MEANFLOWNFT: BRINGING FORWARD-PROCESS RL TO AVERAGE-VELOCITY GENERATORS

Anonymous authors
Paper under double-blind review



Figure 1: Qualitative comparisons on `Wan2.1` 1.3B. Each row shows 3 frames sampled uniformly over time.



Figure 2: Qualitative comparisons on `SD3.5-M`. More visual results can be found in App. I.

ABSTRACT

MeanFlow generators achieve fast few-step sampling by predicting average velocities over time intervals, making them attractive for efficient generation. Reinforcement learning (RL) has become a powerful way to align diffusion and flow models with human preferences and task-specific objectives. In particular, recent work, DiffusionNFT, offers an efficient forward-process RL framework that does not require reverse-process trajectories or likelihood estimation. However, applying such RL methods to MeanFlow remains underexplored. DiffusionNFT optimizes instantaneous velocities, whereas MeanFlow samples with average velocities. To bridge this gap, we introduce **MeanFlowNFT**. Inspired by the MeanFlow identity, which bridges average and instantaneous velocities, we construct an induced instantaneous-velocity predictor. We apply the DiffusionNFT objective to this predictor, making reward optimization well-defined for MeanFlow. Sampling remains based on the average velocity, preserving MeanFlow's fast few-step generation. We further prove that MeanFlowNFT inherits DiffusionNFT's strict policy-improvement guarantee. Experiments on image and video generation show that MeanFlowNFT consistently improves baselines. Moreover, it outperforms prior state-of-the-art RL-tuned few-step generators on most metrics (6 of 8 on `SD3.5-M`), and can even surpass multi-step RL-tuned diffusion while using only a few sampling steps. For instance, on `Wan2.1`, 4-step MeanFlowNFT reaches a VBench score of 84.33, surpassing 50-step `LongCat-Video` RL (82.57).

1 INTRODUCTION

Diffusion (Ho et al., 2020; Song et al., 2021a;b) and flow (Lipman et al., 2023; Liu et al., 2023) models are the dominant paradigm for high-quality image and video generation. However, they

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.

synthesize a sample by integrating an instantaneous velocity over many sequential steps (Podell et al., 2024; Seedream et al., 2025; Qin et al., 2025), which makes generation slow. MeanFlow (Geng et al., 2026a) removes this bottleneck by predicting the *average velocity* over a time interval rather than the instantaneous velocity, so that one or a few steps suffice. This few-step efficiency makes MeanFlow an increasingly practical deployment target.

Reinforcement learning (RL) (Black et al., 2024; Fan et al., 2023; Wallace et al., 2024) from a scalar reward has become a standard tool for aligning such generative models with downstream objectives. For diffusion and flow models, the dominant recipe (Liu et al., 2026a; Xue et al., 2025) discretizes the reverse generative process and applies GRPO-style policy gradients (Guo et al., 2025), which require a stochastic policy and per-step likelihood estimation. DiffusionNFT (Zheng et al., 2026a) instead performs online RL *directly on the forward process*: it contrasts "positive" and "negative" generations (split by rewards) to form an implicit policy-improvement direction and folds rewards into a flow-matching objective. This makes training likelihood-free, solver-agnostic, and far more efficient than GRPO-based methods. However, RL for MeanFlow has received little attention. A key obstacle is that forward-process RL such as DiffusionNFT optimizes the *instantaneous* velocity, whereas MeanFlow samples with the *average* velocity. This raises a natural question:

Can we finetune a pretrained MeanFlow model with efficient forward-process RL to enable superior few-step generation?

In this paper, we answer this question affirmatively with **MeanFlowNFT**, the first forward-process RL framework for MeanFlow generators. Our starting point is the MeanFlow identity (Eq. (3)), an intrinsic link between the average and instantaneous velocity. Through it, a MeanFlow network yields an *induced instantaneous-velocity predictor*, to which we apply the DiffusionNFT-style objective during RL training. This brings two benefits. As in DiffusionNFT, training is likelihood-free and stays on the forward noising process. It also decouples optimization from sampling, so inference still uses MeanFlow's efficient few-step sampler. The objective never acts on the average velocity explicitly. Still, we prove in an idealized setting that the optimal induced predictor recovers DiffusionNFT's improved policy and carries this improvement over to the average-velocity network. On the practical side, we approximate the total-derivative terms in the induced predictors by finite differences. The trainable and reference predictors in our algorithm use the same finite-difference estimate, computed along the forward-process conditional velocity. As a result, MeanFlowNFT delivers strong generation quality (Figs. 1 and 2) while preserving MeanFlow's few-step efficiency.

Our contributions are summarized as follows:

- We propose **MeanFlowNFT**, the first forward-process RL framework for MeanFlow generators: it applies the DiffusionNFT-style objective to an induced instantaneous-velocity predictor derived from the average-velocity network. This keeps training likelihood-free and leaves the efficient few-step sampler unchanged.
- We provide theoretical guarantees: in an idealized setting, the optimum of the induced predictor matches the DiffusionNFT improved-policy target, and this policy improvement provably carries over to MeanFlow's average-velocity generator.
- Comprehensive experiments across text-to-image and text-to-video generation show that MeanFlowNFT consistently improves MeanFlow baselines and outperforms prior few-step methods. It even surpasses multi-step RL with far fewer sampling steps while scaling gracefully at test time.

2 Preliminaries

Throughout, $\boldsymbol{x}_t$ denotes a noised sample at time t and $\boldsymbol{c}$ a conditioning prompt. We write $\boldsymbol{u}$ for an average velocity (Geng et al., 2026a) and $\boldsymbol{v}$ for an instantaneous velocity field.

Notation. *Unless stated otherwise, all velocity fields and predictors are conditioned on $\boldsymbol{c}$, but we omit $\boldsymbol{c}$ from their arguments for notational brevity.*

2.1 Flow Matching

Flow Matching (Lipman et al., 2023; Liu et al., 2023) learns a probability-flow ODE that transports Gaussian noise to the data distribution $\pi(\boldsymbol{x}_0 \mid \boldsymbol{c})$. Given a schedule (α_t, σ_t) and writing

$\dot{f}_t \coloneqq \mathrm{d}f_t/\mathrm{d}t$, the forward process is $\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_0 + \sigma_t \boldsymbol{\epsilon}$, with $\boldsymbol{x}_0 \sim \pi(\cdot \mid \boldsymbol{c})$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Differentiating along a fixed pair $(\boldsymbol{x}_0, \boldsymbol{\epsilon})$ gives the *conditional velocity* $\boldsymbol{v}_t \triangleq \dot{\alpha}_t \boldsymbol{x}_0 + \dot{\sigma}_t \boldsymbol{\epsilon}$. For rectified flow (Liu et al., 2023), $\alpha_t = 1 - t$ and $\sigma_t = t$, so $\boldsymbol{v}_t = \boldsymbol{\epsilon} - \boldsymbol{x}_0$. Flow Matching trains a velocity predictor $\boldsymbol{v}_\theta(\boldsymbol{x}_t, t)$ by minimizing $\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t,\, \boldsymbol{c},\, \boldsymbol{x}_0 \sim \pi(\cdot|\boldsymbol{c}),\, \boldsymbol{\epsilon}} \big[w(t) \, \| \boldsymbol{v}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{v}_t \|_2^2 \big]$. Here $w(t)$ is a time-dependent loss weighting.

Although $\boldsymbol{v}_t$ is random given $\boldsymbol{x}_t$, the optimal predictor under this squared loss is the deterministic *marginal (instantaneous) velocity*

$$\boldsymbol{v}(\boldsymbol{x}_t, t) \triangleq \mathbb{E}[\boldsymbol{v}_t \mid \boldsymbol{x}_t, \boldsymbol{c}, t], \tag{1}$$

the conditional velocity averaged over the posterior of $(\boldsymbol{x}_0, \boldsymbol{\epsilon})$ given $\boldsymbol{x}_t$ (Lipman et al., 2023). At inference one integrates $\mathrm{d}\boldsymbol{x}_t/\mathrm{d}t = \boldsymbol{v}(\boldsymbol{x}_t, t)$ from noise to data using $\boldsymbol{v}_\theta \approx \boldsymbol{v}$, which typically requires many network evaluations.

2.2 MeanFlow

Flow Matching sampling is costly because integrating the generative ODE requires many instantaneous-velocity evaluations. MeanFlow (Geng et al., 2026a) instead learns a finite-interval flow map: predicting the average velocity from time t to an earlier time s lets sampling take large jumps rather than many small steps. It considers the *average velocity* over an interval $[s, t]$ along the ODE induced by $\boldsymbol{v}$ (Eq. (1)),

$$\boldsymbol{u}(\boldsymbol{x}_t, s, t) \triangleq \frac{1}{t-s} \int_s^t \boldsymbol{v}(\boldsymbol{x}_\tau, \tau) \, \mathrm{d}\tau, \qquad \frac{\mathrm{d}\boldsymbol{x}_\tau}{\mathrm{d}\tau} = \boldsymbol{v}(\boldsymbol{x}_\tau, \tau), \tag{2}$$

where the integral runs along the ODE trajectory from $\boldsymbol{x}_t$ to time s, so $\boldsymbol{u}$ depends on both endpoints (t, s) and recovers the instantaneous velocity as $s \to t$, $\lim_{s \to t} \boldsymbol{u}(\boldsymbol{x}_t, s, t) = \boldsymbol{v}(\boldsymbol{x}_t, t)$.

MeanFlow identity and training. Directly regressing $\boldsymbol{u}$ is impractical because Eq. (2) contains a path integral. Differentiating the displacement form $(t - s)\, \boldsymbol{u} = \int_s^t \boldsymbol{v} \, \mathrm{d}\tau$ with respect to t yields the exact *MeanFlow identity*

$$\boldsymbol{v}(\boldsymbol{x}_t, t) \;=\; \boldsymbol{u}(\boldsymbol{x}_t, s, t) \;+\; (t - s)\, \tfrac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{u}(\boldsymbol{x}_t, s, t), \tag{3}$$

with total derivative $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u} = \partial_t \boldsymbol{u} + (\partial_{\boldsymbol{x}} \boldsymbol{u})\, \boldsymbol{v}$. Because the marginal velocity is the posterior mean of $\boldsymbol{v}_t$ (Eq. (1)), replacing this intractable target with the conditional velocity $\boldsymbol{v}_t$ gives a trainable regression target. Specifically, a MeanFlow network $\boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t)$ is trained by minimizing $\mathcal{L}_{\mathrm{MF}}(\theta) = \mathbb{E}_{s,t,\, \boldsymbol{c},\, \boldsymbol{x}_0,\, \boldsymbol{\epsilon}} \big[w(t) \, \| \boldsymbol{u}_\theta - \mathrm{sg}(\boldsymbol{u}_{\mathrm{tgt}}) \|_2^2 \big]$, with target $\boldsymbol{u}_{\mathrm{tgt}} = \boldsymbol{v}_t - (t - s)\, (\partial_t \boldsymbol{u}_\theta + (\partial_{\boldsymbol{x}} \boldsymbol{u}_\theta)\, \boldsymbol{v}_t)$. The stop-gradient $\mathrm{sg}(\cdot)$ is applied to the whole target, so no gradient backpropagates through the total derivative term. When $s = t$, the derivative correction term vanishes and $\mathcal{L}_{\mathrm{MF}}$ reduces to $\mathcal{L}_{\mathrm{FM}}$.

Few-step sampling. Since $\boldsymbol{u}$ is the exact average velocity over $[s, t]$, it obeys the exact displacement identity $\boldsymbol{x}_s = \boldsymbol{x}_t - (t - s)\, \boldsymbol{u}(\boldsymbol{x}_t, s, t)$. The MeanFlow sampler therefore updates

$$\boldsymbol{x}_{t_{i-1}} \;=\; \boldsymbol{x}_{t_i} - (t_i - t_{i-1})\, \boldsymbol{u}_\theta(\boldsymbol{x}_{t_i}, t_{i-1}, t_i), \qquad i = N, \ldots, 1. \tag{4}$$

No instantaneous velocity is evaluated at inference, and a single step suffices in principle when $\boldsymbol{u}_\theta$ matches the true average velocity.

2.3 DiffusionNFT

DiffusionNFT (Zheng et al., 2026a) is an online diffusion RL method built on the forward-process Flow Matching objective rather than on policy gradients through a discretized reverse sampler.

Optimality partition. Let π^{old} be a frozen data-collection policy with marginal velocity $\boldsymbol{v}^{\mathrm{old}}(\boldsymbol{x}_t, t)$. For each prompt $\boldsymbol{c}$ one samples K images $\boldsymbol{x}_0^{1:K} \sim \pi^{\mathrm{old}}(\cdot \mid \boldsymbol{c})$ and scores each with a reward $r(\boldsymbol{x}_0, \boldsymbol{c}) \in [0, 1]$, read as the optimality probability $r(\boldsymbol{x}_0, \boldsymbol{c}) \triangleq p(\mathbf{o} = 1 \mid \boldsymbol{x}_0, \boldsymbol{c})$ in the RL-as-inference view (Levine, 2018). This induces positive and negative posteriors of π^{old}, $\pi^+ \propto r\, \pi^{\mathrm{old}}$ and $\pi^- \propto (1 - r)\, \pi^{\mathrm{old}}$, normalized by $Z(\boldsymbol{c}) \triangleq \mathbb{E}_{\pi^{\mathrm{old}}}[r]$ and $1 - Z(\boldsymbol{c})$, respectively. Under the objective $J(\pi) = \mathbb{E}_{\pi(\cdot|\boldsymbol{c})}[r]$, with $\pi \succ \pi'$ denoting $J(\pi) > J(\pi')$, one has $\pi^+ \succ \pi^{\mathrm{old}} \succ \pi^-$ for any non-degenerate (non-constant) reward, so π^+ is a valid improved policy.

