TRACE: TRAJECTORY-ROUTED MULTI-OBJECTIVE ALIGNMENT FOR FLOW MATCHING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning from image-level rewards has become a standard approach for post-training flow-based text-to-image models. However, most existing methods optimize a single reward, and aligning multiple objectives such as compositional faithfulness, text rendering, and human preference remains challenging. The core difficulty is that multiple objectives conflict when optimized jointly on shared parameters. The difficulty appears along two temporal dimensions: within the denoising trajectory, objectives respond differently across phases; across optimization updates, simultaneous gradients can conflict and sequential updates can overwrite earlier gains. We therefore propose TRACE (**T**rajectory-**R**outed **A**lignment with **C**hunked **E**xperts), which first estimates a per-phase multi-objective utility tensor via bucket-restricted probes and uses it to determine both where each objective is optimized and where each is protected. At the timestep level, TRACE converts this tensor into closed-form softmax routing weights that reweight the policy surrogate loss, while a timestep-conditioned LoRA provides phase-specific updates through a shared adapter. At the training-step level, TRACE trains focused single-objective chunks and protects idle objectives with profile-weighted anchor losses. Experiments on FLUX.1-dev and SD3.5-M with GenEval, OCR, and PickScore show simultaneous improvements over the base models and the highest reported point estimates among the evaluated multi-objective baselines on all three optimized metrics; CLIPScore and Aesthetic Score also remain above the corresponding base-model values. Code is available at `https://anonymous.4open.science/r/TRACE-C427`.

1 INTRODUCTION

Flow-based text-to-image models such as Stable Diffusion 3 Esser et al. (2024) and FLUX Labs (2024) can now generate images of remarkable quality. Reinforcement learning from image-level rewards has emerged as a standard tool for aligning these models with downstream objectives Black et al. (2024); Fan et al. (2023); Liu et al. (2026); Xue et al. (2025). Under a single reward, methods such as Flow-GRPO Liu et al. (2026) and DanceGRPO Xue et al. (2025) substantially improve one objective at a time, whether compositional faithfulness, text rendering, or aesthetic quality. In practice, however, a useful generator must satisfy all of these objectives at once: it should be compositionally correct, render text accurately, and remain visually appealing. This leads to the central problem of multi-objective post-training: how can we align a flow-based generator to several rewards without sacrificing any of them?

Multi-objective alignment is much harder than single-objective alignment. A simple reward mixture on shared parameters often improves the easier objectives while hurting the harder ones, a multi-objective analogue of the alignment tax in RLHF Ouyang et al. (2022). Existing methods mainly fall into two groups. Objective-level methods combine rewards at the loss level, for example through scalarization, calibration, or Pareto-based sample construction Lee et al. (2025). Model-level methods train one expert for each reward and combine them through parameter interpolation Rame et al. (2023), iterative LoRA merging Chen et al. (2026), multi-teacher distillation Fang et al. (2026), or inference-time velocity-field blending Cheng et al. (2026); Li et al. (2024). Although these methods differ in form, they mostly treat multi-objective alignment as either reward aggregation or expert composition, paying little attention to the temporal structure inherent to iterative generation. We

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.

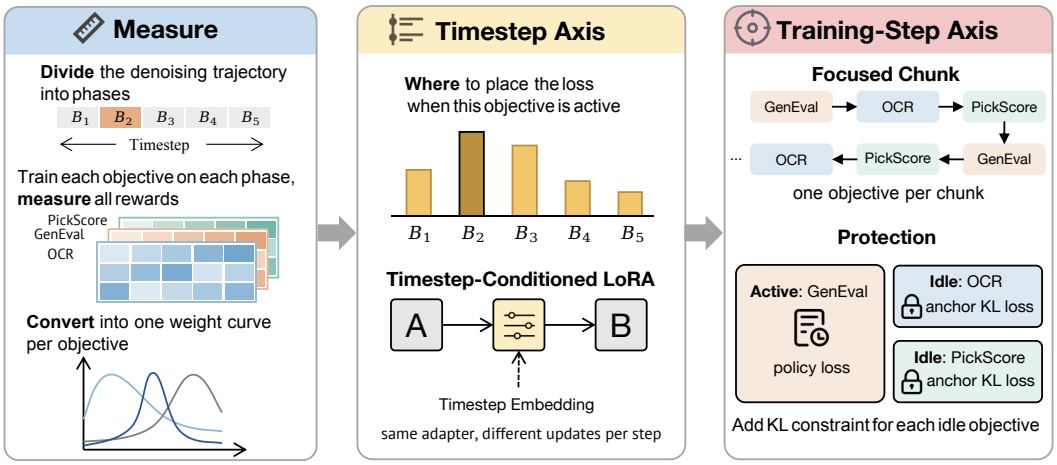


Figure 1: Overview of TRACE. **Measure:** short probes train each objective on each phase, recording how every reward responds, both the targeted gain and cross-objective side effects; these yield a temporal utility tensor, converted into one routing profile per objective. **Timestep Axis:** the active objective's profile concentrates its policy loss on high-utility phases, while a timestep-conditioned LoRA makes one shared adapter phase-specific. **Training-Step Axis:** each chunk optimizes one objective to avoid gradient interference, while idle objectives are protected by anchor losses weighted with their own routing profiles.

argue that this temporal structure is not a peripheral detail but a central lever for multi-objective alignment, and that it should be addressed along two distinct temporal dimensions.

A temporal view of the problem. Reward aggregation and expert composition implicitly treat multi-objective alignment as a static trade-off: rewards are combined into one loss, or separately trained experts are merged after training. In iterative generation, however, this trade-off is temporal. A flow model produces an image through T denoising steps, and an image-level reward is not equally sensitive to updates at all steps; standard RL post-training nevertheless copies the terminal advantage uniformly across the trajectory. This temporal structure can be measured. With short bucket-restricted probes, we estimate, for each objective and phase, both the gain on the targeted reward and the side effects on the other rewards, yielding a temporal utility tensor. This tensor reveals two coupled decisions. Along the timestep axis, each objective has high-utility phases, so updates should be routed there, not spread uniformly. Along the training-step axis, a complementary dilemma arises: joint optimization induces gradient interference, while naive sequential optimization causes forgetting. A method should therefore decide which objective owns each block of updates and where to protect the idle objectives. Underlying both axes is a single measurement, which identifies the phases in which an objective benefits most from active updates; TRACE uses the same phases to weight protection while that objective is idle..

Our approach: TRACE. Motivated by this observation, we propose TRACE (**T**rajectory-**R**outed **A**lignment with **C**hunked **E**xperts), which uses one temporal utility tensor to control both axes. On the timestep axis, TRACE converts the measured tensor into closed-form soft routing weights and instantiates them with a timestep-conditioned LoRA. Routing emphasizes high-utility phases, while time conditioning makes the shared adapter phase-specific. On the training-step axis, TRACE assigns each update chunk to a single objective, removing simultaneous cross-objective gradient interference, and protects inactive objectives with anchor losses weighted by their own routing profiles. The same routing profile therefore serves two roles: it tells TRACE where to spend an objective's budget when the objective is active, and where to protect it when the objective is idle. In this way, TRACE turns phase-specific measurement into a practical multi-objective training procedure.

Our contributions are as follows.

- We identify two temporal axes that govern multi-objective conflicts in flow-based post-training: within a denoising trajectory, objectives respond differently to updates at different phases; across training steps, joint optimization causes gradient interference while sequential optimization causes forgetting.

- We propose TRACE, which estimates a temporal utility tensor via bucket-restricted probes and uses the same tensor to control both temporal axes. On the timestep axis, the tensor is converted into closed-form soft routing weights applied through a timestep-conditioned LoRA, focusing updates on productive denoising phases. On the training-step axis, TRACE trains one objective per chunk while protecting idle objectives with profile-weighted anchors; thus, a single estimate determines where each objective is optimized and protected.
- On FLUX.1-dev and Stable Diffusion 3.5 Medium, TRACE improves GenEval, OCR, and PickScore simultaneously relative to the pretrained model, outperforms both training-time and inference-time multi-objective baselines on all three training rewards, and improves the non-optimized metrics over the base models.

