LEARNING SAMPLING PARAMETERS FOR DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-to-image diffusion models expose many inference-time sampling parameters, including prompts, classifier-free guidance scales, and noise schedules. These parameters are typically manually chosen once and held fixed across prompts and denoising timesteps, despite different prompts and stages of generation benefiting from different parameter values. We introduce LeSAMP, a framework for learning prompt-conditioned, timestep-varying sampling parameters while keeping the diffusion model frozen. We formulate parameter selection as a reinforcement learning problem: given a user prompt, a large language model is trained to emit schedules for the chosen sampling parameters. We optimize our model using rewards from human preference models and VLM-as-a-judge. We evaluate LeSAMP on multiple diffusion backbones and find that compared to baselines, LeSAMP has a win rate of up to 68.12% using human preference scores and 73.37% using VLM-as-a-judge. These gains are validated in a user study where we achieve win rates of up to 59.46% over previous baselines. Our results suggest that learned sampling-parameter policies provide a complementary approach to existing post-training methods for improving diffusion model outputs.

1 INTRODUCTION

Diffusion and flow-matching models (Ho et al., 2020; Song et al., 2021; Lipman et al., 2023; Rombach et al., 2022) have recently become the dominant paradigm for high-fidelity image generation. Progress has come from novel scalable denoising architectures (Peebles & Xie, 2023), large-scale training on paired text-to-image data (Schuhmann et al., 2022), and advances in inference techniques and samplers (Song et al., 2021; Lu et al., 2022; Ho & Salimans, 2022). All these advancements, particularly those at the inference stage, require careful tuning of the hyperparameters (e.g., the sampling parameters of the diffusion process).

While much of the progress in image generation has focused on the training stage and the sampler itself, less attention has been paid to the *choice of sampling parameters*. Modern diffusion models expose a growing hyperparameter space, including classifier-free guidance scale, the noise schedule and step count, and negative prompts. These parameters can interact with text prompts in varying ways. Specifically, different prompts can benefit from different parameters, and the same prompt can benefit from varying parameters throughout the denoising trajectory (Kynkäänniemi et al., 2024; Papalampidi et al., 2025; Huberman et al., 2026). In practice, however, sampling parameters are typically hand-picked once per model and held fixed across all prompts, leaving a substantial gap between default and prompt-optimal sampling parameters.

In this paper, we introduce LeSAMP (**Le**arning **Sam**pling **P**arameters for Diffusion Models), a method that aims to automatically predict prompt-conditioned sampling parameters for text-to-image diffusion models. Concretely, we learn policies that predict four schedulable parameters across denoising timesteps: positive prompt, negative prompt, classifier-free guidance scale, and noise schedule. Rather than using a single fixed value or schedule, we learn to assign different parameter values to different timesteps and prompts.

We formulate predicting each sampling parameter as a reinforcement learning (RL) problem and fine-tune a large language model (LLM) with GRPO (Shao et al., 2024) to emit parameter schedules conditioned on the prompt. We investigate two reward signals: a learned human preference model (Ma et al., 2025) and a VLM-based image-pair critic. Compared to RL post-training of diffusion weights,

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.

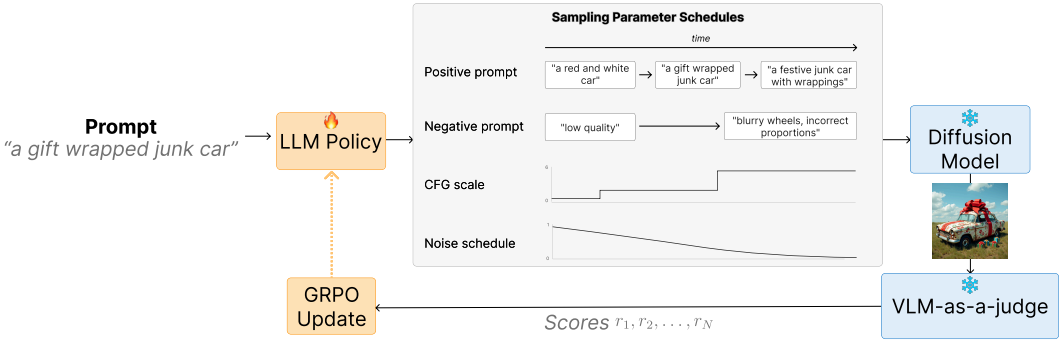

Figure 1: **LeSAMP optimization overview.** Using GRPO and VLM-as-a-judge preference reward, we train an LLM policy to predict timestep-varying positive prompt, negative prompt, CFG scale, or noise schedule given a user prompt.

this design has two attractive properties. First, the diffusion model remains frozen, so its base distribution is preserved. The policy is structurally limited in how far it can drift from the model's natural output, reducing susceptibility to mode-collapse and reward-hacking failures common to RL. Second, each policy update only requires running the diffusion sampler in inference mode, removing the need for backpropagation through the denoising chain.

We evaluate LeSAMP on FLUX.1 [DEV] (Flux) (Labs, 2024), Stable Diffusion 3.5 (SD 3.5) (Esser et al., 2024), FLUX.2 [DEV] (Labs, 2025), and Qwen-Image-2512 (Wu et al., 2025). Against each model's default sampling parameters, SAP (Huberman et al., 2026), and R2F (Park et al., 2025a), LeSAMP trained with VLM-as-a-judge leads to a win rate of up to 68.12% on human preference scores and 73.37% with VLM-as-a-judge. These gains are validated in a user study where we achieve win rates up to 59.46% over SAP and baseline SD 3.5.

Our results suggest that predicting per-step sampling parameters is a simple, model-agnostic, and complementary axis to existing post-training methods for improving diffusion model outputs.

To summarize, our contributions are:

1. We introduce LeSAMP, a method for predicting prompt-conditioned, timestep-varying sampling parameters for text-to-image diffusion models by fine-tuning an LLM policy with GRPO.
2. We define a unified action space over the positive prompt, negative prompt, CFG scale, and noise schedule, with policy-determined span granularity.
3. We show empirically that LeSAMP improves both human and VLM preference win rates over existing baselines over multiple diffusion backbones and two reward signals.

2 Related Work

Diffusion models. Given clean data $x_0 \sim p_{\text{data}}$, the forward diffusion process constructs noisy latents $q(x_t \mid x_0)$ by adding Gaussian noise according to a prescribed noise schedule. A denoising network ϵ_θ is then trained to denoise by predicting the added noise or an equivalent parameterization. In the noise-prediction formulation, the training objective is

$$\mathcal{L}_{\text{DM}}(\theta) = \mathbb{E}_{x_0,\, t,\, \epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t, c)\|_2^2\right], \qquad x_t = \alpha_t x_0 + \sigma_t \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{1}$$

where c denotes optional conditioning information such as a text prompt, and (α_t, σ_t) are determined by the noise schedule (Ho et al., 2020; Song et al., 2021). At inference time, sampling starts from Gaussian noise and repeatedly applies the learned denoiser to produce progressively cleaner samples, using a chosen numerical sampler such as DDIM, DPM-Solver, or an Euler solver for flow-based models (Song et al., 2022; Lu et al., 2022; Lipman et al., 2023).

Tuning diffusion sampling parameters. The quality of samples depends not just on the trained denoiser, but also on inference-time choices such as the sampler, noise schedule, step count, guidance

scale, positive prompt, and negative prompt. Prior work has found that different denoising timesteps benefit from different sampling behavior. Kynkäänniemi et al. (2024) show that classifier-free guidance is harmful at high noise levels and unnecessary at low ones, and propose a fixed middle guidance interval. Papalampidi et al. (2025) perform a greedy *per-prompt* search over CFG scales at each timestep using small latent-space evaluators. Unlike our policy, their method does an online search per sample, is limited to CFG, and requires an inference-time evaluator that scores partial latents. Another work, TPDM (Ye et al., 2025), optimizes the timestep schedule from latent trajectory features. Co-GRPO (Zhou et al., 2025) uses GRPO to co-optimize masked image-model weights and inference schedules. For text generation, Zhou & de Cruys (2026) use PPO to learn CFG scale selection for a frozen diffusion language model.

RL post-training of diffusion models. A large body of work fine-tunes diffusion model weights with RL to optimize non-differentiable rewards. DDPO (Black et al., 2024) reframes denoising as a multi-step MDP and applies PPO-style policy gradients, while DPOK (Fan et al., 2023) adds KL regularization to a pretrained reference. ReFL (Xu et al., 2023) and DRaFT (Clark et al., 2024) instead backpropagate differentiable rewards through a truncated portion of the sampling chain. Preference-based variants such as Diffusion-DPO (Wallace et al., 2023) extend DPO (Rafailov et al., 2024) to image generation, and Flow-GRPO (Liu et al., 2025) adapts GRPO (Shao et al., 2024) to flow-matching models by converting the deterministic ODE into an SDE for exploration. Each method modifies diffusion weights, and some also require backpropagation through the sampling trajectory. Our method is complementary. We freeze the diffusion model and train an LLM policy over its sampling parameters, preserving the diffusion model's distribution and making each policy update much cheaper.

