Penn Engineering
University of Pennsylvania

GMLR

CIS 6280 · FALL 2026

WORLD MODELS

LECTURE 14

Fast Sampling and
Interactive World Models

Jiatao Gu

University of Pennsylvania · October 22, 2026

Course announcements

Assignment 3	Release: Friday, October 30 Due: November 18
Final project	Instructions and updates on Canvas
Office hours	Thursday, 4:30–5:30 PM, AGH 423

Local updates and finite jumps

Local velocity

$$\frac{dx_\tau}{d\tau} = v(x_\tau, \tau)$$

Repeat evaluations as the point moves.

Finite-time map

$$F_{s,t}(x_s) = x_t$$

Predict the destination for a chosen interval.

We will compare teacher-supervised maps with maps learned from data and local flow relations.

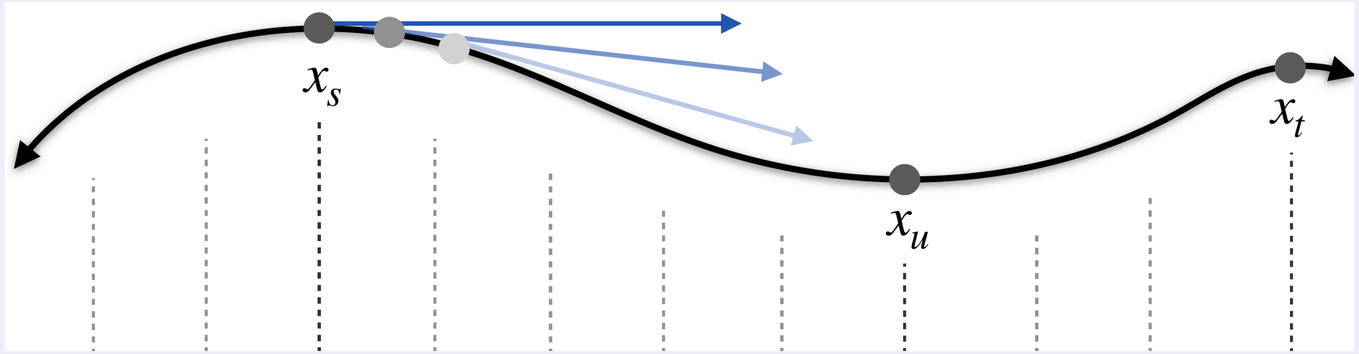
Generation coordinate throughout this section: noise 0 to data 1.

The flow map

$$\frac{dx_\tau}{d\tau} = v(x_\tau, \tau)$$

$$F_{s,t}(x_s) = x_t$$

A finite-time destination along the generative flow.



Original Flow Maps trajectory geometry, excerpt.

Here s, t are generation coordinates, not video positions. $F_{s,t}$ follows the ODE from s to t.

Sampling with a learned map

$$x_0 \sim \mathcal{N}(0, I), \qquad \hat{x}_1 = F_\theta(x_0, 0, 1)$$

$$x_t \approx F_\theta(x_s, s, t)$$

The inputs specify the current point, its generation coordinate, and the requested destination.

$$F_{s,t}(x_s) = F_{r,t}(F_{s,r}(x_s)), \qquad s \le r \le t$$

An exact long jump agrees with two shorter jumps along the same flow.

The teacher velocity

The teacher is an already-trained flow-matching model:

$$\mathcal{L}_{\mathrm{FM}}(w) = \mathbb{E}\,\|v_w(r_\tau, \tau, C) - (y - n)\|^2$$

$r_\tau = (1-\tau)n + \tau y$. n: noise. y: training data.

Freeze the learned weights $w^\star$. Its prediction defines the teacher velocity:

$$v_{\mathrm{T}}(x_\tau, \tau, C) := v_{w^\star}(x_\tau, \tau, C)$$

T: teacher. τ: generation time. C: conditioning, such as a text prompt.

One teacher evaluation supplies a local velocity. A finite-time destination requires integration.

Teacher trajectories as targets

The frozen teacher defines a trajectory. Integrate it to obtain an endpoint target:

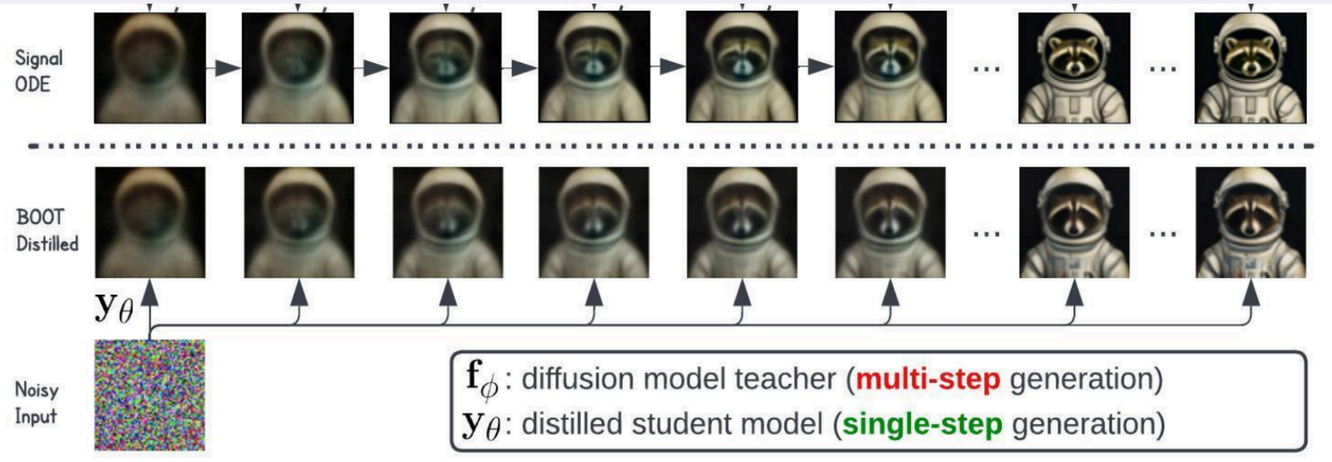
$$x_t^{\mathrm{T}} = \mathrm{ODESolve}(v_{\mathrm{T}}, x_s, s, t)$$

$$\mathcal{L}_{\mathrm{map}} = \mathbb{E}\| F_\theta(x_s, s, t) - x_t^{\mathrm{T}} \|^2$$

Teacher	Integrate the velocity from s to t to obtain the target endpoint.
Student	Predict that same endpoint in one network evaluation.

BOOT predicts a trajectory from noise

Fix the source at noise. Learn to predict any destination along the teacher flow.



BOOT, Figure 3 excerpt. Left to right: toward data. Top: sequential integration. Bottom: direct student predictions.

The student predicts a signal $y_\theta(n, t)$:

$$\hat{x}_t = \alpha_t y_\theta(n, t) + \sigma_t n$$

$$\hat{x}_t \approx F_{0,t}(n)$$

α_t, σ_t: signal and noise weights.

At the data endpoint, $y_\theta(n, 1)$ generates an image in one evaluation.

The same noise identifies the trajectory. The requested time selects the destination.

BOOT in the flow-map framework

Use the same noise n and nearby generation times $0 < s < t \leq 1$.

Same noise

n

→

Student to s

$\hat{x}_s = F_\theta(n, 0, s)$

→

Frozen teacher

Local correction to time t

→

Fixed target

$\tilde{x}_t$

Same noise

n

→

Student to t

$F_\theta(n, 0, t)$

direct prediction →

Train prediction

$\hat{x}_t \quad \approx \quad \tilde{x}_t$

Hold the upper target branch fixed. Update the lower student prediction.

