

DOHF: Online Diffusion Fine-tuning with Doob’s h -transform Guidance

Zhengyi Guo

Jiayuan Sheng

Wenpin Tang

Abstract

Reward-based diffusion fine-tuning faces practical challenges when desirable outcomes are rare or conditioning corrections are costly to estimate. In this work, we propose **Diffusion Online h -guidance Fine-tuning (DOHF)**, which turns Doob’s h -transform into a practical online training algorithm. **DOHF** assigns optimality weights to generated samples, estimates the normalized local correction $\nabla \log h$ under the current rollout policy, and distills it directly into the generative model. Theoretically, we characterize the population-optimal DiffusionNFT update as well as the various classifier free guidance methods through a unified h -transform perspective. Methodologically, our framework accommodates black-box and non-differentiable rewards without additional network evaluations. We further show improved alignments under three empirical scenarios. Our work demonstrates how adapting probabilistic conditioning through inexpensive estimation and iterative distillation can improve generative learning across statistical sampling and visual generation.

1 Introduction

Doob’s h -transform provides a classical probabilistic mechanism for conditioning a Markov process on a terminal event or reweighting its terminal distribution [14]. Given a terminal weight $w(X_0)$, the associated function $h(x_t, t) = \mathbb{E}[w(X_0) \mid X_t = x_t]$ translates terminal information into a local correction of the intermediate dynamics through $\nabla \log h$. This perspective has recently become increasingly relevant to diffusion models. DEFT [7] formulates conditional generation through a generalized h -transform and learns an auxiliary network. Related work studies diffusion guidance under hard constraints [22]. More recently, DOIT [56] uses Doob’s h -transform to construct a training-free adaptation procedure.

From these viewpoints, a central question is how to efficiently estimate local correction for fine-tuning throughout the generative trajectory. The h -transform gives an explicit answer. For a current rollout policy p^{old} and a general reward weight w , the desired terminal distribution is

$$p^{ft}(x_0) \propto w(x_0)p^{old}(x_0),$$

while its intermediate score differs from that of the current model by exactly $\nabla_{x_t} \log h(x_t, t)$.

Existing h -transform approaches use this correction for inference-time guidance [26, 56], typically relative to a fixed reference model. This can become restrictive when the high-reward region is rare under the original proposal: the trajectories used to estimate the correction may provide little information precisely where adaptation is most needed. This motivates an online procedure in which the h -function itself is defined and estimated under the evolving rollout policy, and the resulting local correction is progressively distilled into the generative model.

In this work, we propose **Diffusion Online h -guidance Fine-tuning (DOHF)**. At each iteration, samples from the current rollout policy are assigned an optimality weight $w(X_0)$, from which we estimate the corresponding $\nabla \log h$ and distill the resulting correction directly into the generative model. This online procedure progressively shifts the rollout distribution toward more informative regions, while our simulation-based estimator remains applicable to black-box and

non-differentiable rewards. The same formulation naturally accommodates continuous rewards, hard or soft constraints, and relative optimality objectives.

We further establish a precise connection to DiffusionNFT [54], showing when its population-optimal velocity coincides with the exact h -transformed update. We evaluate **DOHF** on rare-mode adaptation, constrained contingency-table generation, and SD3.5-Medium text-to-image fine-tuning. Across these settings, **DOHF** consistently benefits from jointly adapting the rollout distribution and the h -correction, leading to faster adaptation and strong reward alignment when high-reward regions are poorly represented by the original model.

In conclusion, The key of **DOHF** is not the use of Doob’s h -transform itself, but turning its normalized local correction into an online, self-improving training target: the correction is re-estimated under the evolving rollout policy and distilled into the model that generates the next round of data.

Our main contributions are:

- **Optimality as an unified framework.** We formulate reward-based fine-tuning through a single weight $w : \mathcal{X} \rightarrow [0, 1]$ interpreted as the probability of an optimality event $\{O = 1\}$. Exponential tilting, hard constraints $w = \mathbf{1}\{r \geq z\}$, their smooth relaxations, and relative optimality objectives all become instances of one object, and each induces the exact target $p^{\text{ft}} \propto w p^{\text{old}}$. In contrast to formulations built on scalar advantages or on a positive/negative split of a rollout group, this retains the full conditioning structure, and in particular accommodates indicator-type hard constraints.
- **Online h -guidance distillation.** We turn the terminal weight into a local velocity correction $-\frac{t}{1-t} \nabla_x \log h$ and regress it directly into the generative model, re-estimating h under the evolving rollout policy at every iteration. Our simulation-based estimator reuses the velocity already computed at the current step, so leads to no additional network evaluations, and w need not be differentiable. Estimating h and improving the rollout distribution are two sides of one update: each iteration re-collects trajectories under the tilted policy, so the correction is estimated where adaptation is most needed.
- **An explicit h yields the guidance strength others must tune.** We provide a population-level characterization of DiffusionNFT, showing that its update is an h -attenuated version of the exact h -transform and becomes exact only under the state-dependent choice $\beta = 2h(x_t, t, c)$. Since **DOHF** estimates h explicitly, it realizes this state-dependent strength, whereas methods that do not must leave β a constant — which attenuates the update exactly in the low- h regions where high-reward mass is rare. We verify this prediction and the framework as a whole on rare-mode adaptation, constrained contingency-table generation, and SD3.5-M text-to-image fine-tuning, where our formulation is additionally compatible with classifier-free guidance, a design not available to forward-matching baselines.

Related Work. Existing h -transform methods provide foundations for our work. DEFT [7], hard-constraint guidance [22], and Doob’s Matching [3] learn conditioning corrections, while DOIT [56] estimates them through simulation. SVDD [31], twisted SMC [45], and Feynman-Kac steering [40] offer related inference-time approaches. Tilt Matching [35], Iterative Tilting [34], and Iterative Importance Fine-tuning [6] connects tilted distributions with iterative learning.

In parallel, a rapidly growing literature studies reward-based fine-tuning of diffusion models [42, 44]. Policy-based approaches formulate the reverse generation process as a policy and optimize it through RL, including DDPO [2], FlowGRPO [32], DanceGRPO [50], DPOK [17], and MixGRPO [30]. Sampler stochasticity and the resulting training-inference gap are studied in [39]. ReFL [47] and DRaFT [5] directly differentiate reward signals, while DiffusionDPO [43]

learns from preference pairs. Adjoint Matching [13, 21] and Score as Action [52, 53] cast fine-tuning as stochastic optimal control. Gradient-based construction [55] and flow-map alignment [24, 36] provide alternatives when reward derivatives are available. Regression-based approaches include Advantage Weighted Matching [49] and FlowAWR [18], while online reward-weighted flow matching [16] directly reweights newly generated samples. Reward Score Matching [28], a path-space perspective [48], and reward-based velocity matching [4] clarify the shared structure of these updates.

2 Preliminaries

Throughout, let c denote an optional conditioning context (e.g. a text prompt, a market covariate, or $c = \emptyset$ if unconditional). We suppress c from the notation below whenever it plays no special role. A pre-trained diffusion/flow model generates

$$X_0 \sim p^{\text{base}}(x_0 \mid c).$$

Moreover, we generate samples online, so we define the policy for rolling out is $p^{\text{old}}(x_0 \mid c)$.

2.1 A Unified Optimality Weight

We define optimality through a single weight function

$$w : \mathcal{X} \rightarrow [0, 1], \quad \mathbb{E}_{p^{\text{old}}}[w(X_0)] > 0.$$

We introduce an optimality variable [29, 54] defining whether the generated X_0 is optimal or not:

$$O := \mathbf{1}_{\{U \leq w(X_0)\}},$$

where $U \sim \text{Unif}(0, 1)$ is an auxiliary uniform random variable independent of X_0 . Intuitively, X_0 with higher weight $w(X_0)$ is more likely to be optimal. Then $\mathbb{P}(O = 1 \mid X_0 = x_0) = w(x_0)$, and conditioning the base model on $\{O = 1\}$ gives the fine-tuned target

$$p^{\text{ft}}(x_0 \mid c) := p^{\text{old}}(x_0 \mid O = 1, c) = \frac{w(x_0)}{\mathbb{P}(O = 1 \mid c)} p^{\text{old}}(x_0 \mid c) \propto w(x_0) p^{\text{old}}(x_0 \mid c).$$

Every choice of $w \in [0, 1]$ therefore corresponds to a genuine conditioning event. Note also that w and Cw ($C > 0$) induce the same p^{ft} , so any normalization on w has no effect.

