ViPLE: Visual Inference with Parallel Latent Encoding

Anonymous authors
Paper under double-blind review

ABSTRACT

Explicit visual reasoning generates intermediate text sequentially, increasing decoding cost. We investigate whether useful visual reasoning context can instead be constructed in parallel for a frozen multimodal backbone. We propose ViPLE (Visual Inference with Parallel Latent Encoding), which extends one-pass latent trajectory encoding with a final cross-attention read of fused image–question states. The latent slots attend to these states before conditioning the frozen decoder. Supervised alignment, on-policy distillation, and reinforcement learning train the interface while keeping the backbone frozen. Across six benchmarks, on Qwen3-VL-2B, 4B, and 8B, ViPLE improves macro accuracy by $\mathbf{+(5.12–7.15)}$ pp over the cheapest measured explicit-reasoning settings while decoding fewer tokens. On GLM-4.1V-9B and Qwen3-VL-32B, the gains are $+2.31$ pp and $+2.50$ pp, respectively, at matched decoding costs obtained by interpolation. The cheapest measured explicit-reasoning settings that reach ViPLE's accuracy require $\mathbf{(1.8–5.8)}\times$ as many decoded tokens. Length-preserving replacement controls show the importance of learned latent content, while retraining without the final cross-attention layer reduces accuracy on mathematical tasks. These results demonstrate that a learned parallel interface can improve the accuracy–decode-cost trade-off of frozen multimodal models.

1 INTRODUCTION

Multimodal large language models increasingly solve visual problems by generating intermediate reasoning steps. Supervised reasoning stages improve the structure of these responses (Xu et al., 2025a), and reinforcement learning supports extended reasoning in recent vision-language backbones (GLM-V Team et al., 2025; Bai et al., 2025). This progress builds on chain-of-thought (CoT) prompting (Wei et al., 2022), with each additional rationale token requiring a sequential decoding step.

Existing approaches change how much reasoning text is generated or retained. Budget forcing limits the thinking segment (Muennighoff et al., 2025), and concise drafts express intermediate steps in fewer words (Xu et al., 2025b). LightThinker compresses earlier reasoning (Zhang et al., 2025a); Look and Think combines reasoning-step compression with visual-token management (Cai et al., 2026); VTC-R1 reuses generated text as optical memory (Wang et al., 2026). For methods that compress already generated reasoning, the initial text generation still incurs sequential decoding cost. This motivates constructing a compact conditioning representation before response decoding begins.

> **Research question.** How can a parallel latent block supply useful visual context to a frozen decoder under a small response budget?

A parallel representation must draw on the visual input and be usable by a frozen decoder. SoftCoT supplies soft thought tokens to a frozen text model (Xu et al., 2025c), and Latent Reasoning Tuning (LRT) constructs a latent trajectory in one encoder pass (Jiang et al., 2026). Heima learns compressed multimodal thinking with an adapted consuming model (Shen et al., 2026), while VaLR generates vision-aligned latent tokens before successive explicit steps (Jeon et al., 2026). We therefore study how to construct a fixed latent block in parallel from fused image–question states and make it usable by a frozen multimodal decoder.

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.

In this paper, we introduce ViPLE (Figure 1), a learned interface between the fused prompt and the frozen decoder. Building on LRT, an encoder constructs 256 slots in parallel. A final cross-attention layer lets the completed slots read the fused prompt directly before projection into the decoder's input space. The final read gives the constructed slots direct access to the fused prompt, complementing the prompt information incorporated during trajectory construction. Supervised alignment, on-policy distillation, and reinforcement learning train the encoder through the frozen backbone.

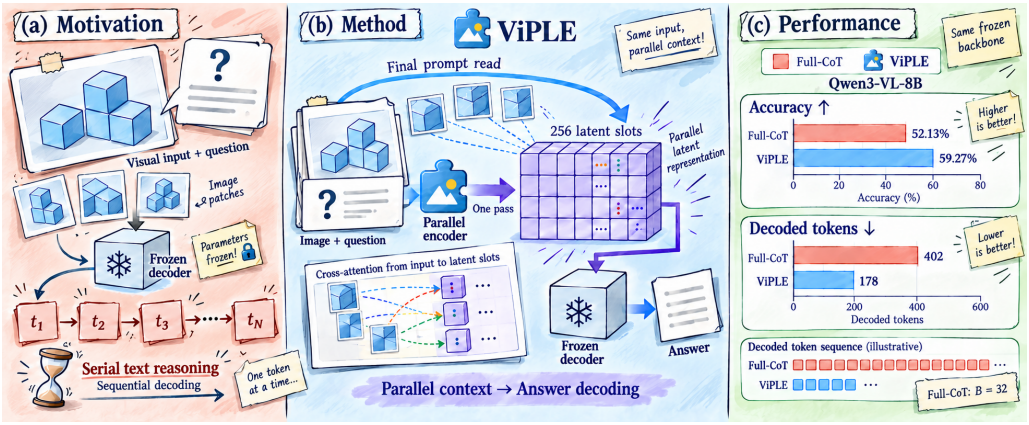


Figure 1: **From serial reasoning to parallel visual conditioning. (a)** Explicit CoT builds reasoning context through sequential token generation. **(b)** ViPLE constructs 256 latent slots in one encoder pass, incorporating a final cross-attention read of fused image–question states. These slots condition a frozen decoder. **(c)** Across six benchmarks on Qwen3-VL-8B, ViPLE achieves 59.27% macro-average accuracy versus 52.13% for Full-CoT with a thinking budget of $B = 32$ tokens, while reducing mean decode length from 402 to 178 tokens.

The contributions of this paper are summarized as follows.

❶ ***Parallel visual conditioning.*** ViPLE extends latent trajectory encoding with a final read of fused prompt states, supplying a fixed conditioning block before text generation.

❷ ***Accuracy at lower decoding cost.*** Five backbones from two families, Qwen3-VL at 2B, 4B, 8B, and 32B, and GLM-4.1V at 9B, improve accuracy at lower measured or interpolated equal decoding costs. The cheapest measured explicit-reasoning settings that reach ViPLE's accuracy require $(\mathbf{1.8}$–$\mathbf{5.8})\times$ as many decoded tokens.

❸ ***Interface and training analysis.*** Length-preserving replacements and retraining without the final cross-attention layer test learned latent content and the added prompt read. Checkpoint analyses show that the largest accuracy gains precede later response shortening.

2 Related Work

Explicit reasoning and budget control. Multimodal CoT introduces written rationales before answers (Zhang et al., 2023), and LLaVA-CoT organises visual reasoning into supervised stages (Xu et al., 2025a). Reinforcement learning further strengthens explicit reasoning in language (DeepSeek-AI et al., 2025) and multimodal models (GLM-V Team et al., 2025). Budget forcing (Muennighoff et al., 2025), concise drafts (Xu et al., 2025b) and token-budget-aware prompting (Han et al., 2024) control the amount of generated text. NoThinking studies when explicit thinking can be omitted (Ma et al., 2025), while work on overthinking examines redundant computation (Sui et al., 2025). These methods motivate comparison with shorter explicit reasoning as well as the largest budget.

Compression and visual grounding during generation. Compression reduces the context retained as reasoning proceeds. LightThinker compresses earlier reasoning (Zhang et al., 2025a); Look and Think combines compression of completed steps with visual-token management (Cai et al., 2026); VTC-R1 converts generated reasoning into reusable optical memory (Wang et al., 2026). VaLR generates vision-aligned latent tokens before each explicit step (Jeon et al., 2026), and Heima learns compressed thinking representations together with the model that consumes them (Shen et al., 2026).

These methods connect intermediate reasoning to compact or visually grounded representations. ViPLE instead constructs a fixed conditioning block before response decoding, using a trainable encoder and a frozen consuming backbone.

Continuous latent reasoning. Distillation transfers explicit CoT into implicit representations (Deng et al., 2023; 2024); Coconut recurrently feeds hidden states back as embeddings (Hao et al., 2024); CODI learns compressed reasoning through self-distillation (Shen et al., 2025); Token Assorted mixes latent and text tokens (Su et al., 2025). SoftCoT supplies soft thoughts to a frozen text model (Xu et al., 2025c). LRT constructs latent trajectories in one encoder pass (Jiang et al., 2026), whereas Latent Thought Flow learns variable-length trajectories under a quality–cost reward (Zou et al., 2026). ViPLE retains LRT's parallel construction and adds a final cross-attention read of fused multimodal prompt states before injection into the frozen decoder. Length-preserving replacement controls test the role of learned content, since additional pause or filler positions can themselves provide computation (Goyal et al., 2024; Pfau et al., 2024).

3 METHOD

ViPLE maps a fused image–question prompt to a fixed latent block and learns that block through a frozen decoder. Figure 2 follows prompt extraction (a), parallel construction and the final prompt read (b), encoder training (c), and response generation (d).