Here $F_\theta(n, 0, t) = \alpha_t y_\theta(n, t) + \sigma_t n$. BOOT matches the corresponding signals y_θ.

No full teacher rollout is needed. A noise-end boundary loss anchors the learned trajectory.

Gu et al., BOOT (ICML 2024)

Why predict a velocity policy?

BOOT predicts a destination

$$F_\theta(n, 0, t) \approx x_t$$

One local teacher velocity does not give a long-jump target. BOOT constructs that target by bootstrapping neighboring times.

π-Flow predicts a velocity rule

$$\pi_s = G_\phi(x_s, s, C)$$

One network call defines a cheap rule that returns a velocity as the sample and time change.

Can one network call support many local updates that directly imitate the teacher?

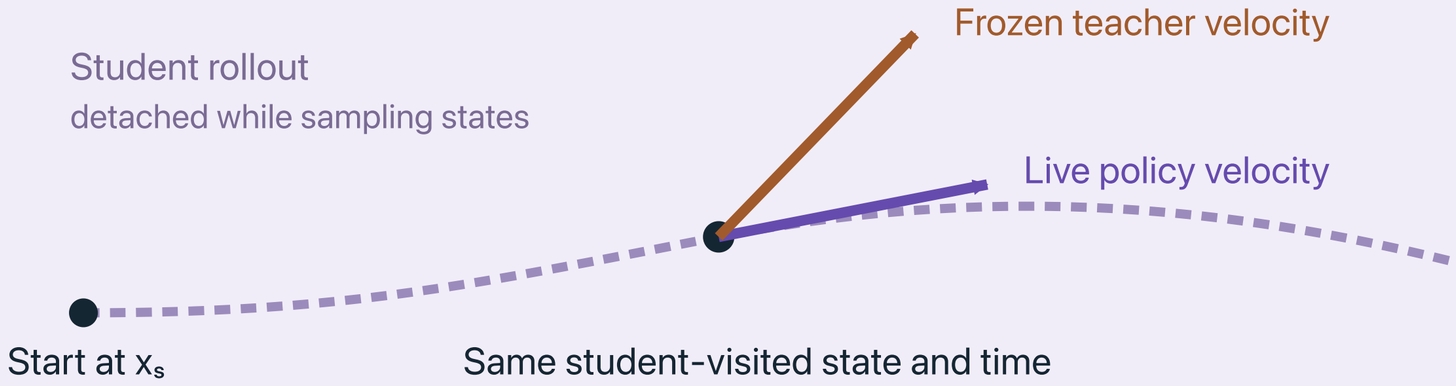
$$\frac{dx_\tau}{d\tau} = \pi_s(x_\tau, \tau), \qquad s \le \tau \le t$$

G_ϕ: student network. π_s: reusable rule returning generation velocities. BOOT already supports one-call generation.

Reuse a cheap rule for integration and supervise its local velocities directly.

π-Flow training on its own trajectory

Predict $\pi_s = G_\phi(x_s, s, C)$. Roll out a detached copy to sample $\bar{x}_\tau$.



$$\mathcal{L}_{\pi\text{-ID}} = \mathbb{E}\left\| \underbrace{\pi_s(\bar{x}_\tau, \tau)}_{\text{live student}} - \underbrace{\mathrm{sg}[v_{\mathrm{T}}(\bar{x}_\tau, \tau, C)]}_{\text{fixed teacher target}} \right\|^2$$

Update G_ϕ through the live policy. Hold the sampled state and teacher target fixed.

sg: stop gradient. The loss compares local velocities at the same input, rather than final images.

BOOT and π-Flow

	BOOT	π-Flow
Network output	A signal prediction for a requested time	Parameters of a reusable velocity rule
Student states	Reconstruct a state from the signal and initial noise	Integrate the current velocity rule
Teacher query	A local denoising correction	A local velocity at the visited state
Inference	Predict the clean endpoint directly	Predict a rule, then integrate it cheaply

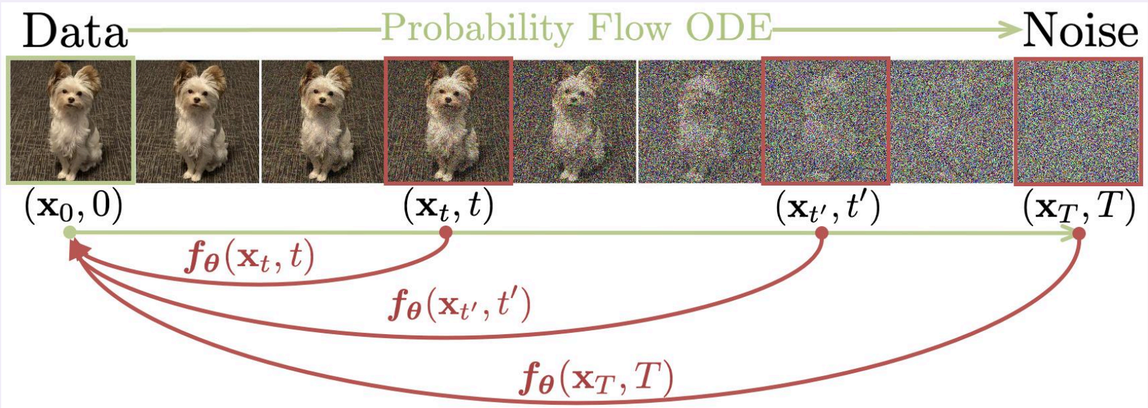
Both use a pretrained teacher to supervise states produced by the student.

π-Flow can cache a time-varying clean estimate (DX) or parameters of a state-dependent Gaussian mixture (GMFlow).

Gu et al., BOOT (ICML 2024) · Chen et al., π-Flow (ICLR 2026)

A common endpoint along a trajectory

Instead of reusing a velocity rule, predict the endpoint from any point on the same trajectory.



Consistency Models, Figure 1. Original time: data 0 to noise T. Our generation coordinate: noise 0 to data 1.

$$f(x_s, s) = F_{s,1}(x_s), \qquad f(x_s, s) = f(x_t, t)$$

$$f(x, 1) = x$$

A consistency model is an endpoint-map construction.

Training the endpoint prediction

$$\tilde{x}_{s+h} = \mathrm{ODEStep}(v_{\mathrm{T}}, x_s, s, s+h)$$

$$\mathcal{L}_{\mathrm{CD}} = \mathbb{E}\, d(f_\theta(x_s, s), \mathrm{sg}[f_{\bar{\theta}}(\tilde{x}_{s+h}, s+h)])$$

Nearby input	Use a teacher step to move along the trajectory
Shared endpoint	Make both endpoint predictions agree
Stable target	sg stops gradients; $\bar{\theta}$ is a target copy

h: a small generation-time step. d: a prediction-distance loss. The boundary remains $f(x, 1) = x$.

Generating with the endpoint map

A boundary-aware parameterization:

$$f_\theta(x, s) = x + (1 - s)g_\theta(x, s), \qquad f_\theta(x, 1) = x$$

One-evaluation sampling:

$$n \sim \mathcal{N}(0, I), \qquad \hat{x}_1 = f_\theta(n, 0)$$

The learned endpoint map turns a fresh noise draw into a generated sample.

Consistency Models include both teacher-based distillation and direct training from data.

Learning without a pretrained teacher

Parameterize displacement as interval length times a learned vector u_θ.