Examples. (i) *Exponential tilting* [13]: $w(x_0) = \exp(\frac{r(x_0, c) - r_{\max}(c)}{\tau})$, temperature $\tau > 0$. (ii) *Hard threshold* [22]: $w(x_0) = \mathbf{1}_{\{r(x_0, c) \geq z\}}$, giving $p^{\text{ft}} \propto p^{\text{old}} \mathbf{1}_{\{r \geq z\}}$. (iii) *Sigmoid relaxation*: $w(x_0) = \sigma(\frac{r(x_0, c) - z}{\beta})$, $\sigma(u) = (1 + e^{-u})^{-1}$. Note that it recovers the hard threshold when $\beta \rightarrow 0^+$. (iv) *Loss/statistic-driven weights* [26]: $w(x_0) = \psi(\ell(x_0))$ for a nondecreasing ψ , e.g. extreme summary statistics of x_0 or loss of data x_0 defined by upstream models.

2.2 The h -Function

Define h -function as the probability of optimality given tuple (x_t, t, c) as

$$h(x_t, t, c) := \mathbb{P}(O = 1 \mid X_t = x_t, c) = \mathbb{E}[w(X_0) \mid X_t = x_t, c].$$

In particular $h(x_0, 0, c) = w(x_0)$. By Bayes' Rule, the fine-tuned marginal at time t is

$$p_t^{\text{ft}}(x_t \mid c) = p_t^{\text{old}}(x_t \mid O = 1, c) = \frac{1}{Z(c)} p_t^{\text{old}}(x_t \mid c) h(x_t, t, c), \quad (2.1)$$

where $Z(c)$ does not depend on x_t . Hence

$$\nabla_{x_t} \log p_t^{\text{ft}}(x_t \mid c) = \nabla_{x_t} \log p_t^{\text{old}}(x_t \mid c) + \nabla_{x_t} \log h(x_t, t, c), \quad (2.2)$$

so the score correction for fine-tuning is $\nabla_{x_t} \log h(x_t, t, c) = \nabla_{x_t} \log \mathbb{E}[w(X_0) \mid X_t = x_t, c]$.

2.3 Flow Matching

Consider Gaussian flow matching

$$X_t = (1 - t)X_0 + tX_1, \quad X_1 \sim \mathcal{N}(0, I), \quad t \in [0, 1],$$

Let $v^{old}(x, t, c)$ denote the velocity field corresponding to old policy. For this Gaussian interpolation, the velocity and score are related (see Appendix A for derivation) through

$$v(x, t, c) = -\frac{x + t\nabla_x \log p_t(x | c)}{1 - t}.$$

By Equation (2.2), the corresponding fine-tuned velocity provides a learning target:

$$v^{ft}(x, t, c) = v^{old}(x, t, c) - \frac{t}{1 - t} \nabla_x \log h(x, t, c). \quad (2.3)$$

3 Diffusion Online h -guidance Fine-tuning

For Flow Matching, the backward process can be reformulated as a SDE [25, 41]:

$$dX_t = \left[(1 + \lambda_t^2) v^{old}(X_t, t, c) + \frac{\lambda_t^2}{1 - t} X_t \right] dt + \lambda_t \sqrt{\frac{2t}{1 - t}} d\bar{B}_t, \quad (3.1)$$

where λ_t is a parameter that we can choose, and when $\lambda_t = 0$ it reverts to the original ODE form. Consider discrete reverse times $0 = t_0 < t_1 < \dots < t_L = 1$, and let $\phi_\ell^{old}(x_{t_{\ell-1}} | x_{t_\ell}, c)$ denote the backward Gaussian transition kernel, with mean $\mu_\ell(x_{t_\ell})$ and variance $\sigma_\ell^2 I$.

In order to do the regression (2.3), we need $\nabla_x \log h(x, t, c)$. Fortunately, we could use Monte Carlo Method to estimate it during the online fine-tuning process.

3.1 Simulation-Based Estimator of $\nabla_x \log h(x, t, c)$ [56]

Given $X_{t_\ell} = x_{t_\ell}$, simulate M backward trajectories $\{x_{t_{\ell-1}}^{(m)}, \dots, x_0^{(m)}\}_{m=1}^M$ using ϕ^{old} . Define

$$W_m := w(x_0^{(m)}), \quad G_m := \nabla_{x_{t_\ell}} \log \phi_\ell^{old}(x_{t_{\ell-1}}^{(m)} | x_{t_\ell}, c) = \frac{1}{\sigma_\ell^2} [\nabla_{x_{t_\ell}} \mu_\ell(x_{t_\ell})]^\top (x_{t_{\ell-1}}^{(m)} - \mu_\ell(x_{t_\ell})).$$

The plug-in estimator is

$$\widehat{\nabla \log h}_A = \frac{\sum_{m=1}^M W_m G_m}{(\sum_{m=1}^M W_m) \vee \tau_\ell}, \quad (3.2)$$

with small truncation level $\tau_\ell > 0$ for numerical stability. To alleviate the cost by M rollouts of length ℓ , we use one-step-lookahead surrogate

$$\hat{x}_0^{(m)} = x_{t_{\ell-1}}^{(m)} - t_{\ell-1} v^{old}(x_{t_\ell}, t_\ell, c), \quad x_{t_{\ell-1}}^{(m)} \sim \mathcal{N}(\mu_\ell(x_{t_\ell}), \sigma_\ell^2 I),$$

in place of $x_0^{(m)}$, at zero extra NFE (reusing $v^{old}(x_{t_\ell}, t_\ell, c)$ already computed). Moreover, it does not require w to be differentiable, which is more general for various terminal rewards mentioned in Section 2.1. If w is differentiable, we also provide an alternative estimator using finite difference method, which is provided in Appendix B.

Algorithm A Diffusion Online **h**-guidance Fine-tuning (**DOHF**)

Require: Pretrained v^{base} , prompt set $\mathcal{C}$, reward model r , schedule $\{t_i\}$, slow-mix schedule η_k

- 1: Initialize model weight $v_\theta \leftarrow v^{\text{base}}$, and sample-generation policy $v_\phi \leftarrow v_\theta$
- 2: **for** $k = 1, 2, \dots$ **do**
- 3: **for** prompt c in mini-batch **do**
- 4: Sample t_ℓ uniformly; roll out $v_\phi(\cdot, \cdot, c)$ to obtain x_{t_ℓ}
- 5: Compute $\widehat{\nabla \log h}(x_{t_\ell}, t_\ell, c)$ ▷ via 3.2 or B.1
- 6: $v^{\text{target}} \leftarrow \text{stopgradient}\left(v_\phi(x_{t_\ell}, t_\ell, c) - \frac{t_\ell}{1-t_\ell} \widehat{\nabla \log h}\right)$
- 7: $\mathcal{L} \leftarrow \mathcal{L} + \|v_\theta(x_{t_\ell}, t_\ell, c) - v^{\text{target}}\|_2^2$
- 8: **end for**
- 9: Update θ by one step on $\nabla_\theta \mathcal{L}$
- 10: $v_\phi \leftarrow \eta_k v_\phi + (1 - \eta_k) v_\theta$ ▷ slow-mix sample-generation policy
- 11: **end for**
- return** v_θ

Algorithm B Sampling with Classifier-Free Guidance

Require: Trained v_θ , prompt c , guidance scale $\gamma \geq 0$, schedule $\{t_i\}_{i=0}^N$

- 1: $x \leftarrow z \sim \mathcal{N}(0, I)$
- 2: **for** $i = 0, 1, \dots, N - 1$ **do**
- 3: $v_{\text{cond}} \leftarrow v_\theta(x, t_i, c)$, $v_{\text{uncond}} \leftarrow v_\theta(x, t_i, \emptyset)$
- 4: $v \leftarrow v_{\text{uncond}} + \gamma (v_{\text{cond}} - v_{\text{uncond}})$
- 5: $x \leftarrow x + v (t_{i+1} - t_i)$
- 6: **end for**
- return** x

3.2 Training and Sampling

We instantiate the training A for Text-to-Image fine-tuning, and for generalized task, prompt could be omitted. In terms of sample rolling-out, we use a slow-mix strategy: starts from on-policy and smooth it with exponential moving average. Note that we index time so that $t=1$ is pure noise and generation runs from $t_L = 1$ to $t_0 = 0$.

