Penn Engineering
University of Pennsylvania

GMLR

CIS 6280 · FALL 2026

WORLD MODELS

LECTURE 13

Video World Models and Fast Sampling

Jiatao Gu

University of Pennsylvania · October 20, 2026

Course announcements

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

The image generator we already have

$$r_\tau = (1-\tau)n + \tau y, \qquad n \sim \mathcal{N}(0, I)$$

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

Training	Generation
Construct a noisy mixture of an image and noise	Start from new noise; repeatedly evaluate the learned field
Supply the condition C	Integrate from $\tau = 0$ to $\tau = 1$

y: clean image. n: Gaussian noise. C: text or another supplied condition.

The complete text-to-image system

Text encoder	Turn the prompt into conditioning features C
Generator	Refine a noisy image or latent using the condition and guidance
Image output	Decode a generated latent, or directly return pixels

$$x_{\tau+\Delta\tau} \approx x_\tau + \Delta\tau\, v_\theta^{\text{guided}}(x_\tau, \tau, C)$$

Text conditioning, generation space, architecture, and sampler are separate choices.

Text-to-video generation



Wan2.1 generated clip, reproduced from the Self Forcing comparison page.

How does one conditional generator produce a coherent sequence?

One distribution over a video



Frame 1



Frame 2



Frame 3



Frame 4



Frame 5

$$p_\theta(V \mid C), \qquad V = (y^{(1)}, \ldots, y^{(T)})$$

V: a complete clip. T: video length. C: supplied text or visual evidence.

Sampling independent images would discard the relationships between frames.

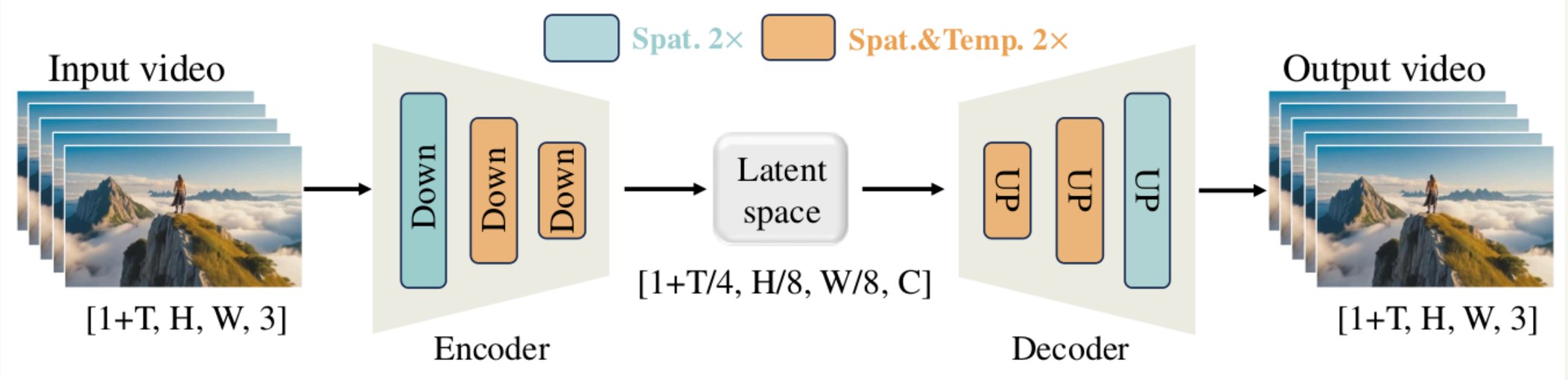
Video time and generation time

Generation coordinate	Frame 1	Frame 2	Frame 3
$\tau = 0 \cdot$ noise	$y_0^{(1)}$	$y_0^{(2)}$	$y_0^{(3)}$
$0 < \tau < 1$	$y_\tau^{(1)}$	$y_\tau^{(2)}$	$y_\tau^{(3)}$
$\tau = 1 \cdot$ generated clip	$y_1^{(1)}$	$y_1^{(2)}$	$y_1^{(3)}$

Across a row: video position. Down the table: progress in generating the candidate frames $y_\tau^{(i)}$.

A joint sampling step updates every video position together.

A representation for a video



Wan, Figure 5: an autoencoder compresses spatial and temporal dimensions.

$$Z_{\text{data}} = E(V), \qquad \tilde{V} = D(Z_{\text{data}})$$

Train the reconstruction system, then hold it fixed while learning the generative prior.

Temporal compression reduces the tensor processed at every sampling step.

Wan Team, Wan (2025)

Space-time tokens

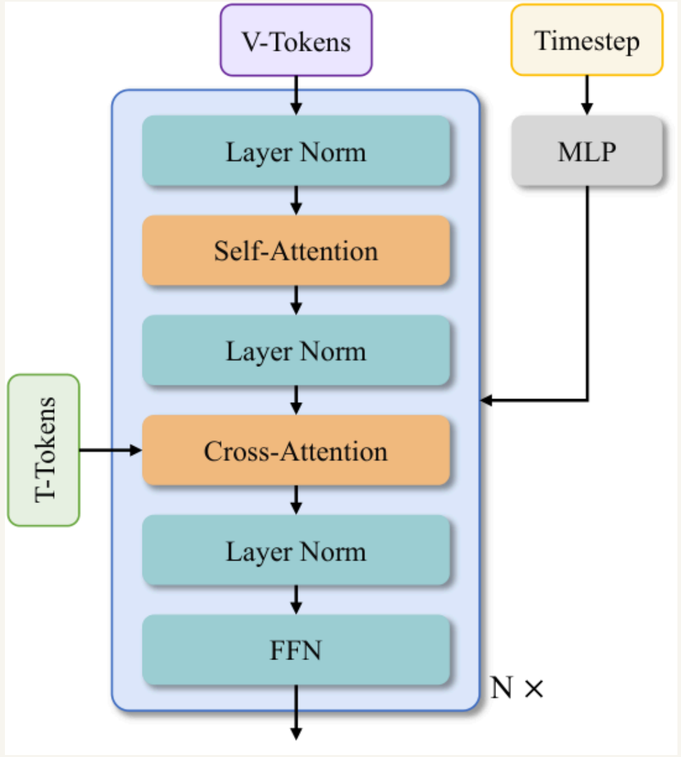
$$Z_\tau \in \mathbb{R}^{T_z \times H_z \times W_z \times d_z}$$

T_z, H_z, W_z: latent time, height, and width. d_z: channels.

Input to the transformer	Information
Patch embeddings	Continuous features from the noisy video latent
Space-time coordinates	Where each token lies in the clip
Generation coordinate τ	How far generation has progressed

Wan Team, Wan (2025)

The video transformer



Wan, Figure 10. Self-attention reads video tokens; cross-attention reads text.

Video self-attention

Connect locations within the candidate clip

Text cross-attention

Read the supplied prompt features

Time modulation

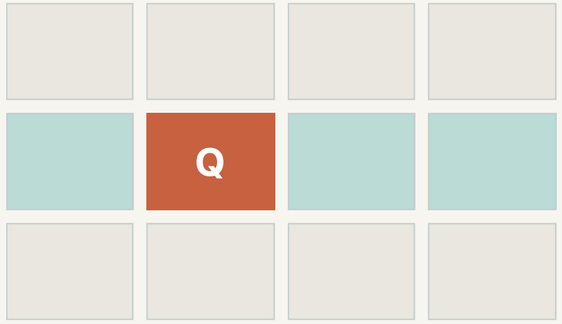
Inject the generation coordinate into the blocks

One network evaluation predicts an update tensor for the whole candidate.

