REFDECODER: Enhancing Visual Generation with Conditional Video Decoding

Xiang Fan[1*] **Yuheng Wang**[1*] **Bohan Fang**[1*] **Zhongzheng Ren**[1,2] **Ranjay Krishna**[1]
[1]University of Washington [2]University of North Carolina at Chapel Hill
[*]Equal contribution
xiangfan@cs.washington.edu
https://refdecoder.github.io/

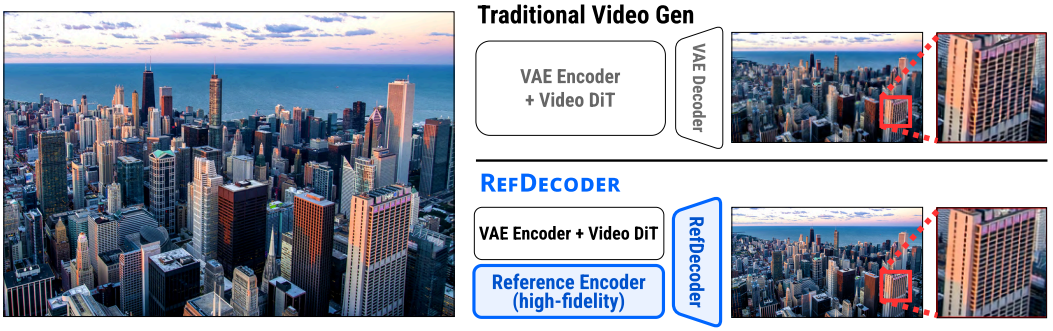




Figure 1: (a) **REFDECODER** improves video VAE decoders by conditioning on a high-fidelity reference signal that bypasses the lossy VAE latent round trip. Given the encoded latents, the decoder injects fine-grained details that are not preserved in the VAE latent space, improving reconstruction fidelity and downstream generation quality. (b) **REFDECODER** integrates with existing video generation pipelines (Wan 2.1 shown) without requiring costly re-training of the diffusion model.

Abstract

Video generation powers a vast array of downstream applications. However, while the *de facto* standard, *i.e.*, latent diffusion models, typically employ heavily conditioned denoising networks, their decoders often remain unconditional. We observe that this **architectural asymmetry** leads to significant loss of detail and inconsistency relative to the input image. To address this, we argue that the decoder requires

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

equal conditioning to preserve structural integrity. We introduce REFDECODER, a reference-conditioned video VAE decoder by injecting high-fidelity reference image signal directly into the decoding process via reference attention. Specifically, a lightweight image encoder maps the reference frame into the detail-rich high-dimensional tokens, which are co-processed with the denoised video latent tokens at each decoder up-sampling stage. We demonstrate consistent improvements across several distinct decoder backbones (*e.g.*, Wan 2.1 and VideoVAE+), achieving up to $+2.1$ dB PSNR over the unconditional baselines on the Inter4K, WebVid, and Large Motion reconstruction benchmarks. Notably, REFDECODER can be directly swapped into existing video generation systems without additional fine-tuning, and we report across-the-board improvements in subject consistency, background consistency, and overall quality scores on the VBench I2V benchmark. Beyond I2V, REFDECODER generalizes well to a wide range of visual generation tasks such as style transfer and video editing refinement.

1 Introduction

In the dominant paradigm of video generation, a diffusion backbone denoises latent representations guided by an input conditioning, while a VAE decoder subsequently maps these latents back to pixel space [2, 40, 52]. While substantial effort has been devoted to improving the conditioning and architecture of the diffusion backbone [28, 13, 31], *the VAE decoder has received comparatively little attention and remain unconditional in mainstream video diffusion models*. It is typically treated as a standalone reconstruction module that operates without access to the primary conditioning signal at inference time [2, 40, 52, 17].

We observe that this **architectural asymmetry** creates a systematic bottleneck: even when the diffusion model faithfully preserves the reference information within the latent, the decoder must reconstruct fine-grained spatial details from a heavily compressed representation without a detailed anchor. This leads to two characteristic failure modes: (1) *Progressive degradation of spatial details*, where textures, edges, and high-frequency content deteriorate in frames that deviate from the conditioning image (e.g. Figure 3 baseline details); and (2) *Temporal inconsistency*, where appearance drifts across the sequence (e.g. Figure 4(a) human face). These artifacts emerge during latent-to-pixel decoding rather than the diffusion process itself, exposing the decoder as an under-exploited component that fails to leverage the visual cues already provided by the reference.

However, injecting the reference image into the decoder is non-trivial for several reasons. (1) Because the decoder operates through hierarchical up-sampling stages, it remains an open question at which specific stage the reference signal should be introduced. (2) The conditioning mechanism must maintain compatibility with pretrained decoder weights, ensuring the system remains functional in its original setting, *i.e.*, no reference image. (3) The approach should be sufficiently generic to generalize across diverse visual generation tasks and various decoder backbones.

To address aforementioned problems, our key insight is that the decoder's hidden space is far richer than the VAE latent space; encoding the reference image into this high-fidelity space and injecting it as additional self-attention tokens lets the decoder exploit details unavailable from the latent code alone. Building on this, we propose **REFDECODER**, a reference-conditioned video VAE decoder with two components: a **reference image encoder**, a lightweight convolutional network mapping the reference frame into the decoder's hidden space as spatially aligned tokens (Sec. 2(a)); and a **conditional token decoder** that, at each up-sampling stage, concatenates reference and video latent tokens along the temporal axis and jointly processes them via a shared transformer block with rotary positional embeddings (RoPE) [33]; the tokens are then separated and up-sampled independently, preserving compatibility with the pretrained decoder (Sec. 2(b)). To prevent the decoder from ignoring the reference, we further introduce a **latent token dropout** strategy that randomly zeroes video latent tokens at a variable rate, forcing the model to recover spatial details from reference tokens via attention (Sec. 2(c)).

Our newly introduced modules are architecture-agnostic and generalize to existing video VAE decoders, with the transformer block shared across decoder stages. Because only the decoder is modified and the encoder remains frozen, REFDECODER is a **drop-in replacement** for any existing video VAE decoder and can immediately benefit downstream diffusion models without retraining.

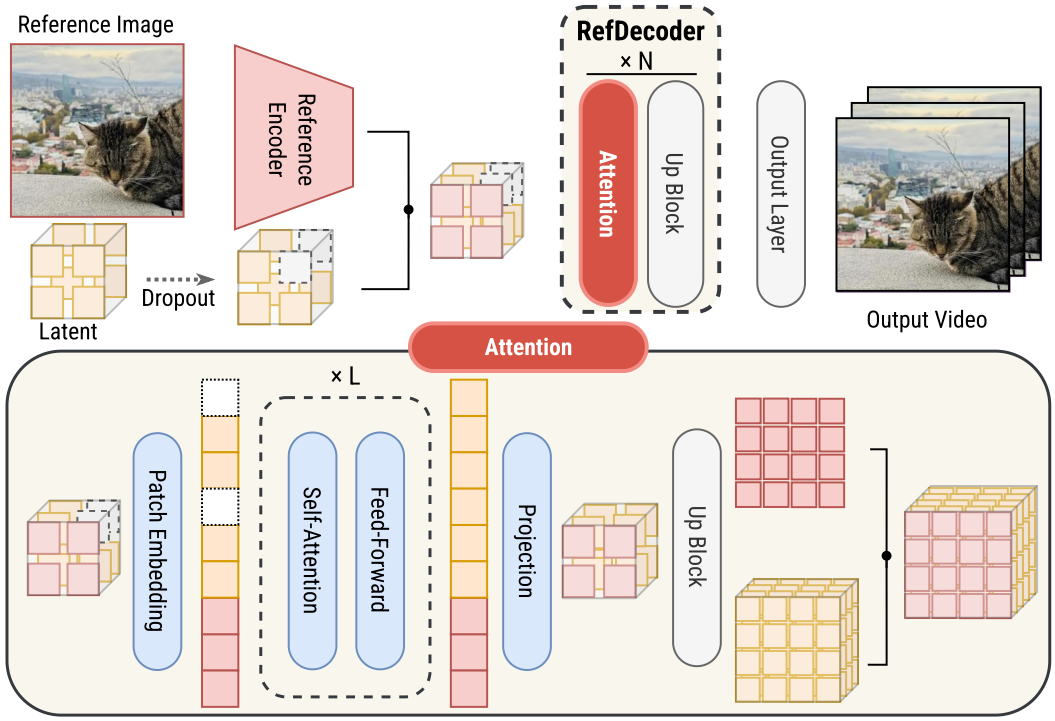


Figure 2: **Architecture overview of the REFDECODER.** The reference image is encoded into tokens and injected into the decoder through shared attention blocks, enabling interactions between reference and latent tokens during decoding.

We test REFDECODER on different video VAE backbones including Wan 2.1 [40] and VideoVAE+ [49], demonstrating consistent improvements across all of them. On the Inter4K reconstruction benchmark, REFDECODER achieves over 1 dB PSNR improvement compared to the unconditional baseline. In generation tasks, on the VBench I2V benchmark [14, 15], our method improves subject consistency, background consistency, motion smoothness, aesthetic quality, and overall scores. We show that the same reference injection mechanism naturally extends to *decode-time style transfer*. By supplying a style image as the reference, REFDECODER produces stylized video without any task-specific architecture modification, validating the generality of conditional token decoding. Furthermore, in video editing tasks, REFDECODER enhances the fidelity of edited videos to the input reference while preserving the intended edits, demonstrating that reference conditioning as a powerful tool to mitigate the trade-off between editability and fidelity [10]. These results suggest a broader message: in image conditioned generation pipelines, the decoder is not merely a passive reconstruction module but an active participant that can—and should—leverage conditioning signals. Don't forget the decoder.

2 Method

We propose **REFDECODER**, a reference-conditioned video VAE decoder that injects reference image tokens directly into the decoding process via joint attention over concatenated reference and video tokens, enhancing spatial fidelity and temporal coherence. The overall architecture is illustrated in Fig. 2. We build upon a standard video VAE backbone and augment its decoder with conditional token interactions, which we detail below.

Reference image encoding. We start with a reference image $I_{\text{ref}} \in \mathbb{R}^{3\times H\times W}$. Typical methods encode the reference into a low-dimensional feature space comparable to the VAE latent bottleneck (*e.g.* 16 channels), and obtain hierarchical features derived therefrom. To preserve high-fidelity information, we instead project the original image patches directly into a high-dimensional feature space (*e.g.* 512 dimensions), which is fed to the decoder at the very first stage, upsampled alongside the video features throughout decoding stages, and interacted with through injected attention layers at every stage (Section 2(b)). This setup allows multiresolution high-dimensional feature extraction within the decoding process.

We keep the reference encoder minimal—applying a single convolution followed by normalization—to avoid any issue with a deeper encoder smoothing out high-frequency details.

After encoding, we obtain reference tokens $\mathbf{z}_{\text{ref}} \in \mathbb{R}^{C \times \frac{H}{p} \times \frac{W}{p}}$, where C is the first decoder stage's channel dimension and p is the VAE's spatial compression ratio. This aligns reference tokens with the decoder's initial feature map, enabling direct position-aware information transfer through attention.

Conditional token decoding. Standard video VAE decoders rely on causal 3D convolutional upsampling layers with limited local receptive fields, which cannot recover fine-grained details lost to the narrow latent bottleneck. To let the decoder selectively retrieve details from the reference, we insert Transformer blocks into the upsampling stages of the pretrained decoder, enabling each video token to query reference tokens and extract relevant information.

Our conditional token decoding involves three design goals: (1) combining reference and video tokens for joint processing; (2) preserving compatibility with the existing VAE upsampling path; and (3) minimizing parameter overhead.

Token concatenation. At decoder stage s, we concatenate video tokens $\mathbf{z}^{(s)} \in \mathbb{R}^{C_s \times T_s \times H_s \times W_s}$ and reference tokens $\mathbf{z}_{\text{ref}}^{(s)} \in \mathbb{R}^{C_s \times 1 \times H_s \times W_s}$ along the temporal dimension, yielding a tensor of shape $\mathbb{R}^{C_s \times (1+T_s) \times H_s \times W_s}$. This is then processed through self attention in a Transformer block.

Separation and upsampling. After attention, the output is split back into reference tokens $\hat{\mathbf{z}}_{\text{ref}}^{(s)}$ and video tokens $\hat{\mathbf{z}}^{(s)}$. Each branch is then passed through the pre-trained upsampling modules independently. The reference tokens are upsampled only spatially, while the video tokens undergo both spatial and temporal upsampling.