For sampling process, especially for Text-to-Image task, we could also implement the similar way used in Classifier-free Guidance [23], as we set some proportion of the prompt to be empty in training process, in order to improve its unconditional generation. For other tasks, just use fine-tuned v_θ to sample. For generalization purpose, we provide Algorithm B used in T2I task.

3.3 Population-Level Relation to DiffusionNFT

DiffusionNFT [54] is the closest online fine-tuning method to **DOHF**. Although its positive-negative construction does not explicitly estimate an h -transform, its population optimum admits an exact characterization in terms of the same h -function.

Let

$$h(x_t, t, c) = \mathbb{E}_{p^{\text{old}}} [w(X_0) \mid X_t = x_t, c].$$

DiffusionNFT parameterizes its positive and negative branches as

$$v_\theta^+ = (1 - \beta)v^{\text{old}} + \beta v_\theta, \quad v_\theta^- = (1 + \beta)v^{\text{old}} - \beta v_\theta,$$

where $\beta > 0$ is a fixed guidance coefficient.

Proposition 3.1. *At the population optimum of the double-branch DiffusionNFT objective, the*

learned velocity satisfies

$$v_{\theta}^*(x_t, t, c) = v^{\text{old}}(x_t, t, c) - \frac{2h(x_t, t, c)}{\beta} \frac{t}{1-t} \nabla_{x_t} \log h(x_t, t, c).$$

Consequently, its population optimum coincides with the exact h -transformed velocity

$$v^{\text{ft}} = v^{\text{old}} - \frac{t}{1-t} \nabla_{x_t} \log h$$

if and only if

$$\beta = 2h(x_t, t, c).$$

A complete derivation is provided in Appendix C.

This result reveals a key distinction between the two methods. With a fixed β , DiffusionNFT effectively follows the unnormalized direction $\nabla h = h \nabla \log h$; hence, its update is attenuated in low- h regions where optimal samples are rare. **DOHF** instead directly estimates the normalized correction $\nabla \log h$ under the evolving rollout policy and distills it into the model. It therefore realizes the exact h -transformed update without requiring a state-dependent guidance hyperparameter.

4 Experiment

4.1 Gaussian Mixture

To isolate the effect of online iteration from confounders, we conduct a controlled study on a 2D synthetic problem. The base distribution is an imbalanced two-mode Gaussian mixture

$$p_{\text{data}}(x) = 0.01 \cdot \mathcal{N}(x; \mu_0, \Sigma) + 0.99 \cdot \mathcal{N}(x; \mu_1, \Sigma),$$

with $\mu_0 = (-1, -1)^\top$, $\mu_1 = (1, 1)^\top$, and $\Sigma = 0.16 \mathbf{I}$. A rectified-flow velocity network (a small MLP) is first pretrained on samples from p_{data} .

The reward is the *soft posterior probability of belonging to the minority mode*:

$$r(x) = \mathbb{P}(\text{cluster} = 0 | x) = \frac{0.01 \cdot \mathcal{N}(x; \mu_0, \Sigma)}{0.01 \cdot \mathcal{N}(x; \mu_0, \Sigma) + 0.99 \cdot \mathcal{N}(x; \mu_1, \Sigma)} \in (0, 1).$$

This turns the alignment task into an extreme mode-seeking problem: the fine-tuned model should transport probability mass from the majority mode μ_1 into the minority mode μ_0 .

We compare two reward-tilting strategies, both starting from the same pretrained base:

- **Off-policy h -guidance [22].** A separate value network $h_{\phi}(x_t, t) \approx \mathbb{E}[r(X_0) | X_t = x_t]$ is regressed on rollouts drawn *exclusively* from the frozen base, then applied as guidance at inference via $\hat{v} = v_{\text{base}} - \frac{t}{1-t} \nabla \log h_{\phi}$.
- **DOHF(ours).** The same h -transform correction is instead distilled into the velocity network itself via online data collection.

Table 1 summarizes the outcome. Off-policy guidance takes 2000 update steps and still lifts the minority-mode fraction to some extent (1.12% $\rightarrow$ 14.88%): most of its training rollouts still land in the majority mode. In contrast, online h -distillation reaches 100% minority-mode concentration after just 200 steps — one order of magnitude fewer updates for a much better outcome: better preserves the target distribution with closer mean $(-0.82, -0.88)$ and standard deviation $(0.38, 0.38)$. Because each iteration re-collects trajectories under tilted policy, the reward-relevant regions of state space are explored progressively as the model itself improves. For visualization, see Figure 1.

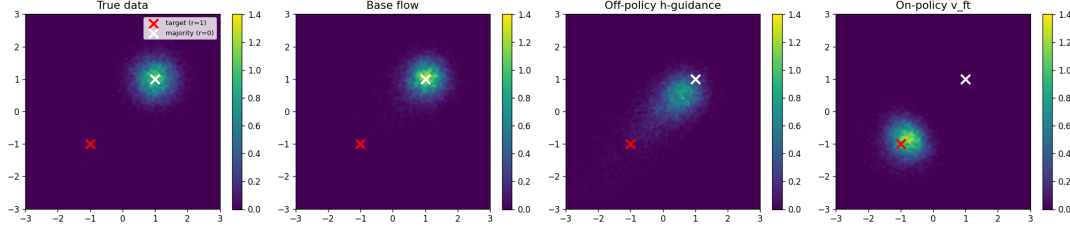


Figure 1: Visualization of on/off-policy h-transform guidance.

Method	Iterations	Minority-mode fraction	Mean $\pm$ std
True data	—	0.96%	$(0.98, 0.98) \pm (0.44, 0.44)$
Base pretrained	—	1.12%	$(0.99, 0.98) \pm (0.42, 0.41)$
Off-policy h -guidance	2000	14.88%	$(0.29, 0.25) \pm (0.73, 0.71)$
DOHF(ours)	200	100.00%	$(-0.82, -0.88) \pm (0.38, 0.38)$

Table 1: Fraction of samples falling in the minority cluster after fine-tuning.

4.2 Contingency Table

We consider contingency-table generation as an application of endogenous conditional sampling under the framework of [22]. Classical references include [8, 11], with connections to Burnside processes and double cosets discussed in [10, Section 3]. Such tables also arise in financial-network and systemic-risk applications [1, 20].

Objective. Let $X = (x_{ij}) \in \mathbb{Z}_+^{m \times n}$ be contingency table with prescribed row and column sums

$$\sum_j x_{ij} = r_i, \quad \sum_i x_{ij} = c_j. \quad (4.1)$$

In practice, these marginals may be noisy or partially observed, so we treat them as soft structural targets.

