EVOLVING PROCEDURAL MEMORY FROM USER TRAFFIC FOR AGENTIC GRAPHIC DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Professional graphic design is a long-horizon agentic task in which structured, editable artifacts emerge from many interdependent actions, yet outcomes admit no reliable programmatic oracle. We introduce a continual adaptation framework in which a frozen frontier model operates professional design software through more than 230 tools, while an external procedural memory of natural-language skills accumulates and refines reusable design procedures from experience. The memory **widens** by acquiring procedures for recurring uncovered subtasks and **deepens** by revising existing procedures against their own successful and failed executions, while a matched replay gate admits only changes that repair failures without regressing observed successes. Five rounds over $1{,}406$ briefs and $1{,}869$ automatically graded trajectories, with no weight updates and no human labels, grow the bank from 76 documentation-derived skills to 139 and raise GenEval2 execution success on `Claude-Sonnet-4` from 72.7% to 99.3% ($+11.99$ points in generation quality), with 61.8% and 67.6% win rates against the no-skill agent across four specialized design benchmarks on `Claude-Sonnet-4` and `Claude-Opus-4.6`. We further show the two mechanisms are effective in combination: on 200 held-out briefs from user-traffic benchmark, widening or deepening alone reaches a 49.4% / 48.6% win rate over the no-skill agent, while their combination reaches 58.5% ($p = 0.025$). Procedural memory offers a practical route to continual adaptation of agents under noisy, unverifiable feedback.

1 INTRODUCTION

Recent generative models can synthesize realistic images from natural-language prompts, but professional graphic design requires structured artifacts that designers can inspect and edit. This has motivated structured graphic-design and layout generation with layered, editable outputs (Yamaguchi, 2021; Hsu et al., 2023; Jia et al., 2023; Inoue et al., 2024; Seol et al., 2024; Hong et al., 2026; Lin et al., 2025; Chen et al., 2025; Lungu-Stan et al., 2026), and agentic systems that construct designs through explicit operations (Wang et al., 2025; Ki et al., 2025). Once creation is represented as manipulable state, design becomes a sequential decision problem: an agent arranges assets, manipulates typography and vectors, builds masks and effects, and revises earlier decisions while preserving editability. Learning this from user traffic is hard: a single design may require dozens of interdependent operations (Ki et al., 2025), so terminal feedback weakly identifies which decisions caused success or failure (Zhang, 2026; Peng et al., 2026; Wang et al., 2026b). Outcomes are also hard to verify: briefs mix concrete requirements (text, colors, placements) with subjective criteria (hierarchy, composition, style) that automated evaluators capture only partially (Wang et al., 2026a; Chang et al., 2025), and unlike code with executable tests, design has no success oracle. Supervised learning therefore needs costly demonstrations, while outcome-based optimization must handle both long-horizon credit assignment and imperfect proxy rewards (Zheng et al., 2023; Huang et al., 2026a), especially when foundation models are externally hosted or impractical to update.

We instead treat the procedural memory surrounding a frozen model as the learning objective: external context can accumulate experience without parameter updates (Suzgun et al., 2026; Zhang et al., 2026b), which agents represent as reusable procedures (Wang et al., 2024a; Forouzandeh et al., 2026; Mi et al., 2026). Creative software has long packaged recurring workflows as replayable routines (spreadsheet macros, Photoshop actions), but we relax the fixed sequence: each procedure is a natural-language guide the model can adapt, reorder, or partially apply (e.g., double exposure:

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

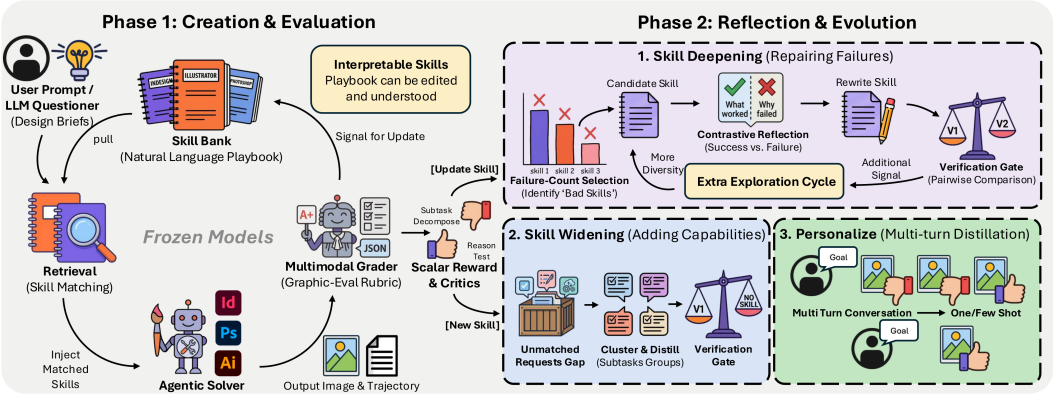


Figure 1: **System overview.** Left — rollout and reward. A brief (from a real User Prompt or in-context LLM Prompter) enters the graphic design agent; the Skill Bank retrieves the top skills with and filters the tool list. The agent executes a chain of tools through intermediate document states to an output image, which the grader turns into a scalar reward. Right — reflection and evolution. Per-skill statistics and recent call histories are the system's assets; skills implicated in failures are edited or rewritten and bad histories are summarized into new skills, each verified by replay gate.

extract a subject, build masks, blend in another asset, refine). A procedure lies between an atomic tool call and an entire trajectory: specific enough to guide execution, general enough to transfer.

We instantiate this approach in a graphic-design agent in which a frozen frontier language model controls equivalents of Adobe Photoshop, Illustrator, and InDesign through more than 230 tools. The memory evolves along two axes: **widening** identifies recurring subtasks in user traffic that the current library does not cover and distills them into new skills, while **deepening** revises existing skills repeatedly associated with failures by contrasting failed executions with successful uses of the same skill. The foundation models, tools, renderer, evaluator, and evolution roles remain fixed; only the skill library changes. But a change should persist only if it improves the system, which is hard because LLM-based evaluators have documented position and order biases (Zheng et al., 2023; Wang et al., 2024b) and a change that helps one request may degrade another. We therefore separate proposal from admission: widening and deepening propose changes from experience (Madaan et al., 2023; Shinn et al., 2023), while a conservative replay gate, inspired by safe policy improvement (Thomas et al., 2015; Laroche et al., 2019), holds upstream context fixed and admits a candidate only when it beats the incumbent on at least one replayed case with no detected regression (Gao et al., 2026). Appendix I studies these procedures to individual user preferences.

We evaluate this loop across five rounds of evolution on user traffic, with no model-weight updates or human reward labels. The evolved skill bank improves multiple frozen backbones across general image-generation and specialized graphic-design benchmarks: on Claude-Sonnet-4, GenEval2 (Kamath et al., 2025) execution success rises from 72.7% to 99.3%, and the evolved agent wins a majority of pairwise comparisons against the same agent without skills. Ablations show that acquiring new procedures and revising existing ones help little in isolation but combine to yield substantially larger gains in task completeness.

Contributions.

- **Skill evolution in a professional graphic-design agent** (§3 & §I). We treat a persistent library of reusable procedures as the object of learning around a frozen foundation model, with mechanisms to acquire and revise procedures from user traffic.
- **Conservative evolution under unverifiable feedback** (§3.3). We introduce a matched replay gate that controls which proposed changes enter the deployed bank under noisy judge and rollout feedback.
- **Coupled acquisition and revision in deployment** (§4). Across five evolution rounds, multiple frozen backbones, and several benchmarks, acquisition and revision are substantially more effective together than either mechanism alone.

2 Background: Agentic System for Graphic Design

We study skill evolution on a professional-level graphic design agent rather than a simplified toy environment. A frozen frontier language model controls equivalents of Adobe Photoshop, Illustrator, and InDesign through more than 230 tools spanning raster editing, vector graphics, page layout, asset retrieval, and verification. Each request invokes an iterative tool-calling loop that constructs a structured, editable artifact. The agent retrieves real assets, renders intermediate document states for multimodal inspection, and supports deterministic offline rendering and evaluation of completed trajectories. Additional details of the underlying agent are provided in Appendix B.

Skill-bank interface. Without skills, the agent selects from the full tool catalog and reconstructs a workflow for each request. The runtime supports progressively disclosed `SKILL.md` playbooks specifying reusable workflows and relevant tools. Before execution, skill retrieval may inject a playbook and reduced tool set into the model context. Neither model weights nor the underlying tools and renderer are modified. Disabling skill retrieval therefore recovers the original agent, providing a natural control for measuring improvements from the evolving skill bank (details in Appendix C).

3 Evolving Loop

Our framework evolves the skill bank through an offline loop (Figure 1) while leaving the model weights unchanged. The loop is organized around four roles:

Role	What it does
Prompter	poses design briefs from user data and LLM
Solver	the agent itself: skill bank + tools → a rendered image
Grader	multimodal; scores each image and says why each unmet requirement failed
Reflector	turns failures into targeted edits of `SKILL.md`

Only the `SKILL.md` files change, along two axes: the bank **widens** by minting skills for uncovered intents (§3.1) and **deepens** by hardening existing skills against failures (§3.2). Personalize is described in Appendix I and excluded from the public skill pool and all main-paper experiments.

3.1 Widening: Minting New Skills from Recurring Uncovered Subtasks

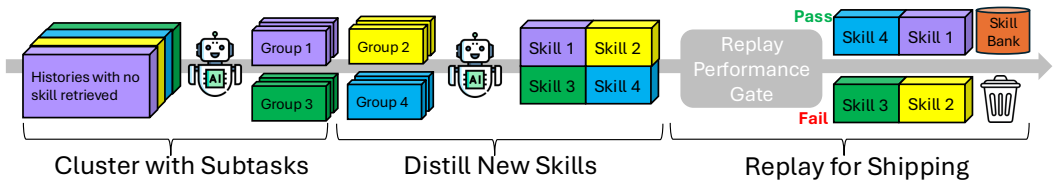


Figure 2: **Widening pipeline.** Uncovered subtasks are clustered into recurring coverage gaps, distilled into candidate skills, and admitted to the bank through the replay gate (§3.3).

For each trajectory, a frozen LLM extracts and canonicalizes the subtasks actually performed from the brief and tool-call sequence. A subtask is uncovered if no retrieved skill addresses it, either because retrieval returns nothing or because the retrieved skills cover a different part of the task. Subtasks associated with a skill blamed for a poor outcome (§3.2) also count as uncovered. Uncovered subtasks accumulate in a persistent coverage pool keyed by canonical label. Once a label reaches $k_{\min} = 3$ occurrences, a frozen LLM distills those cases into a candidate skill. The candidate is admitted only if it passes the replay gate (§3.3) against the no-skill baseline. If rejected, the candidate is discarded but its occurrences remain in the pool, allowing further evidence to accumulate across evolution rounds.

3.2 Deepening: Hardening Skills That Already Exist

Deepening revises existing skills using nothing but their own graded history, and is deliberately asymmetric: selection is a cheap, permissive heuristic, while the gate—not the heuristic—decides

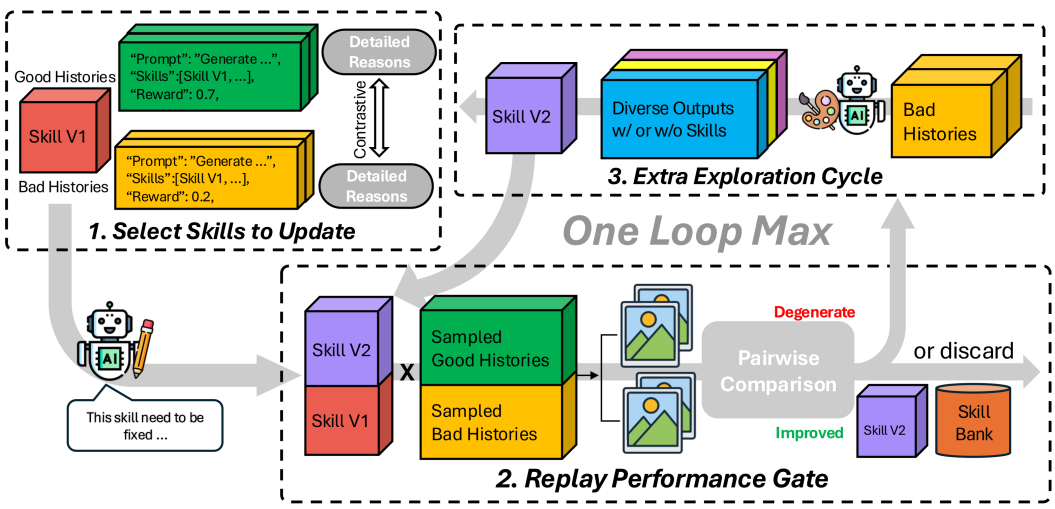


Figure 3: **Deepening pipeline.** Failure-prone skills are revised by contrasting failed and successful histories; a candidate skill enters the bank only if it wins at the replay gate (§3.3) on the skill's worst failing prompts. A rejected candidate triggers an optional one-shot exploration cycle.

what ships. Each trajectory records which skills it retrieved; a trajectory scoring below the success threshold ($s_j < \tau$, $\tau = 0.6$; §E) counts as a failure against every skill it retrieved. We select for revision every skill whose failure-count meets a threshold m (default 2), most-failing first. This failure count is our low-cost prioritization heuristic; prior work instead evolves contextual playbooks or localizes skill passages through paired trajectory contrasts (Zhang et al., 2026b; Gao et al., 2026).

