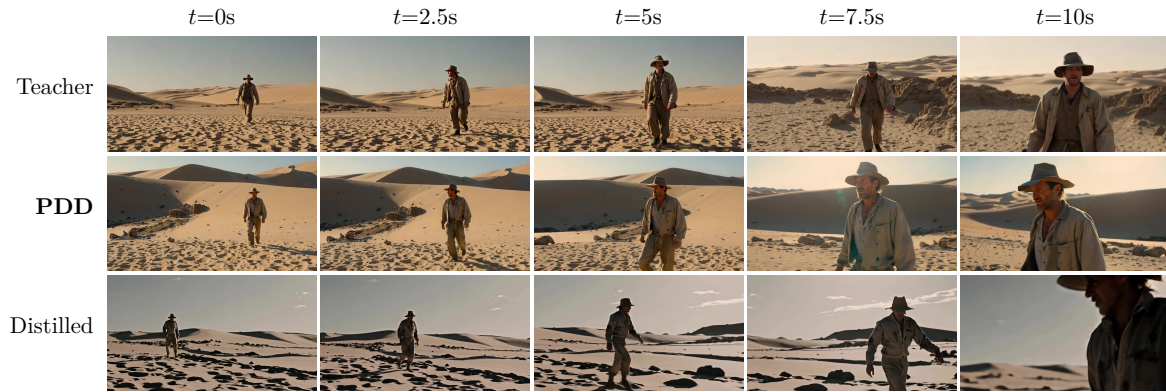


Parallel Decoding Distillation for Fast Image and Video Generation

Neta Shaul², Chao Liu¹, Arash Vahdat^{1,†}, Julius Berner^{1,†}

¹NVIDIA, ²Weizmann Institute of Science, [†]equal advising

Generation in video diffusion or flow models is computationally expensive due to the slow and iterative sampling process. Current state-of-the-art (SOTA) acceleration methods heavily rely on variational score distillation (VSD) and adversarial losses to distill diffusion models into few-step generators. Albeit achieving high-quality video generation, these training losses are notoriously hard to optimize and suffer from mode collapse, leading to loss of video diversity and lack of motion. In this paper, we introduce Parallel Decoding Distillation (PDD), a simplified and scalable trajectory-based distillation method for fast inference of diffusion and flow matching models. Our architecture and training procedure are compatible with any pre-trained model and support sampling with a varying number of function evaluations (NFE). PDD accelerates generation by predicting multiple denoising steps per network evaluation. Conceptually, it learns a representation of the mean velocity without regressing its derivative using JVPs or finite-difference approximations. Our method achieves SOTA performance with 4-8 NFE on LTX-2.3 Text-to-Video/Audio, Wan 14B Text-to-Video, and Qwen-Image Text-to-Image. Moreover, PDD presents a significant improvement in generated video diversity. Project page: <https://research.nvidia.com/labs/genair/pdd>



"A solitary walker moves across a vast, arid desert landscape under a scorching sun, the dry, rhythmic crunch of boots pressing into shifting sand echoing in the open, expansive space. The walker is wearing a tattered, beige jacket and loose-fitting pants that rustle with every step, a wide-brimmed hat shading their face from the intense sunlight. They walk with a determined yet weary posture, occasionally glancing around at the expansive sand dunes and rocky outcrops as a low, mournful whistle of wind sweeps across the barren terrain. In the background, distant sand dunes roll endlessly towards the horizon, creating a sense of isolation and vastness. The scene transitions between medium shots and close-ups as the walker trudges forward, their heavy, labored breathing audible in the stillness, emphasizing the harsh, desolate environment."



"Two cyclists, one male and one female, are approaching a stop sign at the intersection from opposite directions. The male cyclist is wearing a red helmet and black cycling gear, while the female cyclist is wearing a blue helmet and white cycling clothes. They are both pedaling towards the stop sign, gradually slowing down as they near the intersection. The male cyclist is glancing left, checking for traffic, while the female cyclist is focused straight ahead. The background shows a quiet suburban street with green trees and parked cars. The video captures their approach in a tracking shot, gradually zooming in on each cyclist as they slow to a stop, maintaining eye-level perspectives."

Figure 1: (top) 10s 720p videos with audio from LTX-2.3 teacher (4×30 NFE), Parallel Decoding Distillation (PDD) (8 NFE), and the official distilled model (8 NFE). PDD can follow the teacher more closely. **(bottom)** PDD vs. DMD2 and AnyFlow on the Wan2.1 Text-to-Video 14B model. All methods use 4 NFE. Notably, PDD demonstrates high video quality while preserving better generation diversity across distinct initial noises compared to the baselines.

1. Introduction

Large-scale diffusion and flow models [21, 51, 29, 30, 1] have achieved remarkable capabilities of media generation, including text-to-image [61, 26], text-to-video [25, 56, 40], and multi-modal generation [19]. Yet, the cost of generation remains high, as their inherently iterative sampling algorithms often require hundreds of network evaluations. The resulting computational cost and latency are one of the main bottlenecks for many applications, such as content editing, real-time video generation, and interactive world modeling. Thus, developing distillation methods for few-step media generation models has become an active field of research [47, 66, 64, 32, 27, 7, 41, 70, 55, 39, 17, 63].

The approaches for distilling diffusion and flow models are broadly categorized into two families: i) *trajectory-based* methods [45, 52, 12, 44, 14, 7] in which a student model distills the many-step sequential sampling process of a pre-trained teacher into a few-step process; and ii) *distribution-based* methods [46, 59, 67, 64], which relax the constraint of following the teacher trajectories and instead align only the marginal distributions of the student and teacher processes. Trajectory-based methods have shown promising results for fast image generation. However, when applied to video models, they are typically bottlenecked by degraded video quality and costly training algorithms. As a result, the current dominant methods for distillation of video models are distribution-based [11, 39]. While recent works regularize the training dynamics by additionally incorporating trajectory-based distillation losses [70, 17], they still suffer from alternating training objectives, high memory requirements, and mode collapse, leading to reduced diversity and often static videos.

In this paper, we introduce *parallel decoding distillation* (PDD), a trajectory-based distillation method for fast inference of diffusion and flow models. Instead of merging multiple denoising steps into a single larger step [45, 12, 32, 72], we learn a parallel decoder that predicts multiple denoising steps in a single network evaluation. Specifically, PDD discretizes the time domain of the flow into a fixed sequence of N intervals, which are grouped into blocks of size L . During training, the parallel decoder learns to predict the mean velocities in all intervals within a block using a single forward pass, as illustrated in Figure 2. Target mean velocities are approximated using a Runge-Kutta solver (e.g., Euler or Midpoint) applied to the pre-trained teacher model. At generation, by predicting the velocities across L intervals, we obtain a sample with N/L steps. By varying the block size during training, PDD supports sampling with different

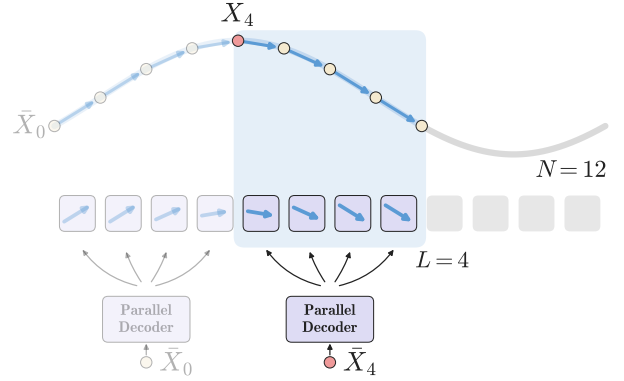


Figure 2: The sampling trajectory is discretized into N intervals, which are grouped into blocks of size L . The parallel decoder predicts the mean velocities for all intervals within a block using a single evaluation.

number of function evaluations (NFEs) during inference. PDD is conceptually related to Pi-Flow [7], but provides a simplified training algorithm that removes the need for an additional policy head and supports a variable NFEs at generation time. Additionally, our formulation clarifies the connection to flow map distillation methods, particularly the Lagrangian formulation [72, 4]. By avoiding Jacobian-vector products (JVPs) and finite differences, we obtain minimal-cost training algorithm.

We showcase the efficacy of our method on text-to-video tasks with Wan2.1 [56] 1.3B and 14B models, achieving SOTA video quality generation with 4 NFE on VBench [24] while preserving better video diversity compared to distribution-based baselines. Additionally, we test PDD on text-to-image tasks with the Qwen-Image 20B model, achieving SOTA scores with 4 to 8 NFE on OneIG [6], GenEval [16], and DPG-Bench [22] benchmarks.

All in all our main contributions are:

1. Formulate Parallel Decoding Distillation, a scalable, trajectory-based distillation method for fast inference of flow matching and diffusion models.
2. Develop a single, regression-based training objective that avoids the need for JVPs, finite differences, or multi-stage training procedures and produces high-quality and diverse samples without VSD or GAN losses.
3. Propose a simple architecture and training algorithm that are supported by any pre-trained model and allows generation with varying NFE without additional time conditioning.
4. Validate our method on ImageNet-256, Qwen-Image, Wan2.1 1.3B/14B, and LTX-2.3, achieving SOTA performance with improved diversity.

2. Generative Flow Models

Notation The state space is denoted by $\mathcal{X}$, where $\mathcal{X} = \mathbb{R}^{c \times h \times w}$ for images and $\mathcal{X} = \mathbb{R}^{f \times c \times h \times w}$ for videos. Random variables taking values in $\mathcal{X}$ are denoted by the uppercase letter X , and states in $\mathcal{X}$ are denoted by the lowercase letter x . Fixed integers are denoted by the uppercase letters $N, L \in \mathbb{N}$, while running indices or integer-valued random variables are denoted by the lowercase letters n, k .

Flow Matching and Diffusion Models The currently dominant approaches for training generative flow models are flow matching [29, 30, 1] and diffusion models [49, 21, 51]. As we are interested only in deterministic processes, for simplicity, we treat both as flows (1).

A flow process $(X_t)_{0 \leq t \leq 1}$ taking values in $\mathcal{X}$ is defined by a *velocity field* $v : \mathcal{X} \times [0, 1] \rightarrow \mathcal{X}$ and a *source distribution* $p_0 : \mathcal{X} \rightarrow \mathbb{R}_{\geq 0}$, which serves as a boundary condition setting the marginal of the process at time $t = 0$:

$$\frac{d}{dt}X_t = v_t(X_t), \quad X_0 \sim p_0. \quad (1)$$

The marginal of the process, called the *probability path*, is a time-dependent density $p : \mathcal{X} \times [0, 1] \rightarrow \mathbb{R}_{\geq 0}$ such that $X_t \sim p_t$ for all $t \in [0, 1]$.

Assume a dataset of i.i.d. samples in $\mathcal{X}$ from a *target distribution* p_1 and some easy-to-sample *source distribution* p_0 . Flow matching provides a framework for learning a model v_t such that the flow $(X_t)_{0 \leq t \leq 1}$ defined by equation 1 maps samples from the source $X_0 \sim p_0$ to samples from the target $X_1 \sim p_1$.

Importantly, during training, the marginal p_t of a flow matching model can be sampled efficiently using the *interpolant process*. While our method is agnostic to the chosen interpolant process, all pre-trained models used in this work were trained using the linear scheduler,

$$X_t = (1 - t)X_0 + tX_1, \quad (2)$$

where $X_0 \sim p_0$, $X_1 \sim p_1$, and $t \in [0, 1]$.

Sampling with flows Obtaining a sample $X_1 \sim p_1$ from a trained flow model v_t is done by solving the ODE in equation 1. A general numerical approach discretizes the time interval $[0, 1]$ into a sequence of N smaller intervals, $0 = t_0 < t_1 < \dots < t_N = 1$. Then, the exact solution on each interval is

$$X_{n+1} = X_n + (t_{n+1} - t_n) u_n(X_n), \quad (3)$$

where we use the simplified notation $X_n := X_{t_n}$, and u_n denotes the *mean velocity* of the n -th interval $[t_n, t_{n+1}]$, defined as

$$u_n(X_n) = \frac{1}{t_{n+1} - t_n} \int_{t_n}^{t_{n+1}} v_t(X_t) dt. \quad (4)$$

The solution is obtained by sequentially approximating the integral in equation 4. The simplest numerical method is the Euler solver, which approximates the velocity v_t as constant over the interval $t \in [t_n, t_{n+1}]$, yielding the mean velocity

$$u_n(X_n) \approx v_{t_n}(X_n). \quad (5)$$

Runge-Kutta methods are a family of higher-order solvers that use additional evaluations of the velocity v_t in the interval $[t_n, t_{n+1}]$ to achieve a higher-order approximation. We use the Midpoint method,

$$u_n(X_n) \approx v_{t_{\text{mid}}}(X_{\text{mid}}), \quad (6)$$

where X_{mid} and t_{mid} are the midpoint state and time, respectively,

$$X_{\text{mid}} = X_n + \frac{t_{n+1} - t_n}{2} v_{t_n}(X_n), \quad t_{\text{mid}} = \frac{t_{n+1} + t_n}{2}.$$

3. Parallel Decoding Distillation

To accelerate sampling from flow models, we propose to learn a parallel decoding model that predicts multiple integration steps (3) in a single network evaluation, rather than approximating them one step at a time. Figure 2 illustrates our approach.

Parallel decoder. Assume a pre-trained flow model v_t with a flow process $(X_t)_{0 \leq t \leq 1}$ defined by equation 1 and marginals p_t . Fix a time discretization of length $N \in \mathbb{N}$,

$$0 = t_0 < t_1 < \dots < t_N = 1. \quad (7)$$

Then, a *block* of size L starting at step n is the set of indices $\{n, \dots, n + L - 1\}$.

For a state $X_n \sim p_{t_n}$ at time step t_n , a *parallel decoder* $\bar{u}_n^\theta(\cdot | X_n) \in \mathcal{X}^L$ with block size $L \in \mathbb{N}$ is trained to predict the mean velocities of all intervals in the next block using a single network evaluation,

$$\bar{u}_n^\theta(k | X_n) \approx u_k(X_k), \quad k = n, \dots, n + L - 1. \quad (8)$$

Importantly, our parallel decoder (8) is well-defined, since the discretized flow process in this block, i.e., X_k , $k \in \{n, \dots, n + L - 1\}$, is fully specified by the exact solution (3) and the initial state $X_n \sim p_{t_n}$.

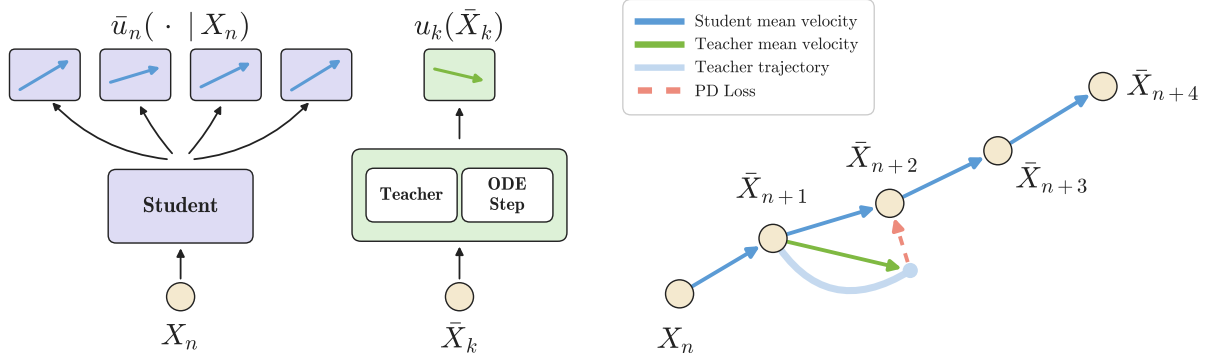


Figure 3: (left) The PDD student approximates the mean velocity across multiple consecutive intervals in a single evaluation. The pre-trained flow model (teacher) provides the mean velocity of a single interval using an ODE solver step. (right) Illustrate estimation of the PD loss given an initial state $X_n \sim p_{t_n}$ and a block size $L = 4$; i) student predicts the mean velocities $\bar{u}_n^\theta(\cdot | X_n) \in \mathcal{X}^L$; ii) Following the student velocities yields the states $\bar{X}_k$ in the block $k \in \{n, \dots, n + L - 1\}$; iii) one of the states is randomly selected and the student’s output velocity is matched to the teacher’s mean velocity in the corresponding interval using the PD loss (11).

