How to Train Your Video World Model for Long-Horizon Rollout

Anonymous authors
Paper under double-blind review

Abstract

Video world models must sustain autoregressive generation long after their initial observations leave the attention window. How should they be trained for this regime? Recent work advocates conditioning on clean history during pre-training to match inference, but the pretrained model is often only the starting point for further training. We study how this choice affects the complete pipeline for action conditioned world models without a separate bidirectional teacher. In this setting, the pretrained causal model provides both the student initialisation and the teacher for post-training. On Minecraft, we compare Diffusion Forcing, clean prefix teacher forcing, and chunked teacher forcing at equal pre-training compute. Each model undergoes flow map finetuning followed by Self Forcing against its own teacher, with evaluation over 1000 generated frames. The recipe rankings change after post-training. Diffusion Forcing initially performs worst beyond the attention window, but subsequently outperforms both alternatives through frame 400 and matches chunked teacher forcing at longer horizons. Its Fréchet Video Distance over frames 401–1000 falls from 329 to 86, compared with 132 to 91 for chunked teacher forcing. Clean prefix teacher forcing is best at neither stage. The post-training stages play complementary roles: qualitative rollouts suggest that flow map finetuning increases dynamic content and Self Forcing improves its persistence. These results show that alignment with inference during pre-training alone does not determine rollout quality. Training recipes should be evaluated through the full pipeline and at long horizons.

1 Introduction

World models predict future observations from past observations and actions, enabling policies to be evaluated and trained without access to the real environment (Ha & Schmidhuber, 2018; Alonso et al., 2024; Guo et al., 2026). Autoregressive (AR) video models trained with diffusion or flow matching offer a natural approach (Bruce et al., 2024; Parker-Holder et al., 2024; Decart et al., 2024; Guo et al., 2025). However, errors accumulate as generated frames become inputs to later predictions. Once the initial observation leaves the self-attention window, that window contains only generated latents; the initial observation remains available through cross-attention. Evaluating beyond this window is therefore essential for understanding whether a world model can sustain interaction.

A central training choice is how to present the conditioning history. Under flow matching (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023), each frame is assigned a flow time that determines its noise level. At inference, the generated history is presented clean, but training can present it differently. Diffusion Forcing (DF) (Chen et al., 2024) independently noises every frame and supervises all frames in a clip, rarely exposing the model to clean history. Causal Forcing (Zhu et al., 2026), Causal Forcing++ (Zhao et al., 2026), and AAPT (Lin et al., 2025) instead advocate pre-training on clean history to align training with inference.

These arguments concern text to video pipelines that use a pretrained bidirectional teacher during distillation or post-training (Huang et al., 2025; Yin et al., 2025; Team Wan et al., 2025). In our action conditioned setting, no bidirectional teacher is available. The pre-trained causal model must provide both the student's initialisation and the teacher scores used during post-training. We therefore ask: which pre-training recipe leads to the best final world model?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

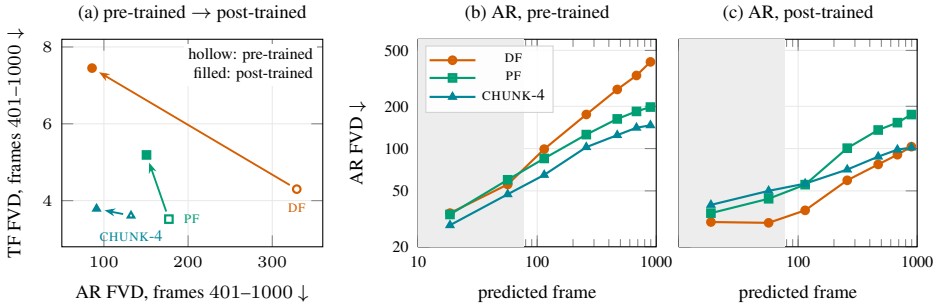


Figure 1: **Recipe rankings change after post-training.** **(a)** FVD over frames 401–1000 with generated history (AR) or ground truth history (TF), before (hollow) and after (filled) post-training. **(b, c)** Rollout quality before and after post-training. Shading marks the period when the initial observation remains in the attention window. DF starts worst beyond this window, then leads through frame 400 and matches CHUNK-4 thereafter. Results use 766 held-out clips, with 10 Euler steps before and 4 flow-map steps after post-training. Lower FVD is better.

We study this question on Minecraft (Baker et al., 2022). At equal pre-training compute, we compare DF, clean prefix teacher forcing (PF), which predicts each frame from its clean history, and chunked teacher forcing, which predicts several frames ahead of each clean prefix. Each model undergoes flow map finetuning with AnyFlow (Gu et al., 2026), followed by Self Forcing (Huang et al., 2025) with distribution matching distillation (DMD) (Yin et al., 2024) against its own causal teacher. We evaluate every stage over 1000 frames of AR generation, more than ten times the attention window.

The ranking changes after post-training (Fig. 1). Chunked teacher forcing performs best after pre-training, while DF performs worst beyond the attention window. After post-training, DF performs best through frame 400 and matches chunked teacher forcing at longer horizons. PF, the recipe designed to match inference, is best at neither stage. These results show that matching the inference history during pre-training is insufficient to select a recipe for the complete pipeline.

Our contributions are:

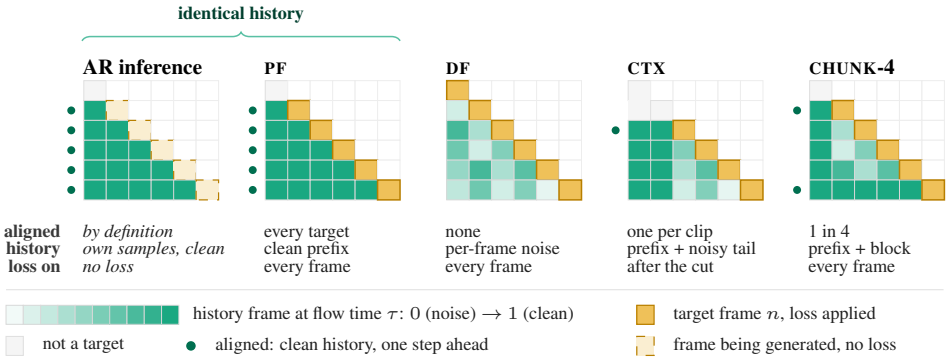
- **C1** A controlled comparison of the full pre-training and post-training pipeline at equal pre-training compute. Evaluation over 1000 frame rollouts shows that recipe rankings change after post-training.
- **C2** Evidence from predictions conditioned on ground truth and history noise experiments that a mismatch between training and inference contributes to DF's initial rollout deficit.
- **C3** An analysis of the distinct effects of post-training. Qualitative rollouts suggest that flow map finetuning increases dynamic content and Self Forcing improves its persistence. The complete post-training pipeline changes the preferred cache policy in favour of retaining K/V as the window slides.

2 Preliminaries and Background

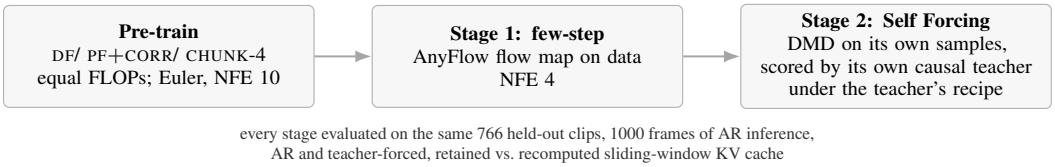
In this section, we fix notation for autoregressive flow matching (Sec. 2.1), introduce the history profile, which describes training and inference in one object (Sec. 2.2), define exposure bias and the inference-time corrections that target it (Sec. 2.3), and describe the two methods our post-training builds on: flow maps (Sec. 2.4) and Self Forcing with distribution matching distillation (Sec. 2.5). Sec. 2.6 then reviews prior work.

2.1 Autoregressive flow matching world models

A clip is a sequence of latents $\boldsymbol{z}_{1:N}$ from the causal WAN VAE (Team Wan et al., 2025), with the first latent given. The first latent encodes one video frame and every later latent four, so "frame" below always means a real video frame and n indexes latents. The actions are grouped the same way: $\boldsymbol{a}_n$ denotes the actions that produce the frames of latent n, with $\boldsymbol{a}_1$ empty. Flow matching (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023) uses the interpolant $\boldsymbol{z}^t = t\boldsymbol{z} + (1-t)\boldsymbol{\epsilon}$ with



(a) what each recipe shows the model



(b) the pipeline every recipe goes through

Figure 2: **Training recipes and the shared pipeline. (a)** Each row shows one prediction target and its history, with colour indicating the history's noise level. Recipes vary which history frames are clean or noisy, while AR inference uses clean generated history. **(b)** Each recipe undergoes pre-training, flow map finetuning on data, and Self Forcing against its own causal teacher.

$\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, so t=1 is clean data and t=0 is noise. A network $v_\theta(\boldsymbol{z}_n^t, t \mid h_n)$ predicts the velocity of latent n given its history h_n (Def. 1), and is trained by regressing onto the conditional velocity,

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{n,t,\boldsymbol{z},\boldsymbol{\epsilon}} \left\| v_\theta(\boldsymbol{z}_n^t, t \mid h_n) - (\boldsymbol{z}_n - \boldsymbol{\epsilon}_n) \right\|_2^2. \tag{1}$$

The network is a diffusion transformer with frame-causal attention and action conditioning, so its output at latent n depends only on latents $\leq n$. We use t for the flow time of a target, τ for the flow times its history is presented at, and s for the destination time of a flow map (Sec. 2.4).

AR inference. To generate latent n the model integrates the ODE from t=0 to t=1 with a fixed number of function evaluations (NFE), commits the result $\hat{\boldsymbol{z}}_n$, and appends it to a sliding window of W=20 latents. Committed latents are stored in a key-value (KV) cache. When a latent leaves the window, the cache can either *retain* the keys and values of the remaining latents, which were computed while the evicted latent was still visible, or *recompute* them from the latents now in the window. Retaining costs nothing extra but is not exact whereas recomputing is exact but costs a forward pass over the entire window per generated latent. The ground-truth first latent leaves the self-attention window at frame 77, after which that window contains only generated latents. The first latent remains available through a separate cross-attention layer (App. B.2). Retained representations can carry information from evicted latents. While this extends the effective history, it also introduces dependencies absent during pre-training, further widening the train-inference gap (Fig. 3).

2.2 History profiles

Every latent in the window, including those being conditioned on, is presented to the network at some flow time. We separate what the history contains from how it is presented.

Definition 1 (History profile)**.** *To predict latent n the model is shown the latents $\boldsymbol{z}_{<n}$ at flow times $\boldsymbol{\tau}_{<n} = (\tau_1, \ldots, \tau_{n-1}) \in [0,1]^{n-1}$, as $\boldsymbol{z}_i^{\tau_i} = \tau_i \boldsymbol{z}_i + (1-\tau_i)\boldsymbol{\epsilon}_i$. We call $\boldsymbol{\tau}_{<n}$ the* history profile *and write $h_n = (\boldsymbol{z}_{<n}^{\boldsymbol{\tau}}, \boldsymbol{\tau}_{<n}, \boldsymbol{a}_{\leq n})$ for everything the network conditions on.*

The latents in the history are its *content* and the profile is its *presentation*. When the content is the model's own samples we write $\hat{\boldsymbol{z}}$ in place of $\boldsymbol{z}$.

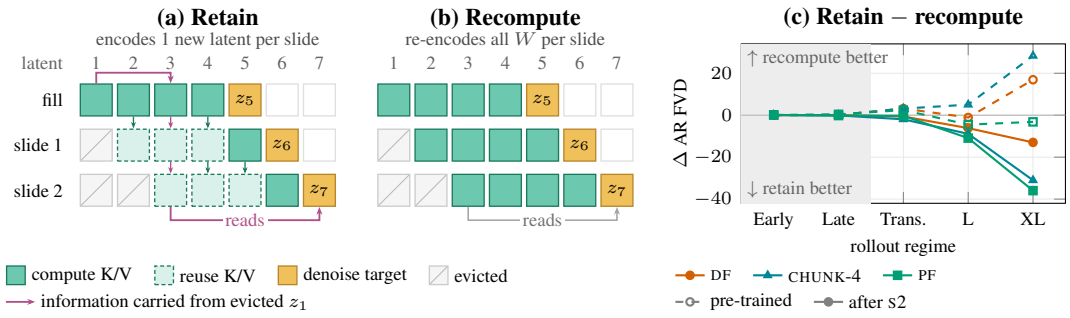


Figure 3: **Retaining versus recomputing the KV cache.** When the window slides, retain **(a)** keeps the surviving K/V and encodes only the newest latent, while recompute **(b)** re-encodes every latent in the window, exactly as in pre-training. Retained K/V were computed while evicted latents were still visible, so they can carry information from outside the window (magenta); schematic drawn with $W{=}4$. **(c)** AR FVD of recompute minus retain at NFE 4 on 766 held-out clips; positive values favour retention. Shading marks frames 1–76, where nothing has been evicted and the policies coincide. Before post-training, retaining hurts CHUNK-4 at long horizons and makes no significant difference for DF or PF; after S2, every recipe prefers it.