Reinforcement guidance. Rather than treating π^+ as an optimization *point* (rejection finetuning, which discards negatives), DiffusionNFT extracts an optimization *direction* from the triplet

$(\pi^+, \pi^{\text{old}}, \pi^-)$. With

$$\alpha(\boldsymbol{x}_t, \boldsymbol{c}) \triangleq \frac{\pi_t^+(\boldsymbol{x}_t|\boldsymbol{c})}{\pi_t^{\text{old}}(\boldsymbol{x}_t|\boldsymbol{c})} Z(\boldsymbol{c}) = \mathbb{E}_{\pi^{\text{old}}}[r \mid \boldsymbol{x}_t, \boldsymbol{c}] \in [0, 1], \tag{5}$$

because $\boldsymbol{v}(\boldsymbol{x}_t, t)$ is a posterior mean (Eq. (1)), the marginal-velocity decomposition $\boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t) = \alpha\, \boldsymbol{v}^+(\boldsymbol{x}_t, t) + (1-\alpha)\, \boldsymbol{v}^-(\boldsymbol{x}_t, t)$ holds (Zheng et al., 2026a, Thm. 3.1), giving the shared *reinforcement guidance*

$$\Delta(\boldsymbol{x}_t, \boldsymbol{c}, t) \triangleq \alpha\big(\boldsymbol{v}^+(\boldsymbol{x}_t, t) - \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t)\big) = (1-\alpha)\big(\boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t) - \boldsymbol{v}^-(\boldsymbol{x}_t, t)\big). \tag{6}$$

Guiding the reference model along Δ with strength $1/\beta$ defines the target $\boldsymbol{v}^*(\boldsymbol{x}_t, t) \triangleq \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t) + \frac{1}{\beta}\Delta$, which mirrors classifier-free guidance (Ho & Salimans, 2021) and recovers $\boldsymbol{v}^*(\boldsymbol{x}_t, t) = \boldsymbol{v}^+(\boldsymbol{x}_t, t)$ at $\beta = \alpha$.

Implicit parameterization. Instead of learning separate positive and negative models, DiffusionNFT uses the implicit parameterization $\boldsymbol{v}_\theta^+(\boldsymbol{x}_t, t) := (1-\beta)\, \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t) + \beta\, \boldsymbol{v}_\theta(\boldsymbol{x}_t, t)$ and $\boldsymbol{v}_\theta^-(\boldsymbol{x}_t, t) := (1+\beta)\, \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t) - \beta\, \boldsymbol{v}_\theta(\boldsymbol{x}_t, t)$, and optimizes the reward-weighted objective

$$\mathcal{L}_{\text{DNFT}}(\theta) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{c},\, \boldsymbol{x}_0 \sim \pi^{\text{old}}(\cdot|\boldsymbol{c}),\\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),\, t}} \left[r\, \|\boldsymbol{v}_\theta^+(\boldsymbol{x}_t, t) - \boldsymbol{v}_t\|_2^2 + (1-r)\, \|\boldsymbol{v}_\theta^-(\boldsymbol{x}_t, t) - \boldsymbol{v}_t\|_2^2 \right]. \tag{7}$$

At the exact optimum, Zheng et al. (2026a, Thm. 3.2) show $\boldsymbol{v}_{\theta^*}(\boldsymbol{x}_t, t) = \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t) + \frac{2}{\beta}\Delta$. In particular, when $\beta = 2\alpha$ the optimum coincides with $\boldsymbol{v}^+(\boldsymbol{x}_t, t)$, so the trained model itself realizes the positive policy improvement.

3 MeanFlow Reinforcement via Forward-process RL

Here, we first build the proposed MeanFlowNFT (Sec. 3.1), which finetunes a MeanFlow generator with DiffusionNFT-style RL. Then, we analyze its closed-form induced optimum and policy-improvement guarantee (Sec. 3.2). Finally, we present the practical implementation (Sec. 3.3).

3.1 MeanFlowNFT

DiffusionNFT acts on the *instantaneous* velocity, while a MeanFlow network predicts the *average* velocity. Our key idea is to *keep the network in average-velocity space, but carry out optimization in instantaneous-velocity space.*

To achieve this, for any interval with $s \le t$, we substitute $\boldsymbol{u}_\theta$ into the MeanFlow identity (Eq. (3)) to build an *induced instantaneous-velocity predictor* $\boldsymbol{V}_\theta$[1],

$$\boldsymbol{V}_\theta(\boldsymbol{x}_t, s, t) \triangleq \boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t) + (t-s)\left[\partial_t \boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t) + (\partial_{\boldsymbol{x}} \boldsymbol{u}_\theta)(\boldsymbol{x}_t, s, t)\, \widehat{\boldsymbol{v}}_\theta(\boldsymbol{x}_t, t)\right]. \tag{8}$$

Here $\widehat{\boldsymbol{v}}_\theta(\boldsymbol{x}_t, t) \triangleq \boldsymbol{u}_\theta(\boldsymbol{x}_t, t, t)$ is the network's instantaneous velocity at time t, obtained by letting $s \to t$ as in Eq. (2). Inside Eq. (8), $\widehat{\boldsymbol{v}}_\theta(\boldsymbol{x}_t, t)$ serves only as the direction that $\partial_{\boldsymbol{x}} \boldsymbol{u}_\theta$ acts on, whereas $\boldsymbol{V}_\theta(\boldsymbol{x}_t, s, t)$ is the quantity we actually optimize. In the idealized setting, Eq. (3) gives $\boldsymbol{V}_\theta(\boldsymbol{x}_t, s, t) = \widehat{\boldsymbol{v}}_\theta(\boldsymbol{x}_t, t)$ for all $s \le t$, so the induced velocity is identical across interval starts for fixed $(\boldsymbol{x}_t, \boldsymbol{c}, t)$. Yet $\boldsymbol{V}_\theta$ is constructed from the full-interval prediction $\boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t)$, so optimizing it still reinforces the average velocity. In contrast, optimizing the single-time $\widehat{\boldsymbol{v}}_\theta$, *i.e.*, using only the $s = t$ case of Eq. (8), would reduce to plain Flow Matching (Liu et al., 2023) and lose MeanFlow's average-velocity parameterization and few-step sampling. To keep the update on $\boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t)$ and reduce costs, we wrap the total-derivative term of Eq. (8) in a stop-gradient during optimization (*i.e.*, $\mathrm{sg}\big(\partial_t \boldsymbol{u}_\theta + (\partial_{\boldsymbol{x}} \boldsymbol{u}_\theta)\, \widehat{\boldsymbol{v}}_\theta\big)$).

Now, we apply a DiffusionNFT-style objective to $\boldsymbol{V}_\theta$ to reinforce the underlying average velocity $\boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t)$. Concretely, let $\boldsymbol{V}^{\text{old}}$ be the same construction applied to the frozen reference $\boldsymbol{u}^{\text{old}}$ [2]. We then define implicit "positive" and "negative" predictors in instantaneous-velocity space,

[1] For clarity, lowercase $\boldsymbol{v}$ denotes marginal instantaneous velocities of policies (*e.g.*, $\boldsymbol{v}^+$ of π^+), while uppercase $\boldsymbol{V}$ denotes instantaneous predictors induced from MeanFlow average-velocity networks (*e.g.*, $\boldsymbol{V}_\theta$ induced by $\boldsymbol{u}_\theta$).

[2] Following DiffusionNFT (Zheng et al., 2026a), $\boldsymbol{u}^{\text{old}}$ is an exponential moving average (EMA) of $\boldsymbol{u}_\theta$.

$\boldsymbol{V}_\theta^+(\boldsymbol{x}_t, s, t) := (1-\beta)\boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t, s, t) + \beta \boldsymbol{V}_\theta(\boldsymbol{x}_t, s, t)$ and $\boldsymbol{V}_\theta^-(\boldsymbol{x}_t, s, t) := (1+\beta)\boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t, s, t) - \beta \boldsymbol{V}_\theta(\boldsymbol{x}_t, s, t)$. Mirroring Eq. (7), we optimize the following objective

$$\mathcal{L}_{\text{MFNFT}}(\theta) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{c},\, \boldsymbol{x}_0 \sim \pi^{\text{old}}(\cdot|\boldsymbol{c}),\\ \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\; s\le t}} \left[r \left\| \boldsymbol{V}_\theta^+(\boldsymbol{x}_t, s, t) - \boldsymbol{v}_t \right\|_2^2 + (1-r) \left\| \boldsymbol{V}_\theta^-(\boldsymbol{x}_t, s, t) - \boldsymbol{v}_t \right\|_2^2 \right]. \quad (9)$$

Here π^{old} is the policy whose average velocity is $\boldsymbol{u}^{\text{old}}$, and $\boldsymbol{x}_0 \sim \pi^{\text{old}}$ means $\boldsymbol{x}_0$ is obtained by running the few-step MeanFlow sampler (Eq. (4)) with $\boldsymbol{u}^{\text{old}}$. The reward r and the other symbols follow the same definitions as in DiffusionNFT (Sec. 2.3).

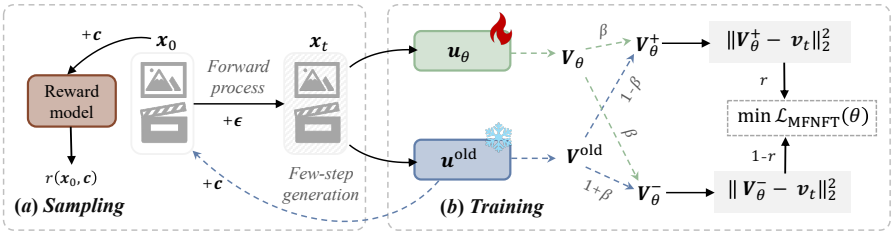


Figure 3: MeanFlowNFT keeps MeanFlow's *average-velocity* parameterization (Eq. (2)) and native few-step sampler (Eq. (4)), while constructing an *induced instantaneous-velocity predictor* $\boldsymbol{V}_\theta$ for forward-process negative-aware finetuning optimization.

In effect, MeanFlowNFT directly optimizes $\boldsymbol{V}_\theta$, and this improvement transfers to the average velocity $\boldsymbol{u}_\theta$. The overall pipeline is depicted in Fig. 3, and it gives two key benefits. **(i)** Training depends only on the forward noising process: following DiffusionNFT, it is likelihood-free and never unrolls the reverse denoising process, in contrast to GRPO-style policy gradients (Liu et al., 2026a; Li et al., 2026a) that discretize the reverse sampler and estimate per-step likelihoods. **(ii)** It decouples optimization from sampling: while optimization acts in *instantaneous-velocity* space, inference and RL sampling still deploy the *average-velocity* $\boldsymbol{u}_\theta$ through its efficient few-step MeanFlow sampler.

3.2 Optimum and Policy Improvement

In this subsection, we provide theoretical guarantees for MeanFlowNFT in three steps: the idealized pointwise optimum in induced-predictor space (Proposition 3.1), its recovery of the improved policy's marginal velocity (Corollary 3.2), and the transfer of this induced-velocity guarantee to the average-velocity network $\boldsymbol{u}_\theta$ (Theorem 3.4). All proofs are provided in App. D.

The objective Eq. (9) matches the predictors $\boldsymbol{V}_\theta^\pm$ to the conditional velocity $\boldsymbol{v}_t$. Treating the induced prediction at each $(\boldsymbol{x}_t, \boldsymbol{c}, s, t)$ as an unconstrained value gives the following closed-form optimum.

Proposition 3.1 (Idealized pointwise optimum)**.** *Conditioned on $(\boldsymbol{x}_t, \boldsymbol{c}, s, t)$, the idealized pointwise minimizer is*

$$\begin{aligned} \boldsymbol{V}_{\theta^*}(\boldsymbol{x}_t, s, t) &= \boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t, s, t) + \frac{2}{\beta}\widehat{\Delta}(\boldsymbol{x}_t, \boldsymbol{c}, s, t), \\ \widehat{\Delta}(\boldsymbol{x}_t, \boldsymbol{c}, s, t) &\triangleq \frac{1}{2}\mathbb{E}_{\pi^{\text{old}}}\left[(2r-1)\left(\boldsymbol{v}_t - \boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t, s, t)\right) \,\middle|\, \boldsymbol{x}_t, \boldsymbol{c}, s, t\right]. \end{aligned} \quad (10)$$

The optimum mirrors the DiffusionNFT fixed point $\boldsymbol{v}_{\theta^*}(\boldsymbol{x}_t, t) = \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t) + \frac{2}{\beta}\Delta$ (Eq. (6)), with the induced reference $\boldsymbol{V}^{\text{old}}$ and guidance $\widehat{\Delta}$ in place of $\boldsymbol{v}^{\text{old}}$ and Δ.