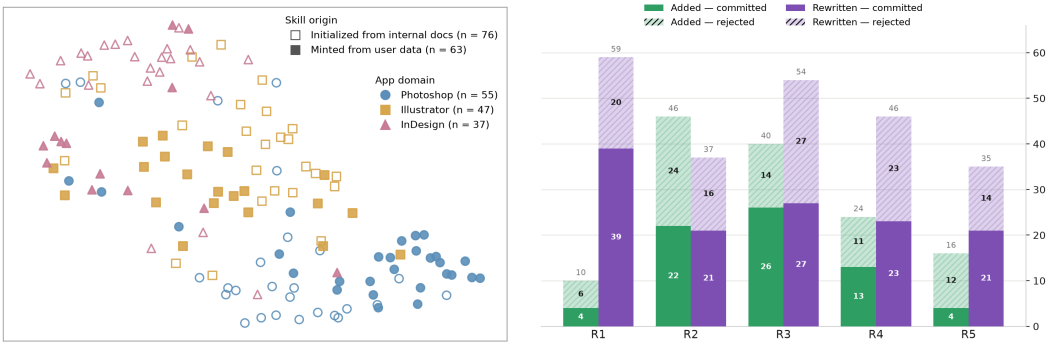
For each selected skill, the Reflector receives the brief, the per-requirement outcomes and "why-bad" rationales, the current `SKILL.md`, and a contrastive set of this same skill's successful calls on similar tasks, represented by the tool-call sequences, intermediate waypoint results, and thinking tokens that actually worked. The successful runs serve as the do-not-regress baseline, while the failure rationales identify what needs improvement, allowing the Reflector to reason over a concrete success↔failure divergence rather than from failure text alone. It emits a targeted edit naming the section and the change; when repeated targeted rewrites of the same skill have failed the gate, it escalates to a major rewrite of the whole skill. Rejected rewrites trigger an optional exploration cycle that probes the skill's failing prompts with and without skills and distills toward whichever arm succeeded, adjudicating the skill's fate as update, delete, or keep—where keep reroutes the unimprovable records into the coverage pool of §3.1.

3.3 Replay Gate

Both axes propose changes; a single gate decides which ones ship. Its design addresses two sources of confounding. First, a VLM Grader's absolute score for the same image drifts across runs, so "accept if the mean score rose" can confuse judge drift with improvement and admit regressions. The gate therefore never uses absolute scores. Second, outcomes depend on more than the skill: asset retrieval and other upstream state can differ between arms, allowing a candidate to win simply because it received better inputs.

We sample prompts that exercise the skill and generate several contexts per prompt, each with distinct retrieved assets and upstream state. Each context is then frozen and replayed fresh in the same batch under both arms: the candidate versus the incumbent for a rewrite, or versus the no-skill agent for a mint. The resulting outputs are judged pairwise under order randomisation, so within each context the only difference under test is the skill condition. A prompt is won only if the candidate wins a majority of its contexts, preventing a large gain in one context from masking losses in others. A change ships only if

$$\left(\nexists \text{ prompt lost}\right) \wedge \left(\exists \text{ prompt won}\right). \tag{1}$$



(a) Skill-bank embedding UMAP

(b) Per-round skill changes

Figure 4: **Skill-bank evolution over five rounds.** (a) Skill embeddings (`BGE-small-en-v1.5`, UMAP, cosine); shape and colour indicate the target application, with hollow markers for the 76 cold-start skills distilled from internal documentation and filled markers for the 63 skills minted from user trajectories. (b) Per-round Added (Widening) and Rewritten (Deepening) skills, split into committed and gate-rejected. R1 is nearly all repair—the gap store has not yet accumulated enough recurring misses to mint from—while minting peaks in R2–R3 and tapers as coverage saturates.

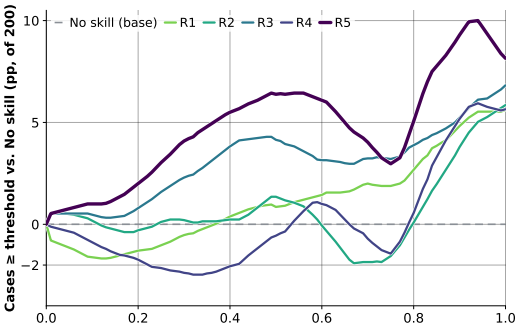
Because replay spans both successful and failed histories, a rewrite must repair failures without regressing existing successes, reflecting the asymmetric cost of regressions in production. Appendix F specifies context construction, the tie band, and per-axis replay budgets.

4 EXPERIMENTS

Agents Setup. We compare the EVOLVE agentic system against a no-skill condition (BASE), holding the underlying agent fixed. We use three foundation models: `claude-opus-4.6` (Anthropic, 2026) and `claude-sonnet-4` (Anthropic, 2025) via Amazon Bedrock, and `Qwen3.6-27B` (Qwen Team, 2026) via vLLM (Kwon et al., 2023) on 8×A100 GPUs (65K-token context; tool calling and native reasoning enabled). Reasoning settings are fixed across generation, evolution, and evaluation. The Claude models use low thinking effort, capped at 2,000 and 5,000 tokens for Opus and Sonnet, respectively. All backbones have a 900s wall-clock cap per prompt.

Internal Benchmark. Each round of evolution consumes $\approx$ 300 design briefs drawn from user traffic and LLM-augmented variants, replayed through the agent to produce the graded trajectories that drive widening and deepening. A design brief is a natural-language request describing the artifact to produce, its concrete requirements (text, colors, placements), and its stylistic goals, which the agent plans and executes into an editable design; a brief can be as short as a few words—like the examples in Figure 7—or as long as a full paragraph. Evaluation uses a separate held-out set of 200 human-authored briefs, disjoint from the evolution briefs and fixed across all five rounds: after each round we freeze the resulting bank and score it on this same set, so per-round skill rewrites, additions, and performance are all measured against a constant target. Evaluation methodology and metrics are detailed in Appendix D.

External Benchmarks and Metrics. We evaluate two categories of benchmarks, sampling 300 prompts from each. For general T2I capability we use **GenEval2** (Kamath et al., 2025) for compositional reasoning over objects and spatial relations, **DPG-Bench** (Hu et al., 2024) for dense prompt following, and **OneIG-EN/OneIG-ZH** (Chang et al., 2025) for cross-lingual subject-element alignment and text rendering; we report the Soft-TIFA (Kamath et al., 2025) geometric mean on GenEval2, the Soft-TIFA arithmetic mean on DPG-Bench, and VQAScore (Lin et al., 2024) on OneIG, judged by `Qwen3-VL-8B-Instruct` (Bai et al., 2025) over successful generations. For design capability we sample the released **OpenCOLE** evaluation data (Inoue et al., 2024), **GraphicBench** (Ki et al., 2025), **CreatiDesign** (Zhang et al., 2026a), and **BannerRequest400** (Wang et al., 2025), which cover multi-step planning, layout and text constraints, and visual quality; here we report pairwise EVOLVE-vs-BASE win rates judged by `GPT-5.4` (OpenAI, 2026) in a blind, two-order comparison to mitigate position bias.



(a) Survival gap $\Delta R(\tau)$ vs. no skill

$\geq \tau$	Base	R1	R2	R3	R4	R5
≥ 0.1	96	95	**98**	97	96	**98**
≥ 0.2	96	94	96	96	94	**98**
≥ 0.3	94	92	94	96	90	**98**
≥ 0.4	91	91	90	94	88	**96**
≥ 0.5	86	88	88	92	85	**93**
≥ 0.6	80	82	80	83	80	**86**
≥ 0.7	74	76	71	76	73	**78**
≥ 0.8	62	64	62	**66**	62	64
≥ 0.9	43	48	46	48	50	**56**
$= 1.0$	24	30	30	31	30	**32**

(b) Retention $R(\tau)$ (% of cases $\geq \tau$)

Figure 5: **Iterative skill bank evolution drives consistent completeness gains** (Claude-Sonnet-4, n=200). Retention $R(\tau)$ denotes the percentage of cases satisfying completeness $\geq \tau$. Subfigure (a) shows the gap to the no-skill baseline $\Delta R(\tau)$, while Table (b) details the underlying absolute values (row max in **bold**). Later rounds (especially R5) dominate across nearly all thresholds, with the largest improvement at τ=0.9 (+13 pp over Base).

4.1 Evolution Dynamics

Skill evolution alternates between widening and deepening without manual scheduling, driven by a coverage–reliability trade-off. First, **widening requires recurrence**: a missing skill is only minted after $k_{\min}$ repeated failures across traffic. Second, **unrefined widening introduces noise**: newly minted skills expand coverage but lack multi-trial verification, occasionally causing false-positive retrievals on neighboring briefs. Finally, **coverage saturation shifts focus back to deepening**: as minting slows, execution failures accumulate against newly added skills, triggering rewrites that convert broad coverage into stable performance.

We run the loop for five rounds on 1,406 non-overlapping briefs from user traffic and LLM-augmented variants, replaying them through the agent and grading every rollout with the evaluation kit of Appendix D. Across five rounds, this yields 1,869 graded trajectories without human labels. The bank grows from 76 documentation-derived skills at cold start to 139, with every change admitted through the replay gate (§3.3).

Figure 4 shows distinct dynamics for widening and deepening. R1 is dominated by repair: 39 of 59 rewrites and 4 of 10 mints pass the gate. Widening lags because gaps must recur across $k_{\min}$ requests before minting (§3.1), peaking in R2–R3 with $22/46$ and $26/40$ committed, growing the bank from 77 to 124. Across five rounds, the gate rejects $100/231$ rewrite proposals (committing 131) and $67/136$ mint candidates (committing 69); net growth (63) trails gross mints since deepening occasionally discards a superseded or merged skill (6 total). The minted skills are not redundant: their nearest-neighbour distance to the cold-start bank exceeds the seed bank's internal spacing (0.215 vs. 0.158 median; Mann–Whitney $p < 10^{-8}$, Cliff's $\delta = 0.58$), indicating widening covers intents the seed missed rather than paraphrasing it.

To track performance, we freeze each round's bank and evaluate `claude-sonnet-4` on 200 fixed, human-authored briefs disjoint from the 1,406 evolution briefs as our internal benchmark. Figure 5 shows gains at essentially every completeness threshold: relative to no skill, R5 raises the share of briefs at ≥ 0.5 from 86% to 93%, at ≥ 0.9 from 43% to 56%, and at $= 1.0$ from 24% to 32%, with the largest survival gap (+13 pp) at $\tau \geq 0.9$.

The trajectory is not monotonic. R4 falls below no skill at completeness ≥ 0.3 (90% vs. 94%) while retaining a +7 pp gain at ≥ 0.9: its high-quality tail remains strong while its lower end regresses. R3 and R4 mint 26 and 13 skills, respectively, leaving R4 with the largest stock of never-revised v1 skills. Because a minted skill is initially verified only against the tools-only baseline of its originating gap cluster, without large-scale revision against failures, it can misfire on requests outside that cluster. R5 reverses the mix, committing 21 rewrites and only 4 mints, and becomes the strongest round at every threshold, recovering the lower end while further improving the high-

Table 1: **Quantitative results on general T2I benchmarks.** Each cell reports the *generation quality* followed by the *success rate (%)*. Agents equipped with the EVOLVE skill bank improve generation quality and success on most benchmarks across backbones compared to the no-skill baseline.

Agent	Method	GenEval2 ↑	DPG-Bench ↑	OneIG-EN ↑	OneIG-ZH ↑	Avg ↑
Claude-Opus-4.6	BASE	**61.24** (96.7)	77.58 **(89.3)**	60.66 (91.3)	68.67 (93.3)	67.04 (92.7)
	EVOLVE	57.92 **(98.0)**	**86.54** (84.7)	**68.79 (94.0)**	**70.21 (94.0)**	**70.87 (92.7)**
	Δ	-3.32 (+1.3)	+8.96 (-4.6)	+8.13 (+2.7)	+1.54 (+0.7)	+3.83 (0.0)
Claude-Sonnet-4	BASE	34.26 (72.7)	68.28 (82.7)	52.00 **(98.0)**	52.00 (96.7)	51.63 (87.5)
	EVOLVE	**46.25 (99.3)**	**83.79 (100)**	**53.47** (96.0)	**53.79** (96.7)	**59.33 (98.0)**
	Δ	+11.99 (+26.6)	+15.51 (+17.3)	+1.47 (-2.0)	+1.79 (0.0)	+7.70 (+10.5)
Qwen3.6-27B	BASE	49.29 (54.7)	**90.11 (44.0)**	61.76 (45.3)	73.53 (45.3)	68.67 (47.3)
	EVOLVE	**64.52 (59.3)**	86.91 (28.7)	**80.88 (51.3)**	**81.03 (58.7)**	**78.34 (49.5)**
	Δ	+15.23 (+4.6)	-3.20 (-15.3)	+19.12 (+6.0)	+7.50 (+13.4)	+9.67 (+2.2)

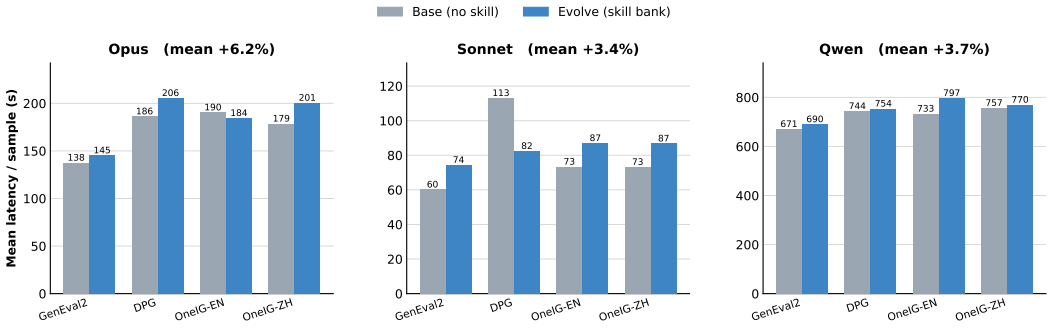


Figure 6: **Generation latency comparison.** Mean wall-clock time per successful generation. The EVOLVE framework introduces minimal computational overhead across most backbones.

quality tail. Widening and deepening are therefore complementary: minting expands coverage, while rewriting converts that coverage into reliability (Section 5).