2 Related Work

2.1 Reinforcement Learning for Diffusion and Flow Matching Models

Reinforcement learning (RL) has become an important paradigm for post-training alignment of generative models, where a pretrained generator is fine-tuned with preference data, reward models, or external evaluators to better align its outputs with human preferences and task objectives. Early methods formulate denoising as a sequential decision problem and apply policy-gradient updates. DDPO Black et al. (2024) directly optimizes diffusion models with terminal-image rewards, while DPOK Fan et al. (2023) adds KL regularization to mitigate distribution drift during reward-driven training. Another line adapts preference optimization from language models to diffusion models, exemplified by Diffusion-DPO Wallace et al. (2024). More recently, the success of GRPO Shao et al. (2024) in language-model alignment has motivated its use in visual generation. Since standard flow matching models follow deterministic ODE trajectories that are difficult to optimize directly with policy gradients, Flow-GRPO Liu et al. (2026) and DanceGRPO Xue et al. (2025) introduce stochasticity into sampling, extending GRPO to flow-based text-to-image models.

2.2 Multi-Objective Alignment

Reward-driven post-training has improved diffusion and flow-based generators on individual objectives, yet most methods optimize only one alignment target, making it difficult to simultaneously balance semantic faithfulness, text rendering, and human preference. Recent work has therefore begun to study multi-objective alignment and multi-preference optimization. Existing approaches fall into two families. Objective-level methods combine rewards at the loss level via scalarization, calibration, or Pareto-based sample construction; for example, CaPO Lee et al. (2025) calibrates cross-objective rewards and constructs Pareto-frontier training pairs to reduce bias from reward-scale and preference-distribution mismatch. Model-level methods train or reuse one expert per objective and combine them through parameter interpolation, LoRA merging, distillation, or inference-time composition, as exemplified by Rewarded Soup Rame et al. (2023), MapReduce-LoRA Chen et al. (2026), Flow-OPD Fang et al. (2026), Diffusion-Blend Cheng et al. (2026), and SELMA Li et al. (2024). Despite these differences, most methods treat multi-objective alignment as reward aggregation or expert composition. In contrast, our work focuses on the temporal structure of the generation trajectory, studying how to allocate objectives across denoising and training steps while preserving previously aligned capabilities.

3 Preliminaries

Low-Rank Adaptation (LoRA) Low-Rank Adaptation Hu et al. (2021) is a parameter-efficient fine-tuning method that freezes the pretrained weights and learns a low-rank update. For the weight matrix W of a linear layer, LoRA parameterizes the adapted weight as

$$W' = W + \Delta W, \qquad \Delta W = BA, \tag{1}$$

where $A \in \mathbb{R}^{r \times d_{\text{in}}}$, $B \in \mathbb{R}^{d_{\text{out}} \times r}$, and $r \ll \min(d_{\text{out}}, d_{\text{in}})$ is the LoRA rank. For an input $x \in \mathbb{R}^{d_{\text{in}}}$, the linear transformation becomes

$$W'x = Wx + BAx. \tag{2}$$

In this way, LoRA restricts the parameter update to a low-rank subspace and only optimizes A and B, which reduces the number of trainable parameters and the memory cost.

Group Relative Policy Optimization (GRPO) GRPO Shao et al. (2024) is a value-free policy optimization method based on group-relative rewards. Given a condition c, the current policy $\pi_{\theta_{\text{old}}}$ samples a group of G candidates $\{x_0^i\}_{i=1}^G$, and each candidate receives a reward $R(x_0^i, c)$. GRPO standardizes the rewards within the group to obtain the relative advantage of the i-th sample:

$$\hat{A}^i \;=\; \frac{R(x_0^i, c) - \text{mean}\left(\{R(x_0^j, c)\}_{j=1}^G\right)}{\text{std}\left(\{R(x_0^j, c)\}_{j=1}^G\right)}. \tag{3}$$

This expression requires nonzero within-group reward standard deviation; in implementation a small stabilizer is added to the denominator, so that a group with identical rewards yields zero advantages and contributes no update.

For diffusion and flow-based models, the denoising process can be viewed as a sequential decision process Black et al. (2024); Fan et al. (2023); Liu et al. (2026). Let s_t be the state at step t, let a_t be the corresponding transition action, and let $\pi_\theta(a_t \mid s_t)$ denote the induced single-step policy. GRPO then uses a PPO-style clipped surrogate Schulman et al. (2017):

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\Bigg[&\frac{1}{GT}\sum_{i=1}^{G}\sum_{t=1}^{T} \min\Big(r_t^i(\theta)\,\hat{A}^i,\, \text{clip}\big(r_t^i(\theta),\,1-\epsilon,\,1+\epsilon\big)\,\hat{A}^i\Big) \\ &- \frac{\beta_{\text{KL}}}{GT}\sum_{i=1}^{G}\sum_{t=1}^{T} D_{\text{KL}}\big(\pi_\theta(\cdot \mid s_t^i)\,\|\,\pi_{\text{ref}}(\cdot \mid s_t^i)\big)\Bigg],\end{aligned} \tag{4}$$

where

$$r_t^i(\theta) = \frac{\pi_\theta(a_t^i \mid s_t^i)}{\pi_{\theta_{\text{old}}}(a_t^i \mid s_t^i)}, \tag{5}$$

and ϵ is the clipping threshold. The outer expectation is over prompts and trajectories from the frozen rollout policy, and π_{ref} is a fixed reference policy. Using group-relative rather than absolute rewards removes the need for a separate value network and reduces sensitivity to reward-scale variation.

4 TRACE

We consider a flow matching text-to-image model π_θ that, given a prompt c, produces an image x_0 in T reverse-time denoising steps $t = T, \ldots, 1$. The state s_t holds the current latent and prompt, and the action a_t transitions to the next state s_{t-1} via a Gaussian with mean $\mu_\theta(s_t, t)$ and variance σ_t^2. We partition the normalized denoising progress $\tau_t = 1 - t/T \in [0, 1)$ into K equal buckets $\mathcal{B}_1, \ldots, \mathcal{B}_K$ ordered from high noise to final refinement, and write $k(t)$ for the bucket of step t. Our goal is to align π_θ to M image-level rewards $\{R_m\}_{m=1}^M$ at once.

Under a fixed policy-update budget, multi-objective post-training poses two coupled allocation decisions: which objective to optimize at each training step, and how to spread its reward-driven loss over the denoising trajectory. The uniform timestep weighting of Eq. (4) ignores how differently rewards respond to updates at different phases. TRACE first measures these responses with a one-off profiling pass that concentrates each probe's loss in one bucket and records every reward's change, yielding a temporal utility tensor of self-gain and cross-objective effects (§4.1). On the timestep axis, TRACE turns this tensor into objective-specific routing profiles and lets Tc-LoRA condition the adapter on the denoising step (§4.2); on the training-step axis, one objective owns each chunk to avoid simultaneous gradient conflict, while anchor losses weighted by each idle objective's own profile prevent forgetting (§4.3). The same measured profile thus marks where to optimize an objective when active and where to protect it when idle. Algorithm 1 summarizes the training loop.

4.1 ONE MEASUREMENT: THE TEMPORAL UTILITY TENSOR

We ask: how do all rewards respond when a specific denoising-phase objective is optimized? A one-off set of short probes measures these responses and supplies the profiles used throughout training.