Spatial, temporal, and joint attention

Spatial

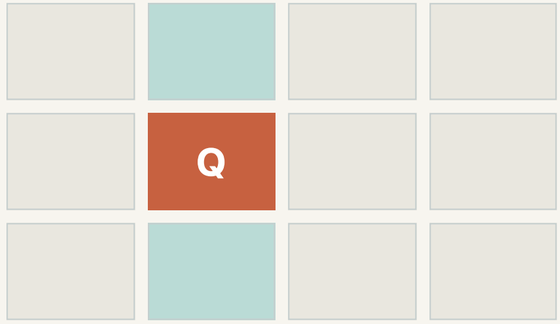
time ↓ space →



Other locations in the same frame

Temporal

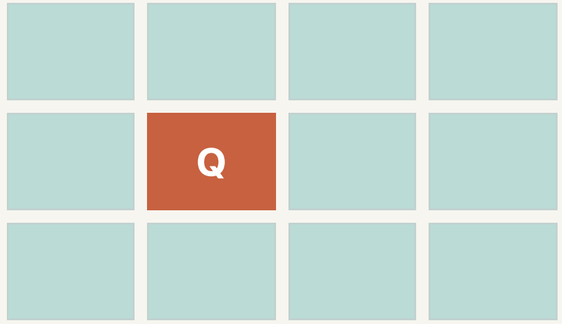
time ↓ space →



The same location across frames

Full space-time

time ↓ space →



All locations across all frames

Q is one query token. Colored cells show the keys it may read.

A moving object can occupy different spatial locations in successive frames.

The cost of full space-time attention

$$L \text{ temporal positions}, \quad P \text{ spatial tokens per position}, \quad N = LP$$

Attention pattern	Token-pair interactions
Spatial, within each temporal position	$O(LP^2)$
Temporal, at each spatial position	$O(PL^2)$
Full space-time attention	$O(L^2P^2) = O(N^2)$

Alternating spatial and temporal attention factorizes the computation. Joint attention connects space and time in one operation.

Compression and sparse attention make longer or higher-resolution clips more affordable.

Temporal access in joint attention

Query position	Key 1	Key 2	Key 3	Key 4	Key 5
1	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓

Click a query row. Cells show permitted access, not attention weights.

A candidate frame can use both earlier and later candidate frames at the same generation coordinate.

Temporal masking is a separate choice from spatial–temporal factorization.

Training the joint video generator

$$Z_{\text{data}} = E(V), \quad Z_\tau = (1-\tau)\eta + \tau Z_{\text{data}}$$

$$\mathcal{L} = \mathbb{E}\|v_\theta(Z_\tau, \tau, C) - (Z_{\text{data}} - \eta)\|^2$$

Construct	An encoded video, same-shaped noise η, and a coordinate τ
Condition	Supply features C from the paired prompt
Learn	Predict the velocity tensor across the complete clip

Wan Team, Wan (2025) · Lipman et al., Flow Matching

Generating the complete clip

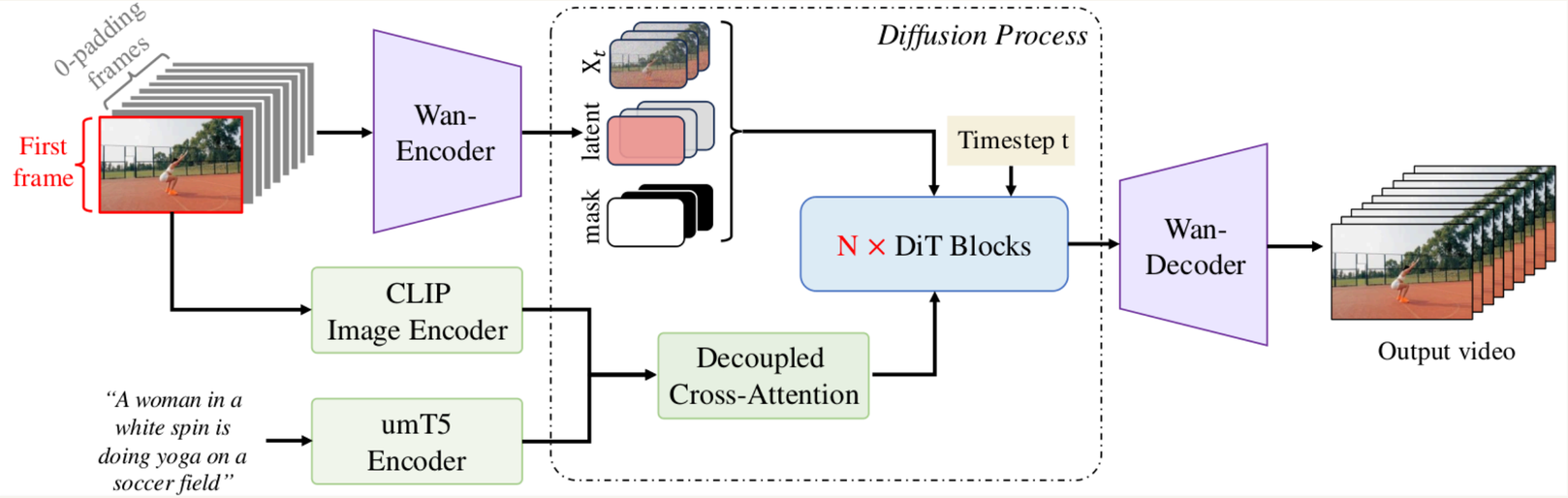
$$Z_{\tau=0} \sim \mathcal{N}(0, I)$$

$$Z_{\tau+\Delta\tau} \approx Z_\tau + \Delta\tau \, v_\theta^{\text{guided}}(Z_\tau, \tau, C)$$

$$\hat{V} = D(Z_{\tau=1})$$

Repeated network evaluations refine the full latent clip; decoding comes afterward.

Image-to-video conditioning



Wan, Figure 18: image features and a masked visual condition accompany the noisy video latent.

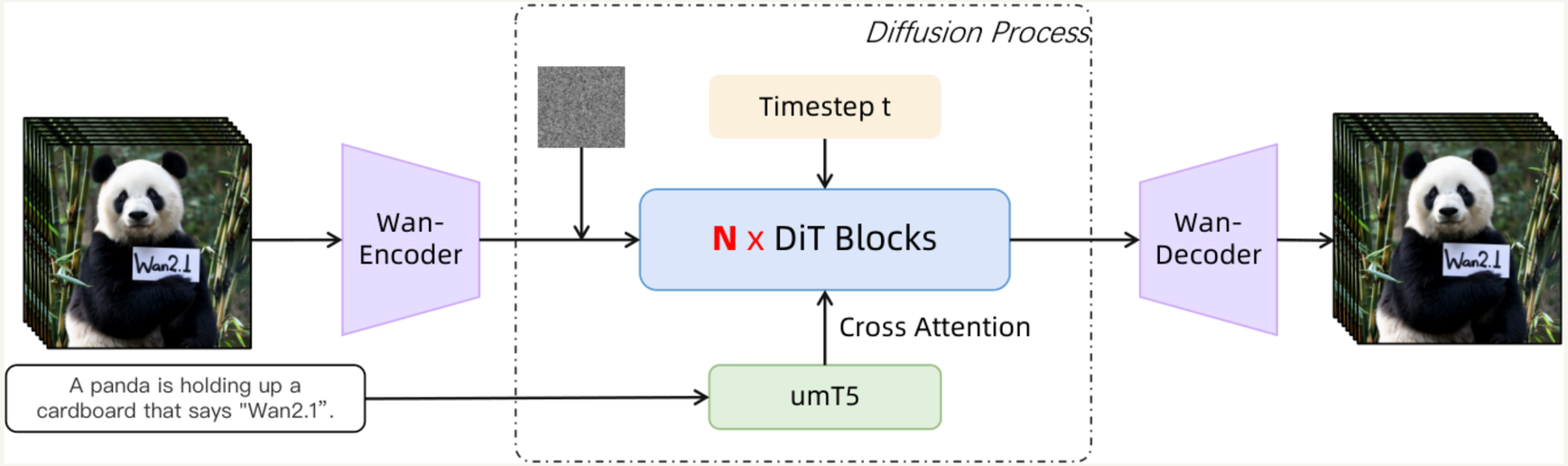
A supplied reference narrows appearance and layout. The model generates possible subsequent motion.