4.2 MAIN RESULTS

In this section, we comprehensively evaluate our framework across both general Text-to-Image (T2I) generation and specialized graphic design tasks.

Performance on General T2I Tasks. Table 1 presents the quantitative comparison between the BASE agent and our EVOLVE agent across four general T2I benchmarks. Overall, equipping agents with the evolved skill bank yields substantial improvements. We highlight three primary takeaways:

- **Generation quality improves on average across all backbones.** EVOLVE raises average quality by $+3.83$, $+7.70$, and $+9.67$ for `Claude-Opus-4.6`, `Claude-Sonnet-4`, and `Qwen3.6-27B`, respectively, although individual benchmarks can regress. Qwen's high absolute quality scores should be interpreted alongside its lower success rate, since quality is evaluated on successful outputs and disproportionately reflects easier prompts.
- **EVOLVE improves execution reliability on most benchmarks.** The effect is strongest for `Claude-Sonnet-4`, whose success rate rises from 72.7% to 99.3% on GenEval2 and from 82.7% to 100% on DPG-Bench.
- **Latency overhead remains modest.** EVOLVE adds only 3.4%–6.2% mean latency overhead across backbones (Figure 6), and can occasionally reduce generation time (e.g., $113 \rightarrow 82$ s for Sonnet on DPG-Bench).

Table 2: **Pairwise win rates on specialized graphic design tasks.** EVOLVE-vs-BASE win rates judged by `GPT-5.4`. Values in green show the margin over a 50% tie baseline.

Agent	OpenCOLE	GraphicBench	CreatiDesign	BannerRequest400	Overall
Claude-Opus-4.6	64.0% +14.0%	63.5% +13.5%	71.7% +21.7%	71.3% +21.3%	67.6% +17.6%
Claude-Sonnet-4	66.0% +16.0%	56.8% +6.8%	68.1% +18.1%	56.2% +6.2%	61.8% +11.8%
Qwen3.6-27B	62.2% +12.2%	49.0% -1.0%	70.9% +20.9%	69.2% +19.2%	62.8% +12.8%

Performance on Specialized Graphic Design Tasks. As presented in Table 2, the EVOLVE agent outperforms the BASE agent overall. For `Claude-Opus-4.6`, the evolution secures a commanding 67.6% overall win rate, peaking at 71.7% on CreatiDesign. Similarly, `Claude-Sonnet-4` achieves 61.8% overall win rate. These margins indicate that the evolution is particularly effective in resolving complex, multi-step design constraints that standard zero-shot generation struggles to handle. Detailed success rates for design benchmarks are provided in Appendix H.

4.3 QUALITATIVE RESULTS

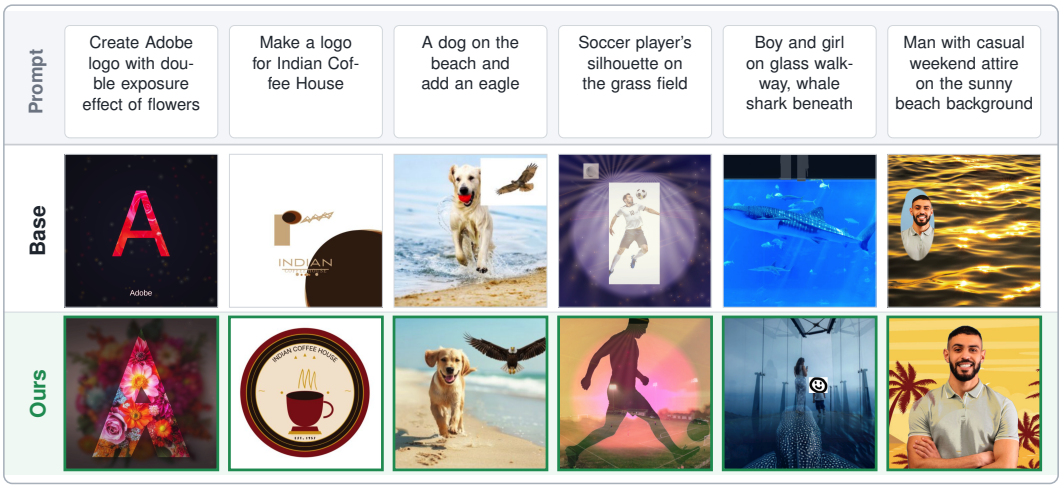


Figure 7: **Qualitative comparison on six design briefs.** Each column is a brief; rows are BASE (no skill) and OURS (EVOLVE). Asset search runs once per brief and both arms are shown the same candidate pool, so retrieval is held fixed and the rows differ only in which assets the agent selects and how it edits them. All outputs are uncurated one-shot `claude-opus-4.6` rollouts at low thinking effort, with no re-sampling and no human intervention, center-cropped to square for display.

Editing. EVOLVE carries multi-step editing procedures through, whereas BASE often places the relevant assets but stops short of the required edit. Consider double exposure: extract the subject, mask it, and blend a second image through it. In Column 1 (Adobe logo with double exposure of flowers), BASE attempts the blend, but the flowers remain faint and muddied; with EVOLVE, they read clearly through the glyph. In Column 4 (soccer player's silhouette on the pitch), BASE never extracts the figure, whereas EVOLVE extracts the silhouette and blends the pitch through it. These failures differ—one attempts the procedure unsuccessfully, while the other never starts—suggesting a missing procedure rather than a missing capability. EVOLVE applies the same workflow in both cases, transferring one procedure from a letterform to a human figure. Columns 2–3 show related failures at different scales: BASE leaves the coffee-house logo unresolved and the eagle visibly unmasked, whereas EVOLVE completes the corresponding composition and masking steps.

Asset selection. EVOLVE also selects assets with the downstream edit in mind, whereas BASE tends to match only the surface nouns of the brief. In Column 5 (boy and girl on a glass walkway, whale shark beneath), BASE selects an underwater shark image with no walkway or suitable vantage point, making the requested spatial relation impossible to stage. EVOLVE instead selects a glass tunnel with suitable figures and places the whale shark beneath them. Column 6 shows the same pattern:

BASE selects an unsuitable water background, whereas EVOLVE selects a beach scene and extracts the foreground figure. Together, these examples show how procedural guidance can influence not only execution but also asset choices needed for downstream editing.

Appendix J shows uncurated one-shot EVOLVE outputs on the full internal benchmark.

5 ABLATION STUDY

We isolate Deepening (rewriting existing skills) and Widening (minting new skills) using two intermediate banks: + *rewrite* (latest cold-start skills) and + *new skills* (all V1 skills) (Table 3). All arms use the same agent and 200 unseen prompts in one batch, evaluated by GPT-5.4 and our Internal Evaluation Kit (Appendix D); token counts include cached contexts.

The documentation-derived cold start does not improve over BASE (68.62 vs. 69.08 completeness; 46.4% win rate). Neither mechanism alone suffices: rewriting reaches 69.02 completeness and 48.6% win rate, while expansion reaches 69.79 and 49.4%, respectively. Combined, they reach 74.04 completeness (+5.42 over cold start) and a 58.5% win rate against BASE ($p = 0.025$). The superadditive gain (+3.85 completeness) reflects loop coupling: minted skills require refined retrieval descriptions to surface, while rewriting reroutes unfixable failures to the gap store for minting.

Gains concentrate in completeness; aesthetics (65.92 $\rightarrow$ 66.53) and critique remain largely unchanged. Although skill retrieval adds $\sim 28\%$ prompt tokens over BASE, evolution adds no marginal token cost: with top-k ($k = 3$) matching, EVOLVE uses fewer prompt/output tokens than cold start (436.6k/5167 vs. 445.2k/5298), consistent with more direct execution and fewer corrective retries.

Table 3: **Ablation of skill-bank update mechanisms.** All arms share identical prompts and evaluation settings. Neither rewriting nor adding new skills alone separates from cold start; applied together, they produce a superadditive interaction (+3.85 completeness). Best per column in **bold**.

		Quality ↑		Critique ↓		Tokens ↓				
Bank	#Sk.	Compl.	Aesth.	Major	Minor	Prompt	Out	Tools	WR	SR
BASE	0	69.08	65.92	3.93	8.02	**346.8k**	**4979**	33.5	—	95.0
Cold start	76	68.62	65.66	3.88	**7.87**	445.2k	5298	35.4	46.4	96.5
+ rewrite	76	69.02	65.98	3.70	8.14	450.1k	5369	35.9	48.6	97.5
+ new skills	139	69.79	64.77	3.70	7.93	445.4k	5342	36.4	49.4	**98.5**
EVOLVE	139	**74.04**	**66.53**	**3.57**	8.13	436.6k	5167	36.7	**58.5**	98.0

6 LIMITATION & CONCLUSION

We studied the evolution of procedural memory, an external library of natural-language skills, for a professional graphic design agent. Widening adds procedures for recurring uncovered subtasks, while deepening revises existing procedures using successful and failed executions, and candidate changes pass through matched replay before entering the deployed bank. Across five rounds over 1,406 briefs, the bank grows from 76 skills to 139, improving three backbones with no weight updates and no human reward labels. Neither axis suffices alone; together they expand coverage and make existing procedures more reliable, yielding clear gains on held-out briefs.

The results also identify limits of this approach. A natural-language skill can describe a preferred procedure, but retrieval alone may not make the model follow it when the model has a strong default strategy. Fine geometric operations remain limited by both model perception and automated verification, and long procedures lose fidelity as instructions accumulate over many execution steps. The replay gate is also local to the evaluated cases: rejecting observed regressions on the replay set does not guarantee monotonic improvement over the full user-traffic distribution. Addressing these limits will require stronger execution mechanisms, including preference-aware retrieval, deterministic primitives, structured plans, and more precise verification. Procedural memory is therefore one mechanism for continual agent adaptation, not a replacement for model learning or structured execution when a task requires capabilities that natural-language guidance cannot reliably induce.

AI DISCLOSURE

In this work, we used generative AI tools to generate synthetic prompt variants; clean, reformat, and augment prompt data; implement the proposed methods and experimental pipeline; design experiments; interpret results; and support automated evaluation and qualitative analysis of generated design artifacts. In particular, LLM APIs are a component of the agentic system studied in this work: they are used to operate software tools that create and edit the reported graphic-design artifacts. As described in the paper, we also use LLMs to clean and augment prompt data, including the generation of augmented prompt variants.

We have not used generative AI tools to develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, or assist with translation; the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to create or edit software code, create visualizations, create research artifacts through the agentic system, and edit the paper to improve readability.

We have reviewed all AI-assisted work. In particular, all AI-assisted code was reviewed and verified by the authors; all AI-assisted text was reviewed, revised, and finalized by the authors; and all AI-assisted data, visualizations, generated artifacts, experimental outputs, and reported claims were human-verified before inclusion in this submission. We take responsibility for the final content of this work, including text, claims, code, data, and artifacts produced with the aid of generative AI.

REFERENCES

Adobe Inc. Adobe stock, 2026. URL https://stock.adobe.com/. Accessed: 2026-08-31.

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Qihan Ren, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=CTr3bovS5F.

Anthropic. Claude 4 system card, May 2025. URL https://www-cdn.anthropic.com/4263b940cabb546aa0e3283f35b686f4f3b2ff47/claude-opus-4-and-claude-sonnet-4-system-card.pdf.

Anthropic. Claude opus 4.6 system card, February 2026. URL https://www-cdn.anthropic.com/14e4fb01875d2a69f646fa5e574dea2b1c0ff7b5.pdf.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 335–336. Association for Computing Machinery, 1998. doi: 10.1145/290941.291025. URL https://doi.org/10.1145/290941.291025.

Jingjing Chang, Yixiao Fang, Peng Xing, Shuhan Wu, Wei Cheng, Rui Wang, Xianfang Zeng, Gang Yu, and Hai-Bao Chen. Oneig-bench: Omni-dimensional nuanced evalua-

tion for image generation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 177093–177128. Curran Associates, Inc., 2025. doi: 10.52202/085713-5330. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/e9e9e5428189a3e49479547ef917e88d-Abstract-Datasets_and_Benchmarks_Track.html.

