Causality in Video Diffusers is Separable from Denoising (Supplementary Material)

Anonymous CVPR submission

Paper ID *****

A. Additional Experimental Setup

A.1. Datasets

TECO-Minecraft[19]. We adopt the long-context, action-conditioned prediction setup popularized by TECO [19]. Each video contains 300 frames, with 128×128 resolution, with per-frame action annotations. We evaluate on 256 video clips; for long-horizon quality (FVD), each clip supplies 36 ground-truth context frames followed by 264 generated frames; for frame-wise metrics (LPIPS, SSIM, PSNR), each clip supplies 144 observed frames followed by 156 generated frames. This set-up exactly aligns with TECO.

UCF0101[14]. We use the UCF-101 dataset to demonstrate the models' capability in real-world motion. This dataset comprises ∼13K unconstrained, unconditional action videos. Following MCVD [15]/ExtDM [22] and FAR [6], we randomly sample 256 videos and, for each, draw 100 stochastic trajectories. Pixel metrics (LPIPS/SSIM/PSNR) are computed best-of-100 per video, and FVD is averaged over all 100 trajectories.

RealEstate10K[1]. We also perform unconditional generation experiments on an auxiliary benchmark, RealEstate10K, a ddataset consisting of real-world indoor scenes. While this dataset is predominantly used in 3D tasks, we use it because it is a relatively small real-world dataset, where pretraining is feasible for our experiments. We use a resolution of 256. Since our model is orthogonal to camera-pose conditioning techniques, we simply perform unconditional prediction tasks with 16 context frames and 48 generated frames. Results are shown in Tab 1.

Tokenizer. Following the common practice, we compress video frames into video latent, and apply diffusion models in the corresponding latent space. To compress a video, we adopt a series of VAE and DCAE models [3]. For Minecraft and UCF, we use the DCAE trained in FAR [6] ; for RealEstate10K, we adopt the E2E-VAE tokenizer from [8] finetuned from VA-VAE [21].

Table 1. **Unconditional generation on RealEstate10K (128^2, 16→48).** Numbers are measured at 400k training steps with 50 denoising steps.

Model	Sec/F	16→48			
		LPIPS↓	SSIM↑	PSNR↑	FVD↓
Causal DiT-B	1.07	0.172	0.594	19.35	101.64
CSD-B	0.44	0.142	0.616	19.67	102.83
CSD-B$^{\mathbf{E}}$	0.45	<u>0.139</u>	<u>0.622</u>	<u>19.95</u>	<u>101.61</u>
CSD-B$^{\mathbf{D}}$	1.03	**0.135**	**0.623**	**20.01**	**85.12**

B. Model and Training Details

B.1. Design Details of Separable Causal Diffusion

Let $\mathcal{E}_\phi$ denote the causal reasoning encoder and $\mathcal{D}_\theta$ the frame-wise diffusion decoder. The encoder runs once per video frame outside the denoising loop to produce a latent c_t, summarizing the temporal context. Then, the decoder denoises each frame with multiple diffusion steps, conditioned on c_t:

$$c_t = \mathcal{E}_\phi(x_{<t}, a_{\le t}) \quad , \qquad \hat{v}_t^{(s)} = \mathcal{D}_\theta\big(x_t^{(s)}, s, c_t\big).$$

Here $x_t^{(s)}$ are noisy frame latents at step s, and $\hat{v}_t^{(s)}$ is the predicted velocity/score used by the sampler. $\mathcal{E}_\phi$ uses frame-wise causal attention and KV caches; $\mathcal{D}_\theta$ is perform intra-frame bidirectional attention. The amortized per-frame cost is therefore $O(\mathcal{E}_\phi) + S \cdot O(\mathcal{D}_\theta)$.

In our experiments, we choose $O(\mathcal{E}_\phi) \gg O(\mathcal{D}_\theta)$ because of two reasons: 1) Empirically, we observe that a large portion of layer features are sharable across the denoising process (Fig.2). 2) the encoder's cost is amortized across multiple denoising steps, so it can be made larger without significantly sacrificing efficiency.

Table 2. **Model variants and depth split.** Depth is split into ℓ causal blocks and m diffusion blocks. BP/frame $= \ell + S{\cdot}m$ with S=50.

