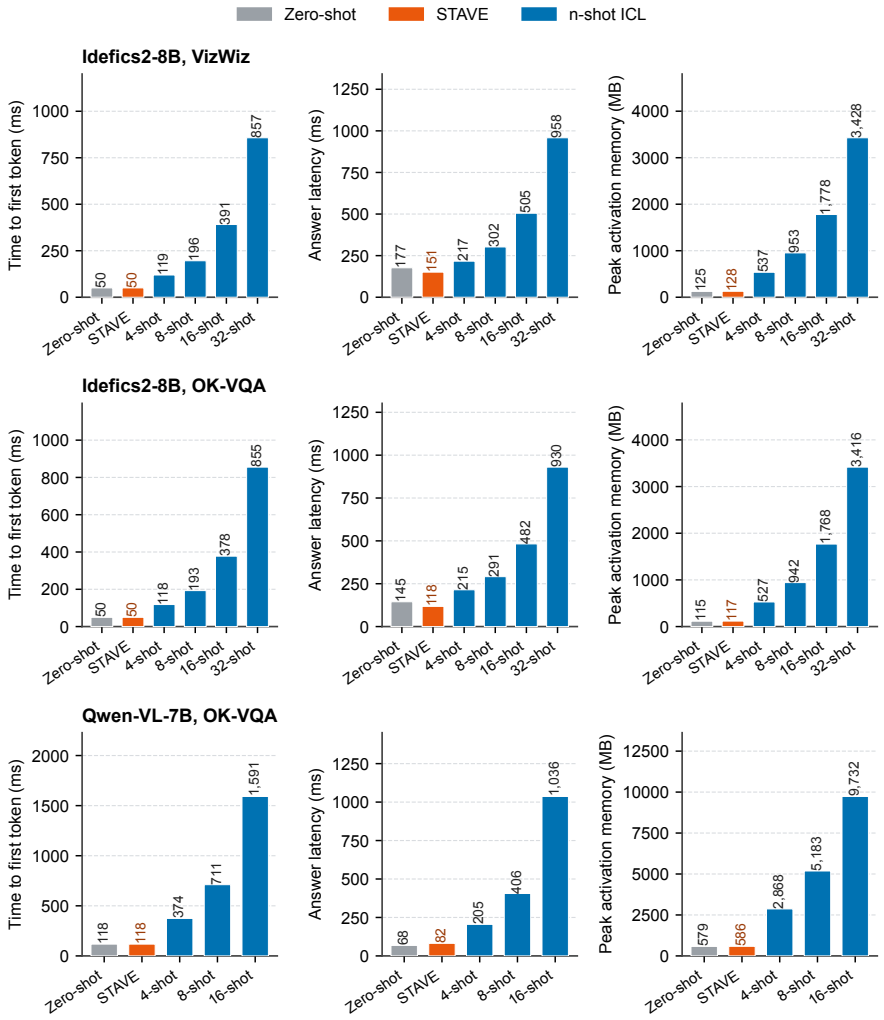


Figure 9: TTFT, answer latency, and peak activation memory against the demo count.

Table 7: Measured per-query inference cost on Idefics2-8B VQAv2.

Method	State (KiB)	Prompt tok.	TTFT (ms)	Decode (ms/tok)	Fixed-20 (ms)	Peak mem. (GiB)
Zero-shot	0	128	**98**	**23.9**	**560**	16.5
8-shot ICL	0	1,026	1,430	24.0	2,021	29.8
16-shot ICL	0	1,922	2,854	24.4	3,605	43.0
32-shot ICL	0	3,717	5,877	26.1	6,933	68.7
LoRA (r=16)	68,704	128	116	33.6	763	16.5
LIVE	512	128	101	27.4	629	16.5
MimIC	1,028	128	106	31.6	713	16.5
HiFICL	8,704	128	108	34.3	767	16.5
STAVE	**32**	128	**98**	**23.9**	**560**	16.5

Table 8: Stored task state per task and backbone ($d = 4096$, fp32).

Method	Stored state	Parameters	Size
Zero-shot, ICL	none	0	0
Single vector (TV, FV)	one embedding vector	4,096	16 KiB
ICV	one vector per layer	131,072	512 KiB
MTV	one vector per head	131,072	512 KiB
LIVE	one vector and one scale per layer	131,104	512 KiB
I2CL	two vectors and four scales per layer	262,272	1.0 MiB
MimIC (Idefics2 / LLaVA)	attention modules	263,168 / 17,040,384	1.0 MiB / 65.0 MiB
HiFICL	attention modules	2,228,224	8.5 MiB
LoRA (Idefics2 / LLaVA)	low-rank weight updates	17,588,224 / 19,652,608	67.1 MiB / 75.0 MiB
STAVE	two embedding vectors	8,192	32 KiB

E.3 Performance overview settings

Figure 1 uses reported results from four tables. From Table 1, it shows LLaVA and Idefics2-8B on VQAv2, OK-VQA, and COCO. From Table 2, it shows Idefics2-8B on VizWiz, DTD, and Flowers and Qwen-VL-7B on VizWiz and OK-VQA. From Table 3, it shows Pythia 2.8B. TV benchmark accuracy aggregates its 18 tasks, weighted by test queries. Bar labels show the reported accuracy percentages or COCO CIDEr scores at the precision printed in each table. Each performance bar's length is the ratio of its score to STAVE's score for that model and dataset, with the inner edge representing zero.

The params sector uses the Idefics2-8B parameter counts of Table 1. Its bar length is $1 - \log_{10}(P_m/P_{\mathrm{STAVE}})/4$ of the full length, where P_m is method m's parameter count, so STAVE has the full length and each factor of 10 removes a quarter of it. The latency sector shows the fixed-20 latencies printed in Table 7. Its bar length is t_{STAVE}/t_m of the full length, where t_m is method m's latency, so a longer bar is faster. These measurements are for Idefics2-8B VQAv2 under the settings of that table. ICL uses 16 demos in the Table 1 and latency sectors, four in Table 2, and 15 on the TV benchmark. LoRA uses rank 8 on the TV benchmark and rank 16 elsewhere. Every value in the overview is copied from the cited table.

F Implementation details

F.1 Backbones

Table 9 lists the eleven frozen backbones used in the experiments. The hidden dimension d fixes the size of the learned matrix, which holds $2d$ parameters on every backbone, and N_{layer} is the number of decoder blocks. Image tokens is the number of decoder-input tokens one query image occupies after the frozen visual encoder and connector expand it, under the image settings used here. The readout vector covers at most a few tokens on every backbone, so these counts are the scale the context vector is normalized against.

Idefics2 and Idefics3 run without image splitting, and Qwen2.5-VL images are resized to 448 pixels, so each of the six LMMs has a fixed count. Idefics2 appears under two checkpoints, `HuggingFaceM4/idefics2-8b-base` in Table 1 and `HuggingFaceM4/idefics2-8b` in Table 2, written Idefics2-8B-base and Idefics2-8B-instruct where both appear, and the model written as LLaVA in Table 1 is `llava-hf/llava-interleave-qwen-7b-hf`. The other LMM checkpoints are `Qwen/Qwen-VL-Chat`, `OpenGVLab/InternVL3_5-8B`, `Qwen/Qwen2.5-VL-7B-Instruct`, and `HuggingFaceM4/Idefics3-8B-Llama3`. The LLM checkpoints are the EleutherAI `pythia-2.8b`, `pythia-6.9b`, and `pythia-12b` releases together with `EleutherAI/gpt-j-6B` and `huggyllama/llama-7b`. Panel (a) names the language model each backbone is built from, and several released checkpoints are instruction-tuned: the Idefics3 decoder starts from Llama-3.1-8B-Instruct, the LLaVA-Interleave decoder starts from Qwen1.5-7B-Chat, and Qwen-VL and Qwen2.5-VL are used in their chat and instruct releases. Training and evaluation run on NVIDIA A100 (40GB) and H200 GPUs.

Table 9: Frozen backbones.

(a) LMMs

Backbone	Language model	Vision encoder and connector	d	N_{layer}	Img.
Idefics2-8B	Mistral-7B	SigLIP-SO400M, perceiver resampler	4096	32	64
LLaVA-Interleave-7B	Qwen1.5-7B	SigLIP-SO400M, MLP projector	4096	32	729
Qwen-VL-7B	Qwen-7B	OpenCLIP ViT-bigG, cross-attention	4096	32	256
InternVL3.5-8B	Qwen3-8B	InternViT-300M, pixel shuffle, MLP	4096	36	256
Qwen2.5-VL-7B	Qwen2.5-7B	Native-resolution ViT, patch merger	3584	28	256
Idefics3-8B	Llama-3.1-8B	SigLIP-SO400M, pixel shuffle, MLP	4096	32	169

(b) LLMs

Backbone	d	N_{layer}	Family
Pythia 2.8B	2560	32	GPT-NeoX
Pythia 6.9B	4096	32	GPT-NeoX
Pythia 12B	5120	36	GPT-NeoX
GPT-J 6B	4096	28	GPT-J
LLaMA 7B	4096	32	LLaMA

F.2 VQAv2, OK-VQA, AND COCO

Table 1 follows the protocol of MimIC and HiFICL, and all methods, STAVE included, share the same data, prompts, and decoding. The training set is the first 1,000 samples of the released shuffle at seed 3407, which is also the demo pool. The trained methods are evaluated without demos on the full evaluation set, with 10,000 VQAv2 questions, the 5,046 validation questions of the official OK-VQA v1.1 release, and 5,000 COCO Karpathy validation images. ICL draws the demos of each query from the same pool without replacement, with seeds 41, 42, and 3407. Zero-shot and 8-shot ICL are scored on the full evaluation set, and 16-shot and 32-shot ICL on the first 1,000 samples of the shuffled evaluation order. Zero-shot has no demos and deterministic decoding, so the three seeds give the same score. The prompts are the released templates, an instruction-style prefix for the VQA tasks and a caption-style prefix for COCO (Appendix F.6). Decoding uses three beams on both backbones, at most 20 new tokens, length penalty 0, and batch size 1. Post-processing and scorers are the released ones, and CIDEr uses pycocoevalcap with CoreNLP 3.4.1. Published CIDEr values reported as fractions are multiplied by 100.

MimIC and HiFICL report results from the best-performing epoch on the evaluation split of these benchmarks, and we follow this protocol: every epoch is scored on the evaluation split and the best epoch is reported, with ties going to the earlier epoch. Tables 2 and 10 instead select the checkpoint on validation samples (Appendix F.3). STAVE trains its $2d = 8{,}192$ parameters for five epochs with the source and target CE losses weighted 1 to 1, using AdamW with weight decay 10^{-3} and a cosine schedule with 10% warmup. The learning rate is 10^{-3} or 3×10^{-3} set per model and dataset, each training step uses $K = 4$ paired examples ($K = 16$ on LLaVA COCO), and the source prompt holds one demo on OK-VQA and eight on COCO.