Jingye Chen, Zhaowen Wang, Nanxuan Zhao, Li Zhang, Difan Liu, Jimei Yang, and Qifeng Chen. Rethinking layered graphic design generation with a top-down approach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16861–16870, October 2025. doi: 10.1109/ICCV51701.2025.01566. URL https://openaccess.thecvf.com/content/ICCV2025/html/Chen_Rethinking_Layered_Graphic_Design_Generation_with_a_Top-Down_Approach_ICCV_2025_paper.html.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=cSimKw5p6R.

Saman Forouzandeh, Wei Peng, Parham Moradi, Xinghuo Yu, and Mahdi Jalili. Learning hierarchical procedural memory for llm agents through bayesian selection and contrastive refinement. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1820–1828. International Foundation for Autonomous Agents and Multiagent Systems, 2026. doi: 10.65109/FKYO8341. URL https://www.ifaamas.org/Proceedings/aamas2026/pdfs/FKYO8341.pdf.

Haowen Gao, Haoran Chen, Can Wang, Shasha Guo, Liang Pang, Zhaoyang Liu, Huawei Shen, and Xueqi Cheng. Skillaudit: Ground-truth-free skill evolution via paired trajectory auditing, 2026. URL https://arxiv.org/abs/2606.14239.

Yicheng He, Chengsong Huang, Zongxia Li, Jiaxin Huang, and Yonghui Yang. Visplay: Self-evolving vision-language models from images, 2025. URL https://arxiv.org/abs/2511.15661.

Dexiang Hong, Zhao Zhang, Weidong Chen, Yutao Cheng, Maoke Yang, Gonglei Shi, Hui Zhang, and Zhendong Mao. Creatiposter: Towards editable and controllable multi-layer graphic design generation, 2026. URL https://arxiv.org/abs/2506.10890.

Hsiao-Yuan Hsu, Xiangteng He, Yuxin Peng, Hao Kong, and Qing Zhang. PosterLayout: A new benchmark and approach for content-aware visual-textual presentation layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6018–6026, June 2023. doi: 10.1109/CVPR52729.2023.00583. URL https://openaccess.thecvf.com/content/CVPR2023/html/Hsu_PosterLayout_A_New_Benchmark_and_Approach_for_Content-Aware_Visual-Textual_Presentation_CVPR_2023_paper.html.

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment, 2024. URL https://arxiv.org/abs/2403.05135.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A. Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20349–20360, 2023. doi: 10.1109/ICCV51070.2023.01866. URL https://doi.org/10.1109/ICCV51070.2023.01866.

Chengsong Huang, Haolin Liu, Tong Zheng, Runpeng Dai, Langlin Huang, Jinyuan Li, Zongxia Li, Zhepei Wei, Yu Meng, and Jiaxin Huang. G-zero: Self-play for open-ended generation from zero data, 2026a. URL https://arxiv.org/abs/2605.09959.

Chengsong Huang, Zifeng Wang, Rujun Han, Jun Yan, Yanfei Chen, Zoey CuiZhu, Ke Jiang, Peng Xia, Han Yu, Yufan Zhuang, Yifei Ming, Jiaqi Pan, Bhavana Dalvi Mishra, Jiaxin Huang, Burak Gokturk, Tomas Pfister, and Chen-Yu Lee. Envharness: Awakening static worlds for agent learning, 2026b. URL https://arxiv.org/abs/2608.19880.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-evolving reasoning llm from zero data. In *International Conference on Learning Representations*, volume 2026, pp. 130770–130790, 2026c. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/d49b9aacebda61051166335af6fd3061-Abstract-Conference.html.

Naoto Inoue, Kento Masui, Wataru Shimoda, and Kota Yamaguchi. Opencole: Towards reproducible automatic graphic design generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 8131–8135, June 2024. URL https://openaccess.thecvf.com/content/CVPR2024W/GDUG/html/Inoue_OpenCOLE_Towards_Reproducible_Automatic_Graphic_Design_Generation_CVPRW_2024_paper.html.

Peidong Jia, Chenxuan Li, Yuhui Yuan, Zeyu Liu, Yichao Shen, Bohan Chen, Xingru Chen, Yinglin Zheng, Dong Chen, Ji Li, Xiaodong Xie, Shanghang Zhang, and Baining Guo. COLE: A hierarchical generation framework for multi-layered and editable graphic design, 2023. URL https://arxiv.org/abs/2311.16974.

Amita Kamath, Kai-Wei Chang, Ranjay Krishna, Luke Zettlemoyer, Yushi Hu, and Marjan Ghazvininejad. Geneval 2: Addressing benchmark drift in text-to-image evaluation, 2025. URL https://arxiv.org/abs/2512.16853.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations*, volume 2024, pp. 54928–54958, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/f1cf02ce09757f57c3b93c0db83181e0-Abstract-Conference.html.

Dayeon Ki, Tianyi Zhou, Marine Carpuat, Gang Wu, Puneet Mathur, and Viswanathan Swaminathan. Graphicbench: A planning benchmark for graphic design with language agents, 2025. URL https://arxiv.org/abs/2504.11571.

Donald E. Knuth and Michael F. Plass. Breaking paragraphs into lines. *Software: Practice and Experience*, 11(11):1119–1184, 1981. doi: 10.1002/spe.4380111102. URL https://doi.org/10.1002/spe.4380111102.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626. Association for Computing Machinery, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. Safe policy improvement with baseline bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3652–3661. PMLR, 2019. URL https://proceedings.mlr.press/v97/laroche19a.html.

Zongxia Li, Hongyang Du, Chengsong Huang, Xiyang Wu, Lantao Yu, Yicheng He, Jing Xie, Xiaomin Wu, Zhichao Liu, Jiarui Zhang, and Fuxiao Liu. Mm-zero: Self-evolving multi-model vision language models from zero data, 2026a. URL https://arxiv.org/abs/2603.09206.

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, Lichao Sun, Haotao Mi, and Leowei Liang. Long-horizon-terminal-bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading, 2026b. URL https://arxiv.org/abs/2607.08964.

Zongxia Li, Dawei Liu, Fuxiao Liu, Yuhang Zhou, Xiyang Wu, Jingxi Chen, Jing Xie, Xiaomin Wu, and Lichao Sun. Comfyclaw: Self-evolving skill harnesses for image generation workflows, 2026c. URL https://arxiv.org/abs/2607.01709.

Jiawei Lin, Shizhao Sun, Danqing Huang, Ting Liu, Ji Li, and Jiang Bian. From elements to design: A layered approach for automatic graphic design composition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8128–8137, June 2025. doi: 10.1109/CVPR52734.2025.00761. URL https://openaccess.thecvf.com/content/CVPR2025/html/Lin_From_Elements_to_Design_A_Layered_Approach_for_Automatic_Graphic_CVPR_2025_paper.html.

Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *Computer Vision – ECCV 2024*, volume 15067 of *Lecture Notes in Computer Science*, pp. 366–384. Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-72673-6_20. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/html/1435_ECCV_2024_paper.php.

Dawei Liu, Zongxia Li, Hongyang Du, Xiyang Wu, Shihang Gui, Yongbei Kuang, and Lichao Sun. Graph-of-skills: Dependency-aware structural retrieval for massive agent skills, 2026. URL https://arxiv.org/abs/2604.05333.

Vlad-Constantin Lungu-Stan, Ionuț Mironică, and Mariana-Iuliana Georgescu. LaDe: Unified multi-layered graphic media generation and decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 6031–6040, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026W/CVEU/html/Lungu-Stan_LaDe_Unified_Multi-Layered_Graphic_Media_Generation_and_Decomposition_CVPRW_2026_paper.html.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents, 2026. URL https://arxiv.org/abs/2602.01869.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Kadous, and Ion Stoica. RouteLLM: Learning to route llms from preference data. In *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5503a7c69d48a2f86fc00b3dc09de686-Abstract-Conference.html.

OpenAI. GPT-5.4 Thinking System Card, March 2026. URL https://deploymentsafety.openai.com/gpt-5-4-thinking. Accessed: 2026-09-04.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22. Association for Computing Machinery, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*, volume 37, pp. 126544–126565. Curran Associates, Inc., 2024. doi: 10.52202/079017-4020. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e4c61f578ff07830f5c37378dd3ecb0d-Abstract-Conference.html.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. Hiper: Hierarchical reinforcement learning with explicit credit assignment for large language model agents, 2026. URL https://arxiv.org/abs/2602.16165.

Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R. Zaiane, and Martin Jagersand. U^2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020. doi: 10.1016/j.patcog.2020.107404. URL https://doi.org/10.1016/j.patcog.2020.107404.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/28e50ee5b72e90b50e7196fde8ea260e-Abstract-Conference.html.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026. URL https://qwen.ai/blog?id=qwen3.6-27b.

Jaejung Seol, Seojun Kim, and Jaejun Yoo. PosterLlama: Bridging design ability of language model to content-aware layout generation. In *Computer Vision – ECCV 2024*, volume 15140 of *Lecture Notes in Computer Science*, pp. 451–468. Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-73007-8_26. URL https://doi.org/10.1007/978-3-031-73007-8_26.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, Rabat, Morocco, March 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-long.333. URL https://aclanthology.org/2026.eacl-long.333/.

Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-confidence off-policy evaluation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1):3000–3006, 2015. doi: 10.1609/aaai.v29i1.9541. URL https://ojs.aaai.org/index.php/AAAI/article/view/9541.

Unicode Consortium. Unicode standard annex #9: Unicode bidirectional algorithm, August 2025. URL https://www.unicode.org/reports/tr9/tr9-51.html. Unicode 17.0.0, Revision 51.

Chonghuinan Wang, Zihan Chen, Yuxiang Wei, Tianyi Jiang, Xiaohe Wu, Fan Li, Wangmeng Zuo, and Hongxun Yao. CREval: An automated interpretable evaluation for creative image manipulation under complex instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9029–9039, June 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/html/Wang_CREval_An_Automated_Interpretable_Evaluation_for_Creative_Image_Manipulation_under_CVPR_2026_paper.html.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. URL https://openreview.net/forum?id=ehfRiF0R3a.

Heng Wang, Yotaro Shimose, and Shingo Takamatsu. BannerAgency: Advertising banner design with multimodal LLM agents. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods*

in Natural Language Processing, pp. 4304–4329, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.214. URL https://aclanthology.org/2025.emnlp-main.214/.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.acl-long.511. URL https://aclanthology.org/2024.acl-long.511/.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-guided policy learning for long-horizon language agents, 2026b. URL https://arxiv.org/abs/2605.06078.

World Wide Web Consortium. Web content accessibility guidelines (wcag) 2.2, December 2024. URL https://www.w3.org/TR/2024/REC-WCAG22-20241212/.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17577–17604. Curran Associates, Inc., 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Kota Yamaguchi. CanvasVAE: Learning to generate vector graphic documents. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5481–5489, October 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Yamaguchi_CanvasVAE_Learning_To_Generate_Vector_Graphic_Documents_ICCV_2021_paper.html.

Chenchen Zhang. From reasoning to agentic: Credit assignment in reinforcement learning for large language models, 2026. URL https://arxiv.org/abs/2604.09459.

Hui Zhang, Dexiang Hong, Maoke Yang, Yutao Cheng, Zhao Zhang, Weidong Chen, Jie Shao, Xinglong Wu, Zuxuan Wu, and Yu-Gang Jiang. Creatidesign: A unified multi-conditional diffusion transformer for creative graphic design. In *International Conference on Learning Representations*, volume 2026, pp. 111201–111214, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/b46b78c353f672d83997d4cce5f1b0fb-Abstract-Conference.html.

Qizheng Zhang, Michael Wornow, and Kunle Olukotun. Agentic plan caching: Test-time memory for fast and cost-efficient llm agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 103270–103296. Curran Associates, Inc., 2025. doi: 10.52202/085713-3451. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/9549f7d06700f0966d5f938f1d11022a-Abstract-Conference.html.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y. Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8a94ff6f922d995d7d3f4ebf4143e442-Abstract-Conference.html.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38 (17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

A Related Work

A.1 Self-Improving Agents and Experience-Based Adaptation

A growing body of work studies how language agents can improve from interaction experience without relying exclusively on parameter updates. Reflexion (Shinn et al., 2023) improves an agent through verbal feedback, storing self-generated reflections in an episodic memory buffer and reusing them across subsequent trials. Self-Refine (Madaan et al., 2023) similarly uses an LLM to generate feedback on its own outputs and iteratively revise them without additional supervised training or reinforcement learning. ExpeL (Zhao et al., 2024) extends experience-based adaptation across tasks by collecting agent trajectories, extracting transferable natural-language insights from them, and retrieving both insights and prior experiences at inference time. Generative Agents (Park et al., 2023) likewise demonstrated that persistent records of experience can be synthesized into higher-level reflections and dynamically retrieved to influence future planning and behavior. These methods differ in persistence scope: Self-Refine focuses on within-task refinement, whereas Reflexion, ExpeL, and Generative Agents retain experience across trials or tasks. None studies a shared, regression-gated skill bank evolved from multi-user deployment traffic.