Model	#Blocks	Hidden	#Heads	Params	BP / frame
DiT-B	12	768	12	131M	600
SCD-B	8+4	768	12	132M	208
SCD-B^E	12+4	768	12	174M	212
SCD-B^D	8+12	768	12	217M	608
FAR-M	12	1024	16	230	600
SCD-M	8+4	1024	16	230M	208
SCD-M^E	12+4	1024	16	306M	212
SCD-M^D	8+12	1024	16	383M	608

Algorithm 1 SCD Training

Require: Videos $\mathbf{x}_{1:N}$ with controls $\mathbf{a}_{1:N}$, where N is the number of frames; temporal reasoning module $\mathcal{E}_\phi$; frame diffusion module $\mathcal{D}_\theta$; diffusion loss $\mathcal{L}$; noisy multi-batch size K.

repeat
  Choose target frame $i \in \{1, \dots, N\}$
  $c_i \leftarrow \mathcal{E}_\phi(\mathbf{x}_{<i}, \mathbf{a}_{\leq i})$
  $\mathcal{L}_{\text{step}} \leftarrow 0$
  for $k = 1$ to K **do**
    Sample $t \sim \mathcal{U}[0,1]$, $\epsilon \sim \mathcal{N}(0, I)$
    $x_i^t \leftarrow (1-t)\, x_i + t\, \epsilon$
    $\hat{u} \leftarrow \mathcal{D}_\theta(x_i^t, t, c_i)$
    $\mathcal{L}_{\text{step}} \leftarrow \mathcal{L}_{\text{step}} + \mathcal{L}(\hat{u},\, x_i,\, \epsilon,\, t)$
  Take a gradient step on $\nabla_{\theta,\phi}\left(\mathcal{L}_{\text{step}}/K\right)$
until converged

Algorithm 2 SCD Generation by Roll-out over Frames

Require: Controls $\mathbf{a}_{1:N}$, where N is the number of frames to be generated; temporal reasoning module $\mathcal{E}_\phi$; frame diffusion module $\mathcal{D}_\theta$; sampler with T denoising steps and schedule $\{t_1, \dots, t_T\}$.

$\hat{\mathbf{x}} \leftarrow [\,]$ % generated frames buffer
for $i = 1, \dots, N$ **do**
  $c_i \leftarrow \mathcal{E}_\phi(\hat{\mathbf{x}}_{<i}, \mathbf{a}_{\leq i})$ % AR context: previously generated frames
  Initialize $z^T \sim \mathcal{N}(0, I)$
  for $t = T, T-1, \dots, 1$ **do**
    $\hat{u} \leftarrow \mathcal{D}_\theta(z^t, t, c_i)$
    $z^{t-1} \leftarrow \textsc{Sampler}(z^t, \hat{u}, t)$
  $\hat{x}_i \leftarrow z^0$; append $\hat{x}_i$ to $\hat{\mathbf{x}}$
return $\hat{\mathbf{x}}_{1:N}$

Table 3. **Ablations on the encoder-decoder interface.** Sequence fusion outperforms channel fusion; temporal RoPE is sslightly better than identical (non-causal) RoPE. For simplicity of ablation, metrics are reported at 400 training steps, with the unified "144 context frames, 156 generated frames" evaluation setup.

Encoder-Decoder Interface	FVD ↓	LPIPS ↓
Channel dim.	25.4	0.231
Frame dim. with temporal RoPE	**24.8**	**0.219**
Frame dim. with identical RoPE	25.1	0.223

B.2. Architectures and Model Variants

We follow the Diffusion Transformer (DiT) structure [12] to implement the SCD neural network. We follow DiT's width/head configuration for hidden size and MLP. To compare with FAR [6], the SOTA model on Minecraft, we also adopt its FAR-M parametrization. Table 2 enumerates the variants used in our experiments and reports BP/frame under S=50.

B.3. Algorithmic Pipeline

We summarize the end-to-end training and inference procedures of Separable Causal Diffusion (SCD) in Algorithms 1 and 2.

C. Ablation Studies

C.1. Encoder-Decoder Interface

We compare two ways of providing the context latent c_t to the frame-wise diffusion decoder $\mathcal{D}_\theta$: (i) *Channel Concatenation* we concatenates c_t with the noisy frame tokens along the channel dimension, projects back to the standard channel dimension with a linear layer, and feeds it to the decoder as input; (ii) *Frame Concatenation* We prepends c_t as a prefix frame for the noisy frame, and then feeds them into the decoder. This effectively positions c_t as the context frame of the current frame, performing self-attention together as a whole sequence. We also ablate the positional embedding applied to the context frame, either embedding it as "the last temporal frame" or "the current frame". As results in Tab. 3 demonstrates, sequence fusion strategy with P.E. embedded as historical frame outperforms other alternative.