All compared methods run with the authors' released code. LIVE (Peng et al., 2024), MimIC (Jiang et al., 2025), and HiFICL (Li et al., 2026) follow the settings of their papers. PT follows the setting of Lester et al. (2021) with vocabulary initialization. The three PT modes learn 20 virtual tokens each on the prompt without demos and train for five epochs with AdamW. We tune the PT learning rate between 10^{-3} and 3×10^{-2}. PT is sensitive to the learning rate and hard to tune, and a learning rate that trains stably at one random seed can collapse at another seed on the same model and dataset.

LoRA appears in Tables 1 and 3 with different ranks and settings. In Table 1 it follows the LoRA configuration released with HiFICL. It applies rank 16 to the attention query, key, value, and output projections of the language model and, on Idefics2, of the perceiver connector, and to the query, key, and value projections of the visual encoder. The MLP connector of LLaVA is not adapted. It trains with AdamW at learning rate 5×10^{-4}, weight decay 10^{-3}, and a cosine schedule with 10% warmup. It uses batch size 2 with 8 accumulation steps, the setting of the released LoRA script. It follows the five-epoch schedule. The rank 8 setting of Table 3 is given in Appendix F.4.

F.3 VizWiz, OK-VQA, and fine-grained classification

Tables 2 and 10 follow the protocol of MTV (Huang et al., 2024) and STV (Ma et al., 2026), and every method uses the STAVE split. The pool is a subsample of the training split drawn with seed 41, the validation block follows the fitting block, and the test set is the full held-out split. MimIC draws its teacher demos only from its training records. On VizWiz and OK-VQA the validation questions are those of STAVE.

Table 2 pairs a weaker and a stronger frozen checkpoint. Qwen-VL-7B, used in its chat release, is the stronger one on the two VQA datasets, with zero-shot accuracy of 45.49% on VizWiz and 57.51% on OK-VQA against 37.74% and 49.44% for Idefics2-8B. Four demos raise its average from 70.35% to 73.53%, whereas they lower the Idefics2-8B average from 70.57% to 69.72%. The pair therefore tests whether the learned vectors help both when the frozen model leaves much room for improvement and when it already answers many questions correctly. STAVE has the highest VizWiz and OK-VQA accuracy and the best average with both checkpoints. On Qwen-VL-7B OK-VQA, PT and the three trained per-layer interventions fall below zero-shot, while STAVE raises accuracy from 57.51% to 62.39%.

All compared methods start from the authors' released code, and we adapt each one to the datasets and protocol of these tables for a fair comparison. Following their papers, MimIC and HiFICL train for 10 epochs with AdamW (Loshchilov & Hutter, 2017) at learning rate 5×10^{-3} and weight decay 10^{-3}, batch size 2 with 2 accumulation steps, a cosine schedule with 10% warmup, and gradient clipping at 1.0. MimIC uses a four-shot teacher with loss weights 0.5 for CE and 1 for alignment. HiFICL uses rank 8 and eight virtual tokens. Decoding is greedy. Each of the 10 epochs is scored on the validation samples with the test metric, ties break by the lower mean teacher-forced answer CE and then by the earlier epoch, and only the selected epoch is evaluated on the test split.

STAVE trains its $2d$ parameters with the source and target CE losses weighted 1 to 1, using AdamW with weight decay 0 and a cosine schedule that decays the learning rate to zero. The source prompt holds one demo on VizWiz and OK-VQA and two on DTD, Flowers, and CUB. Each run takes 250 to 4,000 training steps of $K = 4$ or $K = 8$ paired examples, with a learning rate between 10^{-3} and 10^{-2} set per model and dataset. As for MimIC and HiFICL, and unlike Table 1, the checkpoint is selected on the validation samples, and only the selected checkpoint is evaluated on the test split. The three PT modes in Table 2 also learn 20 virtual tokens each on the prompt without demos. They train for 500 training steps of $K = 4$ samples with AdamW at weight decay 10^{-2} and a cosine schedule after a linear warmup, with the CE of the first answer token as the loss. The learning rate is 10^{-2} in most entries. When a run collapsed at one seed, that seed was trained again at another learning rate between 3×10^{-4} and 5×10^{-3}, so the three seeds of an entry do not always share one learning rate.

F.4 TV benchmark

Table 3 uses the 18 tasks of the TV benchmark (Hendel et al., 2023) with five frozen LLMs. Each task has one split, drawn with seed 41, of at most 256 training, 64 development, and 200 test queries. Every baseline uses the same prompts, training samples, and test queries as STAVE, and every per-task choice of a baseline is made on the demo-free development split, never on the test split. Each baseline runs with the authors' released code and follows the settings of its paper. TV and FV (Todd et al., 2024) choose the injection layer per task, as in their papers. SITE (Park et al., 2026) runs all three stages, with the task embedding averaged over 50 prompts of 10 demos. PT again learns 20 virtual tokens. LoRA follows Hu et al. (2021) and applies rank 8 with scaling $\alpha = 16$ to the attention query and value projections of every decoder block, or to the fused query, key, and value projection on Pythia. It trains with AdamW at weight decay 0.01, $K = 8$, and at most 500 training steps with early stopping on the development loss, and its learning rate is 10^{-4} or 10^{-3}, chosen per task by the development loss. STAVE picks its configuration per task by the lowest demo-free development CE, with one demo in the source prompt. The chosen configurations use AdamW with weight decay 0.01 on 84 of the 90 backbone and task pairs and 0 on the rest, 500 or 1,000 training steps, $K = 8$ paired examples per step on 74 pairs and 16 or 32 on the rest, and a learning rate between 0.01 and 0.5. The context vector uses 0.003 to 0.3 times the learning rate of the readout vector. The average weights each task by its number of test queries.

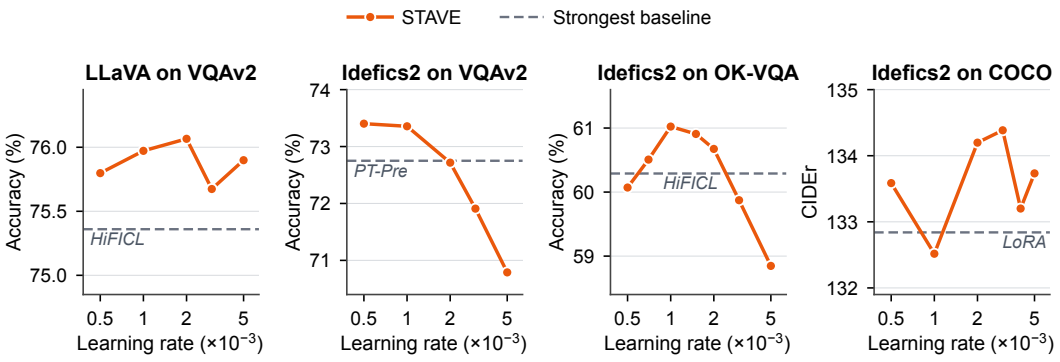


Figure 10: Learning-rate sensitivity on four model and dataset pairs of Table 1.

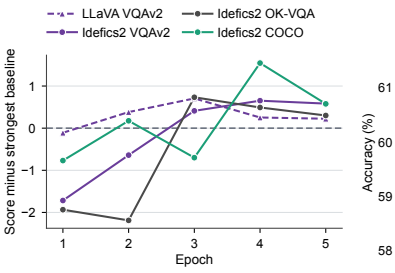


Figure 11: Score after each training epoch, relative to the strongest baseline.

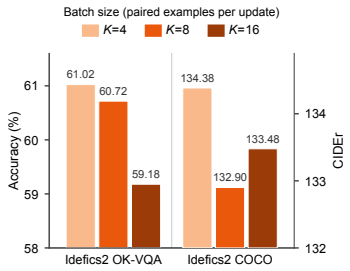


Figure 12: Batch size K on two Idefics2 datasets (axes start above zero).

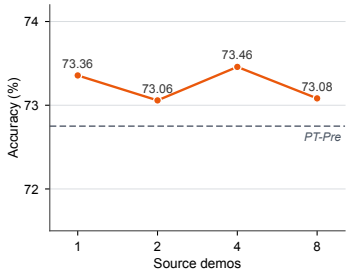


Figure 13: Number of source demos on Idefics2 VQAv2.

F.5 Hyperparameter sensitivity

Figures 10 to 13 vary one training setting at a time on four model and dataset pairs of Table 1, LLaVA VQAv2 and Idefics2 VQAv2, OK-VQA, and COCO. Every run uses seed 41 and the 1,000 training samples of that table and reports its best epoch on the evaluation split, the checkpoint selection used for its entries. Unless a figure varies them, runs use $K = 4$ paired examples per training step in Equation 6, five epochs, weight decay 0.001, and one source demo on OK-VQA or eight on COCO. Figures 11 and 12 use the learning rate with the highest score in Figure 10. The strongest baseline for a model and dataset is the best non-STAVE entry of its column in Table 1. Scores are accuracy (%) for VQAv2 and OK-VQA and CIDEr for COCO.

Learning rate. STAVE stays above the strongest baseline over most of the learning-rate range in Figure 10. On LLaVA VQAv2 every learning rate from 5×10^{-4} to 5×10^{-3} is above it, and on Idefics2 COCO every one except 10^{-3}. On Idefics2 VQAv2 the score decreases as the learning rate grows and remains above the strongest baseline up to 10^{-3}. Idefics2 OK-VQA peaks at 10^{-3} and exceeds the strongest baseline from 7×10^{-4} to 2×10^{-3}.

Training epochs. Figure 11 follows the score after every epoch. All four model and dataset pairs are above the strongest baseline from the fourth epoch on, including the last one, so the gain over the strongest baseline does not depend on selecting a single epoch.

Batch size. Figure 12 varies K on the two Idefics2 datasets and compares STAVE with itself. $K = 4$ gives the highest score on both datasets. With the learning rate and the number of epochs fixed, a larger K means fewer training steps per epoch, 63 at $K = 16$ against 250 at $K = 4$, and the OK-VQA score decreases as K grows.

Number of source demos. Figure 13 varies the number of source demos on Idefics2 VQAv2 from one to eight, with every other setting fixed. The score changes by at most 0.40 points, from 73.06% to 73.46%, and does not rise or fall with the number of demos. Every setting remains above the strongest baseline, PT-Pre at 72.75%. The number of source demos therefore has little effect on STAVE.