More recent work has moved from within-task refinement toward persistent adaptation across tasks. Dynamic Cheatsheet (Suzgun et al., 2026) maintains an evolving test-time memory containing reusable strategies, code snippets, and problem-solving insights, allowing black-box language models to accumulate knowledge across otherwise independent queries without weight updates or explicit human supervision. ACE (Zhang et al., 2026b) treats an agent's context itself as an evolving playbook and uses generator, reflector, and curator roles to incrementally accumulate and refine strategies from execution feedback. More broadly, recent surveys characterize self-evolving agents in terms of what agent components evolve, when adaptation occurs, and what feedback mechanisms drive the evolution (ang Gao et al., 2026).

A related question is which component of an evolving system should be revised after a failure. SkillAudit (Gao et al., 2026) studies ground-truth-free evolution of structured agent skills by comparing paired trajectories with and without a candidate skill and using their behavioral differences to localize passages for refinement or repair. ACE (Zhang et al., 2026b) instead uses generator, reflector, and curator roles to evolve a contextual playbook. Both motivate learning from behavioral feedback, but neither uses our failure-count heuristic for prioritizing deployed skills.

A.2 Agent Memory, Procedural Skills, and Reusable Tool-Use Knowledge

External memory provides a natural mechanism for agents to retain experience without modifying the underlying language model. Early memory-augmented agents primarily represented experience episodically: Generative Agents (Park et al., 2023) store natural-language records of observations and retrieve relevant memories for planning, while Reflexion (Shinn et al., 2023) stores verbal reflections produced after previous trials. A-MEM (Xu et al., 2025) moves toward an adaptive memory substrate in which newly added memories are dynamically indexed and linked to existing memories and can trigger updates to their contextual representations. Dynamic Cheatsheet (Suzgun et al., 2026) similarly maintains persistent, self-curated memory but emphasizes compact and transferable problem-solving strategies rather than complete interaction histories.

A complementary line of work represents reusable experience as procedural knowledge. Voyager (Wang et al., 2024a) introduced an ever-growing library of executable skills in Minecraft, storing successful action programs that can later be retrieved and composed to solve new tasks without model fine-tuning. Agentic Plan Caching (Zhang et al., 2025) extracts structured plan templates from completed agent executions and adapts them for semantically similar future requests, demonstrating that reusable procedural structure can reduce agent inference cost and latency while maintaining task performance. MACLA (Forouzandeh et al., 2026) explicitly formulates external hierarchical procedural memory for frozen LLM agents, extracting reusable procedures from trajectories and refining them contrastively using successful and failed experiences. Skill-Pro (Mi et al., 2026) similarly converts interaction experience into executable skills with activation, execution, and termination conditions, and introduces a non-parametric verification mechanism to control which skills enter procedural memory.

These systems suggest an important distinction between remembering an episode and retaining a reusable procedure: episodic memory preserves information about what happened, whereas procedural memory captures how a class of tasks can be accomplished (Xu et al., 2025; Wang et al., 2024a; Forouzandeh et al., 2026; Mi et al., 2026). The distinction is particularly important for tool-using agents, where repeated success often depends not only on retrieving relevant facts but also on reproducing a reliable sequence of actions (Wang et al., 2024a; Zhang et al., 2025; Forouzandeh et al., 2026; Mi et al., 2026). Some prominent skill-learning settings, such as Voyager's Minecraft tasks, expose programmatic success signals; professional graphic design generally does not.

A.3 Continual Skill Acquisition and Refinement

Lifelong agents must not only reuse existing knowledge but also expand their behavioral repertoire as new tasks are encountered. Voyager (Wang et al., 2024a) addresses this problem through an automatic curriculum coupled with an ever-growing skill library, allowing the agent to continually discover tasks and commit newly mastered executable behaviors for later reuse. Dynamic Cheatsheet (Suzgun et al., 2026) continually adds transferable insights to persistent memory as additional problems are solved, while ACE (Zhang et al., 2026b) explicitly adopts a grow-and-refine strategy in which new knowledge is accumulated and existing contextual knowledge is incrementally updated rather than repeatedly rewriting the full context. Recent self-evolving-agent work similarly emphasizes continual adaptation from interaction data and feedback as a mechanism for moving beyond static agents (ang Gao et al., 2026).

Recent procedural-memory systems increasingly consider refinement in addition to acquisition. MACLA (Forouzandeh et al., 2026) extracts procedures from trajectories, tracks their reliability, and contrastively refines procedural knowledge using differences between successful and failed experiences. Skill-Pro (Mi et al., 2026) accumulates and refines executable procedural skills while using verification and score-based maintenance to control memory quality. SkillAudit (Gao et al., 2026) directly targets deployed skill evolution, distinguishing refinement of broadly useful but noisy guidance from repair of passages that conflict with observed task behavior.

Skill evolution has also been studied in visual generation settings. ComfyClaw (Li et al., 2026c) evolves a progressively disclosed skill library for workflow-based image generation, formulating workflow construction as typed graph editing, automatically reverting invalid edits, and using a region-level vision-language verifier to translate visual failures into actionable repair suggestions; trajectories, execution errors, and verifier feedback from histories are distilled into reusable skills.

Collectively, these works frame continual procedural learning as two complementary processes: acquiring knowledge for capabilities that are not yet represented, and consolidating existing knowledge once experience reveals systematic failure modes (Zhang et al., 2026b; Gao et al., 2026; Wang et al., 2024a; Forouzandeh et al., 2026; Mi et al., 2026).

A.4 Reliable Improvement under Automated Feedback

Automated judges make it possible to evaluate open-ended agent behavior at scale, but their outputs are themselves noisy and biased. Zheng et al. (2023) showed that strong LLM judges can approximate human preferences on open-ended evaluation while also documenting systematic limitations including position, verbosity, and self-enhancement biases. Wang et al. (2024b) independently demonstrated substantial position bias in LLM-based pairwise evaluation and showed that aggregating judgments across balanced presentation orders can mitigate this effect. For visual generation, TIFA (Hu et al., 2023) evaluates prompt faithfulness by decomposing text prompts into question-answer pairs and checking them against generated images, providing a fine-grained alternative to global embedding similarity. VQAScore (Lin et al., 2024) similarly evaluates text-to-visual alignment through visual question answering and reports stronger performance than CLIP-based similarity on complex compositional prompts. Closer to the design domain, QA-decomposition frameworks for creative image manipulation (Wang et al., 2026a) break evaluation of complex editing instructions into structured questions.

Dependence on a proxy judge is itself a documented failure mode rather than a benign implementation detail. G-Zero (Huang et al., 2026a) argues that self-evolving systems succeed in verifiable domains but degrade in open-ended ones precisely because the proxy judge imposes a capability

ceiling and invites reward hacking, and responds by discarding the verifier altogether in favor of an intrinsic, co-evolutionary reward signal.

When automated feedback is used not merely for reporting performance but for modifying a deployed system, evaluation noise becomes an update-safety problem. The broader safe-policy-improvement literature formalizes the goal of improving a policy relative to a deployed baseline while avoiding updates whose performance cannot be established with sufficient confidence (Laroche et al., 2019). High-confidence off-policy evaluation similarly studies how candidate policies can be assessed with confidence bounds before costly or unsafe deployment (Thomas et al., 2015). Although these methods address reinforcement-learning policies rather than natural-language skill artifacts, they motivate a conservative principle relevant to deployed agents: a proposed update should be compared against the incumbent rather than accepted solely because its absolute evaluation score appears high (Laroche et al., 2019; Thomas et al., 2015).

A.5 Skill Retrieval, Tool-Surface Reduction, and Cost-Aware Agent Design

A skill library is only useful if the right procedure reaches the model's context at the right time, and if doing so does not itself become the dominant cost. LLM cascades and routing methods (Chen et al., 2024; Ong et al., 2025) reduce serving cost by choosing among models of different capability and price, while tool-use training (Patil et al., 2024; Qin et al., 2024) teaches models to use large tool APIs through training. Prompt-program systems such as DSPy (Khattab et al., 2024) take a third route, optimizing an LM program or pipeline against a downstream metric, often without updating the base-model weights.

Retrieval becomes a bottleneck of its own as a library grows. Graph of Skills (Liu et al., 2026) observes that loading a full skill set saturates the context window, driving up token cost, hallucination, and latency, while purely semantic retrieval surfaces topically relevant skills but misses their prerequisite chain, leaving the retrieved bundle execution-incomplete. It addresses both by constructing an executable skill graph offline and retrieving a bounded, dependency-aware bundle at inference time through hybrid semantic-lexical seeding and context-budgeted hydration.

A.6 Co-Evolving Curricula and Environments

A parallel line of work improves models rather than memories, by co-evolving the task distribution together with the learner. Self-evolving reasoning via challenger–solver co-evolution (Huang et al., 2026c) removes the dependence on curated task sets by training a challenger to generate problems at the frontier of a solver's competence, and this idea has been extended to a proposer–coder–solver triad for multimodal settings (Li et al., 2026a). VisPlay (He et al., 2025) carries the recipe into vision-language models, splitting a single base model into an image-conditioned questioner that poses challenging but answerable questions and a multimodal reasoner that answers them, and training both jointly from unlabeled images with no human annotation. In all of these systems, the adaptation is stored in the model weights.

A further line co-evolves the environment rather than the policy. EnvHarness (Huang et al., 2026b) wraps a static environment in a programmable plug-in layer that reshapes its behavior through standard interfaces without modifying the underlying logic or its original verifier, with components synthesized from weaknesses diagnosed in a target policy's own execution traces and validated by fresh rollouts, enabling continued co-evolution of a policy and the environment it trains against.

A.7 LLM Agents for Graphic Design

Recent work has begun to extend multimodal agents from well-specified tool-use tasks to graphic design, where agents must reason jointly about content, layout, and visual appearance. OpenCOLE (Inoue et al., 2024) studies reproducible automatic graphic design generation using an open implementation and publicly available training data. GraphicBench (Ki et al., 2025) introduces a planning benchmark of $1{,}079$ creative-design requests together with GraphicTown, an agent environment in which multiple design experts plan and execute workflows using a vocabulary of 46 design actions. BannerAgency (Wang et al., 2025) further uses specialized multimodal agents to coordinate advertising-banner creation and produces editable Figma or SVG components rather than only flattened raster outputs. Layout- and design-generation systems PosterLlama (Seol et al., 2024)

target a narrower slice of the problem, content-aware layout, rather than a full tool-driven editing workflow. Collectively, these systems establish graphic design as an emerging setting for language-agent planning and structured visual creation (Inoue et al., 2024; Ki et al., 2025; Wang et al., 2025).

Execution in this setting is difficult even under a compact action space. GraphicBench reports that execution can fail with a compact vocabulary of 46 abstract design actions because of spatial reasoning, cross-step dependencies, and incorrect action selection (Ki et al., 2025), and professional creative workflows compose far lower-level operations—asset retrieval, selection, masking, compositing, typography, vector manipulation, layout, and verification—into a coherent artifact over a long trajectory.

Long trajectories also make direct trajectory-level learning difficult. Reinforcement learning for language agents typically relies on outcome-level rewards, but as trajectories become longer, a terminal reward provides increasingly weak information about which intermediate decisions caused success or failure (Zhang, 2026; Wang et al., 2026b). Recent work on long-horizon agentic RL identifies credit misattribution and sample inefficiency as central obstacles: an otherwise useful sequence of early actions may receive negative credit because of a much later failure, while sparse successful trajectories provide little learning signal for optimization (Zhang, 2026; Wang et al., 2026b). Long-horizon competence has correspondingly become an evaluation target in its own right. Long-Horizon-Terminal-Bench (Li et al., 2026b) stresses agents on extended terminal tasks and grades them with dense, reward-based signals rather than a single terminal outcome, on the grounds that binary end-state scoring cannot separate substantial partial progress from outright failure.

More fundamentally, graphic design lacks the reliable, verifiable reward available in many domains where agent learning has been most successful. Coding tasks can often be checked with tests, games and embodied environments expose task-relevant state, and other reasoning tasks may admit deterministic final-answer verification. A creative brief, in contrast, routinely combines objectively testable requirements with inherently underspecified judgments about hierarchy, balance, composition, style, and visual quality. BannerAgency (Wang et al., 2025) explicitly characterizes design as an iterative and subjective process, while GraphicBench (Ki et al., 2025) frames creative design as an open-ended setting in contrast to tasks with well-specified goals. There is therefore generally no deterministic oracle that can decide whether a professional design has satisfied its brief, and learning from such interactions requires reasoning from noisy multimodal judgments rather than treating an outcome reward as ground truth.

B Additional Graphic Design Agent Details

Runtime. Each user request invokes an iterative tool-calling loop in which the model plans, executes tools, observes their structured outputs, and revises the design. The runtime supports parallel tool execution, parameter validation and repair, failure recovery, multimodal previews, and context compression for long trajectories.

Asset-grounded creation and rendering. The agent constructs designs from retrieved assets rather than generating all pixels directly. Candidate assets are retrieved from Adobe Stock (Adobe Inc., 2026), enriched with visual attributes such as dominant color, subject position, and text-safe regions, and diversified using maximal marginal relevance (Carbonell & Goldstein, 1998). A production render engine provides typography using global paragraph optimization (Knuth & Plass, 1981), compositing, masks, effects, gradients, multilingual layout following the Unicode bidirectional algorithm (Unicode Consortium, 2025), and subject segmentation based on U^2-Net (Qin et al., 2020). Designs can be exported to editable application formats as well as print-ready PDF.