C.2. Training Noisy Batches: Amortized Multi-Sample Decoding

Because $\mathcal{E}_\phi$ consumes only clean history, it is noise-agnostic. We therefore perform one efficiency trick in training: for each batch of clean videos, we encode once per frame to obtain c_t, then draw K i.i.d. noise/timestep pairs for the current frame and run $\mathcal{D}_\theta$ K times, saving the amortized cost for learning each noisy batch. As Tab. 4 demonstrates, throughput of noisy batch increases with K, which also improves the training speed.

Table 4. **Amortized multi-sample decoding in Training.** Encoder depth 8, decoder depth 4, as in our SCD-B and SCD-M models. For simplicity of ablation, metrics are reported at 400 training steps, with the unified "144 context frames, 156 generated frames" evaluation setup.

K	BP/clean batch	BP/noisy batch	noisy batch/s	FVD
1	$8+4\times1=12$	$8/1+4=12$	*22.0*	*23.9*
2	$8+4\times2=16$	$8/2+4=8$	*39.2*	*23.3*
4	$8+4\times4=24$	$8/4+4=6$	*63.0*	***23.1***

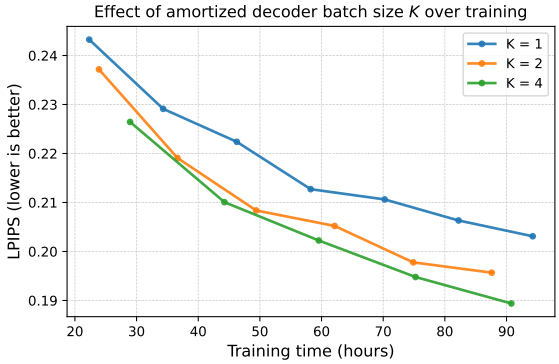


Figure 1. **Effect of amortized decoder batch size K at matched training time.** LPIPS on the validation set versus wall-clock training time (hours) for $K \in \{1, 2, 4\}$. Even when comparing at equal training time rather than equal optimization steps, larger K achieves lower LPIPS, indicating genuine gains from amortizing multiple noisy decoder samples per encoder pass.

Training-time comparison. The main-table ablation in Sec. C.2 compares different K at matched optimization steps, which slightly favors larger K because each step processes more independently noised targets. To control for this, Fig. 1 re-plots LPIPS against *wall-clock training time*. Despite the heavier per-step compute, curves with K=2 and especially K=4 reach lower LPIPS than K=1 at the same elapsed time. This shows that amortized multi-sample decoding does more than just see more noise per step: reusing the once-per-frame encoder output across multiple noisy decoder calls yields a better optimization trajectory even under a fixed compute budget.

C.3. Noisy Context Latent: Context Corruption and CFG

We perturb only the context c_i by adding Gaussian noise $\tilde{c}_t = c_t + \eta_t\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$, where c_t is normalized to unit variance and η_t^2 is the noise variance. Table 5 reports two complementary ablations on TECO–Minecraft at 400k steps. On the *left*, we vary the training-time corruption strength $\eta_t \in \{0, 0.05, 0.10, 0.20, 0.50\}$ and evaluate without any inference-time classifier-free guidance (CFG) on c_t: moderate noise around $\eta_t = 0.05$ improves FVD while stronger corruption eventually hurts. On the *right*, we fix training-time corruption at η_t=0.05 and ablate CFG using a negative branch with the same perturbation $\tilde{c}_t = c_t + 0.05\,\epsilon$ and guidance scale $\eta_{cfg} \in \{0.0, 1.0, 1.5, 2.0\}$; η_{cfg}=1.5 gives the best trade-off. Overall, modest training-time corruption plus a small CFG weight on the corrupted causal prior ($\eta_t \approx 0.05$, $\eta_{cfg} \approx 1.5$) yields the strongest long-horizon visual quality. Because these perturbations act only on the causal interface, they do not require re-caching any context frames.

Table 5. **Training-time causal corruption and test-time CFG with corruption (Minecraft, 400k steps).** Left: vary η_t at training and evaluate with no CFG on c_t (η_{cfg}=0). Right: fix η_t=0.05 at training and sweep inference-time CFG scale η_{cfg} on a corrupted causal prior $\tilde{c}_t = c_t + 0.05\,\epsilon$.