Corollary 3.2 ($\boldsymbol{V}_{\theta^*}$ recovers the improved marginal velocity)**.** *Setting the guidance strength to $\beta = 2\alpha(\boldsymbol{x}_t, \boldsymbol{c})$, Eq. (10) collapses to*

$$\boldsymbol{V}_{\theta^*}(\boldsymbol{x}_t, s, t) = \boldsymbol{v}^+(\boldsymbol{x}_t, t) \quad (11)$$

for all $s \le t$, the marginal instantaneous velocity of the improved policy π^+.

This gives the DiffusionNFT improvement target in $\boldsymbol{V}$-space. To transfer it to the deployed average velocity, we use the following consequence of the MeanFlow identity.

Lemma 3.3 (MeanFlow consistency)**.** *Let $\boldsymbol{v}$ be an instantaneous velocity field. If $\boldsymbol{u}$ satisfies*

$$\boldsymbol{u}(\boldsymbol{x}_t, s, t) + (t-s)\left[\partial_t \boldsymbol{u}(\boldsymbol{x}_t, s, t) + (\partial_{\boldsymbol{x}}\boldsymbol{u})(\boldsymbol{x}_t, s, t)\boldsymbol{v}(\boldsymbol{x}_t, t)\right] = \boldsymbol{v}(\boldsymbol{x}_t, t) \quad (12)$$

for all $s \le t$, then $\boldsymbol{u}$ is the exact average velocity of the ODE $\dot{\boldsymbol{x}}_\tau = \boldsymbol{v}(\boldsymbol{x}_\tau, \tau)$ over $[s, t]$.

Theorem 3.4 (MeanFlow policy improves). *In the setting of Corollary 3.2, if the induced optimum is attained for all intervals $s \le t$, the optimal average velocity $\boldsymbol{u}_{\theta^*}$ is the exact average velocity of the ODE induced by $\boldsymbol{v}^+$. Therefore the MeanFlow policy induced by $\boldsymbol{u}_{\theta^*}$ coincides with π^+, and consequently*

$$J(\pi_{\theta^*}) = J(\pi^+) > J(\pi^{\text{old}}).$$

Indeed, if $\boldsymbol{V}_{\theta^*}(\boldsymbol{x}_t, s, t) = \boldsymbol{v}^+(\boldsymbol{x}_t, t)$ for all $s \le t$, then setting $s = t$ in Eq. (8) removes the $(t-s)$ term and gives $\widehat{\boldsymbol{v}}_{\theta^*}(\boldsymbol{x}_t, t) = \boldsymbol{u}_{\theta^*}(\boldsymbol{x}_t, t, t) = \boldsymbol{v}^+(\boldsymbol{x}_t, t)$. Substituting this identity back into Eq. (8) shows that $\boldsymbol{u}_{\theta^*}$ satisfies Eq. (12) with $\boldsymbol{v} = \boldsymbol{v}^+$. By Lemma 3.3, $\boldsymbol{u}_{\theta^*}$ is then the exact average velocity of the ODE induced by $\boldsymbol{v}^+$.

3.3 Practical Implementation

Eq. (8) provides the theoretical construction analyzed in Sec. 3.2. Guided by this construction, the practical implementation introduces the following design choices to reduce computational cost and improve training stability. Algorithm 1 summarizes the resulting MeanFlowNFT update.

Algorithm 1 MeanFlowNFT (one update step)

Require: pretrained MeanFlow $\boldsymbol{u}_\theta$, frozen reference $\boldsymbol{u}^{\text{old}}$ (EMA of $\boldsymbol{u}_\theta$), reward r, guidance β

// Sampling

sample prompt $\boldsymbol{c}$; roll out $\boldsymbol{x}_0$ with $\boldsymbol{u}^{\text{old}}$; evaluate $r(\boldsymbol{x}_0, \boldsymbol{c}) \in [0,1]$

sample an interval $s \le t$, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\boldsymbol{x}_t \leftarrow \alpha_t \boldsymbol{x}_0 + \sigma_t \boldsymbol{\epsilon}$; $\quad \boldsymbol{v}_t \leftarrow \dot{\alpha}_t \boldsymbol{x}_0 + \dot{\sigma}_t \boldsymbol{\epsilon}$

// Training

$\boldsymbol{x}_{t\pm\Delta t} \leftarrow \boldsymbol{x}_t \pm \Delta t\, \boldsymbol{v}_t$

$\boldsymbol{d} \leftarrow \big[\boldsymbol{u}^{\text{old}}(\boldsymbol{x}_{t+\Delta t}, s, t+\Delta t) - \boldsymbol{u}^{\text{old}}(\boldsymbol{x}_{t-\Delta t}, s, t-\Delta t)\big]/(2\Delta t)$

$\boldsymbol{V}_\theta \leftarrow \boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t) + (t-s)\, \boldsymbol{d}$

$\boldsymbol{V}^{\text{old}} \leftarrow \boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t, s, t) + (t-s)\, \boldsymbol{d}$

$\boldsymbol{V}_\theta^+ \leftarrow (1-\beta)\boldsymbol{V}^{\text{old}} + \beta \boldsymbol{V}_\theta$; $\quad \boldsymbol{V}_\theta^- \leftarrow (1+\beta)\boldsymbol{V}^{\text{old}} - \beta \boldsymbol{V}_\theta$

$\mathcal{L}_{\text{MFNFT}} \leftarrow r\, \|\boldsymbol{V}_\theta^+ - \boldsymbol{v}_t\|_2^2 + (1-r)\, \|\boldsymbol{V}_\theta^- - \boldsymbol{v}_t\|_2^2$

update θ by $\nabla_\theta \mathcal{L}_{\text{MFNFT}}$

Finite-difference derivative. The total-derivative term $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}_\theta = \partial_t \boldsymbol{u}_\theta + (\partial_{\boldsymbol{x}} \boldsymbol{u}_\theta)\, \widehat{\boldsymbol{v}}_\theta$ in Eq. (8) can be seen as a Jacobian-vector product (JVP)[3]. Computing this JVP with forward-mode automatic differentiation (`torch.func.jvp`) is expensive and not fully compatible with Fully Sharded Data Parallel (FSDP) (Zhao et al., 2023) training. Following prior works (Nie et al., 2026; Gu et al., 2026), we instead approximate it by a central finite difference in t,

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t) \;\approx\; \frac{\boldsymbol{u}_\theta(\boldsymbol{x}_{t+\Delta t}, s, t+\Delta t) - \boldsymbol{u}_\theta(\boldsymbol{x}_{t-\Delta t}, s, t-\Delta t)}{2\,\Delta t}, \tag{13}$$

where $\boldsymbol{x}_{t\pm\Delta t} = \boldsymbol{x}_t \pm \Delta t\, \boldsymbol{v}_t$ displace $\boldsymbol{x}_t$ along the direction $\boldsymbol{v}_t$, instead of $\widehat{\boldsymbol{v}}_\theta$ (justified below).

Shared total derivative. Following DiffusionNFT, we take $\boldsymbol{u}^{\text{old}}$ to be an EMA of $\boldsymbol{u}_\theta$, keeping the reference close to the trainable model during online training. Our objective, however, compares the induced predictors. If each predictor uses its own total derivative, their difference is

$$\boldsymbol{V}_\theta(\boldsymbol{x}_t, s, t) - \boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t, s, t) = \boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t) - \boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t, s, t) + (t-s)\left(\tfrac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}_\theta - \tfrac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}^{\text{old}}\right). \tag{14}$$

The EMA keeps the average-velocity difference $\boldsymbol{u}_\theta - \boldsymbol{u}^{\text{old}}$ small, but it does not control the difference between their derivatives. This additional term destabilizes $\boldsymbol{V}_\theta - \boldsymbol{V}^{\text{old}}$ and causes training to collapse (Fig. 6). We therefore use the same derivative $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}^{\text{old}}$ for both predictors, so the derivative-difference term cancels exactly and leaves $\boldsymbol{V}_\theta(\boldsymbol{x}_t, s, t) - \boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t, s, t) = \boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t) - \boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t, s, t)$. This choice also saves computation because the finite-difference derivative is evaluated once for $\boldsymbol{u}^{\text{old}}$ rather than separately for both predictors.

Conditional-velocity direction. The idealized direction $\widehat{\boldsymbol{v}}_\theta(\boldsymbol{x}_t, t) = \boldsymbol{u}_\theta(\boldsymbol{x}_t, t, t)$ in Eq. (8) requires an extra network evaluation. We instead reuse the forward-process conditional velocity $\boldsymbol{v}_t = \dot{\alpha}_t \boldsymbol{x}_0 +$

[3] The Jacobian $[\partial_{\boldsymbol{x}}\boldsymbol{u}_\theta,\ \partial_s \boldsymbol{u}_\theta,\ \partial_t \boldsymbol{u}_\theta]$ stacks the first-order partial derivatives of $\boldsymbol{u}_\theta$, and its product with the direction $[\widehat{\boldsymbol{v}}_\theta,\ 0,\ 1]$ recovers the total derivative $\partial_t \boldsymbol{u}_\theta + (\partial_{\boldsymbol{x}} \boldsymbol{u}_\theta)\, \widehat{\boldsymbol{v}}_\theta$.

$\dot{\sigma}_t\boldsymbol{\epsilon}$, which is already available at no extra cost. In practice, it is used to compute the displaced points $\boldsymbol{x}_{t\pm\Delta t}$ in Eq. (13). This choice is decisive: tying the direction to the shifting $\widehat{\boldsymbol{v}}_\theta$ collapses training, whereas $\boldsymbol{v}_t$ stays stable (Fig. 7).

4 Experiments

4.1 Implementation Details

Models. We validate MeanFlowNFT on both image and video generation. For image generation, we use Stable Diffusion 3.5-Medium (`SD3.5-M`) (Esser et al., 2024) at 512×512 for training and 1024×1024 for evaluation following DiffusionNFT (Zheng et al., 2026a). For video generation, we use `Wan2.1` 1.3B (Wan et al., 2025) at 480p with 81 frames. In each case, the MeanFlow policy is obtained by distilling the base model with AnyFlow (Gu et al., 2026). For video generation, we directly adopt the publicly released AnyFlow checkpoint. Following DiffusionNFT (Zheng et al., 2026a), the entire pipeline is *CFG-free* (Ho & Salimans, 2021). At inference, the model is deployed with its native MeanFlow few-step sampler with different sampling steps.

Rewards. For image RL, we train with `CLIPScore` (Hessel et al., 2021), `PickScore` (Kirstain et al., 2023), and `HPSv2` (Wu et al., 2023) on `PickScore` prompt set. For video RL, we train with `HPSv3-general` and `HPSv3-percentile` (Ma et al., 2025) together with the motion-quality (`MQ`) and text-alignment (`TA`) scores of `VideoAlign` (Liu et al., 2026b), following the reward setup of `LongCat-Video` (Team et al., 2025). Here, the training prompts are those used by DanceGRPO (Xue et al., 2025). More details are deferred to the App. E.

Training. We finetune with LoRA (Hu et al., 2022) (rank 32, scaling factor 64) applied to all linear layers in the attention blocks. We set the guidance strength to β=0.1 and update $\boldsymbol{u}^{\text{old}}$ with the same EMA schedule in DiffusionNFT (Zheng et al., 2026a), $\eta_i = \min(0.001\,i,\ 0.5)$ (i denotes the ith update step), and weight the Kullback–Leibler (KL) regularization term by 10^{-4}. Optimization uses AdamW (Loshchilov & Hutter, 2019) with a constant learning rate of 3×10^{-6}. For each prompt, we adopt the generator with 4 steps to collect rollouts. For `SD3.5-M` we use a group size of 24 with 48 groups per update on 8 NVIDIA H20 GPUs. For `Wan2.1` we use a group size of 16 with 8 groups per epoch on 32 NVIDIA H20 GPUs.

Evaluation. For images, we report `ImageReward` (Xu et al., 2023), `CLIPScore`, `Aesthetic Score` (Schuhmann, 2022), `PickScore`, `HPSv2`, `HPSv3` (Ma et al., 2025), `GenEval2` (Kamath et al., 2025), and `OCR` (Liu et al., 2026a). We evaluate `HPSv3` and `OCR` following Liu et al. (2026a); Xue et al. (2025), adopt the official setting for `GenEval2`, and compute the remaining metrics on `DrawBench` (Saharia et al., 2022). For video, we report `VBench` (Huang et al., 2024) together with the four video training metrics on a held-out set of 256 prompts that are excluded from training.

4.2 Main Results

Table 1: Main results on `SD3.5-M` for image generation with 1024×1024 resolution. Among few-step models, **bold** and underline denote the best and second-best results. †denotes results reproduced using open-source checkpoints. All other results were reproduced by us.

