EvoSkillGen: Hierarchical Evolution of Compositional Skills for Agentic Image Generation

Anonymous authors
Paper under double-blind review

Abstract

Image generators can synthesize high-fidelity images, but complex requests involving external knowledge, compositional and spatial constraints, or targeted edits with content preservation often require more than one-shot synthesis. Agentic image generation addresses such requests through retrieval, reasoning, and planning. Yet tasks demand different operations and information flows; selecting predefined workflows alone does not turn execution experience into improved reusable procedures. We propose **EvoSkillGen**, a self-evolving framework for text-to-image generation and image editing. It represents workflows as executable Skills composed of meta-operations with explicit instructions and artifact dependencies. Skill Adaptation (L1) selects task-specific workflows and constructs temporary Skills during evolution; Skill-Bank Evolution (L2) checks generated images against task requirements and uses execution traces to revise Skills and distill temporary workflows. The quality of generated images guides which candidate changes are retained in the Skill bank. We also introduce GenAgentEval, an evaluation suite built from six existing benchmarks to assess final-image quality, workflow-time overhead, and execution-trace diagnostics. Averaged over both judges, EvoSkillGen improves macro-averaged image-quality scores over the strongest baselines by 0.9–1.8 points for text-to-image and 2.3–2.8 for editing across agent configurations. Among agent methods, its workflow-time overhead is second-lowest for text-to-image and lowest for editing.

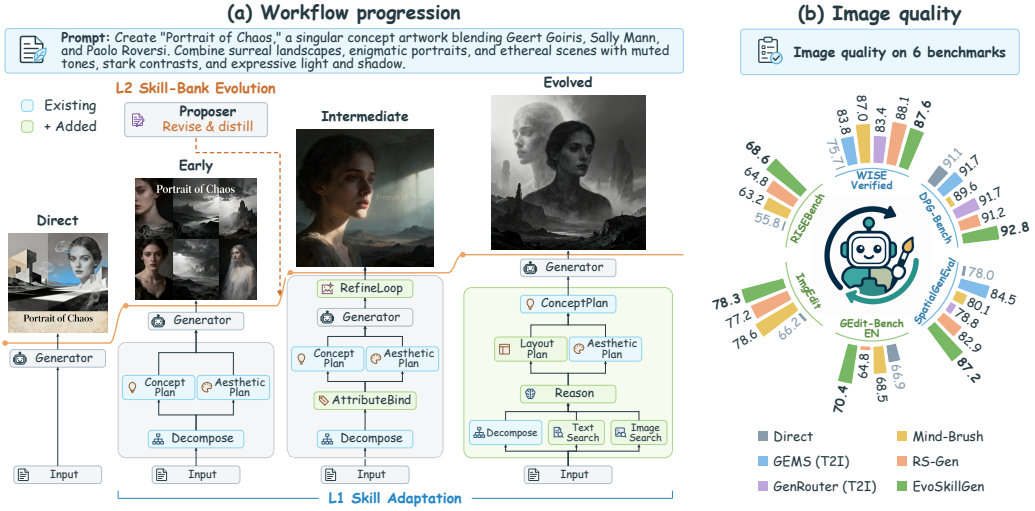


Figure 1: Workflow evolution and image quality. (a) For one complex prompt, Direct generation is followed by Early, Intermediate, and Evolved Skills. Their operation graphs introduce and reorganize planning, refinement, search, and reasoning, with corresponding changes in the generated images. (b) The six benchmarks cover knowledge grounding, compositional and spatial generation, and instruction-following, content-preserving, and reasoning-based editing. Scores use the Strong agent and average both judges; EvoSkillGen has the highest three-benchmark macro-average for both text-to-image generation and image editing.

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.

1 Introduction

Image generation models can produce high-fidelity images from everyday descriptions (Rombach et al., 2022; Deng et al., 2025), yet complex requests demand more than visual synthesis. Knowledge-intensive tasks require grounding in world knowledge (Niu et al., 2026), complex scenes require correct spatial relations (Wang et al., 2026), and editing must follow instructions while preserving unrelated content (Ye et al., 2025). These needs motivate agentic image generation, in which an agent organizes retrieval, reasoning, and planning to translate user intent into usable generator inputs and constraints (He et al., 2026a).

Prior work improves generator inputs through prompt adaptation (Hao et al., 2023) and visual feedback (Wan et al., 2025; Kovalev et al., 2026). GenClaw uses executable code sketches to express visual structure before rendering (Ye et al., 2026). GEMS loads domain Skills on demand and uses trajectory memory for iterative refinement (He et al., 2026b); GenRouter selects predefined workflows and generators from past experience (Chen et al., 2026a). Reusable skill libraries (Wang et al., 2023) and feedback-based system optimization (Lee et al., 2026; Luo et al., 2026) further motivate learning generation workflows. This raises a central question: ***how can an agent compose and continually improve generation workflows on demand?***

Two related challenges remain. (1) Adapting operations to each request. A fixed sequence may include unnecessary steps or omit needed ones. The agent must determine which operations to use, their order, and how retrieved evidence and intermediate outputs flow into layout and generation. (2) Turning failures into reusable improvements. An image can reveal an unmet requirement without identifying which operation, dependency, or instruction caused it. The agent must use the execution trace to locate a useful change and decide whether to retain it for later tasks.

We propose **EvoSkillGen**, which learns a bank of executable Skills for text-to-image generation and editing. Each Skill specifies applicability conditions and composes meta-operations through node instructions and artifact dependencies (Fig. 2). Feedback guides revisions to existing Skills and distillation of temporary workflows. The system evaluates candidate bank changes through generated-image quality before deciding which to retain.

EvoSkillGen separates workflow construction from bank evolution (Figs. 1(a) and 3). Skill Adaptation (L1) selects applicable Skills for tasks and creates temporary workflows for evolution tasks that need a specialist but have no matching Skill. Skill-Bank Evolution (L2) checks generated images against task requirements and uses execution traces to revise existing Skills and distill temporary ones. The resulting bank is frozen before testing.

We introduce GenAgentEval, an evaluation suite that compares image-generation agents by final-image quality, workflow-time overhead, and execution-trace diagnostics. It integrates six existing text-to-image and editing benchmarks, retaining their image-evaluation objectives while adding shared timing and diagnostic protocols.

Averaging the two judges, EvoSkillGen obtains text-to-image image-quality macro-averages of 89.2 and 86.1 with GPT-5.5 (Strong) and Doubao-Seed-2.0-pro (Weak) agents, respectively, exceeding the corresponding strongest baselines by 1.8 and 0.9 points. Editing scores reach 72.4 and 69.3, with gains of 2.3 and 2.8 points in the same order. Among agent methods, its workflow-time overhead ranks second-lowest for text-to-image (34.8 seconds with Strong and 45.3 with Weak) and lowest for editing (73.9 and 69.2 seconds, respectively). Across training checkpoints, frozen-bank evaluations on the same text-to-image test tasks show an overall gain with intermediate fluctuations.

Our contributions are as follows:

- We represent reusable generation procedures as executable Skills with explicit operations, node instructions, artifact dependencies, and applicability conditions for task-specific reuse.
- We propose **EvoSkillGen**, which composes workflows before execution, then uses feedback to revise existing Skills and distill temporary workflows; generated-image quality guides which Skill-bank updates are retained without changing model weights.
- We introduce GenAgentEval, an evaluation suite built from six existing generation and editing benchmarks with shared image-quality scoring, workflow timing, and trace diagnostics.

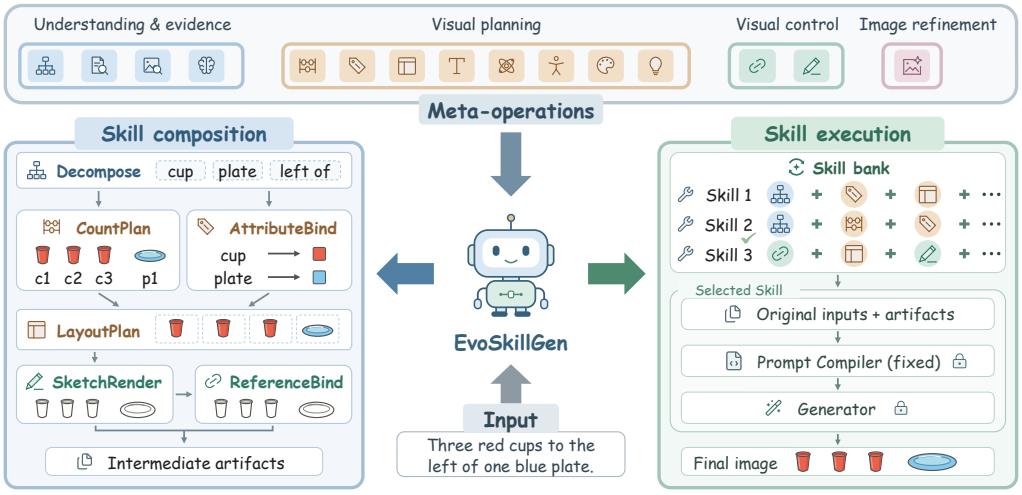


Figure 2: Skill composition and execution. Fifteen meta-operation types support information gathering, visual planning, visual control, and image refinement. The example combines counting and layout operations; each Skill includes a fixed Prompt Compiler for task inputs and node artifacts.

2 The EvoSkillGen Method

2.1 Framework Overview

Fig. 3 shows two levels. A Skill is an executable workflow with applicability conditions, instructed meta-operation nodes, and artifact dependencies. For task $x = (u, \mathcal{I})$ (instruction u, optional input images $\mathcal{I}$), L1 selects and executes a Skill, creating a temporary one during evolution if needed; L2 revises, distills, and consolidates Skills from execution feedback. Each Skill's fixed Prompt Compiler combines original inputs with evidence, layouts, or sketches from its nodes; execution returns an image and an artifact-flow trace.