F.6 Dataset and prompt examples

This subsection shows the decoded, demo-free query block and its reference answer. The VQA and captioning queries come from the evaluation splits. The source branch used during training places demos before the same query block. Angle-bracketed image markers mark where the image shown beside the prompt is placed. Each visual sample is identified by its dataset ID or original file path.

F.6.1 VQA and captioning

VQAv2: question ID 575633002



Decoded LLaVA-Interleave prompt

```
Instruction: provide an answer to the question.
Use the image to answer.
Image:<image>Question:How many sides does the STOP
sign have?
Answer:
```

Reference answer: 8. The ten annotations are {8, 8, 2, 8, 8, 8, 8, 8, 8, 8}.

OK-VQA: question ID 4516835



Decoded prompt

```
<image>What black veggie is on this pizza? Answer:
```

Reference answer: olives. Eight annotators answered “olives” and two answered “olive.”

VizWiz: question ID VizWiz_val_00001816



Decoded prompt

```
First carefully understand the given examples.
Then use the given image and answer the question
in the same way as the examples.
If the question can not be answered, respond
unanswerable. <image>What is this? Answer:
```

Reference answer: coffee creamer. The ten annotations include “coffee creamer,” “coffee mate,” and more specific variants such as “french vanilla coffee creamer.”

COCO Captioning: image ID 306229



Decoded Idefics2 prompt

```
Image:<image><end_of_outterance>
Caption:
```

Reference captions: “This is an air show put on by the US Navy Blue Angels.” Other references describe five airplanes flying in formation, smoke from their tails, and a group of small airplanes in the sky. The separator keeps the spelling of the released MimIC Idefics2 template.

F.6.2 Fine-grained visual classification

The three classification datasets use the same two-choice query format, and evaluation uses the same format on the test split. Each example below is the first query of the fitting split, which the target branch sees without demos. The options are shuffled deterministically. The answer is an option letter, while the class name below makes the label explicit.

Describable Textures Dataset: query banded/banded_0005.jpg



Decoded prompt

```
<image>What is the type of texture in the image?
A.crosshatched B.banded
Answer with the option's letter from the given
choice directly. Answer:
```

Reference answer: B, banded.

Flowers-102: query image_06765.jpg



Decoded prompt

```
<image>What is the type of flower in the image?
A.purple coneflower B.pink primrose
Answer with the option's letter from the given
choice directly. Answer:
```

Reference answer: B, pink primrose.

CUB-200-2011: query Black_Footed_Albatross_0009_34.jpg



Decoded prompt

```
<image>What is the type of bird in the image?
A.Crested Auklet B.Black footed Albatross
Answer with the option's letter from the given
choice directly. Answer:
```

Reference answer: B, Black footed Albatross.

F.6.3 TV BENCHMARK TASKS

Every TV benchmark task uses the demo-free template `example:<input>->`. The answer tokens follow the arrow during supervision. In the translation and linguistic tasks, the input and the answer each start with one space, as in `example: chien->`, and a pair is kept only when its answer is a single token for the backbone's tokenizer. The algorithmic and knowledge tasks add no space and apply no single-token filter. The entries below are input-output pairs that remain after filtering on every backbone, or records from the deterministic algorithmic task generators.

Each task draws up to 200 test queries, and a task with a small input space yields fewer. Next letter and Previous letter have 25 possible inputs and the two case tasks have 26, so their test splits hold 12 or 13 queries. The single-token filter also shrinks some translation and linguistic tasks. English to Spanish holds 33 to 110 test queries depending on the backbone, while French to English and Spanish to English hold 200. An unweighted mean over the 18 tasks would give a 12-query split the same influence as a 200-query split, so every TV benchmark average in this paper weights each task by its number of test queries. The reported figure is therefore the accuracy over the pooled test queries of the suite, 2,138 to 2,202 per backbone. Every column uses the same weighting, and selection is unaffected because it is made per task on the development split.

Algorithmic tasks

Task	Prompt	Reference answer
Next letter	`example:a->`	b
Previous letter	`example:b->`	a
List first	`example:a,b,c->`	a
List last	`example:a,b,c->`	c
To uppercase	`example:a->`	A
To lowercase	`example:A->`	a

Translation tasks

Task	Prompt	Reference answer
French to English	`example: chien->`	dog
Spanish to English	`example: perro->`	dog
English to French	`example: good->`	bon
English to Spanish	`example: two->`	dos

Linguistic tasks

Task	Prompt	Reference answer
Present to gerund	`example: run->`	running
Present to past	`example: run->`	ran
Plural to singular	`example: children->`	child
Antonym	`example: hot->`	cold

Knowledge tasks

Task	Prompt	Reference answer
Country to capital	`example:Germany->`	Berlin
Person to language	`example:Gilad Atzmon->`	Hebrew
Location to continent	`example:Paris->`	Europe
Location to religion	`example:Edwin of Northumbria->`	Christian

G Additional results

G.1 More recent LMMs

Table 10 extends the VizWiz and OK-VQA comparison of Table 2 to InternVL3.5-8B, Qwen2.5-VL-7B, and Idefics3-8B, under the protocol of Appendix F.3, and each trained method is run three times. STAVE has the best mean on five of the six columns, with 14 to 36 times fewer parameters than LIVE and MimIC and 238 to 306 times fewer than HiFICL, and it exceeds 4-shot ICL by more than 10 points on every column.

Table 10: Comparison on VizWiz and OK-VQA with more recent LMMs.

	InternVL3.5-8B			Qwen2.5-VL-7B			Idefics3-8B		
Method	# Params (M)	VizWiz	OK-VQA	# Params (M)	VizWiz	OK-VQA	# Params (M)	VizWiz	OK-VQA
Zero-shot	–	27.7	19.4	–	48.1	54.5	–	29.9	45.2
4-shot ICL	–	56.5	47.4	–	54.5	43.0	–	37.6	45.9
LIVE	0.15 (×18.00)	$63.54_{\pm 0.71}$	$49.98_{\pm 0.13}$	0.10 (×14.00)	$52.97_{\pm 9.08}$	$55.31_{\pm 0.33}$	0.13 (×16.00)	$46.93_{\pm 1.96}$	$39.25_{\pm 1.03}$
MimIC	0.30 (×36.14)	$66.79_{\pm 1.42}$	$57.36_{\pm 0.42}$	0.20 (×28.11)	**73.44**$_{\pm 0.87}$	$\underline{64.55}_{\pm 0.47}$	0.26 (×32.13)	$\underline{66.71}_{\pm 0.86}$	$\underline{57.22}_{\pm 0.22}$
HiFICL	2.5 (×306.00)	$\underline{68.09}_{\pm 0.41}$	$\underline{58.74}_{\pm 0.17}$	1.7 (×238.00)	$\underline{71.74}_{\pm 0.51}$	$59.81_{\pm 0.53}$	2.2 (×272.00)	$65.60_{\pm 0.77}$	$56.53_{\pm 0.64}$
STAVE	**0.008 (×1.00)**	**68.37**$_{\pm 0.71}$	**58.94**$_{\pm 0.11}$	**0.007 (×1.00)**	$71.34_{\pm 0.24}$	**64.91**$_{\pm 0.29}$	**0.008 (×1.00)**	**68.66**$_{\pm 0.18}$	**58.74**$_{\pm 0.37}$

G.2 VQAv2 answer types

Figure 14 splits the VQAv2 scores of Table 1 by the three official answer types on all 10,000 evaluation questions, using one run of each method. Each point is the accuracy gain of STAVE over LIVE, MimIC, or HiFICL, with a paired bootstrap 95% interval over questions (2,000 resamples). STAVE matches or exceeds every baseline on every answer type on both backbones. The other type holds half of the questions and most open-vocabulary answers, and there the gain is 1.3 to 2.6 points, with an interval that excludes zero against every baseline on both backbones.

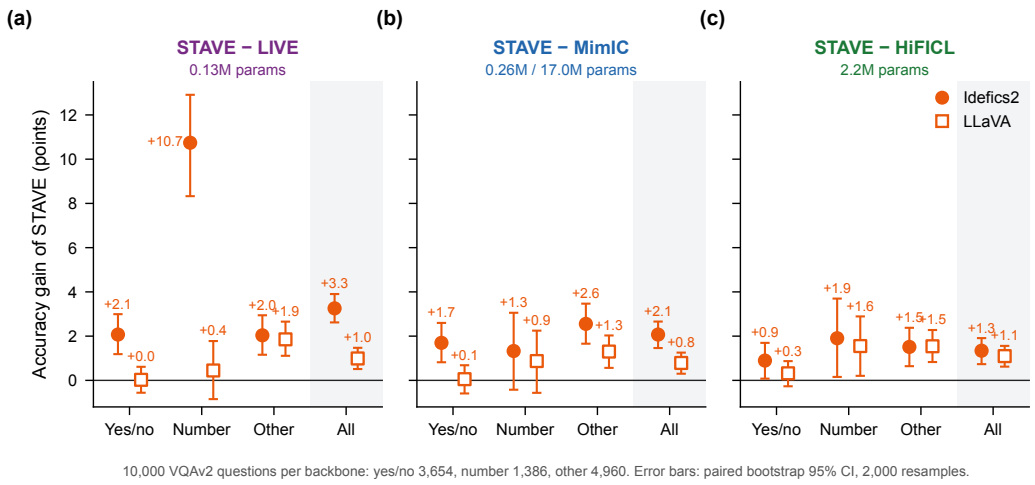


Figure 14: VQAv2 accuracy gain of STAVE over each trained baseline, by answer type.

H Ablation details

H.1 Demos in the prompt on COCO Captioning

Table 1 evaluates every method without demos. Table 11 puts demos back into the prompt at inference and changes nothing else. Every method is read from its own published checkpoint on the first 1,000 images of the COCO Karpathy validation split, and the demos of a query are drawn from the same pool in the same order for every method, so a column compares methods on identical prompts. Columns are the number of demos, and the mismatched column gives four demos whose captions belong to other images. Avg. drop averages the change from the demo-free score over the five demo conditions.

MimIC and HiFICL lose almost all captioning ability on LLaVA as soon as one demo is present, falling from about 135 CIDEr to below 5, and their loss does not shrink as demos are added. On Idefics2, MimIC holds its score within about 3 CIDEr of the demo-free one until the demos are mismatched, and HiFICL does not. Paired STAVE stays within 6 CIDEr of its demo-free score in every condition on both backbones, and within 0.7 CIDEr on Idefics2, including under mismatched demos. Target-only training falls to between 11 and 26 CIDEr on LLaVA, and the paired objective repairs that case.

