Evolving Loop Programs: Discovering How Frozen Transformers Should Loop

Supplementary Material

Contents

712

Main Paper 713

1. Introduction 2 714

2. Related work 2 715
Transformer looping. 2 716
Knowledge discovery by agents. 3 717

3. Method 3 718
3.1. Preliminaries and Problem Setting 3 719
3.2. Evolving Loop Programs 3 720
What to Search: Loop Programs. 3 721
What to Observe: Programmable Diagnostics. 3 722
How to Search: Evidence-Guided Program Evolution. 3 723
What to Evaluate and Select: Objectives, Targets, and Priorities. 4 724

4. Experiments 5 725
4.1. Experimental Setup 5 726
Models and tasks. 5 727
Benchmarks and metrics. 5 728
Search protocol and data splits. 5 729
4.2. Results and Analysis 5 730
RQ1: Can ELP Improve Image Generation? 5 731
Models Trained without Loops. 5 732
Models Trained with Loops. 5 733
RQ2: Which Search Design Choices Matter? 5 734
What to Search: Configurations vs. Programs. 6 735
What to Observe: Programmable Diagnostics. 6 736
How to Search: Choice of Search LLM. 6 737
What to Evaluate: Target Strength. 6 738
What to Evaluate: Search Objectives. 8 739
Discovered Loop Programs. 8 740
RQ3: Can ELP Improve Video Generation? 8 741
Long-Video Consistency. 8 742
Short-Video Quality–Compute Trade-offs. 8 743
RQ4: Does the Approach Extend to Text Reasoning? 8 744
4.3. Search Cost 8 745

5. Conclusion 8 746

Appendix

A. Vocabulary and Terminology **2**

B. Complete Benchmark Results **4**
B.1. Text-to-Image Diffusion: Models Trained without Loops . . . 4
B.2. Text-to-Image Diffusion: Models Trained with Loops . . . 6
B.3. Long-Video Consistency: Loop Window, Depth, and Mechanistic Analysis . . . 7
B.4. Short-Video Diffusion: Wan2.1-1.3B . . . 10
B.5. Text Reasoning: Models Trained with Loops (Dory1.0-4B) . . . 10
B.6. Text Reasoning: Models Trained without Loops (Qwen3-4B) . . . 12

C. Search Design and Search-Target Calibration **12**
C.1. Complete Frozen Loop Programs . . . 12
C.2. Representative Summaries and Replay Records . . . 14
C.3. Recorded Search LLM Episode on Scale-RAE . . . 15
C.4. Search Space, Diagnostics, Search LLMs, and Objectives . . . 15
C.5. Target Strength in Image Generation . . . 17
C.6. Search-Target Calibration in Autoregressive Text Reasoning . . . 17

D. Looped-DiT Case Study: From Diagnostics to Frozen Programs **18**
D.1. Diagnostics, Program Design, and Search Evolution . . . 18
D.2. Held-Out Confirmation and Schedule-Only Comparison . . . 19

E. Runtime Observation Definitions **19**

F. Search LLM Prompts **20**
F.1. Primary Program-Search System Prompt . . . 20
F.2. Per-Turn Observation Payload . . . 23
F.3. Backbone and Ablation Prompt Adaptations . . . 24

G Experiment Settings and Reproduction Details **26**
G.1. Search Protocols, Inference Specifications, and Hardware . . . 26
G.2. Evaluation Scopes, Asset Provenance, and Rollout Conventions . . . 27
G.3. Search Cost and Token Usage . . . 32

H Extended Related Work **32**
H.1. Looped Transformers, Weight Sharing, and Latent Recurrent Depth . . . 32
H.2. Dynamic Depth, Early Exiting, and Conditional Computation . . . 33
H.3. Test-Time Compute Scaling and Inference-Time Diffusion Acceleration . . . 33
H.4. Autoregressive Video Diffusion and Long-Horizon Consistency . . . 34
H.5. Neural Architecture Search and LLM-Guided Program Discovery . . . 34

Supplementary References **34**

A. Vocabulary and Terminology

Table S1 defines the terms used in the main paper and appendix. Short labels used in tables and figures are listed alongside their full forms. In particular, the search LLM operates offline, while the frozen backbone executes the selected loop program during deployment.

Table S1. **Vocabulary separates program search from deployed loop execution.** Definitions and compact labels apply throughout the paper.

Term / compact label	Meaning and usage in this paper
Method, programs, and execution	
Evolving Loop Programs	Our method for discovering executable loop programs offline with an LLM, while keeping pretrained backbone weights fixed.
Search LLM	The LLM that proposes program edits and diagnostic collectors from accumulated evidence. It is separate from the backbone being evaluated.
Loop program $p = (c, \boldsymbol{\lambda})$	Executable code c together with its parameters $\boldsymbol{\lambda}$. A candidate is one particular code–configuration pair.
Configuration Search	Search over predefined configurations while keeping the executable rules fixed.
ELP (Ours)	The full Evolving Loop Programs method: the search LLM edits executable rules and their parameters, supported by programmable diagnostics.
ELP w/o Diagnostics	Program search using scores-only feedback, with no diagnostic probes.
Loop structure	Which transformer blocks or computational paths repeat.
State-update dynamics	How repeated outputs update hidden states or predictions, including damping and extrapolation.
Execution control	When and how much computation is performed, including sampling schedules, gating, guidance, and state reuse where supported.
Programmable diagnostics	The search LLM writes collectors to inspect intermediate states and execution traces, producing observations $\mathcal{O}_i$ for candidate i without changing its computation.
Training-free / frozen weights	No updates to pretrained backbone weights during search or deployment. The backbone may originally have been trained with or without loops.
Frozen program / replay	Selected code and parameters are fixed after search. Replay executes that program without further calls to the search LLM; intermediate states still evolve.
Search round / candidate evaluation	After round r, the search LLM proposes the next batch $\mathcal{C}_{r+1}$; an evaluation measures one candidate under the specified protocol. Neither denotes a transformer loop pass.
Loop count K / sampling step	K counts passes through a repeated block group; $K = 1$ is one pass. A sampling step advances the diffusion sampler and may contain several loop passes.
Core Passes	Total passes through the Looped-DiT core across sampling steps, per reported branch; for variable loop counts this is $\sum_t K_t$.
Operating point / Priority	A selected program serving a stated criterion. Speed, Balanced, and Quality describe quality–compute priorities; Micro and Macro describe text-reasoning aggregation priorities.
Evaluation, selection, and reporting	
Development split (Dev.)	Inputs used to evaluate candidates and guide search. Development scores are search feedback, not independent test evidence.
Held-out split	Inputs withheld from candidate search and used for post-search confirmation.
Full benchmark (Full)	The complete reported benchmark, including development examples where specified. Full-benchmark results are not automatically independent held-out results.
Nondominated program / Pareto frontier	No evaluated alternative is at least as good on every objective and strictly better on one. The frontier $\mathcal{P}_r$ contains their objective vectors after round r.
Pareto hypervolume (HV)	Objective-space volume dominated by the retained frontier relative to a fixed reference point. Compare values only under the same objectives, scaling, and reference point.
Measured anchor	A baseline program and its measured objective values used to initialize or contextualize search. It is distinct from the hypervolume reference point.
Search target	Desired objective values shown to the search LLM. True uses measured reference values; shifted targets guide search without changing benchmark scoring.
Baseline / Native / Full reference	Baseline (Base) denotes standard inference. Native denotes the pretrained loop configuration; Full reference denotes its full-budget setting, not the Full evaluation split.
Hypervolume reference point	The fixed point used to report hypervolume within a study. Changing the search target does not change this reporting reference.

Continued on the next page

Table S1 (continued): Vocabulary and terminology

Term / compact label	Meaning and usage in this paper
Target strength	Mean relative increase of the image-search target over True, averaged over GenEval, DPG, and search-time latency speedup, in percent. Replay FLOPs speedup is reported separately. Text-target shifts are reported separately in percentage points.
TFLOPs / FLOPs speedup	TFLOPs reports cost in units of 10^{12} floating-point operations. FLOPs speedup is reference cost divided by program cost; it excludes offline search and does not measure wall-clock speedup.
DPG / LG	DPG abbreviates DPG-Bench; LG abbreviates LoopGuidance. Loop w/o LG and Loop + LG denote the corresponding baseline variants.

B. Complete Benchmark Results

This section expands the text-to-image comparisons in RQ1, the long- and short-video evaluations in RQ3, and text reasoning across pretrained looped (`Dory1.0-4B`) and unlooped (`Qwen3-4B`) language models in RQ4. We retain the main paper's distinction between models trained without loops and models trained with loops. All pretrained weights remain fixed during offline search and subsequent replay; selected programs execute without further calls to the search LLM. Speed, Balanced, and Quality name operating points with different priorities. DPG denotes DPG-Bench. Throughout the supplementary tables, **bold** and underlined values mark best and second-best results within the displayed comparison group; gray shading identifies our programs. Reported costs are inference TFLOPs unless explicitly marked as measured wall-clock latency (s/img or s/video) and wall-clock speedup; all costs exclude offline search overhead. Appendix G details the evaluation scopes, dataset splits, and checkpoint provenance across all nine backbones.

B.1. Text-to-Image Diffusion: Models Trained without Loops

Table S2 expands Table 1a with full GenEval subtasks, Scale-RAE's 12-step Balanced program, matched 10-step no-loop controls, held-out confirmation splits and deltas versus Baseline, and measured wall-clock inference latency. Read panel (a) for full-benchmark subtask scores and inference FLOPs, and panel (b) for development (320 prompts) versus unseen held-out (1,298 prompts: 441 GenEval, 857 DPG-Bench) generalization alongside RTX 5090 wall-clock timings. Both backbones were trained without repeated blocks; our programs add loops only at inference. Why do our 6–14-step programs outperform 25- and 50-step Loop + LG while standard 10-step inference collapses (42.32 on Scale-RAE; 34.88 on RAE)? In text-to-image diffusion, compositional layout and object-attribute binding crystallize during the high-noise early half ($t/T \leq 0.5$), whereas late outer steps mainly refine high-frequency texture; concentrating recurrence in high-noise steps while eliminating redundant outer-step evaluations via one-step guidance-correction reuse (Scale-RAE) or shared unlooped prefixes and cached conditioning (RAE) preserves layout fidelity at a fraction of the compute. Across the 1,298 unseen held-out prompts, every Speed, Balanced, and Quality program achieves consistent positive gains over standard inference and converts FLOPs reductions into $4.48\times$ and $11.53\times$ measured wall-clock speedups over Loop + LG ($1.61\times$ and $6.13\times$ over Baseline), while Speed and Quality offer complementary operating points across subtasks and metrics (on Scale-RAE held-out prompts, Speed leads GenEval at 59.24 while Quality leads DPG-Bench at 80.18 and fine-grained position and color attribution). For `RAE` (the DiffusionBench [21] `RAE-SigLIP2-B/DDT` 615M adaptation), Appendix G.2 records the checkpoint revision, SHA-256 hash, and replay batch configuration.

We reran standard inference (Baseline, plus matched 10-step Baseline) and Train-Free Loop without and with LoopGuidance [36] (Loop w/o LG and Loop + LG) on both Scale-RAE and RAE under the main-paper protocol with frozen weights. Single and Two denote single- and two-object evaluation; Count, Colors, Pos., and C. Attr. denote counting, colors, position, and color attribution.

Qualitative comparison settings. Figures 3 and S1 compare matched GenEval and DPG-Bench examples on Scale-RAE (1.5B + 2.4B DiT) and RAE (Qwen3-0.6B + DDT). Read the five columns as full-step standard inference (25/50 steps), 10-step standard inference, Train-Free Loop without and with LoopGuidance, and our 10-step programs; compute appears below each heading. Our programs reduce FLOPs by $1.61\times$/$3.12\times$ versus full-step standard inference and $3.71\times$/$5.81\times$ versus Loop + LG. The selected outputs illustrate text rendering, attribute binding, object composition, and color attribution, not the frequency of improvements. Each marked region is enlarged directly from the original image, with the same crop size across all five methods within an example.

Table S2. **ELP improves overall compositional scores, held-out generalization, and measured wall-clock speed over both full- and 10-step no-loop baselines.** Panel (a) expands Table 1a with GenEval subtasks, Balanced, and 10-step no-loop controls; panel (b) reports development (320) and held-out (1,298) splits, held-out gains versus Baseline, and RTX 5090 latency (batch 8 for Scale-RAE, batch 16 for RAE). Rankings are within each backbone.

(a) **Full-Benchmark GenEval Subtasks, DPG-Bench, and Inference FLOPs.**

Model	Method	#step	GenEval (%) ↑							DPG	TFLOPs ↓	Speedup ↑	Speedup ↑
			Single	Two	Count	Colors	Pos.	C. Attr.	Overall	(%) ↑		vs. Base	vs. w/ LG
Scale-RAE [32]	Baseline	25	96.25	50.51	37.50	76.60	27.00	22.00	51.64	77.21	31.58	1.00×	2.29×
	Baseline (10-step)	10	95.00	39.39	23.75	62.77	19.00	14.00	42.32	75.66	**12.63**	**2.50×**	**5.74×**
	Loop w/o LG [36]	25	95.00	63.64	50.00	76.60	**32.00**	25.00	57.04	80.00	56.04	0.56×	1.29×
	Loop + LG [36]	25	97.50	65.66	53.75	74.47	30.00	**30.00**	58.56	79.71	72.46	0.44×	1.00×
	Ours (Speed)	10	**98.75**	**70.71**	60.00	**79.79**	26.00	28.00	60.54	80.19	19.56	1.61×	3.71×
	Ours (Balanced)	12	96.25	65.66	**61.25**	75.53	27.00	24.00	58.28	**80.41**	22.71	1.39×	3.19×
	Ours (Quality)	14	97.50	67.68	60.00	78.72	**32.00**	**30.00**	**60.98**	80.19	27.13	1.16×	2.67×
RAE [21]	Baseline	50	83.75	28.28	32.50	63.83	10.00	13.00	38.56	70.91	29.07	1.00×	1.86×
	Baseline (10-step)	10	78.75	24.24	27.50	62.77	6.00	10.00	34.88	70.69	5.81	5.00×	9.32×
	Loop w/o LG [36]	50	82.50	30.30	**33.75**	60.64	7.00	19.00	38.87	72.44	39.67	0.73×	1.37×
	Loop + LG [36]	50	91.25	41.41	30.00	**67.02**	9.00	21.00	43.28	74.68	54.20	0.54×	1.00×
	Ours (Speed)	6	**96.25**	39.39	30.00	64.89	11.00	25.00	44.42	**74.96**	**5.60**	**5.19×**	**9.68×**
	Ours (Quality)	10	93.75	**42.42**	27.50	65.96	**13.00**	**27.00**	**44.94**	74.47	9.33	3.12×	5.81×

(b) **Held-Out Generalization** (1,298 **Prompts) and Measured Wall-Clock Latency.**

Model	Method	#step	Dev (320) ↑	Held-out (1,298) ↑	Held-out Δ vs. Base		Wall ↓	Wall Speedup ↑
			G / D (%)	G / D (%)	GenEval (pp)	DPG (pp)	(s/img)	(Base / w/ LG)
Scale-RAE [32]	Baseline	25	53.80 / 77.55	51.14 / 77.12	—	—	0.283	1.00× / 2.77×
	Baseline (10-step)	10	45.75 / 76.68	41.47 / 75.42	−9.67	−1.71	**0.111**	**2.54× / 7.05×**
	Loop w/o LG [36]	25	59.88 / 79.70	56.35 / 80.08	+5.21	+2.95	0.498	0.57× / 1.57×
	Loop + LG [36]	25	60.42 / 80.06	57.90 / 79.63	+6.76	+2.51	0.784	0.36× / 1.00×
	Ours (Speed)	10	64.99 / 80.66	**59.24** / 80.07	+8.10	+2.95	0.175	1.61× / 4.48×
	Ours (Balanced)	12	63.31 / 81.55	56.89 / 80.14	+5.75	+3.01	0.203	1.39× / 3.87×
	Ours (Quality)	14	69.33 / 80.23	58.76 / **80.18**	+7.62	+3.06	0.239	1.18× / 3.28×
RAE [21]	Baseline	50	44.02 / 71.84	36.78 / 70.68	—	—	0.313	1.00× / 1.88×
	Baseline (10-step)	10	36.59 / 71.69	34.18 / 70.45	−2.60	−0.23	0.063	4.97× / 9.35×
	Loop w/o LG [36]	50	42.22 / 74.08	37.81 / 72.04	+1.03	+1.35	0.434	0.72× / 1.36×
	Loop + LG [36]	50	45.27 / 75.42	42.80 / 74.50	+6.01	+3.81	0.588	0.53× / 1.00×
	Ours (Speed)	6	50.22 / 74.47	42.98 / **75.07**	+6.19	+4.39	**0.051**	**6.13× / 11.53×**
	Ours (Quality)	10	50.34 / 75.29	**43.59** / 74.27	+6.81	+3.58	0.085	3.68× / 6.93×

Qualitative prompt excerpts. The figures shorten the displayed prompt excerpts to emphasize the attributes being compared. The longer excerpts are:

- **Text rendering:** "An old, vibrant red heavy truck with visible signs of wear stands stationary near a weathered stop sign on a street lined with brick facades…"
- **Attribute binding:** "A vibrant yellow rabbit … bounds across a meadow of wildflowers, its sizeable, red-framed glasses slipping to the tip of its nose…"
- **Object composition:** "A city scene where a yellow school bus stands on an asphalt road … behind the stationary bus, a white van and city buildings stand."
- **Color attribution:** "a photo of a purple wine glass and a black apple"

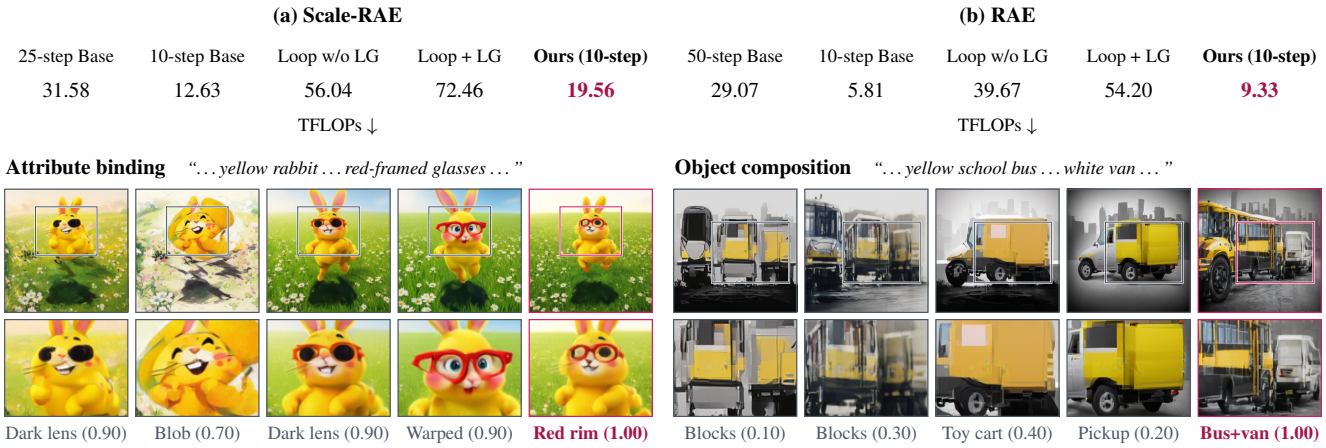


Figure S1. **Additional selected examples illustrate improved attribute binding and object composition with frozen loop programs.** Panels show Scale-RAE (a) and RAE (b), complementing Figure 3. Boxes locate the enlarged details below; magenta frames mark ours. Header numbers are TFLOPs per image; parenthesized values are the reported DPG scores (0–1, higher is better).