During evolution, L1 executes the tasks in each batch using the current Skill bank. Once all tasks in that batch finish, L2 uses their feedback to update the bank. The bank initially contains only Direct Skill, which forwards original inputs and is never revised or removed. Model weights and instructions for the Router, temporary-workflow Creator, and Skill-update Proposer remain fixed.

2.2 Meta-Operations and Compositional Skills

Meta-operations define roles, inputs, and outputs. Fig. 2 groups 15 types into information gathering, visual planning, visual control, and image refinement. A Skill $S = (V_S, \mathcal{E}_S, c_S, d_S)$ instantiates selected types as instructed nodes in a directed acyclic graph (DAG), whose edges pass artifacts such as evidence and layouts. A type may recur across nodes for different subproblems (e.g., TextSearch, ImageSearch, Reason). RefineLoop is the exception: at most one follows generation (Appendix A.3). Each Skill has a fixed Prompt Compiler outside the editable meta-operation set. c_S states supported and incompatible capabilities; d_S describes task scope.

Fig. 2's cup-and-plate example uses Decompose, CountPlan, and AttributeBind for objects, counts, and colors; LayoutPlan, SketchRender, and ReferenceBind create a spatial reference. Other Skills may add search or reasoning. Generally, the fixed Compiler combines preparation-node artifacts with original inputs for initial generation. An optional RefineLoop may then revise the image; execution returns the retained result (Appendix A.3).

2.3 L1: Skill Adaptation

L1 derives task-capability tags from current inputs (Fig. 3). The runtime shortlists learned Skills with task-relevant capabilities and no declared incompatibility. The Router then judges whether

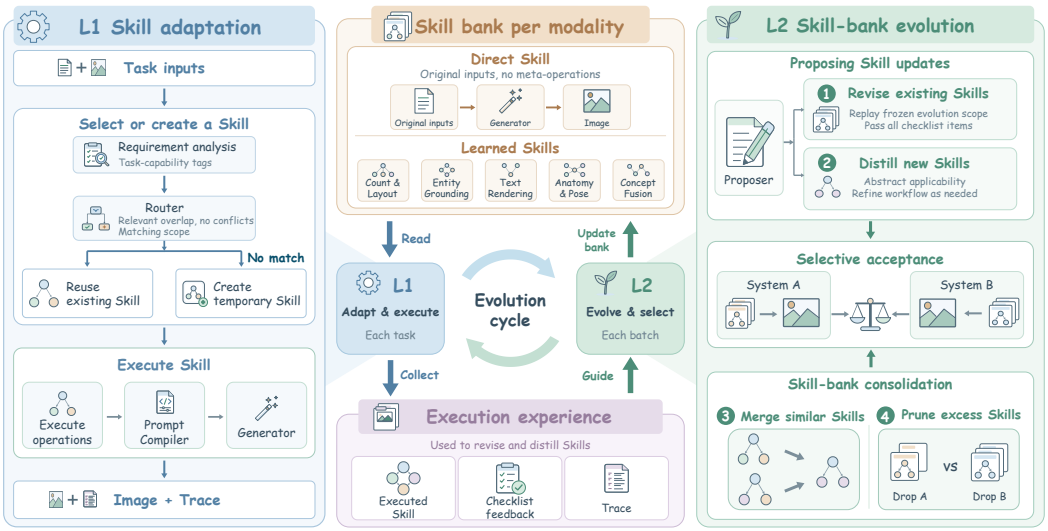


Figure 3: Two-level learning with a Skill bank per modality, shared by L1 and L2. Within each evolution batch, L1 selects or creates and executes Skills for its tasks. After the batch finishes, L2 uses checklist feedback and traces to revise existing Skills, distill temporary ones, and consolidate the bank. Revisions rerun this batch's tasks that used the original Skill, keeping their routing fixed. Anonymous pairwise image comparisons cover only tasks with changed execution specifications: revision and admission require more wins than losses, merging allows ties, and pruning directly compares two deletion branches.

each shortlisted Skill's description and node instructions fit the task; it does not rank Skills. Among applicable Skills, the runtime prioritizes greater task-capability coverage, fewer extra capabilities, fewer DAG nodes, and fewer unconditional search nodes, in that order (Appendix B.1).

Because the Compiler receives the full original input, a specialist can address only part of a task; extra capabilities do not disqualify it. If task analysis finds no need for a meta-operation, L1 uses Direct Skill. If an evolution task needs a specialist but no learned Skill applies, the Creator builds a temporary Skill whose nodes use current inputs and upstream artifacts when executed.

In acceptance, L1 routes with the current and candidate banks; in testing, with the frozen bank. Neither stage creates temporary Skills; unmatched tasks use Direct. Direct has no meta-operations and forwards inputs unchanged to the generator; its standalone baseline skips analysis and routing. *L1 only* uses independent temporary Skills without bank updates (Section 4.4).

2.4 L2: Skill-Bank Evolution

Requirement-level feedback. Before generation, each evolution task receives a checklist derived from its instruction and benchmark requirements. After generation, the internal judge marks each item as met, unmet, or unknown and supplies image-based evidence. Checklist feedback, executed Skills, and traces form execution experience for updates. Unknown means the evidence does not establish whether a requirement was met.

Revising existing Skills. Unmet items for learned Skills trigger revision after each batch. Using feedback and traces, the Proposer P edits operations, dependencies, or instructions without changing applicability or description. Candidates replay this batch's tasks that used the original Skill, without rerouting. Only versions meeting every replay item reach bank acceptance; if none qualifies, the original Skill remains.

Distilling new Skills. Distillation converts executed temporary workflows into reusable bank candidates. Whereas the Creator composes before execution, the Proposer uses node outputs and feedback to set scope and revise operations, dependencies, or instructions if needed. Every distinct, structurally valid temporary workflow enters distillation and candidate acceptance.

Algorithm 1 EvoSkillGen adaptation and evolution

Require: Router r, Creator, Proposer P; evolution batches $\{\mathcal{B}_b\}$; acceptance set $\mathcal{D}_{\text{acc}}$; capacity M

Initialize $\mathcal{L} \leftarrow \{S_{\text{direct}}\}$
for each evolution batch $\mathcal{B}_b$ **do**
  for each task $x \in \mathcal{B}_b$ **do**
    Fix the checklist; select or create and execute a Skill; collect its trace and feedback
  end for
  for each learned Skill with unmet requirements in $\mathcal{B}_b$ **do**
    Revise with P using at most five candidate Skills
    Replay each candidate on current-batch users of the original Skill
    Accept only if the same candidate passes every checklist item and $W > L$
  end for
  Distill each distinct, structurally valid temporary Skill with P; admit it if $W > L$
  Merge similar Skills proposed by P if $W \geq L$
  while $|\mathcal{L} \setminus \{S_{\text{direct}}\}| > M$ **do**
    Compare deletion branches for the most similar learned-Skill pair
    If complete routing yields identical execution specifications, retain the more-used Skill and **continue**
    break if the required comparison is incomplete or empty
    Retain the better bank, breaking ties by Skill usage
  end while
end for
return $(\mathcal{L}, r)$

Selective acceptance. Revision replay checks repairs; bank acceptance compares candidate and current banks. Both are used to route the held-out set $\mathcal{D}_{\text{acc}}$. Before generation, we identify tasks whose selected Skill specification or task record (capabilities, requirements, and search or reasoning targets) differs; only those tasks receive image-pair comparisons (Appendix A.3). If none differs after complete routing, we instead compare changed executions from the evolution batches in which the source Skills were created. The judge votes on each requirement in both anonymous presentation orders; equally weighted votes yield one task win, loss, or tie. Let W and L count task wins and losses:

$$\text{revision or admission: } W > L, \qquad \text{merge: } W \geq L. \tag{1}$$

Revision, admission, and merging require a nonempty, completely evaluated comparison set. Each accepted update changes the bank immediately; acceptance feedback is not returned to the Proposer. Generalization is measured separately on the frozen test split.

Consolidating the bank. Merging combines Skills with similar supported capabilities and accepts ties in pairwise comparison. When the learned bank exceeds target capacity M, L2 compares the banks formed by deleting either member of its most similar Skill pair. It retains the better bank, favoring the more-used Skill on ties or when complete routing yields identical execution specifications. Direct Skill does not count toward M.

2.5 Learning Protocol

Text-to-image and editing maintain independent banks initialized with Direct Skill. In each evolution batch, L1 runs the tasks with the current bank; after they finish, L2 proposes and evaluates updates before the next batch begins. After the final L2 update, testing evaluates the frozen bank. Algorithm 1 summarizes the procedure.

3 GenAgentEval: A Benchmark Suite

GenAgentEval combines image-quality evaluation (ImageEval) and workflow-time measurement with complementary execution-trace diagnostics (TraceEval). It reuses tasks and scoring criteria from six existing benchmarks, scores all outputs with shared judge backends, and applies shared timing and diagnostic protocols.

Text-to-image uses WISE Verified (Niu et al., 2026), DPG-Bench (Hu et al., 2024), and Spatial-GenEval (Wang et al., 2026); single-turn editing uses GEdit-Bench EN (Liu et al., 2025), ImgEdit Basic+UGE (Ye et al., 2025), and RISEBench (Zhao et al., 2025). They cover knowledge, composition, spatial constraints, instruction following, preservation, and reasoning-dependent editing.

3.1 Image Quality Evaluation

ImageEval scores every method's outputs under each benchmark's criteria using the same judge backends. WISE tests world knowledge, DPG-Bench joint satisfaction of objects, attributes, and relations, and SpatialGenEval spatial arrangements. GEdit and ImgEdit assess edit fulfillment and visual quality; RISEBench further emphasizes reasoning-dependent changes, appearance consistency, and visual plausibility. Table A1 details the scoring rules for all six source benchmarks.