A training recipe R is a distribution q_R over history profiles and target flow times, together with a choice of which latents carry a loss. Its objective is Eq. (1) with data content and the presentation drawn from q_R,

$$\mathcal{L}_R(\theta) = \mathbb{E}_{n,\,(\boldsymbol{\tau}_{<n},t)\sim q_R,\,\boldsymbol{\epsilon}} \left\| v_\theta(\boldsymbol{z}_n^t, t \mid h_n) - (\boldsymbol{z}_n - \boldsymbol{\epsilon}_n) \right\|_2^2, \qquad h_n = (\boldsymbol{z}_{<n}^{\boldsymbol{\tau}}, \boldsymbol{\tau}_{<n}, \boldsymbol{a}_{\le n}). \tag{2}$$

AR inference is fixed so that the content is the model's own samples and the profile is clean, $(\hat{\boldsymbol{z}}_{<n}, \boldsymbol{\tau}{=}\mathbf{1})$. Pre-training recipes condition on data, so they choose only the presentation whereas post-training on the model's own samples changes the content. We call a training target *aligned* if it is presented exactly as AR inference queries it, one latent ahead of a clean history, $\boldsymbol{\tau}_{<n}{=}\mathbf{1}$. The recipes we study (Sec. 3.1) differ in how their profiles are distributed and in how many of their targets are aligned.

2.3 Exposure bias and inference-time corrections

For a causal model p, the reverse KL to the data distribution decomposes over latents with the histories drawn from the model itself,

$$D_{\mathrm{KL}}(p \,\|\, p_{\mathrm{data}}) = \sum_n \mathbb{E}_{\hat{\boldsymbol{z}}_{<n}\sim p}\Big[D_{\mathrm{KL}}\big(p(\cdot \mid \hat{\boldsymbol{z}}_{<n}) \,\big\|\, p_{\mathrm{data}}(\cdot \mid \hat{\boldsymbol{z}}_{<n})\big)\Big]. \tag{3}$$

Training on data histories controls each conditional only where $\boldsymbol{z}_{<n} \sim p_{\mathrm{data}}$, while AR inference evaluates it on $\hat{\boldsymbol{z}}_{<n} \sim p$. Error at AR inference therefore has two sources: *on-manifold error*, the error of the conditional on data histories, and *exposure error*, the additional error on the model's own, slightly off-manifold histories, which compounds over the rollout. Throughout this paper, we use teacher-forced (TF) evaluation to measure on-manifold error by conditioning each prediction on ground-truth history and evaluating the first prediction. The gap between AR and TF evaluation then provides a proxy for the *AR drift gap*.

Inference-time corrections. We also consider corrupting the committed history at inference time as a regularisation mechanism. *History noise* moves the history to a single flow time $\bar{\tau} < 1$.

2.4 Few-step generation with flow maps

A flow map (Boffi et al., 2025; Geng et al., 2025) transports a sample between any two flow times in one evaluation,

$$X_\theta(\boldsymbol{z}^t, t, s \mid h) = \boldsymbol{z}^t + (s-t)\, u_\theta(\boldsymbol{z}^t, t, s \mid h), \qquad 0 \le t \le s \le 1, \tag{4}$$

where u_θ is the average velocity over $[t, s]$ and $u_\theta(\boldsymbol{z}^t, t, t \mid h)$ is the instantaneous velocity. It is trained on the self-consistency identity $u(\boldsymbol{z}^t, t, s) = v + (s-t)\frac{\mathrm{d}}{\mathrm{d}t}u(\boldsymbol{z}^t, t, s)$, with the right-hand

side under a stop-gradient and the total derivative taken along the flow. AnyFlow (Gu et al., 2026) applies this to video, conditioning the network on the destination time s as well as t, and samples with any number of steps by chaining X_θ over a partition of $[0, 1]$.

2.5 Self Forcing with distribution matching distillation

Self Forcing (Huang et al., 2025) post-trains a few-step causal student G_θ on its own AR samples, $\hat{\boldsymbol{z}}_n = G_\theta(\boldsymbol{\xi}_n \mid h_n)$ with $\boldsymbol{\xi}_n \sim \mathcal{N}(0, I)$ and a KV cache, so that training sees the content inference will see. Our G_θ uses the compressed denoising pass described in Sec. 3.3. It trains with distribution matching distillation (DMD) (Yin et al., 2024), which minimises a reverse KL between the noised student and teacher distributions. Writing $D(\boldsymbol{z}^t, t) = \boldsymbol{z}^t + (1-t)v(\boldsymbol{z}^t, t)$ for the clean-data prediction from a single teacher or critic velocity evaluation, the update on a noised student sample $\hat{\boldsymbol{z}}^t$ is

$$g = w(t)\big(D_\psi(\hat{\boldsymbol{z}}^t, t) - D_T(\hat{\boldsymbol{z}}^t, t)\big), \qquad \mathcal{L}_{\mathrm{DMD}} = \tfrac{1}{2}\big\|\hat{\boldsymbol{z}} - \mathrm{sg}(\hat{\boldsymbol{z}} - g)\big\|_2^2, \tag{5}$$

where T is a frozen teacher and ψ a critic trained online on detached student samples to track the student's own distribution. The update moves the student's sample from the critic's prediction towards the teacher's. In prior work (Zhu et al., 2026; Zhao et al., 2026), the teacher is bidirectional: it denoises a whole window at one flow time, $D_T(\hat{\boldsymbol{z}}^t_{1:W}, t)_n$, so its score for latent n also reflects how well $\hat{\boldsymbol{z}}_n$ fits the latents after it, while the earlier latents of the window are noised along with it rather than conditioned on.

2.6 Related work

Causal video post-training. CausVid distils a bidirectional video teacher into a causal student using DMD (Yin et al., 2025); Self Forcing trains on the student's own AR rollouts (Huang et al., 2025). Causal Forcing and Causal Forcing++ initialise the student through causal ODE or consistency distillation, respectively, retaining a bidirectional teacher for final DMD (Zhu et al., 2026; Zhao et al., 2026). Self-Forcing++ scores segments of long rollouts with a short-video teacher (Cui et al., 2026). AAPT instead post-trains adversarially against real clips and reports one-minute generation (Lin et al., 2025). We compare recipes through the completed pipeline at fixed pre-training compute, using each causal base model as both student initialisation and DMD teacher. We discuss related work on exposure bias, few-step generation, and interactive world models in Appendix D.

3 Pre-training and post-training design choices

Every model goes through the same pipeline (Fig. 2(b)). It is pre-trained on data with a recipe R (Sec. 3.1) and then post-trained in two stages: few-step finetuning on data (S1, Sec. 3.2), and Self Forcing against its own pre-trained model as a causal teacher (S2, Sec. 3.3). The recipe is the only thing that varies between pipelines.

3.1 Pre-training recipes

We start from three recipes used in prior work, DF, PF and CTX, and add two ingredients that separate the factors CTX changes together: chunked prediction and history corruption. Fig. 2(a) draws the history profile of every recipe. Table 2 in Appendix A.4 summarises their supervision and relative cost. All recipes draw the target flow time from a logit-normal prior (Esser et al., 2024) except DF, which uses a uniform one. We retain the uniform prior to follow the DF literature. Thus, comparisons with DF vary both history presentation and the flow-time prior; they compare complete recipes and do not isolate the effect of history presentation alone.

Diffusion Forcing (DF). Chen et al. (2024) draw an independent flow time for every latent and place a loss on each, so $\tau_i \sim \mathcal{U}[0, 1]$ i.i.d. and every latent is a target in one forward pass. No target is aligned: the clean profile $\boldsymbol{\tau}=\mathbf{1}$ has probability zero under q_{DF}.

Clean-prefix teacher forcing (PF) (Zhu et al., 2026). PF trains every latent against its own clean prefix, $\boldsymbol{\tau}_{<n}=\mathbf{1}$ for every n, so every target is aligned on uncorrupted clips. The history corruption described below is applied to 10% of training clips. It matches the teacher-forcing stage of Causal

Forcing . A clean copy $\boldsymbol{z}_{1:N-1}$ and a noisy copy $\boldsymbol{z}_2^{t_2}, \ldots, \boldsymbol{z}_N^{t_N}$ share an attention mask: clean latents attend causally within the clean sequence, while each noisy latent attends to preceding clean latents. We implement the mask with FlexAttention, whose block sparsity means the doubled mask is never materialised, and one training step costs ρ=1.9× a standard forward pass.

Context-length sampling (CTX) (Po et al., 2025). CTX draws one context length per clip, $c \sim \mathcal{U}\{1, \ldots, N-1\}$, presents $\boldsymbol{z}_{\leq c}$ clean and the latents after it at their own flow times, and places a loss on the latents after the cut. Only the first of them, n=c+1, has a clean history, so one target per uncorrupted clip is aligned. The CTX−TAIL ablation supervises only this first target, removing the loss on all later targets.

Chunked teacher forcing (CHUNK-B). CHUNK-B predicts several latents ahead of each clean prefix. The targets are partitioned into consecutive blocks of B latents such that a target in the block starting at latent b sees the clean prefix $\boldsymbol{z}_{<b}$ and the earlier members of its own block at their own flow times, τ_i=1 for $i < b$ and τ_i=t_i for $b \leq i < n$. On uncorrupted clips, only the first member of each block is aligned, and B=1 recovers PF. It uses the same doubled clean-noised sequence as PF with a different mask; because each target attends to the same number of latents for any B, the cost remains ρ.

History corruption. Because PF and CHUNK-B present a clean prefix by construction, noise on the history becomes a separate knob: on a fraction p of clips the clean prefix is presented at $\tau_i \sim \mathcal{U}[\tau_{\min}, 1]$ instead. We sweep $\tau_{\min}$ in Sec. 4 but everywhere else PF and CHUNK-B use p=0.1 and $\tau_{\min}$=0.7.

Which recipes go forward. Post-training is run for DF, PF and CHUNK-4. We compare CTX in pre-training only, since PF and CHUNK-B vary its factors one at a time. Jump prediction, which deletes the latents between a target and its clean prefix instead of noising them (App. A.3), is a second multi-step recipe. We take CHUNK-4 forward because its quarter-budget AR FVD over frames 401–1000 is lower than JUMP-3's (216 versus 230; Table 7).

3.2 POST-TRAINING STAGE 1: FEW-STEP FINETUNING

S1 turns each pre-trained model into a flow map (Eq. (4)) with AnyFlow, initialising u_θ from v_θ and training on data. PF and CHUNK-B are finetuned under their own pre-training recipe, with histories h_n drawn from q_R as in Eq. (2). For DF, we instead sample histories using p=1.0 and $\tau_{\min}$=0.9, to balance fidelity to its pre-training distribution with alignment to AR inference. At inference, the interval solver chains X_θ over a partition of $[0, 1]$ at the chosen NFE.

Why flow maps may reach narrower modes. Eq. (1) is minimised by the average of the conditional velocities of all paths through a point, $\mathbb{E}[\boldsymbol{z} - \boldsymbol{\epsilon} \mid \boldsymbol{z}^t]$. Where paths heading to different modes overlap, their velocities are averaged into one direction (Frans et al., 2025). With many solver steps the resulting curved trajectories are followed closely, but a few large steps along the averaged direction land between the modes. We hypothesise that this averaging also suppresses rare modes in favour of common ones (Figs. 4 and 7). A flow map regresses the displacement over $[t, s]$ and is conditioned on s, so trajectories that overlap at time t but separate later are assigned different average velocities. We hypothesise that this lets the model sample narrow modes, which in Minecraft include dynamic entities such as villagers: they occupy a small part of the frame and appear in a small fraction of clips. Sec. 4.4 examines this with directional KL and qualitative rollouts.The measured directional KL and an analytic illustration appear in Figs. 4 and 5; the full illustration is in Fig. 7.

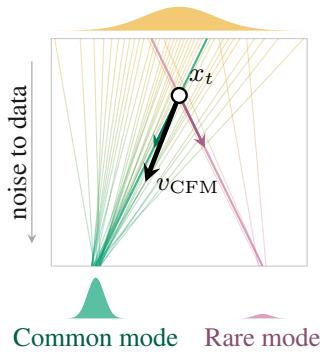


Figure 4: Flow-matching target averaging favors dominant modes, potentially collapsing rare modes and their associated environment dynamics.