Weight sharing across stages. To reduce the parameter overhead of Transformer blocks, Transformer weights are shared across all stages. Stage-specific patch embedding layers project varying channel dimensions C_s to a unified transformer hidden dimension, enabling weight sharing without architectural constraints. We ablate the effect of the number of weight-shared blocks in Section 3.7.

Latent token dropout. To encourage the decoder to actively rely on the reference rather than solely on the video latents, we apply token dropout to the latents before decoding. At each training step, every spatiotemporal position in $\mathbf{z}$ is independently zeroed with probability r, where r is sampled uniformly from $[0, r_{\max})$, forcing the model to retrieve missing details from the reference tokens via attention. We use $r_{\max} = 0.7$ by default.

Training. We initialize the VAE from a pretrained video VAE and freeze the encoder entirely. The decoder is fine-tuned at a reduced learning rate, while the newly added modules (reference image encoder and Transformer blocks) are trained from scratch. The training objective is:

$$\mathcal{L} = \|\mathbf{x} - \hat{\mathbf{x}}\|_1 + \mathcal{L}_{\text{LPIPS}}(\mathbf{x}, \hat{\mathbf{x}})$$

where $\mathbf{x}$ and $\hat{\mathbf{x}}$ denote the ground-truth and reconstructed video frames respectively, and $\mathcal{L}_{\text{LPIPS}}$ is the LPIPS perceptual loss [60].

Random frame reference selection. While the reference image typically serves as the first frame at inference, the model should ideally be robust to a wider range of reference content. We therefore randomly select a frame from the input video as the reference during training, preventing overfitting to a fixed temporal relationship.

Two-stage curriculum training. We first train on short clips of the minimum supported number of frames, then extend to the maximum frame count to generalize to long temporal contexts.

Causal variant for long videos. We also fine-tune a causal variant (4K steps, same data) that decodes the latents chunk by chunk, like the Wan 2.1 VAE decoder, rather than all at once.

3 Experiments

We evaluate REFDECODER on **video reconstruction** and **image-to-video generation**. Throughout, REFDECODER is used as a plug-and-play *drop-in replacement* for the baseline VAE decoder: the encoder, the diffusion backbone, and the inference pipeline remain unchanged. We also ablate the number of Transformer blocks, latent token dropout, and the training curriculum.

3.1 Experimental Setup

Training data. Our training set contains approximately 100K videos including videos from MiraData9K [19, 63], DL3DV [19, 63], and a subset of OpenVidHD-0.4M [22].

Benchmarks. For reconstruction, we use the three test sets provided by VideoVAE+ [49]—an Inter4K [32] test split of 500 high-quality videos, a subset of WebVid [1], and a Large Motion subset of 100 videos (80 from WebVid, 20 from Inter4K) curated for complex camera and object motion—using identical splits to ensure direct comparability with [49] (see Section B.5). For long videos, we reconstruct 81 frames on 500 videos randomly sampled from Inter4K (Inter4K-81; drawn independently of the Inter4K test list, with which it may overlap) and run VBench at 81 frames. We report PSNR, SSIM, and LPIPS for reconstruction, and the standard 12 VBench dimensions [14, 15, 62] for generation.

Implementation details. We apply REFDECODER to two architecturally distinct backbones, Wan 2.1 [40] and VideoVAE+ [49], which differ substantially in their channel widths and decoder depth. We freeze the encoder and train only the decoder, and drop up to 70% latent video tokens during training. We adopt a two-stage curriculum: for Wan 2.1, 5-frame clips at 480×832 followed by 17-frame clips; for VideoVAE+, 4-frame clips at 216×216 (as does [49]) followed by 16-frame clips at the same resolution. All 81-frame results use the causal variant REFDECODER† (Section 2), fine-tuned from the default checkpoint for 4K steps (about 12 hours) on the same data with the same reference sampling, and compared against the original Wan 2.1 decoder.

Baselines. We compare our method with the vanilla VAE autoencoders from frontier open source video models, including Wan 2.1 [40] and HunyuanVAE [17], as well as state-of-the-art academic VAE models including VideoVAE+ [49] and Reducio [36].

3.2 Reconstruction Results

Table 1 reports reconstruction metrics on Inter4K, WebVid, and the Large Motion subset from [49]. REFDECODER consistently outperforms existing baselines, with gains of +1.2 dB PSNR on Inter4K (Wan 2.1) and +2.1 dB PSNR on WebVid (VideoVAE+), and reduced LPIPS across the majority of the settings, demonstrating fine detail recovery and architectural generalization. A per-category breakdown shows that gains are positive across every category and are largest for content-rich scenes (Appendix B.6), a setting that many video generation pipelines struggle with [40, 17, 52].

We additionally compare against three prior reference-conditioned VAEs: H3AE [47], Reducio-VAE [36], and RefTok [11]. Quantitative numbers for Reducio-VAE on our benchmarks are included in Table 1. Since H3AE and RefTok have not released code or weights, we follow each method's reported evaluation protocol and run REFDECODER (VideoVAE+ backbone) on the same dataset—DAVIS [24, 25] for H3AE and BAIR [7] for RefTok. As shown in Table 2, REFDECODER outperforms both under their own evaluation settings: it matches H3AE in PSNR while improving SSIM by 1.5 on DAVIS, and surpasses RefTok by 7.4 dB PSNR and 4.1 SSIM on BAIR, with LPIPS reduced from 0.013 to 0.005.

Long and high-motion videos. On 81-frame reconstruction (Inter4K-81), REFDECODER† improves every metric over the Wan 2.1 decoder ($+0.4$ dB PSNR, $+1.3$ SSIM, -0.012 LPIPS; Table 5). To test whether the gains hold under strong motion, we rank the 500 videos by a per-video motion score computed with RAFT [35] optical flow and keep the top quartile, yielding 125 videos with a mean (median) motion of 16.28 (14.08) px/frame. On this subset the gains grow to $+0.8$ dB PSNR, $+2.0$ SSIM, and -0.023 LPIPS. From the full set to the high-motion subset, Wan 2.1 drops by 0.8 dB PSNR and 1.9 SSIM and its LPIPS rises by 0.017, whereas REFDECODER† drops by only 0.4 dB PSNR and 1.2 SSIM, with LPIPS rising by 0.006. REFDECODER is therefore less sensitive to strong motion and is not limited to static or low-motion videos.

We further examine robustness to temporal distance and reference mismatch. At a distance of 10 frames from the reference, REFDECODER retains a PSNR gain of approximately 0.4 dB. On a scene-boundary stress test, it improves PSNR and SSIM, although LPIPS worsens, revealing a trade-off under reference mismatch. Full results are provided in Appendix D.

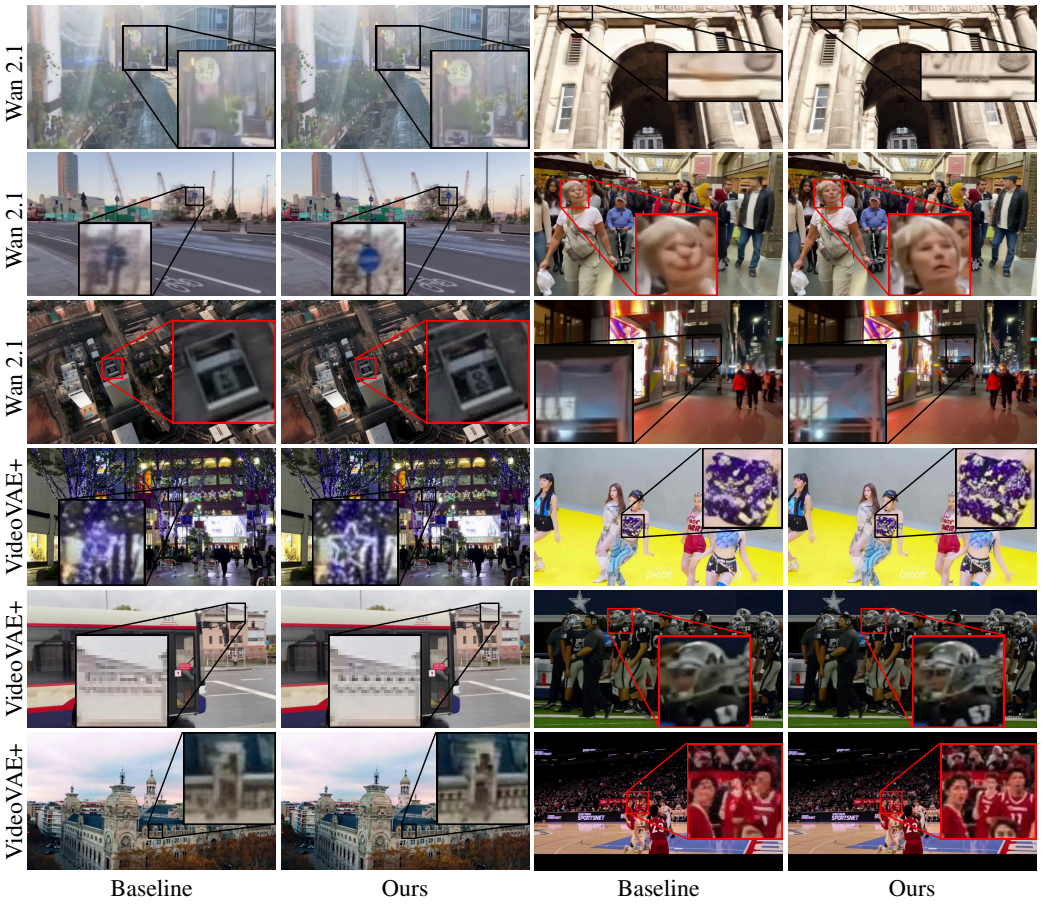


Figure 3: **Reconstruction comparisons.** Cropped regions from two backbones (Wan 2.1 and VideoVAE+) are shown. REFDECODER produces sharper reconstructions and better preserves fine details than the baseline, particularly for high-frequency content like text, human faces, and structural patterns.

3.3 Generation results

We plug REFDECODER into the unmodified Wan 2.1 pipeline as a drop-in decoder replacement, without any retraining of the diffusion model. To isolate the decoder's contribution, we adopt a *fixed-seed* protocol: for each prompt we draw 5 random seeds once, generate diffusion latents with the unmodified Wan 2.1 pipeline, and decode the same latents with both Wan 2.1 baseline decoder and REFDECODER, so that diffusion sampling variance does not affect the pairwise comparison and the improvement is attributable solely to the decoder. Evaluation details are in Appendix B.7.

As shown in Table 3 (17 frames), REFDECODER outperforms the baseline on 10 out of 12 VBench dimensions and is on par in imaging quality, with the largest gains in subject and background consistency, and raises the total VBench score from 87.9 to 88.2 (+0.3).

Long and high-motion videos. Repeating the fixed-seed protocol with 81 frames instead of 17, REFDECODER† raises the total score from 87.9 to 88.1 and improves all three aggregate scores and 6 of the 9 dimensions, with the largest gains in imaging quality and subject consistency (Table 3, 81 frames). We further evaluate the highest-motion quartile of the original, unmodified VBench set, selected using a per-video mean RAFT optical-flow threshold of 14.883. As shown in Table 4, REFDECODER† improves Quality, I2V, and Total by 0.17, 0.35, and 0.26 percentage points, compared with 0.13, 0.25, and 0.19 on the full set, so the generation improvements persist under substantial motion.

Temporal stability. We also assess temporal stability under the same fixed-seed protocol with flow warping error (E_{warp}) [18], temporal LPIPS (E_{tLPIPS}) [60], flicker (E_{flicker}) [4], and CLIP consistency

Table 1: **Video reconstruction results on WebVid, Inter4K (test), and Large Motion datasets.** Best results are shown in bold. Our method consistently outperforms the Wan 2.1 baseline and VideoVAE+ across most metrics.