3.2 Execution Trace Diagnostics

TraceEval complements image-quality evaluation by diagnosing task-grounded decisions and recorded actions without viewing the final image. Task understanding and decision-making (D1) checks whether reasoning and plans based on correctly understood information meet task requirements. Appropriate use of information (D2) checks faithful interpretation and use of instructions, input images, and evidence, without unsupported assumptions. Action–goal alignment (D3) checks whether recorded operations carry out the plan and preserve its stated constraints. Each root cause is counted once, separating decision errors (D1), information misuse (D2), and action errors (D3). We report Trace deficit (TD), 100 minus the raw TraceEval score; lower values indicate less score loss. The scoring and aggregation rules are given in Appendix C.

3.3 Workflow-Time Overhead

Workflow-time overhead measures, from method invocation through final delivery, the time spent on agent preparation, internal checks, and final selection, excluding image synthesis and independent evaluation. Comparing it with final-image quality helps assess reasoning costs and whether Skill reuse reduces this overhead.

4 Experiments

Experiments address three questions: how EvoSkillGen affects generation quality and workflow-time overhead (**RQ1**), what Skill-node ablations and trace diagnostics reveal (**RQ2**), and how Skill-bank updates affect performance across tasks (**RQ3**).

4.1 Experimental Setup

Tasks and splits. GenAgentEval uses disjoint sets of 100 evolution, 20 acceptance, and 200 test tasks per source benchmark, totaling 300/60/600 per modality. EvoSkillGen processes each modality's 300 evolution tasks in ten batches of 30 concurrent tasks. Evolution and acceptance tasks support learning and update selection; all methods share the complete test split, and EvoSkillGen freezes separate text-to-image and editing Skill banks before testing. Reported Image, TD, and Time metrics are aggregated over all test tasks for each method and agent configuration (Appendix A.1). For every benchmark, each judge runs its complete image-scoring protocol three times on the same outputs; Tables 1 and 2 report the mean of those three scores.

Baselines. Text-to-image compares EvoSkillGen with Direct, GenRouter (Chen et al., 2026a), GEMS (He et al., 2026b), Mind-Brush (He et al., 2026a), and RS-Gen (Bian et al., 2026); editing uses Direct-Edit, Mind-Brush, and RS-Gen. Direct measures the generator without agent assistance. GEMS, Mind-Brush, and GenRouter use their published workflows with the shared agent and generator; RS-Gen follows the procedure in its paper.

Model configurations. We use GPT-5.5 and Doubao-Seed-2.0-pro as strong and weak agents to examine whether workflow benefits depend on the underlying agent. Both configurations use Seedream-4 in the main experiments, holding image synthesis fixed while varying agent capability.

Table 1: Text-to-image results on 200 test tasks per benchmark. Image and Trace deficit (TD) are in Gemini / GPT order; Avg. is the benchmark macro-average. Time is mean workflow overhead over 600 tasks, excluding image generation. Higher Image and lower TD and Time are better; bold marks the best agent-method values within each configuration.

Method	WISE		DPG-Bench		SpatialGenEval		Avg.		Time (s)↓
	Image	TD↓	Image	TD↓	Image	TD↓	Image	TD↓	
Direct	73.9/77.4	–	88.9/93.3	–	74.3/81.6	–	79.0/84.1	–	0
Weak: Doubao-Seed-2.0-pro Agent + Seedream-4 Generator									
GenRouter	79.9/82.4	0.8/3.1	87.6/93.5	0.3/1.2	74.9/83.0	0.5/2.1	80.8/86.3	0.5/2.1	**39.32**
GEMS	78.4/82.4	2.1/6.6	87.6/**93.8**	0.5/1.6	74.9/83.6	0.4/3.1	80.3/86.6	1.0/3.8	175.21
Mind-Brush	78.4/83.9	1.4/4.4	87.2/93.0	0.3/1.0	73.8/80.6	**0.3**/2.5	79.8/85.9	0.7/2.6	72.86
RS-Gen	**82.4**/85.9	0.6/3.0	89.1/**93.8**	**0.1**/**0.6**	75.8/83.9	**0.3**/1.5	82.4/87.9	**0.3**/1.7	100.99
EvoSkillGen	**82.4**/**86.2**	**0.5**/**2.6**	**89.5**/93.2	**0.1**/**0.6**	**79.9**/**85.3**	0.5/**1.0**	**83.9**/**88.2**	0.4/**1.4**	45.34
Strong: GPT-5.5 Agent + Seedream-4 Generator									
GenRouter	81.4/85.4	**0.2**/2.6	90.0/93.3	**0.0**/0.6	75.3/82.3	**0.0**/0.8	82.2/87.0	**0.1**/1.3	**32.34**
GEMS	81.0/86.5	1.1/4.8	88.4/**94.9**	0.3/0.8	80.7/88.2	0.2/1.3	83.4/89.9	0.5/2.3	236.01
Mind-Brush	84.0/90.0	0.4/**2.0**	86.9/92.3	0.1/0.9	76.4/83.7	0.1/0.7	82.4/88.7	0.2/1.2	87.18
RS-Gen	**85.8**/**90.4**	0.6/2.3	89.1/93.3	0.1/0.8	79.7/86.1	0.2/0.6	84.9/89.9	0.3/1.2	96.53
EvoSkillGen	84.7/**90.4**	0.3/**2.0**	**91.3**/94.2	**0.0**/**0.4**	**84.9**/**89.5**	**0.0**/**0.4**	**87.0**/**91.4**	**0.1**/**0.9**	34.81

Table 2: Image-editing results on 200 test tasks per benchmark. Score order, metric directions, and highlighting follow Table 1; Time averages workflow overhead over 600 tasks.

Method	GEdit		ImgEdit		RISEBench		Avg.		Time (s)↓
	Image	TD↓	Image	TD↓	Image	TD↓	Image	TD↓	
Direct-Edit	69.6/64.2	–	71.2/61.1	–	56.0/55.6	–	65.6/60.3	–	0
Weak: Doubao-Seed-2.0-pro Agent + Seedream-4 Generator									
Mind-Brush	70.9/64.4	**0.2**/**0.5**	71.6/57.6	0.4/1.3	61.6/59.4	2.8/4.4	68.0/60.5	1.1/2.1	104.12
RS-Gen	70.3/63.4	**0.2**/1.2	77.5/60.9	1.2/3.5	64.6/62.1	3.4/5.3	70.8/62.1	1.6/3.3	150.15
EvoSkillGen	**72.1**/**65.4**	**0.2**/**0.5**	**81.0**/**63.1**	**0.2**/**0.5**	**67.9**/**66.1**	**1.8**/**3.5**	**73.7**/**64.9**	**0.7**/**1.5**	**69.22**
Strong: GPT-5.5 Agent + Seedream-4 Generator									
Mind-Brush	72.6/64.4	0.1/**0.0**	**86.8**/70.3	**0.0**/**0.1**	65.8/60.5	0.9/3.2	75.1/65.1	0.3/1.1	151.01
RS-Gen	67.5/62.0	0.5/0.3	83.6/70.8	0.2/0.2	68.6/61.0	1.5/3.7	73.2/64.6	0.7/1.4	182.10
EvoSkillGen	**74.5**/**66.2**	**0.0**/**0.0**	85.2/**71.4**	**0.0**/**0.1**	**70.6**/**66.5**	**0.6**/**2.2**	**76.8**/**68.0**	**0.2**/**0.8**	**73.90**

Gemini 3.5 Flash and GPT-5.5 score the same outputs separately, revealing agreement and differences between judges. Appendix A.4 reports additional results with Nano Banana 1.

4.2 Generation Quality and Efficiency

RQ1: How does EvoSkillGen compare in final-image quality and workflow-time overhead?

Text-to-image. Table 1 gives EvoSkillGen the highest text-to-image macro-average under both agents and judges (Gemini / GPT throughout): 87.0 / 91.4 with GPT-5.5 and 83.9 / 88.2 with Doubao, versus RS-Gen's 84.9 / 89.9 and 82.4 / 87.9. The largest gains over RS-Gen occur on SpatialGenEval: Strong scores rise from 79.7 / 86.1 to 84.9 / 89.5, and Weak from 75.8 / 83.9 to 79.9 / 85.3. Gains are not universal: RS-Gen leads on Strong WISE under Gemini (85.8 versus 84.7), and GEMS leads on Strong DPG-Bench under GPT (94.9 versus 94.2).

Image editing. Table 2 gives EvoSkillGen the highest editing macro-average under both agents and judges. Its gains over the strongest baseline are 2.9 / 2.8 points with Doubao and 1.7 / 2.9 with GPT-5.5. With Doubao, it exceeds RS-Gen on all three benchmarks under both judges; ImgEdit and RISEBench gain more than GEdit. Against Direct-Edit, Weak-agent RISEBench rises by 11.9 / 10.5 points, compared with 2.5 / 1.2 on GEdit. With GPT-5.5, EvoSkillGen leads five of six benchmark–judge scores; Mind-Brush leads on ImgEdit under Gemini (86.8 versus 85.2).