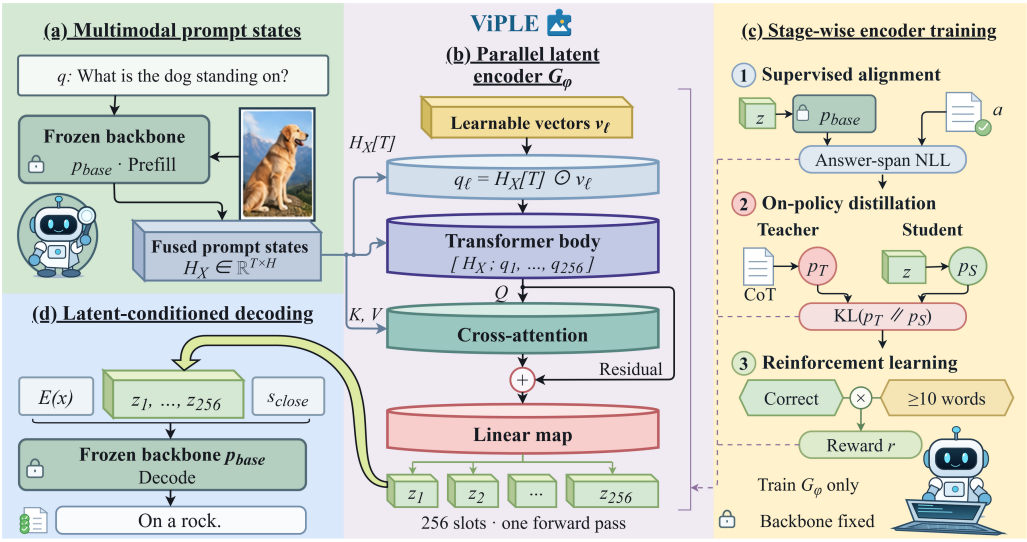


Figure 2: **Parallel visual conditioning for a frozen backbone.** **(a)** Prefill yields fused image–question states H_X. **(b)** The encoder constructs a trajectory whose final cross-attention read queries H_X, producing 256 slots in one pass. **(c)** SFT, on-policy distillation and RL train only G_ϕ. **(d)** The slots condition response decoding. Locks denote frozen parameters; dashed purple paths denote updates. The pictured question is illustrative.

3.1 PARALLEL VISUAL LATENT ENCODING

The latent block $z \in \mathbb{R}^{L\times H}$ contains $L = 256$ slots at the backbone's hidden width H. All trainable parameters belong to the encoder G_ϕ; the multimodal backbone p_{base} remains frozen.

The backbone first prefills the multimodal prompt x, producing last-layer states $H_X \in \mathbb{R}^{T\times H}$. These states already fuse the visual and textual inputs. In Figure 2(b), each learnable trajectory vector v_ℓ is conditioned on the last prompt state by $q_\ell = H_X[T] \odot v_\ell$.

The encoder processes $[H_X; q_1, \ldots, q_L]$ with a Qwen3-Embedding-0.6B transformer body whose token embedding has been removed (Zhang et al., 2025b). Its last L output states form the trajectory and the cross-attention queries Q, following the construction of Jiang et al. (2026).

Final prompt read. Once the trajectory is constructed, a cross-attention layer lets its slots query the full fused prompt directly: Q supplies the queries and H_X supplies the keys and values. A zero-initialised output projection makes this residual contribution initially zero. The learned contribution is added to Q, then a linear map produces the block at the backbone width. The encoder has fewer than one billion parameters. All slots are computed in one forward pass, with a fixed output length independent of an explicit rationale. Section 5.3 tests this added read while retaining the encoder's upstream access to H_X.

As shown in Figure 2(d), z enters the chat template's `<think>` region at the position of the written rationale:

$$\big[\, E(x)\ ;\ z_1, \ldots, z_L\ ;\ s_{\text{close}}\ ;\ E(a_{<t}) \,\big], \qquad z = G_\phi(H_X), \tag{1}$$

where $E(\cdot)$ is the input embedding and $a_{<t}$ is the decoded response history. The suffix s_{close} closes the reasoning region when supplied; an empty suffix allows the model to continue within it. Section 3.3 specifies the main evaluation's thinking and answer stages. All text generated after the block counts as sequential decoding.

3.2 Training the conditioning encoder

The three stages in Figure 2(c) make the block usable by the frozen decoder, transfer the teacher's answer distribution and refine response behaviour through rewards. Each stage starts from the previous checkpoint and updates only G_ϕ. All backbones use three supervised epochs, 150 distillation steps and 60 reinforcement-learning steps.

Stage 1: supervised alignment. An answer-span language-modelling objective trains the encoder to produce states the frozen backbone can use:

$$\mathcal{L}_{\text{SFT}} = -\,\mathbb{E}_{(x,a)\sim\mathcal{D}}\Big[\log p_{\text{base}}\big(a \mid [\,E(x); z; a_{<t}\,]\big)\Big], \qquad z = G_\phi(H_X). \tag{2}$$

Gradients reach G_ϕ through z while the backbone parameters remain fixed.

Stage 2: on-policy distillation. Distillation aligns the latent-conditioned response distribution with that of an explicit-rationale teacher. Both paths use the same frozen backbone: the teacher receives a written chain of thought, and the student receives z. For each question the student samples four candidate responses. At answer position t, the teacher's prefix_t contains the image–question prompt, its recorded rationale and the student's preceding tokens $a_{<t}$. The student uses the same sampled history with z in place of the rationale. A forward KL loss compares their next-token distributions over the answer span:

$$\mathcal{L}_{\text{OPD}} = \operatorname*{masked\,mean}_{t\in\text{answer}}\Big[\,\mathrm{KL}\big(p_{\text{teacher}}(\cdot \mid \text{prefix}_t)\,\big\|\,p_{\text{student}}(\cdot \mid [\,E(x); z; a_{<t}\,])\big)\Big]. \tag{3}$$

Evaluating the loss on student-generated continuations trains the latent block on histories the student actually visits (Agarwal et al., 2024; Gu et al., 2024).

Stage 3: reinforcement learning with a response-length gate. Distillation leaves many responses of at most four tokens (Section 5.2). To encourage correct answers beyond this terse response pattern, the final stage adds a minimum word count to the correctness reward:

$$r_i = \mathbf{1}\big[\,\text{grade}(\text{gold}, y_i)\,\big] \times \mathbf{1}\big[\,\text{words}(y_i) \geq m_{\min}\,\big], \tag{4}$$

where $m_{\min} = 10$. Correct answers below this threshold receive no reward. The gate defines a training constraint; Section 5.2 measures the resulting accuracy and decode length, including their nonmonotonic training trajectory. For each question, $K = 8$ independent decodes share the same z. Their group-relative advantage is $A_i = (r_i - \bar{r})/(\bar{r} + \varepsilon)$, where $\bar{r}$ is the group mean reward. Groups whose rewards are all zero or all one receive zero advantage. The clipped policy-gradient objective (Schulman et al., 2017; Shao et al., 2024) is regularised by a token-level k_3 KL estimate against a frozen copy of the Stage 2 checkpoint:

$$\mathcal{L} = -\,\mathbb{E}_i\Big[\min\big(\rho_i A_i,\ \mathrm{clip}(\rho_i, 1-\epsilon, 1+\epsilon)\, A_i\big)\Big] \;+\; \beta_{\text{KL}}\,\text{masked mean}\big[\exp(d) - d - 1\big], \tag{5}$$

where ρ_i is the importance ratio and $d = \log p_{\text{ref}} - \log p_\theta$. Appendix A lists the shared optimiser, sampling and loss settings.

3.3 Decoding and token accounting

The backbone prefills the prompt, the encoder constructs z, and the backbone generates text conditioned on the block. In the main evaluation, thinking after z ends at the reasoning-close token or a cap of $B = 512$. The reasoning region is closed before answer generation. All five backbones share the sampling settings in Appendix A. Native diagnostics use the model's own termination.

Our cost measure is *live decode length*: the number of sequentially generated tokens per question, including all reasoning and answer tokens after latent insertion. The 256 slots enter through prefill and add no sequential decoding steps. Encoder and backbone-prefill work are outside this metric, so it measures decoding effort rather than end-to-end latency or total FLOPs. The main cost mean excludes immediate zero-token terminations, while accuracy includes every response; Appendix B gives the aggregation rules.

4 Experiments

4.1 Setup

Backbones and data. We evaluate Qwen3-VL-2B, 4B, 8B and 32B-Thinking, and GLM-4.1V-9B-Thinking (Bai et al., 2025; GLM-V Team et al., 2025). Each backbone remains frozen and uses the same training schedule. Training uses 8,056 image–question–answer examples; on-policy distillation uses the subset with an available backbone-specific written rationale.

Benchmarks and aggregation. The in-domain split contains WeMath, MathVista, MathVision and MathVerse (Qiao et al., 2025; Lu et al., 2024; Wang et al., 2024; Zhang et al., 2024). SEED-Bench and MMMU-Pro form the out-of-domain split (Li et al., 2024; Yue et al., 2024). We evaluate 500 examples for each of five benchmarks and all 304 examples in the selected MathVision split, for 2,804 questions per run.

ALL is the equally weighted six-benchmark macro average. Main results average pass@1 over four decoding runs at fixed trained checkpoints; standard errors describe variation across these runs. Underlines identify the closest Full-CoT accuracy within each row, including ties; group headings give ViPLE's ALL standard error.

