LATENTWEAVE: LEARNING HIGH-VALUE THOUGHT FOR EFFICIENT VISUAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling reasoning increasingly relies on longer chains of thought. Because natural language expresses both reasoning progress and elaboration of existing thought, scaling explicit reasoning does not necessarily constitute more intelligence. We hypothesize that reasoning capability depends on high-value thought and its coherent organization. Motivated by this hypothesis, we introduce LatentWeave, combining rubric-guided reinforcement learning to retain high-value explicit reasoning with interleaved latent updates to maintain reasoning continuity. Experiments on six benchmarks covering hard multimodal, vision-dependent, and long-chain reasoning establish an empirical efficiency–accuracy Pareto frontier. Relative to the Qwen3-VL-8B-Thinking baseline, LatentWeave achieves **$2.92\times$ reasoning efficiency with 1.86% relative accuracy loss**, and **$5.53\times$ efficiency and 3.84% loss**. This Pareto frontier supports flexible model selection on varied reasoning budget and performance requirements.

1 INTRODUCTION

Chain-of-thought (CoT) enables vision language models to combine visual observation, symbolic reasoning, and verification (Bai et al., 2025; Chen et al., 2025), but each additional token incurs serial decoding cost. Language can express a new inference or elaborate an existing one: long traces often revisit resolved hypotheses or repeat derivations after an answer is supported (Chen et al., 2024c). Efficient visual reasoning requires deciding what to express and how to continue.

Thought and verbal expression are not interchangeable. Human word prediction reflects constituent structure (Zou et al., 2026), and cognitive evidence dissociates linguistic ability from several forms of complex thought (Fedorenko et al., 2024). In language models, tokens contribute unequally to reinforcement learning (RL), and continuous states can support intermediate computation (Wang et al., 2025; Hao et al., 2024). These observations motivate our hypothesis: **reasoning capability depends on high-value thought and its coherent organization**. We call a thought high-value when it establishes a needed premise, advances the solution, or resolves uncertainty affecting the answer.

Prior work shows that chain-of-thought traces can be shortened through token selection or length-constrained reinforcement learning while largely preserving reasoning accuracy (Xia et al., 2025; Hou et al., 2025). In visual reasoning, compression must preserve the premises and corrections needed for subsequent steps. Neither answer correctness nor a length preference distinguishes a needed visual premise from a repeated derivation.

We therefore ask a rubric judge to inspect successive reasoning segments and their transitions before assigning an overall trajectory score. This assessment guides *selection*: what should remain explicit. *Continuity* concerns how generation proceeds with less verbalized reasoning.

LatentWeave learns selection and continuity jointly. **Rubric RL** ranks grounded progress, necessary correction, and answer support within correctness tiers. **Interleaved Latent Actions** update the next token input without adding an output marker or key–value (KV) position. A semantic router selects Action positions, while frequency regulation maintains sampling opportunities as trajectories shorten. An optional external channel supplies additional evidence, instantiated here with visual information. Joint on-policy training learns what to verbalize and how to route and update the state.

Across six benchmarks, LatentWeave achieves $2.92\times$ reasoning efficiency with 1.86% relative accuracy loss and $5.53\times$ efficiency with 3.84% loss, offering two budget-dependent operating points.

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

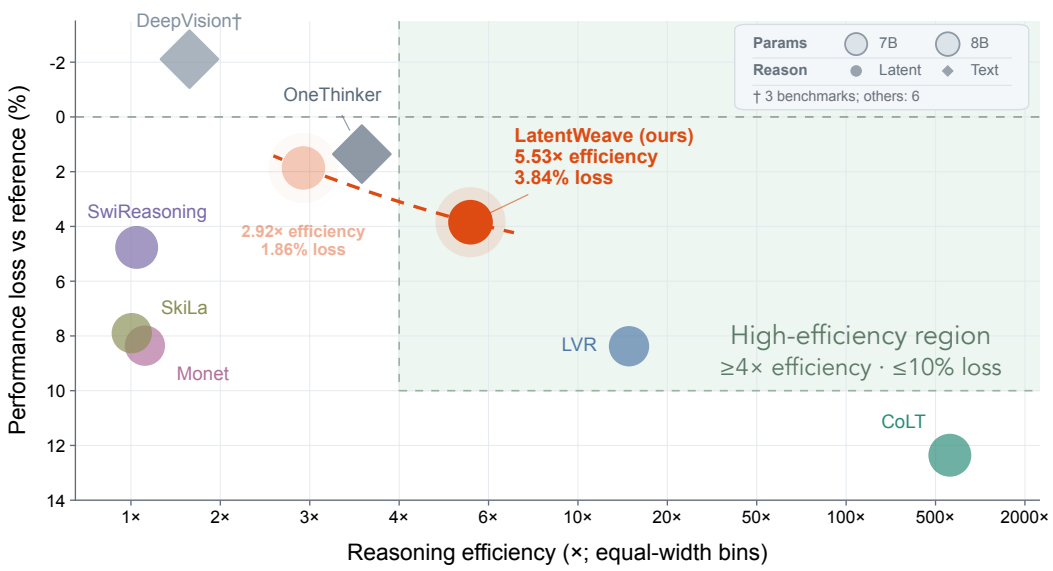

Figure 1: **Efficiency–accuracy trade-off.** Orange points show LatentWeave relative to Qwen3-VL-8B-Thinking; the dashed curve and its short extensions are visual guides through the two measured operating points. Other methods use their respective baselines. Labeled efficiency intervals have equal display width. Results cover six benchmarks except DeepVision (†, three). Appendix H specifies reference baselines and axis scaling.

1. **Learning what to retain:** a correctness-tiered rubric objective that favors high-value explicit reasoning and shorter successful trajectories;
2. **Learning how to continue:** interleaved latent updates with semantic routing and frequency regulation, jointly trained with the decoder through group relative policy optimization (GRPO); and
3. **Characterizing the trade-off:** a six-benchmark efficiency–accuracy frontier, complemented by training controls, inference interventions, and compression dynamics.

2 Related Work

Efficient textual reasoning. Overthinking studies identify repeated exploration and verification with diminishing returns (Chen et al., 2024c; Sui et al., 2025). TokenSkip, CoT-Valve, Extra-CoT, L1, ThinkPrune, and ShorterBetter compress reasoning targets, control length, or reward concise correct completions (Xia et al., 2025; Ma et al., 2025; Tang et al., 2026; Aggarwal & Welleck, 2025; Hou et al., 2025; Yi et al., 2026). Shorthand for Thought merges recurring low-entropy phrases into readable supertokens (Zhao et al., 2026). We use trajectory feedback to guide explicit reasoning jointly with latent updates.

Continuous latent reasoning. Continuous methods reduce reliance on verbalized intermediate steps (Yu et al., 2026). Coconut and CODI carry computation in latent states (Hao et al., 2024; Shen et al., 2025b). HRPO retains stochastic token rollouts while mixing a vocabulary-projected continuous representation into the next embedding (Yue et al., 2026). LatentWeave conditions subsequent generation on a bounded residual from the preceding hidden state and optional external evidence.

Interleaving and structured feedback. ThinkRouter and SWITCH route between textual and latent computation (Xu et al., 2026; Yang et al., 2026). Heima distills predefined multimodal CoT stages into thinking tokens (Shen et al., 2025a), and Tyler combines visible tokens with typed latent operators (Lin et al., 2026). RFTT combines functional tokens, tree search, process rewards, and online RL for mathematics (Zhang et al., 2025). LatentWeave jointly learns *what* remains explicit,

where an Action occurs, and *how* its residual conditions continuation. The rubric scores visible reasoning; ModeSwitcher and ActionNet execute updates at ordinary-token boundaries.

3 LATENTWEAVE

Given an image and question x, the policy generates a reasoning trace and answer $y_{1:T}$. Rubric RL assesses how visible spans contribute to the solution, while Interleaved Latent Actions support subsequent generation (Figure 2).

