BRIDGEPROMPT: BRIDGING THE IMPLICIT MEANING GAP IN TEXT-TO-IMAGE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-to-image (T2I) models can render explicit prompts well, but often fail when the desired image depends on meaning that is only implicit in the user's request, such as figurative meaning, entity knowledge, physical consequences, or scientific relations. We call this failure mode the *implicit visual meaning gap*. We view it as a conditioning-interface problem: the prompt wording may not make explicit the visual semantics that the generator must realize, even when those semantics are recoverable from the request. This motivates a prompt-side objective of making the intended visual meaning explicit before generation. We propose BRIDGEPROMPT, which keeps both the language prompter and T2I generator frozen and optimizes only a reusable natural-language instruction using feedback on generated images from a frozen vision-language evaluator. Candidate instructions are scored by how well their generated images match the original prompt, rather than by the rewritten text itself, while inference requires only one refinement and one generation. Conditioning the generator on benchmark-provided intended-meaning annotations raises the score to 82.3, supporting the view that many failures can be repaired by making missing semantics explicit before generation. With a frozen 1.5B prompter, BRIDGEPROMPT improves SD3.5 Medium on T2I-ReasonBench from 46.8 to 74.6, outperforms zero-shot GPT-4o refinement, and transfers without modification to other reasoning benchmarks and T2I models.

1 INTRODUCTION

Modern text-to-image (T2I) models can render explicit visual descriptions with high fidelity. Given a prompt such as "a red chair beside a wooden table," a modern generator usually places the requested objects correctly. The problem becomes harder when the desired image depends on meaning that is only indirectly specified. In such cases, the model may render the mentioned objects while missing the relation, consequence, or interpretation that determines the intended scene.

We call this failure mode the *implicit visual meaning gap*: the generated image reflects the prompt's explicit content but misses the visual meaning a human reader would infer. The gap is especially visible in idioms, entity references, and physical or scientific relations. We further observe that original prompts and intended-meaning refinements organize differently in the T2I model's text-conditioning space and language-model semantic space, forming more distinct regions in the former while remaining more interleaved in the latter.

This suggests a natural decision problem. Let the observed prompt p imply an unobserved visual intent z, and let r denote a candidate refinement. Ideally, we would choose the refinement whose generated images best realize the intended semantics of p. Searching independently over natural-language refinements for every input, however, would require repeated queries to a stochastic generator. Instead, we restrict refinement to a frozen language model controlled by a reusable instruction s, and ask whether optimizing this instruction using downstream image feedback can systematically expose the missing visual semantics.

We propose BRIDGEPROMPT, which keeps both the language prompter and T2I generator frozen and optimizes only this reusable instruction. During optimization, a frozen VLM evaluates generated images against the original prompt without seeing the refinement; at inference, the evaluator is removed and BRIDGEPROMPT performs only one refinement and one generation. Figure 1 illustrates the idea: "raining cats and dogs" is often rendered literally, while refinement exposes the intended

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

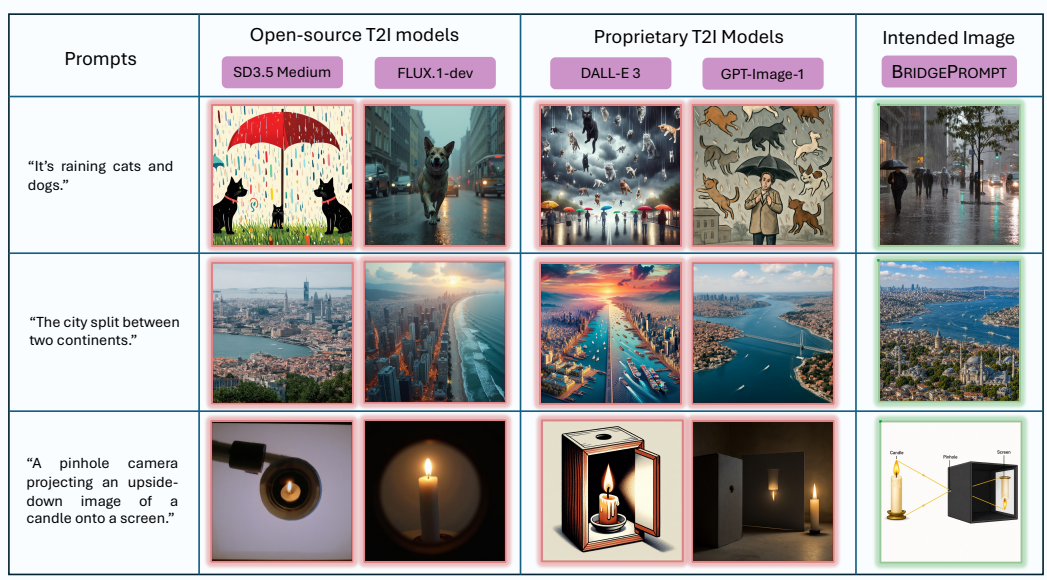

Figure 1: **The implicit visual meaning gap across open-source and proprietary T2I models.** Across SD3.5 Medium, FLUX.1-dev, DALL-E 3, and GPT-Image-1, generations are often realistic but semantically incorrect. The rightmost column shows the image generated from the BRIDGEPROMPT refinement, indicating that the failure is not rendering quality but recovering implicit visual meaning.

heavy-rain scene. On held-out reasoning benchmarks, BRIDGEPROMPT improves over raw prompting and zero-shot refinement and transfers without modification to other benchmarks and generators.

Contributions.

- **Implicit visual meaning gap.** We characterize the implicit visual meaning gap in text-to-image generation and analyze it through paired representation spaces: intended-meaning refinements shift the T2I model's text-conditioning representation while remaining semantically close in the frozen prompter's space.
- **Image-grounded instruction optimization.** BRIDGEPROMPT keeps both the T2I generator and language prompter frozen and optimizes only a reusable natural-language instruction. Candidate instructions are evaluated through their downstream generated images rather than the rewritten text, directly optimizing the visual consequence of prompt refinement.
- **Prompt optimization for T2I reasoning.** We show that optimized refinement can substantially improve reasoning fidelity on implicit prompts and transfer without modification to reasoning benchmarks and generators. We formulate T2I prompt optimization as an interface-level semantic grounding problem, where the goal is to reveal implicit visual intent to a frozen generator.

2 RELATED WORK

Text-to-image generation and reasoning-aware evaluation. Modern T2I systems combine large-scale image–text supervision with increasingly capable generative backbones and language conditioning. Latent diffusion enabled high-resolution generation in compressed latent spaces (Rombach et al., 2022), while Imagen and Parti demonstrated gains from stronger language modeling and larger-scale generative training (Saharia et al., 2022; Yu et al., 2022). DALL-E 3 further highlighted the importance of caption quality and prompt rewriting for improving prompt adherence (Betker et al., 2023). Alongside these advances, evaluation has progressed from image quality and coarse text–image alignment toward fine-grained prompt faithfulness. GenEval measures object-centric skills such as counting, color, and spatial relations (Ghosh et al., 2023); TIFA and VQAScore

evaluate whether generated images satisfy detailed prompt requirements using question answering or image-to-text generation (Hu et al., 2023; Lin et al., 2024); and ImageReward and PickScore model human preferences over generated images (Xu et al., 2023; Kirstain et al., 2023). More recent benchmarks explicitly test reasoning beyond original prompt matching. T2I-ReasonBench evaluates idiom, textual-image-design, entity, and scientific reasoning (Sun et al., 2026), while R2I-Bench and WISE evaluate broader reasoning, world knowledge, and semantic consistency (Chen et al., 2025; Niu et al., 2025). Our work focuses on failures in which the desired image depends on visual meaning that is only implicit in the original prompt.

Prompt refinement and reasoning for T2I. A growing line of work improves generation through prompt-side intervention. Promptist trains a language model to rewrite user prompts using supervised learning and reinforcement learning from image-level rewards (Hao et al., 2023), while Beautiful-Prompt similarly learns a prompt generator with visual-feedback optimization (Cao et al., 2023). OPT2I iteratively revises prompts to improve prompt–image consistency (Mañas et al., 2024), and Idea2Img uses multimodal feedback over repeated generation and prompt-revision cycles (Yang et al., 2024c). More recently, RePrompt trains a reasoning-augmented rewriter with image-level reinforcement learning (Wu et al., 2026). Complementary approaches incorporate reasoning inside the generation pipeline: GoT introduces explicit semantic–spatial reasoning before image synthesis (Fang et al., 2025), T2I-R1 jointly optimizes semantic- and token-level reasoning with reinforcement learning (Jiang et al., 2026), and Think-Then-Generate uses image-grounded reasoning before generation (Kou et al., 2026). In contrast, BRIDGEPROMPT trains neither the rewriter nor the generator: both remain frozen, and only a reusable natural-language instruction controlling the rewriter is optimized from downstream image feedback.