Budget comparisons. The explicit baseline uses the same frozen backbone with thinking budgets $B \in \{32, 64, 128, 256, 512, 1024, 8192\}$ and an answer cap of 2048 tokens. ViPLE uses a force-close budget of $B = 512$ and the shared sampling settings in Appendix A.

If the explicit baseline reaches the answer cap or lacks a complete answer, its fallback requests one short continuation. ViPLE uses direct force-close termination (Appendix B). Reference comparisons interpolate baseline accuracy at ViPLE's measured cost when baseline points bracket it in every run and in the aggregate. Otherwise, they use the cheapest measured baseline (Appendix B). Figure 3(b) marks observed-point comparisons with an asterisk; unmarked entries are interpolated gains.

4.2 Accuracy at measured decoding costs

① **Better accuracy at lower measured or equal interpolated costs.** Accuracy differences are reported in percentage points (pp). The five backbones achieve favourable accuracy–decode-cost operating points (Table 1; Figure 3). For Qwen 2B, 4B and 8B, it uses fewer tokens than the cheapest measured explicit setting while gaining $\mathbf{+5.12}$ pp, $\mathbf{+5.77}$ pp and $\mathbf{+7.15}$ pp, respectively. For GLM-9B and Qwen 32B, the interpolated equal-cost gains are $\mathbf{+2.31}$ pp and $\mathbf{+2.50}$ pp.

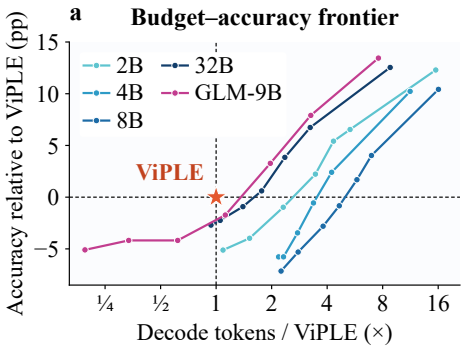
Across the 35 backbone–budget comparisons, 15 explicit settings use more tokens and score lower; none is both cheaper and more accurate than ViPLE on the same backbone. The cheapest measured settings reaching its accuracy use $\mathbf{(1.8–5.8)}\times$ as many decoded tokens. These are observed settings with slightly higher accuracy, so the ratios give the cost of reaching ViPLE's accuracy on the measured budget ladder.

Table 1: **Accuracy and decoding cost.** Black: pass@1 (%, ↑); blue: mean tokens (↓). Underlines mark the closest Full-CoT accuracy to ViPLE per row, including ties.

	ViPLE	Full-CoT thinking budget B													
Benchmark	$B = 512$	32		64		128		256		512		1024		8192	
Qwen3-VL-2B															*ViPLE macro SE:* 0.33 pp
WeMath	49.20	338	44.35	464	47.90	630	51.35	953	56.10	1307	59.80	1549	61.50	1933	70.00 5309
MathVista	51.55	227	45.15	235	46.15	321	50.25	437	55.75	694	60.45	907	61.40	1147	66.15 2968
MathVision	21.22	471	19.33	428	18.83	651	22.12	1062	23.03	1614	23.19	1948	26.32	2434	33.22 7987
MathVerse	35.65	381	30.05	492	29.90	657	33.00	930	38.90	1360	44.70	1679	46.10	1972	55.00 5111
SEED-Bench	70.15	69	63.40	41	66.55	79	69.90	122	69.80	192	70.05	239	70.00	275	70.25 508
MMMU-Pro	28.30	302	23.10	289	22.90	379	23.60	632	25.85	1012	30.35	1425	30.00	1765	35.20 5881
All	42.68	298	37.56	325	38.71	453	41.70	689	44.90	1030	48.09	1291	49.22	1588	54.97 4627
Qwen3-VL-4B															*ViPLE macro SE:* 0.21 pp
WeMath	69.75	265	64.85	919	63.40	870	62.45	987	65.85	1164	69.10	1342	72.20	1510	78.50 3432
MathVista	64.40	185	59.65	346	61.80	354	63.20	398	67.00	528	71.00	671	72.85	883	76.50 1946
MathVision	32.24	520	26.97	982	25.90	928	26.32	948	27.88	1138	30.18	1399	33.72	1909	49.75 6433
MathVerse	57.45	337	46.45	773	46.00	674	44.65	675	47.00	813	52.30	1056	58.15	1353	68.20 3432
SEED-Bench	76.45	63	75.05	101	76.75	115	77.85	143	78.35	175	78.25	199	78.60	235	78.70 287
MMMU-Pro	42.35	397	34.85	892	34.15	911	33.55	947	35.85	1065	38.45	1289	41.55	1582	52.35 4377
All	57.11	294	51.30	669	51.33	642	51.34	683	53.65	814	56.55	993	59.51	1245	67.33 3318
Qwen3-VL-8B															*ViPLE macro SE:* 0.40 pp
WeMath	72.85	164	64.00	361	67.10	574	69.70	787	73.05	966	74.25	1129	76.80	1321	82.00 2542
MathVista	66.65	131	61.85	235	64.80	307	68.05	440	70.45	578	72.80	740	74.55	882	77.30 1595
MathVision	37.50	270	29.03	874	31.17	996	32.65	1232	32.40	1396	37.42	1698	39.47	2097	52.63 6163
MathVerse	58.80	210	46.55	459	47.05	527	50.50	815	54.40	995	58.45	1289	62.75	1465	70.95 2943
SEED-Bench	74.10	43	76.30	67	76.90	89	77.65	125	78.40	157	78.25	183	78.15	206	78.25 251
MMMU-Pro	45.75	252	35.05	413	36.80	489	40.25	673	41.95	892	44.60	1173	48.10	1478	57.05 3695
All	59.27	178	52.13	402	53.97	497	56.47	679	58.44	831	60.96	1035	63.30	1241	69.70 2865
Qwen3-VL-32B															*ViPLE macro SE:* 0.12 pp
WeMath	75.20	252	68.75	185	70.30	191	71.45	276	72.90	382	77.10	567	80.45	858	84.10 2238
MathVista	67.40	198	73.80	160	74.60	193	75.80	277	77.15	350	79.25	485	81.70	599	83.05 1159
MathVision	39.06	483	33.80	544	34.54	604	37.01	738	38.40	831	43.01	1068	47.12	1426	60.44 5384
MathVerse	63.30	330	51.75	206	52.05	257	52.70	345	56.65	498	61.85	706	68.00	1038	74.45 2431
SEED-Bench	75.30	50	79.00	39	79.60	68	80.10	111	80.20	144	80.10	163	80.35	185	80.35 209
MMMU-Pro	53.90	320	50.85	398	49.60	405	51.60	534	52.50	677	55.95	863	56.95	1177	67.00 2922
All	62.36	272	59.66	255	60.11	286	61.44	380	62.97	480	66.21	642	69.10	881	74.90 2391
GLM-4.1V-9B															*ViPLE macro SE:* 0.15 pp
WeMath	68.45	276	51.40	37	52.00	69	52.40	134	57.50	260	67.25	481	73.65	814	82.15 1545
MathVista	64.10	159	65.00	44	67.15	75	67.55	140	70.50	242	75.65	392	78.15	573	81.20 941
MathVision	32.48	243	34.46	44	36.84	74	35.94	137	37.42	269	39.80	519	45.97	981	51.40 3166
MathVerse	54.20	306	38.15	50	37.95	77	38.80	142	40.45	272	50.10	503	60.30	862	68.75 1827
SEED-Bench	76.25	51	76.75	37	77.75	70	77.45	114	78.35	154	78.70	186	78.40	214	78.90 227
MMMU-Pro	42.95	261	42.10	40	41.65	71	41.20	136	43.85	256	46.55	468	49.40	783	56.70 2127
All	56.41	216	51.31	42	52.22	73	52.22	134	54.68	242	59.68	425	64.31	704	69.85 1639

4.3 Transfer beyond the training domain

② **Transfer varies with the backbone and response budget.** Out-of-domain gains on SEED-Bench and MMMU-Pro are positive for Qwen 2B, 4B and 8B, but -0.41 pp and -0.33 pp for GLM-9B and 32B (Figure 3(b)). At $B = 1024$, 32B's interpolated out-of-domain gain rises to $+2.03$ pp and its ALL gain from $+2.50$ pp to $+2.76$ pp, using 44% more decoded tokens. Response budget thus controls the transfer–cost trade-off even with a fixed latent block.



b

Accuracy gain and matching cost

	ID	OOD	All	Cost (×)
2B	**+4.69***	**+5.43**	**+5.12***	**3.5×**
4B	**+6.68***	**+4.45***	**+5.77***	**4.2×**
8B	**+8.59***	**+4.25***	**+7.15***	**5.8×**
32B	**+3.35**	**−0.33***	**+2.50**	**1.8×**
GLM-9B	**+3.64**	**−0.41**	**+2.31**	**2.0×**

Figure 3: **Accuracy and decoding cost across model scales. (a)** All 35 explicit settings, normalised to ViPLE. **(b)** Gains by split: asterisks mark comparisons with the cheapest measured setting; other gains are interpolated at equal cost. Cost gives the Full-CoT/ViPLE token ratio for the cheapest measured setting reaching ViPLE's accuracy. Panel (a) and matching costs use the six-benchmark macro average.