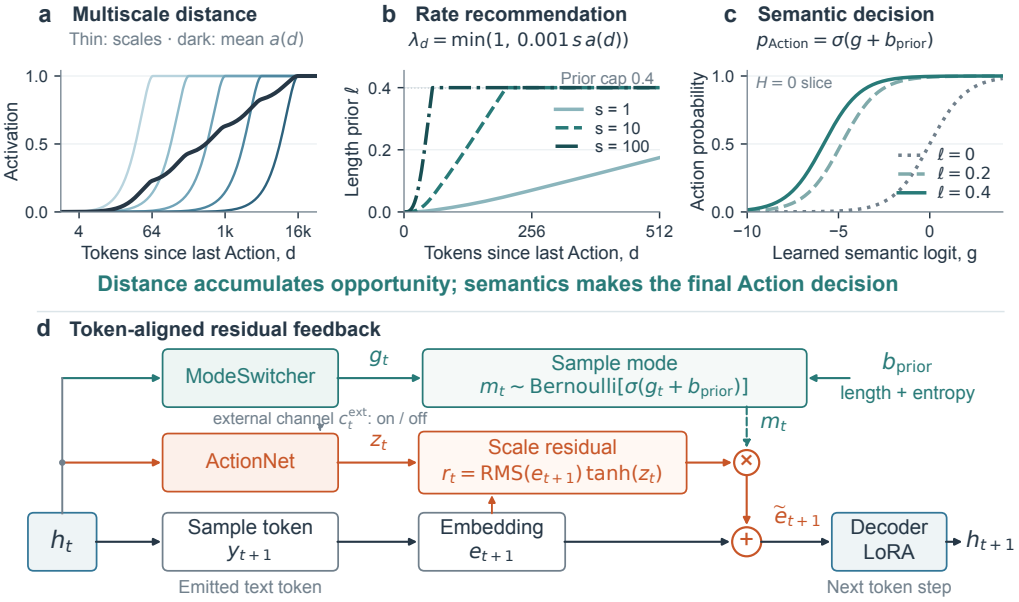

Figure 2: **LatentWeave.** (a) Five half-cosine distance features form the multiscale activation $a(d)$. (b) Accumulated hazard produces a length recommendation capped at 0.4; an Action resets distance and survival. (c) The prior shifts the semantic ModeSwitcher logit without fixing an Action quota. (d) ActionNet may condition on an optional external channel before adding an embedding-scaled residual to the next ordinary token input. No output marker or extra KV position is introduced. Decoder low-rank adaptation (LoRA), ModeSwitcher, and ActionNet are jointly post-trained.

3.1 RUBRIC RL: LEARNING WHAT TO RETAIN

Qwen3.8-Flash-Next assesses the image, question, answer, and Action-delimited reasoning as one trajectory. It assigns six scores, each 0 (not satisfied), 1 (partially satisfied), or 2 (fully satisfied).

Semantic-stage coverage checks necessary setup, operations, checkpoints, and commitment without counting repeated derivations as new stages. **Visible-backbone quality (VB)** checks whether the explicit plan, evidence, and transitions suffice. **Latent-operation alignment** assesses whether Action positions plausibly support updates. **Checkpoint grounding** requires concrete, checkable results after Actions. **Compression safety (CS)** checks that necessary premises survive without unsupported jumps. **Answer grounding (AG)** checks whether the visible reasoning supports the answer, separately from answer correctness.

For trajectory i, let z_{ik} denote these six scores and $f_i \in \{0, 1\}$ indicate rubric failure. The quality score is

$$v_i = \min\left\{ \frac{1}{12}\sum_{k=1}^{6} z_{ik}, \frac{z_i^{\mathrm{VB}}}{2}, \frac{z_i^{\mathrm{CS}}}{2}, \frac{z_i^{\mathrm{AG}}}{2}, 1 - \frac{3}{4} f_i \right\}. \tag{1}$$

The integrity caps prevent strong scores elsewhere from compensating for missing visible structure, unsafe compression, or unsupported answers. A zero in any integrity field sets $v_i = 0$. Quality

modulates one trajectory reward within separate correctness tiers; a correct-only length term favors shorter successful completions. Figure 4 displays the six individual scores, not v_i.

The shared trajectory advantage trains explicit generation, routing, and latent updates without separate segment rewards. Appendix A specifies score validation, reward ranges, the length term, and policy optimization.

3.2 Interleaved Latent Actions: Maintaining State

Removing text can discard information needed later. We add a learned state-feedback path to support continuation. Let h_t be the image-conditioned hidden state after consuming y_t, and E the token embedding. At an eligible reasoning position,

$$y_{t+1} \sim \pi_\theta^{\text{text}}(\cdot \mid h_t), \qquad e_{t+1} = E(y_{t+1}), \qquad m_t \sim \text{Bernoulli}(p_t). \tag{2}$$

The binary mode m_t chooses ordinary Token mode or Action mode. Both emit the ordinary sampled token. Action mode also modifies its input embedding:

$$\begin{aligned} r_t &= \rho(e_{t+1})\, \tanh\big(\text{ActionNet}_\phi(h_t; c_t^{\text{ext}})\big)\,, \\ \widetilde{e}_{t+1} &= e_{t+1} + m_t r_t, \end{aligned} \qquad \rho(e) = \sqrt{\max\left\{ d^{-1} \sum_{j=1}^{d} e_j^2, 10^{-12} \right\}}. \tag{3}$$

ActionNet uses normalized hidden features, a gated feed-forward block, and a zero-initialized output projection, with no additional residual scaling coefficient. It may receive external evidence c_t^{ext}, set to $\varnothing$ when disabled. Root mean square (RMS) scaling matches the local embedding scale, and $\tanh$ bounds $\text{RMS}(r_t) \le \text{RMS}(e_{t+1})$, up to the numerical floor.

The next Transformer forward pass consumes $\widetilde{e}_{t+1}$ and propagates its effect into hidden states and the KV cache. Since y_{t+1} is already sampled, the update affects y_{t+2} onward. It adds neither a latent marker nor a Transformer position. This feedback need not reconstruct omitted verbal steps.

3.3 Semantic Switching with Frequency Regulation

ModeSwitcher predicts an Action logit from the current hidden state. Actions must be sampled from initialization for ActionNet to receive downstream likelihood gradients. Shorter traces reduce eligible boundaries and accumulated distance, limiting these learning opportunities.

We combine this logit with a multiscale distance prior and a small token-entropy recommendation. The prior raises Action probability with distance and resets after an Action. Its scheduled scale maintains exploration as trajectories shorten, while the semantic logit supplies the state-dependent preference. The schedule imposes no Action quota. Appendix B gives the equations; Section 5.2 examines the motivating dynamics.

3.4 Efficient Joint Training and Inference

Joint training applies on-policy GRPO (Shao et al., 2024) to decoder LoRA (Hu et al., 2022) and the Bernoulli mode policy. Text and mode decisions share the trajectory advantage; deterministic ActionNet updates learn through the likelihood of subsequent tokens.

Batched replay records exact token identifiers (IDs), mode decisions, intervention positions, and ActionNet queries, then reconstructs all residual injections in one differentiable decoder pass. At inference, a length-T completion still uses T Transformer positions and KV entries, plus lightweight routing and ActionNet evaluations. The reported efficiency metric is therefore visible-token compression, not a measured wall-clock speedup. Appendix C details replay and computational costs.

4 Experiments

We evaluate six-benchmark accuracy and efficiency, then examine Standard RL's length plateau, segmented rubric feedback, and the contribution of the latent interface.

4.1 Evaluation Setting

Training. We initialize from Qwen3-VL-8B-Thinking and train on a curated subset of approximately 18,000 examples from VLAA-RL (Chen et al., 2025), obtained through cleaning and filtering based on CoT quality, answer validity, and problem difficulty. Four H200-144GB graphics processing units (GPUs) produce each on-policy batch of 256 trajectories from 32 prompts and eight completions per prompt. Training rollouts use an 8,192-token cap, temperature 1.0, top-p 1.0, disabled top-k, and presence penalty 0. Appendix D reports the training configuration.

Benchmarks and evaluation. The six-benchmark suite covers hard multimodal reasoning (Massive Multi-discipline Multimodal Understanding, MMMU (Yue et al., 2024); MathVision (Wang et al., 2024)), vision-dependent understanding (MMStar (Chen et al., 2024a), RealWorldQA (RWQA) (xAI, 2024)), and long-chain or multi-step reasoning (Multi-domain Multi-step Multimodal Chain-of-Thought, M^{3}CoT (Chen et al., 2024b); ScienceQA, SciQA (Saikh et al., 2022)). All our evaluations, including Standard RL, use the same Qwen3-VL Thinking sampling settings: a 40,960-token output cap, temperature 0.7, top-p 0.95, top-k 20, and presence penalty 1.5. We report pass@1 and total visible output tokens. Macro accuracy weights the six benchmarks equally, whereas mean token counts pool 12,914 questions. Answers are judged by `Qwen/Qwen3.8-Flash-Next`.