3.3 POST-TRAINING STAGE 2: SELF FORCING WITH A CAUSAL TEACHER

Two train-inference gaps remain after S1. TF uses data histories and computes their representations within the training window; AR inference uses generated histories and retains K/V across window slides (Fig. 3). S2 applies Self Forcing (Sec. 2.5) to the S1 student, with the recipe's own pre-trained model as the teacher. The teacher is causal: it scores latent n given its history, presented under the recipe it was trained with,

$$D_T\big(\hat{\boldsymbol{z}}_n^t, t \mid h_n\big), \qquad h_n = (\hat{\boldsymbol{z}}_{<n}^{\boldsymbol{\tau}}, \boldsymbol{\tau}_{<n}, \boldsymbol{a}_{\le n}), \quad \boldsymbol{\tau}_{<n} \sim q_R. \tag{6}$$

The DF teacher sees the student's history under independent per-latent noise, and the PF and CHUNK-4 teachers see their pre-training history corruption setup. S2 combines DMD on generated rollouts with the AnyFlow loss on real data. The DMD term corrects exposure error relative to the teacher, while data co-training provides supervision beyond the teacher's distribution.

Rollout and cache. The student rolls out from one ground-truth latent past the window: 18 latents without gradient, then 19 scored latents. For each rollout, we sample a nominal NFE M uniformly from $\{1, 4, 8, 16, 50\}$ and an interval k uniformly from its M-step grid. We compress denoising to $0 \to t_k \to t_{k+1} \to 1$, omitting duplicate endpoints, so each latent takes at most three flow-map evaluations. Warmup and scored latents share this grid. Gradients pass through all denoising steps of each scored latent, but not through earlier generated history. The student rolls out with retained K/V, the cache policy used at inference.

4 EXPERIMENTS

In this section, we evaluate the pipeline of Sec. 3 end to end, organised around the following five research questions:

- **RQ1** Which pre-training recipe gives the most stable long-horizon AR inference, and which ingredient is responsible? (Sec. 4.1)
- **RQ2** Does the ranking of the pre-training recipes survive post-training? (Sec. 4.2)
- **RQ3** Why does DF gain the most from post-training? (Sec. 4.3)
- **RQ4** What does each post-training stage contribute? (Sec. 4.4)
- **RQ5** Can history noise at inference substitute for post-training? (Sec. 4.5)

RQ1 concerns pre-training alone, and answers the question for a practitioner who stops there. RQ2 is the central result of the paper. RQ3 and RQ4 explain it, and RQ5 asks whether a cheaper, training-free correction could have delivered the same.

Setup. Every model generates 1000 frames (250 latents) of AR inference on each of the 766 held-out test clips long enough for it, starting from the ground-truth first latent and following the clip's associated actions (App. B.4). Pre-trained models are sampled with 10 Euler steps per latent and post-trained models with 4 flow-map steps, and every rollout retains the KV cache unless stated otherwise. We report FVD over windows of predicted frames. Over frames 1–76 the ground-truth first latent is still in the self-attention window, so this span is exactly the training sequence length (we shade it in every horizon plot). From frame 77 onwards, the self-attention window contains only generated latents, although the initial observation remains available through cross-attention. Frames 401–1000 are the regime this paper is most interested in. Alongside AR FVD we report TF FVD, which conditions every prediction on ground truth and so measures the one-step conditional in isolation. AR minus TF FVD is our proxy for the drift gap of Sec. 2.3. Note that pre-training recipes are compared at equal training FLOPs (App. A.4) on the final checkpoint. Each recipe is trained once at each reported budget; the two budgets use separate runs. We report a difference between two models as its value followed by its paired 95% bootstrap interval, *e.g.* $-5\,[-17, 5]$ (App. B.5). These intervals resample evaluation videos and do not capture training-run variability. Differences are computed before rounding.

4.1 RQ1: WHICH PRE-TRAINING RECIPE IS MOST STABLE AT LONG HORIZONS?

CHUNK-4 achieves the lowest AR FVD in every evaluated window under the retained cache policy (Table 1). Over frames 401–1000, it reaches 132, compared with 177 for PF ($-45\,[-59, -32]$) and

Table 1: **Rollout quality before and after post-training.** FVD on 766 held-out clips at equal pre-training compute. AR uses generated history, TF uses ground truth, and AR−TF measures their gap. Column ranges denote predicted frames. ρ is relative pre-training cost per clip. Results retain the KV cache. $\Delta_{\text{cache}} = \text{FVD}_{\text{recompute}} - \text{FVD}_{\text{retain}}$, so positive values favour retention.

				AR FVD ↓					FRAMES 401–1000		
	NFE	RECIPE	ρ	1–36	37–76	77–152	153–400	401–1000	TF FVD ↓	AR−TF	Δ_{cache}
PRE	10	DF	1.0	35	56	99	181	329	4.3	325	−10
		CTX−TAIL	1.0	47	91	131	209	271	5.6	266	+6
		CTX	1.0	62	71	75	114	207	7.9	199	−70
		PF	1.9	34	60	85	129	177	**3.5**	174	+7
		CHUNK-4	1.9	**28**	**47**	**65**	**105**	**132**	3.6	129	+9
POST	4	DF	1.0	**30**	**30**	**36**	**61**	**86**	7.5	78	+13
		PF	1.9	35	44	55	104	151	5.2	145	+36
		CHUNK-4	1.9	40	50	56	72	**91**	**3.8**	87	+31

329 for DF. Yet their TF FVD values are similar (3.6, 3.5, and 4.3). The main difference therefore emerges when models condition on their own predictions. Short evaluations obscure this difference: over the first 36 frames, DF and PF are indistinguishable (35 and 34, $+1\,[-3, 5]$).

The controlled ablations support predicting several latents ahead of a clean prefix (Fig. 10a, b). At a quarter of the training budget, adding mild history noise to PF reduces FVD over frames 401–1000 from 327 to 285 ($-42\,[-57, -27]$). Heavier noise gives back this gain ($+3\,[-15, 24]$ at τ_{min}=0), whereas CHUNK-4 reaches 216 at the same compute budget ($-69\,[-80, -57]$ against mild noise). Thus, chunked prediction helps more than history noise alone.

4.2 RQ2: Does the ranking survive post-training?

After post-training, DF achieves the lowest FVD in every window through frame 400, with every paired interval excluding zero (the narrowest is $-5\,[-9, -1]$ against PF over frames 1–36), and is statistically tied with CHUNK-4 over frames 401–1000 (86 versus 91, $-5\,[-17, 5]$). PF reaches 151, worse than DF by $65\,[50, 79]$ and than CHUNK-4 by $59\,[46, 72]$ (Table 1 and Fig. 1). The strongest pre-trained recipe is therefore not uniquely best after post-training.

The improvement persists at matched NFE. At the same 4 function evaluations per latent, pre-trained DF, PF, and CHUNK-4 score 693, 432, and 347 over frames 401–1000, respectively, compared with 86, 151, and 91 after post-training (Table 8). Nor does better rollout quality require better predictions from ground truth history: DF's TF FVD rises from 4.3 to 7.5 ($+3.2\,[2.9, 3.4]$) while its AR FVD falls from 329 to 86 (Table 1). Pre-training recipes must be evaluated through the full pipeline.

4.3 RQ3: Why does DF gain the most?

The evidence points to a mismatch between training and inference as one contributor. DF predicts well from ground truth history but degrades on its own rollouts. Re-noising its history to t_{hist}=0.85 reduces FVD over frames 401–1000 from 329 to 111 ($-218\,[-242, -197]$) without changing its weights (Fig. 10c). The best history noise settings give a smaller gain for PF (177 to 145, $-32\,[-44, -20]$) and no significant gain for CHUNK-4 (132 to 122, $-10\,[-20, 1]$). This suggests that DF's initial deficit does not simply reflect a weaker conditional model.

Post-training exposes the student to its own generated histories. The pre-training recipe also determines the teacher and how its history is presented (Sec. 3.3). We hypothesise that DF's experience with noisy histories helps it score imperfect student rollouts. Our experiments do not isolate this mechanism: the student initialisation and teacher change together. The history noise results support a mismatch explanation, but do not establish why DF benefits most from post-training.

4.4 RQ4: What does each post-training stage contribute?

The stages appear to have complementary effects: qualitative rollouts suggest that flow map finetuning increases dynamic content and Self Forcing improves its persistence. We assess them using directional KL (Fig. 5), qualitative rollouts (Fig. 9 in Appendix C.2), and cache comparisons (Fig. 3c). The KL

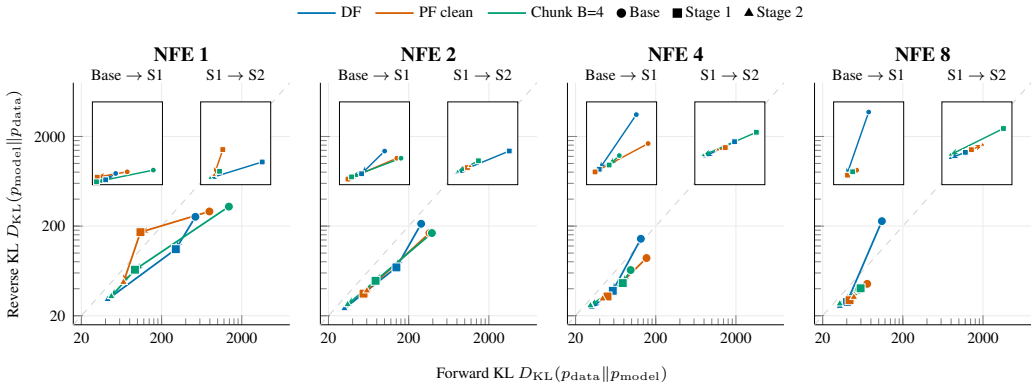


Figure 5: **Directional KL through post-training.** Forward KL (horizontal) and reverse KL (vertical) for the pre-trained model, S1, and S2 at step 300, using 128 validation videos over frames 1–1000. Columns vary the number of function evaluations per latent. Lower is better on both axes.

evaluation uses 128 validation videos and history flow time t_{hist}=0.98, whereas Table 1 uses 766 test clips with clean history.

Flow map finetuning. S1 generally reduces both forward and reverse KL. Lower forward KL is consistent with improved coverage of the data distribution. Qualitatively, moving characters and waterfalls appear more often, but characters can deform or disappear within a few frames (Fig. 9c–e). This suggests recovery of dynamic content but we do not directly quantify its frequency or persistence.

Self Forcing. S2 further improves both KL measures for DF and CHUNK-4 and the qualitative rollouts show characters persisting and moving across more frames (Fig. 9). It also trains with the retained KV cache used at inference. After the complete pipeline, retaining rather than recomputing the cache improves FVD over frames 401–1000 by 13 [3, 24], 36 [23, 50], and 31 [20, 40] for DF, PF, and CHUNK-4, respectively (Table 1). These cache comparisons measure the combined effect of both stages, while the KL figure separates their contributions.

4.5 RQ5: Can history noise at inference substitute for post-training?

History noise recovers some of the benefit, but does not replace post-training. At their best noise settings, pre-trained DF and CHUNK-4 reach FVD of 111 and 122 over frames 401–1000, compared with 86 and 91 after post-training with clean history (-25 [-36, -13] and -31 [-42, -20]; Fig. 10c). For PF, the corresponding results (145 and 151) are statistically indistinguishable ($+5$ [-10, 21]).

History noise can also improve post-trained FVD: over frames 401–1000, DF reaches 68, the lowest point estimate among the evaluated settings, at t_{hist}=0.8, against 86 with clean history (-18 [-28, -9]). However, the inspected noisy rollouts exhibit shifting backgrounds and changing object identities (Fig. 9h). These examples are not matched trajectories, so they illustrate a potential failure rather than quantify its prevalence. We use clean history for the main comparison and note that a lower FVD alone does not establish more coherent interaction over time.

5 Conclusion

Pre-training alignment alone does not determine long-horizon rollout quality. In our controlled Minecraft study, chunked teacher forcing performs best before post-training, whereas Diffusion Forcing leads through frame 400 and matches it at longer horizons after flow map finetuning and Self Forcing. This distinction matters when the pretrained causal model supplies both the student initialisation and the teacher and our findings suggest that its standalone rollout quality does not establish its value for the complete pipeline. We also found that the post-training stages have complementary effects, with qualitative evidence that flow map finetuning increases dynamic content and Self Forcing improves its persistence. We discuss limitations in App. E.