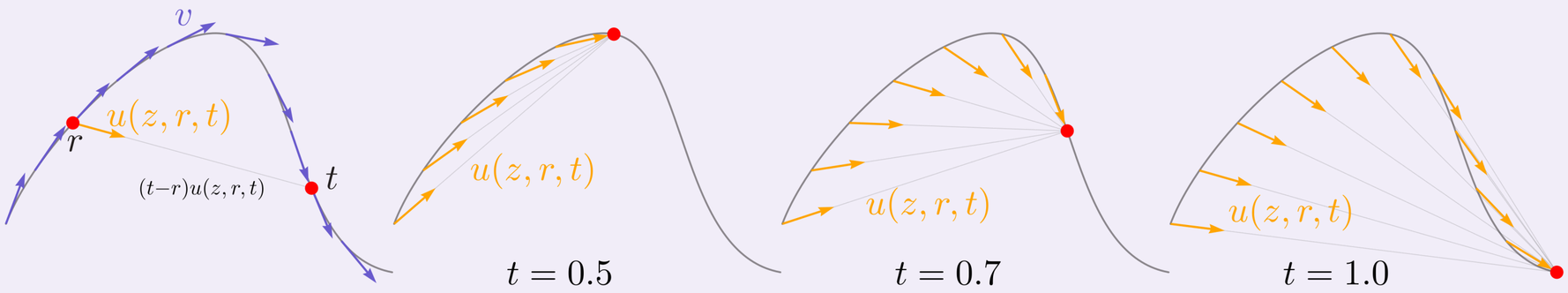
$$F_\theta(x, s, t) = x + (t - s)u_\theta(x, s, t)$$

$$\lim_{t \to s} u(x, s, t) = v(x, s)$$

On a vanishing interval	On a finite interval
Learn the local field from flow-matching data	Enforce compatible map relations or derivative identities
A known regression signal anchors the dynamics	The network extends the signal to larger jumps

Self-consistency needs a data-connected anchor.

Average velocity



MeanFlow, original geometry. The orange chord represents a finite displacement. Original paper uses the opposite time direction.

$$u(x_s, s, t) = \frac{x_t - x_s}{t - s} = \frac{1}{t - s} \int_s^t v(x_\tau, \tau)\, d\tau$$

$$F_{s,t}(x_s) = x_s + (t - s)u(x_s, s, t)$$

The integral follows the generative ODE trajectory. The tangent and interval average can differ.

The MeanFlow identity

Hold the destination t fixed. Move the starting point x_s along the ODE.

$$(t-s)u(x_s, s, t) = \int_s^t v(x_\tau, \tau)\, d\tau$$

$$-u + (t-s)\frac{d}{ds}u = -v(x_s, s)$$

$$\boxed{u = v + (t-s)\frac{d}{ds}u}$$

Product rule on the left. Differentiation of the lower integral limit on the right.

Geng et al., MeanFlow (2025)

The moving-input derivative

$$\frac{d}{ds}u(x_s, s, t) = \partial_s u(x_s, s, t) + J_x u(x_s, s, t)\, v(x_s, s)$$

Term	What changes
$\partial_s u$	The starting-time argument
$J_x u\, v$	The input point moves with velocity v

$J_x u = \partial u / \partial x$. A Jacobian-vector product computes the derivative without constructing that full matrix.

The destination coordinate t stays fixed during this derivative.

Geng et al., MeanFlow (2025)

A velocity target from training data

$$r_s = (1 - s)n + sy, \qquad \delta = y - n$$

$$v(x, s) = \mathbb{E}[\delta \mid r_s = x, s]$$

Flow identity	Constructible training signal
The marginal velocity $v(x, s)$	A sampled noise–image pair gives δ
The exact average-velocity function u	Use the current prediction u_θ to construct a stopped target

The training pair supplies a conditional regression signal for the unknown marginal field.

The MeanFlow training target

$$r_s = (1-s)n + sy, \qquad \delta = y - n$$

$$D_s u_\theta = \partial_s u_\theta + J_x u_\theta \, \delta$$

$$u_{\text{target}} = \delta + (t-s) D_s u_\theta$$

$$\mathcal{L}_{\text{MF}} = \mathbb{E} \| u_\theta(r_s, s, t) - \text{sg}[u_{\text{target}}] \|^2$$

With the current network fixed, the target is linear in δ plus a fixed term. Conditional averaging replaces δ by $v(x, s)$.

$D_s u_\theta$ is evaluated at (r_s, s, t). sg detaches the target.

Geng et al., MeanFlow (2025)

MeanFlow training and sampling

Training

Sample	$y, n, s \le t$
Construct	$r_s = (1-s)n + sy, \quad \delta = y - n$
Differentiate	JVP in direction $(\delta, 1, 0)$
Update	Regress the stopped identity target

Sampling

$$x_0 \sim \mathcal{N}(0, I)$$

$$\hat{x}_1 = x_0 + u_\theta(x_0, 0, 1)$$

One network evaluation predicts the full displacement.

A shorter interval uses $x_t \approx x_s + (t-s)u_\theta(x_s, s, t)$.

The training JVP does not run during sampling.

Geng et al., MeanFlow (2025)

Flow Matching and iMF

Both regress the same velocity target $y - n$. The prediction is parameterized differently.

Training input $x = (1 - s)n + sy$. y: data. n: Gaussian noise.

Ordinary Flow Matching

$$\mathcal{L}_{\text{FM}} = \mathbb{E}\|v_\theta(x, s) - (y - n)\|^2$$

Network output: instantaneous velocity.

Sampling: integrate local updates.

Improved MeanFlow

$$\mathcal{L}_{\text{iMF}} = \mathbb{E}\|\tilde{v}_\theta - (y - n)\|^2$$

Network output: average velocity u_θ.

Sampling: predict a finite jump using u_θ.

$$\tilde{v}_\theta = \underbrace{u_\theta(x, s, t)}_{\text{average-velocity prediction}} - (t - s) \underbrace{\text{sg}[D_s^{\text{pred}} u_\theta]}_{\text{derivative correction}}$$

In iMF, the derivative uses the model's predicted velocity:

$$D_s^{\text{pred}} u_\theta = \partial_s u_\theta + J_x u_\theta \, v_\theta, \qquad v_\theta(x, s) = u_\theta(x, s, s)$$

iMF uses v-loss with u-prediction. At $t = s$, the correction vanishes and the loss reduces to FM.

Matching outputs without fixing pairs

Trajectory-based supervision	Distribution matching
Learn destinations or local relations along a generative flow	The generated population should match the target population
Compare corresponding outputs or relations along a flow	Different noise-to-sample correspondences can be acceptable

A fast generator still needs diverse, plausible samples. It need not reproduce every teacher trajectory.

DMD supplies a distribution-level training signal using denoising models.

The generated distribution

$$z \sim \mathcal{N}(0, I), \qquad x = G_\theta(z, C)$$

G_θ	The student generator. One call produces a clean candidate x.
$q_\theta(x \mid C)$	The distribution of its outputs over fresh z.
$p_{\mathrm{T}}(x \mid C)$	The target distribution represented by the frozen teacher.

C: text or other conditioning. T: teacher. The density of q_θ is generally unavailable.

Yin et al., DMD (CVPR 2024)

Comparing two populations

Student outputs

$$x^{\mathrm{S}} = G_\theta(z, C) \sim q_\theta$$

Gaussian corruption ↓

$$x^{\mathrm{S}}_\tau \sim q_{\theta,\tau}$$

Target outputs

$$x^{\mathrm{T}} \sim p_{\mathrm{T}}$$

The same corruption rule ↓

$$x^{\mathrm{T}}_\tau \sim p_{\mathrm{T},\tau}$$

$$K_\tau(x_\tau \mid x) = \mathcal{N}(x_\tau; \alpha_\tau x, \sigma_\tau^2 I)$$

The comparison is between $q_{\theta,\tau}$ and $p_{\mathrm{T},\tau}$, rather than a clean image and its noisy version.