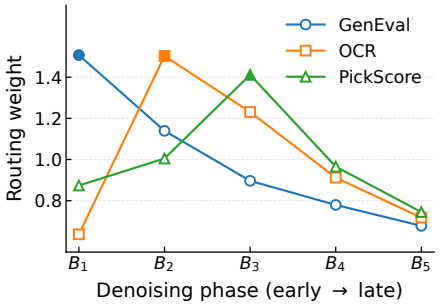
Measure phase-dependent responses. For each objective-bucket pair (m, k), we start a Flow-GRPO probe from the same base model θ_0, generate complete trajectories, and restrict the reward-driven policy loss to $\mathcal{B}_k$ by reweighting the advantage:

$$\hat{A}_{m,t}^i = K\,\mathbb{1}[\tau_t \in \mathcal{B}_k]\hat{A}_m^i. \tag{6}$$

Table 1: Temporal utility tensor from bucket-restricted probes. Rows give self-gain and cross-effects; columns run from early to late denoising. Values are normalized reward changes.

	$\mathcal{B}_1$	$\mathcal{B}_2$	$\mathcal{B}_3$	$\mathcal{B}_4$	$\mathcal{B}_5$
GenEval self-gain / cross-effects					
S_{Gen}	**0.66**	0.46	0.29	0.17	0.09
$C_{\text{Gen}\to\text{OCR}}$	-0.10	-0.06	-0.03	-0.01	0.00
$C_{\text{Gen}\to\text{Pick}}$	-0.06	-0.04	-0.02	0.01	0.01
OCR self-gain / cross-effects					
S_{OCR}	0.20	**0.55**	0.40	0.24	0.12
$C_{\text{OCR}\to\text{Gen}}$	-0.09	-0.05	-0.04	-0.02	0.00
$C_{\text{OCR}\to\text{Pick}}$	-0.05	-0.01	0.03	0.02	0.00
PickScore self-gain / cross-effects					
S_{Pick}	0.34	**0.35**	0.28	0.16	0.08
$C_{\text{Pick}\to\text{Gen}}$	-0.11	-0.06	0.09	0.05	0.02
$C_{\text{Pick}\to\text{OCR}}$	-0.04	-0.03	0.06	0.03	0.01

Figure 2: Routing profiles derived from the temporal utility tensor (Eq. (9)). Weights have mean one; the dashed line is uniform credit. Filled markers highlight each objective's preferred phase.



The factor K preserves the mean loss weight across equally sized buckets. All probes use the same update budget, and all rewards are evaluated on held-out validation prompts drawn from the training distribution. This measures the effect of concentrating the loss in a phase.

Separate self-gain from cross-effects. Let $\bar{R}_j(\theta)$ be the mean evaluation reward and let $g_j > 0$ be the gain of a fully trained single-objective expert for reward j over the base model, a per-reward normalization constant. For each probe checkpoint $\theta^{\text{probe}}_{m,k}$ obtained by profiling objective m in bucket k, define

$$\mathcal{T}_{m,j,k} = \frac{\bar{R}_j(\theta^{\text{probe}}_{m,k}) - \bar{R}_j(\theta_0)}{g_j}, \qquad S_m(k) = \mathcal{T}_{m,m,k}, \quad C_{m\to j}(k) = \mathcal{T}_{m,j,k} \quad (j \neq m). \tag{7}$$

The tensor records which objective is optimized, which reward is evaluated, and where the loss is applied. Because each entry is a fraction of the evaluated reward's expert gain, entries across different rewards are comparable: S_m measures the self-gain of objective m, while a positive or negative $C_{m\to j}$ indicates benefit or damage to another objective. We combine them into a net utility:

$$U_m(k) = S_m(k) + \sum_{j\neq m} C_{m\to j}(k). \tag{8}$$

The utility tensor provides an objective-specific net utility U_m, from which we derive one routing profile w_m per objective; the profile guides policy optimization when the objective is active (§4.2) and weights its anchor penalty when it is inactive (§4.3).

4.2 Timestep Axis: Spending the Budget Where It Pays

From utility to routing profile. An entropy-regularized allocation over buckets yields nonnegative, mean-one softmax weights used directly in training:

$$w_m(k) = \frac{K \exp(U_m(k)/T_r)}{\sum_{k'} \exp(U_m(k')/T_r)}. \tag{9}$$

We set $T_r = 0.5$ and keep the profiles fixed during training. Appendix A.2 derives the softmax form, for any fixed temperature, as the optimizer of a surrogate allocation problem.

Routed GRPO objective. Substituting w_m from Eq. (9) into the clipped surrogate of Flow-GRPO yields the routed policy loss for objective m:

$$\mathcal{L}^{\text{pol}}_m(\theta) = -\,\mathbb{E}\left[\frac{1}{GT}\sum_{i=1}^{G}\sum_{t=1}^{T} w_m(k(t))\,\min\Big(r^i_t(\theta)\,\hat{A}^i_m,\ \text{clip}\big(r^i_t(\theta),\,1-\epsilon,\,1+\epsilon\big)\,\hat{A}^i_m\Big)\right]. \tag{10}$$

Here, $r^i_t(\theta)$ and ϵ follow the GRPO definitions. Routing changes only the timestep weights of the policy surrogate; uniform weights recover its unweighted form.

Timestep-conditioned LoRA. The routing profiles emphasize different denoising phases. To let the adapter respond differently at different timesteps, we condition the LoRA update on the backbone's time embedding $e(t) \in \mathbb{R}^{d_t}$. Two linear heads produce channel-wise scale and shift vectors of the low-rank representation $Ax \in \mathbb{R}^r$ (Perez et al., 2018):

$$\gamma(t) = W_\gamma e(t) + b_\gamma, \qquad \beta(t) = W_\beta e(t) + b_\beta, \qquad W_\gamma, W_\beta \in \mathbb{R}^{r \times d_t}. \tag{11}$$

The modulated update replaces BAx in Eq. (2) with

$$\Delta y_t = B\big(\gamma(t) \odot (Ax) + \beta(t)\big). \tag{12}$$

The scale $\gamma(t)$ and shift $\beta(t)$ modulate the low-rank representation before projection by B. Equivalently, the adapter applies an effective weight update $\Delta W(t) = B\,\mathrm{diag}(\gamma(t))A$ and bias $\Delta b(t) = B\beta(t)$. Thus, a shared set of adapter parameters produces different transformations at different denoising steps, without requiring a separate adapter for each step. On FLUX.1-dev, Tc-LoRA increases the standard adapter parameter count by 6.27%.. Implementation details are given in Appendix D.

4.3 Training-Step Axis: One Objective at a Time

Focused chunks. Routing determines the temporal weighting of each objective, while chunking determines which objective supplies the reward-driven update. Concretely, we divide the training run into contiguous chunks of on-policy policy updates; within chunk q, a single objective $m^{(q)}$ remains active for every update in the chunk, and the following chunk may switch to a different objective. The active objective supplies the prompts, reward, and routing profile for the policy loss; inactive objectives contribute only anchor losses. This avoids combining multiple reward-driven policy gradients within an update. Unlike model-level methods that keep one trainable expert per objective, chunking maintains a single shared adapter: the expert of chunk q is a behavioral regime induced by the active objective's reward and routing profile, not a separate parameter set.

Anchors hold the inactive objectives. Optimizing one reward can degrade previously learned capabilities (Table 4). For each objective j, we therefore maintain an anchor snapshot θ_j^-, initialized at θ_0 and tracking the checkpoint that attained its best validation reward while active. When $m^{(q)}$ is active, each inactive objective j is protected by an anchor loss on replay trajectories $\xi = (s_1, \ldots, s_T) \sim \mathcal{D}_j$, weighted by its own routing profile w_j, which emphasizes the phases that objective prefers:

$$\mathcal{L}_j^{\mathrm{anc}}(\theta) = \mathbb{E}_{\xi \sim \mathcal{D}_j}\Big[\textstyle\sum_t w_j(k(t))\, D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid s_t) \,\|\, \pi_{\theta_j^-}(\cdot \mid s_t)\big)\Big], \tag{13}$$

where $\pi_\theta(\cdot \mid s_t)$ denotes the Gaussian transition with mean $\mu_\theta(s_t, t)$ and covariance $\sigma_t^2 I$, for which the conditional KL reduces to the variance-normalized mean-matching term $\|\mu_\theta(s_t, t) - \mu_{\theta_j^-}(s_t, t)\|^2/(2\sigma_t^2)$. The total loss for chunk q is

$$\mathcal{L}^{(q)} = \mathcal{L}_{m^{(q)}}^{\mathrm{pol}} + \sum_{j \neq m^{(q)}} \kappa_j \mathcal{L}_j^{\mathrm{anc}}. \tag{14}$$

The anchor for each inactive objective uses its own routing profile w_j, emphasizing the phases preferred by that objective. These objective-specific anchors replace the global KL penalty in Eq. (4).