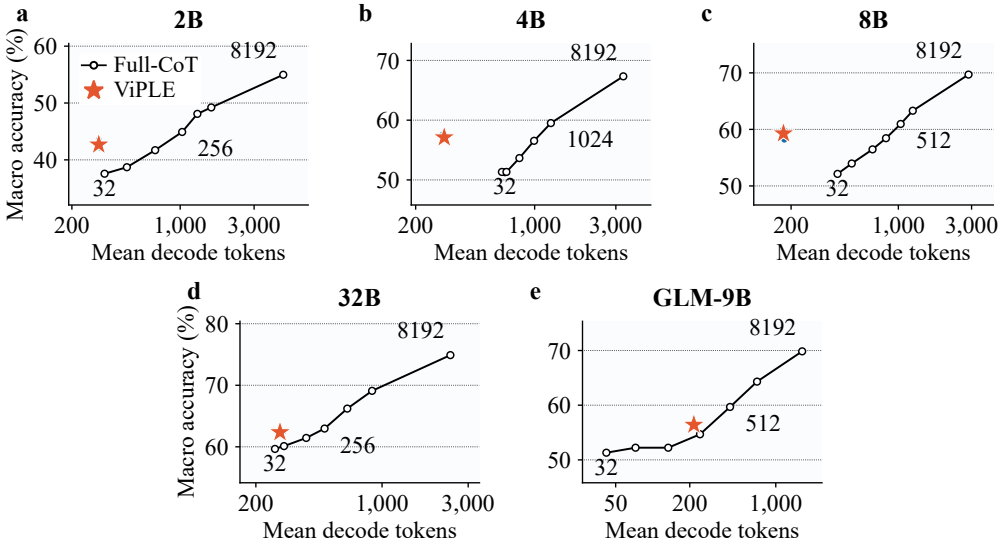


Figure 4: **Accuracy–cost curves for individual backbones.** Black curves connect seven Full-CoT budgets; labels identify the endpoints and matching setting. Stars mark ViPLE means, points show four decoding runs and error bars show ±1 standard error. Accuracy is the six-benchmark macro average.

The full curves locate the benefit in the constrained-budget regime (Figure 4). Long explicit reasoning reaches higher accuracy, while ViPLE attains a lower-cost operating point. Because the curves use measured response length, increasing the thinking cap need not increase either token count or accuracy monotonically.

4.4 Paired response examples

The MathVista examples in Appendix E show shorter correct responses (Figure A1). Both methods recover the rectangle's correct perimeter, with ViPLE using 83 tokens against 969 for Full-CoT (91.4% fewer). For the angle question, ViPLE returns the correct value of 100 in 181 tokens; the displayed Full-CoT response returns 49 in 1,029 tokens (82.4% fewer tokens for ViPLE). These selected cases complement the aggregate comparison in Table 1.

5 ANALYSIS

We examine which parts of the interface support its performance: trained conditioning states (Section 5.1) and the added prompt read (Section 5.3). Checkpoints in Section 5.2 then describe when accuracy and response length change.

5.1 EFFECTS OF LATENT INTERVENTIONS

The block adds both sequence positions and a trained conditioning representation. To test whether the selected untrained alternatives can provide the same benefit at fixed length (Goyal et al., 2024; Pfau et al., 2024), we remove the block or replace its contents while fixing the Qwen3-VL-4B backbone, injection site and native decoding protocol (Figure 5). The diagnostic covers four benchmarks with 500 examples each and MathVision with 304. The unmodified system reaches 59.65% macro accuracy at 347 mean decode tokens; Appendix C specifies all five controls.

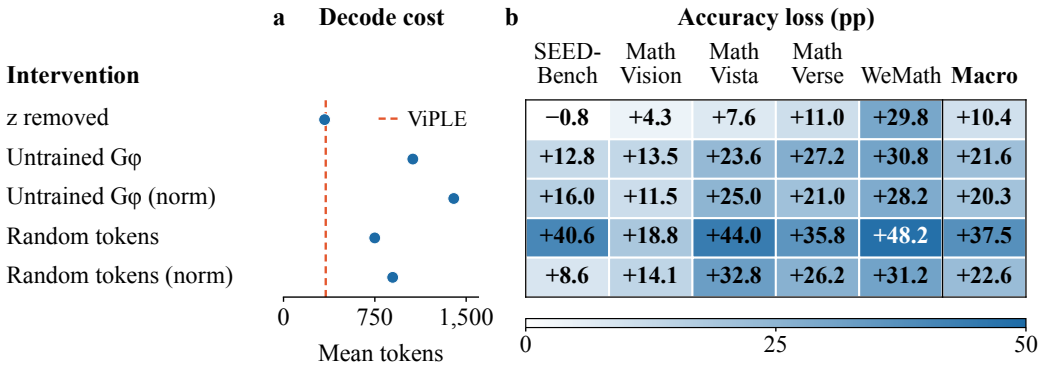


Figure 5: **Latent interventions on Qwen3-VL-4B. (a)** Mean decode tokens; dashed line: ViPLE. **(b)** Accuracy loss (ViPLE minus intervention, pp); Macro averages five benchmarks. White denotes nonpositive values; blue intensity shows increasing loss. "Norm" denotes RMS matching.

③ **The trained representation matters at fixed block length.** All four untrained replacements preserve the 256 slots, but accuracy falls by **20.34–37.47** pp and responses use **(2.2–4.0)**× as many decoded tokens. The trained representation therefore matters for both accuracy and response length even when the number of conditioning positions is held fixed.

Block removal primarily affects mathematical tasks. Removing the block lowers macro accuracy by 10.38 pp, with the largest loss on WeMath (29.80 pp). All four mathematical benchmarks decline, while SEED-Bench improves by +0.8 pp. Across the five interventions, 24 of 25 benchmark-level losses are positive. The strongest dependence on the block thus appears in the mathematical tasks.

5.2 TRAINING TRAJECTORIES AND DECODING BEHAVIOUR

Intermediate checkpoints track the emergence of useful response behaviour alongside changes in latent geometry (Figure 6). We re-decode Qwen3-VL-4B on WeMath and MathVista and measure mean-block geometry over five benchmarks at 14 checkpoints.

④ **Early accuracy gains precede the final response shortening.** Distillation ends with many very short responses: 77% on WeMath and 44% on MathVista have at most four tokens. The first ten RL steps eliminate this pattern, raise accuracy by **+30.80** pp and **+8.40** pp, and increase typical response length. By RL60, mean responses are 61% and 54% shorter than at RL10, with a further +5.0 pp and +3.0 pp in accuracy. This is a net endpoint improvement: accuracy falls at RL30 before recovering, and responses lengthen again from RL30 to RL60 (Appendix D).

Mean-block direction changes earlier. The largest sampled mean-block direction change occurs between initialisation and SFT ($1 - \cos = 0.996$). The OPD-to-RL10 change is only 0.0071 despite its large accuracy gains. Mean-block direction and response behaviour thus change at different points in training; Appendix D defines the checkpoint-based measurement.

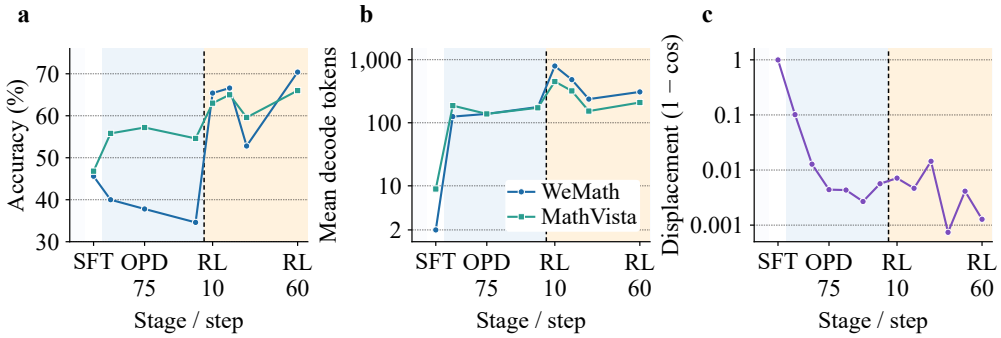


Figure 6: **Response behaviour and latent geometry evolve on different schedules.** **(a)** Accuracy and **(b)** mean decode length on WeMath and MathVista (500 examples each). **(c)** Consecutive mean-block displacement, $1 - \cos$, across 14 checkpoints (five-benchmark mean and minimum–maximum band). Backgrounds mark training stages; the dashed line marks RL onset.

5.3 Contribution of the final prompt read

We test the final cross-attention route by holding its output projection at zero throughout all three training stages. The ablated encoder retains the upstream fused prompt states, corpus, schedules and effective batch sizes. Both systems use the same evaluation items and matched settings (Table 2).

⑤ **The added prompt read benefits all four mathematical benchmarks.** Route removal lowers accuracy and increases decode length on each mathematical benchmark, despite retaining the encoder's upstream access to fused states.

