Designing Instance-Level Sampling Schedules via REINFORCE with James-Stein Shrinkage

Supplementary Material

A. Proofs

A.1. Proof of Proposition 3.1.

Proof. First, we show the unbiasedness of the baseline. We abbreviate the policy as $\pi_\theta(\tau) \equiv \pi_\theta(\tau \mid \mathbf{x}_T, \mathbf{c})$,

$$g(\tau) \;=\; \nabla_\theta \log \pi_\theta(\tau), \qquad w(\tau) \;=\; \|g(\tau)\|^2,$$

and let $r(\tau)$ denote the scalar reward. The single–sample REINFORCE estimator with a scalar baseline b is

$$\widehat{G}(b) \;=\; (r(\tau) - b)\, g(\tau). \tag{11}$$

By swapping integral and gradient operators

$$\mathbb{E}_{\tau\sim\pi_\theta}[g(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(\tau)] = \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[1] = 0. \tag{12}$$

Hence the estimator *remains unbiased* for any baseline b:

$$\mathbb{E}[\widehat{G}(b)] = \mathbb{E}[(r-b)g] = \mathbb{E}[rg] - b\,\mathbb{E}[g] = \mathbb{E}[rg]. \tag{13}$$

Next, we study the variances of the estimator. The covariance of $\widehat{G}(b)$ is

$$\mathrm{Var}[\widehat{G}(b)] = \mathbb{E}[(r-b)^2 g g^\top] - \mathbb{E}[(r-b)g]\,\mathbb{E}[(r-b)g]^\top.$$

Since $\mathbb{E}[(r-b)g] = \mathbb{E}[rg]$ does not depend on b, minimizing the total scalar variance (trace) is equivalent to minimizing

$$V(b) \;=\; \mathbb{E}\big[(r-b)^2\, w(\tau)\big] \;-\; \text{const}, \;\; \text{Recall}\; w(\tau) = \|g(\tau)\|^2.$$

The function $V(b)$ is differentiable and strictly convex in b. Differentiating and setting the derivative to zero yields

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}b} V(b) &= \frac{\mathrm{d}}{\mathrm{d}b}\,\mathbb{E}\big[(r-b)^2 w\big] = \mathbb{E}\big[2(b-r)w\big] \\ &= 2\Big(b\,\mathbb{E}[w] - \mathbb{E}[rw]\Big) = 0,\end{aligned}$$

which gives the unique minimizer

$$b^* = \frac{\mathbb{E}[r(\tau)\, w(\tau)]}{\mathbb{E}[w(\tau)]} = \frac{\mathbb{E}\big[r(\tau)\, \|\nabla_\theta \log \pi_\theta(\tau)\|^2\big]}{\mathbb{E}\big[\,\|\nabla_\theta \log \pi_\theta(\tau)\|^2\big]}. \tag{14}$$

Thus b^* minimizes the trace of $\mathrm{Var}[\widehat{G}(b)]$. □

Remarks. (i) Because b is a scalar shared across all components of $g(\tau)$, the same minimizer b^* arises whether one minimizes each component variance or the trace. (ii) Our derivation shares the spirit of state-dependent baselines derived for GPOMDP in Greensmith et al. [6], which optimize a different variance criterion conditioned on Markov states. (iii) When the policy becomes nearly deterministic for a fixed context $(\mathbf{x}_T, \mathbf{c})$, the variation of $\nabla_\theta \log \pi_\theta(\tau)$ is small. In this regime, Eq. (14) reduces to the mean reward $\mathbb{E}[r(\tau) \mid \mathbf{x}_T, \mathbf{c}]$, which motivates the empirical approximations used in the main text.

A.2. Proof of Theorem 3.2.

Proof. Now we prove the theoretical results related to the James-Stein reward baseline.