B.2. Text-to-Image Diffusion: Models Trained with Loops

Table S3 expands Table 1b with all baseline settings (100-step $K{=}1$ and 50/20/14-step $K{=}4$), core-pass counts, GenEval subtasks, and the 20-step Balanced program. Even on a backbone already pretrained with recurrent loops, native inference applies uniform $K{=}4$ passes across both flat and high-curvature trajectory regions; by combining curvature-warped timestep allocation, dual-stream extrapolation, and bounded late-loop residual reuse, ELP improves every matched-step regime (up to $+4.07$ pp GenEval and $+3.80$ pp DPG at 14 steps) and enables Balanced (20 steps, $5.50\times$ fewer FLOPs) and Quality (25 steps, $4.00\times$ fewer FLOPs) to surpass the 100-step Full reference on overall GenEval (85.03% and 85.08% vs. 84.65%) while staying within 0.23–0.34 pp of its DPG score (85.00–85.11% vs. 85.34%). The diagnostics, program design, and held-out confirmation appear in Appendix D.

We reran the Looped-DiT [6] baselines using the published inference rule with the stated sampling steps and K core passes per step. Core Passes counts core passes across sampling steps. Full reference denotes the 100-step $K{=}4$ reference. The complete table ranks all displayed settings together, so its bold and underlined values can differ from the smaller matched-step groups in the main table.

Table S3. **ELP improves every matched-step regime and recovers full-reference compositional quality at $4\times$–$7.8\times$ lower compute.** This expands Table 1b with all baseline settings, Balanced, and GenEval subtasks. Rankings cover the full displayed set.

Method	#step	Core Passes	GenEval (%) ↑ Single	Two	Count	Colors	Pos.	C. Attr.	Overall	DPG (%) ↑	TFLOPs ↓	Speedup ↑ vs. Full
Looped-DiT [6] ($K{=}4$)	100	400	99.38	93.18	**71.56**	92.29	81.50	70.00	84.65	**85.34**	53.18	1.00×
Looped-DiT [6] ($K{=}1$)	100	100	99.38	94.19	69.69	**92.82**	**83.00**	67.75	84.47	84.26	28.98	1.84×
Looped-DiT [6] ($K{=}4$)	50	200	99.69	94.19	70.94	92.29	79.25	72.25	84.77	85.29	26.59	2.00×
Looped-DiT [6] ($K{=}4$)	25	100	**100.00**	93.69	65.94	**92.82**	78.25	68.25	83.16	84.76	13.30	4.00×
Looped-DiT [6] ($K{=}4$)	20	80	**100.00**	92.93	65.94	90.96	78.50	65.25	82.26	84.18	10.64	5.00×
Looped-DiT [6] ($K{=}4$)	14	56	99.69	89.39	63.12	86.44	76.75	60.75	79.36	81.31	7.45	7.14×
Ours (Speed)	14	48	**100.00**	93.43	67.81	92.55	78.00	68.75	83.43	85.11	**6.80**	**7.82×**
Ours (Balanced)	20	68	99.38	**94.44**	70.31	92.02	80.00	**74.00**	85.03	85.00	9.67	5.50×
Ours (Quality)	25	100	**100.00**	**94.44**	71.25	92.29	81.50	71.00	**85.08**	85.09	13.30	4.00×

B.3. Long-Video Consistency: Loop Window, Depth, and Mechanistic Analysis

Table S4 expands the long-video evaluation along three axes: panel (a) varies the loop window and depth for LongLive 1.0 1.3B across all 100 RememBench scenes (20 search, 80 held-out); panel (b) isolates the 80 unseen held-out scenes across LongLive 2.0 5B, LongLive 1.0 1.3B, and Self-Forcing 1.3B; and panel (c) ablates edit granularity (whole-block loop scoping versus unscoped sub-layer edits) and runtime diagnostics on LongLive 2.0 5B under a matched 80-candidate budget and 20/80 RememBench split. In (a), all windowwise loop configurations act on early high-noise steps $\{0, 1\}$ using zero-based block indices. Comparing bands reveals a clear depth–smoothness trade-off within middle layers: shifting the repeated window slightly deeper from blocks 9–15 to 12–18 sharpens semantic alignment and perceptual revisit fidelity (raising mean CLIP up to 0.9403 at K=4 and lowering LPIPS down to 0.1334 at K=8), whereas earlier middle blocks (9–15) maximize local frame-to-frame temporal smoothness (TempSSIM = 0.9890); within the optimal 12–18 band, increasing K from 4 to 8 trades extra passes for tighter pixel-level LPIPS recovery while preserving strong CLIP and TempSSIM. Across all three backbones in (b), consistency gains transfer cleanly to the 80 unseen scenes with negligible attenuation.

Table S4. **Window/depth choices trade off long-video consistency, held-out gains persist across backbones, and block-level scoping regularizes small-panel search.** Panel (a) compares LongLive 1.0 1.3B loop windows across all 100 scenes; panel (b) reports 80-scene held-out confirmation across backbones; panel (c) compares unscoped sub-layer search (w/o and w/ diagnostics), a fixed layerwise loop, and block-scoped ELP on LongLive 2.0 5B (diagnostic runs and the fixed loop share final-pass KV updates). Entries report mean (or mean (median)); **bold**/underlined: best/second-best in (a),(c), better per backbone pair in (b).

(a) **LongLive 1.0 1.3B Loop Window and Depth Study (All 100 Scenes).**

Window	K	CLIP ↑	LPIPS ↓	TempSSIM ↑
Native [38]	—	0.8670 (0.8919)	0.2609 (0.2591)	0.9819 (0.9839)
11–17	8	0.9218 (0.9526)	0.1480 (0.1363)	0.9869 (0.9903)
9–15	8	0.9218 (0.9407)	0.1533 (0.1404)	**0.9890** (0.9894)
11–17	4	0.9231 (0.9407)	0.1592 (0.1213)	0.9876 (0.9888)
12–18	4	**0.9403** (**0.9585**)	0.1387 (0.1107)	0.9875 (**0.9906**)
12–18	8	0.9371 (0.9504)	**0.1334** (**0.0975**)	0.9866 (0.9904)

(b) **80-Scene Held-Out Confirmation Across Long-Video Backbones.**

Backbone	Method	CLIP ↑	LPIPS ↓	TempSSIM ↑
LongLive 2.0 5B [5]	Native (no loop)	0.8551 (0.8947)	0.3907 (0.4033)	0.9658 (0.9689)
	ELP (Ours, 11–18, K=6, Table 1d)	**0.9213** (**0.9563**)	**0.1577** (**0.1017**)	**0.9900** (**0.9938**)
LongLive 1.0 1.3B [38]	Native (no loop)	0.8676 (0.8919)	0.2604 (0.2556)	0.9819 (0.9835)
	ELP (Ours, 11–17, K=8, Table 1d)	**0.9173** (**0.9432**)	**0.1556** (**0.1380**)	**0.9861** (**0.9888**)
Self-Forcing 1.3B [18]	Native (no loop)	0.6465 (0.6416)	0.5086 (0.5075)	0.9771 (0.9799)
	ELP (Ours, 14–19, K=6)	**0.7375** (**0.7463**)	**0.3808** (**0.3765**)	**0.9788** (**0.9817**)

(c) **Controlled Ablation of Edit Scope and Diagnostics on LongLive 2.0 5B (RememBench T2V).**

Method	Diag.	Block Scope	Dev HV ↑	Dev (20) Mean		Held-out (80) Mean (Median)		
				CLIP ↑	LPIPS ↓	CLIP ↑	LPIPS ↓	TempSSIM ↑
Native (no loop) [5]	✗	✗	—	0.8720	0.3633	0.8551 (0.8947)	0.3907 (0.4033)	0.9658 (0.9689)
Fixed Layerwise Loop (12–19, K=4, no search)	✗	✓	—	0.9083	0.1631	0.9057 (0.9340)	0.1982 (0.1519)	0.9890 (0.9930)
Unscoped Sub-layer (w/o Diag.)	✗	✗	0.711	0.9133	0.3152	0.8588 (0.9043)	0.3878 (0.3703)	0.9636 (0.9669)
Unscoped Sub-layer (w/ Diag., best CLIP)	✓	✗	0.712	0.9266	0.3103	0.8477 (0.8792)	0.4020 (0.4023)	0.9648 (0.9684)
Unscoped Sub-layer (w/ Diag., joint)	✓	✗		0.9244	0.3037	0.8354 (0.8834)	0.4029 (0.4132)	0.9653 (0.9689)
ELP (Ours, per-block 11–18, K=5)	✓	✓		0.9462	0.1225	0.9092 (0.9446)	0.1850 (0.1279)	0.9893 (0.9932)
ELP (Ours, windowwise 11–18, K=4)	✓	✓	**0.857**	0.9550	0.1384	**0.9228** (**0.9564**)	0.1670 (0.1212)	0.9894 (**0.9940**)
ELP (Ours, windowwise 11–18, K=6, Tab. 1d)	✓	✓		**0.9561**	**0.1153**	0.9213 (0.9563)	**0.1577** (**0.1017**)	**0.9900** (0.9938)

Additional LongLive 2.0 qualitative comparisons. Figure S2 supplements Figure 3c with four additional RememBench T2V video sequences from the frozen LongLive 2.0 (5B) loop program in Table 1d (block-group / windowwise recurrence

with loop count $K{=}6$, blocks 11–18, denoising steps $\{0, 1, 2\}$). Across these selected departure-to-revisit examples (3.6 s to 15.8 s), the baseline drifts or corrupts scene structure, whereas the frozen loop program closes each container mid-sequence and restores its original contents: in (a), the baseline collapses the grill grate and sausages into distorted red clutter while our program seals the lid and restores the charred sausages; in (b), the baseline melts the hinged bread-maker lid into a deformed double slab while our program restores the loaf; in (c), the baseline hallucinates a reflective screen inside the milk crate while our program closes the lid and recovers the newspapers; and in (d), the baseline embeds dark arc artifacts into the cracker barrel and shifts viewpoint while our program seals cleanly and restores the crackers. As with other qualitative visualizations, these examples illustrate revisit recovery rather than improvement frequency.

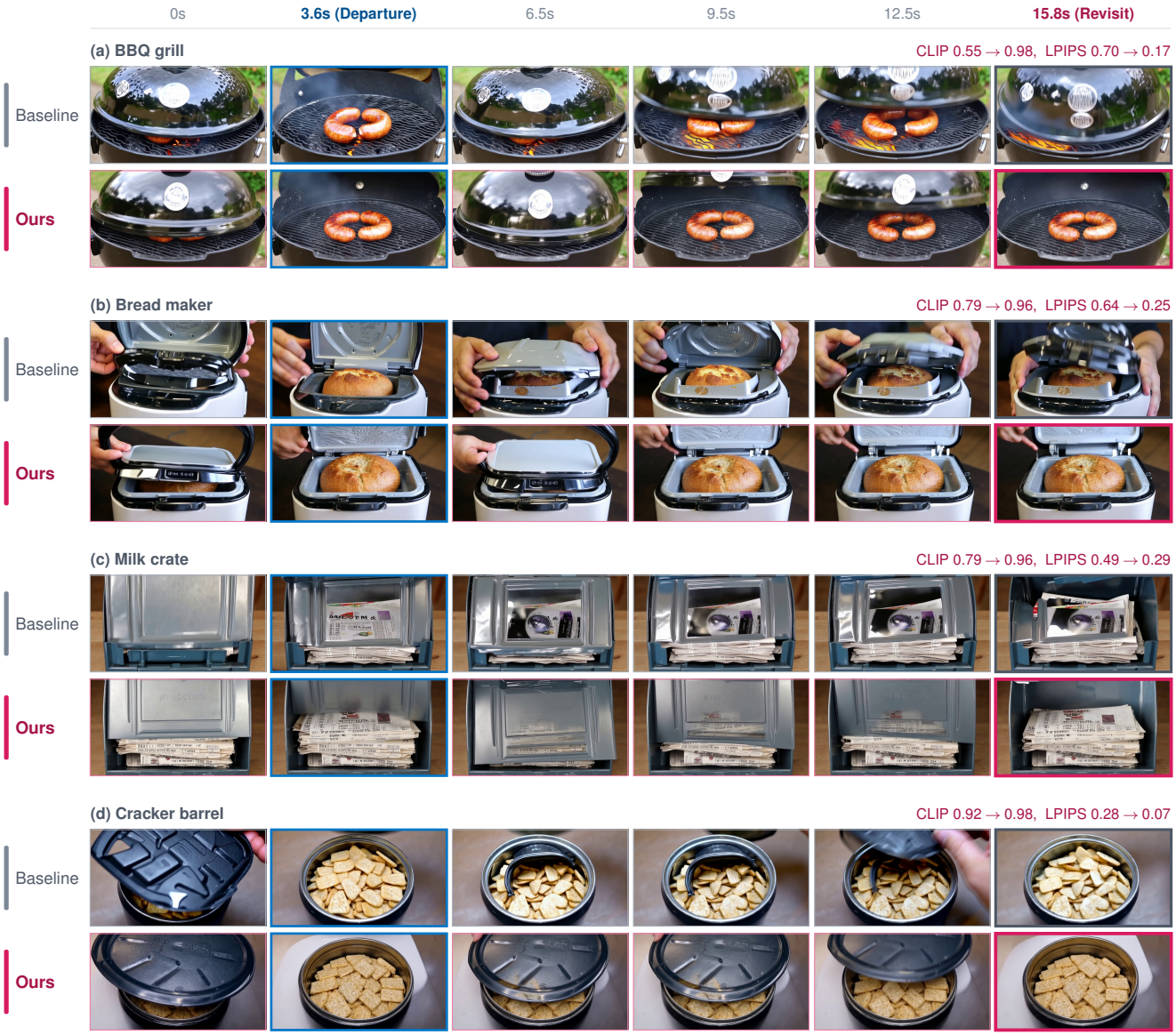


Figure S2. **Frozen LongLive 2.0 (5B) loop programs preserve container geometry and contents on revisit where the baseline drifts or hallucinates.** Matched RememBench T2V frames compare the baseline and our Table 1d program from departure (3.6 s, blue) to revisit (15.8 s, slate vs. magenta) on selected examples: (a) BBQ grill, (b) bread maker, (c) milk crate, and (d) cracker barrel.

Why middle-band block-group recurrence with final-pass KV updates stabilizes long-horizon memory. Across Tables 1d, S4, and S9, independent searches on LongLive 1.0 and LongLive 2.0 converge on the same structural motif:

contiguous middle-band block-group (windowwise) Euler recurrence (blocks 11–18 or 11–17 of 30) with step-size-normalized updates ($\Delta \mathbf{x}^{(r)} = \frac{1}{K}(W(\mathbf{x}^{(r)}) - \mathbf{x}^{(r)})$) and causal key–value (KV) cache writes deferred to the final pass. In autoregressive chunk-wise video diffusion, small per-chunk errors compound once committed to the causal KV cache and reused by all future chunks. Baseline diagnostics explain why boundary layers are unsuitable loop targets—early blocks show strongly negative input–residual alignment (e.g., block 1 in LongLive 2.0) and terminal blocks exhibit abrupt residual spikes (block 29)—whereas middle blocks inhabit a smoother semantic–geometric latent space amenable to iterative refinement. Looping a contiguous block group rather than isolated blocks lets cross-frame attention and feed-forward layers co-adapt across passes (on LongLive 2.0 held-out scenes, block-group $K{=}4$ surpasses per-block $K{=}5$ with fewer passes), while deferring KV updates allows intermediate passes to read frozen historical context without contaminating future-chunk memory.

Recurrence tracks high-noise structure formation and induces strong coupling across search axes. Test-time looping is not an isotropic compute multiplier: it acts primarily during high-noise denoising steps, where global scene layout and object identity are formed before low-noise steps finalize texture. Consequently, active steps follow noise magnitude rather than raw step index (Table S16): LongLive 1.0 (schedule $[1000, 750, 500, 250]$) peaks when looping only early steps $\{0, 1\}$, whereas LongLive 2.0's shift-5 UniPC schedule ($[999, 937, 833, 624]$) retains high noise at step 2 ($t{=}833$) and benefits from steps $\{0, 1, 2\}$. This regime dependence creates strong non-additive coupling across design choices. In the LongLive 2.0 search, excluding step 0 caused Round 2 to stall—both raising K and extending the starting block to block 11 degraded CLIP—yet once step 0 was activated, the same block-11 boundary produced the largest single-round gain ($\Delta\mathrm{HV} = +0.036$). Similarly, in LongLive 1.0, an early-depth band (blocks 1–8) led CLIP in initial rounds before saturating at higher K, and maintaining multi-objective Pareto diversity allowed the middle band (blocks 11–17) to overtake it at $K{=}8$. Greedy coordinate-wise sweeps over an incomplete step schedule would prematurely discard these winning regimes.

Test-time looping acts as a tail-collapse stabilizer rather than a uniform polish. Aggregate averages in Table 1d obscure where consistency gains arise: frozen loop programs do not merely polish already-stable trajectories, but disproportionately suppress catastrophic drift on failure-prone scenes while leaving stable scenes intact. Grouping the 100 RememBench scenes into baseline-drift quartiles, the hardest quartile receives the bulk of the improvement—reducing LPIPS by 63.0% ($+0.159$ CLIP) on LongLive 2.0 and 58.9% ($+0.127$ CLIP) on LongLive 1.0—while the lowest-drift quartile stays unchanged or improves slightly. Across the benchmark, severe perceptual collapse (LPIPS$>$0.40) drops nearly tenfold on LongLive 2.0 ($48 \rightarrow 5$ scenes; $41 \rightarrow 4$ on the 80 held-out scenes) and sixfold on LongLive 1.0 ($12 \rightarrow 2$), and severe semantic failures (CLIP$<$0.80) fall by more than half on both models. Because this tail stabilization transfers cleanly from 20 development scenes to 80 unseen scenes without retraining, test-time looping serves as a general reliability mechanism, while Table S4 shows that tuning K and block depth shifts the remaining balance among semantic fidelity (CLIP), perceptual similarity (LPIPS), and frame-to-frame smoothness (TempSSIM).

Matching edit granularity to small video panels. Table S4c studies how edit granularity and runtime diagnostics interact when long-video rollout cost limits development to a small panel ($N{=}20$ RememBench scenes, 80-candidate budget with `GPT-6-Astra`). On benchmarks with larger development panels—320 image prompts on Scale-RAE, RAE, and Looped-DiT, 48 short videos on Wan2.1, and 113–829 tasks on Dory1.0 and Qwen3—full executable program editing generalizes cleanly without restricting sub-layer edits. When 16-second autoregressive rollouts narrow the panel to 20 scenes, however, exposing arbitrary sub-layer feed-forward and temporal rewrites creates a capacity-to-sample mismatch: unscoped sub-layer search readily fits the 20 scenes (0.9133–0.9266 development CLIP, with or without diagnostics) via scene-specific feed-forward perturbations, while remaining near native fidelity on the 80 unseen scenes (0.8354–0.8588 CLIP vs. 0.8551 Native). Scoping edits to ELP's core block-level loop interface—whole-block or block-group Euler recurrence with final-pass KV commits, without fixing block indices, window structure, active denoising steps, or loop depth K—aligns hypothesis-space capacity with $N{=}20$ so that candidates transfer reliably (as confirmed by the unsearched fixed layerwise control at 0.9057 CLIP and 0.1982 LPIPS). Crucially, diagnostic-guided search inside this minimal interface still drives the decisive gains: it lifts development hypervolume from 0.712 to 0.857, discovers that contiguous windowwise recurrence over middle blocks 11–18 and high-noise steps $\{0, 1, 2\}$ outperforms per-block looping even at lower depth ($K{=}4$ vs. $K{=}5$), and improves held-out consistency to 0.9228 CLIP ($K{=}4$) and 0.1577 LPIPS ($K{=}6$)—a further 20.4% LPIPS reduction beyond the fixed loop (59.6% below Native).