Verification and offline evaluation. During execution, the agent periodically renders its current document for multimodal inspection and runs deterministic checks over properties including text overflow, occlusion, alignment, visual balance, and contrast, including WCAG-style contrast constraints (World Wide Web Consortium, 2024). Document state is serializable, allowing completed trajectories to be deterministically rendered and evaluated offline. This separates the comparatively expensive multimodal reward computation from the user-facing execution path.

C THE SKILL BANK

The skill bank is the agent's evolving, interpretable memory: a versioned collection of natural-language skills, each the stored form of a reusable procedure (Section 1) for one class of image-creation task. This section describes the static object (representation, retrieval, injection); Section 3 describes how the harness changes it.

C.1 SKILL REPRESENTATION

A skill is a single Markdown file (`SKILL.md`) with YAML front matter. The bank is partitioned into three sub-banks by `app_mode`—raster (Photoshop equivalent), vector (Illustrator equivalent), and page_layout/any (InDesign equivalent and cross-app)—and the folder name must equal the skill's `name`. The front matter carries:

- `description` — 200–600 characters of trigger phrases a designer would type. This is the **sole retrieval signal**; the body text does not affect whether a skill is retrieved, only what the agent reads once it is.
- `app_mode` $\in$ {`raster`, `vector`, `page_layout`, `any`} — scopes relevance scoring to the matching sub-bank.
- `tool_sequence` — the 5–10-tool critical path that defines the workflow (and, optionally, a scoped tool list).
- `prerequisites` — other skills pulled in automatically (one hop) when this one matches.
- `status` $\in$ {`candidate`, `stable`, `deprecated`} and a `version` counter.

The body has five required sections: when to use, clarify before starting, steps (each step names a tool and its key parameters), tips (domain knowledge the model would not otherwise have), and error handling (a situation→action table). Per-skill statistics live outside the file, in a separate store, so the skill itself stays in clean, shippable form; see Section 3.

C.2 RETRIEVAL

Because briefs can be long and subject-heavy while skills are operation-centric, retrieval is two-stage.

(1) Query distillation. A frozen LLM maps the request to the core operations it requires—e.g. "double exposure, background removal"—and is instructed to ignore subject matter (names, places, brands), adjectives, and color values, returning at most six such keywords. This makes the query short, focused, and aligned with the operation-centric `description` fields, so each operation retrieves the procedure that implements it.

(2) Overlap scoring for ranking. Each skill s is scored by token overlap between the distilled query q and its description:

$$\text{score}(s) \;=\; \frac{\big|\,\text{tokens}(q) \cap \text{tokens}(\text{desc}(s))\,\big|}{\big|\,\text{tokens}(q)\,\big|}. \tag{2}$$

Each distilled operation is scored independently, with a bonus for matching a skill's name so an operation maps to the skill actually named for it rather than one that merely shares a token, and `app_mode` scopes the score to the matching app (a raster skill scores 0 in a vector session). Crucially, the scores do not gate what the agent can see—they only rank the bank: **2–4** skills are flagged as likely-relevant per turn—those whose name the operation matches, that clear $\tau = 0.2$, and that lie within $0.6\times$ the top score (up to 4), padded up to 2 when fewer clear the bar. Retrieval runs once per turn; how the ranked bank is then presented is described next.

Why token overlap, not embeddings. Token overlap keeps the cold-start seeding (Section G) and the runtime retriever aligned on the same tokenizer, so trigger phrases written into a skill's description during seeding are guaranteed retrievable by the same procedure. An embedding retriever would force the seeding pipeline to guess what an embedding space prefers, decoupling authoring from routing.

C.3 INJECTION: PROGRESSIVE DISCLOSURE VIA LOAD_SKILL

The agent surfaces skills by progressive disclosure rather than forcing their bodies into the context. Once per turn the injector appends a compact catalog to the system prompt: every skill as its name, `app_mode` tag, and one-line `description`, with the ranked matches from §C.2 starred and floated to the top and a token budget bounding the listing. The full bodies are not injected. When the model judges a skill relevant, it calls the `load_skill(name)` tool, which returns that skill's steps, tool sequence, tips, and error handling as a tool result in the conversation history—so only the skills the agent chooses to read enter the context. The catalog is app-agnostic: skills from all three apps are shown and tagged, so a cross-app request can be planned from the outset, while app separation happens at execution, where the tool list is already scoped to the active document's app. Optionally, that tool list can be narrowed further to the union of the loaded skills' `tool_sequence` plus a small always-on core—shrinking the 230-tool catalog to the handful a task needs—though this is off by default in our experiments. When nothing is starred, the turn is recorded as an uncovered coverage gap (Section 3).

C.4 TRAJECTORY RECORDING

When enabled, a recorder writes one record per turn: the distilled query, the retrieved skills with their scores, the tool calls made, and a pointer to the turn's `XML` document snapshot, tagged covered or uncovered. Scores are left empty and back-filled by the offline grader. This record is the substrate for both credit assignment (which skills were active on a failure) and coverage-gap detection (which requests had no skill or misused skills).

D EVALUATION KIT: THE GRAPHIC-EVAL RUBRIC

The reward that drives self-evolution and the metric that reports progress are the same frozen judge, which we call **Graphic-Eval**. Reusing one judge for both roles is deliberate: a skill is only rewritten against the criterion it will later be measured on. Graphic-Eval scores a single rendered design against its brief along three axes—completeness, aesthetics, and critique—each realized as a constrained, JSON-only judge call. This section specifies each axis exactly as used.

D.1 RENDERING

Grading operates on images, not document models. A turn's `XML` snapshot is rendered to a PNG at a fixed scale; production turns are judged against the cumulative request (initial brief plus all refinements up to that turn), since the design state at that point should satisfy everything asked so far, not just the latest instruction.

D.2 COMPLETENESS (DECOMPOSE THEN CHECK)

Completeness measures the degree to which the output delivers what the brief asked for, in two LLM calls.

Decompose. A first call, prompted as a senior graphic designer, extracts 3–10 scorable requirements from the brief. It is instructed to write each requirement so it can be judged met/partial/not-met, to bake flexibility into soft or metaphorical asks ("evoke vintage") while keeping concrete instructions precise, to group related attributes into one requirement, and—critically—to **exclude** requirements about which tool or software to use or how to implement. Completeness thus scores outcomes, not the path taken, which is what makes it a fair reward for an agent free to choose its own tools.

Check. A second, multimodal call receives the brief, the extracted requirement list, and the rendered image, and labels each requirement `met`, `partial`, or `not_met`, with a reason required for

anything less than met. The score is the earned fraction,

$$\text{completeness} \;=\; \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1.0 & \text{if requirement } i \text{ is } \texttt{met}\\ 0.5 & \text{if } \texttt{partial}\\ 0.0 & \text{if } \texttt{not_met}\end{cases} \tag{3}$$

The per-requirement `reason` strings for unmet items are exactly the "why-bad" signal the reflector consumes (§3).

D.3 Aesthetics (anchored 1–10)

A separate multimodal call rates the design purely on visual quality—composition, balance, negative space, hierarchy, typography, color harmony, form language, and craft (alignment, overlap, readability)—explicitly ignoring whether the brief was followed, so that aesthetics and completeness stay disentangled. The judge returns an integer 1–10 against fixed anchors (6 = passable but unremarkable, 8 = professional and polished, 9–10 = exceptional, $\leq$ 3 = weak) plus a short summary; we store $\text{aesthetic} = \text{score}/10 \in [0, 1]$. Keeping aesthetics free of brief-adherence is what justifies the smaller weight w_a in Eq. 4: a beautiful design that ignores the brief should not earn a high pass rate.

D.4 Critique (counted problems)

A third multimodal call, prompted as a senior design director, performs an aesthetic critique: for each of eight dimensions—hierarchy/typography, contrast/readability, spacing/alignment, color, composition/balance, craft/execution, consistency, originality—it lists every visible problem and marks its severity `major` or `minor` (skipping dimensions with none, and suggesting no fixes). We report the major and minor counts. Critique is a diagnostic axis: unlike the two scalar scores it localizes where a design is weak, which is useful for qualitative analysis and for sanity-checking that aesthetic scores move for the right reasons.

D.5 Combining axes and judge backends

The scalar pass rate combines the two scores via Eq. 4 with $(w_c, w_a) = (0.7, 0.3)$; this single number feeds the difficulty estimate $\bar{p}$ and the good/bad banding defined in Appendix E. The judge is pluggable across two backends—an Azure-hosted GPT model and a Bedrock-hosted Claude model—both driven with strict JSON-mode outputs. Backend choice matters for one risk: when the grader and solver are from the same model family, the grader may over-reward its sibling's outputs. The pairwise gate of Section 3.3 mitigates this risk by comparing solver outputs pairwise rather than relying on absolute scores; the grader can also be switched to the cross-family backend if $\bar{p}$ skews uniformly high.

D.6 Cost signals

Alongside quality, every rollout records cost: the number of tool calls, wall-clock latency, and input/output token counts (parsed from the agent's completion marker, needing no telemetry backend). Because the skill bank's second effect is to shrink the tool surface and the number of iterations, these cost signals are reported as first-class outcomes, not afterthoughts.

E Scoring and outcome binning

Every rollout is scored by the frozen `Graphic-Eval` grader (§D), which returns a completeness fraction, an aesthetic score, and a per-requirement "why-bad" rationale for anything unmet. We reduce the rubric to a single scalar that leads with completeness,

$$s_j \;=\; w_c \cdot \text{completeness}_j \;+\; w_a \cdot \text{aesthetic}_j, \qquad (w_c, w_a) = (0.7,\, 0.3). \tag{4}$$

Completeness leads because a design can look good yet ignore the brief; the smaller aesthetic weight keeps a polished-but-wrong output from masking a task failure.

Binning. The scalar bins each trajectory with a single threshold $\tau = 0.6$: $s_j \geq \tau$ is a success and $s_j < \tau$ is a failure. A failure counts against every skill the trajectory retrieved, with no blame weighting; a success gives uniform credit. This is the only place absolute scores are used—to select what to revise. Nothing ships on the strength of an absolute score, because the same image drifts run-to-run under a VLM grader (Appendix F).

Selection and triggering. When 50 new graded records accumulate, an evolve round fires. We select for revision every skill whose failure-count—the number of failed trajectories in which it was retrieved—meets $m = 2$, ordered most-failing first and capped at 50 skills per round. The rule is intentionally permissive: failure-count is the sole signal—no blame weighting, no score floor, and success counts are not consulted—since a skill that both succeeds and fails often is still worth a rewrite attempt. A per-skill quality score is maintained for observability as an exponential moving average of the per-outcome signal (good $\rightarrow$ 1, bad $\rightarrow$ 0), reset to 0.5 on each accepted rewrite (version bump); it does not currently drive selection or routing.

F REPLAY GATE DETAILS

Why relative. A VLM grader's absolute score for a fixed image varies across runs, and the solver is itself stochastic. A rule of the form "accept if the mean score rose" therefore confuses three sources of variation—real improvement, judge drift, and solver noise—and admits regressions whenever the latter two happen to align. The gate removes two of the three by construction: both arms are re-run fresh in the same batch, so any drift in that batch shifts them equally and cancels in the difference, and the comparison is pairwise rather than absolute, so the judge is only ever asked which of two images better satisfies the brief.

Replay set and arms. The replay set and the baseline arm differ by axis. For a rewrite (deepening), the replay prompts are drawn from trajectories that retrieved the skill, stratified by outcome: two are sampled from well-scoring trajectories and two from poorly-scoring ones. The poor half tests whether the candidate fixes the failures that motivated the rewrite; the good half tests whether it preserves what the incumbent already handled, since a revision tuned only on failures can silently break cases that previously succeeded. The candidate (V2) is replayed against the incumbent (V1). For a mint (widening), there is no incumbent version to protect, and the skill is proposed to fill a coverage gap rather than to repair a specific failure; the replay prompts are therefore sampled from the trajectories in the triggering cluster without conditioning on their scores, and the candidate is replayed against the agent with no skill retrieved. In both cases the two arms are executed in the same batch with identical context otherwise.

Judging and tie band. For each record, every candidate rollout is judged against every baseline rollout by a pairwise grader asked which of the two images better satisfies the brief, with presentation order swapped to mitigate position bias (Wang et al., 2024b; Zheng et al., 2023). The votes aggregate into a per-record win rate $r \in [0, 1]$ for the candidate. A tie band of half-width δ around 0.5 absorbs judge jitter on near-identical outputs: a record is an improvement if $r > 0.5 + \delta$, a regression if $r < 0.5 - \delta$, and a tie otherwise.

Acceptance.

$$\textbf{accept} \iff \big(\nexists \text{ record with } r < 0.5 - \delta\big) \wedge \big(\exists \text{ record with } r > 0.5 + \delta\big). \tag{5}$$

A candidate that merely ties everywhere is rejected: it is not worth a version bump, and shipping it would reset the skill's statistics for no measured gain. The rule never trades a regression on one case for a gain on another. This is strictly more conservative than maximising expected quality, and deliberately so—in a deployed product a visible regression costs far more than a missed improvement, a stance shared with safe policy improvement against a deployed baseline (Laroche et al., 2019; Thomas et al., 2015).