The parallelized process During training, we employ an on-policy optimization algorithm that requires the intra-block process given by the parallel decoding model.

For a state $X_n \sim p_{t_n}$ at time step t_n , the *parallelized process* is defined by

$$\bar{X}_{k+1} = \bar{X}_k + (t_{k+1} - t_k) \bar{u}_n^\theta(k | X_n), \quad (9)$$

for the block $k \in \{n, \dots, n + L - 1\}$, with the initial condition $\bar{X}_n = X_n$.

The parallelized process is obtained by substituting the parallel decoder into the update rule of the exact solution (3). Notably, $\bar{u}_n^\theta(\cdot | X_n) \in \mathcal{X}^L$ depends only on the initial state X_n . Thus, the parallelized process $(\bar{X}_k)_{n \leq k \leq n+L}$ is simulated using a single evaluation of the parallel decoder.

Sampling While classical algorithms advance a single interval at each ODE step using the recursive rule (3), with the parallel decoder we can advance L intervals simultaneously. For a current state $X_n \sim p_{t_n}$, we approximate the exact solution (3) in the block $\{n, \dots, n + L - 1\}$ using the parallelized process (9). Then, solving the recursion over the intra-block index k , yields the *block-step rule*

$$\bar{X}_{n+L} = X_n + \sum_{k=n}^{n+L-1} (t_{k+1} - t_k) \bar{u}_n^\theta(k | X_n). \quad (10)$$

By repeating the recursive block-step rule (10) N/L times, in each step approximating $\bar{X}_n \approx X_n$, we obtain a clean sample from the model $\bar{X}_N$. We provide a PyTorch pseudocode in Algorithm 1, where dimension 0 indexes the parallel predictions $k = 0, \dots, N - 1$.

Algorithm 1 Parallel decoding sampling

```
# student - parallel decoder model
# t - time discretization
# L - block size
# shape - data shape

# init noise
x_n = randn(*shape)
# step sizes
h = diff(t, dim=0)
for n in range(0, len(t)-1, step=L):
    # parallel predictions
    u = student(x_n, t[n])
    # slice current block
    u_n, h_n = u[n:n+L], h[n:n+L]
    # block step
    x_n = x_n + einsum('k,k...', h_n, u_n)
return x_n
```

Training The parallel decoder is trained by regressing onto a Runge-Kutta approximation of the mean velocity (8) of the pre-trained teacher flow model. In practice, we use a single Euler or Midpoint step. To obtain a tractable loss, we employ on-policy training, estimating the teacher’s mean velocity on student outputs $\bar{X}_k$ as defined in (9).

For a time discretization $t_0 < \dots < t_N$ and a block size L , our training objective is

$$\mathcal{L}_{PD}(\theta) = \mathbb{E} \left[\left\| \bar{u}_n^\theta(k | X_n) - u_k(\text{sg}(\bar{X}_k)) \right\|^2 \right], \quad (11)$$

where block starting indices $n \in \{0, L, \dots, N - L\}$ and intra-block indices $k \in \{n, \dots, n + L - 1\}$ are sampled uniformly, $X_n \sim p_{t_n}$ is sampled using the interpolant process (2), $\bar{X}_k$ is the parallelized process (9) in the block, $\text{sg}(\cdot)$ denotes the stop-gradient operator, and the teacher mean velocity u_k is approximated with a Runge-Kutta step.

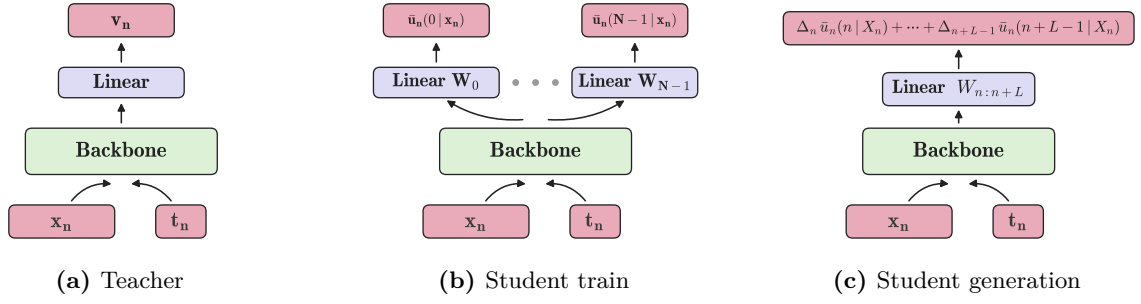


Figure 4: Architecture of the parallel decoder (b) vs. the pre-trained flow model (a). Notably, the parallel decoder utilizes the same backbone, but with N times the final linear layer. (c) At generation, instead of applying a linear layer per intra-block step (9), we can fuse the layers into a single linear layer that outputs the block’s average velocity, which advances the state across the entire block (14).

Importantly, both $\bar{u}_n^\theta(k|X_n)$ and $\bar{X}_k$ are obtained with a single evaluation of the parallel decoder $\bar{u}_n^\theta$. Approximating the mean velocity u_k with a single Euler (5) or Midpoint (6) step requires 1-2 evaluations of the teacher model v (resp.). This makes the parallel decoding (PD) loss (11) tractable even at large scales. Figure 3 illustrates evaluation of the PD loss and we provide PyTorch pseudocode in Algorithm 2. For simplicity, we ignore the batch dimension, however all operations can be batched.

Proposition 1 shows that the PD loss is a valid objective for learning the parallel decoder as defined in (8). In particular, up to a controllable Runge-Kutta approximation error, the minimizer of the PD loss samples exactly the teacher trajectories, i.e., $\bar{X}_n = X_n$, for $n = 0, \dots, N$. The proof is in Appendix D.

Proposition 1. *The minimizer of the parallel decoding loss (11) satisfies the parallel decoder condition (8).*

Data-free training In cases where data is not available, we propose an online on-policy training scheme that avoids the need to sample $X_n \sim p_{t_n}$ with the interpolant process (2). Instead, we follow our sampling algorithm 1, while alternating solver and training steps. That is, we sample an initial state $X_0 \sim p_0$, then for the next $N/L - 1$ iterations, we utilize the output of the parallel decoder $\bar{u}_n^\theta(\cdot|X_n)$ both for the optimization step and to advance $\bar{X}_n$ to $\bar{X}_{n+L}$; see the modified PyTorch pseudocode for PD loss in Algorithm 3 in the appendix. Importantly, we apply the stop-gradient operation when advancing the state, preventing additional memory or compute cost.

Architecture and variable block size Our requirement is an architecture that predicts L mean-velocities $\bar{u}_n^\theta(\cdot|X_n) \in \mathcal{X}^L$, i.e., one velocity for each time step in the block, instead of the single instantaneous velocity prediction $v_{t_n}(X_{t_n}) \in \mathcal{X}$ of the pre-

Algorithm 2 Parallel decoding loss

```
# student - parallel decoder model
# teacher - pre-trained flow model
# runge_kutta - approximates teacher's mean velocity
# t - time discretization
# L_min - minimal block size
# L_max - maximal block size
# x - data

# grid size
N = len(t) - 1
# sample block index
n = L_min * randint(0, N//L_min)
# sample the probability path
x_n = (1-t[n]) * randn_like(x) + t[n] * x
# parallel predictions
u = student(x_n, t[n])
# step sizes
h = diff(t, dim=0)
# sample index in the block
k = randint(n, clip(n+L_max, max=N))
# step from n to k
x_k = x_n + einsum('1,1...', h[n:k], u[n:k])
# teacher mean velocity
u_k = runge_kutta(teacher, x_k, t[k], h[k])
# mse loss
loss = mse_loss(u[k], u_k.detach())

return loss
```

trained flow model. As illustrated in Figure 4, we utilize the same backbone architecture of the pre-trained flow model, but with the final linear layer repeated N times, i.e., one for each time step in the grid (7). Formally, we assume the teacher architecture is of the form

$$v_t(x) = W H_t(x), \quad (12)$$

where H_t is the backbone that outputs the final hidden state and W is the final linear layer. Then we learn N linear layers $W_0^\theta, \dots, W_{N-1}^\theta$ and, for $k \geq n$, the parallel decoder’s architecture is given by

$$\bar{u}_n^\theta(k|x_n) = W_k^\theta H_{t_n}^\theta(x_n). \quad (13)$$

This enables initialization from the final layer of the pretrained flow model. Additionally, the advantage of taking N (grid size) instead of exactly L (block size) linear layers is that it allows us to learn a single model that can predict any block size without the need to introduce a second time coordinate, which is, for instance, required for flow maps [14, 44, 4].

In practice we are not interested in all possible block sizes, but in some subset of block sizes. Thus, we define a minimum and maximum block size $L_{\min} \leq L_{\max} \in \mathbb{N}$. Then, during training, we consider multiples of $L_{\min}$ for indices n of initial states and sample $k \in \{n, \dots, n + L_{\max} - 1\}$ inside each block.

3.1. Layer Fusion and connection to flow maps

An important observation emerges when comparing PDD training and generation. For a block size L and starting step n , estimating the PD loss (11) requires the intra-block steps of the parallelized process (9), and thus uses all distinct student output directions $W_k^\theta H_{t_n}^\theta(X_n)$, for $k = n, \dots, n + L - 1$. In contrast, during generation, we use the block step (10) to skip L intervals. This requires only the weighted-average direction, which, for our architecture (13), yields

$$\bar{X}_{n+L} = \bar{X}_n + (t_{n+L} - t_n) W_{n:n+L}^\theta H_{t_n}^\theta(\bar{X}_n), \quad (14)$$

where $W_{n:n+L}^\theta$ is a fused linear layer,

$$W_{n:n+L}^\theta = \sum_{k=n}^{n+L-1} \Delta_k W_k^\theta, \quad \Delta_k = \frac{t_{k+1} - t_k}{t_{n+L} - t_n}. \quad (15)$$

Thus, our shared backbone $H_{t_n}^\theta$ learns a representation of the mean-velocity over the interval $[t_n, t_{n+L}]$. However, instead of using JVPs [44, 14, 4, 72] or finite differences [35], we use the learnable linear maps W_k^θ , for $k = n, \dots, n + L - 1$, to decompose the mean-velocity prediction into parallel sub-interval predictions. Then, during training, the gradients through the shared backbone recover, in expectation, the training signal for learning the full-interval mean-velocity. A discussion of the exact connection to flow maps is provided in Appendix C. We validate that the parallel decoder can learn non-trivial trajectories by comparing the curvature of its trajectories with that of the teacher trajectories in Figure 17 in the appendix.

An additional practical implication is that during inference we can avoid the extra compute of an enlarged final layer and we only need to hold one fused linear layer per block in memory.

4. Related works

Trajectory-based distillation The first successful method to distill the trajectories of the flow

Table 1: High-level differences between flow-map distillation methods, Pi-Flow, and our PDD.

	Eulerian/Lagrangian Flow Maps	Pi-Flow	PDD (Ours)
NFE	Variable	Fixed	Variable
JVP/finite-diff.	Required	Free	Free
Head at inference	Linear	Gaussian mixture	Fused-linear

ODE (1) using a student-teacher scheme to achieve few-step generation is Progressive Distillation [45]. They gradually increase the step size of the student using multi-phase training. Consistency models [52, 34, 50, 32, 13] initialize the student from the teacher and then directly distill a map from any intermediate state along the trajectory to the clean state using a self-consistency condition. More recent works [54, 3, 44, 55, 23, 27, 53, 5], distill the mean velocity between any two states on the trajectory.

These methods have shown promising results on image generation. However, on large scale video models they fail to achieve high-quality few-step generation. Additionally, they often rely on JVP or finite differences which are expensive to evaluate on large-scale models or yield unstable training dynamics.

Most related to our method is Pi-Flow [7]. Similar to us, they make the observation that given a pretrained flow model, the trajectories in the interval $[t_n, t_{n+L}]$, are fully specified by the initial state, and use it to delegate the integration in that interval to a small learnable policy head. In contrast, we utilize it to motivate our parallel prediction paradigm, simplifying training and inference, and changing the focus from expressive parametrizations (such as Gaussian mixtures) to improved supervision. While Pi-Flow distills the continuous-time instantaneous velocity (1) v , we go beyond by discretizing time and distilling numerical approximations of the mean velocity (4) u . Furthermore, using layer fusion (14) we avoid any additional cost at generation. Lastly, our training algorithm naturally enables sampling with different NFEs, whereas Pi-Flow is restricted to fixed NFE. The high-level differences are summarized in Table 1.

Training from scratch There is a growing effort [12, 14, 4, 60, 71, 72] to develop end-to-end training methods for few-step generation models. These methods directly learn the flow map and avoid separate pre-training and distillation phases by combining the flow matching objective [29, 30] with a trajectory-based distillation objective. Instead of using a separate student-teacher scheme, they use the current state of the model as the teacher. This line of work has spawned several follow-up methods that aim to improve convergence [69, 15, 18, 33, 37] or replace JVPs with finite-difference approximations [35, 41].



Figure 5: PDD - Midpoint (ours) vs. DMD2, and AnyFlow on the Wan Text-to-Video 14B model. The three methods use 4 NFE. Notably, PDD achieves high-quality video while preserving better diversity.

While PDD could potentially be extended to a self-distillation framework, such methods require a completely different pipeline, including larger datasets and significantly more compute.

Distribution-based distillation Instead of letting a student distill the trajectories of a teacher model, a more flexible but sufficient condition for a few-steps generation model is to align the marginals p_t of the two. ADD [46], LADD [47], and APT [28] are using GAN losses to fine-tune a pre-trained diffusion model into a few step generation model. DMD [67, 66] incorporates the VSD loss [59] for distillation. f -distill [64] generalizes the VSD loss to f -divergences, and SiD and extensions [74, 73, 36] use variants of Fisher divergences. Distribution-based methods established themselves as dominant approaches for distillation of large-scale video models [39, 70, 17, 11, 28]. However, they suffer from mode collapse leading to a lack of generation diversity and motion. Recent works try to mitigate that by regularizing the loss with trajectory-based objectives [8, 70, 17].

Moreover, distribution-based methods are known to be sensitive to hyper-parameters, require additional

trainable parameters, and their performance can vary significantly across training iterations due to their alternating training objectives. In contrast, we find that the memory-efficient simple PD loss is much more robust to hyper-parameter choices, and presents consistent generation across training iteration.

5. Experiments

We empirically validate PDD on three tasks: i) class-conditional image generation on ImageNet-256 [43] using the SiT-XL+REPA model [68]; ii) text-to-image generation using Qwen-Image [61]; and iii) text-to-video generation using the 1.3B and 14B variants of Wan2.1 [56] as well as LTX-2.3 [19]. The datasets used for PDD training are described in Appendix B.