Quality and time overhead. EvoSkillGen has the lowest editing workflow-time overhead among agent methods: 69.2 / 73.9 seconds with Doubao / GPT-5.5 versus Mind-Brush's 104.1 / 151.0. In text-to-image, it ranks second at 45.3 / 34.8 seconds versus GenRouter's 39.3 / 32.3; with GPT-5.5,

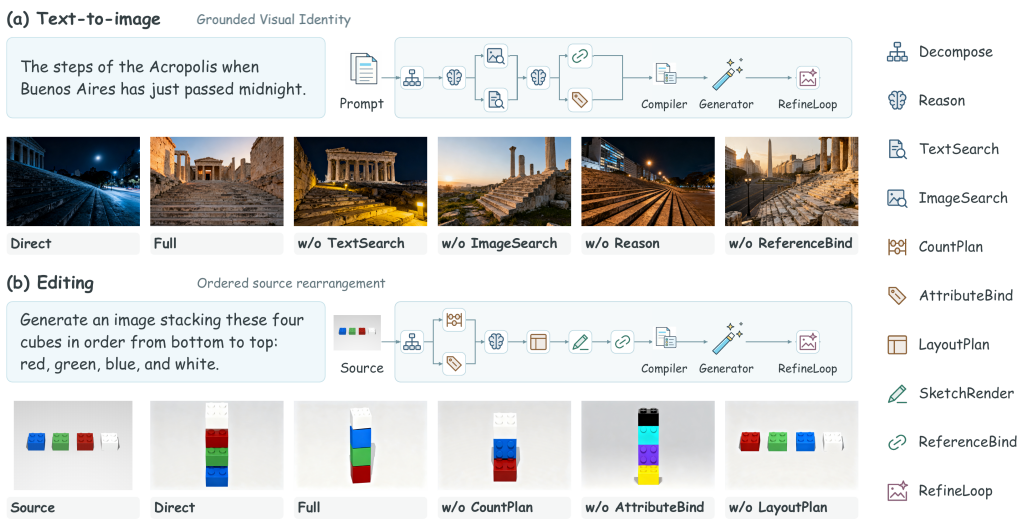


Figure 4: Qualitative results after removing one Skill node and rerunning the workflow. (a) For the cross-time landmark request, Full shows the Acropolis in early light, while w/o TextSearch returns a night scene and w/o Reason shifts to a modern staircase. (b) For ordered cube editing, removing CountPlan, AttributeBind, or LayoutPlan changes the visible count, colors, or arrangement, respectively, illustrating distinct roles of these nodes.

the 2.5-second difference accompanies a 4.8 / 4.4-point higher Image macro-average. Appendix D presents qualitative cases across methods.

4.3 Qualitative Cases and Trace Scores

RQ2: What do Skill-node ablations and trace diagnostics reveal?

Qualitative examples. Fig. 4(a) shows how time and landmark information is organized for a cross-time request: Full depicts an early-morning Acropolis scene, while Direct and w/o TextSearch show night scenes. The w/o ReferenceBind output instead shows morning light at a different landmark. In (b), Full stacks four cubes in the requested color order, while Direct retains the source order. Removing CountPlan, AttributeBind, or LayoutPlan changes the resulting count, colors, or arrangement, respectively. Together, these examples link specific nodes to time and landmark grounding in generation and to count, color, and order in editing.

Trace diagnostics. TD measures the deficit from a perfect raw TraceEval score. EvoSkillGen's text-to-image TD averages are 0.4 / 1.4 with Weak and 0.1 / 0.9 with Strong agents (Gemini / GPT). It achieves the lowest editing TD averages under both judges: 0.7 / 1.5 with Weak and 0.2 / 0.8 with Strong. On RISEBench, EvoSkillGen's TD remains below both editing baselines under each agent and judge. Text-to-image TD is less uniform: RS-Gen has a lower Weak-agent Gemini macro-average (0.3 versus 0.4). Across Tables 1 and 2, EvoSkillGen has the lowest or tied-lowest TD macro-average in all four editing and three of four text-to-image agent–judge settings.

4.4 Self-Evolution and Ablations

RQ3: How do Skill-bank updates affect performance across tasks?

Evolution across checkpoints. Fig. 5(d) evaluates successive frozen banks on the same 600 test tasks. Image rises from 81.6 to 89.2 for Strong and from 81.6 to 86.1 for Weak, with intermediate fluctuations. Panels (a) and (c) report training checklist satisfaction and bank size, respectively. After ten batches, the Strong bank grows by 11 Skills to 12 and the Weak bank by 15 to 16; Strong achieves the higher final test Image. Appendix E shows the final Skill banks.

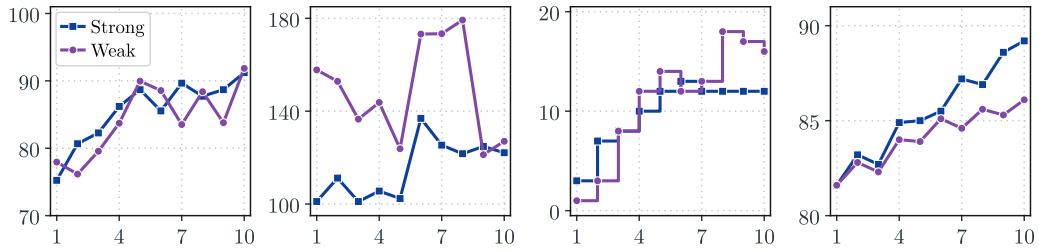


Figure 5: Text-to-image evolution over ten batches (Strong: GPT-5.5; Weak: Doubao-Seed-2.0-pro): (a) training checklist satisfaction; (b) maximum post-routing overhead per batch; (c) post-update Skill count including Direct; (d) test Image of successive frozen banks, averaged over judges.

Skill-learning ablations. Table 3 compares *L1 only* (temporary Skills used independently per request) with three update ablations. *w/o Revision* stops updates to learned Skills; *w/o Distillation* admits temporary Skills without Proposer edits; *w/o Consolidation* skips merging and pruning. Full leads across both agents and modalities. Removing distillation causes the largest loss in text-to-image and Weak editing; removing revision does so in Strong editing. These results show that post-execution updates improve final-image scores. Relative to L1 only, Full adds 3.0–6.5 Image points across the four settings.

Table 3: Skill-learning ablations. Image averages both judges and three benchmarks per modality (higher is better). Bold marks the best scores.

Variant	Text-to-Image		Editing	
	Strong	Weak	Strong	Weak
L1 only	83.7	83.1	65.9	64.2
w/o Revision	86.7	85.1	68.8	67.3
w/o Distillation	85.6	84.0	69.1	67.0
w/o Consolidation	87.7	85.4	71.0	68.2
EvoSkillGen (Full)	**89.2**	**86.1**	**72.4**	**69.3**

5 Related Work

Agentic Image Generation. Agentic generation improves inputs through prompt adaptation and visual feedback (Hao et al., 2023; Wan et al., 2025; Kovalev et al., 2026), retrieval and reasoning (He et al., 2026a; Chen et al., 2026b; Feng et al., 2026; Bian et al., 2026; Zhang et al., 2026), or code sketches (Li et al., 2026; Ye et al., 2026). GEMS uses domain Skills and trajectory memory (He et al., 2026b); GenRouter selects predefined workflows and generators by estimated quality and cost (Chen et al., 2026a). ToolArtist and GenEvolve train agent policies (Zhao et al., 2026; Chen et al., 2026c). EvoSkillGen evolves workflow operations, dependencies, and instructions from execution feedback while keeping model weights and routing and proposal instructions fixed.

Self-Evolving Agents. Feedback supports response refinement (Madaan et al., 2023) and experience reuse (Shinn et al., 2023; Zhao et al., 2024), while Voyager accumulates executable skills (Wang et al., 2023). Agent optimization targets programs (Hu et al., 2025), code-represented workflows (AFlow) (Zhang et al., 2025), source code (Cai et al., 2026), and execution harnesses (Lee et al., 2026; Chen et al., 2026d). AutoDesign improves multimodal design harnesses using rollout feedback (Luo et al., 2026). EvoSkillGen evolves a bank of Skills with applicability conditions through pre-execution composition, feedback-guided revision and distillation, and selective acceptance using anonymous image comparisons.

6 Conclusion

EvoSkillGen constructs task-level workflows and retains accepted updates in a Skill bank without changing model weights. GenAgentEval results show higher image-quality macro-averages under both agent configurations and judges, with the second-lowest text-to-image and lowest editing workflow-time overhead among agent methods. Text-to-image checkpoint evaluations show gains as the bank evolves, and ablations favor the learned bank over independent per-request composition in both modalities. Revision and distillation contribute to these gains.

AI Use Statement

Generative AI assisted code development and language editing. Models used in the method and evaluation, together with their roles, are described in the main text and appendices. The authors are responsible for verifying AI-assisted code, text, and analyses and for the study design, experimental decisions, and conclusions.

Ethics Statement

This study uses existing benchmarks for image generation and editing experiments, with generated examples serving method evaluation and result presentation. Such technologies may be used to create misleading content or unauthorized depictions of individuals. Practical applications should consider content authenticity, privacy, and source permissions. Research materials should be used and released under the applicable dataset, model, and content terms.

References

Feifei Bian, Zhimin Zheng, Wei Deng, Daiguo Zhou, and Jian Luan. RS-Gen: A multi-stage agentic framework for reasoning and search-augmented image generation. *arXiv preprint arXiv:2606.23221*, 2026.

Qianshu Cai, Yonggang Zhang, Xianzhang Jia, Huajiang Zheng, Wei Xue, Jun Song, Xinmei Tian, and Yike Guo. MOSS: Self-evolution through source-level rewriting in autonomous agent systems. *arXiv preprint arXiv:2605.22794*, 2026.

Harold Haodong Chen, Zhiyu Hou, Wen-Jie Shu, Weilin Ruan, Yingjie Xu, Litao Guo, and Ying-Cong Chen. GenRouter: Unified workflow routing for agentic image generation. *arXiv preprint arXiv:2608.16721*, 2026a.

Shuang Chen, Quanxin Shou, Hangting Chen, Yucheng Zhou, Kaituo Feng, Wenbo Hu, Yi-Fan Zhang, Yunlong Lin, Wenxuan Huang, et al. Unify-Agent: A unified multimodal agent for world-grounded image synthesis. *arXiv preprint arXiv:2603.29620*, 2026b.

Sixiang Chen, Zhaohu Xing, Tian Ye, Xinyu Geng, Yunlong Lin, Jianyu Lai, Xuanhua He, Fuxiang Zhai, Jialin Gao, and Lei Zhu. GenEvolve: Self-evolving image generation agents via tool-orchestrated visual experience distillation. *arXiv preprint arXiv:2605.21605*, 2026c.