AI USE STATEMENT

In this work, we used generative AI tools for two tasks, *(i)* generating figures and *(ii)* polishing the text. For figures, AI tools helped write the plotting and diagram code that renders our results. Every plotted value comes from our own experiments and the authors checked each figure against the underlying results. For text, AI tools suggested edits to wording, grammar and concision of drafts written by the authors. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Michael Samuel Albergo and Eric Vanden-Eijnden. Building Normalizing Flows with Stochastic Interpolants. In *International Conference on Learning Representations*, 2023.

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for World Modeling: Visual Details Matter in Atari. In *Advances in Neural Information Processing Systems*, 2024.

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video PreTraining (VPT): Learning to act by watching unlabeled online videos. In *Advances in Neural Information Processing Systems*, 2022.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=cqDH0e6ak2.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative Interactive Environments. In *International Conference on Machine Learning*, 2024.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion. In *Advances in Neural Information Processing Systems*, 2024.

Justin Cui, Jie Wu, Ming Li, Tao Yang, Xiaojie Li, Rui Wang, Andrew Bai, Yuanhao Ban, and Cho-Jui Hsieh. Self-forcing++: Towards minute-scale high-quality video generation. In *International Conference on Learning Representations*, 2026.

Decart, Julian Quevedo, Quinn McIntyre, Spruce Campbell, Xinlei Chen, and Robert Wachen. Oasis: A universe in a transformer. https://oasis-model.github.io/, 2024. Open-weights 500M variant: https://huggingface.co/Etched/oasis-500m.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, 2025.

Samuel Garcin, Thomas Walker, Steven McDonagh, Tim Pearce, Hakan Bilen, Tianyu He, Kaixin Wang, and Jiang Bian. Beyond pixel histories: World models with persistent 3D state. In *International Conference on Machine Learning*, 2026.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Advances in Neural Information Processing Systems*, 2025.

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. In *European Conference on Computer Vision*, 2026.

Junliang Guo, Yang Ye, Tianyu He, Haoyu Wu, Yushu Jiang, Tim Pearce, and Jiang Bian. MineWorld: a real-time and open-source interactive world model on minecraft. *arXiv preprint arXiv:2504.08388*, 2025.

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-World: A Controllable Generative World Model for Robot Manipulation. In *International Conference on Learning Representations*, 2026.

David Ha and Jürgen Schmidhuber. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems*, 2018.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems*, 2025.

Shanchuan Lin, Ceyuan Yang, Hao He, Jianwen Jiang, Yuxi Ren, Xin Xia, Yang Zhao, Xuefeng Xiao, and Lu Jiang. Autoregressive Adversarial Post-Training for Real-Time Interactive Video Generation. In *Advances in Neural Information Processing Systems*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *International Conference on Learning Representations*, 2023.

Jack Parker-Holder, Philip Ball, Jake Bruce, Vibhavari Dasagi, Kristian Holsheimer, Christos Kaplanis, Alexandre Moufarek, Guy Scully, Jeremy Shar, Jimmy Shi, Stephen Spencer, Jessica Yung, Michael Dennis, Sultan Kenjeyev, Shangbang Long, Vlad Mnih, Harris Chan, Maxime Gazeau, Bonnie Li, Fabio Pardo, Luyu Wang, Lei Zhang, Frederic Besse, Tim Harley, Anna Mitenkova, Jane Wang, Jeff Clune, Demis Hassabis, Raia Hadsell, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 2: A large-scale foundation world model. Google DeepMind blog post, 2024. URL https://deepmind.google/discover/blog/genie-2-a-large-scale-foundation-world-model/.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Ryan Po, Yotam Nitzan, Richard Zhang, Berlin Chen, Tri Dao, Eli Shechtman, Gordon Wetzstein, and Xun Huang. Long-Context State-Space Video World Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Ryan Po, David Junhao Zhang, Amir Hertz, Gordon Wetzstein, Neal Wadhwa, and Nataniel Ruiz. MultiGen: Level-design for editable multiplayer worlds in diffusion game engines. *arXiv preprint arXiv:2603.06679*, 2026. URL https://arxiv.org/abs/2603.06679.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling Diffusion Models. In *International Conference on Machine Learning*, 2024.

Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-Guided Video Diffusion. In *International Conference on Machine Learning*, 2025.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252, 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and Advanced Large-Scale Video Generative Models. *arXiv preprint arXiv:2503.20314*, 2025.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Frédo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/54dcf25318f9de5a7a01f0a4125c541e-Abstract-Conference.html.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Min Zhao, Hongzhou Zhu, Kaiwen Zheng, Zihan Zhou, Bokai Yan, Xinyuan Li, Xiao Yang, Chongxuan Li, and Jun Zhu. Causal forcing++: Scalable few-step autoregressive diffusion distillation for real-time interactive video generation. *arXiv preprint arXiv:2605.15141*, 2026.

Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. In *International Conference on Machine Learning*, 2026.

APPENDIX

The appendix follows the order in which the main text introduces its material. App. A states every pre-training recipe precisely and shows how each is computed in a single forward pass. App. B gives the data, architecture, training, inference and evaluation settings needed to reproduce the runs. App. C extends the results of Sec. 4 in the order of the research questions. App. D reviews related work in full, and App. E discusses limitations.

- App. A: *Training recipes.* Objectives, attention masks, exactness and cost, jump prediction, compute accounting.
- App. B: *Experimental setup.* Data, architecture, pre-training, inference, evaluation protocol.
- App. C: *Extended results*, one subsection per research question.
- App. D: *Related work in full.*
- App. E: *Limitations.* Scale, training-run variability, attribution, evaluation of dynamic content.

A TRAINING RECIPES

This section writes each pre-training recipe as an instance of Eq. (2) (App. A.1), shows how every recipe is computed in one forward pass under its own attention mask (App. A.2), defines jump prediction (App. A.3), and accounts for compute (App. A.4).

A.1 OBJECTIVES

A clip has $N{=}20$ latents with the first $\boldsymbol{z}_1$ given. A recipe R then fixes four things: *(i)* the distribution of the history profile $\boldsymbol{\tau}_{<n}$ (Def. 1), *(ii)* the prior over target flow times t_n, *(iii)* the set $\mathcal{S}_R$ of latents that carry a loss, and *(iv)* a weight w_n for each. Writing the expectation over n in Eq. (2) as a weighted sum over the supervised latents of a clip,

$$\mathcal{L}_R(\theta) = \mathbb{E}\Big[\sum_{n\in\mathcal{S}_R} w_n \big\| v_\theta(\boldsymbol{z}_n^{t_n}, t_n \mid h_n) - (\boldsymbol{z}_n - \boldsymbol{\epsilon}_n)\big\|_2^2\Big], \qquad h_n = (\boldsymbol{z}^{\boldsymbol{\tau}}_{<n}, \boldsymbol{\tau}_{<n}, \boldsymbol{a}_{\le n}), \tag{7}$$

where the expectation is over the clip, the noise, and flow times from the recipe's prior. Target flow times are drawn independently per latent from $\mathrm{LogitNormal}(0,1)$ for every recipe except DF, which draws them from $\mathcal{U}[0,1]$. This choice was to make our evaluation consistent with the DF literature as we note that using $\mathrm{LogitNormal}(0,1)$ for DF would place even less probability near clean flow times. All other methods use clean prefix $t=1$ for 90% of their batch so this would put the DF training distribution even further from the inference time distribution.

Single-sequence recipes. DF and CTX only require a single forward pass for a sampled video clip. Each latent then has one flow time and it plays both roles: latent i is a target at t_i and appears in the history of every later target at the same value, $\tau_i{=}t_i$ (or $\tau_i{=}1$ if it is presented clean).

- **DF.** In DF, every latent is noised, $\tau_i{=}t_i$ with $t_i \sim \mathcal{U}[0,1]$ i.i.d., and every latent is supervised with unit weight, $\mathcal{S}_{\text{DF}}{=}\{1,\dots,N\}$. As such, no target has a clean history.
- **CTX.** In CTX, we draw $c \sim \mathcal{U}\{1,\dots,N{-}1\}$ per clip, then present $\boldsymbol{z}_{\le c}$ clean ($\tau_i{=}1$ for $i \le c$) whilst the rest are at their own flow times, and we then supervise $\mathcal{S}_{\text{CTX}}{=}\{c{+}1,\dots,N\}$. The first target has weight $w_{c+1}{=}1$ and is the only aligned one. A later target n lies in the tail only when $c < n{-}1$, which happens with probability $\pi_n = (n{-}2)/(N{-}1)$, so the tail is weighted by $w_n = \lambda/\pi_n$ to make every position contribute equally in expectation. CTX uses $\lambda{=}1$, and CTX$-$TAIL uses $\lambda{=}0$, which leaves one aligned target per clip.

Doubled-sequence recipes. PF, CHUNK-B and JUMP-$h_{\max}$ use a clean history of latents whilst later targets are noisy. As such, they use a clean copy and a noisy copy of the clip (App. A.2). Every latent after the first is supervised with unit weight, $\mathcal{S}{=}\{2,\dots,N\}$ and the recipes differ only in the history each target sees.

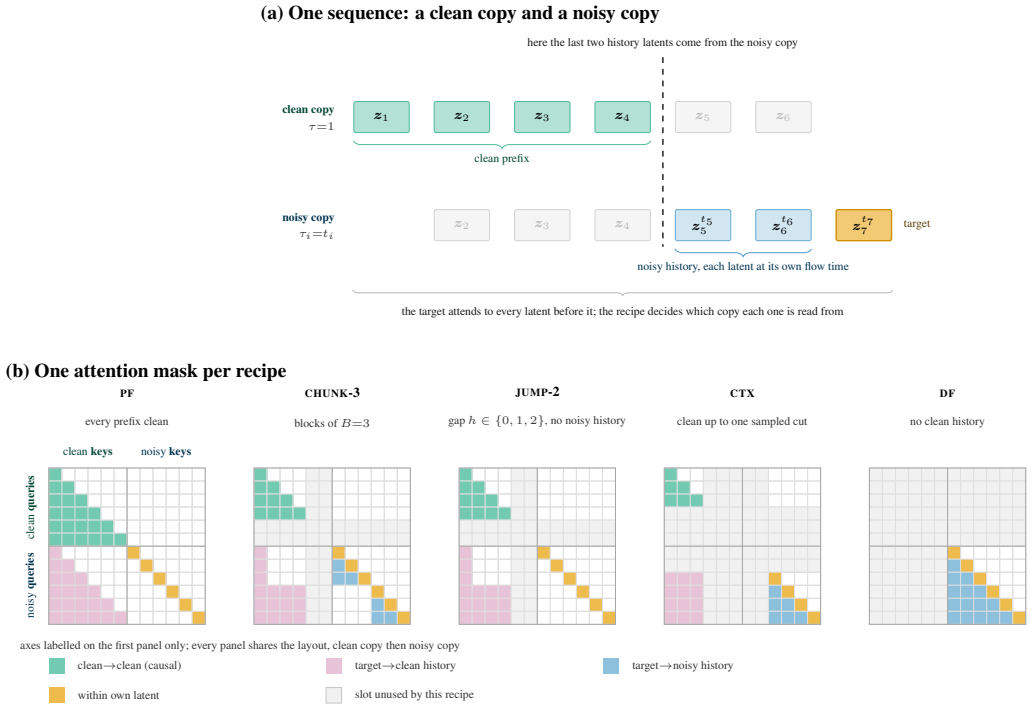


Figure 6: **Every recipe as one sequence and one mask. (a)** The doubled sequence: a clean copy of the clip and a noisy copy, one column per latent. A target attends to every latent before it and the recipe decides which copy each one is read from. In the example, the two most recent history latents come from the noisy copy. **(b)** The attention mask of each recipe, drawn for a short clip, with B=3 and $h_{\max}$=2 so that the patterns are visible. The clean→clean block is the same causal attention in every doubled-sequence recipe but how the blocks read by the noisy slots change. Note that PF has no noisy history, so its noisy→noisy block is the identity. JUMP-2 has the clean cutoffs of CHUNK-3 but no noisy history. CTX and DF are drawn in the same layout for comparison, although they are trained on a single copy: CTX reads clean latents up to one sampled cut and noisy latents after it whilst DF has no clean history at all.