B.4. Short-Video Diffusion: Wan2.1-1.3B

Table S5 reports the complete Wan2.1-1.3B short-video results for frozen programs searched separately from the autoregressive RememBench study, alongside our reruns of the native 50- and 30-step baselines under the local 256-video VBench protocol (Appendix G.2). Panel (a) presents the full 256-video Quality, Semantic, and Overall scores together with temporal and visual sub-dimensions across all four selected programs (Quality names the Overall-selected program, whereas Quality subscore names the separate program selected by that subscore). Panel (b) separates the 48-video development split from the 208 videos excluded from candidate evaluation, the 192 videos strictly unseen by prior search pools, and measured per-video wall-clock generation latency.

Table S5 reveals two complementary operating regimes uncovered by test-time search on Wan2.1-1.3B. First, in the accelerated regime (Speed and Balanced at 26 steps), pairing conditional cross-attention looping on blocks 10–13 with step reduction and interval guidance reuse delivers 2.24× and 2.11× measured wall-clock speedups over the 50-step baseline (52.4 s and 55.7 s vs. 117.4 s per video—faster even than the 30-step native control at 72.4 s) while consistently outperforming 50-step native inference across every temporal-consistency axis, with Balanced and Speed reaching up to 95.54 Subject Consistency (vs. 94.20), 98.60 Background Consistency (vs. 98.09), 99.63 Temporal Flickering (vs. 99.53), and 98.68 Motion Smoothness (vs. 98.52). Second, in the quality-focused regime (50 steps), multi-window spacetime curvature-lookahead looping raises full-benchmark Semantic (84.61 vs. 83.04) and Scene (71.47 vs. 52.17) alignment while retaining +0.88 pp Semantic gain on the 208 held-out videos (83.93 vs. 83.05). Because VBench splits prompts across 16 distinct dimensions (leaving only 4–5 videos per semantic dimension in the 48-video development panel), aggregate Overall scores sit near parity on Held-out 208 (84.81 vs. 84.86) and Unseen 192 (84.94 vs. 84.92), whereas the 2.1×–2.2× speedups, temporal stability gains, and held-out semantic improvements persist. FLOPs speedup uses the 50-step Baseline (28,300.2 TFLOPs). VBench [19] sub-dimensions Subj. Cons., Bg. Cons., Temp. Flick., Motion Smooth., Aesth. Quality, and Temp. Style denote subject consistency, background consistency, temporal flickering, motion smoothness, aesthetic quality, and temporal style.

Table S5. **Wan2.1-1.3B programs deliver 2.1×–2.2× wall-clock speedups with stronger temporal consistency and +0.88 pp held-out semantic gains.** Panel (a) expands VBench aggregates and sub-dimensions on all 256 videos; panel (b) separates Development (48), Held-out (208), and Unseen (192) splits alongside measured generation latency (s/video). **Bold**/underlined: best/second-best.

(a) **Full 256-Video VBench Sub-Dimensions and Inference FLOPs.**

Method	#step	VBench Sub-Dimensions (%) ↑							VBench (%) ↑			TFLOPs ↓	Speedup ↑
		Subj. Cons.	Bg. Cons.	Temp. Flick.	Motion Smooth.	Aesth. Quality	Scene	Temp. Style	Quality	Semantic	Overall		vs. Base
Wan2.1 [31]	50	94.20	98.09	99.53	98.52	67.82	52.17	26.42	**85.39**	83.04	84.92	28,300.2	1.00×
Wan2.1 [31]	30	94.48	98.10	99.57	98.55	67.81	57.07	26.54	85.33	80.54	84.37	16,980.1	1.67×
Ours (Speed)	26	95.46	**98.60**	**99.63**	**98.68**	67.11	55.43	26.84	84.85	80.32	83.94	**11,916.1**	**2.37×**
Ours (Balanced)	26	**95.54**	98.57	**99.63**	**98.68**	67.24	65.22	**26.87**	84.90	82.03	84.33	12,765.1	2.22×
Ours (Quality subscore)	50	94.14	97.91	99.53	98.56	**68.10**	65.76	26.55	85.32	83.57	84.97	30,903.6	0.92×
Ours (Quality)	50	94.01	98.00	99.53	98.57	67.71	**71.47**	26.56	85.25	**84.61**	**85.12**	30,903.6	0.92×

(b) **Split Breakdown (Dev 48 / Held-out 208 / Unseen 192 / Full 256) and Measured Wall-Clock Latency.**

Method	#step	Dev 48 (%) ↑			Held-out 208 (%) ↑			Unseen 192	Full 256	Wall ↓	Wall ↑
		Qual.	Sem.	Ov.	Qual.	Sem.	Ov.	Ov. (%) ↑	Ov. (%) ↑	(s/vid)	Speedup
Wan2.1 [31]	50	85.71	83.00	85.17	**85.31**	83.05	**84.86**	84.92	84.92	117.4	1.00×
Wan2.1 [31]	30	85.70	74.51	83.46	85.23	81.80	84.55	84.83	84.37	72.4	1.62×
Ours (Speed)	26	86.19	86.44	86.24	84.51	78.94	83.39	83.46	83.94	**52.4**	**2.24×**
Ours (Balanced)	26	**86.35**	**89.14**	**86.91**	84.54	80.44	83.72	83.80	84.33	55.7	2.11×
Ours (Quality subscore)	50	86.30	86.72	86.38	85.08	82.89	84.64	84.90	84.97	129.2	0.91×
Ours (Quality)	50	86.11	87.78	86.44	85.03	**83.93**	84.81	**84.94**	**85.12**	129.4	0.91×

B.5. Text Reasoning: Models Trained with Loops (Dory1.0-4B)

Analogous to Looped-DiT-B32 [6] in image generation, we also evaluate ELP on Dory1.0-4B [S1], an open-weight 4B language model pretrained with elastic recurrent depth over a weight-tied recurrent block. All pretrained weights remain frozen while offline search discovers inference-time loop programs modifying recurrent depth, inter-loop state transitions, and KV-cache reuse. Each evaluation below reports complete benchmark scores alongside unseen held-out confirmation splits.

Open-ended generative reasoning, coding, and instruction following. We first evaluate open-ended reasoning, instruction following, and coding across IFBench [S2], GPQA Diamond [29], and SciCode [S3], averaging eight stochastic runs per prompt across both the complete benchmarks (300 IFBench, 198 GPQA Diamond, 65 SciCode) and the unseen 80% held-out split (240, 158, and 52 tasks, respectively). Table S6 compares native $K{=}2$ and $K{=}4$ baselines with our two highest-scoring $K{=}2$ programs. On a pretrained elastic-depth LM, the first two recurrent passes establish the dominant trajectory of each token's hidden state; applying bounded geodesic state extrapolation after pass 2 therefore approximates deeper $K{=}4$ thinking without paying extra recurrent passes, while committing converged last-loop KV states carries those refined representations across subsequent decoding steps. Both $K{=}2$ programs surpass native $K{=}2$ and $K{=}4$ on full (61.58% and 61.45% vs. 60.92% and 61.18%) and unseen held-out averages (62.50% and 62.09% vs. 61.82% and 61.87%) while cutting FLOPs and wall-clock time by roughly 1.7× relative to native $K{=}4$. Between the two KV-caching strategies, converged last-loop KV leads every benchmark on both splits (notably +1.74 pp over native $K{=}2$ on held-out GPQA Diamond), whereas per-loop KV retains exact native-$K{=}2$ wall-clock speed (1.00×) at the lowest total FLOPs.

Table S6. **ELP surpasses native $K{=}2$ and $K{=}4$ on both full and unseen held-out open-ended generation at $K{=}2$-level compute.** Means across eight runs per prompt on the full benchmarks (300 IFBench, 198 GPQA-D, 65 SciCode) and the unseen 80% held-out split (240, 158, and 52 tasks); GPQA-D denotes GPQA Diamond. Speedups use ratios of full-set means relative to native $K{=}2$ and $K{=}4$. Gray shading: ours; **bold**/underlined: best/second-best per column (ties bolded only).

Method	Full Benchmarks (%) ↑				Held-out (80%) (%) ↑				Total TFLOPs ↓	Wall Speedup ($K{=}2$ / $K{=}4$) ↑	FLOPs Speedup ($K{=}2$ / $K{=}4$) ↑
	IFBench	GPQA-D	SciCode	Avg.	IFBench	GPQA-D	SciCode	Avg.			
Native ($K{=}2$, per-loop KV)	78.48	70.39	33.90	60.92	79.30	70.57	35.61	61.82	96,300	**1.00× / 1.72×**	1.00× / 1.67×
Native ($K{=}4$, per-loop KV)	78.71	71.02	33.81	61.18	79.27	71.44	34.90	61.87	161,198	0.58× / 1.00×	0.60× / 1.00×
ELP (Ours, $K{=}2$, last-loop KV)	**78.94**	**71.72**	**34.07**	**61.58**	**79.48**	**72.31**	**35.71**	**62.50**	95,719	0.96× / 1.65×	**1.01×** / 1.68×
ELP (Ours, $K{=}2$, per-loop KV)	78.88	71.53	33.94	61.45	79.06	72.15	35.06	62.09	**95,223**	**1.00× / 1.72×**	**1.01× / 1.69×**

Multiple-choice STEM reasoning. Table S7 next compares native inference and our frozen programs on Dory1.0-4B at matched recurrent passes ($K{\in}\{2,3,4\}$) under the zero-shot log-likelihood suite used for Qwen3-4B [37] (Appendix G.2), covering MMLU-STEM [15], GPQA-Main [29], ARC-Challenge [7], and SciQ [34]; in the table headers, ARC N. denotes ARC-Challenge normalized accuracy, while M. Norm, M. Grp, and M. Dom denote Micro Acc Norm, Macro Group, and Macro Domain. Across recurrent depths, Table S7 shows that uniform native iteration under-utilizes even a recurrently pretrained block, and that geometry-aware state transitions unlock compounding gains. At $K{=}2$, span-based recurrence cuts per-token compute by 1.17× (9.45 vs. 11.03 GFLOPs/tok) while raising overall Micro, Macro Group, and held-out Macro Domain accuracy (+0.74 pp), led by harder reasoning tasks (GPQA-Main +1.56 pp). At $K{=}3$ and $K{=}4$, tailored transition dynamics—spherical velocity extrapolation with calibrated output scaling at $K{=}3$, and annealed input injection, depth-wise rate scaling, and forward continuation at $K{=}4$—improve every individual benchmark and held-out aggregate at identical per-token compute (+0.87 and +0.60 pp Micro Acc). Notably, our $K{=}3$ program (60.68% Micro Acc) already surpasses both native $K{=}4$ (60.29%) and deeper native $K{=}6$ (60.24%), demonstrating that better state-transition dynamics can substitute for extra recurrent passes.

Table S7. **ELP improves matched-K STEM accuracy–compute trade-offs on Dory1.0-4B.** Held-out reports Macro Group / Macro Domain on the 3,316 unseen questions. Speedup reports per-token GFLOPs speedup vs. each matched native-K baseline. Gray shading: ours; **bold**: better within each matched-K pair (ties unbolded).

Setting	Method	GFLOPs / tok ↓	Speedup (vs. K) ↑	Individual Benchmarks (%) ↑					Aggregates (%) ↑				Held-out M. Grp / M. Dom ↑
				MMLU	GPQA	ARC	ARC N.	SciQ	Micro	M. Norm	M. Grp	M. Dom	
Matched $K{=}2$	Native	11.03	1.00×	54.69	28.57	**47.78**	**49.66**	95.00	59.64	**60.14**	56.51	51.98	56.86 / 52.53
	ELP (Ours)	**9.45**	**1.17×**	**55.02**	**30.13**	46.76	48.21	**95.20**	**59.69**	60.07	**56.78**	**52.93**	**56.95 / 53.27**
Matched $K{=}3$	Native	14.74	1.00×	55.02	28.79	47.95	49.83	94.90	59.81	60.29	56.67	52.15	57.18 / 52.85
	ELP (Ours)	14.74	1.00×	**55.67**	**30.13**	**49.32**	**50.85**	**95.30**	**60.68**	**60.97**	**57.61**	**53.05**	**57.93 / 53.45**
Matched $K{=}4$	Native	18.44	1.00×	55.54	29.69	48.89	49.91	94.60	60.29	60.68	57.18	53.02	57.50 / 53.43
	ELP (Ours)	18.44	1.00×	**56.07**	**30.80**	**49.49**	**50.51**	**95.10**	**60.89**	**61.21**	**57.86**	**53.96**	**58.24 / 54.45**

B.6. Text Reasoning: Models Trained without Loops (Qwen3-4B)

Table S8 expands Table 1c for the frozen Qwen3-4B program (ELP (Ours), selected under the task-balanced Macro criterion) with 3,316-question held-out aggregates alongside the full 4,145-question suite. Contrasting Qwen3-4B with Dory1.0-4B (Section B.5) clarifies how pretraining shapes test-time loop headroom in language models. Because Qwen3-4B's frozen weights were trained without loops, its residual blocks operate within a tighter multi-pass stability envelope; nevertheless, ELP's non-overlapping dual-window RK3 schedule with read-only intermediate probes (Table S9) delivers consistent held-out gains over standard Non-looped inference (+1.42 pp Micro Acc from 63.93% to 65.35%, +1.56 pp Macro Domain from 58.46% to 60.02%), with the largest full-suite gains on knowledge-intensive multi-step reasoning (GPQA-Main +2.90 pp, MMLU-STEM +2.29 pp). On the 3,316 unseen questions, the frozen program matches or edges past the strong Fixed Loop [3] baseline on Micro Acc (65.35% vs. 65.26%), Macro Group (61.60%), and Macro Domain (60.02% vs. 59.81%), with tighter margins than on development questions (and 65.65% vs. 65.77% on Micro Acc Norm). All methods use the same 4,145-question evaluation suite (Appendix G.2); we reran Non-looped and Fixed Loop on the exact same split.

Table S8. **The frozen Qwen3-4B program transfers held-out gains over Non-looped inference and matches or improves upon Fixed Loop across key aggregates.** ELP (Ours) matches the single task-balanced (Macro) program column in Table 1c. Full covers 4,145 questions (829 dev + 3,316 held-out). **Bold**/underlined: best/second-best within each row.

Metric (%) ↑	Non-looped [37]	Fixed Loop [3]	ELP (Ours)
Individual benchmarks (Full, 4,145 questions)			
MMLU-STEM Acc	62.43	64.20	**64.72**
GPQA-Main Acc	31.92	33.26	**34.82**
ARC Acc	50.60	**51.71**	**51.71**
ARC Acc Norm	54.18	**54.95**	54.61
SciQ Acc	95.70	96.20	**96.50**
Full-benchmark aggregates (4,145 questions)			
Micro Acc	63.81	65.04	**65.48**
Micro Acc Norm	64.27	65.72	**65.91**
Macro Group	60.16	61.34	**61.94**
Macro Domain	58.37	59.60	**60.36**
Held-out aggregates (3,316 unseen questions)			
Micro Acc	63.93	65.26	**65.35**
Micro Acc Norm	64.35	**65.77**	65.65
Macro Group	60.24	**61.60**	**61.60**
Macro Domain	58.46	59.81	**60.02**

C. Search Design and Search-Target Calibration

This section expands the search-design comparisons in RQ2. We compare search spaces, search LLMs, and objectives, followed by search-target calibration for image diffusion and autoregressive text reasoning. A search target is the quality–cost reference presented to the LLM during search. It is distinct from the fixed hypervolume reference point used to compute Pareto hypervolume. The image and text studies use different tasks and scoring protocols, so their target shifts and outcomes are interpreted separately.

C.1. Complete Frozen Loop Programs

Table S9 expands the two examples in Table 3 to eight backbones, adding Dory1.0-4B [S1] alongside unlooped image, video, and text backbones; Self-Forcing 1.3B uses the pure AST loop contract summarized in Prompt 4c and Table S15. Read each row from the operating priority through loop structure, state-update dynamics, and execution control. Across all eight backbones, discovered programs converge on a shared mechanistic principle: they concentrate recurrence inside middle semantic blocks during high-noise or multi-step reasoning stages, while pairing it with modality-specific state transitions and memory controls—cross-step correction or residual reuse in image diffusion, deferred final-pass KV commits in autoregressive

video and text rollouts, and geodesic/spherical state extrapolation in pretrained looped models. All rules describe frozen programs replayed without further search-LLM calls; the complete settings retain the backbone-specific mechanisms omitted from the compact main-paper table.

Table S9. **ELP discovers programs with distinct loop structures, state-update dynamics, and execution controls across backbones.** Rules summarize the frozen operating points reported in Tables 1, S5, S6, and S7, plus the LongLive 1.0 setting in Table 3 from the additional window/depth study; s denotes outer sampling steps.

Task	Backbone	Priority	Loop structure	State-update dynamics	Execution control
Image generation	Scale-RAE [32]	Speed ($10s$) / Quality ($14s$)	Layerwise dense loop on blocks 12–27 of 32 (K=2)	Damped Euler ($\Delta\mathbf{h}$=0.5($B_l(\mathbf{h})-\mathbf{h}$)) + output LoopGuidance (w=3.0)	Early half ($t/T\leq$0.5); skip D_{ref} if K_t=0; interval-2 D_{ref} refresh + 1-step Δ_{cached}
Image generation	RAE [21]	Speed ($6s$) / Quality ($10s$)	Modality-split sparse loop on encoder blocks 19–22 of 28 (K=4; image queries at k=1..3)	Split Euler (λ/K=0.25) with cached Δ_{cond} on text/time + LoopGuidance (w=15/12)	Early half ($t/T\leq$0.5); share unlooped prefix blocks 0–18 across D_{ref} and D_{loop}
Image generation (pretrained loops)	Looped-DiT-B32 [6]	Speed ($14s$) / Quality ($25s$)	Adaptive $K_t\in\{4,2\}$ (48 passes) or full K_t=4 (100 passes) on 5-block core	Dual-stream latent extrapolation (α=0.20/0.22); cached $\Delta_{2\to4}$ (Speed); damped/tapered x_0 AB2 (β=0.32)	Curvature-warped SNR schedule (s=1.38/1.30); symmetric $4/2$ stride (age $\leq$2); cached text preamble
Text reasoning	Qwen3-4B [37]	Micro / Macro	Non-overlapping dual window (54 vs. 36 layers): $[14]+[15..18]$ and $[25..28]$ (K=3)	Post-L_{14} tapered RK3 (H=1.12, τ=0.20, norm γ=0.25) in W1; anchored Ralston RK3 in W2	Full-token FP32 state updates; read-only intermediate RK probes (`use_cache=False`), single KV commit
Text reasoning (pretrained loops)	Dory1.0-4B [S1]	STEM (K=2/3/4) / Open (K=2)	Weight-tied elastic recurrent block; span recurrence (K=2 STEM) or matched $K\in\{2,3,4\}$ passes	STEM: spherical velocity extrapolation + calibrated scaling (K=3); annealed input injection, rate scaling & continuation (K=4). Open: geodesic extrapolation after pass 2	Matched-K compute (1.17× speedup at K=2 STEM); converged last-loop or per-loop KV states on open-ended decoding
Long video generation	LongLive 2.0 5B [5]	w/ Ours (consistency)	Contiguous windowwise loop over causal Wan-DiT blocks 11–18 of 30 (K=6)	Windowwise Euler (K=6): $\Delta\mathbf{x}^{(r)}=\frac{1}{6}(W(\mathbf{x}^{(r)})-\mathbf{x}^{(r)})$	First 3 of 4 chunk-wise UniPC steps ($t\in\{999,937,833\}$); defer causal KV commit to final pass r=5
Long video generation	LongLive 1.0 1.3B [38]	Highest CLIP in window/ depth study	Contiguous windowwise loop over causal blocks 12–18 of 30 (K=4)	Windowwise Euler: $\Delta\mathbf{x}^{(r)}=\frac{1}{4}(W(\mathbf{x}^{(r)})-\mathbf{x}^{(r)})$	Loop only at denoising steps $\{0,1\}$ of 4; defer causal KV commit to final pass r=3
Short video generation	Wan2.1-1.3B [31]	Speed ($26s$) / Quality ($50s$)	Speed: cond. `cross_attn` on 10–13 (K=3); Quality: 3 windows $[[10,11],[12,13],[18,19]]$ (K=2)	Speed: convex ODE weights $[0.2,0.3,0.5]$; Quality: spacetime curvature-lookahead RK2 ($[0.63,0.37]$)	Speed: $t/T\in[0.05,0.40)$, CFG decay 6.5→4.0 + interval-5 reuse; Quality: $t/T\in[0,0.46)$, shift 7.5

Representative frozen loop programs. The listings below distill three frozen programs from Table S9 (for Scale-RAE Speed, Looped-DiT-B32 Speed, and LongLive 2.0 5B) to illustrate how discovered rules jointly coordinate loop structure, state-update dynamics, and execution controls at deployment without calls to the search LLM.