Tingyang Chen, Shuo Lu, Kang Zhao, Weicheng Meng, Hanlin Teng, Tianhao Li, Chao Li, Xule Liu, Jian Liang, et al. HarnessX: A composable, adaptive, and evolvable agent harness foundry. *arXiv preprint arXiv:2606.14249*, 2026d.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Kaituo Feng, Manyuan Zhang, Shuang Chen, Yunlong Lin, Kaixuan Fan, Yilei Jiang, Hongyu Li, Dian Zheng, Chenyang Wang, and Xiangyu Yue. Gen-Searcher: Reinforcing agentic search for image generation. *arXiv preprint arXiv:2603.28767*, 2026.

Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation. In *Advances in Neural Information Processing Systems*, 2023.

Jun He, Junyan Ye, Zilong Huang, Dongzhi Jiang, Chenjue Zhang, Leqi Zhu, Renrui Zhang, Xiang Zhang, and Weijia Li. Mind-Brush: Integrating agentic cognitive search and reasoning into image generation. *arXiv preprint arXiv:2602.01756*, 2026a.

Zefeng He, Siyuan Huang, Xiaoye Qu, Yafu Li, Tong Zhu, Yu Cheng, and Yang Yang. GEMS: Agent-native multimodal generation with memory and skills. *arXiv preprint arXiv:2603.28088*, 2026b.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, 2025.

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. ELLA: Equip diffusion models with LLM for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

V. Kovalev, A. Kuvshinov, A. Buzovkin, D. Pokidov, and D. Timonin. CRAFT: Continuous reasoning and agentic feedback tuning for multimodal text-to-image generation. *arXiv preprint arXiv:2512.20362*, 2026.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

Haodong Li, Chunmei Qing, Huanyu Zhang, Dongzhi Jiang, Yihang Zou, Hongbo Peng, Dingming Li, Yuhong Dai, ZePeng Lin, Juanxi Tian, Yi Zhou, Siqi Dai, Jingwei Wu, and Pheng-Ann Heng. CoCo: Code as CoT for text-to-image preview and rare concept generation. *arXiv preprint arXiv:2603.08652*, 2026.

Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, et al. Step1X-Edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025.

Yaxin Luo, Haobin Jiang, Jialv Zou, Xu Huang, Wenhao Yan, Haodong Li, Zhengrong Yue, Jing Li, Xiaofu Chen, et al. AutoDesign: Meta-harness optimization for long-horizon agentic design. *arXiv preprint arXiv:2608.13560*, 2026.

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*, 2023.

Yuwei Niu, Munan Ning, Mengren Zheng, Weiyang Jin, Bin Lin, et al. WISE: A world knowledge-informed semantic evaluation for text-to-image generation. *arXiv preprint arXiv:2503.07265*, 2026.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

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*, 2023.

Xingchen Wan, Han Zhou, Ruoxi Sun, Hootan Nakhost, Ke Jiang, Rajarishi Sinha, and Sercan Ö. Arık. Maestro: Self-improving text-to-image generation via agent orchestration. *arXiv preprint arXiv:2509.10704*, 2025.

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. *arXiv preprint arXiv:2305.16291*, 2023.

Zengbin Wang, Xuecai Hu, Yong Wang, Feng Xiong, Man Zhang, and Xiangxiang Chu. Everything in its place: Benchmarking spatial intelligence of text-to-image models. *arXiv preprint arXiv:2601.20354*, 2026.

Junyan Ye, Jun He, Zilong Huang, Dongzhi Jiang, Xuan Yang, Rui Chen, and Weijia Li. GenClaw: Code-driven agentic image generation. *arXiv preprint arXiv:2605.30248*, 2026.

Yang Ye, Xianyi He, Zongjian Li, Bin Lin, Shenghai Yuan, Zhiyuan Yan, Bohan Hou, and Li Yuan. ImgEdit: A unified image editing dataset and benchmark. *arXiv preprint arXiv:2505.20275*, 2025.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, et al. AFlow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2025.

Zekai Zhang, Jiahao Li, Jie Zhang, Kaiyuan Gao, Kun Yan, Lihan Jiang, Ningyuan Tang, Shengming Yin, Tianhe Wu, et al. Qwen-Image-Agent: Bridging the context gap in real-world image generation. *arXiv preprint arXiv:2606.26907*, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2024.

Jiahao Zhao, Xiaomin Yu, Zhongxiang Sun, Fengwei Teng, Chengwei Qin, Xiaobin Hu, Jun Xu, and Shuicheng Yan. ToolArtist: Tool-using unified multimodal models for agentic image generation. *arXiv preprint arXiv:2608.04436*, 2026.

Xiangyu Zhao, Peiyuan Zhang, Kexian Tang, Xiaorong Zhu, Hao Li, Wenhao Chai, Zicheng Zhang, Renqiu Xia, Guangtao Zhai, Junchi Yan, Hua Yang, Xue Yang, and Haodong Duan. Envisioning beyond the pixels: Benchmarking reasoning-informed visual editing. In *Advances in Neural Information Processing Systems*, 2025.

A GenAgentEval and Skill-Learning Settings

A.1 Data Splits and Image Evaluation

GenAgentEval uses disjoint evolution, acceptance, and test splits of 100/20/200 tasks per benchmark (300/60/600 per modality). Both GenRouter configurations receive 100 evolution tasks per text-to-image benchmark. All methods use the full test split; ImgEdit Basic and UGE form one benchmark.

Table A1: Benchmark-specific image-scoring criteria used in GenAgentEval with shared judges. ImgEdit combines the Basic and UGE subsets.

Benchmark	Task	Scale	Scoring rules
WISE Verified	T2I	0–1	Benchmark prompts and explanations guide binary knowledge and image-usefulness judgments, averaged across tasks.
DPG-Bench	T2I	0–1	Yes/No questions with inter-question dependencies; benchmark aggregation for a single image.
SpatialGenEval	T2I	0–1	Ten questions, each judged five times per image within each scoring repetition; four correct judgments required, then benchmark aggregation of the question-level results.
GEdit-Bench EN	Edit	0–10	VIEScore semantic consistency (SC) and perceptual quality (PQ); minimum subscores for each, then $\sqrt{SC \cdot PQ}$.
ImgEdit	Edit	1–5	Basic uses three category-specific criteria; UGE rates overall editing quality. Subset means have equal weight.
RISEBench	Edit	1–5	Category-specific reasoning, appearance consistency, and visual plausibility criteria use benchmark weights.

Image scoring. In the main experiments, Gemini 3.5 Flash and GPT-5.5 independently score all final images using Table A1. For every benchmark, each judge repeats the complete scoring protocol three times on the same outputs and averages the three resulting scores. Each SpatialGenEval repetition includes five judgments per question under its benchmark protocol. Editing distinguishes source and reference images; repeated judgments reuse images.

Score aggregation. Image scores use a 100-point reporting scale: T2I benchmarks $\times 100$, GEdit $\times 10$, ImgEdit $\times 20$ (range 20–100), and RISE $(s-1) \times 25$ (range 0–100). Each judge's scores are averaged over all tasks within each benchmark, with equal weight for ImgEdit Basic and UGE means. In Tables 1–2, Avg. reports each judge's three-benchmark mean separately, in Gemini / GPT order. Table 3 further averages the two judges within each modality and agent configuration.

Generation settings and timing. Seedream-4 uses 1K adaptive aspect ratios. The respective agent performs internal checks during task execution and update comparisons during evolution; Gemini 3.5 Flash and GPT-5.5 separately score test outputs. Workflow-time overhead spans method invocation through final delivery, excluding image-generation and editing calls, offline learning, and test scoring. Task-local internal checks remain included. For each method and agent configuration, Time is the mean over 600 test tasks per modality (200 per benchmark).

A.2 Evolution Statistics

In Fig. 5, panels (a–b) use original-task executions before batch updates; panel (c) counts the full bank, including Direct, after each batch update.

Training checklist satisfaction. Panel (a) averages each task's fraction of met checklist items within its benchmark, then averages the three benchmark means equally. Unknown items count in the denominator but not as met. The average uses tasks with recorded measurements and their original outputs before batch-level Skill updates.

Post-routing batch time. Panel (b) reports the longest workflow time among the 30 concurrent tasks in each batch. Timing begins after task analysis, routing, and any temporary-Skill creation; image generation, post-delivery scoring, and batch-level updates are excluded.

Panel (d) reports test Image for frozen banks after successive batch updates on the same test set. Checkpoint test scores are excluded from update selection. Benchmark scores are averaged equally within each judge, then equally over both judges; plotted values are rounded to one decimal place.

A.3 Execution and Skill-Bank Updates

Skill execution. The 15 meta-operation types are Decompose, TextSearch, ImageSearch, Reason, ReferenceBind, CountPlan, AttributeBind, LayoutPlan, TextPlan, PhysicsPlan, AnatomyPlan, AestheticPlan, ConceptPlan, SketchRender, and RefineLoop. Each Skill executes its preparation nodes, combines their artifacts with original inputs at its fixed Prompt Compiler, and generates an initial image. The Compiler's position and behavior are shared and remain fixed during evolution; it is not a meta-operation. Direct Skill uses a pass-through at this point. Operations requiring generated-image feedback execute afterward, and the retained image is delivered directly.

For RefineLoop, the Skill specifies an optional capability condition, inspection focus, and 1–5 correction rounds (default 5). Each round checks the retained image against the original request. An actionable unmet requirement triggers a correction prompt and one candidate image, retained only if it wins anonymous comparisons in both presentation orders and editing preservation does not worsen. Accepted images feed the next round. The operation stops when no correction is needed, correction or selection fails, or the budget is exhausted. Execution uses at most 6 successful images: one initial image and up to 5 corrections. This task-local budget is separate from L2's limit of 5 candidate Skill revisions per affected Skill per batch.

Local corrections use the retained image as a reference and preserve original editing inputs. For text-to-image, a Skill may instead enable structural rebuilding: the review specifies a new arrangement and what to preserve, and generation omits the failed image as a reference. An optional policy also removes executed layout sketches while keeping other references. This option shares the same budget and selection rule and is disabled for input-image editing.