- **PF.** In PF, every target sees its full clean prefix, $\boldsymbol{\tau}_{<n}=\mathbf{1}$. As such, on uncorrupted clips every term of Eq. (7) is aligned with inference and the objective is the teacher-forced objective AR inference implies,

$$\mathcal{L}_{\text{PF}}(\theta) = \sum_{n=2}^{N} \mathbb{E}_{t_n,\boldsymbol{\epsilon}_n} \left\| v_\theta\left(\boldsymbol{z}_n^{t_n}, t_n \mid \boldsymbol{z}_{<n}, \mathbf{1}, \boldsymbol{a}_{\le n}\right) - (\boldsymbol{z}_n - \boldsymbol{\epsilon}_n) \right\|_2^2. \tag{8}$$

- **CHUNK-B.** The targets are partitioned into consecutive blocks of B latents, with the block boundaries shifted by a phase that is redrawn every step so that every latent occupies every position within a block over training. A target n in the block starting at latent b sees τ_i=1 for $i < b$ and τ_i=t_i for $b \le i < n$: the clean prefix before its block and the earlier members of its block at their own flow times. The first member of each block is aligned such that B=1 recovers PF.
- **JUMP-$h_{\max}$.** A target n is assigned a gap $h_n \in \{0, \dots, h_{\max}\}$ and sees the clean prefix $\boldsymbol{z}_{<n-h_n}$ only: the h_n latents in between are removed from its keys rather than noised (App. A.3). As such, only targets with h_n=0 are aligned.

Under CHUNK-B and JUMP-$h_{\max}$ the network is told each target's depth, its position within its block or its gap, through a learned embedding.

History corruption. On a fraction p of clips, every latent that a recipe presents clean is presented at $\tau_i \sim \mathcal{U}[\tau_{\min}, 1]$ instead, drawn independently per latent, with the flow time declared to the

network. Every recipe except DF uses p=0.1, $\tau_{\min}$=0.7 unless stated otherwise. It is worth noting that corrupted clips have no aligned targets and that DF has no clean latents to corrupt.

A.2 ATTENTION MASKS

Every recipe is trained with a single forward pass, however, the sequence length, and thus FLOPs per forward pass varies. DF and CTX use the frame-causal mask on one copy of the clip, with the flow times above. The doubled-sequence recipes place a clean copy $\boldsymbol{z}_{1:N-1}$, at τ=1 and a noisy copy $\boldsymbol{z}_2^{t_2}, \ldots, \boldsymbol{z}_N^{t_N}$ in one sequence of $2(N-1)$=38 latents, and a recipe is then nothing but a mask over it (Fig. 6). Let us write c_j for the clean slot of latent j, u_n for the noisy slot of latent n, and $\ell(\cdot)$ for a slot's latent index. Under the PF mask,

- a clean slot attends causally to clean slots, $c_j \to c_i$ for $i \leq j$;
- a noisy slot attends to the clean slots strictly before it, $u_n \to c_i$ for $i < n$;
- every slot is attended within its own latent space, and there is no other attention between noisy slots.

The last condition is what keeps the $N-1$ targets independent: if noisy slots attended to each other, a target would see other targets' noisy latents, and its history would no longer be clean. CHUNK-B additionally lets u_n attend to the earlier noisy slots of its own block and moves its clean cutoff to the start of the block. Similarly, JUMP-$h_{\max}$ moves the clean cutoff to $n-h_n$ but does not maintain a noisy history. All masks are expressed as FlexAttention block masks, so blocks of the 38×38 latent mask that are fully masked are skipped and the dense token-level mask is never materialized.

A.3 JUMP PREDICTION

CHUNK-B fills the gap between a target and its clean prefix with the inbetween latents, each at its own flow time. In contrast, JUMP-$h_{\max}$ removes them from the target's keys. That is, target n predicts from the clean prefix $\boldsymbol{z}_{<n-h_n}$, with latents $n-h_n, \ldots, n-1$ absent rather than noisy. The gap is assigned rather than sampled so that h_n cycles through $0, \ldots, h_{\max}$ through consecutive targets, and the cycle is rotated every step so that every target sees every gap during training. As a result, one target in $h_{\max}+1$ is aligned, the same count as CHUNK-B at B=$h_{\max}+1$, and the two recipes share their clean cutoffs (Fig. 6). They differ only in what fills the gap, nothing under jump and noisy latents under chunk, which makes jump the control that separates the two ways in which predicting several latents ahead could help. That is, either *(i)* seeing a corrupted recent history, or *(ii)* training the network to bridge a gap from a clean one. It is worth noting that jump is not a history profile in the sense of Def. 1, since the latents in the gap are absent rather than presented at some flow time, and no sampler produces its configuration at inference. We run one arm, JUMP-3, at the quarter budget and report its results in App. C.1.

A.4 COMPUTE ACCOUNTING

All comparisons are at equal training FLOPs, counted with the measured mask density, which it is worth noting is hardware independent (Table 3). At the full budget, the single-sequence recipes train for 350k steps and the doubled-sequence recipes for 185k, both reaching $\approx$680 EFLOP. For the quarter budget experiments, every arm trains for 45k steps to 166 EFLOP. We note that equal FLOPs is not equal GPU-hours as the doubled sequence realises a lower model FLOP utilisation (0.187 against 0.199), so at equal FLOPs it takes 8.7% more GPU-hours. The gap comes from the block-sparse attention kernel on the longer sequence, not from the objective. Table 2 summarises supervision per clip and per unit of compute. Table 3 reports the measured training costs.

Table 2: **What each recipe supervises**, per clip of $N{=}20$ latents with the first given. A counts aligned targets, D all targets, and ρ is the measured cost relative to one standard forward pass. For CTX, D is $\mathbb{E}[N-c]$. History corruption makes a fraction p of PF and CHUNK-B targets unaligned.

recipe	A	D	ρ	A/ρ	D/ρ
DF	0	20	1.0	0	20
PF	19	19	1.9	10	10
CTX	1	10	1.0	1	10
CHUNK-B	$19/B$	19	1.9	$10/B$	10

Table 3: **Cost accounting**, measured at resolution $128{\times}224$, with $N{=}20$ latents of 112 tokens, at an effective batch of 240 clips. The FLOP count per clip is identical to six significant figures across three batch configurations, so it is linear in batch and independent of the number of GPUs. CHUNK-B and JUMP-$h_{\max}$ share the doubled-sequence column.

	single sequence DF, CTX, CTX−TAIL	doubled sequence PF, CHUNK-B, JUMP-$h_{\max}$	ratio ρ
Sequence slots	$N = 20$	$2(N{-}1) = 38$	
GPUs × (micro × accum)	$3 \times (40 \times 2)$	$4 \times (30 \times 2)$	
Effective batch	240	240	1.00
Mask density	0.525	0.263	0.50
FLOPs / clip (TF)	8.08996	15.3736	**1.9003**
FLOPs / step, total (TF)	1941.6	3689.7	1.900
Realised MFU	0.199	0.187	0.94
Wall-clock / step (s)	2.54	3.84	
Peak memory / GPU (GiB)	82.3	76.0	
Steps at the full budget	350,000	185,000	
Total compute (EFLOP)	679.6	682.6	1.004
GPU-hours	794	863	1.087

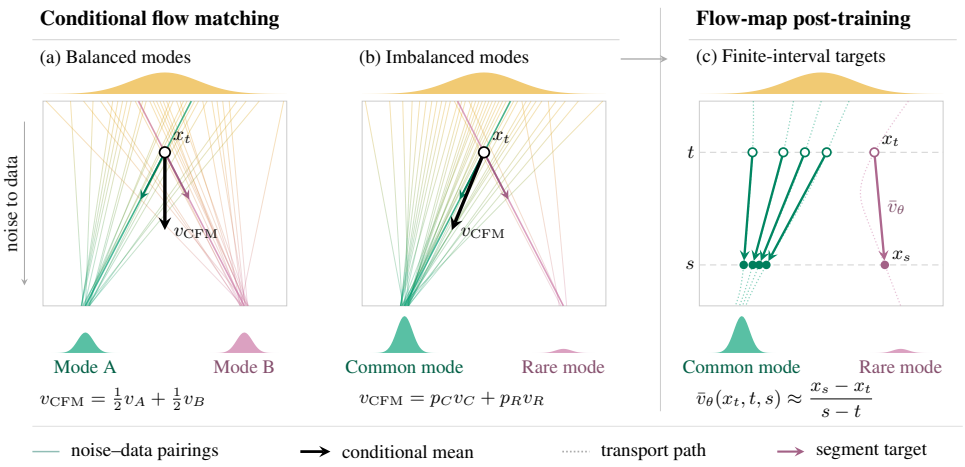


Figure 7: **Conditional averaging and finite-interval targets in an analytic toy. (a,b)** With random noise–data pairings, opposing component velocities at the marked state average to zero for balanced modes; with conditional weights $(0.9, 0.1)$, the mean points toward the common mode. Colored arrows are component-conditional targets and black arrows are their conditional means. **(c)** Open and filled markers show inputs and exact finite-interval targets from $t = 0.25$ to $s = 0.80$ on transport paths of the imbalanced toy field; dotted curves give path context. Standard Gaussian noise is paired with Gaussian data modes at ± 1.5 of standard deviation 0.1. At $(x, t) = (0, 0.25)$, the mean velocities in (a) and (b) are 0 and -1.598. These are analytic training targets, not outputs from a trained model or evidence that exact integration of the conditional-mean field loses a mode.

B EXPERIMENTAL SETUP

In this section, we provide additional details of the experimental setup, *(i)* the data and tokeniser (App. B.1), *(ii)* the network shared by every recipe (App. B.2), *(iii)* the pre-training settings for every experiment (App. B.3), *(iv)* how rollouts are generated (App. B.4), and *(v)* how they are scored (App. B.5).

B.1 DATA AND TOKENISER

Data. We use the Minecraft contractor data released with VPT (Baker et al., 2022), restricted to the *build-house*, *find-cave*, *pen-animals* and *waterfall* tasks. Videos are at 20 frames per second and are resized to 128×224. We removed the initial frame from each clip as the initial frame in the majority of clips is the load screen. Each frame transition carries a 25-dimensional action consisting of 23 binary keys and the camera's pitch and yaw, which we compressed with a μ-law following (Guo et al., 2025) (App. B.2). Training samples clips of $N_{\text{px}}=77$ frames, *i.e.* $N=20$ latents, at a stride of one latent. The held-out evaluation uses the 766 test clips long enough for 1000 predicted frames: 232 from build-house, 112 from find-cave, 224 from pen-animals and 198 from waterfall. We will release the encoded latents and the splits.

Tokeniser. Clips are encoded once with the WAN 2.2 VAE (Team Wan et al., 2025), which compresses $4\times$ temporally and $16\times$ spatially into 48 channels. We pre-encode entire clips and store the posterior means in `fp16` after validating that fp16 storage causes no measurable degradation relative to fp32. The first latent covers one frame whilst every later latent compresses four, so a clip of 77 frames is 20 latents on an 8×14 grid. The encoder is frozen throughout, so one set of stored latents serves every run. We finetune only the decoder at 128×224, which improves PSNR and SSIM over the public decoder at this resolution (Table 4).

Table 4: **Encode–decode reconstruction** on held-out Minecraft episodes at 128×224. The encoder is shared; only the decoder differs.

VAE (128×224)	PSNR ↑	SSIM ↑	LPIPS ↓
public WAN	29.25	0.8542	0.0325
finetuned (ours)	32.44	0.8916	0.0360

B.2 ARCHITECTURE

In order to provide a controlled study, every recipe uses the same network (Fig. 8). Only the attention mask, the flow-time prior and the loss change between recipes. See Table 5 for an overview.

Tokens. A 1×1 patch embedding turns each latent into $S=112$ tokens. A learned embedding $\boldsymbol{e}^{\text{first}}$ marks the episode's first latent, which the VAE does not compress in time and which therefore has different statistics. There is no learned positional table as we encode positional information via 3D RoPE. A quarter of each head's rotary pairs encode time (base 100) and the rest height and width (base 1000), each band with its own frequency ladder. Time is indexed by absolute latent position, so cached keys stay valid when the window slides.

Blocks. The DiT (Peebles & Xie, 2023) has 24 pre-norm blocks of width 1024. Self-attention has 8 query heads and 4 key-value heads (grouped-query), with QK-norm and a frame-causal mask, which each recipe replaces with its own (see App. A.2). Each block runs four sublayers in order: self-attention, cross-attention to the actions, cross-attention to $\boldsymbol{z}_1$, and a SwiGLU FFN with hidden width 2816. Every sublayer has a zero-initialised, elementwise gate on its residual branch, so the network starts as the identity. Self-attention and the FFN use AdaLN, a LayerNorm followed by a per-latent scale and shift, and per-block linear maps produce these and the two gates from a conditioning vector $\boldsymbol{c}_n$. The two cross-attention sublayers use a plain LayerNorm and have their own zero-initialised gates on $\boldsymbol{c}_n$. The vector $\boldsymbol{c}_n$ is built per latent from the flow time (sinusoidal features and an MLP), a clean or noisy role embedding $\boldsymbol{e}_n^{\text{role}}$, and a residual refinement MLP. Under CHUNK-B and JUMP-$h_{\max}$ a zero-initialised depth embedding is added as well (see App. A.1). The

flow-time embedding and the application of every scale, shift and gate run in `fp32`, and the maps that produce them under `bf16` autocast.