Given a pretrained flow model v^{base} , define

$$\mathcal{S}_2 = \sum_i \left(\sum_j x_{ij} - r_i \right)^2 + \sum_j \left(\sum_i x_{ij} - c_j \right)^2.$$

Exact feasibility corresponds to $\mathcal{S}_2 = 0$. Since exact matches are extremely rare for large tables, directly using the indicator reward leads to high-variance h -guidance estimates, usually there is no such table in a batch satisfying $\mathcal{S}_2 = 0$. We therefore use the smooth reward

$$w(x) = \exp(-\beta \mathcal{S}_2),$$

where larger β more closely approximates the hard constraint.

Birthday–deathday contingency table. We evaluate on the classical 12×12 birthday–deathday table of 82 descendants of Queen Victoria [12]. The target margins are

$$r = (6, 5, 5, 12, 12, 3, 10, 7, 3, 7, 9, 3), \quad c = (13, 4, 7, 10, 8, 4, 5, 3, 4, 9, 7, 8),$$

with total count 82, the original table is provided as Table 4.

We pre-train a flow model that generates nonnegative integer-valued 12×12 tables with total count 82; implementation details are given in Appendix D.2. Starting from the same pretrained

Method	Updates	Avg. Margin Error ↓	Max Margin Error ↓	W_2 vs. DG sampler [9] ↓
Uniform	–	0.802	1.975	13.48
Off-policy	800	0.379	1.005	11.00
DOHF	300	0.210	0.531	9.84

Table 2: Comparison of margin matching and distributional similarity. The margin errors are standardized by the corresponding generated marginal standard deviations.

model, we compare frozen off-policy h -guidance with **DOHF** using the reward $\exp(-\beta \mathcal{S}_2)$, where we take $\beta = 0.02$.

Among 500,000 uniformly sampled pre-training tables, none exactly satisfies all target margins, illustrating the sparsity of the target event and the necessity of using soft targets. As shown in Table 2, uniform sampling (as how we generate pre-training data) yields an average standardized margin error of 0.802 and a maximum error of 1.975. After only 300 updates, **DOHF** reduces these errors to 0.210 and 0.531, respectively, outperforming the off-policy baseline, which reaches 0.379 and 1.005 after 800 updates.

The Diaconis–Gangolli (DG) sampler [9] is a Markov Chain based method, generating approximately uniform contingency tables with fixed margins through repeated local 2×2 integer-preserving moves, which requires initial burn-in (in our experiment 200,000 steps are used). Although we are generating tables with perturbed margins, it is still worthwhile to check the difference between generated distributions. It is shown that **DOHF** also achieves the smallest Wasserstein distance to the DG sampler baseline among the three methods.

The advantage comes from adapting the rollout distribution toward the target region, which provides increasingly informative samples for subsequent h -transform updates. For this experiment, however, we set $v_\phi = v^{\text{base}}$ at each iteration rather than using the EMA update in Algorithm A, since the latter empirically caused collapse to nearly identical tables. This modification preserves rollout diversity while retaining the benefit of online h -distillation.

4.3 Image Fine-tuning

Our flow-based Text-to-Image model fine-tuning task is based on **SD3.5-Medium** [15] at 256×256 resolution. We demonstrate the potential of our method trained with the sum of three reward models: HPSV2.1 [46], PICKSCORE [27] and CLIPSCORE [37].

We will evaluate our method and other baselines on both in-domain and out-of-domain metrics: IMAGEREWARD [47], AESTHETIC SCORE [38], GENEVAL [19] and pair-wise LPIPS [51]. Note that we used average pair-wise LPIPS per prompt to measure image diversity. We aim to show that rolling out online is not only more efficient compared to off-policy fashion, but also provide a higher reward, combined with better semantic alignment. For Qualitative comparison, refer to Figure 2.

The training and testing prompt set is the ones used in Pick-a-Pic [27]. We finetuned with LoRA($\alpha = 32$, $r = 32$). Each iteration consists of 28 prompts, and generates $M = 8$ images per prompt via one-step-lookahead surrogate discussed in Section 3.1. For both training and testing roll-outs, 40 time steps are applied. Additional details are provided in Appendix E.

As shown in Table 3, our **DOHF** methodology works very well even without classifier-free-guidance (CFG) [23] using only 2k iterations, surpassing FlowGRPO [32] on most of the metrics using 9k iterations. Moreover, it performs slightly better than DiffusionNFT [54] overall in 2k iterations. To better explain their relationship, detailed discussion are provided in Appendix C.

However, our online h -guidance has CFG design which is not available in DiffusionNFT. During training, we set 10% of the prompts to be empty, in order to enforce model to have better unconditional generation, thus $v_\theta(\cdot, \cdot, c) - v_\theta(\cdot, \cdot, \emptyset)$ acts like guidance. During inference, velocity field is $v_\theta(\cdot, \cdot, c) + \gamma(v_\theta(\cdot, \cdot, c) - v_\theta(\cdot, \cdot, \emptyset))$, where $\gamma = 4$ is the default guidance scale.



Figure 2: Qualitative comparison across different guidance and fine-tuning methods. The prompts are taken from PICK-A-PIC [27].

It is shown in Table 3 that with guidance, all the quality scores increase to some extent, at little cost of diversity. It is worth noting that GENEVAL, the out-of-domain rule-based reward measuring how accurately generated images satisfy compositional prompts, increase significantly from 44.27% to 62.52%, showing its potential in semantic alignment.

5 Conclusion

We introduced **DOHF**, an online diffusion fine-tuning framework based on Doob’s h -transform. Our method turns reward-based adaptation into a local score or velocity correction that can be estimated online and directly distilled into the generative model. This perspective provides a unified treatment of continuous rewards, hard and soft constraints, and relative optimality objectives, while also clarifying the connection to existing methods such as DiffusionNFT.

Across rare-mode adaptation, constrained contingency table generation, and text-to-image fine-tuning, **DOHF** consistently demonstrates the benefit of adapting the rollout distribution together with the model, particularly when reward-relevant regions are difficult to sample via base model. These results suggest that online h -transform fine-tuning offers a promising alternative to both inference-time guidance and trajectory-based reinforcement learning. More broadly, the framework provides a flexible interface between probabilistic conditioning and generative-model optimization, and may be useful for a wide range of problems involving rare events, structured constraints, black-box rewards, and distributional adaptation.

Model	#Iter	GenEval	PickScore	ClipScore	HPSv2.1	Aesthetic	ImgRwd	LPIPS
SD3.5-M w/o CFG	–	4.49%	0.179	0.178	0.150	4.513	-1.809	0.771
SD3.5-M w/ CFG	–	41.35%	0.203	0.266	0.248	5.428	0.250	0.674
SD3.5-L w/ CFG	–	25.00%	0.193	0.243	0.208	4.919	-0.452	0.733
DOHF w/o CFG	2k	44.27%	<u>0.216</u>	<u>0.264</u>	<u>0.327</u>	6.327	1.115	0.538
DOHF w/ CFG	2k	62.52%	0.218	0.277	0.336	<u>6.240</u>	1.329	0.521
Off-policy Guidance	2k	19.52%	0.184	0.201	0.177	4.739	-1.320	0.746
	8k	38.96%	0.197	0.240	0.249	5.422	0.190	<u>0.655</u>
DiffusionNFT	2k	<u>49.39%</u>	0.214	0.261	0.305	6.151	<u>1.118</u>	0.525
FlowGRPO	2k	35.89%	0.205	0.252	0.272	5.924	0.636	0.602
	9k	45.66%	0.211	0.259	0.302	6.162	1.020	0.588

Table 3: Evaluation Results under resolution 256×256 . **Bold**: best; Underline: second best.