Bayesian optimality. Recall that we have the random–effects model in Eq. (6),

$$r^{(c,i)} \mid \mu_c \sim \mathcal{N}(\mu_c, \sigma^2), \qquad \mu_c \sim \mathcal{N}(\mu_0, \delta^2), \tag{15}$$

the per–context mean satisfies

$$\bar{r}_c \;=\; \frac{1}{K_c}\sum_{i=1}^{K_c} r^{(c,i)}, \qquad \bar{r}_c \mid \mu_c \sim \mathcal{N}(\mu_c, \sigma^2/K_c). \tag{16}$$

By normal–normal conjugacy, the posterior distribution $\mu_c \mid \{r^{(c,i)}\}_{i=1}^{K_c}$ is Gaussian with mean

$$\mathbb{E}[\mu_c \mid \{r^{(c,i)}\}] = (1-\alpha_c^*)\,\bar{r}_c + \alpha_c^*\,\mu_0, \tag{17}$$

where the (population) shrinkage coefficient is

$$\alpha_c^* = \frac{\sigma^2/K_c}{\sigma^2/K_c + \delta^2}. \tag{18}$$

For any data–dependent predictor t_c of μ_c, the posterior expected squared error given the observed rewards is

$$\begin{aligned}\mathbb{E}\big[(\mu_c - t_c)^2 \,\big|\, \{r^{(c,i)}\}\big] &= \mathbb{E}[\mu_c^2 \mid \{r^{(c,i)}\}] \\ &\quad - 2t_c\,\mathbb{E}[\mu_c \mid \{r^{(c,i)}\}] + t_c^2,\end{aligned} \tag{19}$$

which, as a quadratic in t_c, is minimized at $t_c = \mathbb{E}[\mu_c \mid \{r^{(c,i)}\}]$. Thus the Bayes–optimal predictor of μ_c under squared error is the posterior mean in Eq. (17), i.e., the convex combination of the within–context mean $\bar{r}_c$ and the population mean μ_0.

In our REINFORCE setting, we require a baseline that is independent of the current rollout (c, i) to preserve unbiasedness. We therefore replace $\bar{r}_c$ and the global mean μ_0 by their leave–one–out analogues:

$$\bar{r}_c^{(-i)} = \frac{1}{K_c - 1} \sum_{j \neq i} r^{(c,j)},$$
$$\bar{r}_{\cdot\cdot}^{(-c,-i)} = \frac{1}{(\sum_{c'} K_{c'}) - 1} \left(\sum_{c'=1}^{B} \sum_{j=1}^{K_{c'}} r^{(c',j)} - r^{(c,i)} \right). \tag{20}$$

The ideal Bayes baseline for rollout (c, i) is then

$$b_{\text{Bayes}}^{(c,i)} = (1 - \alpha_c^*)\, \bar{r}_c^{(-i)} + \alpha_c^*\, \bar{r}_{\cdot\cdot}^{(-c,-i)}.$$

In practice, σ^2 and δ^2 are unknown. The method–of–moments estimators in Eq. (10) yield empirical counterparts $\widehat{\sigma}^2$ and $\widehat{\delta}^2$, which in turn define the empirical shrinkage weights

$$\widehat{\alpha}_c = \frac{\widehat{\sigma}^2/(K_c - 1)}{\widehat{\sigma}^2/(K_c - 1) + \widehat{\delta}^2}.$$

Substituting $\widehat{\alpha}_c$ and the leave–one–out means into the Bayes form above recovers exactly the empirical James–Stein baseline

$$b_{\text{JS}}^{(c,i)} = (1 - \widehat{\alpha}_c)\, b_{\text{RLOO}}^{(c,i)} + \widehat{\alpha}_c\, b_{\text{xctx}}^{(c,i)}$$

from Eq. (7), where both $b_{\text{RLOO}}^{(c,i)}$ and $b_{\text{xctx}}^{(c,i)}$ are computed in a leave–one–out way that excludes (c, i). Since $b_{\text{JS}}^{(c,i)}$ does not depend on the current reward $r^{(c,i)}$, the REINFORCE gradient remains unbiased. In summary, $b_{\text{JS}}^{(c,i)}$ by construction coincides with the empirical Bayes posterior mean of μ_c under the model (6), and is Bayes–optimal.