LLM-guided prompt and sampling control for text-to-image generation. Another line of work uses LLMs to control diffusion models. Promptist (Hao et al., 2023) fine-tunes a language model with PPO to rewrite the user prompt, optimizing a combined aesthetic and CLIP reward. Other training-free methods prompt an LLM to plan time-varying conditioning – R2F (Park et al., 2025a) uses an LLM to alternate between rare concepts in the prompt and surrogate concepts with similar visual features at each denoising step. SAP (Huberman et al., 2026) targets prompts containing contextual contradictions by querying an LLM to decompose the target prompt into a sequence of proxy prompts applied at different timesteps. LeSAMP combines complementary aspects of these methods. Like Promptist, we train an LLM with RL to output sampling parameters that optimize an image-level reward, and like R2F and SAP, our policy emits timestep-varying sampling parameters rather than a static value. Recent work has also explored automating negative prompts. NPC (Park et al., 2025b) uses an LLM to propose negative prompts and searches for the best candidate during inference time. In contrast, LeSAMP emits conditioning parameters in a single pass, avoiding test-time search, and predicts timestep-varying parameter schedules rather than static negative prompts.

3 Learning to Predict Diffusion Sampling Parameters via RL

In this section, we present our method, LeSAMP (Figure 1). We formulate the task of choosing diffusion sampling parameters as a Markov decision process. Then, we utilize reinforcement learning to optimize an LLM to produce these parameters, given a reward function and a prompt.

3.1 Preliminaries

Classifier-free guidance. For text-to-image models, sampling is generally conditioned on a prompt. *Classifier-free guidance* (CFG) (Ho & Salimans, 2022) sharpens conditioning by extrapolating between a conditional and an unconditional noise prediction:

$$\tilde{\epsilon}_\theta(x_t, c_+, c_-, w) = \epsilon_\theta(x_t, c_-) + w\big(\epsilon_\theta(x_t, c_+) - \epsilon_\theta(x_t, c_-)\big), \tag{2}$$

where c_+ is the conditioning (positive) prompt, c_- is an unconditional (negative) prompt, and $w \geq 1$ is the guidance scale. Larger w strengthens prompt conditioning and can improve sample fidelity. However, it can also reduce diversity and introduce artifacts such as oversaturation (Ho & Salimans, 2022; Lin et al., 2024; Sadat et al., 2025).

Noise schedule. The noise schedule specifies the sequence of noise levels $\sigma_0 > \sigma_1 > \cdots > \sigma_T$ followed by the sampler during denoising. Larger values of σ_t correspond to earlier, more corrupted

latents, while smaller values correspond to later, cleaner latents. At inference time, the sampler starts from an initial high-noise latent and iteratively moves through this sequence toward $\sigma_T \approx 0$, applying the denoising model at each step. The choice of schedule determines how the finite sampling budget is allocated across high to low noise regimes, and can affect global structure, fine details, prompt adherence, and overall image quality. Standard text-to-image pipelines fix this sequence in advance, often using a hand-designed rule such as a linear, cosine, Karras, or model-specific default schedule.

Policy controls. We train policies to control each of these diffusion sampling parameters: positive prompt c_+, negative prompt c_-, CFG scale $w \in [w_{\min}, w_{\max}]$, and noise schedule $\sigma = [\sigma_0, \sigma_1, \ldots, \sigma_T]$. While standard practice fixes c_+, c_-, w, and the noise schedule $\{\sigma_t\}$ for the full trajectory, we treat each of these as a *schedulable* parameter that an LLM policy can vary across the denoising trajectory (Section 3.3).

Group Relative Policy Optimization (GRPO). GRPO (Shao et al., 2024) is a policy gradient method that replaces the learned value function of PPO (Schulman et al., 2017) with a group-relative baseline. For each prompt x in a training batch, GRPO samples a group of G rollouts $\{o_1, \ldots, o_G\}$ from the current policy $\pi_{\theta_{\text{old}}}$ and assigns each a reward $r_i = R(o_i)$. The policy is updated by maximizing a PPO-style clipped surrogate objective with a KL penalty to a reference policy π_{ref}, which in our case is the initialization checkpoint:

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\big(\rho_{i,t}\hat{A}_i,\ \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\hat{A}_i\big) - \beta\, D_{\text{KL}}[\pi_\theta \,\|\, \pi_{\text{ref}}]\right], \quad (3)$$

where $\rho_{i,t} = \frac{\pi_\theta(o_{i,t}|x, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|x, o_{i,<t})}$ is the per-token importance ratio, ϵ and β are hyperparameters controlling clipping and KL regularization, and $\hat{A}_i = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^G)}{\text{std}(\{r_j\}_{j=1}^G)}$ is rollout i's advantage. Because our reward is defined only at the end of the episode (after parsing the rollout, running the diffusion model, and scoring the image), every token in a rollout shares the same advantage $\hat{A}_i$.

3.2 Problem Setting

We formalize our setting as a finite-horizon MDP, treating the diffusion model as part of the environment. The LLM policy generates tokens that are parsed into sampling parameters, and the diffusion model is called at the episode end to produce a sample that is assigned a reward.

The state space S consists of all possible token sequences in the LLM's context window, so a state $s_t \in S$ corresponds to the original prompt concatenated with all tokens generated up to step t. The action space A is the LLM's vocabulary $\mathcal{V}$, and an action $a_t \in A$ is the next token sampled from the policy. The transition kernel P is deterministic. Given (s_t, a_t), s_{t+1} is obtained by appending a_t to s_t. The reward R is given by parsing the produced tokens into sampling parameters, calling a diffusion model with these parameters, and scoring the sample with a chosen reward model. Finally, we set $\gamma = 1$, since each episode terminates at the maximum token budget or earlier.

3.3 Generating Sampling Parameters

LLM Inputs. The LLM is prompted with a description of each parameter and its allowed value range, and responds with a JSON object containing a subset of the four keys. Any parameter not emitted falls back to the diffusion model's default sampling parameters. This design lets us study each parameter in isolation by restricting the prompt to a single key, and also allows the policy to combine them. See Appendix J for prompts used.

LLM Outputs. The policy outputs a JSON string that is parsed into a schedule over the denoising trajectory. Let the diffusion model run for T denoising steps. For a sampling parameter p, a schedule is a partition of $[0, T]$ into contiguous spans $\{[\tau_0^p, \tau_1^p), [\tau_1^p, \tau_2^p), \ldots, [\tau_{K-1}^p, \tau_K^p]\}$ with $\tau_0^p = 0$ and $\tau_K^p = T$, together with a value v_k^p for each span k. This allows the policy to use coarse or fine schedules depending on the prompt and token budget. See Appendix I for example outputs.

Outputs that contain invalid values (e.g., a span not in $[0, T]$) are assigned a penalty reward relative to the GRPO group's valid rewards. Concretely, for a prompt x with group $\mathcal{G}(x) = \{o_1, \ldots, o_G\}$ of G rollouts, let $\mathcal{G}_{\text{valid}}(x) \subseteq \mathcal{G}(x)$ denote the rollouts that parse and produce an image. Let σ_{valid} denote

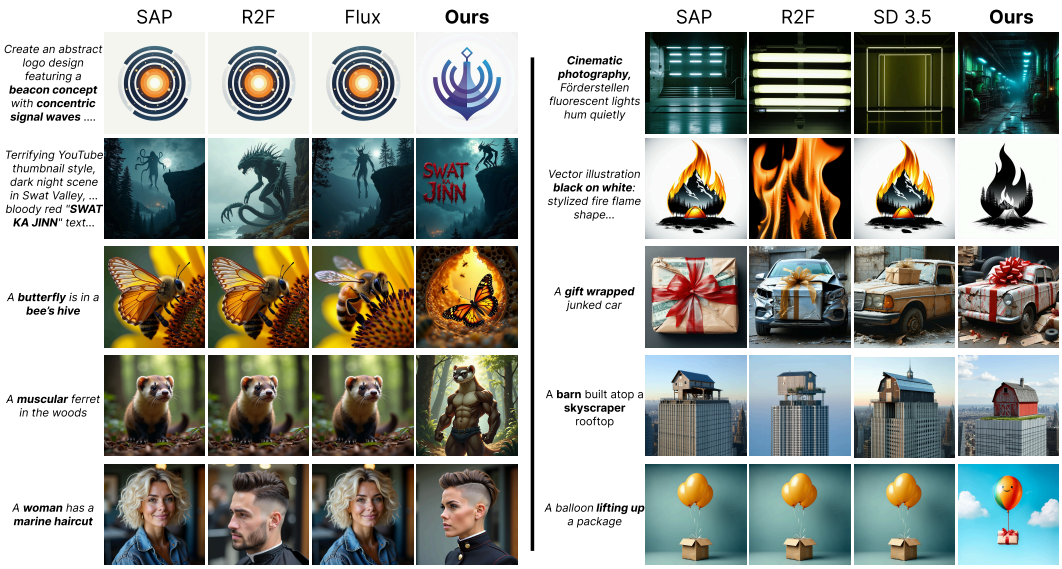

Figure 2: Comparison between LeSAMP, SAP, and R2F, applied to SD 3.5 and Flux. Each row shows images generated from a prompt shown on the left, using the methods in the columns.