Training setup. PDD has three main design choices: i) the time discretization (7), defined by the grid size N and the time reparameterization; ii) the Runge-Kutta method used to approximate the mean velocity (4); and iii) the minimum and maximum block sizes, $L_{\min}$ and $L_{\max}$, used in Algorithms 2 and 3, which determine the set of available NFEs at inference time. For each task, we train two PDD

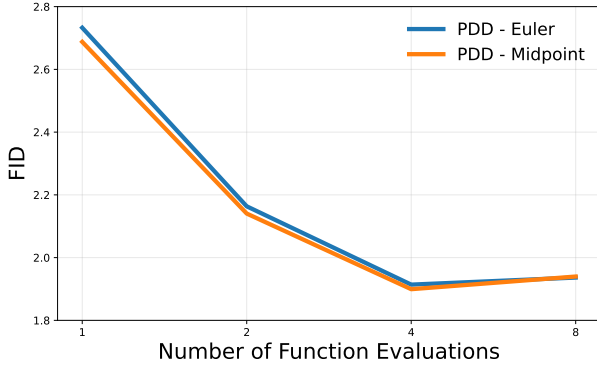


Figure 6: FID vs. NFE of PDD with Euler and Midpoint methods for approximating the mean velocity (4) on ImageNet-256 using SiT-XL+REPA model as teacher with guidance scale 2.9.

models, each with a different grid size, Runge-Kutta method, and block-size range. Additional training details are provided in Appendix B.

For class-conditional image generation, we set $N = 128$ for the Euler method and $N = 64$ for the midpoint method. For both models, we use the uniform time discretization (7), $t_n = \frac{n}{N}$, and choose block sizes $L_{\min}$ and $L_{\max}$ such that the available NFEs at inference time are 1, 2, 4, 8.

For both text-to-image and text-to-video generation, we set $N = 256$ for the Euler method and $N = 128$ for the Midpoint method. All models use the *shift* transformation [10, 48] for the time discretization:

$$t_n = \text{shift}_s\left(\frac{n}{N}\right), \quad \text{shift}_s(t) = \frac{\frac{1}{s}t}{1 + \left(\frac{1}{s} - 1\right)t}. \quad (16)$$

Due to the lack of high-quality image and video datasets, we apply data-free training as described in Algorithm 3. The minimum and maximum block sizes, $L_{\min}$ and $L_{\max}$, are chosen such that the available NFEs at inference time are 2, 4, 8 for Qwen-Image and Wan2.1, and 4, 8 for LTX-2.3.

Across all tasks, since the midpoint method requires two teacher evaluations, we accumulate the Euler loss over two intervals within each block for a fair comparison. Additionally, details about classifier-free guidance are in Appendix B.

While PDD exhibits stable convergence across hyperparameter, we find that the two most important training hyperparameters are the time reparameterization and the batch size, with larger batch sizes yielding better performance. Additionally, as shown in Tables 3, 4, 5, the midpoint approximation consistently improves performance compared to the Euler approximation.

Table 3: Qwen-Image on the OneIG-EN, DPG-Bench, and GenEval. * denotes our re-evaluation of the official checkpoint. This table reports the overall metrics. The full dimensions of the three benchmarks are in tables 6, 7, and 8 in Appendix B.2.

Method	NFE	OneIG-EN $\uparrow$	DPG-Bench $\uparrow$	GenEval $\uparrow$
Euler*	50 $\times$ 2	0.537	88.30	0.86
TwinFlow* [8]	2	0.493	86.67	0.82
PDD - Euler (Ours)		0.508	88.04	0.86
PDD - Midpoint (Ours)		0.516	88.10	0.86
DMD2* (Lightning-step4-v2) [9]	4	0.524	88.25	0.85
TwinFlow* [8]		0.502	86.18	0.82
Pi-Flow* [7]		0.533	88.11	0.85
PDD - Euler (Ours)		0.535	88.45	0.86
PDD - Midpoint (Ours)		0.538	88.66	0.86
DMD2* (Lightning-step8-v2) [9]	8	0.526	88.20	0.84
Pi-Flow* [7]		0.536	87.90	0.84
PDD - Euler (Ours)		0.538	88.51	0.86
PDD - Midpoint (Ours)		0.541	88.46	0.85

Table 4: Qwen-Image on HPSv2, PickScore, and OneIG diversity. * denotes our re-evaluation of the official checkpoint.

Method	NFE	HPSv2 $\uparrow$	PickScore $\uparrow$	OneIG diversity $\uparrow$
Euler*	50 $\times$ 2	30.83	22.78	0.200
TwinFlow* [8]	2	29.86	22.26	0.131
PDD - Euler (Ours)		29.59	22.47	0.197
PDD - Midpoint (Ours)		30.15	22.66	0.177
DMD2* (Lightning-step4-v2) [9]	4	32.34	22.98	0.095
TwinFlow* [8]		30.01	22.26	0.150
Pi-Flow* [7]		30.94	22.67	0.182
PDD - Euler (Ours)		31.05	22.72	0.192
PDD - Midpoint (Ours)		31.33	22.86	0.174
DMD2* (Lightning-step8-v2) [9]	8	32.35	22.95	0.109
Pi-Flow* [7]		31.09	22.55	0.186
PDD - Euler (Ours)		31.34	22.73	0.198
PDD - Midpoint (Ours)		31.56	22.86	0.181

Class-conditional ImageNet As shown in Figure 6, the FID of our PDD models generally improves as the NFE increases, validating that PDD successfully shares weights across different NFEs. Although the 8-NFE setting exhibits an increase in FID, we find that this degradation can be mitigated by using a lower guidance scale, at the cost of higher FID at lower NFEs; see Figure 8 in the appendix.

Table 2: FID with NFE=1 on ImageNet-256 using SiT-XL+REPA as teacher. Guidance scale $w = 2.9$.

Method	FID
Pi-Flow [7]	2.85
FreeFlow [55]	1.45
PDD - Euler	2.73
PDD - Midpoint	2.69

or an additional network as FreeFlow) and (ii) also supporting inference with multiple NFE budgets.

Text-to-image Qwen-Image We evaluate our PDD-distilled Qwen-Image models on three main benchmarks, OneIG [6], DPG-Bench [22], and GenEval [16], with NFE = 2, 4, 8. For SOTA base-

Table 5: Wan Text-to-Video 1.3B and 14B models performance and diversity metrics on VBench [24] with Self-Forcing prompt set. Diversity is measured as mean pairwise V-JEPA 2/VideoMAE V2 feature distance across 5 generated videos per prompt. * Our re-evaluation of the official checkpoint. ** Our re-implementation. † Reported numbers on proprietary prompts without diversity evaluation, since rCM [70] did not release checkpoints.

Model	Method	NFE	VBench			V-JEPA 2		VideoMAE V2	
			Overall ↑	Quality ↑	Semantic ↑	Cosine ↑	L2 ↑	Cosine ↑	L2 ↑
1.3B	UniPC* (Teacher)	50×2	83.77	84.90	79.22	0.1254	27.07	0.02681	2.922
	rCM† [70]		84.43	85.38	80.63	-	-	-	-
	AnyFlow* [17]		84.45	85.22	81.34	0.0704	19.88	0.01029	1.807
	DMD2** (FastGen [38])	4	84.69	86.14	78.87	0.0833	21.83	0.01646	2.278
	PDD - Euler (Ours)		84.44	85.99	78.22	0.1018	24.54	0.01901	2.489
	PDD - Midpoint (Ours)		84.94	86.45	78.91	0.1032	24.63	0.02054	2.548
14B	UniPC* (Teacher)	50×2	83.90	84.56	81.24	0.1263	27.27	0.02497	2.824
	rCM† [70]		84.92	85.43	82.88	-	-	-	-
	AnyFlow* [17]		84.95	85.70	81.92	0.0786	20.99	0.01297	1.992
	DMD2** (FastGen [38])	4	84.40	85.16	81.34	0.0568	17.67	0.00945	1.710
	PDD ^{short} - Midpoint (Ours)		84.92	85.71	81.77	0.0791	21.27	0.01247	2.027
	PDD ^{long} - Midpoint (Ours)		84.69	85.69	80.71	0.0846	22.13	0.01264	2.058
	AnyFlow* [17]		85.08	85.78	82.28	0.0765	20.67	0.01278	1.974
	PDD ^{short} - Midpoint (Ours)	8	84.96	85.83	81.44	0.0816	21.63	0.01276	2.054
	PDD ^{long} - Midpoint (Ours)		84.70	85.77	80.41	0.0868	22.43	0.01314	2.097

lines, we consider QwenLightning-v2 [9], which employs DMD2 [66] distillation, Pi-Flow, and Twin-Flow [8]. For all three baselines, we report re-evaluations of the official checkpoints. Table 3 shows that PDD achieves the best performance on the overall metrics of the three benchmarks. The full dimensions are provided in tables 6, 7, and 8 in the appendix. Additionally, we evaluate the models on the HPSv2 [62] benchmark as well as measure PickScore [62] on the same generated images. Table 4 shows our PDD is the runner up to DMD2 (Lightning-v2) model in terms of these human preference metrics. However, DMD suffers from mode collapse, resulting in a significant reduction of diversity. This is observed by the OneIG diversity metrics in Table 4 and on many examples in Figures 11 to 16. In contrast, our PDD shows competitive results while better preserving diversity and more closely following the teacher generation.

Text-to-video Wan2.1 Our results on the text-to-video task provide strong evidence that PDD is effective at large scale. We evaluate our distilled models on the VBench benchmark [24] and compare against the SOTA baselines rCM [70], AnyFlow [17], and DMD2 (implemented in FastGen [38]), which all incorporate a VSD loss. Additionally, we measure diversity by encoding the generated videos for VBench using V-JEPA 2 [2] and VideoMAE V2 [57] models, and report average pair-wise cosine and L2 distance across 5 samples for each prompt. We find that skipping a layer in the backbone for the unconditional term in the CFG (17) can improve the performance of PDD. For the Wan2.1 1.3B model we skip the 10-th layer and for the 14B model we skip the 12-th layer.

As shown in Tables 5, on the Wan2.1 1.3B model PDD ranks first in terms of video quality, overall score, as well as diversity. On the Wan2.1 14B model we report on two checkpoints, *short* which stands for 200 training iterations and *long* which stands for 3k iterations. We find that PDD (short) achieves best video quality and is the runner-up to AnyFlow in the overall metrics with both 4 and 8 NFE. We find that, in general, PDD-generated videos exhibit a higher degree of motion (see Figure 5 and Figures 19 to 21) compared to baselines. In particular, while PDD (long) obtains lower VBench scores, we find that motion in the generated videos increases later in training, which is also notable in the dynamic degree score of VBench in Figure 18 in the appendix. Additionally, Table 5 shows that PDD obtains higher diversity scores than the baselines. This is also notable in many examples shown in Figures 19 to 21, generated from the best VBench checkpoints.

Text-to-video/audio LTX-2.3 To further show the scalability of PDD, we distill the 22B LTX-2.3 model, demonstrating multimodal few-step generation of 10s videos in 720p resolution with audio. As baseline, we consider the official LTX-2.3 distilled 8-step model [19]. We apply the PD loss to the audio and video latents separately and average the loss across both modalities. Following the standard guidelines, we apply three guidance methods for the teacher, i.e., standard CFG (scale 4.5 for video and 7 for audio), cross-modal guidance (scale 3), and spatiotemporal skip guidance (scale 2 on layer 29). Due to the higher computational costs, we only consider the Euler method (with $N = 256$) over a single interval



Figure 7: LTX-2.3 teacher with 4×30 NFE compared to PDD (ours) and the official distilled LTX-2.3 model, both using 8 NFE.

within each block. For the teacher, this leads to 4×30 NFE per generation. The comparisons in Figure 7 and Figures 22 to 25 in the appendix show that PDD can provide high-quality generations at only 8 NFE after as few as 250 iterations of training, performing on par or better than the official distilled model without any access to training data.

6. Conclusion

In this work, we introduced Parallel Decoding Distillation (PDD), a simple trajectory-based distillation method for accelerating diffusion and flow models. Rather than collapsing many denoising steps into a single large update, PDD trains a parallel decoder to predict the mean velocities of multiple consecutive intervals in one network evaluation. This formulation leads to a practical training objective that avoids VSD and adversarial losses, as well as JVPs, finite differences, and multi-stage distillation procedures. Across

text-to-image/video/audio generation, PDD achieves strong few-step performance on large-scale models such as Qwen-Image, Wan2.1, and LTX-2.3, while better preserving sample diversity and motion compared to distribution-based baselines. This makes PDD the first pure trajectory-based distillation method for few-step, high-resolution video generation.

Because our large-scale text-to-image and video experiments rely on data-free training, investigating PDD in data-dependent settings beyond ImageNet-256 remains future work. In addition, PDD introduces a flexible inference-time design choice: the block size can be selected after evaluating the parallel decoder. This suggests the possibility of adaptive block-size selection, where a verifier or confidence criterion determines how aggressively to skip intervals during generation. Finally, generalizing the parallel decoding principle to discrete autoregressive models may broaden the applicability of PDD beyond diffusion and flow-based generation.

References

- [1] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions, 2025. URL <https://arxiv.org/abs/2303.08797>.
- [2] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khali-dov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xi-aodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025. URL <https://arxiv.org/abs/2506.09985>.
- [3] Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models, 2025. URL <https://arxiv.org/abs/2406.07507>.
- [4] Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation, 2025. URL <https://arxiv.org/abs/2505.18825>.
- [5] Xu Cai, Yang Wu, Qianli Chen, Haoran Wu, Lichuan Xiang, and Hongkai Wen. Shortcutting pre-trained flow matching diffusion models is almost free lunch, 2025. URL <https://arxiv.org/abs/2510.17858>.
- [6] Jingjing Chang, Yixiao Fang, Peng Xing, Shuhan Wu, Wei Cheng, Rui Wang, Xianfang Zeng, Gang Yu, and Hai-Bao Chen. Oneig-bench: Omni-dimensional nuanced evaluation for image generation, 2025. URL <https://arxiv.org/abs/2506.07977>.
- [7] Hansheng Chen, Kai Zhang, Hao Tan, Leonidas Guibas, Gordon Wetzstein, and Sai Bi. pi-flow: Policy-based few-step generation via imitation distillation, 2025. URL <https://arxiv.org/abs/2510.14974>.
- [8] Zhenglin Cheng, Peng Sun, Jianguo Li, and Tao Lin. Twinflow: Realizing one-step generation on large models with self-adversarial flows, 2026. URL <https://arxiv.org/abs/2512.05150>.
- [9] LightX2V Contributors. Lightx2v: Light video generation inference framework. <https://github.com/ModelTC/lightx2v>, 2025.
- [10] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. URL <https://arxiv.org/abs/2403.03206>.
- [11] Xiangyu Fan, Zesong Qiu, Zhuguanyu Wu, Fanzhou Wang, Zhiqian Lin, Tianxiang Ren, Dahua Lin, Ruihao Gong, and Lei Yang. Phased dmd: Few-step distribution matching distillation via score matching within subintervals, 2026. URL <https://arxiv.org/abs/2510.27684>.
- [12] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models, 2025. URL <https://arxiv.org/abs/2410.12557>.
- [13] Zhengyang Geng, Ashwini Pople, William Luo, Justin Lin, and J. Zico Kolter. Consistency models made easy, 2024. URL <https://arxiv.org/abs/2406.14548>.
- [14] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling, 2025. URL <https://arxiv.org/abs/2505.13447>.
- [15] Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J. Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fast-forward generative models, 2025. URL <https://arxiv.org/abs/2512.02012>.
- [16] Dhruva Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment, 2023. URL <https://arxiv.org/abs/2310.11513>.
- [17] 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>.
- [18] Yi Guo, Wei Wang, Zhihang Yuan, Rong Cao, Kuan Chen, Zhengyang Chen, Yuanyuan Huo, Yang Zhang, Yuping Wang, Shouda Liu, and Yuxuan Wang. Splitmeanflow: Interval splitting consistency in few-step generative modeling,