The released evaluator turns a generation into a caption by cutting it at the first of a list of stop words, and the HiFICL Idefics2 entries at four and at mismatched demos hold 130 and 160 generations of 1,000 that open with a template marker, which places a stop word at the first token and leaves an empty caption. Those two entries are scored after one leading marker is removed, which recovers 93 and 59 captions. The rest are empty generations, so both entries stay far below the demo-free score.

H.2 Object hallucination by training objective

Table 12 scores the Idefics2 COCO captions of the three objectives of Table 4 with the CHAIR metrics of Table 5. Paired training has the highest recall, and its captions name absent objects less often than target-only captions on both CHAIR measures. Source-only captions name absent objects least but recall the fewest objects.

H.3 Training objective on the TV benchmark

The paired objective has two branches. The source branch trains the learned vectors inside a one-demo prompt and the target branch trains them in the demo-free prompt. Table 13 removes one branch at a time. STAVE-source sets the target weight to zero, STAVE-target sets the source weight to zero, and STAVE-paired is the configuration of Table 3. Each entry reuses the configuration that Table 3

Table 11: COCO CIDEr with demos added to the prompt at inference (first 1,000 Karpathy validation images).

Method	0	1	2	4	8	mismatched	Avg. drop
LLaVA-Interleave, COCO							
MimIC	134.6	3.3	3.0	3.4	3.9	4.3	-131.0
HiFICL	138.0	3.4	3.0	3.4	4.5	4.0	-134.4
STAVE (source-only)	134.0	**135.2**	**134.8**	**135.8**	**137.4**	**135.7**	+1.8
STAVE (target-only)	135.2	11.3	21.8	21.6	25.5	15.0	-116.2
STAVE (paired)	**138.8**	133.9	134.7	135.5	134.4	133.0	-4.5
Idefics2, COCO							
MimIC	140.4	139.3	139.1	137.3	137.6	94.5	-10.8
HiFICL	137.7	114.6	129.7	66.5	136.0	21.4	-44.1
STAVE (source-only)	138.1	140.4	141.4	140.8	141.0	139.8	+2.6
STAVE (target-only)	141.5	**141.8**	**142.0**	142.4	**143.4**	138.1	+0.0
STAVE (paired)	**141.8**	141.1	141.6	**142.5**	142.1	**142.1**	+0.0

Table 12: Object hallucination by training objective on Idefics2 COCO.

Objective	CHAIRs ↓	CHAIRi ↓	Recall ↑
Paired	2.62	1.76	**44.81**
Target-only	2.84	1.93	44.47
Source-only	**2.36**	**1.64**	43.08

selected for that backbone and task, with no further tuning, and is evaluated twice on the same test queries: without a demo, and with the same learned vectors placed in a one-demo prompt. Accuracy weights each task by its test queries. The TV bench rows of Table 4 average the three LLMs of Table 13.

Each control holds only in the prompt format it was trained in. STAVE-source loses 2.4 to 25.4 points when the demo is removed, and STAVE-target loses up to 4.3 points when one is added. STAVE-paired stays within half a point of itself in both formats and is above each control in the format that control was not trained for. The one entry where a control is ahead is Pythia 6.9B with a demo, where STAVE-source is 0.2 points above STAVE-paired. The same pattern holds on Pythia 2.8B and Pythia 12B, where STAVE-source loses 4.4 and 24.5 points without a demo. SITE is listed for reference and is read the same way. It loses 0.8 to 1.0 points when a demo is added, a milder form of the STAVE-target pattern, so STAVE-paired is the only entry that holds its accuracy in both formats.

Table 13: Training objective ablation on the TV benchmark (accuracy, %). Best objective in bold.

	Pythia 6.9B			GPT-J 6B			LLaMA 7B		
Method	**w/o demo**	**w/ 1 demo**	**Δ**	**w/o demo**	**w/ 1 demo**	**Δ**	**w/o demo**	**w/ 1 demo**	**Δ**
SITE	90.4	89.4	−1.0	91.6	90.6	−1.0	92.1	91.3	−0.8
STAVE-source	66.6	**92.0**	+25.4	82.8	92.3	+9.5	89.6	92.0	+2.4
STAVE-target	91.3	88.1	−3.2	91.7	87.4	−4.3	91.6	91.2	−0.4
STAVE-paired	**91.6**	91.8	+0.2	**92.3**	**92.4**	+0.1	**92.3**	**92.5**	+0.2

H.4 Structural token groups

Figure 4(a) and Table 14 hold the training configuration fixed and vary the group-to-vector assignment or the updated tokens. Every configuration trains for five epochs at the learning rate and demo count of the Table 1 entry of its model and dataset, and reports its best epoch on the evaluation split, the

checkpoint selection used for that entry. Configuration names list the tokens covered by each learned vector, with vectors separated by a slash. The readout tokens are the final prompt token and the answer cue. The context tokens are all other eligible tokens. The image tokens are the query-image embeddings. In "Text / image", the text tokens are all eligible tokens outside the query image, and in "Readout / text / image", they exclude the readout tokens. "Readout / context" is the configuration used in the main experiments. A configuration marked "only" learns one vector and leaves all other tokens unchanged.

Table 14: Group ablation on COCO (CIDEr) and VQAv2 (accuracy, %). Best: **bold**, second: underlined.

	COCO		VQAv2	
Configuration	Idefics2	LLaVA	Idefics2	LLaVA
One vector				
Context only	132.65	129.57	72.84	75.51
Readout only	132.24	129.85	71.78	75.86
All tokens	131.95	129.18	72.75	75.74
Two vectors				
Readout / context	**134.38**	**131.78**	73.46	**76.12**
Last / rest	133.28	128.98	73.33	75.77
Text / image	133.35	129.12	72.67	75.98
Three vectors				
Readout / text / image	133.79	130.79	**73.62**	75.96
One vector per group				
Per group	133.91	130.19	72.72	75.84

One-vector configurations. "All tokens" ties every eligible group to one vector. "Readout only" learns the readout vector and leaves the context tokens unchanged. "Context only" learns the context vector and leaves the readout tokens unchanged. Excluded groups are zeroed rather than reassigned. The LLaVA prompt has no image delimiters, so its IMAGE-GATE group is empty. Demos, BOS tokens, padding, and answer tokens remain excluded in every configuration.

Two-vector, three-vector, and per-group configurations. "Last / rest" learns one vector at the final prompt token and one vector shared by every other eligible token, so the answer cue joins the context tokens instead of the final token. "Text / image" learns one vector for the text tokens, including IMAGE-GATE, and one vector for the image tokens. "Readout / text / image" keeps the readout vector and splits the context tokens into one vector for the query-image tokens and one vector for the other context tokens. "Per group" is the full group set, with a separate vector for every structural token group. Every configuration recomputes the count normalization on its merged groups, so each learned vector is divided by the square root of the number of tokens it updates in the prompt.

What the comparisons support. Figure 4(a) shows the mean of the two backbones in Table 14, on axes that start above zero. Overall, "Readout / context" is the best configuration. One learned vector is not enough: "All tokens", "Readout only", and "Context only" are below it for every model and dataset, by up to 1.7 points on Idefics2 VQAv2 and up to 2.6 CIDEr on LLaVA COCO. Two-vector partitions that mix readout and context tokens are also below it for every model and dataset, "Text / image" by up to 2.7 CIDEr and "Last / rest" by up to 2.8 CIDEr. More vectors bring no consistent gain. "Per group" is below it for every model and dataset, and "Readout / text / image" is above it only on Idefics2 VQAv2, by 0.16 points, inside the paired-bootstrap interval against "Readout / context", which spans about half a point on VQAv2 (2,000 resamples over validation samples). Appendix I.3 examines what each vector changes in the trained matrix.

H.5 ALLOCATION OF A SHARED UPDATE

Resolution sweep. Figure 15 tests the dilution of Equation 1 by changing only the number of image tokens. From 64 to 4,096 image tokens, one vector shared by all eligible tokens loses 81% of its

first-order decrease on OK-VQA and stays within about 3% of the prediction with zero gradient at the image tokens of Proposition C.2. The readout/context split at the same total norm keeps its effect.

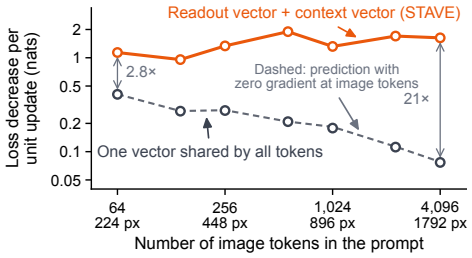


Figure 15: One shared vector against the readout/context split as the image grows (Qwen2.5-VL, OK-VQA).

The figure uses the frozen Qwen2.5-VL-7B backbone of Table 10 with the demo-free OK-VQA prompt of that table. Each image is resized to $s \times s$ pixels with s from 224 to 1792, which gives $(s/28)^2$ image tokens and leaves every text token unchanged. Besides the image, the prompt holds one FIRST token, one IMAGE-GATE token, about nine question tokens, and two readout tokens, ANSWER-CUE and LAST. The same 160 training-pool samples are used at every size, 96 to fit the update directions and 64 to measure the loss. For each configuration, the direction is the negative mean zero-update gradient of its vector over the fitting samples, scaled to unit total norm, so that it changes the input embeddings by one in Frobenius norm. The reported value is the first-order decrease of the answer loss of Equation 5 along this direction, averaged over the held-out samples. The prediction with zero gradient at the image tokens is the value of one vector on the text tokens, multiplied by $\sqrt{N_{\text{text}}/N_1}$, where N_{text} counts the eligible text tokens and N_1 all eligible tokens. The readout/context configuration ties LAST and ANSWER-CUE to one vector and all other eligible tokens, including the image, to a second vector. Two vectors on random token sets with the readout and context counts, under three partition seeds, never exceed one shared vector by more than 0.3%, so the gain comes from separating the readout tokens rather than from the second vector. Updates of norm 0.1 give the same pattern in the actual loss. At 64 image tokens one shared vector lowers it by 0.047 nats and the readout/context split by 0.115 nats, and at 4,096 image tokens by 0.0002 and 0.167 nats. Smaller norms are not informative on this backbone, because bf16 rounding moves the loss of a single sample by up to about 0.1 nats under any perturbation of its embeddings. Table 15 lists the values for OK-VQA and for VizWiz, whose prompt also carries a 39-token instruction, so that the shared vector falls later there. Real photos fall inside this range. The default Qwen2.5-VL processor limits each image to 12,845,056 pixels and rounds its sides to multiples of 28. On 1,500 OK-VQA images drawn at random from the same pool, it gives 234 to 391 image tokens between the 10th and 90th percentiles, with a median of 345. On 1,500 VizWiz images the median is 1,610 and the 90th percentile 2,552.