Table 2: **Removing the final cross-attention read of fused prompt states on Qwen3-VL-4B.**

	Accuracy (%) ↑			Mean decode tokens ↓			Cost
Benchmark	ViPLE	No route	Δ_{acc}	ViPLE	No route	Δ_{tok}	ratio
SEED-Bench	76.10	77.10	−1.00	60.3	2.8	−57.5	0.05×
WeMath	69.00	58.65	+10.35	309.1	742.8	+433.7	2.40×
MathVista	65.60	47.50	+18.10	222.6	341.3	+118.7	1.53×
MathVision	31.91	21.55	+10.36	715.0	904.9	+189.9	1.27×
MathVerse	56.50	40.95	+15.55	441.9	655.3	+213.4	1.48×
Macro	59.82	49.15	+10.67	349.8	529.4	+179.6	1.51×

Δ_{acc} = ViPLE − No route; Δ_{tok} = No route − ViPLE; cost ratio = No route/ViPLE.

Across the five benchmarks, route removal lowers macro accuracy by 10.67 pp and adds 179.6 decode tokens (1.51× the original length). Paired exact McNemar tests are significant in every mathematical-benchmark evaluation (largest $p = 1.8 \times 10^{-2}$). On SEED-Bench, the ablated encoder averages 2.8 tokens and scores within 1 pp of ViPLE, with no significant paired comparison.

6 Conclusion

ViPLE extends parallel latent encoding with a final read of fused image–question states, learning a conditioning interface for a frozen multimodal decoder. Across five backbones, it improves the accuracy–decode-cost trade-off under constrained budgets. Replacement and retraining controls support the trained representation and added prompt read; checkpoints locate the largest accuracy gains before the final response shortening. These results establish parallel visual conditioning as a useful interface for frozen models operating within a limited text-decoding budget.

AI USE STATEMENT

Generative AI tools were used to improve the clarity and readability of the manuscript. The authors reviewed the resulting text and take responsibility for the final paper.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, et al. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Xidi Cai, Junhao Zheng, Jingye Li, Boyuan Li, Shaowei Zhang, and Qianli Ma. Look and think: Efficient multimodal reasoning via modality-decoupled compression. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 24787–24806, 2026. doi: 10.18653/v1/2026.findings-acl.1241. URL https://aclanthology.org/2026.findings-acl.1241/.

DeepSeek-AI, Daya Guo, Dejian Yang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Yuntian Deng, Kiran Prasad, Roland Fernandez, et al. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit CoT to implicit CoT: Learning to internalize CoT step by step. *arXiv preprint arXiv:2405.14838*, 2024.

GLM-V Team, Wenyi Hong, Wenmeng Yu, et al. GLM-4.5V and GLM-4.1V-Thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*, 2025. URL https://arxiv.org/abs/2507.01006.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, et al. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations*, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8ac015d409635f196f9e3e9dcfb9a94e-Abstract-Conference.html.

Tingxu Han, Zhenting Wang, Chunrong Fang, et al. Token-budget-aware LLM reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, et al. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Byungwoo Jeon, Yoonwoo Jeong, Hyunseok Lee, Minsu Cho, and Jinwoo Shin. Vision-aligned latent reasoning for multi-modal large language model. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.04476.

Cong Jiang, Xiaofeng Zhang, Fangzhi Zhu, XiaoWei Chen, Junxiong Zhu, and Zheng Zhang. Rethinking LLM reasoning: From explicit trajectories to latent representations. In *International Conference on Learning Representations*, 2026.

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. SEED-Bench: Benchmarking multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13299–13308, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Li_SEED-Bench_Benchmarking_Multimodal_Large_Language_Models_CVPR_2024_paper.html.

Pan Lu, Hritik Bansal, Tony Xia, et al. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, 2024.

Wenjie Ma, Jingxuan He, Charlie Snell, et al. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*, 2025.

Niklas Muennighoff, Zitong Yang, Weijia Shi, et al. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

Runqi Qiao, Qiuna Tan, Guanting Dong, et al. We-Math: Does your large multimodal model achieve human-like mathematical reasoning? In *Annual Meeting of the Association for Computational Linguistics*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, et al. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Xuan Shen, Yizhou Wang, Yufa Zhou, et al. Efficient reasoning with hidden thinking. In *International Conference on Machine Learning*, 2026.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, et al. CODI: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

DiJia Su, Hanlin Zhu, Yingchen Xu, et al. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint arXiv:2502.03275*, 2025.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Ke Wang, Junting Pan, Weikang Shi, et al. Measuring multimodal mathematical reasoning with MATH-Vision dataset. In *Advances in Neural Information Processing Systems*, 2024.

Yibo Wang, Yongcheng Jing, Shunyu Liu, Hao Guan, Rong-cheng Tu, Chengyu Wang, Jun Huang, and Dacheng Tao. VTC-R1: Vision-text compression for efficient long-context reasoning. *arXiv preprint arXiv:2601.22069*, 2026. URL https://arxiv.org/abs/2601.22069.

Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Guowei Xu, Peng Jin, Ziang Wu, et al. LLaVA-CoT: Let vision language models reason step-by-step. In *IEEE/CVF International Conference on Computer Vision*, 2025a.

Silei Xu, Wenhao Xie, Lingxiao Zhao, et al. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025b.

Yige Xu, Xu Guo, Zhiwei Zeng, et al. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In *Annual Meeting of the Association for Computational Linguistics*, 2025c.

Xiang Yue, Tianyu Zheng, Yuansheng Ni, et al. MMMU-Pro: A more robust multi-discipline multimodal understanding benchmark. *arXiv preprint arXiv:2409.02813*, 2024.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, et al. LightThinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*, 2025a.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, et al. MathVerse: Does your multi-modal LLM truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, 2024.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b. URL https://arxiv.org/abs/2506.05176.

Zhuosheng Zhang, Aston Zhang, Mu Li, et al. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023.

Xiandong Zou, Jing Huang, Jianshu Li, and Pan Zhou. Latent thought flow: Efficient latent reasoning in large language models. *arXiv preprint arXiv:2606.16222*, 2026. URL https://arxiv.org/abs/2606.16222.

APPENDIX CONTENTS

Appendix A. Training and decoding configuration 13
Appendix B. Evaluation and accuracy–cost comparisons 14
Appendix C. Latent intervention construction 15
Appendix D. Checkpoint measurements 16
Appendix E. Paired response case studies 17
Appendix F. Paired correctness and decoding cost 18
Appendix G. Chart-reading examples 19
Appendix H. Natural-image examples 20
Appendix I. Response-length distributions across backbones 21
Appendix J. Benchmark-level response-length profiles 22
Appendix K. Limitations 22

A TRAINING AND DECODING CONFIGURATION

Supervised alignment and reinforcement learning use all 8,056 training examples. Distillation uses examples with a non-empty backbone-specific rationale: 3,889 (2B), 3,895 (4B), 4,729 (8B), 4,554 (GLM-9B) and 5,001 (32B). The response-length threshold is $m_{\min} = 10$ words.

Table 3 gives the shared configuration. Data-parallel rank counts provide the stated effective batch size and number of questions per step. Training follows three supervised epochs and fixed distillation and reinforcement-learning schedules, with no evaluation-based early stopping. The main comparison uses force-close termination; the intervention and checkpoint analyses use native decoding. Visual-route retraining uses the paired canonical evaluation settings. The reproduction records retain exact random-state identifiers and per-experiment configurations.

Table 3: **Training and decoding configuration shared across backbones.**

	Stage 1 (SFT)	Stage 2 (OPD)	Stage 3 (gated RL)
Objective	answer-span LM	forward KL on answer span	clipped policy gradient
Optimiser	AdamW	AdamW	AdamW
Learning rate	5×10^{-4}	10^{-5}	5×10^{-6}
Warmup steps	100	–	–
Length	3 epochs	150 steps	60 steps
Effective batch	8 sequences	128 questions/step	64 questions/step
Samples/question	–	4	8
Rollout cap	–	64 new tokens	384 new tokens
Rollout sampling	–	–	$T = 1.0$, top-p 1.0
KL coefficient	–	–	0.05
PPO clip ϵ	–	–	0.2
Gradient clipping	–	–	1.0
Reasoning gate	–	–	≥ 10 words

Shared across stages	
Encoder body G_ϕ	Qwen3-Embedding-0.6B, token embedding removed
Latent slots L	256
Training corpus	8,056 image-question-answer rows; rationale subsets above
Decoding (main comparison)	
Force-close budget B	512 (ViPLE arm; the ladder arm instead sweeps B per rung)
Sampling	$T = 0.6$, top-p 0.95

Teacher records. All five training tables contain the same image–question–answer triples and backbone-specific rationales. The 2B, 4B and 8B tables share a row order; 32B and GLM-9B use different permutations. Each backbone reads its corresponding table. Qwen teacher rationales were collected greedily; GLM collection used temperature 0.6, top-p 0.95 and top-k 20. These collection settings are distinct from student rollout and evaluation sampling. During OPD, the rationale supplies

teacher context, and the loss compares teacher and student answer distributions along student-sampled continuations. At each answer position, both distributions are conditioned on the same preceding student tokens; the teacher additionally has its recorded rationale, while the student has the latent block.