MSE improvement. Under the same random–effects model, consider the convex estimator

$$\tilde{\mu}_c(\alpha) = (1 - \alpha)\, \bar{r}_c^{(-i)} + \alpha\, \bar{r}_{\cdot\cdot}^{(-c,-i)}.$$

Assuming independence between the within–context and across–context noise, its mean–squared error satisfies

$$\text{MSE}(\tilde{\mu}_c(\alpha)) = (1 - \alpha)^2\, \frac{\sigma^2}{K_c - 1} + \alpha^2\, \delta^2,$$

which is minimized at

$$\alpha^\star = \frac{\sigma^2/(K_c - 1)}{\sigma^2/(K_c - 1) + \delta^2}.$$

Thus the same shrinkage coefficient $\alpha^\star$ that defines the posterior mean also uniquely minimizes the frequentist MSE of the baseline, strictly improving not only over the purely contextual baseline (RLOO), but also over all the convex combination between RLOO and the XCTX baselines including themselves. □

Remarks. If we collect the context-specific means into the vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_B)$ and likewise stack their noisy estimators, we obtain the classical homoscedastic normal-mean problem in dimension B. In that setting, James–Stein shrinkage estimators that shrink the vector of means toward a common target are known to uniformly dominate the unshrunk sample mean in total squared error risk whenever the dimension is at least three; see, e.g., James et al. [7]. In our notation, the "dimension" plays the role of the number of contexts B, which is the reason for the $B \geq 3$ condition mentioned in the main text. Our empirical Bayes baseline has the same shrinkage-to-a-global-mean structure as these classical estimators, but here we focus on the random–effects and REINFORCE viewpoints rather than re-proving the classical domination result.

B. Scheduler Network Architectures

We summarize the network architecture we used for implementing the scheduler network in Tab. 5. The scheduler is 0.1–1% the size (∼20M parameters) of the backbone networks used in Stable Diffusion and Flux model families, and incurs at most ∼ 0.1–0.3% extra end-to-end sampling compute in the step counts we used ranging from 5 to 80.

C. Experiment Settings

C.1. Synthetic Sanity Check for Baseline Variance

RLOO is an unbiased baseline but fragile in exactly the regime most relevant to our policy learning scenarios: limited number of rollouts per context and highly heterogeneous reward scales. To isolate this failure mode and evaluate variance reduction in a controlled setting, we construct a synthetic experiment where the policy is analytic, rewards follow a spiky random–effects model. We measure the variance of the REINFORCE gradient estimators since they are all unbiased. This setup cleanly reveals the effect of the proposed James–Stein (JS) baseline.

Setup. We adopt an analytic Dirichlet policy $\pi(\tau) = \text{Dirichlet}(\boldsymbol{\alpha})$ with concentration $\boldsymbol{\alpha} = 2\mathbf{1}_T$ and horizons $T \in \{4, 16, 64\}$. Its score function has a closed form:

$$\nabla_{\alpha_j} \log \pi(\tau) = \psi\Big(\sum_t \alpha_t\Big) - \psi(\alpha_j) + \log \tau_j,$$

where $\psi(\cdot)$ denotes the digamma function and $\log \tau_j$ is the elementwise logarithm of the j-th coordinate of τ. To mimic the two-level random–effects model used in our theoretical analysis, we generate synthetic rewards as

$$r^{(c,i)} = \mu_c + \varepsilon^{(c,i)} + \delta^{(c,i)},$$
$$\mu_c \sim \mathcal{N}(0, 1), \quad \varepsilon^{(c,i)} \sim \mathcal{N}\big(0, s_c^2\big),$$
$$s_c \sim \text{LogNormal}(0, 1), \quad \delta^{(c,i)} \sim \text{Bernoulli}(0.15) \times 8, \tag{21}$$

Table 5. **Network architecture for the Dirichlet schedule policy $\alpha_\theta(x, c)$.** The network takes an image-shaped noise tensor x, a sequence of text embeddings e_{text}, and an optional pooled text embedding $\bar{e}_{\text{text}}$ (consistent with SD-XL, SD3.5M/L and FLUX models), and outputs Dirichlet parameters $\boldsymbol{\alpha} \in \mathbb{R}_+^{L+1}$ over L schedule intervals plus one skipped interval. Default network architecture hyperparameters: `text_embed_dim` $d_{\text{text}} = 2048$, `pooled_text_embed_dim` $d_{\text{pool}} = 1280$, `image_encoder_depth` $n_{\text{conv}} = 2$, `image_encoder_width` $d_{\text{conv}} = 32$, `attention_dim` $d_{\text{attn}} = 256$, `cross_attention_heads` $n_{\text{attn}} = 4$, `number_of_transformer_blocks` $N_b = 4$, `hidden_dim` $d_h = 256$, `num_mlp_layers` $n_{\text{mlp}} = 2$.