Frozen Program 1: Scale-RAE 10-step Speed (`step10_open_r6_euler2_lg300_reuse2_s10`)

```
# Deployed inside Euler sampling over steps t_idx = 0..9 (total T = 10)
w, interval, reuse_scale = 3.0, 2, 1.0
delta_cache, refresh_step = None, -999

for t_idx in range(T):
    if t_idx / T <= 0.5:  # Early active loop window; repeat blocks 12..27 with K=2
```

```
        def loop_forward(x_t, cond):
            h = prefix_blocks(0..11)(x_t, cond)
            for l in range(12, 28):
                for k in range(2):  # Damped Euler step size lambda/K = 0.5
                    h = h + 0.5 * (block[l](h) - h)
            return suffix_blocks(28..31)(h)

        if delta_cache is None or (t_idx - refresh_step) >= interval:
            d_ref = unlooped_forward(x_t, cond)
            d_loop = loop_forward(x_t, cond)
            delta_cache, refresh_step = (d_loop - d_ref).detach(), t_idx
            pred = d_ref + w * delta_cache
        else:  # Skip reference pass on odd steps; reuse cached correction
            d_loop = loop_forward(x_t, cond)
            pred = d_loop + (w - 1.0) * reuse_scale * delta_cache
    else:  # Skip loops & reference pass in late steps (K_t = 0)
        pred = unlooped_forward(x_t, cond)
    x_t = euler_step(x_t, pred, t_idx)
```

Frozen Program 2: Looped-DiT-B32 14-step Speed (`step14_open_r1_s1_s14_rescache_cap8`)

```
# Curvature-warped SNR schedule (s=1.38) over 14 steps; 48 total conditional core passes
res_cache, cache_step = {}, {}
prev_pred = None
txt_pre, pos_emb = precompute_text_preamble(prompt)  # Cached across all timesteps

for idx, (t0, dt_ratio) in enumerate(warped_snr_schedule(steps=14, s=1.38)):
    n_cond = 4 if (idx in refresh_steps or t0 < low_t) else 2  # Adaptive K_t in {4, 2}
    img, txt = pre_blocks(x_t, txt_pre, pos_emb)
    hist_img, hist_txt = [img], [txt]
    for r in range(1, n_cond + 1):
        prev_img, prev_txt = img, txt
        img, txt = run_weight_tied_core(img, txt)  # 5 weight-tied blocks
        hist_img.append(img); hist_txt.append(txt)
    if n_cond == 4:
        if 0.15 <= t0 < 0.90:  # Dual-stream latent extrapolation on full refresh
            img = img + 0.20 * (img - prev_img)
            txt = txt + 0.20 * (txt - prev_txt)
        res_cache[2], cache_step[2] = (img - hist_img[2], txt - hist_txt[2]), idx
    elif 2 in res_cache and (idx - cache_step[2]) <= 2:  # Bounded residual reuse (age <= 2)
        d_img, d_txt = res_cache[2]
        img, txt = img + res_weight * d_img, txt + res_weight * d_txt
    pred = post_blocks(img, txt)
    if prev_pred is not None:  # Tapered x0 second-order AB2 update
        pred = pred + (0.32 * taper(t0) * dt_ratio) * (pred - prev_pred)
    prev_pred = pred
    x_t = step_update(x_t, pred, idx)
```

Frozen Program 3: LongLive 2.0 5B Consistency (`r08_ww_layers11_18_steps012_k6`)

```
# Autoregressive 12-chunk UniPC rollout; shift=5 noise schedule [999, 937, 833, 624]
for chunk in range(12):
    x_c = init_chunk_noise(chunk)
    for step_idx in range(4):
        x = prefix_blocks(0..10)(x_c, causal_kv_cache, write_kv=True)
        if step_idx in (0, 1, 2):  # Contiguous middle window blocks 11..18, K=6
            for r in range(6):
                commit_kv = (r == 5)  # Defer KV cache writes until final loop pass
                x_w = window_blocks(11..18)(x, causal_kv_cache, write_kv=commit_kv)
                x = x + (x_w - x) / 6.0  # Step-normalized windowwise Euler update
        else:  # Step 3 (t=624) executes unlooped
            x = window_blocks(11..18)(x, causal_kv_cache, write_kv=True)
        x_c = unipc_step(x_c, suffix_blocks(19..29)(x, write_kv=True), step_idx)
```

C.2. Representative Summaries and Replay Records

The image-search coordinator requests quality-, speed-, and balanced-priority representatives in the search LLM's final summary. This is a recommendation based on the accumulated evidence, rather than an implemented universal weighted-score maximizer or tie-breaking rule. The named operating points in the result tables identify particular frozen programs; Pareto membership alone does not determine these names.

For Scale-RAE, the saved two-anchor `Gemini-4-Argon` final summary recommends the 15-step candidate `step15_open_r6_step15_ce2_b12_30_t0_58_e30_a54_g075` for high development GenEval at above-baseline speed; its

full-benchmark replay reproduces 57.30/79.64 GenEval/DPG, matching the Speed row in Table 2b.

For RAE, the replay manifest identifies `step6_r7_b19_22_condition_reuse_k4_lg15` (Speed) and `step10_r7_b19_22_condition_reuse_k4_lg12` (Quality), both frozen from the Round 7 source snapshot (SHA-256: `f6a53bf1791cbcba8c78464f74984473831611c0b6613d7f1f45acdb5b902dbf`). Full-benchmark replay reproduces their Table 1a GenEval/DPG pairs (44.42/74.96 and 44.94/74.47). Appendix G summarizes the selection notes and reproduction settings across all backbones.

C.3. Recorded Search LLM Episode on Scale-RAE

Figure S3 records a complete Scale-RAE decision episode (`GPT-6-Astra`, rounds 4–6), linking diagnostic falsification, executable code edits, and controlled evaluation. Across 208 traces, raw predictions stay aligned across steps (0.994–0.995 cosine similarity), yet reusing the looped-minus-unlooped LoopGuidance correction from two steps back incurs 0.86–1.10 relative error versus 0.56–0.63 from one step back, falsifying lag-≥ 2 caching and linear extrapolation despite high cosine alignment (a,b). In the resulting 10-step edit (c), all ten image updates still run—five with fresh looped predictions followed by five unlooped steps—while extra reference calls during loop-active steps drop from five to three by reusing the prior correction at odd steps. In a separate matched 12-step test with three reference calls (d), reusing rather than omitting corrections at skipped-reference steps raises development GenEval from 60.23 to 64.14 (+3.91 pp), yielding the Speed and Quality programs in Table 1a. This episode illustrates within-step computation changes rather than step reduction or an isolated diagnostics ablation.

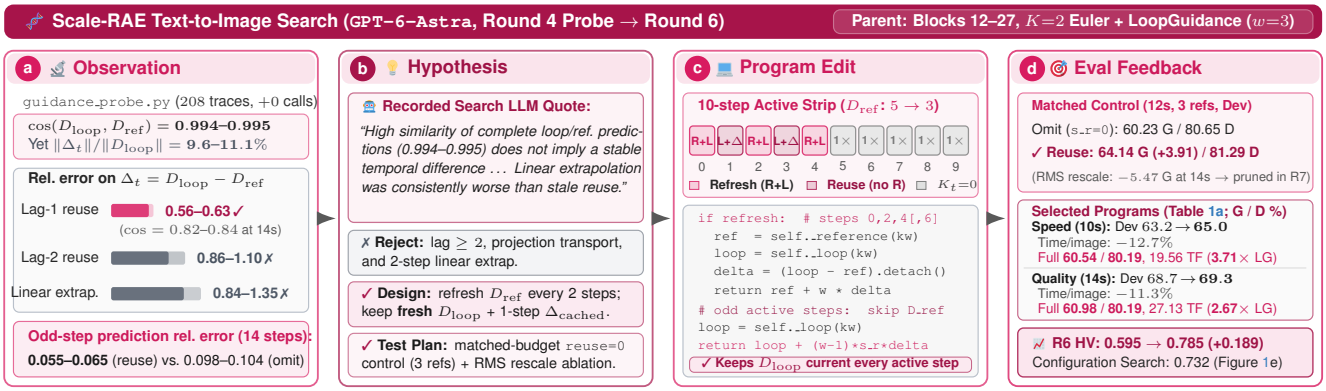


Figure S3. **Recorded Search LLM reasoning links diagnostic falsification, executable code edits, and controlled evaluation.** In the Scale-RAE search (`GPT-6-Astra`, Round 4 probe to Round 6), a collector written by the search LLM falsifies lag-≥ 2 caching and linear extrapolation despite 0.995 output alignment (a,b), motivating one-step correction reuse with fresh D_{loop} passes (c) that outperforms a matched-call omission control and yields the Speed and Quality programs in Table 1a (d).

C.4. Search Space, Diagnostics, Search LLMs, and Objectives

Table S10 expands Tables 2a, 2b and 2d with Balanced programs, step counts, development Pareto hypervolume (HV), FLOPs speedups relative to standard inference (Base) and Loop + LG (w/ LG), and explicit 1,298-prompt held-out scores (441 GenEval, 857 DPG-Bench). Panel (a) unifies the search-space and diagnostic-access comparisons: Configuration Search varies a predefined configuration space using fixed probes, whereas ELP also edits executable rules with or without programmable diagnostics. Panels (b) and (c) expand the search-LLM and objective comparisons following the RQ2 protocol.

Design principles for held-out generalization. Across our ablations and cross-modal studies, three practical principles govern how offline search transfers beyond the development panel: (i) *Diagnostics separate transferable mechanisms from development fitting*: on Scale-RAE (Table S10(a)), scores-only search (ELP w/o Diagnostics) fits the 112-prompt development panel (59.16 GenEval at 8 steps) yet drops to 50.71 on the 441 held-out prompts (below Baseline 51.14) and misses cross-step guidance reuse, whereas programmable diagnostics expose intermediate prediction stability, lifting held-out GenEval to 59.24 and development HV from 0.792 (Configuration Search) to 0.915. (ii) *Match edit granularity to panel size*: full program editing generalizes on 48–829-item panels, whereas at N=20 LongLive 2.0 scenes (Table S4(c)), unscoped sub-layer edits overfit (0.9266 → 0.8477 CLIP); block-level loop scoping regularizes N=20 search, cutting held-out LPIPS 20.4% past a

fixed loop (0.9228 CLIP, 0.1577 LPIPS). (iii) *Panel granularity and objective alignment govern which axes generalize*: dense panels (Scale-RAE, RAE, Looped-DiT, Dory1.0) sustain simultaneous gains across all metrics and splits, whereas finely stratified small panels such as 48-video VBench (4–5 clips per semantic dimension) transfer most reliably on compute speedups and targeted semantic or temporal axes; likewise, quality-only LongLive search trades additional compute (1.61×–1.80× FLOPs) for nearly tenfold reductions in severe long-horizon drift (LPIPS>0.40).

HV in (a) measures the development frontier over GenEval, DPG, and FLOPs speedup; held-out (441 GenEval, 857 DPG) and full-benchmark (553 GenEval, 1,065 DPG) scores are shown separately. Parenthesized factors beside TFLOPs in (b) are speedups over the 25-step Baseline. Rankings are within each panel.

Table S10. **Search space, diagnostic access, search LLMs, and objectives govern both held-out generalization and full-benchmark trade-offs on Scale-RAE.** Panels (a)–(c) expand Table 2 with Balanced programs, development HV, FLOPs speedups, and 1,298-prompt held-out confirmation (441 GenEval, 857 DPG-Bench). Gray shading: our full method; **bold**/underlined: best/second-best within each panel.

(a) **Search Space and Diagnostic Access (Scale-RAE).**

Search variant	Diag.	Priority	#step	HV ↑	Held-out (%) ↑		Full (%) ↑		TFLOPs ↓	Speedup ↑
					GenEval	DPG	GenEval	DPG		(Base / w/ LG)
Configuration Search	✓	Speed	10	0.792	53.83	79.21	55.05	79.45	17.34	1.82× / 4.18×
		Quality	10		58.35	80.10	59.88	80.26	23.65	1.34× / 3.06×
ELP w/o Diagnostics	✗	Speed	8	—	50.71	79.75	52.41	80.05	**11.99**	**2.63× / 6.04×**
		Quality	14		56.37	79.62	57.63	79.68	22.08	1.43× / 3.28×
ELP (Ours)	✓	Speed	10		**59.24**	80.07	60.54	80.19	19.56	1.61× / 3.71×
		Balanced	12		56.89	80.14	58.28	**80.41**	22.71	1.39× / 3.19×
		Quality	14	**0.915**	58.76	**80.18**	**60.98**	80.19	27.13	1.16× / 2.67×

(b) **Search LLM: `GPT-6-Astra` vs. `Gemini-4-Argon`.**

Search LLM	Priority	#step	Held-out (%) ↑		Full (%) ↑		TFLOPs ↓ (vs. Base)
			GenEval	DPG	GenEval	DPG	
`GPT-6-Astra`	Speed	10	**59.24**	80.07	60.54	80.19	**19.56 (1.61×)**
	Balanced	12	56.89	80.14	58.28	**80.41**	22.71 (1.39×)
	Quality	14	58.76	**80.18**	**60.98**	80.19	27.13 (1.16×)
`Gemini-4-Argon`	Speed	15	55.53	79.41	57.30	79.64	25.53 (1.24×)
	Quality	25	57.27	79.86	58.48	80.18	41.85 (0.75×)

(c) **Search Objectives: Joint vs. Single-Metric.**

Setting	#step	Held-out (%) ↑		Full (%) ↑		TFLOPs ↓	Speedup ↑
		GenEval	DPG	GenEval	DPG		(Base / w/ LG)
Baseline [32]	25	51.14	77.12	51.64	77.21	31.58	1.00× / 2.29×
Loop w/o LG [36]	25	56.35	80.08	57.04	80.00	56.04	0.56× / 1.29×
Loop + LG [36]	25	57.90	79.63	58.56	79.71	72.46	0.44× / 1.00×
Joint (Speed)	10	59.24	80.07	60.54	80.19	**19.56**	**1.61× / 3.71×**
Joint (Balanced)	12	56.89	80.14	58.28	80.41	22.71	1.39× / 3.19×
Joint (Quality)	14	58.76	80.18	60.98	80.19	27.13	1.16× / 2.67×
GenEval only	20	**61.77**	80.04	**63.15**	80.15	—	—
DPG only	20	54.79	**80.43**	56.09	**80.61**	—	—

Single-objective search settings. The GenEval-only and DPG-only searches use `GPT-6-Astra`, the same two measured anchors as joint search, and 80 candidates each, with development panels of 112 GenEval and 208 DPG-Bench prompts, respectively. The GenEval-only row reports the 20-step blocks 12–27 program and the DPG-only row uses the development-set winner. Both are evaluated on full GenEval (553 prompts) and DPG-Bench (1,065 prompts) at seed 42; joint rows reuse the programs in Table 1a.

C.5. Target Strength in Image Generation

Table S11 expands Table 2c with the True Speed and 15-step Ultra Balanced programs and separate development, held-out, and full results. Compare target settings within each split: Development is the search panel, and Full includes both development and held-out prompts. The original target coordinates allow the compact target strength percentages to be interpreted; the split columns test whether the selected trade-offs persist beyond search.

Target strength. For compact reporting in Table 2c, target strength is the mean relative increase in the search target's three coordinates: GenEval G, DPG D, and latency speedup s in the search prompt. Here $s = \tau_{\text{base}}/\tau$, where τ is warm inference time per image; it is not the FLOPs ratio reported in the replay columns. With target $\mathbf{a} = (G, D, s)$ and the measured True target $\mathbf{a}^{\text{True}}$, we report

$$\text{Target strength}(\mathbf{a}) = \frac{1}{3}\sum_{j=1}^{3}\left(\frac{a_j}{a_j^{\text{True}}} - 1\right) \times 100\%. \tag{6}$$

The original target coordinates in Table S11 give 0.00% (True), 17.86% (Strong), and 29.28% (Ultra). These percentages use the rounded table values and summarize shifts that differ across objectives. They change the search target shown to the search LLM, not the fixed hypervolume reference point.

Each `Gemini-4-Argon` search evaluates 80 candidates on 320 development prompts, followed by replay on 1,298 held-out prompts and all 1,618 prompts. The target columns retain the original GenEval/DPG/latency-speedup coordinates. Replay TFLOPs and FLOPs speedups are computed separately, as are the manuscript's FLOPs-based frontier and hypervolume summaries; they do not describe the cost signal originally shown to the search LLM. The Strong-target program uses uncached 20-step guidance (44.17 TFLOPs). Ultra-target programs use strided LoopGuidance with cross-step velocity-delta reuse (16.09–31.23 TFLOPs).

Table S11. **Stronger search targets lift held-out GenEval and expand lower-compute trade-offs on Scale-RAE.** Target s denotes latency speedup; replay Speedup denotes FLOPs speedup. Development, held-out, and full-benchmark scores are shown separately. Target strength percentages in Table 2c summarize the coordinate shifts.