	WebVid			Inter4K (test)			Large Motion		
Model (Backbone)	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**
Cosmos-Tokenizer [23]	31.2	89.6	0.107	31.3	88.6	0.103	30.2	86.8	0.119
CogVideoX-VAE [52]	32.5	92.3	0.053	32.9	92.1	0.050	31.1	89.8	0.069
EasyAnimate-VAE [50]	31.5	90.5	0.057	32.1	90.9	0.041	30.5	88.5	0.060
CV-VAE [61]	31.6	90.6	0.064	32.3	90.8	0.055	30.7	88.7	0.073
Open-Sora-Plan (WF-VAE-L) [20]	32.5	91.5	0.051	33.1	92.1	0.040	31.2	89.6	0.060
IV-VAE [46]	33.2	92.7	0.041	33.2	92.3	0.038	31.9	90.9	0.049
VidTok [34]	33.4	93.1	0.043	34.4	93.6	0.034	31.9	90.8	0.053
HunyuanVAE [17]	33.2	92.7	0.037	33.4	92.7	0.032	32.1	91.2	0.045
Reducio-VAE [36] (T/4)	32.4	88.8	0.057	29.7	86.3	0.096	26.8	78.6	0.126
Wan 2.1 VAE [40]	32.8	91.4	0.032	33.7	93.0	0.034	30.8	89.1	0.048
RefDecoder (Wan 2.1)	33.5	92.6	0.037	**34.9**	**94.9**	**0.031**	31.4	90.4	0.053
	(+0.7)	(+1.2)	(+.005)	(+1.2)	(+1.9)	(-.003)	(+0.6)	(+1.3)	(+.005)
VideoVAE+ [49]	32.9	93.0	0.041	33.9	93.3	0.034	31.9	90.7	0.050
RefDecoder (VideoVAE+)	**35.1**	**93.5**	**0.031**	34.7	94.7	0.033	**32.9**	**92.3**	**0.043**
	(+2.1)	(+0.5)	(-.010)	(+0.8)	(+1.5)	(-.001)	(+1.1)	(+1.6)	(-.007)

Table 2: **Comparison with prior reference-conditioned VAEs** on each method's own benchmark: H3AE on DAVIS [24, 25] and RefTok on BAIR [7]. RefDecoder uses the VideoVAE+ backbone.

DAVIS			BAIR			
Model	PSNR↑	SSIM↑	**Model**	PSNR↑	SSIM↑	LPIPS↓
H3AE [47]	33.0	89.9	RefTok [11]	28.8	95.0	0.013
RefDecoder	**33.1**	**91.4**	**RefDecoder**	**36.2**	**99.1**	**0.005**

($\text{CLIP}_{\text{cons}}$) [8]. RefDecoder reduces all three degradation metrics by 8–15% and slightly improves CLIP consistency, producing less flicker and more temporally coherent videos (Appendix C).

3.4 Human evaluation

We conduct a blind preference study comparing RefDecoder with four VAE baselines: Wan 2.1, HunyuanVAE, VideoVAE+, and Reducio-VAE. Participants use a slider-overlay interface in which the two videos are shown on randomized sides and model identities are hidden. Overall, RefDecoder is preferred over every baseline. In reconstruction tasks, RefDecoder is preferred over Wan 2.1, HunyuanVAE, VideoVAE+, and Reducio-VAE, with preference rates of 92.3%, 69.2%, 63.4%, and 89.6%, respectively. In generation tasks, RefDecoder is preferred over Wan 2.1 with a preference rate of 82.4%. Appendix F provides further detail.

3.5 Qualitative results

Figure 3 shows reconstruction results. Our decoder recovers sharper edges and finer textures than the baseline, particularly in high-frequency regions such as text, human faces, and fine structural details. These improvements are consistent across backbones. Figure 4 compares generation results produced using the baseline decoder vs. RefDecoder. As we can see, RefDecoder better preserves the structure and produces sharper, more consistent details across frames.

3.6 Efficiency

RefDecoder adds overhead only to the lightweight VAE decoding stage, whereas the DiT denoising stage dominates the diffusion pipeline. When run on Wan 2.1, the total pipeline latency is nearly equal with minimal overhead ($72{,}037 \pm 16$ ms vs. $74{,}584 \pm 40$ ms).

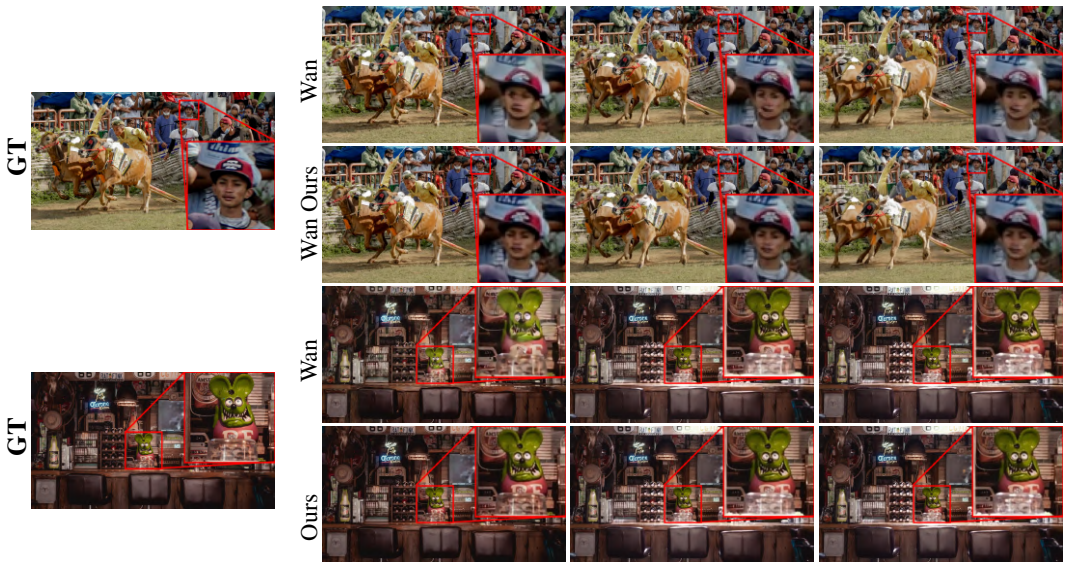


Figure 4: **Video generation comparisons.** For the same input reference, three frames generated at different timesteps are shown. The baseline distorts the underlying scene structure, whereas REFDECODER preserves the structure while producing sharper and more consistent details across frames.

3.7 Ablation study

Effect of the number of blocks. We ablate the number of Transformer blocks on Wan 2.1 with 3, 5, 7, and 10. Reconstruction quality improves with depth: the 10-blocks model achieves 34.9 dB PSNR on Inter4K, up from 34.3 for the 3-blocks variant, and is best across all benchmarks (Appendix I).

Latent token dropout and two-stage curriculum. We further ablate two training-time choices: latent token dropout and the two-stage curriculum. We find that higher latent-token dropout consistently improves reconstruction. Gains appear on both the reference and non-reference frames, indicating that stronger dropout encourages better use of reference information. The two-stage curriculum substantially improves over one-stage training (e.g., overall PSNR $30.5 \rightarrow 34.3$ dB), helping the model adapt from short to longer-video decoding. Full training details and results are in Appendix J.

Alternative reference injection strategy. We also compare against a ControlNet-style [59] alternative that adds residual features from a parallel reference encoder to the decoder at each stage. This approach lacks temporal reasoning and converges to lower quality than our attention-based design. Details and qualitative results are in Appendix K.

Reference injection stages. We evaluate stage-specific reference injection by retaining conditioning at only one decoder stage and fine-tuning each variant under the same settings. Injecting reference information at all three stages achieves the best reconstruction quality among the tested configurations, indicating that conditioning across multiple decoding scales provides complementary benefits. The stage-specific variants offer lower per-chunk decoding latency. Full results are provided in Appendix L.

4 Applications

4.1 Style transfer

To demonstrate the generality of our reference injection design, we apply REFDECODER to decode-time style transfer: by supplying a style image as the reference and training on OmniStyle150K [45], REFDECODER transfers the reference style to the input video at inference while preserving its structural content. Figure 5 shows qualitative results.

Table 3: **VBench generation eval results (12 dimensions) on the Wan 2.1 backbone at 17 and 81 frames.** Subj. / BG denote subject/background consistency; I2V Subj./BG are their I2V-specific variants. REFDECODER†: causal variant (Section 2). Best results per frame count in bold.

		Quality Dimensions						I2V Dimensions			Aggregate Scores		
Model	**Frames**	Subj.	BG	Motion	Dyna.	Aesth.	Imag.	I2V Subj.	I2V BG	Camera	Quality	I2V	Total
Wan 2.1 [40]	17	96.6	97.9	98.1	47.5	65.6	**70.1**	98.0	99.1	**19.3**	81.3	94.5	87.9
REFDECODER		**96.8**	**98.2**	**98.3**	**47.6**	**66.0**	**70.1**	**98.3**	**99.3**	19.1	**81.6**	**94.8**	**88.2**
Wan 2.1 [40]	81	94.9	**97.7**	98.2	**46.6**	**65.1**	70.2	98.0	99.3	28.9	80.8	95.0	87.9
REFDECODER†		**95.3**	97.5	**98.5**	46.3	64.4	**70.9**	**98.2**	**99.4**	**29.1**	**80.9**	**95.3**	**88.1**

Table 4: **81-frame generation on high-motion VBench** (highest-motion quartile of the original VBench set by RAFT optical flow; Wan 2.1 backbone). REFDECODER†: causal variant (Section 2).

Model	Quality↑	I2V↑	Total↑
Wan 2.1 [40]	81.93	93.22	87.57
REFDECODER† (Wan 2.1)	**82.10** (+0.17)	**93.56** (+0.35)	**87.83** (+0.26)

Table 5: **81-frame reconstruction on Inter4K-81 and its high-motion subset** (top quartile by RAFT motion score; Wan 2.1 backbone). REFDECODER†: causal variant (Section 2).

	All (500 videos)			High motion (125 videos)		
Model	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Wan 2.1 [40]	31.0	90.0	0.042	30.2	88.1	0.059
REFDECODER† (Wan 2.1)	**31.4** (+0.4)	**91.3** (+1.3)	**0.030** (−0.012)	**31.0** (+0.8)	**90.1** (+2.0)	**0.036** (−0.023)



Figure 5: **Qualitative results on the style transfer task.** The reconstructed images follow the reference style while preserving the structural content of the input images.

4.2 Video editing

First-frame-guided video editors take a source video together with an edited version of its first frame, and produce a video in which the user-specified edit is applied consistently across all frames while everything else is expected to stay identical to the source. In practice, however, the editing model often noticeably degrades the *non-edited* regions.

We experiment with video editing on LoRA-Edit [12], a first-frame-guided editor that fine-tunes the Wan 2.1 model with a mask-aware LoRA. We compare decoding the latents with the original Wan 2.1

Table 6: **Evaluation of REFDECODER on LoRA-Edit [12].** We decode the same latents with the original decoder vs. REFDECODER and score the non-edited region against the source video using a per-frame mask. Numbers in teal are gains of REFDECODER over the baseline.

Decoder	PSNR↑	SSIM↑	LPIPS↓
Wan 2.1 [40]	30.2	90.7	0.0477
REFDECODER (Wan 2.1)	**31.6** (+1.4)	**92.6** (+1.9)	**0.0297** (−0.018)

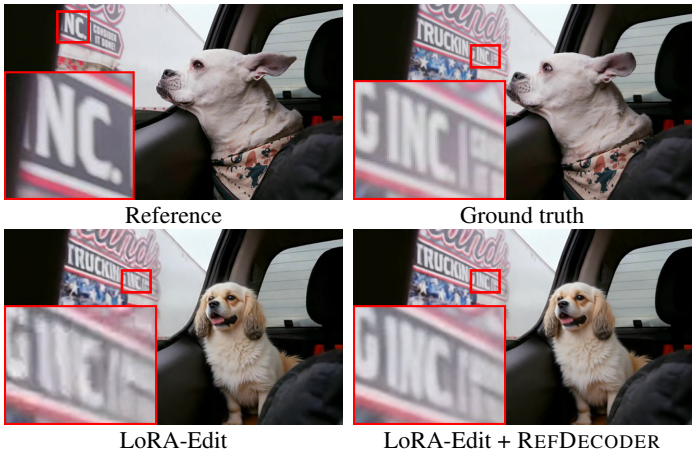


Figure 6: **Qualitative comparison on LoRA-Edit [12].** *Reference*: first frame of the source video. The baseline decoder distorts non-edited regions, while REFDECODER regenerates them faithfully from the reference, preserving high-quality details such as the text.

decoder vs. REFDECODER. We follow LoRA-Edit's pipeline to track the edited object across frames, and score masked PSNR, SSIM, and LPIPS over the non-edited region, averaged over all frames. As shown in Table 6, REFDECODER improves the non-edited region on every metric ($+1.4$ dB PSNR, $+1.9$ SSIM (%), -0.018 LPIPS). Figure 6 further shows that REFDECODER preserves high-quality details that the baseline decoder blurs out, despite significant movement from the reference frame.