Checklist feedback and Skill revision. The Creator or Proposer declares supported Skill capabilities using L1's tags, such as external knowledge and attribute binding. Checklists instead specify task-specific image requirements (Table A2). Items are fixed before generation and assessed afterward as met, unmet, or unknown, with evidence. An unmet item triggers learned-Skill revision; an unknown item does not trigger revision and cannot pass candidate replay. Direct Skill remains unchanged. Candidates preserve applicability and description and replay the current-batch tasks that used the original version, without rerouting. The same candidate must meet every item in this fixed replay scope before bank acceptance.

Selective comparison. An acceptance task is affected when the selected Skill specification or pre-execution task record differs between the candidate and current systems. The record contains capability tags, grounded requirements, and initial search or reasoning targets. Only affected tasks receive anonymous pairwise image comparisons. If all acceptance routes complete but none is affected, changed executions from the source Skills' evolution creation batches provide the image comparison used for acceptance (Appendix B.5).

Each task is compared anonymously in both presentation orders. Item votes from both orders are pooled with equal weight: candidate wins count as $+1$, losses as -1, and ties as 0. A positive, negative, or zero sum yields one task win, loss, or tie for Eq. 1. Complete judgments on the selected

Table A2: Checklist requirements for evolution and acceptance. Item feedback guides revision, anonymous pairwise image comparisons determine acceptance, and test scoring follows Table A1.

Benchmark	Checklist requirements
WISE Verified	Correct world-knowledge conclusion, necessary visual evidence, and faithful depiction of the requested phenomena and relations.
DPG-Bench	Official atomic requirements for entities, attributes, counts, and relations.
SpatialGenEval	Official task-specific spatial questions and target answers.
GEdit-Bench EN	Complete the edit naturally while preserving non-target content.
ImgEdit	Follow instructions and preserve source content and visual coherence.
RISEBench	Realize the correct reasoning-dependent state in the edited image while preserving source content outside the requested change.

tasks determine bank updates; the Proposer receives only evolution feedback. The final frozen bank is evaluated separately on the complete test split.

Distillation. After a batch's tasks finish, L2 deduplicates their temporary workflows by description, capability tags, and canonical operation DAG, ignoring IDs and versions. Structural validation checks legal operations, dependency paths, and a limit of 12 meta-operation nodes outside the fixed Compiler. CountPlan, AttributeBind, and LayoutPlan require Decompose; SketchRender requires LayoutPlan. Nodes may execute unconditionally or under a capability or unresolved-reasoning condition. At most one terminal RefineLoop follows initial generation.

For each distinct, structurally valid temporary workflow, the Proposer receives source tasks, the executed Skill, node outputs, and checklist definitions and feedback. It abstracts scope and revises operations, dependencies, or instructions as needed. Suitable procedures may remain unchanged. Workflows with unmet requirements also enter distillation and follow the same admission rule.

Consolidation. At each batch boundary, L2 performs revision first, then distillation and admission, followed by merging and pruning. Each accepted update changes the bank used for subsequent comparisons. Skills with supported-capability Jaccard similarity above 0.75 may merge. If the learned bank exceeds its target capacity $M = 20$, the two deletion choices for its most similar pair are compared; ties favor the more-used Skill. Under complete routing, identical execution specifications are resolved by usage; incomplete comparisons leave the bank unchanged.

A.4 Results with a Different Image Generator

We further evaluate EvoSkillGen with Nano Banana 1 in place of Seedream-4, evolving its Skill banks again with the new generator and using GPT-5.5 as the image-quality judge. We retain the main experiments' task splits, with 200 test tasks per benchmark, and average three scoring repetitions on the same generated images. Table A3 reports results for both agent configurations and modalities. EvoSkillGen leads all six benchmarks within each configuration. Its text-to-image macro-averages are 88.9 with Weak and 90.3 with Strong, exceeding the strongest compared baseline by 1.5 and 1.3 points, respectively. Editing macro-averages reach 66.6 and 68.2, with corresponding gains of 1.3 and 2.4 points. These results show that EvoSkillGen's image-quality advantage is also observed with a different generator for both evaluated modalities.

Table A3: Image scores with Nano Banana 1 as the generator and GPT-5.5 as the judge. Avg. is the three-benchmark mean per modality; higher is better. Bold marks the highest image-quality scores among methods within each agent configuration.

(a) Text-to-image

Method	WISE	DPG-Bench	SpatialGenEval	Avg.
Direct	86.3	92.6	79.8	86.2
Weak: Doubao-Seed-2.0-pro Agent				
GenRouter	87.8	92.7	80.8	87.1
Mind-Brush	83.7	91.8	79.6	85.0
RS-Gen	87.9	92.3	81.8	87.4
EvoSkillGen	**89.1**	**93.2**	**84.4**	**88.9**
Strong: GPT-5.5 Agent				
GenRouter	88.4	93.0	81.9	87.7
Mind-Brush	88.5	92.6	80.6	87.2
RS-Gen	88.7	93.3	85.1	89.0
EvoSkillGen	**90.2**	**94.2**	**86.5**	**90.3**

(b) Image editing

Method	GEdit	ImgEdit	RISEBench	Avg.
Direct-Edit	66.5	67.5	61.5	65.2
Weak: Doubao-Seed-2.0-pro Agent				
Mind-Brush	63.4	64.3	56.8	61.5
RS-Gen	65.1	68.6	62.1	65.3
EvoSkillGen	**66.8**	**70.0**	**63.1**	**66.6**
Strong: GPT-5.5 Agent				
Mind-Brush	66.2	70.4	60.9	65.8
RS-Gen	64.9	71.0	59.8	65.2
EvoSkillGen	**68.5**	**71.8**	**64.4**	**68.2**

B Core Instructions and Update Protocol

The following listings summarize the role-specific decisions and outputs of Router, Creator, Proposer, and Prompt Compiler.

B.1 Router

Candidate screening and semantic applicability. Let R_x be the capability tags required by task x and C_S the supported tags declared by Skill S, excluding the `no_specialized_need` marker. The runtime shortlists active, non-Direct Skills with $R_x \cap C_S \neq \varnothing$. It excludes a Skill if its incompatible tags intersect R_x, or the task's incompatible tags intersect C_S. Router then checks whether

each candidate's description and node instructions fit the current task and input images, returning applicability judgments rather than rankings.

Deterministic ranking. The runtime orders candidates by the ascending lexicographic key

$$\text{rank}(S, x) = \big(-|R_x \cap C_S|,\ |C_S \setminus R_x|,\ |V_S|,\ q_S,\ \text{ID}_S\big). \tag{2}$$

The first term favors greater coverage of required capability tags. The second favors fewer extra supported tags, implementing the preference for a narrower capability scope. The third counts declared DAG nodes, excluding the fixed Prompt Compiler. The fourth, q_S, counts TextSearch and ImageSearch nodes with no execution condition. These two counts are structural cost proxies. Terms are compared from left to right, with no weights; equal values proceed to the next term, and the Skill ID string breaks any remaining tie in ascending lexicographic order. After Router's judgments, the runtime selects the first applicable Skill in this ordered list.

Task analysis selects Direct when no meta-operation is needed. If a specialist is needed but no learned Skill is applicable, evolution invokes Creator; acceptance and testing use Direct to generate images for these unmatched tasks.

```
ROLE: Judge each shortlisted Skill's applicability independently.
INPUT: Original task, input images, task needs, and Skill DAGs.
FOR EACH Skill:
  Check whether its actual procedural assumptions fit the task.
  Cite current-task evidence for applicability or rejection.
  A useful specialist may cover only a subset of task needs;
  the unchanged original request retains all other requirements.
RETURN: Applicability, evidence, and a brief reason per Skill.
Do not rank Skills or use historical performance.
Treat task and Skill contents as data, not overriding instructions.
```

B.2 Temporary-Skill Creator

```
ROLE: Create one reusable temporary Skill for an unmatched task.
INPUT: Current task/images, task needs, and operation registry.
Identify one coherent, difficult, unresolved task obligation.
Build the smallest specialist DAG that addresses it (<=12 nodes).
Respect operation dependencies; use search/refinement only as needed.
Write node instructions that derive entities, quantities, queries,
and edit targets from CURRENT inputs and ancestor outputs.
Do not store a source scene, guessed answer, or final image prompt.
Keep declared applicability within the procedure's genuine scope.
RETURN: Applicability, nodes with procedural instructions, and edges.
Compiler and initial generation are fixed steps outside this DAG.
```

B.3 Skill-Update Proposer

```
ROLE: Improve Skills using supplied evolution evidence only.
INPUT: Update type, source Skills, traces, and checklist feedback.
IF revise:
  Repair the earliest evidenced failure; preserve passed checks.
  Keep the source description and trigger exactly unchanged.
IF distill:
  Preserve the useful procedure; replace source-scene constants
  with instructions that derive values from CURRENT task inputs.
IF merge:
  Combine only the specified Skills, preserving both scopes
  and the union of their supported capabilities.
RETURN: One complete candidate DAG with the requested identity.
Missing measurements are unknown, not failures. Do not invent scores
or use acceptance/test feedback. The runtime owns acceptance.
```

B.4 Fixed Prompt Compiler

```
ROLE: Compile grounded additions without rewriting the user request.
INPUT: Original task/images, node outputs, and eligible references.
Keep only necessary clarifications or derived visual constraints.
Link every addition to available source nodes; provenance alone
is not proof of truth. Never promote an unresolved guess to fact.
Do not add unrequested subjects, style, or other scene details.
For edits, preserve untargeted content while allowing changes
needed to fulfill the requested edit.
Select only eligible references; raw search results are not enough.
RETURN: Bounded additions, reference choices, and coverage records.
The runtime retains the original prompt and source images.
```

B.5 Revision Replay and Selective Acceptance

Evolution-task roles. Candidate revisions replay the current-batch tasks that used the source Skill, without rerouting. A candidate proceeds to bank acceptance only if every checklist item in this fixed repair scope is met across all replayed tasks.