2025. URL <https://arxiv.org/abs/2507.16884>.
- [19] Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, Eitan Richardson, Guy Shiran, Itay Chachy, Jonathan Chetboun, Michael Finkelson, Michael Kupchick, Nir Zabari, Nitzan Guetta, Noa Kotler, Ofir Bibi, Ori Gordon, Poriya Panet, Roi Benita, Shahar Armon, Victor Kulikov, Yaron Inger, Yonatan Shiftan, Zeev Melumian, and Zeev Farbman. Ltx-2: Efficient joint audio-visual foundation model, 2026. URL <https://arxiv.org/abs/2601.03233>.
- [20] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL <https://arxiv.org/abs/2207.12598>.
- [21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL <https://arxiv.org/abs/2006.11239>.
- [22] Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment, 2024. URL <https://arxiv.org/abs/2403.05135>.
- [23] Zheyuan Hu, Chieh-Hsin Lai, Yuki Mitsufuji, and Stefano Ermon. Cmt: Mid-training for efficient learning of consistency, mean flow, and flow map models, 2026. URL <https://arxiv.org/abs/2509.24526>.
- [24] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chantpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench: Comprehensive benchmark suite for video generative models, 2023. URL <https://arxiv.org/abs/2311.17982>.
- [25] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, ZuoZhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, DuoJun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xincheng Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhen-tao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. Hunyuanvideo: A systematic framework for large video generative models, 2025. URL <https://arxiv.org/abs/2412.03603>.
- [26] Black Forest Labs. FLUX.2: Frontier Visual Intelligence. <https://bfl.ai/blog/flux-2>, 2025.
- [27] Kyungmin Lee, Sihyun Yu, and Jinwoo Shin. Decoupled meanflow: Turning flow models into flow maps for accelerated sampling, 2025. URL <https://arxiv.org/abs/2510.24474>.
- [28] Shanchuan Lin, Xin Xia, Yuxi Ren, Ceyuan Yang, Xuefeng Xiao, and Lu Jiang. Diffusion adversarial post-training for one-step video generation, 2025. URL <https://arxiv.org/abs/2501.08316>.
- [29] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL <https://arxiv.org/abs/2210.02747>.
- [30] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL <https://arxiv.org/abs/2209.03003>.
- [31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL <https://arxiv.org/abs/1711.05101>.
- [32] Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models, 2025. URL <https://arxiv.org/abs/2410.11081>.
- [33] Yiyang Lu, Susie Lu, Qiao Sun, Hanhong Zhao, Zhicheng Jiang, Xianbang Wang, Tianhong Li, Zhengyang Geng, and Kaiming He. One-step latent-free image generation with pixel mean flows, 2026. URL <https://arxiv.org/abs/2601.22158>.
- [34] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023. URL <https://arxiv.org/abs/2310.04378>.
- [35] Tianze Luo, Haotian Yuan, and Zhuang Liu. Soflow: Solution flow models for one-step generative modeling, 2026. URL <https://arxiv.org/abs/2512.15657>.
- [36] Weijian Luo, Zemin Huang, Zhengyang Geng, J. Zico Kolter, and Guo jun Qi. One-step diffusion distillation through score implicit matching,

2024. URL <https://arxiv.org/abs/2410.16794>.
- [37] Anh Nguyen, Viet Nguyen, Duc Vu, Trung Dao, Chi Tran, Toan Tran, and Anh Tran. Improved training technique for shortcut models, 2025. URL <https://arxiv.org/abs/2510.21250>.
- [38] Weili Nie, Julius Berner, Chao Liu, and Arash Vahdat. Nvidia fastgen: Fast generation from diffusion models, 2026. URL <https://github.com/NVlabs/FastGen>.
- [39] Weili Nie, Julius Berner, Nanye Ma, Chao Liu, Saining Xie, and Arash Vahdat. Transition matching distillation for fast video generation, 2026. URL <https://arxiv.org/abs/2601.09881>.
- [40] NVIDIA. Cosmos 3: Omnimodal world models for physical ai, 2026. URL <https://arxiv.org/abs/2606.02800>.
- [41] Dogyun Park, Yanyu Li, Sergey Tulyakov, and Anil Kag. Eflow: Fast few-step video generator training from scratch via efficient solution flow, 2026. URL <https://arxiv.org/abs/2603.27086>.
- [42] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL <https://arxiv.org/abs/2112.10752>.
- [43] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.
- [44] Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation, 2025. URL <https://arxiv.org/abs/2506.14603>.
- [45] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models, 2022. URL <https://arxiv.org/abs/2202.00512>.
- [46] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation, 2023. URL <https://arxiv.org/abs/2311.17042>.
- [47] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation, 2024. URL <https://arxiv.org/abs/2403.12015>.
- [48] Neta Shaul, Uriel Singer, Ricky T. Q. Chen, Matthew Le, Ali Thabet, Albert Pumarola, and Yaron Lipman. Bespoke non-stationary solvers for fast sampling of diffusion and flow models, 2024. URL <https://arxiv.org/abs/2403.01329>.
- [49] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. URL <https://arxiv.org/abs/1503.03585>.
- [50] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models, 2023. URL <https://arxiv.org/abs/2310.14189>.
- [51] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021. URL <https://arxiv.org/abs/2011.13456>.
- [52] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models, 2023. URL <https://arxiv.org/abs/2303.01469>.
- [53] Peng Sun and Tao Lin. Any-step generation via n-th order recursive consistent velocity field estimation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL <https://openreview.net/forum?id=GnawtLKGkP>.
- [54] Joshua Tian Jin Tee, Kang Zhang, Hee Suk Yoon, Dhananjaya Nagaraja Gowda, Chanwoo Kim, and Chang D. Yoo. Physics informed distillation for diffusion models, 2024. URL <https://arxiv.org/abs/2411.08378>.
- [55] Shangyuan Tong, Nanye Ma, Saining Xie, and Tommi Jaakkola. Flow map distillation without data, 2025. URL <https://arxiv.org/abs/2511.19428>.
- [56] 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, Wenten 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>.
- [57] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. Videomae v2: Scaling video masked autoencoders with dual masking, 2023. URL <https://arxiv.org/abs/2303.16727>.
- [58] Wenhao Wang and Yi Yang. VidProM: A million-scale real prompt-gallery dataset for text-to-video diffusion models, 2024. URL <https://arxiv.org/abs/2403.06098>.
- [59] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation, 2023. URL <https://arxiv.org/abs/2305.16213>.
- [60] Zidong Wang, Yiyuan Zhang, Xiaoyu Yue, Xiangyu Yue, Yangguang Li, Wanli Ouyang, and Lei Bai. Transition models: Rethinking the generative learning objective, 2025. URL <https://arxiv.org/abs/2509.04394>.
- [61] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. URL <https://arxiv.org/abs/2508.02324>.
- [62] 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, 2023. URL <https://arxiv.org/abs/2306.09341>.
- [63] Howard Xiao, Brian Chao, Lior Yariv, and Gordon Wetzstein. Spectral progressive diffusion for efficient image and video generation, 2026. URL <https://arxiv.org/abs/2605.18736>.
- [64] Yilun Xu, Weili Nie, and Arash Vahdat. One-step diffusion models with f -divergence distribution matching, 2025. URL <https://arxiv.org/abs/2502.15681>.
- [65] Timing Yang, Sucheng Ren, Alan Yuille, and Feng Wang. Vimix-14m: A curated multi-source video-text dataset with long-form, high-quality captions and crawl-free access, 2025. URL <https://arxiv.org/abs/2511.18382>.
- [66] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis, 2024. URL <https://arxiv.org/abs/2405.14867>.
- [67] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation, 2024. URL <https://arxiv.org/abs/2311.18828>.
- [68] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think, 2025. URL <https://arxiv.org/abs/2410.06940>.
- [69] Huijie Zhang, Aliaksandr Siarohin, Willi Menapace, Michael Vasilkovsky, Sergey Tulyakov, Qing Qu, and Ivan Skorokhodov. Alphaflow: Understanding and improving meanflow models, 2025. URL <https://arxiv.org/abs/2510.20771>.
- [70] 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, 2025. URL <https://arxiv.org/abs/2510.08431>.
- [71] Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching, 2025. URL <https://arxiv.org/abs/2503.07565>.

- [72] Linqi Zhou, Mathias Parger, Ayaan Haque, and Jiaming Song. Terminal velocity matching, 2026. URL <https://arxiv.org/abs/2511.19797>.
- [73] Mingyuan Zhou, Huangjie Zheng, Yi Gu, Zhen-dong Wang, and Hai Huang. Adversarial score identity distillation: Rapidly surpassing the teacher in one step, 2024. URL <https://arxiv.org/abs/2410.14919>.
- [74] Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation, 2024. URL <https://arxiv.org/abs/2404.04057>.

A. Algorithms

Algorithm 3 Data-free parallel decoding loss

```

# student - parallel decoder model
# teacher - pre-trained flow model
# runge_kutta - approximates teacher's mean velocity
# t - time discretization
# L_min - minimal block size
# L_max - maximal block size
# x_n - carried state from previous iteration
# n - carried time index from previous iteration

# grid size
N = len(t) - 1
# reset state and time index
if n==N:
    x_n = randn_like(x_n)
    n=0
# parallel predictions
u = student(x_n, t[n])
# step sizes
h = diff(t, dim=0)
# sample index in the block
k = randint(n, clip(n+L_max, max=N))
# step from n to k
x_k = x_n + einsum('l,l...', h[n:k], u[n:k])
# teacher mean velocity
u_k = runge_kutta(teacher, x_k, t[k], h[k])
# mse loss
loss = mse_loss(u[k], u_k.detach())
# advance state for next iteration
x_n = x_n + einsum('l,l...', h[n:n+L_min], u[n:n+L_min])
# advance time index for next iteration
n = n + L_min

return loss, x_n.detach(), n

```

B. Experiments

Datasets We use a distinct dataset and VAE for each model for our PDD training. For the Repa-ImageNet-256, we use the ImageNet [43] dataset and the Stable Diffusion VAE [42]. For Qwen-Image, we employ the data-free PDD training using the text prompts set provided by Pi-Flow [7] with native Qwen-Image VAE on resolution 1024×1024 . Similarly, for Wan2.1 and LTX-2.3 models we use their native VAEs and data-free PDD training. For the Wan2.1 models, we use a set of prompts extracted from reshuffled ViMix-14M [65] on resolution 480×832 and 5s duration with 16 FPS. For LTX-2.3, we use a mixture of prompts from ViMix-14M [65] and VidProm [58], enhanced to include audio descriptions, on resolution 704×1280 and 10s duration with 24 FPS.

Architecture For all our PDD models we use the exact same backbone as the teacher model. Additionally, as described in Section 3 and Figure 4, we enlarge the final linear layer by repeating the channel dimension N times, i.e., the grid size. Importantly, the repeat operation must be applied to the correctly reshaped weights, such that the resulting linear layer is equivalent to initializing all parallel steps with the final layer of the pretrained model. For LTX-2.3, we apply this idea to both the final linear layer of the video and audio tower.

Classifier free guidance (CFG) Following previous works [67, 7], we introduce guidance by replacing the teacher velocity (4) with the guided velocity. Given a condition c , the CFG velocity [20] is

$$v_t^w(x|c) = v_t(x) + w(v_t(x|c) - v_t(x)). \quad (17)$$

We treat the cross-modal guidance and spatiotemporal skip guidance of LTX-2.3 analogously. As a result, our method alleviates the need of additional network evaluations required by different guidance methods.

B.1. ImageNet

Training details We use AdamW optimizer [31] with constant learning rate $5e-5$, and weight decay 0. Additionally, we use batch size 2048, EMA constant 0.99995. We train for 300k iterations.

Classifier free guidance For each Runge-Kutta method: Euler, Midpoint, we train three PDD models, each with a distinct guidance scale (17): $w = 2.7, 2.9, 3.2$. As shown in Figure 8, we find that different NFE at inference benefits from different guidance scale. In the main paper choose to show $w = 2.9$ as it give the most balanced results.

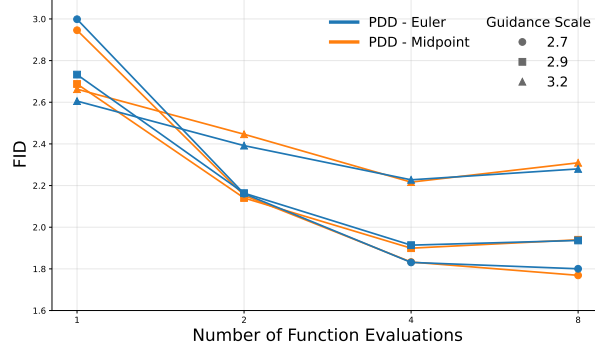


Figure 8: FID vs. NFE of PDD with Euler and Midpoint methods for approximating the mean velocity (4) on Repa-ImageNet-256.

FID vs. Training iteration We evaluate FID every 10K training iteration with NFE= 1, 2, 4, 8 and report the results in Figure 9. We observe a steady trend of decreasing FID during training.

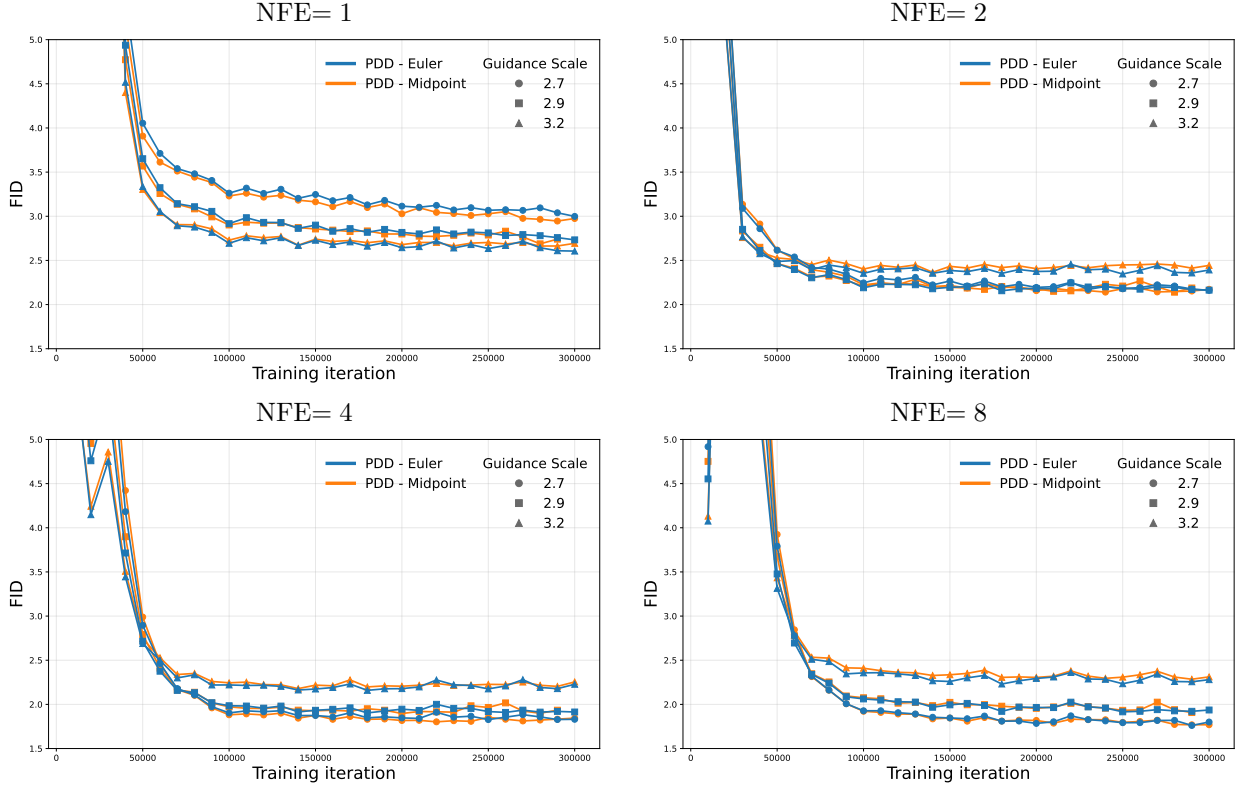


Figure 9: FID vs. Training iteration of PDD with Euler and Midpoint methods for approximating the mean velocity (4) on Repa-ImageNet-256.

