INTERNALIZING AGENT EXPERIENCE INTO DIFFUSION MODEL WEIGHTS VIA ON-POLICY CONTEXT DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Wrapping an image generation model in an agentic harness can effectively boost Text-to-Image task performance: the harness can leverage memory, skills, workflow orchestration, result verification, and iterative refinement to continually construct and revise prompts, thereby eliciting better images. These gains, however, remain external to the diffusion model and are realized only while the full harness runs. We propose Diffusion On-Policy Context Distillation (D-OPCD), which treats the agent-improved prompt as privileged context and distills the knowledge encoded in the agent harness into the weights of the diffusion model, so that the model retains part of the harness's benefit when conditioned on the original query alone. Using a Text-to-Image agent equipped with our proposed Auto Skill Evolver (ASE), we show that D-OPCD can internalize harness capabilities into the generator's weights, raising the average direct-generation score from 60.52 to 65.09 across four benchmarks. With this knowledge absorbed into the weights, the harness can shed its saturated skills and resume evolving: a second ASE round on the updated generator improves on a skill-free harness by additional 1.83 points, pointing toward text-to-image systems in which harness and model keep improving each other through continual co-evolution.

1 INTRODUCTION

Agentic harnesses have become an effective way to improve text-to-image (T2I) generation. A harness wraps a fixed generator with reasoning, memory, skills, and verification: within a task, it reasons about and iteratively refines its generations (Wang et al., 2025; He et al., 2026); across tasks, it accumulates experience so that useful strategies carry over to later requests (Chen et al., 2026b). As the harness evolves, it learns to use the generator better. The generator itself, however, does not change: the acquired gains live in external guidance that must be retrieved and applied again on every subsequent request.

Keeping these gains outside the generator has two costs. First, as memories and skills accumulate, relevant guidance becomes harder to retrieve. Second, reapplying this guidance to every request incurs repeated reasoning, verification, and image generation. Internalizing established gains could reduce both costs: a one-time offline training cost is amortized over future requests, and obsolete strategies can be discarded so the harness can target the updated generator's remaining limitations. This raises our central question: can gains discovered through harness evolution be internalized into the generator, enabling it to retain these improvements without external guidance at inference time?

Answering this question requires an interface between harness experience and generator learning. Although agents represent experience through different memories, skills, and reasoning traces, all ultimately condition the T2I generator on a text prompt. We therefore retain, for each completed task, the original query and the harness-generated prompt that yielded the selected image. This captures prompt-mediated gains, but not gains from verification or stochastic sample selection. Prior work uses visual experience to improve the agent policy that constructs generator inputs (Chen et al., 2026a), leaving the harness responsible for realizing these gains at inference time. In contrast, we use the resulting query–prompt pairs to train the generator itself, making the gains available from the original query alone.

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.

To perform this transfer, we introduce *Diffusion On-Policy Context Distillation (D-OPCD)*, a self-distillation algorithm for diffusion models. A teacher conditioned on both the original query and the harness-produced prompt supervises a student that receives only the original query, at states sampled from the student's own denoising trajectories. We formulate harness-to-generator transfer as on-policy context distillation (Ye et al., 2026) for diffusion models, with the evolving harness providing privileged textual context. Unlike prior on-policy diffusion self-distillation (Jiang et al., 2026a), which obtains privileged information from target or reference images, D-OPCD distills the gains elicited by the harness. It thereby internalizes prompt-mediated harness gains into the generator's weights, making them available without the harness at inference time.

We study this process with Auto Skill Evolver (ASE), a test-time learning mechanism that turns completed tasks into reusable prompt guidance; the resulting query–prompt pairs train D-OPCD. Distilling these pairs improves the generator's direct-generation performance across all four benchmarks. With the skill-free harness reattached, the updated generator outperforms its base-generator counterpart and even slightly exceeds the original skill-equipped agent on average, indicating that its learned skills have been internalized. Crucially, this establishes an iterative improvement cycle: after the generator internalizes the learned skills, resetting those now-redundant skills allows ASE to evolve again, yielding further improvements.

In summary, our main contributions are as follows:

- We introduce D-OPCD, an on-policy context distillation algorithm that uses harness-produced prompts as privileged teacher context to internalize prompt-mediated gains into diffusion model weights, making part of the harness's benefit available from the original query alone.
- We develop Auto Skill Evolver (ASE), a test-time learning mechanism that turns task experience into reusable skills and provides query–prompt pairs for D-OPCD. Experiments across four benchmarks demonstrate improved direct generation after internalization.
- We formulate harness–model co-evolution for T2I agents by coupling harness adaptation with generator learning. Internalizing accumulated harness gains into the generator allows the harness to shed its saturated skills and resume evolving around the updated model.

2 Related Work

2.1 Agentic Text-to-Image Generation

The quality of a generated image depends in part on how a user's request is translated into instructions for the generator. Prompt adaptation improves this interface by rewriting requests into prompts better suited to a fixed T2I model (Hao et al., 2023). For more complex requests, image-generation agents combine planning, generation, visual inspection, and prompt revision, using feedback from earlier attempts to address missing objects or unsatisfied constraints (Wang et al., 2025; Jiang et al., 2026b; Kovalev et al., 2025; Zhang et al., 2026b). This process can also extend beyond a single task: GEMS equips the agent with memory and skills, MemoGen reuses experience from earlier requests, and GenEvolve learns from generation trajectories to improve the agent's decisions (He et al., 2026; Chen et al., 2026b;a). These approaches show how better prompt construction and accumulated experience can improve the use of an image generator over time.

2.2 Training Diffusion Models

Updating a diffusion generator's weights provides another route to improving its outputs. Supervised fine-tuning can teach new visual concepts from a small set of examples, as in DreamBooth, or support continual customization across concepts (Ruiz et al., 2023; Smith et al., 2023). For qualities that are harder to specify with target images, training can instead use evaluative signals: ImageReward supplies preference-based feedback for ReFL, DDPO optimizes rewards over denoising trajectories, DRaFT backpropagates differentiable rewards through sampling, and Diffusion-DPO learns from preferred image pairs (Xu et al., 2023; Black et al., 2024; Clark et al., 2024; Wallace et al., 2024). Recent methods extend online reward optimization to flow-matching models and use teacher predictions along a student's own sampling trajectory to improve diffusion or flow generators (Liu et al.,

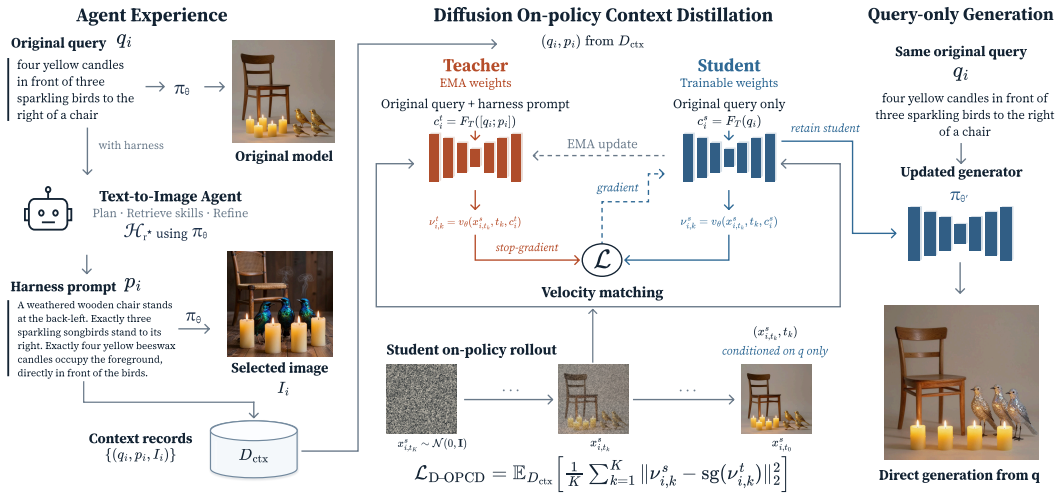

Figure 1: **Method overview.** As tasks accumulate, a T2I harness learns reusable skills that improve its prompts while the generator remains fixed. D-OPCD conditions an EMA teacher on the original query q_i and the harness-selected prompt p_i, and trains a student conditioned only on q_i to match the teacher's predictions along the student's own denoising trajectory. The updated generator can generate from q_i alone, while the harness resets its skills to continue adapting around it.

2025; Fang et al., 2026; Jiang et al., 2026a). This literature offers several ways to turn examples or feedback into persistent changes in generation behavior.

2.3 Harness–Model Co-Evolution

Experience-driven agent improvement begins by recording what succeeded or failed and turning those observations into guidance for later tasks. ExpeL extracts reusable lessons from trajectories, while MemSkill learns and revises skills for managing an agent's memory (Zhao et al., 2024; Zhang et al., 2026a). Recent work connects this external adaptation to policy learning by evolving skills alongside reinforcement learning (Xia et al., 2026a; Zhang et al., 2026c), updating an agent's harness and model across interactions (Xia et al., 2026b; Karten et al., 2026), or deriving hindsight skills that supervise the current policy (Wu et al., 2026). Across these settings, the current model shapes the experience available for the next update, while accumulated experience changes the model and the guidance surrounding it. We study this interaction in T2I generation, where the evolving harness constructs prompts for a separate image generator.

3 Method

3.1 Harness Gains at the Generator Interface

In a training-free T2I agent in which the harness may invoke the generator multiple times, the final performance gains arise mainly from two main channels: improving the conditioning supplied to a frozen diffusion model and using external tools to compensate for its limitations. Harness components—including memory, skills, planning, verification, and reflection—may interact in harness-specific ways to realize these gains. We focus on the prompt-mediated gain exposed at the generator interface: for a task with its original query q_i, each invocation of the diffusion model produces a prompt–image pair,

$$\mathcal{G}_i = \{(p_{i,j}, I_{i,j})\}_{j=1}^{J_i}, \qquad I_{i,j} \sim \pi_\theta(\cdot \mid p_{i,j}) \tag{1}$$

where each $p_{i,j}$ records a generation condition constructed by the harness and J_i is the number of diffusion model generation made by the harness for task i. Let $j_i^\star$ denote the call selected by the harness for submission, and define

$$(p_i, I_i) := (p_{i,j_i^\star}, I_{i,j_i^\star}) \in \mathcal{G}_i. \tag{2}$$

Under a fixed generator and sampling protocol, the quality of I_i provides an observable proxy for the effectiveness of p_i. The selected pair therefore provides a design-agnostic interface for internalization: p_i encodes the condition constructed by the harness, while I_i records its realized outcome.