Prompt optimization. Automatic prompt optimization has been developed for language-model pipelines through textual gradients, iterative instruction search, evolutionary methods, and language-model program optimization (Pryzant et al., 2023; Yang et al., 2024b; Fernando et al., 2023; Khattab et al., 2023; Yuksekgonul et al., 2024; Opsahl-Ong et al., 2024; Agrawal et al., 2026). These methods provide alternative mechanisms for searching over natural-language instructions. BRIDGEPROMPT uses GEPA as the practical optimizer; our contribution is the T2I formulation in which candidate instructions are evaluated through their downstream generated images while both the prompter and generator remain frozen.

3 PROBLEM SETUP

We study reasoning-aware prompt refinement for text-to-image generation. Let $p \in \mathcal{P}$ denote the observed user prompt and let $z \in \mathcal{Z}$ denote its intended visual meaning. The variable z represents information that is inferable from the prompt but need not be stated explicitly, such as the figurative interpretation of an idiom, a physical consequence, an implied entity, or a scientific relation. We use z only as a conceptual variable; our method does not explicitly construct it or estimate $P(z \mid p)$. T2I-ReasonBench provides concrete instances of this setting: for idioms, entity references, and scientific prompts, the original prompt p may differ substantially from the visual meaning z that the generated image must realize (Sec. 4).

Given a candidate refinement r, a frozen T2I generator produces the following image:

$$x \sim G(\cdot \mid r). \tag{1}$$

Let $U(z, x) \in [0, 1]$ denote how well an image x realizes the intended visual meaning z. The expected utility of refinement r is as follows:

$$V(p, r) = \mathbb{E}_{z \sim P(z|p)} \mathbb{E}_{x \sim G(\cdot|r)}[U(z, x)]. \tag{2}$$

The ideal prompt-side repair would therefore be:

$$r^\star(p) = \arg\max_{r \in \mathcal{R}} V(p, r). \tag{3}$$

This formulation distinguishes semantic refinement from generic prompt expansion, namely, a useful refinement is one that changes the conditioning so that the frozen generator better realizes the intended meaning of the original request.

Equation 3 defines the ideal prompt-side objective, but BRIDGEPROMPT does not optimize r directly. Instead, we restrict the refinement to the output of a frozen language prompter M controlled by a reusable instruction s:

$$r_s(p) := M(p; s), \qquad x = G(r_s(p); \eta). \tag{4}$$

The prompter and generator remain fixed. The only optimized object is s. This converts the prompt-side repair problem into a black-box optimization problem over natural-language instructions.

4 DIAGNOSING THE IMPLICIT VISUAL MEANING GAP

We first ask whether failures on implicit prompts arise because the generator cannot render the intended scene, or because the intended visual content is not sufficiently exposed at the conditioning interface. Our analyses are consistent with the latter view; the embedding projections below are qualitative diagnostics rather than quantitative evidence of a universal representation gap.

We analyze T2I-ReasonBench (Sun et al., 2026), which contains 800 reasoning-focused prompts across idiom interpretation, textual image design, entity reasoning, and scientific reasoning. Idioms make the gap clearest, but the same structure appears across categories. "The final exam was a piece of cake" should not produce a cake; "a rubber duck and a metal ball in a water tank" should show the duck floating and the ball sinking; and "the city hosting the first modern Olympic Games" should resolve to Athens. In each case, the intended scene is specified only indirectly. Figure 2 visualizes this effect in SD3.5's CLIP text-encoder space. Although each refinement is paired with its original prompt and preserves the same underlying intent, the two often occupy noticeably different regions of the embedding space. This suggests that making the intended meaning explicit can substantially alter the conditioning representation presented to the generator, even when the semantic intent remains unchanged.

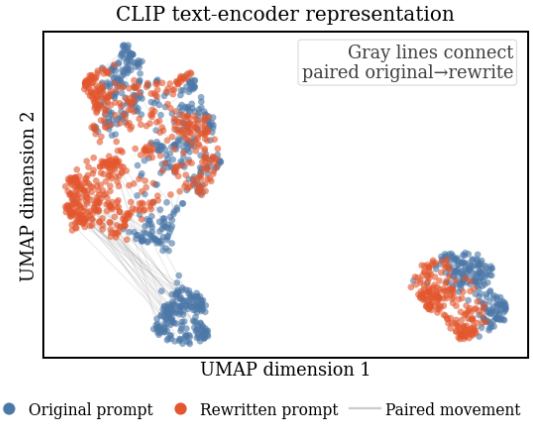

Figure 2: **SD3.5 CLIP text-encoder space.** Original prompts and intended-meaning refinements occupy visibly different regions in this projection of one component of SD3.5's text-conditioning stack. Gray lines connect paired originals and refinements.

Prompt-preserving inference-time controls leave the conditioning text unchanged. Best-of-50 searches over 50 samples from the original prompt, while the CFG sweep uses scales $\{3, 5, 7.5, 10, 12\}$. Both remain within one point of the published base score, whereas category-specific GPT-4o refinement reaches 68.7. We interpret this result narrowly: substantially increasing image-space search under the original conditioning recovers little of the missing performance, while changing the conditioning can produce much larger gains. For controlled comparisons in Sec. 6.1, all zero-shot rewriters instead use the shared instruction s_0; GPT-4o reaches 57.4 in this setting.

Table 1: Refinement vs. prompt-preserving controls; Base and GPT-4o rows from Sun et al. (2026).

Method	Generations	Score
Base model	1	46.8
Best-of-N	50	47.6
CFG sweep	5	47.1
GPT-4o(category-specific)	1	68.7

The same prompt pairs organize differently in a language-model semantic space. Figure 3 shows that originals and intended-meaning refinements are heavily interleaved, suggesting that their semantic relation remains accessible to the language model even when the CLIP representation separates them.

For example, "raining cats and dogs" can remain semantically close to heavy rain in language space while CLIP remains tied to the literal tokens "cats" and "dogs." This asymmetry motivates prompt refinement as a semantic bridge from implicit meaning to explicit visual instruction.

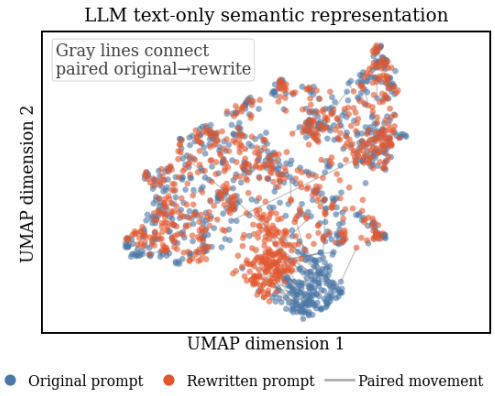

Figure 3: **Qwen2.5-1.5B semantic space.** Original prompts and intended-meaning refinements are heavily interleaved in the frozen prompter's representation space, consistent with the intended meaning being more accessible to the prompter than to SD3.5's CLIP text encoder.

Hidden-constraint recovery. We next test whether refinements expose missing semantics or merely make prompts longer. For each example i, let p_i be the original prompt, m_i the benchmark-provided intended meaning, and r_i the refined prompt. Let $C(x)$ denote the lemmatized content words in text x after removing stop words and generic visual terms. We define the hidden constraints as

$$H_i = C(m_i) \setminus C(p_i),$$

and, for prompts with non-empty H_i, measure recovery as

$$\mathrm{R}(i) = \frac{|H_i \cap C(r_i)|}{|H_i|}.$$

We average over prompts with non-empty H_i and restrict this analysis to idiom, entity, and scientific prompts; textual image design is excluded because its targets concern layout or typography rather than lexical content.

Table 2 uses generic visual expansion as a length control rather than a competing reasoning method: it is deliberately instructed not to infer hidden meaning. Despite producing long prompts, it recovers almost none of the hidden constraints, showing that prompt length alone does not explain the recovery achieved by reasoning-aware refinement. Reasoning-aware refinement instead recovers 40.1% of hidden constraints on average and improves over generic expansion on 66.9% of prompts, exposing visual evidence needed to render the intended scene. Together, these results support the view that intended semantics can remain accessible in language space while being poorly exposed in the T2I conditioning representation, and useful refinements recover missing constraints rather than merely adding visual detail. Changing the conditioning also produces larger gains than searching additional samples under the original prompt.

The remaining question is how to learn a refinement strategy that exposes this information consistently. Rather than searching over rewrites at inference time, BRIDGEPROMPT fixes the prompter and generator and optimizes only the reusable instruction controlling the prompter. We formulate this instruction search next.

Table 2: Hidden-constraint recovery.

Refinement	Words	Recovery	Improved
Generic expansion	43.2	0.2%	–
Reasoning-aware	56.0	40.1%	66.9%

5 BRIDGEPROMPT: IMAGE-GROUNDED INSTRUCTION OPTIMIZATION

BRIDGEPROMPT keeps the prompter and T2I generator frozen and optimizes only the instruction s:

$$r = M(p; s), \qquad x = G(r; \eta),$$

where η denotes sampling randomness. During optimization, a frozen VLM scores x against the original prompt p, and GEPA uses the resulting reward and textual feedback to revise s (Fig. 4).