MathV-G100 diagnostics. MathV-G100 is a fixed set of 100 MathVision questions selected deterministically from question identifiers and text, independently of model outputs. All checkpoint comparisons, training controls, and inference interventions reuse this set and the same sampling, answer-scoring, and pass@k procedure. We generate eight completions per question and report pass@1/4/8, averaging over questions. Appendix D.1 specifies the selection algorithm and scoring rules. Section 5 explains why the full-suite compression results motivate this focused diagnostic.

4.2 Six-Benchmark Accuracy–Compression Frontier

Textual post-training trades length for retained accuracy. OneThinker and Standard RL reach different operating points: the former compresses more relative to Instruct ($3.55\times$), whereas the latter largely preserves Thinking accuracy at $1.92\times$. OneThinker is the only external text-only method here with complete six-benchmark coverage, but its different starting model prevents a controlled comparison. Standard RL supplies the same-backbone reference for testing whether further compression requires different training feedback.

LatentWeave offers a wider choice of inference budgets. Both reported checkpoints exceed the Instruct model's macro accuracy while using fewer tokens than Standard RL. Step 300 preserves more of the Thinking baseline's accuracy; step 750 nearly halves output again, at a further accuracy cost. The empirical frontier therefore supports budget-dependent selection rather than a single checkpoint that dominates at every budget. Figure 1 gives the cross-model context; Appendix Figure 5 shows the recorded same-backbone trajectories.

The accuracy cost depends on the reasoning task. Hard multimodal reasoning bears the largest loss. In contrast, both checkpoints exceed Thinking on M^{3}CoT and ScienceQA, and step 300 also improves RealWorldQA. Shortening therefore does not cause a uniform decline across the suite. This concentration of errors matters for checkpoint selection and motivates analyzing difficult visual reasoning rather than relying on the aggregate trade-off alone.

5 Analysis

5.1 From Full-Suite Compression Dynamics to MathV-G100

MathVision shows the largest accuracy decline as compression increases: 5.69 points between steps 300 and 750 (Table 1). We therefore use MathV-G100 to study difficult visual reasoning under compression. All following checkpoint comparisons, training controls, and interventions use the fixed set and shared protocol defined in Section 4. Appendix Table 5 lists all checkpoints.

Table 1: **Per-benchmark accuracy and total visible output tokens.** LatentWeave is reported at two inference-budget operating points.

		Hard multimodal		*Vision-dependent*		*Long-chain*			
Method	**Backb.**	**MMMU**	**MathV.**	**MMStar**	**RWQA**	**M^{3}CoT**	**SciQA**	**Avg.**	**Tokens**
Text-only reasoning									
Qwen2.5-VL	Q2.5-7B	58.60	25.10	63.90	68.50	71.53	88.55	62.70	312
Qwen3-VL	Q3-8B-I	69.60	53.90	70.90	71.50	70.83	92.00	71.46	4,995
	Q3-8B-T	74.10	62.70	75.30	73.50	71.31	91.23	74.69	3,649
OneThinker	Q3-8B-I	70.60	47.50	70.60	71.63	71.01	91.57	70.49	1,408
DeepVision†‡	Q3-8B-I	71.33	55.49	–	–	71.61	–	66.14†	3,173†
XMCC‡	Q3-8B-T	54.80	–	59.00	–	–	–	–	–
Standard RL (step-900)		**70.00**	**64.08**	**73.67**	**72.29**	**74.16**	**92.03**	**74.37**	**1,900**
Δ vs. Q3-8B-T		−4.10	+1.38	−1.63	−1.21	+2.85	+0.80	−0.32	1.92×
Latent reasoning									
LVR	Q2.5-7B	49.81	20.36	64.80	67.97	58.02	83.71	57.45	21
Monet	Q2.5-7B	51.24	22.66	61.13	68.40	59.32	82.00	57.46	280
SkiLa	Q2.5-7B	47.52	20.95	64.80	66.93	63.93	82.34	57.75	310
LaRe‡	Q3-8B-I	–	–	77.10	–	–	91.50	–	180
CoLT	Q3-8B-I	56.86	25.95	68.90	70.20	61.04	92.80	62.62	8
SwiReasoning	Q3-8B-T	68.39	59.44	74.73	72.30	68.90	83.02	71.13	3,491
LatentWeave (step-300)		**67.33**	**58.49**	**72.80**	**75.56**	**74.03**	**91.58**	**73.30**	**1,252**
Δ vs. Q3-8B-T		−6.77	−4.21	−2.50	+2.06	+2.72	+0.35	−1.39	2.92×
LatentWeave (step-750)		**68.09**	**52.80**	**72.51**	**73.25**	**72.84**	**91.42**	**71.82**	**660**
Δ vs. Q3-8B-T		−6.01	−9.90	−2.79	−0.25	+1.53	+0.19	−2.87	5.53×

Accuracy is reported in percent; Avg. is the unweighted benchmark mean. Backb. and MathV. denote backbone and MathVision. Tokens is total visible output length, summing thinking and answer tokens when reported separately. Gray scores indicate local measurements. Standard RL and LatentWeave use Q3-8B-T. †Average uses the three reported tasks. ‡No public checkpoint. Q2.5 and Q3 denote Qwen2.5-VL and Qwen3-VL; I and T denote Instruct and Thinking.

The checkpoint trajectory distinguishes single-sample accuracy from eight-sample solution coverage. Between steps 100 and 250, pass@1 falls as reasoning shortens, while pass@8 remains unchanged: correct solutions are still sampled for the same fraction of questions, but less consistently. Coverage remains close to this level at step 300 and then drops markedly by step 400. Both accuracy and coverage are therefore needed to assess reasoning retention under stronger compression.

5.2 From Length Saturation to Structured Feedback

The limitation of conventional RL feedback. Longer training alone does not sustain compression under Standard RL. After the length plateau begins around step 550, another 300 updates reduce output by only 2.0%, while accuracy continues to fluctuate (Figure 3a,b). The near-2× full-suite operating point thus reflects a plateau in the current recipe, rather than premature stopping.

Standard RL uses answer correctness and a length term. These signals assess the outcome and output length, but do not distinguish needed premises, error repair, and repeated derivations. Its plateau motivates rubric assessment along explicit segment boundaries.

Segmented assessment, trajectory-level reward. Action boundaries make the reasoning segments explicit to the judge. It checks whether each segment advances the solution and whether transitions follow from grounded intermediate results. Repeated derivations and unsupported jumps can receive different scores despite identical final correctness. These checks yield one trajectory reward, without independent segment rewards or local credit assignment.

Maintaining opportunities for segmented reasoning. The manual run illustrates why segmentation needs continued sampling support (Figure 3c–f). Actions are sampled from initialization, but

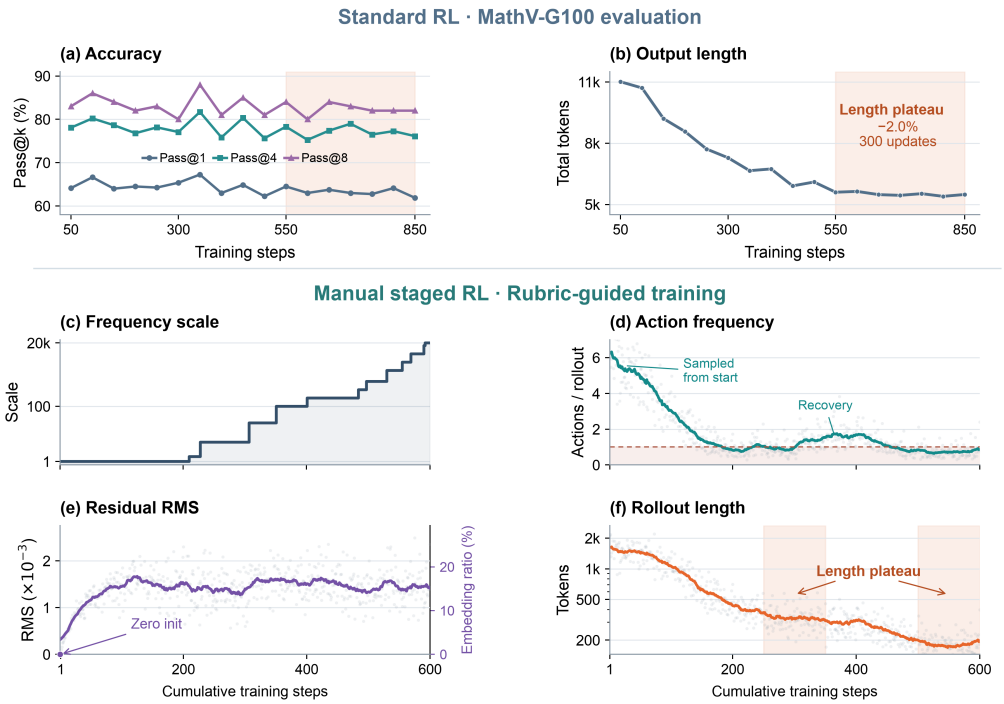

Figure 3: **Length saturation and rubric-guided training dynamics.** (a,b) Standard RL on MathV-G100, evaluated every 50 updates with eight completions per question: length falls only 2.0% from steps 550 to 850. (c–f) The first 600 updates of Manual staged RL relate scheduled scale, Action frequency, residual RMS, and rollout length; lines are 25-update means. Matching peach shading marks length plateaus. The two runs use distinct evaluation and rollout statistics. RMS is rescaled consistently across stages; the embedding ratio is approximate.

become sparse as trajectories shorten. A scale increase is followed by recovered Action frequency and renewed shortening, whereas later increases fail to sustain either. Frequency regulation is therefore motivated by the shrinking opportunity to execute and learn updates; increasing scale alone does not guarantee continued compression.