Native resolution. The LLaVA measurements below use the frozen LLaVA backbone of Table 1 on VQAv2, OK-VQA, and COCO Captioning, with the prompt templates and structural token groups of the main experiments. All prompts are demo-free, the loss is the answer CE of Equation 5 in nats per answer token, and all images come from the training pool, never from an evaluation split. Each configuration learns one or two vectors, which are the rows of W in Equation 2.

Effect of one token. The eligible tokens form three sets: the readout tokens (LAST and ANSWER-CUE), the query-image tokens, and the remaining context tokens, which are all text tokens in the LLaVA prompts. For each set, one vector is tied to its tokens with the count normalization of Equation 2, and all other tokens stay unchanged. On 300 prompts per task, the negative mean gradient of this vector at zero update is estimated on one half and scaled to unit norm, and the first-order loss decrease along this direction is averaged over the other half. Dividing it by the square root of the number of tokens in the set gives the average decrease for a unit shift at one token of the set. A readout token exceeds a query-image token by a factor of about 19,000 on VQAv2 and OK-VQA and about 14,000 on COCO. With the same halves, query-image tokens contribute 0.2% of the first-order decrease of one shared vector on VQAv2 and 0.04% on COCO, although they form 95% and 99% of the eligible tokens.

Equal update size. For each configuration, the update direction is the negative mean zero-update gradient of its vectors over 96 images, scaled to unit total norm. Because the columns of $H(p)$ in Equation 2 are orthonormal when every row of W is nonempty, vectors of total norm ρ change the input embeddings by exactly ρ in Frobenius norm. The loss is measured on 64 other images at $\rho = 0.01$, a value fixed before the measurement, so no optimizer or learning rate is involved. The random configurations assign the eligible tokens of each prompt at random to two vectors with the

Table 15: First-order loss decrease per unit update as the image grows (Qwen2.5-VL).

Image tok.	Size (px)	One shared	Predicted	Readout / context	Ratio	Image share (%)
OK-VQA						
64	224	0.408	0.409	1.137	2.8	1.95
144	336	0.270	0.271	0.956	3.5	1.80
256	448	0.274	0.276	1.337	4.9	1.41
576	672	0.209	0.209	1.897	9.1	1.27
1,024	896	0.178	0.184	1.320	7.4	1.35
2,304	1344	0.112	0.113	1.702	15.2	0.68
4,096	1792	0.077	0.078	1.630	21.2	0.74
VizWiz						
64	224	1.082	1.089	1.397	1.3	0.01
144	336	0.936	0.947	1.399	1.5	0.01
256	448	0.856	0.867	1.501	1.8	0.02
576	672	0.806	0.820	2.125	2.6	0.04
1,024	896	0.877	0.897	3.352	3.8	0.04
2,304	1344	0.550	0.564	3.100	5.6	0.04
4,096	1792	0.443	0.453	3.213	7.3	0.06

One shared: one vector on all eligible tokens. Predicted: the same vector with zero gradient at the image tokens. Readout / context: one vector on LAST and ANSWER-CUE and one on all other eligible tokens, with the same total norm. Ratio: readout / context over one shared. Image share: part of the one-shared value contributed by image tokens.

readout and context counts, under three partition seeds. Table 16 reports the resulting loss decreases. At the same update norm, the readout/context split lowers the held-out loss 3.4 to 7.5 times as much as one shared vector, while random splits with the same counts stay within 5% of it.

Table 16: Held-out loss decrease at update norm 0.01 on LLaVA. Ratios are relative to one shared vector.

Task	One shared vector (nats)	Readout / context	Random splits
VQAv2	0.113	7.52×	0.99–1.00×
OK-VQA	0.094	7.31×	1.01–1.03×
COCO	0.028	3.43×	0.95–1.01×

H.6 COUNT NORMALIZATION

Table 17 varies only the scale applied to each learned vector and keeps every other setting of the Table 1 entry of the model and dataset. Write n_k for the number of tokens of structural token group k in the prompt and N_r for the merged count of the groups that share vector r. "Vector count" divides a vector by $\sqrt{N_r}$ and is the normalization used in all other experiments. "Group count" divides each token by the square root of its own group count, so a vector that ties several groups is not renormalized after merging. "Vector mean" divides by N_r. "None" applies no scale. Every configuration uses the two-vector assignment, paired supervision, five epochs, and the learning rate and demo count of the model and dataset, and reports its best epoch on the evaluation split.

What the comparisons support. Vector count is the best entry for every model and dataset. Group count is below vector count for every model and dataset, by 1.10 points on Idefics2 VQAv2 and by 2.68 CIDEr on LLaVA COCO, which supports computing the scale on merged groups rather than on the original groups. Removing the scale is the most damaging choice on VQAv2 for both models, by 5.85 points on Idefics2 and 1.16 points on LLaVA. The readout vector covers the final token and the answer cue, while the context vector covers all image tokens, 64 on Idefics2 and 729 on LLaVA, so an unscaled update injects far more energy through the context vector than through the readout vector at the same per-coordinate step.

Table 17: Count normalization ablation on VQAv2 (accuracy, %) and COCO (CIDEr).

		VQAv2		COCO	
Normalization	**Scale**	Idefics2	LLaVA	Idefics2	LLaVA
Vector count	$1/\sqrt{N_r}$	**73.46**	**76.12**	**134.38**	**131.78**
Group count	$1/\sqrt{n_k}$	<u>72.36</u>	75.73	133.14	129.10
Vector mean	$1/N_r$	72.10	<u>75.87</u>	<u>133.80</u>	130.09
None	1	67.61	74.96	132.98	<u>130.23</u>

H.7 Injection depth

The depth ablation replaces the embedding injection by an update after a decoder block while retaining the readout vector, context vector, group assignments, and prompt-only mask. The Qwen-VL depths use the VizWiz and OK-VQA entries in Table 2. Both backbones have 32 decoder blocks. A displayed depth $k/8$ means injection after block $4k$ for $k = 1, \ldots, 8$. Depth zero denotes the input embeddings.

Raw scores and retained gain. Figure 16 contains all four raw curves. VQA scores are accuracies in percentages. Captioning uses CIDEr (Vedantam et al., 2015), on the scale reported in Table 1. The retained gain in Figure 4(b) is computed as

$$\mathrm{RetainedGain}(\ell) = \frac{s_\ell - s_{\mathrm{zero}}}{s_{\mathrm{emb}} - s_{\mathrm{zero}}} \times 100\%. \tag{H.1}$$

Here s_ℓ is the score at injection depth ℓ, s_{zero} is the unadapted zero-shot score, and s_{emb} is the embedding endpoint. A negative retained gain means the depth-specific score is below zero-shot.

Embedding injection is best or tied for best in every curve. Later Idefics2 scores degrade sharply, while Qwen-VL retains more of its embedding gain on VizWiz than on OK-VQA. At depth one, after the last block, Proposition 2.1 explains why the context vector receives zero gradient and why later teacher-forced stopping decisions are not directly controllable.

I Analysis of the learned vectors

I.1 Promoted tokens

The learned vectors are added to input embeddings and act through the frozen decoder, so Table 18 describes them by their effect on the next-token distribution. For a category with demo-free test prompts $p_1, \ldots, p_n$, the promotion of a vocabulary token u is

$$\Delta(u) = \frac{1}{n} \sum_{i=1}^{n} \Big[p_\theta\big(u \mid \widetilde{E}(p_i, W)\big) - p_\theta\big(u \mid E(p_i)\big) \Big], \tag{I.1}$$

where both distributions are taken at the final prompt token. Spellings with and without a leading space are merged, and whitespace tokens are omitted. Each cell lists the four tokens with the largest $\Delta(u)$ over 30 to 476 test prompts. A token is black when, lowercased and stripped of punctuation, it begins at least one reference answer of the category, and gray otherwise.

The checkpoints are the seed-41 runs of Tables 1 and 3 and the reported entries of Table 2. Prompts are rendered as in the corresponding evaluation. VQAv2 and VizWiz categories are assigned from the question wording, the VizWiz unanswerable column uses the answerability label of the dataset, and the OK-VQA columns use its annotated knowledge categories. No category depends on a model prediction.

The same two vectors are added to every prompt of a task, yet the promoted tokens change with the category and are the answers it asks for. Over the 68 categories of the eight VQA checkpoints, excluding the catch-all other category, 95.9% of the five most promoted tokens begin a reference answer. The table shows a subset of these categories. Adding demos without the learned vectors, as in the source prompt p_i^{src} of paired training, gives 91.8% on the same categories, with four demos on VQAv2 and one on the other tasks. On the 90 TV benchmark checkpoints the two rates are 92.2% and 82.2%.

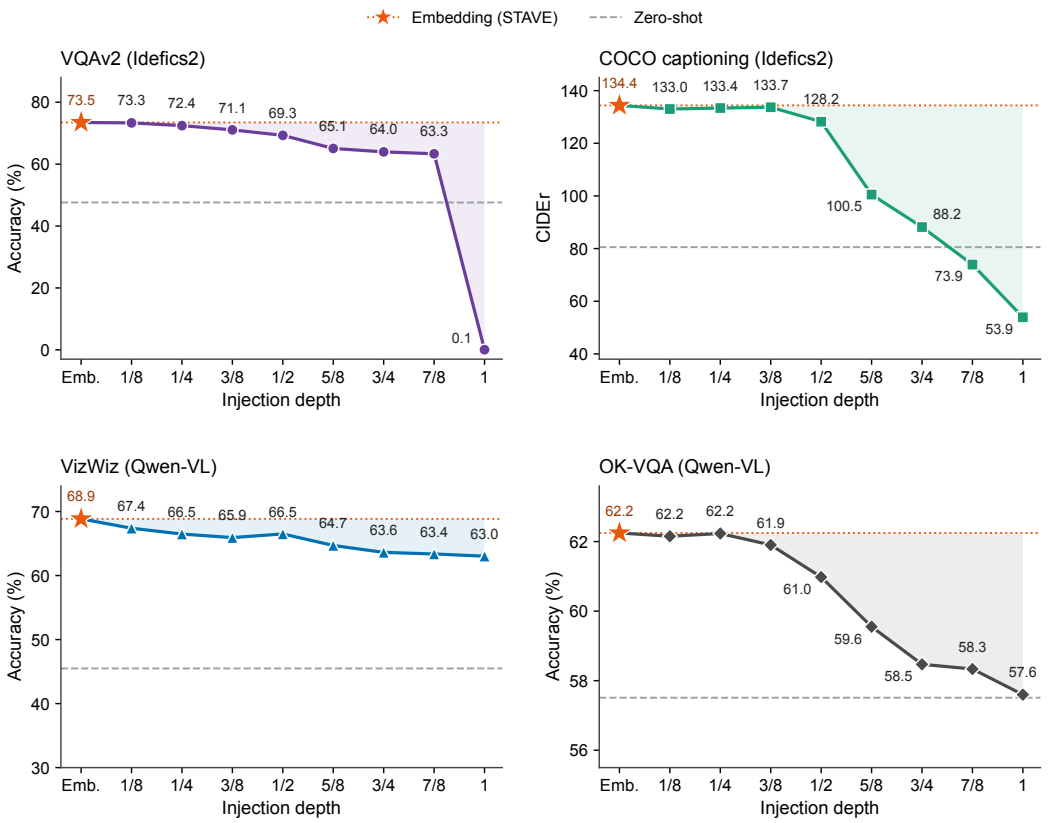


Figure 16: Scores at each injection depth on Idefics2 (top) and Qwen-VL (bottom).