Image-level reward. We evaluate candidate instructions through their downstream visual effect. Given the original prompt and generated image (p, x), a frozen VLM evaluator E scores three criteria: whether the image expresses the intended meaning of p, whether the required visual evidence is

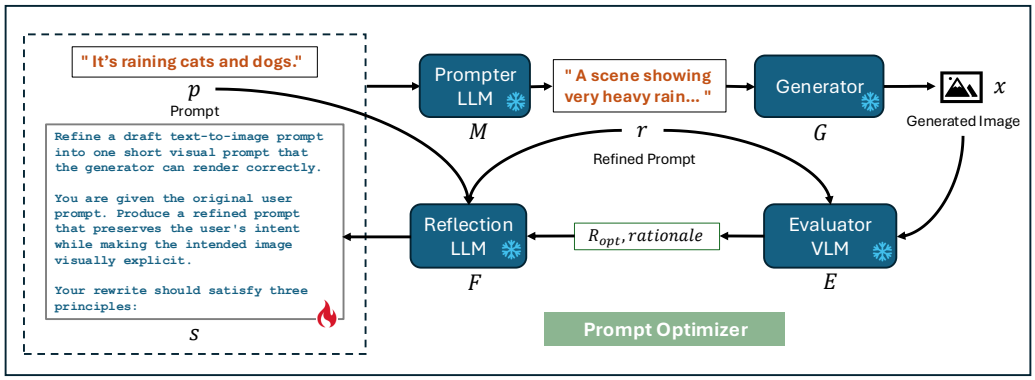

Figure 4: Overview of BRIDGEPROMPT. A frozen prompter LM refines the original prompt under instruction s, and a frozen T2I generator renders the refinement. During optimization, a frozen VLM evaluates the generated image against the original prompt. GEPA uses the resulting rewards and textual feedback to search over candidate instructions. At inference, only the optimized instruction, frozen prompter, and frozen generator are used.

present, and whether the scene is coherent. Each criterion is scored in $\{0, 0.5, 1\}$ and their mean defines the following reward:

$$R_{\text{opt}}(p, x) \in [0, 1]. \tag{5}$$

The evaluator sees the original prompt p and generated image x, but does not see the refined prompt r. The optimization therefore rewards the *visual consequence* of a refinement rather than whether the refinement itself appears linguistically convincing. In our experiments, E is Qwen2.5-VL-7B-Instruct; the evaluator prompt is provided in the Supplementary Sec. B.4.

Instruction objective. For an instruction s, define its empirical image-level reward as follows:

$$J(s) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{p_i \in \mathcal{D}_{\text{train}}} R_{\text{opt}}\left(p_i, G(M(p_i; s); \eta_i)\right). \tag{6}$$

where η_i denotes the sampling randomness used to generate the image for prompt p_i. The instruction optimization problem solved by BRIDGEPROMPT is therefore

$$s^\star = \arg\max_{s \in \mathcal{S}} J(s), \tag{7}$$

where $\mathcal{S}$ denotes the discrete space of candidate natural-language instructions.

Equation 7 is a point-estimation objective over a large discrete space of natural-language instructions. A complementary probabilistic view is useful for characterizing this reward landscape: instead of representing only its maximizer, we can assign greater probability mass to instructions with larger downstream image-level reward. This view connects instruction search to standard energy-based optimization and annealing, while remaining independent of the particular search algorithm used in practice.

5.1 ENERGY-BASED INTERPRETATION OF INSTRUCTION OPTIMIZATION

We formalize this probabilistic interpretation by assigning each instruction an energy equal to the negative of its image-level reward:

$$E(s) = -J(s). \tag{8}$$

For inverse temperature $\beta > 0$, consider the following unnormalized distribution:

$$\widetilde{\pi}_\beta(s) = \rho(s) \exp(\beta J(s)), \tag{9}$$

where $\rho(s)$ is a prior over plausible natural-language instructions. The corresponding normalized distribution is then:

$$\pi_\beta(s) = \frac{\rho(s)\exp(\beta J(s))}{Z_\beta}, \tag{10}$$

where Z_β is the partition function. The value of Z_β is generally intractable because the instruction space is combinatorial, but sampling methods such as Metropolis-Hastings only require the target density up to normalization (Andrieu et al., 2003).

This formulation exposes the connection between sampling and optimization. At low inverse temperature β, probability is spread across a broad set of candidate instructions. As β increases, probability increasingly concentrates on instructions with larger image-level reward $J(s)$. In the low-temperature limit, the distribution concentrates on maximizers of Eq. 7. For example, given a proposal distribution:

$$s' \sim q(s' \mid s), \tag{11}$$

a Metropolis–Hastings transition targeting Eq. 10 would accept the proposal with probability:

$$A(s \to s') = \min\left\{1, \frac{\rho(s')e^{\beta J(s')}q(s \mid s')}{\rho(s)e^{\beta J(s)}q(s' \mid s)}\right\}. \tag{12}$$

The partition function cancels from the ratio. With a uniform prior and symmetric proposal, this reduces to the following:

$$A(s \to s') = \min\left\{1, \exp\left[\beta\big(J(s') - J(s)\big)\right]\right\}. \tag{13}$$

This construction makes the exploration-exploitation trade-off explicit. At small β, many plausible instructions retain non-negligible probability, whereas increasing β concentrates mass around instructions with larger $J(s)$. In the low-temperature limit, the modes of π_β recover the maximizers of Eq. 7. Metropolis-Hastings and simulated annealing therefore provide canonical search procedures for this same reward landscape. We use this formulation as an interpretation of the optimization objective rather than as the implemented sampler.

Optimization with GEPA. The energy-based formulation characterizes the instruction-level reward landscape but does not prescribe our practical optimizer. We optimize Eq. 7 with GEPA (Agrawal et al., 2026), which uses execution traces and textual feedback to propose revised natural-language instructions and maintains a Pareto frontier over candidate performance. Unlike MCMC, GEPA does not sample from π_β or inherit its stationary-distribution guarantees; both formulations operate on the same reward $J(s)$, but GEPA provides a practical feedback-driven search procedure for the discrete instruction space.

Inference and evaluation. After optimization, $s^\star$ is fixed and inference requires only $r = M(p; s^\star)$ followed by $x = G(r; \eta)$. No evaluator, reflection LM, optimization, reranking, or test-time search is used. Final results use the official benchmark metric rather than R_{opt}.

6 Experiments

BridgePrompt keeps the prompter and T2I generator frozen, optimizes only the prompter instruction with image-level feedback, and performs one refinement before generation. We evaluate (i) held-out reasoning accuracy on T2I-ReasonBench, (ii) zero-shot transfer to other reasoning benchmarks, (iii) the effects of prompter scale and interface optimization, and (iv) transfer across generators.

6.1 Setup

The frozen T2I generator is Stable Diffusion 3.5 Medium (Esser et al., 2024), the prompter M is Qwen2.5-1.5B-Instruct (Yang et al., 2024a), and the optimization-time evaluator E is Qwen2.5-VL-7B-Instruct (Bai et al., 2025). Unless otherwise noted, our generated baselines and BridgePrompt use the same generator, sampling settings, and evaluation seeds. We optimize on T2I-ReasonBench (Sun et al., 2026), splitting each category (idiom, textual image design, entity, scientific; 200 prompts each) into 90/10/100 train/val/test prompts, and evaluate zero-shot transfer to

R2I-Bench (Chen et al., 2025) and WISE (Niu et al., 2025). Per-example category labels are never exposed to M, E, the reflection LM, or the optimizer. The manually written seed s_0 is nevertheless benchmark-informed: it names the broad reasoning types represented in T2I-ReasonBench without identifying the category of any individual example.

During optimization, E sees only (p, x) and scores intent, visual evidence, and coherence in $\{0, 0.5, 1\}$, whose mean defines $R_{\text{opt}} \in [0, 1]$ (§5). Final results use the official T2I-ReasonBench question-criterion evaluation scripts, which score images with Qwen2.5-VL-72B-Instruct, and are rescaled to $[0, 100]$. Thus, optimization and reporting use different evaluator checkpoints (7B vs. 72B) and different rubrics; evaluator sensitivity is analyzed in Sec. 6.3. The exact s_0 is provided in Supplementary Sec. B.3.

Implementation details. We optimize s with GEPA (Agrawal et al., 2026) using `max_full_evals= 1`, reflection minibatches of 3, and one optimization thread. For each reflection example, the frozen Qwen2.5-1.5B reflection LM receives the current execution trace, including the refinement, image-level reward, and evaluator rationale, rather than a scalar reward alone. Before candidate admission, a static validity filter rejects malformed or interface-breaking proposals (e.g., truncated instructions, broken output contracts, or requests for scores or category labels); it contains no benchmark examples or target-image semantics. Images use 1024×1024, CFG 4.5, and a 64-word refinement cap: 24 steps with seed 0 during compilation and 28 steps with seeds $\{0, 1, 2\}$ at evaluation. Full settings and $s^\star$ appear in the Supplementary.