The remaining frames can still be generated jointly.

The complete video pipeline

Observed-video encoder: training only

Generation starts at the Gaussian-noise inlet ↓



Wan, Figure 9. The observed clip on the left supplies training latents; generation does not require that clip.

Learning	Generation
Encode observed clips; learn a conditional field	Draw noise; refine a latent clip; decode
Videos supply temporal evidence	The sampler determines repeated network evaluations

Seedance: motion across shots



Seedance 2.0 official T2V example: figure skating. Generated footage, classroom 1080p copy.

A rotating subject, a moving camera, and changes of shot all have to remain coherent.

Seedance 2.0 official examples (2026) · Seedance 2.5 official page (checked Oct. 19, 2026)

Seedance: generation from references



Seedance 2.0 official R2V example, with reference material shown in the source demonstration. Classroom 720p copy.

The condition can specify a storyboard, a character, a scene, and props.

Seedance 2.0 official examples (2026) · Seedance 2.0 report (2026)

The published Seedance training recipe

Training evidence	Curate motion and audiovisual alignment; describe both the visual event and its sound.
Joint generation	Let video and audio representations interact inside the generative transformer.
Output refinement	Fine-tune on selected examples, optimize human-preference rewards, and distill for faster generation.

Documented in Seedance 1.5 Pro. The cited 2.5 product page does not specify a full training recipe.

Architecture determines what can interact. Data and training determine what the interaction learns.

Seedance 1.5 Pro technical report · Seedance 2.0 report (2026) · Seedance 2.5 official page (checked Oct. 19, 2026)

MiniMax H3: a shared generation backbone

Given references

Text / visual meaning · visual VAE latents · audio VAE latents

↓

Condition tokens

New Gaussian noise

Not the reference inputs

↓

Output candidates: video + audio tokens

↓ Combined token sequence ↓

33B dense single-stream Transformer

Shared attention and FFN; modality-specific inputs, outputs, and modulation

Video VAE decoder → video

Audio VAE decoder → stereo sound

A common token sequence lets reference content, video, and audio inform generation.

MiniMax H3 official release (checked Oct. 19, 2026) · MiniMax H3 released weights and license

The released H3 model and the complete system

Context-IR

Interpret and organize multimodal instructions

Hosted

→

H3-Base

Generate 768p video and stereo audio

Released weights

→

Regenerate-2K

Refine the result using the original context

Hosted

The release documentation describes full-attention inference; native sparse attention is not included in the initial release.

A released base model and a hosted product can differ in their conditioning and refinement stages.

Video models as world models

Video learning provides	Interactive prediction additionally asks
A representation of visible appearance and change	What happens from this observed history?
A distribution over possible visual events	How does the future depend on the supplied interaction?

The predicted world can be represented through observations, without an explicit physical state vector.

For an interactive world model, specify the evidence and actions that condition its predictions.

Wan Team, Wan (2025) · Chen et al., Diffusion Forcing (2024)

Predicting futures from experience and interaction

$$h_t = (o_{\leq t}, a_{<t})$$

$$p_\theta(o_{t+1:t+H} \mid h_t, a_{t:t+H-1})$$

Evidence h_t	Observed history and previous actions
Candidate actions	The interactions whose consequences we want to predict
Future observations	A distribution of possible visual outcomes over horizon H

Action conditioning requires suitable data and tests of the predicted consequences.

Chen et al., Diffusion Forcing (2024) · Zheng et al., Causal-rCM (2026)

Joint prediction and online interaction

Joint future block	Online interaction
Condition on the known history and a supplied action sequence	Receive an action, predict a segment, then accept another action
Bidirectional computation can operate within the unknown future block	Already displayed outputs constrain the next prediction
Regenerating a block can revise candidate frames	Long blocks can delay the effect of new controls

Future chunks can use bidirectional attention while successive chunks follow causal order.

Huang et al., Self Forcing (2025) · Zheng et al., Causal-rCM (2026)

Autoregression across video chunks

$$B_{1:K} = (B_1, \ldots, B_K)$$

$$p_\theta(B_{1:K} \mid h_t, A, C) = \prod_{k=1}^{K} p_\theta(B_k \mid h_t, B_{<k}, A_{\leq k}, C)$$

Across chunks	Within a chunk
Generate the next block from the available prefix	Use diffusion or flow matching over its continuous tensor
Condition on controls available for that block	Refine all current-block tokens together

Encode the observed history h_t into the initial context. B_k: a future latent chunk; A_k: its controls; C: other conditions.

Huang et al., Self Forcing (2025) · Zheng et al., Causal-rCM (2026)

Sampling the next video chunk

$$Z^{(k)}_{\tau=0} \sim \mathcal{N}(0, I)$$

$$\frac{dZ^{(k)}_{\tau}}{d\tau} = v_\theta(Z^{(k)}_{\tau}, \tau, h_t, \hat{B}_{<k}, A_{\leq k}, C)$$

Inner loop	Denoise or integrate the current chunk
Commit	Decode or emit the generated segment $\hat{B}_k$
Outer loop	Keep the observed history; append this future chunk and receive new controls

There are now two loops: generation steps inside a chunk, and rollout across chunks.

Huang et al., Self Forcing (2025) · Zheng et al., Causal-rCM (2026)

Causal access across chunks

Query chunk	Key 1	Key 2	Key 3	Key 4	Key 5
1	✓	—	—	—	—
2	✓	✓	—	—	—
3	✓	✓	✓	—	—
4	✓	✓	✓	✓	—
5	✓	✓	✓	✓	✓

Click a query row. Cells show permitted access, not attention weights.

The current chunk can read its prefix and itself. Later chunks remain unavailable.

KV caching reuses prefix features across current-chunk denoising steps.

Huang et al., Self Forcing (2025)

Teacher forcing: learn from the real prefix

$$Z_\tau^{(k)} = (1-\tau)\eta_k + \tau B_k$$

$$\mathcal{L}_{\mathrm{TF}} = \mathbb{E}\|v_\theta(Z_\tau^{(k)}, \tau, h_t, B_{<k}, A_{\le k}, C) - (B_k - \eta_k)\|^2$$

Context	Noisy network input	Velocity target
Observed history and clean real prefix $B_{<k}$	$Z_\tau^{(k)}$	$B_k - \eta_k$

The loss trains a conditional generator, not a deterministic next-frame regressor.