Shared shading links the plateaus across evaluation and rollout statistics. This historical run motivates the design without isolating frequency's causal effect. Appendix B.3 records the stages and RMS normalization.

5.3 Retaining Reasoning Beyond Textual Compression

Compression introduces a retention problem. The MathVision accuracy loss exposes the cost of further shortening. Segmented rubric feedback guides what remains visible; latent updates additionally condition continuation from reduced text. Table 2 compares the complete latent method with rubric-guided text generation.

Table 2: **LatentWeave versus Pseudo action on MathV-G100.** Tokens and Actions report mean output length and executed latent updates per completion.

Method	Steps	Pass@1	Pass@4	Pass@8	Tokens	Actions
Pseudo action	300	56.50	70.36	76.00	2,927	0
LatentWeave	300	58.25	75.46	83.00	3,357	87
LatentWeave	350	56.88	74.39	82.00	2,712	61

Textual control. Pseudo action adds judge-only boundaries at random intervals of 20–100 reasoning tokens, without continuous updates. Its output falls below the Standard RL plateau, showing that rubric-guided textual training enables further compression. We test whether latent feedback better retains reasoning at this compression level.

Equal training steps. At step 300, LatentWeave outperforms Pseudo action on all three pass@k measures, with a larger gain in solution coverage. Its longer traces, however, leave the contribution of additional inference length unresolved.

Comparable inference length. At step 350, LatentWeave generates less text than Pseudo action step 300 while retaining higher pass@1/4/8. The pass@1 margin is modest (0.38 points), but the pass@8 advantage remains six points. The coverage advantage persists at shorter output length, but requires 50 additional updates; the two comparisons separately address training budget and inference length, without matching both simultaneously.

These comparisons support the joint design's retention advantage, not continuous updates alone: learned versus random boundaries can also change rubric rewards. The following interventions test the trained policy's dependence on each component.

5.4 Module Ablations at Inference

Table 3 tests the external visual channel, latent updates, and Action routing by intervening on the same trained checkpoint under the shared evaluation protocol.

All three interventions reduce accuracy and coverage, supporting dependence on latent feedback and external visual access beyond the backbone. Randomized routing causes the largest loss, but changes both update pattern and placement, so it tests routing as a whole.

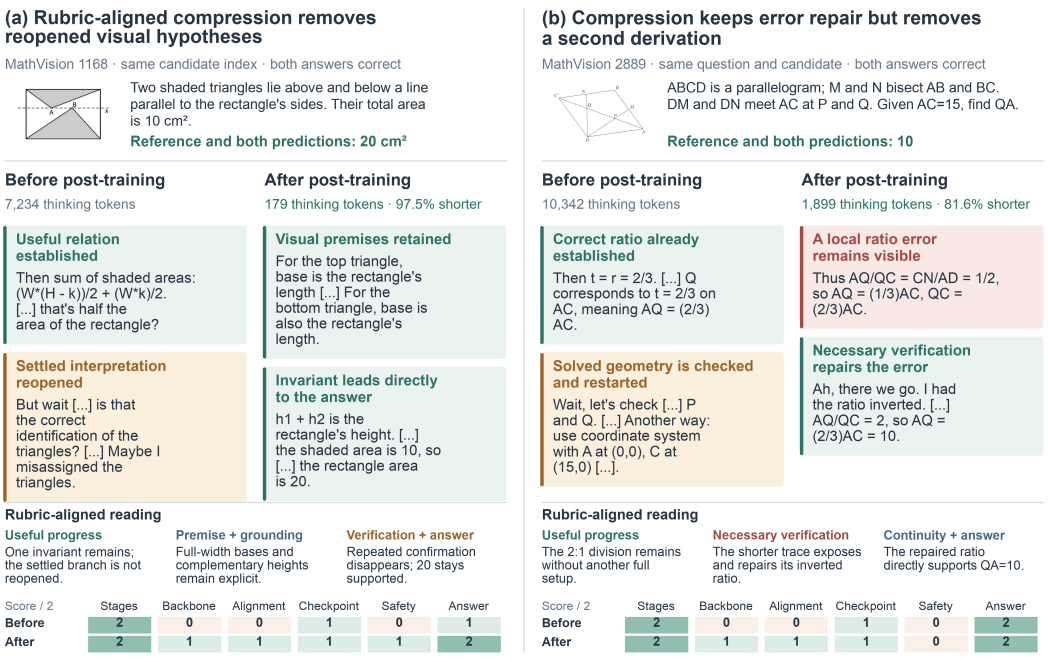

Figure 4: **What high-value compression retains.** Two MathVision pairs illustrate: (a) removing a reopened visual interpretation after establishing a sufficient invariant, and (b) retaining an error correction while omitting a later re-derivation. Excerpts are verbatim; [. . .] marks omissions. Rubric scores (0–2; Section 3.1) compare before and after post-training.

Table 3: **Inference interventions on LatentWeave step 300.** All rows evaluate the same checkpoint on MathV-G100; the unmodified policy is the reference.

Inference mode	Pass@1	Pass@4	Pass@8
LatentWeave	58.25	75.46	83.00
Visual channel off	55.98	72.67	78.00
Latent Action off	54.25	69.41	74.00
Random Action position	50.29	63.84	70.00

5.5 What High-Value Compression Retains

Figure 4 distinguishes useful verification from redundant reconsideration. Panel (a) retains a sufficient area invariant without reopening the visual interpretation. Panel (b) retains ratio-error detection and repair while omitting a second derivation. The rubric favors checks that resolve uncertainty over repeated confirmation.

The rubric scores in Figure 4 support this reading at case level. Both pairs retain full semantic-stage coverage and improve visible-backbone quality and operation alignment. Question 1168 also improves compression safety and answer grounding. Checkpoint grounding remains partial in both cases. Question 2889 retains zero compression safety, so its improved scores do not increase the capped quality reward. These independently generated, jointly correct pairs illustrate selective retention, without establishing its prevalence or attributing it to the rubric or a particular Action.

6 Discussion and Conclusion

LatentWeave couples the selection of explicit reasoning with continuous feedback for subsequent generation. Rubric-guided reinforcement learning assesses the contribution of reasoning segments, while Interleaved Latent Actions provide a learned feedback path without adding sequence positions. Joint optimization aligns what the model verbalizes with where and how it updates its state. Across six visual reasoning benchmarks, the resulting policies offer distinct accuracy–efficiency operating points for different inference budgets. Together, these results support jointly learning explicit reasoning and latent continuation as an approach to efficient multimodal reasoning.

Learned compression across reasoning tasks. Under the same training recipe, the model learns which reasoning to retain without task-specific compression rules. The benchmark groups reveal the resulting differences: hard multimodal reasoning incurs the largest accuracy cost, long-chain and multi-step reasoning retains performance more readily, and vision-dependent understanding shows mixed responses. These patterns suggest that compressibility depends on the role of intermediate reasoning rather than text length or visual dependence alone. They also expose a remaining challenge: preserving the precise visual premises and coupled deductions needed for harder problems.

Computational scope. Matrix-operation counts give 15.14 billion floating-point operations (FLOPs) per backbone decoding token and 2.22 GFLOPs per Action for a reference input with 1,024 visual tokens, including repeated visual key/value projections. At the Action frequencies in Table 2, routing and Actions add approximately 0.33–0.38% to same-length backbone dense decoding cost. Each multiply-add counts as 2 FLOPs. Visual cost scales linearly with visual-token count. This estimate excludes prefill, elementwise operations, and padding; Appendix C.3 specifies the accounting. No positions are added.