Training corruption			**Inference CFG with corruption**		
Noise η_t	FVD ↓	LPIPS ↓	η_{cfg}	FVD ↓	LPIPS ↓
0.00	24.8	0.199	0.0	24.2	0.254
0.05	**23.8**	0.195	1.0	23.1	0.223
0.10	24.5	0.195	**1.5**	**22.3**	**0.219**
0.20	25.1	**0.191**	2.0	22.7	0.221
0.50	27.6	0.199	(η_t=0.05)		

D. Additional Analysis of Section 4

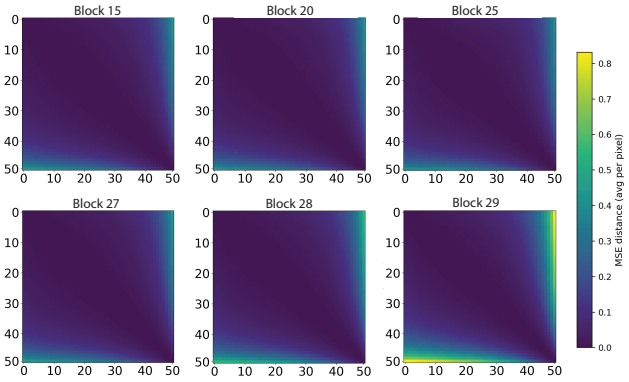
D.1. Redundancy across Denoising Steps

Feature similarity across denoising steps and depth. We extend the analysis in Fig. 2 to multiple depths of an autoregressively fine-tuned WAN-2.1 T2V-1.3B model. For a fixed set of prompts/seeds, we roll out the model for 50 denoising steps and, at every transformer block ℓ and step s, record the hidden features. We then compute a step–step *mean-squared-error (MSE) distance* matrix $\mathbf{S}_\ell \in \mathbb{R}^{T\times T}$ with entries $[\mathbf{S}_\ell]_{s,s'} = \|f_{\ell,s} - f_{\ell,s'}\|_2^2$, where $f_{\ell,s}$ denotes the layer-ℓ features at step s. Representative matrices for $\ell \in \{10, 15, 20, 25, 28, 29\}$ are shown in Fig. 2a and Fig. 2b

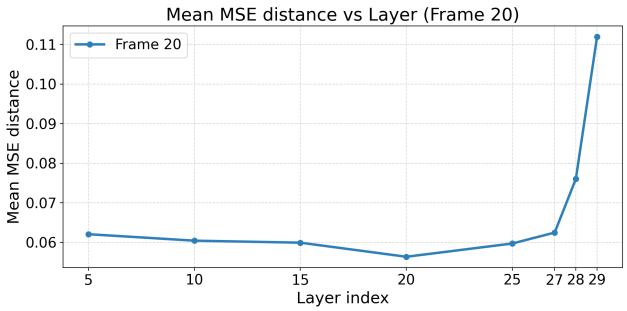
Layers 10–25 exhibit pronounced step-wise invariance: their similarity maps contain broad, near-uniform high-value bands (also visible at the 15th layer in the main paper), indicating that middle/early denoiser blocks repeatedly recompute almost the same features across the diffusion trajectory. In contrast, the last two layers ($\ell = 28, 29$) display markedly lower and more step-dependent similarity, consistent with these blocks performing step-specific, intra-frame rendering.

To further verify the redundancy observation, we fine-tune the baseline with a skip-layer design, in which the majority of denoising steps skip the middle-layers computation.

To further verify the redundancy finding, we fine-tune the causal baseline with a *skip-layer* schedule in which most



(a) Step–step *MSE-distance* matrices for multiple layers.



(b) Mean MSE distance versus block index. Each value here represenets the average value across the entire corresponding matrix on the left.

Figure 2. **Redundant computation across denoising steps.** (a) Step–step feature *MSE-distance* matrices of a fine-tuned AR WAN-2.1 T2V-1.3B model at several layer depths (layers 5, 10, 15, 20, 25, 28, 29). Middle layers (10–25) show broad, *low-distance* bands across all 50 denoising steps, indicating that their features are mostly invariant along the diffusion trajectory, whereas the last few layers exhibit more step-dependent distances. (b) A complementary per-layer summary: the average MSE distance across all pairs of denosing steps remains small in the early and middle layers, but increase dramatically in late layers.. Together, these views support our claim that early/middle blocks perform largely redundant computation across denoising steps, while the deepest blocks remain step-specific for intra-frame rendering, motivating our design that amortizes the first 25 layers once per frame and reuses them across all denoising steps.