The score of the corruption kernel

Fix the noise level τ. For now, $p(x)$ can be either clean population.

$$s_p(x_\tau, \tau) := \nabla_{x_\tau} \log p_\tau(x_\tau)$$

$$\log K_\tau(x_\tau \mid x) = \text{const} - \frac{\|x_\tau - \alpha_\tau x\|^2}{2\sigma_\tau^2}$$

$$\nabla_{x_\tau} \log K_\tau(x_\tau \mid x) = \frac{\alpha_\tau x - x_\tau}{\sigma_\tau^2}$$

This derivative assumes a known clean source x. A denoiser must average over its possible sources.

Averaging over possible clean sources

$$p_\tau(x_\tau) = \int p(x) K_\tau(x_\tau \mid x)\, dx$$

$$\begin{aligned}\nabla_{x_\tau} \log p_\tau(x_\tau) &= \frac{\int p(x) \nabla_{x_\tau} K_\tau(x_\tau \mid x)\, dx}{p_\tau(x_\tau)} \\ &= \int \underbrace{\frac{p(x) K_\tau(x_\tau \mid x)}{p_\tau(x_\tau)}}_{p(x \mid x_\tau)} \nabla_{x_\tau} \log K_\tau(x_\tau \mid x)\, dx\end{aligned}$$

$$s_p(x_\tau, \tau) = \frac{\alpha_\tau \mathbb{E}_p[x \mid x_\tau] - x_\tau}{\sigma_\tau^2}$$

Bayes' rule turns the population average into a posterior average over clean sources.

Noise prediction estimates the score

$$\epsilon = \frac{x_\tau - \alpha_\tau x}{\sigma_\tau} \quad \Longrightarrow \quad s_p(x_\tau, \tau) = -\frac{\mathbb{E}_p[\epsilon \mid x_\tau]}{\sigma_\tau}$$

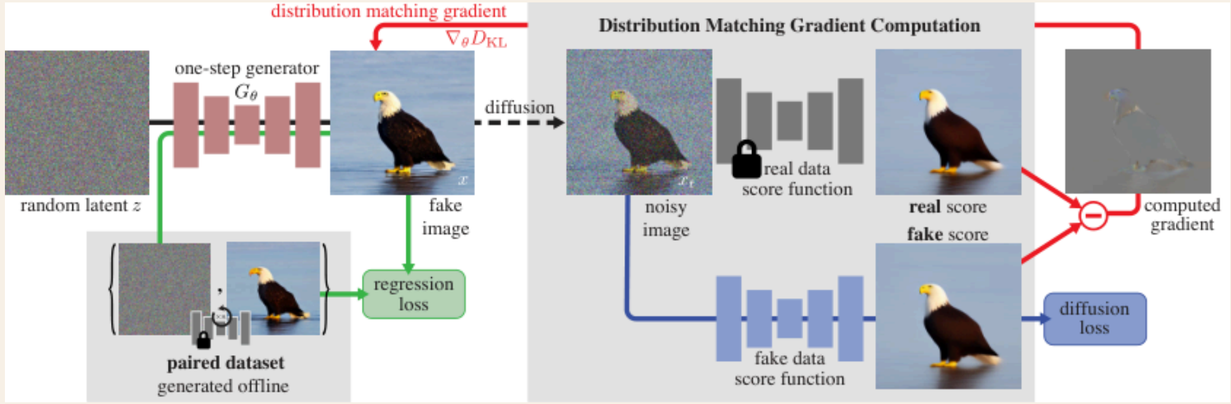
Squared-error noise prediction learns this conditional mean:

$$\hat{\epsilon}_p^* = \arg\min_f \mathbb{E}\|f(x_\tau, \tau) - \epsilon\|^2 = \mathbb{E}_p[\epsilon \mid x_\tau]$$

$$\hat{s}_p(x_\tau, \tau) = -\hat{\epsilon}_p(x_\tau, \tau)/\sigma_\tau$$

Different clean populations give different denoisers and different scores, even at the same noisy input.

Two score models



DMD, original method figure. Read the two score branches first. The original method also includes paired regression.

Frozen teacher	Trainable fake score model
Estimates $s_{\mathrm{T}} = \nabla \log p_{\mathrm{T},\tau}$	Estimates $s_\phi \approx \nabla \log q_{\theta,\tau}$
What the target population looks like	What the current generator population looks like

Both read the same noisy student sample (x_τ, τ, C). The fake model estimates a score, not a real/fake probability.

Why the generated-distribution score matters

Target score

$$s_{\mathrm{T}}(x_\tau, \tau)$$

Indicates where the target density increases locally.

Distribution difference

$$s_\phi(x_\tau, \tau) - s_{\mathrm{T}}(x_\tau, \tau)$$

Compares the current generated density with the target density.

$$q_{\theta,\tau} = p_{\mathrm{T},\tau} \quad \Longrightarrow \quad s_\phi - s_{\mathrm{T}} \approx 0$$

At a match, the distribution correction should vanish, even though the target score need not be zero.

The two terms in reverse KL

Fix one noise level τ and the condition C. Both populations use the same corruption rule.

$$x_\tau = \alpha_\tau G_\theta(z, C) + \sigma_\tau \epsilon \sim q_{\theta,\tau}$$

$$\mathcal{J}_\tau = \mathrm{KL}(q_{\theta,\tau} \| p_{\mathrm{T},\tau}) = \mathbb{E}_{z,\epsilon}\left[\underbrace{\log q_{\theta,\tau}(x_\tau)}_{\text{student term}} - \underbrace{\log p_{\mathrm{T},\tau}(x_\tau)}_{\text{target term}}\right]$$

Target term	Rewards outputs that the target distribution regards as plausible.
Student term	Accounts for how probability spreads across the generator's outputs.

Yin et al., DMD (CVPR 2024)

Differentiating the target term

The teacher distribution stays fixed. Changing θ only moves the sample x_τ.

$$s_{\mathrm{T}}(x_\tau, \tau) = \nabla_{x_\tau} \log p_{\mathrm{T},\tau}(x_\tau)$$

$$\nabla_\theta[-\mathbb{E} \log p_{\mathrm{T},\tau}(x_\tau)] = -\mathbb{E}\left[\left(\frac{\partial x_\tau}{\partial \theta}\right)^\top s_{\mathrm{T}}(x_\tau, \tau)\right]$$

Input derivative	The score tells us how log density changes when the sample moves.
Chain rule	$\partial x_\tau / \partial \theta$ maps a parameter change to a sample change.

This contribution supplies the target score, with a minus sign in the loss gradient.

Differentiating the student term

Here, both the sample and its density change:

$$\nabla_\theta \log q_{\theta,\tau}(x_\tau) = \underbrace{\left(\frac{\partial x_\tau}{\partial \theta}\right)^\top s_q(x_\tau, \tau)}_{\text{sample moves}} + \underbrace{\partial_\theta \log q_{\theta,\tau}(y)|_{y=x_\tau}}_{\text{density changes at fixed input}}$$

$$\begin{aligned}\mathbb{E}_{y\sim q_{\theta,\tau}}[\partial_\theta \log q_{\theta,\tau}(y)] &= \int q_{\theta,\tau}(y)\, \partial_\theta \log q_{\theta,\tau}(y)\, dy \\ &= \int \partial_\theta q_{\theta,\tau}(y)\, dy \\ &= \partial_\theta \int q_{\theta,\tau}(y)\, dy = \partial_\theta 1 = 0\end{aligned}$$

Only the second term cancels in expectation. The student-score contribution remains.