AI USE STATEMENT

We used generative AI tools to assist with writing up mathematical proofs, provide feedback on the research methodology and experiments, implement parts of the method, and create or edit code. We also used a generative AI model as a judge to assess reasoning trajectories. Generative AI tools assisted with figures and with organizing, drafting, and polishing the manuscript. We did not use them to generate synthetic data, develop theoretical models or conceptual frameworks, originate mathematical claims or proof arguments, propose research hypotheses, translate material, clean or reformat datasets, conduct qualitative analysis, or interpret experimental results. The authors reviewed and verified all AI-assisted work and take responsibility for the final content.

ETHICS STATEMENT

This work studies the efficiency of visual reasoning on existing research benchmarks. Compression reduces the visibility of intermediate computation and can remove necessary visual checks; a shorter answer is therefore not evidence of a more reliable decision. The reported results do not establish suitability for high-stakes use.

REPRODUCIBILITY STATEMENT

Section 3 defines the method. Appendices A–C specify the rubric and reward, routing and residual construction, replay gradient boundary, and FLOPs accounting. Appendix D records hardware, sampling settings, deterministic MathV-G100 selection, and answer scoring. Appendices E–G document checkpoint trajectories and paired-trace provenance; Appendix H gives the efficiency normalization and reference counts.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

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

Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. Sft or rl? an early investigation into training r1-like reasoning large vision-language models. *arXiv preprint arXiv:2504.11468*, 2025.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *Advances in Neural Information Processing Systems*, 37:27056–27087, 2024a.

Qiguang Chen, Libo Qin, Jin Zhang, Zhi Chen, Xiao Xu, and Wanxiang Che. M^3cot: A novel benchmark for multi-domain multi-step multi-modal chain-of-thought. *arXiv preprint arXiv:2405.16473*, 2024b.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024c.

Evelina Fedorenko, Steven T Piantadosi, and Edward AF Gibson. Language is primarily a tool for communication rather than thought. *Nature*, 630(8017):575–586, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Hanyu Lin, Min Cai, Jiawei Wen, and Haodi Zhang. Tyler: Typed latent reasoning for language models–when to think, what to compute, and how much to allocate. *arXiv preprint arXiv:2606.16360*, 2026.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6025–6035, 2025.

Tanik Saikh, Tirthankar Ghosal, Amish Mittal, Asif Ekbal, and Pushpak Bhattacharyya. Scienceqa: A novel resource for question answering on scholarly articles. *International Journal on Digital Libraries*, 23(3):289–301, 2022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, 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, Xiangxi Shi, Pu Zhao, Yanzhi Wang, and Jiuxiang Gu. Efficient reasoning with hidden thinking. *arXiv preprint arXiv:2501.19201*, 2025a.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025b.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Yuntian Tang, Bohan Jia, Wenxuan Huang, Lianyue Zhang, Jiao Xie, Wenxi Li, Wei Li, Jie Hu, Xinghao Chen, Rongrong Ji, and Shaohui Lin. Towards efficient large language reasoning models via extreme-ratio chain-of-thought compression. *arXiv preprint arXiv:2602.08324*, 2026.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 37:95095–95169, 2024.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning, 2025. URL https://arxiv.org/abs/2506.01939.

xAI. RealworldQA: A benchmark for real-world visual understanding. https://huggingface.co/datasets/xai-org/RealworldQA, 2024.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3351–3363, 2025.

Xin Xu, Tong Yu, Xiang Chen, Haoliang Wang, Julian McAuley, and Saayan Mitra. Thinkrouter: Efficient reasoning via routing thinking between latent and discrete spaces. *arXiv preprint arXiv:2602.11683*, 2026.

Jiayu Yang, Chao Chen, Shengen Wu, Yinhong Liu, Yuxuan Fan, Lujundong Li, Songning Lai, Chengwei Qin, and Zhijiang Guo. Demystifying hidden-state recurrence: Switchable latent reasoning with on-policy reinforcement learning. *arXiv preprint arXiv:2606.13106*, 2026.

Jingyang Yi, Jiazheng Wang, and Sida Li. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. *Advances in Neural Information Processing Systems*, 38: 39011–39043, 2026.

Xinlei Yu, Zhangquan Chen, Yongbo He, Tianyu Fu, Guanting Dong, Cheng Yang, Chengming Xu, Yue Ma, Xiaobin Hu, Zhe Cao, et al. The latent space: Foundation, evolution, mechanism, ability, and outlook. *arXiv preprint arXiv:2604.02029*, 2026.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9556–9567, 2024.

Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei Han, and Dong Wang. Hybrid latent reasoning via reinforcement learning. *Advances in Neural Information Processing Systems*, 38:5501–5530, 2026.

Kongcheng Zhang, Qi Yao, Baisheng Lai, Jiaxing Huang, Wenkai Fang, Dacheng Tao, Mingli Song, and Shunyu Liu. Reasoning with reinforced functional token tuning. *arXiv preprint arXiv:2502.13389*, 2025.

Zhenyu Zhao, Sander Land, Daniel M Bikel, and Waseem Alshikh. Shorthand for thought: Compressing llm reasoning via entropy-guided supertokens. *arXiv preprint arXiv:2604.26355*, 2026.

Jiajie Zou, David Poeppel, and Nai Ding. Constituent-constrained word prediction during language comprehension. *Nature Neuroscience*, pp. 1–12, 2026.

A RUBRIC RL: ASSESSMENT AND OPTIMIZATION

A.1 SCORE VALIDATION AND REWARD CONSTRUCTION

The implementation validates the six integer scores defined in Section 3.1 before applying Equation 1. The rubric-failure flag caps quality at 0.25.

Let $c_i \in \{0, 1\}$ be answer correctness after structural validation. Empty, malformed, or degenerate-repetition outputs receive $c_i = 0$ and zero rubric fields. The main reward is

$$R_i^{\text{rubric}} = \begin{cases} 4 + 4v_i, & c_i = 1, \\ -6 + v_i, & c_i = 0. \end{cases} \tag{4}$$

The resulting reward lies in $[4, 8]$ for correct answers and $[-6, -5]$ for incorrect answers.

For group G, correct subset G^+, and unpadded response length L_i, the correct-only adjustment is

$$\Delta A_i = -\alpha\, \mathbf{1}[i \in G^+] \left[\log(L_i + 1) - \frac{1}{|G^+|} \sum_{j \in G^+} \log(L_j + 1) \right], \qquad \alpha = 0.1. \tag{5}$$

We set $\Delta A_i = 0$ when G^+ is empty. The adjustment sums to zero among successful members of a group, and incorrect samples receive no direct shortening reward. The separately recorded Action-count preference is

$$C(n) = \begin{cases} 1, & 0 \le n \le 16, \\ 1 - 2(n - 16)/n, & n > 16. \end{cases} \tag{6}$$

The implementation supplies $C(n)$ as a separate deterministic reward column when count regularization is enabled, before reward normalization. It is not part of v_i or the judge's assessment, and it does not enter the post-normalization length adjustment. Constant count scores within a group supply no relative preference. We use `Qwen/Qwen3.8-Flash-Next` for holistic assessment of the image, question, visible trajectory, and extracted answer. Action locations and counts are reconstructed from the rollout trace; the generated text contains no latent markers.

A.2 JOINT POLICY OPTIMIZATION AND REWARD CONTROLS

For trajectory $\tau = (y_{1:T}, m_{\mathcal{E}})$, the joint token–mode likelihood is

$$\log \pi(\tau \mid x) = \sum_t \log \pi_\theta^{\text{text}}(y_{t+1} \mid h_t) + \sum_{t \in \mathcal{E}} \log \pi_\psi^{\text{mode}}(m_t \mid h_t, \ell_t, u_t). \tag{7}$$

We optimize this policy with on-policy GRPO (Shao et al., 2024). Group-normalized trajectory advantages apply to text-token likelihood ratios and Bernoulli mode likelihood ratios at eligible recorded events. Both $m = 0$ and $m = 1$ decisions enter the objective. ActionNet is deterministic and learns through downstream token likelihoods conditioned on its residual. Each rollout batch is generated by the current policy and consumed once for its update.