Actions. Actions are conditioned on via cross-attention after embedding them and temporally compressing to match the latent sequence length. The five mutually exclusive key pairs (forward and back, left and right, sprint and sneak, attack and use, drop and pick) are three-way categoricals, the hotbar is a ten-way categorical, and the remaining flags and the camera each pass through a small MLP. We concatenate each of them and pass them through a two-layer MLP to width 1024. A strided 1D convolution with kernel and stride 4, the VAE's temporal stride, turns the 76 per-frame embeddings into 19 tokens (one per latent after the first), and a learned anchor token is prepended for $\boldsymbol{z}_1$, giving $e^{\text{act}}_{1:N}$. The tokens of latent n attend causally to $e^{\text{act}}_{\leq n}$, with temporal RoPE at latent index. As such, both slots of latent n in the doubled sequence see the same actions.

First-latent anchor. Each block also cross-attends to the 112 tokens of $\boldsymbol{z}_1$, placed at time 0 with full spatial RoPE. Every latent therefore sees the episode's first latent after it has left the self-attention window (App. B.4). Its keys and values are fixed for the whole rollout and are computed at the start.

Size. The network has 653.5M trainable parameters, 648.0M in the DiT and 5.5M in the action encoder, excluding the VAE. Each block has 3.1M parameters in self-attention, 8.4M in the two cross-attention sublayers and 8.7M in the FFN, and most of the rest are the per-block modulation maps (153M in total). The depth embedding of CHUNK-B and JUMP-$h_{\max}$ adds 4,096.

Engineering. Attention uses FlexAttention with a latent-level block mask, so fully masked tiles are skipped and the dense token-level mask is never materialised. Tiles are 128 tokens, since the kernel requires a power of two and $S{=}112$ is not one, so tiles that straddle a latent boundary are evaluated per element. The FFN linears run in `float8` (torchao) and everything else under `bf16` autocast. When training on H200s, blocks use activation checkpointing, whilst on B300s we found that turning off gradient checkpointing and increasing gradient accumulation was significantly faster. For all experiments, the forward pass was compiled with `torch.compile`. It is worth noting that these choices change throughput, and so the wall-clock figures of Table 3, but not the FLOP count.

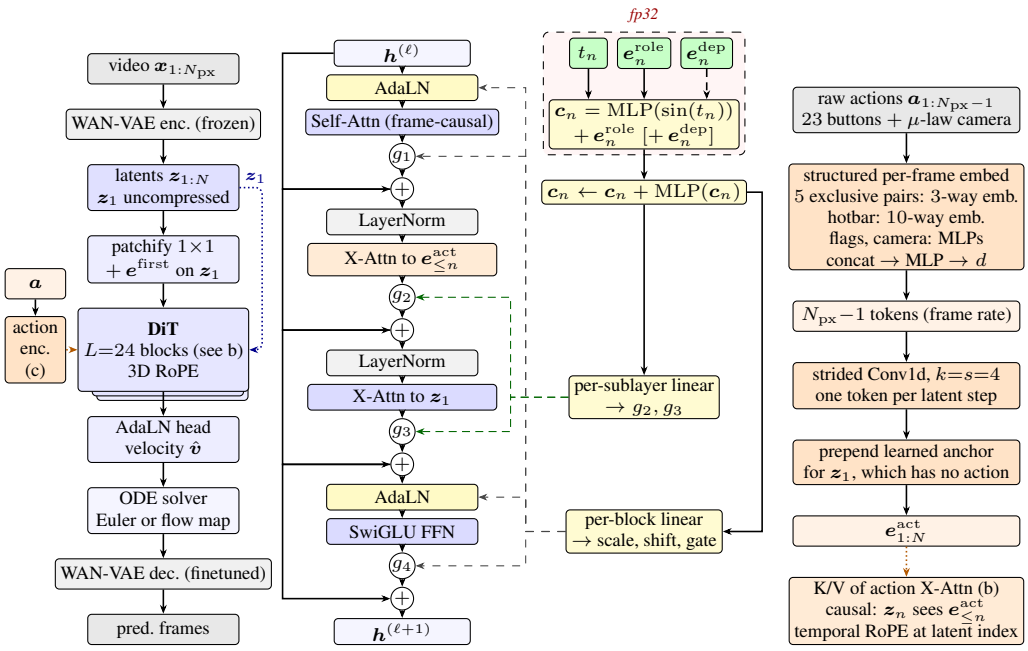
B.3 PRE-TRAINING

Table 5 lists every setting shared by the pre-training arms whilst Table 6 details the experiment specific settings.

Matching compute. For each experiment, we set the budget in training FLOPs specific to the implementation, not in steps or wall-clock (App. A.4). The full budget is 350k steps for the single-sequence recipes (679.6 EFLOP) and 185k for the doubled-sequence ones (682.6 EFLOP). The quarter budget experiments are trained for 45k doubled-sequence steps (166.0 EFLOP). Note that CHUNK-B costs exactly what PF costs, since every target attends to the same number of latents for any B. JUMP-$h_{\max}$ costs slightly less than the counter reports as FLOP matching would give it 45,379 steps. Nevertheless, we keep it at 45k so that its evaluation grid aligns with the other arms but note that this counts against it.

Matching the schedule. Every experiment uses the same peak learning rate, a linear warmup, and a linear decay to zero that ends at the experiment's final step. We also match learning rate warmup in compute, and so as a fraction of the run, *i.e.*, 5,000 single-sequence steps (1.43%), or 2,600 doubled-sequence steps (1.41%), or 650 steps at the quarter budget (1.44%). Within a budget, equal FLOPs therefore approximately means the same point in the schedule. Note that across FLOP budgets the schedules do not line up, so no comparison in the paper crosses budgets. That is to say, the quarter budget runs are completely separate runs from the full budget runs.

Checkpoints. Throughout the paper we use the final checkpoint, with no checkpoint selection and no weight averaging.



(a) Data flow. **(b)** DiT block (layer ℓ), frame n. **(c)** Action encoder.

Figure 8: **Architecture. (a)** Latents from the frozen WAN-VAE encoder are patchified and denoised by a 24-block frame-causal DiT. The head's output is integrated by the ODE solver, Euler for pre-trained models and the flow map for post-trained ones, and the finetuned decoder maps the output to frames. Actions (orange) and the first latent z_1 (blue) reach the DiT only as cross-attention keys and values. **(b)** One block consisting of self-attention and the FFN which are AdaLN-modulated by c_n, which is built from the flow time, the role, and, for CHUNK-B and JUMP-$h_{\max}$ only, the prediction depth. The two cross-attention sublayers use a plain LayerNorm and have their own gates. Note that all four gates are zero-initialised. **(c)** Shows the structured per-frame action embedding, compressed to one token per latent by a strided convolution, with a learned anchor for z_1. Latent frame n then cross-attends to the current action as well as all past actions $e^{act}_{\le n}$.

B.4 INFERENCE

AR Inference. An autoregressive (AR) rollout starts from the ground-truth first latent z_1 of a test clip and generates one latent, *i.e.* four frames, at a time, conditioned on the clip's recorded actions. The self-attention window holds W=20 latents to match the training sequence length. It grows for the first 19 generated latents and then slides, evicting the oldest latent each step. Predicted frames 1-76 are therefore generated with z_1 in the self-attention window, and from frame 77 onwards every latent in the window is the model's own output. Note that z_1 does stay visible through the first-latent cross-attention (see App. B.2). This is the shaded region in every horizon plot. Throughout the paper, we roll out for 1000 frames, which corresponds to 250 generated latents.

Solvers. For the pre-trained checkpoints we integrate the velocity field from t=0 to t=1 with 10 Euler steps per latent. For post-trained models we use the flow map of Eq. (4) with 4 steps per latent, chaining X_θ over a partition of $[0, 1]$. As controls, App. C.2 also reports pre-trained models with 4 Euler steps and post-trained models with 10 flow-map steps.

KV cache. During inference, we store committed latents in a KV cache, so each generated latent costs one pass over the new latent's tokens per solver step. When a latent is evicted, the cache can either *retain* the keys and values of the remaining latents, which were computed while the evicted latent was still visible, or *recompute* them from the latents now in the window, which costs about 1.8× as much per generated latent. Throughout the paper we use the retaining strategy as default and

Table 5: **Settings shared by every pre-training arm.**

	setting	value
data	source / resolution	VPT Minecraft (Baker et al., 2022), 128×224
	tokeniser	WAN 2.2 VAE, encoder frozen, decoder finetuned (App. B.1)
	latent grid / channels	8×14 / 48; latents are posterior means
	clip	N_{px}=77 frames = N=20 latents, first latent given
	window stride	1 latent
	actions	76 per clip, 25-d (23 binary, 2 μ-law camera)
model	width / depth / heads	1024 / 24 / 8 query, 4 key–value (GQA)
	patch / tokens per latent	1×1 / S=112
	attention	frame-causal, QK-norm (LayerNorm), no QKV bias
	FFN	SwiGLU, hidden 2816 ($\frac{2}{3} \cdot 4d$ rounded up to 128)
	norm / conditioning	LayerNorm (ϵ=10^{-5}); AdaLN-zero, per-block linear, fp32
	RoPE	3D; time fraction 0.25; bases 100 (time), 1000 (space); per-band
	action path	cross-attention only; causal; temporal RoPE
	first-latent anchor	cross-attention to $\boldsymbol{z}_1$, spatial + temporal RoPE
	parameters	653.5M (+4,096 with the depth embedding)
flow matching	path	rectified: $\boldsymbol{z}^t = t\,\boldsymbol{z} + (1-t)\,\boldsymbol{\epsilon}$, t=1 clean
	target / loss	velocity $\boldsymbol{z} - \boldsymbol{\epsilon}$, MSE, mean over weighted target elements
	flow time	independent per latent; $\text{LogitNormal}(0, 1)$ except DF (uniform)
	time embedding	sinusoidal (256 features, $t \times 1000$) + MLP
optimisation	optimiser	AdamW, β=$(0.9, 0.95)$, fused
	weight decay	0.01 on every tensor with $\geq$ 2 dims (incl. embedding tables); 0 on the rest
	peak learning rate	4×10^{-4}
	schedule	linear warmup, then linear decay to 0 at the arm's own final step
	gradient clipping	global norm 1.0
	effective batch	240 clips (Table 6 for the split)
	precision	bf16 autocast; float8 FFN linears (torchao); fp32 modulation
	EMA	none
	seed	42 (one seed per arm)
sampling	solver	pre-trained: Euler, 10 steps; post-trained: flow map, 4 steps; no guidance
	rollout	one latent per step; window of 20 latents; KV cache
	KV cache on eviction	retain, unless stated (App. B.4)
	held-out evaluation	test split, 766 clips × 1000 predicted frames, batch 128, final checkpoint

Table 6: **Settings and budgets for each pre-training run.** $p(t)$ is the target flow-time prior, uniform ($\mathcal{U}$) or $\text{LogitNormal}(0, 1)$ (LN). Tail is the weight on targets past the first and whether inverse-inclusion reweighting (IIP) is applied (App. A.1). $(p, \tau_{\min})$ is the history corruption: with probability p a clip's clean history is presented at $\tau \sim \mathcal{U}[\tau_{\min}, 1]$, independently per latent; DF has no clean history to corrupt. Depth is the block length or maximum gap, which also turns on the depth embedding. Every arm uses an effective batch of 240 clips. *Resumed on a second cluster with a different split of the same batch. †The FLOP counter ignores the JUMP-3 mask and overstates its compute by 0.83%.

recipe	$p(t)$	tail	$(p, \tau_{\min})$	depth	GPUs×(micro×acc.)	steps / warmup	EFLOP
Full budget: linear decay to 0 at 350k (single-sequence) or 185k (doubled-sequence) steps							
DF	$\mathcal{U}$	all, 1	—	—	3×(40×2)	350,000 / 5,000	679.6
CTX	LN	1, IIP	(0.1, 0.7)	—	3×(40×2)	350,000 / 5,000	679.6
CTX−TAIL	LN	0	(0.1, 0.7)	—	3×(40×2)	350,000 / 5,000	679.6
PF	LN	—	(0.1, 0.7)	—	4×(30×2) / 3×(80×1)*	185,000 / 2,600	682.6
CHUNK-4	LN	—	(0.1, 0.7)	B=4	3×(80×1)	185,000 / 2,600	682.6
Quarter budget: all doubled-sequence, linear decay to 0 at 45k steps							
PF	LN	—	(0, —)	—	4×(30×2)	45,000 / 650	166.0
PF	LN	—	(0.1, 0.7)	—	4×(30×2)	45,000 / 650	166.0
PF	LN	—	(0.1, 0.5)	—	4×(30×2)	45,000 / 650	166.0
PF	LN	—	(0.1, 0.0)	—	4×(30×2)	45,000 / 650	166.0
CHUNK-4	LN	—	(0.1, 0.7)	B=4	4×(30×2) / 3×(80×1)*	45,000 / 650	166.0
JUMP-3	LN	—	(0.1, 0.7)	$h_{\max}$=3	4×(30×2)	45,000 / 650	166.0†

every number uses it unless stated. Note that both methods agree exactly until the first eviction and their difference is the policy alone (see App. C.4).