5 Discussion

We introduced REFDECODER, a reference-conditioned video VAE decoder that addresses a fundamental asymmetry in conditional generation: while the diffusion backbone is richly conditioned, the decoder remains generally unconditional. By injecting reference image tokens via self-attention, REFDECODER serves as a lightweight, architecture-agnostic drop-in replacement for existing decoders. Consistent improvements across several backbones, including mainstream open-source foundation video models, on both reconstruction and generation benchmarks, together with its natural extension to style transfer and video editing, demonstrate the generality and task-agnostic nature of conditional decoding. We hope this work encourages the community to rethink the decoder's role in conditional generation.

Several limitations suggest future directions: extending from a single reference to multiple references (*e.g.*, for interpolation or multi-view tasks), exploring richer reference encoders beyond a single convolution layer, automating hyperparameter selection across architectures, and scaling beyond 81 frames, where the reference becomes more temporally distant. Because REFDECODER amplifies the perceptual fidelity of existing video generation models, it also inherits the same dual-use risks as the underlying model (*e.g.*, deepfake misuse).

Acknowledgments and Disclosure of Funding

This work was supported in part by the NSF through award IIS-2211133. The authors declare no additional competing interests.

References

[1] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *IEEE International Conference on Computer Vision*, 2021.

[2] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets, 2023.

[3] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models, 2023.

[4] Nicolas Bonneel, James Tompkin, Kalyan Sunkavalli, Deqing Sun, Sylvain Paris, and Hanspeter Pfister. Blind video temporal consistency. *ACM Transactions on Graphics (TOG)*, 34(6):1–9, 2015.

[5] Haoxin Chen, Yong Zhang, Xiaodong Cun, Menghan Xia, Xintao Wang, Chao Weng, and Ying Shan. Videocrafter2: Overcoming data limitations for high-quality video diffusion models, 2024.

[6] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets, 2019.

[7] Frederik Ebert, Chelsea Finn, Alex X. Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections, 2017.

[8] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models, 2023.

[9] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

[10] Xiang Fan, Anand Bhattad, and Ranjay Krishna. Videoshop: Localized semantic video editing with noise-extrapolated diffusion inversion, 2024.

[11] Xiang Fan, Xiaohang Sun, Kushan Thakkar, Zhu Liu, Vimal Bhat, Ranjay Krishna, and Xiang Hao. Reftok: Reference-based tokenization for video generation. *arXiv preprint arXiv:2507.02862*, 2025.

[12] Chenjian Gao, Lihe Ding, Xin Cai, Zhanpeng Huang, Zibin Wang, and Tianfan Xue. Lora-edit: Controllable first-frame-guided video editing via mask-aware lora fine-tuning. *arXiv preprint arXiv:2506.10082*, 2025.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[14] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[15] Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, Yaohui Wang, Xinyuan Chen, Ying-Cong Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench++: Comprehensive and versatile benchmark suite for video generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[17] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. Hunyuanvideo: A systematic framework for large video generative models, 2025.

[18] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In *ECCV*, 2018.

[19] Teng Li, Guangcong Zheng, Rui Jiang, Shuigen Zhan, Tao Wu, Yehao Lu, Yining Lin, and Xi Li. Realcam-i2v: Real-world image-to-video generation with interactive complex camera control. *arXiv preprint arXiv:2502.10059*, 2025.

[20] Zongjian Li, Bin Lin, Yang Ye, Liuhan Chen, Xinhua Cheng, Shenghai Yuan, and Li Yuan. Wf-vae: Enhancing video vae by wavelet-driven energy flow for latent video diffusion model, 2025.

[21] Willi Menapace, Aliaksandr Siarohin, Ivan Skorokhodov, Ekaterina Deyneka, Tsai-Shien Chen, Anil Kag, Yuwei Fang, Aleksei Stoliar, Elisa Ricci, Jian Ren, and Sergey Tulyakov. Snap video: Scaled spatiotemporal transformers for text-to-video synthesis, 2024.

[22] Kepan Nan, Rui Xie, Penghao Zhou, Tiehan Fan, Zhenheng Yang, Zhijie Chen, Xiang Li, Jian Yang, and Ying Tai. Openvid-1m: A large-scale high-quality dataset for text-to-video generation. *arXiv preprint arXiv:2407.02371*, 2024.

[23] NVIDIA. Cosmos tokenizer: A suite of image and video neural tokenizers, November 2024.

[24] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.

[25] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[27] Fitsum Red, Jinwei Gu, Xian Liu, Songwei Ge, Ting-Chun Wang, Haoxiang Wang, and Ming-Yu Liu. Cosmos tokenizer: A suite of image and video neural tokenizers, 2024.

[28] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[29] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data, 2022.

[30] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2, 2022.

[31] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

[32] Alexandros Stergiou and Ronald Poppe. Adapool: Exponential adaptive pooling for information-retaining downsampling, 2022.

[33] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024.

[34] Anni Tang, Tianyu He, Junliang Guo, Xinle Cheng, Li Song, and Jiang Bian. Vidtok: A versatile and open-source video tokenizer, 2024.

[35] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020.

[36] Rui Tian, Qi Dai, Jianmin Bao, Kai Qiu, Yifan Yang, Chong Luo, Zuxuan Wu, and Yu-Gang Jiang. REDUCIO! generating 1024×1024 video within 16 seconds using extremely compressed motion latents. In *ICCV*, 2025.

[37] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation, 2017.

[38] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *NeurIPS*, 2017.

[39] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022.

[40] 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.

[41] Junke Wang, Yi Jiang, Zehuan Yuan, Binyue Peng, Zuxuan Wu, and Yu-Gang Jiang. Omnitokenizer: A joint image-video tokenizer for visual generation, 2024.

[42] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. Cogvlm: Visual expert for pretrained language models, 2023.

[43] Wenhao Wang and Yi Yang. Tip-i2v: A million-scale real text and image prompt dataset for image-to-video generation. 2024.

[44] Xiang Wang, Hangjie Yuan, Shiwei Zhang, Dayou Chen, Jiuniu Wang, Yingya Zhang, Yujun Shen, Deli Zhao, and Jingren Zhou. Videocomposer: Compositional video synthesis with motion controllability, 2023.

[45] Ye Wang, Ruiqi Liu, Jiang Lin, Fei Liu, Zili Yi, Yilin Wang, and Rui Ma. Omnistyle: Filtering high quality style transfer data at scale. In *CVPR*, 2025.

[46] Pingyu Wu, Kai Zhu, Yu Liu, Liming Zhao, Wei Zhai, Yang Cao, and Zheng-Jun Zha. Improved video vae for latent video diffusion model, 2024.

[47] Yushu Wu, Yanyu Li, Ivan Skorokhodov, Anil Kag, Willi Menapace, Sharath Girish, Aliaksandr Siarohin, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. H3AE: High compression, high speed, and high quality autoencoder for video diffusion models. *arXiv preprint arXiv:2504.10567*, 2025.

[48] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Xintao Wang, Tien-Tsin Wong, and Ying Shan. Dynamicrafter: Animating open-domain images with video diffusion priors, 2023.

[49] Yazhou Xing, Yang Fei, Yingqing He, Jingye Chen, Jiaxin Xie, Xiaowei Chi, and Qifeng Chen. Videovae+: Large motion video autoencoding with cross-modal video vae. In *ICCV*, 2025.

[50] Jiaqi Xu, Xinyi Zou, Kunzhe Huang, Yunkuo Chen, Bo Liu, MengLi Cheng, Xing Shi, and Jun Huang. Easyanimate: A high-performance long video generation method based on transformer architecture, 2024.

[51] Wilson Yan, Matei Zaharia, Volodymyr Mnih, Pieter Abbeel, Aleksandra Faust, and Hao Liu. Elastictok: Adaptive tokenization for image and video, 2024.

[52] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Xiaotao Gu, Yuxuan Zhang, Weihan Wang, Yean Cheng, Ting Liu, Bin Xu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer, 2024.

[53] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models, 2023.

[54] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, et al. Magvit: Masked generative video transformer. In *CVPR*, 2023.

[55] Lijun Yu, José Lezama, Nitesh B. Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language model beats diffusion – tokenizer is key to visual generation, 2024.

[56] Sihyun Yu, Weili Nie, De-An Huang, Boyi Li, Jinwoo Shin, and Anima Anandkumar. Efficient video diffusion models via content-frame motion-latent decomposition. In *International Conference on Learning Representations*, 2024.

[57] Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin. Generating videos with dynamics-aware implicit generative adversarial networks, 2022.

[58] David Junhao Zhang, Jay Zhangjie Wu, Jia-Wei Liu, Rui Zhao, Lingmin Ran, Yuchao Gu, Difei Gao, and Mike Zheng Shou. Show-1: Marrying pixel and latent diffusion models for text-to-video generation, 2023.

[59] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[60] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[61] Sijie Zhao, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Muyao Niu, Xiaoyu Li, Wenbo Hu, and Ying Shan. Cv-vae: A compatible video vae for latent generative video models. *arXiv preprint arXiv:2405.20279*, 2024.

[62] Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, and Ziwei Liu. VBench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025.

[63] Guangcong Zheng, Teng Li, Rui Jiang, Yehao Lu, Tao Wu, and Xi Li. Cami2v: Camera-controlled image-to-video diffusion model. *arXiv preprint arXiv:2410.15957*, 2024.

A Related Work

Visual tokenization for generation. Variational autoencoders (VAEs) [16] learn latent-space probabilistic models for efficient representation. VQ-VAE [38] introduces a vector-quantized discrete latent space, and VQGAN [9] extends it to high-resolution images with adversarial training. Latent diffusion models (LDMs) [28] leverage these encoded latent spaces for high-quality image generation. For video, MAGVIT [54] employs a 3D tokenizer for efficient video encoding, and MAGVIT v2 [55] introduces a causal 3D CNN. ElasticTok [51] offers adaptive tokenization for long videos, and OmniTokenizer [41] proposes joint image-video tokenization. Recent continuous video autoencoders such as Cosmos Tokenizer [27] and VideoVAE+ [49] serve as the visual backbone for state-of-the-art video generation models.

Video generation. Early video generation relied on GANs [30, 37, 6, 57], but these are limited to short, low-resolution outputs. The field has since shifted to video diffusion models [2, 29, 3, 21, 5, 44, 58], which produce high-quality results at the cost of many denoising steps. Frontier open-source systems such as Wan 2.1 [40], HunyuanVideo [17], and CogVideoX [52] adopt the latent diffusion paradigm with powerful VAE backbones, achieving state-of-the-art quality. Image-to-video (I2V) models condition the generation process on a reference frame to produce temporally coherent video. Stable Video Diffusion [2] conditions a video diffusion model on a single image via concatenation in latent space. DynamiCrafter [48] animates open-domain images by injecting them as visual context into the diffusion backbone. IP-Adapter [53] introduces decoupled cross-attention to inject image prompt features into diffusion models, enabling image-guided generation with a lightweight adapter. A common thread across these approaches is that conditioning is applied exclusively to the *diffusion model*; the VAE decoder remains a purely unconditional reconstruction module and does not have access to the reference image.

Reference-conditioned decoding. Recent attempts to introduce reference conditioning into the VAE decoder have struggled to achieve full quality recovery using the reference signal. For example, RefTok [11] achieves improved reconstruction only at low resolutions using a quantized codebook and struggles to yield a high-quality video generation model; H3AE [47] instead focuses on high compression ratios, with imperfect reference information forming blurred reconstructions; Reducio-VAE [36] employs low-dimensional latent features without capturing high-fidelity reference signal. CMD [56] learns a video autoencoder that decomposes videos into a content frame and compact motion latents.

Prior methods exploring this space develop their own video autoencoder architectures and are tightly coupled to their respective systems. This further limits their quality and applicability, as their corresponding diffusion models are tied to the specific autoencoder representations. For example, compared to the 1.5 billion training videos of Wan 2.1, H3AE and Reducio are only trained on million-scale data sources. In contrast, REFDECODER is designed as a *plug-and-play* decoder module that can be integrated with existing, pretrained frontier video model VAE decoders without modifying the encoder or retraining the diffusion backbone, making it immediately deployable in production pipelines.