The distribution-matching gradient

$$\frac{\partial x_\tau}{\partial \theta} = \alpha_\tau \frac{\partial G_\theta(z, C)}{\partial \theta}$$

$$\nabla_\theta \mathcal{J} = \mathbb{E}_{\tau,z,\epsilon}\left[w_\tau \alpha_\tau \left(\frac{\partial G_\theta}{\partial \theta}\right)^\top \Big(\underbrace{s_q}_{\text{student term}} - \underbrace{s_{\mathrm{T}}}_{\text{target term}} \Big) \right]$$

Gradient

$$s_q - s_{\mathrm{T}}$$

The expression above differentiates the loss.

Descent direction

$$s_{\mathrm{T}} - s_q$$

Gradient descent reverses that sign.

Both scores read x_τ. $\mathcal{J} = \mathbb{E}_\tau[w_\tau \mathcal{J}_\tau]$ uses fixed noise-level weights $w_\tau \geq 0$.

Updating the generator

Evaluate both scores and hold their outputs fixed during this update:

$$d = \mathrm{sg}[w_\tau \alpha_\tau (s_\phi(x_\tau, \tau) - s_{\mathrm{T}}(x_\tau, \tau))]$$

$$\mathcal{L}_{\text{surrogate}} = d^\top G_\theta(z, C)$$

$$\nabla_\theta \mathcal{L}_{\text{surrogate}} = \left(\frac{\partial G_\theta}{\partial \theta}\right)^\top d$$

sg stops gradients. This scalar implements the desired gradient; its value is not the KL value.

Backpropagate through the student generator, without differentiating through either score network.

Learning the fake score

$$\bar{x} = \mathrm{sg}[G_\theta(z, C)], \qquad x_\tau = \alpha_\tau \bar{x} + \sigma_\tau \epsilon$$

$$\mathcal{L}_{\text{fake}}(\phi) = \mathbb{E}\,\|\hat{\epsilon}_\phi(x_\tau, \tau, C) - \epsilon\|^2$$

$$s_\phi(x_\tau, \tau, C) = -\hat{\epsilon}_\phi(x_\tau, \tau, C)/\sigma_\tau$$

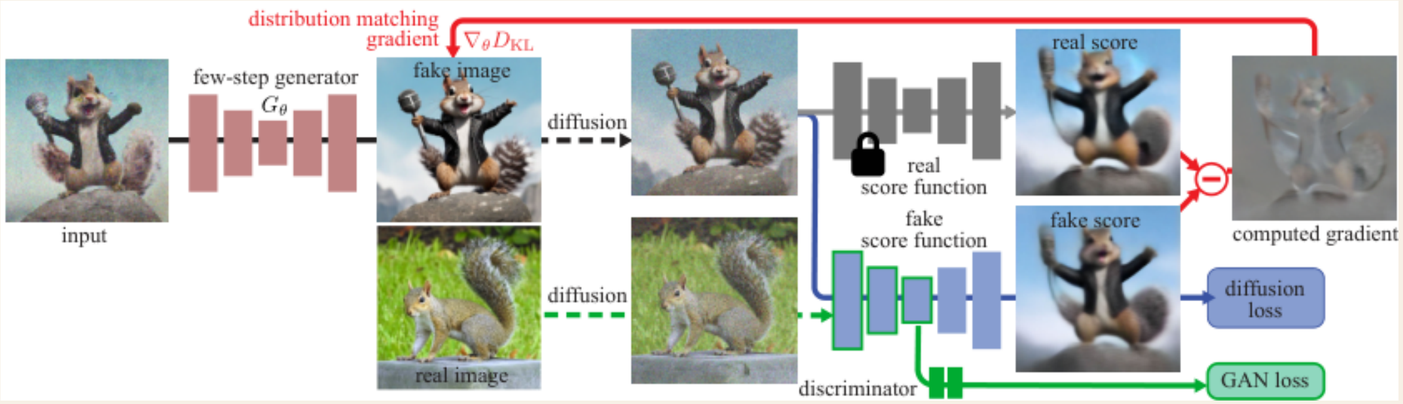
Training images	Fresh outputs from the current generator
Known target	The Gaussian noise ϵ that we just added
Updated weights	Only ϕ, the fake denoising model

The alternating training loop

Update the generator θ	Update the fake score ϕ
Generate a sample and add evaluation noise	Draw fresh samples from the current generator
Query the frozen teacher and fixed fake score	Add known noise and train a denoiser
Backpropagate the distribution-matching direction	Backpropagate the denoising loss
Keep teacher and fake-score weights fixed	Keep generator and teacher weights fixed

After training, keep G_θ for sampling. The two score models only supply training signals.

DMD2



DMD2, original training figure. Blue estimates the generated distribution; green adds a real-data adversarial signal.

Original DMD	Adds paired teacher-output regression to stabilize training
DMD2	Removes that regression and trains the fake score fast enough to track the generator
Additional signal	A GAN discriminator compares generated outputs with real images

The GAN component belongs to this recipe. It is not required by the DMD gradient identity.

Yin et al., DMD (CVPR 2024) · Yin et al., DMD2 (NeurIPS 2024)

Few-step generation and its limits



DMD2, original Figure 1: four-step SDXL samples. A mechanism example, not a current model ranking.

Sampling

Run only the distilled generator.

Evaluation

Check sample quality and diversity as well as latency.

Training objectives

The same inference budget can use different training constraints.

Distribution matching directly from samples

$$x = G_\theta(z) \sim q_\theta$$

Real reference samples

$$y_1^+, \dots, y_N^+ \sim p$$

Examples of the distribution we want.

Generated reference samples

$$y_1^-, \dots, y_M^- \sim q_\theta$$

Examples of the generator's current distribution.

These two batches provide the training direction. No auxiliary denoising model needs to learn the student score.

A local average of real examples

For the current output x, assign larger weights to more similar real examples.

$$w_j^+(x) = \frac{k(x, y_j^+)}{\sum_\ell k(x, y_\ell^+)}, \qquad \sum_j w_j^+ = 1$$

$$\mu_p(x) = \sum_j w_j^+(x) y_j^+$$

$$V_p^+(x) = \mu_p(x) - x$$

$k(x, y) > 0$: a similarity weight. $\mu_p(x)$: the weighted local mean of real examples.

V_p^+ points from the current output toward that local mean.

The current generator also has a local mean

Apply the same calculation to the generated reference batch.