AR and TF. In AR inference the window holds the model's own committed latents, whilst in TF inference it holds ground-truth latents at every step. That is to say, for TF, the model generates the

next latent from the true history and that latent is scored but not committed. Both run for the same 1000 frames and TF FVD in a window measures the one-step conditional at that depth of the clip.

History noise. Under history noise at $t_{\text{hist}} < 1$, each committed latent is re-noised once, when it enters the window, to $\boldsymbol{z}_i^{t_{\text{hist}}} = t_{\text{hist}}\hat{\boldsymbol{z}}_i + (1 - t_{\text{hist}})\boldsymbol{\epsilon}_i$, and presented with its flow time declared as t_{hist}. The content and the declared flow time therefore move together, as in training-time corruption (see App. A.1). It is worth noting that across experiments, the noise is drawn from a fixed seed, so it is identical across the models being compared.

B.5 Evaluation protocol

FVD. Throughout the paper, every number is FVD (Unterthiner et al., 2018) on decoded frames, where features are the 400-dimensional logits of the I3D network trained on Kinetics-400. Clips are 16 frames, resized to 224×224 and scaled to $[-1, 1]$, and start on latent boundaries (frames $1, 5, 9, \dots$) at a stride of one latent, 247 per video. A clip counts toward a window only if it lies wholly inside it, which gives 6, 7, 16, 59 and 147 clips per video in the windows 1-36, 37-76, 77-152, 153-400 and 401-1000. Horizon plots split the last two windows further, at frames 365, 577 and 789.

Directional KL. Fig. 5 complements FVD with two directional KLs between the distributions of generated and real rollouts. Forward KL, $D_{\text{KL}}(p_{\text{data}} \| p_{\text{model}})$, penalises missing data modes (mode collapse); reverse KL, $D_{\text{KL}}(p_{\text{model}} \| p_{\text{data}})$, penalises spurious model modes. We fit Gaussian distributions to I3D features of the real and generated clips and compute both closed-form Gaussian KL directions. They are computed over frames 1–1000 of 128 validation videos with history noise at t_{hist}=0.98.

The test panel. Every window is scored over the same 766 test clips, the ones long enough for 1000 predicted frames, because FVD is a population statistic and windows scored over different sets of videos are not comparable. Every model is rolled out on these clips with the same seed.

Confidence intervals. The bands in Fig. 10(b) and the ties marked bold in tables are 95% percentile intervals of differences over 500 bootstrap resamples of videos, with the same resample applied to every model, so every difference is paired. They quantify variability across evaluation videos, not across training runs. Resampling biases FVD upwards as the number of distinct videos falls, but the bias is shared by both sides of a difference and cancels there. In tables, an entry is in bold if it is the best in its column or if its paired interval against the best contains zero. For this reason we attach intervals to differences only. Plots of absolute FVD show point estimates, and every comparison the text draws from them is quoted with its paired interval.

C Extended results

C.1 Pre-training

Table 7 reports every pre-trained recipe in every window, in AR and TF, at both budgets. The two budgets are separate runs with their own learning rate schedules, so rows are compared only within a block. Two patterns hold at both budgets: recipes that are close over the first 76 frames but then separate past them, and TF FVD varies far less across recipes than AR FVD does.

C.2 Post-training

Sampling budget. Table 8 evaluates every model at both 4 and 10 function evaluations per latent. At either budget, every post-trained model is better than its pre-trained counterpart past frame 76. Pre-trained models degrade sharply with fewer Euler steps, while post-trained models change far less between the two budgets. The improvement therefore persists at matched NFE, but this comparison does not isolate the solver: it changes with the stage, from Euler integration to the learned flow map.

Qualitative rollouts. Figure 9 illustrates how dynamic content and its persistence change through post-training, alongside artifacts from history noise at inference.

Table 7: **Pre-trained recipes at both budgets.** AR and TF FVD by window of predicted frames, 766 held-out clips, 10 Euler steps, retained KV cache. Unlabelled PF, CHUNK-4 and JUMP-3 use history noise p=0.1, $\tau_{\min}$=0.7. Bold is the best entry of a column within a block, with every entry whose paired 95% interval against it contains zero.

		AR FVD ↓					TF FVD ↓				
		GT IN WINDOW					GT IN WINDOW				
RECIPE	ρ	1–36	37–76	77–152	153–400	401–1000	1–36	37–76	77–152	153–400	401–1000
full budget, 680 *EFLOP*											
DF	1.0	35	56	99	181	329	22.6	8.4	5.6	4.1	4.3
CTX−TAIL	1.0	47	91	131	209	271	23.0	9.1	6.5	5.1	5.6
CTX	1.0	62	71	75	114	207	22.6	8.5	8.2	6.9	7.9
PF	1.9	34	60	85	129	177	**18.6**	**7.1**	**4.5**	**3.2**	**3.5**
CHUNK-4	1.9	**28**	**47**	**65**	**105**	**132**	19.3	**7.4**	**4.7**	3.4	3.6
quarter budget, 166 *EFLOP*											
DF	1.0	**33**	**79**	193	344	596	31.1	11.4	7.9	6.4	7.0
CTX	1.0	75	94	131	199	254	26.2	10.9	10.3	9.2	10.3
PF, p=0	1.9	52	101	156	252	327	23.0	**9.1**	6.7	5.3	6.1
PF	1.9	47	92	137	224	285	22.3	**9.0**	**6.5**	**5.3**	**6.0**
PF, $\tau_{\min}$=0.5	1.9	45	90	136	225	299	**21.3**	**9.1**	**6.4**	**5.2**	**6.0**
PF, $\tau_{\min}$=0	1.9	44	84	130	217	330	**20.8**	**9.0**	**6.3**	5.4	6.1
CHUNK-4	1.9	44	81	**114**	**176**	**216**	23.7	**9.1**	7.0	5.8	6.5
JUMP-3	1.9	42	**76**	**112**	**179**	230	23.0	10.0	7.0	6.2	6.9

Table 8: **Pre-trained and post-trained models at** 4 **and** 10 **evaluations per latent.** AR FVD by window and TF FVD past frame 400, 766 held-out clips, clean history, retained KV cache. AR−TF is the drift gap past frame 400. Bold is the best entry of a column within a recipe, with every entry whose paired 95% interval against it contains zero.

				AR FVD ↓					FRAMES 401–1000	
				GT IN WINDOW						
RECIPE	MODEL	SOLVER	NFE	1–36	37–76	77–152	153–400	401–1000	TF FVD ↓	AR−TF
DF	pre-trained	Euler	4	73	129	217	307	693	7.5	686
	pre-trained	Euler	10	35	56	99	181	329	**4.3**	325
	post-trained	flow map	4	**30**	**30**	**36**	**61**	**86**	7.5	78
	post-trained	flow map	10	**29**	**30**	**38**	**65**	108	6.9	101
PF	pre-trained	Euler	4	74	150	225	309	432	5.8	426
	pre-trained	Euler	10	**34**	60	85	129	177	**3.5**	174
	post-trained	flow map	4	**35**	44	**55**	**104**	**151**	5.2	145
	post-trained	flow map	10	**35**	**40**	**55**	**100**	**147**	4.8	143
CHUNK-4	pre-trained	Euler	4	61	117	171	243	347	6.1	340
	pre-trained	Euler	10	**28**	**47**	65	105	132	**3.6**	129
	post-trained	flow map	4	40	**50**	**56**	**72**	**91**	3.8	87
	post-trained	flow map	10	39	**50**	**59**	**71**	**86**	4.1	82

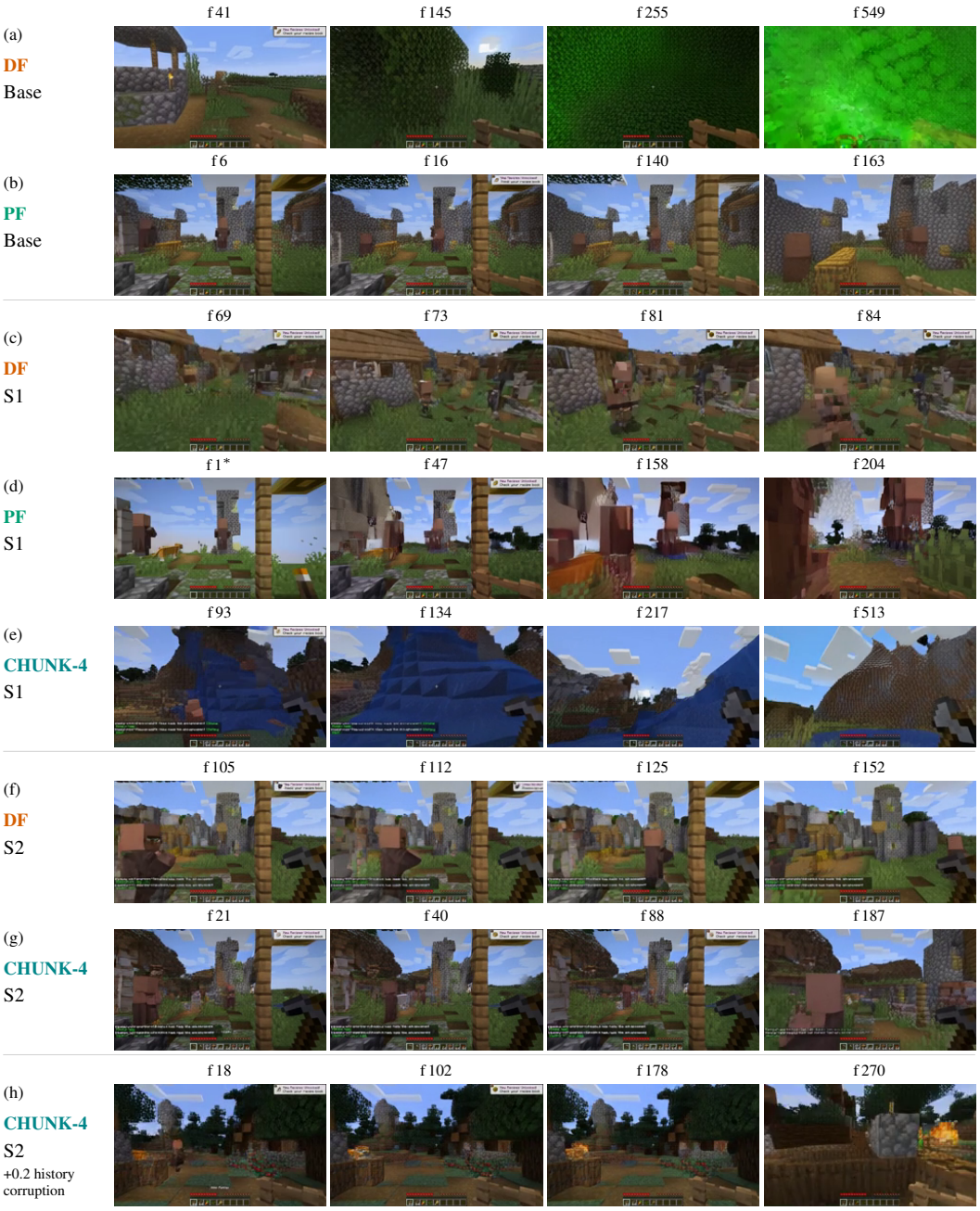


Figure 9: **Qualitative AR rollouts through post-training (NFE 4).** Each row follows one rollout, with captured frame indices above the images. Note that rows are illustrative excerpts, not matched trajectories. Base DF degrades **(a)**, while Base PF villagers change little in pose **(b)**. S1 generates NPCs and waterfalls **(c–e)**, but NPCs can deform **(d)**. After S2, villagers persist and move around **(f,g)**. With 0.2 history corruption **(h)**, objects and landscape morph: a pumpkin becomes a cat, then a torch. Images show the generated panel, except f 1^*, the observed initialisation frame.