Advantage and loss reduction. Let A_i^{group} denote the advantage after reward normalization. The correct-only length term is then added as $A_i = A_i^{\text{group}} + \Delta A_i$, without another normalization of ΔA_i. This same A_i is broadcast to the text decisions and eligible mode events of trajectory i. The implementation reduces text loss over valid response tokens within each trajectory and then across trajectories. Mode loss is averaged over eligible events in the batch, including both executed and unexecuted Actions, and added with unit coefficient. Thus, the two factors share a trajectory signal but have separate normalization denominators. Padding, ineligible boundaries, and terminal events do not enter the mode denominator.

Textual control definitions. Standard RL generates ordinary thinking traces using only answer-correctness rewards and a length term, without rubric assessment, a latent path, or synthetic boundaries. Pseudo action retains ordinary generation but submits randomly segmented text to the six-field rubric. Successive boundary intervals are sampled uniformly from the integers 20–100, stopping before the reasoning region ends; the number of boundaries therefore depends on trace length. The

deterministic generator uses seed $0 + 1{,}000{,}003s + 9{,}176i$, where s is the training step and i the sample index within the batch. Markers appear only in the judge input, not the generated or optimized token sequence. Latent-specific structure and count rewards are disabled for this textual control. Thus, it tests rubric-guided textual compression rather than isolating the residual while holding every reward input fixed.

The main-text controls assess the resulting behavior from complementary directions. Pseudo action provides synthetic boundaries for the same rubric judge to assess a textual trajectory without executing continuous updates. It preserves the judge-facing interleaving format needed by the rubric. Inference interventions instead remove the external channel, remove the latent update, or randomize its placement on one trained checkpoint. These controls compare rubric-guided textual compression with dependence on the learned interface; judge scores are not independent evidence.

B Exact Routing Features and Residual Construction

B.1 Eligibility and Boundary Calibration

An Action can be executed only at a nonterminal position inside the reasoning region, when the next ordinary token will actually be consumed. Thinking delimiters and answer tokens are excluded. The first response token has no preceding response-level Action. These rules attach every intervention to an existing decoding step and avoid updates with no subsequent consumer.

The historical calibrated initialization used for the paired cases uses compressed reasoning text with boundary annotations outside the token sequence. A removed span supplies a positive boundary at the last retained token before the gap, provided a valid next token exists. Binary cross-entropy trains ModeSwitcher at response positions. The decoder and ActionNet are frozen, and Action execution is disabled. This supplies an initial switching policy without teaching the decoder to emit special markers or assuming that the annotated boundaries remain optimal during joint training.

B.2 Multiscale Frequency Features

Length activation. Let d count thinking-content tokens since the last executed Action. With deadzone $d_0 = 2$ and scales $\mathcal{K} = \{64, 256, 1024, 4096, 16384\}$, the monotone distance feature is

$$a(d) = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \frac{1 - \cos(\pi \min\{\max(d - d_0, 0)/k, 1\})}{2}. \tag{8}$$

It increases smoothly across several distance scales and saturates at one. The Action-count decay is

$$D(n) = \left(1 + \frac{\max(n - 16, 0)}{4}\right)^{-1/2}. \tag{9}$$

With schedule scale s, the hazard, survival, and length recommendation are

$$\lambda_t = \min(1, 0.001\, s\, a_t), \qquad S_t = S_{t-1}(1 - \lambda_t), \qquad \ell_t = \min\{0.4, (1 - S_t)D(n_t)\}. \tag{10}$$

The first 16 Actions leave the length recommendation unmodified. This is not a hard Action budget. After an executed Action, S resets to one and d to zero. Thinking delimiters and answer tokens do not advance this state. The two-token deadzone leaves semantic routing active.

Entropy feature. The runtime obtains the current policy's top-20 token probabilities $p_1, \ldots, p_{20}$. It aggregates the remaining mass as $p_{\text{tail}} = 1 - \sum_{j=1}^{20} p_j$. The normalized entropy is

$$H = -\frac{\sum_{j=1}^{20} p_j \log p_j + p_{\text{tail}} \log p_{\text{tail}}}{\log 21}, \tag{11}$$

with $0 \log 0 = 0$. This is instantaneous entropy, without a moving-average history. It supplies the fixed probability component $u = 0.1H$. Both u and ℓ are detached auxiliary features of the mode policy; the semantic head supplies its trainable component. Entropy measures current uncertainty and is not interpreted as the causal value of the corresponding text.

Let g_t be the semantic ModeSwitcher logit and $p_0 = 0.002$ the reference probability. The fixed recommendations combine as

$$q_t = 1 - (1 - p_0)(1 - u_t)(1 - \ell_t), \qquad p_t = \sigma(g_t + \mathrm{logit}(q_t) - \mathrm{logit}(p_0)) \,. \tag{12}$$

Only one Bernoulli mode is sampled. The 0.4 cap applies to ℓ_t, not to the final Action probability.

Prior interpretation. If $g = \mathrm{logit}(p_0)$, Equation 12 yields $p = q$: the mode policy reproduces the fixed recommendation at its reference semantic logit. A lower learned g suppresses the recommendation and a higher g reinforces it. Large hazard scales accelerate survival accumulation but do not remove this semantic dependence. Increasing the hazard scale does not prescribe an Action count.

B.3 Provenance of the Frequency Dynamics

Figure 3c–f uses the first 400 updates of Manual staged RL followed by 200 updates of its continuation from step 400, with an optimizer restart at the stage boundary. The horizontal axis is cumulative training updates. The plotted Action counts and lengths are rollout statistics from this lineage, not MathV-G100 evaluation statistics or the delivered LatentWeave training trajectory. The scale is an externally specified schedule. Adaptive routing refers to the state-dependent Bernoulli decisions conditioned on the semantic logit, entropy, and accumulated distance.

Each curve is smoothed once over the combined 600-update sequence using a centered 25-update window, shortened at the endpoints. The first stage logged residual RMS after an additional factor of 0.25, which was removed in the continuation. We divide first-stage RMS values by 0.25 to express both stages under the continuation's parameterization. The plotted RMS therefore compares normalized residual magnitudes rather than the original executed magnitudes across the restart. An independent step-191 measurement associates RMS 1.09×10^{-3} with 11.68% of embedding RMS, supplying the approximate ratio axis. The simultaneous training and stage changes preclude treating this trajectory as a controlled frequency ablation.

B.4 Embedding-Scaled Action Construction

ActionNet uses a normalized query, a gated feed-forward residual block, a final normalization, and a zero-initialized output projection. The resulting output passes through $\tanh$ before the scaling in Equation 3. The optional external input is present in the reported LatentWeave configuration and is set to $c_t^{\mathrm{ext}} = \varnothing$ in the channel-off intervention.

The RMS bound controls intervention scale while preserving the ordinary embedding as an additive reference. The external channel is a generic input path rather than a fixed architectural assumption; the reported instantiation supplies visual information. Table 3 tests whether the trained policy depends on this query-specific continuous update.

Visual instantiation of the external channel. The saved LatentWeave configuration reads the native keys and values at visual prompt positions from the first language-model layer (zero-based layer 0). Keys are transformed back from their cached multimodal rotary coordinates before use. Each visual position supplies eight key/value heads of dimension 128. The evidence adapter flattens these heads, independently RMS-normalizes keys and values, and projects each to width 512. A query projected from the normalized reasoning hidden state attends to these visual positions with eight attention heads. The attended summary is projected to hidden width 4,096 and added to the ActionNet input before its gated feed-forward block, whose intermediate width is 1,024. Padding is masked, and a missing or disabled external channel contributes no summary. Disabling this channel retains the backbone's original image-conditioned computation.

Trainable parameters. The decoder uses LoRA rank 8, scaling parameter 16, and dropout 0.05 on its query, key, value, attention-output, and feed-forward gate/up/down projections. ModeSwitcher and ActionNet are jointly trained alongside these adapters. The visual key/value projections and summary projection belong to ActionNet. Its final hidden-to-hidden projection is initialized to zero, so the initial residual vanishes even when the external channel is enabled.

C Efficient Training and Inference

Replay preserves recorded token and Action alignment while batching the residual computation. Below we specify its gradient boundary and costs.

C.1 Token-Aligned Generation

An event is anchored to a carrier token and the immediately following consume token. Positions are completion-relative before collation and are mapped to the padded training tensor using the response offset. Replay checks adjacency, attention-mask validity, and the consumed token ID. Text likelihoods retain the exact rollout token IDs and sampled terminal token when present. The trace also retains mode probabilities and the fixed auxiliary features used at eligible events.