6.2 Results

Held-out results on T2I-ReasonBench. Table 3 reports test-split accuracy. The published base generator score is 46.8. Zero-shot rewriting under s_0 improves the base, showing that exposing intended meaning helps but that a hand-written instruction does not specialize well to the generator. BridgePrompt reaches 74.6 overall, a $+27.8$-point gain over raw prompting and $+20.7$ over the same 1.5B prompter under s0. Gains are largest on idiom, entity, and scientific prompts ($+35.1$, $+33.5$, $+27.2$), where surface tokens most often omit the visual content the generator needs; textual image design improves by $+15.4$. Supplementary Fig. 6 shows representative held-out examples.

Table 3: T2I-ReasonBench test accuracy (400 prompts). Zero-shot baselines use shared s_0; the SD3.5 Medium (base) is from Sun et al. (2026). Parentheses show absolute gain.

Method	Idiom	Textual	Entity	Scientific	Overall
SD3.5 Medium (base)	34.4	58.0	44.8	49.9	46.8
+ Qwen2.5-1.5B zero-shot	51.2	52.0	56.5	55.8	53.9
+ GPT-4o zero-shot	54.8	55.1	60.2	59.5	57.4
+ BridgePrompt	**69.5** (↑35.1)	**73.4** (↑15.4)	**78.3** (↑33.5)	**77.1** (↑27.2)	**74.6** (↑27.8)

Transfer to other reasoning benchmarks. We apply $s^\star$ unchanged to R2I-Bench and WISE, neither seen during optimization. Figure 5a shows improvements from $0.47 \rightarrow 0.63$ on R2I-Bench and $0.45 \rightarrow 0.71$ on WISE, with scores of 0.58 and 0.69 for GPT-4o zero-shot. Thus, despite optimization on T2I-ReasonBench, the learned instruction retains utility on two unseen reasoning benchmarks with different evaluation protocols.

Human evaluation. As an evaluator-independent check, 13 annotators compare BridgePrompt and GPT-4o generations on 24 held-out prompts (312 judgments). BridgePrompt is preferred in 53.2% of judgments, GPT-4o in 34.9%, and neither over the base image in 11.9%. We treat these as descriptive proportions because prompts and annotators contribute repeated judgments; full details are in the Supplementary Sec. D.8.

6.3 Ablations on prompter and evaluator scale

Figure 5b,c studies prompter and evaluator sensitivity. Zero-shot rewriting changes only modestly with prompter scale (49.5–54.5 for Qwen-0.5B–7B; GPT-4o: 57.4), whereas optimizing the instruction for

(a) Benchmark transfer

Method	R2I	WISE
Base	0.47	0.45
Qwen-1.5B	0.55	0.48
GPT-4o	0.58	0.69
BRIDGEPROMPT	**0.63**	**0.71**

(b) Prompter scale

Model	Size	Score
No prompter	–	46.8
Qwen-0.5B	0.5B	49.5
Qwen-1.5B	1.5B	53.9
Qwen-3B	3B	54.1
Qwen-7B	7B	54.5
GPT-4o	Prop.	57.4
BRIDGEPROMPT	1.5B	**74.6**

(c) Evaluator scale

Evaluator	Size	Score
Qwen2.5-VL-3B	3B	67.9
Qwen2.5-VL-7B	7B	74.6
Qwen2.5-VL-72B	72B	**75.3**

(d) Generator transfer

Generator	Base	GPT-4o	BP
SDXL	38.4	43.7	**51.5**
Sana-1.5	43.8	55.2	**63.0**
FLUX.1-dev	47.4	59.9	**70.3**

Figure 5: **Transfer and scale analysis.** (a) Zero-shot transfer of the SD3.5-optimized $s^\star$ to unseen reasoning benchmarks. (b) Effect of prompter scale under the shared zero-shot instruction s_0. (c) Optimization-evaluator scale; all scores use the official 72B judge, which the 72B row also optimizes against. (d) Cross-generator transfer of the same $s^\star$ without further optimization.

the 1.5B prompter reaches 74.6. Changing the optimization-time evaluator yields 67.9, 68.4, 74.6, and 75.3 with Qwen2.5-VL-3B, GPT-4o-mini, Qwen2.5-VL-7B, and Qwen2.5-VL-72B, respectively. All final scores use the official Qwen2.5-VL-72B reporting protocol. These results show that successful instruction optimization is not restricted to a single optimization-time evaluator: all tested evaluators substantially improve over the unoptimized 1.5B prompter, including the cross-family GPT-4o-mini evaluator.

6.4 GENERALIZATION ACROSS GENERATORS

We apply the SD3.5-optimized $s^\star$ unchanged to SDXL (Podell et al., 2023), Sana-1.5 (Xie et al., 2025), and FLUX.1-dev (Black Forest Labs, 2024), which differ substantially in backbone and text-conditioning architecture. Figure 5d shows gains of 13.1, 19.2, and 22.9 points over their respective base generators, with BRIDGEPROMPT also exceeding GPT-4o zero-shot on all three. On Qwen-Image, the same transferred $s^\star$ improves overall performance from 67.5 to 73.5 without recompilation. Generator-specific optimization further raises the score to 80.5, while the category-wise results reveal that some generator-specific adaptation remains beneficial (Supplementary Sec. D.7).

7 CONCLUSION

We study the *implicit visual meaning gap*, where T2I models miss visual content that is only indirectly specified by the prompt. Our analyses suggest that many such failures arise at the conditioning interface: making the intended semantics explicit can substantially improve a frozen generator. BRIDGEPROMPT operationalizes this idea by optimizing a reusable instruction with downstream image feedback while keeping both the prompter and generator frozen. With a 1.5B prompter, it improves SD3.5 Medium from 46.8 to 74.6 on T2I-ReasonBench and transfers to unseen benchmarks and heterogeneous generators. These results support prompt-side semantic repair as a lightweight way to improve reasoning fidelity without modifying the generator.

Limitations. BRIDGEPROMPT relies on a VLM for optimization-time feedback, and performance varies with evaluator choice, although gains persist across evaluator scales and with a cross-family GPT-4o-mini evaluator. Our human evaluation provides complementary evaluator-independent evidence but remains modest in scale. Future work can further study alternative optimization signals, broader prompt-rewriting baselines, and settings where generator limitations or ambiguous intent constrain prompt-side repair.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology, experimental design, and interpretation of results, and to revise the manuscript for readability and format figures and tables. LLM/VLMs are also components of the method itself, as described in Sections 5-6. All AI-assisted work was reviewed and verified by the authors, who take full responsibility for the final content, claims, and reported results.

REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce BRIDGEPROMPT throughout the main paper and Supplementary. Section 5 specifies the optimization objective, image-level reward, energy-based formulation, and practical GEPA optimization, while Section 6.1 describes the models, data splits, evaluation protocol, and main experimental settings. The Supplementary provides the complete hyperparameter configuration, seed and optimized instructions, evaluator and reflection prompts, instruction guard, inference-time baselines, embedding and hidden-constraint analyses, optimization procedure, and robustness experiments. These details specify the training, inference, and evaluation procedures used for the reported results.

ETHICS STATEMENT

Our human evaluation involved annotators assessing generated images for prompt adherence. Participation was voluntary, and no sensitive personal information was collected.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, Wesam Manassra, Prafulla Dhariwal, Casey Chu, Yunxin Jiao, and Aditya Ramesh. Improving image generation with better captions. *Technical report, OpenAI*, 2023.

Black Forest Labs. FLUX. `https://github.com/black-forest-labs/flux`, 2024.

Tingfeng Cao, Chengyu Wang, Bingyan Liu, Ziheng Wu, Jinhui Zhu, and Jun Huang. Beautifulprompt: Towards automatic prompt engineering for text-to-image synthesis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1–11, 2023.

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 *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12595–12630, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.636.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Rongyao Fang, Chengqi Duan, Kun Wang, Linjiang Huang, Hao Li, Shilin Yan, Hao Tian, Xingyu Zeng, Rui Zhao, Jifeng Dai, et al. Got: Unleashing reasoning capability of multimodal large language model for visual generation and editing. *arXiv preprint arXiv:2503.10639*, 2025.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

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*, volume 36, 2023.

Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:66923–66939, 2023.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A. Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. *arXiv preprint arXiv:2303.11897*, 2023.

Dongzhi Jiang, Ziyu Guo, Renrui Zhang, Zhuofan Zong, Hao Li, Le Zhuo, Shilin Yan, Pheng-Ann Heng, and Hongsheng Li. T2i-r1: Reinforcing image generation with collaborative semantic-level and token-level cot. *Advances in Neural Information Processing Systems*, 38:39856–39890, 2026.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Siqi Kou, Jiachun Jin, Zetong Zhou, Ye Ma, Yugang Wang, Quan Chen, Peng Jiang, Xiao Yang, Jun Zhu, Kai Yu, and Zhijie Deng. Think-then-generate: Reasoning-aware text-to-image diffusion with llm encoders. *arXiv preprint arXiv:2601.10332*, 2026.

Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *Computer Vision – ECCV 2024*, pp. 366–384. Springer, 2024.