B Implementation Details

We apply REFDECODER to two pretrained video VAE backbones: Wan 2.1 [40] and VideoVAE+ [49]. Below we describe the architecture and training details using Wan 2.1 and VideoVAE+ as backbones.

B.1 Model Architecture

The Wan 2.1 decoder operates in three stages at progressively higher spatial resolutions, with channel dimensions of 384, 192, and 192. At each stage, spatio-temporal feature tokens are extracted using 3D patch embeddings with temporal stride 1 and spatial strides of 2×2, 4×4, and 8×8, respectively.

For comparison, VideoVAE+ also adopts a three-stage decoder but with channel dimensions of 512, 256, and 128. Spatio-temporal tokens are extracted via 3D patch embeddings with temporal stride 1 and spatial strides of 8×8, 16×16, and 16×16, respectively.

For both backbones, Transformer blocks are shared across all three stages. Each uses $H{=}12$ heads with head dimension $d_h{=}128$, resulting in a total hidden dimension of $d{=}1536$. We adopt Rotary

Position Embeddings (RoPE) [33] for positional encoding and GELU activations in the feed-forward layers.

The reference image encoder is implemented as a lightweight tokenizer consisting of a single strided 3D convolution layer (patch size 8, Xavier-uniform initialization) that maps the reference frame into tokens matching the channel dimension of the first decoder stage (384 for Wan 2.1 and 512 for VideoVAE+). This is followed by normalization (RMSNorm for Wan 2.1 and GroupNorm for VideoVAE+). During training, the reference frame is sampled uniformly at random from the input video sequence.

B.2 Training Setup

We train on 8 NVIDIA H200 GPUs with a per-GPU batch size of 2 (effective batch size 16) for 5 days. Model parameters are divided into two groups: (1) the attention block and reference image encoder, optimized at a base learning rate of 2×10^{-4} (effective learning rate 1.6×10^{-3} after linear scaling); and (2) the pretrained VAE decoder, fine-tuned at 0.1 of the effective rate. Both groups use AdamW with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

The learning rate schedule consists of a linear warmup from 1% to 100% of the base rate over the first 1,000 steps, followed by cosine annealing over 100,000 total steps. The encoder, quantization convolutions, and post-quantization convolutions remain frozen throughout; only the decoder and attention block are updated.

B.3 Loss Function

The training objective combines pixel-wise and perceptual losses:

$$\mathcal{L} = \|\mathbf{x} - \hat{\mathbf{x}}\|_1 + \mathcal{L}_{\text{LPIPS}}(\mathbf{x}, \hat{\mathbf{x}})$$

where $\mathbf{x}$ and $\hat{\mathbf{x}}$ denote the ground-truth and reconstructed video frames, and $\mathcal{L}_{\text{LPIPS}}$ is the LPIPS perceptual loss [60]. Since the encoder is frozen, KL divergence regularization is disabled.

B.4 Data and Evaluation

For Wan 2.1, training videos are resized to 480×832 pixels, with 5 frames per sample in the first training stage and 17 frames in the second stage. For VideoVAE+, training videos are resized to 216×216 pixels, with 4 frames per sample in the first stage and 16 frames in the second stage. We evaluate reconstruction quality using PSNR, SSIM, and LPIPS [60].

B.5 Reconstruction Benchmarks

We use three reconstruction benchmarks. Across all of them we use the exact video lists provided by VideoVAE+ [49] so that our numbers are directly comparable with those reported in [49].

- **Inter4K (test).** The test split of Inter4K [32], 500 high-quality videos.
- **WebVid.** An in-the-wild subset of WebVid [1].
- **Large Motion.** 100 videos (80 from WebVid [1], 20 from Inter4K [32]) manually curated to exhibit complex motion dynamics—significant camera motion, fast-moving subjects, and large inter-frame displacements.

For all three benchmarks, videos are decoded at the same resolution and frame count as the corresponding training configuration of each backbone, and PSNR / SSIM / LPIPS are averaged over all frames.

B.6 Per-Category Reconstruction on Inter4K

Table 7 reports per-category PSNR on Inter4K (Wan 2.1 backbone). Gains are positive across every category. They are largest on *content-rich* scenes with strong high-frequency structure (urban streets, neon night, driving POVs, aerial skylines, indoor venues, sports, underwater wildlife, and nature landscapes), and smallest on *content-sparse* scenes that are already smooth or close-up (animation

graphics, performance events, close-up macros, and people portraits), where the latent bottleneck loses comparatively less detail and the baseline already reconstructs faithfully. This pattern supports the intuition that the reference signal contributes most where the latent code is least sufficient.

Categorization protocol. To group the Inter4K test clips into 12 semantic categories without manual labeling, we use zero-shot CLIP [26].

Visual features. For each clip we sample three frames at evenly-spaced positions. Each frame embedding is L_2-normalized, and we mean-pool the three to obtain a single 768-d clip embedding v_i.

Category prompts. We define 12 categories and write 3–4 short natural-language prompts per category (42 prompts in total), encode each with CLIP's text tower, and L_2-normalize. Example prompts include "a busy city street with cars and buildings" (urban street), "neon signs and billboards at night in a city" (neon night), "first-person view of driving down a road" (driving pov), and "an aerial drone view of a city" (skyline aerial); the remaining categories are indoor venue, sports action, nature landscape, close up macro, people portrait, underwater wildlife, performance event, and animation graphics. The full prompt list is included in the released code.

Category score and assignment. For clip i and category c with prompt embeddings $\{t_{c,k}\}$, the score is the mean cosine similarity over prompts in that category, $s_{i,c} = \frac{1}{|c|} \sum_k v_i^\top t_{c,k}$, and each clip is assigned to a single category by $\arg\max_c \, s_{i,c}$. Per-category counts are reported in Table 7.

Table 7: Per-category reconstruction PSNR on Inter4K (Wan 2.1 backbone). n denotes the number of videos per category; Δ is the PSNR gain of REFDECODER over the Wan 2.1 baseline. Categories are grouped into content-rich (top) and content-sparse (bottom) scenes.

Category	n	**Wan 2.1**	**REFDECODER**	**Δ PSNR**
Content-rich scenes				
Urban street	43	29.0	31.4	**+2.4**
Neon night	34	31.6	33.6	**+2.1**
Driving POV	36	31.3	33.4	**+2.1**
Skyline aerial	48	31.3	33.3	**+1.9**
Indoor venue (arena/mall)	6	30.8	32.6	+1.8
Sports action	29	32.7	34.2	+1.6
Underwater wildlife	37	32.8	34.3	+1.4
Nature landscape	14	32.3	33.7	+1.4
Content-sparse scenes				
Animation graphics (CGI / fluid sim)	164	33.8	35.0	+1.2
Performance event (concert/dance)	45	34.2	35.2	+1.0
Close-up macro	23	37.4	38.2	+0.8
People portrait	18	38.5	39.2	+0.7

B.7 VBench Evaluation Protocol

Since REFDECODER is a drop-in decoder, our VBench [14, 15, 62] evaluation follows a *fixed-seed* protocol that controls for diffusion-side stochasticity. We describe the protocol here so that any per-prompt difference reflects the choice of decoder rather than differences in noise or sampling.

Fixed latent generation. For each prompt in the VBench info list we generate $K = 5$ samples, matching the default VBench-evaluation. For each (prompt, sample-index) pair we draw a 32-bit seed once via `random.randint(0, 2^32 - 1)`, persist it to a per-GPU JSON log, and use it to instantiate a generator. The Wan 2.1 pipeline is then run with that generator at 50 inference steps, classifier-free guidance scale 5.0, 17 frames, and the same 16:9 480p reference image used by the official VBench benchmark. The pipeline is invoked with `output_type="latent"`, so the noised input, denoising trajectory, and final latents are fully determined by the seed. The latents and the seed used to produce them are saved together to disk.

Decoder swap. The same set of saved latents is then decoded twice—once with the original Wan 2.1 VAE decoder and once with our REFDECODER –and both decoded videos are scored with the official VBench-I2V evaluation pipeline. Because both decoders consume exactly the same latents produced

Table 8: Temporal-stability metrics on VBench (Wan 2.1 backbone, fixed-seed protocol). E_{warp}, E_{tLPIPS}, and E_{flicker} measure temporal degradation (lower is better); $\text{CLIP}_{\text{cons}}$ measures temporal consistency (higher is better). REFDECODER reduces flickering and warping error across all degradation metrics. Δ is the relative change vs. Wan 2.1.

Model	$E_{\text{warp}} \downarrow$	$E_{\text{tLPIPS}} \downarrow$	$E_{\text{flicker}} \downarrow$	$\text{CLIP}_{\text{cons}} \uparrow$
Wan 2.1 [40]	0.03274	0.09136	0.02727	0.99526
REFDECODER (Wan 2.1)	**0.02941**	**0.08367**	**0.02303**	**0.99588**
	(−10.2%)	(−8.4%)	(−15.5%)	(+0.06%)

Table 9: **Reconstruction under scene-boundary reference mismatch.** REFDECODER improves PSNR and SSIM, while LPIPS increases.

Model	PSNR ↑	SSIM ↑	LPIPS ↓
Wan 2.1	31.45	0.905	**0.039**
REFDECODER	**31.82**	**0.917**	0.045

from exactly the same seeds, the per-prompt comparison is paired: differences in VBench dimensions can only arise from the decoder, not from a different noise sample, prompt order, or guidance schedule. This is why we report the comparison without an explicit per-seed standard deviation; sample-level variance from the diffusion sampler is shared across the two methods.

Reproducibility. The seed log files are kept alongside the saved latents, so any third party with access to the Wan 2.1 checkpoint can reproduce both the latents and the decoded videos bit-for-bit. The decoding scripts and the latent-generation script will be released together with the code.

C Temporal Stability

Table 8 reports the temporal-stability metrics summarized in the main paper: flow warping error (E_{warp}) [18], temporal LPIPS (E_{tLPIPS}) [60], flicker (E_{flicker}) [4], and CLIP consistency ($\text{CLIP}_{\text{cons}}$) [8]. The first three measure temporal degradation (lower is better) and the last consistency (higher is better). All metrics are computed on the VBench generations under the fixed-seed protocol of Sec. B.7, so any per-prompt difference reflects the choice of decoder rather than diffusion-side variance.

D Robustness to Reference Mismatch

D.1 Scene-Boundary Stress Test

We construct a stress test using 100 Inter4K videos containing scene-boundary cuts. We place the boundary near the middle of each evaluated clip and randomly select the reference frame, excluding the boundary frame. The reference and evaluated frames may therefore come from different shots with substantially different backgrounds.

Table 9 shows that REFDECODER improves PSNR by 0.37 dB and SSIM by 0.012, while LPIPS only increases by 0.006. Thus, pixel-level and structural improvements persist in this reference-mismatch setting.

D.2 Performance versus Reference Distance

We examine reconstruction quality as a function of temporal distance from the reference on 81-frame clips from 500 randomly sampled Inter4K videos. The source videos are recorded at 60 fps and sampled with temporal stride 4, yielding an effective frame rate of 15 fps. For each clip, we randomly select one reference frame and measure per-frame PSNR as a function of the absolute frame-index difference from the reference.

At a distance of 10 sampled frames from the reference, PSNR increases from approximately 31.1 to 31.5 dB. This improvement indicates that the benefit is not confined to the reference frame or its immediate neighbors.

E Comparison with H3AE

We provide a qualitative head-to-head comparison with H3AE [47], a prior reference-conditioned VAE. Since H3AE has not released training or evaluation code, training data, or model weights, we cannot run it on our benchmark splits; instead, we take the examples directly from the figures in the H3AE paper and run REFDECODER on the same input frames so that both methods are evaluated under identical inputs. As shown in Figure 7, REFDECODER preserves fine structures and textures (*e.g.*, the lizard's dots and the spectator faces) noticeably better than H3AE.