References

- [1] S. Albanesi and D. F. Vamossy. Predicting consumer default: A deep learning approach. Working Paper 26165, National Bureau of Economic Research, August 2019.
- [2] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024.
- [3] J. Chang, C. Duan, Y. Jiao, Y. Xu, and J. Z. Yang. Inference-time alignment for diffusion models via variationally stable doob’s matching, 2026.
- [4] J. Choi, W. Guo, Y. Zhu, A. Vahdat, M. Tao, J. Berner, and Y. Chen. Scaling reinforcement learning for diffusion models via velocity matching, 2026.
- [5] K. Clark, P. Vicol, K. Swersky, and D. J. Fleet. Directly fine-tuning diffusion models on differentiable rewards, 2024.
- [6] A. Denker, S. Padhy, F. Vargas, and J. Hertrich. Iterative importance fine-tuning of diffusion models, 2026.
- [7] A. Denker, F. Vargas, S. Padhy, K. Didi, S. Mathis, V. Dutordoir, R. Barbano, E. Mathieu, U. J. Komorowska, and P. Lio. DEFT: Efficient fine-tuning of diffusion models by learning the generalised h -transform. In *Advances in Neural Information Processing Systems*, volume 37, 2024.
- [8] P. Diaconis and B. Efron. Testing for independence in a two-way table: New interpretations of the chi-square statistic. *The Annals of Statistics*, 13(3):845–874, 1985.
- [9] P. Diaconis and A. Gangolli. Rectangular arrays with fixed margins. In *Discrete Probability and Algorithms*, volume 72 of *IMA Volumes in Mathematics and Its Applications*, pages 15–41. Springer, 1995.
- [10] P. Diaconis and M. Howes. Random sampling of contingency tables and partitions: Two practical examples of the burnside process. *Statistics and Computing*, 35(6), Aug. 2025.
- [11] P. Diaconis and B. Sturmfels. Algebraic algorithms for sampling from conditional distributions. *The Annals of Statistics*, 26(1):363–397, Feb. 1998.
- [12] P. Diaconis and B. Sturmfels. Algebraic algorithms for sampling from conditional distributions. *The Annals of Statistics*, 26(1):363–397, 1998.
- [13] C. Domingo-Enrich, M. Drozdal, B. Karrer, and R. T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control, 2025.
- [14] J. L. Doob. Conditional brownian motion and the boundary limits of harmonic functions. *Bulletin de la Société Mathématique de France*, 85:431–458, 1957.
- [15] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, D. Podell, T. Dockhorn, Z. English, K. Lacey, A. Goodwin, Y. Marek, and R. Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024.
- [16] J. Fan, S. Shen, C. Cheng, Y. Chen, C. Liang, and G. Liu. Online reward-weighted fine-tuning of flow matching with wasserstein regularization, 2025.

- [17] Y. Fan, O. Watkins, Y. Du, H. Liu, M. Ryu, C. Boutilier, P. Abbeel, M. Ghavamzadeh, K. Lee, and K. Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models, 2023.
- [18] Z. Fu, R. He, W. Shang, X. Ma, L. Wang, C. Liu, and S. Fu. Flowawr: Online adaptive flow reinforcement via advantage-weighted rectification, 2026.
- [19] D. Ghosh, H. Hajishirzi, and L. Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment, 2023.
- [20] R. Greenwood, S. G. Hanson, A. Shleifer, and J. A. Sørensen. Predictable financial crises. *The Journal of Finance*, 77(2):863–921, 2022.
- [21] Z. Guo, J. Sheng, D. D. Yao, and W. Tang. Improved techniques for fine-tuning flow models via adjoint matching: a deterministic control pipeline, 2026.
- [22] Z. Guo, W. Tang, and R. Xu. Conditional diffusion guidance under hard constraint: A stochastic analysis approach. *arXiv preprint arXiv:2602.05533*, 2026.
- [23] J. Ho and T. Salimans. Classifier-free diffusion guidance, 2022.
- [24] P. Holderrieth, D. Chen, L. Eyring, I. Shah, G. Anantharaman, Y. He, Z. Akata, T. Jaakkola, N. M. Boffi, and M. Simchowitz. Diamond maps: Efficient reward alignment via stochastic flow maps, 2026.
- [25] C.-W. Huang, J. H. Lim, and A. Courville. A variational perspective on diffusion-based generative models and score matching. In *Advances in Neural Information Processing Systems*, volume 34, 2021.
- [26] H. Kim, T. Lee, H. Kim, and G. D. Veciana. Generating informative samples for risk-averse fine-tuning of downstream tasks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.
- [27] Y. Kirstain, A. Polyak, U. Singer, S. Matiana, J. Penna, and O. Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation, 2023.
- [28] J. Lee, J. Chang, J. Kim, and J. C. Ye. Reward score matching: Unifying reward-based fine-tuning for flow and diffusion models, 2026.
- [29] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.
- [30] J. Li, Y. Cui, T. Huang, W. Kong, Y. Cheng, C. Zeng, Y. Ma, C. Fan, M. Yang, Z. Zhong, and L. Bo. Mixgrpo: Unlocking flow-based grpo efficiency with mixed ode-sde, 2026.
- [31] X. Li, Y. Zhao, C. Wang, G. Scalia, G. Eraslan, S. Nair, T. Biancalani, S. Ji, A. Regev, S. Levine, and M. Uehara. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding, 2024.
- [32] J. Liu, G. Liu, J. Liang, Y. Li, J. Liu, X. Wang, P. Wan, D. Zhang, and W. Ouyang. Flow-grpo: Training flow matching models via online rl, 2025.
- [33] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. Dpm-solver++. *Machine Intelligence Research*, 22(4):730–751, June 2025.
- [34] J. Pachebat, G. Conforti, A. Durmus, and Y. Janati. Iterative tilting for diffusion fine-tuning, 2025.

- [35] P. Potapchik, C.-K. Lee, and M. S. Albergo. Tilt matching for scalable sampling and fine-tuning, 2025.
- [36] P. Potapchik, A. Saravanan, A. Mammadov, A. Prat, M. S. Albergo, and Y. W. Teh. Meta flow maps enable scalable reward alignment, 2026.
- [37] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision, 2021.
- [38] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022.
- [39] J. Sheng, H. Zhao, H. Chen, D. D. Yao, and W. Tang. Understanding sampler stochasticity in training diffusion models for RLHF. *arXiv preprint arXiv:2510.10767*, 2025.
- [40] R. Singhal, Z. Horvitz, R. Teehan, M. Ren, Z. Yu, K. McKeown, and R. Ranganath. A general framework for inference-time scaling and steering of diffusion models, 2025.
- [41] W. Tang and H. Zhao. Score-based diffusion models via stochastic differential equations. *Statistic Surveys*, 19:28–64, 2025.
- [42] M. Uehara, Y. Zhao, T. Biancalani, and S. Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review. *arXiv preprint arXiv:2407.13734*, 2024.
- [43] B. Wallace, M. Dang, R. Rafailov, L. Zhou, A. Lou, S. Purushwalkam, S. Ermon, C. Xiong, S. Joty, and N. Naik. Diffusion model alignment using direct preference optimization, 2023.
- [44] G. I. Winata, H. Zhao, A. Das, W. Tang, D. D. Yao, S.-X. Zhang, and S. Sahu. Preference tuning with human feedback on language, speech, and vision tasks: a survey. *J. Artificial Intelligence Res.*, 82:2595–2661, 2025.
- [45] L. Wu, B. L. Trippe, C. A. Naesseth, D. M. Blei, and J. P. Cunningham. Practical and asymptotically exact conditional sampling in diffusion models, 2024.
- [46] X. Wu, Y. Hao, K. Sun, Y. Chen, F. Zhu, R. Zhao, and H. Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis, 2023.
- [47] J. Xu, X. Liu, Y. Wu, Y. Tong, Q. Li, M. Ding, J. Tang, and Y. Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation, 2023.
- [48] Y. Xu, Y. Zhang, S. Luo, L. Wang, and D. He. Designing reinforcement learning for diffusion models: A unified path-space view, 2026.
- [49] S. Xue, C. Ge, S. Zhang, Y. Li, and Z.-M. Ma. Advantage weighted matching: Aligning rl with pretraining in diffusion models, 2026.
- [50] Z. Xue, J. Wu, Y. Gao, F. Kong, L. Zhu, M. Chen, Z. Liu, W. Liu, Q. Guo, W. Huang, and P. Luo. Dancegrp: Unleashing grp on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.
- [51] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