the standard deviation of the rewards assigned to valid rollouts in the group. The reward is

$$R(o_i) = \begin{cases} R_{\text{model}}(\text{image}(o_i)) & o_i \in \mathcal{G}_{\text{valid}}(x), \\ \min\limits_{o_j \in \mathcal{G}_{\text{valid}}(x)} R_{\text{model}}(\text{image}(o_j)) - \max(\lambda_{\text{fail}}\sigma_{\text{valid}}, 1) & o_i \notin \mathcal{G}_{\text{valid}}(x). \end{cases} \tag{4}$$

Here we set the failure penalty coefficient λ_{fail} to 2. This way, invalid outputs receive a strictly lower reward than any valid output in their group. If no valid completions exist in the group, that prompt is dropped from the training batch.

3.4 Reward Functions

We explore a variety of reward functions to guide the LLM to generate sampling parameters that lead to preferred images. These reward functions are used to score each generated image.

VLM-as-a-judge. We use VLM-as-a-judge by prompting a VLM (e.g., Gemini 3.5 Flash) to compare two images and return the index of the image with better prompt adherence and aesthetics. We repeat this for all pairs of images in a group, collecting win statistics over each pair. The score of each image is its group-level win rate. If both images are equally good or equally bad, the model is prompted to return a tie. To prevent bias from image position, we query the VLM twice with both orderings of the images. If any response contains a tie or responses disagree, we consider the final evaluation to be a tie. In Appendix D, we measure alignment between reward models and human preferences and find that Gemini 3.5 Flash–human agreement is comparable to human–human agreement. These results support our use of Gemini 3.5 Flash-as-a-judge as the training reward in our main experiments.

HPSv3. HPSv3 (Ma et al., 2025), a human preference model trained on pairwise text-to-image rankings, is another reward function candidate. Given an image and a prompt, it outputs a scalar that aims to represent how well the image aligns with the prompt. We note that HPSv3 is susceptible to reward hacking via its bias toward aesthetic but non-prompt-adherent images. For this reason, we find that HPSv3 is not an optimal reward signal to train on. We show the results of training with HPSv3 and discuss its limitations in Appendix B.

Table 1: Pairwise win rates. Each cell reports the tie-adjusted win rate $(W + 0.5T)/(W + L + T)$ where W, L, and T are wins, losses, and ties, respectively. 95% confidence intervals are reported with $\pm$ margins, using the number of prompts as the effective sample size. Each row is evaluated on the dataset in the top-level column using the reward function in the second-level column.

	Whoops-Hard ∪ ContraBench		**User Requests**	
Comparison	Gemini	HPSv3	Gemini	HPSv3
Ours vs $\text{SAP}_{\text{SD 3.5}}$	64.58 ± 6.23	57.25 ± 8.25	62.71 ± 4.93	56.67 ± 6.27
Ours vs $\text{R2F}_{\text{SD 3.5}}$	67.84 ± 6.51	68.12 ± 7.77	68.91 ± 4.77	61.25 ± 6.16
Ours vs SD 3.5	66.49 ± 6.19	60.14 ± 8.16	60.10 ± 5.00	52.08 ± 6.32
Ours vs SAP_{Flux}	67.48 ± 6.14	51.54 ± 8.33	64.27 ± 4.93	64.58 ± 6.05
Ours vs R2F_{Flux}	73.37 ± 5.91	63.04 ± 8.05	69.27 ± 4.92	63.75 ± 6.08
Ours vs Flux	69.38 ± 5.84	55.80 ± 8.28	60.31 ± 4.92	57.50 ± 6.25

4 EXPERIMENTS

We evaluate whether LeSAMP can improve image aesthetics and prompt adherence over baselines, and discuss the distribution of learned sampling parameters. Our main experiments fine-tune Qwen3-32B with GRPO to produce the sampling parameters described in Section 3.1. We train the policy to maximize VLM-as-a-judge reward, using Gemini 3.5 Flash. We use Stable Diffusion 3.5 and FLUX.1 [DEV] as our main diffusion backbones. Additional experiments on FLUX.2 [DEV] and Qwen-Image-2512 assess whether LeSAMP also improves generation on newer backbones, with details provided in Appendix A. For implementation details, see Appendix H.

To capture the distribution of common image generation requests, we collect and evaluate on a held-out set of 240 user image generation requests, which we refer to as User Requests. We also evaluate on out-of-distribution and contextually contradictory prompts, from the union of Whoops-Hard and ContraBench, which we refer to as Whoops-Hard ∪ ContraBench (Huberman et al., 2026). Unless otherwise stated, we evaluate each prompt using two generation seeds.

Baselines. We compare LeSAMP against three baselines. First, we use each diffusion model's default sampling parameters, keeping the original prompt fixed. We also compare against SAP (Huberman et al., 2026), which uses an LLM to decompose contextually contradictory prompts into a sequence of proxy prompts for different denoising stages. Finally, we compare to R2F (Park et al., 2025a), which uses an LLM to swap rare prompt concepts with related common concepts, switching back and forth between the original and common prompts. We replace SAP and R2F's original GPT-4o LLM with GPT-5.4 mini, while keeping everything else unchanged. Since GPT-5.4 mini is stronger than off-the-shelf Qwen3-32B, this makes the prompt-scheduling baselines more competitive and ensures our gains come from LeSAMP's method rather than from using a stronger base LLM.

Qualitative Results. Figure 2 presents a qualitative comparison of LeSAMP to baselines. The first two rows use prompts from our User Requests dataset, and the remaining rows use prompts from Whoops-Hard ∪ ContraBench. Using the default sampling parameters often creates images that miss important details on User Requests and do not accurately represent the contextually contradictory prompts from Whoops-Hard ∪ ContraBench. SAP and R2F improve some contradictory cases, but because they rely on prompt decomposition or concept substitution, they can degrade ordinary user requests. For example, in the "Terrifying YouTube thumbnail..." prompt, both methods introduce the required "SWAT KA JINN" text only at later denoising steps, after the image layout has largely formed. As a result, the final image fails to render the text. On contextually contradictory prompts such as "A gift wrapped junk car," SAP and R2F correctly recognize the contradiction, but their prompt-swapping strategies still fail to produce an image that fully captures the intended composition.

Our trained policy takes a different approach. Instead of swapping between common and rare concepts, our policy outputs shorter prompts at the beginning and gradually adds detail in later steps. With these prompt schedules, we observe that our policy leads to images that are both prompt-adherent and aesthetically pleasing. These findings suggest that *concept-substitution methods are not strictly*

Table 2: Pairwise preference data collected from a user study reporting tie-adjusted win rates.

Dataset	Comparison	Win rate
Whoops-Hard ∪ ContraBench	Ours vs $SAP_{SD\ 3.5}$	53.68
Whoops-Hard ∪ ContraBench	Ours vs SD 3.5	59.46
User Requests	Ours vs $SAP_{SD\ 3.5}$	54.29
User Requests	Ours vs SD 3.5	55.81

Table 3: Win rates against default sampling parameters when using Gemini-as-a-judge preference or HPSv3 score after training with Gemini preference reward, evaluated on User Requests.

Diffusion model	Trained parameter	Gemini	HPSv3
Stable Diffusion 3.5	Positive prompt	60.10	52.08
Stable Diffusion 3.5	Negative prompt	56.46	58.75
Stable Diffusion 3.5	CFG Scale	54.79	53.75
Stable Diffusion 3.5	Noise Schedule	51.46	48.33
Flux.1 [dev]	Positive prompt	60.31	57.50
Flux.1 [dev]	Negative prompt	62.29	35.00
Flux.1 [dev]	CFG Scale	48.85	58.75
Flux.1 [dev]	Noise Schedule	51.04	51.25

needed to achieve good results on contextually contradictory prompts. Rather, gradually adding more detail to prompts throughout the denoising process can satisfy requirements for both contextually contradictory and in-distribution user requests.