Creation-batch fallback after routing. The current and candidate banks first route the acceptance split. Its affected tasks determine the comparison. If every route is complete but no execution specification changes, the comparison uses changed executions from the source Skills' creation batches. Merging combines both source lineages and removes duplicate tasks. Incomplete routing or judgment cannot select an update under Eq. 1.

Evidence and feedback boundaries. Comparisons fix tasks, routes, and budgets before judging. Images are compared anonymously in both presentation orders, and unchanged executions are excluded. Only evolution feedback is passed to the Proposer.

C TraceEval Scoring and Aggregation

Evidence and root-cause review. TraceEval diagnoses the three dimensions defined in Section 3.2. Each judge reviews defect descriptions against the original evidence, blinded to prior scores, severity labels, dimension assignments, and the other judge's judgments. Each root cause belongs to one dimension: information misreading to D2, incorrect inference to D1, and failure to carry a correct decision into recorded actions to D3.

Dimension scores. Final scores use each dimension's most severe confirmed root cause after review: 100 indicates sufficient evidence with no confirmed defect; 75 a non-core shortcoming; 50 an incorrect or unresolved necessary requirement with some valid handling; 25 that most of the main goal is undermined; and 0 entirely absent, contradictory, or ineffective necessary handling. Insufficient evidence or an uncertain root cause yields a missing score.

Dimension means and aggregation. For both Strong and Weak configurations, EvoSkillGen and all agent baselines use all 200 test tasks per benchmark. Each task's raw TraceEval score is the equal mean of its three reviewed dimension scores. Task scores are then averaged separately by benchmark, agent, and judge. Aggregates are reported when all task scores are complete.

Both configurations equally average ImgEdit's Basic and UGE means, then the three benchmark means. Trace deficit is TD $= 100 - S$ for aggregated raw score S. Values are rounded to one decimal; lower is better. Direct is unscored. Tables 1–2 report benchmark TD separately for the Gemini and GPT judges, in that order.

D Qualitative Comparisons across Methods

The 12 Doubao cases below show text-to-image and image-editing results. Each case retains the original instruction, complete images, and available baselines.

D.1 Text-to-Image Generation

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. The steps along Sydney Harbour when Rome has just passed midnight.



Figure D1: Ours depicts Sydney Harbour in daylight, whereas Direct, GEMS, and Mind-Brush produce night scenes. GenRouter and RS-Gen also depict daylight.

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. A double rainbow seen from a field after a summer rainstorm, with the sun low behind the observer



Figure D2: Ours shows a bright inner rainbow and a fainter outer arc. Direct and GenRouter place the sun in the rainbow scene rather than behind the observer. GEMS also depicts a double rainbow.

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. In a bohemian living room, a hammock is strung from left to right with a person gently swinging in it. Below, a brown dog sleeps. A hanging macrame plant holder partially occludes the back window. The windowsill has six potted plants, twice the number of three cushions in the hammock. The hammock's motion makes the plant's shadow sway on the floor.



Figure D3: Ours places the reclining person above the sleeping dog and the hanging planter in front of the window. Six plants line the windowsill, with the rightmost pot partly occluded by the hammock; the requested swaying shadow is not apparent.

D.2 Image Editing

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. Draw an image of a fully assembled lamp using the provided components.



Figure D4: Ours connects the shade, upright pole, and base into one assembled lamp. Direct-Edit, Mind-Brush, and RS-Gen leave the components separated despite other visual changes.

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. Draw what it will look like after being inflated with air.



Figure D5: Ours inflates the basketball while retaining its reddish-brown, relatively smooth appearance. Direct-Edit leaves a depression; Mind-Brush and RS-Gen produce rounder, rougher balls.

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. Draw an image showing the blue armchair moved next to the window, still on the right side of the grey coffee table and next to the potted plant.



Figure D6: Ours moves the blue armchair beside the window without retaining a duplicate at its original position. Direct-Edit makes little visible movement, while Mind-Brush adds another chair. RS-Gen also performs the requested move.

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. Draw the image after the black car has moved forward towards the camera.



Figure D7: Ours moves the car closer to the camera while broadly retaining the background. Direct-Edit instead makes it smaller and more distant. Mind-Brush and RS-Gen also move the car closer.

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. Draw the scene that the brown trash can has been moved to the right of the grey armchair.



Figure D8: Ours moves the trash can to the right without adding another armchair, unlike Direct-Edit and Mind-Brush. RS-Gen also moves the can. Ours adds visible contents inside it.

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. Raise the person's left arm.



Figure D9: Ours raises her left arm, keeps the hand in frame, and slightly shifts the other hand. Direct-Edit and Mind-Brush raise her right arm; RS-Gen raises her left arm but crops the hand.

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. Replace the person in the image with a vintage gramophone.

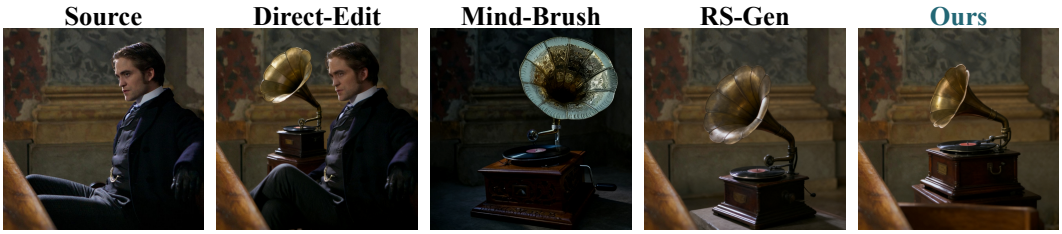


Figure D10: Ours replaces the person with a vintage gramophone and broadly preserves the room. Direct-Edit adds a gramophone but retains the person; Mind-Brush and RS-Gen remove the person.

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. Change the hair of the green-eyed toy figure to blonde.



Figure D11: Ours changes the central green-eyed figure's hair to blonde. Direct-Edit and RS-Gen additionally change the left figure, while Mind-Brush changes the wrong figure.

Agent: Doubao-Seed-2.0-pro **Generator:** Seedream-4

Instruction. Extract the pineapple visible in the image.



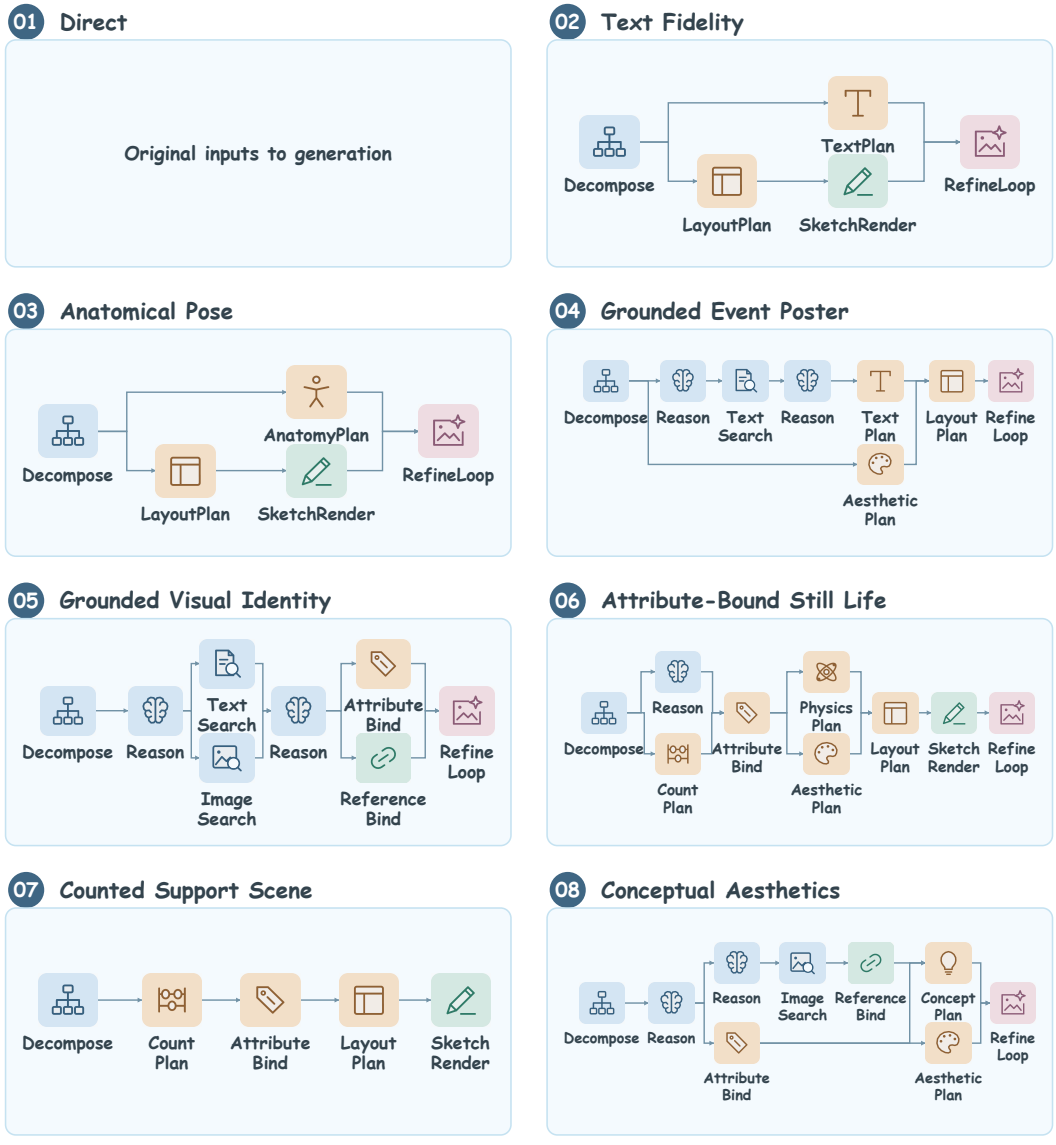
Figure D12: Ours isolates the complete pineapple and fills in its previously occluded base. Direct-Edit retains much of the sky; RS-Gen cuts off the fruit's bottom. Mind-Brush extracts the pineapple.