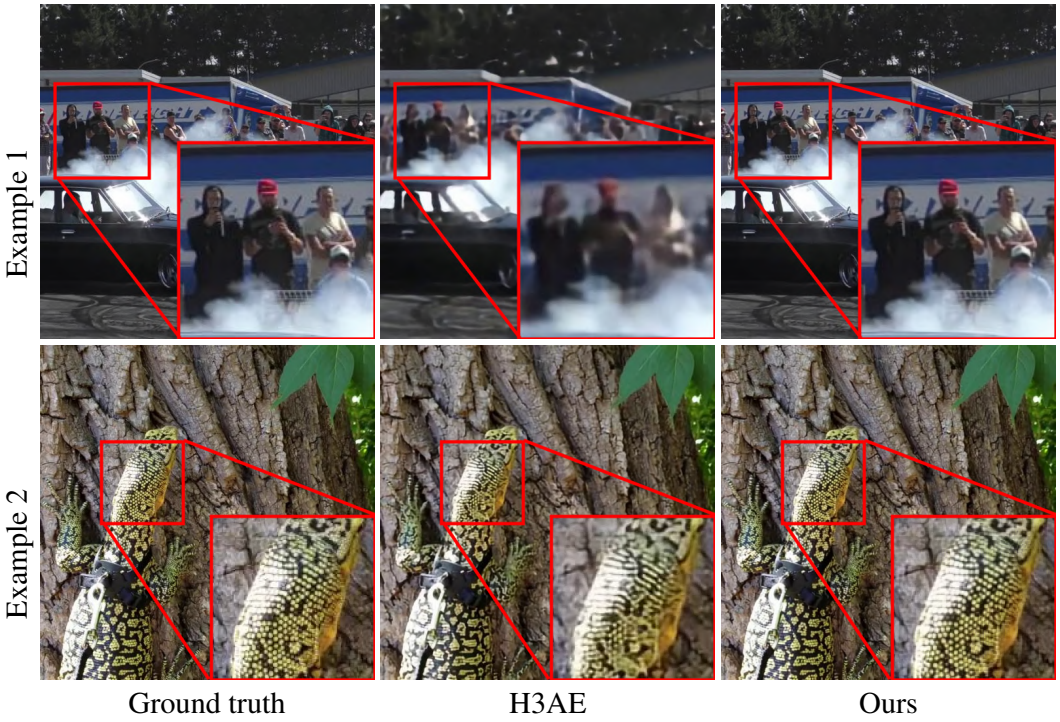


Figure 7: **Qualitative comparison with H3AE.** The ground-truth and H3AE crops are taken directly from the figures in the H3AE paper, since H3AE's training data, code, and weights are not publicly released. REFDECODER reconstructions are produced by us on the *same* input frames so that both methods are evaluated under identical inputs. REFDECODER recovers fine textures and structural details (lizard dots, background people) more faithfully than H3AE.

F Human Evaluation

Interface. Evaluations are collected through a custom web tool. Each pair is rendered as a single slider-overlay: the two videos are stacked and synchronously played, and the evaluator drags a vertical divider to reveal one model on the left half and the other on the right half. Left/right assignment is randomized per pair (and cached so reloads do not reshuffle), and model identities are hidden until after voting. The evaluator clicks one of two buttons, "Left wins" or "Right wins", to register a preference. For a tie, the evaluator leaves both options unselected.

Evaluation protocol and votes. Evaluators are instructed to view and rate a random subset of the pairs assigned to a given baseline based on overall visual quality. All participants took part voluntarily and received no compensation. We collect 252 evaluation records in total, including ties: 102 for Wan 2.1 (89 generation, 13 reconstruction) and 50 each for VideoVAE+, HunyuanVAE, and Reducio-VAE (all reconstruction). Evaluators leave both preference options unselected when

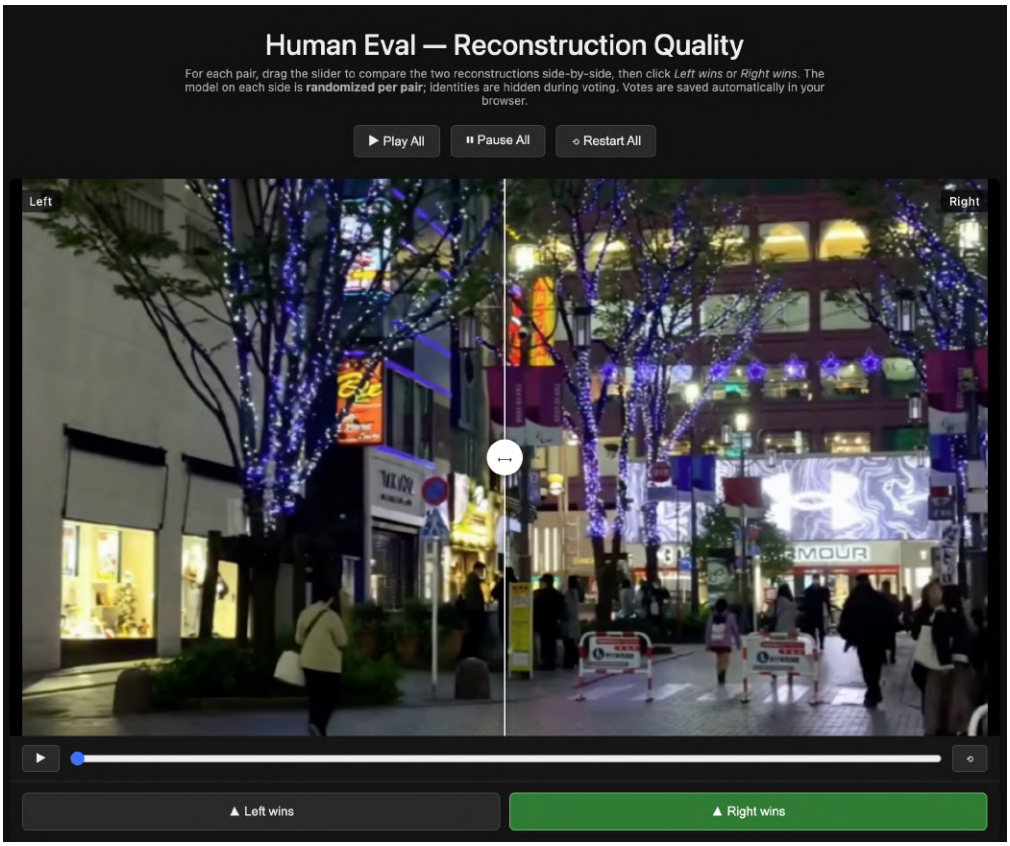


Figure 8: **Human-evaluation interface.** Each pair of videos is rendered as a slider-overlay: the two videos are stacked and synchronously played, and the evaluator drags a vertical divider to reveal one decoder's output on the left half and the other on the right half. Left/right assignment is randomized per pair and cached across reloads, and model identities are hidden until after voting. The evaluator registers a preference using the "Left wins" / "Right wins" buttons. Preference rates are normalized over non-tied votes; ties are not shown.

Table 10: Human evaluation. Evaluators compare the visual quality of REFDECODER against four baselines on video pairs sampled from Inter4K reconstructions and VBench generations, presented via a slider-overlay interface with randomized left/right order and hidden model identities. The **blue** segment is preference for REFDECODER and the gray tail is preference for the baseline; the two numbers next to each bar show REFDECODER's win percentage versus the baseline.

REFDECODER (Ours) vs. ...	Preference (Ours % vs. Baseline %)
Wan 2.1 Generation [40]	82.4% vs. 17.6%
Wan 2.1 Reconstruction [40]	92.3% vs. 7.7%
VideoVAE+ [49]	63.4% vs. 36.6%
HunyuanVAE [17]	69.2% vs. 30.8%
Reducio-VAE [36]	89.6% vs. 10.4%

they judge the two videos to be tied. These ties are included in the total count but excluded when computing preference rates.

Results. Table 10 summarizes the outcomes; REFDECODER is consistently preferred over every baseline: 92.3% vs. 7.7% on Wan 2.1 reconstruction, 82.4% vs. 17.6% on Wan 2.1 generation, 89.6% vs. 10.4% vs. Reducio-VAE, 69.2% vs. 30.8% vs. HunyuanVAE, and 63.4% vs. 36.6% vs. VideoVAE+. These human preferences are consistent with the quantitative reconstruction (Table 1) and VBench (Table 3) results, as well as the temporal-stability results in Table 8.

Table 11: VBench evaluation results across 12 evaluation metrics. We compare the Wan 2.1 baseline with REFDECODER under different training settings, including latent fine-tuning and bidirectional training. Our chosen strategies improve performance across most dimensions and lead to higher aggregate scores.

	Quality Dimensions						I2V Dimensions			Aggregate Scores		
Model	**Subj.**	**BG**	**Motion**	**Dynamic**	**Aesth.**	**Imaging**	**I2V Subj.**	**I2V BG**	**Camera**	**Quality**	**I2V**	**Total**
Wan 2.1 [40]	0.9656	0.9788	0.9814	0.4748	0.6557	0.7011	0.9796	0.9912	0.1934	0.8127	0.9446	0.8786
REFDECODER (w/o FT)	0.9675	0.9820	0.9828	**0.4764**	**0.6598**	0.7006	0.9830	0.9930	0.1908	0.8156	0.9475	0.8815
REFDECODER (Forward)	0.9691	0.9858	0.9829	0.4732	0.6582	**0.7027**	0.9846	0.9949	0.1927	**0.8167**	0.9497	0.8832
REFDECODER (Bidirectional)	**0.9693**	**0.9859**	**0.9833**	0.4724	0.6586	0.7010	**0.9847**	**0.9950**	**0.1937**	0.8167	**0.9499**	**0.8833**

G Latent Training

To further improve the performance of REFDECODER on the generation task, we additionally fine-tune the model using latent data. Specifically, we collect 30k images from TIP-I2V [43] and re-caption all images using CogVLM [42]. We then use the Wan 2.1 model [40] to generate corresponding latent video frames.

During training, we use 30k videos from the original video dataset together with 30k latent videos generated by diffusion models. The latent videos correspond to the output of the diffusion model rather than the output of the VAE encoder. As in the main training stage, the encoder remains frozen and only the decoder is optimized. Training alternates between real videos and latent videos in successive batches.

For real video data, the reconstruction loss is computed between the input video and the reconstructed video. For latent video data, the reconstruction loss is applied only to the first frame by comparing the reconstructed first frame with the input reference frame. Reconstruction loss is not applied to the remaining frames.

In addition, we aim to train the model with latent sequences where the reference frame is not always the first frame, encouraging the model to learn longer temporal dependencies. Empirically, we find that when training only with forward sequences, early frames tend to preserve the reference appearance, while later frames may gradually deviate from it. To alleviate this issue, we construct reversed training sequences so that later frames are encouraged to attend to the reference frame.

Since a single latent video frame encodes multiple video frames, we cannot directly reverse the latent sequence. Instead, we first decode the latent sequence using the original Wan 2.1 decoder to obtain the video frames, reverse their temporal order, and then pass the reversed video through the original Wan 2.1 encoder to obtain the corresponding latent representation used for training.

This training also consists of two stages. In Forward Training, latent sequences are used in the original temporal order. In Bidirectional Training, each batch randomly uses either the forward or reversed sequence with equal probability.

Table 11 presents VBench evaluation results on the Wan 2.1 backbone. Introducing REFDECODER already improves most evaluation dimensions compared to the original Wan 2.1 model. Applying latent fine-tuning further improves the performance, indicating that training on diffusion-generated latent videos helps the decoder better adapt to latent-space generation. Finally, the proposed bidirectional training strategy achieves the best overall performance, suggesting that exposing the model to reversed temporal sequences improves reference consistency across frames.

H Random Reference Frame

We further study how the choice of reference frame during training affects model performance. Specifically, we train two models on Wan 2.1 backbone with different reference-frame strategies: one uses the first frame as the reference during training, while the other randomly samples a reference frame from the video. Both models use Wan 2.1 as the backbone with 5 Transformer blocks and a dropout rate of 0.7, and are evaluated on the 17 frames 480×832 Inter4K test set.

Table 12: Effect of reference-frame selection during training and evaluation. Models are trained using either the first frame or randomly sampled frames as the reference. Each model is evaluated using both first-frame and random-frame references. Training with random references leads to more robust performance across evaluation settings.

Training Ref.	Eval Ref.	Overall			First Frame		
		PSNR ↑	SSIM (%) ↑	LPIPS ↓	PSNR ↑	SSIM (%) ↑	LPIPS ↓
First frame	First frame	34.1	94.06	0.0371	**40.8**	**98.57**	**0.0046**
First frame	Random frame	31.9	91.06	0.0556	22.9	66.44	0.2194
Random frame	First frame	34.2	94.14	0.0371	40.0	98.24	0.0073
Random frame	Random frame	**34.6**	**94.64**	**0.0337**	37.2	96.82	0.0215

Table 13: Effect of the number of Transformer blocks on reconstruction (Wan 2.1). Increasing number of blocks consistently improves reconstruction quality, with the 10-blocks model achieving the best results.