Quantitative Results. As shown in Table 1, LeSAMP achieves win rates above 50% across each baseline, dataset, and evaluator. While our policy was trained to maximize Gemini preference win rate, it was never trained to maximize HPSv3, reducing the likelihood that our results stem from reward hacking. In Table 10, we also evaluate using SAP's prompt alignment and image quality scoring. We observe that LeSAMP matches or exceeds SAP and R2F on both scores. Finally, we compare learning sampling parameters for a frozen diffusion model to updating the diffusion model itself. In Appendix C, we compare LeSAMP with Flow-GRPO (Liu et al., 2025). We observe that LeSAMP improves on both in-distribution and out-of-distribution prompts, while Flow-GRPO only improves on in-distribution prompts, suggesting that it overfits more easily compared to LeSAMP.

User Study. To validate our previous results, we conduct a user study comparing LeSAMP, SAP, and default sampling parameters. Users compare two images for each prompt, considering prompt adherence and image aesthetics. Across Whoops-Hard ∪ ContraBench and User Requests, 13 evaluators provide 511 pairwise preference labels. As shown in Table 2, we see that images from LeSAMP are preferred over those from SAP and SD 3.5, supporting our previous findings.

4.1 Ablations

Effect of predicting each sampling parameter. In Table 3 and Figure 3, we show the results of training our policy to predict each sampling parameter in Section 3.1, and evaluate using the reward functions in Section 3.4 on Flux and SD 3.5. These results indicate that our policy is able to output positive and negative prompts that outperform the baseline across nearly all settings. Detailed examples of outputs are shown in Appendix I.

Across both diffusion models, we find that our policy learns to output progressively longer prompts at later denoising steps. More detail about this behavior is discussed in the Qualitative Results. Negative prompts behave differently between Flux and SD 3.5. In SD 3.5, the policy has learned a mostly universal negative prompt schedule, cycling through generic photographic-quality terms (e.g., *blurry, low resolution*) at intermediate and late steps. In Flux, the negative prompt at the first and last



Figure 3: Results from four LeSAMP-trained policies predicting schedules for positive prompt, negative prompt, CFG, and noise on SD 3.5. Areas of interest are boxed in red. Images without boxes indicate that the global context is of interest. Please zoom in for the best details.

denoising steps is mostly fixed, similar to SD 3.5. However, intermediate steps adapt to the prompt content (e.g., *blurry faces* on portrait prompts).

With CFG and noise schedule, we see less consistent improvements. In SD 3.5, we do observe an improvement from predicting CFG scale. Here, our policy has learned to output a monotonically increasing CFG schedule, starting near 4 and ending around 8. However, we do not see the same improvement in Flux. We believe this is because Flux is guidance-distilled, resulting in a less rich and harder-to-predict action space. For noise schedule, we observe that across both models, our

Table 4: Evaluation of smaller policy win rates against default sampling parameters, using Gemini 3.5 Flash as a judge.

Policy	User Requests	Whoops-Hard ∪ ContraBench
Qwen3-1.7B	58.75	59.06
Qwen3-4B	62.92	65.22

policy learns to predict an approximately linearly decreasing schedule, similar to the default schedule. Given that our win rates with noise schedule are $\sim$50%, this behavior makes sense.

Finally, we observe that positive prompt gives the best combination of gains on the training reward and on a held-out human preference model. For this reason, we choose to predict the positive prompt for the majority of our experiments.

Effect of timestep-varying sampling parameters. To isolate the effect of predicting time-varying positive prompts as opposed to a single prompt, we train a policy to predict the positive prompt for SD 3.5 without changing it at different timesteps, similarly to Promptist (Hao et al., 2023). While we observe a Gemini-judge win rate of 55.62%, indicating improvement over the default sampling parameters, this win rate is still lower than the 60.10% win rate we observe when predicting timestep-varying positive prompts in Table 3. This gap isolates the benefit of timestep variation, suggesting that the gains are not solely due to generic prompt optimization, but also from scheduling the conditioning over time. Both policies can rewrite the user prompt, but only LeSAMP can change the conditioning over the denoising trajectory.

Effect of policy size. To investigate whether image preference gains hold with smaller policies, we train Qwen3-1.7B and Qwen3-4B policies with an SD 3.5 renderer. As shown in Table 4, both smaller policies achieve win rates $\geq 58\%$ over the baseline. This indicates that LeSAMP is an accessible method for improving image generation quality in compute-constrained environments.

5 Discussion, Limitations, and Future Work

Our results suggest that learning prompt-conditioned sampling parameters is an effective way to improve text-to-image generation without modifying the underlying diffusion model. Across diffusion backbones, datasets, and evaluation metrics, LeSAMP improves over default sampling parameters and prior LLM-based prompt-scheduling baselines. This indicates that a substantial part of the gap between default and preferred generations can be closed through better inference-time conditioning.

Limitations and future work. We find that the choice of reward model is critical. When training with HPSv3, we observe severe reward hacking. Generated images are often aesthetically strong, but no longer satisfy user intent. We elaborate on our reward hacking observations in Appendix B.

We also find that jointly training over all four sampling parameters does not substantially improve over training on positive-prompt-only. The jointly trained policy achieves similar win rates to the positive-prompt-only policy, with a 60.42% Gemini win rate and a 54.58% HPSv3 win rate against the default sampler. The larger action space may require a larger training budget for exploration, or the most accessible gains for our models and datasets already come from time-varying prompts. We believe investigation in both areas can lead to new techniques for setting optimal diffusion model sampling parameters. Finally, policies trained with our method still fail on some difficult prompts. Because our training distribution consists of ordinary user requests on which such failures are rare, one future direction could explore synthetically generating difficult user requests for training data within our framework.

AI USE STATEMENT

In this work, we used generative AI tools for polishing writing, finding related work, and for experiment execution. We have not used generative AI tools for drafting sections of the paper, generating synthetic datasets, or proving mathematical claims. We have reviewed all AI-assisted work by checking writing and claims, validating the related work, and reviewing AI-written code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Users were compensated for this study as part of their regular job requirements. For more details, see Appendix G.

REPRODUCIBILITY STATEMENT

We hope to release this paper so that others can reproduce our work and improve on it. The high-level experimental setting is stated in Section 4. Here we describe the models, data, and results for each of the experiments we ran. We also provide low-level implementation details in Appendix H. Finally, we will release code that implements our method upon acceptance.

REFERENCES

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning, 2024. URL https://arxiv.org/abs/2305.13301.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards, 2024. URL https://arxiv.org/abs/2309.17400.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. URL https://arxiv.org/abs/2403.03206.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models, 2023. URL https://arxiv.org/abs/2305.16381.

Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation, 2023. URL https://arxiv.org/abs/2212.09611.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Saar Huberman, Or Patashnik, Omer Dahary, Ron Mokady, and Daniel Cohen-Or. Image generation from contextually-contradictory prompts, 2026. URL https://arxiv.org/abs/2506.01929.

Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. *Advances in Neural Information Processing Systems*, 37:122458–122483, 2024.

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

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. https://bfl.ai/blog/flux-2, 2025.

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed, 2024. URL https://arxiv.org/abs/2305.08891.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

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.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, 2022. URL https://arxiv.org/abs/2206.00927.

Yuhang Ma, Yunhao Shui, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. Hpsv3: Towards wide-spectrum human preference score, 2025. URL https://arxiv.org/abs/2508.03789.

Pinelopi Papalampidi, Olivia Wiles, Ira Ktena, Aleksandar Shtedritski, Emanuele Bugliarello, Ivana Kajic, Isabela Albuquerque, and Aida Nematzadeh. Dynamic classifier-free diffusion guidance via online feedback, 2025. URL https://arxiv.org/abs/2509.16131.

Dongmin Park, Sebin Kim, Taehong Moon, Minkyu Kim, Kangwook Lee, and Jaewoong Cho. Rare-to-frequent: Unlocking compositional generation power of diffusion models on rare concepts with llm guidance, 2025a. URL https://arxiv.org/abs/2410.22376.

Sangha Park, Eunji Kim, Yeongtak Oh, Jooyoung Choi, and Sungroh Yoon. Guiding what not to generate: Automated negative prompting for text-image alignment, 2025b. URL https://arxiv.org/abs/2512.07702.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL https://arxiv.org/abs/2305.18290.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Seyedmorteza Sadat, Otmar Hilliges, and Romann M. Weber. Eliminating oversaturation and artifacts of high guidance scales in diffusion models, 2025. URL https://arxiv.org/abs/2410.02416.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022. URL https://arxiv.org/abs/2210.08402.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021. URL https://arxiv.org/abs/2011.13456.