Huang et al., Self Forcing (2025) · Zheng et al., Causal-rCM (2026)

Real history and generated history

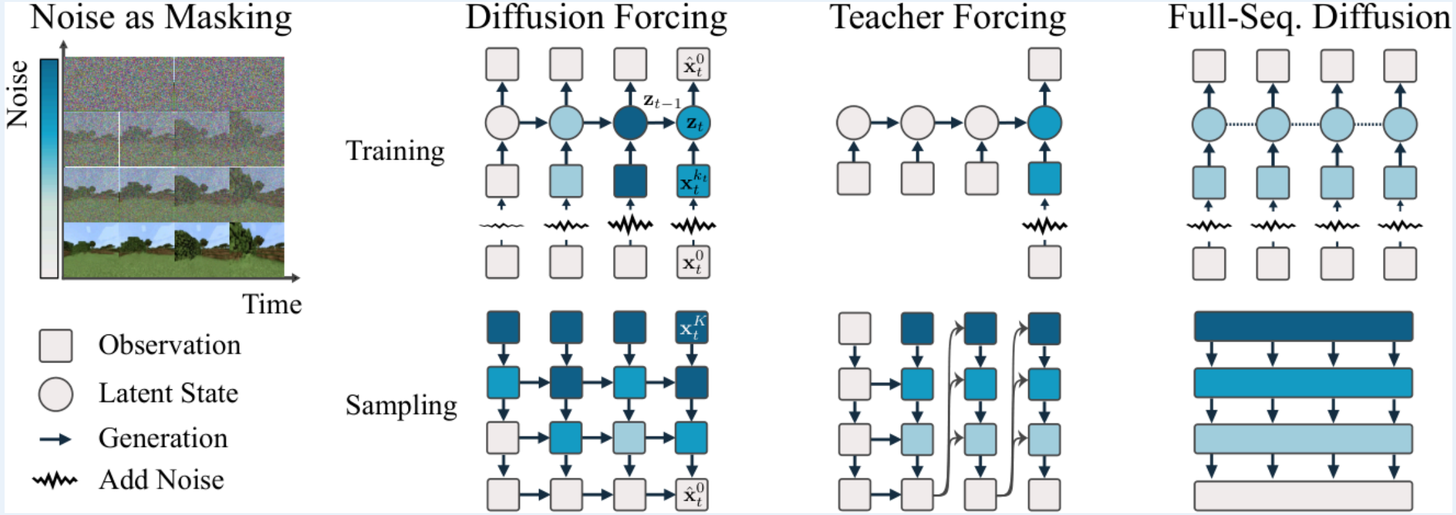
During teacher-forced training	During a simulator rollout
$B_{<k}$	Previously generated $\hat{B}_{<k}$
Consistent context from a recorded video	A prefix that may contain model errors
Learn the next-block conditional on data	Continue from the consequences of earlier predictions

A small error can alter the context of every later prediction.

Causal factorization does not remove the train–rollout mismatch.

Diffusion forcing: different noise levels

$$Z_{\tau_i}^{(i)} = (1-\tau_i)\eta_i + \tau_i B_i$$

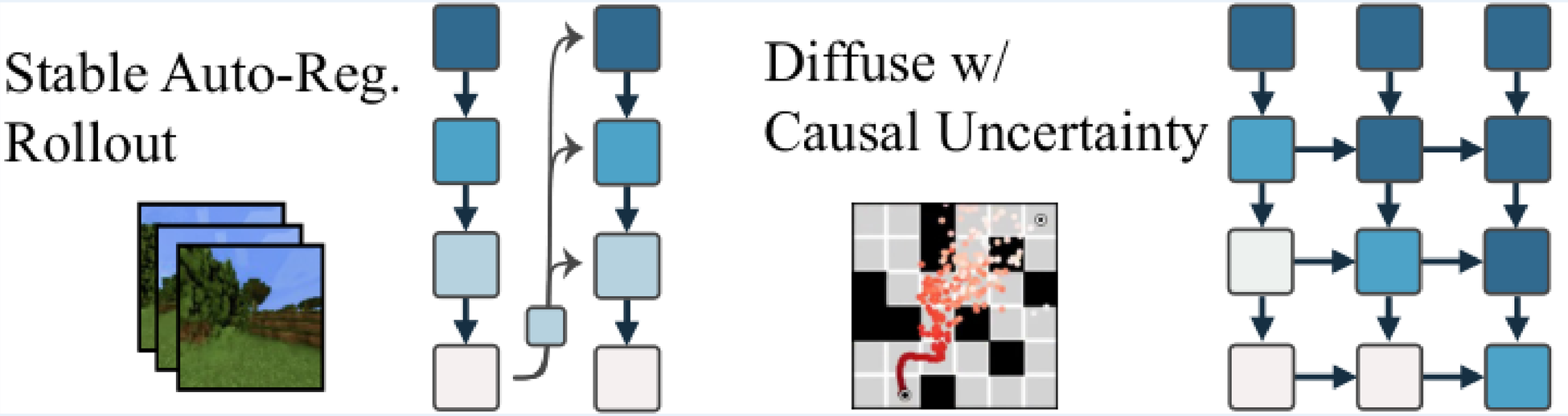


Diffusion Forcing, Figure 2. Squares: observations at different noise levels. Circles: recurrent memory.

Paper: larger corruption index k → more noise. Here: larger τ → cleaner.

Sample noise levels independently across positions; supply each level to the causal model.

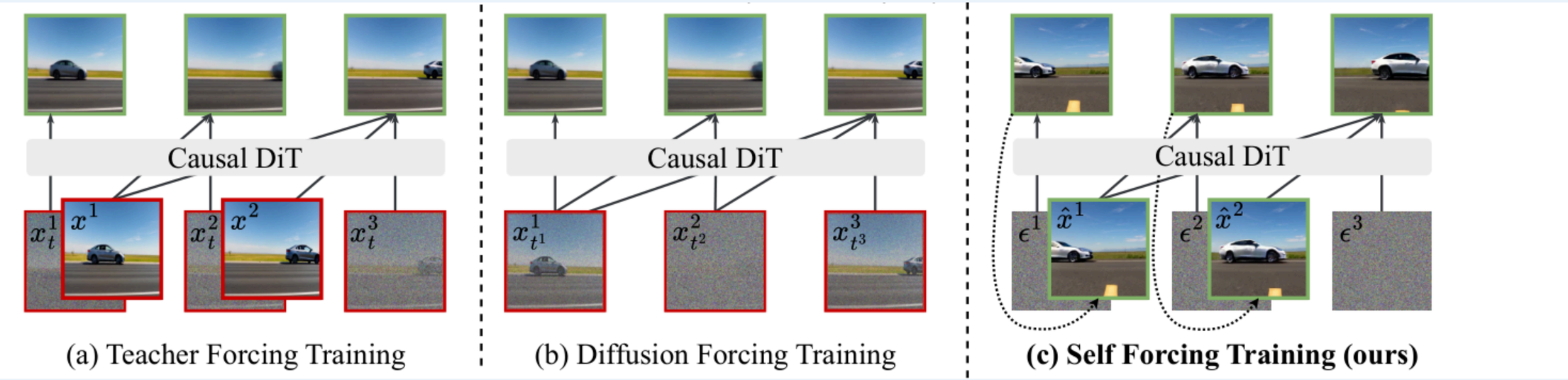
Noise schedules across a sequence



Diffusion Forcing, Figure 3 excerpt. Across: video position. Down: denoising progress. Darker: noisier.