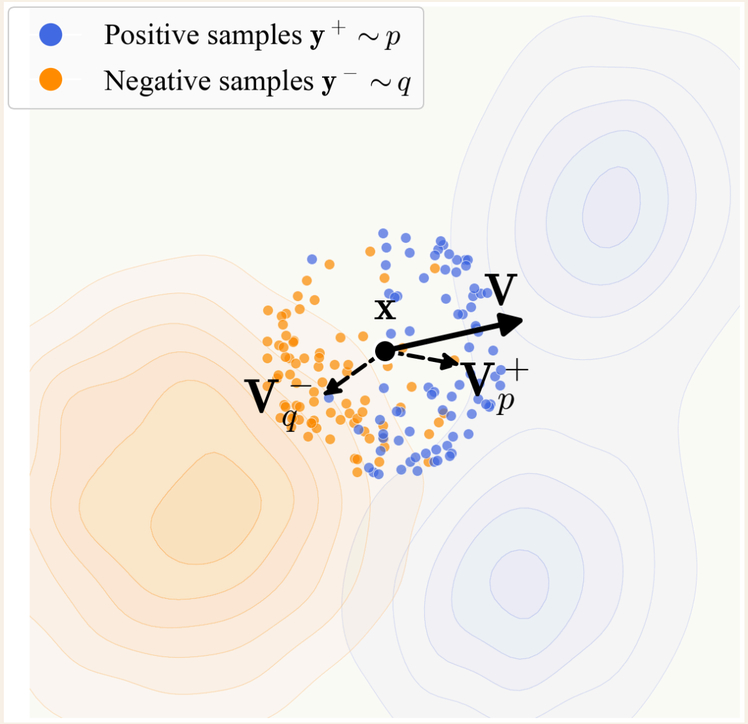
$$w_j^-(x) = \frac{k(x, y_j^-)}{\sum_\ell k(x, y_\ell^-)}, \qquad \mu_q(x) = \sum_j w_j^-(x) y_j^-$$

$$V_q^-(x) = \mu_q(x) - x$$

What it measures	Where the current generated samples concentrate near x.
Why it matters	If generated and real populations already agree, the correction should stop.

V_q^- itself points toward generated examples. Subtracting it will reverse that contribution.

The difference of the two local means



Original Drifting figure: real references in blue, generated references in orange, current output in black.

$$V_p^+ = \mu_p - x$$

$$V_q^- = \mu_q - x$$

$$\begin{aligned} V(x) &= V_p^+(x) - V_q^-(x) \\ &= \mu_p(x) - \mu_q(x) \end{aligned}$$

The two copies of x cancel. The net movement compares local real and generated populations.

Equal populations give equal local means, so the population drift is zero. Finite batches add estimation noise.

The drift supplies a frozen training target

$$x = G_\theta(z), \qquad x_{\text{target}} = \text{sg}[x + V(x)]$$

$$\mathcal{L} = \tfrac{1}{2}\mathbb{E}_z \| G_\theta(z) - x_{\text{target}} \|^2$$

$$\nabla_\theta \mathcal{L} = \mathbb{E}_z \left[\left(\frac{\partial G_\theta}{\partial \theta} \right)^\top (G_\theta(z) - x_{\text{target}}) \right]$$

$$= -\mathbb{E}_z \left[\left(\frac{\partial G_\theta}{\partial \theta} \right)^\top V(x) \right]$$

The residual equals $-V(x)$. Gradient descent therefore trains the generator to follow the drift.

sg: stop all gradients through the target, including the reference samples.

The practical Drifting recipe

The paper uses an exponential of unsquared distance, with additional batch and drift normalization.

$$k(x, y) = \exp(-\|x - y\|/h)$$

$$\mathcal{L}_{\text{feat}} = \tfrac{1}{2}\mathbb{E}\|\Phi(x) - \text{sg}[\Phi(x) + V(\Phi(x))]\|^2$$

Training	Φ is a frozen feature encoder. Gradients through its predicted features reach the generator.
Sampling	Run only the generator. The reference batches and encoder disappear.

Next, a Gaussian teaching case makes the score connection explicit. The paper's kernel and full feature recipe require separate care.

Gaussian weights define a smoothed density

For the score connection, choose a Gaussian kernel with width $h > 0$.

$$K_h(x \mid y) = \mathcal{N}(x; y, h^2 I), \qquad p_h(x) = \frac{1}{N} \sum_j K_h(x \mid y_j^+)$$

$$\nabla_x K_h(x \mid y) = K_h(x \mid y) \frac{y - x}{h^2}$$

$$\nabla_x \log p_h(x) = \frac{\sum_j K_h(x \mid y_j^+)(y_j^+ - x)}{h^2 \sum_j K_h(x \mid y_j^+)}$$

$$= \frac{\mu_p(x) - x}{h^2} = \frac{V_p^+(x)}{h^2}$$

p_h: a kernel density estimate from the real batch. The normalization gives exactly the weights used in μ_p.

The generated batch supplies the student score

Use the same Gaussian mixture calculation for the current generated examples.

$$q_h(x) = \frac{1}{M}\sum_j K_h(x \mid y_j^-), \qquad y_j^- \sim q_\theta$$

$$\nabla_x \log q_h(x) = \frac{\sum_j K_h(x \mid y_j^-)(y_j^- - x)}{h^2 \sum_j K_h(x \mid y_j^-)} = \frac{\mu_q(x) - x}{h^2} = \frac{V_q^-(x)}{h^2}$$

The student-distribution information is still present. A batch-based estimate replaces the auxiliary score network.

Fresh generated references track the current generator. Kernel width, batch size and representation affect the estimate.

The Gaussian drift is a score difference

$$V_p^+(x) = h^2 \nabla_x \log p_h(x), \qquad V_q^-(x) = h^2 \nabla_x \log q_h(x)$$

$$\boxed{V(x) = h^2 \left[\nabla_x \log p_h(x) - \nabla_x \log q_h(x) \right]}$$

DMD

Two denoisers estimate the target and generated scores.

Gaussian Drifting

Kernel-weighted reference batches compute the same type of score difference.

This proves the field-level connection. Matching the DMD parameter gradient also requires matching where we evaluate the field.

When the DMD parameter gradient matches

Use population Gaussian smoothing and evaluate the drift at the same noisy student sample as DMD.

$$u = G_\theta(z) + h\epsilon \sim q_h, \qquad V(u) = h^2(s_{p_h}(u) - s_{q_h}(u))$$

$$\mathcal{L}_{\text{aligned}} = \frac{1}{2h^2}\mathbb{E}\|u - \text{sg}[u + V(u)]\|^2$$

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\text{aligned}} &= -\frac{1}{h^2}\mathbb{E}\left[\left(\frac{\partial G_\theta}{\partial \theta}\right)^\top V(u)\right] \\ &= \mathbb{E}\left[\left(\frac{\partial G_\theta}{\partial \theta}\right)^\top (s_{q_h} - s_{p_h})\right] = \nabla_\theta \text{KL}(q_h \| p_h)\end{aligned}$$

Gaussian kernel, exact population scores and the same noisy query give the DMD-form gradient. The original clean-query recipe differs.

FD-Loss: matching feature statistics

A frozen encoder Φ represents real and generated images. Compare their feature means and covariances:

$$(\mu_p, \Sigma_p) \text{ for real data}, \qquad (\mu_q, \Sigma_q) \text{ for generated images}$$

$$\mathcal{L}_{\mathrm{FD}} = \|\mu_p - \mu_q\|^2 + \mathrm{tr}\left[\Sigma_p + \Sigma_q - 2(\Sigma_p^{1/2}\Sigma_q\Sigma_p^{1/2})^{1/2}\right]$$

Distribution signal	Mean and covariance differences provide generator gradients.
No extra score model	Differentiate through generated-image features and their statistics.

This is the Gaussian Fréchet distance in the chosen representation. Matching two moments does not identify every image distribution.

Yang et al., Representation Fréchet Loss (2026)

A large population with a small gradient batch

Reliable feature statistics need more samples than one affordable training batch.

Stored population	Keep recent generated features in a queue, or maintain running first and second moments.
Current batch	Generate fresh images and compute live features through the frozen encoder.
Generator update	Compute FD using the combined statistics. Backpropagate only through current-batch features.

Statistics use a broad sample history. The gradient graph covers only the current batch.

FD-Loss can post-train a fast generator or repurpose a multi-step model. Stale statistics and representation-specific optimization remain concerns.

Yang et al., Representation Fréchet Loss (2026)

Trajectory and distribution constraints

Trajectory-based methods

$$F_\theta(z) \approx F_{\mathrm{T}}(z)$$

A representative paired objective preserves the teacher's noise-to-output correspondence.