	Inter4K (test)			**WebVid**			**Large Motion**		
# Blocks	**PSNR↑**	**SSIM (%)↑**	**LPIPS↓**	**PSNR↑**	**SSIM (%)↑**	**LPIPS↓**	**PSNR↑**	**SSIM (%)↑**	**LPIPS↓**
3	34.3	94.4	0.035	33.2	92.3	0.038	31.1	89.9	0.055
5	34.6	94.6	0.034	33.4	92.6	0.038	31.3	90.2	0.055
7	34.5	94.7	0.032	33.3	92.3	0.037	31.2	90.1	0.054
10	**34.9**	**94.9**	**0.031**	**33.5**	**92.6**	**0.037**	**31.4**	**90.4**	**0.053**

During evaluation, we test both models under two settings: using the first frame or a randomly sampled frame as the reference. This allows us to analyze how each training strategy generalizes to different reference-frame conditions.

Table 12 reports the quantitative results. When the model is trained using the first frame as the reference, it achieves the best reconstruction quality when evaluated with the same setting (first-frame reference). However, its performance drops significantly when evaluated with a randomly selected reference frame, indicating that the model overfits to the training reference strategy and generalizes poorly to other reference conditions. In particular, we observe that the model degenerates when the reference frame is fixed to frame 0, becoming overly dependent on the reference appearance.

In contrast, training with randomly sampled reference frames leads to more robust performance across evaluation settings. The model trained with random references achieves the best overall performance when evaluated with random references and remains competitive when evaluated with the first frame as the reference. Notably, even under the first-frame evaluation setting, random-reference training still outperforms training that always uses the first frame as the reference. These results suggest that random reference training improves the model's ability to adapt to varying reference-frame conditions during inference.

I Effect of the Number of Transformer blocks

Table 13 reports the block-count ablation summarized in the main paper. We train the Wan 2.1 backbone with $\{3, 5, 7, 10\}$ Transformer blocks under the same two-stage curriculum and evaluate on all three reconstruction benchmarks. Reconstruction quality improves consistently with depth across PSNR, SSIM, and LPIPS, and the 10-blocks model attains the best results on every benchmark.

J Latent Token Dropout and Two-Stage Curriculum

Setup. Table 14 reports the two training-time ablations summarized in the main paper. Both use the Wan 2.1 backbone with 3 Transformer blocks and are evaluated on Inter4K. The top block sweeps the maximum latent-token dropout probability $(0.0, 0.3, 0.7)$, with each variant trained for $10{,}000$ steps under the two-stage curriculum. The bottom block compares one-stage training (the first curriculum stage only) against the full two-stage curriculum at fixed dropout 0.7. We additionally report metrics

Table 14: Ablations on the Wan 2.1 backbone with 3 Transformer blocks, evaluated on Inter4K. *Top*: effect of maximum latent token dropout probability (with two-stage curriculum). *Bottom*: effect of two-stage curriculum training (with dropout 0.7).

		Overall			Reference Frame		
	Setting	PSNR ↑	SSIM (%) ↑	LPIPS ↓	PSNR ↑	SSIM (%) ↑	LPIPS ↓
Dropout	0.0	27.4	88.3	0.083	27.3	88.6	0.082
	0.3	27.5	86.7	0.077	27.6	87.1	0.075
	0.7	**30.1**	**90.1**	**0.060**	**31.1**	**92.1**	**0.054**
Curriculum	One-stage	30.5	89.7	0.058	34.9	95.7	0.018
	Two-stage	**34.3**	**94.4**	**0.035**	**39.6**	**98.2**	**0.008**

over both the entire reconstructed video (*Overall*) and the reference frame alone (*Reference Frame*) to show whether gains come primarily from regenerating the reference or from propagating reference information to the rest of the video.

Latent token dropout. Increasing the maximum dropout from 0.0 to 0.7 improves overall PSNR by $+2.71$ dB and the reference-frame PSNR by $+3.85$ dB. The reference-frame gain is larger than the overall gain, indicating that with dropout the decoder learns to lean on the reference token when the corresponding latent is missing rather than producing degraded content. Qualitative reconstructions in Figure 9 show the same trend: at dropout 0.7, reconstructions exhibit sharper edges and more faithful textures, while dropout 0.0 produces blurrier results.

Two-stage curriculum. At fixed dropout 0.7, the two-stage curriculum improves overall PSNR by $+3.80$ dB and reference-frame PSNR by $+4.64$ dB compared to one-stage training. The much larger gap on reference-frame PSNR ($+4.64$ dB vs. $+3.80$ dB overall) suggests that the second stage primarily refines reference-token decoding to longer temporal contexts, supporting the choice to first train on short 5-frame clips and then fine-tune on 17-frame sequences.



GT Dropout 0.0 Dropout 0.3 Dropout 0.7

Figure 9: **Qualitative comparison of different dropout rates** on Wan 2.1. Higher dropout encourages the decoder to rely more on reference features, leading to sharper and more detailed reconstructions.

K Alternative Reference Injection Strategy

We compare our attention-based reference injection with a ControlNet-style [59] alternative that injects the reference image through a parallel encoder branch, adding residual features to the decoder at each stage. We identify two fundamental limitations of this approach:

1. *No temporal reasoning.* ControlNet operates via spatial addition of residual features, applying the reference signal identically to every frame at the same spatial position. Unlike joint attention, it lacks a mechanism to modulate the reference contribution based on temporal context. The decoder cannot distinguish whether a frame is temporally close to or far from the reference, nor can it selectively attend to different reference regions for different frames.
2. *Suboptimal convergence.* The ControlNet variant converges quickly to a suboptimal reconstruction quality and plateaus. We hypothesize that the spatially rigid injection forces the model into a local minimum where it learns to uniformly blend the reference signal rather than adaptively retrieving fine-grained details.



Figure 10: **Qualitative comparison of alternative reference injection methods.** Our attention-based reference injection produces sharper and more temporally consistent reconstructions, while the ControlNet-style approach introduces ghosting artifacts and temporal inconsistency.

Table 15: Reconstruction quality and per-chunk decoding latency for different reference injection stages. "None" denotes the original Wan 2.1 decoder. Best reconstruction results are shown in bold.

Injection stages	Latency/chunk ↓	PSNR ↑	SSIM ↑	LPIPS ↓
None (Wan 2.1)	239.2 ms	–	–	–
0	264.1 ms	31.5272	0.9093	0.0574
1	279.9 ms	31.6146	0.9103	0.0582
2	329.3 ms	32.4959	0.9223	0.0508
0, 1, 2 (ours)	393.5 ms	**33.2817**	**0.9328**	**0.0414**

L Stage-Specific Reference Injection

To evaluate the effect of reference injection at different decoder stages, we construct stage-specific variants of RefDecoder. Starting from the same pretrained RefDecoder checkpoint, we keep reference injection active at only one decoder stage and disable it at all remaining stages. We then fine-tune each stage-specific model under the same training settings. This setup allows us to compare the reconstruction quality and decoding cost of injection at individual stages with those of the full model, which injects reference information at all three stages.

As shown in Table 15, reference injection at all three stages achieves the best reconstruction performance. Compared with injection at stage 2 alone, the full model improves PSNR by 0.7858 dB and SSIM by 0.0105, while reducing LPIPS by 0.0094 (18.5%). These results indicate that reference conditioning across multiple decoding scales provides complementary benefits.

M Code License

Our work is built upon the HuggingFace Diffusers [39] library, which is licensed under the Apache License 2.0 (`https://github.com/huggingface/diffusers/blob/main/LICENSE`). VideoVAE+-related code is licensed under Attribution-NonCommercial-NoDerivatives 4.0 International, per the VideoVAE+ [49] license (`https://github.com/VideoVerses/VideoVAEPlus/blob/main/LICENSE`).

N More Qualitative Results

We present more qualitative comparisons for both video reconstruction and image-to-video (I2V) generation.

Figure 11 shows reconstruction comparisons on the Wan 2.1 backbone. Each pair displays a cropped region from the same video frame, with the baseline decoder on the left and REFDECODER on the right. Figure 12 provides an analogous comparison on the VideoVAE+ backbone with cropped region pairs. In both cases, REFDECODER recovers finer textures, sharper edges, and more faithful structural details such as text and facial features.

Figure 13 shows video generation comparisons on the Wan 2.1 backbone. For each example, the ground-truth reference image is shown alongside three generated frames (first, middle, and last) from both the baseline model and REFDECODER. Our method better preserves scene structure and fine-grained appearance from the reference image while maintaining temporal consistency across frames.

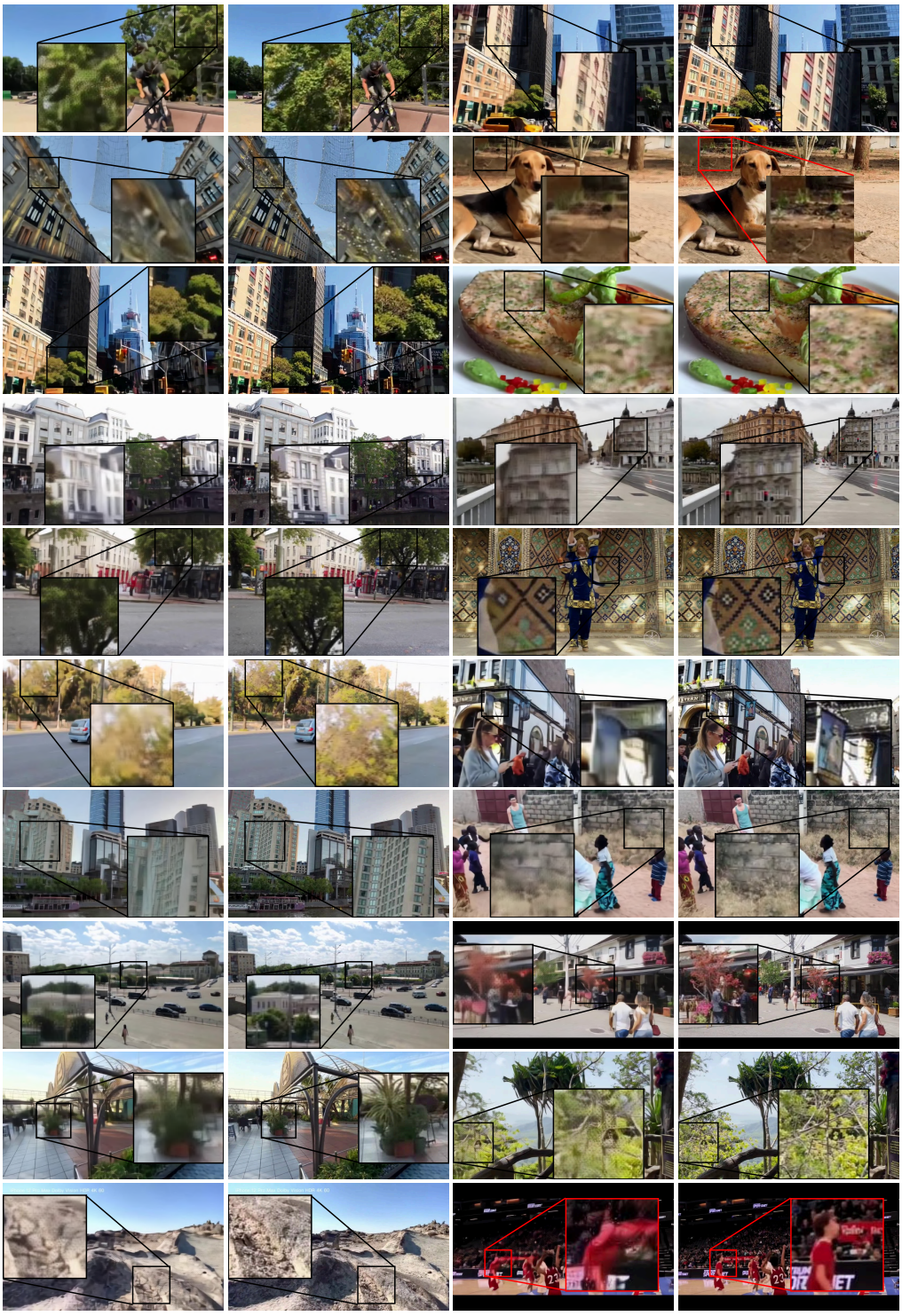


Figure 11: **Additional Wan 2.1 reconstruction comparisons.** Highlighted regions are shown for baseline vs. REFDECODER. Our method consistently recovers sharper textures and finer details across diverse scenes.