denoising steps bypass the middle of the network. Concretely, only the first five denoising steps run the full denoiser; all subsequent steps traverse just a short *prefix* of 5 early layers and a *suffix* of 10 late layers, skipping the middle chunk. We retain the prefix because, as shown in Fig. 7 of the main text, early layers are particularly important during fine-tuning. The outcome (Fig. 3 in the main text) is that the skipped model produces high-quality videos and recovers the visual fidelity of the fully causal baseline, validating that most cross-step computation in early/middle layers is redundant and can be shared. This experiment is designed to test the observation in the simplest setting. In later SCD fine-tuning, we adopt a more aggressive recipe that also works: the first 25 layers run *once per frame* (amortized across steps), and only the final 5+5 layers participate in per-step denoising under our SCD design.

D.2. Evidence on Other Models

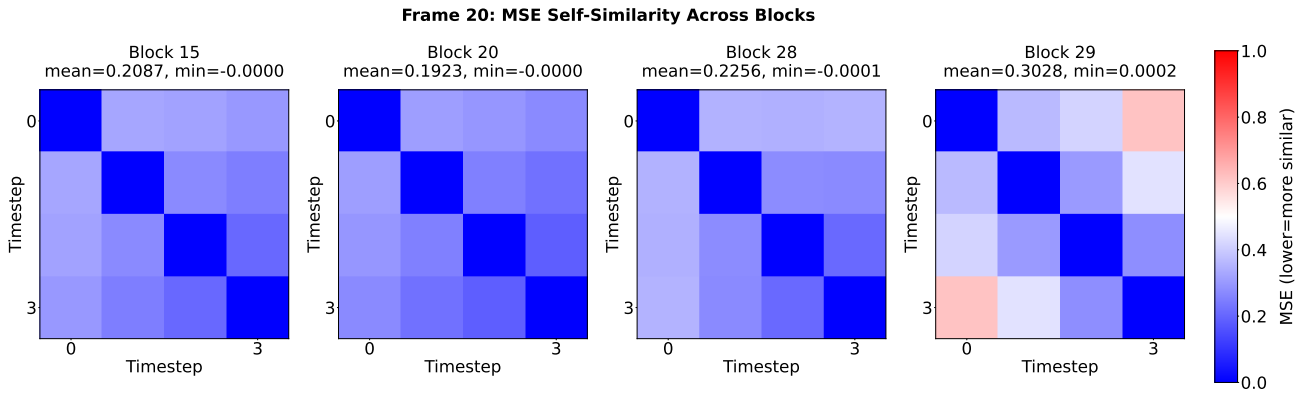
Beyond the WAN-2.1 (1.3B) model, we also evaluate an open-source, open-weight **Self-Forcing** (SF) 1.3B model [7]—a few-step, *block-autoregressive* student distilled from a bidirectional teacher that predicts three latent frames per block. We chose this model deliberately for two reasons. **(i) Block-autoregression** is a widely used generative pattern in contemporary video systems, and even in emerging language diffusion models [2], so validating our analysis on a block-AR student makes the conclusions relevant to a broad class of architectures [9, 13, 20]. **(ii) Self-forcing** is the prevailing post-training recipe for large autoregressive video models, bridging the train–test gap and distilling many-step teachers into efficient few-step samplers [4, 5, 7, 10]. By observing our claims on both a many-step WAN and a few-step, block-autoregressive, self-forced student, we cover complementary ends of the design space; the aligned observations across these regimes would substantially strengthen the generality of our claims.

Observations. We see the same patterns as in Section 4. Early and middle *layers* produce very similar features across denoising steps. PCA views change little with the step index and already capture global structure in the *first block*. Deeper layers are temporally sparse and focus on intra-frame rendering. These results indicate that the separability trends hold for both a 50-step WAN and a distilled 4-step *block-autoregressive* self-forcing model.