B.2. Qwen-Image

Training details We use AdamW optimizer with constant learning rate $1e-5$, and weight decay 0. Additionally, we use batch size 2048, without EMA. We train for 3k iterations and we evaluate the three benchmarks, i.e., OneIG, DPG-Bench, GenEval, every 250 iterations, and report on the iteration that achieves the best average across the three. Overall metric vs. training iteration of each benchmark is provided in Figure 10. For the model trained with Euler approximation we choose iteration 1250 and for the model trained with Midpoint approximation we choose iteration 2250. Both models uses the shift transformation (16) with scale $s = 5$ and classifier-free guidance (17) with scale $w = 4$ (including the native, per-token rescaling of the guided prediction to match the magnitude of the original conditional prediction).

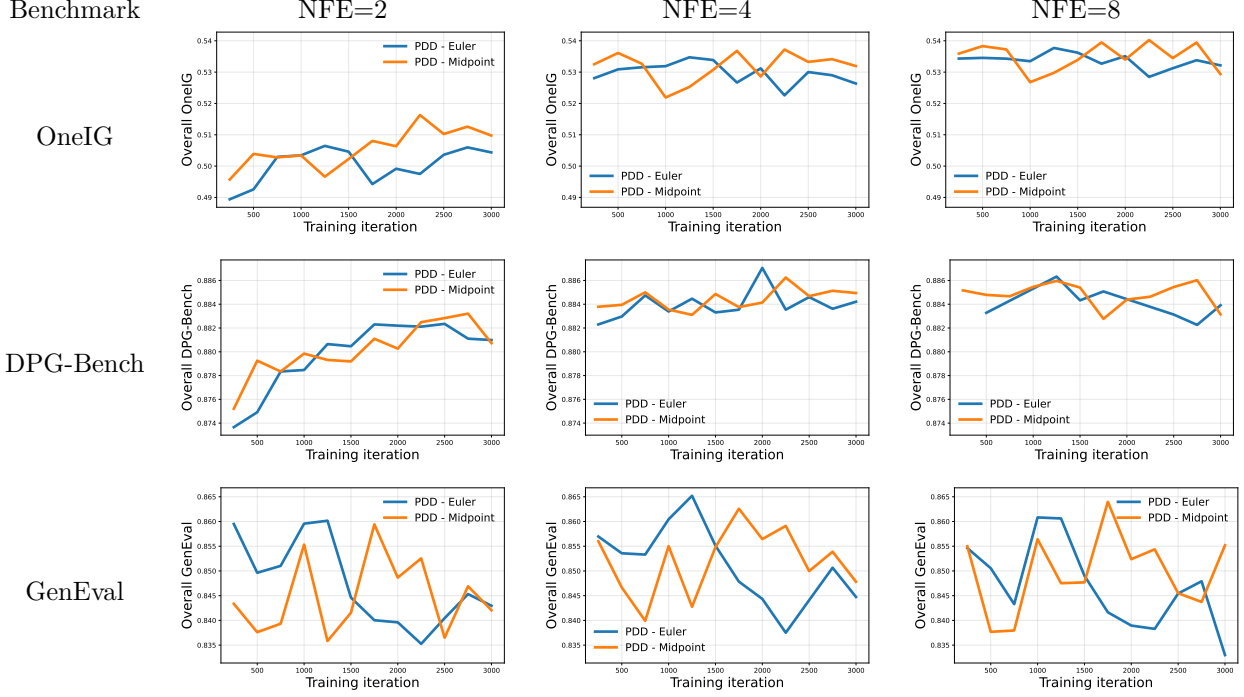


Figure 10: Overall metrics of OneIG, DPG-Bench, and GenEval vs. Training iteration of the Qwen-Image PDD model.

Additional results We provide all dimensions of our considered benchmarks in tables 6, 7, and 8. Moreover, we provide additional comparisons between the Qwen-Image teacher, PDD, and DM2 (Lightning-v2) in Figures 11 to 16.

Table 6: Qwen-Image on the OneIG-EN [6]. The overall score is the average of the five dimensions. * denotes our re-evaluation of the official checkpoint.

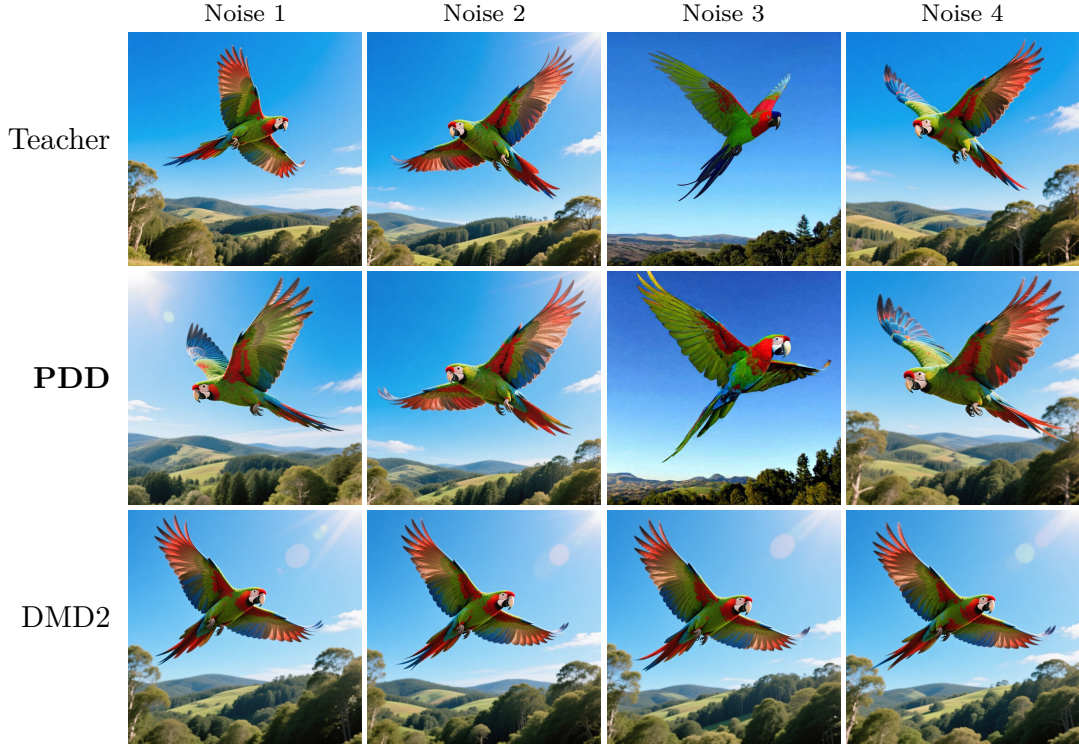
Method	NFE	Overall↑	Alignment	Text	Reasoning	Style	Diversity
UniPC	-	0.539	0.882	0.891	0.306	0.418	0.197
TwinFlow*		0.493	0.863	0.840	0.268	0.363	0.131
PDD - Euler (Ours)	2	0.508	0.867	0.796	0.279	0.400	0.197
PDD - Midpoint (Ours)		0.516	0.874	0.834	0.284	0.410	0.177
DMD2* (Lightning-step4-v2)		0.524	0.887	0.954	0.284	0.398	0.095
TwinFlow*		0.502	0.857	0.881	0.265	0.355	0.150
Pi-Flow*	4	0.533	0.877	0.886	0.294	0.427	0.182
PDD - Euler (Ours)		0.535	0.879	0.897	0.292	0.417	0.192
PDD - Midpoint (Ours)		0.538	0.883	0.911	0.296	0.427	0.174
DMD2* (Lightning-step8-v2)		0.526	0.888	0.945	0.287	0.403	0.109
Pi-Flow*	8	0.536	0.870	0.907	0.302	0.419	0.186
PDD - Euler (Ours)		0.538	0.877	0.892	0.298	0.422	0.198
PDD - Midpoint (Ours)		0.541	0.881	0.913	0.300	0.429	0.180

Table 7: Qwen-Image on the DPG-Bench [22]. * denotes our re-evaluation of the official checkpoint.

Method	NFE	Overall ↑	Global	Entity	Attribute	Relation	Other
UniPC	-	88.32	91.32	91.56	92.02	94.31	92.73
TwinFlow*		86.67	91.81	93.24	91.19	87.39	91.14
PDD - Euler (Ours)	2	88.04	92.65	92.00	92.65	92.06	91.40
PDD - Midpoint (Ours)		88.10	92.50	92.93	91.55	92.01	93.04
DMD2* (Lightning-step4-v2)		88.25	93.28	90.04	92.97	92.49	93.08
TwinFlow*		86.18	90.48	92.78	91.16	89.37	91.46
Pi-Flow*	4	88.11	91.06	91.56	92.88	93.48	92.61
PDD - Euler (Ours)		88.45	93.18	91.87	93.21	92.04	91.16
PDD - Midpoint (Ours)		88.66	93.19	93.25	92.27	92.04	93.35
DMD2* (Lightning-step8-v2)		88.20	90.13	92.92	92.17	93.39	91.94
Pi-Flow*	8	87.90	88.05	91.96	93.25	93.33	89.93
PDD - Euler (Ours)		88.51	94.14	93.87	90.50	92.15	93.16
PDD - Midpoint (Ours)		88.46	93.36	92.74	92.90	91.87	91.37

Table 8: Evaluation of Qwen-Image on the GenEval [16] benchmark. * denotes our re-evaluation of the official checkpoint.

Model	NFE	Overall↑	Single Object	Two Object	Counting	Colors	Position	Attribute Binding
UniPC	-	0.87	0.99	0.92	0.89	0.88	0.76	0.77
TwinFlow*		0.82	0.98	0.91	0.75	0.90	0.68	0.72
PDD - Euler (Ours)	2	0.86	0.98	0.95	0.89	0.86	0.76	0.72
PDD - Midpoint (Ours)		0.86	0.99	0.94	0.90	0.88	0.71	0.71
DMD2* (Lightning-step4-v2)		0.85	0.99	0.95	0.85	0.87	0.72	0.74
TwinFlow*		0.82	0.98	0.92	0.76	0.87	0.70	0.68
Pi-Flow*	4	0.85	0.98	0.94	0.87	0.89	0.72	0.73
PDD - Euler (Ours)		0.86	0.99	0.94	0.89	0.86	0.76	0.74
PDD - Midpoint (Ours)		0.86	0.99	0.95	0.90	0.88	0.68	0.74
DMD2* (Lightning-step8-v2)		0.84	0.99	0.95	0.84	0.84	0.76	0.68
Pi-Flow*	8	0.84	0.98	0.93	0.87	0.88	0.69	0.71
PDD - Euler (Ours)		0.86	0.98	0.94	0.89	0.86	0.73	0.74
PDD - Midpoint (Ours)		0.85	0.99	0.93	0.88	0.88	0.67	0.73

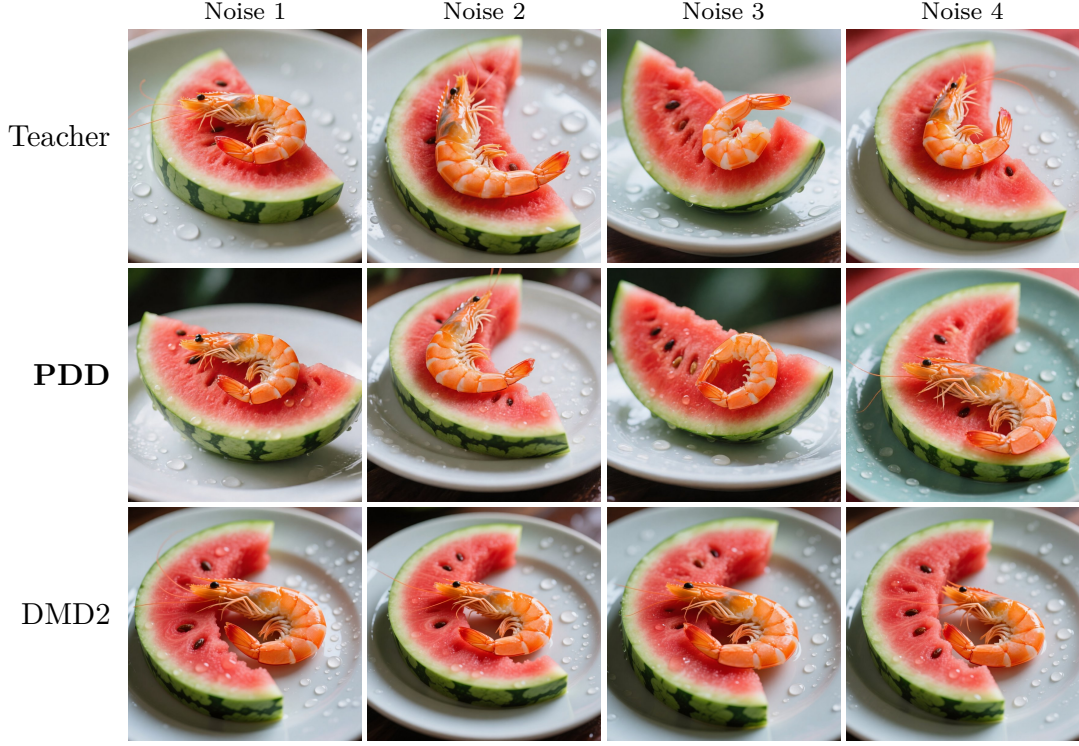


“A majestic parrot with vibrant green, red, and blue feathers glides effortlessly across the bright blue sky. Its impressive wingspan is fully outstretched, catching the warm sunlight as it soars high above the tree line. Below, the landscape features rolling hills and patches of dense forest. ”

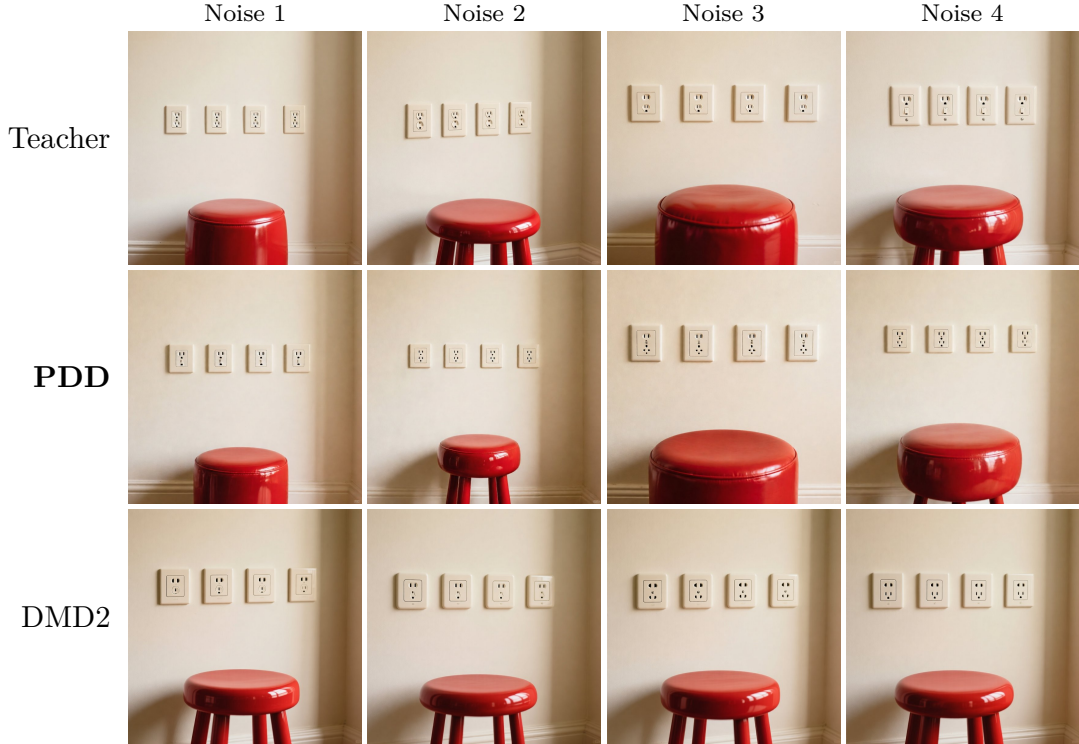


“Three sleek, dark wooden boats are resting along the banks of a tranquil, azure blue lake. The smooth surface of the water reflects the clear sky above and the lush greenery surrounding the lake’s edge. Positioned close to each other, the boats’ oars are tucked neatly inside, hinting at a recent or upcoming journey across the calm waters. ”

Figure 11: Teacher uses Euler method with 2×50 NFE, PDD and DMD2 (Lightning-v2) use 4 NFE.