Oscar Mañas, Pietro Astolfi, Melissa Hall, Candace Ross, Jack Urbanek, Adina Williams, Aishwarya Agrawal, Adriana Romero-Soriano, and Michal Drozdzal. Improving text-to-image consistency via automatic prompt optimization. *arXiv preprint arXiv:2403.17804*, 2024.

Yuwei Niu, Munan Ning, Mengren Zheng, Bin Lin, Peng Jin, Jiaqi Liao, Kunpeng Ning, Bin Zhu, and Li Yuan. Wise: A world knowledge-informed semantic evaluation for text-to-image generation. *arXiv preprint arXiv:2503.07265*, 2025.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, 2024.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.494.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022. doi: 10.1109/CVPR52688.2022.01042.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Kaiyue Sun, Rongyao Fang, Chengqi Duan, Xian Liu, Aoxue Li, and Xihui Liu. T2I-ReasonBench: Benchmarking reasoning-informed text-to-image generation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8919–8944, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.433. URL https://aclanthology.org/2026.findings-acl.433/.

Mingrui Wu, Lu Wang, Pu Zhao, Fangkai Yang, Jianjin Zhang, Jianfeng Liu, Yuefeng Zhan, Weihao Han, Hao Sun, Jiayi Ji, et al. Reprompt: Reasoning-augmented reprompting for text-to-image generation via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 14030–14057, 2026.

Enze Xie, Junsong Chen, Yuyang Zhao, Jincheng Yu, Ligeng Zhu, Chengyue Wu, Yujun Lin, Zhekai Zhang, Muyang Li, Junyu Chen, Han Cai, Bingchen Liu, Daquan Zhou, and Song Han. SANA 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. *arXiv preprint arXiv:2501.18427*, 2025.

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. *arXiv preprint arXiv:2304.05977*, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024b.

Zhengyuan Yang, Jianfeng Wang, Linjie Li, Kevin Lin, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. Idea2img: Iterative self-refinement with gpt-4v for automatic image design and generation. In *European conference on computer vision*, pp. 167–184. Springer, 2024c.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

A Qualitative Examples

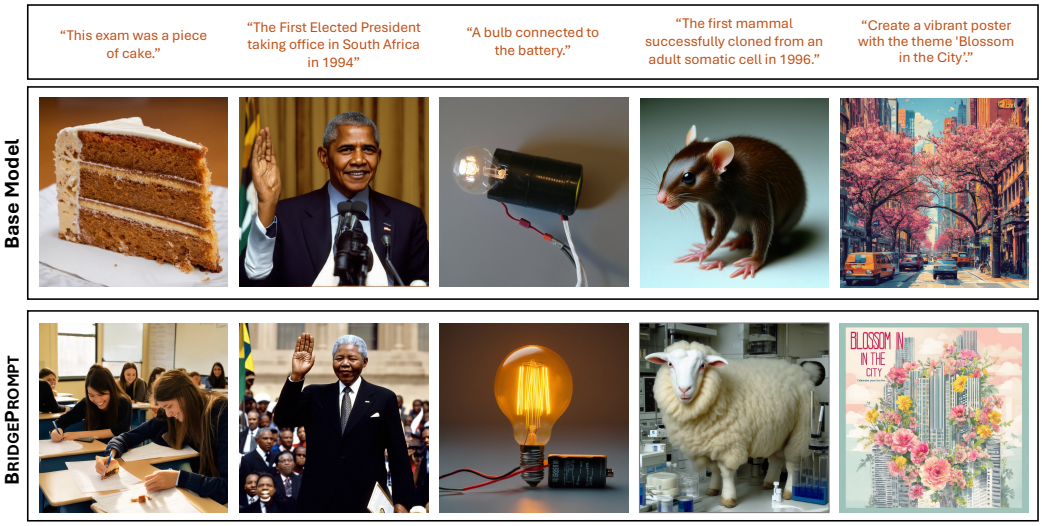

Figure 6: Qualitative comparison on five held-out T2I-ReasonBench prompts spanning idiom, entity, scientific, and textual-image-design categories. *Top:* SD3.5 Medium conditioned on the original prompt p. *Bottom:* the same frozen generator conditioned on $r = M(p; s^\star)$ from BridgePrompt. The base model often follows surface cues rather than the intended visual meaning, while the optimized refinement makes the intended scene explicit without modifying the generator.

B Implementation Details

B.1 Hyperparameters

Table 4 lists the configuration used for our generated baselines and BridgePrompt in Table 3. These runs share the same generator, sampler, evaluation seeds, and decoding budget; the SD3.5 Medium base row is the published benchmark result.

B.2 Compute and inference cost

A single GEPA compile occupies one A100 80GB GPU and takes approximately 6 hours. Test-set evaluation on the full T2I-ReasonBench test split takes approximately 1 hour per generator. The full set of experiments, including baselines, ablations, and cross-generator transfer, totals approximately 200 A100-hours.

The evaluator VLM and reflection LM are used only during compilation. At inference, BridgePrompt uses the frozen 1.5B prompter for one prompt rewrite, followed by one generator run. Thus, it does not require image reranking or VLM evaluation at inference.

The compilation cost is amortized when the optimized instruction is reused across prompts and benchmarks. As shown in §D.4, increasing the tested optimization budget by approximately 3–4$\times$ changes the idiom score by only about one to two points.

B.3 Seed instruction s_0

Rather than exposing dataset category labels to the rewriter, we initialize GEPA from a single fused instruction that covers all prompt types. This instruction was manually written to combine the four recurring failure modes in T2I-ReasonBench: figurative meaning, entity/reference resolution, scientific or causal reasoning, and text-rich visual design. The exact seed instruction is:

Table 4: Configuration for the headline T2I-ReasonBench result.

Component	Setting	Value
Generator	Model	Stable Diffusion 3.5 Medium
	Resolution	1024×1024
	Guidance scale (CFG)	4.5
	Compile-time steps / seeds	24 / $\{0\}$
	Eval-time steps / seeds	28 / $\{0, 1, 2\}$ (averaged)
	Max sequence length	256
Prompter M	Model	Qwen2.5-1.5B-Instruct
	Decoding	Greedy ($T = 0$)
	Max refined-prompt length	64 words
	Output contract	`{"refined_prompt": "..."}`
Optimization evaluator E	Model	Qwen2.5-VL-7B-Instruct
	Decoding	Greedy, `max_new_tokens`=512
	Protocol	three-axis describe-then-score
Reporting evaluator	Model	Qwen2.5-VL-72B-Instruct
	Decoding	Official script defaults, `max_new_tokens`=1000
	Protocol	Official T2I-ReasonBench question-criterion scripts
Reflection LM F	Model	Qwen2.5-1.5B-Instruct
	Wrapper	prose-only system prompt (§B.5)
GEPA	`max_full_evals`	1
	Reflection minibatch size	3
	Threads	1
	Train / val / test per category	90 / 10 / 100
	Candidate acceptance	strict improvement on minibatch
	Selection criterion	best validation R_{opt}
Splits	T2I-ReasonBench prompts/category	200
	Categories	idiom, design, entity, scientific
	Random seed	42

Table 5: Inference-time cost relative to prompt-preserving search baselines.

Method	Generator runs	Additional inference-time model use	Δ vs. reported base
Base model	1	None	0.0
Best-of-50	50	VLM selection	+0.8
CFG sweep	5	VLM selection	+0.3
BRIDGEPROMPT	1	One 1.5B prompt rewrite	+27.8

```
Refine a draft text-to-image prompt into one short visual
prompt that the generator can render correctly.

You are given the original user prompt.  Produce a refined
prompt that preserves the user's intent while making the
intended image visually explicit.

Your rewrite should satisfy three principles:

(1) Intent recovery.  Identify what image the user actually
wants, not just the surface wording.  If the prompt contains
figurative language, indirect references, implied relations,
causal effects, or background knowledge, make the intended
visual scene explicit.  If the prompt is already literal
and complete, preserve the literal scene without adding
unsupported interpretation.

(2) Visual evidence.  Include the concrete visible cues that
would let a viewer recognize the intended meaning from the
image alone.  Prefer observable objects, actions, spatial
relations, materials, symbols, settings, text elements, and
outcomes over abstract descriptions.  Replace vague concepts
```

```
with visible evidence: show the consequence, identifying
cue, relation, or communicative layout that the prompt
implies.
(3) Renderability and coherence. Phrase the scene so that
a text-to-image generator can plausibly render it. Avoid
unnecessary complexity, tiny details, long text passages,
dense labels, crowded scenes, fine-print elements, unusual
anatomy, or many small objects when a simpler visual cue
communicates the same meaning. Keep all details mutually
consistent.
Front-load the visual payload. Put the most distinguishing
information first. Good: "Curved iron-filing field lines
arc between two bar magnet poles on white paper." Bad: "A
photo of iron filings on paper near magnets, showing field
behavior caused by magnetism."
Output rules: - Return ONLY valid JSON: {"refined_prompt":
"<one sentence>"}. - Exactly one sentence, under 64 words.
- Put the most visually distinguishing detail in the first
20 words. - Do not add unsupported facts, identities, text,
symbols, or settings. - No subjective terms such as "easy",
"epic", or "successful". - No vague abstractions such as
"chaos" or "success"; show visible evidence instead. - No
camera jargon unless it is necessary to satisfy the prompt.
- Do not output analysis, labels, scores, or feedback.
```

This instruction is category-agnostic: it names the kinds of reasoning that may be required, but it does not provide the benchmark category label for any example. GEPA starts from this fused instruction and searches for improved instruction text using validation-set feedback.