Which objective goes next. We prioritize objectives that have regressed or received fewer chunks. Let $R_m^\star$ be the best observed validation reward, and $\bar{R}_m^{(q-1)}$ the latest value. We sample the active objective for chunk q with probability:

$$p^{(q)}(m) \propto \frac{[R_m^\star - \bar{R}_m^{(q-1)}]_+}{R_m^\star} + \frac{1}{n_m + 1}, \tag{15}$$

where n_m is the number of completed chunks assigned to objective m before chunk q, incremented by one after each chunk assigned to m. The first term grows with the shortfall of m's current validation reward below its historical best, normalized by that best value so that objectives with different reward scales contribute comparably to the priority score. The second term decreases with n_m, raising the priority of objectives that have received fewer chunks and giving every objective nonzero sampling probability without guaranteeing a maximum waiting time.

Table 2: Main results on FLUX.1-dev. GenEval, OCR, and PickScore are the three training objectives, evaluated on their corresponding Flow-GRPO test prompts. For GenEval, we report all six sub-scores and the official overall score. CLIPScore and Aesthetic Score serve as non-optimized evaluation metrics, computed on the same test prompts as PickScore. **Bold** and underlined scores indicate the best and second-best values across all methods; higher is better. Results on SD3.5-M are shown in Table 5.

Method	GenEval							OCR	Pick.	CLIP	Aes.
	Single	Two	Count.	Colors	Position	C.-Attr.	Overall				
FLUX.1-dev	0.988	0.748	0.713	0.745	0.200	0.470	0.644	0.591	21.993	34.428	6.216
Single-objective alignment											
Flow-GRPO (GenEval)	1.000	0.980	0.888	0.851	0.860	0.760	<u>0.890</u>	0.678	22.094	34.928	6.210
Flow-GRPO (OCR)	0.975	0.859	0.738	0.787	0.220	0.540	0.686	**0.972**	22.014	34.916	6.186
Flow-GRPO (PickScore)	1.000	0.919	0.500	0.702	0.250	0.560	0.655	0.710	**23.591**	34.622	6.450
Multi-objective alignment											
Joint training	1.000	0.910	0.825	0.780	0.450	0.610	0.763	0.889	22.638	35.201	6.298
Diffusion-Blend	1.000	0.899	0.738	0.819	0.380	0.680	0.753	0.732	23.083	35.659	6.375
SELMA	1.000	0.950	0.838	0.800	0.530	0.680	0.799	0.879	22.947	<u>36.056</u>	6.344
MapReduce-LoRA	1.000	0.980	0.875	0.915	0.820	0.670	0.877	0.951	23.015	**36.391**	6.313
Flow-OPD	1.000	0.970	0.913	0.862	0.810	0.700	0.876	0.951	23.454	34.231	<u>6.479</u>
TRACE (Ours)	1.000	0.939	0.925	0.883	0.930	0.700	**0.896**	<u>0.962</u>	<u>23.576</u>	35.886	**6.566**

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We conduct experiments on two flow-based text-to-image models: FLUX.1-dev Labs (2024) and Stable Diffusion 3.5 Medium (SD3.5-M), both at 512×512 resolution. Following prior work on flow-based alignment, we study multi-objective post-training under three complementary reward dimensions: GenEval for compositional faithfulness, OCR for rendered-text accuracy, and PickScore as a learned proxy for human preference Ghosh et al. (2023); Cui et al. (2025); Kirstain et al. (2023). Unless otherwise noted, we use the same training and test prompt splits as Flow-GRPO Liu et al. (2026), with the per-split training and test prompt counts listed in Appendix B.2. All three rewards are targeted during training and are also reported as the primary evaluation metrics.

We compare TRACE against four categories of baselines. *(1) Single-objective experts.* We train one LoRA expert per reward using Flow-GRPO, yielding three reward-specialized models for GenEval, OCR, and PickScore. *(2) Joint training.* A single model is trained on all three rewards simultaneously under the same Flow-GRPO and LoRA setup, the simplest multi-objective baseline. *(3) Expert composition.* Diffusion-Blend Cheng et al. (2026) blends reward-specific reverse processes at inference time; SELMA Li et al. (2024) merges independently trained LoRA experts. *(4) Training-time multi-objective methods.* MapReduce-LoRA Chen et al. (2026) and Flow-OPD Fang et al. (2026) integrate multiple reward-specific experts during training. Baselines that we retrain share a common LoRA rank, prompt splits, and reward functions; externally released checkpoints are evaluated as provided, under their reported settings. Appendix B details the datasets, evaluators, and baselines.

5.2 MAIN RESULTS

Table 2 and Table 5 report the quantitative comparison on FLUX.1-dev and SD3.5-M, respectively. The three single-objective Flow-GRPO experts exhibit a clear specialization pattern: each performs best or near-best on its own target objective but does not transfer well to the other two. This pattern is consistent across both models, confirming that reward-specialized experts alone are insufficient for balanced multi-objective alignment.

Among the multi-objective alignment methods, TRACE achieves the strongest overall performance across all three training rewards on both models. On FLUX.1-dev, it obtains the highest GenEval overall score (0.896), the best OCR among multi-objective methods (0.962), and the best PickScore among multi-objective methods (23.576). On SD3.5-M, TRACE similarly leads all multi-objective



Figure 3: Qualitative comparison across representative prompts. The examples illustrate compositional, text-rendering, and visual-style outputs; they do not constitute a human preference evaluation. For the Flow-GRPO column, we use the single-objective expert corresponding to the primary reward type of each prompt.

baselines on all three rewards. On FLUX.1-dev, TRACE slightly exceeds the reported point estimate of the GenEval-specialized Flow-GRPO expert on the overall GenEval score (0.896 versus 0.890). On the non-optimized quality metrics, TRACE remains competitive on CLIPScore Hessel et al. (2021) and Aesthetic Score, indicating that its gains on the aligned objectives do not come at the expense of general image quality.

5.3 ABLATION STUDY

We ablate each component of TRACE while keeping the others unchanged. All results are on FLUX.1-dev.

Component contributions. Table 3 (top) removes one component at a time while keeping the others fixed; every removal lowers all three objectives, so each component is necessary. Anchor protection is the key to retention: without it the model forgets previously aligned objectives, matching the forgetting measured in Table 4. Policy time routing is the key to spending efficiency: without it updates spread uniformly and the measured phase-specific utility is discarded. Adaptive scheduling is replaced in this ablation by a fixed round-robin cycle over the three objectives; the resulting smaller but consistent decreases show that demand-aware prioritization, rather than mere alternation, provides a meaningful but secondary benefit.

Routing form spectrum. Table 3 (bottom) varies only the routing weight assignment while keeping all other components fixed; uniform, random, reversed, and hard replacements all score lower, and the profiled soft weights remain highest. Random weights (entropy-matched to the profiled softmax) and reversed weights (concentrating updates on the least productive phases) are non-uniform yet ignore the measured utility. Both underperform the profiled routing weights, disfavoring non-

Table 3: Component and routing ablations on FLUX.1-dev. The top group removes one component at a time while keeping the others fixed. The bottom group varies the routing weight assignment while keeping all other components fixed. Higher is better for all metrics. Best results are in **bold**.