- [52] H. Zhao, H. Chen, J. Zhang, D. D. Yao, and W. Tang. Scores as Actions: a framework of fine-tuning diffusion models by continuous-time reinforcement learning. *arXiv preprint arXiv:2409.08400*, 2024.
- [53] H. Zhao, H. Chen, J. Zhang, D. D. Yao, and W. Tang. Scores as Actions: fine tuning diffusion generative models by continuous-time reinforcement learning. In *ICML*, 2025.
- [54] K. Zheng, H. Chen, H. Ye, H. Wang, Q. Zhang, K. Jiang, H. Su, S. Ermon, J. Zhu, and M.-Y. Liu. Diffusionnft: Online diffusion reinforcement with forward process. *arXiv preprint arXiv:2509.16117*, 2026.
- [55] W. Zhou, X. Zhu, L. Kong, B. Chen, L. Zhang, Y. Liang, X. Hou, Y. Tian, X. Sun, Y. Wang, L. Li, S. Wu, L. Qu, F. Li, W. Liu, J. McAuley, and T.-S. Chua. On-policy self-distillation in diffusion models, 2026.
- [56] Q. Zhu, Z. Ye, H. Liu, Z. Wang, and M. Chen. Training-free adaptation of diffusion models via doob’s h -transform, 2026.

A Relation between score and velocity

Consider the rectified-flow interpolation

$$X_t = (1 - t)X_0 + t\epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

The corresponding velocity field is

$$v(x, t) = \mathbb{E}[\epsilon - X_0 \mid X_t = x].$$

Conditioned on X_0 , we have

$$X_t \mid X_0 \sim \mathcal{N}((1 - t)X_0, t^2 I),$$

and hence

$$\nabla_x \log p_t(x \mid X_0) = -\frac{x - (1 - t)X_0}{t^2} = -\frac{\epsilon}{t}.$$

Using the score marginalization identity,

$$\nabla_x \log p_t(x) = \mathbb{E}[\nabla_x \log p_t(x \mid X_0) \mid X_t = x],$$

we obtain

$$\nabla_x \log p_t(x) = -\frac{1}{t} \mathbb{E}[\epsilon \mid X_t = x].$$

Therefore,

$$\mathbb{E}[\epsilon \mid X_t = x] = x + (1 - t)\mathbb{E}[\epsilon - X_0 \mid X_t = x] = x + (1 - t)v(x, t).$$

Substituting this into the previous expression gives

$$\nabla_x \log p_t(x) = -\frac{x + (1 - t)v(x, t)}{t}.$$

Therefore, we have

$$v(x, t) = -\frac{x + t \nabla_x \log p_t(x)}{1 - t}.$$

B Alternative Estimator of $\nabla_x \log h$ [26]:

Approximate $h(x, t, c) \approx w(\hat{x}_0(x, t, c))$ via the Tweedie point estimate

$$\hat{x}_0(x, t, c) = x - t v^{old}(x, t, c),$$

so that $\nabla_x \log h(x, t, c) \approx [\partial_x \hat{x}_0(x, t, c)]^\top u$, with $u := \nabla_{\hat{x}_0} \log w(\hat{x}_0(x, t, c))$. Moreover, we should avoid the Jacobian $\partial_x \hat{x}_0$ via a finite-difference step of size $\epsilon > 0$:

$$\widehat{\nabla \log h_B} = \frac{1}{1 - t} \left[u + \frac{t^2}{\epsilon} \left(\nabla_x \log p_t^{old}(x + \epsilon u \mid c) - \nabla_x \log p_t^{old}(x \mid c) \right) \right], \quad (\text{B.1})$$

where $\nabla_x \log p_t^{old}(x \mid c) = -\frac{x + (1 - t)v^{old}(x, t, c)}{t}$. This method doesn't require M rollouts in Estimator 3.2, only 2 evaluations of v^{old} plus one backward pass through w . However, it requires w to be differentiable.

C Relation to Diffusion Negative-aware Fine-Tuning (NFT)

DiffusionNFT[54] arrives at a guidance mechanism built from the optimality variable $r(x_0, c) = p(O=1 \mid x_0, c)$, which is exactly our weight $w(x_0)$. We show that its guidance direction, and both its single-branch and double-branch training objectives, are special cases of the h -transform framework developed above.

C.1 Distribution split coefficient $\alpha(x_t)$ is equivalent to $h(x_t, t, c)$

DiffusionNFT defines $\pi^+(x_0 | c) := \pi^{old}(x_0 | O=1, c)$ and $\pi^-(x_0 | c) := \pi^{old}(x_0 | O=0, c)$. It can be shown that the posterior distribution of old policy can be separated as

$$\pi_{0|t}^{old}(x_0 | x_t, c) = \alpha(x_t)\pi_{0|t}^+(x_0 | x_t, c) + [1 - \alpha(x_t)]\pi_{0|t}^-(x_0 | x_t, c),$$

where

$$\alpha(x_t) := \frac{\pi_t^+(x_t | c)}{\pi_t^{old}(x_t | c)} \cdot \mathbb{E}_{\pi^{old}(x_0|c)}[w(x_0)].$$

Also,

$$v^{old}(x_t, t, c) = \alpha(x_t)v^+(x_t, t, c) + [1 - \alpha(x_t)]v^-(x_t, t, c). \quad (\text{C.1})$$

Since $\pi^+(x_0 | c) = p^{old}(x_0 | O=1, c) = p^{\text{ft}}(x_0 | c)$ by definition, we have $\pi_t^+ = p_t^{\text{ft}}$, and from equation 2.1,

$$\frac{p_t^{\text{ft}}(x_t | c)}{p_t^{old}(x_t | c)} = \frac{h(x_t, t, c)}{Z(c)}.$$

Since $\mathbb{E}_{\pi^{old}(x_0|c)}[w(x_0)] = Z(c)$, substituting gives

$$\alpha(x_t) = \frac{h(x_t, t, c)}{Z(c)} \cdot Z(c) = h(x_t, t, c).$$

As noted above, $\pi_t^+ \equiv p_t^{\text{ft}}$ as distributions, hence their posteriors over x_0 given x_t coincide, and since the velocity field is a linear functional of the posterior mean,

$$v^+(x_t, t, c) = v^{\text{ft}}(x_t, t, c) = v^{\text{base}}(x_t, t, c) - \frac{t}{1-t} \nabla_x \log h(x_t, t, c).$$

C.2 The Guidance Direction Δ and v^- as Anti-Guidance

According to equation C.1, v^{old}, v^+, v^- are on the same line. DiffuionNFT defines velocity improvement direction Δ as:

$$\Delta := \alpha(x_t)[v^+ - v^{old}] = [1 - \alpha(x_t)][v^{old} - v^-].$$

Using $\alpha = h$ and $v^+ - v^{old} = v^{\text{ft}} - v^{old} = -\frac{t}{1-t} \nabla_x \log h$,

$$\Delta = h \cdot \left(-\frac{t}{1-t} \nabla_x \log h \right) = -\frac{t}{1-t} h \nabla_x \log h = -\frac{t}{1-t} \nabla_x h,$$

using $h \nabla \log h = \nabla h$. That is, Δ is the (unnormalized) gradient $\nabla_x h$, rescaled by $-\frac{t}{1-t}$.