Left: autoregressive rollout	Right: staggered refinement
Finish the next position before moving on	Refine several future positions at different noise levels
Earlier positions become clean context	More distant positions can remain more uncertain

What context does training provide?



Self Forcing, Figure 1, excerpt: real history, corrupted real history, and self-generated history.

Teacher forcing	Diffusion forcing	Self-rollout training
Clean real prefix	Corrupted real prefix	The model’s own generated prefix

Gaussian-corruption robustness and robustness to generated-history errors are different questions.

Huang et al., Self Forcing (2025) · Zheng et al., Causal-rCM (2026)

Why teacher forcing remains useful

A clear conditional target	Learn next-block generation from available clean history
Parallel offline training	Use packed clean context and noisy targets with causal masks
Separate rollout refinement	Later training can expose the model to its own histories

Causal-rCM (2026) combines offline learning on real prefixes with refinement on generated prefixes.

Teacher forcing and rollout training can serve different stages of one system.

Zheng et al., Causal-rCM (2026)

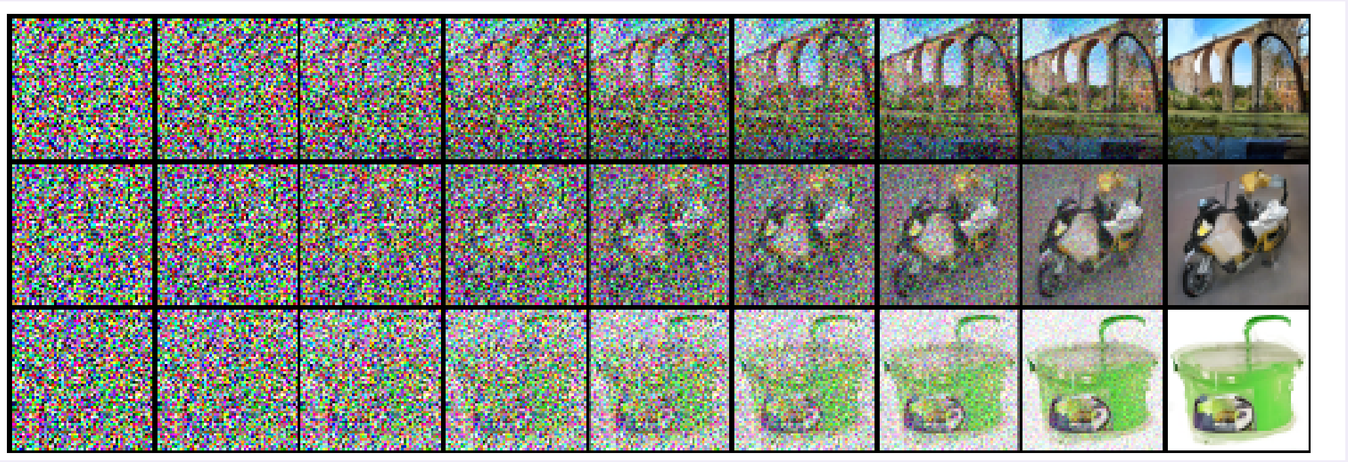
Streaming generation



Self Forcing: a published autoregressive rollout. Streaming output alone does not establish action accuracy.

History consistency, response to controls, and latency need separate evaluation.

The cost of another prediction



Flow Matching paper: successive states during image generation.

$$\text{sampling cost} \approx \text{network evaluations} \times \text{cost per evaluation}$$

A streaming video model repeats this computation for every new chunk.

We return to images to study how to reduce expensive network evaluations.

Two routes to fewer evaluations

Existing model, better sampler	A model trained for few-step generation
Keep the learned velocity or denoising field fixed	Change the learned mapping or training objective
Approximate its generative dynamics more efficiently	Learn a finite jump or a reusable velocity policy
DDIM / ODE solvers	Flow maps, consistency, π-Flow, distribution matching

Both reduce sampling work, but only the second route changes what the network learns.

Geng et al., MeanFlow (2025) · Boffi et al., Flow Maps via Self-Distillation (2025)

Two main learning approaches

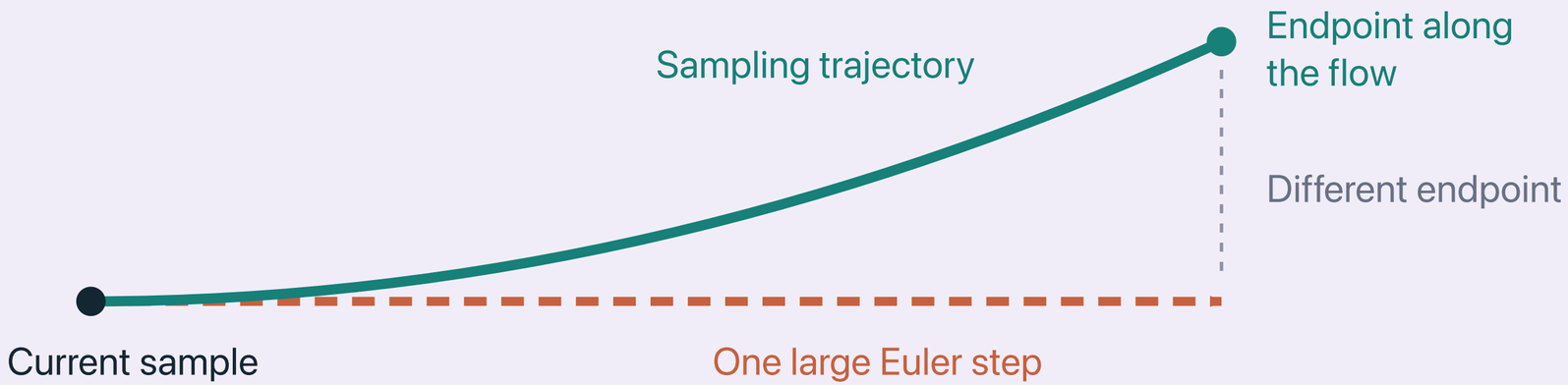
Trajectory-based methods	Distribution matching
Learn relationships along a generative trajectory	Match generated samples to a target distribution
Constrain maps, derivatives, consistency, or policy velocities	Do not require a fixed sample-by-sample correspondence
Flow maps / Consistency / MeanFlow / π-Flow	DMD / IMM / Drifting

These describe the main training constraint. Hybrid objectives can combine them.

Distillation versus training from scratch is a separate choice.