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, 2023. URL https://arxiv.org/abs/2311.12908.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. URL https://arxiv.org/abs/2508.02324.

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, 2023. URL https://arxiv.org/abs/2304.05977.

Zilyu Ye, Zhiyang Chen, Tiancheng Li, Zemin Huang, Weijian Luo, and Guo-Jun Qi. Schedule on the fly: Diffusion time prediction for faster and better image generation, 2025. URL https://arxiv.org/abs/2412.01243.

Fan Zhou and Tim Van de Cruys. Guidance is not a hyperparameter: Learning dynamic control in diffusion language models, 2026. URL https://arxiv.org/abs/2605.07701.

Renping Zhou, Zanlin Ni, Tianyi Chen, Zeyu Liu, Yang Yue, Yulin Wang, Yuxuan Wang, Jingshu Liu, and Gao Huang. Co-grpo: Co-optimized group relative policy optimization for masked diffusion model, 2025. URL https://arxiv.org/abs/2512.22288.

Table 5: Pairwise winrates of LeSAMP against each backbone's default sampling configuration, using Gemini 3.5 Flash as a judge.

Diffusion backbone	User Requests	Whoops-Hard ∪ ContraBench
FLUX.2 [DEV]	67.00	60.51
Qwen-Image-2512	63.75	55.80
PFIG	57.20	60.10

Table 6: Pairwise preference data collected from a user study, showing tie-adjusted win rates of LeSAMP against PFIG.

Diffusion backbone	User Requests	Whoops-Hard ∪ ContraBench
PFIG	68.00	55.00

A Applying LeSAMP on additional diffusion backbones

To evaluate whether LeSAMP works broadly across different diffusion backbones, we train additional policies for FLUX.2 [DEV] (Labs, 2025) and Qwen-Image-2512 (Wu et al., 2025). We evaluate these policies on the same datasets as the main experiments, using Gemini 3.5 Flash as a judge. As shown in Table 5, our policy wins more often against the baseline on both datasets, suggesting that LeSAMP is effective across varying diffusion models.

We also evaluate LeSAMP on a proprietary frontier image generation model released in 2026 (which we will refer to as PFIG) whose architecture has significant differences from other models we evaluated. A natural concern with new approaches is that they mainly compensate for capability gaps in weaker models. However, we observe that LeSAMP still improves upon baseline PFIG, even though PFIG is a much stronger model than others we evaluated. These findings are validated by a user study, where we ask 10 people to each annotate 50 pairs of PFIG vs LeSAMP. We show tie-adjusted win rates from this study in Table 6, demonstrating that LeSAMP is consistently preferred over the baseline in terms of user preference. We also show qualitative results comparing PFIG and LeSAMP in Figure 4.

B HPSv3 Experiments

Here, we discuss the performance of policies after training with HPSv3 reward. We also discuss the reward hacking and biases that occur when training and evaluating with HPSv3.

B.1 HPSv3 Training Results

In Table 7, we show the results of training each available sampling parameter on SD 3.5 and Flux. Across most sampling parameters we are able to increase HPSv3 win rate, but some increases come at the expense of Gemini win rate. This suggests that the two rewards are not strongly correlated. We discuss a possible reason why in Appendix B.2.

B.2 HPSv3 Reward Hacking

In Figure 5, we can see an example of HPSv3's bias toward aesthetic but non-prompt-adherent images. The prompt requests a black and white image. The image on the left is in color, and the image on the right is in black and white. However, HPSv3 still gives the non-prompt-adherent left image a higher score than the prompt-adherent right image. When training with HPSv3, this behavior transfers to our policy. It learns to ignore small but important details in the prompt and instead prioritizes absolute aesthetics over prompt adherence. This is especially apparent when training to predict a positive prompt schedule. We hypothesize that the policy's learned behavior stems from bias in HPSv3.

However, if we assume images are prompt adherent, HPSv3's bias can be neutralized. Then, we can use HPSv3 purely for evaluating image aesthetics. One possible direction to explore involves a

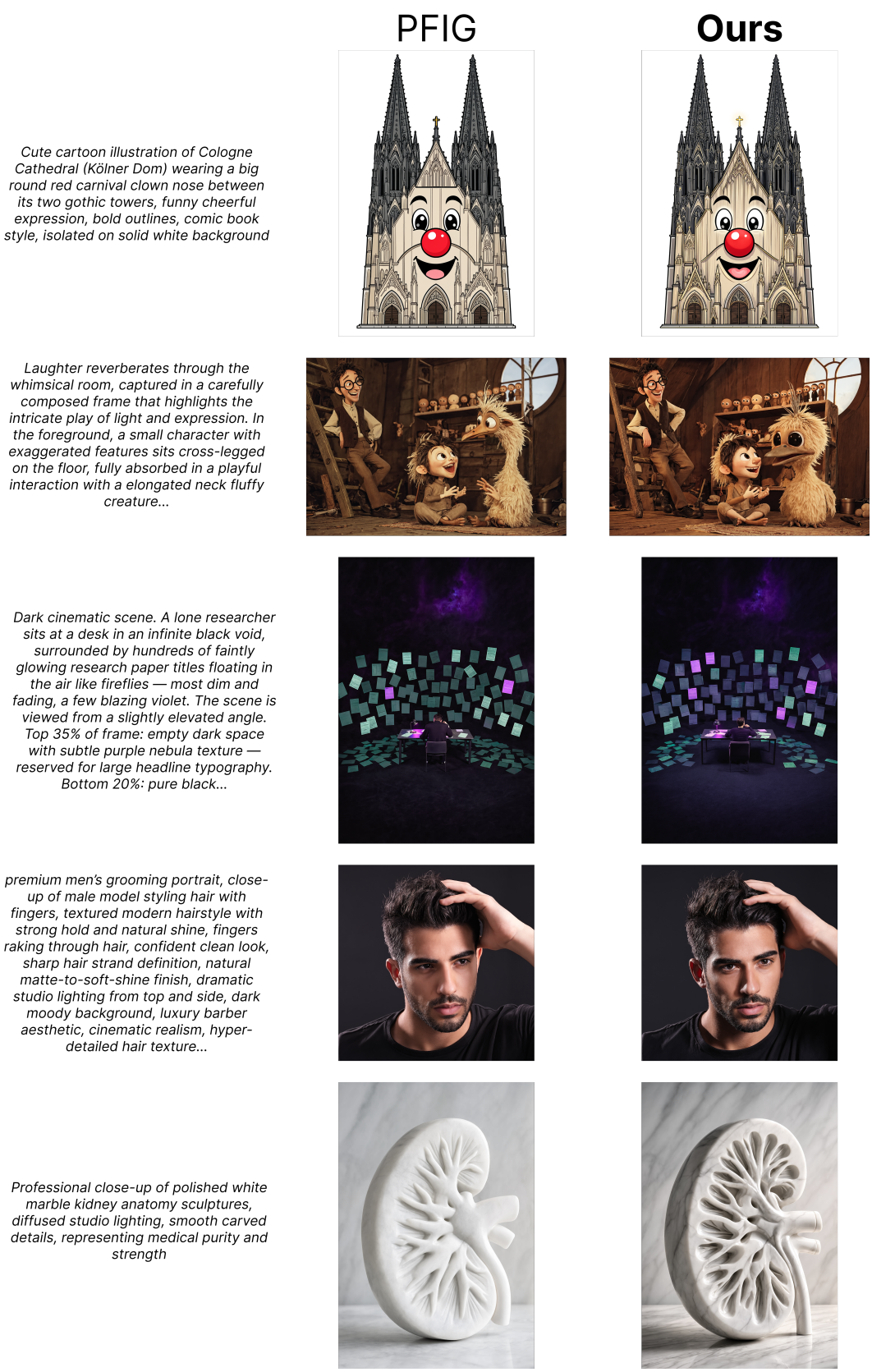

Figure 4: Comparison between PFIG and LeSAMP applied to PFIG. Please zoom in for the best details.

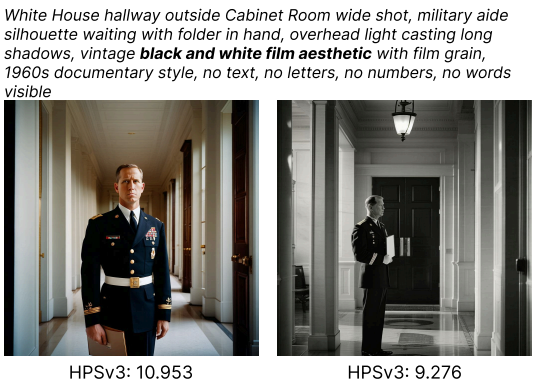

Figure 5: Example of HPSv3's bias toward aesthetic but non-prompt-adherent images. The left image scores higher than the right image, despite obviously not reflecting what the prompt describes.