B EVALUATION AND ACCURACY–COST COMPARISONS

Examples and aggregation. Every backbone and budget uses the same benchmark snapshots, examples and order (Table 4). For a benchmark with N questions, accuracy in decoding run r is $A_r = 100N^{-1}\sum_i h_{ri}$, where h_{ri} is the scored correctness of response i. The reported pass@1 averages four decoding runs of the fixed checkpoint; its standard error describes decoding variation, not variation across independently trained models. ID, OOD and ALL weight their member benchmarks equally, so MathVision has the same benchmark weight as each 500-question set.

Table 4: **Evaluation examples per backbone and decoding run.**

Benchmark	Split	Examples
WeMath	ID	500
MathVista	ID	500
MathVision	ID	304
MathVerse	ID	500
SEED-Bench	OOD	500
MMMU-Pro	OOD	500
Total		**2,804**

Sequential decoding cost. The main comparison uses the mean live decode length. For token count t_{ri} and $I_r = \{i : t_{ri} > 0\}$, it is

$$T_{\text{live}} = \frac{1}{4}\sum_{r=1}^{4}\frac{\sum_{i\in I_r} t_{ri}}{|I_r|}.$$

Accuracy includes every evaluated response. The cost mean excludes immediate zero-token terminations; a run with no non-empty responses receives zero in the reducer. The count includes reasoning and answer tokens. The 256 latent slots enter through prefill and add zero sequential decoding steps. Encoder and backbone-prefill computation remain separate from this token metric. The native intervention and checkpoint tables use the arithmetic mean over all recorded responses, together with the median where shown.

Budgets and comparison points. Full-CoT follows its thinking caps with a separate 2,048-token answer cap; ViPLE uses a force-close thinking budget of $B = 512$. Thus B differs from total decoded length. Full-CoT's fallback requests one continuation of at most 64 tokens after a capped or incomplete answer; ViPLE uses direct force-close termination.

For each run, baseline points are sorted by measured cost. Between adjacent points (T_0, A_0) and (T_1, A_1), the baseline accuracy at T is

$$\widehat{A}(T) = A_0 + \frac{T - T_0}{T_1 - T_0}(A_1 - A_0), \qquad T_0 \le T \le T_1.$$

Coincident-cost endpoints use the average of their accuracies. Equal-cost gains average the paired run differences and require bracketing in all four runs and in the aggregate curve. Otherwise, Figure 3(b) uses the cheapest measured baseline, marked with an asterisk. The matching-cost comparison instead selects the measured setting with the fewest tokens whose accuracy reaches ViPLE's accuracy.

Table 5: **Measured Full-CoT settings reaching ViPLE accuracy (six-benchmark average).**

		Accuracy (%)		Decode tokens		
Backbone	B	ViPLE	Full-CoT	ViPLE	Full-CoT	Ratio
2B	256	42.68	44.90	298.1	1029.9	3.5×
4B	1024	57.11	59.51	294.4	1245.4	4.2×
8B	512	59.27	60.96	178.5	1035.3	5.8×
32B	256	62.36	62.97	272.1	480.3	1.8×
GLM-9B	512	56.41	59.68	216.0	425.0	2.0×

C LATENT INTERVENTION CONSTRUCTION

The five controls in Figure 5 use the final Qwen3-VL-4B encoder and native response termination. They hold the backbone, input processing and answer scorer fixed. The intervention hook substitutes the latent block immediately before generation. Each benchmark contributes its own macro-average component; MMMU-Pro is outside this five-benchmark diagnostic set.

Table 6: **Construction of the five latent interventions.**

Condition	Block supplied to the frozen backbone
z removed	Replace the latent block with embeddings of the reasoning-close string and generate the answer.
Untrained G_ϕ	Restore the encoder's construction state, including the pretrained transformer body and initial trajectory/projection parameters.
Untrained G_ϕ (norm)	Apply slot-wise norm matching to the untrained encoder output for each question.
Random tokens	Use 256 ordinary-vocabulary token embeddings sampled once and shared across questions; exclude special tokens.
Random tokens (norm)	Rescale each random-token embedding to its corresponding trained-encoder reference norm.

Slot-wise norm matching. For a benchmark with N inputs, the trained encoder defines the reference magnitude $r_\ell = \|N^{-1}\sum_i z_{i\ell}\|_2$ at slot ℓ. This reference uses input states without answer labels. A replacement vector u_ℓ is rescaled as

$$\widetilde{u}_\ell = u_\ell \frac{r_\ell}{\max(\|u_\ell\|_2, 10^{-6})}.$$

With a fixed hidden width, this is also RMS matching. Each slot retains its replacement direction. The untrained-encoder control restores the constructor state, including its pretrained transformer body and initial trajectory/projection parameters.

Table 7: **Five-benchmark results for the five latent interventions.**

Condition	Accuracy (%)	Decode tokens	Loss (pp)	Cost ratio
ViPLE	59.65	346.8	+0.00	1.00×
z removed	49.28	337.3	+10.38	0.97×
Untrained G_ϕ	38.08	1062.0	+21.58	3.06×
Untrained G_ϕ (norm)	39.31	1398.0	+20.34	4.03×
Random tokens	22.18	749.1	+37.47	2.16×
Random tokens (norm)	37.06	896.5	+22.59	2.59×

Accuracy loss is ViPLE minus intervention accuracy, and the cost ratio is intervention tokens divided by ViPLE tokens. Both use the native-decoding reference in Figure 5, consistently across the five controls.

Visual-route retraining. Table 2 uses a separate retraining control. The final cross-attention output projection remains at its zero initialisation throughout SFT, OPD and RL. Upstream fused prompt states remain available, and both arms use the same examples and matched evaluation settings. The reported paired exact McNemar tests count question-level outcomes where only one arm is correct, using the two-sided binomial test on these discordant pairs.

D CHECKPOINT MEASUREMENTS

Accuracy and response length. Table 8 gives the numerical values underlying the response trajectories in Figure 6. Accuracy and Short are percentages; Mean and Median are decoded-token counts. Short denotes responses of at most four tokens. These diagnostics use native termination at the indicated training checkpoints. The short-response threshold is separate from the ten-word reward gate used during RL.

Table 8: **Checkpoint response statistics on 500 examples per benchmark.**

	WeMath				MathVista			
Checkpoint	Acc.	Mean	Median	Short	Acc.	Mean	Median	Short
SFT (3 epochs)	45.6	2.0	2	100.0	46.8	8.9	6	46.8
OPD 25	40.0	125.2	2	59.8	55.8	186.8	34	37.0
OPD 75	37.8	138.3	2	82.6	57.2	138.2	30	47.0
OPD 150	34.6	177.8	2	77.2	54.6	172.8	57	43.6
RL 10	65.4	793.9	427	0.0	63.0	450.5	211	0.0
RL 20	66.6	482.4	289	0.0	65.0	320.1	164	0.0
RL 30	52.8	237.2	96	13.2	59.6	153.3	61	9.2
RL 60	70.4	308.5	125	0.0	66.0	209.5	93	0.0

The OPD WeMath median is two tokens despite a long response tail. By RL10, accuracy and typical length rise; final mean length is lower. This is a nonmonotonic trajectory: accuracy falls at RL30, then recovers as responses lengthen again by RL60.

Latent geometry. For benchmark b at checkpoint k, flatten the mean latent block to a vector $v_{b,k}$. The consecutive displacement is

$$d_{b,k} = 1 - \frac{v_{b,k}^{\mathsf{T}} v_{b,k-1}}{\|v_{b,k}\|_2 \, \|v_{b,k-1}\|_2}.$$

Table 9 lists the mean and range across the five diagnostic benchmarks for each of the 13 transitions between 14 checkpoints. Its mean and range correspond to the line and band in Figure 6(c).

Table 9: **Consecutive mean-block displacement across five benchmarks.**

Transition	Mean	Minimum	Maximum
Initialisation → SFT	0.99581	0.99021	0.99914
SFT → OPD 25	0.10120	0.09933	0.10529
OPD 25 → OPD 50	0.01277	0.01246	0.01337
OPD 50 → OPD 75	0.00441	0.00431	0.00457
OPD 75 → OPD 100	0.00433	0.00418	0.00453
OPD 100 → OPD 125	0.00269	0.00262	0.00280
OPD 125 → OPD 150	0.00564	0.00528	0.00590
OPD 150 → RL 10	0.00713	0.00694	0.00749
RL 10 → RL 20	0.00467	0.00459	0.00472
RL 20 → RL 30	0.01441	0.01367	0.01513
RL 30 → RL 40	0.00074	0.00072	0.00075
RL 40 → RL 50	0.00413	0.00405	0.00425
RL 50 → RL 60	0.00128	0.00121	0.00131

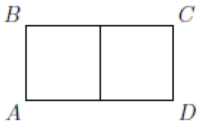
The largest sampled direction change spans initialisation to SFT (three epochs); OPD and RL are sampled more frequently. The small OPD-to-RL10 change accompanies a large accuracy gain, separating mean-block direction changes from behavioural gains.