3.2 Diffusion On Policy Context Distillation

3.2.1 Harness Internalization as Context Transfer

The selected prompt p_i is constructed and retained by the harness after reasoning, refinement, and verification. It therefore serves as a task-specific compilation of the harness's prompt-mediated experience, while the associated image I_i provides evidence that this condition is effective. Our goal is not merely to reproduce I_i from p_i, but to transfer the generation advantage *encoded* in p_i to the model when it receives only the original query q_i.

Direct SFT cannot realize this transfer. Training on (p_i, I_i) leaves the harness knowledge in the input and does not teach query-only generation, whereas training on (q_i, I_i) discards this privileged signal and supervises image-derived rather than student-visited states, creating an off-policy mismatch. We therefore introduce *Diffusion On-Policy Context Distillation* (D-OPCD), which the teacher has access to privileged context that is hidden from the student (Ye et al., 2026).

Specifically, D-OPCD uses a teacher diffusion model conditioned on both q_i and p_i to supervise a student model conditioned only on q_i, with both evaluated along the student's own denoising trajectory, following the on-policy diffusion construction (Jiang et al., 2026a). This provides a training signal without additional evaluator queries and directly targets the transfer from harness-assisted generation to generation under the original query.

3.2.2 On-Policy Distillation Objective

Harness version and context dataset. Let $\mathcal{H}_r$ denote a snapshot of the harness, where r indexes its version. The harness may evolve through self-evolution (Section 3.3) or human tuning, producing successive versions. A version $\mathcal{H}_r$ is admitted for internalization only if it improves over query-only generation on the held-out task set $\mathcal{Q}_{\text{hold}}$ (see the data protocol in Section 3.4). Let π_θ denote the generator on which $\mathcal{H}_r$ operates. This generator stays frozen while $\mathcal{H}_r$ produces data and is modified only by the D-OPCD update described below.

Running $\mathcal{H}_r$ with π_θ on $\mathcal{Q}_{\text{train}}$ produces completed tasks. Each task has a submitted pair (p_i, I_i), defined in Equation 2. Recall that the harness selects p_i itself: p_i is the final generation condition whose realized image passed the harness's internal verification. We retain N_r completed-task records to form the context dataset for $\mathcal{H}_r$:

$$\mathcal{D}_{\text{ctx}}^{(r)} = \{(q_i, p_i, I_i)\}_{i=1}^{N_r}. \tag{3}$$

The image I_i is used only for record selection, where it filters out tasks whose submitted outputs fail the selection criteria. D-OPCD only uses p_i as the sole privileged context.

Asymmetric conditioning. The text-conditioning encoder F_{T} maps text into the generator's conditioning space. We define two branches:

$$c_i^s = F_{\text{T}}(q_i), \qquad c_i^t = F_{\text{T}}([q_i; p_i]), \tag{4}$$

where $[q_i; p_i]$ denotes a structured text condition consisting of the original query followed by the harness-selected prompt. The student receives only the original user condition. The teacher receives the same condition together with the context constructed by the harness.

On-policy rollout. Following D-OPSD (Jiang et al., 2026a), let $v_\theta(x_t, t, c)$ denote the velocity field of the student generator, initialized from θ. The teacher $v_{\bar{\theta}}$ is initialized from the same parameters and is updated as an exponential moving average of the student. Let $0 = t_1 < \cdots < t_K < t_{K+1} = 1$ be the rollout schedule, and let Φ be the diffusion solver used at deployment. Starting from $x_{i,t_1}^s = \epsilon$, where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, the student produces its own rollout under c_i^s:

$$x_{i,t_{k+1}}^s = \Phi\big(x_{i,t_k}^s, t_k, t_{k+1}, v_\theta(\cdot, \cdot, c_i^s)\big), \qquad k = 1, \ldots, K. \tag{5}$$

The visited states are treated as constants, so gradients do not propagate through the rollout. At each visited state, the student and the teacher predict velocities under their respective conditions:

$$u^s_{i,k} = v_\theta(x^s_{i,t_k}, t_k, c^s_i), \qquad u^t_{i,k} = v_{\bar{\theta}}(x^s_{i,t_k}, t_k, c^t_i). \tag{6}$$

Objective. We train the student to match the privileged-context teacher on these shared on-policy states:

$$\mathcal{L}_{\text{D-OPCD}} = \mathbb{E}_{(q_i,p_i)\sim\mathcal{D}^{(r)}_{\text{ctx}},\ \epsilon\sim\mathcal{N}(0,\mathbf{I})}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|u^s_{i,k} - \text{sg}\left(u^t_{i,k}\right)\right\|_2^2\right], \tag{7}$$

where $\text{sg}(\cdot)$ denotes stop-gradient. This objective combines OPCD's context asymmetry with D-OPSD's diffusion-level on-policy alignment: both branches share q_i, p_i supplies additional teacher-only advantage information, and their predictions are compared on states visited by the student.

3.3 Self-Evolving Harness via Skill Extraction

To demonstrate a harness whose prompt-mediated gain improves with experience, we construct the *Auto Skill Evolver* (ASE), a hierarchical test-time learning framework that distills agent execution trajectories into reusable skills. Figure 2 summarizes the harness evolution process.

ASE organizes experience into three levels: 1) An *episode* is the agent's complete execution trajectory on a single task together with task-level feedback returned by an external evaluator after image submission; 2) An *insight*, extracted from episodes, records what the agent discovered during execution that could have been provided in advance; 3) A *skill* consolidates multiple insights into reusable guidance for constructing the first generation prompt; within the scope of this paper, skills do not invoke tools. At the start of each task, the agent loads the current skill library $\mathcal{S}$ and selects applicable skills to construct its first generation prompt, applying guidance learned from earlier tasks to the current request. Verification feedback and task-local history guide subsequent prompt refinement. Further details of the agent harness and ASE implementation are given in Appendix C.

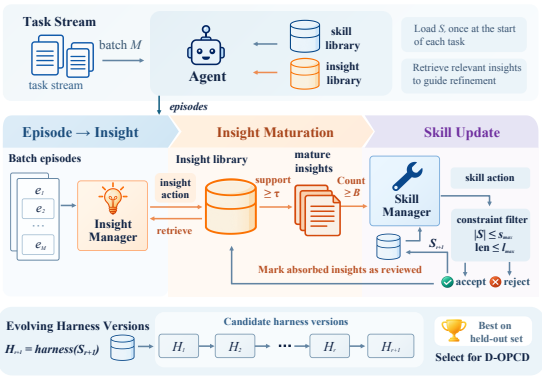

Figure 2: Harness evolution with ASE, from task episodes through insights to skill updates.

Insight extraction. Tasks arrive as a stream and are processed in batches of M tasks. During harness evolution, the agent may retrieve relevant insights from the pool $\mathcal{P}$ after failed verification to guide prompt refinement. The retrieved insights and the task outcome are recorded together, providing evidence about their usefulness across tasks. After each batch, an insight manager separately retrieves relevant insights from $\mathcal{P}$ to match against the completed episodes and may create, support, revise, contradict, merge, or archive insights. An insight becomes *mature* once its support count reaches τ and exceeds its contradiction count.

Skill update. Once $\mathcal{P}$ contains at least B mature insights not yet absorbed into the library, a skill manager examines them together with $\mathcal{S}$ and may leave the library unchanged or propose to create, revise, split, merge, or retire skills. Invalid plans are retried with validation feedback; if the staged library exceeds size or length limits, the manager reorganizes it before acceptance. When a proposal is accepted, its changes are committed and the insights it absorbs are marked as *reviewed*, excluding them from subsequent updates so that each update draws only on new evidence.

Harness versions. Frozen skill-only replay and evaluation disable cross-task insight retrieval and ASE updates. Accumulated experience therefore reaches the agent only through the committed skill library, so each committed update defines a new harness version. We denote by $\mathcal{H}_r$ the harness equipped with the r-th skill library $\mathcal{S}_r$. Among the resulting versions, the one to internalize with D-OPCD is selected on the held-out task set $\mathcal{Q}_{\text{hold}}$.

3.4 HARNESS–GENERATOR CO-EVOLUTION

Harness evolution improves the agent without training, but the harness has limited capacity: each added skill consumes context and makes it harder for the agent to choose the right one at inference time. D-OPCD relieves this limit by internalizing part of the harness knowledge into the generator weights. After the generator update, we clear the learned skills to examine whether the harness can acquire new guidance around the updated generator.

Data protocol. For each benchmark, we split the tasks into three disjoint sets, $\mathcal{Q} = \mathcal{Q}_{\text{train}} \sqcup \mathcal{Q}_{\text{hold}} \sqcup \mathcal{Q}_{\text{eval}}$. $\mathcal{Q}_{\text{train}}$ is used for harness evolution and D-OPCD training; $\mathcal{Q}_{\text{hold}}$ only for selecting harness versions and training checkpoints; and $\mathcal{Q}_{\text{eval}}$ only for reporting final results.

Harness evolution and version selection. With the current generator π_θ fixed, the agent processes $\mathcal{Q}_{\text{train}}$ as a stream and evolves its skill library with ASE (Section 3.3), producing harness versions $\{\mathcal{H}_r\}$. We evaluate each version on $\mathcal{Q}_{\text{hold}}$ and select the best-performing version $\mathcal{H}_{r^\star}$ using the external evaluator's aggregate score J.

$$r^\star = \arg\max_r \; J\big(\mathcal{H}_r;\; \pi_\theta,\; \mathcal{Q}_{\text{hold}}\big). \tag{8}$$

Replay and internalization. We freeze the skill library of $\mathcal{H}_{r^\star}$ and replay it on $\mathcal{Q}_{\text{train}}$ with π_θ. The resulting records form $\mathcal{D}_{\text{ctx}}^{(r^\star)}$ in Equation 3, and D-OPCD uses this dataset to update the generator to $\pi_{\theta'}$. We replay rather than reuse the streaming records so that all contexts come from a single harness version operating on the generator from which D-OPCD is initialized.