Several entries are the first subword of a longer answer, for example `gir` for giraffe, `sam` and `son` for Samsung and Sony, `vit` for vitamin, and `un` for unanswerable. LLaVA answers without a leading space, and its tokenizer then splits more answers, as in `base`, `ten`, and `soc` for baseball, tennis, and soccer.

I.2 States and decoded tokens

The state of a prompt is the output of the last decoder block at the final prompt token, computed without the update and with the STAVE update.

On LLaVA VQAv2 we leave out the catch-all other category, which keeps 1,585 validation questions. Without the update, the next token is `<|im_start|>` for 99.1% of them. In a principal component analysis (PCA) of the states of both conditions, the first component explains 50% of the variance and has cosine 0.99 with the difference between the two condition means, so Figure 5(a) shows the second and third components. Along them the no-update states stay in one cluster, and the STAVE states leave it in a different direction for each answer type. Panel (b) refits the PCA on the STAVE states of the six object categories. The no-update states still encode the question type: in the full space, a nearest-centroid classifier recovers it with 99.2% accuracy under five-fold cross-validation.

Figure 17(a) decodes the state after each block with the final normalization layer and the output projection of the frozen model, for three questions of Figure 5(b). We call an answer found late and lost when the reference first token, with and without a leading space, is the top decoded token after some block in the last 40% of the decoder but not at the output. Without the update this holds for 10% to 58% of the test prompts of the seven LMM checkpoints in panel (e) and for 15% of the TV benchmark prompts. STAVE outputs that token for 73% to 95% of these prompts. For these prompts the answer is present inside the frozen decoder, and the learned vectors keep it through the last blocks.

Table 18: Tokens most promoted by STAVE at the final prompt token. Gray tokens begin no reference answer of the column.

(a) VQAv2, Table 1 checkpoints

Model	Color	Sport	Animal
Idefics2-8B-base	blue, white, black, brown	tennis, baseball, soccer, sk	cat, horse, gir, z
LLaVA	white, black, blue, red	base, ten, soc, sk	cat, g, dog, horse
Model	**Food**	**Vehicle**	**Time or weather**
Idefics2-8B-base	pizza, hot, sandwich, cake	bus, motor, truck, train	afternoon, winter, morning, sun
LLaVA	pizza, hot, grass, sand	bus, bike, motor, pickup	winter, day, summer, s

(b) OK-VQA, Table 1 and Table 2 checkpoints

Model	Vehicles and transport	Brands and products	Objects and material
Idefics2-8B-base	stop, double, fighter, dies	sam, son, apple, laptop	t, clock, wood, plastic
Idefics2-8B-instruct	stop, double, vol, c	apple, son, sam, computer	t, bear, wood, metal
Qwen-VL-7B	bo, schw, east, stop	apple, c, schw, monitor	sit, cut, suit, leather
Model	**Sports and recreation**	**Cooking and food**	**Weather and climate**
Idefics2-8B-base	ski, surf, tennis, sk	b, vit, c, cake	sun, rain, winter, cold
Idefics2-8B-instruct	ski, tennis, sk, surf	b, c, vit, f	rain, sun, cloud, cold
Qwen-VL-7B	surf, fall, ser, run	carrot, lunch, spinach, k	dusk, str, rain, star

(c) VizWiz, Table 2 checkpoints

Model	Unanswerable	Identify the object	Color
Idefics2-8B-instruct	un, no, yes, windows	un, laptop, computer, tv	white, grey, black, blue
Qwen-VL-7B	un, priority, pink, unknown	un, computer, sweet, mac	un, white, black, blue

(d) TV benchmark, Table 3 checkpoints

Model	Location to continent	Location to religion	Person to language
Pythia 6.9B	An, Europe, Asia, Af	Muslim, Christian, Jewish, Chinese	English, French, Italian, Spanish
Pythia 12B	An, Europe, Asia, Af	Muslim, Christian, Jewish, Jew	English, French, Italian, Spanish
GPT-J 6B	Ant, Europe, Asia, Af	Muslim, Christian, Jewish, Chinese	English, French, Spanish, Italian
Model	**Country to capital**	**Present to gerund**	**English to French**
Pythia 6.9B	Paris, Val, Pr, Ber	staying, blessing, begging, providing	par, comme, m, toujours
Pythia 12B	Paris, Ber, Pr, Val	rushing, staying, handing, guiding	comme, par, image, toujours
GPT-J 6B	Paris, Ber, Pr, Val	singing, making, banning, baking	dire, par, personnel, pour

Panels (b) to (d) project the states of all 18 TV benchmark tasks onto the two leading principal components of the STAVE states, and the no-update states onto the same axes. Several tasks share test prompts exactly, which we detect as prompts with identical no-update states. A shared prompt has one no-update state and one next token, while STAVE moves it with the vectors of each task. List first and list last share all 200 test prompts. With the vectors of each task, STAVE answers both tasks correctly on all of them on every LLM, against at most 2.5% for the frozen model. Present to gerund and present to past share 9 to 29 prompts per LLM, with 100% against 0%. Panel (f) counts every pair of tasks with a shared test prompt, 265 to 277 per LLM. STAVE answers both tasks correctly for 92% to 94% of them and the frozen model for at most 2.6%. On GPT-J 6B and Pythia 12B the cards show next letter and to uppercase in place of the translation pair.

Figure 18 applies the same decoding to the first answer token on the Idefics2 and LLaVA VQAv2 and COCO checkpoints of Table 1, with 128 held-out demo-free prompts per model and dataset and a prompt with four demos and no update as a reference. Without the update, on three of the four model and dataset pairs the correct token is the top decoded token before the last block for 33% to 48% of the prompts and is lost at the output. With both vectors it stays top-1 through the output for 64.8% to 70.3% of the prompts, against 55.5% to 64.1% with four demos. In panel (b), the change that the vectors cause in the state stays between 0.7 and 1.3 times the norm of the state from block 16 to the output, so the frozen blocks do not suppress the update. In panel (c), the attention from the final token to the query image, averaged over layers, rises with the vectors for every model and dataset.

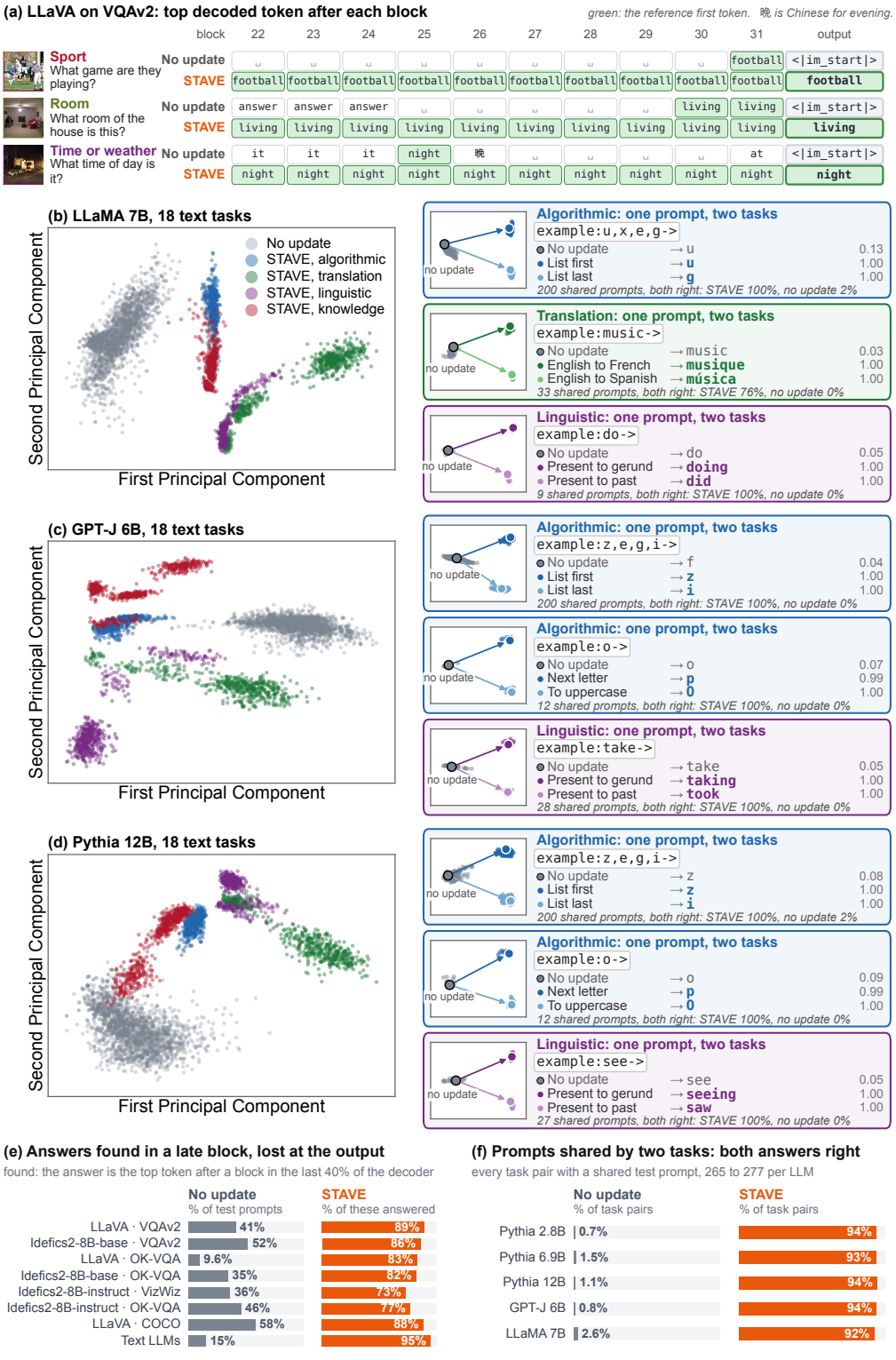


Figure 17: Final-token analysis beyond Figure 5. (a) Top decoded token after each block. (b) to (d) TV benchmark states. (e) Answers found late and lost. (f) Shared prompts answered correctly for both tasks.