For v^- : by the same construction with $w^-(x_0) := 1 - w(x_0)$, $\pi^- = p^{old}(\cdot | O=0, c)$ has $h^-(x_t, t, c) = 1 - h(x_t, t, c)$, and

$$v^-(x_t, t, c) = v^{old}(x_t, t, c) - \frac{t}{1-t} \nabla_x \log (1 - h(x_t, t, c)).$$

Substituting into the first form of Δ , with $\nabla_x \log(1 - h) = -\nabla_x h / (1 - h)$,

$$-[1 - h][v^{old} - v^-] = -[1 - h] \cdot \frac{t}{1-t} \cdot \frac{\nabla_x h}{1-h} = -\frac{t}{1-t} \nabla_x h,$$

confirming consistency. Thus v^- is precisely the h -transform velocity for the *complementary* event $\{O = 0\}$: it is not an independent construction but an “anti-guidance” field pointing away from the optimal region.

C.3 Double-Branch Optimal Velocity

DiffusionNFT trains v_θ (it acts as an anchor for improvement guidance while training, and used for inference after training) via

$$\mathcal{L}(\theta) = \mathbb{E}_{c, \pi^{old}(x_0|c), t} \left[w(x_0) \|v_\theta^+ - v\|_2^2 + (1 - w(x_0)) \|v_\theta^- - v\|_2^2 \right],$$

where

$$v_\theta^+ := (1 - \beta)v^{old} + \beta v_\theta, \quad v_\theta^- := (1 + \beta)v^{old} - \beta v_\theta. \quad (\text{C.2})$$

The 'good velocity' is parametrized as a linear combination between anchor v_θ and v^{old} , and 'bad velocity' is pushing v^{old} to the opposite direction from the 'good' one.

By definition $p^{ft}(x_0 | c) = \frac{w(x_0)}{Z(c)} p^{old}(x_0 | c)$, i.e.

$$w(x_0) p^{old}(x_0 | c) = Z(c) p^{ft}(x_0 | c).$$

Multiplying both sides by the forward transition kernel $\phi(x_t | x_0)$, which is shared by p^{old} and p^{ft} , and applying Bayes' rule $p(x_0 | c)\phi(x_t | x_0) = p_t(x_t | c)p_{0|t}(x_0 | x_t, c)$ on each side:

$$w(x_0) p_t^{old}(x_t | c) \pi_{0|t}^{old}(x_0 | x_t, c) = Z(c) p_t^{ft}(x_t | c) \pi_{0|t}^+(x_0 | x_t, c).$$

Dividing both sides by $p_t^{old}(x_t | c)$ and using $\frac{p_t^{ft}(x_t | c)}{p_t^{old}(x_t | c)} = \frac{h(x_t, t, c)}{Z(c)}$ in Equation 2.1

$$w(x_0) \pi_{0|t}^{old}(x_0 | x_t, c) = Z(c) \cdot \frac{h(x_t, t, c)}{Z(c)} \cdot \pi_{0|t}^+(x_0 | x_t, c) = h(x_t, t, c) \pi_{0|t}^+(x_0 | x_t, c). \quad (\text{C.3})$$

The same argument with w replaced by $1 - w$ (equivalently h replaced by $1 - h$, π^+ by π^-) gives $(1 - w(x_0)) \pi_{0|t}^{old} = (1 - h(x_t, t, c)) \pi_{0|t}^-$.

Reduction to a weighted regression at fixed x_t . In NFT, loss function is defined as

$$\mathcal{L}(\theta, c) = \mathbb{E}_{\pi^{old}(x_0, x_t|c)} [w(x_0) \|v_\theta^+(x_t, t, c) - v\|_2^2 + (1 - w(x_0)) \|v_\theta^-(x_t, t, c) - v\|_2^2]$$

Since the joint density $\pi^{old}(x_0, x_t | c)$ factorizes as $\pi_t^{old}(x_t | c) \pi_{0|t}^{old}(x_0 | x_t, c)$, the inner expectation of the positive term at fixed (x_t, t, c) is

$$\int w(x_0) \pi_{0|t}^{old}(x_0 | x_t, c) \|v_\theta^+ - v\|^2 dx_0 = h(x_t, t, c) \int \pi_{0|t}^+(x_0 | x_t, c) \|v_\theta^+ - v\|^2 dx_0 = h \mathbb{E}_{\pi_{0|t}^+} [\|v_\theta^+ - v\|^2],$$

using the identity C.3 and v_θ^+ being independent of x_0 . Since v_θ^+ is a constant w.r.t. x_0 ,

$$\mathbb{E}_{\pi_{0|t}^+} [\|v_\theta^+ - v\|^2] = \|v_\theta^+ - \mathbb{E}_{\pi_{0|t}^+}[v]\|^2 + \underbrace{\text{Var}_{\pi_{0|t}^+}(v)}_{\text{independent of } \theta} = \|v_\theta^+ - v^+\|^2 + C_+,$$

where $\mathbb{E}_{\pi_{0|t}^+}[v] = v^+(x_t, c, t)$ by definition of the velocity field. The negative term reduces identically to $(1 - h) [\|v_\theta^- - v^-\|^2 + C_-]$. Dropping the θ -independent constants, the loss at fixed x_t, t, c is equivalent to minimizing

$$h \|v_\theta^+ - v^+\|_2^2 + (1 - h) \|v_\theta^- - v^-\|_2^2.$$

Let $Y := v_\theta - v^{old}$. Let $D := v^+ - v^{old}$ and $D^- := v^- - v^{old}$. Substituting the parameterization,

$$v_\theta^+ - v^+ = (1 - \beta)v^{old} + \beta v_\theta - (v^{old} + D) = \beta(v_\theta - v^{old}) - D = \beta\left(Y - \frac{D}{\beta}\right),$$

$$v_\theta^- - v^- = (1 + \beta)v^{old} - \beta v_\theta - (v^{old} + D^-) = -\beta(v_\theta - v^{old}) - D^- = -\beta\left(Y + \frac{D^-}{\beta}\right).$$

Hence $\|v_\theta^+ - v^+\|^2 = \beta^2 \|Y - D/\beta\|^2$ and $\|v_\theta^- - v^-\|^2 = \beta^2 \|Y + D^-/\beta\|^2$; factoring out $\beta^2 > 0$ (which does not affect the minimizer) gives the weighted least-squares problem

$$f(Y) = h \left\| Y - \frac{D}{\beta} \right\|_2^2 + (1 - h) \left\| Y + \frac{D^-}{\beta} \right\|_2^2.$$

Solving for the minimizer. f is a weighted sum of squared distances to two points, D/β (weight h) and $-D^-/\beta$ (weight $1-h$); its minimizer is the weighted average

$$Y^* = h \cdot \frac{D}{\beta} + (1-h) \cdot \left(-\frac{D^-}{\beta} \right) = \frac{hD - (1-h)D^-}{\beta}.$$

Since $v^{old} = h v^+ + (1-h) v^-$, subtracting v^{old} from both sides gives $0 = hD + (1-h)D^-$, i.e. $hD = -(1-h)D^-$. Substituting,

$$\begin{aligned} Y^* &= \frac{hD - (1-h)D^-}{\beta} = \frac{hD + hD}{\beta} = \frac{2hD}{\beta} \\ \implies v_{\theta^*} &= v^{old} - \frac{2h(x_t, t, c)}{\beta} \cdot \frac{t}{1-t} \nabla_x \log h(x_t, t, c). \end{aligned}$$

For fixed scalar β , this is a state-dependent damped version of v^{ft} : the guidance strength scales with the local optimality probability h , vanishing where $h \rightarrow 0$.

Therefore our algorithm is choosing β in the Definition C.2 to be $2h(x_t, t, c)$. This is reasonable because when you have $(x_t, t \mid c)$ is more likely to be optimal, that is $h(x_t, t, c) > 0.5$, we are more confident to push fine-tuned sampling strategy v_θ to be farther on the current improvement direction. On the other hand if $h(x_t, t, c) < 0.5$, we are not willing to trust v_θ that much and keep closer to v^{old} . However, in DiffusionNFT, they don't estimate h -function explicitly, so choosing β as a function depending on (x_t, t, c) is not viable. They have to leave β as a hyper-parameter (in their implementation, $\beta = 1$ is default value). Our algorithm provided more flexibility on this aspect.