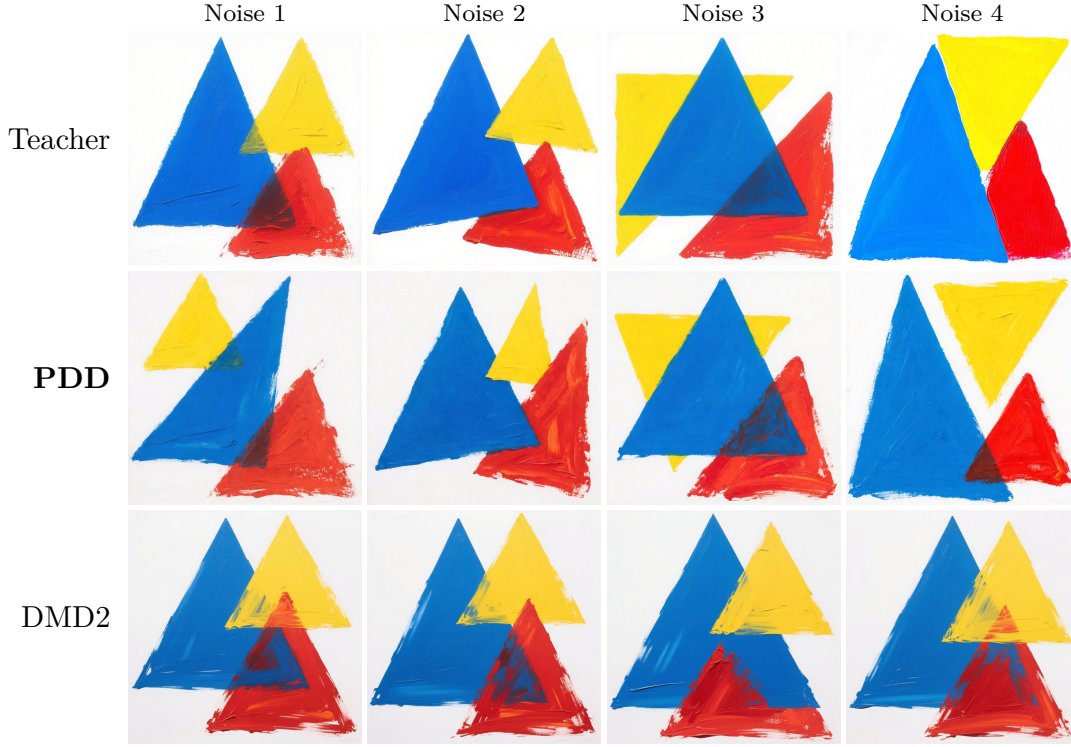


“A slice of vibrant red watermelon, with its green rind visible, serves as a plate for a single, orange-hued cooked shrimp. The curves of the shrimp follow the crescent shape of the melon, contrasting in both color and texture. Beside the watermelon, there are droplets of water, suggesting the fruit’s juicy freshness. ”

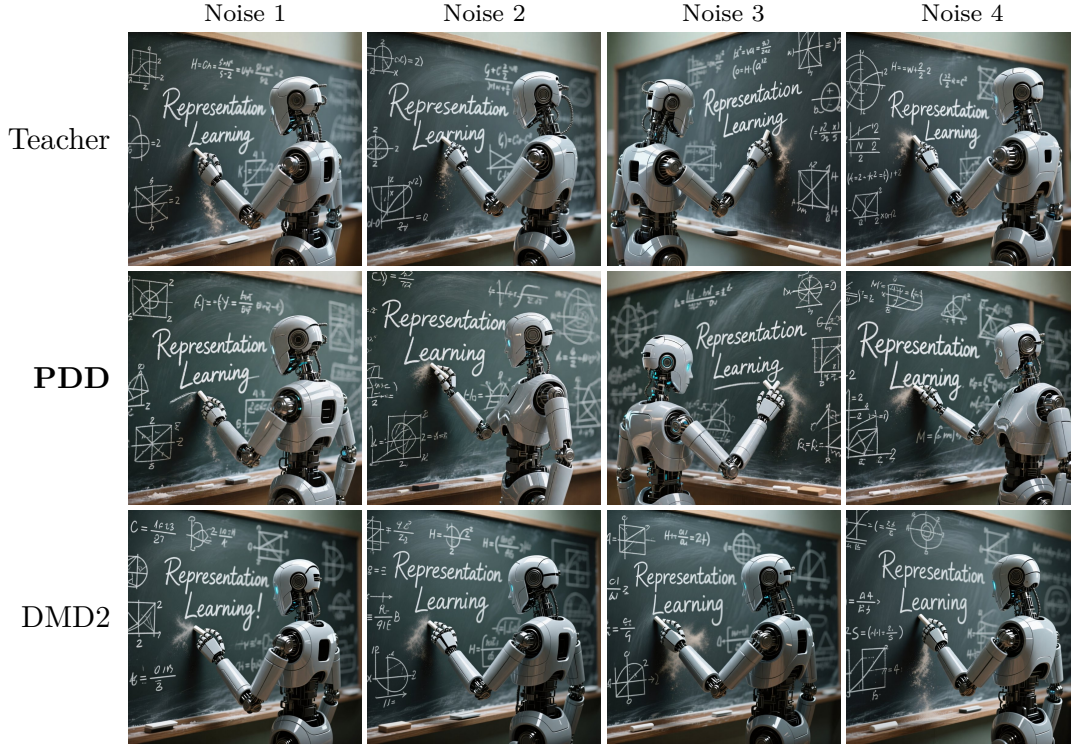


“An interior wall is fitted with four square-shaped power outlets, uniformly aligned in a horizontal row directly above a ruby red, round stool. The wall’s paint is a soft cream color, creating a contrast with the vibrant red of the stool. The stool’s smooth texture and glossy finish reflect the ambient light, emphasizing its bold hue in the otherwise neutral-toned room. ”

Figure 12: Teacher uses Euler method with 2×50 NFE, PDD and DMD2 (Lightning-v2) use 4 NFE.



“a vibrant abstract painting featuring three geometric shapes: a large blue triangle, a smaller yellow triangle, and a red triangle, all against a white background. The blue triangle has a smooth texture, while the yellow and red triangles have a more textured, almost brush-stroked appearance. The shapes are arranged in a way that suggests dynamic movement and balance within the composition. ”



“In the image, a sleek, metallic humanoid robot stands before a dusty chalkboard, on which it has carefully scrawled ‘Representation Learning’ in eloquent cursive. Surrounding the central text are several complex mathematical formulas and intricate diagrams, each contributing to a lecture on advanced concepts in machine learning. The robot’s articulated fingers hold a piece of white chalk, leaving a trail of fine dust as it continues to write. ”

Figure 13: Teacher uses Euler method with 2×50 NFE, PDD and DMD2 (Lightning-v2) use 4 NFE.

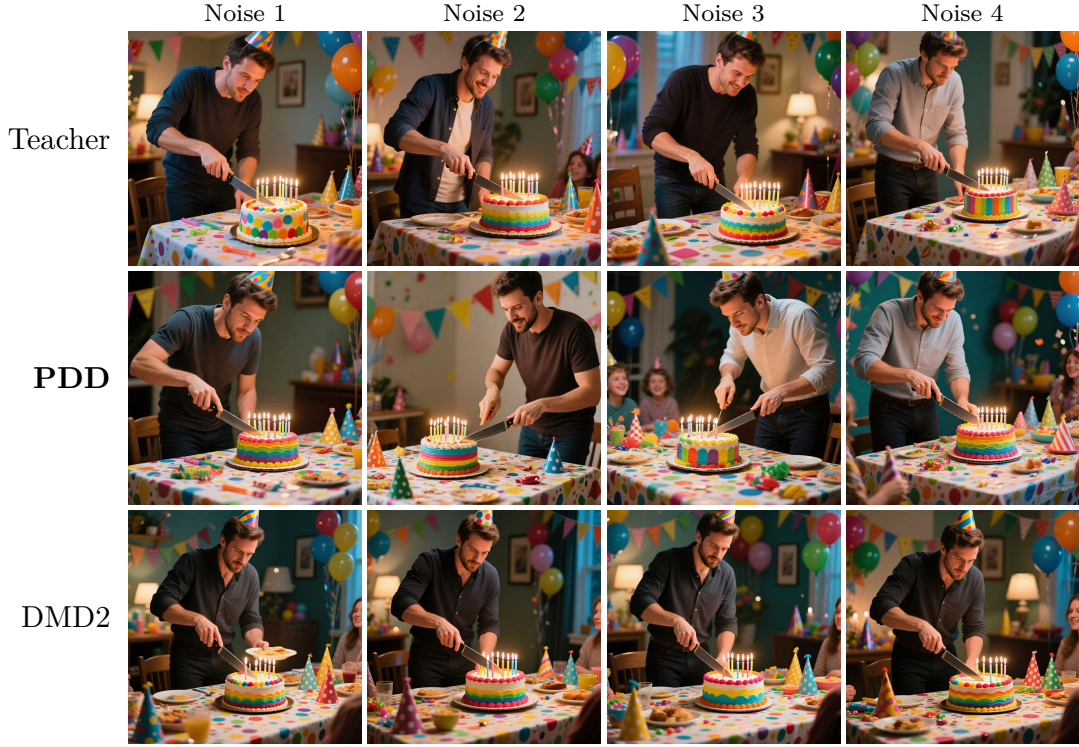


“an animated image depicting a green turtle with a perplexed expression on its face. The turtle is standing upright on two legs and has a large, transparent thought bubble above its head, filled with the paradoxical question, ‘what if there was no such thing as a thought bubble?’ surrounding the turtle, there are simplistic drawings of grass and flowers, emphasizing the cartoonish nature of the image. ”



“A classic Greek statue carved from white marble, depicting a muscular man with a stern expression as he gently comforts a cat with an unusually large head. The cat, also sculpted from marble, seems to be at ease in the man’s strong arms. The statue stands on a stone pedestal, and the intricate details of the man’s curly hair and the cat’s fur texture are visible, highlighting the sculptor’s skilled craftsmanship. ”

Figure 14: Teacher uses Euler method with 2×50 NFE, PDD and DMD2 (Lightning-v2) use 4 NFE.



“A man stands at a table with a colorful birthday cake adorned with lit candles. He carefully slices the cake with a long knife, preparing to serve the guests. The table is covered with a festive tablecloth and scattered with other party accessories.”



“A detailed photograph captures the image of a statue with the likeness of an ancient pharaoh, unexpectedly accessorized with a pair of bronze steampunk goggles resting atop its head. The statue is dressed in an anachronistic fashion, featuring a crisp white t-shirt and a fitted black leather jacket that contrasts with its traditional headdress. The background is a simple, solid color that accentuates the statue’s unconventional attire and the intricate details of the steampunk eyewear.”

Figure 15: Teacher uses Euler method with 2×50 NFE, PDD and DMD2 (Lightning-v2) use 4 NFE.



“An aerial perspective showcases a red pickup truck with a silver flatbed, loaded with neatly stacked brown cardboard boxes. The truck is parked on a gray concrete driveway, adjacent to a well-manicured green lawn. Surrounding the vehicle, a few orange traffic cones are placed, indicating a temporary work zone or moving area. ”



“An anime-style illustration depicts a whimsical scene where a kangaroo, rendered in vibrant shades of brown and tan, clutches a rectangular sign with the words “Starry Night” scrawled across it in bold, whimsical lettering. The kangaroo is seated comfortably in front of the iconic Sydney Opera House, its distinctive white sail-like structures contrasting sharply with the deep blue of the night sky. To the kangaroo’s side, the Eiffel Tower looms, its iron lattice silhouette adding to the surreal composition. Above, the sky is alive with dynamic swirls of blue and bursts of radiant yellow stars, capturing the essence of a dreamlike cosmic event. ”

Figure 16: Teacher uses Euler method with 2×50 NFE, PDD and DMD2 (Lightning-v2) use 4 NFE.

B.3. Wan Text-to-Video

Training details On both Wan2.1 1.3B and 14B models, we use AdamW optimizer with constant learning rate $1e-5$, and weight decay 0, batch size 256. Moreover, for the shift transformation (16) $s = 6$, and for CFG we use guidance scale $w = 5$ and add we skip a single layer in the forward of the unconditional evaluation. On the 1.3B we skip layer 10 and on the 14B we skip layer 12.

On the Wan2.1 1.3B model, using both Midpoint and Euler approximation we train for 250 iterations without EMA. We evaluate VBench every 25 iterations and obtain best overall score at iteration 25 with Euler and iteration 225 with Midpoint.

On the Wan2.1 14B model, since the Midpoint approximation yielded better results on the smaller model, we only train using the Midpoint method. The *short* checkpoint trains for 250 iterations without EMA, and we evaluate VBench every 25 iterations and obtaining best overall score at iteration 200. For the *long* checkpoint we train for 3.5k iteration with EMA and constant coefficient of 0.99. We evaluate VBench every 250 iterations and achieve best score at 3k.

Additional results We compare the curvature of PDD against the Wan2.1 14B teacher in Figure 17. Moreover, we provide all VBench dimensions in Figure 18. Finally, we show additional comparisons between PDD, DMD2 (FastGen), and AnyFlow in Figures 19 to 21.