Boffi et al., Flow Maps via Self-Distillation (2025) · Chen et al., π-Flow (ICLR 2026) · Yin et al., Distribution Matching Distillation · Zhou et al., Inductive Moment Matching (2025) · Deng et al., Drifting Models (2026)

A large step can miss the trajectory



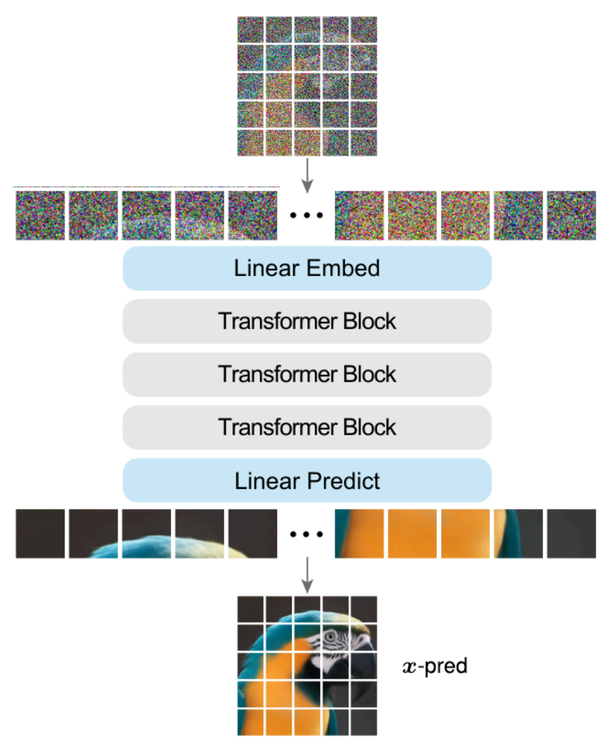
$$x_{\tau+\Delta\tau} \approx x_\tau + \Delta\tau\, v_\theta(x_\tau, \tau)$$

The model predicts a local velocity. As the sample moves, that direction can change.

Taking a larger step alone does not make a reliable one-step generator.

Geng et al., MeanFlow (2025)

Why a clean-image prediction still needs sampling



JiT: the direct network output is a clean-image prediction.

Denoising regression learns a conditional estimate:

$$\hat{y}(r_\tau, \tau, C) \approx \mathbb{E}[y \mid r_\tau, \tau, C]$$

One noisy input can be compatible with many clean images.

A trajectory endpoint map instead selects the endpoint reached from this particular starting point.

Changing the output parameterization does not train a one-step sampler.

Progressive distillation

The teacher takes two deterministic DDIM steps. Train the student to reach the same result in one step.



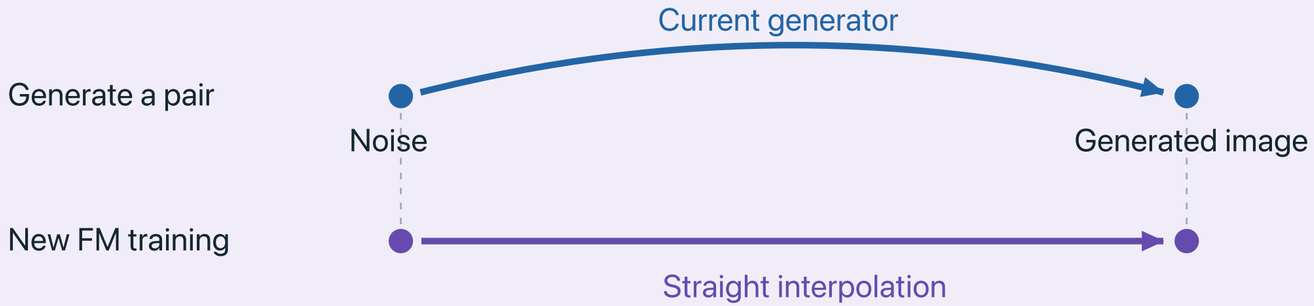
$$\mathcal{L}_{\text{endpoint}} = \mathbb{E} \left\| S_\theta(x_s, s, t) - x_t^{\mathrm{T}} \right\|^2$$

S_θ: a complete student DDIM update from generation coordinate s to t. x_t^{T}: the frozen teacher target.

Use the trained student as the next teacher, then halve the sampling step count again.

Salimans and Ho, Progressive Distillation (2022)

Reflow



y^{T} is the image generated from n. Keep this pair together when training the new velocity network v_ϕ.

$$r_\tau = (1-\tau)n + \tau y^{\mathrm{T}}$$

$$\mathcal{L}_{\text{reflow}} = \mathbb{E}\left\|v_\phi(r_\tau, \tau) - (y^{\mathrm{T}} - n)\right\|^2$$

Initial FM: independently paired noise and data. Reflow: each noise and its own generated image.

The new pairing encourages straighter trajectories for more accurate coarse sampling.

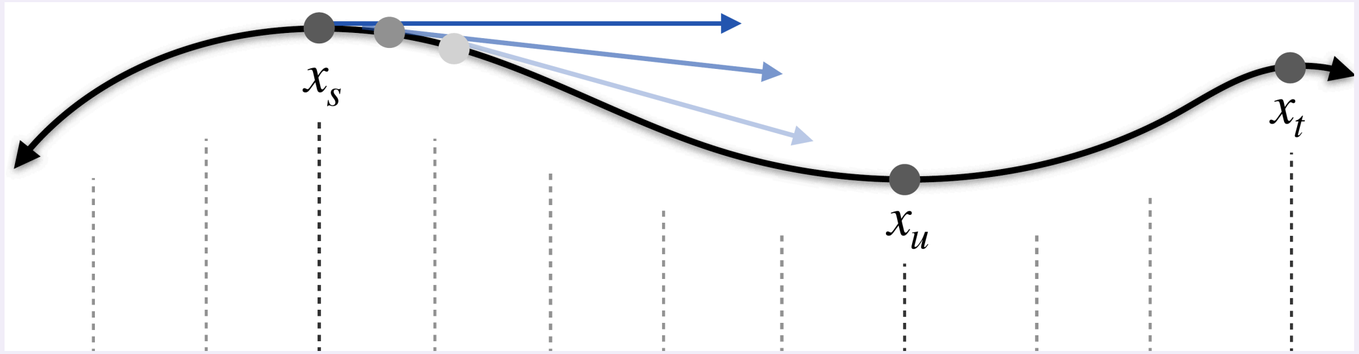
Liu et al., Rectified Flow / Reflow (2022)

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

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

$$x_{t_{i+1}} = F_\theta(x_{t_i}, t_i, t_{i+1})$$

One evaluation	Several evaluations
Ask for the full noise-to-data jump	Ask for a sequence of shorter learned jumps
The random input supplies diversity	Approximation quality can improve with refinement

The network receives both the current coordinate and the destination coordinate.

Boffi et al., Flow Maps via Self-Distillation (2025) · Geng et al., MeanFlow (2025)

Consistent finite-time jumps

$$F_{s,s}(x) = x$$

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

No elapsed generation time	The map returns its input
An intermediate coordinate r	A long jump agrees with two consecutive shorter jumps

These identities describe the map, not a particular teacher network.

Learning a map from teacher trajectories

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

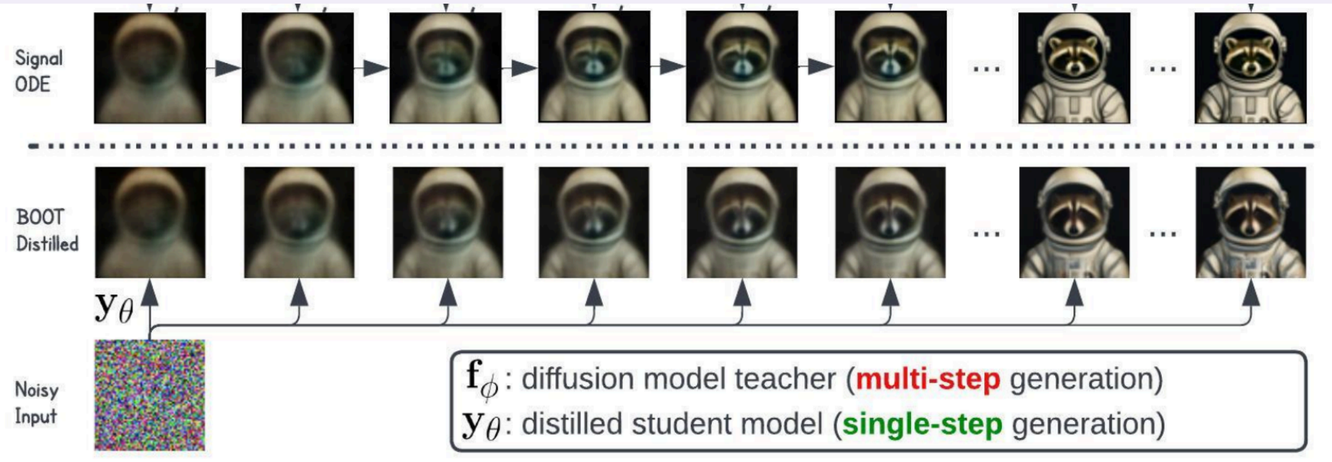
$$\mathcal{L}_{\mathrm{map}} = \mathbb{E}\| F_\theta(x_s, s, t) - x_t^{\mathrm{T}} \|^2$$