Method	ImageReward↑	CLIPScore↑	Aesthetic↑	PickScore↑	HPSv2↑	HPSv3↑	GenEval2↑	OCR↑
Multi-step models (40 steps)								
SD3.5-M (w/o CFG)†	-0.4770	0.2391	5.1514	20.6587	0.2109	3.4601	0.1171	0.1439
+ DiffusionNFT†	1.4066	0.2889	5.9647	23.6440	0.3236	13.5959	0.2613	0.9098
SD3.5-M (w/ CFG)†	0.9253	0.2880	5.3811	22.4638	0.2822	11.2489	0.2038	0.5449
+ Flow-GRPO†	1.0193	0.2912	5.2843	22.4783	0.2657	9.8059	0.2638	0.6282
Few-step models (4 steps)								
DMD	0.9241	0.2841	5.5055	22.2807	0.2874	11.7156	0.2042	0.3996
CDM	1.0307	0.2819	5.5721	22.4160	0.2976	12.5190	0.2020	0.3225
AnyFlow	1.1125	0.2886	5.4203	22.4789	0.2969	12.0691	0.1896	0.4520
RTDMD†	1.2315	0.2775	**6.1290**	<u>23.2825</u>	<u>0.3265</u>	**13.9253**	0.2042	0.2965
R_{dm}	0.7236	0.2720	5.7537	22.0748	0.2759	11.2115	0.1619	0.3759
DMD + DiffusionNFT	0.7158	0.2843	5.3784	21.9571	0.2708	9.6985	0.2249	0.4865
CDM + DiffusionNFT	0.1455	0.2745	4.8335	21.3849	0.2143	-3.2834	0.2103	0.3303
AnyFlow + DiffusionNFT	<u>1.2394</u>	<u>0.2915</u>	<u>5.9489</u>	23.0876	0.2919	12.1378	<u>0.2335</u>	<u>0.5948</u>
MeanFlowNFT (Ours)	**1.4504**	**0.2967**	5.9275	**23.5019**	**0.3269**	<u>13.8826</u>	**0.2375**	**0.6534**

Image generation. Tab. 1 reports the quantitative comparison on `SD3.5-M`. Among all few-step models, MeanFlowNFT attains the best results on 6 of the 8 metrics. It clearly outperforms the few-step distillation baselines DMD (Yin et al., 2024b), CDM (Liu et al., 2026c), and AnyFlow (Gu et al., 2026), and also beats the recent few-step RL methods R_{dm} (Fan et al., 2026) and RTDMD (Huang et al., 2026) on most metrics (*e.g.*, OCR 0.65 *vs.* 0.30 against RTDMD). Remarkably, with only 4 sampling steps MeanFlowNFT already matches or exceeds the 40-step forward-process RL method DiffusionNFT (Zheng et al., 2026a) on several reward metrics (ImageReward 1.45 *vs.* 1.41, CLIPScore 0.297 *vs.* 0.289, *etc.*) at 10× fewer function evaluations. Qualitatively (Figs. 13 to 15), MeanFlowNFT produces more faithful samples with fewer reward-hacking artifacts, such as oversaturated colors and implausible object scales, than DiffusionNFT and RTDMD.

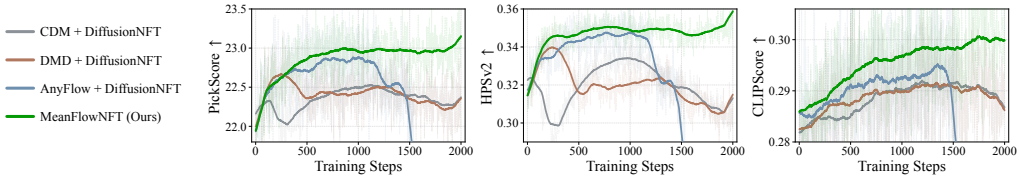


Figure 4: Training reward curves of MeanFlowNFT compared with the baselines on `SD3.5-M`.

Additionally, we apply DiffusionNFT directly to few-step generators. AnyFlow trains an average-velocity network, while DMD and CDM are trained by distribution matching (Yin et al., 2024b) rather than flow matching. In each case the network does not predict the instantaneous velocity as a posterior mean, so the linearity behind DiffusionNFT's improvement guarantee is absent and no policy improvement can be derived (App. C). As a result, AnyFlow+DiffusionNFT[4] and DMD/CDM+DiffusionNFT fall far short of MeanFlowNFT and train very unstably, collapsing early (Fig. 4). For instance, CDM+DiffusionNFT diverges within 400 steps.

Table 2: Main results on `Wan2.1` 1.3B for video generation. We report `VBench` scores and four metrics on 256 held-out prompts (`HPSv3-G`/`HPSv3-P`: HPSv3 general/percentile; `MQ`/`TA`: VideoAlign motion-quality/text-alignment). Among few-step models, **bold** and underline denote the best and second-best results. †denotes results reproduced using open-source checkpoints. All other results were reproduced by us. Full `VBench` per-dimension results can be found in App. H.

Method	VBench			256 held-out prompts			
	Total↑	Quality↑	Semantic↑	HPSv3-G↑	HPSv3-P↑	MQ↑	TA↑
Multi-step models (50 steps)							
Wan2.1 1.3B (w/ CFG)†	82.94	84.69	75.97	3.9099	8.2868	0.8684	1.2255
+ LongCat-Video RL†	82.57	84.44	75.10	4.7099	9.2730	0.5493	1.6321
Few-step models (4 steps)							
rCM†	82.43	84.58	73.82	3.9660	8.7198	0.2740	1.6290
DMD	82.64	84.65	74.57	3.8598	8.7955	0.1810	1.6845
SC-DMD	83.36	84.76	**77.77**	–	–	–	–
AnyFlow†	<u>83.71</u>	<u>85.36</u>	77.11	<u>6.0536</u>	<u>10.450</u>	<u>0.7504</u>	**1.7356**
MeanFlowNFT (Ours)	**84.33**	**85.99**	<u>77.68</u>	**6.5959**	**10.793**	**0.9535**	<u>1.7235</u>

Video generation. Tab. 2 further evaluates MeanFlowNFT on video generation with `Wan2.1` 1.3B. Since there is currently no open-source few-step video-generation RL baseline, we compare with few-step distillation methods (rCM (Zheng et al., 2026b), DMD (Yin et al., 2024b), SC-DMD (Ge et al., 2026)[5], AnyFlow (Gu et al., 2026)) and the 50-step flow-matching RL baseline `LongCat-Video` RL (Team et al., 2025)[6]. With only 4 sampling steps, MeanFlowNFT obtains the best few-step results on 5 of the 7 reported metrics and improves over AnyFlow on VBench total, HPSv3, and motion quality. It also outperforms `LongCat-Video` RL on all reported metrics with far fewer function evaluations, confirming its applicability to video MeanFlow models.

Test-time scaling. Since $\boldsymbol{u}_\theta$ approximates the average velocity over any interval $[s, t]$, MeanFlowNFT supports any-step inference via the sampler of Eq. (4), sweeping the number of steps $N \in \{2, 4, 8, 16, 32\}$. MeanFlowNFT exhibits the same test-time scaling trend as AnyFlow (Gu

[4] AnyFlow+DiffusionNFT plugs the average velocities $\boldsymbol{u}^{\mathrm{old}}$ and $\boldsymbol{u}_\theta$ into Eq. (7) in place of the instantaneous velocities $\boldsymbol{v}^{\mathrm{old}}$ and $\boldsymbol{v}_\theta$, and optimizes it exactly as DiffusionNFT does.

[5] We use the numbers reported in the SC-DMD paper.

[6] https://huggingface.co/lightx2v/Wan2.1-T2V-1.3B-longcat-step1500

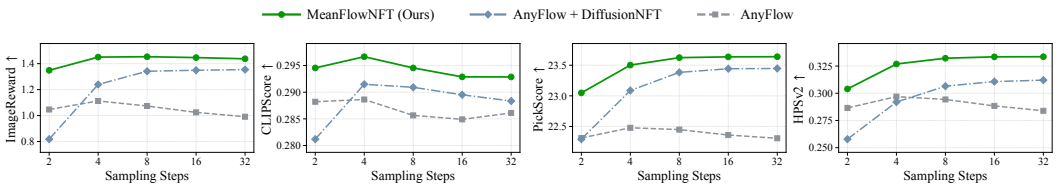


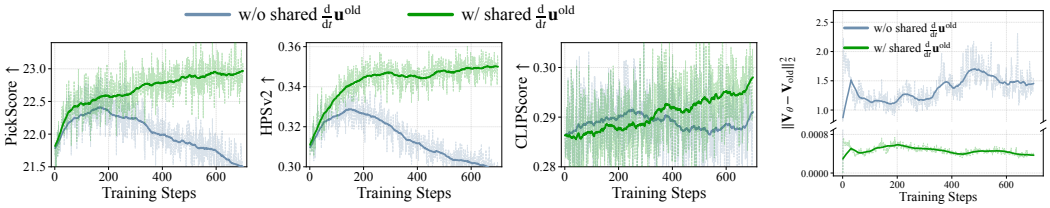
Figure 5: Quantitative results of test-time scaling for MeanFlow-based methods on `SD3.5-M`. More results for `Wan2.1` 1.3B and `SD3.5-M` can be found in App. G.

et al., 2026): most metrics improve as N increases (Fig. 5). More notably, MeanFlowNFT is noticeably more step-consistent than AnyFlow, with samples that vary far less across N in both layout and content (Figs. 11 and 12 in App. G). As a more accurate average-velocity estimation yields more N-invariant sampling, this consistency indicates that our RL improves AnyFlow.

4.3 Analysis

We analyze several design choices in MeanFlowNFT on `SD3.5-M`. A further ablation on training only with $s{=}t$ pairs is deferred to App. F.

Effect of shared $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}^{\mathrm{old}}$. We examine whether the trainable and reference predictors should share a single derivative term $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}^{\mathrm{old}}$ or form one each, and find sharing essential for stability. With a shared derivative, reward rises smoothly and monotonically across `PickScore`, `HPSv2`, and `CLIPScore`, whereas forming the two derivatives independently lets reward rise briefly before collapsing (Fig. 6a). The induced deviation $\|\boldsymbol{V}_\theta - \boldsymbol{V}^{\mathrm{old}}\|_2^2$ exposes the mechanism, which grows by orders of magnitude without sharing (Fig. 6b). The gap between the two predictors is then governed by the uncontrolled derivative-difference term rather than the EMA-bounded $\boldsymbol{u}_\theta - \boldsymbol{u}^{\mathrm{old}}$ signal (Sec. 3.3). This reference drift is what destabilizes training, and sharing eliminates it exactly.



(a) Training reward with and without shared $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}^{\mathrm{old}}$. **(b)** Velocity deviation.

Figure 6: Effect of sharing the derivative term $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}^{\mathrm{old}}$ between $\boldsymbol{V}^{\mathrm{old}}$ and $\boldsymbol{V}_\theta$ on `SD3.5-M`.

Direction for $\boldsymbol{x}_{t\pm\Delta t}$. We compare the two directions that can drive $\boldsymbol{x}_{t\pm\Delta t}$ in the finite-difference derivative (Sec. 3.3): $\widehat{\boldsymbol{v}}_\theta = \boldsymbol{u}_\theta(\boldsymbol{x}_t, t, t)$ (Eq. (8)), and the model-free conditional velocity $\boldsymbol{v}_t = \dot{\alpha}_t \boldsymbol{x}_0 + \dot{\sigma}_t \boldsymbol{\epsilon}$ (Algorithm 1). Using $\boldsymbol{v}_t$ improves reward steadily across all three metrics, whereas anchoring the direction to the trainable $\widehat{\boldsymbol{v}}_\theta$ peaks early and then degrades (Fig. 7). As $\boldsymbol{v}_t$ is model-free and saves an additional network evaluation, we adopt it by default.

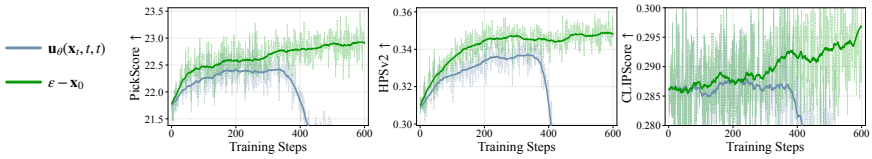


Figure 7: Training reward curves for MeanFlowNFT with different choices of direction for $\boldsymbol{x}_{t\pm\Delta t}$.

5 Conclusion

In this work, we presented MeanFlowNFT, the first forward-process RL method tailored for MeanFlow models. MeanFlow predicts an average velocity, but the DiffusionNFT framework we build on works with the instantaneous velocity. To bridge this gap, we use the MeanFlow identity to optimize the reward on the instantaneous velocity while keeping the average-velocity network for fast sampling. We prove that, in an idealized setting, this reaches DiffusionNFT's improved policy and carries the gain to the average-velocity model. Experiments show that MeanFlowNFT consistently improves MeanFlow baselines and can even surpass multi-step RL with only a few sampling steps. We discuss limitations and future work in App. A.

AI USE STATEMENT

We acknowledge the use of large language models (LLMs), such as OpenAI's `GPT-5.6`, as a writing-assistance tool in this work. Their role was strictly limited to proofreading and rephrasing sentences to enhance linguistic quality, without any contribution to the research ideation or experimental results.