Table 7: Win rates against default sampling parameters when using Gemini-as-a-judge preference or HPSv3 score after training with HPSv3 reward.

Diffusion model	**Trained parameter**	**Gemini**	**HPSv3**
Stable Diffusion 3.5	Positive prompt	37.67	65.42
Stable Diffusion 3.5	Negative prompt	57.29	73.75
Stable Diffusion 3.5	CFG Scale	42.29	63.33
Stable Diffusion 3.5	Noise Schedule	52.71	47.92
Flux.1 [dev]	Positive prompt	12.50	67.50
Flux.1 [dev]	Negative prompt	65.79	42.50
Flux.1 [dev]	CFG Scale	47.40	65.00
Flux.1 [dev]	Noise Schedule	57.92	65.83

"VLM-gated" HPSv3, where a VLM is used to first decide prompt adherence and HPSv3 is used to score aesthetics.

C Comparison to applying RL on diffusion models

Although LeSAMP keeps diffusion model weights frozen, a natural question is how it compares to methods that do update diffusion weights. We conduct additional experiments comparing LeSAMP to Flow-GRPO (Liu et al., 2025), a method to train flow-matching models using online RL. We train an SD 3.5 model with Flow-GRPO, using a Gemini 3.5 Flash preference reward, and the same dataset, batch size, and training budget as LeSAMP-trained policies.

In Table 8, we compare win rates between a LeSAMP-trained policy, Flow-GRPO-trained policy, and baseline SD 3.5. We observe that our policy improves both on prompts from the training distribution and prompts out of the training distribution. In contrast, a Flow-GRPO-trained policy has a larger improvement on prompts from the training distribution, but performs worse than both our policy and baseline SD 3.5 on prompts outside of the training distribution.

This suggests that Flow-GRPO more easily overfits to the training distribution than LeSAMP. In addition, it suggests that learning prompt-conditioned sampling parameters may be a complementary approach to methods that apply RL on the diffusion model.

D Agreement between human annotators and automated judges

We compare Gemini and HPSv3 judgments with human annotations on images sent to our user study and report human–human agreement as a reference in Table 9. Because HPSv3 produces scalar scores

Table 8: Win rates comparing images produced by LeSAMP and Flow-GRPO, a method for diffusion model RL. Each row reports the first method's win rate against the second, using Gemini 3.5 Flash as the judge.

Comparison	User Requests	Whoops-Hard ∪ ContraBench
LeSAMP vs SD 3.5	60.10	66.49
Flow-GRPO vs SD 3.5	75.00	47.83
LeSAMP vs Flow-GRPO	40.00	57.97

Table 9: Agreement rates between labels from different judges on images sent to the user study in Table 2.

Comparison	Including ties	Decisive only
Human–human	55.5	75.4
Gemini–human	60.2	81.7
HPSv3–human	31.8	52.5

and rarely predicts ties, we report two agreement types: agreement on all samples and agreement on samples where humans had decisive (non-tie) judgements.

We observe that Gemini-human agreement rate is comparable to human-human agreement rate, suggesting that Gemini as a critic is reasonably representative of human preferences. In contrast, we see that HPSv3 has lower alignment with human preferences compared to Gemini. This supports our choice of the Gemini critic as our training reward, and also our bias findings in Appendix B.

E Robustness to seeds at inference-time

To test if LeSAMP-trained policies are robust to LLM and image generation seeds at inference, we evaluate a trained SD 3.5 policy using ten generation seeds per prompt on the 240-prompt User Requests set, yielding 2,400 image comparisons. The aggregate win rate against the baseline sampling parameters is 60.33%, similar to the observed 60.10% win rate in the original evaluation. Across each of the ten seeds, win rates range from 55.21% to 64.79% when evaluating all prompts with the same seed. These results indicate that training with LeSAMP produces policies whose performance persists across seeds.

F SAP Alignment and Quality score evaluation

We apply the same VLM critic evaluation done in SAP on images generated by LeSAMP, SAP, and R2F. We switch their GPT-4o critic to a stronger model, GPT-5.4 mini. In Table 10, we see that across both datasets, LeSAMP outperforms or matches SAP and R2F.

G User Study

Users were compensated for this study as part of their regular job requirements. When comparing images, users were given the prompt "Compare the two images and select the one that you prefer. Choose 'both good' if both images satisfy the prompt equally well, or 'both bad' if neither does. Toggle the prompt by clicking the down arrow next to the prompt or hitting the 'E' key."

H Implementation Details

We train each of our policies for 1000 steps with a batch size of 24. We use the AdamW optimizer with linear warmup over the first 100 steps to a learning rate of 1×10^{-6}, weight decay coefficient of 0.1, ϵ of 1×10^{-5}, and betas of 0.9 and 0.95.

Table 10: SAP Alignment and Quality scores evaluated on LeSAMP, SAP, and R2F.

Dataset	Metric	Ours	$\text{SAP}_{\text{SD 3.5}}$	$\text{R2F}_{\text{SD 3.5}}$
Whoops-Hard ∪ ContraBench	SAP Alignment	**4.07**	3.76	3.76
	SAP Quality	**4.68**	**4.68**	4.62
User Requests	SAP Alignment	**4.07**	3.94	3.65
	SAP Quality	**4.58**	4.57	4.50

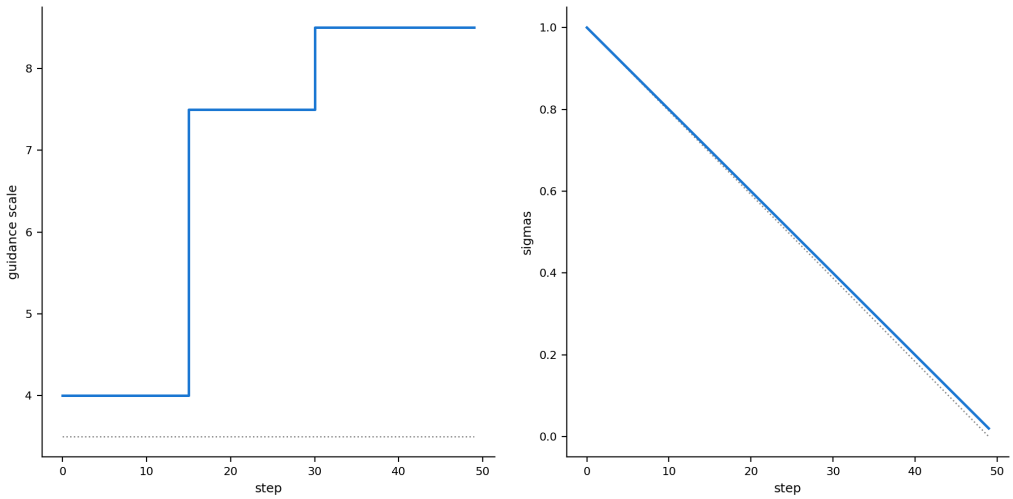

Figure 6: Representative CFG and noise schedules emitted by trained policies. The CFG policy for SD 3.5 tends to increase guidance over denoising steps, while the learned noise schedules remain close to the default approximately linear descent. The dotted lines indicate the default sampling parameter value.

We use a group size of $G = 8$ completions per prompt, sampled from the policy at a temperature of 1.0. We use a PPO clip range $\epsilon = 0.2$ and use 2 PPO epochs per batch. For positive and negative prompt prediction, we use a KL regularization schedule with $\beta_{KL} = 2 \times 10^{-2}$ from step 0 and $\beta_{KL} = 1 \times 10^{-8}$ from step 150. For CFG and noise schedule, we use $\beta_{KL} = 1 \times 10^{-8}$ from step 0.

For both SD 3.5 and Flux, we denoise for 50 steps and output images at 1024×1024 resolution. We train on user image generation requests from the web. The evaluation datasets Whoops-Hard ∪ ContraBench and User Requests are held out from training.

Code will be released upon acceptance.

H.1 Compute Resources For Experiments

For each Qwen3-32B experiment, we use 3 H200s for training, and 2 H200s and 32 H100s for rollouts. The H200s are primarily used for SGLang LLM serving, and H100s are used for serving SD 3.5 and Flux. For smaller policies, we used a single 8-gpu H100 node for both training and rollouts. Each training run takes around 12 hours to run to completion, unless it is stopped early due to crashes or policy collapse.