Method	Search Target (G / D / s)	Priority	#step	Development (%) ↑		Held-out (%) ↑		Full (%) ↑		TFLOPs ↓	Speedup ↑ (Base / w/ LG)
				GenEval	DPG	GenEval	DPG	GenEval	DPG		
Baseline [32]	—	—	25	53.80	77.55	51.14	77.12	51.64	77.21	31.58	1.00× / 2.29×
Loop w/o LG [36]	—	—	25	59.88	79.70	56.35	80.08	57.04	80.00	56.04	0.56× / 1.29×
Loop + LG [36]	—	—	25	60.42	80.06	57.90	79.63	58.56	79.71	72.46	0.44× / 1.00×
Ours (True)	59.88 / 79.70 / 0.56×	Speed	15	64.50	80.60	55.53	79.41	57.30	79.64	<u>25.53</u>	<u>1.24×</u> / <u>2.84×</u>
		Quality	25	63.38	<u>81.52</u>	57.27	79.86	58.48	80.18	41.85	0.75× / 1.73×
Ours (Strong)	65.17 / 81.21 / 0.80×	Quality	20	66.45	**81.64**	**59.53**	80.21	61.03	<u>80.49</u>	44.17	0.72× / 1.64×
Ours (Ultra)	71.31 / 83.26 / 0.92×	**Speed**	9	60.60	81.07	57.70	<u>80.24</u>	58.32	80.40	**16.09**	**1.96× / 4.50×**
		Balanced	15	<u>68.74</u>	80.95	<u>59.45</u>	**80.45**	<u>61.47</u>	**80.55**	27.73	1.14× / 2.61×
		Quality	16	**70.49**	80.84	59.29	80.08	**61.63**	80.22	31.23	1.01× / 2.32×

Comparing True, Strong, and Ultra reveals how target ambition interacts with algorithmic headroom in image diffusion. Under the unshifted True target, search settles conservatively on 15–25-step schedules (25.53–41.85 TFLOPs; held-out GenEval 55.53–57.27). By contrast, aspirational targets push the search LLM past local optima to invent lower-compute strided LoopGuidance with cross-step velocity-delta reuse (16.09–31.23 TFLOPs), lifting held-out GenEval to 59.53 (Strong) and held-out DPG-Bench to 80.45 (Ultra Balanced) while cutting Ultra Speed cost to 16.09 TFLOPs (1.96× fewer FLOPs than Baseline and 4.50× fewer than Loop + LG). Because Strong retains the highest held-out GenEval and Ultra expands the lower-compute Pareto frontier, target strength acts as an exploration catalyst whose optimal level depends on the operating priority.

C.6. Search-Target Calibration in Autoregressive Text Reasoning

Table S12 expands the text-reasoning target study in RQ2 and RQ4. Read panel (a) as development (829 questions) and full-benchmark individual scores, and panel (b) as separate 3,316-question held-out and 4,145-question full-benchmark aggregates. Each `Gemini-4-Argon` search evaluates 80 candidates on 829 development questions, then replays frozen programs on all 4,145 questions. The target is development Acc Norm, with shifts in percentage points rather than the

three-coordinate target strength percentage for image diffusion. Non-looped is standard Qwen3-4B inference; Fixed Loop follows Training-Free Looped Transformers [3]. Ours (True) is the instance-weighted (Micro) program from the True-target search; it is distinct from the task-balanced (Macro) program shown as Ours in Table 1c and Table S8.

Unlike multi-step image diffusion, a frozen unlooped LLM evaluated on discrete multiple-choice questions operates within a narrower test-time gain envelope. Consequently, the grounded True target (+0.00 pp) delivers the strongest generalization to the 3,316 unseen questions—achieving the highest held-out Micro Acc (65.29%), Macro Domain (60.16%), and full-set decision margin (0.3204)—whereas artificially inflated targets (+1.57 to +6.03 pp) push search toward development-set specialization (raising development Acc from 66.10% to 66.47% under Strong while lowering held-out Micro Acc from 65.29% to 64.84%, with all shifted targets falling slightly below Fixed Loop on held-out Micro Acc at 64.84–64.87% vs. 65.26%). Taken together, Sections C.5 and C.6 yield a clear calibration rule: search targets should match the domain's available algorithmic headroom—aspirational targets catalyze algorithmic shortcuts in multi-step diffusion, whereas measured targets protect held-out generalization in tighter discrete-reasoning regimes.

Table S12. **Measured (True) search targets generalize best on Qwen3-4B, illustrating domain-dependent target calibration.** Panel (a) reports development (829) and full-benchmark individual scores; panel (b) contrasts 3,316-question held-out aggregates against full 4,145-question aggregates. Target shifts are percentage points, unlike the relative target strength percentages for image generation. **Bold**/underlined: best/second-best per column.

(a) **Development and Individual-Benchmark Scores.**

Method	Search Target (Acc Norm %, Δ)	Development (%) ↑		MMLU-STEM	GPQA-Main	ARC-Challenge (%) ↑		SciQ
		Acc	Acc Norm	Acc (%) ↑	Acc (%) ↑	Acc	Acc Norm	Acc (%) ↑
Non-looped [37]	—	63.33	63.93	62.43	31.92	50.60	54.18	95.70
Fixed Loop [3]	—	64.17	65.50	64.20	33.26	**51.71**	**54.95**	96.20
Ours (True)	65.50 (+0.00)	66.10	**67.31**	**64.85**	34.60	51.54	**54.95**	**96.50**
Ours (Mild)	67.07 (+1.57)	66.22	**67.31**	64.66	**34.82**	51.02	53.67	96.00
Ours (Strong)	68.52 (+3.02)	**66.47**	67.19	64.79	34.38	51.02	53.24	96.10
Ours (Extreme)	71.53 (+6.03)	65.50	67.07	64.59	34.60	50.60	53.67	96.10

(b) **Held-Out** (3,316 **Unseen Questions) vs. Full-Benchmark** (4,145 **Questions) Aggregate Scores.**

Method	Held-out (3,316) (%) ↑				Full (4,145) (%) ↑				
	Micro Acc	Micro Acc Norm	Macro Group	Macro Domain	Micro Acc	Micro Acc Norm	Macro Group	Macro Domain	Margin ↑
Non-looped [37]	63.93	64.35	60.24	58.46	63.81	64.27	60.16	58.37	0.2919
Fixed Loop [3]	65.26	**65.77**	**61.60**	59.81	65.04	65.72	61.34	59.60	0.3132
Ours (True)	**65.29**	65.71	61.53	**60.16**	**65.45**	**66.03**	**61.87**	60.46	**0.3204**
Ours (Mild)	64.87	65.29	61.16	59.96	65.14	65.69	61.63	60.40	0.3080
Ours (Strong)	64.84	65.17	61.18	59.84	65.16	65.57	61.57	60.25	0.3128
Ours (Extreme)	64.87	65.23	61.10	60.11	64.99	65.60	61.47	**60.51**	0.3092

D. Looped-DiT Case Study: From Diagnostics to Frozen Programs

This case study expands the pretrained-looped image-model comparison in RQ1 (Appendix B.2) and illustrates how programmable diagnostics drive executable edits in structure, dynamics, and control. We run Evolving Loop Programs offline on frozen Looped-DiT-B32 [6] (6 pre-loop blocks, a weight-tied 5-block core repeated K=4 times per step, and 6 post-loop blocks; effective depth 32, with a 100-step / 400-core-pass native reference) using `Gemini-4-Argon` over 80 unique candidates across nine rounds on 320 stratified development prompts (112 GenEval, 208 DPG-Bench; 4 seeds per prompt), holding out 1,298 prompts (441 GenEval, 857 DPG-Bench) strictly for post-search confirmation. Pretrained weights remain frozen throughout, and full protocol details appear in Appendix G.

D.1. Diagnostics, Program Design, and Search Evolution

Programmable diagnostics. Before proposing candidates in R0, the search LLM wrote lightweight diagnostic collectors (defined formally in Appendix E) and verified bit-exact image parity on 16 anchor prompts. The recorded trajectory statistics separated stable vs. unstable reuse axes:

- **Velocity curvature.** Relative velocity second-difference curvature peaks at 0.836 (t=0.02), drops to 0.036 at t=0.10 and 0.018–0.021 across $t \in [0.15, 0.50]$, and rises to 0.100 at t=0.90, motivating nonuniform timestep allocation.
- **Loop-residual stability.** Adjacent steps maintain cosine similarity 0.9978 at t=0.10 and 0.9989–0.9997 across $t \in [0.15, 0.90]$ for $(\mathbf{h}^{(4)} - \mathbf{h}^{(2)})$, while within-step updates $(\mathbf{h}^{(k)} - \mathbf{h}^{(k-1)})$ align progressively with depth (0.746–0.983 at k=4), motivating bounded residual reuse.
- **Guidance-branch variation.** By contrast, cross-step relative variation in the classifier-free guidance difference remains high (0.528 over $t \in [0.0, 0.2]$ and 0.735 at t=0.90), explaining why early R0 attempts at cross-step guidance reuse or asymmetric unconditional-loop pruning degraded GenEval (32.37%–53.30%) and were discarded.

Discovered program mechanisms. Guided by these diagnostics (complementing the direct diagnostic-access ablation on Scale-RAE in Table 2a), the search LLM co-designed four complementary mechanisms across structure, dynamics, and control:

1. **Control: curvature-based timestep allocation.** An SNR shift ($s \in [1.30, 1.38]$) paired with a sinusoidal curvature warp ($c_u \in [0.04, 0.05]$) concentrates steps in early high-curvature regions, raising 25-step development GenEval at fixed K=4 from 84.56% to 85.49% (+0.93 pp).
2. **Dynamics: latent extrapolation in both streams.** On full 4-loop refresh steps over $t \in [0.15, 0.90]$, both image and text states extrapolate along their last core increment $\Delta\mathbf{h}^{(4)} = \mathbf{h}^{(4)} - \mathbf{h}^{(3)}$ ($\alpha \in [0.18, 0.22]$) with zero extra core passes.
3. **Dynamics: tapered x_0 extrapolation.** A step-size-aware second-order Adams–Bashforth update ($\beta \in [0.32, 0.35]$) tapers linearly over $t \in [0.80, 0.94]$ to prevent late-step oversharpening as $(1 - t) \to 0$, lifting the 25-step program to 86.34% development GenEval.
4. **Structure and control: bounded residual reuse.** Alternating $K_t \in \{2, 4\}$ steps along an LCM-aligned stride reuses cached $(\mathbf{h}^{(4)} - \mathbf{h}^{(2)})$ in both streams with cache age capped at ≤ 2 steps, reducing core passes from 80 to 68 at 20 steps and from 56 to 48 at 14 steps.

Search evolution. Across the 80 validated candidates (excluding exact duplicates and three malformed proposals), cumulative development hypervolume over GenEval, DPG, and FLOPs speedup rises by +102.13% (2.8756 → 5.8126). Approximately 97% of this improvement arrives in the first two rounds (R0–R1, 24 evaluations) because Round 0 diagnostics immediately identified early velocity curvature and 0.998+ late-loop residual alignment, enabling the search to land directly on curvature-matched schedules and bounded residual reuse. Later rounds refine the frontier through tapered x_0 multistep updates (R4, 86.34% development GenEval) and stale-cache-age pruning (R6, 85.17% at 7.21× speedup). (Absolute hypervolume is specific to this backbone and reference, distinct from Scale-RAE in Figure 1e; full-benchmark evaluations appear in Tables 1b and S3.)

D.2. Held-Out Confirmation and Schedule-Only Comparison

Table S13 compares frozen Speed, Balanced, and Quality programs with native inference and a 15-step Schedule-only comparator (46 core passes) that omits residual reuse and latent/x_0 updates. Two main findings emerge on the 1,298 unseen held-out prompts. First, development gains generalize with near-zero attenuation and hardware-faithful speedups: Balanced reaches 84.92% held-out GenEval—surpassing the 100-step native reference (84.43%) at 5.53× measured 32 × H100 wall-clock speedup (0.147 s vs. 0.814 s/image)—while Speed raises matched-14-step native scores by +4.21 pp GenEval and +3.96 pp DPG at 7.86× wall-clock speedup (0.104 s/image), with measured wall-clock ratios tracking FLOPs speedups within ∼1%. Second, at matched compute (6.80 vs. 6.85 TFLOPs, ∼7.8× speedup), the complete Speed program exceeds the Schedule-only comparator by +1.32 pp GenEval (83.39% vs. 82.07%) and +1.13 pp DPG (85.28% vs. 84.15%), confirming that co-designed latent/x_0 dynamics and bounded residual reuse yield accuracy unattainable from curvature-aware timestep scheduling alone. As in RQ1, the 100-step native baseline retains a slight DPG lead (85.52% vs. 85.28% held-out) at 4×–7.8× higher compute, and Full merges development and held-out splits (Δ_{step} reports full-benchmark gains over matched-step native K=4 inference).

E. Runtime Observation Definitions

Table S14 formalizes representative runtime probes used in the RQ2 diagnostic loop of Section 3 and Section 4.2—spanning Scale-RAE configuration probes (Appendix F) and guidance or Looped-DiT probes written during program search (Appendix D.1)—to diagnose local block changes, core-pass dynamics, late-loop residual stability, and guidance temporal reuse.

Table S13. **The complete Speed program outperforms the schedule-only comparator at similar compute on held-out prompts.** Δ_{step} reports full-benchmark gains vs. matched-step native inference; Speedup gives FLOPs and 32 × H100 wall-clock ratios (0.814 s/img at 100 steps). Rankings cover all rows; gray marks ours.

Method	#step	Core	GenEval (%) ↑				DPG (%) ↑				TFLOPs ↓	Speedup ↑
		Passes	Dev.	Held-out	Full	Δ_{step}	Dev.	Held-out	Full	Δ_{step}		(FLOPs / Wall)
Native (K=4) [6]	100	400	85.89	84.43	84.65	+0.00	**84.64**	**85.52**	**85.34**	+0.00	53.18	1.00× / 1.00×
Native (K=1) [6]	100	100	84.21	84.68	84.47	−0.18	83.18	84.53	84.26	−1.08	28.98	1.84× / —
Native (K=4) [6]	50	200	85.12	84.78	84.77	+0.00	84.41	85.50	85.29	+0.00	26.59	2.00× / —
Native (K=4) [6]	25	100	84.56	82.88	83.16	+0.00	84.51	84.82	84.76	+0.00	13.30	4.00× / 4.00×
Native (K=4) [6]	20	80	83.21	82.06	82.26	+0.00	84.14	84.19	84.18	+0.00	10.64	5.00× / 5.00×
Native (K=4) [6]	15	60	81.27	79.46	79.80	+0.00	82.77	82.35	82.44	+0.00	7.98	6.67× / 6.67×
Native (K=4) [6]	14	56	80.26	79.18	79.36	+0.00	81.26	81.32	81.31	+0.00	7.45	7.14× / 7.14×
Ours (Speed)	14	48	83.77	83.39	83.43	+4.07	84.40	85.28	85.11	+3.80	**6.80**	**7.82×** / 7.86×
Ours (Balanced)	20	68	85.72	**84.92**	85.03	+2.76	84.51	85.12	85.00	+0.82	9.67	5.50× / 5.53×
Ours (Quality)	25	100	**86.34**	84.84	**85.08**	+1.92	84.36	85.27	85.09	+0.33	13.30	4.00× / 4.01×
Ours (Schedule-only)	15	46	83.99	82.07	82.44	+2.65	83.78	84.15	84.08	+1.64	6.85	7.77× / **7.87×**

The search LLM inspects these diagnostics offline and can revise the collectors; deployed programs run without collecting this full observation suite.

Notation. At sampling step j, let $\mathbf{a}, \mathbf{b}$ be an observed block's input and output. Let $\mathbf{h}_j^{(k)}$ and $\mathbf{z}_j^{(k)}$ be the conditional image and text states after core pass k, with increments $\Delta\mathbf{h}_j^{(k)} = \mathbf{h}_j^{(k)} - \mathbf{h}_j^{(k-1)}$ and $\Delta\mathbf{z}_j^{(k)} = \mathbf{z}_j^{(k)} - \mathbf{z}_j^{(k-1)}$. The late-loop residual is $\mathbf{r}_j = \mathbf{h}_j^{(4)} - \mathbf{h}_j^{(2)}$. For Scale-RAE guidance, $\mathbf{p}_j^L$ and $\mathbf{p}_j^R$ denote looped and reference predictions; $\mathbf{d}_j = \mathbf{p}_j^L - \mathbf{p}_j^R$ is their difference and w is the LoopGuidance weight.

Reductions and availability. Write $\mathrm{rms}(\mathbf{u}) = \sqrt{\mathrm{mean}(\mathbf{u}^2)}$, $\rho(\mathbf{u}) = \max(\mathrm{rms}(\mathbf{u}), \epsilon)$, and $\epsilon = 10^{-8}$. Norms and means cover the observed tensor elements within one example; $\langle\cdot\rangle_B$ averages per-example measurements over a batch. The Scale-RAE guidance probe subsamples up to 4096 flattened elements per example and reports means over finite observations. For cosine similarity, Scale-RAE clamps the product of norms below at 10^{-12}; Looped-DiT clamps each norm below at ϵ. History-dependent measurements require previous observations; late-loop measurements require four passes. Records include step/time and applicable branch/block identifiers.

F. Search LLM Prompts

For the search framework in Section 3, this section reproduces (i) the Round-0 instructions prompt instantiated for our primary Scale-RAE program search (Prompt 1, with internal workspace paths anonymized), (ii) the structured per-turn observation payload returned by the coordinator (Prompt 2), and (iii) the prompt adaptations used across ablations and model families (Prompts 3–4). In the verbatim prompts below, "OPAL" refers to the sandboxed code-and-diagnostics execution profile [2], and coordinator template variables (`$objective`, `$search_prompts`, `$last_group`, `$final_summary`) are resolved prior to Round 0.

Search context. Each search uses one persistent multi-turn conversation and workspace across all nine rounds; separate runs share no context. The workspace retains code snapshots, hypotheses, notes, evaluation history, and diagnostic outputs. `GPT-6-Astra` receives the full evaluation history inline; `Gemini-4-Argon` receives it inline when compact and otherwise a summary with tool access to the full record. Diagnostics provide numerical summaries and saved tensor statistics (Appendix E); generated images and videos are never shown to the search LLM. During search, evaluation and probes use only development inputs with frozen backbone weights. Held-out inputs and scores, full-split scores, ground-truth labels, evaluator internals, external network access, and other runs are excluded from the LLM's prompts and tools.

F.1. Primary Program-Search System Prompt

Prompt 1 reproduces the Round-0 instructions supplied to `GPT-6-Astra` for the Scale-RAE joint program search (instantiated with 320 development prompts across canonical batch groups `0..39`, evaluating $16 + 8 \times 8 = 80$ unique candidates over Rounds 0–8).