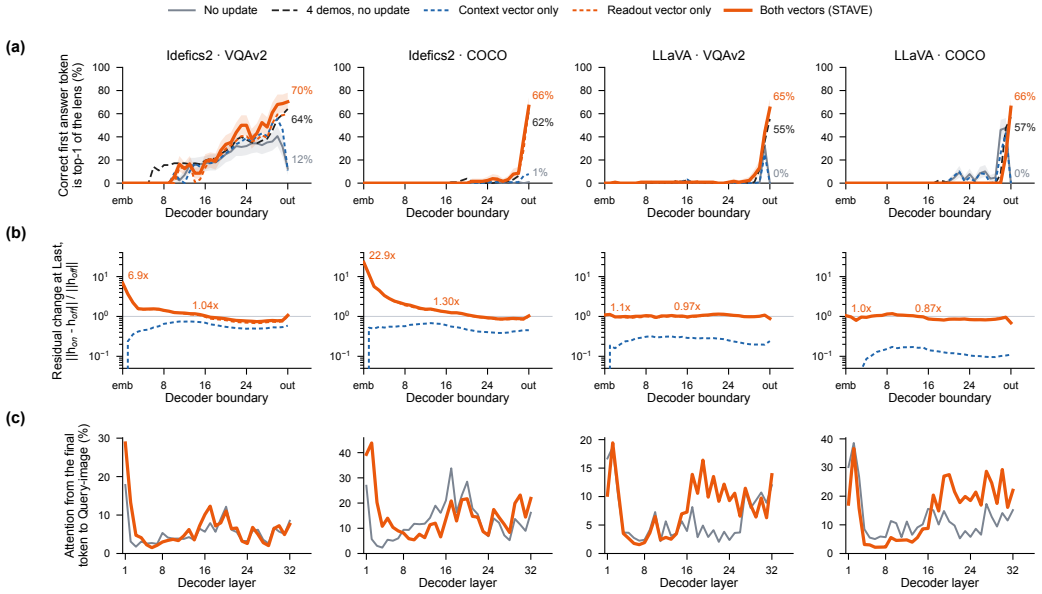


Figure 18: Layerwise effect of the learned vectors at the final prompt token. (a) Correct first token decoded as top-1. (b) Relative change of the state. (c) Attention to the query image.

I.3 What each learned vector changes

Figures 6 and 19 apply the deployed vectors of the Idefics2 entries of Table 1 to held-out demo-free prompts with no update, with one of the two vectors, and with both vectors. No vector is retrained, so the four conditions differ only in which vectors are added. Figure 6(a) and Figure 19(a) read the first answer token on 1,000 prompts per task, for VQAv2 and OK-VQA. A token counts as correct when it matches the answer of one annotator, the reference used for training, so these rates are lower than the VQA scores of Table 1, which give full credit to any answer given by at least three annotators. VQAv2 number questions are left out because every number answer begins with the same token on this backbone. The left group of bars asks whether the correct token is the most probable among the first tokens of the task's answers, and the right group whether it is the generated token.

On VQAv2 and OK-VQA, the context vector alone makes the correct answer the most probable task answer for 10 and 14 more percentage points of the prompts, yet it is generated for only 0.3% and 10.5% of them, because without the readout vector the most probable next token is usually not an answer. The readout vector alone makes the model answer. Both vectors give the highest rate on both measures, so the improvement that the context vector brings to the choice among answers reaches the generated answer mostly when the readout vector is present. In Figure 6(b), for each prompt, let Δ be a change of the state at the final prompt token. The figure reports the share $\|\bar{\Delta}\|_2^2 / \overline{\|\Delta\|_2^2}$ that is common to all prompts, where the bars denote means over prompts.

Figure 19(b) to (d) show what the prompt-specific change of the context vector does. On VQAv2 yes/no questions whose reference is no, the readout vector alone still answers yes for 45% of them, and adding the context vector lowers this rate to 26% while the rate on questions whose reference is yes stays near 80%. In panel (c), the readout vector is kept at every layer, while the tokens after the context see the change that the context vector makes to the context states only inside one block of four decoder layers. Layers 12 to 15 alone recover 94% of the yes/no accuracy gain and layers 16 to 19 alone 65%, while layers 20 to 31 recover almost none. In layers 12 to 15 the context vector also raises the attention from the final prompt token to the query image the most. Panel (d) replaces the query image by the image of another evaluation sample and keeps the question and the reference answer. On all three tasks the answer loss rises more when the context vector is present, by 0.23, 0.16, and 0.06 nats per token, and every paired 95% bootstrap interval excludes zero. The context vector therefore makes the answer depend more on the query image, and the readout vector makes this dependence visible in the generated answer.

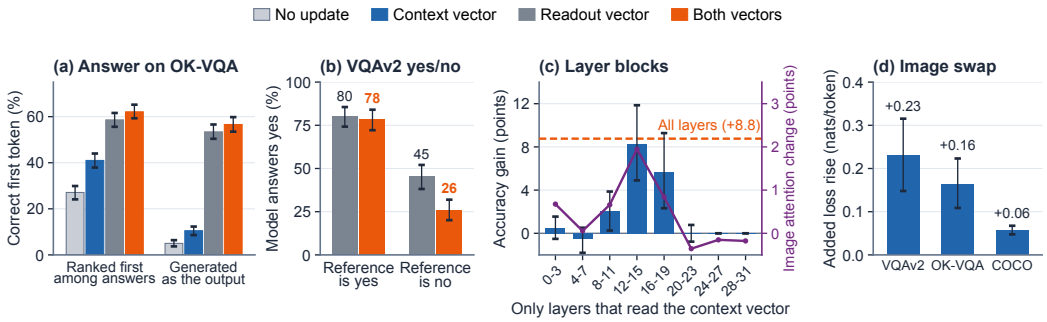


Figure 19: More tests of each learned vector on held-out Idefics2 prompts.

I.4 Answer-prefix probabilities

Figure 20 compares the LLaVA and Idefics2 entries of Table 1 and the Idefics2 and Qwen-VL entries of Table 2. In each of its four rows, STAVE has the largest share of examples above the diagonal and the highest mean answer-prefix probability of the four trained methods. Table 1 rows compare VQAv2 and OK-VQA against PT-Pre, using its 20-token checkpoint. Table 2 rows compare VizWiz and OK-VQA against zero-shot. Each row uses the corresponding table's method checkpoints and 512 fixed seed-41 evaluation examples per dataset, paired across methods. The two OK-VQA protocols are kept separate and assigned different colors.

Each point compares the probability that a continuation begins with an annotated answer. We sum disjoint reference-prefix probabilities after removing duplicate token sequences and extensions of shorter reference prefixes. We include tokenizations with and without a leading space, without an end-of-sequence token or length normalization. Each method retains its reported inference prompt. The axes use a logit scale with probabilities displayed as percentages. Marginal histograms show sample counts, and dashed ellipses summarize the empirical 90% sample spread. Probabilities below 0.0001% are clipped for display only. Each row is read separately, since its reference method and protocol differ.

I.5 Qualitative examples

Figures 21 and 22 compare STAVE with LIVE, MimIC, and HiFICL on demo-free evaluation examples. The VQA examples come from VQAv2, OK-VQA, and VizWiz with the checkpoints of Tables 1 and 2, and each is a question that STAVE answers correctly and all three baselines answer incorrectly. The captions come from the Idefics2 and LLaVA COCO checkpoints of Table 1. STAVE answers questions that depend on text in the image, as for the bill and the toothpaste, and its captions name the object that the reference captions mention where the baselines name a more common one, for example a girl instead of a doll or a book instead of a laptop.

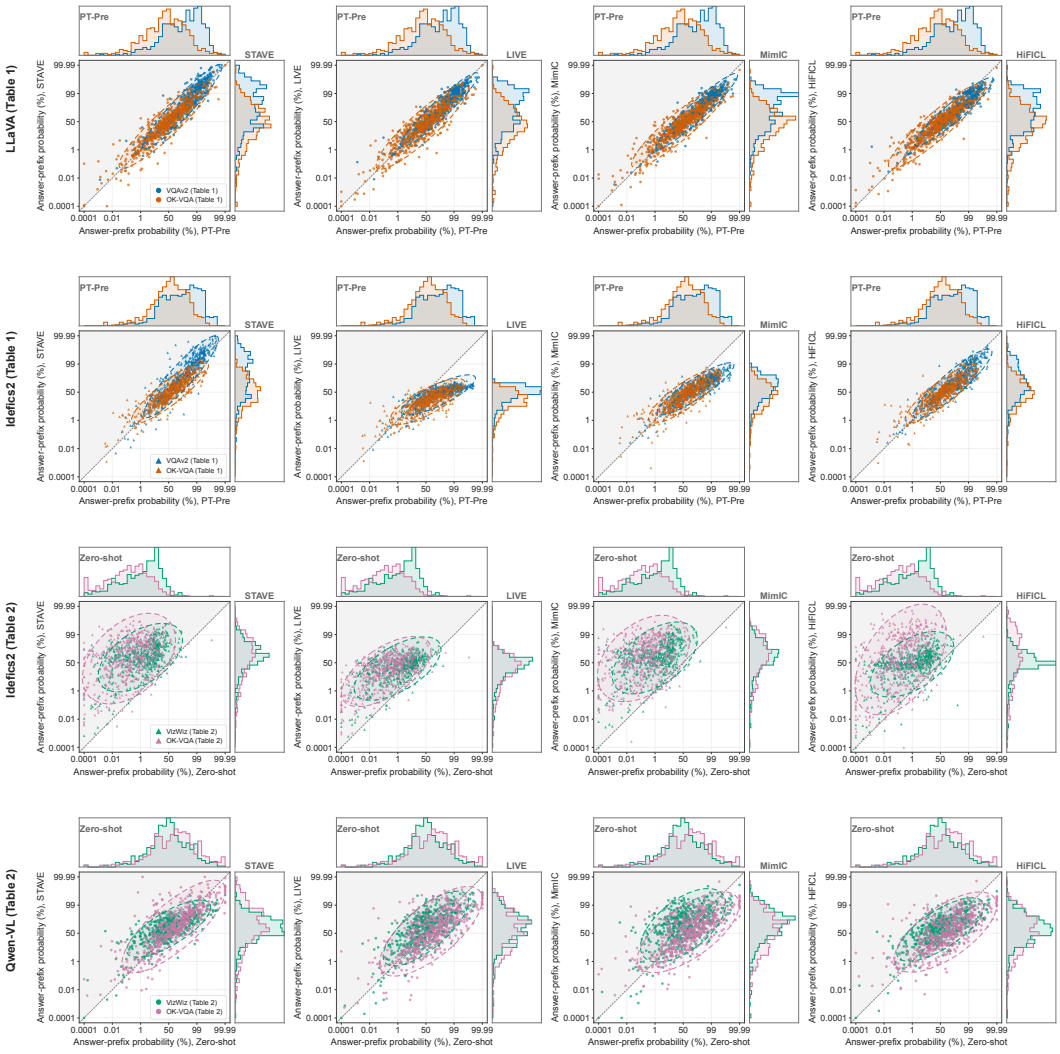


Figure 20: Answer-prefix probabilities against PT-Pre (top two rows) and zero-shot (bottom two rows).