C.3 HISTORY NOISE AT INFERENCE

Table 9 lists the full sweep behind Fig. 10, and Fig. 11 plots it in every window. Two regularities stand out. The best t_{hist} generally falls with horizon: noise that helps past frame 400 costs quality over the first 36 frames. And post-training changes how a recipe tolerates noise: post-trained DF is nearly flat from t_{hist}=0.99 to 0.7, while post-trained PF degrades sharply below 0.95, although pre-trained PF did not.

Table 9: **History noise at inference, pre-trained and post-trained.** AR FVD by window when every committed latent is presented at flow time t_{hist}, 766 held-out clips, retained KV cache. Pre-trained models use 10 Euler steps, post-trained models 4 flow-map steps. Bold is the best entry of a column within a recipe, with every entry whose paired 95% interval against it contains zero. Settings are selected on this evaluation.

	PRE-TRAINED, AR FVD ↓					POST-TRAINED, AR FVD ↓				
	GT IN WINDOW					GT IN WINDOW				
t_{hist}	1–36	37–76	77–152	153–400	401–1000	1–36	37–76	77–152	153–400	401–1000
DF										
1 (clean)	**35**	56	99	181	329	**30**	**30**	36	61	86
0.99	**34**	61	133	324	789	**31**	**29**	**34**	57	76
0.98	**34**	61	127	340	875	31	**29**	**33**	58	**75**
0.95	36	**53**	73	155	347	33	**29**	37	57	**74**
0.90	39	**52**	62	89	125	37	**30**	**36**	55	**75**
0.85	44	**52**	58	82	**111**	45	31	**34**	**52**	**74**
0.80	51	**51**	**55**	**77**	**112**	55	35	**34**	**48**	**68**
0.70	70	56	**53**	**75**	**115**	79	55	48	55	76
0.60	90	73	65	91	151	103	84	74	81	100
0.50	107	93	84	118	223	123	110	100	106	133
PF										
1 (clean)	**34**	**60**	85	129	177	**35**	44	55	104	151
0.98	**34**	61	87	138	178	**34**	42	**49**	**87**	**123**
0.95	**35**	61	83	127	165	**35**	**39**	**53**	98	144
0.90	37	60	79	118	**154**	45	55	88	184	272
0.85	41	**58**	**77**	**109**	**145**	59	68	117	237	331
0.80	50	**56**	**74**	**112**	**146**	75	84	142	252	348
0.70	76	69	80	**115**	**147**	132	186	261	388	484
0.60	109	105	110	137	190	241	399	490	608	720
CHUNK-4										
1 (clean)	**28**	**47**	65	105	**132**	**40**	**50**	**56**	**72**	91
0.98	**29**	**47**	68	104	**129**	**40**	**52**	**58**	72	**86**
0.95	**29**	**46**	64	98	**128**	**41**	**53**	**57**	**68**	**80**
0.90	31	**45**	59	94	**126**	44	**52**	**59**	**69**	**83**
0.85	35	**45**	**55**	**87**	**123**	49	**51**	**57**	**67**	**87**
0.80	42	**45**	**53**	**84**	**122**	59	55	**58**	**69**	98
0.70	62	57	60	**85**	**124**	79	63	63	**68**	113
0.60	82	76	82	97	**131**	108	94	93	86	118

C.4 KV CACHE

Table 10 reports $\Delta_{\text{cache}} = \text{FVD}_{\text{recompute}} - \text{FVD}_{\text{retain}}$ for every model, so positive values favour retaining the cache. The two policies agree until the first eviction, so the first two windows are zero up to evaluation noise. Before post-training, only CTX at 10 steps and CHUNK-4 at 4 steps separate from zero past frame 400, both in favour of recomputation. After post-training, all three recipes favour the retained cache. In TF the window holds ground truth, so the TF column isolates the effect of stale keys and values without compounding; it stays below one FVD for every model except pre-trained CTX.

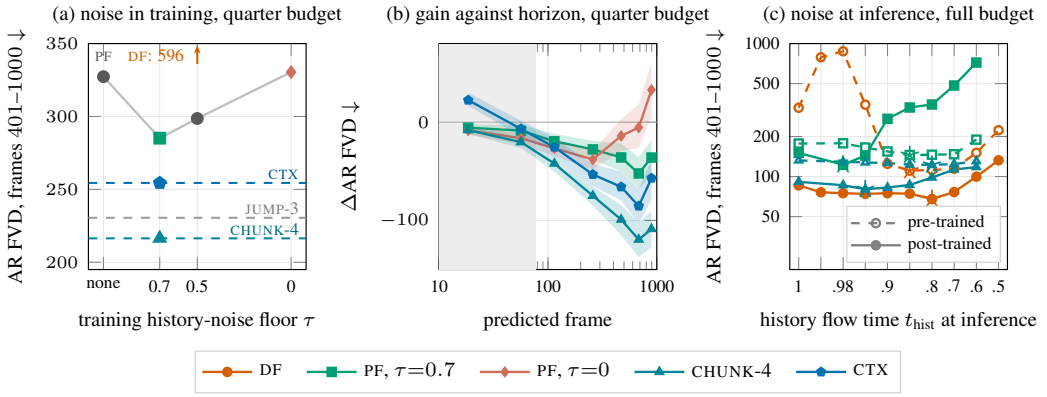


Figure 10: **History noise helps, but does not replace post-training. (a, b)** At equal quarter-budget compute, PF receives history noise on 10% of clips, with flow time sampled from $[\tau, 1]$ (τ=1: no noise). Panel (a) compares FVD over frames 401–1000 with other recipes. Panel (b) shows changes relative to PF without history noise, with paired 95% intervals. Lower is better. Heavy noise loses its benefit after frame 400, while chunked prediction retains its advantage. **(c)** After post-training at full budget, inference history is re-noised to match t_{hist}. Dashed curves use 10 Euler steps before post-training, solid curves 4 flow-map steps after. Results use 766 held-out clips.

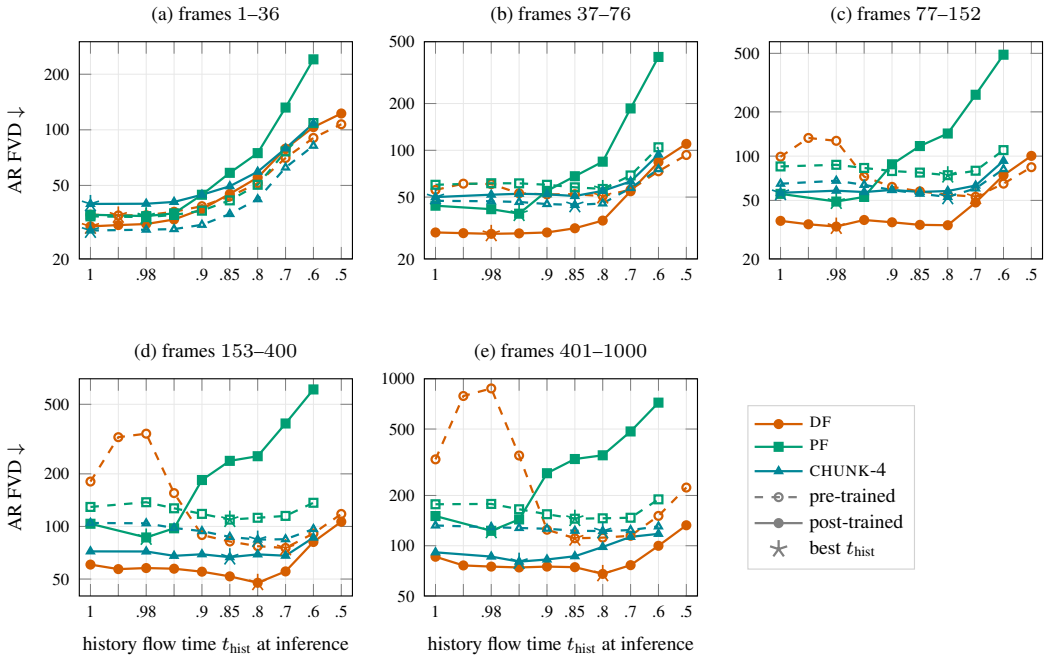


Figure 11: **History noise at inference, in every window.** The data of Table 9. Dashed, hollow: pre-trained. Solid, filled: post-trained. A star marks each curve's best t_{hist} in that window, which for most curves moves towards heavier noise as the horizon grows.

Table 10: **Retaining versus recomputing the KV cache.** $\Delta_{\text{cache}} = \text{FVD}_{\text{recompute}} - \text{FVD}_{\text{retain}}$ by window, 766 held-out clips where positive favours retaining. Grey entries have a paired 95% interval containing zero; the interval past frame 400 is given in full. "–": not evaluated.

			AR Δ_{cache}					FRAMES 401–1000	
			GT IN WINDOW						
MODEL	NFE	RECIPE	1–36	37–76	77–152	153–400	401–1000	AR 95% CI	TF Δ_{cache}
pre-trained	10	DF	0	0	−2	−1	−10	[−34, +11]	−0.1
		CTX−TAIL	0	0	−1	+1	+6	[−7, +18]	0
		CTX	0	0	−4	−13	−70	[−87, −56]	−3.3
		PF	0	0	−2	+3	+7	[−6, +20]	0
		CHUNK-4	0	0	−2	−3	+9	[−2, +19]	−0.2
pre-trained	4	DF	0	0	−3	+1	−17	[−54, +16]	–
		PF	0	0	−2	+4	+3	[−12, +22]	–
		CHUNK-4	0	0	−3	−5	−28	[−47, −13]	–
post-trained	4	DF	0	0	+1	+6	+13	[+3, +24]	0
		PF	0	0	0	+11	+36	[+23, +50]	+0.1
		CHUNK-4	0	0	+2	+9	+31	[+20, +40]	−0.2

D Extended related work

Exposure bias and history conditioning. Scheduled sampling and sequence-level training address the mismatch between training on data histories and generating from a model's own outputs (Bengio et al., 2015; Ranzato et al., 2016). Video diffusion also chooses how history is noised: DF samples independent frame noise levels (Chen et al., 2024), while rolling diffusion assigns more noise to later frames (Ruhe et al., 2024). History guidance combines predictions under different history configurations at inference (Song et al., 2025). Our profiles distinguish these presentation choices from the shift to generated content.

Few-step generation. Consistency models map points on a denoising trajectory to its data endpoint (Song et al., 2023). Flow map matching learns transitions between two times (Boffi et al., 2025); shortcut models condition on step size (Frans et al., 2025), and MeanFlow learns interval-average velocities (Geng et al., 2025). AnyFlow extends flow-map distillation to video with variable sampling budgets and on-policy supervision (Gu et al., 2026). We use its objective in S1 to study changes in mode coverage and AR stability across pre-training recipes.

Interactive world models and memory. DIAMOND trains agents inside a diffusion world model (Alonso et al., 2024), while Genie learns latent actions from unlabelled video (Bruce et al., 2024). Oasis and MineWorld provide interactive Minecraft models using diffusion and discrete tokens, respectively (Decart et al., 2024; Guo et al., 2025). PERSIST's evolving 3D state (Garcin et al., 2026) and MultiGen's editable spatial memory (Po et al., 2026) preserve information beyond a temporal context window. Such memory complements our study of training choices that improve robustness to generated histories.

E Limitations

A controlled end-to-end comparison is expensive, and we traded breadth for depth. Each of the three pipelines comprises full pre-training and two post-training stages, evaluated over 1000 frames, so we fixed one domain, one architecture and a resolution of 128×224. Doubling the resolution alone would quadruple the tokens per latent and multiply the cost of every pipeline. For the same reason each recipe was trained once, so our intervals capture variation across evaluation videos but not across training runs. Whether the rankings hold at larger scale, in other domains, or in text-to-video pipelines with a bidirectional teacher remains open. Our comparisons are also between complete recipes: DF keeps the uniform flow-time prior of its literature, and in post-training the recipe sets both the student initialisation and the teacher. We can therefore identify the better pipeline but not the responsible factor so crossing students with teachers is the natural next step.

Finally, our evidence on dynamic content and its persistence is qualitative. This reflects a gap in the field rather than in this study alone. FVD over short windows rewards realistic frames rather than a consistent world, and history noise at inference lowers FVD while changing object identities (Sec. 4.5). Measuring this is an open problem that we believe the community should prioritise.