REFERENCES

Andreas Bergmeister, Stefanie Jegelka, Nikolas Nüsken, Carles Domingo-Enrich, and Jakiw Pidstrigach. Reinforce adjoint matching: Scaling rl post-training of diffusion and flow-matching models, 2026. URL https://arxiv.org/abs/2605.10759. 15, 16

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=YCWjhGrJFD. 2, 15

Guanjie Chen, Shirui Huang, Yifu Sun, Kai Liu, Jianchen Zhu, Xiaoye Qu, Yu Cheng, and Peng Chen. Flash-dmd: Towards high-fidelity few-step image generation with efficient distillation and joint reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6010–6020, 2026. 16

Linwei Dong, Ruoyu Guo, Ge Bai, Zehuan Yuan, Yawei Luo, and Changqing Zou. Guiding distribution matching distillation with gradient-based reinforcement learning, 2026. URL https://arxiv.org/abs/2604.19009. 16

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. 7, 15

Linqian Fan, Peiqin Sun, Tiancheng Wen, Shun Lu, and Chengru Song. r_{dm}: Re-conceptualizing distribution matching as a reward for diffusion distillation, 2026. URL https://arxiv.org/abs/2603.28460. 8, 16

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. 2, 15

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OlzB6LnXcS. 15

Xingtong Ge, Yi Zhang, Yushi Huang, Dailan He, Xiahong Wang, Bingqi Ma, Guanglu Song, Yu Liu, and Jun Zhang. Salt: Self-consistent distribution matching with cache-aware training for fast video generation. In *European Conference on Computer Vision*. Springer, 2026. 8

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2026a. 2, 3, 15

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30467–30476, 2026b. 15

Yuchao Gu, Guian Fang, Yuxin Jiang, Weijia Mao, Song Han, Han Cai, and Mike Zheng Shou. Anyflow: Any-step video diffusion model with on-policy flow map distillation, 2026. URL https://arxiv.org/abs/2605.13724. 6, 7, 8, 15, 16, 19

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. 2, 16

Xiaoxuan He, Siming Fu, Yuke Zhao, Wanli Li, Jian Yang, Dacheng Yin, Fengyun Rao, and Bo Zhang. TEMPFLOW-GRPO: WHEN TIMING MATTERS FOR GRPO IN FLOW MODELS. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7mCo3R3Wyn. 16

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. 7

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI. 4, 7

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 15

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 7

Yushi Huang, Xiangxin Zhou, Ruoyu Wang, Chi Zhang, Jun Zhang, and Tianyu Pang. Reinforcing few-step generators via reward-tilted distribution matching, 2026. URL https://arxiv.org/abs/2605.26108. 8, 16

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024. 7

Dengyang Jiang, Dongyang Liu, Zanyi Wang, Qilong Wu, Liuzhuozheng Li, Hengzhuang Li, Xin Jin, David Liu, Zhen Li, Bo Zhang, et al. Distribution matching distillation meets reinforcement learning. In *European Conference on Computer Vision*. Springer, 2026. 16

Amita Kamath, Kai-Wei Chang, Ranjay Krishna, Luke Zettlemoyer, Yushi Hu, and Marjan Ghazvininejad. Geneval 2: Addressing benchmark drift in text-to-image evaluation, 2025. URL https://arxiv.org/abs/2512.16853. 7

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ymjI8feDTD. 15

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. 7

Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024. 15

Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023. 15

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018. 3

Junzhe Li, Yutao Cui, Tao Huang, Yinping Ma, Chun Fan, Miles Yang, and Zhao Zhong. Mixgrpo: Unlocking flow-based grpo efficiency with mixed ode-sde. In *European Conference on Computer Vision*. Springer, 2026a. 5, 16

Zhiqi Li, Wen Zhang, and Bo Zhu. Flow-map grpo: Reinforcement learning for few-step flow-map generators via anchored stochastic composition, 2026b. URL https://arxiv.org/abs/2607.00535. 16

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t. 1, 2, 3, 15

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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026a. URL https://openreview.net/forum?id=oCBKGw5HNf. 2, 5, 7, 16

Jie Liu, Gongye Liu, Jiajun Liang, Ziyang Yuan, Xiaokun Liu, Mingwu Zheng, Xiele Wu, Qiulin Wang, Menghan Xia, Xintao Wang, et al. Improving video generation with human feedback. *Advances in Neural Information Processing Systems*, 38:82155–82192, 2026b. 7

Tao Liu, Hao Yan, Mengting Chen, Taihang Hu, Zhengrong Yue, Zihao Pan, Jinsong Lan, Xiaoyong Zhu, Ming-Ming Cheng, Bo Zheng, and Yaxing Wang. Continuous-time distribution matching for few-step diffusion distillation. 2026c. URL https://arxiv.org/abs/2605.06376. 8, 17

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z. 1, 2, 3, 4, 15

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7. 7

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023a. 15

Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36:76525–76546, 2023b. 15

Yihong Luo, Tianyang Hu, Weijian Luo, and Jing Tang. TDM-r1: Reinforcing few-step diffusion models with non-differentiable reward. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=4w9DpowGcs. 16

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024. 15

Yuhang Ma, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. Hpsv3: Towards wide-spectrum human preference score. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15086–15095, 2025. 7

Zichen Miao, Zhengyuan Yang, Kevin Lin, Ze Wang, Zicheng Liu, Lijuan Wang, and Qiang Qiu. Tuning timestep-distilled diffusion model using pairwise sample optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=fXnE4gB64o. 16

Weili Nie, Julius Berner, Nanye Ma, Chao Liu, Saining Xie, and Arash Vahdat. Transition matching distillation for fast video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4645–4655, June 2026. 6, 15

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023. 15

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=di52zR8xgf. 2, 15

Qi Qin, Le Zhuo, Yi Xin, Ruoyi Du, Zhen Li, Bin Fu, Yiting Lu, Xinyue Li, Dongyang Liu, Xiangyang Zhu, Will Beddow, Erwann Millon, Victor Perez, Wenhai Wang, Yu Qiao, Bo Zhang, Xiaohong Liu, Hongsheng Li, Chang Xu, and Peng Gao. Lumina-image 2.0: A unified and efficient image generative framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20031–20042, October 2025. 2, 15

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022. 15

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=08Yk-n5l2Al. 7

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI. 15

Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024a. 15

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pp. 87–103. Springer, 2024b. 15

Christoph Schuhmann. Laion-aesthetics. https://laion.ai/blog/laion-aesthetics/, 2022. 7

Team Seedream, Yunpeng Chen, Yu Gao, Lixue Gong, Meng Guo, Qiushan Guo, Zhiyao Guo, Xiaoxia Hou, Weilin Huang, Yixuan Huang, et al. Seedream 4.0: Toward next-generation multimodal image generation. *arXiv preprint arXiv:2509.20427*, 2025. 2, 15

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP. 1, 15

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS. 1, 15

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252. PMLR, 2023. 15

Meituan LongCat Team, Xunliang Cai, Qilong Huang, Zhuoliang Kang, Hongyu Li, Shijun Liang, Liya Ma, Siyu Ren, Xiaoming Wei, Rixu Xie, and Tong Zhang. Longcat-video technical report, 2025. URL https://arxiv.org/abs/2510.22200. 7, 8, 15, 19

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8228–8238, 2024. 2, 16

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan

Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314. 7, 15

Ruoyu Wang, Boye Niu, Xiangxin Zhou, Yushi Huang, Tongliang Liu, and Chi Zhang. Exploring the design space of reward backpropagation for flow matching. *arXiv preprint arXiv:2606.11075*, 2026. 16

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023. 7

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023. 7, 16

Shuchen Xue, Chongjian GE, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning RL with pretraining in diffusion models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=nLY2pOYBrJ. 15, 16

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. 2, 7, 16

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and Bill Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024a. 15

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6613–6623, 2024b. 8, 15, 17

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12):3848–3860, August 2023. ISSN 2150-8097. doi: 10.14778/3611540.3611569. URL https://doi.org/10.14778/3611540.3611569. 6

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. DiffusionNFT: Online diffusion reinforcement with forward process. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=VJZ477R89F. 2, 3, 4, 7, 8, 16, 19

Kaiwen Zheng, Yuji Wang, Qianli Ma, Huayu Chen, Jintao Zhang, Yogesh Balaji, Jianfei Chen, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Large scale diffusion distillation via score-regularized continuous-time consistency. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=2uNlM353RI. 8

Appendix

CONTENTS

A Limitations and Future Work **15**
B Related Work **15**
C Why Direct Plug-Ins Lack DiffusionNFT's Guarantee **16**
D Proofs for MeanFlowNFT **17**
E More Implementation Details **19**
F Training Only on $s{=}t$ Pairs **19**
G Additional Test-time Scaling Results **20**
H VBench Full Results **20**
I More Qualitative Results **20**

A LIMITATIONS AND FUTURE WORK

In terms of limitations, we only explore DiffusionNFT-style forward-process RL on MeanFlow. We do not study other forward-process objectives, such as RAM (Bergmeister et al., 2026) and AWM (Xue et al., 2026). Since these methods also act on the instantaneous velocity and rely on a frozen reference, we expect our induced instantaneous-velocity construction and practical implementation to carry over to them. Additionally, we consider only MeanFlow, one instance of the broader family of flow-map models. These models learn a direct map between two time points, so one or a few steps replace iterative sampling. MeanFlow realizes this by predicting the average velocity over an interval, while other instances include shortcut models (Frans et al., 2025), consistency trajectory models (Kim et al., 2024), *etc*. Because the induced-predictor construction in Eq. (8) is not specific to MeanFlow, we believe our approach can also extend to them. In short, both directions lie outside the scope of this paper, and we leave them to future work.

B RELATED WORK

Diffusion and flow models. Diffusion (Ho et al., 2020; Song et al., 2021b;a) and flow models (Liu et al., 2023; Lipman et al., 2023) have become the dominant paradigm for high-quality image and video generation. Diffusion models synthesize data by learning to reverse a fixed noising process, and latent diffusion (Rombach et al., 2022) runs this process in a compressed latent space for efficiency. Transformer backbones improve scalability (Peebles & Xie, 2023; Ma et al., 2024), and scaling rectified-flow transformers enables high-resolution synthesis (Esser et al., 2024). Flow matching (Lipman et al., 2023) and rectified flow (Liu et al., 2023) recast generation as regressing a velocity field that transports noise to data. Building on these formulations, large text-to-image (Podell et al., 2024; Labs, 2024; Qin et al., 2025; Seedream et al., 2025) and text-to-video (Wan et al., 2025; Team et al., 2025) systems reach strong visual quality.

Few-step generation. Because iterative sampling is slow, many methods distill pretrained diffusion or flow models into few-step generators, including progressive distillation (Salimans & Ho, 2022), consistency models (Song et al., 2023; Luo et al., 2023a), distribution matching distillation (Yin et al., 2024b;a; Luo et al., 2023b), and adversarial distillation (Sauer et al., 2024b;a). A closely related line learns *flow maps*, long-range transport operators that move samples directly between two time points instead of integrating an instantaneous velocity (Frans et al., 2025; Kim et al., 2024). In particular, MeanFlow (Geng et al., 2026a;b) parameterizes the average velocity over a time interval, so one evaluation advances a sample across it and enables one or few-step generation. AnyFlow (Gu et al., 2026) and transition matching (Nie et al., 2026) distill such flow maps. Our work builds on this average-velocity parameterization.

Reinforcement learning for diffusion models. Reinforcement learning is now a standard tool for aligning diffusion and flow models with human preferences and task rewards. Existing approaches rely on policy gradients (Black et al., 2024; Fan et al., 2023), reward-weighted training (Lee et al.,

2023), preference optimization (Wallace et al., 2024), or reward backpropagation (Xu et al., 2023; Wang et al., 2026). A dominant recent recipe discretizes the reverse sampling process into a Markov decision process and applies GRPO-style policy gradients (Guo et al., 2025; Liu et al., 2026a; Xue et al., 2025; Li et al., 2026a; He et al., 2026), which need a stochastic policy and per-step likelihood estimation. A complementary line runs reinforcement learning on the forward process, recasting reward optimization as a regression objective on analytically noised samples and thereby avoiding reverse rollouts and likelihoods. It includes advantage weighted matching (Xue et al., 2026), DiffusionNFT (Zheng et al., 2026a), and reinforce adjoint matching (Bergmeister et al., 2026). MeanFlowNFT follows this forward-process line.

Reinforcement learning for few-step generation. A growing line of work brings reinforcement learning to few-step generators. Most methods reinforce distribution-matching distillation, either by combining a reward objective with the matching loss (Jiang et al., 2026; Chen et al., 2026; Fan et al., 2026; Dong et al., 2026) or by reward-tilting the target distribution (Huang et al., 2026). Others post-train distilled few-step generators with surrogate reward learning (Luo et al., 2026) or pairwise sample objectives (Miao et al., 2025). Overall, reinforcement learning for flow-map generators remains underexplored. We study it through forward-process RL, bringing the DiffusionNFT (Zheng et al., 2026a) paradigm to MeanFlow. The concurrent Flow-Map GRPO (Li et al., 2026b) introduces path-preserving stochastic flow-map transitions during RL training and applies GRPO with stochastic rollouts and per-step likelihood ratios. MeanFlowNFT instead uses analytically noised forward-process samples and a likelihood-free regression objective.

C Why Direct Plug-Ins Lack DiffusionNFT's Guarantee

This section explains why the direct DiffusionNFT plug-ins for AnyFlow, DMD, and CDM in Sec. 4.2 do not inherit the policy improvement guarantee of DiffusionNFT. The issue is the meaning of the network output rather than the number of reverse sampling steps.