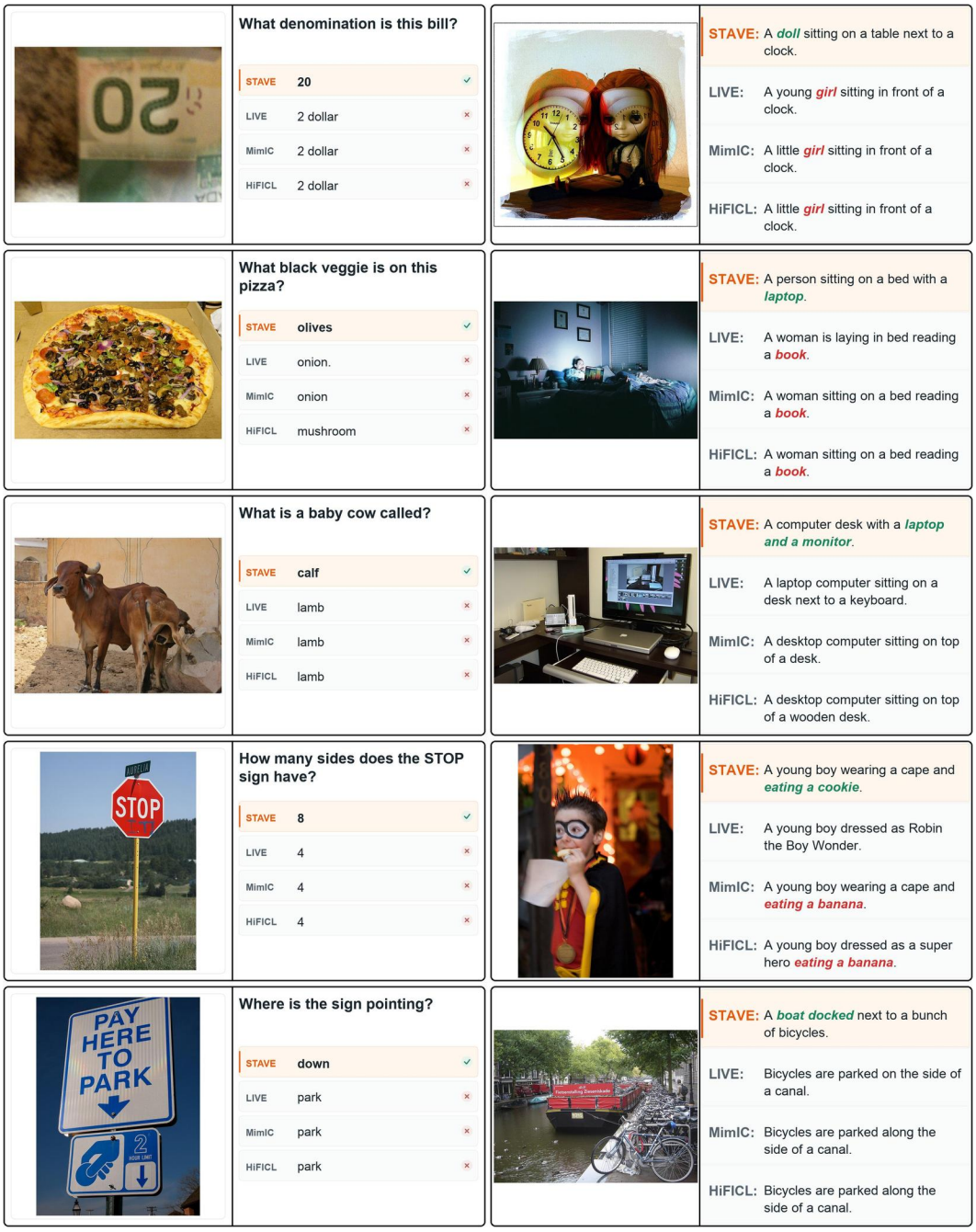


Figure 21: More qualitative examples on VQA (left) and COCO (right), set 1.

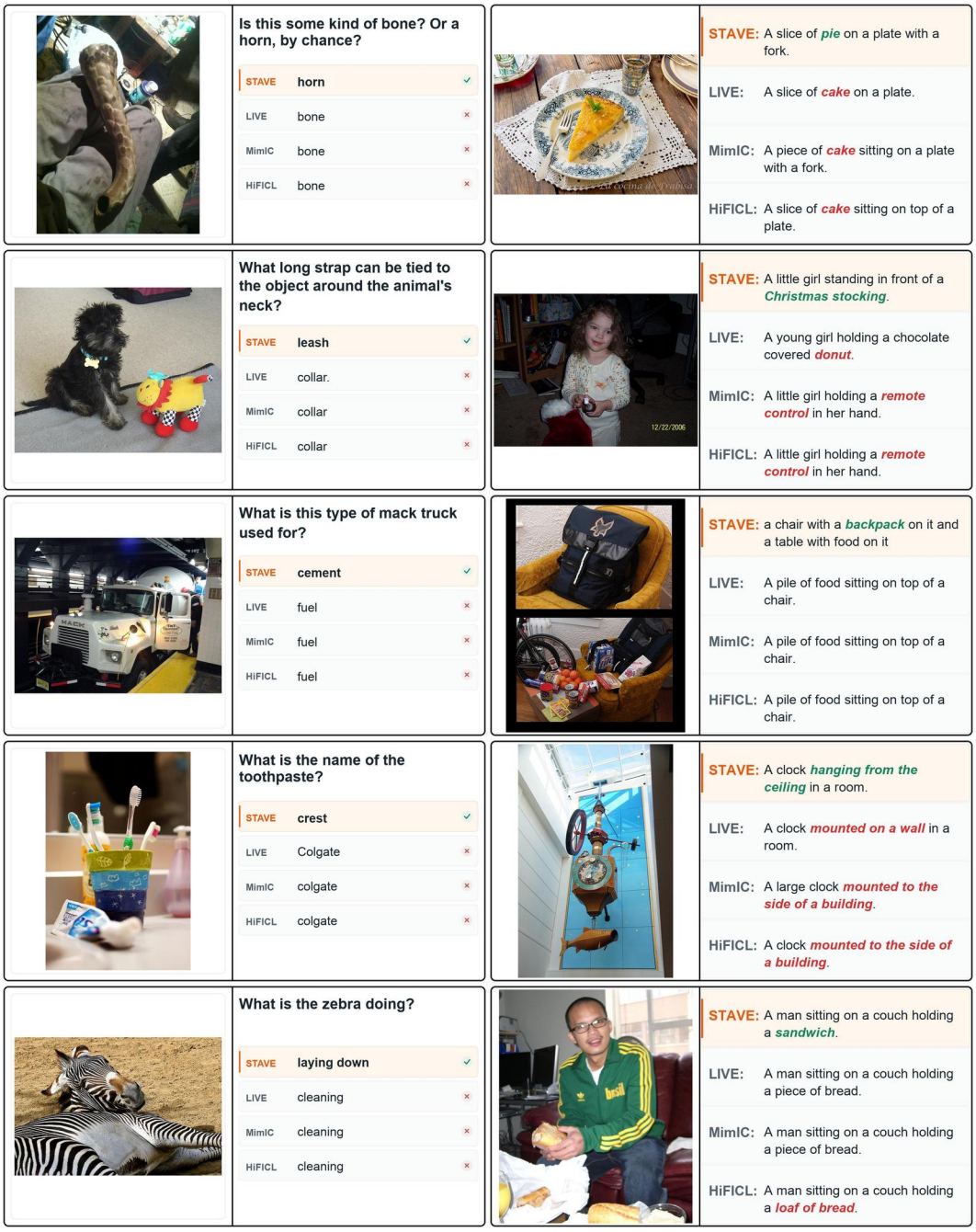


Figure 22: More qualitative examples on VQA (left) and COCO (right), set 2.

J LIMITATIONS AND FUTURE WORK

STAVE runs at zero-shot inference cost, and its two vectors are trained once per task by backpropagation through the frozen decoder, so training memory and time grow with the backbone. STAVE learns each task from labeled training samples. With a tenth of them it loses a few points, less than MimIC and HiFICL on Idefics2 (Figure 7). Training and injection act on the input embeddings, so STAVE requires access to the model weights.

Our LMM experiments cover prompts with one query image and short answers or captions. A natural next step is to apply STAVE to interleaved multi-image and video prompts (Li et al., 2024b; Zhang et al., 2025) and to tasks that require temporal understanding (Ding & Wang, 2025). Another application is long-form reasoning, where steering vectors already change the behavior of thinking language models (Venhoff et al., 2025). More broadly, a task stored in two vectors adds no prompt tokens and only $2d$ parameters, so one deployed backbone can serve many downstream tasks, including on devices with limited memory.

K IMPACT STATEMENT

This work aims to make task adaptation of frozen LLMs and LMMs cheaper. STAVE removes demos from the prompt at inference, which lowers the compute, memory, and energy spent on each query, and it stores a task in two vectors, so many tasks can share one deployed backbone. Because the backbone stays frozen, STAVE inherits the biases and failure modes of the pretrained model, including hallucinated content in generated captions, and the same low cost also applies to tasks trained on harmful data. Deployments should therefore keep the safeguards and usage policies of the underlying model.