Algorithm 1 One token-aligned latent intervention

Require: Current hidden h_t, optional evidence c_t^{ext}, token policy, routing state (S, d, n)
1: Sample $y_{t+1} \sim \pi^{\text{text}}(\cdot \mid h_t)$; set $e \leftarrow E(y_{t+1})$
2: **if** y_{t+1} terminates the request or will not be consumed **then**
3: Finalize the ordinary token trace without a pending residual
4: **return**
5: **end if**
6: **if** t is an eligible reasoning position **then**
7: Compute a_t, λ_t, ℓ_t, u_t, and p_t using Equations 8–12
8: Sample $m_t \sim \text{Bernoulli}(p_t)$
9: Set $\delta_t \leftarrow 0$; if $m_t = 1$, compute $\delta_t \leftarrow \text{RMS}(e)\tanh(\text{ActionNet}(h_t; c_t^{\text{ext}}))$
10: Record carrier/consume positions, token ID, m_t, and old mode log probability
11: If $m_t = 1$, record the Action query; reset (S, d) and increment n
12: **else**
13: Set $\delta_t \leftarrow 0$; create no executable Action event
14: **end if**
15: Consume $e + \delta_t$ in the next ordinary Transformer forward
16: Obtain the next hidden state and ordinary KV entry

The runtime commits an intervention only when the next ordinary token will actually be consumed. Requests terminated by an end-of-sequence (EOS) token, stop conditions, or scheduler decisions must not retain a pending residual. When interrupted decoding is reconstructed, historical residuals must be reapplied at the same consume positions. Ordinary token history alone does not determine the acted KV cache. Text and mode sampling use separate random-number streams so that zero residuals do not alter the text-sampling stream merely by drawing mode decisions.

C.2 Batched Replay and Its Gradient Boundary

Batched replay injects residuals from saved on-policy queries in one decoder pass.

For saved rollout features $\bar{h}_t$ and $\bar{c}_t^{\text{ext}}$ with gradients stopped, batched replay evaluates

$$r_t = \rho(e_{t+1}) \tanh\big(\text{ActionNet}_\phi(\bar{h}_t; \bar{c}_t^{\text{ext}})\big) . \tag{13}$$

Gradients update ActionNet, including its evidence projections, through subsequent token likelihoods. They do not pass back through the rollout computation that produced the query or visual evidence. ModeSwitcher likewise receives saved detached query features. Decoder LoRA still receives gradients from the replayed text likelihood. This is a surrogate with fixed rollout features rather than full differentiation through the recurrent sequence of Action queries.

Policy versioning. A rollout batch uses one joint policy snapshot, and training consumes that batch before updating the snapshot. Exact token-ID replay avoids text-normalization drift, while policy fingerprints reject mismatched versions. Replay additionally verifies carrier–consume adjacency, attention-mask validity, and consumed token IDs before applying an intervention.

C.3 Cost Accounting

Let $F(T)$ denote the cost of ordinary Transformer decoding for a completion of length T, including attention to its prefix. Let C_M and C_A be the costs of one mode evaluation and one ActionNet evaluation, and let $N \leq T$ be the number of executed Actions. Ignoring shared input preprocessing, a simple operation-count model is

$$C_{\text{normal}}(T) = F(T), \qquad C_{\text{LatentWeave}}(T, N) = F(T) + O(TC_M + NC_A). \tag{14}$$

At fixed output length, routing and Actions add computation without changing the backbone attention pattern or KV-cache length. Shorter completions reduce full decoder steps.

Training stores $O(Nd)$ Action-query state per trajectory alongside token, mode, and alignment records. Batched replay computes all residuals in one pass under the stated gradient boundary.

Counting convention and execution path. We derive the estimate from the saved backbone and latent-module dimensions and the executed inference branches. Each multiply–add counts as two FLOPs. ModeSwitcher runs at eligible reasoning boundaries; ActionNet and visual attention run only for sampled Actions. Native visual keys and values are cached per request, but the evidence adapter recomputes their projections at each Action. The count therefore includes repeated visual projection rather than assuming an unimplemented projection cache.

Backbone reference. The language model has 36 layers, hidden width 4,096, multilayer perceptron (MLP) width 12,288, 32 query heads, and eight KV heads of width 128. Per generated token, the query/key/value/output projections cost 83.89 million FLOPs (MFLOPs) per layer, and the gated MLP costs 301.99 MFLOPs per layer. Summing all layers and adding the 4,096-to-151,936 vocabulary projection, which costs 1.245 GFLOPs, gives 15.136 GFLOPs per token. This is the dense decoding reference used in the percentage calculation.

Additional computation. The router's 4,096-to-one projection costs 8,192 FLOPs per eligible token. ActionNet maps 4,096 features to two 1,024-wide branches and projects their gated product back to width 4,096, costing 25.17 MFLOPs. Its final 4,096-to-4,096 projection adds 33.55 MFLOPs, for 58.72 MFLOPs per Action without visual feedback. With visual feedback, the 4,096-to-512 query projection and 512-to-4,096 summary projection add 8.39 MFLOPs. Each visual token supplies 1,024 native key and value features, each projected to width 512. These two projections cost 2.097152 MFLOPs per visual token; the query–key and attention–value products add 2,048 FLOPs. The resulting visual contribution is 2.0992 MFLOPs per visual token per Action. At the reference input size of 1,024 visual tokens, the total is 2.2167 GFLOPs per Action.

Amortization over a completion. Table 2 reports means of 87 Actions per 3,357 output tokens at step 300 and 61 Actions per 2,712 tokens at step 350. For each checkpoint, we multiply the per-Action cost by its mean Action count, add the routing cost, and divide by the dense backbone cost at the same mean output length. We conservatively charge routing to every output token, including answer tokens. This gives 0.38% and 0.33% additional matrix FLOPs, respectively.

The 1,024-visual-token input is a reference scenario, not a measured benchmark mean. Visual cost grows linearly with input size; at 4,096 visual tokens the corresponding overheads are 1.48% and 1.28%. The backbone denominator excludes context-dependent self-attention products. Counts also exclude input prefilling and vision encoding, elementwise operations, batch padding, data movement, and unmerged LoRA overhead. These estimates describe unpadded matrix arithmetic and do not predict runtime latency.

D Experimental Protocol and Training Stages

Table 4 lists the shared configuration. DP denotes data parallelism and TP tensor parallelism.

Table 5 reports one LatentWeave training trajectory; Table 1 selects two full-suite operating points. These checkpoint evaluations are distinct from the same-checkpoint interventions in Table 3.

Table 4: Training and evaluation configuration.

Item	Setting
Backbone	Qwen3-VL-8B-Thinking
Training data	Curated VLAA-RL subset (approximately 18k examples)
Trainable modules	Decoder LoRA, ModeSwitcher, ActionNet
Training hardware	$4\times$ NVIDIA H200-144GB
Training batch	4 GPUs $\times 4$/device $\times 16$ accumulation $= 256$ trajectories (32 prompts $\times 8$)
Training limit	8,192 output tokens
Training sampling	Temperature 1.0; top-p 1.0; top-k disabled; presence penalty 0
Replay	Batched injection from saved rollout queries
Routing scale	Scheduled from 1 to 20,000
Evaluation hardware	$6\times$ NVIDIA RTX 4090D (DP $= 6$, TP $= 1$)
Evaluation limits	40,960 output tokens; 49,152-token context
Evaluation sampling	Temperature 0.7; top-p 0.95; top-k 20; presence penalty 1.5
Judge	`Qwen/Qwen3.8-Flash-Next`; $2\times$ RTX PRO 6000 Blackwell-96GB

Hazard schedule. Table 5 evaluates each delivered-run checkpoint with its training-time routing scale. Figure 3 instead reports the earlier Manual staged RL lineage that motivates frequency regulation. Its scale changes are not independent ablations, and its rollout statistics are not pooled with the MathV-G100 checkpoint evaluations.

Generation limits. All formal evaluations, including Standard RL and MathV-G100, use the evaluation settings in Table 4. Training sampling differs; historical 8,192-token diagnostics are excluded.

D.1 MathV-G100: Fixed Question Set and Shared Scoring

Deterministic selection. We load the full MathVision tab-separated values (TSV) file used by the evaluation code. The loader specifies the Message-Digest Algorithm 5 (MD5) checksum `93f6de14f7916e598aa1b7165589831e`. For each row, the sampler concatenates its `index` and `question` fields without a separator, encodes the resulting string in the 8-bit Unicode Transformation Format (UTF-8), and computes its MD5 hexadecimal digest. It sorts rows by this digest, using original row order to break ties, takes the first 100 rows, and restores their original dataset order. This rule defines one reproducible, fixed question set without a random sampling seed. It uses neither difficulty strata nor model predictions, correctness, or response length. The same selected identifiers are reused across checkpoints and controls.