Generator adoption and harness reset. After D-OPCD, we adopt $\pi_{\theta'}$ as the new generator. To measure how much of the harness gain has been transferred into the weights, we define the signed internalization gap

$$\delta^{(r^\star)} = J\big(\mathcal{H}_\emptyset; \pi_{\theta'}\big) - J\big(\mathcal{H}_{r^\star}; \pi_\theta\big). \tag{9}$$

Both scores are measured on $\mathcal{Q}_{\text{eval}}$. A value near zero means that the updated generator without learned skills matches the original skill-equipped agent; a positive value favors the updated generator, while a negative value indicates a remaining deficit. We clear the skill library and its associated episode and insight records, retain the skill-free harness $\mathcal{H}_\emptyset$, and evolve new harness versions against $\pi_{\theta'}$. We denote by $\mathcal{H}_{r^\dagger}$ a version selected on $\mathcal{Q}_{\text{hold}}$ under $\pi_{\theta'}$ after this renewed harness evolution.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Tasks and benchmarks. We evaluate on GenEval (Ghosh et al., 2023), GenEval2 (Kamath et al., 2025), WISE (Niu et al., 2025), and R2I-Bench (Chen et al., 2025), which cover compositional, knowledge-intensive, and reasoning-intensive generation; T2I-CompBench++ (Huang et al., 2025) is reserved for out-of-distribution (OOD) evaluation. Each benchmark forms a sequential task stream: the agent receives a query and obtains evaluator feedback after submitting its final image. We use $\mathcal{Q}_{\text{train}}$, $\mathcal{Q}_{\text{hold}}$, and $\mathcal{Q}_{\text{eval}}$ as defined in Section 3.4, and report evaluation results on $\mathcal{Q}_{\text{eval}}$. The official evaluators provide GenEval Overall, GenEval2 Soft-TIFA AM, WIScore, and R2I-Score, respectively. Training-method comparisons use direct generation from q_i; the main results also evaluate the base and D-OPCD-updated generators with skill-free and evolved harnesses. Appendix E.1 gives split sizes, and Appendix E details the evaluation protocols.

Agent and harness evolution. We use Z-Image-Turbo as the initial T2I generator. The agent starts from a GEMS-based scaffold (He et al., 2026) with an empty skill library (the skill-free harness) and is equipped with the Auto Skill Evolver (ASE) described in Section 3.3. With the generator fixed, it processes $\mathcal{Q}_{\text{train}}$ as a task stream; ASE extracts insights from completed episodes and consolidates mature insights into skills, producing successive harness versions. Appendix C.1 describes the scaffold design, and Appendix C.4 reports the benchmark-specific ASE settings.

Table 1: Base and D-OPCD generator scores under direct generation, a skill-free harness, and evolved skills. Parenthesized Δ values compare base + rows with base direct generation and D-OPCD rows with the corresponding rows in the base-generator block above, in the same order.

Method	GenEval ↑	GenEval2 ↑	WISE ↑	R2I-Bench ↑	Avg. ↑
Base generator, direct	78.22	78.14	42.86	42.87	60.52
+ skill-free harness ($\mathcal{H}_\emptyset$)	83.51 (+5.29)	87.50 (+9.36)	78.00 (+35.14)	68.12 (+25.25)	79.28 (+18.76)
+ evolved skills ($\mathcal{H}_{r^\star}$)	87.63 (+9.41)	88.83 (+10.69)	81.43 (+38.57)	69.41 (+26.54)	81.83 (+21.31)
D-OPCD generator, direct	81.28 (+3.06)	82.24 (+4.10)	50.00 (+7.14)	46.83 (+3.96)	65.09 (+4.57)
+ skill-free harness ($\mathcal{H}_\emptyset$)	86.08 (+2.57)	88.41 (+0.91)	81.14 (+3.14)	**73.68** (+5.56)	82.33 (+3.05)
+ newly evolved skills ($\mathcal{H}_{r^\dagger}$)	**91.75** (+4.12)	**89.74** (+0.91)	**85.71** (+4.28)	69.45 (+0.04)	**84.16** (+2.33)

Compared training methods. For a matched comparison, all methods use the same prompt-changed task records replayed from the selected skill-equipped harness $\mathcal{H}_{r^\star}$ on $\mathcal{Q}_{\text{train}}$. They start from the same Z-Image-Turbo weights and train LoRA adapters with the same optimization budget. D-OPCD uses each (q_i, p_i) pair: an EMA teacher conditioned on $[q_i; p_i]$ supervises a student conditioned only on q_i along the student's own sampling trajectory, with p_i hidden at inference. Vanilla SFT fits the agent-submitted image I_i from q_i with a standard flow-matching objective. Diffusion-DPO (Wallace et al., 2024) treats I_i as chosen and a same-seed image generated directly from q_i as rejected, training both under q_i. D-OPSD (Jiang et al., 2026a) uses an image-conditioned teacher based on (q_i, I_i) to supervise a q_i-conditioned student along its own trajectory. Appendix D.1 details the record selection; Appendix D gives training objectives and settings.

4.2 Main Results

Harness gains at the generator interface. To assess the gains provided by the agent harness, we evaluate the base generator π_θ on $\mathcal{Q}_{\text{eval}}$ alone, with the skill-free harness $\mathcal{H}_\emptyset$, and with the selected skill-equipped harness $\mathcal{H}_{r^\star}$ evolved by ASE. Both harness configurations improve performance over direct generation on all four benchmarks (Table 1). The skill-free harness raises the four-benchmark average from 60.52 to 79.28 (+18.76 points), including gains from 42.86 to 78.00 on WISE and from 42.87 to 68.12 on R2I-Bench. ASE-evolved skills raise the average further to 81.83 (+2.55 points over $\mathcal{H}_\emptyset$), with additional gains on every benchmark, ranging from +1.29 points on R2I-Bench to +4.12 on GenEval. The prompts selected by $\mathcal{H}_{r^\star}$ then provide the privileged contexts for D-OPCD.

Internalizing harness gains. To test whether the harness's gains can be internalized, we train the generator on (q_i, p_i) pairs from $\mathcal{H}_{r^\star}$ with D-OPCD and evaluate $\pi_{\theta'}$ directly from q_i, without a harness. It outperforms the base generator on all four benchmarks, raising the average score from 60.52 to 65.09 (Table 1). Some of the harness benefit therefore persists in the model weights.

Harness benefits after internalization. To test whether internalization reduces reliance on the original evolved skills, we attach the skill-free harness $\mathcal{H}_\emptyset$ to $\pi_{\theta'}$ and compare it with the base generator under $\mathcal{H}_\emptyset$ and $\mathcal{H}_{r^\star}$. The updated skill-free agent outperforms its base-generator counterpart on all four benchmarks (Table 1). It scores 82.33 on average and slightly exceeds the original skill-equipped agent, yielding a signed internalization gap of $\delta^{(r^\star)} = +0.50$. On R2I-Bench, it is also ahead, with $\delta^{(r^\star)} = +4.27$. Thus, much of the earlier skill benefit can be retained after removing those skills, although the gap varies across benchmarks.

Continued co-evolution after model update. After adopting $\pi_{\theta'}$, we reset the learned skills and rerun ASE to obtain $\mathcal{H}_{r^\dagger}$. The renewed skills improve the updated agent on three benchmarks and raise its average above both the updated skill-free agent and the original skill-equipped agent (Table 1). R2I-Bench instead declines after skill evolution. The D-OPCD generator with the skill-free harness already scores highly, which may leave less room for additional gains. These results illustrate how D-OPCD unlocks a harness–model co-evolution mechanism, with renewed skills building on the updated generator. Matched image examples are provided in Appendix A.

Table 2: Direct-generation comparison on $\mathcal{Q}_{\text{eval}}$. Updated models train on matched records from $\mathcal{H}_{r^\star}$: D-OPCD uses (q_i, p_i), while baselines use the agent image I_i. Parenthesized Δ values are differences from the base generator; Avg. is the unweighted mean across four benchmarks.

Method	**GenEval** ↑	**GenEval2** ↑	**WISE** ↑	**R2I-Bench** ↑	**Avg.** ↑
Base generator	78.22	78.14	42.86	42.87	60.52
Vanilla SFT	80.09 (+1.87)	80.94 (+2.80)	46.00 (+3.14)	46.51 (+3.64)	63.38 (+2.86)
Diffusion-DPO (Wallace et al., 2024)	77.56 (-0.66)	79.04 (+0.90)	45.43 (+2.57)	46.70 (+3.83)	62.18 (+1.66)
D-OPSD (Jiang et al., 2026a)	80.72 (+2.50)	76.48 (-1.66)	48.86 (+6.00)	44.96 (+2.09)	62.75 (+2.23)
D-OPCD (ours)	**81.28** (+3.06)	**82.24** (+4.10)	**50.00** (+7.14)	**46.83** (+3.96)	**65.09** (+4.57)

Table 3: D-OPCD performance with different teacher inputs on $\mathcal{Q}_{\text{eval}}$. The teacher receives the original query with its matched harness prompt ($[q_i; p_i]$), the harness prompt alone (p_i only), or the query with a prompt from another task (Shuffled). The student always receives q_i.

Teacher input	**GenEval** ↑	**GenEval2** ↑	**WISE** ↑	**R2I-Bench** ↑	**Avg.** ↑
D-OPCD: $[q_i; p_i]$	**81.28**	**82.24**	**50.00**	46.83	**65.09**
p_i only	80.93	81.13	49.14	**47.32**	64.63
Shuffled: $[q_i; p_{\sigma(i)}]$	74.50	75.38	48.57	45.76	61.05

4.3 Ablation Studies and Further Analysis

4.3.1 Comparison with Alternative Training Methods

To compare methods for internalizing agent experience, we train D-OPCD and three baselines on matched records collected by the evolved harness and evaluate direct generation from the original queries (Table 2). All four improve the base generator's four-benchmark average, with gains ranging from 1.66 to 4.57 points. D-OPCD achieves the highest mean of 65.09, 1.71 points above Vanilla SFT, and leads on all four benchmarks. On WISE, D-OPCD improves over the base generator by 7.14 points. Overall, these results suggest that D-OPCD has an advantage over the tested baselines in internalizing agent experience into the generator's model weights.

4.3.2 Ablation of Teacher Conditioning

With the student conditioned on q_i, we compare three teacher inputs: matched $[q_i; p_i]$, p_i alone, and shuffled $[q_i; p_{\sigma(i)}]$ using another task's prompt. The matched input averages 65.09, ahead of p_i alone (64.63) and shuffled prompts (61.05). It leads on three benchmarks, while p_i alone reaches the highest R2I-Bench score of 47.32. Shuffling reduces scores on all four benchmarks, underscoring the value of task-matched teacher prompts (Table 3).