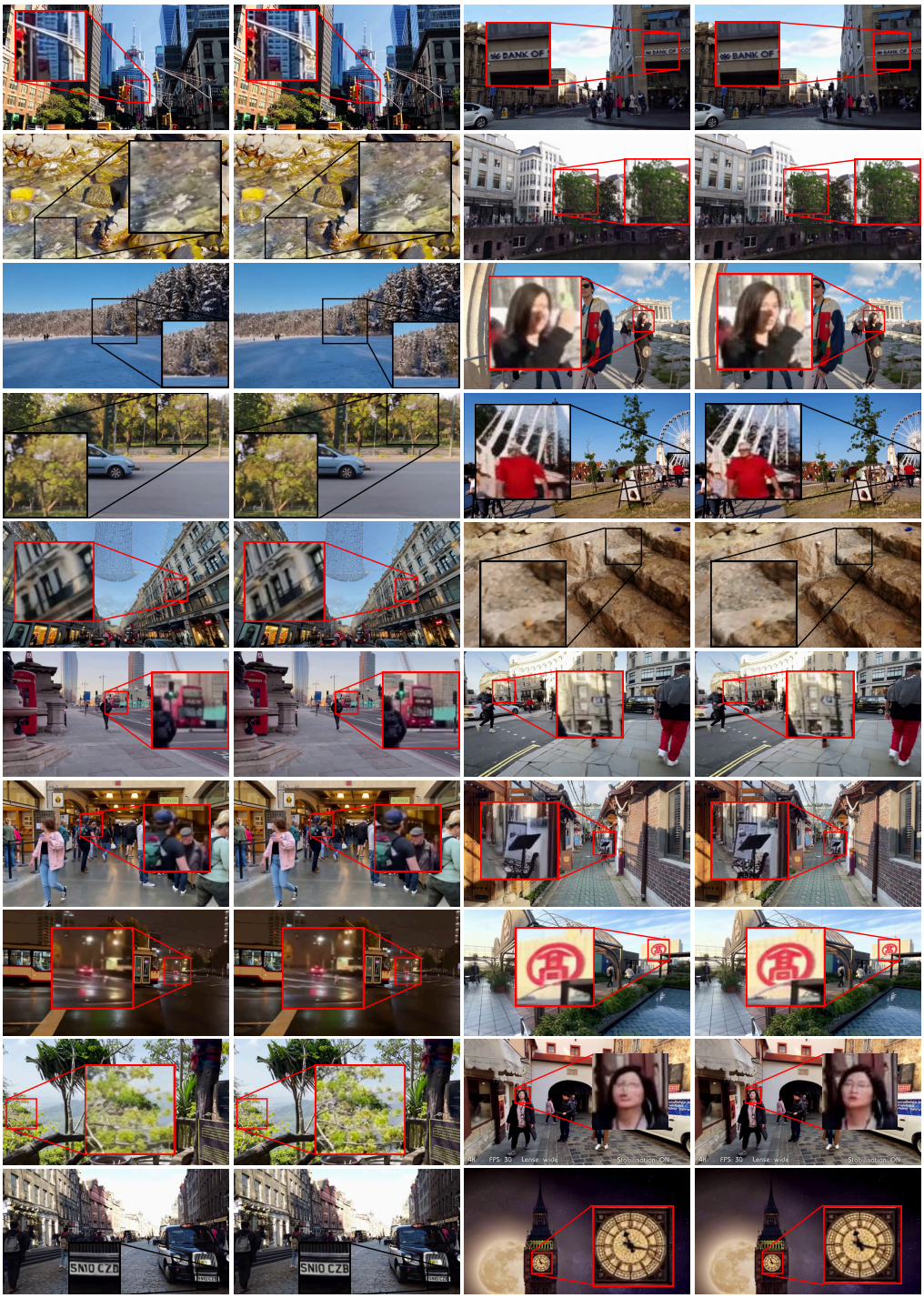


Figure 12: **Additional VideoVAE+ reconstruction comparisons.** Highlighted regions are shown for baseline vs. REFDECODER. Our method preserves fine-grained details such as text, facial features, and structural patterns more faithfully.

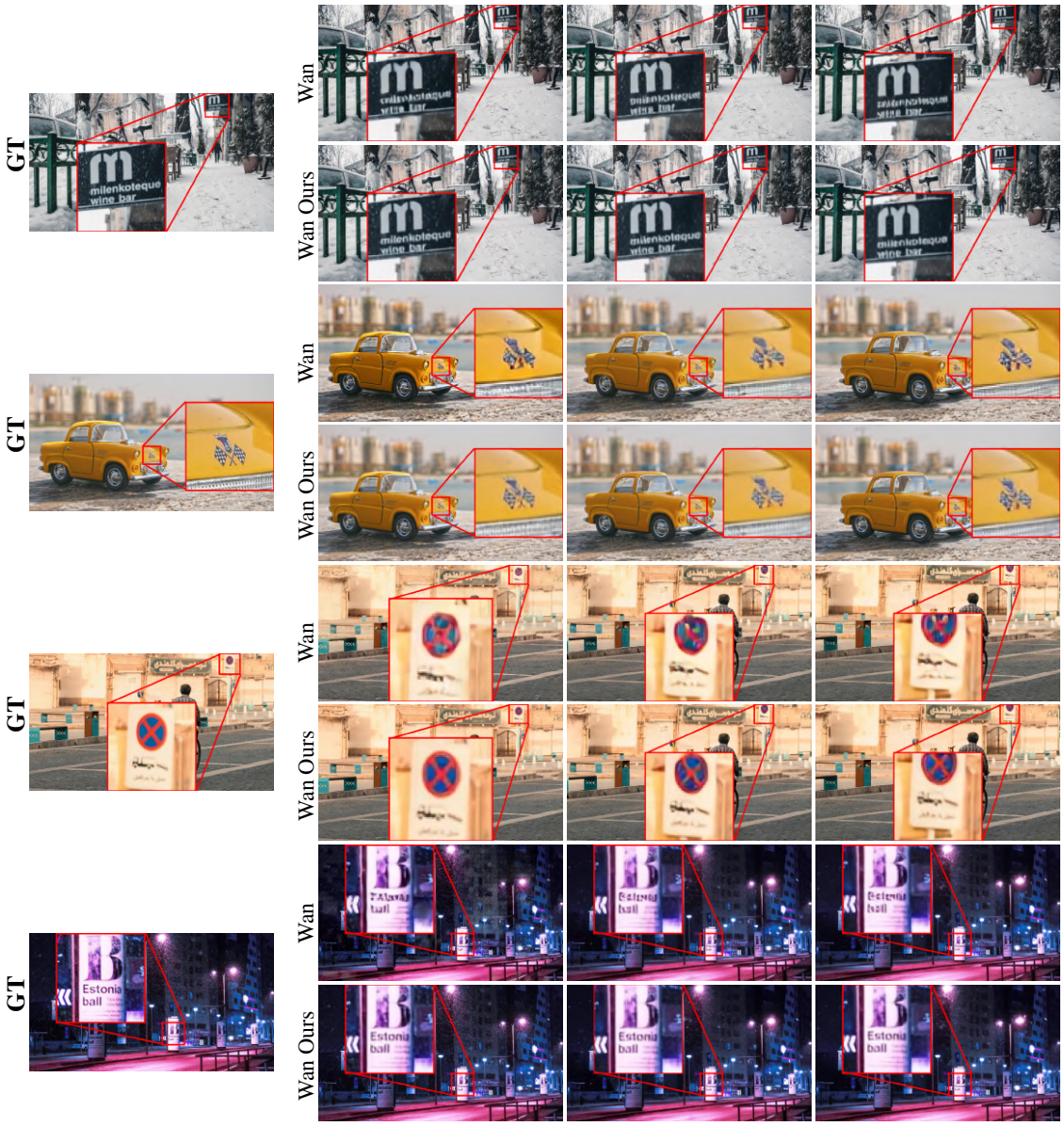


Figure 13: **Additional video generation comparisons.** For each example, the GT reference image and three generated frames at different timesteps are shown. REFDECODER preserves scene structure and produces sharper, more consistent details across frames compared to the baseline.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction describe a reference-conditioned retrofit of pretrained video VAE decoders that preserves the encoder and latent interface and does not require retraining the diffusion backbone. These claims are supported by reconstruction and generation experiments in Sec. 3, with additional evaluation of the causal variant on 81-frame videos in Sec. 3.1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 5 discusses limitations of single-reference conditioning, the lightweight reference encoder, architecture-specific hyperparameter selection, and scaling beyond the evaluated 81-frame setting. The scene-boundary stress test in Sec. D.1 additionally shows that PSNR and SSIM improvements can coexist with worse LPIPS under reference mismatch.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper presents an empirical contribution and contains no formal theorems or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The architecture and training procedure are described in Sec. 2, Sec. 3, and the supplementary Implementation Details. Sec. B.7 describes the paired VBench evaluation, in which both decoders process the same saved latents and the seed for each prompt–sample pair is recorded. The original reconstruction benchmarks use the VideoVAE+ splits (Sec. B.5), while the independently sampled Inter4K-81 evaluation and the causal variant are described in Sec. 3.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code and instructions for reproducing the main experiments are available at `https://github.com/RefDecoder/RefDecoder`. The release provides the training and evaluation scripts and configurations described in the paper. Data sources and evaluation splits are described in Sec. B.5 and Sec. 3.1.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 3 and the supplementary Implementation Details describe the training data, optimizer, learning-rate schedule, batch size, training resolutions and frame counts, latent dropout, and parameter-freezing strategy. Sec. B.7 describes the paired generation protocol, and Sec. 3.1 describes the 81-frame evaluation and the additional 4K-step adaptation of the causal variant.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report confidence intervals or statistical significance tests for the main reconstruction and generation results. The VBench evaluation uses a paired protocol in which both decoders process identical latents, controlling for differences in diffusion sampling between methods. This pairing does not eliminate variability across prompts and samples.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Sec. B.2 reports the hardware, effective batch size, and training duration for the main training setup: eight NVIDIA H200 GPUs, an effective batch size of 16, and five days of training. The causal variant additionally undergoes 4K adaptation steps(about 12 hours), as described in Sec. 3.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: The work uses pretrained models and existing datasets and includes a human preference evaluation described in Sec. F. Potential misuse of improved video fidelity is discussed in Sec. 5.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper presents benefits for visual generation and editing and discusses potential misuse of improved video fidelity, including deepfake synthesis, in Sec. 5.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [No]

Justification: We do not introduce additional safeguards specific to RefDecoder. The method improves the decoding fidelity of existing video generation models and may inherit their misuse risks, including deepfake synthesis, as discussed in Sec. 5.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All backbones (Wan 2.1, VideoVAE+, HunyuanVideo, CogVLM) and benchmark datasets (Inter4K, WebVid, VBench-I2V, OpenVidHD-0.4M, MiraData, DL3DV, TIP-I2V) are cited in Sec. 3 and the apendix, and are used under their original public research licenses; we did not modify or relicense any existing asset. Our work is built upon the HuggingFace Diffusers [39] library, which is licensed under the Apache License 2.0 (`https://github.com/huggingface/diffusers/blob/main/LICENSE`). VideoVAE+-related code is licensed under Attribution-NonCommercial-NoDerivatives 4.0 International. The VideoVAE+ [49] license (`https://github.com/VideoVerses/VideoVAEPlus/blob/main/LICENSE`).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released code and model checkpoints are accompanied by documentation describing installation, model usage, training and evaluation configurations, licensing, and limitations.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [Yes]

Justification: Sec. F provides the evaluation instructions, interface, and preference-recording procedure, including the treatment of ties. All participants took part voluntarily and received no compensation.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [No]

Justification: We conducted a volunteer human preference evaluation but did not obtain IRB approval or equivalent institutional review for this study. Participants took part voluntarily and received no compensation.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method does not depend on an LLM. CogVLM [42] is used only as an off-the-shelf captioning tool to re-caption 30k images for the latent fine-tuning stage (Sec.G); this is a standard pre-processing step and is not an original or non-standard component of the method.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.