Shared generation and answer scoring. Each selected question receives eight completions under the common evaluation settings above. Question selection is separate from answer eligibility: under the Thinking answer format, the scorer uses the nonempty answer after the final `</think>` delimiter. A completion without this answer is marked incorrect and remains in the candidate count. The scorer does not search unfinished reasoning for a substitute answer. Reaching the token limit does not automatically reject a completion that already contains an eligible answer. Every eligible answer follows the same MathVision extraction and correctness-checking procedure. The eight-candidate evaluation entry point checks the candidate count for each question before aggregation.

Pass@k. For eight completions and c_i correct samples on question i, the diagnostic uses the standard without-replacement estimator

$$\text{pass@}k = \frac{1}{100}\sum_{i=1}^{100}\left[1 - \frac{\binom{8-c_i}{k}}{\binom{8}{k}}\right], \qquad k \in \{1, 4, 8\}, \tag{15}$$

where the numerator is zero when $8 - c_i < k$. In particular, pass@8 is the fraction of questions with at least one correct completion among eight draws. The eight completions from a question are not treated as eight independent benchmark items.

E RECORDED SAME-BACKBONE ACCURACY–COMPRESSION TRAJECTORIES

Figure 5 compares the recorded six-benchmark trajectories within the Qwen3-VL-8B family under a 40,960-token evaluation limit. The Standard RL point provides the textual reference; the other curves show how compression and accuracy change along distinct training lineages. Checkpoints are linked only within their own lineage, so this comparison describes operating ranges rather than isolating component effects at matched training steps.

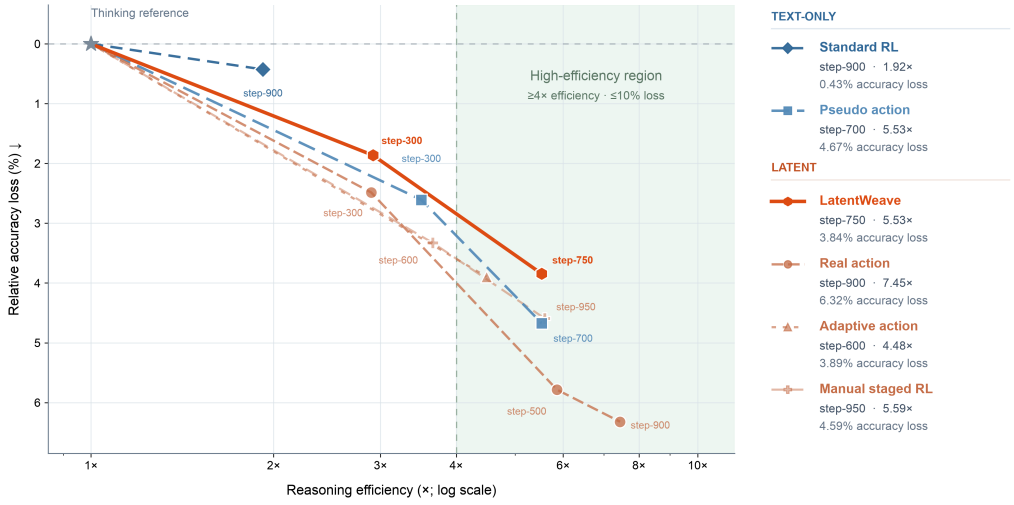

Figure 5: **Recorded full-suite accuracy–efficiency trajectories.** All points cover six benchmarks under a 40,960-token output cap. Efficiency and relative macro-accuracy loss use the Thinking reference. Standard RL (step-900) anchors the comparison at $1.92\times$ efficiency and 0.43% loss. Lines connect the reference and recorded checkpoints within each lineage; step labels do not imply matched training conditions across lineages. Blue curves denote text-only methods; orange curves denote latent methods, with LatentWeave emphasized by a saturated solid line. The key reports final plotted checkpoints. Real action disables visual feedback; shading marks efficiency above $4\times$.

F LatentWeave Training Trajectory on MathV-G100

This appendix reports the complete recorded LatentWeave trajectory used by the main-text analysis. Each checkpoint is evaluated on the same MathV-G100 questions under the same 40,960-token limit.

Table 5: **LatentWeave training trajectory on MathV-G100.** Each checkpoint is evaluated on the same 100 questions with eight completions per question and its training-time routing scale. Pass@1/4/8 are percentages; Think and Ans. are mean thinking and answer token counts.

Updates	Pass@1	Pass@4	Pass@8	Think	Ans.	Actions
50	65.4	80.0	84	10,357	615	127
100	66.9	79.8	84	9,241	585	171
200	63.5	78.3	84	5,631	422	165
250	61.6	75.8	84	3,926	359	126
300	58.2	75.5	83	3,097	259	87
350	56.9	74.4	82	2,507	205	61
400	54.5	69.3	75	2,095	173	43

G Paired Trace Provenance and Rubric Scores

G.1 Source and Selection for Figure 4

We paired saved MathV-G100 completions by question and candidate index before and after joint post-training. Panel (a) uses question 1168, completion 6 of 8; both outputs answer 20 and are judged correct. Panel (b) uses question 2889, also completion 6 of 8; both outputs answer 10 and are judged correct. Question and image identifiers match within each pair. Selection targets joint correctness and complementary reasoning patterns, not population representativeness. Candidate indices do not guarantee matched random streams. The calibrated initial policies differ from the LatentWeave trajectory reported in Table 5.

Token, Action, and excerpt alignment. Each output matches one saved rollout trace. IDs before the first `</think>` give 7,234 and 179 tokens for panel (a), and 10,342 and 1,899 for panel (b), agreeing with retokenization. For panel (b), full traces contain 11,782 and 2,250 IDs including termination; its 114 and 25 Actions are each applied to the next token. Images are uniformly resized without content edits. Verbatim excerpts retain source order, with ellipses marking gaps. Question summaries, headings, highlights, and the rubric-aligned readings are analyst annotations. The numerical rubric scores are separate assessments of the paired traces. In panel (a), both traces preserve the full-width bases and complementary heights that make the shaded area half the rectangle area. In panel (b), the preserved geometry is $AD \parallel CN$, $AD = 2CN$, hence $AQ/QC = 2$; the short trace still inverts and repairs this ratio.

G.2 Case-Level Rubric Scores

The score columns in Figure 4 follow the six-criterion order in Section 3.1, which defines their scale and aggregation.

The scores assess these selected traces, not pattern prevalence or the causal contribution of an Action. Under Equation 1, question 2889 has zero quality reward before and after training because compression safety remains zero. Its retained correction illustrates behavior, not evidence that the aggregate reward favored this change.

H Reference Normalization for the Efficiency Landscape

Figure 1 contextualizes the task-level comparison in Table 1 using each method's collected reference. For a method with benchmark subset B, the vertical coordinate is $100[1 - \overline{A}_B^{\text{method}} / \overline{A}_B^{\text{reference}}]$; the horizontal coordinate is reference total visible output tokens divided by method total visible output tokens. All methods cover six benchmarks except DeepVision, which uses MMMU, MathVision, and M3CoT. Bubble area encodes backbone size; shading is not a fitted frontier.

For readability, the horizontal display assigns equal width to each adjacent labeled interval: 1, 2, 3, 4, 6, 10, 20, 50, 100, 500, and 2000×. Values are log-interpolated within their enclosing interval. This is a display transformation only: tick labels and point annotations report the true compression ratios, and no interval or observation is omitted.

Table 6: Denominators and total generated-token counts used in the contextual landscape. Total tokens include both thinking and answer regions when they are reported separately.

Method	Accuracy reference	Reference total	Method total
LatentWeave step 300	Q3-8B-Thinking	3,649	1,252
LatentWeave step 750	Q3-8B-Thinking	3,649	660
LVR / Monet / SkiLa	Q2.5-7B-Instruct	312	21 / 280 / 310
CoLT	Q3-8B-Instruct	4,995	8
OneThinker / DeepVision	Q3-8B-Instruct	4,995	1,408 / 3,173
SwiReasoning	Q3-8B-Thinking	3,649	3,491

Table 1 and Figure 1 report total visible output, summing thinking and answer tokens when available separately, including explicit reasoning in Instruct answer fields. Counts exclude latent computation and do not measure latency explicitly. Both LatentWeave points use Qwen3-VL-8B-Thinking for both denominators. The dashed curve and extensions do not estimate unmeasured performance.