Variant	GenEval	OCR	PickScore
Component Contributions			
TRACE (full)	**0.896**	**0.962**	**23.576**
w/o policy time routing	0.845	0.922	23.149
w/o anchor protection	0.813	0.891	23.173
w/o adaptive scheduling	0.874	0.946	23.492
Routing Form Spectrum			
Uniform weights ($T_r \to \infty$)	0.845	0.922	23.149
Random weights (same entropy)	0.852	0.928	23.098
Reversed weights	0.781	0.890	22.741
Hard interval ($T_r \to 0$)	0.871	0.944	23.388

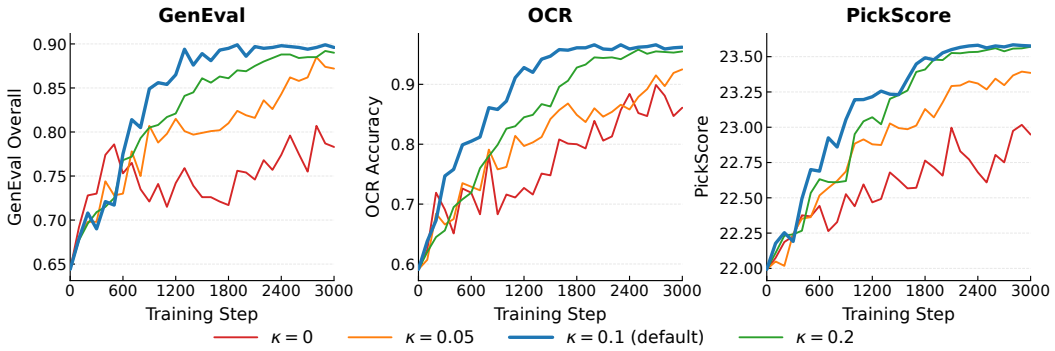


Figure 4: Training dynamics under different anchor weights κ on FLUX.1-dev. The curves span 3000 training steps for $\kappa \in \{0, 0.05, 0.1, 0.2\}$. Without protection ($\kappa = 0$) the rewards show sawtooth oscillations, the default $\kappa = 0.1$ rises steadily, and $\kappa = 0.2$ converges more slowly.

uniformity alone as an explanation for the observed gains. Hard routing ($T_r \to 0^+$) assigns the whole budget to the best bucket and underperforms the soft profile.

Anchor protection dynamics. The anchor weight κ_j in Eq. (14) controls the strength of the replay-state constraint. Figure 4 compares $\kappa \in \{0, 0.05, 0.1, 0.2\}$ over 3000 training steps. Without protection ($\kappa = 0$), the active objective rises within each chunk but drops when idle, producing sawtooth oscillations; the intermediate $\kappa = 0.05$ damps but does not remove them; at the default $\kappa = 0.1$ only slight jitter remains and the trend rises steadily, while $\kappa = 0.2$ converges more slowly because stronger pulling restricts per-chunk learning. The curves thus suggest a trade-off between retention and adaptation, motivating the default $\kappa = 0.1$.

6 CONCLUSION

We study multi-objective post-training for flow-based text-to-image models and show that its main difficulties arise along two temporal dimensions: within the denoising trajectory, where self-gain and collateral damage are non-proportional across phases, and across optimization steps, where joint training induces gradient conflict and sequential training causes forgetting. Addressing both dimensions, TRACE uses phase-restricted probes to construct objective-specific routing profiles, a shared timestep-conditioned adapter to represent time-dependent updates, and focused chunks with replay-based anchors to limit drift on inactive objectives. On FLUX.1-dev and Stable Diffusion 3.5 Medium, TRACE improves all three optimized rewards over the corresponding base models and attains the highest reported point estimates among the evaluated multi-objective baselines.

AI USE STATEMENT

We used large language models to polish the manuscript and assist in retrieving related literature. The authors revised all AI-assisted text and manually verified every suggested reference against the original source. Large language models were not used for idea generation, experimental design, code implementation, or data analysis. The authors take full responsibility for the final content of the paper.

REFERENCES

Akhil Agnihotri, Rahul Jain, Deepak Ramachandran, and Zheng Wen. Multi-objective preference optimization: Improving human alignment of generative models. *arXiv preprint arXiv:2505.10892*, 2025.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 4965–4987, 2024.

Chieh-Yun Chen, Zhonghao Wang, Qi Chen, Zhifan Ye, Min Shi, Yue Zhao, Yinan Zhao, Hui Qu, Wei-An Lin, Yiru Shen, et al. Mapreduce lora: Advancing the pareto front in multi-preference optimization for generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 34375–34384, 2026.

Min Cheng, Fatemeh Doudi, Dileep Kalathil, Mohammad Ghavamzadeh, and Panganamala Kumar. Diffusion blend: Inference-time multi-preference alignment for diffusion models. In *International Conference on Learning Representations*, volume 2026, pp. 51151–51181, 2026.

Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, et al. Paddleocr 3.0 technical report. *arXiv preprint arXiv:2507.05595*, 2025.

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.

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. *Advances in neural information processing systems*, 36:79858–79885, 2023.

Zhen Fang, Wenxuan Huang, Yu Zeng, Yiming Zhao, Shuang Chen, Kaituo Feng, Yunlong Lin, Lin Chen, Zehui Chen, Shaosheng Cao, et al. Flow-opd: On-policy distillation for flow matching models. *arXiv preprint arXiv:2605.08063*, 2026.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36: 52132–52152, 2023.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7514–7528, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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. *Advances in neural information processing systems*, 36:36652–36663, 2023.

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

Kyungmin Lee, Xiahong Li, Qifei Wang, Junfeng He, Junjie Ke, Ming-Hsuan Yang, Irfan Essa, Jinwoo Shin, Feng Yang, and Yinxiao Li. Calibrated multi-preference optimization for aligning diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18465–18475. IEEE, 2025.

Jialu Li, Jaemin Cho, Yi-Lin Sung, Jaehong Yoon, and Mohit Bansal. Selma: Learning and merging skill-specific text-to-image experts with auto-generated data. *Advances in Neural Information Processing Systems*, 37:38530–38558, 2024.

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. *Advances in neural information processing systems*, 38:40783–40818, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Alexandre Rame, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *Advances in Neural Information Processing Systems*, 36:71095–71134, 2023.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=M3Y74vmsMcY.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

A Temporal Utility Tensor: Profiling and Analysis

This appendix specifies the measurement protocol for §4.1, derives the routing weights used in §4.2, and explains how the measured tensor determines the temporal profiles.

A.1 Profiling Protocol and Interpretation

For profiling, we partition $T = 5$ denoising steps into $K = 5$ equally sized buckets, ordered from high noise to final refinement. Each objective–bucket pair (m, k) is evaluated by an independent standard LoRA probe initialized from the same base checkpoint θ_0. Every probe runs for $N_{\text{probe}} =$

50 updates with batch size 32, generating complete trajectories while restricting the reward-driven loss to $\mathcal{B}_k$ through Eq. (6). The factor K preserves the mean loss coefficient across the trajectory, but does not equalize gradient norms. With three objectives, this yields $MK = 15$ probes per seed. Additional implementation settings are given in Appendix D.

For each probe checkpoint, we evaluate all rewards to obtain both self-gain and cross-objective effects. Let $\bar{R}_j(\theta)$ denote the mean evaluation reward for objective j. Its reference gain is

$$g_j = \bar{R}_j(\theta_j^{\mathrm{SO}}) - \bar{R}_j(\theta_0) > 0,$$

where θ_j^{SO} is the trained single-objective expert. Equation (7) normalizes each probe's change in reward j relative to the base model by g_j. The denominator therefore depends on the reward being evaluated, not on the objective optimized by the probe, so self-gain and incoming cross-effects for that reward share the same scale.