Observation	Definition
Local block changes (Scale-RAE)	
delta_rms	$\mathrm{rms}(\mathbf{b}-\mathbf{a})$
relative_delta_rms	$\frac{\mathrm{rms}(\mathbf{b}-\mathbf{a})}{\max(\mathrm{rms}(\mathbf{a}), 10^{-12})}$
cosine	$\cos(\mathbf{a}, \mathbf{b})$
selected_fraction	$\mathrm{mean}(\mathbf{m})$, for the binary token-selection mask $\mathbf{m}$
Changes across core passes (Looped-DiT-B32)	
img_loop_rel_delta_cond	$\left\langle \frac{\lVert \Delta \mathbf{h}_j^{(k)} \rVert_2}{\max(\lVert \mathbf{h}_j^{(k-1)} \rVert_2, \epsilon)} \right\rangle_B$
txt_loop_rel_delta_cond	$\left\langle \frac{\lVert \Delta \mathbf{z}_j^{(k)} \rVert_2}{\max(\lVert \mathbf{z}_j^{(k-1)} \rVert_2, \epsilon)} \right\rangle_B$
img_loop_delta_cos	$\left\langle \cos(\Delta \mathbf{h}_j^{(k)}, \Delta \mathbf{h}_j^{(k-1)}) \right\rangle_B, k \geq 2$
Late-loop residual stability (Looped-DiT-B32)	
res_r2_to_r4_cos_prev	$\langle \cos(\mathbf{r}_j, \mathbf{r}_{j-1}) \rangle_B$
res_r2_to_r4_rel_err	$\left\langle \frac{\lVert \mathbf{r}_j - \mathbf{r}_{j-1} \rVert_2}{\max(\lVert \mathbf{r}_j \rVert_2, \epsilon)} \right\rangle_B$
Guidance magnitude and temporal reuse (Scale-RAE)	
delta_to_loop_ratio	$\frac{\rho(\mathbf{d}_j)}{\rho(\mathbf{p}_j^L)}$
guidance_extra_to_loop_ratio	$\lvert w-1 \rvert \frac{\rho(\mathbf{d}_j)}{\rho(\mathbf{p}_j^L)}$
lagℓ_cosine	$\cos(\mathbf{d}_j, \mathbf{d}_{j-\ell}), \ell \in \{1,2,3\}$
lagℓ_relative_error	$\frac{\mathrm{rms}(\mathbf{d}_{j-\ell} - \mathbf{d}_j)}{\rho(\mathbf{d}_j)}, \ell \in \{1,2,3\}$

Table S14. **Diagnostics measure block updates, loop dynamics, and the stability of reusable corrections.** Field names follow the implemented collectors. Scale-RAE formulas give per-example measurements; Looped-DiT formulas include the batch reduction. These passive probes preserve candidate outputs.

Prompt 1: Primary Program Search Instructions (Scale-RAE, Joint Objective)

```
You are the research agent for a NEW open-code Scale-RAE loop experiment.
Use one persistent gpt-6-astra / xhigh session for the whole experiment.
Optimize GenEval, DPG and inference speed jointly on the fixed development panel using Pareto tradeoffs.
First round: 16 NEW candidates. Then 8 rounds of 8 NEW candidates, 80 total successful candidates.
Use paired feedback. Every candidate needs a
testable loop-related hypothesis. Both single and combined changes are allowed.

The old discrete action space DOES NOT APPLY. You may change the copied loop source code,
add loop methods, change schedules, steps, layers, K/lambda, guidance, token selection/reuse,
state-dependent mechanisms, and add arbitrary diagnostic code. Stay related to loop inference.
Do not change pretrained model weights, language conditions/prompts, noise/seed, precision,
AMP, batch size, decoder, output validation, timing/reporting or official scoring semantics.
Do not train, memorize prompt-specific answers, hard-code benchmark-item behavior, fetch
external weights/data, or optimize the evaluator. Use runtime features for general mechanisms.
Fixed execution: checkpoint tensor dtypes, BF16 AMP, seed42, generation/decode batch8.
Each formal candidate uses 16 warmups followed by 320 scored images, canonical batch groups.
Timing includes candidate hook setup/cleanup. Diagnostics are separate and their cost is logged.
The executor also checks 8 unchanged no-loop images before and after each formal round;
these controls are unscored and do not consume the 80 candidate slots. Preserve disabled-loop
behavior and restore cross-candidate state; a failed control invalidates the round.

You can see the SAME permitted anchors as the previous search: 25-step no-loop and paper
dense loop without LoopGuidance, measured on THIS fixed search panel. Full/held-out results,
other experiments, scorer code/target annotations, private notes, credentials and other sessions
are inaccessible. Do not try to infer or request those files.
```

```
You have native shell, file reads, apply_patch and file writes in your isolated workspace.
Use these tools directly to inspect, implement and test loop code; you do not need to ask
the coordinator to read/edit files for you. Following official OPAL, shell network access,
Web search, browser/apps/plugins and reading authentication/session files are disabled.
Model-service connectivity belongs to the transport, not your shell. No sandbox escape.
Your shell uses the supplied PyTorch interpreter on CPU, as in official OPAL. Task-specific
additions include model assets, the selected 320 conditions and a GPU diagnostic/evaluation worker.
The environment supplies python, torch, numpy and ordinary shell tools. Read-only reference
generation code is at /reference; the selected cache is /inputs/conditions.sqlite3.
Your own experiment's runtime results are read-only at /experiment_results.
You may implement arbitrary feature collectors and run local analysis scripts directly.
The managed diagnostic runner below executes your custom code on the GPU in the exact
generation pipeline, returns observations and records its cost. It is not a fixed list of
allowed features. Returned tensors/statistics can also be analyzed by your own shell scripts.
Source, notes and small results can live in this workspace; use scratch/ for large temporary
outputs. Reference inputs and official evaluation infrastructure are not editable.

Use native tools freely during each turn. End the turn with one JSON object for the
coordinator, containing experiment_id, round, rationale and ONE request below. This final
JSON schedules evaluation/diagnostics or closes the experiment; it does not restrict your
intermediate tool calls. Read/edit requests remain optional compatibility helpers.

1. Optional read-code helper (native shell reads are preferred):
{"request":"read","files":[{"path":"<loop_patch_module>","start_line":1,"line_count":250}]}
Read at most 6 files/request, up to 500 lines each. The observation includes available file names.

2. Optional edit helper (native apply_patch/shell edits are preferred):
{"request":"edit","edits":[{"path":"candidate.py","content":"complete UTF-8 Python source"}]}
Or exact replacement: {"path":"...","old":"exact existing text","new":"replacement"}.
Editable: candidate module, loop library modules, and notes.
Do not edit immutable interface/reference files. Changes are syntax checked and snapshotted.

3. Run your diagnostic code on any fixed-panel groups 0..39:
{"request":"diagnostic","batch_groups":[0],"actions":[ACTION,...],"question":"specific hypothesis"}
Use only as many samples as needed. Write results under context["output_dir"] from your
hook's close() method. Any data observed during a DiT forward is available to your custom
collector: inputs, outputs, intermediate activations, Q/K/V, attention logits/probabilities,
AdaLN/modulation values, residuals, guidance branches, token decisions, or other internal
tensors and state. These examples are not an allowlist. Add hooks or instrument the copied
code wherever necessary. You may save full tensors, e.g. .pt/.npy/.npz/.safetensors files,
and inspect them with your native shell/PyTorch tools after the worker returns. Small JSON
summaries are inlined; large JSON/tensor files are returned by path under
/experiment_results/<run_id>/diagnostics/. Message-size limits do not restrict feature
content or file size. Prefer selective capture when sufficient for iteration efficiency.
Diagnostics return real runtime artifacts plus costs; no hypothetical features.
Start by installing a lightweight DiT feature collector and requesting the no-loop and paper
anchors on group0. Confirm those images are unchanged before the first formal 16 candidates.
Then request more diagnostics whenever they can inform the next decision; there is no old
fixed feature allowlist. Full tensors are expensive: summarize meaningful norms/deltas/cosines,
attention/token statistics, guidance branches or convergence first. You may implement new probes.

4. Run this round's candidates:
{"request":"evaluate","actions":[ACTION,...],"decision_summary":"why these candidates"}
Exactly 16 actions in round0 and 8 in rounds1..8. The executor freezes the entire current code
snapshot plus every action before running. Later rounds may edit the code again. Use params to
support multiple algorithm variants in one snapshot. Never silently apply one candidate's hooks
to another. Restore overrides/hooks and reset transient state on close().

ACTION schema:
{"id":"unique_lowercase_id","step_count":10,
 "base_variant":"step25_b12-27_euler_k4_lam100_t0-50_full",
 "runtime":{"num_loops":3,"block_indices":[12,13,14,15,16,17,18,19,20,21,22,23]},
 "params":{"arbitrary_new_loop_parameter":1.2},
 "hypothesis":"mechanism and expected effect on the search objective", "changed_fields":["..."],
 "parent_id":"anchor or earlier candidate"}
base_variant supplies built-in defaults; runtime may override any existing LoopConfig field.
step_count is any positive integer, not the previous discrete list. Put NEW fields in params.
Custom code sees them via candidate.configure(model, params, context). Return None or an object
with close(). configure() is called once per generation batch after built-in loop patching;
the returned handle is closed after inference. context contains step_count, variant, diagnostic,
output_dir. Calls execute under the existing AMP context. You can also modify the loop modules
directly. Your hooks must not disable AMP or mutate fixed dtype/weight policies.

5. After all rounds:
{"request":"finish","summary":"quality, speed and balanced representatives, evidence, limitations"}

If an executor defect outside your editable code prevents progress, return
{"request":"blocked","reason":"exact failure and required runtime repair"}.
```

```
The coordinator will retain this session without claiming completion or forcing more
GPU/model-call retries. Do not resubmit an unchanged infrastructure failure repeatedly.

Always identify expected changes, observed evidence, failures and uncertainty. Missing results
are not zeros. Do not call the development subset a held-out test. Do not assert significance
from overlapping confidence intervals or repeated adaptive selection. Avoid repeated identical
code+effective configurations. Failures return observations and may be repaired; do not pretend
they succeeded. Input/output tokens, public responses/reasoning summaries, requests, code diffs,
configuration/source hashes, GPU metrics and wall times are recorded. Hidden reasoning is not
requested or exported. Prioritize GPU iteration efficiency.
```

F.2. Per-Turn Observation Payload

On every turn, the coordinator serializes the current experiment state into a structured JSON observation payload and passes it as the user message. Single-objective runs replace `pareto` with `best_geneval` or `best_dpg`, and no-diagnostics runs withhold `worker_log_tail` on infrastructure failures. Prompt 2 shows the canonical JSON turn payload at Round 0 initialization together with the `feedback` fields updated after subsequent `diagnostic` and `evaluate` turns.

Prompt 2: Per-Turn Observation JSON Schema and Round-0 Initialization Payload

```
{
  "protocol": {
    "experiment_id": "<run_id>",
    "agent_backend": "codex",
    "model": "gpt-6-astra",
    "effort": "xhigh",
    "round_sizes": [16, 8, 8, 8, 8, 8, 8, 8, 8],
    "batch_size": 8,
    "amp": "bf16",
    "precision": "checkpoint_actual",
    "seed": 42,
    "search_prompts": 320,
    "scored_splits": ["geneval", "dpg"],
    "diagnostics_enabled": true,
    "objectives": ["geneval", "dpg", "speed_ratio"],
    "action_space": "unrestricted_loop_code_and_configuration",
    "anchors": ["step25_disabled", "step25_b12-27_euler_k4_lam100_t0-50_full"],
    "anchor_run": "<anchor_run_id>",
    "synthetic_anchor_tier": null,
    "held_out_access": false,
    "old_experiment_access": false,
    "evaluator_access": false,
    "panel_sha256": "<sha256_of_320_prompt_manifest>"
  },
  "round": 0,
  "available_trial_slots": 16,
  "files": [["<candidate_module>", "<sha256>"], ["<loop_patch_module>", "<sha256>"], "..."],
  "trials": [
    {"variant": "step25_disabled", "run_id": "<anchor_run_id>",
     "geneval": 0.5380, "dpg": 0.7755, "seconds_per_image": 0.2828, "speed_ratio": 1.0000},
    {"variant": "step25_b12-27_euler_k4_lam100_t0-50_full", "run_id": "<anchor_run_id>",
     "geneval": 0.5988, "dpg": 0.7970, "seconds_per_image": 0.5042, "speed_ratio": 0.5609}
  ],
  "pareto": ["step25_disabled", "step25_b12-27_euler_k4_lam100_t0-50_full"],
  "feature_chain_verified": false,
  "completed_rounds": [],
  "feedback": {
    "phase": "initialize",
    "anchors": {"...": "subtask/domain scores and timing for measured anchors on dev split"},
    "instruction": "Read needed loop code, implement an informative lightweight feature collector, request both
  anchors on group0; then propose 16 new candidates."
  }
}

// After a "diagnostic" request:
//   feedback = {"both_anchors_exact": true, "artifacts": {...}, "artifact_files": [...], "cost": {...}}
//   (when both_anchors_exact && non-empty artifacts, feature_chain_verified becomes true)
// After an "evaluate" request completes Round r:
//   completed_rounds.append({"round": r, "run_id": "<id>", "configs": N, "code_revision": "<sha256>",
  "joint_hypervolume_increment": <delta_hv>})
//   feedback = {
//     "last_round": {"trials": [...], "comparisons": {"vs_baseline": {...}, "vs_paper": {...}},
  "subtask_breakdowns": {...}},
```

```
//     "instruction": "Iterate on the loop code/configuration using evidence; diagnose when useful. Finish only
  after all 9 rounds."
//   }
```

F.3. Backbone and Ablation Prompt Adaptations

Prompts 3a–3c record the controlled ablation variations in RQ2 (Table 2, Figure 1e, and Table 2c); Prompts 4a–4c record the backbone-specific instructions for the remaining image, text-reasoning, and video architectures in RQ1, RQ3, and RQ4.

Prompt 3a: ELP w/o Diagnostics Adaptation (RQ2)

```
Diagnostics are disabled. Submit formal candidates directly. Do not run exploratory
model forwards, add instrumentation to collect activations or internal statistics,
or try to recover worker logs or artifacts. Static code inspection and CPU unit tests
of your inference code are available. Candidate inference may use its internal state
to compute its output within the forward pass. The evaluator returns formal benchmark
feedback only.
```

Prompt 3a replaces lines 60–76 (`diagnostic` interface) in Prompt 1 and omits `feature_chain_verified` checks, while retaining the identical 9-round, 80-candidate budget and editable loop source rules. Prompt 3b shows the coordinator substitutions for the two single-objective runs (GenEval-only and DPG-only) and the anchor blocks distinguishing the `Gemini-4-Argon` search-target calibration runs (True, Strong, and Ultra).

Prompt 3b: Single-Objective and Search-Target Calibration Adaptations (RQ2)

```
--- Variant A: GenEval-only search ($search_prompts = 112, $last_group = 13) ---
$objective := Maximize official GenEval overall (equal-task macro mean) on the fixed GenEval development panel.
  GenEval is the sole selection objective. Inference timing is reported as telemetry.
$final_summary := top GenEval programs, evidence, limitations

--- Variant B: DPG-only search ($search_prompts = 208, $last_group = 25) ---
$objective := Maximize official DPG-Bench overall (mean official per-prompt score) on the fixed DPG-Bench
  development panel. DPG-Bench is the sole selection objective. Inference timing is reported as telemetry.
$final_summary := top DPG-Bench programs, evidence, limitations

--- Variant C: Search-target anchor blocks (Gemini-4-Argon, 320-prompt dev panel) ---
[True target (2 anchors):]
1. step25_disabled: 25-step no-loop baseline (GenEval=0.5380, DPG=0.7755, 0.2828 s/img, speed_ratio=1.0000).
2. step25_b12-27_euler_k4_lam100_t0-50_full: 25-step paper dense loop without LoopGuidance
   (GenEval=0.5988, DPG=0.7970, 0.5042 s/img, speed_ratio=0.5609).

[Strong target (3 anchors, target strength +17.86%):]
3. step20_open_ref_strong_loop: 20-step strong reference loop baseline
   (GenEval=0.6517, DPG=0.8121, 0.3535 s/img, speed_ratio=0.8000).

[Ultra target (3 anchors, target strength +29.28%):]
3. step18_open_ref_ultra_loop: 18-step strong reference loop baseline
   (GenEval=0.7131, DPG=0.8326, 0.3074 s/img, speed_ratio=0.9200).
NOTE: For the initial group0 diagnostic verification (both_anchors_exact), still request the two
built-in physical anchors (base_variant="step25_disabled" and
base_variant="step25_b12-27_euler_k4_lam100_t0-50_full", both with step_count=25 and no runtime
overrides) so the coordinator can verify exact byte preservation against the physical reference images.
```

For Configuration Search (RQ2), the session prompt locks the Scale-RAE weights, 320-prompt panel, 9-round schedule ($16 + 8 \times 8 = 80$ candidates), and BF16 AMP settings, but prohibits shell access, file edits, and custom feature hooks. Instead, proposals are restricted to the fixed discrete action grid and JSON response schema in Prompt 3c, with fixed `dit_features` summary queries capped at ≤ 2 diagnostic calls per round after initial anchor parity verification.

Prompt 3c: Configuration-Search Discrete Action Space and JSON Turn Schema (RQ2)

```
--- Permitted discrete action space (28 loopable DiT blocks 0..27; loop_operator="euler") ---
step_count: [10, 15, 20, 25, 30]
layer_start: [8, 10, 12, 14, 16, 18, 20, 22, 24, "all"]
layer_count: [4, 8, 12, 16]   # null when layer_start="all" (expands to b0-27)
k: [2, 3, 4, 6, 8]
total_lambda_pct: 100
loop_schedule: ["t0-30", "t0-40", "t0-50", "t0-60", "t0-75"]
token_operator: ["full", "abaCB"]
selector: ["r", "rt", "adaln", "aorwn"]   # required for abaCB; null for full
```

```
aratio_pct: [50, 70, 85]   # required for abaCB; null for full
loop_guidance: ["off", 1.5, 2.0, 2.8, 4.0, 6.0, 9.0]

--- Strict JSON-only turn response schema (no shell or file editing; <= 2 fixed probes/round) ---
{
  "experiment_id": "<run_id>",
  "round": 0,
  "analysis_summary": "...",
  "diagnostics": [{"type": "dit_features", "variants": ["known_variant"],
                   "layers": [12, 20], "steps": [0, 6, 12], "batch_groups": [0], "hypothesis": "..."}],
  "next_actions": [{"variant": "<variant_grammar>", "parent_variant": "<id_or_null>",
                    "role": "improvement|exploration|control", "changed_fields": ["..."],
                    "hypothesis": "...", "decision_rationale": "..."}],
  "hypothesis_register_updates": [],
  "final_summary": ""
}
```

Prompts 4a–4c reproduce the architecture-specific blocks for the remaining diffusion and autoregressive models. Notice that the two image diffusion backbones use the same `diagnostic` / `evaluate` / `finish` loop and `candidate.configure(model, params, context)` API as Prompt 1, differing primarily in backbone interfaces and physical anchors.

Prompt 4a: RAE (615M DDT) and Looped-DiT-B32 Backbone Protocols (RQ1)

```
--- RAE-SigLIP2-B/DDT 615M (GPT-6-Astra, 320-prompt dev panel) ---
The permitted anchors are measured on THIS search panel: 50-step no-loop, paper Dense Loop,
and paper Dense LoopGuidance9. The paper loop uses encoder blocks 15..22, K4, total strength 1,
layerwise Euler residuals and sampling progress [0,0.5]. CFG is disabled; output is 256x256.
The model has 28 encoder blocks processing 256 image + 4 time + 256 text tokens, then 2 decoder blocks.

candidate.configure(model,params,context), called once per generation batch AFTER the built-in loop patch.
Return None or a handle with close(). The handle may also expose model_fn to wrap context['model_fn'].

--- Looped-DiT-B32 (Gemini-4-Argon, 320-prompt dev panel, 4 measured native anchors) ---
- Model: sensenova/Looped-DiT-B32 (LoopedMMDiT, 512x512 pixel-space flow-matching MMDiT,
  patch_size=32, 16x16=256 image tokens, 256 FLAN-T5-large text tokens, 2 text preamble blocks,
  loop_split=(6, 5, 6) pre/core/post DoubleStreamBlocks, trained with num_loops=4 and deep supervision).
- Checkpoint dtype: BF16 EMA weights with BF16 CUDA autocast; noise_scale=2.0, official cfg_scale=6.0,
  samples_per_prompt=4, prompt_batch_size=16 (effective image batch = 64, fused CFG batch = 128).
- 4 initial anchors: steps100_loops4, steps50_loops4, steps25_loops4, steps100_loops1.
- Custom sampler hook: optional handle.sample_batch(model, text, text_mask, x_init, *, step_count,
  cfg_scale, num_loops, context) -> Tensor returning [B, 3, 512, 512] in [-1, 1]; step_count in 1..200,
  runtime.num_loops in 1..32; Pareto HV over (geneval, dpg, min(t_steps100_loops4 / t_candidate, 8.0)).
```

For autoregressive text reasoning on `Qwen3-4B`, the search prompt evaluates an 829-question 4-benchmark development panel under a strict `quality_only_no_time` objective and enforces genuine state recurrence (`require_loop = true`, audited via `loop_mechanism` metadata and `context['loop_trace']`). For the search-target calibration ablation in Table S12, identical search rules hold except that the `paper` anchor's target `acc_norm` coordinate is shifted by $+1.57$, $+3.02$, or $+6.03$ percentage points.