E PAIRED RESPONSE CASE STUDIES

Figure A1 shows two MathVista geometry questions from the paired Qwen3-VL-4B evaluation. Full-CoT uses $B = 1024$ and ViPLE uses $B = 512$. Each displayed answer and token count comes from its recorded response. The examples cover a shared correct answer and a case where ViPLE supplies the correct answer while the displayed Full-CoT response fails.

a Perimeter

Rectangle ABCD is subdivided into two identical square regions, as in the figure above. If the area of each square is 9, what is the perimeter of ABCD?



Reference answer: 18

Full-CoT 969 tokens

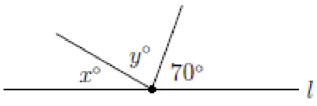
"Therefore, the final answer is 18."

ViPLE **83 tokens**

"Therefore, the final answer is 18."

b Angle

In the figure above, two line segments meet at a point on line l. If the value of y is equal to the square of the value of x, what is the value of y?



Reference answer: 100

Full-CoT 1,029 tokens

"Therefore, the final answer is 49."

ViPLE **181 tokens**

"Therefore, the final answer is 100."

Figure A1: **Paired MathVista responses on Qwen3-VL-4B.** Full-CoT uses $B = 1024$ and ViPLE uses $B = 512$. Quoted final-answer text is reproduced verbatim from the stored responses; token counts include reasoning and the answer. Questions are transcribed and diagram regions come from the original inputs; the diagrams are not to scale.

Perimeter. Each square has area 9, so its side length is 3. The rectangle is 6×3 and its perimeter is $2(6 + 3) = 18$. Both displayed responses return 18; ViPLE uses 83 decoded tokens and Full-CoT uses 969.

Angle. The three angles on a straight line satisfy $x + y + 70 = 180$, and the question specifies $y = x^2$. The positive solution of $x^2 + x - 110 = 0$ is $x = 10$, giving $y = 100$. ViPLE returns 100 in 181 tokens, while the displayed Full-CoT response returns 49 in 1,029 tokens. These calculations verify the reference answers.

Table 10: **Four decoded responses for each selected question and method.**

Question	Method	Correct	Mean tokens	Token range
Perimeter	ViPLE	4/4	90.5	83–109
Perimeter	Full-CoT	4/4	952.0	781–1029
Angle	ViPLE	4/4	201.8	179–231
Angle	Full-CoT	1/4	1716.2	1029–3137

Selection and traceability. The examples were selected from geometry questions for which ViPLE produced a correct, shorter response. Table 10 includes all four decoding draws for each question and method, while Table 1 gives full-evaluation performance. The accompanying records retain question identifiers, source rows and hashes; image pixels and answers were checked against the fixed dataset snapshot.

F PAIRED CORRECTNESS AND DECODING COST

The paired analysis uses Qwen3-VL-4B on all 2,804 evaluation questions, with the same four decoding runs as the main comparison. Full-CoT uses a fixed $B = 1024$ and ViPLE uses $B = 512$ throughout. Each pair matches the benchmark, question and decoding run. Figure A2 displays all 11,216 pairs; within each benchmark, the tracks share one ordering by Full-CoT's total decoded length.

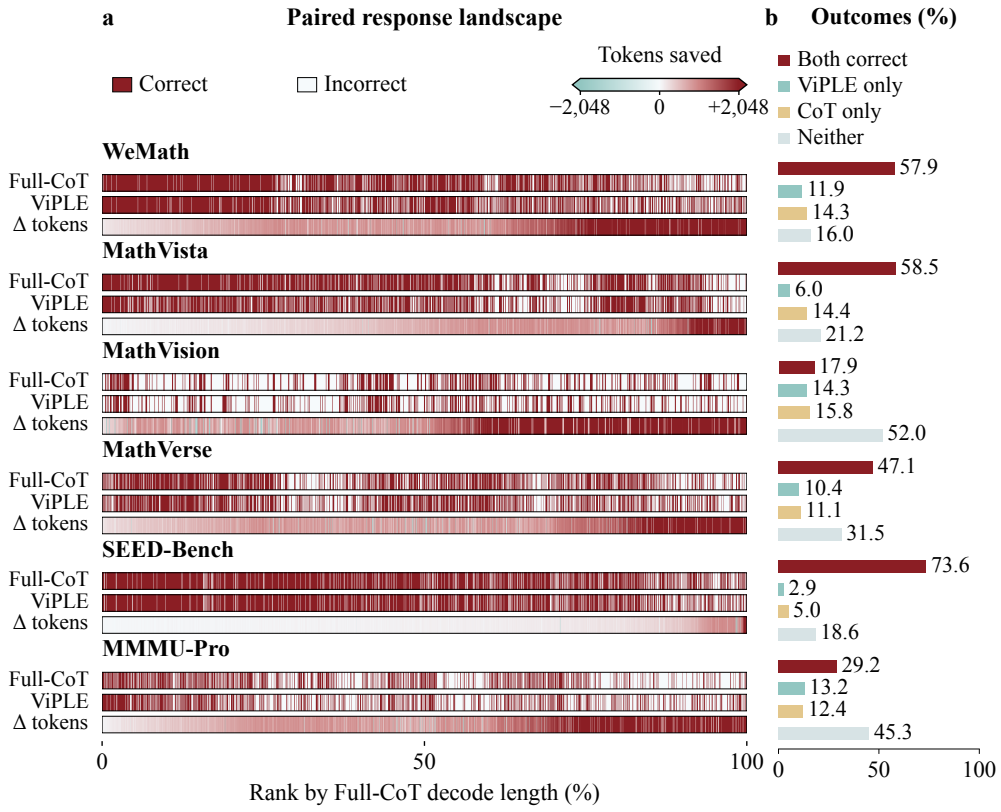


Figure A2: **Paired correctness and token savings across six benchmarks.** (a) Each column is a response pair. Dark cells mark correct answers; the lower band shows Full-CoT minus ViPLE tokens, with colors saturating at ±2,048. (b) Four outcome shares per benchmark, each measured from zero; labels give percentages. CoT denotes Full-CoT.

Table 11: **Paired correctness outcomes and token reductions (%).**

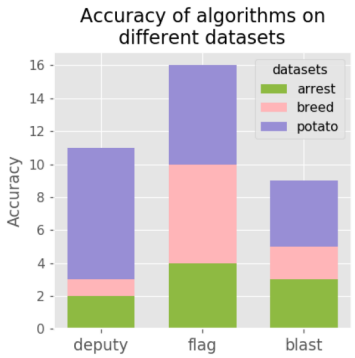
Benchmark	Both correct	ViPLE only	CoT only	Neither	Fewer tokens
WeMath	57.9	11.9	14.3	16.0	98.5
MathVista	58.5	6.0	14.4	21.2	98.6
MathVision	17.9	14.3	15.8	52.0	95.2
MathVerse	47.1	10.4	11.1	31.5	96.6
SEED-Bench	73.6	2.9	5.0	18.6	98.5
MMMU-Pro	29.2	13.2	12.4	45.3	97.7

Shorter responses are also common when both methods answer correctly. ViPLE uses fewer tokens in 97.7% of all pairs (10,956/11,216) and in 98.7% of pairs with two correct answers (5,472/5,543). Table 11 separates the four correctness outcomes. These token comparisons include every recorded response, including immediate terminations.

G CHART-READING EXAMPLES

Figure A3 extends the geometry examples with two MathVista chart-reading questions. The comparison uses Qwen3-VL-4B, Full-CoT at $B = 1024$ and ViPLE at $B = 512$, as in Appendix E. Table 12 reports all four draws for these selected examples.

a Reading a stacked value



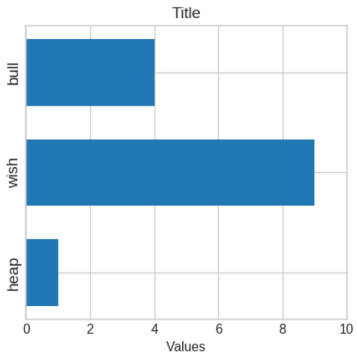
What is the lowest accuracy reported in the whole chart?

Reference: 1

Full-CoT 710 tokens
"Therefore, the final answer is 1."

ViPLE **73 tokens**
"Therefore, the final answer is 1."

b Applying a strict threshold



How many bars have values smaller than 1?

Reference: 0

Full-CoT 1,028 tokens
"Therefore, the final answer is 1."

ViPLE **79 tokens**
"Therefore, the final answer is 0."

Figure A3: **Reading individual values and applying a threshold.** Original input images, complete question text and verbatim stored final answers. Token counts include reasoning and the answer.

Reading the chart. In (a), the pink segment of the first stacked bar spans 2 to 3, giving the minimum component value of 1. Both responses recover this value. In (b), the three horizontal bars have values 4, 9 and 1. None is strictly below 1, so the reference answer is 0; ViPLE returns 0 in 79 tokens and Full-CoT returns 1 in 1,028 tokens.