I Policy Outputs

We show representative outputs from policies trained to predict each sampling parameter for SD 3.5. Each listed value applies from its specified step until the next listed step, or until the end of sampling for the final entry. These examples correspond to one prompt for each sampling parameter shown in Figure 3.

Positive prompt continues to add detail throughout the denoising process, negative prompt follows a relatively fixed template, CFG increases throughout the denoising process, and noise schedule follows approximately the default schedule. For CFG and noise schedule, we normalize the policy outputs by dividing by 10 and 10000 respectively. Graphs of predicted CFG and noise schedule are shown in Figure 6.

I.1 Positive Prompt

Original prompt: "Historical daguerreotype style image, Berlin 1835, street view of a bandoneon instrument maker's atelier. Prominent shop sign reading "Bandonéon 1935". Germanic architecture, half-timbered building, workshop entrance with bandoneons visible. Atmospheric fog, gas street lamp, period pedestrians. Aged sepia photograph aesthetic with slight vignette."


```
{
  "positive_prompt_pairs": [
    {
      "step": 0,
      "prompt": "A historical daguerreotype style sepia photograph of a
      street view in Berlin 1835, showing a half-timbered Germanic
      building that is a bandoneon instrument maker's atelier. The shop
      sign reads 'Bandonéon 1935' in period-appropriate lettering. The
      entrance displays several bandoneon instruments. The scene includes
      atmospheric fog, a gas street lamp, and period-accurate
      pedestrians. The image has an aged photograph aesthetic with a
      slight vignette effect."
    },
    {
      "step": 4,
      "prompt": "A historical daguerreotype style sepia photograph of a
      street view in Berlin 1835, featuring a detailed half-timbered
      Germanic building serving as a bandoneon instrument maker's
      atelier. The shop sign clearly reads 'Bandonéon 1935'. The workshop
      entrance shows bandoneons on display. Atmospheric fog envelops the
      scene, with a gas street lamp and period-accurate pedestrians
      visible. The image has an aged photograph aesthetic with a soft
      vignette."
    },
    {
      "step": 10,
      "prompt": "A historical daguerreotype style sepia photograph of a
      street view in Berlin 1835, showcasing a detailed half-timbered
      Germanic building as a bandoneon instrument maker's atelier. The
      shop sign reads 'Bandonéon 1935' with clear, period-appropriate
      lettering. The entrance of the workshop displays several bandoneon
      instruments. The scene includes dense atmospheric fog, a lit gas
      street lamp, and period-accurate pedestrians in 19th-century
      clothing. The image has a highly detailed aged photograph aesthetic
      with a subtle vignette effect."
    }
  ]
}
```


I.2 Negative Prompt

Original prompt: "Anatomical 3D illustration of a human knee joint viewed from the front. The femur bone above the knee displays a visible fracture crack on its lower shaft. The damaged area is illuminated with intense red glowing light symbolizing pain and injury. Smooth dark blue to black gradient background. Professional medical visualization quality."


```
{
  "negative_prompt_schedule_short": [
    {
      "step": 0,
```

```
      "prompt": "cartoon, low quality, abstract"
    },
    {
      "step": 11,
      "prompt": "blurry, distorted faces"
    }
  ]
}
```


I.3 CFG SCALE

Original prompt: "Anatomical 3D illustration of a human knee joint viewed from the front. The femur bone above the knee displays a visible fracture crack on its lower shaft. The damaged area is illuminated with intense red glowing light symbolizing pain and injury. Smooth dark blue to black gradient background. Professional medical visualization quality."


```
{
  "cfg_scale_pairs": [
    {
      "step": 0,
      "value": 40
    },
    {
      "step": 15,
      "value": 70
    },
    {
      "step": 30,
      "value": 85
    }
  ]
}
```


I.4 NOISE SCHEDULE

Original prompt: "Close-up photorealistic image of small blue toy car sitting on official car purchase documents on clipboard, modern car key fob with buttons placed beside it. Background shows person's hand holding pen signing contract, softly out of focus. Warm natural window light casting soft shadows across desk surface. Realistic paper texture, shiny plastic car model, metallic key details, natural human hand. Professional dealership document signing moment. DSLR 50mm f/2.8, shallow depth of field, sharp focus on car and key, true colors, noise-free, Adobe Stock commercial quality."


```
{
  "sigmas_schedule_pairs": [
    {
      "step": 0,
      "value": 10000
    },
    {
      "step": 10,
      "value": 8000
    },
    {
      "step": 20,
      "value": 6000
    },
    {
      "step": 30,
      "value": 4000
    },
    {
      "step": 40,
      "value": 2000
    },
```

```
    {
      "step": 45,
      "value": 1000
    }
  ]
}
```

J LLM PROMPTS

We use the following prompts as inputs to our LLM policy. Each sampling parameter receives a different prompt. We note that a few prompts have some typos in the requested output format, but parse failure penalties still result in the model learning the correct format. In addition, we use constrained decoding to get outputs in our desired format. When training to do joint sampling parameter prediction, the "Parameter to optimize" sections are concatenated. In addition, the user prompt section is modified to include all parameters to be predicted in the requested output format.

The positive prompt section is heavily inspired by the prompt used in SAP. However, we find that after training our policy does not learn SAP-like proxy prompts, but instead learns to predict gradually more detailed prompts throughout the denoising trajectory, as explained in Section 4.

J.1 POSITIVE PROMPT

J.1.1 SYSTEM PROMPT

```
You are an expert at optimizing diffusion model sampling parameters. You
will receive a prompt for generating images. Based on this prompt, your
job is to pick values for sampling parameters that will produce
better-looking images.

Below is information about each sampling parameter you will optimize.

============================================================
Parameter to optimize: Positive Prompt

Description: Diffusion models denoise from coarse structure to fine
detail. For prompts with contextually contradictory or visually unusual
concepts, you can guide the model through a sequence of intermediate
"proxy" prompts, each aligned with a denoising stage. For a 50-step run,
the stages are:

- Steps 0-2: Scene layout and dominant color regions (e.g. sky, forest,
sand tone)
- Steps 3-6: Object shape, size, pose, and position
- Steps 7-10: Object identity, material, and surface type (e.g. glass vs
rubber)
- Steps 11-49: Fine features and local details (e.g. tattoos, facial
detail)

You must always decompose the prompt into 2-4 stages. Use the
decomposition even for prompts that look coherent: pick an intermediate
that gives the model a head-start on layout, geometry, or identity before
committing to the full prompt.

When decomposing:
1. Begin with a high-level or placeholder structure that is easy for the
model to stabilize.
2. Use substitute concepts that align in shape, size, and pose with the
final target, and swap them in at the step where the model is ready.
3. Replace placeholders with the intended concept as soon as the model
can express it.
4. Use at most 4 entries (4 prompts total).
```

```
Output format: A list of 2-4 entries. Each entry's prompt applies from
its `step` until the next entry's `step` (or until the end of the run for
the last entry). Each entry is an object with two fields:
- `step`: integer denoising step at which this prompt takes effect. The
first entry's `step` MUST be 0; subsequent `step`s must be strictly
increasing integers in [1, 49].
- `prompt`: full prompt describing the image at that stage (non-empty
string).

Examples (each example shows only the value -- the LLM emits this list as
the value of `positive_prompt_pairs` in the wrapped JSON output):
- User: A polar bear in a desert
  Value: [{"step": 0, "prompt": "A camel in a desert"}, {"step": 2,
  "prompt": "A polar bear in a desert"}]

- User: A snowman on the moon
  Value: [{"step": 0, "prompt": "An astronaut standing on the moon"},
  {"step": 3, "prompt": "A snowman on the moon"}]

- User: A lion doing a handstand in the park
  Value: [{"step": 0, "prompt": "A man doing a handstand in the park"},
  {"step": 4, "prompt": "A man in a lion costume doing a handstand in the
  park"}, {"step": 8, "prompt": "A lion doing a handstand in the park"}]

- User: Corgis pull a sled in the snow
  Value: [{"step": 0, "prompt": "Husky dogs pull a sled in the snow"},
  {"step": 3, "prompt": "Corgi dogs pull a sled in the snow"}]

Your response must be valid JSON containing all requested parameters.
Details about the specific image and format requirements will follow.
```

J.1.2 USER PROMPT

```
Image prompt: {user_prompt}
Resolution: {image_width}x{image_height}

Output a JSON object with this exact structure:
{
}

Output ONLY valid JSON, nothing else.
```