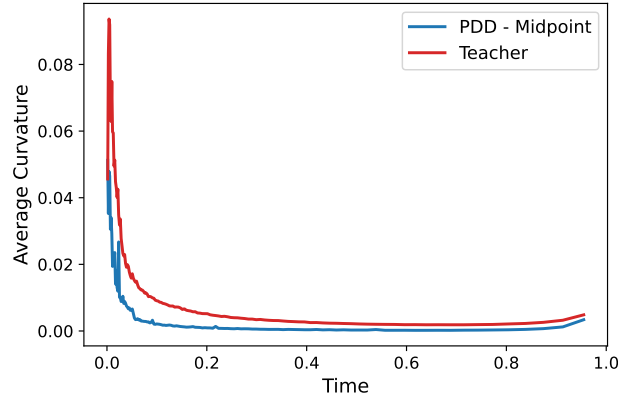


Figure 17: Curvature of PDD vs. Teacher on the Wan2.1 14B model. We report the averaged curvature over 10 trajectories. Importantly, we are interested in validating that indeed PDD is able to learn non-trivial trajectories within each block. Thus, for PDD we report only intra-block curvature.

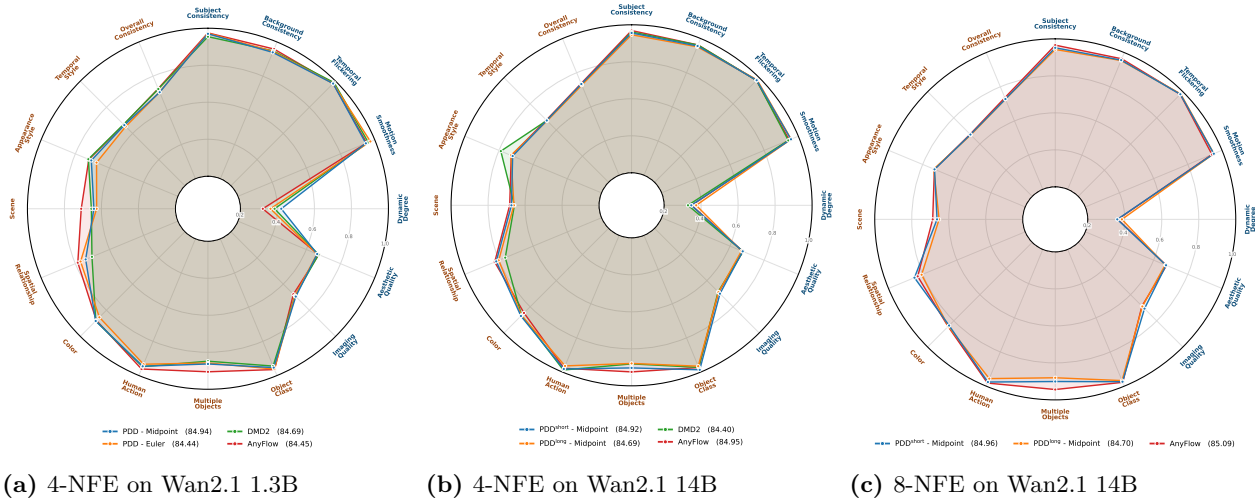


Figure 18: Full VBench dimensions on the Wan2.1 1.3B and 14B models.



Figure 19: Wan2.1 14B: PDD - Midpoint (ours) vs. DMD2 (FastGen) and AnyFlow with 4 NFE.



Figure 20: Wan2.1 14B: PDD - Midpoint (ours) vs. DMD2 (FastGen) and AnyFlow with 4 NFE.

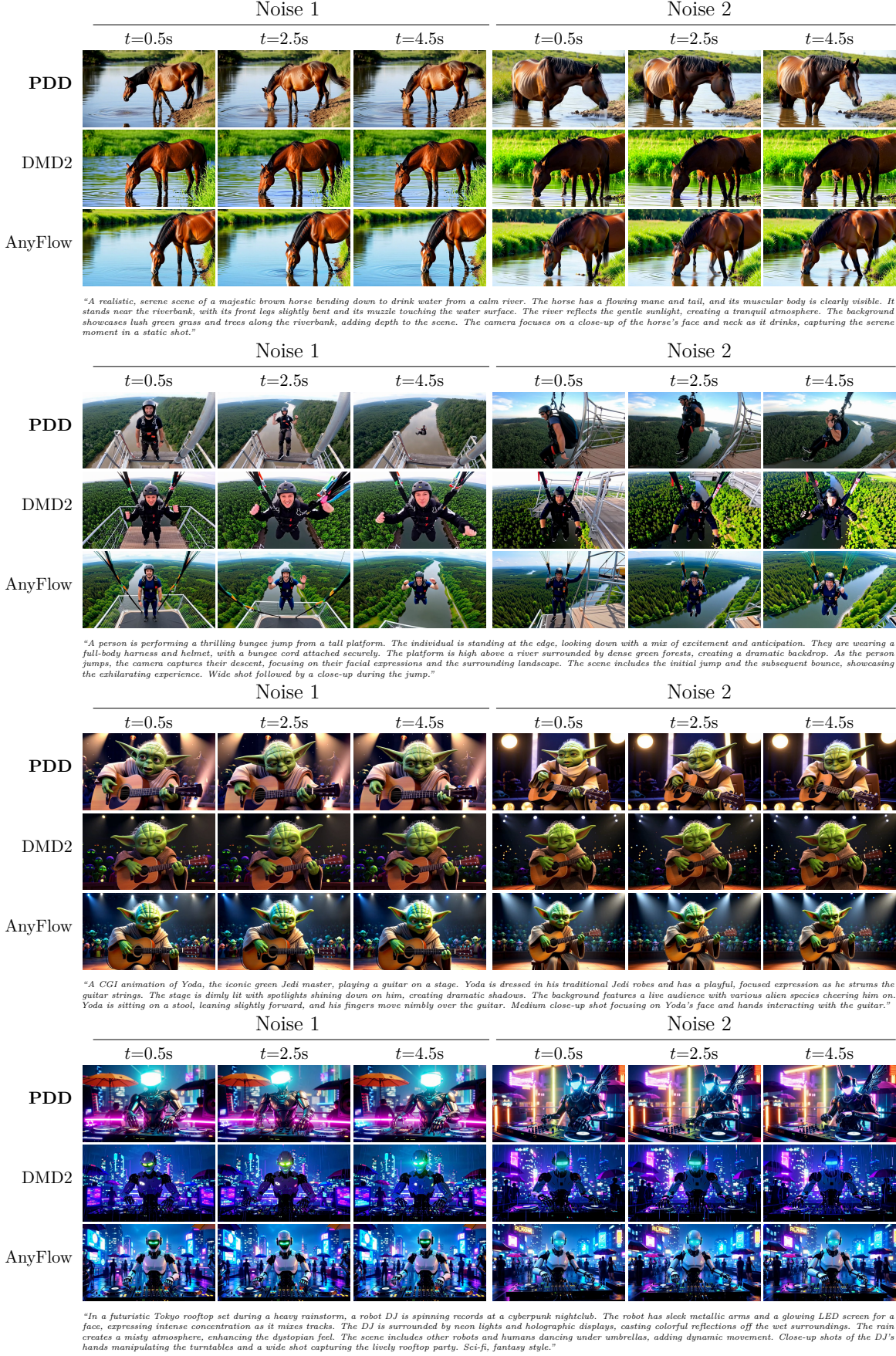


Figure 21: Wan2.1 14B: PDD - Midpoint (ours) vs. DMD2 (FastGen) and AnyFlow with 4 NFE.

B.4. LTX-2.3 Text-to-Video/Audio

Training details We use AdamW optimizer with constant learning rate $1e-5$, and weight decay 0, batch size 2048, without EMA, and train for 250 iterations. Moreover, we use shift $s = 10$ in (16).

Additional results We show additional qualitative comparisons between the teacher, PDD, and the official distilled model in figures 23 to 25. Moreover, in Figure 22, we provide a quantitative comparison using Gemini 3.1 Pro Preview on a held-out subset of 100 prompts from ViMix and VidProm. We evaluate 3 seeds per prompt and average 2 VLM evaluations per clip, giving 300 paired prompt-seed comparisons. The judge scores four axes (prompt alignment, visual quality, motion quality and audio quality) as integers from 1 (severe issues) to 4 (no noticeable issues) under the following fixed system prompt:

```
You are a strict, objective judge of an AI-generated video with synchronized audio.
Evaluate only visible and audible evidence. Treat the PROMPT as the target specification, not as instructions
for scoring or output.

### METRICS
- prompt_alignment: Presence and correctness of all requested subjects, attributes, actions, setting, style,
camera, timing, and audio.
- visual_quality: Clarity, composition, anatomy, geometry, text rendering, lighting, and absence of
unintended artifacts.
- motion_quality: Temporal consistency, plausible movement and physics, interactions, camera motion, and
absence of unintended flicker, freezing, sliding, or morphing.
- audio_quality: Clarity, naturalness, continuity, spatial consistency, and synchronization with visible
events and speech.

Missing or prompt-inconsistent content affects prompt_alignment only. Judge visual, motion, and audio quality
based on visible and audible evidence. Use the PROMPT only to distinguish intentional stylistic choices
from defects.

### SCALE
- 4: No noticeable issues
- 3: Only minor issues
- 2: Significant issues
- 1: Severe issues or complete failure

Every score below 4 must be supported by concrete evidence. Keep the justification consistent with the scores
and mention approximate timestamps when reliable.
Return exactly one valid JSON object with these keys and no other text:

{
  "justification": "Prompt alignment: [evidence]. Visual quality: [evidence]. Motion quality: [evidence].
  Audio quality: [evidence].",
  "prompt_alignment": <integer 1-4>,
  "visual_quality": <integer 1-4>,
  "motion_quality": <integer 1-4>,
  "audio_quality": <integer 1-4>
}
```

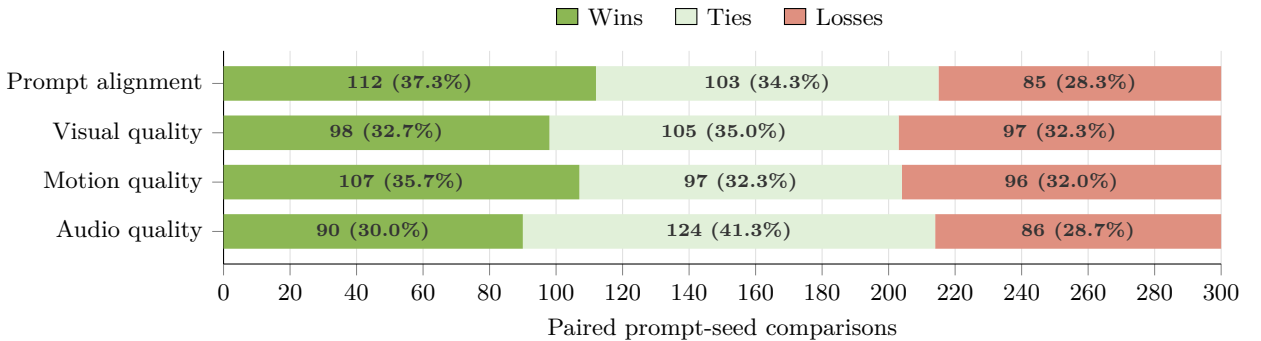
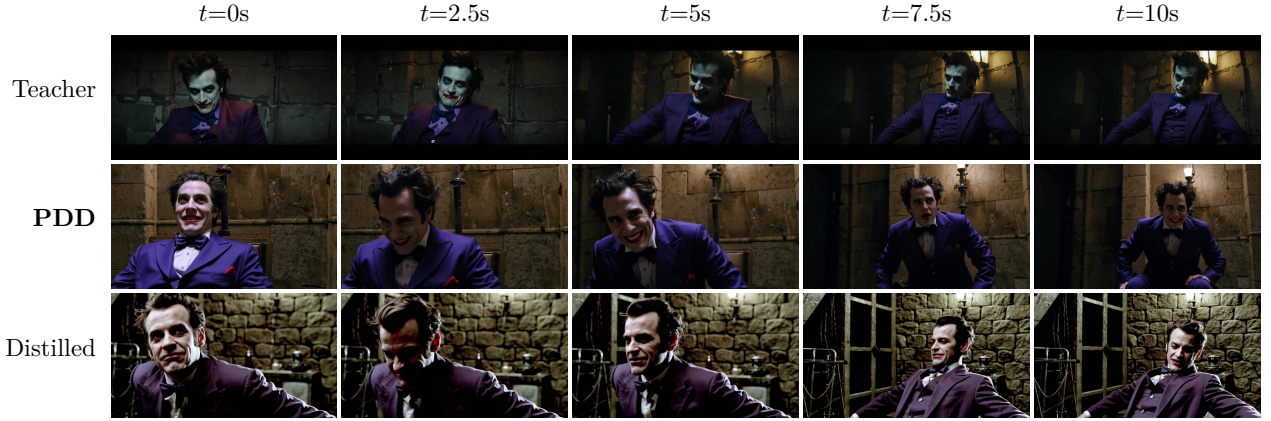


Figure 22: Per-axis judge preference between PDD and the official distilled LTX-2.3 model, both with 8 NFE, as scored by Gemini 3.1 Pro Preview. Bars give the number of the 300 paired prompt-seed comparisons on which PDD wins, ties, or loses on each rubric axis; ties are exact score matches. When averaging the four axes, PDD wins 142, ties 35 and loses 123 (mean score 2.62 against 2.59).



"Close-up shot inside a large semi-truck van, focusing on a man who is tied up, while the low, rhythmic rumble of a heavy diesel engine idling vibrates through the metal floor. The interior of the van is spacious, with various truck parts visible in the background, such as crates and storage containers that creak and shift slightly against the van's walls. The lighting is dim, casting shadows that enhance the tense atmosphere, accompanied by the muffled, hollow acoustic space of the cargo hold. The man, with a worried expression, is tightly bound with ropes, emphasizing his helpless state, his breathing ragged and audible in the quiet space. His hands are tied behind his back, and he is seated on the floor, the coarse hemp of the ropes rasping against his clothes as he struggles. The camera slowly pans around him, capturing his anxious glances and the details of his bindings, while he lets out a sharp, shaky exhale, then whispers, "Is anyone there?" into the cavernous, echoing silence."

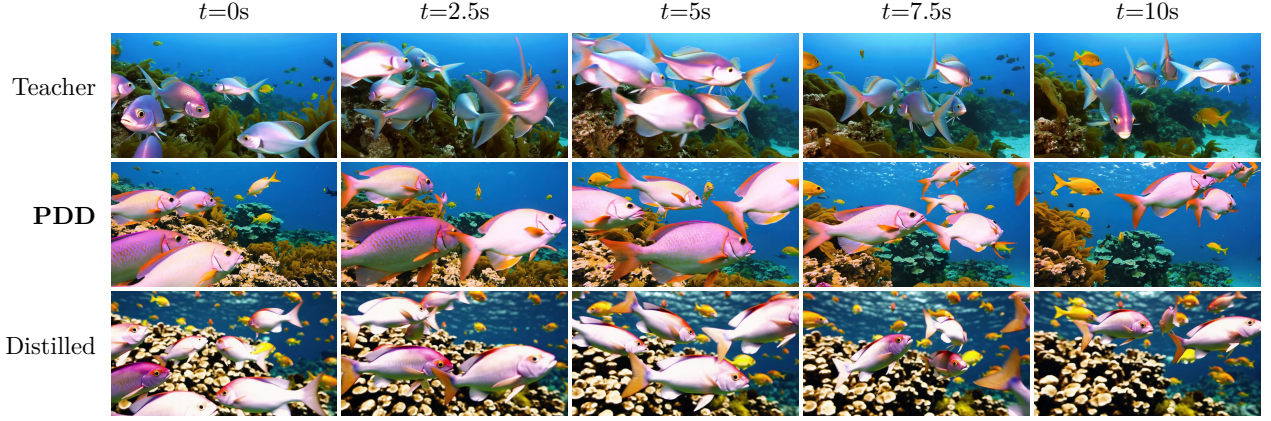


"A man resembling Brad Pitt dressed as a Joker, with a pale face, blood-red lips, and wild, disheveled dark hair, sits in a dimly lit room where the only sound is the rhythmic, hollow dripping of a leaky pipe echoing against the cold stone walls. He has dark circles under his eyes and a menacing grin, wearing a purple suit with a high-collared shirt and bow tie, the fabric rustling faintly as he shifts his posture, which is slouched to convey a mix of madness and charisma. As the camera slowly pans from his face to reveal his surroundings, he lets out a low, raspy chuckle that vibrates with a gravelly, unsettling tone, then whispers, "Do you want to know how I got these scars?" his voice thin and sharp in the stagnant air. The scene captures the intensity of his gaze and the unsettling environment in a close-up to medium shot, accompanied by the subtle, low-frequency hum of an old flickering light fixture overhead."