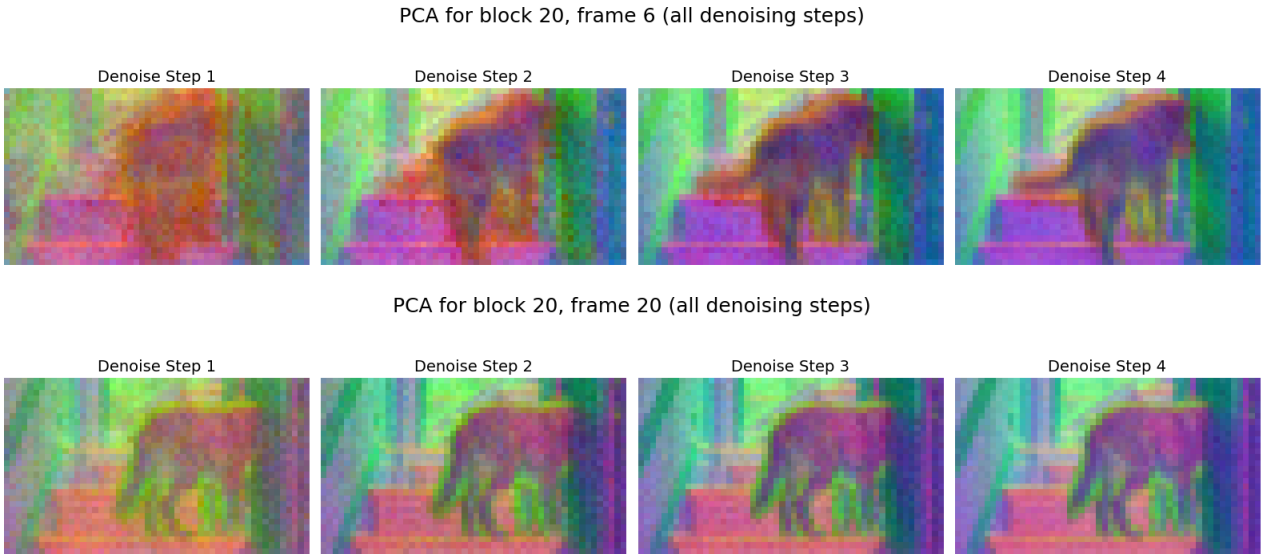
E. SCD Fine-tuning Details

This section describes fine-tuning details omitted from the main text. We first specify the teacher, data, and architecture adaptations used to convert a pretrained bidirectional video diffusion model into our Separable Causal Diffusion (SCD), which contains a causal encoder + a frame-wise diffusion decoder. We then detail the self-forcing rollout/distillation protocol used for post-training, and finally discuss capacity splits between encoder and decoder that highlight the flexibility of SCD.

Bidirectional Teacher. Unless otherwise noted, we fine-tune from a high-quality, bidirectional T2V checkpoint of WAN 2.1 T2V-1.3B [17, 18], whose weights are transplanted into our decoupled backbone (§E.1). All training



(a) **Self-forcing: step–step feature similarity** across multiple layers (analogous to Fig. 2(a) in Section 4). Early/middle layers show broad high-similarity (small MSE) bands over denoising steps.



(b) **Self-forcing: PCA of activations** (combined view; analogous to Fig. 2(b) in Section 4). Principal components remain stable across denoising steps and across *blocks* in the block-autoregressive rollout.

Figure 3. **Evidence on another model family (self-forcing, 4-step, block-autoregressive).** We replicate the Section 4 analyses on a few-step self-forcing student. (a) Early/middle layers exhibit high step–step feature similarity. (b) PCA views confirm that principal directions stabilize early and change little across denoising steps and *blocks*.

lies in the latent spaces derived from the original VAE of WAN 2.1.

Datasets. We use the text prompts from a 1M subset of VidProM [16] following the same filtering process in [7]. For fine-tuning with diffusion loss with our architecture, we use 70k synthetic data generated by WAN 2.1 T2V-14B with the above text prompts. For self-rollout training, we use the full 1M text prompts as conditions.

Training Specifications. We jointly train the encoder and decoder with the conditional flow-matching objective (Eq. (1) in the main text). For time step distribution, following WAN, we employ the timestep shifting $t'(k,t) = \frac{kt}{1+(k-1)t}$ and the forward interpolation is given as $x_t = (1-t')x + t'\epsilon$, $\epsilon \sim \mathcal{N}(0,I), t \in \mathcal{U}(0,1)$. We use the AdamW [11] optimizer for all experiments. Detailed hyperparameters can be found in the Table 6 and Table 7.