Each tensor entry measures the response to phase-restricted loss weighting over the probe's update budget. It does not imply that policy changes are confined to the selected bucket: shared parameters can affect other buckets, and altered states can propagate to later denoising steps. We use these responses to guide allocation and keep the tensor fixed during Tc-LoRA training.

A.2 Derivation of Temporal Routing Weights

For an active objective m, we allocate nonnegative weights with mean one by solving

$$\max_{w \geq 0,\, \sum_k w(k) = K} \frac{1}{K} \sum_{k=1}^{K} \big[w(k) U_m(k) - T_r w(k) \log w(k) \big],$$

where $T_r > 0$ and $0 \log 0 = 0$. The objective is strictly concave and its optimizer is positive. Introducing a multiplier ν for the mean-one constraint gives

$$U_m(k) - T_r \big(\log w(k) + 1 \big) - \nu = 0.$$

Thus $w(k) \propto \exp(U_m(k)/T_r)$. Normalizing the sum to K yields the unique optimizer w_m in Eq. (9).

For equally sized buckets, $T^{-1} \sum_t w_m(k(t)) = 1$: routing preserves the average loss coefficient, not the gradient norm. As $T_r \to \infty$, the weights become uniform; as $T_r \to 0^+$, they concentrate on the maximum-utility buckets, with equal weights for ties.

A.3 Pseudocode of Trace

Algorithm 1 Trace training loop

Require: Base model θ_0, update budget, and chunk length H
1: Measure $\mathcal{T}$ with independent probes from θ_0; derive fixed routing profiles w_j (Eqs. (7)–(9))
2: Initialize policy, anchors, replay buffers, and scheduling records from θ_0
3: **for** each chunk q until the update budget is exhausted **do**
4: Select the active objective $m^{(q)}$ using Eq. (15)
5: **for** up to H updates within the remaining budget **do**
6: Collect on-policy trajectories for $m^{(q)}$; compute its routed loss (Eq. (10))
7: Sample replay for inactive objectives; compute their anchor losses (Eq. (13))
8: Update the shared policy using the combined loss (Eq. (14))
9: **end for**
10: Evaluate all objectives and update scheduling records
11: Replace the active anchor if its validation reward exceeds its stored best
12: Refresh the replay buffers
13: **end for**
Ensure: Trained shared policy θ

B EXPERIMENTAL SETTINGS

This appendix supplements Section 5.1 with full details on base models, datasets, reward models, and baselines.

B.1 BASE MODELS

We evaluate two flow-based text-to-image models at 512×512 resolution. Both parameterize a velocity field and are sampled with an ODE solver. Because deterministic ODE sampling does not provide stochastic actions for policy-gradient optimization, Flow-GRPO introduces an SDE-based sampling process during training. This deterministic process requires stochastic perturbations for policy-gradient optimization, making them amenable to Flow-GRPO.

- **FLUX.1-dev** Labs (2024): A 12B-parameter rectified-flow transformer (MM-DiT) from Black Forest Labs. Inference uses $T = 28$ denoising steps.
- **Stable Diffusion 3.5 Medium** Esser et al. (2024): A 2.5B-parameter MMDiT-X from Stability AI. Inference uses $T = 40$ steps.

B.2 DATASETS AND PROMPT SPLITS

We adopt the exact training and test prompt splits from Flow-GRPO Liu et al. (2026). Each reward dimension uses its own prompt set, and all methods within each backbone share identical splits; per-split counts are listed below. Profiling and validation use a separate validation split held out from both the training and test prompts..

- **GenEval prompts** Ghosh et al. (2023): $50{,}000$ training prompts and 553 test prompts; the test prompts span the six subtasks (single object, two objects, counting, colors, position, and complex attributes) and carry ground-truth annotations for automated evaluation.
- **OCR prompts**: $19{,}653$ training and $1{,}018$ test text-rendering prompts following the Flow-GRPO split.
- **PickScore prompts** Kirstain et al. (2023): $25{,}432$ training and $2{,}048$ test prompts drawn from Pick-a-Pic v2.

B.3 REWARD MODELS AND EVALUATION METRICS

Training-aligned rewards. Three rewards serve as both training signals and primary evaluation metrics.

- **GenEval** Ghosh et al. (2023): An object-focused framework using pre-trained detectors. It verifies object presence, count, color binding, and spatial relationships. It reports six sub-scores and one overall score. We use the overall score as R_{Gen}.
- **PaddleOCR** Cui et al. (2025): The PaddleOCR 3.0 toolkit (PP-OCRv5, 70M parameters) extracts rendered text. OCR accuracy serves as R_{OCR}.
- **PickScore** Kirstain et al. (2023): A CLIP-based preference model fine-tuned on Pick-a-Pic v2. It computes a scalar preference score via a linear head over CLIP embeddings. This score serves as R_{Pick}.

Non-optimized metrics. Two general image-quality metrics are computed on PickScore test prompts. They are never optimized during training.

- **CLIPScore** Hessel et al. (2021): Computed as $100 \times \max(\cos(\mathbf{z}_I, \mathbf{z}_T), 0)$ using CLIP ViT-H/14 Schuhmann et al. (2022). It measures global text-image alignment.
- **Aesthetic Score**: An MLP predictor trained on LAION-Aesthetics ratings. It operates on CLIP image embeddings to quantify visual quality.

B.4 BASELINES

All baselines share the same LoRA rank, prompt splits, and reward functions. Composition-based and distillation-based baselines reuse the experts trained in the first baseline below.

Single-objective Flow-GRPO Liu et al. (2026); Shao et al. (2024): We train one independent LoRA expert per reward using the official Flow-GRPO code. Flow-GRPO extends GRPO to flow-matching models by injecting stochasticity into the ODE trajectory. This enables PPO-style clipped policy updates. Training uses denoising reduction ($T = 5$ steps) to accelerate rollouts. Inference retains the full schedule. Each expert is trained for 1000 updates on $8\times$A100 GPUs, at which point all three experts have converged (reward plateaus confirmed on held-out prompts).

Joint training: A single LoRA model optimizes all three rewards simultaneously. At each step, we compute three reward-specific advantages from the same rollout group. Their clipped surrogate losses are summed with equal weights.

Diffusion-Blend Cheng et al. (2026): An inference-time composition method. It maintains one denoising trajectory per objective. These trajectories are driven by the single-objective Flow-GRPO experts defined above. At each step, it blends the predicted velocities using equal weights. Model weights remain unmodified.

SELMA Li et al. (2024): A parameter-space merging method. It combines multiple skill-specific experts into one model. We directly use the single-objective Flow-GRPO experts as these skills. Their low-rank updates are averaged with equal weights into a unified LoRA adapter.

MapReduce-LoRA Chen et al. (2026): A training-time iterative method. It alternates between two phases. *Map*: Train reward-specific LoRA experts in parallel using Flow-GRPO. *Reduce*: Merge experts back into a shared base model using equal weights. The merged model initializes the next round. We use the official pretrained checkpoints for both FLUX.1-dev and SD3.5-M, which follow the default three-round configuration.

Flow-OPD Fang et al. (2026): A training-time distillation method. It replaces sparse scalar rewards with dense vector-field guidance. Each single-objective Flow-GRPO expert acts as a teacher. The student's velocity prediction is supervised by a combination of teacher outputs. We apply equal weights to all teachers. For SD3.5-M, we use the official pretrained checkpoint. For FLUX.1-dev, we train the student using the official code for 1500 updates on $8\times$A100 GPUs, at which point the student has converged.