Output required by DiffusionNFT. The idealized DiffusionNFT analysis assumes that the optimized output is the marginal instantaneous velocity

$$\boldsymbol{v}^q(\boldsymbol{x}_t, t) = \mathbb{E}_{\pi^q}[\boldsymbol{v}_t \mid \boldsymbol{x}_t, \boldsymbol{c}, t], \qquad \boldsymbol{v}_t = \dot{\alpha}_t \boldsymbol{x}_0 + \dot{\sigma}_t \boldsymbol{\epsilon}, \quad q \in \{\text{old}, +, -\}.$$

The map from $(\boldsymbol{x}_0, \boldsymbol{\epsilon})$ to $\boldsymbol{v}_t$ is identical for all three policies. Reward reweighting changes the posterior distribution of $(\boldsymbol{x}_0, \boldsymbol{\epsilon})$ but leaves this map unchanged. At fixed $(\boldsymbol{x}_t, \boldsymbol{c}, t)$, the definition $\pi^+ \propto r\pi^{\text{old}}$ gives $\mathbb{E}_{\pi^{\text{old}}}[r\boldsymbol{v}_t] = \alpha \boldsymbol{v}^+$. The definition $\pi^- \propto (1-r)\pi^{\text{old}}$ similarly gives $\mathbb{E}_{\pi^{\text{old}}}[(1-r)\boldsymbol{v}_t] = (1-\alpha)\boldsymbol{v}^-$. All expectations here are conditioned on the fixed tuple, and we write α for $\alpha(\boldsymbol{x}_t, \boldsymbol{c})$ in Eq. (5). Adding these identities and using $r + (1-r) = 1$ gives $\boldsymbol{v}^{\text{old}} = \alpha \boldsymbol{v}^+ + (1-\alpha)\boldsymbol{v}^-$. This decomposition yields the guidance Δ in Eq. (6) (Zheng et al., 2026a, Thm. 3.1). DiffusionNFT then minimizes Eq. (7) over an instantaneous velocity predictor. Its pointwise optimum is $\boldsymbol{v}_{\theta^*} = \boldsymbol{v}^{\text{old}} + \frac{2}{\beta}\Delta$ (Zheng et al., 2026a, Thm. 3.2). At points where $\alpha > 0$, substituting $\beta = 2\alpha$ and $\Delta = \alpha(\boldsymbol{v}^+ - \boldsymbol{v}^{\text{old}})$ gives $\boldsymbol{v}_{\theta^*} = \boldsymbol{v}^+$. The guarantee therefore uses both the posterior mean identity above and the fact that the optimized output is an instantaneous velocity. Reusing the same loss for another output quantity preserves the algebraic form but not this policy interpretation.

Direct substitution of AnyFlow outputs. AnyFlow (Gu et al., 2026) is a MeanFlow network that predicts the interval average velocity $\boldsymbol{u}$ in Eq. (2), rather than the marginal instantaneous velocity $\boldsymbol{v}$ in Eq. (1). The AnyFlow+DiffusionNFT baseline replaces $\boldsymbol{v}_\theta$ and $\boldsymbol{v}^{\text{old}}$ in Eq. (7) with $\boldsymbol{u}_\theta$ and $\boldsymbol{u}^{\text{old}}$. It keeps the regression target $\boldsymbol{v}_t$. Fix $(\boldsymbol{x}_t, \boldsymbol{c}, s, t)$ and regard $\boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t)$ as a free output value. The resulting conditional objective is

$$\begin{aligned}\ell\big(\boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t)\big) = \mathbb{E}_{\pi^{\text{old}}}\Big[& r\big\|\boldsymbol{u}^{\text{old}} + \beta(\boldsymbol{u}_\theta - \boldsymbol{u}^{\text{old}}) - \boldsymbol{v}_t\big\|_2^2 \\ & + (1-r)\big\|\boldsymbol{u}^{\text{old}} - \beta(\boldsymbol{u}_\theta - \boldsymbol{u}^{\text{old}}) - \boldsymbol{v}_t\big\|_2^2 \mid \boldsymbol{x}_t, \boldsymbol{c}, s, t\Big],\end{aligned}$$

where every average velocity in the display is evaluated at $(\boldsymbol{x}_t, s, t)$. The interval sampler makes s independent of $(\boldsymbol{x}_0, \boldsymbol{\epsilon}, r)$ given $(\boldsymbol{c}, t)$. Conditioning on s therefore leaves the required posterior moments unchanged.

Differentiating the conditional objective with respect to its output gives

$$\frac{1}{2\beta}\nabla_{\boldsymbol{u}_\theta}\ell = \beta\big(\boldsymbol{u}_\theta(\boldsymbol{x}_t, s, t) - \boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t, s, t)\big) - \mathbb{E}_{\pi^{\text{old}}}\left[(2r-1)\big(\boldsymbol{v}_t - \boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t, s, t)\big) \mid \boldsymbol{x}_t, \boldsymbol{c}, s, t\right].$$

For $\beta > 0$, the objective is strictly convex in this output value. Its unique minimizer sets the displayed gradient to zero. The conditional residual can be written as

$$\begin{aligned}
&\mathbb{E}_{\pi^{\text{old}}}\big[(2r-1)\big(\boldsymbol{v}_t - \boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t,s,t)\big) \mid \boldsymbol{x}_t,\boldsymbol{c},s,t\big] \\
&= 2\mathbb{E}_{\pi^{\text{old}}}[r\boldsymbol{v}_t \mid \boldsymbol{x}_t,\boldsymbol{c},t] - \mathbb{E}_{\pi^{\text{old}}}[\boldsymbol{v}_t \mid \boldsymbol{x}_t,\boldsymbol{c},t] - \big(2\alpha-1\big)\boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t,s,t) \\
&= 2\alpha\boldsymbol{v}^{+}(\boldsymbol{x}_t,t) - \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t,t) - \big(2\alpha-1\big)\boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t,s,t) \\
&= 2\Delta(\boldsymbol{x}_t,\boldsymbol{c},t) + \big(2\alpha-1\big)\big(\boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t,t) - \boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t,s,t)\big).
\end{aligned}$$

Denoting the unique minimizer by $\boldsymbol{u}^{\dagger}(\boldsymbol{x}_t,s,t)$ and substituting the residual into the zero gradient condition gives

$$\begin{aligned}
\boldsymbol{u}^{\dagger}(\boldsymbol{x}_t,s,t) &= \boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t,s,t) + \frac{2}{\beta}\Delta(\boldsymbol{x}_t,\boldsymbol{c},t) + \frac{2\alpha-1}{\beta}\big(\boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t,t) - \boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t,s,t)\big) \\
&\overset{\beta=2\alpha}{=} \boldsymbol{v}^{+}(\boldsymbol{x}_t,t) + \frac{\boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t,s,t) - \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t,t)}{2\alpha}, \qquad \alpha > 0.
\end{aligned} \tag{15}$$

The last equality uses $\Delta = \alpha(\boldsymbol{v}^{+} - \boldsymbol{v}^{\text{old}})$ from Eq. (6) and collects the terms involving $\boldsymbol{u}^{\text{old}}$ and $\boldsymbol{v}^{\text{old}}$. Let $\boldsymbol{u}^{+}(\boldsymbol{x}_t,s,t)$ denote the exact average velocity induced by $\boldsymbol{v}^{+}$ over $[s,t]$. This is the desired output, whereas Eq. (15) gives

$$\boldsymbol{u}^{\dagger}(\boldsymbol{x}_t,s,t) - \boldsymbol{u}^{+}(\boldsymbol{x}_t,s,t) = \boldsymbol{v}^{+}(\boldsymbol{x}_t,t) - \boldsymbol{u}^{+}(\boldsymbol{x}_t,s,t) + \frac{\boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t,s,t) - \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t,t)}{2\alpha}.$$

Both terms vanish when s approaches t. For a finite interval, the DiffusionNFT identities do not force them to cancel. The direct objective therefore does not guarantee $\boldsymbol{u}^{\dagger}(\boldsymbol{x}_t,s,t) = \boldsymbol{u}^{+}(\boldsymbol{x}_t,s,t)$.

Direct substitution of DMD and CDM outputs. DMD (Yin et al., 2024b) and CDM (Liu et al., 2026c) are trained with distribution matching. Their distribution matching terms compare the generated and target marginal distributions after averaging over latent noise. Matching these distributions does not identify a unique generator map at each fixed $(\boldsymbol{x}_t,\boldsymbol{c},t)$. Both methods include additional regression or alignment terms, but these terms do not regress the network output against the forward conditional target $\boldsymbol{v}_t$. Consequently, even when the output is parameterized as a velocity, their objectives do not establish that the network output equals the posterior mean $\mathbb{E}[\boldsymbol{v}_t \mid \boldsymbol{x}_t,\boldsymbol{c},t]$ required by DiffusionNFT. The posterior decomposition therefore does not apply to the outputs inserted into Eq. (7). As a result, DiffusionNFT's regression analysis cannot identify the optimum of the substituted objective with $\boldsymbol{v}^{+}$. Its Theorems 3.1 and 3.2 do not establish that DMD+DiffusionNFT or CDM+DiffusionNFT realizes π^{+}.

D Proofs for MeanFlowNFT

This appendix restates and proves each result of Sec. 3.2.

Proposition 3.1 (Idealized pointwise optimum). *Conditioned on $(\boldsymbol{x}_t,\boldsymbol{c},s,t)$, the idealized pointwise minimizer is*

$$\begin{aligned}
\boldsymbol{V}_{\theta^*}(\boldsymbol{x}_t,s,t) &= \boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t,s,t) + \frac{2}{\beta}\,\widehat{\Delta}(\boldsymbol{x}_t,\boldsymbol{c},s,t), \\
\widehat{\Delta}(\boldsymbol{x}_t,\boldsymbol{c},s,t) &\triangleq \frac{1}{2}\,\mathbb{E}_{\pi^{\text{old}}}\big[(2r-1)\big(\boldsymbol{v}_t - \boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t,s,t)\big) \,\big|\, \boldsymbol{x}_t,\boldsymbol{c},s,t\big].
\end{aligned}$$

Proof. Fix $(\boldsymbol{x}_t,\boldsymbol{c},s,t)$. Under Eq. (8), the induced predictor $\boldsymbol{V}_\theta(\boldsymbol{x}_t,s,t)$ is a deterministic function of the conditioned variables for fixed θ. Its value is therefore shared by all posterior draws $(\boldsymbol{x}_0,\boldsymbol{\epsilon})$ consistent with the same $\boldsymbol{x}_t$, and the idealized pointwise optimum can be obtained by minimizing $\mathbb{E}_{\pi^{\text{old}}}[\ell \mid \boldsymbol{x}_t,\boldsymbol{c},s,t]$ over the value $\boldsymbol{V}_\theta(\boldsymbol{x}_t,s,t)$. By the definitions of $\boldsymbol{V}_\theta^{\pm}$, the per-sample integrand of Eq. (9) is

$$\begin{aligned}
\ell\big(\boldsymbol{V}_\theta(\boldsymbol{x}_t,s,t)\big) = r\big\|(1-\beta)\boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t,s,t) + \beta\boldsymbol{V}_\theta(\boldsymbol{x}_t,s,t) - \boldsymbol{v}_t\big\|_2^2 \\
+ (1-r)\big\|(1+\beta)\boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t,s,t) - \beta\boldsymbol{V}_\theta(\boldsymbol{x}_t,s,t) - \boldsymbol{v}_t\big\|_2^2.
\end{aligned}$$

Dividing the gradient by 2β and collecting terms gives

$$\tfrac{1}{2\beta}\,\nabla_{\boldsymbol{V}_\theta}\ell = \beta\big(\boldsymbol{V}_\theta(\boldsymbol{x}_t,s,t) - \boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t,s,t)\big) + (2r-1)\big(\boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t,s,t) - \boldsymbol{v}_t\big).$$

Since this is a strictly convex quadratic in $\boldsymbol{V}_\theta(\boldsymbol{x}_t, s, t)$ for $\beta > 0$, the stationarity condition $\mathbb{E}_{\pi^{\text{old}}}[\nabla_{\boldsymbol{V}_\theta}\ell \mid \boldsymbol{x}_t, \boldsymbol{c}, s, t] = 0$ yields

$$\begin{aligned}\beta\big(\boldsymbol{V}_{\theta^*}(\boldsymbol{x}_t, s, t) - \boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t, s, t)\big) \\ = \mathbb{E}_{\pi^{\text{old}}}\big[(2r-1)\big(\boldsymbol{v}_t - \boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t, s, t)\big) \mid \boldsymbol{x}_t, \boldsymbol{c}, s, t\big] \\ = 2\widehat{\Delta}(\boldsymbol{x}_t, \boldsymbol{c}, s, t),\end{aligned}$$

which is exactly Eq. (10). □

Corollary 3.2 ($\boldsymbol{V}_{\theta^*}$ recovers the improved marginal velocity). *Setting the guidance strength to $\beta = 2\alpha(\boldsymbol{x}_t, \boldsymbol{c})$, Eq. (10) collapses to*

$$\boldsymbol{V}_{\theta^*}(\boldsymbol{x}_t, s, t) = \boldsymbol{v}^+(\boldsymbol{x}_t, t)$$

for all $s \le t$, the marginal instantaneous velocity of the improved policy π^+.