Prompt 4b: Autoregressive Text Reasoning Instructions (Qwen3-4B, RQ4)

```
You are the research agent for an open-code training-free LLM loop experiment on Qwen/Qwen3-4B
  (Qwen3-4B-Instruct, 36 decoder layers 0..35).
Use this one persistent session for all rounds on the frozen 829-question (20%) multi-task zero-shot
  multiple-choice loglikelihood search panel:
- mmlu_hard_stem (10 subjects, 305 questions)
- sciq (200 questions)
- arc_challenge (234 questions)
- gpqa_main (Biology, Chemistry, Physics, 90 questions)

The two initial anchors are:
1. "baseline": unmodified Qwen/Qwen3-4B (loop_config: null)
2. "paper": paper Euler loop at inclusive zero-based layers [15, 18], num_loops=3, strategy="euler",
  iteration_mode="block", cache_strategy="first", decode_mode="full"
Anchors do not consume candidate slots.

CRITICAL OPTIMIZATION OBJECTIVE -- QUALITY ONLY (TIME IS NOT INCLUDED):
- Inference time / latency is NOT part of the optimization objective (optimization_objective =
  "quality_only_no_time").
- Timing fields are intentionally excluded from the trial observations, and there is NO accuracy-vs-time
```

```
  hypervolume or latency penalty.
- Do NOT propose reduced-loop, shallow, or early-exit candidates merely to save inference time.
- Optimize prediction quality purely across: acc_norm, accuracy, macro_group_accuracy, margin_score,
  logprob_score.

OPEN-CODE ALGORITHM FREEDOM & MANDATORY LOOP REQUIREMENT (require_loop = true):
- You may edit the candidate module, loop library modules, and notes.
- EVERY formal candidate MUST contain a real, substantive loop that repeatedly applies a transformer layer,
  window, or model-derived operator to an evolving hidden state, and the iterated result must contribute to the
  returned hidden states/logits.
- Pure layer skipping/pruning, single-pass depth changes, static residual scaling without recurrence, or dummy
  discarded loops are ineligible.
- Every evaluate action MUST include a "loop_mechanism" object:
  {"operator": "...", "state_update": "...", "output_dependency": "...", "scope": "..."}
- Original decoder layer/submodule forwards are observed automatically by the loop audit. For a custom
  functional loop body that bypasses original module forwards, wrap its substantive loop body via
  context['loop_trace']('body_name', body_callable) during configure and call the returned callable each
  iteration.
```

Finally, Prompt 4c summarizes the loop-invariance contracts across the three video architectures: Wan2.1-1.3B short-video diffusion (enforcing fixed unit-sum residual strength λ=1 with a required `lambda_contract`), Self-Forcing 1.3B (constraining each candidate to a pure `def loop(x, evaluate, info, params, memory)` update with deferred chunk KV commits), and LongLive 1.0/2.0 RememBench search (searching unit-budget layerwise or contiguous-window Euler loops over layer spans, repetition counts, active denoising steps, and chunks across 20 development scenes to jointly improve departure-to-revisit CLIP and LPIPS).

Prompt 4c: Video Diffusion Loop Contracts (Wan2.1-1.3B, Self-Forcing 1.3B, LongLive 1.0/2.0, RQ3)

```
--- Wan2.1-1.3B Short Video (unit-sum residual rule) ---
Total loop residual lambda is FIXED at 1 (strength=1, total_lambda_pct=100). Each of K equal residual
substeps uses 1/K. Unequal/adaptive executed residual weights must still sum to 1 for every enabled
loop. Do not change total lambda via per-layer/per-step scaling or early stopping. LoopGuidance is a
separate coefficient, not residual lambda. Include a concise lambda_contract explaining the executed
coefficient sum with every evaluate request; code is retained for protocol review.

--- Self-Forcing 1.3B Causal Chunked Video (pure AST-checked loop API + KV transaction) ---
Implement:
  def loop(x, evaluate, info, params, memory):
      ...
      return {"residuals": [r0, r1, ...], "weights": [w0, w1, ...]}
Only imports torch and math are allowed. Weights must be finite scalar numbers summing to 1 (lambda=1);
negative coefficients are allowed for LoopGuidance or secant corrections (magnitude <= 8). Self/block
repeated evaluations use a cache transaction: the first evaluation commits native current-chunk K/V;
subsequent refinement evaluations restore that committed slice and indices. Clean-context refresh always
runs the native model through all 30 layers; it is never looped, skipped or approximated.

--- LongLive 1.0 1.3B / 2.0 5B RememBench (20 dev scenes; 80 candidates) ---
Propose recurrent loop programs over the 30-layer causal denoiser to improve departure-to-revisit
consistency (maximizing mean CLIP and minimizing mean LPIPS) on the frozen 20-scene development split.
Loop either individual layers (layerwise) or a contiguous sequence of layers (windowwise) for K in
2..8 passes, applying unit-budget Euler updates x_next = x + (F(x) - x) / K across selected denoising
steps (0..3) and autoregressive chunks (0..11 for 2.0; 0..31 for 1.0). Defer causal KV-cache commits
to the final logical pass of each physical layer, and leave native clean-context refreshes unmodified.
On-demand read-only diagnostics may probe layer activations, residuals, alignment, repeat dynamics,
and KV boundaries between rounds without consuming candidate slots.
```

G. Experiment Settings and Reproduction Details

G.1. Search Protocols, Inference Specifications, and Hardware

Tables S15 and S16 summarize search settings and replay specifications for all nine frozen backbones. Read the modality groups in Table S15 to compare edit scope, data partitions, and objectives; Table S16 records sampling, hardware, and native compute references. Full evaluations combine development and held-out partitions and therefore include search examples. All entries describe the programs evaluated in the manuscript.

Shared search budgets and LLMs. Unless noted here, each search evaluates 80 unique candidates over nine rounds ($16 + 8 \times 8$). Dory open-ended search instead uses five rounds of eight candidates (40 total). Anchor counts in Table S15 are per run; "2 + 1 synthetic" means two measured anchors plus one shifted target. The table abbreviates `GPT-6-Astra`

(xhigh) as Astra, Gemini-4-Argon (high) as Argon, and Gemini-4-Carbon (high) as Carbon. "Program + probes" denotes program editing with diagnostics; the configuration and zero-probe ablations are marked explicitly.

Image-search objectives and confirmation. Here, s_{lat} denotes measured warm-batch latency speedup; the joint image objective is the three-dimensional Pareto vector of GenEval, DPG, and s_{lat}. Replay tables report inference FLOPs separately. GenEval-only and DPG-only optimize one score, retain latency only as telemetry, and cross-evaluate the other benchmark. Held-out and full results are reported for every setting; development results are also reported for Scale-RAE main/Argon, RAE, Looped-DiT, Qwen, and Wan. RAE uses the DiffusionBench RAE-SigLIP2-B/DDT 615M adaptation documented below.

Text- and video-search objectives. Qwen's five quality coordinates are acc_norm, accuracy, macro_group_accuracy, margin_score, and logprob_score; no cost coordinate is optimized. Mild/Strong/Extreme shift the target by $+1.57/+3.02/+6.03$ pp. Qwen and Dory STEM use the four-benchmark local suite detailed below; Dory open-ended uses IFBench, GPQA Diamond, and SciCode, with eight runs per prompt. Its STEM cost is measured in GFLOPs/token. Long-video search optimizes clip_mean and $1/(1+\text{lpips_mean})$, excludes cost, and reports TempSSIM post hoc; the unscoped LongLive 2.0 variants correspond to Table S4(c). Wan uses the local vbench_expanded_text_v1 collection and a native 50-step anchor. Speed/Balanced target higher quality_score and semantic_score with lower measured latency; Quality/Quality-subscore target those scores plus total_score, without cost.

Software and rollout details. Table S16 separates execution settings from the native compute references. Scale-RAE, RAE, and Wan retain checkpoint tensor dtypes under BF16 autocast. Qwen3-4B uses torch 2.8.0 and transformers 4.57.1 with sdpa; Dory STEM uses multiple-choice log-likelihood with torch 2.7.1 and transformers 4.57.6. Dory open-ended generation uses vLLM with CUDA graphs and 8 GPUs per candidate, temperature 1.0, and top-p 0.95. The native Dory $K = 6$ reference attains 60.24% STEM accuracy. Each long video lasts 15.79 s, with evaluation at frames 86 and 378. LongLive 2.0 uses 12×8 latent-frame chunks; LongLive 1.0 uses 32×3. Self-Forcing uses a 21-frame attention window. Wan videos last 5.0625 s.

G.2. Evaluation Scopes, Asset Provenance, and Rollout Conventions

Text and short-video evaluation scope. The text suite contains a local 1,525-question STEM subset of MMLU [15], 448 GPQA-Main questions, 1,172 ARC-Challenge questions, and 1,000 SciQ questions. "MMLU-STEM" in the tables denotes this local subset, rather than a separate benchmark defined by the original MMLU paper. The benchmark families follow Training-Free Looped Transformers [3]; its reported gains on knowledge-intensive tasks motivate our STEM focus. The local subset and aggregate summaries are specific to this study, rather than an exact reproduction of that paper's evaluation suite. With benchmark accuracy a_b and count n_b, Micro Acc weights each question equally, $\sum_b n_b a_b / 4{,}145$, whereas Macro Group gives each of the four benchmarks equal weight, $\frac{1}{4}\sum_b a_b$. These definitions reproduce the displayed aggregates up to rounding. Micro Acc Norm uses the normalized-accuracy column where it is reported for ARC-Challenge. Macro Domain is a separate evaluator-reported aggregate and is not interchangeable with the four-benchmark mean (the breakdown below details the 10 MMLU Hard-STEM subjects, SciQ test split, answer scoring, and 15-domain unweighted aggregation).

The short-video results use VBench scoring on a local 256-video collection. This count identifies the evaluated sample, not coverage of all prompts in the official benchmark. All Wan2.1-1.3B baselines and frozen programs are compared under this common 256-video protocol (see Tables S15–S16 and the breakdown below for resolution, frame count, seed, evaluator revision, and baseline reference).

RAE checkpoint and replay provenance. RAE denotes the DiffusionBench [21] RAE-SigLIP2-B/DDT 615M text-to-image adaptation evaluated here, not the official RAEv2 release. The asset manifest records the source and checkpoint identities below. **Source:** diffusion-bench/diffusion-bench
Revision: a4a0f5c22e70f7f253f47a5250b478ae5084acd7
EMA checkpoint: t2i-ddt-en28d1152hd72-dn2d2048hd128-rae-siglip2-vit-b-vpred-t4-v1.pt
Recorded SHA-256: 557c878ed8ab288dadc62a6b08ac21ad720c56165c1ba9d88b77d9dc59a7ef00
Text conditioning uses Qwen/Qwen3-0.6B; decoding uses the SigLIP2-ViT-B decoder. Generation retains the stored checkpoint tensor dtypes with BF16 autocast, seed 42, and generation/decode batches of 16 at native 256×256 resolution. GenEval uses batch size 4; DPG uses question batch size 32 and crop resolution 256. The three control rows and the

Table S15. **Search settings vary by backbone, edit scope, and objective.** Split counts are development / held-out / full. Anchors are measured unless marked synthetic.

Backbone / setting	LLM	Edit scope	Dev / Held / Full	Anchors	Search objectives
Image generation GenEval (553) and DPG (1,065); counts in prompts					
Scale-RAE [32]: config.	`Astra`	Configuration; fixed probes	320/1,298/1,618	2	GenEval, DPG, s_{lat}
Scale-RAE: main	`Astra`	Program + probes	320/1,298/1,618	2	GenEval, DPG, s_{lat}
Scale-RAE: no diagnostics	`Astra`	Program; zero probes	320/1,298/1,618	2	GenEval, DPG, s_{lat}
Scale-RAE: True	`Argon`	Program + probes	320/1,298/1,618	2	GenEval, DPG, s_{lat}
Scale-RAE: Strong / Ultra	`Argon`	Program + probes	320/1,298/1,618	2 + 1 synthetic	Same; shifted target
Scale-RAE: GenEval-only	`Astra`	Program + probes	112/441/553	2	GenEval only
Scale-RAE: DPG-only	`Astra`	Program + probes	208/857/1,065	2	DPG only
RAE [21]	`Astra`	Program + probes	320/1,298/1,618	3	GenEval, DPG, s_{lat}
Looped-DiT-B32 [6]	`Argon`	Program + probes	320/1,298/1,618; four seeds	4	GenEval, DPG, s_{lat} (cap $8\times$)
Text reasoning STEM suite: 4,145 questions; open-ended suite: 563 tasks					
Qwen3-4B [37]: True	`Argon`	Program + probes	829/3,316/4,145	2	Five quality metrics; no cost
Qwen3-4B: Mild / Strong / Extreme	`Argon`	Program + probes	829/3,316/4,145	2 + 1 synthetic	Same; shifted target
Dory1.0-4B [S1]: open-ended	`Carbon`	Program + probes; transitions, KV reuse	113/450/563; eight runs/prompt	2	Accuracy, time, FLOPs; $K{=}2$
Dory1.0-4B: STEM	`Carbon`	Program + probes; depth, transitions	829/3,316/4,145	1–2	Quality–compute at matched $K \in \{2, 3, 4\}$
Long-video generation RememBench T2V (`t2v/test`); counts in scenes					
LongLive 2.0 5B [5]	`Astra`	Block-scoped program + probes	20/80/100	1	CLIP, $1/(1 + \text{LPIPS})$
LongLive 2.0: unscoped	`Astra`	Sub-layer program; with / without probes	20/80/100	1	Same quality pair
LongLive 1.0 1.3B [38]	`Astra`	Block-scoped program + probes	20/80/100	1	Same quality pair
Self-Forcing 1.3B [18]	`Astra`	Program + probes; AST loop API	20/80/100	1	Same quality pair
Short-video generation Local VBench-1.0 (16 dimensions); counts in videos					
Wan2.1-1.3B [31]: Speed / Balanced	`Astra`	Program + probes	48/208/256	1	Quality, Semantic, latency
Wan2.1-1.3B: Quality / Quality subscore	`Argon`	Program + probes	48/208/256	1	Quality, Semantic, Total; no cost

six-/ten-step programs in Table S2 match the saved full-replay reports under this protocol. These results establish behavior of this evaluated checkpoint, not of a later official RAEv2 model.

Evaluation protocols. Table S17 separates the evaluated datasets from their subset-selection rules. Table S18 records evaluator versions and numerical settings, and Table S19 gives the subject-level counts for the STEM suite. Development subsets are used during search; the evaluation scope reported for each experiment is specified in Table S15.

Table S17. **Evaluation protocols separate benchmark coverage from development-subset selection.** Counts refer to prompts, questions, or videos as indicated.

Evaluation	Dataset and size	Scoring or subset selection
GenEval	553 prompts	All prompts scored.
DPG-Bench	1,065 prompts	All prompts scored.
Qwen3-4B STEM	4,145 questions; 829 development	Zero-shot log-likelihood evaluation; stratified 20% development sampling with `seed=1234`.
Dory1.0-4B STEM	Same 4,145-question suite	Reuses the STEM benchmarks listed in Table S19.
Dory1.0-4B generation	IFBench [S2], GPQA Diamond [29], SciCode [S3]	Open-ended generation evaluation.
RememBench	`t2v/test`	Departure/revisit frames 86/378; CLIP features, LPIPS, and TempSSIM.
VBench-1.0	256 videos; 48 development	Extended prompts sampled from a 946-prompt pool with `selection_seed=42`. See the VBench subset explanation.

Aggregation. *Macro Domain* is the unweighted mean accuracy across 15 domains: the ten MMLU Hard-STEM subjects, SciQ, ARC-Challenge, and the three GPQA domains. This differs from the four-benchmark mean described above.

VBench subset construction. The 48-video development panel is selected from a 64-video, 50-step baseline pool within the 256-video evaluation set. Therefore, 208 videos are withheld from candidate evaluation ($256 - 48$), while 192 lie outside the baseline pool ($256 - 64$). These counts describe different exclusions.

Table S18. **Evaluator identities and numerical settings are fixed for reproducible scoring.** Dataset revisions and scorer commits are listed separately.

Evaluator	Implementation or checkpoint	Version and numerical settings
GenEval	`mmdetection 2.28.2`; Mask2Former Swin-S; OpenCLIP ViT-L/14 Checkpoint: `timm/vit_large_patch14_clip_224.openai`	FP32. Repository: `djghosh13/geneval` Commit: `af4902f2`.
DPG-Bench	mPLUG VQA	Crop resolutions: 224/256/512. Repository: `TencentQQGYLab/ELLA` Commit: `3c228f1d`.
Qwen3-4B STEM	`lm_eval 0.4.11`	`apply_chat_template=False`.
RememBench data	`mosaichunk/RememBench` Split: `t2v/test`	Dataset revision: `cdeabf0fce9c075876a903a1d5234483ad677fd5`.
RememBench scoring	MosaiChunk `ClipFeatures`; `LPIPS(net="alex",version="0.1")`; `score_tempssim._win()`	Scorer commit: `0eafcb91`. TempSSIM: `allow_tf32=False`.
VBench-1.0	Vendored VBench evaluator	Revision: `VBench-official-fd18b3d`.

Table S19. **The STEM suite contains 4,145 questions, including 829 development questions.** Full and development counts are shown for each subject or domain; MMLU, SciQ, and ARC-Challenge use their `test` splits.

Subject or domain	Exact task identifier	Full	Dev.
MMLU Hard-STEM			
College physics	`college_physics`	102	20
High-school physics	`high_school_physics`	151	30
High-school mathematics	`high_school_mathematics`	270	54
Abstract algebra	`abstract_algebra`	100	20
College chemistry	`college_chemistry`	100	20
High-school statistics	`high_school_statistics`	216	43
Econometrics	`econometrics`	114	23
College computer science	`college_computer_science`	100	20
Professional medicine	`professional_medicine`	272	55
Medical genetics	`medical_genetics`	100	20
MMLU subtotal		**1,525**	**305**
SciQ	`sciq`	1,000	200
ARC-Challenge	`arc_challenge`	1,172	234
GPQA-Main			
Chemistry	`gpqa_main_zeroshot`	183	37
Physics	`gpqa_main_zeroshot`	187	37
Biology	`gpqa_main_zeroshot`	78	16
GPQA subtotal		**448**	**90**
Total		**4,145**	**829**

Candidate selection, program freezing, and scope boundaries. After Round 8 completes on the development split, the coordinator records search-LLM frontier recommendations and freezes candidate snapshots in replay manifests before confirmation (Section C.2). For *RAE*, replay evaluates the Round 7 6- and 10-step programs (*Speed* and *Quality*) from the agent's 15/10/6-step frontier summary. Finally, *Self-Forcing 1.3B* in Table 1d (0.6585 $\rightarrow$ 0.7617 CLIP) and *Wan2.1-1.3B* in Table S5 report the full frozen evaluations under the RememBench and VBench-1.0 protocols summarized in Table S15.