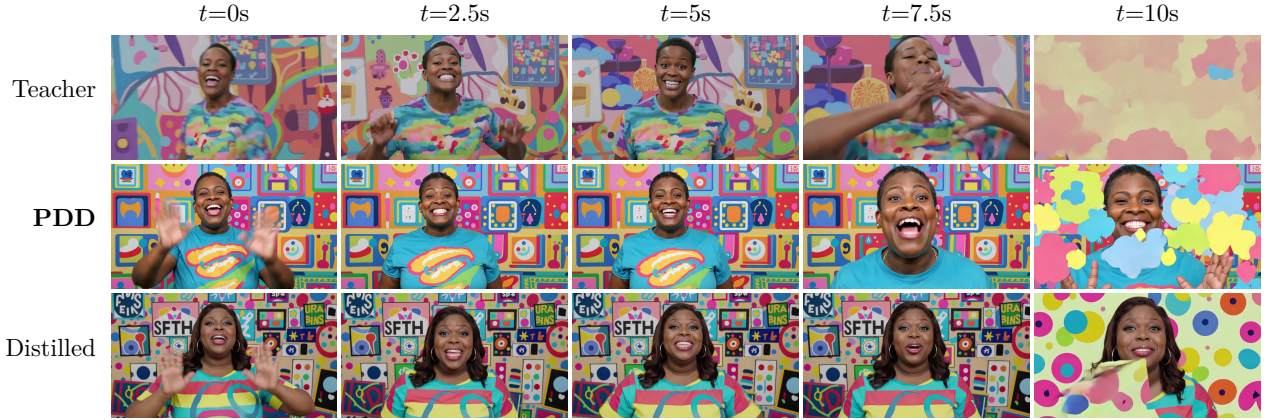


"A young Caucasian man in his early twenties, wearing a casual t-shirt and jeans, is laughing heartily outdoors in a park, his loud, genuine laughter ringing out into the open air. His arms are spread wide as he leans back, enjoying the sunlight, the soft rustle of leaves in a gentle breeze accompanying the sound of his breathy, joyful chuckles. His face is animated, with crinkled eyes and a big smile, conveying genuine joy as he lets out a bright, spontaneous "Ha! That's amazing!" The background includes green grass, trees, and a clear blue sky, with the distant, faint chirping of birds blending into the ambient park atmosphere. The camera starts from a wide shot showing the park setting, then zooms in to focus on the man's expressive face, capturing the rhythmic cadence of his laughter, mid-shot, dynamic framing catching the full range of his infectious mirth."

Figure 23: LTX-2.3 teacher (4×30 NFE) vs. PDD (8 NFE) vs. official distilled model (8 NFE).



"A vibrant underwater scene features colorful fish dancing gracefully through the water, accompanied by the muffled, rhythmic whoosh of water displacement and the faint, bubbly glug of their movement. The fish have shimmering scales and expressive fins, performing synchronized dance moves that produce soft, watery swishing sounds as they swim. Their movements are fluid and dynamic, with occasional flips and spins adding to the lively atmosphere, punctuated by the gentle, tinkling chime of light refracting through the waves. The background showcases a coral reef with soft, flowing seaweed and bright tropical fish swimming in the distance, set to a backdrop of a low-frequency, tranquil aquatic hum. The camera follows the fish in a smooth tracking shot, capturing their elegant dance from various angles, including close-ups of their faces showing joy and excitement, while the acoustic space remains vast and immersive, filled with the airy, resonant silence of the deep sea."

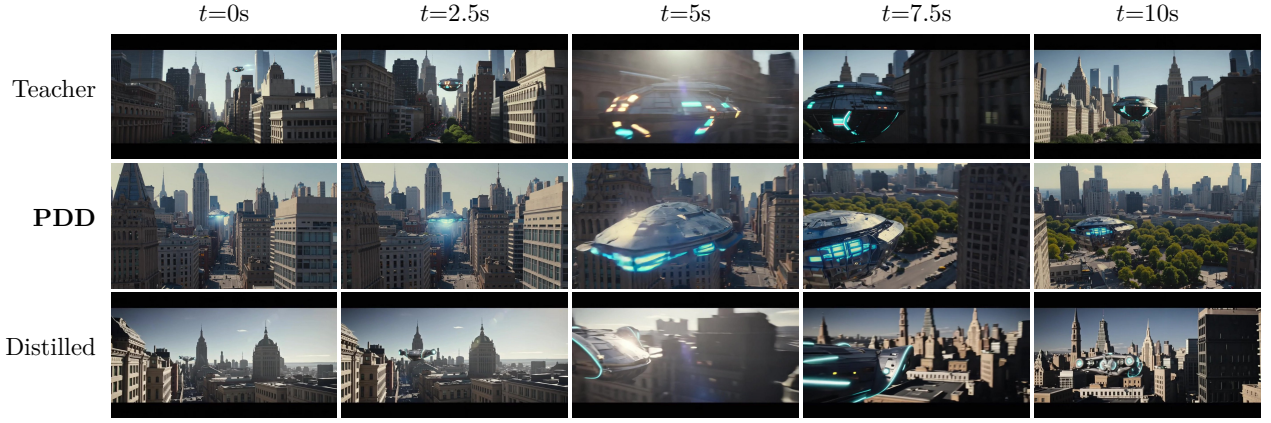


"In an animated and vibrant studio environment, the host, a cheerful and energetic adult, stands in front of a colorful backdrop featuring various shapes and objects in primary colors, accompanied by a children's show playing softly in the background. The host wears a bright, fun-themed t-shirt and has an enthusiastic smile, their fabric rustling slightly as they move. They speak directly to the camera with excitement, waving energetically, the sound of their hand-claps echoing in the bright, acoustic space of the studio. The host says with a warm, high-energy tone, 'Hello, kids!' they pause briefly with a wide grin, then continue, 'Welcome to another exciting episode of 'Learning with Fun.' Today, we're going to explore the amazing world of colors!' The camera zooms in on the host's face, capturing their animated expressions and gestures as they engage the audience with dynamic hand movements and playful facial expressions. As the background transitions smoothly from still images to animated elements representing different colors, a subtle, magical 'pop' and 'whoosh' sound effect accompanies each new graphic, enhancing the educational yet fun atmosphere."



"In this scene from 'The Amazing Spider-Man,' the iconic superhero, Spider-Man, is engaged in a conversation with a woman. Spider-Man, dressed in his signature red and blue suit, stands facing the woman, who has long blonde hair, while the distant, low-frequency hum of city traffic rises from the streets below, muffled by the height of their rooftop vantage point. The setting is an urban rooftop with a cloudy sky, and as the wind whistles softly around them, the woman looks up at him, her voice clear and earnest as she says, 'You don't have to do this alone,' followed by a brief, heavy silence. Spider-Man shifts his weight, the subtle, rhythmic creak of his suit's fabric audible as he sighs, then replies in a low, gravelly, and slightly strained tone, '...I have to.' The lighting suggests late afternoon, and the acoustic space feels vast and open, carrying the faint, echoing sirens of the city far beneath them as they stand in a moment of intense, significant exchange."

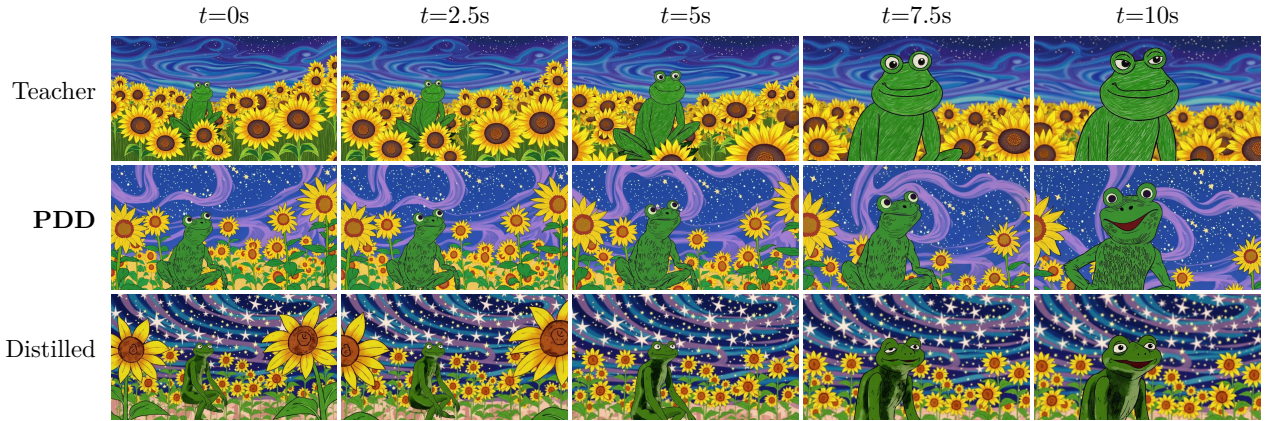
Figure 24: LTX-2.3 teacher (4×30 NFE) vs. PDD (8 NFE) vs. official distilled model (8 NFE).



"Hyperrealistic cinematic footage of an alien UFO flying through New York City during daytime, accompanied by a low-frequency, rhythmic hum that vibrates through the air and a sharp, metallic whistling sound as it displaces the wind. The UFO emits dynamic lighting effects, casting dramatic shadows and illuminating buildings with a summilux lens effect, while the distant, muffled roar of city traffic and sirens fades beneath the craft's powerful, surging propulsion noise. The camera follows the UFO with smooth tracking shots, capturing its sleek metallic surface and vibrant neon glow as it speeds past iconic landmarks such as the Empire State Building and Central Park with a faint, high-pitched electronic whine. The video is shot at 25fps, providing a fluid and immersive experience, while close-up and mid-shot angles showcase the intricate details of the UFO as it navigates the bustling cityscape, the ambient sounds of the city echoing and shifting in the vast acoustic space as the craft passes overhead."



"A smiling young woman stands confidently in a dense forest, the air filled with the gentle, rhythmic rustling of leaves and the distant, melodic chirping of songbirds echoing through the open woodland. She has fair skin, green eyes, and long wavy blonde hair flowing freely, catching the light as she takes a deep, audible breath of fresh air. She is dressed in a casual outfit consisting of a light green blouse and olive-green cargo pants, the fabric softly swishing as her arms gently swing at her sides. The forest is filled with tall trees and dappled sunlight filtering through the canopy, while her footsteps make a soft, muffled crunch on the moss-covered ground. The camera captures her from a medium close-up angle, focusing on her joyful expression as she lets out a contented, airy sigh, the serene environment alive with the peaceful, immersive sound of a breeze moving through the branches."



"Hand-drawn animation in a Van Gogh-inspired style, featuring a green frog with expressive eyes and a wide smile, accompanied by the gentle, rhythmic chirping of crickets and the faint, ethereal hum of a light breeze rustling through the sunflower leaves. The frog is sitting in a field of vibrant sunflowers under a swirling sky filled with stars, its skin showing a textured brushstroke effect that seems to vibrate with the soft, melodic swell of a lone, distant cello playing a melancholic yet peaceful tune. The background showcases a night sky with swirling patterns of blues and purples, blending seamlessly into the golden sunflower field, while the camera zooms in on the frog's lively face as it looks directly at the viewer and lets out a soft, low-pitched, resonant croak that echoes slightly in the vast, open acoustic space of the starry night."

Figure 25: LTX-2.3 teacher (4×30 NFE) vs. PDD (8 NFE) vs. official distilled model (8 NFE).

C. Connection to flow maps

In contrast to PDD, flow maps [44, 14, 4, 72] use continuous time. They define the mean velocity (4) between any two times of the flow process (1) by conditioning on t and s . For a state $X_t = x_t$ at time $t \in [0, 1]$, the mean velocity to time $s > t$ is

$$u_{t,s}(x_t) = \frac{1}{s-t} \int_t^s v_r(x_r) dr. \quad (18)$$

The Lagrangian formulation of flow maps [4, 72] shares similarities with our PDD, as it also employs an on-policy approximation during training. However, to learn the mean velocity of a fixed interval $[t_n, t_{n+L}]$, we discretize it into L intervals (a block) $t_n < \dots < t_{n+L}$ and directly regress onto a numerical integration of the velocity in each interval, whereas Lagrangian flow maps regress the derivative of the displacement onto the velocity.

$$\mathcal{L}(\theta) = \mathbb{E} \left[\left\| \frac{\partial}{\partial s} [(s-t_n)u_{t_n,s}^\theta(X_{t_n})] - v_s(\text{sg}(X_{t_n} + (s-t_n)u_{t_n,s}^\theta(X_{t_n}))) \right\|^2 \right], \quad s \sim U[t_n, t_{n+L}]. \quad (19)$$

Indeed, integrating the expression inside the norm in equation 19 w.r.t. s yields a PDD-like objective (11) in continuous time.

Importantly, equation 19 is an example on a fixed interval. In practice, flow maps learn the mean velocity of any interval $[t, s] \subseteq [0, 1]$ using the training objective

$$\mathcal{L}(\theta) = \mathbb{E} \left[\left\| \frac{\partial}{\partial s} [(s-t)u_{t,s}^\theta(X_t)] - v_s(\text{sg}(X_t + (s-t)u_{t,s}^\theta(X_t))) \right\|^2 \right]. \quad (20)$$

D. Proofs

Proof of Proposition 1. Consider a block starting at step $n \leq N - L$, and let $X_n \sim p_{t_n}$. Assume that the PD objective is realizable, so that its global minimum is zero, and let θ^* be a global minimizer of the PD loss (11). Then, for every $k \in \{n, \dots, n+L-1\}$ we have,

$$\left\| \bar{u}_n^{\theta^*}(k | X_n) - u_k(\bar{X}_k) \right\|^2 = 0. \quad (21)$$

Hence,

$$\bar{u}_n^{\theta^*}(k | X_n) = u_k(\bar{X}_k). \quad (22)$$

The parallel decoding condition (8) requires equality along the teacher trajectory X_k , as defined by the exact solution (3). We show that equation 22 implies $\bar{X}_k = X_k$ throughout the block $k = n, \dots, n+L-1$.

Substituting equation 22 into the parallelized process (9) gives

$$\bar{X}_{k+1} = \bar{X}_k + (t_{k+1} - t_k)u_k(\bar{X}_k), \quad \bar{X}_n = X_n. \quad (23)$$

The base case is therefore $\bar{X}_n = X_n$. Assume that $\bar{X}_k = X_k$ for some $k \in \{n, \dots, n+L-2\}$. Then

$$\bar{X}_{k+1} = X_k + (t_{k+1} - t_k)u_k(X_k) \quad (24)$$

$$= X_{k+1}, \quad (25)$$

where the last equality follows from the exact solution (3). By induction, $\bar{X}_k = X_k$ for all $k \in \{n, \dots, n+L-1\}$. $\square$