Layers	Output size	Note
Inputs		
Input: x	$B \times H \times W \times C$	image-shaped noise
Input: e_{text}	$B \times L_{\text{text}} \times d_{\text{text}}$	token text emb.; d_{text}=2048
Input: $\bar{e}_{\text{text}}$	$B \times d_{\text{pool}}$	optional pooled emb.; d_{pool}=1280
Image encoder & cross-attention blocks		
Image encoder conv stack (inside block i, layers $j = 0, \ldots, n_{\text{conv}}-1$)	$B \times H_i \times W_i \times C_i$	n_{conv} conv layers, 3×3, stride 1 GroupNorm, SiLU $C_i = d_{\text{conv}} \cdot 2^{\min(4,\, i+j)}$
Flatten spatial dims	$B \times (H_i W_i) \times C_i$	prepare for attention
Query projection (Dense)	$B \times (H_i W_i) \times C_i$	applied to image features (`query_projection`)
Key/Value projection (Dense)	$B \times L_{\text{text}} \times C_i$	applied to e_{text} (`key_value_projection`)
MultiHeadDotProductAttention	$B \times (H_i W_i) \times C_i$	cross-attn: query from image, key/value from text, n_{attn} heads
Residual + LayerNorm	$B \times (H_i W_i) \times C_i$	LayerNorm(query + attn_output) (`cross_attn_norm`)
Reshape to image grid	$B \times H_i \times W_i \times C_i$	output of `SchedulerTransformerBlock`
Global avg. pooling (per block)	$B \times C_i$	mean over (H_i, W_i) for feature fusion
Downsample Conv 3×3, stride 2 + GroupNorm + SiLU (between blocks)	$B \times H_{i+1} \times W_{i+1} \times C_{i+1}$	from reshaped image grid, for $i < N_b-1$; $C_{i+1} = d_{\text{conv}} \cdot 2^{\min(4,\,(i+1)\, n_{\text{conv}}-1)}$
Feature pyramid fusion		
Concat. over blocks	$B \times \sum_i C_i$	feature pyramid $\{\text{block } i\}_{i=0}^{N_b-1}$
Concat. with $\bar{e}_{\text{text}}$	$B \times \left(\sum_i C_i + d_{\text{pool}}\right)$	only if pooled emb. used
MLP Dirichlet head		
Dense, SiLU $\times\, (n_{\text{mlp}} - 1)$	$B \times d_h$	default $d_h = 256$
Dense (Dirichlet logits head)	$B \times (L{+}1)$	$L =$ `num_timesteps`
Softplus $+10^{-3}$	$B \times (L{+}1)$	$\boldsymbol{\alpha} = \text{softplus}(\cdot){+}10^{-3}$

where μ_c acts as the context-level effect (analogous to $\mu_0 + \xi^{(c)}$ in Eq. (6)) and $\varepsilon^{(c,i)}$ represents within-context noise. The additional sparse term $\delta^{(c,i)}$ introduces occasional outliers, creating a heavy-tailed, heteroskedastic reward landscape that stresses RLOO in the small-K regime.

We sweep through different settings for (B, K, L). For each (B, K, L) we generate 500 i.i.d. batches, compute

$$\hat{g}(b) = \frac{1}{BK} \sum_{c=1}^{B} \sum_{i=1}^{K} \left(r^{(c,i)} - b^{(c,i)}\right) \nabla_\alpha \log \pi(\tau^{(c,i)}),$$

once with RLOO and once with JS, and report the empirical per-dimension variance $\frac{1}{L} \sum_{j=1}^{L} \text{Var}(\hat{g}_j)$. We sweep $B \in \{8, 16, 32\}$ and $K \in \{2, 4, 8, 16\}$.

Observations. As shown in Tab. 6, the James–Stein (JS) baseline delivers uniformly lower gradient variance than RLOO across all batch sizes, rollout counts, and horizons. For the most challenging regime ($K = 2$), JS reduces variance by roughly 45–50%, effectively matching the stability one would expect from doubling the rollout count—without any additional computation. Even with moderate rollouts ($K = 4$), variance drops by 20–25%, demonstrating that the improvement is systematic rather than case-specific. Across all settings, JS remains consistently superior, reflecting its adaptive shrinkage across both within- and cross-context variability. In practical terms, this means JS acts as a computationally free variance amplifier: it increases the *effective* number of rollouts available to policy-gradient estimation, providing a principled and efficient replacement for

Table 6. Synthetic variance sanity check. Average per-dimension variance of REINFORCE for an analytic Dirichlet policy. Rewards follow Eq. (21). Entries are RLOO → JS. JS consistently gives gain over RLOO baselines for various settings.