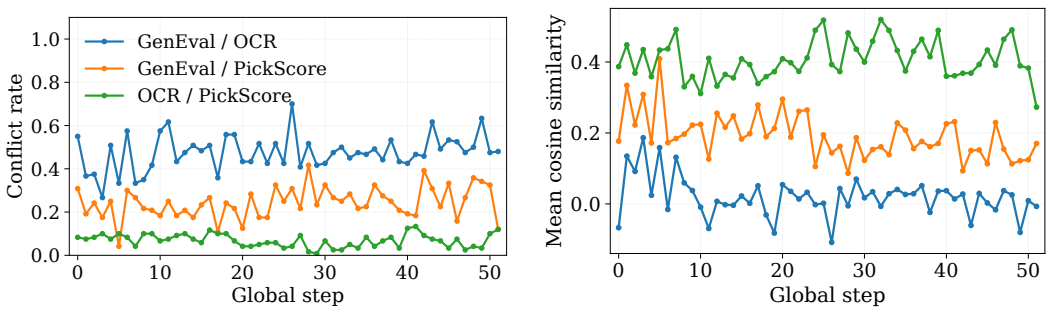
C ADDITIONAL EXPERIMENTS

C.1 ANALYSIS OF MULTI-OBJECTIVE ALIGNMENT

Gradient interference under joint training. A natural baseline for multi-objective alignment is to mix data and rewards from all objectives and jointly optimize a shared parameter set Agnihotri et al. (2025); Lee et al. (2025); Rame et al. (2023). This approach implicitly assumes that different reward gradients are largely compatible in the shared parameter space. To test this assumption, we compute the per-objective policy gradient on the LoRA parameters at every 50 training updates during a Joint training run on FLUX.1-dev. We then measure the pairwise cosine similarity between the flattened gradient vectors.

Figure 5(a) reports the conflict rate, defined as the fraction of steps where the cosine similarity is negative, for each objective pair. It shows that GenEval/OCR has the highest conflict rate, GenEval/PickScore an intermediate rate, and OCR/PickScore the lowest. Figure 5(b) shows the reverse ordering in mean cosine similarity: GenEval/OCR remains near zero, whereas OCR/PickScore is the most positively aligned. These measurements reveal persistent differences in pairwise gradient compatibility during joint training.

Capability forgetting under sequential training. An alternative strategy that avoids same-step conflicts is to optimize one objective at a time and switch periodically. While this eliminates gradient interference within each step, it introduces a distinct problem. When the active objective updates the shared parameters, updates to the shared parameters can degrade previously aligned capabilities. Table 4 demonstrates this phenomenon on both backbones. On FLUX.1-dev, after training GenEval,



(a) Conflict rate

(b) Mean cosine similarity

Figure 5: Gradient interference between objective pairs during Joint training on FLUX.1-dev. (a) Conflict rate, the fraction of steps with negative cosine similarity. (b) Mean cosine similarity.

Table 4: Sequential single-objective training exhibits forgetting of previously aligned capabilities. Each row adds one objective trained on top of the previous row's checkpoint. Percentages indicate the relative drop from the peak value (shown in **bold**).

Training stage	GenEval	OCR	PickScore
FLUX.1-dev			
Base model	0.644	0.591	21.993
+ GenEval	**0.883**	0.691	22.037
+ OCR	0.799 (↓9.5%)	**0.932**	22.024
+ PickScore	0.740 (↓16.2%)	0.828 (↓11.2%)	**23.273**
SD3.5-M			
Base model	0.671	0.616	21.775
+ GenEval	**0.955**	0.661	21.724
+ OCR	0.811 (↓15.1%)	**0.940**	21.615
+ PickScore	0.768 (↓19.6%)	0.812 (↓13.6%)	**23.357**

OCR, and PickScore sequentially, GenEval drops from 0.883 to 0.740 (a relative decrease of 16.2%) and OCR drops from 0.932 to 0.828 (a relative decrease of 11.2%). SD3.5-M exhibits the same qualitative pattern, with both previously optimized rewards decreasing after subsequent stages. This indicates that focused optimization, while resolving per-step gradient conflicts, requires explicit protection of inactive capabilities.

C.2 Main Results on SD3.5-M

Table 5 reports the quantitative comparison on SD3.5-M. Consistent with FLUX.1-dev results in Table 2, TRACE achieves the strongest overall performance across all three training rewards among the multi-objective baselines. Moreover, on the two non-optimized automatic metrics, TRACE remains above the base model and competitive with the evaluated baselines.

C.3 Hyperparameter Sensitivity

We study the sensitivity of TRACE to the chunk length H and to timestep-conditioned LoRA.

Chunk length H. The chunk length H controls the number of consecutive updates assigned to an objective before switching. Table 6 varies $H \in \{25, 50, 100, 200\}$ under a fixed budget of 3000 main-policy updates. Under the fixed 3,000-update budget, $H = 50$ achieves the highest reported scores among the tested values. Both shorter chunks ($H = 25$) and longer chunks ($H = 100$ or 200) reduce performance, with $H = 200$ producing the largest degradation. This pattern is consistent with a trade-off between per-chunk optimization depth and the frequency with which the scheduler can revisit regressing objective.

Table 5: Main results on SD3.5-M. GenEval, OCR, and PickScore are the three training-aligned objectives. They are evaluated on their corresponding Flow-GRPO test prompts. For GenEval, we report all six sub-scores and the official overall score. CLIPScore and Aesthetic Score serve as non-optimized evaluation metrics. Both are computed using the exact same test prompts as PickScore. **Bold** and underlined scores indicate the best and second-best values across all methods; higher is better.

Method	GenEval							OCR	Pick.	CLIP	Aes.
	Single	Two	Count.	Colors	Position	C.-Attr.	Overall				
SD3.5-M	0.988	0.808	0.625	0.798	0.270	0.540	0.671	0.616	21.775	36.983	5.957
Single-objective alignment											
Flow-GRPO (GenEval)	1.000	0.980	0.925	0.947	1.000	0.920	**0.962**	0.667	21.591	37.810	5.890
Flow-GRPO (OCR)	0.988	0.828	0.600	0.809	0.280	0.530	0.672	**0.963**	21.797	37.717	5.863
Flow-GRPO (PickScore)	1.000	0.929	0.700	0.883	0.300	0.640	0.742	0.737	**23.571**	37.125	**6.432**
Multi-objective alignment											
Joint training	1.000	0.889	0.775	0.748	0.425	0.570	0.735	0.755	22.285	37.421	6.058
Diffusion-Blend	1.000	0.960	0.775	0.904	0.610	0.770	0.837	0.787	22.681	**38.056**	6.165
SELMA	1.000	0.970	0.913	0.894	0.570	0.760	0.851	0.876	22.543	37.977	6.119
MapReduce-LoRA	1.000	0.990	0.913	0.840	0.740	0.830	0.886	0.923	22.564	38.017	6.114
Flow-OPD	1.000	1.000	0.913	0.862	0.780	0.810	0.894	0.933	22.649	37.198	6.300
TRACE (Ours)	1.000	0.948	0.938	0.895	0.855	0.790	0.904	0.946	22.982	37.925	6.398

Table 6: Sensitivity to chunk length H on FLUX.1-dev. The total main-policy update budget is fixed at 3000. The default $H = 50$ has the highest reported scores.

H	GenEval	OCR	PickScore
25	0.882	0.947	23.421
50 (default)	**0.896**	**0.962**	**23.576**
100	0.889	0.951	23.498
200	0.855	0.923	23.143

Tc-LoRA beyond routing. Table 7 provides a single-objective control. In the multi-objective setting, removing Tc-LoRA reduces all three objectives. In the single-objective setting, Tc-LoRA still improves all three reward-specific experts over standard LoRA. These single-objective results suggest that Tc-LoRA can also help outside the multi-objective setting; a parameter-matched control is needed to isolate time conditioning from additional capacity.

D IMPLEMENTATION DETAILS

Compute resources. All training experiments use eight NVIDIA A100 GPUs; latency measurements use a single NVIDIA A100 80GB GPU.