On rejection. A rejected rewrite is discarded and the skill's consecutive-failure counter—which drives the targeted→major escalation—is incremented; a rejected mint is discarded but its occurrences remain in the coverage pool, eligible for a later attempt once more evidence accumulates. In

both cases the proposal and its before/after snapshot are logged, so the bank's history records what was tried and refused, not only what shipped.

G Cold Start from Documentation

The evolution loop improves an existing bank, but a fresh deployment has no traffic to learn from and an empty bank retrieves nothing. We therefore seed the bank from product help documentation before any user arrives, via a four-phase pipeline. Phases that call an LLM reuse the same frozen model as the rest of the system; two of the four phases use no LLM at all.

Phase 1 — Intent clustering. Stream the documentation corpus (help articles keyed by a stable id), filter to English, dedupe, and extract per-product pages and their topics. A few LLM calls cluster the topics into 20–25 coherent themes, each a candidate skill.

Phase 2 — Page assignment (no LLM). Score every documentation page against every theme with the same token-overlap tokenizer the runtime retriever uses, and attach the top pages to each theme. Using the runtime tokenizer here guarantees that what seeds a skill is what will later retrieve it.

Phase 3 — Skill generation. One LLM call per theme produces a `SKILL.md`: trigger phrases for the `description`, the relevant documentation pages as source, a tool whitelist drawn from the real catalog, and the existing bank as dedup context. A `NO_SKILL` escape hatch lets a theme that is purely UI chrome (no reusable workflow) emit nothing.

Phase 4 — Validation and save (no LLM). Validate every referenced tool name against the live tool catalog, reject near-duplicate descriptions, check structural completeness (all five body sections present), write the files into the correct sub-bank, and initialize each skill's statistics at a neutral score.

Human review checkpoints follow Phase 1 (inspect the theme clusters) and Phase 3 (inspect the raw skills). In our deployment this pipeline produced an initial bank of **76 skills** (26 Photoshop-, 25 Illustrator-, and 25 InDesign-equivalent or cross-app). From that point the loop of Section 3 takes over: production and prompter traffic grade the seeded skills, failures drive reflection, and the gate ships only improvements. Cold start thus provides an initial recall floor (a skill exists for common intents); the harness then deepens those skills and widens the bank.

H Success Rate on Specialized Design Tasks

Table 4: **Success rates on specialized graphic design tasks.** Percentage of prompts completed with a valid design within 900 s. On hosted Claude backbones, EVOLVE reaches near-perfect completion. Results for `Qwen3.6-27B` reflect local deployment stability constraints (see text).

Agent	Method	OpenCOLE	GraphicBench	CreatiDesign	BannerRequest400	Avg
Claude-Opus-4.6	BASE	100.0%	98.7%	94.7%	100.0%	98.3%
	EVOLVE	**100.0%**	**100.0%**	**96.7%**	**100.0%**	**99.2%**
	Δ	0.0%	+1.3%	+2.0%	0.0%	+0.9%
Claude-Sonnet-4	BASE	98.0%	97.3%	94.0%	74.7%	91.0%
	EVOLVE	**100.0%**	**99.3%**	**100.0%**	**100.0%**	**99.8%**
	Δ	+2.0%	+2.0%	+6.0%	+25.3%	+8.8%
Qwen3.6-27B	BASE	**77.3%**	50.0%	**56.7%**	**78.0%**	**65.5%**
	EVOLVE	63.3%	**72.7%**	52.7%	64.0%	63.2%
	Δ	-14.0%	+22.7%	-4.0%	-14.0%	-2.3%

Table 4 details the generation success rates of the BASE and EVOLVE agents on the specialized graphic design benchmarks. While highly capable models like `Claude-Opus-4.6` al-

ready exhibit strong baseline stability, the EVOLVE framework further pushes their completion rates to near-perfect levels (averaging 99.2%). The stabilizing effect is most pronounced for `Claude-Sonnet-4` on the highly complex BannerRequest400 benchmark. Without the skill bank, Sonnet struggles to handle intricate layout and typographical constraints, leading to a noticeable drop in success rate (74.7%). By leveraging pre-verified workflows, the EVOLVE agent completely mitigates these catastrophic tool-use failures, achieving a flawless 100% success rate (an absolute improvement of $+25.3\%$). This demonstrates that the skill bank acts as a critical safety net, ensuring high reliability in demanding, multi-step professional design tasks. For `Qwen3.6-27B`, completion rates are primarily dominated by timeouts, and connection drops rather than reasoning or tool-use failures. Consequently, its success rate fluctuations across benchmarks (e.g., $+22.7\%$ on GraphicBench vs. drops elsewhere) are largely confounded by local deployment stability.

I PERSONALIZE: WHERE RETRIEVAL-AND-INJECTION IS NOT ENOUGH

Even after the skill bank widens and deepens coverage, a residual set of cases on our internal user-data benchmark remains unsatisfactory—covering an intent does not guarantee that a particular user is satisfied, and users may judge a result differently from the automated verifier. We selected a small number of such flagged prompts and group them into three personalization modes, with one representative case each in Figure 8. Throughout, "No skill" denotes the BASE agent and "With skill" (Ours) denotes the agent that retrieves and applies the user's personalized skill; both run on the same frozen backbone.

Mode 1 — Method preference (Fig. 8, top). Here the effect is well within the agent's reach, but the user prefers a different realization than the harness default. For "add a falling-snow effect", the default synthesizes snow with a procedural particle field, whereas the user's tutorial specifies the classic layered recipe (fill $\rightarrow$ add-noise $\rightarrow$ blur $\rightarrow$ threshold $\rightarrow$ screen, repeated for depth) that creates a more realistic (less artificial) snow effect. Injecting the user's technique as a skill is by itself insufficient: recognizing an effect it already "knows", the model reverts to its default method. Making the preference take effect required two ingredients beyond injection—conditioning retrieval on the user (encoding that this user prefers the noise-based method) and scoping the tool set so the default shortcut (the particle generator) is unavailable. The agent then reliably reproduces the requested technique, yielding the "With skill" panel.

Mode 2 — Perception and precision limits (Fig. 8, middle). For a Droste (recursive picture-in-picture) effect inside a photographed frame, the bottleneck is not method choice but the limited perception of fine geometric detail—by the agent and, crucially, by its automated verifier. Correctly locating a frame's inner boundary and deciding whether an inset sits exactly on it is beyond reliable VLM judgment, which passes placements that are visibly off. A textual skill does not change behavior; a deterministic placement primitive fixes an individual step but not the overall composition; and replacing the VLM verifier with a numeric one (code that measures the pixel band straddling the boundary) improves detection—it catches misalignments the VLM accepts—yet the agent still cannot execute pixel-exact placement, and per-layer localization degrades as recursion deepens. With the skill the result is still far from perfect or stable, but it is a modest improvement over the default: the nested frames are seated somewhat more squarely and concentrically ("With skill", right) than the visibly broken, misaligned insets the BASE agent leaves behind (left). Across our attempts the residual error is a perception/verification gap that current automation does not close; certifying that "the boundary was found correctly" still requires a human in the loop. We include this as an honest, mostly negative result: the skill helps at the margin but does not solve the precision limit.

Mode 3 — Complex hand-crafted targets: vibe drawing (Fig. 8, bottom). We use vibe drawing to name a characteristic failure mode of the agent: given a prompt—here, together with a target picture of the intended result—it produces something that roughly resembles, is spiritually like, what was asked, but is not the actual artefact and cannot be used directly; to get the real thing one would have to draw it again. Our example asks for the base unit tile of a De Nigris "3/4 Decorative" pattern, with the intended tile supplied as a reference image. Without the skill, the BASE agent produces exactly such a vibe drawing (bottom left): it reads like the reference—diagonal gray bands and corner arcs—but is a different shape and structure from the target, so it is not the requested tile and would have to be rebuilt from scratch. With the skill—distilled from the user's own hand-guided

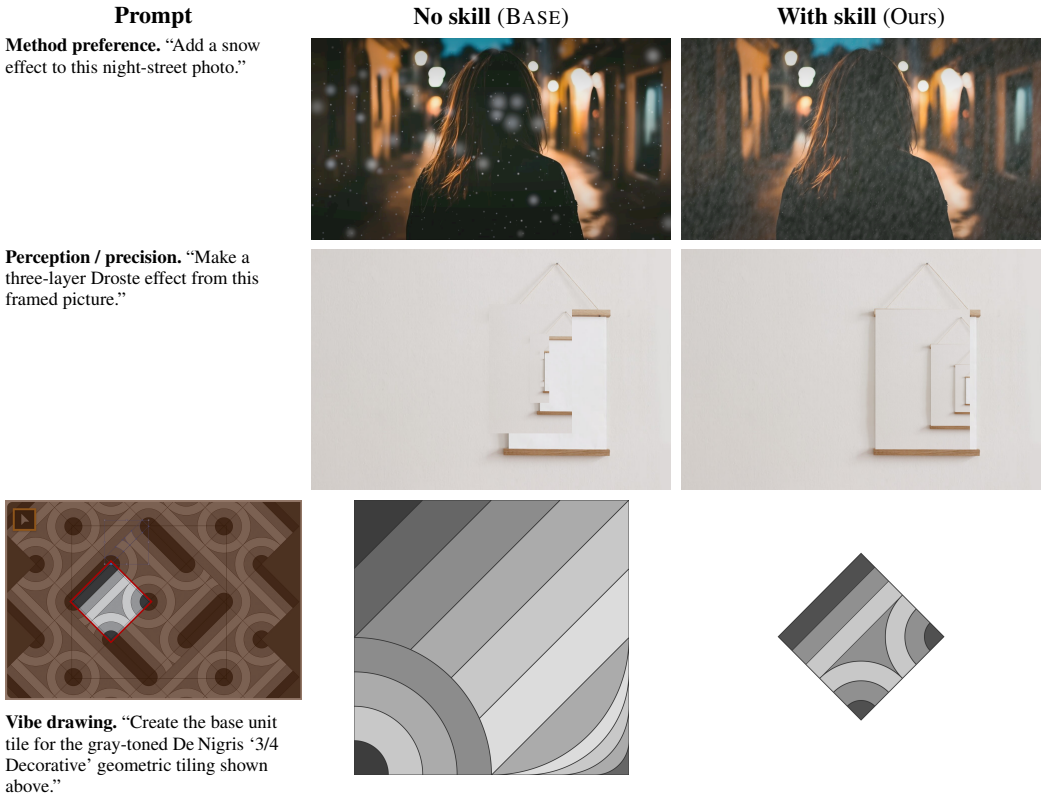
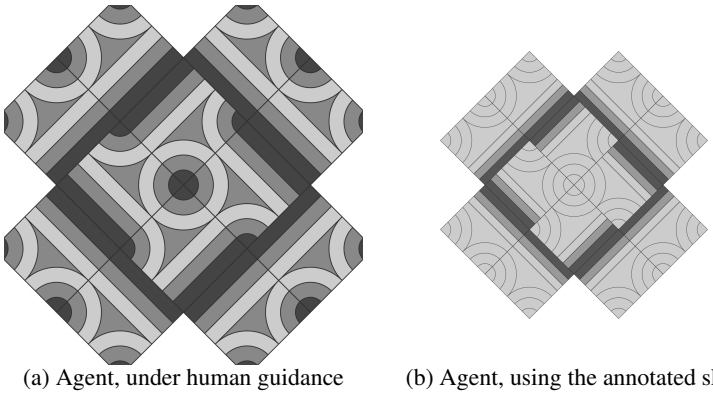


Figure 8: **Three personalization modes.** Each row is a user-flagged prompt run without the personalized skill (BASE, left) and with it (Ours, right), on the same frozen backbone. Top — method preference: the default renders snow as a sparse particle field; the skill, enforced via preference-conditioned retrieval and tool scoping, reproduces the user's layered noise technique. Middle — perception/precision limit: both attempts at a recursive in-frame Droste effect leave the nested frames imperfectly seated; the skill helps but the residual misalignment reflects a verification gap that still needs a human in the loop. Bottom — vibe drawing: for the base unit tile of a De Nigris "3/4 Decorative" pattern, the BASE agent drifts into an uncontrolled design, while the distilled skill reconstructs the user's hand-tuned tile from a single prompt. The mosaic prompt is given together with the target pattern shown above it (a De Nigris "3/4 Decorative" tiling); the skill's tile is a correct unit that tessellates into it, whereas the BASE vibe drawing does not.

construction and backed by deterministic construction primitives—the agent instead reproduces the target exactly, in one pass (bottom right), and delivers it as a fully editable vector state: layered paths, arcs, and fills that can be re-edited and tiled, not a flattened look-alike. Because it is a correct unit tile, it repeats seamlessly into the full "3/4 Decorative" pattern the user is after (shown with the prompt in Fig. 8), which is what the single tile is ultimately for. This mode captures where an otherwise capable agent still falls short—it can already vibe-draw a convincing approximation—and shows the personalized skill supplying what is missing: the exact, directly usable, editable artefact the user actually asked for.