Video rollout and compute conventions. LongLive 2.0, LongLive 1.0, and Self-Forcing generate chunks autoregressively with diffusion denoising within each chunk. RememBench rollouts follow the official T2V test prompt-switching schedule aligned to chunk boundaries; departure and revisit occur at frames 86 and 378 (zero-indexed). Figure 1d reports $100(o-b)/b$ for CLIP and $100(b-o)/b$ for LPIPS, where b and o are the baseline and program scores. LongLive 2.0 repeats blocks 11–18 with $K{=}6$; LongLive 1.0 uses $K{=}8$ over blocks 11–17 in the first two denoising steps of each chunk. For Self-Forcing, inference FLOPs use the model's configured 21-frame attention window.

Text loop configurations. The Fixed Loop baseline follows the frozen-checkpoint, damped-Euler approach of Training-Free Looped Transformers [3], repeating Qwen3-4B blocks $[15, 18]$ with $K{=}3$. Searched programs use dual-window RK3 updates on Qwen3-4B and elastic recurrent-depth updates with KV reuse on Dory1.0-4B; Table S9 records their detailed settings.

Table S16. **Replay uses fixed sampling and numerical settings within each frozen backbone.** Panel (a) lists execution settings; (b) identifies the native and loop references used for compute comparisons.

(a) Inputs, sampling, and execution.

Backbone	Input / decoding	Samples / seeds	Precision / batch	GPUs
Image generation				
Scale-RAE [32]	224 × 224; DPG crop 224	1/prompt; seed 42	BF16 AMP; FP32 decode; batch 8	1× RTX 5090
RAE [21]	256 × 256; DPG crop 256	1/prompt; seed 42	BF16 AMP; FP32 decode; batch 16	1× RTX 5090
Looped-DiT-B32 [6]	512 × 512; DPG crop 512	4 seeds/prompt; seed 0; 8 GPU RNGs	BF16 AMP; batch 16 prompts (64 images)	32× H100
Text reasoning and generation				
Qwen3-4B [37]	Multiple-choice log-likelihood	1/question; zero-shot; seed 1234	BF16, SDPA; batch 8	1× RTX 5090
Dory1.0-4B [S1]	STEM: log-likelihood Open: generation	STEM: 1/question, zero-shot, seed 1234 Open: 8/prompt	STEM: BF16, batch 8 Open: BF16, 16 seqs/GPU	STEM: 16× H100 Open: 32× H100
Video generation				
LongLive 2.0 5B [5]	1280 × 704; 379 frames at 24 fps	1/scene; official per-scene seeds	BF16; 1 scene/GPU	32× H100
LongLive 1.0 1.3B [38]	832 × 480; 379 frames at 24 fps	1/scene; official per-scene seeds	BF16; 1 scene/GPU	32× H100
Self-Forcing 1.3B [18]	832 × 480; 379 frames at 24 fps	1/scene; official per-scene seeds	BF16; 1 scene/GPU	32× H100
Wan2.1-1.3B [31]	832 × 480; 81 frames at 16 fps	1/prompt; seed 42	BF16 AMP; FP32 time/head/VAE; batch 1/GPU	24×/32× H100

(b) Native configurations and compute references.

Backbone	Reference configuration	Compute
Scale-RAE	25-step Base, guidance 1.0; Loop + LG: blocks 12–27, $K = 4$, LG 2.8	Base: 31.58 TFLOPs Loop + LG: 72.46 Loop w/o LG: 56.04
RAE	50-step Base; Loop + LG: blocks 15–22, $K = 4$, LG 9.0	Base: 29.07 TFLOPs Loop + LG: 54.20 Loop w/o LG: 39.67
Looped-DiT-B32	100 Euler steps, $K = 4$, CFG 6.0, noise scale 2.0	Full reference: 53.18 TFLOPs
Qwen3-4B	Standard inference; Fixed Loop uses Euler updates on layers 15–18, $K = 3$	FLOPs and latency not reported
Dory1.0-4B	STEM: native $K \in \{2, 3, 4\}$ Open-ended: native $K \in \{2, 4\}$	STEM: 11.03/14.74/18.44 GFLOPs/token Open: 96,300/161,198 TFLOPs
LongLive 2.0 5B	No-loop, 4-step chunk UniPC; shift 5; steps $[999, 937, 833, 624]$	Base: 7,621 TFLOPs Ours: 13,716
LongLive 1.0 1.3B	No-loop, 4-step chunk denoising; steps $[1000, 750, 500, 250]$	Base: 4,748 TFLOPs Ours: 7,627
Self-Forcing 1.3B	No-loop autoregressive chunk diffusion	Base: 5,972 TFLOPs Ours: 7,564
Wan2.1-1.3B	50-step UniPC; shift 8.0, CFG 6.0; 30-step control	Base: 28,300 TFLOPs Control: 16,980

G.3. Search Cost and Token Usage

Search is performed offline for each setting. Search LLM usage is similar across tasks, while candidate evaluation runtime varies with the backbone and task. Per nine-round (80-candidate) search, reasoning and code generation by the search LLM take roughly 0.7–2 hours of wall-clock time (2.7 hours with extra diagnostic turns on Looped-DiT-B32). Over a full executable-program search, `GPT-6-Astra` processes roughly 6–20 M input tokens (about 90% served from prompt cache; 1.8 M for restricted configuration search) and generates 0.07–0.19 M output and reasoning tokens, whereas `Gemini-4-Argon` uses roughly 11–33 M uncached input tokens (31–187 M cached reads) and 1.9–4.4 M output and thinking tokens (85 M and 8.1 M in the 22-turn Looped-DiT-B32 run).

Candidate evaluation on the development split accounts for the remaining runtime: on a single RTX 5090 GPU, evaluating 80 candidates takes 1.3–1.8 hours for Qwen3-4B and 4.0–5.9 hours for RAE and joint Scale-RAE (3.4–7.2 hours end-to-end per search), whereas Looped-DiT-B32 (0.9 hours evaluation, 3.8 hours end-to-end) and the video searches distribute candidate rollouts across 32 H100 GPUs.

All search time and token costs are incurred once offline per setting; at deployment, the frozen program runs inside the backbone without calls to the search LLM or associated token overhead.

H. Extended Related Work

This section expands Section 2 of the main paper with a broader synthesis of looped-transformer foundations, dynamic depth architectures, inference-time compute scaling and caching, autoregressive video rollouts, and LLM-guided algorithm discovery. To preserve exact main-paper bibliography numbering, works appearing only in the supplementary material are cited as [S1] and listed in Supplementary References at the end of the appendix.

H.1. Looped Transformers, Weight Sharing, and Latent Recurrent Depth

Foundations of weight tying and continuous computation. Increasing effective depth without growing parameter counts traces back to recurrent and adaptive computation in sequence models. Adaptive Computation Time (ACT) [S15] equips recurrent networks with a learned halting unit to execute variable numbers of transition steps per input, while PonderNet [S6] reformulates adaptive halting as a probabilistic geometric process with unbiased low-variance gradient estimators. Within Transformer architectures, Universal Transformers [8] replace distinct layers with a shared parallel-in-time recurrent block paired with dynamic halting, and ALBERT [S18] demonstrates that cross-layer parameter sharing substantially shrinks encoder memory while preserving downstream accuracy. Pushing parameter tying to the infinite-depth limit, Deep Equilibrium Models (DEQ) [S5] bypass fixed unrolling during training by solving directly for a fixed-point representation $\mathbf{z}^\star = f_\theta(\mathbf{z}^\star, \mathbf{x})$ via root-finding and differentiating through the equilibrium point via implicit differentiation. Complementing discrete equilibrium formulations, Neural Ordinary Differential Equations (Neural ODEs) [S8] reinterpret residual transitions as continuous-time vector fields integrated by adaptive numerical solvers; this differential perspective directly illuminates why discovered ELP rules replace naive fixed-point steps with higher-order Runge–Kutta, Adams–Bashforth, or hyperspherical geodesic transitions. On the theoretical side, Levine et al. [22] establish when depth efficiency outstrips width in self-attention networks, framing why iterative refinement within a block can express compositions unreachable in a single shallow pass.

Expressivity, optimization dynamics, and generalization of looped transformers. Recent theoretical and mechanistic work studies what algorithms looped transformers naturally learn and implement in context. Giannou et al. [S14] construct explicit weight configurations showing that shallow Transformers executed inside an iterative loop act as general-purpose programmable computers capable of running multi-step linear algebra and control-flow routines. For in-context learning, Yang et al. [S30] show empirically that looped Transformers outperform matched-parameter deep Transformers at learning iterative optimization algorithms across diverse function classes, while Gatmiry et al. [S13] prove that weight-tied looped Transformers can represent multi-step preconditioned gradient descent with exponentially better depth efficiency than non-looped counterparts. Investigating out-of-distribution scaling, Fan et al. [S12] demonstrate that adaptive loop iteration substantially improves algorithmic length generalization when test instances require more computational steps than seen during training. From a systems and training-cost standpoint, Xiong et al. [35] analyze scaling laws under truncated backpropagation through time (TBPTT), characterizing how training compute and memory trade off against recurrent unroll length during pretraining.

Pretrained looped backbones and training-free test-time looping. Building on these insights, a wave of pretrained models explicitly builds recurrent depth into large language, diffusion, and world-model architectures. In language modeling,

Huginn [11] scales latent-space reasoning by iterating a weight-tied recurrent core between fixed prelude and coda stacks, Ouro [42] optimizes adaptive recurrent depth allocation during large-scale pretraining, Mixture-of-Recursions [1] couples parameter sharing with token-level recursive depth routing, and Dory1.0-4B [S1] trains an open-weight 4B normalized-GPT natively with elastic recurrent depth. Addressing the conversion cost of existing pretrained models, Relaxed Recursive Transformers [S4] initialize weight-tied blocks from standard LLMs and recover specialization with lightweight layer-wise LoRA adapters, Inner Thinking Transformer [S9] elicits adaptive latent thinking via residual routing over shared layers, and Coconut [S16] feeds continuous hidden states directly back as thought tokens instead of projecting every reasoning step to vocabulary logits. To stabilize multi-pass feedback, Rectified Looped Transformer [23] interpolates intermediate states toward predicted recurrent endpoints. In visual and embodied domains, Looped-DiT [6] trains diffusion Transformers with weight-tied core blocks, intermediate step supervision, and inner-loop attention regulation; LiFT [10] supervises a shared flow Transformer along targets parameterized by continuous computational depth; and LoopWM [25] trains interactive world models with variable test-time loop depth. Finally, training-free looping methods inject recurrence into *frozen, unlooped* checkpoints: Training-Free Looped Transformers [3] formulate hidden-state repetition as damped pseudo-time residual steps, Yan et al. [36] introduce LoopGuidance and block-band repetition for flow-matching generators, and GramLoop [4] gates spatial feature replay using Gram-matrix consistency.

Relation to ELP. Prior looped-transformer approaches either (i) modify pretraining or finetune adapters to learn recurrent dynamics, or (ii) apply a manually chosen, fixed recurrence rule to frozen checkpoints. ELP bridges both regimes with zero weight updates: given any frozen checkpoint—either initially unlooped (Scale-RAE, RAE, Qwen3-4B, LongLive, Self-Forcing, Wan2.1) or natively pretrained with loops (Looped-DiT-B32, Dory1.0-4B)—offline diagnostic-guided program search jointly discovers loop structure (blocks, windows, adaptive K_t), state-update dynamics (Euler, RK3, AB2, dual-stream and geodesic extrapolation), and outer execution/memory controls (deferred causal KV commits, last-loop KV caching, strided D_{ref} reuse, and bounded residual caching).

H.2. Dynamic Depth, Early Exiting, and Conditional Computation

Rather than repeating layers to deepen computation, a complementary line of work skips layers on easy tokens or samples to reduce average cost. In language models, CALM [S26] attaches per-token confidence classifiers to enable calibrated early exiting during autoregressive decoding while preserving consistency across KV states of skipped upper layers. LayerSkip [S11] trains transformers with structured layer dropout and an early-exit loss so lower layers can draft tokens verified via self-speculative decoding against the remaining layers. Mixture-of-Depths (MoD) [S25] replaces fixed uniform computation with a fixed per-layer compute budget routed to the top-k selected tokens, later unified with parameter-shared recursion in Mixture-of-Recursions [1]. Unlike early exiting and learned routers—which require training auxiliary gates, auxiliary losses, or router weights and strictly sub-select from an existing forward graph—ELP acts purely at inference time without weight training, and unifies both *depth expansion* (targeted middle-band recurrence on hard structure-forming steps) and *compute skipping* (skipping late-step loops, reference passes, or redundant core iterations via deterministic schedule conditions).

H.3. Test-Time Compute Scaling and Inference-Time Diffusion Acceleration

Test-time compute in language reasoning. Scaling inference compute beyond greedy single-pass decoding has become central to LLM reasoning. Explicit token-space scaling began with Chain-of-Thought prompting [S29] and evolved into large-scale outcome-verifiable RL policies such as DeepSeek-R1 [S10], which allocate long explicit reasoning chains to difficult problems. Snell et al. [S27] systematically compare proposal, verifier, and sequential revision strategies, proving that compute-optimal test-time allocation can outperform scaling model parameters by several-fold. Whereas token-level test-time scaling pays $O(L_{\text{cot}})$ autoregressive decoding steps and KV-cache growth per trace, recurrent-depth models [11, 23, 42] and ELP's loop programs operate in *continuous latent depth*. Our Dory1.0-4B results (Appendix B.5) further show that smarter latent-state dynamics (bounded geodesic extrapolation plus converged last-loop KV commits) can extract deeper-than-$K{=}4$ reasoning quality at $K{=}2$-level compute without lengthening output token sequences.

Diffusion test-time scaling, fast solvers, and feature caching. In visual generative models, Ma et al. [S23] scale test-time compute by pairing best-of-N and zero-order noise search with domain verifiers, complementing Self-Guided [16] and token-sparse [20] guidance mechanisms. Orthogonal to outer search over initial noises, fast numerical integrators such as DPM-Solver [S22] and UniPC [S32] reduce outer NFEs through exponential integrators and unified predictor–corrector schemes. Training-free caching further amortizes repeated denoiser calls across timesteps: DeepCache [S24] reuses high-level U-Net features along skip branches across consecutive steps, TeaCache [S19] dynamically schedules video-diffusion feature reuse

from timestep-modulated embedding shifts, and TaylorSeer [S21] replaces naive feature copying with Taylor-series forecasting of intermediate Transformer activations. Crucially, hand-crafted caching heuristics apply a uniform reuse assumption across fixed unlooped backbones. In contrast, ELP writes lightweight pass-through diagnostics to test reuse validity empirically (for instance, falsifying lag-≥ 2 caching, linear extrapolation, and cross-step CFG reuse despite 0.995 output alignment in Scale-RAE and Looped-DiT-B32) and co-adapts bounded reuse with recurrent refinement inside a single frozen program.

H.4. Autoregressive Video Diffusion and Long-Horizon Consistency

Generating minute-scale or cyclic camera-revisit videos requires bridging the mismatch between parallel bidirectional diffusion training and causal chunk-wise autoregressive rollout. Diffusion Forcing [S7] introduces independent per-frame noise levels so trajectories can mix planning, stabilization, and streaming causal decoding, while History-Guided Video Diffusion [S28] extends arbitrary-length history conditioning to stabilize transitions and long-horizon memory. To achieve real-time streaming synthesis, CausVid [S31] distills bidirectional video teachers into few-step causal autoregressive generators, Self-Forcing [18] rolls out self-generated frames during distribution-matching training to curb exposure bias, and LongLive 1.0/2.0 [5, 38] scale interactive KV-cached chunk streaming with FP4 kernel optimizations. MosaiChunk and RememBench [40] establish standardized camera-departure and camera-revisit benchmarks alongside spatiotemporal KV memory compositing. Rather than altering training objectives, adding distillation stages, or redesigning KV-eviction topologies, ELP discovers that repeating middle causal blocks with deferred final-pass KV commits slashes revisit drift across frozen LongLive 2.0, LongLive 1.0, and Self-Forcing checkpoints without contaminating future chunks during intermediate loop passes.

H.5. Neural Architecture Search and LLM-Guided Program Discovery

Automated network and algorithm design traditionally framed structural search over differentiable continuous relaxations such as DARTS [S20] or discrete black-box evolutionary controllers that optimize static architecture topologies requiring full training from scratch. With the advent of coding-capable LLMs, evolutionary program search operates directly over readable, executable source code: FunSearch [30] couples LLM mutators with island-based evolutionary evaluators to uncover new combinatorial bounds and online bin-packing heuristics, AlphaEvolve [28] scales full-codebase evolution to hardware kernels and scheduling algorithms, EoH [24] and ReEvo [39] co-evolve natural-language algorithmic concepts and reflective feedback, Eureka [26] synthesizes reinforcement-learning reward functions via vectorized fitness feedback, and Automated Design of Agentic Systems (ADAS) [S17] searches over complete executable agent control flows alongside domain-specific pipelines like DS-Agent [14] and PromptAgent [33]. Moving from blind scalar score evaluation toward mechanistic inspection, OPAL [2] equips a coding agent to inspect neural training diagnostics and rewrite optimizers online, while OODA-Tool [13] structures multi-turn action generation around explicit controller-verified state transitions. ELP adapts diagnostic-driven program synthesis to multi-modal *frozen-weight test-time inference*: all LLM exploration and passive trajectory probing occur once offline under strict split isolation, retaining multi-objective Pareto frontiers whose selected candidates compile directly into zero-overhead, deterministic inference modules.

Supplementary References

[S1] Maxim Fishman, Simeng Sun, Shantanu Acharya, Faisal Ladhak, Brian Chmiel, Santiago Akle Serano, Ron Banner, Boris Ginsburg, and Krishna C. Puvvada. Dory1.0-4B: A looped normalized-GPT. *arXiv preprint arXiv:2611.03998*, 2026.

[S2] Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *NeurIPS Datasets and Benchmarks*, 2025.

[S3] Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, et al. SciCode: A research coding benchmark curated by scientists. *arXiv preprint arXiv:2407.13168*, 2024.

[S4] Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. In *ICLR*, 2025.

[S5] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *NeurIPS*, 2019.

[S6] Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

[S7] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *NeurIPS*, 2024.

[S8] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018.

[S9] Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. In *ACL*, 2025.

[S10] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[S11] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A. Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling early exit inference and self-speculative decoding. In *ACL*, 2024.

[S12] Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *ICLR*, 2025.

[S13] Khashayar Gatmiry, Nikunj Saunshi, Sashank J. Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? In *ICML*, 2024.

[S14] Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *ICML*, 2023.

[S15] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[S16] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E. Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *COLM*, 2025.

[S17] Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *ICLR*, 2025.

[S18] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR*, 2020.

[S19] Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *CVPR*, 2025.

[S20] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019.

[S21] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with TaylorSeers. In *ICCV*, 2025.

[S22] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *NeurIPS*, 2022.

[S23] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps. In *CVPR*, 2025.

[S24] Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *CVPR*, 2024.

[S25] David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

[S26] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *NeurIPS*, 2022.

[S27] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *ICLR*, 2025.

[S28] Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-guided video diffusion. In *ICML*, 2025.

[S29] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[S30] Liu Yang, Kangwook Lee, Robert D. Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *ICLR*, 2024.

[S31] Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Frédo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *CVPR*, 2025.

[S32] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *NeurIPS*, 2023.