4.3.3 Context Properties and Model Internalization

Table 1 shows that training on contexts replayed from the selected skill-equipped harness $\mathcal{H}_{r^\star}$ improves direct generation from the original query on all four benchmarks. To explore how context properties relate to model internalization, we examine two observable properties of the collected (q_i, p_i) pairs: the submitted image's advantage over direct-query generation and variation in prompt wording. We compare these data-level properties with direct-generation scores after training.

Context design. We vary the default prompt-changed replay set in two ways. **1) Improvement filter.** We retain a pair only when the agent's internal verifier gives the submitted image a strictly higher pass count than the direct-query image. **2) Diversity adapter.** We change the agent's prompt-

Table 4: D-OPCD direct-generation test scores after training on three context sets. Pair counts follow GenEval/GenEval2/WISE/R2I-Bench order; Avg. is the four-benchmark mean before rounding.

Setting	Pairs	GenEval ↑	GenEval2 ↑	WISE ↑	R2I-Bench ↑	Avg. ↑
Default ($p \neq q$)	276/400/490/392	81.28	**82.24**	**50.00**	46.83	65.09
Improvement filter	43/264/319/314	**82.98**	81.76	47.71	**48.13**	**65.14**
Diversity adapter	276/400/491/381	82.74	81.75	48.29	44.92	64.42

Table 5: Out-of-distribution category scores on T2I-CompBench++ for the base generator and four D-OPCD checkpoints. The suffix in each D-OPCD method name identifies its training benchmark.

Method	Color	Shape	Texture	2D-Spatial	3D-Spatial	Numeracy	Non-Spatial
Base generator	77.45	57.54	72.79	32.32	41.27	67.98	31.24
D-OPCD-GenEval	**81.61**	57.47	73.00	36.19	41.53	70.26	**31.52**
D-OPCD-GenEval2	81.44	56.56	72.36	**37.33**	**42.02**	**71.05**	31.31
D-OPCD-WISE	77.17	57.21	72.67	32.34	39.74	67.09	31.34
D-OPCD-R2I-Bench	77.06	**57.84**	**73.09**	32.00	41.07	68.42	31.33

sampling policy to reduce repeated prompt patterns, then collect its prompt-changed pairs. Appendix E.3 gives the construction details for both variants.

Results and analysis. The improvement-filtered pairs show larger observed image-score gaps on their retained tasks, while the diversity adapter increases character-level variation in the collected prompts (Appendix E.3). However, downstream gains are mixed: both variants improve individual benchmarks, but the filter leaves the four-benchmark mean nearly unchanged and the adapter lowers it (Table 4). Filtering also changes the training set's size and composition, so its effect cannot be attributed solely to task improvement. A prompt that is more useful during generation is therefore not necessarily easier for the query-only student to internalize when used as context. Identifying which harness-produced contexts best support model internalization remains an open question.

4.3.4 Out-of-Distribution Generalization

We evaluate the base generator and four source-specific D-OPCD checkpoints through direct generation without a harness on seven T2I-CompBench++ categories reserved for out-of-distribution testing (Huang et al., 2025). Color scores rise from 77.45 for the base generator to 81.61 and 81.44 for the GenEval- and GenEval2-trained checkpoints, respectively. Their 2D-Spatial scores likewise reach 36.19 and 37.33, up from 32.32 for the base (Table 5). The WISE- and R2I-Bench-trained checkpoints remain close to the base overall. These results show good out-of-distribution performance, especially with GenEval and GenEval2 as training sources.

5 Conclusion

We introduce D-OPCD to internalize prompt-mediated gains from an evolving T2I harness into the generator's weights, allowing the model to retain part of the harness's benefit from the original query alone. With ASE turning task experience into reusable skills, D-OPCD improves direct generation across all four benchmarks. The updated generator with a skill-free harness slightly exceeds the original skill-equipped agent on average, and resetting the old skills allows ASE to evolve again and further improve performance. We hope this work guides the development of T2I systems in which accumulated experience becomes lasting generator capability, creating room for the harness and model to keep improving each other through continual co-evolution.

REFERENCES

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 4965–4987, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/14f75513f0f1ca01de1e826b52e6b840-Paper-Conference.pdf.

Kaijie Chen, Zihao Lin, Zhiyang Xu, Ying Shen, Yuguang Yao, Joy Rimchala, Jiaxin Zhang, and Lifu Huang. R2I-bench: Benchmarking reasoning-driven text-to-image generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12595–12630, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.636. URL https://aclanthology.org/2025.emnlp-main.636/.

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, 2026a. URL https://arxiv.org/abs/2605.21605.

Wenshuo Chen, Kuimou Yu, Bowen Tian, Jianfei Song, Shaofeng Liang, Haozhe Jia, Kan Cheng, Haosen Li, Kaishen Yuan, Lei Wang, Jiemin Wu, Songning Lai, and Yutao Yue. Memogen: Can past experience improve future text-to-image generation?, 2026b. URL https://arxiv.org/abs/2606.03243.

Kevin Clark, Paul Vicol, Kevin Swersky, and David Fleet. Directly fine-tuning diffusion models on differentiable rewards. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 4793–4822, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/13e8be77982beb73d7ed0bbf122f9f3c-Paper-Conference.pdf.

Zhen Fang, Wenxuan Huang, Yu Zeng, Yiming Zhao, Shuang Chen, Kaituo Feng, Yunlong Lin, Lin Chen, Zehui Chen, Shaosheng Cao, and Feng Zhao. Flow-opd: On-policy distillation for flow matching models, 2026. URL https://arxiv.org/abs/2605.08063.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *Advances in Neural Information Processing Systems 36*, NeurIPS 2023, pp. 52132–52152. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-2270. URL http://dx.doi.org/10.52202/075280-2270.

Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 66923–66939. Curran Associates, Inc., 2023. doi: 10.52202/075280-2923. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d346d91999074dd8d6073d4c3b13733b-Paper-Conference.pdf.

Zefeng He, Siyuan Huang, Xiaoye Qu, Yafu Li, Tong Zhu, Yu Cheng, and Yang Yang. Gems: Agent-native multimodal generation with memory and skills, 2026. URL https://arxiv.org/abs/2603.28088.

Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench++: An enhanced and comprehensive benchmark for compositional text-to-image generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(5):3563–3579, May 2025. ISSN 1939-3539. doi: 10.1109/tpami.2025.3531907. URL http://dx.doi.org/10.1109/TPAMI.2025.3531907.

Dengyang Jiang, Xin Jin, Dongyang Liu, Zanyi Wang, Mingzhe Zheng, Ruoyi Du, Xiangpeng Yang, Qilong Wu, Zhen Li, Peng Gao, Harry Yang, and Steven Hoi. D-opsd: On-policy self-distillation for continuously tuning step-distilled diffusion models, 2026a. URL https://arxiv.org/abs/2605.05204.

Kaixun Jiang, Yuzheng Wang, Junjie Zhou, Pandeng Li, Zhihang Liu, Chen-Wei Xie, Zhaoyu Chen, Yun Zheng, and Wenqiang Zhang. Genagent: Scaling text-to-image generation via agentic multimodal reasoning, 2026b. URL https://arxiv.org/abs/2601.18543.

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.

Seth Karten, Joel Zhang, Tersoo Upaa, Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual harness: Online adaptation for self-improving foundation agents, 2026. URL https://arxiv.org/abs/2605.09998.

V. Kovalev, A. Kuvshinov, A. Buzovkin, D. Pokidov, and D. Timonin. Craft: Continuous reasoning and agentic feedback tuning for multimodal text-to-image generation, 2025. URL https://arxiv.org/abs/2512.20362.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 110–119. Association for Computational Linguistics, 2016. doi: 10.18653/v1/n16-1014. URL http://dx.doi.org/10.18653/v1/N16-1014.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl, 2025. URL https://arxiv.org/abs/2505.05470.

Yuwei Niu, Munan Ning, Mengren Zheng, Weiyang Jin, Bin Lin, Peng Jin, Jiaqi Liao, Chaoran Feng, Fanqing Meng, Kunpeng Ning, Bin Zhu, and Li Yuan. Wise: A world knowledge-informed semantic evaluation for text-to-image generation, 2025. URL https://arxiv.org/abs/2503.07265.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22500–22510, June 2023.

James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. Continual diffusion: Continual customization of text-to-image diffusion with c-lora, 2023. URL https://arxiv.org/abs/2304.06027.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8228–8238. IEEE, 2024. doi: 10.1109/cvpr52733.2024.00786. URL http://dx.doi.org/10.1109/cvpr52733.2024.00786.

Kaishen Wang, Ruibo Chen, Tong Zheng, and Heng Huang. Imagent: A unified multimodal agent framework for test-time scalable image generation, 2025. URL https://arxiv.org/abs/2511.11483.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. Seed: Self-evolving on-policy distillation for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2607.14777.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning, 2026a. URL https://arxiv.org/abs/2602.08234.

Peng Xia, Jianwen Chen, Xinyu Yang, Haoqin Tu, Jiaqi Liu, Kaiwen Xiong, Siwei Han, Shi Qiu, Haonian Ji, Yuyin Zhou, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Metaclaw: Just talk – an agent that meta-learns and evolves in the wild, 2026b. URL https://arxiv.org/abs/2603.17187.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems 36*, NeurIPS 2023, pp. 15903–15935. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-0700. URL http://dx.doi.org/10.52202/075280-0700.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models, 2026. URL https://arxiv.org/abs/2602.12275.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents, 2026a. URL https://arxiv.org/abs/2602.02474.

Zekai Zhang, Jiahao Li, Jie Zhang, Kaiyuan Gao, Kun Yan, Lihan Jiang, Ningyuan Tang, Shengming Yin, Tianhe Wu, Xiaoyue Chen, Xiao Xu, Yan Shu, Yanran Zhang, Yixian Xu, Yuxiang Chen, Zhendong Wang, Zihao Liu, Zikai Zhou, Huishuai Zhang, Dongyan Zhao, and Chenfei Wu. Qwen-image-agent: Bridging the context gap in real-world image generation, 2026b. URL https://arxiv.org/abs/2606.26907.

Zhiwei Zhang, Yudi Lin, Nikki Lijing Kuang, Linlin Wu, Xiaomin Li, Songtao Liu, and Fenglong Ma. Co-evolving skill generation and policy optimization, 2026c. URL https://arxiv.org/abs/2606.08755.

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, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i17.29936. URL http://dx.doi.org/10.1609/aaai.v38i17.29936.