Distribution matching

$$q_\theta \approx p_{\mathrm{T}}$$

Different correspondences can satisfy the same population-level objective.

Consistency and MeanFlow also constrain local or finite-time relations. They need not use an explicit paired endpoint loss.

A correct output distribution does not uniquely determine which output belongs to each input noise.

What each objective makes easier

	Trajectory-based methods	Distribution matching
Training signal	A target destination or a local flow relation	A signal comparing two populations
Main advantage	Structured supervision for a chosen transformation	Freedom to change the noise-to-output correspondence
Main difficulty	Accumulated target error or an incompatible teacher map	Estimating the population mismatch reliably
Typical cost	Teacher trajectories, local evaluations, or JVPs	Score tracking, adversarial training, or sample interactions

Initialization and post-training can use different objectives. Causal video generation will make that distinction concrete.

Boffi et al., Flow Maps via Self-Distillation (2025) · Yin et al., DMD (CVPR 2024) · Yin et al., DMD2 (NeurIPS 2024) · Zhu et al., Causal Forcing (ICML 2026)

Two loops in video generation

Video time k

Generate the next chunk, append it to history, then repeat.

$$\hat{Y}_k \sim q_\theta(Y_k \mid \hat{Y}_{<k}, C)$$

Generation coordinate τ

Within each chunk, transform noise into a clean prediction.

$$x_{k,0} \sim \mathcal{N}(0, I), \qquad x_{k,1} = \hat{Y}_k$$

Fast sampling reduces the computation inside each next-chunk prediction.

A fast generator for the next chunk

$$\hat{Y}_k = G_\theta(z_k, \hat{Y}_{<k}, C)$$

The fresh noise selects a possible continuation. The generated prefix supplies the current history.

The same few-step generator runs again for the next chunk.



Self Forcing, original project clip. Observe the continuously generated sequence, not just one frame.

Errors in one generated chunk become part of the next input.

Huang et al., Self Forcing (NeurIPS 2025)

The history seen during training

Training procedure	Prefix supplied to the predictor
Teacher Forcing	Clean frames from a real training video
Diffusion Forcing	Real-video frames with independently sampled corruption levels
Self Forcing	Earlier outputs produced by the current generator

Noise corruption can cover different noise levels without reproducing the model’s own content errors.

Self-generated history changes the distribution of training inputs.

Evaluating a generated video

$$\hat{V} = (\hat{Y}_1, \ldots, \hat{Y}_K), \qquad V_\tau = \alpha_\tau \hat{V} + \sigma_\tau E$$

Online student

Generate chunk k using only its available prefix.

Offline supervision

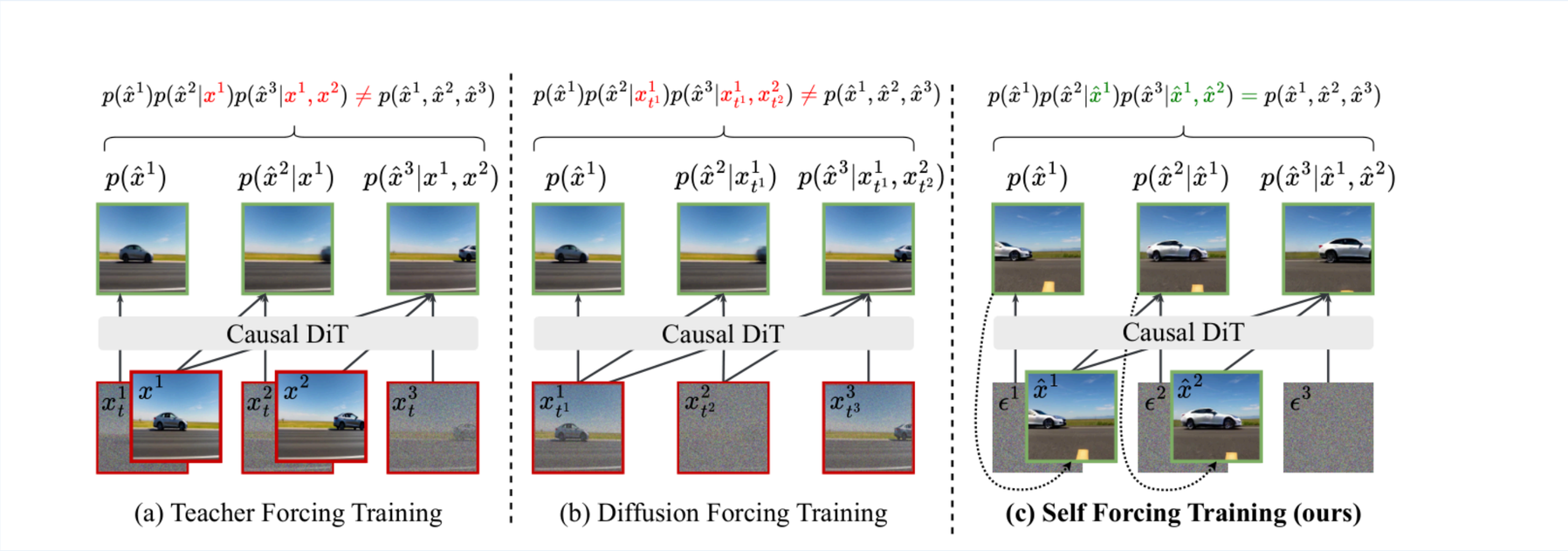
After rollout, evaluate the noisy full clip with a bidirectional teacher.

$$s_{\mathrm{T}}(V_\tau, \tau, C), \qquad s_\phi(V_\tau, \tau, C)$$

E: fresh video-shaped Gaussian noise. Both scores now describe distributions over complete clips.

The teacher's training-time view does not give the deployed student access to future chunks.

Self Forcing rollout



Self Forcing, original Figure 1. The right-hand procedure feeds generated outputs into later predictions during training.

$$\hat{Y}_1 = G_\theta(z_1, C), \quad \hat{Y}_2 = G_\theta(z_2, \hat{Y}_1, C), \quad \dots$$

Cache the generated prefix as the rollout advances. Evaluate the resulting full clip.

Distribution matching on the rollout

$$q_\theta(V \mid C) = \prod_{k=1}^{K} q_\theta(Y_k \mid Y_{<k}, C)$$

$$\mathcal{J}_{\text{video}} = \mathbb{E}_\tau[w_\tau \, \mathrm{KL}(q_{\theta,\tau}(V \mid C) \,\|\, p_{\mathrm{T},\tau}(V \mid C))]$$

$$\text{score correction:} \qquad s_\phi(V_\tau, \tau, C) - s_{\mathrm{T}}(V_\tau, \tau, C)$$

Samples	Complete rollouts from the causal student
Supervision	A distribution-level signal on the noisy complete video

Self Forcing specifies the rollout procedure. DMD is one choice of holistic training objective.