C.4 Single-Branch Optimal Velocity

Training on the positive branch alone, $\mathcal{L}^+(\theta) = \mathbb{E}_{c, t, \pi^{old}(x_0|c)} [w \|v_\theta^+ - v\|_2^2]$, and using $w \pi_{0|t}^{old} = h \pi_{0|t}^+$,

$$\mathcal{L}^+(\theta) = \mathbb{E}_{c, t, x_t} \left[h(x_t, t, c) \cdot \mathbb{E}_{x_0 \sim \pi_{0|t}^+} \|v_\theta^+ - v\|_2^2 \right].$$

Pointwise in x_t , the L^2 -minimizer is the conditional mean regardless of the (positive) weight h :

$$v_\theta^{+*} = \mathbb{E}_{\pi_{0|t}^+} [v] = v^+ = v^{\text{ft}} \implies v_\theta^* = v^{\text{base}} + \frac{D}{\beta} = v^{\text{base}} - \frac{1}{\beta} \cdot \frac{t}{1-t} \nabla_x \log h,$$

which is *exact*, with no h -dependent damping, for any fixed β .

C.5 Unifying the Three via h

Comparing the two population optima:

$$\text{double-branch: } v_{\theta^*} = v^{\text{base}} - \frac{2h}{\beta} \cdot \frac{t}{1-t} \nabla \log h, \quad \text{single-branch: } v_{\theta^*} = v^{\text{base}} - \frac{1}{\beta} \cdot \frac{t}{1-t} \nabla \log h.$$

Setting $\beta(x_t, t, c) = 2h(x_t, t, c)$ in the double-branch objective exactly cancels the damping factor, giving $v_{\theta^*} = v^{\text{ft}}$; equivalently, this coincides with the single-branch optimum at $\beta = 1$. Thus

$$\text{double-branch, } \beta = 2h \iff \text{single-branch, } \beta = 1 \iff v_\theta^* = v^{\text{ft}} = v^{\text{base}} - \frac{t}{1-t} \nabla_x \log h.$$

D Discussion on Contingency Table

D.1 Birthday-Deathday Data

The following Table 4 is taken from the classical contingency-table example of [12].

Birth \ Death	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total
Jan	1	0	0	0	1	2	0	0	1	0	1	0	6
Feb	1	0	0	1	0	0	0	0	0	1	0	2	5
Mar	1	0	0	0	2	1	0	0	0	0	0	1	5
Apr	3	0	2	0	0	0	1	0	1	3	1	1	12
May	2	1	1	1	1	1	1	1	1	1	1	0	12
Jun	2	0	0	0	1	0	0	0	0	0	0	0	3
Jul	2	0	2	1	0	0	0	0	1	1	1	2	10
Aug	0	0	0	3	0	0	1	0	0	1	0	2	7
Sep	0	0	0	1	1	0	0	0	0	0	1	0	3
Oct	1	1	0	2	0	0	1	0	0	1	1	0	7
Nov	0	1	1	1	2	0	0	2	0	1	1	0	9
Dec	0	1	1	0	0	0	1	0	0	0	0	0	3
Total	13	4	7	10	8	4	5	3	4	9	7	8	82

Table 4: Birthday and death month of 82 descendants of Queen Victoria.

D.2 Pretraining the Contingency-Table Flow Model

To obtain the pretrained generative model used in the contingency-table experiment, we first construct a training distribution over 12×12 nonnegative tables with a fixed total count $N = 82$. For each training sample, we randomly generate an integer table $X \in \mathbb{Z}_+^{12 \times 12}$ satisfying

$$\sum_{i,j} X_{ij} = N.$$

We then normalize the table by its total count,

$$P_{ij} = \frac{X_{ij}}{N}, \quad \sum_{i,j} P_{ij} = 1,$$

so that each table can be viewed as a probability vector on the 144-dimensional simplex. A continuous flow model is pretrained on these normalized probability tables.

At inference time, the generated probability table $\hat{P}$ is converted back to an integer contingency table with total count exactly N . We first compute

$$Y_{ij} = N\hat{P}_{ij}, \quad \tilde{X}_{ij} = \lfloor Y_{ij} \rfloor.$$

Since rounding down may leave a residual

$$R = N - \sum_{i,j} \tilde{X}_{ij},$$

we assign one additional count to the R entries having the largest fractional parts

$$Y_{ij} - \lfloor Y_{ij} \rfloor.$$

This largest-remainder rounding procedure guarantees

$$\sum_{i,j} \tilde{X}_{ij} = N$$

while keeping the resulting integer table as close as possible to the continuous model output under this rounding scheme. If the total number is not hard-constrained to be the total number, we could let

$$Y_{ij} = N\hat{P}_{ij}, \quad \tilde{X}_{ij} = \text{round}(Y_{ij}).$$

E Additional Details on Text-to-Image Task

ODE Integration. Roll-out time steps are 40 for both training and inference. Note that we only solve the flow ODE step-by-step till $\nabla \log h$, then use one step estimation no matter using Estimator 3.2 or B.1. However, the solver used in training roll-outs is Euler, and during inference is DPM-Solver++[33], and CFG scale for online h -guidance w/ CFG is 4 by default, for SD3.5-M is 3.5 by default (both of them are taken due to the relatively best performance among different metrics).

Online Data Collection. Slow mixed policy is used to generate online samples. Follow the default design used in DiffusionNFT[54], $\eta_k = \min(0.001 \cdot k, 0.5)$, which means the data collection starts from on-policy, and gradually converges to $0.5 \cdot v^{old} + 0.5 \cdot v_\theta$ after 500 iterations. On the other hand, for the off policy guidance baseline, the roll-out policy and regression target are based on the base model v^{base} .

Other Configurations. In our T2I implementation, 7 Nvidia L40S is used. For each iteration per GPU, 4 prompts are randomly picked, and 12 initial noise is generated for each prompt. We do not use all time steps in training because two denominators in Flow-equivalent SDE 3.1 will approach to zero when $t \rightarrow 1$, therefore we set $t_{\min} = 0.05$ and $t_{\max} = 0.9$. Moreover, stochasticity parameter $\lambda_t = 1$ in SDE 3.1. Learning Rate is $3e - 5$, Gradient Clip is set to be 1.0, and slight LoRA regularization $1e - 5$ is applied.

F Additional Qualitative Comparison

In this section, we will provide more images showing the aesthetic quality and semantic alignment provided by our algorithm. For baselines, SD3.5-Medium, DiffusionNFT, FlowGRPO and off-policy h -guidance are compared in images.

For aesthetic quality, refer to Figure 3. For semantic alignment, refer to Figure 4.



Figure 3: Qualitative comparison across different methods and prompts. The prompts are: (Prompt 1) “Pixel art of a space station launching with two suns and a comet, featuring a retro space helmet, minigun, and a bright explosion in a dream-like atmosphere with CGA shading.”; (Prompt 2) “The devil using a heat press.”; (Prompt 3) “Ink drawing of a Shaolin monk in fighting pose.”; (Prompt 4) “A side view of a fox, with a flat design, Artwork of t-shirt graphic.”

	Prompt 1	Prompt 2	Prompt 3	Prompt 4
SD3.5M (w/o CFG)				
SD3.5M (w/ CFG)				
DiffusionNFT				
FlowGRPO				
Off-policy <i>h</i> -guidance				
DOHF (w/o CFG)				
DOHF (w/ CFG)				

Figure 4: Qualitative comparison across different methods and prompts. The prompts are: (Prompt 1, color) “A young woman with **red** hairs and **green** eyes”; (Prompt 2, spatial) “A penguin astronaut, in a penguin spacesuit, standing next to the apollo landing”; (Prompt 3, text) “A pikachu holding a sign that says ”**I’m cute!**””; (Prompt 4, text) “Gandalf holding a sign with a text ”**you shall not pass**””.