E.1. Architecture Adaptation for Decoupling

As described in the main paper, our decoupled backbone implements once-per-frame temporal reasoning in a causal encoder $\mathcal{E}_\phi$ and iterative rendering in a light frame-wise diffusion decoder $\mathcal{D}_\theta$, which differs from the teacher architecture. Therefore, to align the two architectures, in practice we make two adaptations when initializing from a bidirec-

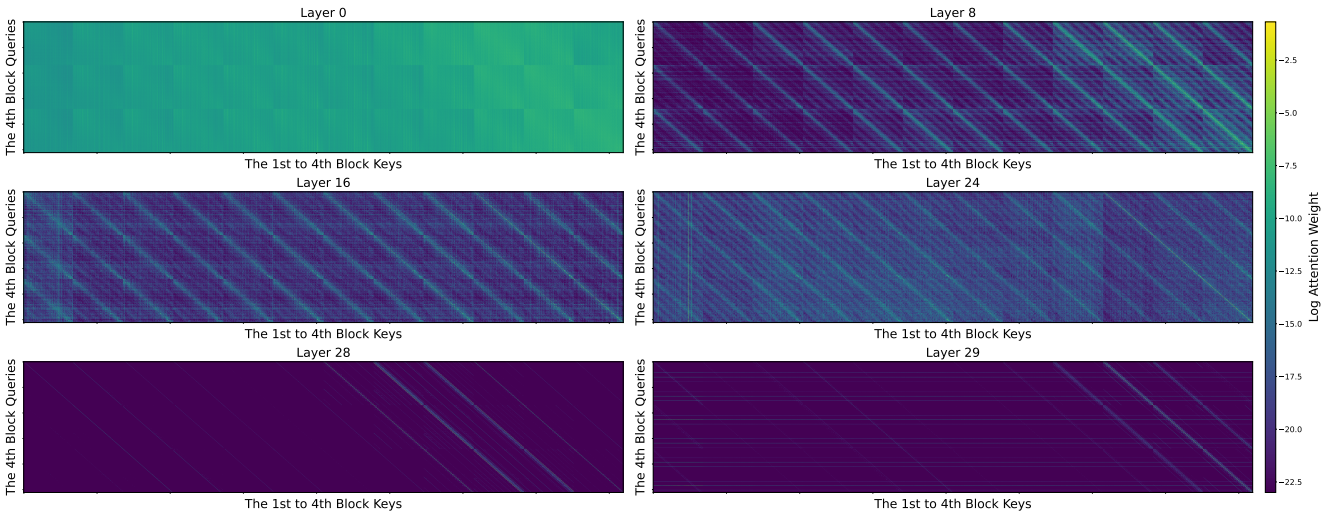


Figure 4. **Evidence on another model family (self-forcing, 4-step, block-autoregressive).** We replicate the Section 4 analyses on a few-step self-forcing student. As with the multi-step models, its deeper layers place minimal attention on past blocks, again revealing strong temporal sparsity.

tional teacher:

(i) Input reparameterization for the encoder. Pre-trained video diffusers consume a *noisy* current frame at each denoising step, while our encoder must operate on last generated frame instead of current frame. During fine-tuning, we therefore feed the encoder a corrupted current frame x_i^t at relatively *high* noise levels (e.g., top 20% of the diffusion/flow schedule), and at inference we replace it with pure Gaussian noise. This aligns the encoder's input distribution with the teacher while preserving the decoupled compute pattern (once per frame for $\mathcal{E}_\phi$, multi-step for $\mathcal{D}_\theta$).

(ii) Layer decomposition. Directly treating early layers as encoder and late layers as decoder often harms generation performance from finetuning. As discussed in the main text, the early layers play an important role in converting model input scale to an internal model scale. Guided by leave-one-out loss probing, we allocate the first 25 layers to $\mathcal{E}_\phi$ and build $\mathcal{D}_\theta$ by combining the first 5 and last 5 layers.

E.2. Hyperparameters

Table 6. **Fine-tuning hyperparameters.**

Resolution / Frames	832×480 / 81 frames
Batch size	64
LR / WD / Optimizer	2×10^{-5} / 0.01 / AdamW(0.9, 0.99)
EMA decay	0.99
Time sampler	$\frac{5t}{1+4t}, t \sim \mathcal{U}(0,1)$

Table 7. **Self-Forcing rollout training hyperparameters.**

Resolution / Frames	832×480 / 81 frames
Teacher	WAN 2.1 14B
Teacher CFG	3.0
Critic initialization	WAN 2.1 1.3B
Batch size	64
Student LR / WD / Optimizer	2×10^{-6} / 0.01 / AdamW(0.0, 0.99)
Critic LR / WD / Optimizer	4×10^{-7} / 0.01 / AdamW(0.0, 0.99)
Student EMA decay	0.99
Critic/student update ratio	5
Time sampler	$\frac{5t}{1+4t}, t \sim \mathcal{U}(0,1)$

Throughput and latency. We report wall-clock throughput (FPS) and per-frame latency with batch size 1 on 1×H100 80 GB, charging the initial frame's extra compute. SCD fine-tuned from a strong T2V teacher achieves ∼11.1 FPS with 0.29 s latency at 832×480 while retaining competitive VBench scores; the self-forcing baseline at the same scale reaches 8.9 FPS and 0.45 s latency.(Table 8).