Table 12: **All recorded draws for the two chart questions.**

Case	Method	Correct	Mean tokens	Token range
(a)	Full-CoT	4/4	431.5	306–710
(a)	ViPLE	4/4	86.8	53–130
(b)	Full-CoT	3/4	528.8	251–1,028
(b)	ViPLE	4/4	81.2	66–99

H NATURAL-IMAGE EXAMPLES

Figure A4 compares responses to two SEED-Bench questions using Qwen3-VL-4B, Full-CoT at $B = 1024$ and ViPLE at $B = 512$. Both methods give the correct answer in each displayed case. Selection required correct ViPLE answers across all four recorded draws and readable Full-CoT explanations; Table 13 reports every draw for these questions.

a Counting a musical group



How many bands are performing on the stage?
A. Four B. Two
C. Three D. One

Reference: D

Full-CoT **1,128 tokens**

"The stage has one band performing. The question is asking how many bands, so if there's one group, it's one. [...] The stage has multiple musicians, but they are part of one band. So the number of bands is one. So the answer is D."

ViPLE **137 tokens**

"Therefore, the final answer is 1 band, so option D. D"

b Counting pink flowers



How many pink flowers are in the image?
A. 1 B. 2
C. 3 D. 4

Reference: B

Full-CoT **1,112 tokens**

"First, identify the pink flowers by their petal color. The image contains two flowers with pinkish petals: one at the top center (petals transitioning from pink to white) and one on the left side (petals with pink and white hues). [...] Counting the pink flowers: 2."

ViPLE **38 tokens**

"Therefore, the final answer is 2. Final answer: B"

Figure A4: **Correct scene answers with shorter responses.** Full-CoT passages are verbatim excerpts from the stored answer text; bracketed ellipses mark omissions. Images are reproduced intact. Token counts cover the complete response, including reasoning and the answer.

Table 13: **All recorded draws for the two scene questions.**

Case	Method	Correct	Mean tokens	Token range
(a)	Full-CoT	3/4	494.8	210–1,128
(a)	ViPLE	4/4	59.5	28–137
(b)	Full-CoT	4/4	960.5	769–1,112
(b)	ViPLE	4/4	48.5	38–62

I RESPONSE-LENGTH DISTRIBUTIONS ACROSS BACKBONES

The distributional analysis pairs every ViPLE response with Full-CoT at the measured matching-cost setting in Table 5: $B = 256$, 1024, 512, 256 and 512 for 2B, 4B, 8B, 32B and GLM-9B, respectively. Each backbone contributes 11,216 pairs from all 2,804 questions and four decoding draws, giving 56,080 pairs in total.

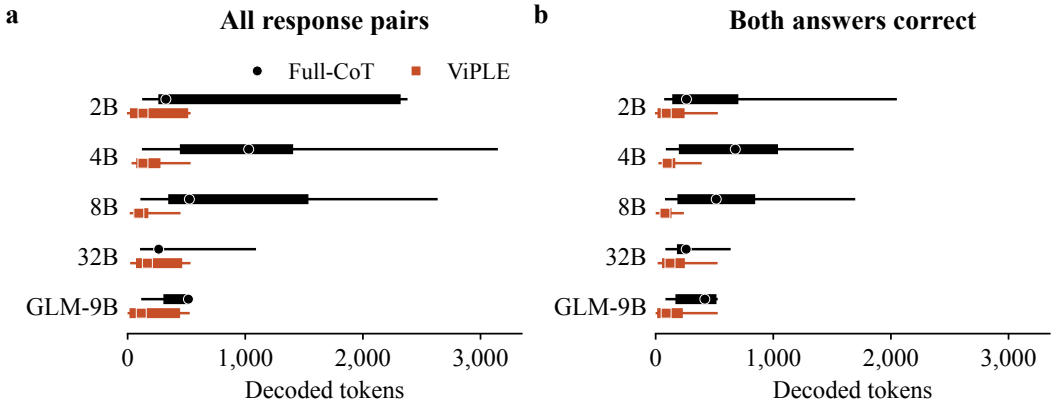


Figure A5: **Typical and upper-tail response lengths.** Markers show medians, thick segments the 25th–75th percentiles and thin segments the 10th–90th percentiles. Panels share the same token axis; (b) retains pairs where both answers are correct. These are distributional intervals.

Table 14: **Response-length quantiles and paired token reductions.**

Backbone	Median	P90	P95	Fewer (%)	Both n	Both (%)
2B	131 / 325	525 / 2,369	542 / 2,369	85.5	3,424	88.5
4B	130 / 1,029	525 / 3,137	676 / 3,137	97.7	5,543	98.7
8B	95 / 525	439 / 2,625	523 / 2,625	98.3	5,738	99.1
32B	168 / 264	524 / 1,081	526 / 2,315	75.8	6,077	83.6
GLM-9B	116 / 517	517 / 519	517 / 525	78.3	5,564	85.1

Quantiles and weighting. Each quantile entry lists ViPLE / Full-CoT tokens. P90 and P95 are the 90th and 95th percentiles. For sorted lengths $t_{(1)}, \ldots, t_{(N)}$, the reported quantile is $Q(p) = t_{(\lceil pN \rceil)}$. Every recorded response has equal weight, including zero-token outputs. These pooled distributions describe individual responses; the main accuracy–cost comparison uses equal benchmark weights. Fewer is the percentage of all pairs with strictly fewer ViPLE tokens; Both applies the same test to the subset in which both answers are correct, with the subset size given by Both n.

Typical cost and the long tail. ViPLE lowers the median response length on all five backbones, with different effects on the upper tail. For 4B, the median falls from 1,029 to 130 tokens and the 90th percentile from 3,137 to 525. For 32B, these decrease from 264 to 168 and from 1,081 to 524, respectively. GLM-9B has a lower median (517 to 116) but nearly unchanged 90th percentiles (519 and 517). The distributional view thus separates typical savings from tail behaviour.

Budget interpretation. The Full-CoT thinking budget is followed by a separate answer allowance. Consequently, total response lengths can exceed B, as seen in the upper quantiles. The latent block is supplied during prefill.

J BENCHMARK-LEVEL RESPONSE-LENGTH PROFILES

Table 15 expands the pooled distributions in Appendix I into all 30 backbone–benchmark combinations. Each cell lists ViPLE / Full-CoT tokens under the same matching-cost settings. Each benchmark contributes 2,000 paired responses per backbone, except MathVision with 1,216. Quantiles include every response and use the same nearest-rank definition.

Table 15: **Response-length profiles for every backbone and benchmark.**

Benchmark	Median tokens	P90 tokens	Fewer tokens (%)
Qwen3-VL-2B; Full-CoT $B = 256$			
WeMath	224 / 1,038	529 / 2,369	88.3
MathVista	126 / 263	520 / 2,369	83.9
MathVision	493 / 2,369	536 / 2,369	84.7
MathVerse	275 / 1,261	527 / 2,369	83.8
SEED-Bench	35 / 131	141 / 263	90.2
MMMU-Pro	158 / 292	532 / 2,369	81.9
Qwen3-VL-4B; Full-CoT $B = 1024$			
WeMath	137 / 1,041	526 / 3,137	98.5
MathVista	101 / 791	408 / 1,695	98.6
MathVision	293 / 1,092	1,147 / 3,137	95.2
MathVerse	189 / 1,040	524 / 3,137	96.6
SEED-Bench	45 / 130	86 / 529	98.5
MMMU-Pro	190 / 1,050	549 / 3,137	97.7
Qwen3-VL-8B; Full-CoT $B = 512$			
WeMath	112 / 700	306 / 2,625	99.3
MathVista	72 / 516	209 / 2,085	98.6
MathVision	161 / 1,994	518 / 2,625	97.9
MathVerse	135 / 917	438 / 2,625	98.2
SEED-Bench	27 / 116	59 / 436	98.2
MMMU-Pro	118 / 580	528 / 2,625	97.2
Qwen3-VL-32B; Full-CoT $B = 256$			
WeMath	178 / 270	524 / 534	71.8
MathVista	113 / 260	516 / 526	85.1
MathVision	515 / 269	525 / 2,369	53.4
MathVerse	258 / 270	524 / 1,078	63.8
SEED-Bench	32 / 118	96 / 259	97.8
MMMU-Pro	261 / 269	527 / 2,307	74.2
GLM-9B; Full-CoT $B = 512$			
WeMath	205 / 517	517 / 517	69.7
MathVista	64 / 487	517 / 517	83.6
MathVision	150 / 517	517 / 523	72.9
MathVerse	230 / 517	518 / 528	75.9
SEED-Bench	12 / 128	105 / 450	93.6
MMMU-Pro	150 / 517	517 / 520	72.2

The final column gives the share of pairs in which ViPLE uses strictly fewer tokens, regardless of correctness. Together with the median and P90, it shows whether savings are widespread or concentrated in long responses. Table 1 supplies the corresponding benchmark accuracies.

K LIMITATIONS

ViPLE's fixed 256-slot block targets constrained decoding budgets. Long explicit reasoning reaches higher accuracy, and out-of-domain gains vary with the backbone and response budget. Adaptive latent capacity and decoding budgets are natural extensions of this operating regime. The evaluation measures sequential decode tokens under each system's termination fallback, with four decoding runs characterising sampling variation at fixed checkpoints. Full latency and memory profiling, including

encoder and prefill work, would establish deployment costs. Matched-termination controls, repeated training and comparisons with other latent methods would further separate the effects of conditioning, optimisation and evaluation protocol.