K	B	L=4	L=16	L=64
2	8	1.055 → **0.612**	1.445 → **0.725**	1.148 → **0.625**
	16	0.479 → **0.285**	0.663 → **0.339**	0.606 → **0.339**
	32	0.261 → **0.140**	0.328 → **0.174**	0.310 → **0.165**
4	8	0.393 → **0.284**	0.402 → **0.307**	0.386 → **0.299**
	16	0.175 → **0.143**	0.193 → **0.154**	0.210 → **0.163**
	32	0.086 → **0.066**	0.099 → **0.079**	0.106 → **0.084**
8	8	0.140 → **0.126**	0.191 → **0.170**	0.182 → **0.166**
	16	0.071 → **0.064**	0.099 → **0.090**	0.086 → **0.078**
	32	0.038 → **0.034**	0.042 → **0.038**	0.045 → **0.040**
16	8	0.072 → **0.070**	0.081 → **0.078**	0.094 → **0.089**
	16	0.031 → **0.030**	0.038 → **0.037**	0.039 → **0.038**
	32	0.017 → **0.016**	0.020 → **0.019**	0.020 → **0.019**

Table 7. Learning rates for schedule training on HPDv2 prompts.

Backbone	L=5	L=10	L=20	L=40	L=80
SD-XL	5e−5	5e−5	5e−5	5e−5	5e−5
SD3.5-M	5e−5	2.5e−5	2.5e−5	1.25e−5	5e−5
SD3.5-L	5e−5	5e−5	5e−5	5e−5	5e−5
Flux-Dev	2.5e−5	5e−5	2.5e−5	5e−5	5e−5

RLOO in rollout-expensive training regimes. We visualize a side-by-side comparison in Fig. 2a for the $B = 32, L = 16$ case with different number of rollouts K.

C.2. General Text-to-Image Experiments on HPDv2

General settings. For all the experiments, we consistently use the AdamW [15] optimizer for learning scheduler parameters. We set the `weight_decay` to $1e-4$ throughout experiments, tune the learning rates (see Tab. 7), and use default values for the remaining hyperparameters. We apply the gradient clipping with a maximum norm of 1. For all models we consistently use a rollout number of 2, and set the output resolution to 1024×1024. Due to memory limit, we use a batch size of 32 for smaller models like SD-XL or SD-3.5M, and a batch size of 16 for larger ones like SD-3.5L and Flux-Dev. We use a guidance weight of 7.5 for SD-XL and 5.0 for SD3.5-M/L and Flux models. For SD-XL model we use the DDIM solver [21] for per step denoising update; for the flow-matching-based SD3.5 and Flux models we use the Euler solver.

TPDM baseline. We reimplement the TPDM baseline using (i) the same network architecture we used, and (ii) the same set of seeds, hyperparmeters and base model settings for training. We follow the customized PPO implementation in TPDM with no ratio clipping.

Table 8. Learning rates for schedule training on text rendering and geneval-style prompts. Num. of sampling steps is 40.

Backbone	**SD3.5-M**	**Flux-Dev**
Text Render	1.25e−5	5e−5
Geneval	1.25e−6	5e−5

We would like to explicitly mention that (i) we *exclude* any regularizer including KL constraint between a reference schedule and the learned schedule during training to be consistent with our settings. This is different from the vanilla implementation of TPDM, where KL constraint is used. (ii) TPDM in its basic form *does not* involve text condition as its input. We experiment with different settings: (i) we faithfully nullify the textual inputs by setting the embeddings to zero vectors to train the model, and (ii) we include the textual inputs as in our settings. We report in Tab. 1 the results for (i). For (ii) the variant, especially in our setting, reduces to the RLOO baseline, which we also report in Tab. 1.