Training within a memory budget

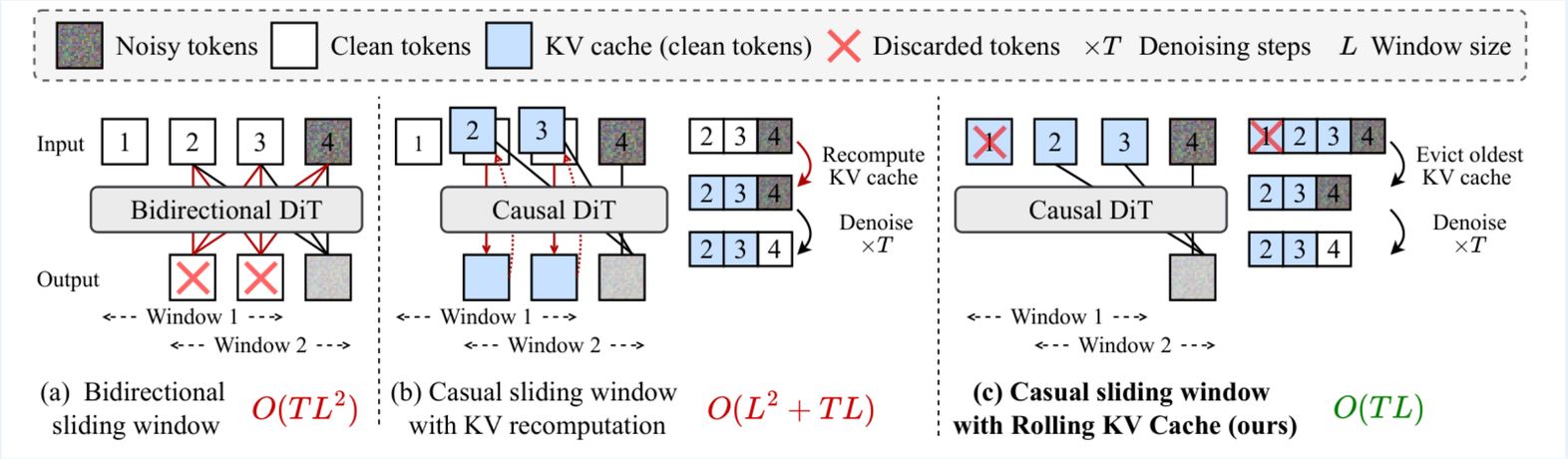
Computation	Gradient treatment
Run the chunk’s early denoising steps	No gradient through these earlier sampling steps
Use a randomly selected denoising exit	Backpropagate through that final selected denoising call
Condition on earlier generated chunks	Detach their cached keys and values
Evaluate the completed generated video	Use the holistic objective to supervise the retained paths

Random exits give different denoising steps a chance to receive supervision.

The practical update does not backpropagate through all video history and all sampling steps.

Huang et al., Self Forcing (NeurIPS 2025)

Rolling KV cache



Self Forcing, original Figure 3. The rolling cache retains recent context without recomputing the full overlapping prefix.

Reuse	Keep attention keys and values for recent generated chunks
Roll	Drop the oldest entries when the context window fills

A finite cache reduces computation. It does not guarantee persistent geometry or unlimited reliable memory.

A teacher target the student cannot reproduce

Bidirectional teacher

$$Y_k^{\mathrm{T}} = [F_{\mathrm{T}}(z_{1:K}, C)]_k$$

Its current-chunk target can depend on future noise $z_{>k}$.

Causal student

$$Y_k^{\mathrm{S}} = [F_\theta(z_{\le k}, C)]_k$$

It cannot read that future noise.

Hold the student-visible input fixed. Vary the hidden future noise. The teacher's current target may change.

Paired squared-error regression averages those conflicting targets.

F: full noise-to-video sampling computation. $[\cdot]_k$: its output chunk k.

The mismatch concerns a particular correspondence, not whether an AR model can represent the video distribution.

Paired regression averages hidden futures

$$u = z_{\le k}, \qquad h = z_{>k}, \qquad T_k(u,h) = [F_{\mathrm{T}}(z_{1:K}, C)]_k$$

The student sees u. Its paired teacher target also depends on hidden h.

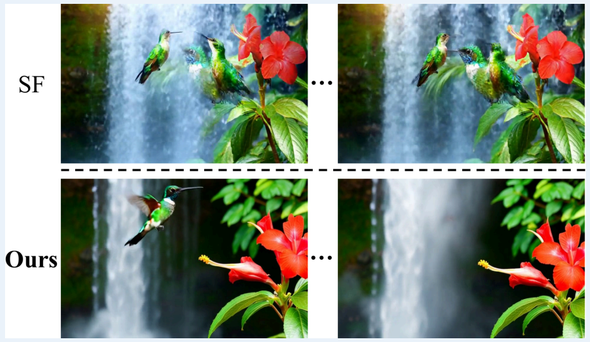
$$f_k^*(u) = \arg\min_f \mathbb{E}_{h|u} \| f(u) - T_k(u,h) \|^2 = \mathbb{E}[T_k(u,h) \mid u]$$

$$\mathbb{E}\| f(u) - T_k \|^2 = \mathbb{E}\| f(u) - \mathbb{E}[T_k \mid u] \|^2 + \mathbb{E}\| T_k - \mathbb{E}[T_k \mid u] \|^2$$

A causal teacher removes this inaccessible dependence from the paired target. Distribution matching can instead relax the pairing.

Causal initialization and asymmetric DMD

Train an AR teacher	Learn next-chunk denoising with a causal information interface
Distill its causal map	Initialize the few-step AR student with compatible paired ODE targets
Match full-video distributions	Use a strong bidirectional teacher for the DMD post-training stage



Causal Forcing, original qualitative comparison of initialization choices followed by DMD.

Paired map supervision and distribution supervision can use different teacher architectures.

Self-Forcing++: the supervision horizon

Self Forcing

Train on self-generated clips that fit within the teacher's short video horizon.

Self-Forcing++

Train on long self-rollouts. Ask the short-video teacher to supervise sampled segments.

Teacher horizon	Short segment K
Student training	Long self-rollout $L \gg K$
Student inference	Long self-rollout

Seeing generated history during training and seeing long generated history are different requirements.

Short windows from a long self-rollout

Long student rollout

Earlier history	**Selected window**	Later continuation

The teacher evaluates only the selected segment

$$\hat{V} = (\hat{Y}_1, \ldots, \hat{Y}_L), \qquad i \sim \mathrm{Unif}\{1, \ldots, L - K + 1\}$$

$$W_i = \hat{Y}_{i:i+K-1}, \qquad W_{i,\tau} = \alpha_\tau W_i + \sigma_\tau E$$

$$\text{DMD direction at this window:} \qquad s_\phi(W_{i,\tau}, \tau, C) - s_{\mathrm{T}}(W_{i,\tau}, \tau, C)$$

L: rollout length. K: the teacher's segment length. E: fresh Gaussian noise.

Training on late, degraded contexts

Long rollout	Use a rolling KV cache during training, as at inference.
Backward noise initialization	Re-noise the student's generated latents before teacher evaluation. The noisy input retains content from this rollout.
Window supervision	Apply distribution matching across sampled positions, including late segments.

Re-noising is the DMD operation we already derived. The extension changes which student outputs receive supervision.

Matching short windows does not guarantee that an object remains correct after a long occlusion.

Other distribution-matching methods

Inductive Moment Matching (IMM)	Learns generative jumps using distribution-level moment matching and bootstrap targets.
Adversarial objectives	A learned discriminator can supply another population comparison, as in the DMD2 recipe.

These objectives can complement map-based initialization. Their training signals make different demands on features, batches and auxiliary models.

Zhou et al., Inductive Moment Matching (ICML 2025) · Yin et al., DMD2 (NeurIPS 2024)

Active research directions

Long-term consistency	How can supervision over short windows preserve identity and geometry across long gaps?
Compatible supervision	Which teacher information can an online causal student actually reproduce?
Efficient credit assignment	How can late errors improve earlier decisions without retaining an entire rollout graph?
Controlled prediction	How do we evaluate the consequences of new actions, beyond visual plausibility?

Fast generation makes an interactive loop possible. Reliable world prediction adds requirements that these objectives do not settle.