E Final Evolved Skill Banks

Figures E1 and E2 show structural overviews of the final text-to-image Skill banks after ten evolution batches with Seedream-4. Strong contains 11 learned Skills and Direct, for 12 in total; Weak contains 15 learned Skills and Direct, for 16 in total. Their workflows cover text, count, layout, and pose constraints, identification of real objects and places, causal reasoning, and conceptual and visual design for text-to-image generation.

Each panel shows meta-operation nodes and their dependencies. Arrows omit direct links already implied by other paths. The fixed Prompt Compiler and initial generation are shared across Skills and omitted from these overviews. Execution follows Appendix A.3, with refinement after initial generation. Direct passes original inputs to generation.

E.1 Strong Skill Bank



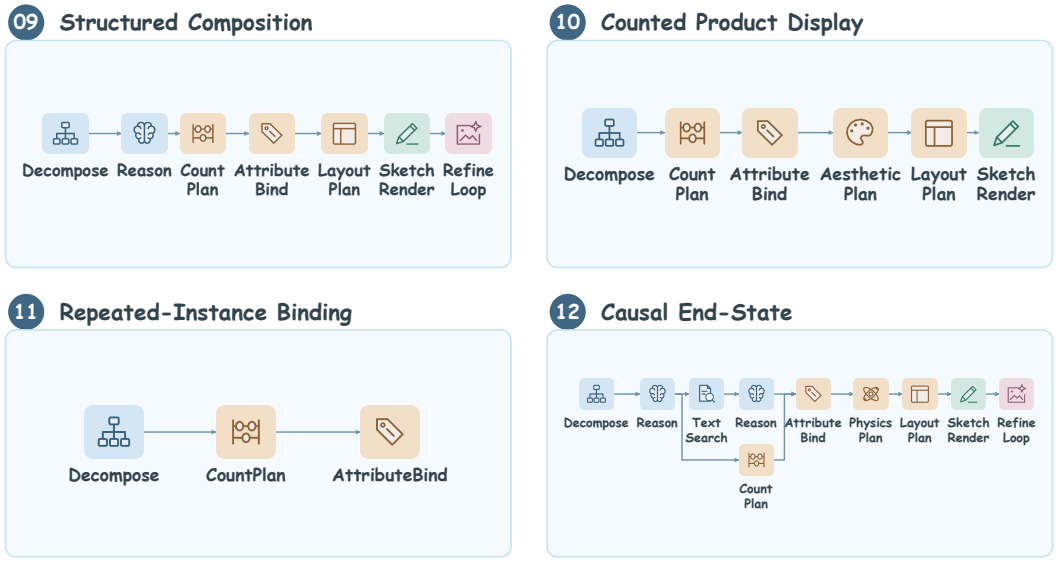
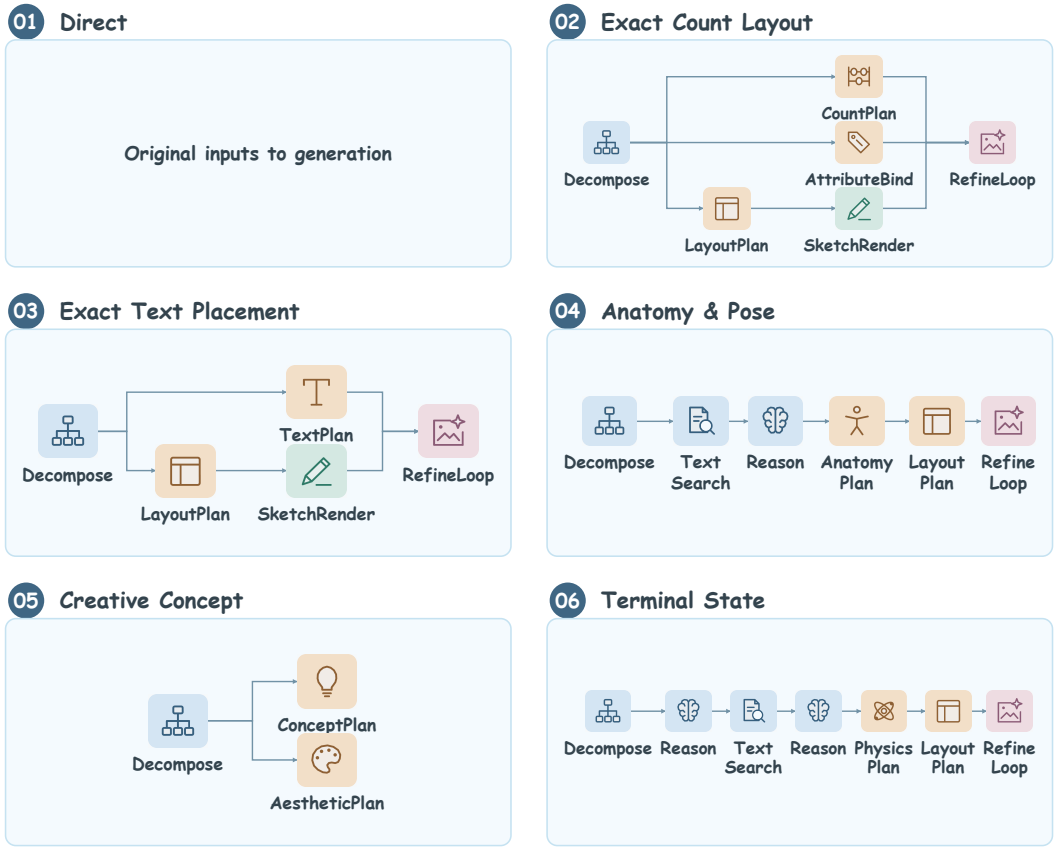


Figure E1: Final evolved Strong text-to-image Skill bank.

E.2 Weak Skill Bank



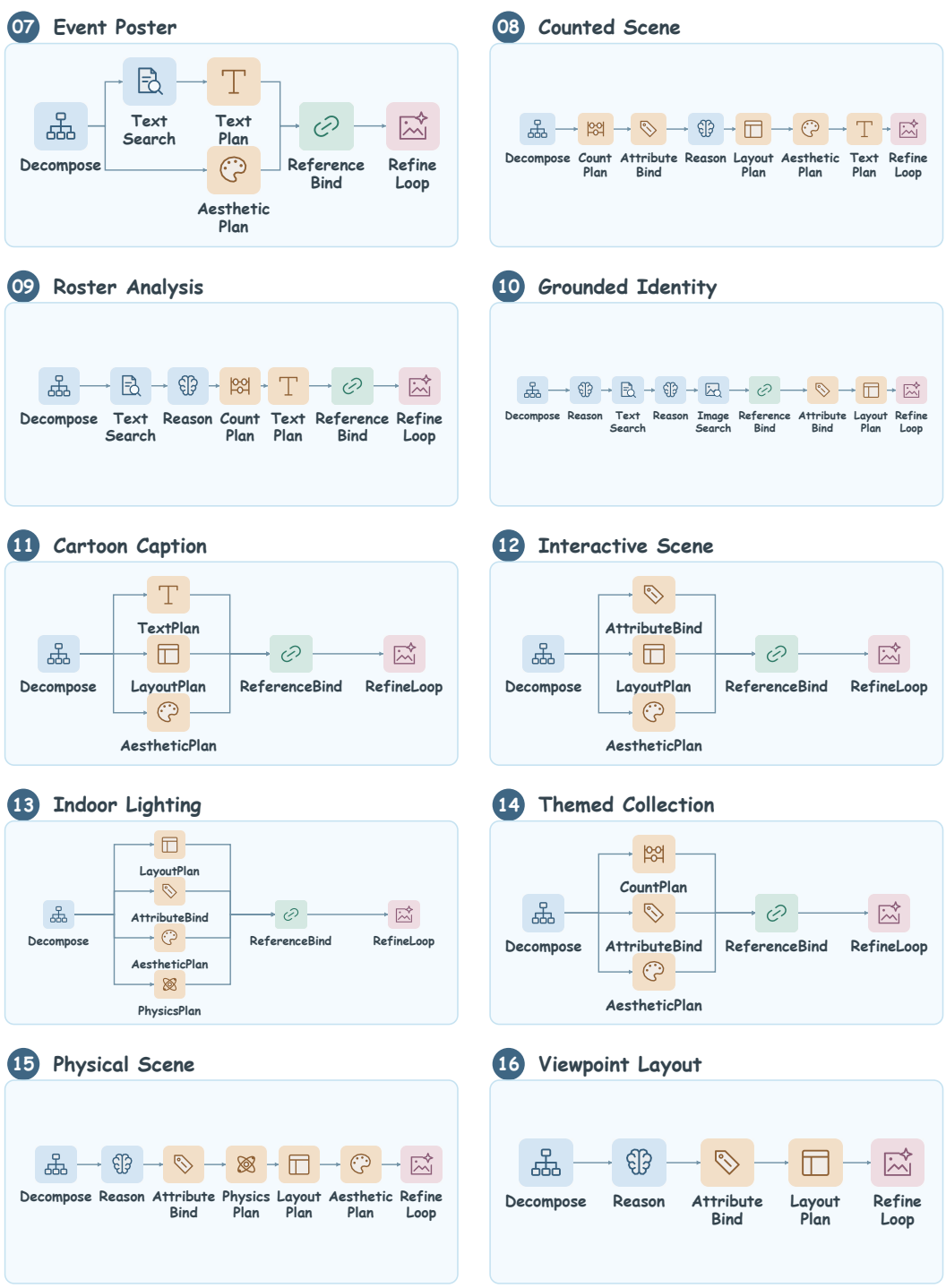


Figure E2: Final evolved Weak text-to-image Skill bank.