C.3. Text Rendering and Fine-Grained Alignment

We use the same settings as in Sec. C.2, except for tuning the learning rates (see Tab. 8). For the OCR metrics, we adopt the commonly used mask textspotter v3 model [11] to calculate the character level accuracy, precision and recall scores for each image, and then calculate the average across the testing samples as the final results in Tab. 3.

D. Additional qualitative results

We provide further T2I results for 5-step sampling (Figs. 8 and 9), general T2I generation on HPDv2 prompts (Figs. 10 and 11), text rendering (Figs. 12 and 13) and fine-grained object-focused generation (Figs. 14 and 15). We follow the format of Fig. 1 and plot the corresponding schedules for generation at the top left corner.

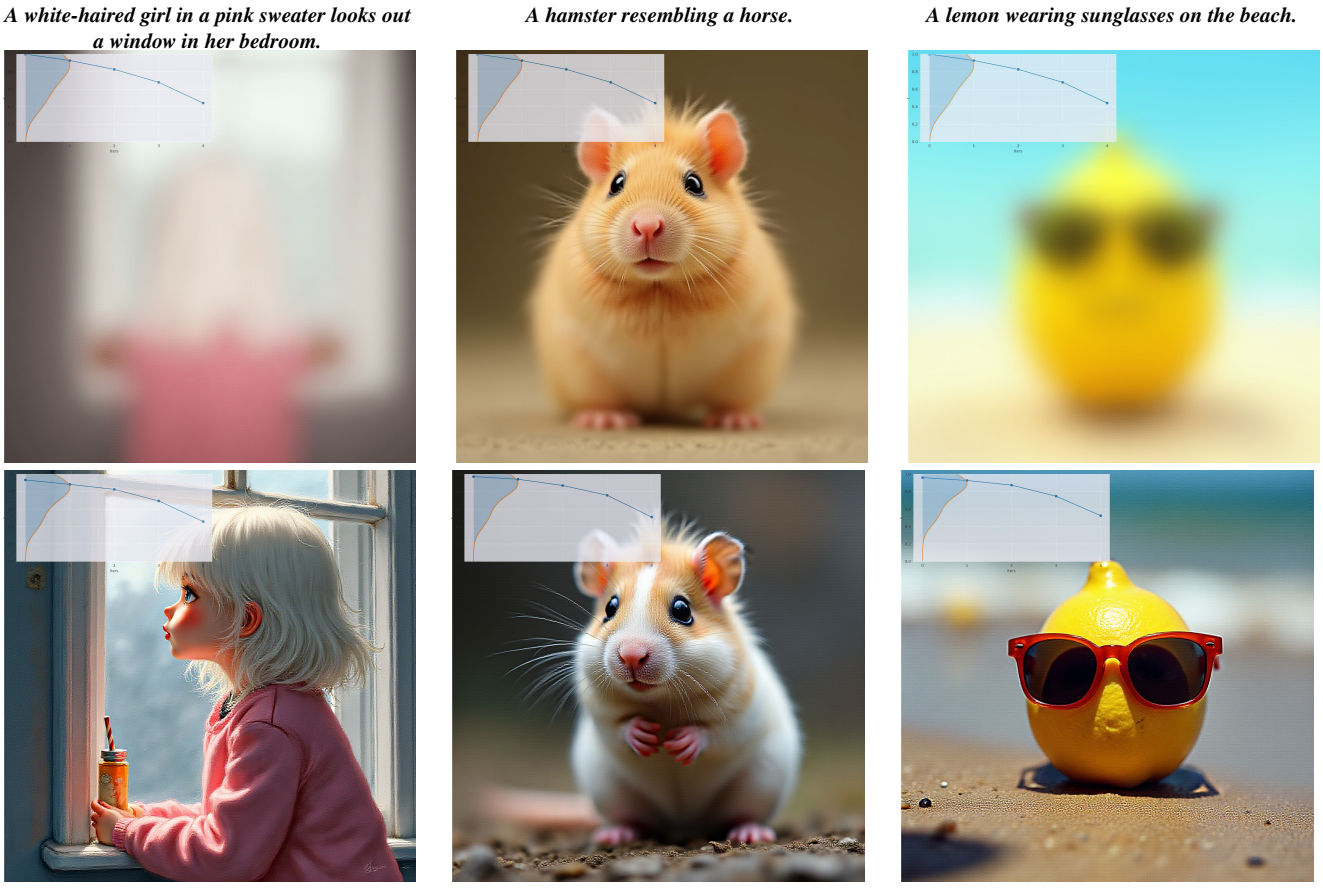

Figure 8. **Rescheduling improves few-step sampling**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev, with 5 steps.