Teacher	Compute the destination using a trained field v_{T}
Student	Predict that destination in one evaluation
Pairing	Keep the starting point and teacher destination matched

This target preserves a specific input-output correspondence.

BOOT: predicting a teacher trajectory

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 $y_\theta(n, t)$:

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

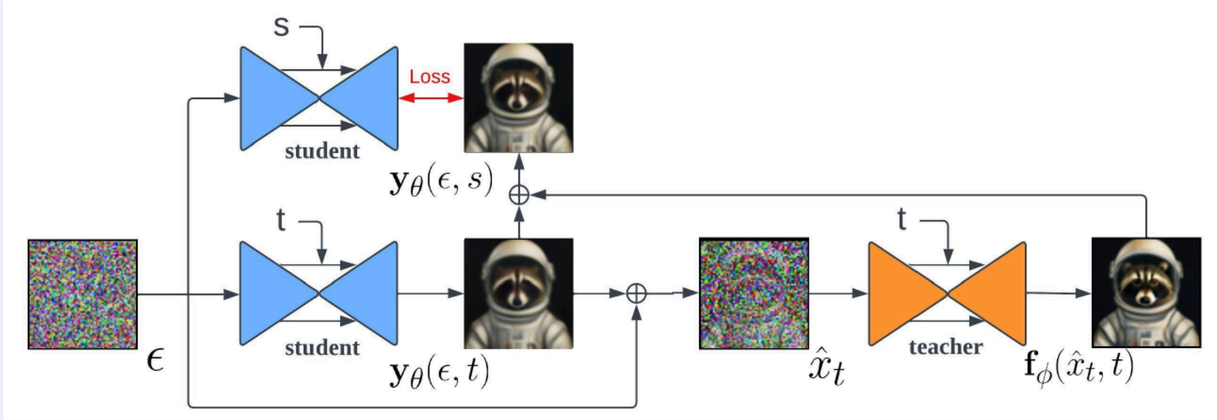
$$\hat{x}_t \approx F_{0,t}(n)$$

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: bootstrapping neighboring times



BOOT, Figure 2. Original paper notation: s is cleaner than t. Blue: shared student. Orange: frozen teacher.

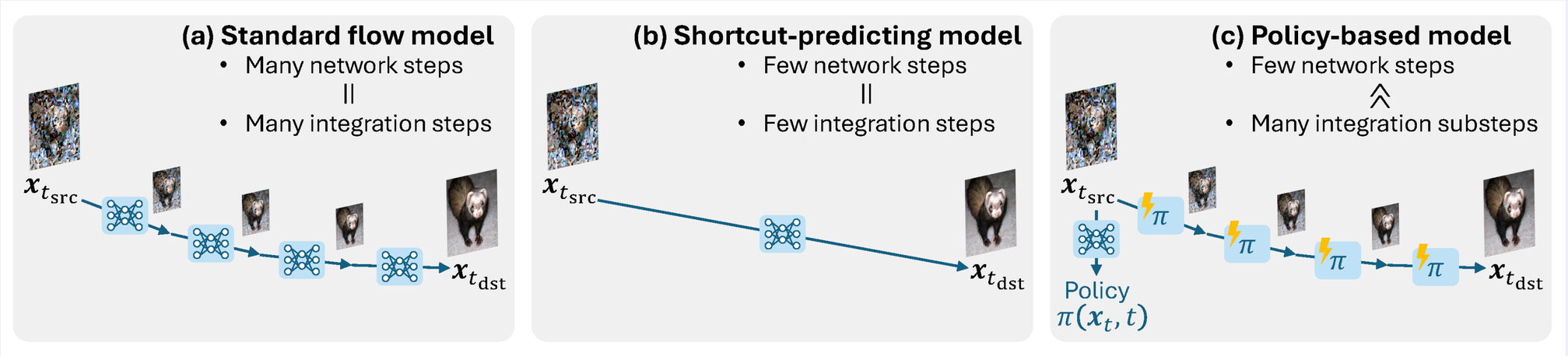
Local target	Predict a signal, reconstruct its noisy state, and apply a short teacher update.
Student update	From the same noise, predict the next signal and match the stopped target.

Training needs noise and optional prompts, not real images. A noise-end boundary loss anchors the trajectory.

Local teacher corrections replace complete teacher rollouts during training.

Gu et al., BOOT (ICML 2024)

π-Flow: a policy for the generation trajectory



π-Flow, Figure 2. The expensive network predicts a policy that is reused during integration.

$$\pi_s = G_\phi(x_s, s, C), \qquad \frac{dx_\tau}{d\tau} = \pi_s(x_\tau, \tau), \quad s \le \tau \le t$$

One network call predicts compact policy parameters. The policy returns a generation velocity, not an environment action.

A few network calls can support many inexpensive integration substeps.

A fixed policy produces changing velocities

The DX policy predicts clean estimates on a time grid, then interpolates between them.

$$\pi_s^{\mathrm{DX}}(x_\tau, \tau) = \frac{\hat{y}_s(\tau) - x_\tau}{1 - \tau}, \qquad s \le \tau < 1$$

Policy	Cached network output	Calculation at each substep
Dynamic clean estimate (DX)	A grid of clean-image estimates	Interpolate $\hat{y}_s(\tau)$ and evaluate the velocity
Gaussian mixture (GMFlow)	Mixture weights, means, and variance	Compute a clean estimate from the cached mixture and current (x_τ, τ)

The GMFlow policy lets the denoising estimate respond to the current state.

Within a segment, fixed policy parameters produce velocities that change with state and time.

The teacher velocity

Start with a model pretrained using the usual flow-matching objective:

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

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

Freeze its learned weights $\theta^\star$. Query it at the student-visited state:

$$v_{\mathrm{T}}(\bar{x}_\tau, \tau, C) := v_{\theta^\star}(\bar{x}_\tau, \tau, C)$$

T: teacher. $\bar{x}_\tau$: student rollout state. τ: generation time. C: conditioning.

Teacher inference supplies the local velocity target. No full teacher rollout is required.