B.4 Evaluator rubric

The training reward $R_{\text{opt}}(p, x)$ is the unweighted mean of three sub-scores in $\{0, 0.5, 1\}$. The evaluator first describes the generated image and then scores three axes with respect to the original prompt p. The refined prompt r and category label are not shown to the evaluator.

Axis 1: intent direction.

> **Question.** Score the image's intent direction by working through three checks. (1) Prompt classification: is the prompt *literal* (plainly describes a scene to render, no figurative language, no reasoning, no description-by-reference) or *interpretive* (idiomatic, figurative, describes an entity by reference, or requires world knowledge or reasoning)? (2) Literal-content presence: does the image visually contain the literal content of the surface words? (3) Interpreted-content presence: does the image visually contain the interpreted content the prompt actually intends? For literal prompts only Check 2 matters; for interpretive prompts both apply.
>
> **Criterion.** Literal+Check 2 yes $\rightarrow$ 1; literal+Check 2 no $\rightarrow$ 0; interpretive+Check 3 yes & Check 2 no $\rightarrow$ 1; interpretive+Check 3 yes & Check 2 yes $\rightarrow$ 0.5 (partial: both meanings depicted); interpretive+Check 3 no & Check 2 yes $\rightarrow$ 0 (literal trap); interpretive+Check 3 no & Check 2 no $\rightarrow$ 0. Faint atmospheric cues do not count as interpreted content; when in doubt, the answer is no.

Axis 2: detail faithfulness.

> Given the meaning the image is attempting (as judged in Axis 1), are the specific visual details required to make that meaning recognizable actually present and correct? This is conditional on the attempt: an image that took the wrong direction in Axis 1 can still be richly detailed for its own attempted meaning, and a right-direction image can still be detail-poor.
>
> **Criterion.** 1 if the required details are clearly present (figurative event with identifying cues; named entity with key attributes); 0.5 if some are present but others are missing or wrong; 0 if largely absent.

Axis 3: coherence.

> Setting aside meaning and detail, is the rendering technically coherent: legible required text, intact subjects, plausible anatomy, no garbled regions, no duplicated or broken objects?
>
> **Criterion.** 1 if the rendering is clean; 0.5 for minor issues that do not undermine the image (one mildly distorted hand, slightly garbled secondary text); 0 for major rendering failures (illegible required text, severely mangled main subject, broken composition).

The reward is $R_{\text{opt}} = \frac{1}{3}(s_1 + s_2 + s_3)$ where $s_i \in \{0, 0.5, 1\}$. The evaluator returns one paragraph of free-text rationale per axis; these rationales are forwarded to the reflection LM as feedback.

B.5 Reflection LM prompt

The reflection LM proposes revised instructions from per-prompt rewards and rationales. We wrap it with a prose-only system prompt that suppresses JSON-formatted instruction proposals; the wrapper does not change the underlying weights and is removed when the reflection LM is reused as the rewriter. The system prompt is:

```
You are a prompt engineer revising the system instruction
of a small text-to-image rewriter model.  You will be shown
the model's current instruction together with examples of
inputs, outputs, and judge feedback.  Your job is to write
an improved instruction.
STRICT OUTPUT FORMAT REQUIREMENTS: Respond ONLY in natural
English prose.  Write the new instruction as a human would
write a system prompt for an LLM: paragraphs, bullet points,
and clear directives.  Do NOT respond with JSON, YAML,
dictionaries, code blocks, function definitions, or any
structured data format.  Do NOT wrap your output in {}, [],
<>, or backticks.  Do NOT use keys like 'task_description',
'input_format', 'output_format', or 'rules' as JSON-style
headings.  The output of the rewriter model itself must
remain JSON (specifically {"refined_prompt":  "..."}); do
not change that.  But your instruction TO the rewriter is
plain English prose.
Begin your response with the first sentence of the new
instruction.  Do not preface with explanation or commentary.
```

In our optimization runs, the reflection LM occasionally produced JSON-formatted proposals rather than prose instructions. The wrapper reduces this failure mode and increases the fraction of usable proposals.

B.6 Instruction guard

The guard is a static filter applied to every candidate instruction before it can enter the GEPA candidate pool. It exists because reflection LMs occasionally emit instructions that are syntactically plausible but break the rewriter's contract: a JSON object instead of prose, a truncated proposal, an instruction that asks the rewriter to emit scores or category labels, or one that drops the JSON output schema. Such candidates produce silent failures: the rewriter's output is then unparseable, the evaluator sees a degenerate image, and GEPA may still score the run as comparable to the seed because both decay to similar behavior. Without the guard, the optimizer can converge on an instruction whose effective behavior is identical to s_0.

The guard rejects an instruction if any of the following holds, with negation-aware matching so that prohibitions in the seed (e.g. "do not output scores") are not themselves rejected:

- **Format-leak header.** Contains "`input format:`", "`output format:`", "`task description:`" or their JSON-key spellings.
- **JSON-blob proposal.** The instruction begins with `{` and a majority of its lines look like JSON keys (the reflection LM emitted an object instead of prose).

- **Structural truncation.** Brace or bracket imbalance, or unterminated string literal in the instruction body.
- **Affirmative ask for prohibited fields.** The instruction asks the rewriter to emit `score`, `scores`, `feedback`, `rating`, or `ratings`, with affirmative-verb context.
- **Affirmative dataset-taxonomy mention.** The instruction asks the rewriter to emit dataset or reasoning category labels.
- **Missing JSON output contract.** The instruction does not retain the requirement that the rewriter return JSON with key `refined_prompt`.
- **Missing refinement directive.** The instruction does not clearly instruct prompt refinement.
- **Length.** The instruction exceeds 450 words.

Table 6 compares GEPA with and without the guard while holding all other settings fixed. Without the guard, malformed proposals consume much of the evaluation budget and the final score remains close to the zero-shot baseline. With the guard, the same optimization budget reaches 74.6.

Table 6: Effect of the instruction guard on the T2I-ReasonBench overall score.

Configuration	Overall
Qwen2.5-1.5B zero-shot (s_0)	53.9
BRIDGEPROMPT, no guard	55.4
BRIDGEPROMPT, with guard	74.6

The most common failure is a proposal formatted as a JSON object (e.g., beginning with `{"task_description": ...}`) rather than as a prose instruction. Such proposals conflict with the rewriter's output contract and can lead to unparseable outputs. The guard contains no benchmark examples or semantic information about the target image.

C ADDITIONAL RESULTS

C.1 INFERENCE-TIME SEARCH BASELINES

Table 1 (main paper) compares inference-time search over the original prompt to language-model refinement. Implementation details:

Best-of-N. For each prompt we generate $N = 50$ images using SD3.5 Medium with seeds $\{0, 1, \ldots, 49\}$ and the same sampling configuration as the headline result. The selector is Qwen2.5-VL-7B-Instruct, prompted to choose the image that best satisfies the user prompt. The selector prompt uses the same describe-then-score protocol as the evaluator (§B.4) but exposes the original prompt p, not the refined prompt. The selector returns one integer index in $[0, N)$; ties are broken by sampling order. We re-run the protocol three times to estimate variance; the value reported in the main paper is the mean.

CFG sweep. For each prompt we generate one image at each of CFG $\in \{3, 5, 7.5, 10, 12\}$ holding all other settings fixed (including the seed). The reported value uses the same VLM selector as Best-of-N, restricted to the five candidates.

Both protocols leave the conditioning text unchanged. Their small gains support the hypothesis that many implicit-prompt failures arise from the conditioning rather than from image-space sampling alone.

C.2 UMAP ANALYSIS

Figures 2 and 3 (main paper) project original and intended-meaning refinements in the generator-facing CLIP space and the language-model semantic space, respectively. Implementation details:

Embeddings. For the CLIP panel, we encode each prompt with SD3.5 Medium's primary CLIP text encoder (OpenCLIP ViT-L/14) and take the *penultimate* layer hidden states (`hidden_states[-2]`), then apply attention-masked mean pooling over non-padding tokens. We use the penultimate-layer hidden states rather than the pooled output because our analysis is intended to reflect the token-level conditioning representation supplied to the generator. For the language-model panel, we encode each prompt with Qwen2.5-1.5B-Instruct, the same frozen prompter M used by BRIDGEPROMPT, and apply attention-masked mean pooling over the final hidden states.

Sample. We use all 800 T2I-ReasonBench prompts and their intent-aware refinements (1,600 embeddings per panel). Each original prompt is paired with its corresponding intended-meaning refinement; the projections visualize the resulting paired representations.