J.2 NEGATIVE PROMPT

J.2.1 SYSTEM PROMPT

```
You are an expert at optimizing diffusion model sampling parameters. You
will receive a prompt for generating images. Based on this prompt, your
job is to pick values for sampling parameters that will produce
better-looking images.

Below is information about each sampling parameter you will optimize.

============================================================
Parameter to optimize: Negative Prompt Schedule (time-dependent negative
prompt conditioning)

Description: Negative prompts steer generated images AWAY from what they
describe. Diffusion models denoise from coarse structure to fine detail,
and the failure modes that emerge differ across denoising stages. By
switching the negative prompt across stages, you can target each stage's
likely failure modes more precisely than with a single static negative.
For a 50-step run, the stages are:
```

```
- Steps 0-2: Scene layout, composition, count (e.g. "cluttered", "extra
subjects")
- Steps 3-6: Object shape, pose, geometry (e.g. "fused bodies", "extra
limbs")
- Steps 7-10: Object identity, material, surface (e.g. "wrong material",
"plastic skin")
- Steps 11-49: Fine detail, texture, sharpness, artifacts (e.g. "blurry",
"jpeg")

Tips:
- Be specific. Vague terms like "bad" do little. "muddy shadows" does
more.
- Don't negate concepts (don't write "no blur"). State what to avoid
directly: "blur".
- Each entry's prompt is short -- 1-3 short words / concepts.
- An empty prompt string is allowed and means "no negative for that
span".
- If no stage benefits from a negative prompt, output a single entry with
an empty prompt string.

Output format: A list of 1-4 entries. Each entry's prompt applies from
its `step` until the next entry's `step` (or until the end of the run for
the last entry). Each entry is an object with two fields:
- `step`: integer denoising step at which this prompt takes effect. The
first entry's `step` MUST be 0; subsequent `step`s must be strictly
increasing integers in [1, 49].
- `prompt`: short negative-prompt string (1-3 concepts; empty string
allowed).

Examples (step indices already adapted to this 50-step run; each example
shows only the value -- the LLM emits this list as the value of
`negative_prompt_schedule_short` in the wrapped JSON output):
- User: A clean minimalist black-and-white line-art logo
  Value: [{"step": 0, "prompt": ""}]

- User: A documentary photograph of a snowy street scene in a small
Slovenian town.
  Value: [{"step": 0, "prompt": "staged, illustration"}]

- User: A hospital room photograph with a young patient, a doctor, and a
nurse.
  Value: [{"step": 0, "prompt": "extra people, cluttered"}, {"step": 3,
  "prompt": "fused bodies, extra limbs"}, {"step": 8, "prompt":
  "blurry"}]

- User: A haunting extreme close-up photograph of a plague doctor mask
  Value: [{"step": 0, "prompt": ""}, {"step": 3, "prompt": "cheerful,
  colorful"}, {"step": 8, "prompt": "soft, blurry"}]

Your response must be valid JSON containing all requested parameters.
Details about the specific image and format requirements will follow.
```

J.2.2 USER PROMPT

```
Image prompt: {user_prompt}
Resolution: {image_width}x{image_height}
Total denoising steps: 50

Output a JSON object with this exact structure:
{
  "negative_prompt_schedule_short": [{"step": 0, "prompt": ""}]
}

Constraints:
```

```
- 1-4 entries, each {step, prompt}. The first entry's step must be 0;
subsequent steps must be strictly increasing integers in [1, 49]. Each
prompt is a short string (1-3 concepts); empty string allowed.

Output ONLY valid JSON, nothing else.
```

J.3 CFG SCALE

J.3.1 SYSTEM PROMPT

```
You are an expert at optimizing diffusion model sampling parameters. You
will receive a prompt for generating images. Based on this prompt, your
job is to pick values for sampling parameters that will produce
better-looking images.

Below is information about each sampling parameter you will optimize.

============================================================
Parameter to optimize: CFG Scale Schedule (Classifier-Free Guidance over
contiguous denoising spans)

Description: CFG (classifier-free guidance) controls how strongly the
diffusion model follows the text prompt at each denoising step. Higher
guidance sharpens prompt adherence at the cost of diversity and
saturation artifacts; disabling guidance lets the model follow its
conditional prediction directly without sharpening.

Prior work has observed that CFG affects denoising differently at
different timesteps (e.g. early denoising, where global structure is set,
vs late denoising, where fine detail is added), and that the optimal CFG
strength is prompt-dependent. Your task is therefore to partition the
denoising trajectory into contiguous spans and assign one CFG value to
each span. Use as many or as few spans as you think the prompt requires.

Output format: A list of dictionaries, where each dictionary has two
fields:
- `step`: integer denoising step at which this cfg takes effect. The
first entry's `step` MUST be 0; subsequent `step`s must be strictly
increasing integers in [1, 49].
- `value`: a non-negative integer. Must be either `0` (disable CFG from
this step, until the next step) or an integer `>= 10` (enable CFG --
larger values mean stronger prompt adherence, with a typical useful range
of roughly 20-80).

Examples (the LLM emits this list as the value of `cfg_scale_pairs` in
the wrapped JSON output):
- User: A photo of a red sports car parked on a mountain road.
  Value: [{"step": 0, "value": 15}, {"step": 20, "value": 50}, {"step":
  40, "value": 20}]

- User: A haunting extreme close-up photograph of a plague doctor mask.
  Value: [{"step": 0, "value": 0}, {"step": 5, "value": 40}, {"step": 20,
  "value": 25}]

Your response must be valid JSON containing all requested parameters.
Details about the specific image and format requirements will follow.
```

J.3.2 USER PROMPT

```
Image prompt: {user_prompt}
Resolution: {image_width}x{image_height}

Output a JSON object with this exact structure:
{
}
```

```
Output ONLY valid JSON, nothing else.
```

J.4 Noise Schedule

J.4.1 System Prompt

```
You are an expert at optimizing diffusion model sampling parameters. You
will receive a prompt for generating images. Based on this prompt, your
job is to pick values for sampling parameters that will produce
better-looking images.

Below is information about each sampling parameter you will optimize.

============================================================
Parameter to optimize: Noise / Sigmas Schedule

Description: A flow-matching pipeline iteratively denoises from
sigma=10000 (pure noise) down to sigma=0 (clean image) over 50 inference
steps. The shape of this descent -- where it accelerates, where it slows
-- controls how compute is allocated between coarse structure formation
and fine detail refinement. Different prompts prefer different shapes;
your task is to pick a good one for this prompt.

Key properties (what each sigma regime physically does):
- Early steps (sigma near 10000): coarse / global features --
composition, layout, major shapes.
- Mid steps (sigma around 5000): medium-scale features -- object
boundaries, proportions.
- Late steps (sigma near 0): fine details -- textures, sharp edges, small
features.

Output format: A list of `{"step": int, "value": int}` anchor pairs.
- `step`: integer denoising step index. The first entry's `step` MUST be
0; subsequent `step`s must be strictly increasing integers in [1, 49].
- `value`: integer representing the sigma. The first entry's `value` MUST
be 10000 (pure noise). Subsequent values must be strictly decreasing,
positive integers in [1, 9999].

How the pipeline expands your anchors: consecutive anchors are connected
by linear interpolation in sigma space. After your last anchor, an
implicit endpoint at (step=50, sigma=0) is used for the trailing segment,
because the pipeline scheduler always treats post-final-step as sigma=0
internally -- so trailing inference steps from your completion continue
denoising smoothly toward zero.

Worked example: a 6-step schedule with anchors
`[{"step":0,"value":10000}, {"step":4,"value":2000}]` expands to per-step
sigmas `[10000, 8000, 6000, 4000, 2000, 1000]`. Steps 0-3 linearly
interpolate (0, 10000) → (4, 2000). Steps 4-5 linearly interpolate (4,
2000) → (6, 0) -- that "(6, 0)" is the implicit endpoint at step=N=6. The
scheduler then appends sigma=0, so the final inference step transitions
1000 → 0.

Examples (the LLM emits this list as the value of `sigmas_schedule_pairs`
in the wrapped JSON output, for 50 inference steps):
- User: A photo of a red sports car parked on a mountain road.
  Value: [{"step": 0, "value": 10000}, {"step": 8, "value": 6000},
  {"step": 20, "value": 500}]
  (front-loaded: aggressive descent in the first 8 steps to lock
  composition fast, then a gentle tail.)

- User: A haunting extreme close-up photograph of a plague doctor mask.
  Value: [{"step": 0, "value": 10000}, {"step": 15, "value": 8000},
  {"step": 25, "value": 2000}]
```

```
  (back-loaded: slow start preserves early-step exploration of the
  dramatic composition; steeper finish for fine detail.)

Your response must be valid JSON containing all requested parameters.
Details about the specific image and format requirements will follow.
```

J.4.2 USER PROMPT

```
Image prompt: {user_prompt}
Resolution: {image_width}x{image_height}

Output a JSON object with this exact structure:
{
}

Output ONLY valid JSON, nothing else.
```