Hyperparameters. Following the denoising reduction strategy of Flow-GRPO (Liu et al., 2026), we use $T = 5$ denoising steps during training to accelerate rollout collection, while retaining the full default schedules at inference ($T = 28$ for FLUX.1-dev and $T = 40$ for SD3.5-M). We partition the normalized trajectory into $K = 5$ equal buckets. Other shared settings include: group size $G = 24$, LoRA rank $r = 64$, time embedding dimension $d_t = 256$, routing temperature $T_r = 0.5$, PPO clip $\epsilon = 0.2$, anchor weight $\kappa_j = 0.1$, chunk length of 50 updates.

Efficiency and Tc-LoRA overhead. Table 8 reports the per-image inference latency of TRACE on both backbones. Because TRACE relies on a single shared Tc-LoRA adapter and requires only one forward pass per denoising step, its inference pipeline remains structurally identical to standard LoRA. The only architectural difference is the introduction of lightweight timestep-conditioning heads. For FLUX.1-dev, we apply rank-64 LoRA to 342 linear layers. Each adapted layer is equipped with two conditioning projections that reuse the backbone's 256-dimensional time embedding. This adds exactly 32,896 parameters per layer, totaling 11.25M extra parameters. Compared

Table 7: Effect of timestep-conditioned LoRA (Tc-LoRA) under (a) the full multi-objective setting and (b) the single-objective Flow-GRPO setting. In (b), each row optimizes one reward and reports that reward.

(a) Multi-objective setting			
Variant	GenEval	OCR	PickScore
w/o Tc-LoRA	0.861	0.954	22.982
Full	**0.896**	**0.962**	**23.576**
(b) Single-objective Flow-GRPO setting			
Reward optimized	Standard LoRA	Tc-LoRA	Δ
GenEval	0.890	**0.911**	+0.021
OCR	0.972	**0.980**	+0.008
PickScore	23.591	**23.642**	+0.051

Table 8: Per-image inference latency (in seconds) on FLUX.1-dev ($T = 28$) and SD3.5-M ($T = 40$). All timings are measured at batch size 1 on a single NVIDIA A100 80GB GPU, averaged over 100 prompts.

Method	**FLUX.1-dev**	**SD3.5-M**
Base model	36.88	17.26
Standard LoRA	36.93	17.28
TRACE (Tc-LoRA)	38.96	18.50

to the standard LoRA baseline (179.3M), this corresponds to a marginal parameter overhead of 6.27%. Relative to standard LoRA, Tc-LoRA adds 2.03/1.22 seconds per image on both backbones, corresponding to increases of 5.5% on FLUX.1-dev and 7.1% on SD3.5-M.

Training-loop initialization. Initialize the Tc-LoRA policy and frozen anchors at the base model, populate each replay buffer with base-model trajectories on its objective's prompts, and initialize reward records and visit counts as described in Section 4.3. The independent probes do not initialize the main policy. For each policy update, the rollout policy is held fixed when computing importance ratios, and anchor snapshots are frozen when computing protection losses.

E LIMITATIONS AND FUTURE WORK

This paper presents TRACE, a framework for multi-objective alignment through temporal routing and anchor protection. We note several scope considerations and opportunities for further study:

- **Scaling to more objectives:** Practical text-to-image models must satisfy diverse alignment goals, including compositionality, text rendering, aesthetics, and safety. In this work, we validate TRACE on three representative objectives to demonstrate the effectiveness of our core mechanisms. Since the framework is inherently reward-agnostic, evaluating it on a broader and more diverse reward portfolio remains a necessary step.
- **Dynamic utility profiling:** The utility tensor is currently profiled once from the base model and remains fixed during training. However, the relative productivity of denoising phases may shift as the policy improves. Implementing periodic re-measurement could dynamically refine the routing profiles, representing a practical direction for future optimization.
- **Extension to process rewards:** Our method assumes terminal image-level rewards evaluated only on the final generation. Extending temporal routing to process reward models is a promising open direction. Since these models provide dense feedback at intermediate denoising steps, integrating them would enable fine-grained credit assignment across the entire trajectory.



Figure 6: Qualitative comparisons on FLUX.1-dev using GenEval test prompts. From left to right: base model, single-objective Flow-GRPO expert, and TRACE. Red text marks the compositional constraint under test.



Figure 7: Qualitative comparisons on FLUX.1-dev using OCR test prompts. From left to right: base model, single-objective Flow-GRPO expert, and TRACE. Red text marks the target string to be rendered.

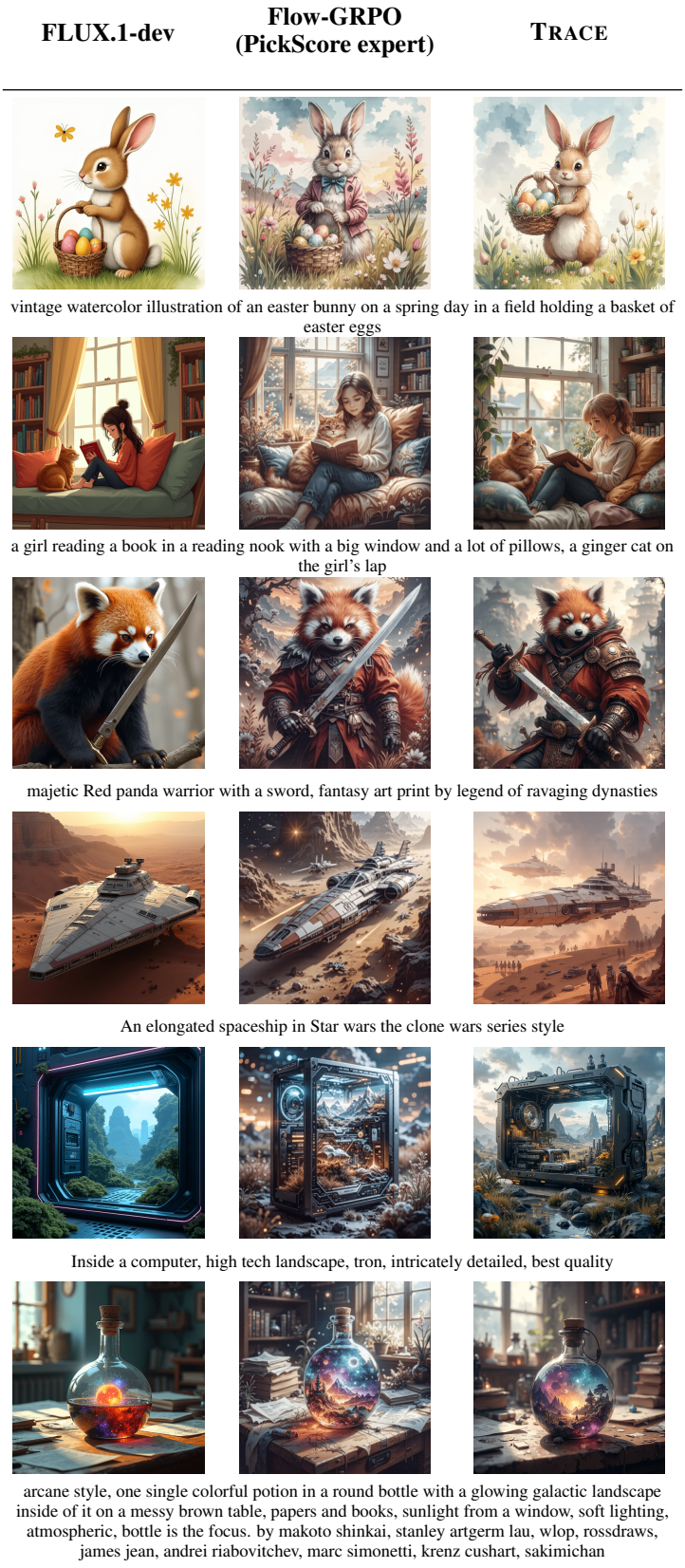


Figure 8: Qualitative comparisons on FLUX.1-dev using Pick-a-Pic v2 test prompts. From left to right: base model, single-objective Flow-GRPO expert, and TRACE.