Figure 9. **Rescheduling improves few-step sampling**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev, with 5 steps.

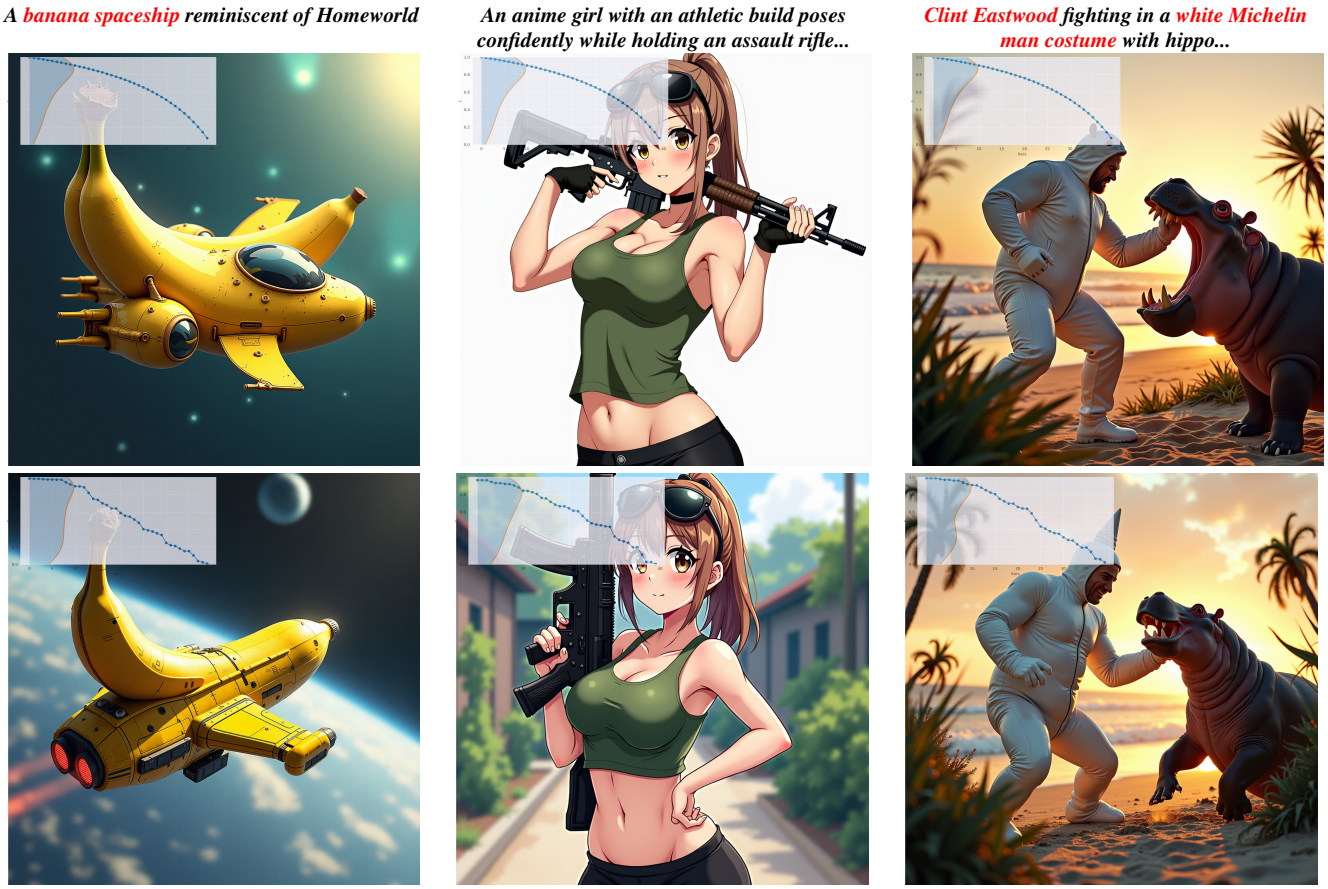

Figure 10. **Rescheduling improves general T2I alignment.** Head-to-head comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev with 40 steps.