Proof. Since $\boldsymbol{u}^{\text{old}}$ is the MeanFlow average velocity of π^{old}, its MeanFlow-induced instantaneous velocity is

$$\widehat{\boldsymbol{v}}^{\text{old}}(\boldsymbol{x}_t, t) = \boldsymbol{u}^{\text{old}}(\boldsymbol{x}_t, t, t) = \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t).$$

Applying the MeanFlow identity to this exact reference therefore gives $\boldsymbol{V}^{\text{old}}(\boldsymbol{x}_t, s, t) = \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t)$ for all $s \le t$. Thus $\boldsymbol{V}^{\text{old}}$ is a deterministic function of $(\boldsymbol{x}_t, \boldsymbol{c}, t)$ and may be pulled out of the conditional expectation. Then

$$\begin{aligned}\widehat{\Delta} &= \frac{1}{2}\mathbb{E}_{\pi^{\text{old}}}\big[(2r-1)\big(\boldsymbol{v}_t - \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t)\big) \mid \boldsymbol{x}_t, \boldsymbol{c}, s, t\big] \\ &= \mathbb{E}_{\pi^{\text{old}}}[r\,\boldsymbol{v}_t \mid \boldsymbol{x}_t, \boldsymbol{c}] - \tfrac{1}{2}\mathbb{E}_{\pi^{\text{old}}}[\boldsymbol{v}_t \mid \boldsymbol{x}_t, \boldsymbol{c}] - \tfrac{1}{2}\mathbb{E}_{\pi^{\text{old}}}[2r-1 \mid \boldsymbol{x}_t, \boldsymbol{c}]\,\boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t).\end{aligned}$$

Writing $\alpha = \mathbb{E}_{\pi^{\text{old}}}[r \mid \boldsymbol{x}_t, \boldsymbol{c}]$ and using the posterior-mean identities $\mathbb{E}_{\pi^{\text{old}}}[\boldsymbol{v}_t \mid \boldsymbol{x}_t, \boldsymbol{c}] = \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t)$ (Eq. (1)) and $\mathbb{E}_{\pi^{\text{old}}}[r\,\boldsymbol{v}_t \mid \boldsymbol{x}_t, \boldsymbol{c}] = \alpha\boldsymbol{v}^+(\boldsymbol{x}_t, t)$, the last display becomes

$$\widehat{\Delta} = \alpha\boldsymbol{v}^+(\boldsymbol{x}_t, t) - \tfrac{1}{2}\boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t) - \tfrac{1}{2}(2\alpha - 1)\boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t) = \alpha\big(\boldsymbol{v}^+(\boldsymbol{x}_t, t) - \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t)\big) = \Delta,$$

the DiffusionNFT reinforcement guidance in Eq. (6). Substituting this into Eq. (10) and using $\beta = 2\alpha$ gives, for all $s \le t$,

$$\boldsymbol{V}_{\theta^*}(\boldsymbol{x}_t, s, t) = \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t) + \frac{2}{2\alpha}\alpha(\boldsymbol{v}^+(\boldsymbol{x}_t, t) - \boldsymbol{v}^{\text{old}}(\boldsymbol{x}_t, t)) = \boldsymbol{v}^+(\boldsymbol{x}_t, t),$$

as claimed. □

Lemma 3.3 (MeanFlow consistency). *Let $\boldsymbol{v}$ be an instantaneous velocity field. If $\boldsymbol{u}$ satisfies*

$$\boldsymbol{u}(\boldsymbol{x}_t, s, t) + (t-s)\left[\partial_t\boldsymbol{u}(\boldsymbol{x}_t, s, t) + (\partial_{\boldsymbol{x}}\boldsymbol{u})(\boldsymbol{x}_t, s, t)\boldsymbol{v}(\boldsymbol{x}_t, t)\right] = \boldsymbol{v}(\boldsymbol{x}_t, t)$$

for all $s \le t$, then $\boldsymbol{u}$ is the exact average velocity of the ODE $\dot{\boldsymbol{x}}_\tau = \boldsymbol{v}(\boldsymbol{x}_\tau, \tau)$ over $[s, t]$.

Proof. Fix $s < t$ and a trajectory $\{\boldsymbol{x}_\tau\}_{\tau\in[s,t]}$ of the velocity field $\boldsymbol{v}$ with endpoint $\boldsymbol{x}_t$ at time t, i.e., $\dot{\boldsymbol{x}}_\tau = \boldsymbol{v}(\boldsymbol{x}_\tau, \tau)$. Define $G(\tau) = (\tau - s)\boldsymbol{u}(\boldsymbol{x}_\tau, s, \tau)$. Differentiating along the trajectory,

$$\begin{aligned}G'(\tau) &= \boldsymbol{u}(\boldsymbol{x}_\tau, s, \tau) + (\tau - s)\left[\partial_t\boldsymbol{u}(\boldsymbol{x}_\tau, s, \tau) + (\partial_{\boldsymbol{x}}\boldsymbol{u})(\boldsymbol{x}_\tau, s, \tau)\dot{\boldsymbol{x}}_\tau\right] \\ &= \boldsymbol{u}(\boldsymbol{x}_\tau, s, \tau) + (\tau - s)\left[\partial_t\boldsymbol{u}(\boldsymbol{x}_\tau, s, \tau) + (\partial_{\boldsymbol{x}}\boldsymbol{u})(\boldsymbol{x}_\tau, s, \tau)\boldsymbol{v}(\boldsymbol{x}_\tau, \tau)\right] \\ &= \boldsymbol{v}(\boldsymbol{x}_\tau, \tau),\end{aligned}$$

where the last equality uses Eq. (12) at time τ. Since $G(s) = 0$, integrating from s to t gives

$$(t-s)\boldsymbol{u}(\boldsymbol{x}_t, s, t) = \int_s^t \boldsymbol{v}(\boldsymbol{x}_\tau, \tau)\,\mathrm{d}\tau.$$

This is precisely the average velocity of the ODE induced by $\boldsymbol{v}$ over $[s, t]$. □

Theorem 3.4 (The deployed MeanFlow policy improves). *In the setting of Corollary 3.2, if the induced optimum is attained for all intervals $s \le t$, the optimal average velocity $\boldsymbol{u}_{\theta^*}$ is the exact average velocity of the ODE induced by $\boldsymbol{v}^+$. Therefore the MeanFlow policy induced by $\boldsymbol{u}_{\theta^*}$ coincides with π^+, and consequently*

$$J(\pi_{\theta^*}) = J(\pi^+) > J(\pi^{\text{old}}).$$

Proof. By the premise, the optimum in Corollary 3.2 is attained for all intervals $s \le t$, so

$$\boldsymbol{V}_{\theta^*}(\boldsymbol{x}_t, s, t) = \boldsymbol{v}^+(\boldsymbol{x}_t, t).$$

Taking $s = t$ in Eq. (8), the correction term vanishes and hence

$$\widehat{\boldsymbol{v}}_{\theta^*}(\boldsymbol{x}_t, t) = \boldsymbol{u}_{\theta^*}(\boldsymbol{x}_t, t, t) = \boldsymbol{V}_{\theta^*}(\boldsymbol{x}_t, t, t) = \boldsymbol{v}^+(\boldsymbol{x}_t, t).$$

Substituting this identity back into Eq. (8) and using $\boldsymbol{V}_{\theta^*}(\boldsymbol{x}_t, s, t) = \boldsymbol{v}^+(\boldsymbol{x}_t, t)$ yields

$$\boldsymbol{u}_{\theta^*}(\boldsymbol{x}_t, s, t) + (t - s)\left[\partial_t \boldsymbol{u}_{\theta^*}(\boldsymbol{x}_t, s, t) + (\partial_{\boldsymbol{x}} \boldsymbol{u}_{\theta^*})(\boldsymbol{x}_t, s, t)\boldsymbol{v}^+(\boldsymbol{x}_t, t)\right] = \boldsymbol{v}^+(\boldsymbol{x}_t, t),$$

which is Eq. (12) with $\boldsymbol{v} = \boldsymbol{v}^+$. By Lemma 3.3, $\boldsymbol{u}_{\theta^*}$ is the exact average velocity of the ODE induced by $\boldsymbol{v}^+$. Since $\boldsymbol{v}^+$ is the marginal instantaneous velocity of the positive policy π^+, this ODE has marginals corresponding to π^+, so the exact MeanFlow update driven by $\boldsymbol{u}_{\theta^*}$ samples from π^+. Consequently $J(\pi_{\theta^*}) = J(\pi^+) > J(\pi^{\text{old}})$ for any non-degenerate reward. □

E More Implementation Details

This appendix provides more implementation details for MeanFlowNFT.

Reward. For image training, we follow the multi-reward setup of DiffusionNFT (Zheng et al., 2026a), using equally weighted `CLIPScore`, `PickScore`, and `HPSv2` as reward signals on the `PickScore` prompt set. For video training, we follow the multi-reward setup of `LongCat-Video` (Team et al., 2025). The `HPSv3-general` reward evaluates visual quality by scoring each frame with the generic prompt *"A high-quality image"* and averaging over all frames. The `HPSv3-percentile` reward instead uses the video caption as the text prompt and averages the top 30% frame scores, reducing the effect of occasional low scores caused by temporal content changes. These two HPSv3 rewards are combined with the `VideoAlign` motion-quality and text-alignment rewards described in Sec. 4.1.

Training. For `SD3.5-M`, we first construct the MeanFlow policy with the two-stage AnyFlow recipe (Gu et al., 2026). The first stage is flow-map pretraining for 6000 steps on precomputed latent–prompt pairs. It uses AnyFlow's three-mode endpoint sampling: 50% of samples take $s = t$ for standard flow matching, 25% take $s = 0$ for endpoint consistency, and the remaining samples draw $s \sim \mathcal{U}(0, t)$. Following AnyFlow (Gu et al., 2026), we use the reverse-CFG fusion with scale 4.5, so the resulting policy can be sampled CFG-free. The second stage performs on-policy AnyFlow distillation for 12000 steps, initialized from the stage-one LoRA checkpoint, with sampling steps drawn from $\{2, 4, 8, 16, 40\}$.

MeanFlowNFT is then applied on top of the trained AnyFlow policy. We run RL finetuning for 2000 steps on `SD3.5-M` and 1600 steps on `Wan2.1`, where the video run starts from the publicly released AnyFlow-Wan checkpoint. Both runs use CFG-free 4-step rollouts, equally weighted reward dimensions, $\beta = 0.1$, KL weight 10^{-4}, AdamW with learning rate 3×10^{-6}, and fresh training-time (s, t) pairs sampled from the same three-mode AnyFlow schedule. For video, following `LongCat-Video`, we compute a group-normalized relative advantage for each reward dimension independently, and then average the normalized advantages for the RL update.

Evaluation. Besides the evaluation protocol described in Sec. 4.1, we evaluate videos with the official `VBench` suite following AnyFlow (Gu et al., 2026). Specifically, we compute all 16 VBench dimensions and report their aggregated `Total`, `Quality`, and `Semantic` scores in Tab. 2, with the full per-dimension breakdown in Tabs. 3 and 4. All evaluations, including the image benchmarks, are run on 8 NVIDIA H20 GPUs.

F Training Only on $s{=}t$ Pairs

MeanFlow training mixes two kinds of pairs. Zero-length pairs ($s{=}t$) supervise the instantaneous velocity $\boldsymbol{u}_\theta(\boldsymbol{x}_t, t, t)$, as in standard flow matching. Finite-interval pairs ($s{<}t$) supervise the average velocity behind few-step sampling. Training only on $s{=}t$ pairs keeps just the instantaneous velocity and drops this average velocity. The 4-step training rollouts then degrade, so the reward briefly rises and collapses. In contrast, our default keeps both pair types throughout training and improves stably (Fig. 8).

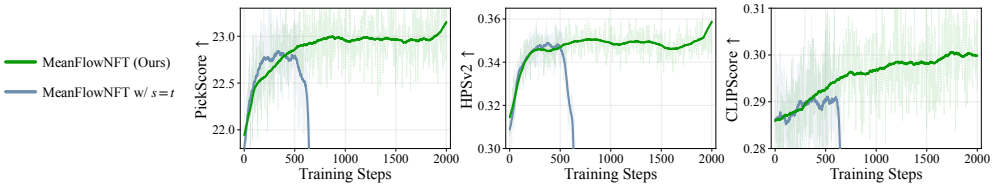


Figure 8: Training reward curves for MeanFlowNFT and a variant trained only on $s{=}t$ pairs, *i.e.*, only the instantaneous velocity $\boldsymbol{u}_\theta(\boldsymbol{x}_t, t, t)$.

G Additional Test-time Scaling Results

This section completes the test-time scaling study of Fig. 5. Quantitatively, Fig. 9 reports the remaining metrics on `SD3.5-M` and Fig. 10 the corresponding results on `Wan2.1` 1.3B. MeanFlowNFT retains AnyFlow's any-step scaling behavior while consistently delivering stronger generation quality. Qualitatively, Figs. 11 and 12 compare samples across sampling steps N. AnyFlow varies markedly in layout and content, whereas MeanFlowNFT stays consistent.