Table 8. **Frame-Autoregressive Text-to-Video on VBench (832×480, batch 1).** Reported throughput includes first-frame overhead.

Model	FPS ↑	Latency (s) ↓	Total ↑	Quality / Semantic ↑
Self Forcing	8.9	0.45	84.26	85.25 / 80.30
SCD (Ours)	**11.1**	**0.29**	84.03	85.14 / 79.60

E.3. Flexibility of Separable Causal Diffusion

A practical benefit of SCD is that temporal reasoning capacity and per-frame rendering capacity can be traded *in-*

dependently. Let total depth be $\ell+m$ with encoder depth ℓ (causal reasoning, amortized once per frame) and decoder depth m (frame-wise denoising, repeated S steps in inference, where $S = 4$ in standard self-forcing settings). For fixed parameters:

- **Encoder-heavier variants** slightly increase reasoning cost per frame but systematically improve motion/layout adherence and long-horizon stability; it only slightly reduce the throughput since the encoder runs once per frame.
- **Decoder-heavier variants** improve per-frame detail at the cost of $S\times m$ block passes per frame; quality gains can be notable when targeting high-fidelity or very few denoising steps, but latency rises accordingly.

F. Additional Visualization

We provide an HTML viewer with additional video samples in `Video_Samples/view_T2V.html`. It contains selected text-to-video generations from our 25-layer encoder, 10-layer decoder Separable Causal Diffusion model under T2V setting; each clip is shown alongside its full text prompt so that content and conditioning can be inspected together. The folder also includes additional generations from models trained *from scratch* on TECO-Minecraft and RealEstate10K, illustrating long-horizon prediction and real-scene rendering in our decoupled architecture beyond the WAN fine-tuning regime.

References

[1] Realestate10k: A large dataset of camera trajectories from videos. https://google.github.io/realestate10k/, 2018. 1

[2] Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025. 4

[3] Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models, 2025. 1

[4] 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. *arXiv preprint*, 2025. 4

[5] FastVideo Team. Fastvideo CausalWan2.2-I2V-A14B-Preview-Diffusers. https://huggingface.co/FastVideo/CausalWan2.2-I2V-A14B-Preview-Diffusers, 2025. Hugging Face model card. 4

[6] Yuchao Gu, Weijia Mao, and Mike Zheng Shou. Long-context autoregressive video modeling with next-frame prediction. *arXiv preprint*, 2025. 1, 2

[7] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint*, 2025. 4, 5

[8] Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. Repa-e: Unlocking vae for end-to-end tuning with latent diffusion transformers. *arXiv preprint arXiv:2504.10483*, 2025. 1

[9] 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. *arXiv preprint arXiv:2506.09350*, 2025. Seaweed APT-2. 4

[10] Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. *arXiv preprint arXiv:2509.25161*, 2025. 4

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 5

[12] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2

[13] Joonghyuk Shin, Zhengqi Li, Richard Zhang, Jun-Yan Zhu, Jaesik Park, Eli Shechtman, and Xun Huang. Motionstream: Real-time video generation with interactive motion controls. *arXiv preprint arXiv:2511.01266*, 2025. 4

[14] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human action classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 1

[15] Vikram Voleti, Alexia Jolicoeur-Martineau, and Christopher Pal. Mcvd: Masked conditional video diffusion for prediction, generation, and interpolation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1

[16] Wenhao Wang and Yi Yang. Vidprom: A million-scale real prompt-gallery dataset for text-to-video diffusion models. 2024. 5

[17] WanTeam. Wan: Open and advanced large-scale video generative models. *arXiv preprint*, 2025. 4

[18] WanTeam. Wan 2.1 T2V-1.3B: Open-source text-to-video model. *arXiv preprint arXiv:2503.00123*, 2025. 4

[19] Wilson Yan, Danijar Hafner, Stephen James, and Pieter Abbeel. Temporally consistent transformers for video generation. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. 1

[20] Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, Song Han, and Yukang Chen. Longlive: Real-time interactive long video generation. *arXiv preprint*, 2025. 4

[21] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models, 2025. 1

[22] Zhicheng Zhang, Junyao Hu, Wentao Cheng, Danda Paudel, and Jufeng Yang. Extdm: Distribution extrapolation diffusion model for video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1