A Qualitative Examples for the Main Results

Figure 3 shows selected test queries for which D-OPCD improves direct generation. Table 1 reports the aggregate results over the full evaluation sets.

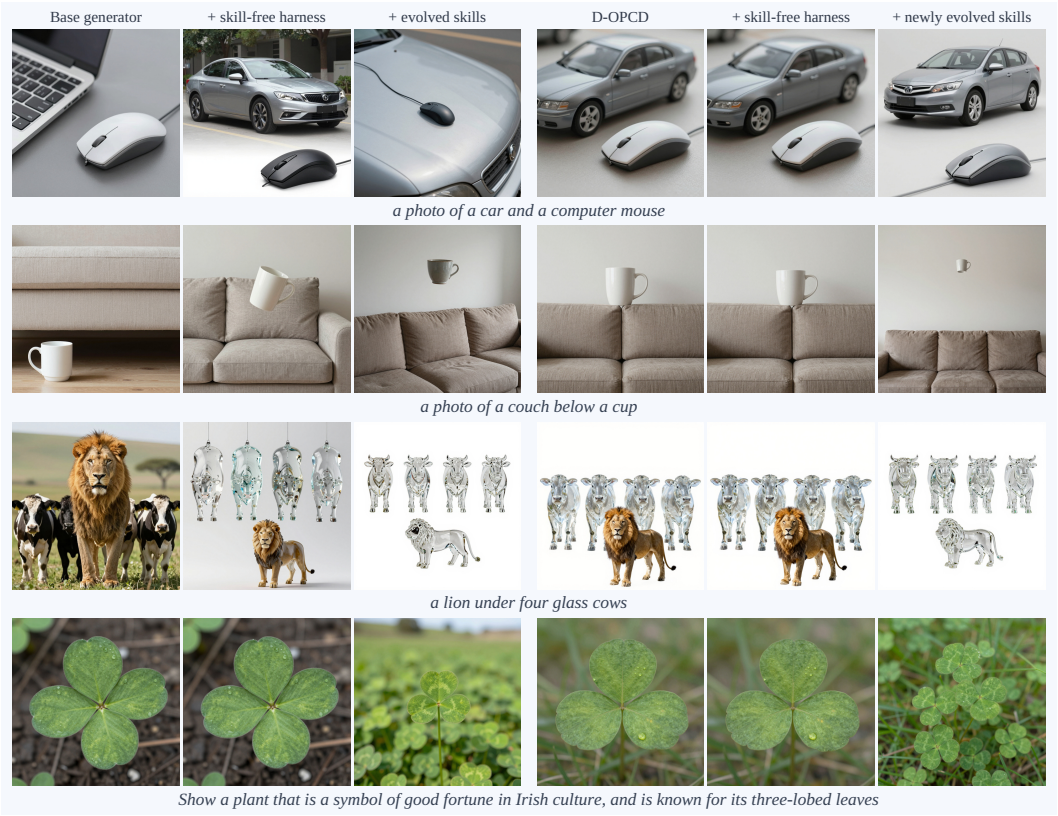

Figure 3: Matched outputs for four test queries. The left triplet uses the base generator π_θ and the right triplet uses the D-OPCD-updated generator $\pi_{\theta'}$. Within each triplet, the columns show direct generation, the skill-free harness $\mathcal{H}_\emptyset$, and evolved skills ($\mathcal{H}_{r^\star}$ or $\mathcal{H}_{r^+}$). The text below each row is the original query; harness runs may use selected generation prompts.

B Discussion on Limitations and Future Works

Limits of internalizing skill-evolution experience. Internalizing the experience from skill evolution yields only modest gains for the standalone generator: D-OPCD raises the four-benchmark direct-generation average from 60.52 to 65.09, still far below the 81.83 achieved by the original skill-equipped harness (Table 1). One possible explanation is that prompts produced by evolved skills, while effective when used by the harness, may be poorly aligned with the diffusion model's latent space for learning the same behavior from the original query q_i alone. Further studies could explore which harness designs produce experience that supports both immediate task success and subsequent model learning, and which algorithms can internalize this broader harness experience. Stronger mechanisms for jointly adapting the harness and generator may be needed to realize sustained co-evolution.

Evidence for repeated co-evolution. Our experiments evaluate one base generator and one D-OPCD update on four benchmarks, followed by one subsequent round of skill evolution. They show that a newly evolved harness can build on the updated generator, but do not establish what happens over many update-and-reset cycles or across different generators and task distributions. Longer-

term studies could measure retention of earlier gains, transfer to new tasks, and the tradeoff between offline training cost and recurring harness inference cost. Repeated model updates may also require methods that prevent forgetting while adapting the harness to each new generator.

Beyond text-to-image agents. The agent studied here improves generation through prompt construction, verification, and refinement, without external knowledge search or reference-image tools. Other image-generation agents may search the web, retrieve reference images, and use additional tools. In addition to text-to-image generation, they may handle image-to-image generation and other image-generation tasks. Their experience includes tool choices, retrieved information, and visual evidence that a final text prompt cannot fully represent. Extending internalization to these settings requires learning which reusable capabilities can be transferred into model weights and which information should remain available through tools at inference. A further open problem is how to use tool-augmented experience gathered at test time to guide model updates and then evolve the harness again around the updated generator.

C Implementation Details of the Agent Harness and Skill Evolution

C.1 GEMS-Based Agent Harness Design

Planning and requirements. We adapt the GEMS agent architecture (He et al., 2026) to a text-to-image harness with five roles: Planner, Decomposer, Generator, Verifier, and Refiner. Given an original query q_i, the Planner selects applicable Skills using their names and descriptions, then loads their instructions to construct the first generation prompt $p_{i,1}$. If it selects no Skill, $p_{i,1} = q_i$. The Decomposer then turns q_i into a fixed set of yes/no visual requirements $\mathcal{C}_i = (c_{i,1}, \ldots, c_{i,m_i})$. Deriving these checks from the original query keeps the target of verification fixed as prompts are revised.

Generation and refinement. The Generator produces an image $I_{i,j}$ from the current prompt $p_{i,j}$. The multimodal Verifier checks the image against every requirement in $\mathcal{C}_i$. If any check fails, the Refiner uses the prompt, image, verification feedback, and task-local history of earlier attempts to produce $p_{i,j+1}$. That history retains the prompts, images, check results, and short experience summaries used during refinement. The agent stops when all checks pass or the attempt budget is exhausted. During evolution, relevant Insights may also be retrieved to guide refinement after a failure; frozen skill-only replay and evaluation disable this retrieval. External benchmark scores do not guide refinement within a task.

Returned output. Let $n_{i,j}$ be the number of requirements satisfied by attempt j, and let J_i be the number of attempts executed. The agent returns the prompt and image from the earliest attempt with the highest count:

$$j_i^* = \min\left\{ j : n_{i,j} = \max_{1 \le \ell \le J_i} n_{i,\ell} \right\}, \qquad (p_i, I_i) = (p_{i,j_i^*}, I_{i,j_i^*}). \tag{10}$$

Thus, a later unsuccessful revision can leave an earlier image as the returned output. The resulting (p_i, I_i) is the harness output used for context-data collection. The agent prompt templates are listed in Appendix C.5.

C.2 Episode Construction and Insight Extraction

Episode records. Each completed task yields one Episode. An Episode records (i) the task metadata, including the original request q_i and the deployed generator version; (ii) the ordered generation attempts and verification results; (iii) the Skills selected at the start of the task and any Insights retrieved for refinement; and (iv) the returned attempt, its passed and failed checks, the first-attempt check score, and any feedback from the external evaluator. Multiple attempts within the same Episode still count as one evidence unit. An optional LLM-written summary may be attached, but we disable it so that consolidation uses the factual attempt record and outcome.

Table 6: Action space of the Insight manager $\mathcal{M}_I$. All actions require at least one Episode reference from the current batch; targets must be active Insights in the frozen working set.

Action	Parameters	Semantics	Key validation rules
ADD	`text`, `episode_refs`	Create an Insight capturing a distinct, transferable refinement lesson.	Exact or semantic (≥ 0.92 cosine) duplicates are not created; the action is resolved as SUPPORT for the matching Insight instead.
SUPPORT	`insight_ref`, `episode_refs`	Attach supporting Episodes without changing the text.	Cannot carry text; the same Episode supports an Insight at most once per batch.
REVISE	`insight_ref`, `text`, `episode_refs`	Refine the scope, boundary, or wording of an Insight and attach the motivating support.	Requires non-empty text; an Insight may be revised at most once per batch.
CONTRADICT	`insight_ref`, `episode_refs`, `reason`	Record Episode evidence that weakens or limits an Insight.	Requires a reason; one Episode cannot both support and contradict the same Insight.
MERGE	`target_ref`, `merged_refs`, `text`, `episode_refs`	Fold overlapping Insights into the target with a complete reformulation; sources are archived.	Target and sources must be distinct active entries; the merged evidence is re-attributed to the target.
ARCHIVE	`insight_ref`, `episode_refs`, `reason`, [`replacement_ref`]	Retire obsolete or redundant knowledge, optionally pointing to a replacement Insight.	Requires a reason; the replacement must be active and different from the archived entry.

Frozen working set. At the batch barrier, the learner builds one retrieval query per Episode from its request and failure evidence, retrieves the top-4 active Insights whose cosine similarity to the query exceeds 0.35 using BGE-M3 embeddings, and takes the union over Episodes as the working set $\mathcal{N}_k$. This management-only retrieval includes immature single-Episode hypotheses, so that new evidence can strengthen, correct, or refute them. The working set is frozen before the manager runs: the Insight manager $\mathcal{M}_I$ may reference only these entries and only Episodes of the current batch.

Action space. $\mathcal{M}_I$ returns an ordered list of operations. Table 6 summarizes the six actions, and Figure 10 shows the manager's decision prompt. The action space is deliberately small: evidence moves are separated from text moves, so support and contradiction accumulate as Episode references while wording changes remain rare, explicit, and evidence-backed.

Validation and commit. Operations are checked in order against the frozen working set: references must resolve, text constraints must hold per action, and one Episode may not take both a supporting and a contradicting direction toward the same Insight. Only operations satisfying these rules update the Insight pool.

Evidence and maturity. Support and contradiction counts use distinct Episodes, not the number of attempts within an Episode. The support threshold τ and the minimum number of supporting batches required before an Insight is passed to the Skill manager vary by benchmark (Table 9). The Insight store retains contradictory Episode references so they can affect maturity and motivate a revision or archival.