Projection. We project each d-dimensional embedding bank to two dimensions with UMAP (`n_neighbors`=15, `min_dist`=0.1, `metric`=cosine, `random_state`=42). The two panels are projected independently; absolute coordinates are not comparable across panels.

C.3 HIDDEN-CONSTRAINT RECOVERY

We provide additional details for the analysis in Table 2.

Tokenization and lemmatization. We tokenize with the spaCy English model (`en_core_web_sm`, v3.7), lemmatize each token, lowercase, and remove punctuation. Stopwords are the spaCy default English stopword list. *Content words* are the remaining tokens with POS in {NOUN, PROPN, VERB, ADJ}. The *hidden-constraint set* for prompt p_i is the set of content-word lemmas that appear in the intended-meaning annotation m_i but not in p_i.

Scoring. For each prompt p_i with refinement r_i, the recovery $\mathrm{R}(i)$ is the fraction of hidden-constraint lemmas that appear in r_i. We average over prompts with at least one hidden constraint. The textual image design category is excluded because its targets are layout and typography rather than lexical content.

Worked examples.

- **Idiom.** p_i = "the final exam was a piece of cake"; m_i = "a relaxed student finishing an exam easily in a classroom"; hidden-constraint set = {relaxed, student, finish, classroom} (after lemmatization). A refinement that names "a relaxed student in a classroom finishing an exam" recovers $4/4$.
- **Entity.** p_i = "the city hosting the first modern Olympic Games"; m_i = "Athens with the Panathenaic Stadium and the Acropolis"; hidden-constraint set = {Athens, Panathenaic, Stadium, Acropolis}. A refinement that names Athens but omits the landmarks recovers $1/4$.
- **Scientific.** p_i = "a rubber duck and a metal ball in a water tank"; m_i = "the duck floating on the surface and the ball sinking to the bottom"; hidden-constraint set = {float, surface, sink, bottom}. A refinement that names floating and sinking recovers $2/4$ (the lemmas "surface" and "bottom" are missed).

Generic baseline. The generic visual-expansion baseline is GPT-4o prompted with: "Rewrite the following prompt as a detailed visual description for a text-to-image model. Add style, lighting, camera, and composition cues. Do not infer hidden meaning." The intent-aware refinement uses our optimized $s^\star$.

C.4 EVALUATION PROTOCOL

The evaluator follows a two-stage protocol. First, it receives the generated image and produces a free-form image description. Second, it receives this description, the original prompt p, and the corresponding scoring block, and returns JSON containing a list of scores and a textual rationale:

```
{"reason": [...], "score": [s_1, s_2, ...]}
```

Each score is constrained to lie in $\{0, 0.5, 1\}$.

For final test-set reporting, we use the official T2I-ReasonBench evaluation scripts. All four scripts use Qwen2.5-VL-72B-Instruct as the reporting evaluator, with the scripts' default generation settings and `max_new_tokens`=1000. Idiom and textual-design prompts are scored using their corresponding question-criterion blocks. Entity prompts are scored as

$$0.7\, s_{\text{entity}} + 0.3\, s_{\text{detail}},$$

and scientific prompts are scored as

$$0.7\, s_{\text{scientific}} + 0.3\, s_{\text{detail}}.$$

Overall performance is the mean of the four category scores, so each category contributes equally.

During optimization, by contrast, we use Qwen2.5-VL-7B-Instruct with our generic three-axis rubric for intent direction, detail faithfulness, and image coherence. Optimization and final reporting therefore differ in both evaluator checkpoint (7B vs. 72B) and scoring rubric.

D Additional Analyses and Robustness Studies

D.1 Optimized instruction and instruction evolution

Table 7 shows two accepted intermediate instructions along the optimization trajectory from s_0 to $s^\star$. The corresponding validation reward increases from 0.52 to 0.71. The seed instruction s_0 is given in §B.3.

Table 7: Accepted instructions along one optimization trajectory.

Instruction	Validation reward
s_0 (seed)	0.52
s_1	0.58
s_2	0.64
$s^\star$ (final)	0.71

Intermediate instruction s_1.

```
Refine a draft text-to-image prompt into one short visual
prompt that the generator can render correctly.
You are given the original user prompt.  Produce a refined
prompt that preserves the user's intent while making the
intended image visually explicit.
Before writing, silently decide whether the prompt is
literal and complete or whether it requires interpretation.
If it is literal, preserve the requested scene, objects,
relations, and text without adding an unsupported story.
If it requires interpretation, resolve the intended scene
first and describe that scene directly.  Do not preserve the
surface expression merely because it appeared in the user
prompt, and do not combine the literal and intended meanings
unless the user asks for both.
Recover the visual information that the original wording
leaves unstated.  For figurative language, show the
ordinary real-world situation that the expression means,
using visible actions, expressions, posture, setting,
or outcomes instead of abstract words.  For an indirect
reference, name the person, place, object, or event when the
reference determines it uniquely, and add one or two visible
identifying cues.  A uniquely entailed identity is supported
information; unrelated attributes are not.  If the reference
remains genuinely ambiguous, do not guess.
For physical, causal, or scientific content, show the
relevant objects in their correct states and make the
```

decisive relation, change, or result visible in a single scene. For a requested poster, sign, diagram, or other text-dependent image, preserve the requested wording exactly and make it prominent and readable without adding extra text.

Use the smallest set of concrete visual details that makes the intended meaning recognizable. Prefer direct nouns and verbs, explicit spatial relations, and observable consequences. Avoid generic lighting, style, mood, or camera additions that do not help identify the intended scene. Avoid crowded compositions, long explanations, tiny details, and mutually inconsistent requirements.

Front-load the main identity, action, relation, or outcome in the first 20 words.

Return ONLY valid JSON: {"refined_prompt": "<one sentence>"}. The value must be exactly one sentence under 64 words. Do not output analysis, labels, scores, feedback, or additional keys.

Intermediate instruction s_2.

Refine the user's prompt into one short, generator-ready visual sentence. Preserve every intended requirement, but convert implicit meaning into concrete visual evidence that can be rendered in one coherent image.

Silently classify the prompt as literal or interpretive. For a literal prompt, keep the requested scene and do not add an interpretation. For an interpretive prompt, determine what the viewer should actually see before writing.

State the intended scene directly. Do not repeat metaphorical wording, description-by-reference phrases, or trigger objects that invite the wrong literal image. Omit unwanted literal content rather than negating it: do not write "without X" or "not Y" when X or Y names the incorrect interpretation, because mentioning it can still cause it to appear. Do not depict both meanings unless both are explicitly requested.

Recover the minimum sufficient evidence for the intended meaning. Begin with the principal subject and the decisive identity, action, spatial relation, state change, or outcome. Add only the strongest supporting cues. Every additional object, adjective, symbol, or text element becomes a rendering obligation, so remove decorative, redundant, speculative, or conflicting detail. Do not simplify by dropping a requirement from the original prompt.

When an indirect reference has one clear answer, name it directly and include one to three visible cues that distinguish it. Treat information logically entailed by the prompt as supported, but do not invent unrelated appearance, setting, or biography. If more than one answer remains plausible, do not guess.

For physical, causal, or scientific prompts, show all relevant objects in the correct states and make the interaction or result unmistakable. Connected components must visibly form the required relation; contrasting objects must display the correct different outcomes.

For posters, signs, diagrams, and text-rich designs, preserve only the exact requested wording. Make required text large, central, high-contrast, and legible in a simple layout, and add no extra slogan, label, or fine print.

Use direct nouns and verbs, explicit spatial relations, and visible evidence rather than explanations. Avoid negative

```
phrasing, generic style or camera filler, multiple panels,
crowded scenes, tiny secondary elements, and details that
cannot all be rendered clearly.  Front-load the decisive
visual content in the first 20 words.
Return ONLY valid JSON: {"refined_prompt":  "<one
sentence>"}.  The value must be exactly one grammatical
sentence under 64 words.  Do not output analysis,
explanations, category labels, scores, feedback, or
additional keys.
```

Final optimized instruction $s^\star$.

```
Rewrite the user's prompt into one short, generator-ready
visual sentence.  Preserve the user's intent, but convert
every required inference into concrete, visible content.
Before writing, silently decide whether the prompt is
literal and complete or requires interpretation.  If it
is literal, keep it close to the original and preserve every
requested object, attribute, relation, and text element.
If it is implicit, resolve the intended scene first and
describe that scene directly.
For an interpretive prompt, do not repeat metaphorical
wording, indirect reference phrases, or trigger objects
that invite the wrong literal image.  Omit unwanted literal
content rather than negating it.  Do not combine the literal
and intended meanings unless the user explicitly requests
both.
Choose the smallest set of visible evidence that makes
the intended meaning recognizable:  the main subject, the
decisive action, relation, or outcome, and a few identifying
cues.  Every added detail becomes a rendering obligation,
so omit decorative or speculative details.  Use direct
nouns and verbs, explicit spatial relations, and mutually
consistent facts.  Do not explain the reasoning, symbolism,
or concept.
When relevant:
- Resolve a uniquely implied person, place, object, or
event and name it directly, with one to three recognizable
visual cues.  Information strongly entailed by the prompt
is supported; do not invent unrelated attributes.  If the
reference is genuinely ambiguous, do not guess.
- For physical, causal, or scientific prompts, show the
objects in the correct states and make the interaction,
contrast, cause, or result unambiguous in one scene.
- For figurative language, render the intended real-world
situation and omit the expression's literal imagery.
- For posters, signs, diagrams, or text-rich designs,
preserve requested wording exactly, make it large, central,
legible, and high-contrast, use a simple layout, and add no
extra text.
Keep the scene easy to render:  one coherent composition,
few subjects, no alternatives, no dense labels, no long
passages, no tiny secondary details, and no generic style,
lighting, or camera filler unless requested.  Front-load the
decisive event, relation, or identity in the first 20 words.
Return ONLY valid JSON: {"refined_prompt":  "<one
sentence>"}.  The value must be exactly one grammatical
sentence under 64 words.  Do not output analysis,
explanations, labels, scores, feedback, or extra keys.
```

Across the trajectory, the instruction becomes more explicit about three behaviors: distinguishing literal from interpretive prompts, retaining only the visual evidence needed to convey the intended meaning, and avoiding wording that can trigger an incorrect literal interpretation.