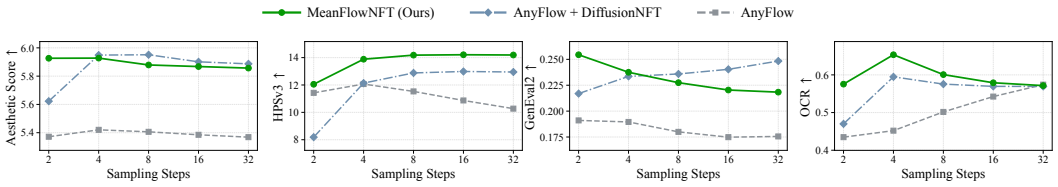


Figure 9: Additional quantitative results of MeanFlowNFT test-time scaling on `SD3.5-M`.

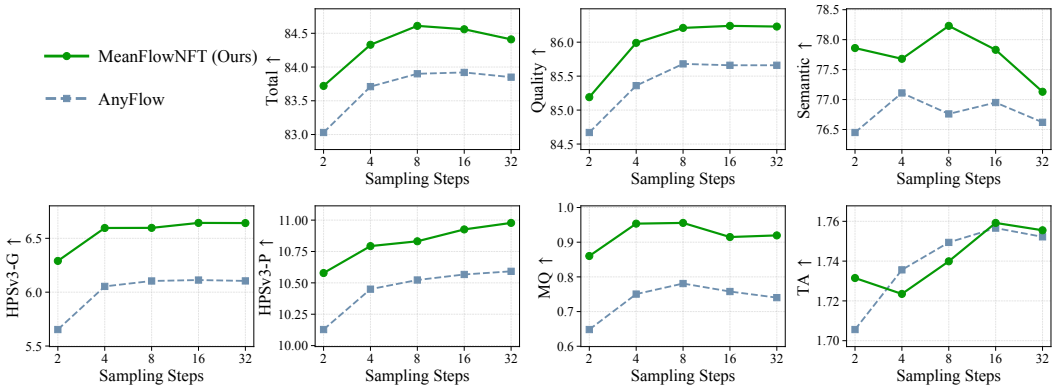


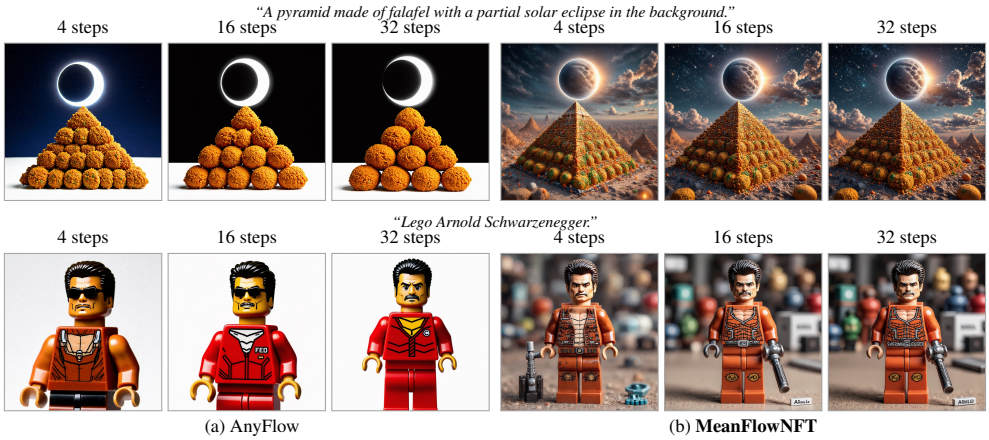
Figure 10: Quantitative results of MeanFlowNFT test-time scaling on `Wan2.1` 1.3B. `Total`, `Quality`, and `Semantic` are `VBench` scores, while the remaining metrics (`HPSv3-G`/`HPSv3-P` and `MQ`/`TA`) are evaluated on the 256 held-out prompts.

H VBench Full Results

In this section, we report the per-dimension `VBench` breakdown for the `Wan2.1` 1.3B video-generation experiment of Tab. 2, split across Tabs. 3 and 4 for readability.

I More Qualitative Results

This section provides additional qualitative comparisons for both image and video generation. The image examples compare MeanFlowNFT with multi-step/few-step RL methods, few-step distillation baselines, and directly applying DiffusionNFT to few-step generators. The video examples



(a) AnyFlow (b) **MeanFlowNFT**

Figure 11: Qualitative results of test-time scaling on `SD3.5-M`.



(a) AnyFlow (b) **MeanFlowNFT**

Figure 12: Qualitative results of test-time scaling on `Wan2.1` 1.3B.

Table 3: Full `VBench` per-dimension results on `Wan2.1` 1.3B (part 1 of 2). Among few-step models, **bold** and underline denote the best and second-best results. †denotes results reproduced using open-source checkpoints. All other results were reproduced by us.

Method	Dynamic Degree↑	Temporal Flickering↑	Human Action↑	Overall Consistency↑	Multiple Objects↑	Color↑	Appearance Style↑	Scene↑
Multi-step models (50 steps)								
`Wan2.1` 1.3B (w/ CFG)†	65.56	99.32	93.80	25.47	74.36	89.43	21.32	44.91
+ `LongCat-Video` RL†	52.78	99.12	92.00	25.43	75.90	87.23	21.19	39.55
Few-step models (4 steps)								
rCM†	**88.89**	97.28	91.20	24.72	71.42	88.12	20.59	41.82
DMD	88.61	97.37	94.00	24.90	76.16	86.00	20.19	39.52
AnyFlow†	58.89	98.83	93.20	**25.17**	82.48	88.42	20.81	**43.71**
MeanFlowNFT (Ours)	59.45	**99.44**	**94.40**	25.09	**84.79**	**89.21**	**21.08**	43.41

Table 4: Full `VBench` per-dimension results on `Wan2.1` 1.3B (part 2 of 2). Among few-step models, **bold** and underline denote the best and second-best results. †denotes results reproduced using open-source checkpoints. All other results were reproduced by us.

Method	Object Class↑	Spatial Relationship↑	Aesthetic Quality↑	Motion Smoothness↑	Temporal Style↑	Imaging Quality↑	Subject Consistency↑	Background Consistency↑
Multi-step models (50 steps)								
`Wan2.1` 1.3B (w/ CFG)†	90.98	71.87	65.95	98.76	23.33	67.42	94.98	96.63
+ `LongCat-Video` RL†	89.68	75.56	67.07	98.87	23.08	70.08	96.38	96.27
Few-step models (4 steps)								
rCM†	88.78	71.80	65.27	97.89	22.67	68.95	94.04	93.96
DMD	87.71	76.88	65.86	97.88	23.04	68.03	94.41	94.16
AnyFlow†	90.28	**81.96**	**70.00**	98.61	22.77	**69.57**	97.89	96.55
MeanFlowNFT (Ours)	**90.49**	81.36	69.53	**99.22**	**23.06**	68.82	**98.44**	**97.07**

further compare against `Wan2.1`, `LongCat-Video` RL, and few-step video distillation baselines. Across these examples, MeanFlowNFT produces more faithful and visually coherent results, while preserving strong quality across different sampling steps.



Figure 13: Text-to-image comparison on `SD3.5-M`. Here "+NFT" abbreviates "+DiffusionNFT".



Figure 14: Text-to-image comparison on `SD3.5-M`. Here "+NFT" abbreviates "+DiffusionNFT".



Figure 15: Text-to-image comparison on `SD3.5-M`. Here "+NFT" abbreviates "+DiffusionNFT".

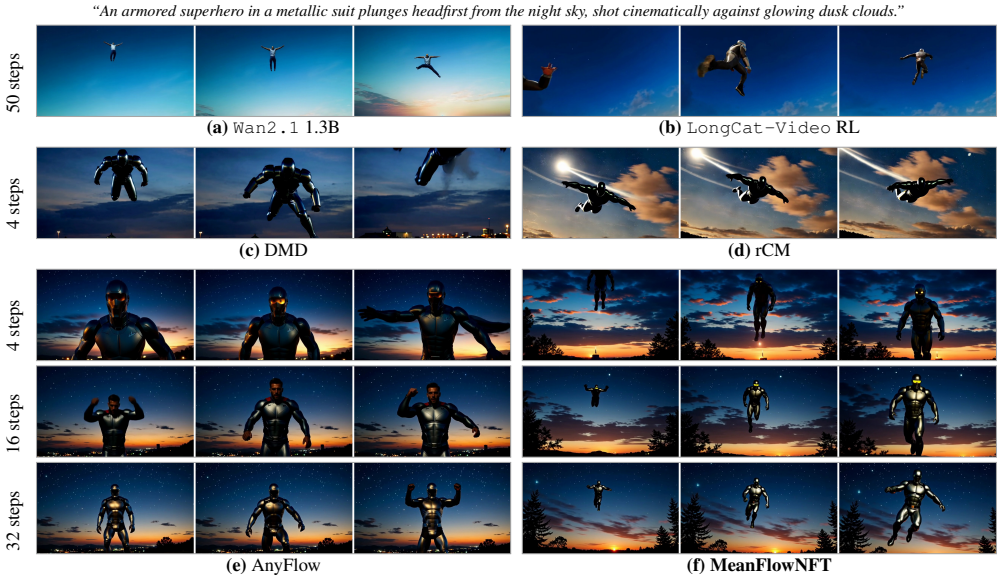


Figure 16: Text-to-video comparison on `Wan2.1` 1.3B.

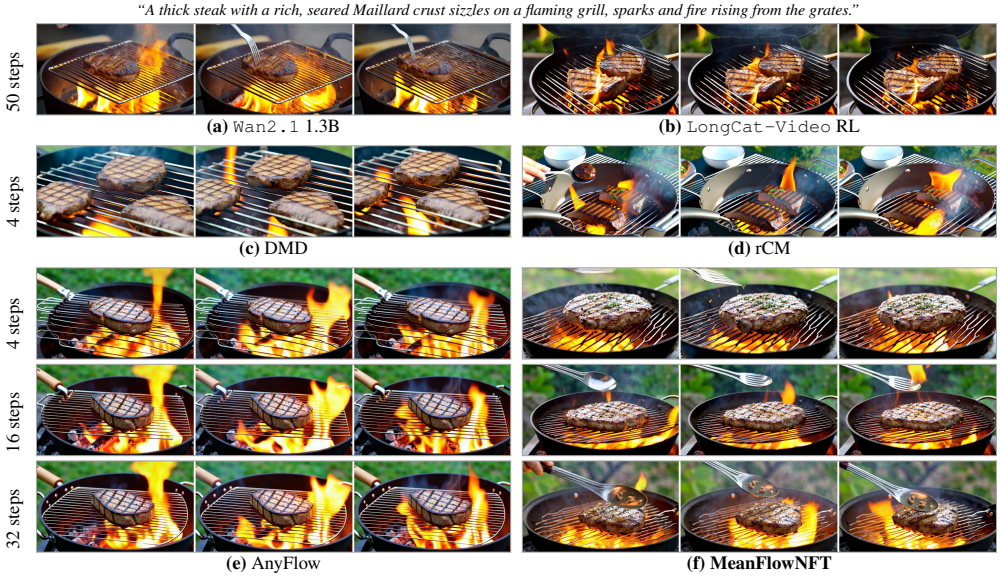


Figure 17: Text-to-video comparison on `Wan2.1` 1.3B.

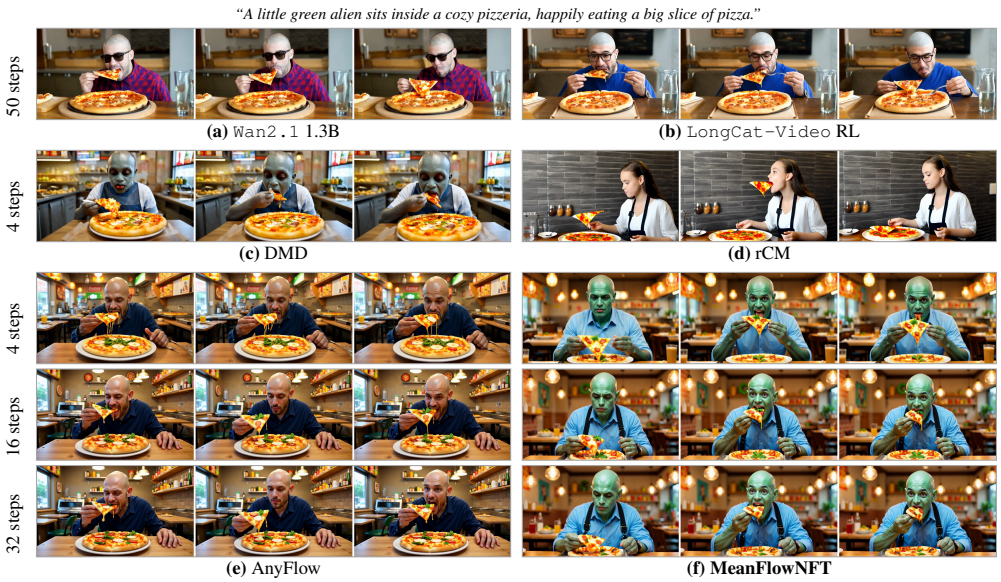


Figure 18: Text-to-video comparison on `Wan2.1` 1.3B.