Beyond the base tile: limits of the skill-based route. Encouraged by the base-tile result, we tried to push the same recipe one level further—to the full tessellated pattern, not just its unit tile. Here the method visibly strains, and the difficulty appears along all three axes a skill must survive: annotating it, distilling it, and using it. **(1) Annotation.** Even with patient step-by-step guidance, the human effort grows at least proportionally with the number of steps: every additional step (another ring of tiles, a mirror, a recolor) needs its own round of correction, so hand-guiding a long construction to completion becomes progressively more laborious. **(2) Distillation.** The resulting trajectory runs to hundreds of turns interleaving correct and mistaken steps; deciding which to keep and how to compress a long, noisy demonstration into a clean, reusable skill is itself unsolved. **(3) Use.** Even



(a) Agent, under human guidance (b) Agent, using the annotated skill

Figure 9: **Pushing past the base tile to the full pattern.** (a) The full "3/4 Decorative" pattern a user builds by hand—the annotated target. (b) What the agent produces from that same hand-annotated skill: the tiling layout is roughly preserved, but the three-tone gray fills and fine structure are lost, so the result is a discounted version of the target. Fidelity leaks at each of the three stages—annotating the demonstration, distilling it into a compact skill, and getting the frozen model to follow a long skill—and the losses compound.

when the distilled skill is written correctly, a long, many-step skill exceeds what the frozen model reliably follows—partway through it stops adhering to the script and reverts to its own behavior. Because each axis leaks a little fidelity, the losses compound: Figure 9 contrasts the human-annotated full pattern with what the agent produces from that same annotated skill—the overall tiling layout survives, but the three-tone fills and fine detail wash out, and the result is visibly discounted. The takeaway is that skill distillation is most effective for short, self-contained procedures such as the base unit tile; for targets that demand very long, many-step constructions, a purely skill-based route degrades on all three axes at once, and reaching them reliably likely needs mechanisms beyond prose skills—deterministic macros, plan-level orchestration, or model updating.

Discussion: limits and future work. Taken together, these cases delineate where a retrieval-and-injection skill bank stops, and point to personalization as a distinct axis of a continually evolving system. Three limits recur. (i) A skill's text can carry knowledge but cannot, on its own, override a model's default procedure; personalizing method choice therefore needs preference-conditioned retrieval and tool-level enforcement, not merely better-worded guidance. (ii) Some failures are perception and verification limits of the underlying multimodal model; moving verification from VLM judgment to numeric measurement helps but does not eliminate them, leaving human-in-the-loop as the current fallback for precision-critical detail. (iii) Skill distillation is powerful but scale-limited: it excels at short, self-contained procedures—like a base unit tile, which it makes reproducible from a single prompt—but degrades on very long, many-step constructions, where fidelity leaks at every stage (annotating the demonstration, distilling it into a compact skill, and getting the frozen model to follow a long skill) so the losses compound. Extending the approach to such targets likely requires representations beyond prose skills—deterministic macros for the precise sub-steps, plan-level orchestration, or model updating. More broadly, beyond widening (new capabilities) and deepening (more reliable capabilities), a deployed system must also specialize to individual users' methods, standards, and hard-to-specify targets. Our current framework addresses this personalization axis only partially, and we view preference-aware retrieval, numeric and human-assisted verification, and plan-level enforcement as the natural next steps.

J Gallery on the Held-Out Prompt Set

Figures 10 and 11 show the designs our system produces on the held-out prompt set, one panel per prompt, for the Opus 4.6 and Sonnet 4 backbones respectively, both run with a low thinking budget. The two figures cover the same 193 prompts and are rendered with the round-5 skill bank; the corresponding no-skill baselines and the side-by-side pairs are omitted here for space. The prompts span the full range of the benchmark—posters, flyers, invitations, logos, brand and UI layouts,

photographic composites, and low-level primitive requests—and vary widely in canvas aspect ratio, which is preserved for every panel: images are uniformly scaled and tiled into rows of equal height, never cropped or stretched.

The benchmark is an internal test set, and our content policy does not permit releasing the prompt text. Within that split nothing further is selected: both galleries show every prompt, in a fixed random order. This allows readers to assess the system's typical output quality rather than a hand-picked selection of its best results. Individual panels are necessarily small at this density; they are meant to convey the aggregate distribution of quality, layout structure, and stylistic variety.

K Ethical Statement

This work studies an offline evolution and replay pipeline for a graphic-design agent. The evolution data include design briefs derived from pre-existing user traffic collected in the course of normal product use, together with LLM-augmented variants. No users have been recruited and no additional user data or annotations have been collected specifically for this study to date. We plan to conduct a small-scale user study to evaluate the effectiveness of EVOLVE from a human perspective. The held-out internal benchmark is human-authored, and the main experiments use automated evaluation rather than human reward labels. User-guided personalization examples are evaluated separately and are excluded from the shared skill pool and all main-paper experiments.

The use of pre-existing user data raises privacy and confidentiality considerations. User-derived data were accessed and processed within the organization's established data-governance and access-control framework. User-derived inputs were filtered and processed to reduce the exposure of personally identifiable or other sensitive information. We do not release user-derived prompts or trajectories. Because the evaluation benchmark is internal, we also withhold its full prompt set and trajectory contents; the qualitative gallery contains only rendered outputs for 193 held-out prompts, without the corresponding prompt text.

The system uses hosted proprietary language-model services through Amazon Bedrock and Azure, locally served models, and assets retrieved from Adobe Stock. User-derived data may be processed by these hosted model services under the organization's enterprise agreements with the respective providers, which prohibit the use of such data for training the providers' models. Third-party assets and user-derived materials are not publicly released except where such disclosure is authorized and consistent with applicable licenses and data-use requirements.

Finally, graphic-design automation can potentially be used to create misleading, deceptive, or rights-infringing content. Deployment of such systems should therefore retain appropriate content safeguards, access controls, and human oversight.

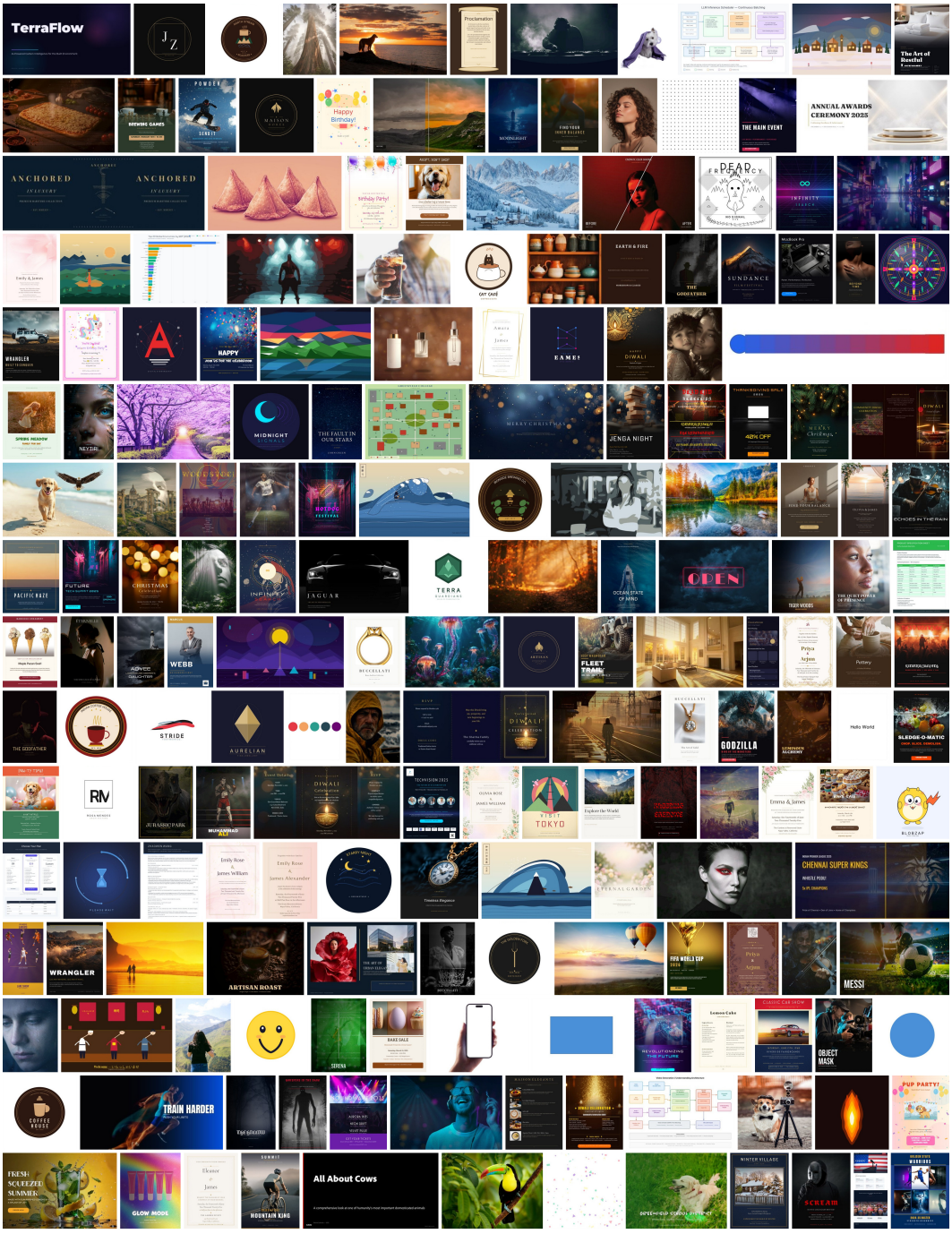


Figure 10: **Opus 4.6 backbone (low thinking).** All 193 held-out prompts rendered with the round-5 skill bank, in random order. Prompt text is withheld (internal test set); aspect ratios are preserved and no panel is selected for quality.

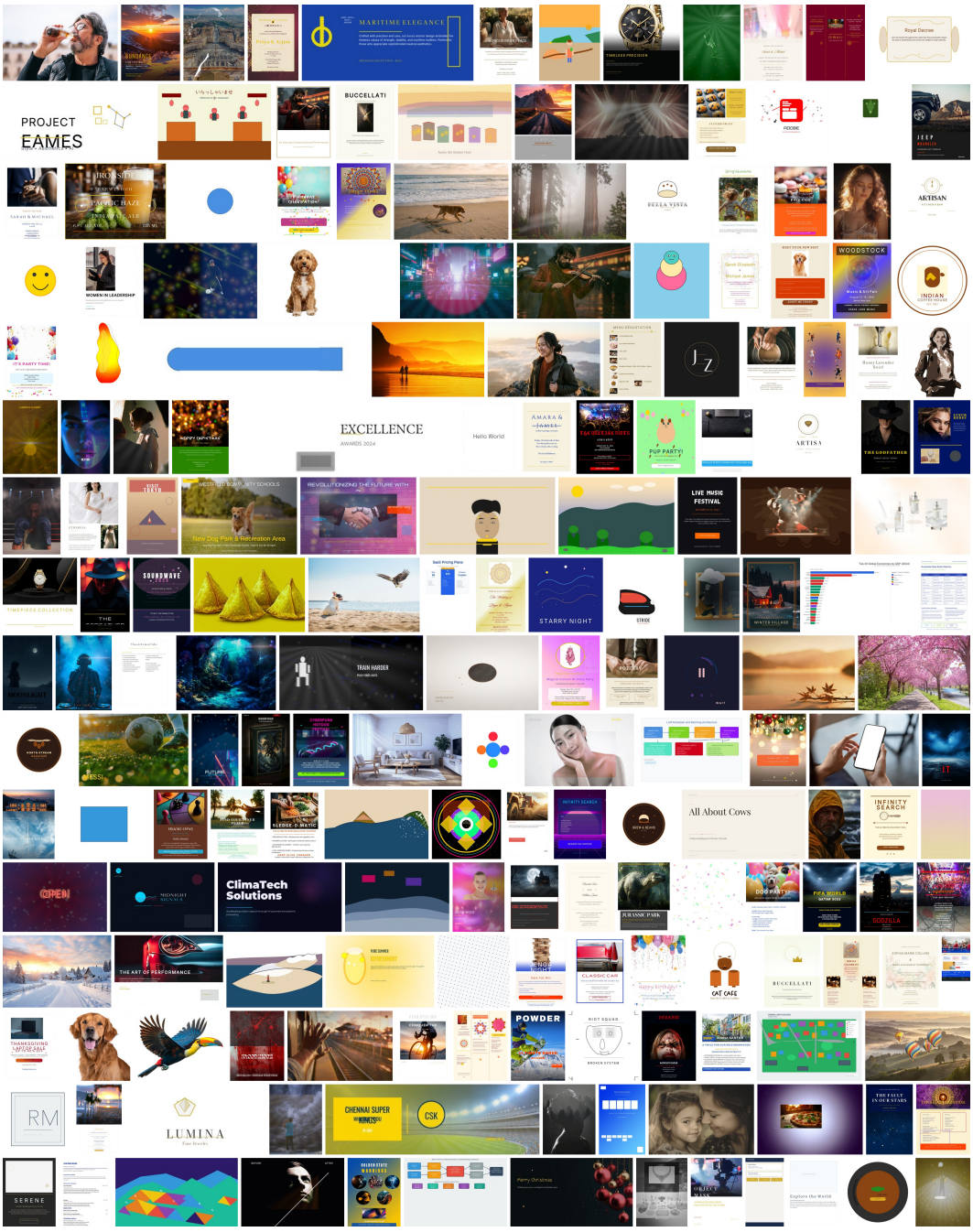


Figure 11: **Sonnet 4 backbone (low thinking).** The same 193 prompts as Figure 10, rendered on the smaller backbone under the same protocol.