π-Flow training on its own rollout

$$\pi_s = G_\phi(x_s, s, C), \qquad \pi_s^{\mathrm{D}} = \mathrm{sg}[\pi_s]$$

$$\bar{x}_\tau = \mathrm{sg}[\mathrm{ODESolve}(\pi_s^{\mathrm{D}}, x_s, s, \tau)]$$

$$\mathcal{L}_{\pi\text{-ID}} = \mathbb{E}\,\|\pi_s(\bar{x}_\tau, \tau) - \mathrm{sg}[v_{\mathrm{T}}(\bar{x}_\tau, \tau, C)]\|^2$$

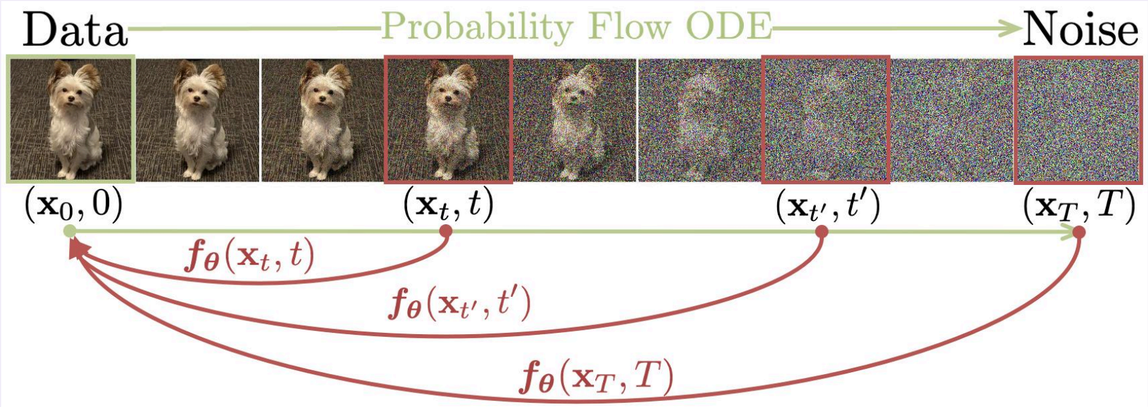
Visit	Integrate the detached student policy to obtain its own intermediate states.
Imitate	Match the policy velocity to v_{T}, the frozen teacher network output at the same state.

T: teacher. sg stops gradients through the rollout and teacher target.

Update the live policy using both velocities evaluated at the same student-visited state.

Consistency: one 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.

A consistency-distillation target

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

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

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 $f(x, 1) = x$.

Song et al., Consistency Models (2023) · Boffi et al., Flow Maps via Self-Distillation (2025)

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.

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.

MeanFlow: average velocity over an interval

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

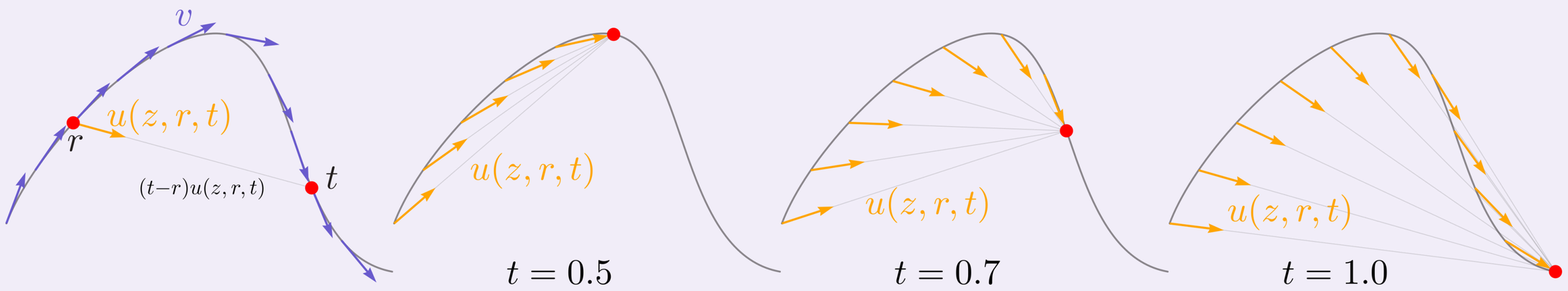
$$F_{s,t}(x_s) = x_s + (t - s) u(x_s, s, t)$$

Instantaneous velocity v	Average velocity u
Tangent direction at the current point	Net displacement divided by the interval length
Describes an infinitesimal change	Describes a finite jump

The integral follows the generative ODE trajectory, not an arbitrary straight training pair.

Geng et al., MeanFlow (2025)

A finite displacement and a local direction



MeanFlow, Figure 3. Purple: local direction; orange: interval displacement. Original r,t correspond to our two interval coordinates.

A long displacement need not point along the initial tangent.

The average-velocity network must know both where the interval starts and where it ends.

A finite jump can follow a curved flow without tracing every intermediate point.

Deriving 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.

The derivative follows the moving input

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

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)

The velocity available 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.

A trainable MeanFlow 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)

The training computation

Sample	A data image y, noise n, and $s \le t$, including diagonal examples
Construct	The mixture r_s and pair velocity $\delta = y - n$
Evaluate	Average velocity and its JVP in direction $(\delta, 1, 0)$
Update	Match the detached identity target

$$t = s: \qquad u_{\text{target}} = \delta, \qquad \mathcal{L}_{\text{local}} = \left\| u_\theta(r_s, s, s) - \delta \right\|^2$$

Diagonal examples recover local flow matching. Finite intervals teach larger jumps.

One-step and few-step generation



MeanFlow, Figure 1: a published one-evaluation ImageNet sample.

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

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

For a shorter jump:

$$x_t \approx x_s + (t - s)u_\theta(x_s, s, t)$$

Inference uses the learned average velocity; it does not compute the training JVP.

iMF: predict the derivative direction

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

$$\tilde{v}_\theta = u_\theta - (t - s)\,\text{sg}[D_s^{\text{pred}} u_\theta]$$

$$\mathcal{L}_v = \mathbb{E}\|\tilde{v}_\theta - \delta\|^2$$

Use the predicted marginal velocity v_θ inside the JVP. The sampled pair velocity δ appears only as the regression target.

The prediction now depends on the noisy input, without using its paired target direction.

Guidance in a few-step generator

$$F_\theta(x_s, s, t, C, w)$$

Condition C	Specify the desired content
Guidance scale w	Specify how strongly generation follows that condition
Learned jump	Train the map to represent the corresponding guided transformation

iMF treats guidance settings as explicit conditioning variables.

A guided endpoint map is not generally a linear combination of unguided endpoint maps.

What a fast model learns

Object	Meaning
Velocity $v(x_s, s)$	Local motion along a generative flow
Flow map $F_{s,t}(x_s)$	A finite-time destination on that flow
Endpoint map $f(x_s, s)$	The data-end destination
Average velocity $u(x_s, s, t)$	Displacement divided by interval length
Velocity policy $\pi_s(x_\tau, \tau)$	A compact rule reused across integration substeps

Finite jumps and reusable policies reduce expensive network evaluations.

Boffi et al., Flow Maps via Self-Distillation (2025) · Geng et al., MeanFlow (2025) · Chen et al., π-Flow (ICLR 2026)

Next: distribution matching

Must a fast generator preserve the same noise-to-image correspondence?

Today’s constraint	Another possibility
Learn a trajectory’s destination or its velocity policy	Make the generated sample distribution match the target
Compare paired predictions, flow relations, or local velocities	Compare distributions without fixing every correspondence

This opens a different route to few-step generation, and can complement trajectory-based training.

Yin et al., Distribution Matching Distillation · Zhou et al., Inductive Moment Matching (2025) · Deng et al., Drifting Models (2026) · Zheng et al., Causal-rCM (2026)