D.2 STRONG INSTRUCTION BASELINES

To test whether the gain can be explained by stronger instruction design alone, we compare the human-written seed s_0, an instruction authored by GPT-4o from the task description and representative failure modes without optimization, and the optimized instruction $s^\star$. The prompter, generator, sampling configuration, and evaluation seeds are fixed across the three settings.

Table 8: Instruction-quality baselines on T2I-ReasonBench.

Instruction	Feedback used	Score
Human-written s_0	None	53.9
GPT-4o-authored instruction	None (one-shot)	59.6
BRIDGEPROMPT $s^\star$	Image-grounded	74.6

The GPT-4o-authored instruction improves over s_0 but remains substantially below $s^\star$, indicating that the observed gain is not explained by stronger one-shot instruction engineering alone. Because the GPT-4o instruction is not itself optimized, this comparison does not isolate image-level feedback from automatic instruction optimization.

D.3 OPTIMIZATION PROCEDURE

The optimization procedure is:

1. Initialize the candidate pool with the seed instruction s_0.
2. Sample a reflection minibatch from the T2I-ReasonBench training split.
3. For each prompt in the minibatch, run the frozen prompter, frozen image generator, and frozen VLM evaluator to obtain the generated image, per-example reward, and textual rationale.
4. Condition the frozen reflection LM on the current instruction and these execution traces to propose a revised instruction.
5. Apply the instruction guard in §B.6; malformed or interface-breaking proposals are rejected before consuming further evaluation budget.
6. Evaluate valid proposals on the minibatch. Candidate admission requires strict improvement on the minibatch, while GEPA maintains a Pareto frontier over per-prompt rewards.
7. Under the configured full-evaluation budget, evaluate retained candidates and select $s^\star$ as the candidate with the best validation reward.

Neither the prompter nor the image generator is updated during this procedure.

D.4 OPTIMIZATION STABILITY AND BUDGET SENSITIVITY

Each optimization rollout requires one prompter call, one full diffusion generation, and one VLM evaluation. The main configuration uses `max_full_evals`$= 1$ and reflection minibatch size 3, which in GEPA corresponds to a budget of approximately 400 metric calls (one pass over the 360 training and 40 validation prompts) and about six A100 GPU-hours per full compile. Proposals rejected by the instruction guard (§B.6) are filtered before evaluation and consume no rollouts.

We measure sensitivity on the idiom subset while holding the remaining settings fixed. Table 9 varies the reflection minibatch size, and Table 10 varies the full-evaluation budget.

Table 9: Sensitivity to reflection minibatch size with `max_full_evals`$= 1$.

Minibatch size	Idiom score	Compile time
3	69.5	1.5 h
6	68.8	2.6 h
12	70.1	4.9 h

Table 10: Sensitivity to full-evaluation budget with reflection minibatch size 3.

max_full_evals	Idiom score	Compile time
1	69.5	1.5 h
2	70.4	3.0 h
4	71.0	6.1 h

For reference, the same 1.5B prompter under s_0 scores 51.2 on idiom. Increasing either optimization parameter yields only small changes relative to the gain over s_0 while increasing compute substantially.

Seed and compile variability. Across the three evaluation seeds $\{0, 1, 2\}$, BRIDGEPROMPT scores 74.6 ± 2.54 (mean $\pm$ standard deviation). Across three independent GEPA compilations, the mean final score is 73.2 ± 3.10.

D.5 CONDITIONING ON INTENDED-MEANING ANNOTATIONS

We further test the conditioning-bound hypothesis by replacing the original prompt with the benchmark-provided intended-meaning annotation, a human-written statement of what the image should depict. This intervention changes the conditioning text without changing the generator.

Table 11: Conditioning intervention on T2I-ReasonBench.

Conditioning input	Overall score
Original prompt	46.8
GPT-4o rewrite	57.4
BRIDGEPROMPT rewrite	74.6
Intended-meaning annotation	82.3

The intended-meaning annotation reaches 82.3, showing that the frozen generator can render a substantial fraction of the target content once the missing semantics are stated explicitly. Combined with the small gains from Best-of-50 ($+0.8$) and the CFG sweep ($+0.3$), this supports the view that a substantial portion of the observed failure arises at the conditioning interface rather than from generator capacity alone.

D.6 CROSS-FAMILY EVALUATOR ROBUSTNESS

To test dependence on the optimization-time evaluator family, we repeat the full compile with GPT-4o-mini as the evaluator while holding the prompter, generator, seed instruction, and optimization budget fixed. All resulting instructions are evaluated using the official T2I-ReasonBench reporting protocol with Qwen2.5-VL-72B-Instruct. The GPT-4o-mini-optimized instruction scores 68.4; for context, instructions optimized with Qwen2.5-VL-3B, 7B, and 72B score 67.9, 74.6, and 75.3, respectively.

Table 12: Effect of the optimization-time evaluator. All final scores use the official Qwen2.5-VL-72B reporting protocol.

Optimization-time evaluator	Final score
Qwen2.5-VL-3B	67.9
GPT-4o-mini	68.4
Qwen2.5-VL-7B	74.6
Qwen2.5-VL-72B	75.3

The cross-family result remains above the non-optimized s_0 (53.9) and GPT-4o zero-shot rewriting baseline (57.4), showing that successful instruction optimization is not unique to the Qwen evaluator family. The spread across optimization evaluators nevertheless indicates sensitivity to evaluator choice.

D.7 QWEN-IMAGE: TRANSFER AND GENERATOR-SPECIFIC RECOMPILATION

Qwen-Image provides a direct test on a recent T2I model with an LLM text encoder. We first transfer the SD3.5-optimized $s^\star$ without recompilation. Table 13 reports the T2I-ReasonBench test results.

Table 13: Transfer of the SD3.5-optimized instruction to Qwen-Image without recompilation.

Conditioning	Idiom	Textual	Entity	Scientific	Overall
Original prompt	62.7	77.1	61.8	68.2	67.5
GPT-4o zero-shot	69.5	79.5	68.5	75.0	73.1
$s^\star$ (optimized on SD3.5)	74.0	67.9	73.5	78.5	73.5

We then perform a full BRIDGEPROMPT compile with Qwen-Image as the frozen generator while keeping the prompter, evaluator, seed instruction, and optimization budget unchanged. The Qwen-Image-specific instruction reaches 80.5 overall, compared with 67.5 from the original prompt and 73.5 from transferring the SD3.5-optimized instruction.

Table 14: Qwen-Image-specific recompilation.

Conditioning	Overall
Original prompt	67.5
Transferred SD3.5 $s^\star$	73.5
Qwen-Image-specific $s^\star$	80.5

The 7.0-point gap between transferred and generator-specific optimization shows that transfer is not seamless: part of the learned repair policy transfers across generators, while recompilation can further adapt the instruction to the target generator.

D.8 HUMAN EVALUATION

We conduct a preliminary human evaluation on 24 held-out T2I-ReasonBench prompts, with six prompts from each category. For both GPT-4o refinement and BRIDGEPROMPT, images are generated with SD3.5 Medium at 1024×1024, CFG 4.5, 28 denoising steps, and seed 0. Thirteen annotators evaluate all 24 prompts, yielding 312 judgments. For each prompt, annotators see the original prompt, the base image as a reference, and the images generated from the GPT-4o and BRIDGEPROMPT refinements. They select which refined image better follows the original prompt, or whether neither improves over the base reference.

Table 15: Human preference evaluation.

Per-rating outcome	Count	Rate
BRIDGEPROMPT image preferred	$166/312$	53.2%
GPT-4o image preferred	$109/312$	34.9%
Neither preferred over base	$37/312$	11.9%

These are descriptive per-rating proportions. Because each prompt is rated by multiple annotators and each annotator rates multiple prompts, we do not treat the 312 judgments as independent samples or report naive binomial confidence intervals.