C.3 SKILL UPDATES AND LIBRARY VERSIONING

Update trigger. After each batch k, Skill evolution is considered when at least B mature, previously unreviewed Insights have accumulated and the minimum interval since the preceding completed Skill review has elapsed. Both settings vary by benchmark (Table 9). The mature-insight trigger controls Skill maintenance independently of generator updates.

Action space. Once triggered, the Skill manager $\mathcal{M}_S$ examines the pending mature Insights together with summaries of the current library and the recent change history, and returns one ordered

Table 7: Action space of the Skill manager $\mathcal{M}_S$. `source_insight_refs` identifies the mature Insights absorbed by an edit. An empty plan is a deliberate no-op.

Action	Parameters	Semantics	Key validation rules
CREATE	`suggested_id`, `content`, `source_insight_refs`	Add a distinct high-level Skill as complete natural-language content.	Requires at least one source Insight; subject to the capacity bound C.
REPLACETEXT	`skill_id`, `old_text`, `new_text`, `source_insight_refs`	Replace an exact passage in one Skill.	`old_text` must match a unique location in the target Skill.
INSERTTEXT	`skill_id`, `anchor`, `text`, `position`, `source_insight_refs`	Insert text immediately before or after an anchor.	The anchor must match uniquely; `position` $\in$ {`before`, `after`}.
DELETETEXT	`skill_id`, `text`, `source_insight_refs`	Remove an exact passage from one Skill.	The passage must match uniquely.
RENAME	`skill_id`, `new_title`, `source_insight_refs`	Change a Skill's title while preserving its stable identifier and body.	Requires a non-empty title.
SUMMARY	`skill_id`, `content`, `source_insight_refs`	Replace an overgrown Skill with shorter complete content, preserving its identity.	Result must satisfy the length bound L.
SPLIT	`skill_id`, `new_skills`, `source_insight_refs`	Replace one over-broad Skill with two complete, independently routeable Skills.	Exactly two replacement Skills; subject to C.
MERGE	`target_skill_id`, `merged_skill_ids`, `content`, `source_insight_refs`	Merge overlapping Skills into the target with complete resulting content.	Sources are retired; result must satisfy C and L.
RETIRE	`skill_id`, `replacement_skill_id`	Retire a redundant Skill in favor of an active replacement.	The replacement must be an active Skill.

plan from the nine-action space in Table 7. Figure 11 shows its decision prompt. Text-level edits (REPLACE, INSERT, DELETE) operate on exact, uniquely matching passages copied from the target Skill, which makes every proposed diff verifiable before it is applied. Structural actions (SPLIT, MERGE, RETIRE) keep the library routeable under a hard capacity bound. Every edit cites the Insights it absorbs, linking each Skill revision back to its evidence.

Validation and commit. Proposed plans are checked against two hard constraints: at most $C{=}5$ active Skills, and at most $L{=}4500$ characters per Skill. Only a valid final library is committed. Each committed edit records the Insights it absorbs.

Reviewed evidence and harness versions. The review state is recorded per Insight at its latest evidence revision. An accepted Skill update marks the Insights it absorbs as reviewed, preventing the same unchanged evidence from immediately triggering another update. A deliberate NOOP also completes a review but changes neither the active library nor its version. If a reviewed Insight later gains new evidence, its new revision can be considered in a subsequent update. Each committed library change defines a new harness version.

Deployment-time use. At inference the agent routes each new request to the relevant Skills once, before the first generation. Generation calls under the evolving library are recorded with their harness and generator version annotations and become the prompt–image pairs available to D-OPCD.

C.4 EXPERIMENTAL CONFIGURATION AND FROZEN REPLAY

Experimental configuration. GPT-5.6 Terra with medium reasoning effort serves as the agent's base reasoning model, the Insight and Skill managers, and the in-task verifier. Table 8 reports settings shared across benchmarks; Table 9 reports benchmark-specific ASE settings. Evaluation leaves the agent state unchanged, so it cannot affect subsequent predictions or the adaptation stream.

Table 8: Harness-evolution settings shared across benchmarks.

Component	Parameter	Value
Agent loop	max verification questions	10
Agent loop	max generation attempts	5
Episode store	LLM summary	disabled
Insight extraction	task batch size M	8
Insight retrieval	text embedder	BGE-M3
Insight retrieval	top-k / semantic gate	4 / 0.35 cosine
Insight consolidation	duplicate threshold	0.92 cosine
Skill library	max active Skills C	5
Skill library	max Skill length L	4500 characters

Table 9: Benchmark-specific ASE settings during evolution on $\mathcal{Q}_{\text{train}}$. The review interval is measured in task batches.

Parameter	GenEval	GenEval2	WISE	R2I-Bench
Insight support τ	2	4	4	4
Min. support batches	2	3	3	3
Skill trigger B	3	8	8	8
Min. review interval	4	6	6	6

Generation seeds. For task i, generation attempt j uses seed $s_i + j - 1$, where s_i is the task's base seed.

Selecting and freezing a skill library. Library selection is performed on $\mathcal{Q}_{\text{hold}}$, independently of evaluation on $\mathcal{Q}_{\text{eval}}$. Each candidate is evaluated without persistent updates; the selected library is subsequently frozen for replay on the entire $\mathcal{Q}_{\text{train}}$ and for evaluation on $\mathcal{Q}_{\text{eval}}$. The replayed tasks do not update the Skill library or the persistent Insight store. Table 10 identifies which Skill-update snapshot was selected in each evolution round and how many tasks had contributed experience to that snapshot. The skill-free harness $\mathcal{H}_\emptyset$ uses the empty library and requires no version selection.

Prompt-pair collection. For each query $q_i \in \mathcal{Q}_{\text{train}}$, frozen replay returns (p_i, I_i) according to Equation 10. We retain the submitted record (q_i, p_i, I_i); all compared training methods drop tasks whose prompt equals the query after normalization, following the context-data protocol in Appendix D.1.

C.5 Prompt Templates

C.5.1 Agent Execution Prompts

Figures 4–9 show the prompts used by the agent during each generation task in execution order. Braced capitalized fields denote runtime substitutions, and line wrapping is for presentation only. The optional cross-task Insight field in the refinement template is used during evolution and left empty in frozen skill-only replay and evaluation.

C.5.2 Harness Evolution Prompts

The Insight and Skill managers use separate prompts to update cross-task knowledge. Figures 10 and 11 show their core decision instructions and runtime inputs. The operation schemas are given in Tables 6 and 7, rather than repeated in the prompt figures.

Prompt 1 — Skill routing

```
You are an image-generation Skill Router. Every available Skill is an
 initial-prompt rewriter that may run exactly once before the first
image generation. Choose the subset of candidate Skills whose
transformations are applicable and useful for the original request.
You may combine multiple complementary Skills or choose none.

### Candidate Initial-Prompt Skills
{SKILL_MANIFEST}

### Original User Request
{ORIGINAL_QUERY}

Return ONLY a JSON array containing the selected SKILL_ID strings in
the order they should be considered. Return [] when no Skill is
useful.
```

Figure 4: **Skill routing prompt.** Selects applicable Skills from their manifest before the first generation attempt.

Prompt 2 — Initial prompt rewrite

```
Produce the one prompt used for the first image-generation attempt.
Preserve every requested fact. Apply all selected initial-prompt
Skill instructions together as one coherent transformation. Combine
compatible guidance according to the original request and return only
 the resulting prompt.

### Initial-Prompt Skill Instructions
{SELECTED_SKILL_INSTRUCTIONS}

### Original Prompt
{ORIGINAL_QUERY}

Return ONLY the final enhanced prompt.
```

Figure 5: **Initial prompt rewrite.** Combines selected Skill instructions with the original query.

Prompt 3 — Requirement decomposition

```
Analyze the user's image generation prompt and break it into specific
 visual requirements. For each requirement, write a question
answerable with yes or no. The questions must verify whether the
requirement is present in an image. Produce at most 10 independent
questions. Prioritize the most important requirements and constraints
 that an image generator is likely to miss.
Respond ONLY with a JSON array of strings.

USER PROMPT:
{ORIGINAL_QUERY}
```

Figure 6: **Requirement decomposition.** Converts the original query into visual yes/no checks.

Prompt 4 — Batched image verification

```
Image: <image>
Evaluate every verification question below against the provided image
. Return ONLY one JSON array with exactly the same number of items
and in the same order as the questions. Each item must be an object
with an 'answer' string and a 'passed' JSON boolean. Use true only
when the image clearly satisfies the question; otherwise use false.
Do not use markdown or add commentary outside the JSON array.

VERIFICATION QUESTIONS:
1. {QUESTION_1}
2. {QUESTION_2}
...
```

Figure 7: **Batched image verification.** Checks an image against all visual requirements in one request.

Prompt 5 — Task-local experience summary

```
Task: Summarize the experience of the current image generation
attempt.

--- CURRENT ATTEMPT ---
Prompt used: {CURRENT_PROMPT}
Passed requirements: {PASSED_REQUIREMENTS}
Failed requirements: {FAILED_REQUIREMENTS}
Reasoning/Thought before generation: {CURRENT_THOUGHT}
Image: <image>

--- PREVIOUS EXPERIENCES ---
{PREVIOUS_TASK_LOCAL_EXPERIENCES}

--- ANALYSIS ---
Based on the image, internal verification results, thought process,
and prior attempts from this task, summarize what worked, what failed
, and what should be changed in the next attempt. Keep it under 100
words. Do not include an introduction.
```

Figure 8: **Task-local experience summary.** Records lessons from the current attempt and earlier attempts on the same task.

Prompt 6 — Feedback-based refinement

```
Task: Refine the image generation prompt after failed internal checks
.

ORIGINAL INTENT:
{ORIGINAL_QUERY}

--- RELEVANT CROSS-TASK INSIGHTS ---
{RELEVANT_INSIGHTS_IF_ENABLED}

--- ATTEMPT HISTORY ---
{ATTEMPT_HISTORY}

--- REQUIREMENTS ---
1. Reinforce requirements that failed in the latest attempt.
2. Preserve requirements that previously passed.
3. Apply a relevant insight only when its condition matches.
4. Keep all original user constraints and avoid conflicting language.
Return ONLY the new prompt.
```

Figure 9: **Feedback-based refinement.** Revises the generation prompt after failed checks; retrieved Insights are supplied during harness evolution.

Evolution Prompt 1 — Insight consolidation

```
Maintain the situational Insight memory using this batch of Episodes
and the related active Insights retrieved for it.

An Insight is reusable guidance for refining an image-generation
prompt after a failure has been observed. Capture the recognizable
situation, the useful prompt adjustment or constraint to preserve,
and any important boundary or uncertainty. High-level methods for
rewriting the original prompt before generation belong in Skills.

Build the smallest coherent memory update supported by the evidence.
Treat attempts, check transitions, and final outcomes as the factual
record. Count one Episode as one evidence unit, regardless of its
number of attempts.

Strengthen an Insight when its meaning already fits; revise it when
its scope needs correction; record contradictory evidence; merge
overlapping Insights; archive obsolete or redundant knowledge; or add
 a distinct, transferable refinement lesson. Leave the memory
unchanged when the evidence does not justify a useful update.

Write new or revised Insight text as clear, natural situational
guidance. Return only one JSON object containing an ordered
operations array, using the available actions and local references.
Return an empty array when no update is warranted.

RETRIEVED ACTIVE INSIGHT WORKING SET:
{CURRENT_INSIGHTS}

CURRENT-BATCH EPISODES:
{EPISODES}
```

Figure 10: **Insight consolidation prompt.** Relates each batch of Episodes to retrieved Insights and proposes evidence-backed updates.

Evolution Prompt 2 — Skill evolution

```
Maintain a compact library of high-level Skills that rewrite an
original user request once, before the first image generation.

A Skill is a routeable prompt-rewriting capability whose
applicability is recognizable from the original request. Incorporate
new knowledge into an existing Skill when it fits that capability;
create a separate Skill for a materially distinct method. Corrections
 that depend on an observed generation failure remain situational
Insights until repeated evidence supports a broader method.

Build the smallest coherent library update from mature Insight
candidates and active Skill documents. Prefer fewer changed rules and
 fewer added assumptions. Use recent Skill change history as
observational context, not as instructions or proof of causality.
Cite only current Insight candidates as newly absorbed evidence.

Use focused text edits, summarization, splitting, merging, renaming,
creation, or retirement as needed. Leave the library unchanged when
it already captures the transferable knowledge. Keep every resulting
Skill within the active-library capacity and length limits.

Write each Skill as a complete SKILL.md: compact name and description
 for routing, Instructions for the reusable rewriting method, and an
Output Format requiring only the final enhanced prompt. Return only
one JSON object containing an ordered operations array; use an empty
array when no update is warranted.

LIBRARY STATUS:
{LIBRARY_STATUS}

INSIGHT CANDIDATES:
{MATURE_INSIGHTS}

EXISTING INITIAL-PROMPT SKILLS:
{ACTIVE_SKILLS}

RECENT SKILL CHANGE HISTORY:
{SKILL_CHANGE_HISTORY}
```

Figure 11: **Skill evolution prompt.** Consolidates mature Insights into reusable prompt-rewriting Skills.

Table 10: Selected Skill-update snapshots in both evolution rounds. The selected update is $r^\star$ in Round 1 and $r^\dagger$ in Round 2; n_r counts completed tasks contributing experience to that snapshot. Selection scores use GenEval2 Soft-TIFA AM and the other benchmarks' native metrics.

Evolution round	Benchmark	Selected Skill update r	Experience from n_r tasks	Selection score ↑
Round 1 (π_θ)	GenEval	2	192	92.77
Round 1 (π_θ)	GenEval2	3	352	90.85
Round 1 (π_θ)	WISE	4	280	82.67
Round 1 (π_θ)	R2I-Bench	4	216	69.60
Round 2 ($\pi_{\theta'}$)	GenEval	2	240	92.77
Round 2 ($\pi_{\theta'}$)	GenEval2	1	128	90.40
Round 2 ($\pi_{\theta'}$)	WISE	3	272	82.00
Round 2 ($\pi_{\theta'}$)	R2I-Bench	4	328	68.44

D Implementation Details of D-OPCD and Baseline Training

D.1 Training Data Preparation

Frozen replay of the selected skill-equipped harness $\mathcal{H}_{r^\star}$ produces records (q_i, p_i, I_i) on $\mathcal{Q}_{\text{train}}$. All compared training methods use the same subset for which p_i differs from q_i after Unicode normalization, whitespace collapse, and case folding. D-OPCD trains from (q_i, p_i); I_i determines record eligibility but is not a loss target. Diffusion-DPO additionally pairs the submitted image I_i with a rejected image generated directly from q_i using the same seed. Vanilla SFT and D-OPSD train from the corresponding (q_i, I_i) records. The per-benchmark counts and filtering yields are reported in Appendix E.2.

D.2 D-OPCD Implementation

The student receives q_i. For the teacher, the text encoder receives q_i followed by a blank line, the label `Privileged generation context:`, and p_i. The student and teacher use the same frozen text encoder, with maximum sequence lengths of 512 and 1024 tokens, respectively. Starting from Gaussian noise, the student follows a four-point Euler rollout at $t \in \{0, 0.1, 0.25, 0.5\}$, with its final update ending at $t = 1$. The state is detached before each velocity comparison, so gradients update the student at visited states without backpropagating through earlier rollout steps. An EMA copy of the student, with decay 0.9999, supplies the teacher prediction under the additional context. We minimize the mean squared velocity difference across the four states, as defined in Section 3.2; the teacher prediction is stop-gradient.

D.3 Baseline Objectives

Vanilla SFT. The student receives q_i, while the agent-submitted image I_i supplies the clean image latent for the standard flow-matching target. Each batch samples a noise level from a logit-normal distribution with logit mean zero and standard deviation one. No privileged prompt enters the model.

Diffusion-DPO. Both the chosen image I_i and the same-seed rejected image are trained under q_i. A frozen copy of the initial generator supplies reference losses, and the chosen–rejected difference in model and reference flow-matching losses forms the preference objective with $\beta = 2500$. Two logit-normal noise levels are sampled per pair.

D-OPSD. The student receives q_i and visits states along its own sampling trajectory. The EMA teacher, updated with decay 0.9999, receives an embedding of (q_i, I_i) from Qwen3-VL-4B-Instruct and supplies a detached clean-output endpoint target. This baseline shares SFT's image-paired records, but uses on-policy student states rather than the image-derived states used by SFT.

Table 11: Generator checkpoints and held-out scores by method and benchmark. All methods use records from $\mathcal{H}_{r^\star}$; scores use GenEval Overall, GenEval2 Soft-TIFA AM, WIScore, and R2I-Score (0–100).

Benchmark	Step	Hold score
D-OPCD		
GenEval	1800	86.31
GenEval2	1600	82.38
WISE	1000	48.67
R2I-Bench	400	49.33
Vanilla SFT		
GenEval	1000	82.98
GenEval2	1600	82.13
WISE	200	50.67
R2I-Bench	600	51.30
Diffusion-DPO		
GenEval	200	79.29
GenEval2	1400	79.49
WISE	1800	49.33
R2I-Bench	1200	47.87
D-OPSD		
GenEval	1200	79.64
GenEval2	1800	75.29
WISE	400	51.33
R2I-Bench	200	49.05

D.4 Optimization and Checkpoint Selection

All methods start from Z-Image-Turbo and train LoRA adapters on its transformer feed-forward projections and attention query, key, value, and output projections. The common settings are rank 64, scale parameter $\alpha = 128$, zero LoRA dropout, 1024-pixel square images, and seed 42. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, zero weight decay, a constant learning rate of 10^{-4} without warmup, gradient clipping at norm 1, and bfloat16 precision. Training runs for 2000 optimizer steps and saves a checkpoint every 200 steps. Direct evaluation uses eight sampling steps and guidance scale zero.

Compute resources. Model training and image-generation inference used NVIDIA A100 GPUs (80 GB each). Training jobs allocated four GPUs; inference used one GPU per generation worker, with multiple workers run in parallel where applicable.

Each saved checkpoint is evaluated on $\mathcal{Q}_{\text{hold}}$; the benchmark-specific selection metrics are defined in Appendix E. Table 11 lists each benchmark and compared method using skill-equipped harness data, with the reported optimizer step and held-out score. Hold scores use the same benchmark-native metric used for checkpoint evaluation and for the main evaluation tables.

E Experimental and Evaluation Details

E.1 Task Set Allocations

Table 12 gives the disjoint task sets used for harness evolution, selection, and final evaluation. GenEval, GenEval2, and WISE use approximately 50/15/35 allocations; R2I-Bench uses a 780-task experimental subset with its supplied split.

Table 12: In-distribution task allocations. Ratios give the train/hold/evaluation shares of the task pool actually used.

Benchmark	Pool	$\lvert\mathcal{Q}_{\text{train}}\rvert$	$\lvert\mathcal{Q}_{\text{hold}}\rvert$	$\lvert\mathcal{Q}_{\text{eval}}\rvert$	Ratio (%)
GenEval	553	276	83	194	49.9/15.0/35.1
GenEval2	800	400	120	280	50/15/35
WISE Verified	1000	500	150	350	50/15/35
R2I-Bench subset	780	480	100	200	61.5/12.8/25.6

Table 13: Replay-record yields from the selected skill-equipped harness $\mathcal{H}_{r^\star}$. The last column gives the prompt-changed tasks used by all four compared training methods.

Benchmark	Source	Records	$p_i = q_i$	Changed
GenEval	$\mathcal{H}_{r^\star}$	276	0	276
GenEval2	$\mathcal{H}_{r^\star}$	400	0	400
WISE	$\mathcal{H}_{r^\star}$	500	10	490
R2I-Bench	$\mathcal{H}_{r^\star}$	480	88	392

The first three splits use seed 42; GenEval2 and WISE preserve earlier task assignments, while R2I-Bench uses its supplied split.

E.2 Replay Data Construction and Counts

Frozen replay on $\mathcal{Q}_{\text{train}}$ yields one submitted (q_i, p_i, I_i) record per task. Table 13 reports the records retained before and after excluding normalized $p_i = q_i$ pairs. D-OPCD, Diffusion-DPO, Vanilla SFT, and D-OPSD use the prompt-changed tasks, with method-specific supervision. The replay uses the selected skill-equipped harness version in Table 10.

E.3 Context Effectiveness and Diversity Diagnostics

Improvement-filtered pairs. The default set retains records from the selected skill-equipped harness when the submitted prompt differs from the original query after normalization. The improvement filter instead starts from all saved records, including unchanged prompts. For each task, we pair the saved image from the harness's selected attempt with the first image generated directly from q_i. We take the requirement checklist $\mathcal{C}_i$ from the selected harness attempt and score both saved images against that same checklist using the harness's internal verifier. Previously recorded answers are reused when the checklist matches; otherwise, the direct-query image is verified against $\mathcal{C}_i$. If $V_i(I)$ is the number of passed checks for image I, we retain (q_i, p_i) iff $V_i(I_i^p) > V_i(I_i^q)$. Missing or invalid verifier responses are excluded. This comparison requires neither new image generation nor external benchmark scores. Since the saved images may use different seeds, the filter selects observed verifier improvements rather than estimating a seed-controlled prompt effect. It selects 43, 264, 319, and 314 pairs on GenEval, GenEval2, WISE, and R2I-Bench, respectively. Three R2I-Bench pairs have $p_i = q_i$ but satisfy the image-level criterion.

Diversity-adapted sampling. When the Skill router selects applicable Skills, the default harness combines their guidance with q_i in one initial prompt rewrite. To broaden the contexts produced by the same Skills, the diversity adapter acts at this step, before the first image-generation attempt. It samples one of four expression strategies at random for each task with selected Skills. *Local edit* preserves the query's wording and order where possible while adding relevant guidance. *Restate* rewrites the request in new words. *Integrate* weaves guidance into a single coherent image description. *Separate* states the original request first and places guidance in separate sentences. All four instruct the rewriter to preserve the task requirements rather than create variety by changing the requested content. Table 14 gives the exact strategy instructions, and Figure 12 shows the shared rewrite prompt that receives q_i, the selected Skill instructions, and the sampled strategy.

If no Skill is selected, the first attempt uses q_i directly. The Skill library, router, internal verifier, and later refinement steps remain the same, with at most five generation attempts per task. We record the prompt of the agent's final selected image as p_i, which need not be its first or last attempted prompt. After excluding normalized $p_i = q_i$ cases, the adapted replay yields 276, 400, 491, and 381 pairs on GenEval, GenEval2, WISE, and R2I-Bench. All three context sets use the same D-OPCD objective and student/$[q_i; p_i]$ teacher interface.

Table 14: Exact expression-strategy instructions inserted into the diversity-adapter rewrite template.

Mode	Instruction
`local_edit`	Keep the original request's wording and order where possible. Incorporate applicable Skill guidance through local edits or additions.
`restate`	Write the resulting prompt afresh in your own words rather than retaining the original sentence structure. Express the same request together with applicable Skill guidance.
`integrate`	Weave applicable Skill guidance directly into the description of the request, as one coherent image-generation prompt.
`separate`	State the original request first, then express applicable Skill guidance in separate sentences within the same final image-generation prompt. Do not output explanations of your work.

Diversity adapter — initial prompt rewrite

```
Produce the prompt for the first image-generation attempt. Use the
supplied Skills as guidance for fulfilling the original request.
Apply guidance only where it is relevant to that request, and
preserve the request's meaning and requirements. If a Skill conflicts
 with the original request, the request takes priority. Return only
the resulting prompt. Returning the original request unchanged is
allowed.

### Initial-Prompt Skill Instructions
{SELECTED_SKILL_INSTRUCTIONS}

### Original Prompt
{ORIGINAL_QUERY}

### Expression strategy
{STRATEGY_INSTRUCTION}
This strategy changes expression, not the requested content. It does
not authorize dropping applicable guidance or adding new requirements
 merely to create variety.

Return ONLY the final enhanced prompt.
```

Figure 12: **Diversity-adapter rewrite template.** The strategy instruction is chosen from Table 14; all other text is shared across modes.

Paired task effectiveness. For each retained (q_i, p_i), we generate one image from q_i and one from p_i using the same initial generator, seed, sampling settings, and task. Both images are scored against the original q_i and its benchmark annotations. Let $S_b(q)$ and $S_b(p)$ denote the benchmark-native aggregate scores for setting b on exactly its retained tasks, expressed on a 0–100 scale. Table 15 reports $\Delta_b = S_b(p) - S_b(q)$ in percentage points, $100\Delta_b/S_b(q)$ as a relative percentage, and the percentage of individual tasks whose p_i image scores strictly above its paired q_i image. GenEval uses Overall and GenEval2 uses Soft-TIFA AM. The same-seed external evaluation is distinct from the agent's internal-verifier criterion used for filtering.

Table 15: Effectiveness of the three training-pair settings on their retained train tasks. $S(q)$ and $S(p)$ use the original query as the scoring target: GenEval Overall, GenEval2 Soft-TIFA AM, WIScore, or R2I-Score, respectively. $\Delta = S(p) - S(q)$ is in percentage points; Rel. is $100\Delta/S(q)$, and Win is the percentage of individual pairs with a strictly higher p-image score. The external benchmark scorer evaluates both images in each pair.

Benchmark	Setting	N	$S(q)$	$S(p)$	Δ	Rel. (%)	Win (%)
GenEval	Default	276	73.76	85.71	+11.95	+16.20	19.57
	Improvement filter	43	12.50	78.99	+66.49	+531.94	69.77
	Diversity adapter	276	73.76	81.84	+8.08	+10.95	16.30
GenEval2	Default	400	77.43	85.74	+8.31	+10.73	70.75
	Improvement filter	264	73.37	85.35	+11.98	+16.33	78.79
	Diversity adapter	400	77.43	86.76	+9.33	+12.06	72.75
WISE	Default	490	43.74	80.00	+36.26	+82.89	38.78
	Improvement filter	319	29.92	79.91	+49.99	+167.12	51.72
	Diversity adapter	491	43.87	80.67	+36.80	+83.90	40.33
R2I-Bench	Default	392	42.58	77.17	+34.59	+81.23	75.00
	Improvement filter	314	37.87	76.43	+38.56	+101.83	79.62
	Diversity adapter	381	42.48	75.79	+33.31	+78.43	74.54

Table 16: Text diversity of the retained privileged contexts p_i. Normalized character Vendi and Distinct-1 are higher for more varied contexts; mean pairwise character TF–IDF cosine is lower. Pool N is the number of retained pairs, and Eval n is the equal-size subsample. Length is the mean number of words.

Benchmark	Setting	Pool N	Eval n	Char Vendi ↑	Char cosine ↓	Distinct-1 ↑	Length
GenEval	Default	276	43	0.7754	0.1519	0.1768	96.4
	Improvement filter	43	43	0.7864	0.1431	0.1767	116.7
	Diversity adapter	276	43	0.7841	0.1459	0.1817	83.5
GenEval2	Default	400	264	0.4914	0.1374	0.0483	189.3
	Improvement filter	264	264	0.4816	0.1386	0.0476	197.4
	Diversity adapter	400	264	0.5664	0.1094	0.0590	111.0
WISE	Default	490	319	0.7583	0.0796	0.1680	112.3
	Improvement filter	319	319	0.7339	0.0856	0.1592	122.9
	Diversity adapter	491	319	0.8133	0.0610	0.1940	84.7
R2I-Bench	Default	392	314	0.7668	0.0708	0.1183	143.7
	Improvement filter	314	314	0.7582	0.0721	0.1135	153.3
	Diversity adapter	381	314	0.7946	0.0632	0.1265	123.5

Context-text diversity. We jointly fit character 3–5-gram TF–IDF features to all three settings within each benchmark. We measure normalized character Vendi score (Friedman & Dieng, 2023) (Vendi divided by sample size), mean within-set pairwise cosine similarity, and corpus Distinct-1 (Li et al., 2016) (unique words divided by all words). Mean prompt length aids interpretation. To reduce the effect of cohort size, Table 16 uses equal-size subsamples, fixed by the smallest setting in each benchmark: 43/264/319/314 for GenEval/GenEval2/WISE/R2I-Bench. For larger settings, it averages 100 subsamples drawn with seed 42. These are text-diversity proxies rather than image-diversity scores. The R2I-Bench improvement-filtered pool includes three unchanged $p_i = q_i$ pairs; we retain them to match the model-training data.

A separate earlier matched-400 GenEval2 audit compared the default SkillOnly contexts with those from the diversity adapter, fitting TF–IDF jointly to those two corpora. It found normalized lexical Vendi scores of 0.7402 and 0.7921, respectively, and Distinct-2 values of 0.3533 and 0.3862. Because this two-setting word-level audit used a different feature space and sample protocol, its numbers are not inserted into Table 16.

Character n-grams probe repetition in the prompt wording that the adapter is designed to reduce. A word unigram–bigram audit agrees on GenEval2, WISE, and R2I-Bench, but its GenEval differences are small and mixed: normalized word Vendi is 0.9221 for the default and 0.9205 for the adapter, while Distinct-2 is 0.5469 and 0.5430, respectively.

E.4 Evaluation Metrics and Protocols

For in-distribution evaluation, each method generates one image per task on $\mathcal{Q}_{\text{eval}}$. We report each benchmark by its native metric under its official evaluation protocol. GenEval (Ghosh et al., 2023) uses the Overall score from its detector-based checker; GenEval2 (Kamath et al., 2025) uses prompt-level Soft-TIFA AM; WISE (Niu et al., 2025) uses WiScore under the WISE_Verified binary protocol; and R2I-Bench (Chen et al., 2025) uses R2I-Score from its checklist-based judge. Checkpoint candidates are evaluated on $\mathcal{Q}_{\text{hold}}$, with Overall score for GenEval, Soft-TIFA AM for GenEval2, WiScore for WISE, and R2I-Score for R2I-Bench. Reported hold scores use these same evaluation metrics.

T2I-CompBench++ (Huang et al., 2025) is evaluated separately on 2,100 prompts from the seven categories in Table 5 of its official *validation* split, using three images per prompt with seeds 42–44 and reporting scores by category. This OOD set is not $\mathcal{Q}_{\text{hold}}$ and is used neither for skill evolution nor for model or harness selection. The base generator is evaluated once, and each other row in Table 5 reports one source-specific D-OPCD checkpoint, without averaging across training sources.

F Use of AI Assistants

AI assistants were used in three roles. First, the experimental harness uses GPT-5.6 Terra to plan and refine generation prompts, verify images, and consolidate task experience into Insights and Skills (Appendix C). Second, outputs are scored with the benchmark-native automated evaluators described in Appendix E, including the checklist-based judge for R2I-Bench. Third, general-purpose AI writing tools were used to improve wording and readability. They were not used to originate the research questions, choose the experimental design, produce the reported results, or draw the paper's conclusions. The authors reviewed and approved the experimental design, analyses, results, and manuscript text.