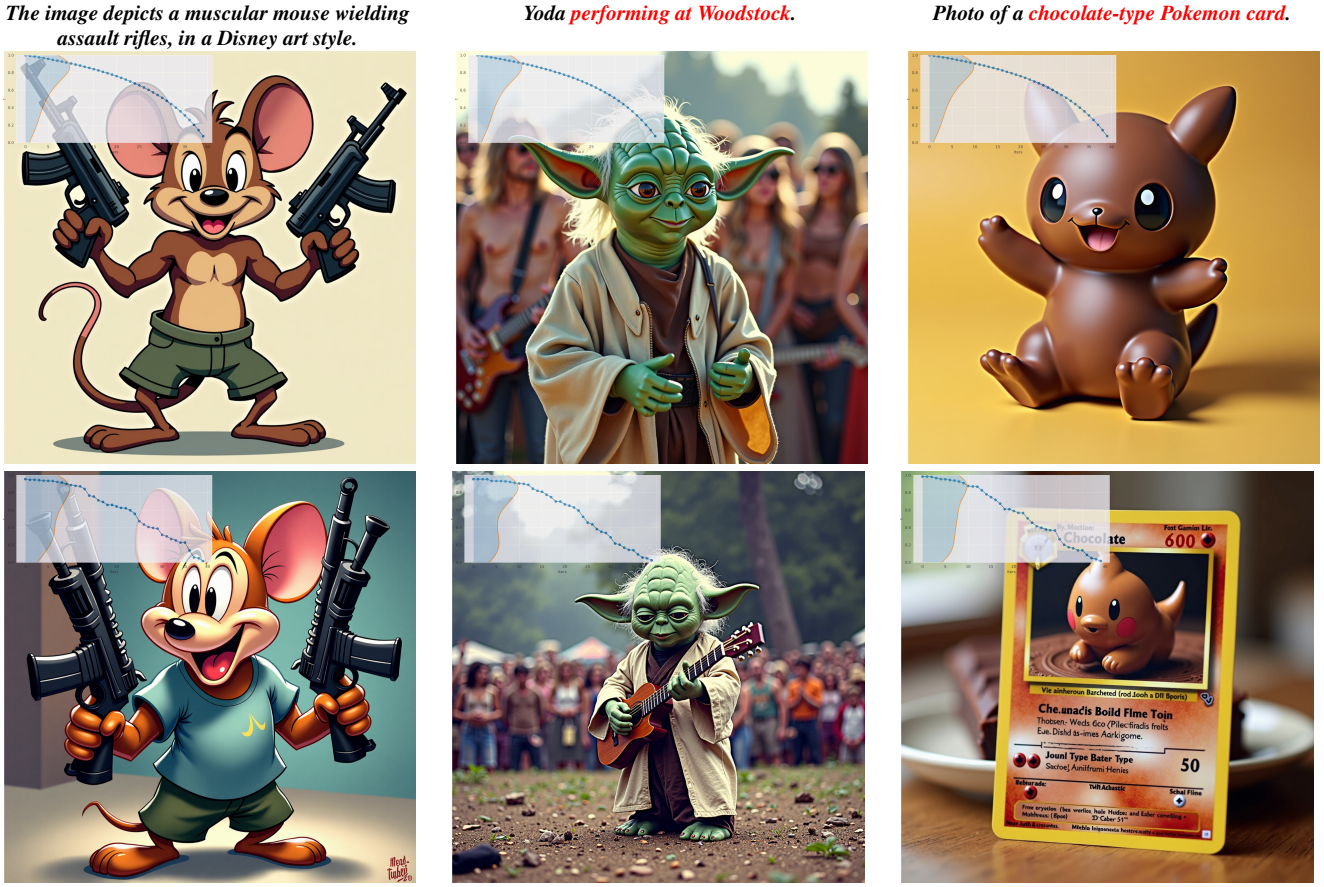

Figure 11. **Rescheduling improves general T2I alignment.** Head-to-head comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev with 40 steps.

Figure 12. **Rescheduling improves text rendering**. We present comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev.

Figure 13. **Rescheduling improves text rendering**. We present comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev.

Figure 14. **Rescheduling improves fine grained alignment**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev.

Figure 15. **Rescheduling improves fine grained alignment**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev.