GENERATION-AWARE LATENT COMPRESSION FOR EFFICIENT VIDEO GENERATION

Anonymous authors
Paper under double-blind review

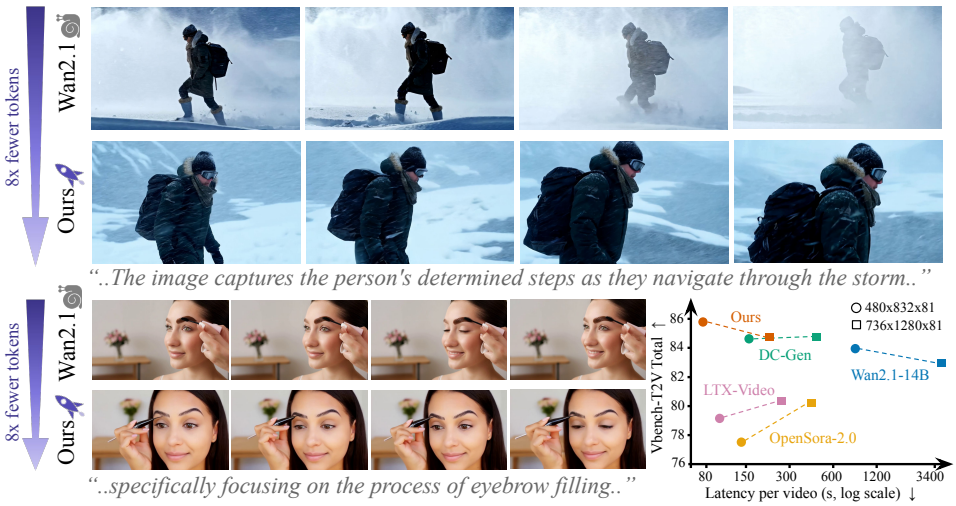


Figure 1: We fine-tune a pretrained video autoencoder and DiT to generate from $8\times$ fewer latent tokens. **(Top)** Text-to-video (T2V) samples from the pretrained Wan2.1-14B and ours at 736p, from the same prompt and seed. **(Bottom left)** The same comparison at 480p. **(Bottom right)** Ours matches the pretrained model on VBench while generating $11.1\times$ faster at 480p and $15.6\times$ faster at 736p, and outperforms autoencoders trained from scratch.

ABSTRACT

Highly compressed video autoencoders offer an effective way to accelerate video diffusion models, as the Diffusion Transformer (DiT) operates on far fewer tokens. However, such autoencoders are challenging to train, since a higher compression ratio degrades reconstruction quality and recovering it requires more channels, which is known to slow the convergence of the DiT. The compressed latent also differs from the one the DiT was trained on, so the pretrained DiT must be either retrained from scratch or adapted at considerable cost. Compressing the autoencoder the DiT was trained with appears to preserve compatibility, yet optimizing it for reconstruction alone still shifts the latent away from the distribution the DiT has learned. To address this, we propose **GRACE**, a two-stage framework that compresses a pretrained video autoencoder while keeping it compatible with the pretrained DiT. Specifically, we keep a frozen base latent from the pretrained encoder and learn a residual latent for the information lost under stronger compression, while aligning the compressed latent with the pretrained one in the feature space of the frozen DiT so that the autoencoder is optimized for generation. We then adapt the DiT with lightweight fine-tuning and asymmetric denoising, where the base is denoised ahead of the residual. GRACE reduces the token count of Wan2.1-I2V-14B by $8\times$ and its latency by $11.1\times$ at $480\times832\times81$, while matching the generation quality of the pretrained pipeline on VBench.

1 INTRODUCTION

Recent advances in video diffusion models (Wan et al., 2025; Ma et al., 2025; HaCohen et al., 2024) have demonstrated remarkable generation quality, yet their computational cost remains a significant

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

bottleneck for both training and inference. This computational cost largely depends on the number of tokens processed by the Diffusion Transformer (DiT) (Peebles & Xie, 2023), whose attention scales quadratically with sequence length.

The number of input tokens scales with the spatio-temporal resolution of the latent produced by the video autoencoder (Wan et al., 2025; HaCohen et al., 2024). Therefore, increasing the autoencoder compression ratio provides a straightforward way to reduce computation. However, fewer tokens limit the representational capacity of the latent, leading to a significant drop in reconstruction quality (Yao et al., 2025).

To preserve reconstruction quality with fewer tokens, previous works (Chen et al., 2025a;c; HaCohen et al., 2024; Zheng et al., 2026; Ma et al., 2025) increase the channel dimension of each token. However, higher-dimensional latents are known to compromise generation quality, introducing a trade-off between reconstruction and generation quality (Yao et al., 2025), and they converge more slowly, raising the training cost. Moreover, modifying the autoencoder architecture typically requires retraining the autoencoder from scratch, making it difficult to fully leverage pretrained knowledge and incurring substantial training cost. Reusing a released high-compression autoencoder avoids such cost but passes the burden to the pretrained DiT, which must adapt to a latent space never seen during pretraining and may not fully converge even on 160 GPUs (Zheng et al., 2026). A natural alternative is to leverage a pretrained autoencoder by adding a lightweight compression block to its architecture. However, we empirically find that this approach still degrades generation quality. Optimizing the autoencoder and the additional compression block solely for reconstruction shifts the resulting latent distribution away from that learned by the pretrained DiT, degrading generation quality (He et al., 2026; Zhao et al., 2024).

These observations point to two key challenges in efficient latent compression: (1) efficiently adapting pretrained autoencoder and DiT models while fully leveraging their pretrained knowledge, and (2) reducing the gap between reconstruction and generation quality introduced by compression. Our key insight is that a pretrained autoencoder and DiT already form a well-aligned generative pipeline, and this alignment can be exploited during compression. We therefore propose GRACE, a training framework that compresses a pretrained video autoencoder while explicitly accounting for both reconstruction and generation quality.

First, to fully leverage the pretrained autoencoder, we introduce a dual-latent representation that separates the pretrained representation from the information lost to compression. The base latent is obtained from the pretrained autoencoder using a spatially and temporally downsampled input, preserving the representation already learned by the pretrained autoencoder. The residual latent then captures the spatial detail and inter-frame motion that the low-resolution base latent cannot represent. This design allows the compressed autoencoder to retain the pretrained representation while using the residual latent only to complement what the downsampled input leaves out.

Second, we introduce asymmetric denoising to exploit this separation during generation. The base latent is denoised at an earlier timestep than the residual latent, allowing the denoised base to provide a stable anchor for the residual to recover the missing information. Both latents are denoised within a single pass, introducing no additional sampling cost.

Finally, to preserve compatibility with the pretrained DiT, we introduce generation-aware alignment, which matches the compressed autoencoder latent with the pretrained latent in the DiT's feature space. Rather than relying solely on conventional reconstruction objectives such as L1/L2 and perceptual losses (Zhang et al., 2018), we use intermediate DiT features to measure the distance between the compressed and pretrained latents. This generation-aware alignment encourages the compressed latent to remain compatible with the distribution learned by the pretrained DiT, thereby reducing the gap between reconstruction and generation quality.

Our method reduces the token count by $\mathbf{8}\times$ and end-to-end generation latency by $\mathbf{11.1}\times$ at $480\times832\times81$, while matching the generation quality of the pretrained pipeline on VBench (Huang et al., 2023; 2024) (Fig. 1), compressing both axes at once to $16\times$ spatial and $8\times$ temporal. The gain grows with resolution, reaching $15.6\times$ at $736\times1280\times81$. Compared with our single-latent baseline and existing high-compression video autoencoders on the same generation backbone, our method achieves better generation quality at the same or fewer tokens, even though its reconstruction on Panda-70M (Chen et al., 2024) is not the best among them, showing that reconstruction fidelity alone does not determine generation quality.

We summarize our contributions as follows:

- We propose GRACE, a diffusion training framework that efficiently compresses a pretrained video autoencoder while preserving the pretrained autoencoder–DiT pipeline and narrowing the reconstruction–generation gap.
- We introduce a dual-latent representation with asymmetric denoising to preserve the pretrained autoencoder representation while recovering fine-grained details lost by compression.
- We introduce a generation-aware alignment objective in the feature space of the pretrained DiT to preserve its compatibility with the compressed autoencoder.
- Our method reduces the token count by $8\times$, compressing both axes at once to $16\times$ spatial and $8\times$ temporal, while matching the generation quality of the pretrained pipeline with an $11.1\times$ reduction in latency at $480\times832\times81$ and a $15.6\times$ reduction at $736\times1280\times81$.

2 Related Work

2.1 Latent Compression for Efficient Video Generation

Video diffusion models are computationally expensive since they repeatedly denoise a large number of latent tokens. Existing acceleration methods either reduce the number of denoising steps (Song et al., 2022; Lu et al., 2022; Zhao et al., 2023; Wang et al., 2024; Li et al., 2024), often at the cost of quality, or reduce the computation per step (Xi et al., 2025; Sun et al., 2025; Huang et al., 2025). Several works instead propose to compress the latent beyond the common f8t4 setting (HaCohen et al., 2024; Zheng et al., 2026; Tian et al., 2025; Chen et al., 2025b; Ma et al., 2025), which significantly lowers the token count by a fixed ratio. Existing approaches increase spatial compression through autoencoder architectural changes (Chen et al., 2025a; Tian et al., 2025), or progressively increase temporal compression (Mahapatra et al., 2025). A complementary strategy assigns more channels to each token to preserve reconstruction quality at high compression ratios, but wider latents can degrade generation quality, which recent work addresses by structuring the channel dimension (Chen et al., 2025c). Our work compresses the video autoencoder in both space and time by fine-tuning a pretrained autoencoder and DiT rather than training either from scratch, keeping training cost low while mainly addressing the generation quality and convergence issues introduced by increased channel capacity.

2.2 Autoencoder Adaptation for Pretrained Generators

A few recent works redesign or improve the autoencoder in generation pipelines (Zheng et al., 2025; Yao et al., 2025; Chen et al., 2025c). However, changing the latent space typically requires retraining the DiT from scratch at prohibitive cost, since the new latent distribution no longer matches the one on which the DiT was trained. To avoid this cost, other works reuse the pretrained DiT and close the resulting mismatch during training (Zheng et al., 2026). Some constrain the new latent to remain reconstructable by the pretrained decoder (Zhao et al., 2024), while others align patch embeddings between the pretrained and modified pipelines after the latent space has already diverged (He et al., 2026). We instead optimize the autoencoder latent with intermediate features from the frozen DiT, keeping it close to the representation space where the pretrained generator operates.

3 Preliminaries

3.1 Video Autoencoders

A video autoencoder maps a video into a compressed latent space in which a diffusion model operates. Given an input video $\mathbf{x} \in \mathbb{R}^{3\times(1+L)\times H\times W}$, a causal video autoencoder produces $\mathbf{z} = \mathcal{E}(\mathbf{x}) \in \mathbb{R}^{C\times(1+\frac{L}{t})\times\frac{H}{f}\times\frac{W}{f}}$, where f and t are the spatial and temporal compression factors, and the decoder reconstructs $\hat{\mathbf{x}} = \mathcal{D}(\mathbf{z})$. Modern video autoencoders compress space and time jointly with 3D causal convolutions (Wan et al., 2025; HaCohen et al., 2024), commonly at f8t4.

3.2 Latent Video Diffusion Models

A diffusion transformer $\mathbf{v}_\theta$ (Peebles & Xie, 2023) generates in this latent space, patchifying the latent with a patch size p, so the token count is set by f, t, and p together. Following the rectified flow formulation (Esser et al., 2024), a schedule time $u \in [0, 1]$ is mapped to the flow matching

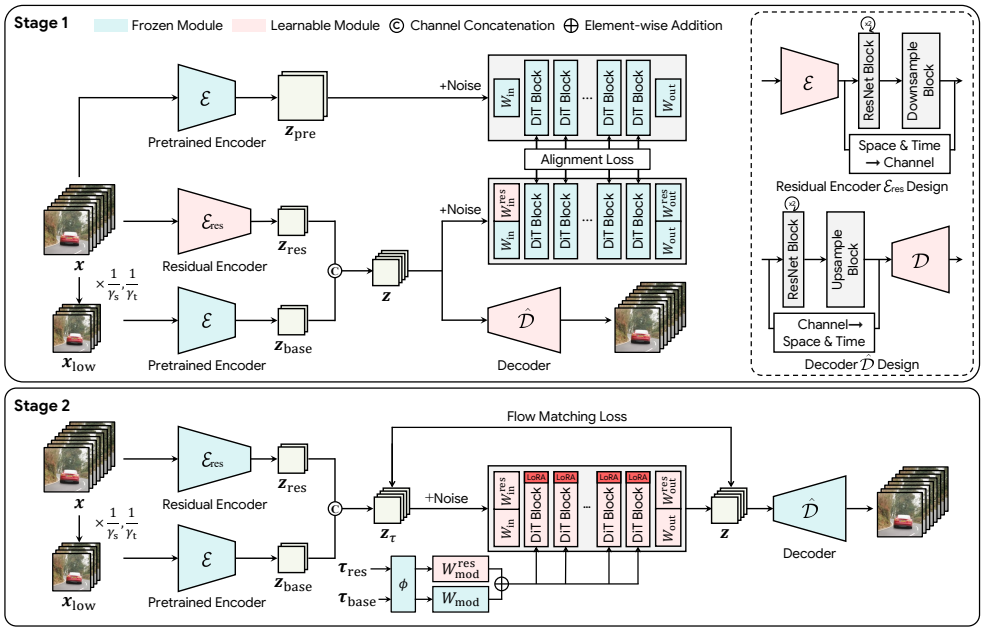


Figure 2: **Overall architecture.** Stage 1 trains the autoencoder. The frozen encoder $\mathcal{E}$ maps the downsampled input to $\mathbf{z}_{\text{base}}$, the residual encoder $\mathcal{E}_{\text{res}}$ maps the full-resolution input to $\mathbf{z}_{\text{res}}$, and $\hat{\mathcal{D}}$ decodes both, while $\mathcal{L}_{\text{align}}$ matches the compressed latent to the pretrained one inside the frozen DiT. Stage 2 adapts the DiT to the compressed latent with LoRA, denoising $\mathbf{z}_{\text{base}}$ ahead of $\mathbf{z}_{\text{res}}$.

timestep τ by a shift function ψ_s,

$$\tau = \psi_s(u) = \frac{s\,u}{1 + (s-1)\,u}\,, \tag{1}$$

where larger s places more of the schedule at higher noise, and the noisy latent is

$$\mathbf{z}_\tau = (1-\tau)\mathbf{z}_0 + \tau\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}). \tag{2}$$

A timestep embedding $\phi(\tau)$ is mapped by a projection W_{mod} to the modulation $\mathbf{m}$ that conditions every transformer block through adaptive layer normalization (AdaLN). Conditioned on $\mathbf{m}$ and text embeddings $\mathbf{c}$, $\mathbf{v}_\theta$ predicts $(\boldsymbol{\epsilon} - \mathbf{z}_0)$:

$$\mathcal{L}_{\text{velocity}} = \mathbb{E}_{\mathbf{z}_0, \boldsymbol{\epsilon}, \tau} \|\mathbf{v}_\theta(\mathbf{z}_\tau, \tau, \mathbf{c}) - (\boldsymbol{\epsilon} - \mathbf{z}_0)\|^2. \tag{3}$$

4 Method

4.1 Overview

We propose GRACE, a two-stage framework that compresses the latent of a pretrained pipeline while preserving generation quality (Fig. 2). **Stage 1** fine-tunes the autoencoder into a dual latent of a frozen base and a learned residual, guided by the frozen DiT (Section 4.2). **Stage 2** adapts the DiT to that latent, denoising the base ahead of the residual (Section 4.3). In both stages, the pretrained pipeline guides its own compression.

4.2 Stage 1: Autoencoder Training

We make full use of the pretrained autoencoder, keeping its architecture and weights, which keeps the latent close to the pretrained DiT's latent space and avoids training from scratch. The straightforward way to reach a higher ratio from there is to add compression blocks inside the encoder and train under the pretrained reconstruction objective (Wan et al., 2025; Zhang et al., 2018),

$$\mathcal{L}_{\text{recon}} = \mathcal{L}_1 + \lambda_{\text{LPIPS}}\mathcal{L}_{\text{LPIPS}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}\,, \tag{4}$$

which already gives a strong baseline, reconstructing on par with high-compression autoencoders trained from scratch (Tab. 1). However, reconstruction quality does not carry over to generation,

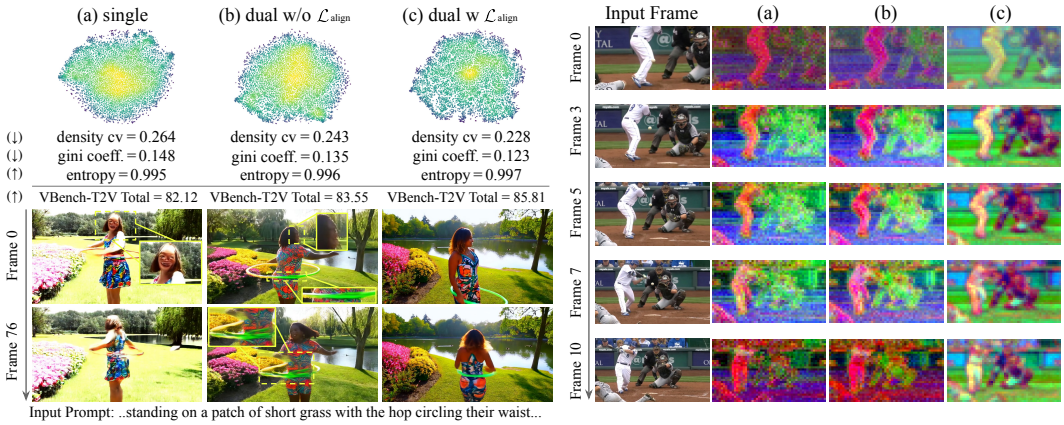


Figure 3: **Effect of the dual latent and $\mathcal{L}_{\text{align}}$.** **(a)** single latent, **(b)** dual-latent representation without $\mathcal{L}_{\text{align}}$, and **(c)** ours, all generated at $480\times832\times81$. **Left:** latent distribution, projected with t-SNE and coloured by kernel density, with its uniformity metrics and the total VBench score below, followed by T2V samples generated after Stage 2. **Right:** spatio-temporal structure of the latent, shown as its principal components mapped to RGB under the input frames.

even with the pretrained initialization. The latent drifts to optimize the objective, leaving the DiT with more to adapt to in Stage 2.

We address this during Stage 1 in two ways: (i) *dual-latent representation*, which anchors the compressed latent in the pretrained latent space, and (ii) *generation-aware alignment in the pretrained DiT feature space*, which matches the compressed latent to the pretrained one inside the frozen DiT.

Dual-latent representation. We first design the compressed latent as a dual representation, with a base latent and a residual latent. The base latent, taken from the frozen pretrained encoder $\mathcal{E}$, anchors the compressed latent in the pretrained latent space, so the DiT has less to adapt to in Stage 2. Since $\mathcal{E}$ compresses only by f and t, reaching the higher ratio requires reducing the input before encoding: we downsample it spatially by r_{s} and subsample it temporally by r_{t}, producing $\mathbf{x}_{\text{low}}$ within the resolution range $\mathcal{E}$ was trained on (Wan et al., 2025). Then $\mathcal{E}$ processes $\mathbf{x}_{\text{low}}$ with its original weights as

$$\mathbf{z}_{\text{base}} = \mathcal{E}(\mathbf{x}_{\text{low}}) \in \mathbb{R}^{C\times(1+\frac{L}{t\cdot r_t})\times\frac{H}{f\cdot r_s}\times\frac{W}{f\cdot r_s}} . \tag{5}$$

This reduced input lowers the token count, which limits the reconstruction capacity of $\mathbf{z}_{\text{base}}$ (Chen et al., 2025a;c), since the decoder cannot restore what never reached the latent. We therefore introduce a residual encoder $\mathcal{E}_{\text{res}}$, initialized from the pretrained weights, that takes the video at full resolution and carries the missing information,

$$\mathbf{z}_{\text{res}} = \mathcal{E}_{\text{res}}(\mathbf{x}) \in \mathbb{R}^{C'\times(1+\frac{L}{t\cdot r_t})\times\frac{H}{f\cdot r_s}\times\frac{W}{f\cdot r_s}} . \tag{6}$$

To match the spatial and temporal size of $\mathbf{z}_{\text{base}}$, $\mathcal{E}_{\text{res}}$ includes an additional downsampling stage, paired with a parameter-free shortcut that folds space and time into channels (Chen et al., 2025a). The two latents are concatenated along the channel axis,

$$\mathbf{z} = [\mathbf{z}_{\text{base}}; \mathbf{z}_{\text{res}}] \in \mathbb{R}^{(C+C')\times(1+\frac{L}{t\cdot r_t})\times\frac{H}{f\cdot r_s}\times\frac{W}{f\cdot r_s}} . \tag{7}$$

Separating the two parts organizes the latent, which prior work finds to help generation at high latent dimensions (Chen et al., 2025c). Compared with a single latent, the dual latent distorts the body less and keeps the colours natural, raising the total VBench score and the uniformity of the latent distribution (Fig. 3). The decoder is fully fine-tuned, with the residual channels zero-initialized so that training starts at the pretrained reconstruction quality. For image-to-video (I2V), we also pass the first frame to the decoder to restore detail the compressed latent cannot carry.

Generation-aware alignment in pretrained DiT feature space. The dual-latent representation alone narrows the gap between the compressed latent and the pretrained latent space, but we find it insufficient at higher compression ratios, where the residual latent must carry more information missing from the base. The residual channels are optimized only through the reconstruction objective, which improves fidelity but does not directly account for how easily the diffusion model can

learn the resulting latent. We introduce generation-aware alignment, a regularizer that supervises the compressed latent inside the feature space of the pretrained DiT without modifying the autoencoder architecture. Our approach builds on the same insight as prior autoencoder adaptation methods: reconstruction fidelity alone does not necessarily yield a compressed latent that the diffusion backbone can learn effectively (Yao et al., 2025; Chen et al., 2025c). Since the pretrained DiT is already adapted to the original latent space, we compare the two latents in its intermediate feature space. We draw τ from the pretrained pipeline's schedule and noise both latents at that level with independent noise. The alignment loss is then defined by passing both noised latents through this frozen DiT and matching their intermediate representations. For the pretrained latent, the frozen encoder encodes the full-resolution video, and v_θ patchifies the resulting latent through the original input projection W_{in} before the transformer blocks:

$$\mathbf{z}_{\text{pre}} = \mathcal{E}(\mathbf{x}) \in \mathbb{R}^{C\times(1+\frac{L}{t})\times\frac{H}{f}\times\frac{W}{f}}, \qquad \mathbf{h}^l_{\text{pre}} = \mathbf{v}^l_\theta(\mathbf{z}_{\text{pre},\tau}, \tau)\,, \tag{8}$$

where τ is a randomly sampled flow matching timestep, $\mathbf{z}_{\text{pre},\tau}$ and $\mathbf{z}_\tau$ denote the noised pretrained and compressed latents following Eq. 2, and $\mathbf{v}^l_\theta$ is the feature from the l-th transformer layer of $\mathbf{v}_\theta$. For the compressed latent $\mathbf{z}$, we use $\mathbf{v}_{\theta,+}$, which shares the frozen transformer body of $\mathbf{v}_\theta$ but extends the input projection along the channel axis with a zero-initialized $W^{\text{res}}_{\text{in}}$ to handle the residual channels. Since this branch runs at a lower latent resolution, its features contain fewer tokens. We reshape them to their spatio-temporal grid, trilinearly upsample them to the full-resolution grid, and apply a zero-initialized per-layer residual projection P_l:

$$\mathbf{h}^l_{\text{cmp}} = \mathbf{v}^l_{\theta,+}(\mathbf{z}_\tau, \tau)\,, \qquad \hat{\mathbf{h}}^l_{\text{cmp}} = \text{interp}(\mathbf{h}^l_{\text{cmp}}) + P_l\big(\text{interp}(\mathbf{h}^l_{\text{cmp}})\big)\,. \tag{9}$$

We then compare the two features at each supervised layer:

$$\mathcal{L}_{\text{align}} = \sum_{l\in S}\Big[1 - \text{sim}\big(\mathbf{h}^l_{\text{pre}}, \hat{\mathbf{h}}^l_{\text{cmp}}\big)\Big]\,, \tag{10}$$

where $\text{sim}(\cdot,\cdot)$ is the cosine similarity averaged over tokens and S is the set of supervised layers. The features from $\mathbf{z}_{\text{pre}}$ are kept fixed, so $\mathcal{L}_{\text{align}}$ only optimizes the compressed representation.

Since $\mathcal{L}_{\text{align}}$ and $\mathcal{L}_{\text{recon}}$ operate on different scales, a fixed weight makes training unstable. Following prior work (Yao et al., 2025), we set the weight as the ratio of their gradient norms with respect to the last convolutional layer of the encoder, so that the two losses contribute at a comparable scale without manual hyperparameter tuning. The final training objective is

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + w_{\text{adaptive}}\,\mathcal{L}_{\text{align}}\,, \qquad w_{\text{adaptive}} = \frac{\|\nabla\mathcal{L}_{\text{recon}}\|}{\|\nabla\mathcal{L}_{\text{align}}\|}\,. \tag{11}$$

We analyze the resulting latent in Fig. 3 from two complementary perspectives. Following prior work that relates the uniformity of the latent distribution to generation quality (Yao et al., 2025), we fit a kernel density estimate to the t-SNE (van der Maaten & Hinton, 2008) projection and measure its coefficient of variation, Gini coefficient, and normalized entropy. All three improve with the alignment loss, and fewer artifacts remain in the samples, with the total VBench score following. These metrics summarize the latent as a whole, while the PCA visualization in Fig. 3 shows how it varies within and across frames: the dual latent makes the components less noisy, but they stay weak until the alignment loss makes them follow object regions and hold structure across frames.

4.3 Stage 2: DiT Adaptation

In this section, we adapt the pretrained DiT to the compressed latent while keeping the autoencoder frozen. Since the compressed latent adds C' unseen channels, we extend the DiT input and output projections from C to $C + C'$ channels. We fully fine-tune these projections and apply LoRA (Hu et al., 2021) to the transformer blocks, keeping the adaptation cost low while preserving the pretrained generation capability (He et al., 2026). The DiT is trained with the flow matching loss $\mathcal{L}_{\text{velocity}}$, applied to the base and the residual latent at different noise levels, which we describe next.

Asymmetric denoising. Our key motivation is to denoise the two parts of the latent asymmetrically, keeping the base ahead of the residual so that the residual builds on a reliable foundation throughout generation. The residual encoder is trained to carry what the base does not, so what the residual should contain becomes clearer as the base settles, which makes the residual easier to generate.

During training, we sample $\delta \sim \mathcal{U}(\delta_{\min}, \delta_{\max})$ and $\hat{u} \sim \mathcal{U}(0, 1+\delta)$ per video, and set

$$u_{\text{base}} = \max(\hat{u} - \delta, 0)\,, \qquad u_{\text{res}} = \min(\hat{u}, 1)\,, \tag{12}$$

Autoencoder	Config	AE Training	Latent Tokens	Reconstruction				VBench-I2V
				PSNR ↑	SSIM ↑	LPIPS ↓	rFVD ↓	Total ↑
Wan2.1-VAE (Wan et al., 2025)	f8t4c16p2	scratch†	32.8k	**35.15**	**0.958**	**0.016**	**1.13**	**87.92**
Step-Video-VAE (Ma et al., 2025)	f16t8c64p1	scratch	17.2k	33.88	0.950	0.029	3.16	84.05
Video DC-AE (Zheng et al., 2026)	f32t4c128p1	scratch	8.2k	34.61	0.956	0.024	3.70	84.94
LTX-VAE (HaCohen et al., 2024)	f32t8c128p1	scratch	4.3k	31.97	0.914	0.051	19.53	87.06
Baseline	f16t8c32p2	fine-tuned	4.3k	33.76	0.956	0.031	13.11	86.44
GRACE (ours)	f16t8c32p2	fine-tuned	4.3k	32.63	0.930	0.032	13.53	87.86

Table 1: **Video autoencoder comparison at** $256\times256\times81$ **reconstruction and** $480\times832\times81$ **generation.** We report the VBench-I2V total after adapting the same pretrained Wan2.1-I2V-14B to every latent under the same budget. *Config* lists f, t, channel count c, and patch size p, which set the number of *latent tokens*. Our method is shaded. †Initialized by inflating its own 2D image autoencoder.

where $u = 1$ corresponds to pure noise and $u = 0$ to a clean latent, so the base is always the less corrupted of the two. We obtain the noise levels τ_{base} and τ_{res} from Eq. 1, and noise the base and the residual separately with Eq. 2:

$$\mathbf{z}_\tau = \left[(1-\tau_{\text{base}})\mathbf{z}_{\text{base}}+\tau_{\text{base}}\boldsymbol{\epsilon}_{\text{base}};\,(1-\tau_{\text{res}})\mathbf{z}_{\text{res}}+\tau_{\text{res}}\boldsymbol{\epsilon}_{\text{res}}\right], \tag{13}$$

where $\boldsymbol{\epsilon}_{\text{base}}, \boldsymbol{\epsilon}_{\text{res}} \sim \mathcal{N}(0, \mathbf{I})$. We sample δ rather than fixing it, which trains one model across offsets. The two timesteps share ϕ and enter through separate projections, which keeps the conditioning path closer to the pretrained path. We apply the pretrained W_{mod} to the base and a zero-initialized $W^{\text{res}}_{\text{mod}}$ to the residual, so adaptation begins from the pretrained behavior,

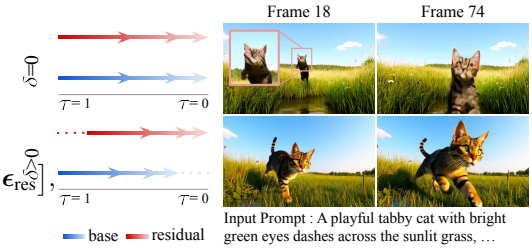


Figure 4: **Denoising order.** Each row shows the sampling steps of $\mathbf{z}_{\text{base}}$ and $\mathbf{z}_{\text{res}}$ (**Left**) and the generated frames (**Right**), with $\tau = 1$ pure noise and $\tau = 0$ clean. **(Top)** Both parts are denoised at the same noise level. **(Bottom)** $\mathbf{z}_{\text{base}}$ stays δ ahead of $\mathbf{z}_{\text{res}}$ at every step.

$$\mathbf{m} = W_{\text{mod}}\,\phi(\tau_{\text{base}}) + W^{\text{res}}_{\text{mod}}\,\phi(\tau_{\text{res}})\,. \tag{14}$$

The modulation $\mathbf{m}$ conditions every transformer block, and the DiT predicts a velocity for each part,

$$[\hat{\mathbf{v}}_{\text{base}}, \hat{\mathbf{v}}_{\text{res}}] = \mathbf{v}_\theta(\mathbf{z}_\tau, [\tau_{\text{base}}, \tau_{\text{res}}], \mathbf{c})\,, \tag{15}$$

and the training objective is the average of the flow matching loss of Eq. 3 over the base and the residual. At inference, we fix a single offset $\delta \in [\delta_{\min}, \delta_{\max}]$ and let the base lead the residual by δ throughout sampling. Both parts are denoised in the same forward pass, rather than one after the other, so the number of function evaluations remains unchanged. Fig. 4 shows that the asymmetric schedule recovers detail on the subject's face that a shared schedule degrades where motion is large.

5 Experiments

5.1 Setup

Model configuration. We use Wan2.1 (Wan et al., 2025) as the pretrained pipeline and compress its latent from f8t4 to f16t8 with r_s=2, r_t=2, and C'=16 residual channels. Full training and architecture details are in Appendices A and B.

Single-latent baseline. Under the same configuration above, our single-latent baseline only adds compression blocks to the pretrained autoencoder and fine-tunes it, without any further modification. The DiT is adapted following the training recipe of Wan2.1 (Wan et al., 2025).

5.2 Video Reconstruction Results

Evaluation details. We evaluate reconstruction on Panda-70M (Chen et al., 2024) with PSNR, SSIM (Wang et al., 2004), LPIPS (Zhang et al., 2018), and rFVD (Unterthiner et al., 2019), and generation on VBench-I2V (Huang et al., 2023; 2024). The compared autoencoders are trained from scratch at their own compression ratios, while our single-latent baseline and GRACE are fine-tuned from Wan2.1-VAE. To measure generation, we train a DiT for every latent, starting from the

Model	Autoencoder	Config	Diffusion Model	Latent Tokens	DiT Params	NFE	Latency (s) ↓		VBench-T2V			VBench-I2V		
							T2V	I2V	Quality ↑	Semantic ↑	Total ↑	I2V ↑	Quality ↑	Total ↑
Wan2.1-14B	Wan2.1-VAE	f8t4c16p2	Wan2.1-14B	32.8k	14B	100	851.5	863.2	85.24	78.70	83.93	**95.82**	80.01	87.92
Open-Sora 2.0	Video DC-AE	f32t4c128p1	Open-Sora	8.2k	11B	150	138.6	138.4	78.80	72.35	77.51	91.17	76.96	84.07
DC-Gen	DC-AE-V	f32t4c32p1	Wan2.1-14B	8.2k	14B	100	157.1	165.0	85.85	79.70	84.62	88.25	80.01	84.13
LTX-Video 0.9.7	LTX-VAE	f32t8c128p1	LTX-Video	4.3k	13B	150	99.6	104.1	82.88	64.31	79.17	95.62	**80.32**	**87.97**
Baseline	Baseline	f16t8c32p2	Wan2.1-14B	4.3k	14B	100	76.1	77.8	83.21	77.75	82.12	93.67	79.21	86.44
GRACE (ours)	**GRACE**	f16t8c32p2	Wan2.1-14B	4.3k	14B	100	**75.8**	**77.7**	**86.02**	**84.98**	**85.81**	95.61	80.11	87.86

Table 2: **Video generation on VBench at** $480\times832\times81$**.** Notation follows Tab. 1. Our method preserves the quality of the pretrained pipeline on both tasks while using $8\times$ fewer tokens and running $11.1\times$ faster. NFE counts the DiT forward passes per video over 50 sampling steps. Per-dimension scores are in Appendix D.1.



Figure 5: **Qualitative comparison of video generation.** VBench samples at $480 \times 832 \times 81$ from the pretrained Wan2.1-14B and GRACE (ours), generated from the same prompt, for text-to-video (**top**) and image-to-video (**bottom**). Best viewed when zoomed in.

pretrained Wan2.1-I2V-14B, with a budget of 38.5 H200 GPU days each. Our adaptation builds on the frozen Wan2.1 base latent, which is unavailable to the compared autoencoders, so we transfer them with the released implementation of DC-Gen (He et al., 2026; Chen et al., 2025b) instead.

Main results. As shown in Tab. 1, better reconstruction does not mean better generation. Step-Video-VAE reconstructs 1.91 dB above LTX-VAE but scores 3.01 lower on VBench-I2V, even at $4\times$ the tokens, and our single-latent baseline reconstructs 1.13 dB above GRACE yet scores 1.42 lower. GRACE reaches the highest generation quality among the compressed autoencoders, at the smallest token count in the table, and comes within 0.06 of the uncompressed pipeline. Video DC-AE and Step-Video-VAE use $2\times$ and $4\times$ more tokens than GRACE, yet fall 2.98 and 3.87 behind that pipeline.

5.3 Video Generation Results

Evaluation details. We evaluate on VBench (Huang et al., 2023; 2024) for both image-to-video (I2V) and text-to-video (T2V) at $480\times832\times81$ with 50 sampling steps, using each model's default guidance configuration, and measure latency on a single A100 GPU. The autoencoders we adapt to Wan2.1-14B share the same pretrained DiT and adaptation budget, while LTX-Video and Open-Sora 2.0 were trained with their own generators, differ in scale and data, and each run a third guidance branch for 150 forward passes per video rather than 100.

Main results. As shown in Tab. 2, GRACE stays within 0.06 of the pretrained pipeline on I2V and exceeds it on T2V, while running $11.1\times$ faster. The T2V gain comes almost entirely from the semantic score, where GRACE leads the next best model by 5.28. Generation-aware alignment not only preserves the pretrained latent under compression but also improves its semantic structure, as the latent analysis in Fig. 3 shows. Compared with DC-Gen, which adapts the same pretrained DiT at twice the token count, GRACE is faster on both tasks and scores higher on every VBench total. Although LTX-Video reaches a higher I2V total at the same token count, it is slower than GRACE,

Method	Components			Reconstruction			VBench-T2V			VBench-I2V		
	$[\mathbf{z}_{\text{base}}; \mathbf{z}_{\text{res}}]$	$\mathcal{L}_{\text{align}}$	$\delta > 0$	PSNR ↑	LPIPS ↓	rFVD ↓	Quality ↑	Semantic ↑	Total ↑	I2V ↑	Quality ↑	Total ↑
(I) Wan2.1-I2V-14B (pretrained)	–	–	–	35.15	0.016	1.13	85.24	78.70	83.93	95.82	80.01	87.92
(II) Baseline	×	×	×	33.76	0.031	13.11	83.21	77.75	82.12	93.67	79.21	86.44
(III) + dual-latent	✓	×	×	33.10	0.031	13.89	84.77	78.65	83.55	93.96	79.32	86.64
(IV) + $\mathcal{L}_{\text{align}}$	✓	✓	×	32.63	0.032	13.53	85.22	82.55	84.68	95.23	79.78	87.51
(V) + $\delta > 0$ (**Ours**)	✓	✓	✓	32.63	0.032	13.53	86.02	84.98	85.81	95.61	80.11	87.86
(VI) Baseline w/o pretrained DiT	×	×	×	33.76	0.031	13.11	74.20	58.40	71.04	53.06	73.94	63.50

Table 3: **Ablation on design components.** Row (VI) keeps the autoencoder of (II) but trains the DiT from scratch for the same number of steps.

since a third guidance branch for temporal consistency adds a DiT pass per step. We also observe that LTX-Video often produces near-static videos in this setting, with the lowest dynamic degree among the compared models (Tab. D.3b).

5.4 Training Cost and Convergence

Fig. 6 tracks the flow matching loss on the residual channels during DiT adaptation, under identical settings except for $\mathcal{L}_{\text{align}}$. The residual converges faster and reaches a lower loss with alignment, while the base channels behave almost identically, which is what the design assumes: the DiT already models that space (Appendix F.2). In total, our pipeline takes 8.5 H200-days for the autoencoder and 30 for the DiT, whereas adapting a pretrained DiT to the Video DC-AE latent of Open-Sora 2.0 is reported to produce blurry videos and to fall short of training from scratch (Zheng et al., 2026; He et al., 2026).

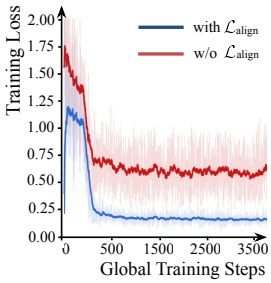


Figure 6: **Convergence during DiT adaptation.**

5.5 Ablations and Discussion

Ablation on design components. Each component in Tab. 3 improves generation on both tasks. The dual latent and the offset help much more on T2V (+1.43 and +1.13) than on I2V (+0.20 and +0.35), as both determine what the generation is anchored to, whereas I2V already provides the conditioning image as an anchor. Alignment behaves differently, raising the I2V score as well as both T2V scores, since the conditioning image anchors the content but does not resolve the mismatch between the compressed latent and the DiT's pretrained denoising space. Alignment therefore recovers most of the gap to the pretrained pipeline on I2V and goes further on T2V, where the total ends up above that pipeline, which may relate to the latent analysis in Fig. 3.

Alignment target. We align the compressed latent inside the pretrained DiT (Section 4.2), and Tab. 4 assesses the impact of the alignment target while keeping everything else fixed. Prior work regularizes a high-dimensional latent with a pretrained vision foundation model so that it is easier for a diffusion model to learn (Yao et al., 2025). For that comparison, we use V-JEPA 2.1 (Mur-Labadia et al., 2026), a video foundation model whose dense features are spatially structured and temporally consistent. Aligning to V-JEPA 2.1 raises the semantic score from 78.65 to 80.04, while aligning inside the pretrained DiT raises it far higher, to 82.55, with the totals following the same order. These results suggest that vision foundation models provide useful semantic structure in general, but in our setting, the relevant structure is the one already represented by the pretrained DiT, since the compressed autoencoder is adapted to a fixed denoising model.

Alignment target	Reconstruction		VBench-T2V		
	PSNR ↑	rFVD ↓	Quality ↑	Semantic ↑	Total ↑
None	33.10	13.89	84.77	78.65	83.55
V-JEPA-2.1 features	32.61	13.93	84.98	80.04	83.99
Pretrained DiT features (**ours**)	32.63	13.53	85.22	82.55	84.68

Table 4: **Alignment target.** Asymmetric denoising is disabled in all rows.

Denoising schedule design. Tab. 5 compares the denoising order, used in both DiT adaptation and sampling. Denoising the base first improves the total, while reversing it drops below the synchronous schedule, showing that the direction of the offset is what matters.

Denoising Order	Offset	VBench-I2V		
		I2V ↑	Quality ↑	Total ↑
$\tau_{\text{base}} = \tau_{\text{res}}$	$\delta = 0$	95.23	79.78	87.51
$\tau_{\text{base}} > \tau_{\text{res}}$	$\delta < 0$	94.61	79.72	87.17
$\tau_{\text{base}} < \tau_{\text{res}}$ (ours)	$\delta > 0$	95.61	80.11	87.86

Table 5: **Ablation on the denoising order.**

6 Conclusion

We presented GRACE, a framework that compresses the latent of a pretrained video diffusion pipeline by $8\times$ and adapts the DiT to the compressed latent. On Wan2.1-I2V-14B, GRACE reaches the generation quality of the pretrained pipeline at $11.1\times$ lower latency, and the gain grows with resolution.

AI USE STATEMENT

In this work, we used generative AI tools for writing assistance, including polishing prose, shortening captions, and formatting results into LaTeX tables, and for assistance with implementing our method. We have not used generative AI tools for research ideation, methodology design, or experiment design, and generating synthetic data, formulating mathematical claims, and assisting with proofs are not applicable to this work. Additionally, we used generative AI tools to search for related work, whose relevance and content we verified by reading the cited papers ourselves. We have reviewed all AI-assisted work: AI-assisted code was read and tested by the authors, every number reported in this paper was produced by our own experiments and transcribed into the tables by the authors, and all AI-assisted text was rewritten or approved by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The architecture and training setup of our autoencoder are described in Appendix A, and those of the DiT adaptation in Appendix B, with all hyperparameters listed in the corresponding tables. Appendix C describes how each compared autoencoder is adapted to the pretrained DiT and how every model is evaluated, including the sampling configuration of each model and the protocol used to measure latency.

REFERENCES

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models, 2025a. URL https://arxiv.org/abs/2410.10733.

Junyu Chen, Wenkun He, Yuchao Gu, Yuyang Zhao, Jincheng Yu, Junsong Chen, Dongyun Zou, Yujun Lin, Zhekai Zhang, Muyang Li, Haocheng Xi, Ligeng Zhu, Enze Xie, Song Han, and Han Cai. Dc-videogen: Efficient video generation with deep compression video autoencoder, 2025b. URL https://arxiv.org/abs/2509.25182.

Junyu Chen, Dongyun Zou, Wenkun He, Junsong Chen, Enze Xie, Song Han, and Han Cai. Dc-ae 1.5: Accelerating diffusion model convergence with structured latent space, 2025c. URL https://arxiv.org/abs/2508.00413.

Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang wei Chao, Byung Eun Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, and Sergey Tulyakov. Panda-70m: Captioning 70m videos with multiple cross-modality teachers, 2024. URL https://arxiv.org/abs/2402.19479.

Yu Cheng and Fajie Yuan. Leanvae: An ultra-efficient reconstruction vae for video diffusion models, 2025. URL https://arxiv.org/abs/2503.14325.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. URL https://arxiv.org/abs/2403.03206.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. Ltx-video: Realtime video latent diffusion, 2024. URL https://arxiv.org/abs/2501.00103.

Wenkun He, Yuchao Gu, Junyu Chen, Junyi Wu, Wenhang Ge, Dongyun Zou, Yujun Lin, Zhekai Zhang, Haocheng Xi, Muyang Li, et al. Dc-gen: Post-training diffusion acceleration with deeply compressed latent space. In *European Conference on Computer Vision*, pp. 249–269. Springer, 2026.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion, 2025. URL https://arxiv.org/abs/2506.08009.

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, 2023. URL https://arxiv.org/abs/2311.17982.

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, 2024. URL https://arxiv.org/abs/2411.13503.

Jiachen Li, Weixi Feng, Tsu-Jui Fu, Xinyi Wang, Sugato Basu, Wenhu Chen, and William Yang Wang. T2v-turbo: Breaking the quality bottleneck of video consistency model with mixed reward feedback, 2024. URL https://arxiv.org/abs/2405.18750.

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. URL https://arxiv.org/abs/2411.17459.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, 2022. URL https://arxiv.org/abs/2206.00927.

Guoqing Ma, Haoyang Huang, Kun Yan, Liangyu Chen, Nan Duan, Shengming Yin, Changyi Wan, Ranchen Ming, Xiaoniu Song, Xing Chen, Yu Zhou, Deshan Sun, Deyu Zhou, Jian Zhou, Kaijun Tan, Kang An, Mei Chen, Wei Ji, Qiling Wu, Wen Sun, Xin Han, Yanan Wei, Zheng Ge, Aojie Li, Bin Wang, Bizhu Huang, Bo Wang, Brian Li, Changxing Miao, Chen Xu, Chenfei Wu, Chenguang Yu, Dapeng Shi, Dingyuan Hu, Enle Liu, Gang Yu, Ge Yang, Guanzhe Huang, Gulin Yan, Haiyang Feng, Hao Nie, Haonan Jia, Hanpeng Hu, Hanqi Chen, Haolong Yan, Heng Wang, Hongcheng Guo, Huilin Xiong, Huixin Xiong, Jiahao Gong, Jianchang Wu, Jiaoren Wu, Jie Wu, Jie Yang, Jiashuai Liu, Jiashuo Li, Jingyang Zhang, Junjing Guo, Junzhe Lin, Kaixiang Li, Lei Liu, Lei Xia, Liang Zhao, Liguo Tan, Liwen Huang, Liying Shi, Ming Li, Mingliang Li, Muhua Cheng, Na Wang, Qiaohui Chen, Qinglin He, Qiuyan Liang, Quan Sun, Ran Sun, Rui Wang, Shaoliang Pang, Shiliang Yang, Sitong Liu, Siqi Liu, Shuli Gao, Tiancheng Cao, Tianyu Wang, Weipeng Ming, Wenqing He, Xu Zhao, Xuelin Zhang, Xianfang Zeng, Xiaojia Liu, Xuan Yang, Yaqi Dai, Yanbo Yu, Yang Li, Yineng Deng, Yingming Wang, Yilei Wang, Yuanwei Lu, Yu Chen, Yu Luo, Yuchu Luo, Yuhe Yin, Yuheng Feng, Yuxiang Yang, Zecheng Tang, Zekai Zhang, Zidong Yang, Binxing Jiao, Jiansheng Chen, Jing Li, Shuchang Zhou, Xiangyu Zhang, Xinhao Zhang, Yibo Zhu, Heung-Yeung Shum, and Daxin Jiang. Step-video-t2v technical report: The practice, challenges, and future of video foundation model, 2025. URL https://arxiv.org/abs/2502.10248.

Aniruddha Mahapatra, Long Mai, Yitian Zhang, David Bourgin, and Feng Liu. Progressive growing of video tokenizers for highly compressed latent spaces. *arXiv preprint arXiv:2501.05442*, 2025.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning, 2026. URL https://arxiv.org/abs/2603.14482.

NVIDIA, :, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma,

Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezanali, Fitsum Reda, Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchapmi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Lin Yen-Chen, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. Cosmos world foundation model platform for physical ai, 2025. URL https://arxiv.org/abs/2501.03575.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Jaeyo Shin, Jiwook Kim, and Hyunjung Shim. Representation alignment for just image transformers is not easier than you think, 2026. URL https://arxiv.org/abs/2603.14366.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Wenhao Sun, Rong-Cheng Tu, Jingyi Liao, Zhao Jin, and Dacheng Tao. Asymrnr: Video diffusion transformers acceleration with asymmetric reduction and restoration, 2025. URL https://arxiv.org/abs/2412.11706.

Jordan Taylor, William Agnew, Maarten Sap, Sarah E Fox, and Haiyi Zhu. The algorithmic gaze of image quality assessment: An audit and trace ethnography of the laion-aesthetics predictor. In *Proceedings of the 2026 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '26, pp. 6383–6402. ACM, 2026. doi: 10.1145/3805689.3806462. URL http://dx.doi.org/10.1145/3805689.3806462.

Rui Tian, Qi Dai, Jianmin Bao, Kai Qiu, Yifan Yang, Chong Luo, Zuxuan Wu, and Yu-Gang Jiang. Reducio! generating 1k video within 16 seconds using extremely compressed motion latents, 2025. URL https://arxiv.org/abs/2411.13552.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric challenges, 2019. URL https://arxiv.org/abs/1812.01717.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

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

Fu-Yun Wang, Zhaoyang Huang, Weikang Bian, Xiaoyu Shi, Keqiang Sun, Guanglu Song, Yu Liu, and Hongsheng Li. Animatelcm: Computation-efficient personalized style video generation without personalized video data, 2024. URL https://arxiv.org/abs/2402.00769.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

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. URL https://arxiv.org/abs/2411.06449.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity, 2025. URL https://arxiv.org/abs/2502.01776.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models, 2025. URL https://arxiv.org/abs/2501.01423.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. URL https://arxiv.org/abs/1801.03924.

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, 2024. URL https://arxiv.org/abs/2405.20279.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models, 2023. URL https://arxiv.org/abs/2302.04867.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders, 2025. URL https://arxiv.org/abs/2510.11690.

Zangwei Zheng, Xiangyu Peng, Yuxuan Lou, Chenhui Shen, Tom Young, Xinying Guo, Binluo Wang, Hang Xu, Hongxin Liu, Mingyan Jiang, Wenjun Li, Yuhui Wang, Anbang Ye, Gang Ren, Qianran Ma, Wanying Liang, Xiang Lian, Xiwen Wu, Yuting Zhong, Zhuangyan Li, Chaoyu Gong, Guojun Lei, Leijun Cheng, Limin Zhang, Minghao Li, Ruijie Zhang, Silan Hu, Shijie Huang, Xiaokang Wang, Yuanheng Zhao, Yuqi Wang, Ziang Wei, and Yang You. Open-sora 2.0: Training a commercial-level video generation model in $200k$, $2026. URL$.

APPENDIX

This appendix provides supplementary material to support the main paper.

- Appendix A describes the architecture and the training setup of our autoencoder, and Appendix B does the same for the DiT, including how it is extended to the compressed latent.
- Appendix C describes how the compared autoencoders are adapted to the pretrained DiT and how every model is evaluated.
- Appendix D reports per-dimension VBench scores, results at a higher resolution, and a human evaluation.
- Appendix E ablates the autoencoder design and the alignment design.
- Appendix F analyzes the compressed latent and the convergence of each of its parts, and Appendix G reports the training and inference cost.
- Appendix H extends the discussion of related work, Appendix I collects the qualitative samples referenced throughout the paper, and Appendix J states the limitations of our method.

Contents

A GRACE Autoencoder Details **16**
A.1 Architectural Details 16
A.2 Implementation Details 17

B GRACE Diffusion Model Details **18**
B.1 Architectural Details 18
B.2 Implementation Details 18

C More Details on Evaluation **19**
C.1 Adaptation of Other Autoencoders 19
C.2 Evaluation Protocol 19

D Additional Evaluation **20**
D.1 Video Generation 20
D.2 Human Evaluation 23

E Additional Ablations **25**
E.1 Autoencoder Design 25
E.2 Alignment Design 25

F Additional Analysis **26**
F.1 Latent Analysis 26
F.2 Convergence Behavior 27

G Computational Cost **28**

H More Discussion on Related Work **28**

I Additional Qualitative Results **29**

J Limitations **39**

A GRACE AUTOENCODER DETAILS

A.1 ARCHITECTURAL DETAILS

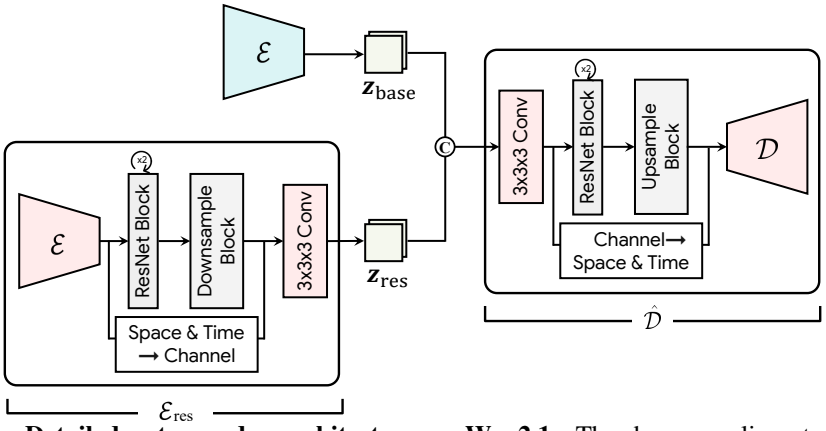


Figure A.1: **Detailed autoencoder architecture on Wan2.1.** The downsampling stage inserted in the middle of $\mathcal{E}_{\text{res}}$, consisting of residual blocks and a strided causal convolution, paired with a parameter-free shortcut that folds space and time into channels.

Fig. A.1 shows the architecture of our autoencoder in detail. We reuse the encoder and decoder backbones of Wan2.1 and describe our additions below.

Base encoding. We obtain $\mathbf{z}_{\text{base}}$ from the frozen encoder $\mathcal{E}$. Since $\mathcal{E}$ compresses at fixed factors, we reduce the video first. We downsample the frames by r_s with bilinear interpolation, and subsample them by r_t in time. We experiment with average pooling, strided sampling, and bilinear interpolation as reduction strategies, and the setting above reconstructs best.

Residual encoding. The residual encoder $\mathcal{E}_{\text{res}}$ has the same architecture as $\mathcal{E}$, but takes the full-resolution video, so it needs an additional compression step to match the target resolution. In the pretrained encoder, four downsampling stages are followed by middle blocks that refine the feature at the final resolution through residual and attention layers, and then by a head that projects it to the latent. We add one downsample block between the middle blocks and the head. The block reuses the design of the pretrained downsample blocks, which pass the feature through two residual blocks before a strided convolution over space and a strided causal convolution over time. We attach a non-parametric shortcut to this block, following the residual autoencoding of DC-AE (Chen et al., 2025a). Since the block compresses time as well as space, the shortcut moves both axes into the channel axis, then averages channel groups to match the channel number of the block output. Its result is added to the block output, which leaves the block to learn the residual.

Decoding. The decoder $\hat{\mathcal{D}}$ extends the input convolution of $\mathcal{D}$ from C to $C{+}C'$ channels for the concatenated latent, and adds one upsample block symmetric to the downsample block of $\mathcal{E}_{\text{res}}$. The weights for the residual channels are zero-initialized, so training starts at the pretrained reconstruction quality. The non-parametric shortcut of this block runs in the opposite direction, duplicating channels and then applying a channel-to-space-and-time operation. In the image-to-video setting, the first frame is available at inference as well as during training, and we pass it to $\hat{\mathcal{D}}$ to restore detail the compressed latent cannot carry on its own. We encode the first frame with $\mathcal{E}_{\text{res}}$ and inject its intermediate features into $\hat{\mathcal{D}}$ through gated cross-attention (Tian et al., 2025). We take the feature before each downsampling stage of the encoder and attend to it at the matching stage of $\hat{\mathcal{D}}$. A learned gate on each block controls how much of the first frame reaches $\hat{\mathcal{D}}$.

Single-latent Baseline. The baseline replaces the base and residual pair with a single encoder, so there is no frozen $\mathcal{E}$ and no reduction of the input. The compression block keeps the same design and placement, and both the non-parametric shortcut and the first-frame cross-attention are unchanged.

A.2 Implementation Details

We train the autoencoder in two phases. In the first phase, we update $\mathcal{E}_{\text{res}}$ and $\hat{\mathcal{D}}$ at $256 \times 256 \times 81$, along with $W_{\text{in}}^{\text{res}}$ and the per-layer projections P_l. All other parameters of $\mathbf{v}_\theta$ and $\mathbf{v}_{\theta,+}$ remain frozen. In the second phase, we raise the resolution to $512 \times 512 \times 81$ and train $\hat{\mathcal{D}}$ alone, following the decoupled high-resolution adaptation of DC-AE (Chen et al., 2025a), which freezes $\mathcal{E}_{\text{res}}$ as well so that the latent space stays fixed while the decoder adapts. $\mathcal{E}$ stays frozen in both phases.

Training setup. We train on Panda-70M (Chen et al., 2024) with AdamW, using the hyperparameters in Tab. A.1. The clip length stays at 81 frames, since reconstruction degrades quickly on shorter clips under high temporal compression. The modules we add on the DiT side, $W_{\text{in}}^{\text{res}}$ and P_l, are trained with a separate learning rate.

Initialization. We initialize $\mathcal{E}_{\text{res}}$ and $\hat{\mathcal{D}}$ from the pretrained weights. The added downsample and upsample blocks are trained from scratch, and the input convolution of $\hat{\mathcal{D}}$ is zero-initialized on the residual channels. The gate of each first-frame cross-attention block is also zero, so the first frame has no effect at the start of training.

	Hyperparameter	Phase 1	Phase 2
Architecture	pretrained autoencoder	Wan2.1-VAE	Wan2.1-VAE
	(C, C')	$(16, 16)$	$(16, 16)$
	(r_s, r_t)	$(2, 2)$	$(2, 2)$
Training Setup	input shape	$256\times256\times81$	$512\times512\times81$
	trained modules	$\mathcal{E}_{\text{res}}$, $\hat{\mathcal{D}}$	$\hat{\mathcal{D}}$
	optimizer	AdamW	AdamW
	learning rate	8e-5	4e-5
	betas	$(0.9, 0.999)$	$(0.9, 0.999)$
	weight decay	1e-4	1e-4
	scheduler	constant	constant
	precision	bf16	bf16
	effective batch size	32	64
$\mathcal{L}_{\text{recon}}$	λ_{L1}	1.0	1.0
	λ_{LPIPS}	3.0	3.0
	λ_{KL}	3e-6	3e-6
$\mathcal{L}_{\text{align}}$	reference DiT	Wan2.1-14B	–
	alignment depth	10	–
	P_l bottleneck	16	–
	added-module learning rate	1e-4	–

Table A.1: **Stage 1 hyperparameters.** All settings follow the training recipe of the pretrained autoencoder.

Training objective. We follow the loss configuration of the pretrained autoencoder, with the weights listed in Tab. A.1. We apply $\mathcal{L}_{\text{align}}$ only in the first phase, since the second phase leaves the latent unchanged. Each P_l is a two-layer projection with a bottleneck of 16 channels, zero-initialized on the output so that the alignment starts from no contribution. We sample τ with the scheduler of the pretrained pipeline, as in Stage 2.

Single-latent Baseline. The baseline trains with $\mathcal{L}_{\text{recon}}$ alone, since there is no residual to shape and $\mathcal{L}_{\text{align}}$ never applies. The two phases and every hyperparameter in Tab. A.1 are the same.

B GRACE DIFFUSION MODEL DETAILS

B.1 ARCHITECTURAL DETAILS

We reuse the Wan2.1 DiT and modify only what the compressed latent requires.

Input and output projections. The pretrained model patchifies a latent of C channels, and for image-to-video it also takes a mask and the latent of the conditioning frame. We extend all three along the channel axis. The mask width is tied to the temporal compression factor, so it doubles with the latent, and we copy the pretrained weights into the channels it had before and zero-initialize the rest. The output projection is widened the same way, with the added channels zero-initialized, so the model starts from its pretrained behavior.

Position encoding. Compression places tokens farther apart in the video than the DiT was trained to expect. We therefore scale the RoPE (Su et al., 2023) indices by r_s in space and r_t in time. This restores the token spacing the DiT was pretrained with.

Timestep conditioning. The base and the residual latents are denoised at different noise levels, so the DiT receives two timesteps. For the transformer blocks, we add a second modulation projection $W^{\text{res}}_{\text{mod}}$ for the residual timestep, and the two modulations are summed. The two output heads are conditioned separately, each on its own timestep.

Single-latent Baseline. The baseline extends the input and output projections in the same way, but the latent is not split, so one timestep suffices and $W^{\text{res}}_{\text{mod}}$ is not added.

B.2 IMPLEMENTATION DETAILS

We adapt the pretrained DiT to the compressed latent while the autoencoder stays frozen. Before the latent enters the DiT, we standardize each part with statistics measured on the training set, following the pretrained pipeline. We fully fine-tune the input projection, the output heads, and $W^{\text{res}}_{\text{mod}}$, together with the modulation and normalization parameters of each block. LoRA (Hu et al., 2021) is applied to all linear layers in the attention and feed-forward blocks, as well as to the key and value projections of the image cross-attention for image-to-video.

Training setup. We train on Panda-70M (Chen et al., 2024) at $480 \times 832 \times 81$ with AdamW, using the hyperparameters in Tab. B.2. We apply the standard video preprocessing, filtering out clips with scene cuts or excessive motion, and rewrite the captions with GPT-4o (OpenAI et al., 2024) so that they match the length and detail of the prompts the pretrained model was trained on. Text conditioning is dropped with probability 0.1 for classifier-free guidance.

Initialization. We keep the pretrained weights for the base channels, and zero-initialize the weights for the residual channels along with $W^{\text{res}}_{\text{mod}}$.

	Hyperparameter	I2V	T2V
Architecture	pretrained DiT	Wan2.1-I2V-14B	Wan2.1-T2V-14B
	input dim	72	32
	hidden dim	5120	5120
	blocks	40	40
	num. heads	40	40
	patch size	2	2
	(C, C')	$(16, 16)$	$(16, 16)$
Training setup	input shape	$480\times832\times81$	$480\times832\times81$
	optimizer	AdamW	AdamW
	learning rate	1e-4	1e-4
	effective batch size	128	128
	offset δ	$[0, 0.6]$	$[0, 0.6]$
	shift s	$[2, 5]$	$[2, 5]$
LoRA	rank	512	512
	α	512	512
	image cross-attention	✓	×
Sampling	steps	50	50
	guidance scale	5.0	5.0
	offset δ	0.15	0.15
	shift s	3	3

Table B.2: **Stage 2 hyperparameters.** The offset and shift are sampled from a range during training and fixed at inference.

Denoising schedule. We sample the offset at each step rather than fixing it, so a single model covers a range of offsets at inference. The shift s of Eq. 1 is sampled the same way, and we find that a lower shift, which spends more sampling steps near the clean end, recovers detail better on the compressed latent. The values are listed in Tab. B.2.

Single-latent Baseline. The baseline follows the training recipe of Wan2.1, where a single timestep leaves no offset or shift to sample. The weights added to the projections are randomly initialized, following DC-Gen (He et al., 2026).

C More Details on Evaluation

C.1 Adaptation of Other Autoencoders

This section details how the pretrained DiT is adapted to each compared autoencoder in Tab. 1. Our own autoencoder is adapted as described in Section 4.3, which extends the pretrained input and output projections and denoises the base ahead of the residual. Both steps build on the base latent from the frozen Wan2.1 encoder, which is unavailable to the compared autoencoders, as their latent spaces are entirely different from the one the DiT was trained on. For each of them, we therefore adapt Wan2.1-I2V-14B with the released implementation of DC-Gen (He et al., 2026), a recent method that adapts a pretrained DiT to a new latent space in three stages: first the patch embedding, then the input and output layers, and finally the whole transformer with LoRA (Hu et al., 2021). Each autoencoder keeps the patch size of 1 used in its original generation model. We train every autoencoder with the same budget of 38.5 H200 GPU days that DC-Gen reports, sharing all other settings.

C.2 Evaluation Protocol

Autoencoder comparison. In Tab. 1, we use the Step-Video VAE v2 (Ma et al., 2025), the Video DC-AE of Open-Sora 2.0 (Zheng et al., 2026), and the LTX-Video VAE 0.9.7 (HaCohen et al., 2024), whose 13B release is closest in size to the generation backbone we compare against. Video DC-AE is evaluated with the spatial and temporal tiling of its released implementation, which uses spatial tiles of 256 pixels and temporal tiles of 32 frames with an overlap factor of 0.25, and reconstructs better than the untiled variant.

Generation comparison. Every model adapted from Wan2.1, including ours, the single-latent baseline, and the latents in Tab. 1, is sampled with 50 steps, a guidance scale of 5.0, a flow shift of 3.0, and seed 42, on the full prompt and conditioning image set of VBench-I2V (Huang et al., 2024). The other generators in Tab. 2 are run from their released checkpoints, the 13B development release of LTX-Video 0.9.7, the high-compression Video DC-AE release of Open-Sora 2.0, and the released DC-Gen model (He et al., 2026). We run LTX-Video as a single 50-step pass at the target resolution, without its multiscale pipeline and spatial upscaler, so that every model denoises the same number of steps at the same resolution. For text-to-video, each model is run with the prompt pipeline recommended by its authors: Wan2.1-14B, GRACE, and DC-Gen use the expanded prompts released with Wan2.1, LTX-Video uses its built-in prompt enhancer, and Open-Sora 2.0 follows its default text-to-image-to-video path without prompt refinement. Every model takes 50 sampling steps, but the number of forward passes through the DiT differs. LTX-Video combines classifier-free guidance with spatio-temporal guidance, and Open-Sora 2.0 guides on text and image separately, so both evaluate the DiT three times per step rather than twice. Latency covers the full pipeline, from encoding the inputs through denoising to decoding the output video, measured on a single NVIDIA A100 SXM4-80GB in PyTorch with bfloat16 and no inference-time compilation.

D Additional Evaluation

D.1 Video Generation

We report the VBench scores for each dimension in Tab. D.3a and Tab. D.3b. Three dimensions rank differently from the totals, and we discuss each below.

Aesthetic quality. DC-Gen scores above GRACE on this dimension in T2V settings, while scoring below on every total. We observe that its samples are consistently more saturated than those of the pretrained pipeline (Figs. I.7–I.10), and the LAION aesthetic predictor behind this dimension has been shown to track photographic taste rather than fidelity to the prompt or to the conditioning image (Taylor et al., 2026). The dimension therefore rewards a shift in appearance that the other dimensions penalize.

Motion smoothness and temporal flickering. In image-to-video at 480×832, LTX-Video leads on these two dimensions. It runs an extra guidance branch for temporal consistency, and in the image-to-video setting it also moves the least of all compared models, with a dynamic degree of 24.80 against 30.08 for GRACE. Videos that move less score higher on both dimensions (Huang et al., 2024).

Preserving the pretrained appearance. Given the same prompt and seed, GRACE stays close to the color and style of the pretrained pipeline in text-to-video, while DC-Gen often departs from it (Figs. I.9 and I.10). The effect is weaker in image-to-video, where the conditioning image fixes the appearance for every model.

We also evaluate at $736 \times 1280 \times 81$, above the resolution we train on. GRACE reaches the highest text-to-video total (84.74) while running $15.6\times$ faster than the pretrained pipeline, and stays within 0.21 of it on image-to-video (87.67 against 87.88). DC-Gen again leads on aesthetic quality in text-to-video while trailing on both totals. In image-to-video, LTX-Video keeps the highest subject consistency, while GRACE now leads on background consistency, motion smoothness, and temporal flickering.

Dimension	Wan2.1-14B	Open-Sora 2.0	LTX-Video 0.9.7	Wan2.1-14B + DC-Gen	Wan2.1-14B + Baseline	Wan2.1-14B + **GRACE (ours)**
Latent tokens	32.8k	8.2k	4.3k	8.2k	4.3k	4.3k
Latency (s) ↓	851.5	138.6	99.6	157.1	76.1	**75.8**
Quality						
Subject consistency	96.48	94.55	96.74	96.42	94.91	**96.76**
Background consistency	98.14	96.98	96.20	97.88	98.04	**98.70**
Temporal flickering	99.20	98.98	99.35	99.25	99.22	**99.42**
Motion smoothness	98.71	99.28	**99.45**	97.74	99.20	99.36
Dynamic degree	56.94	58.33	51.39	**70.83**	47.22	62.50
Aesthetic quality	70.41	56.09	60.62	**71.06**	64.77	68.66
Imaging quality	67.55	41.81	64.18	67.50	65.10	**67.65**
Semantic						
Object class	92.09	89.08	81.33	92.25	88.77	**94.15**
Multiple objects	79.50	61.28	40.62	84.68	72.94	**85.67**
Human action	97.00	98.00	86.00	97.00	98.00	**99.00**
Color	89.68	69.01	72.10	88.36	86.41	**97.29**
Spatial relationship	78.94	57.87	58.16	79.18	79.23	**91.16**
Scene	47.31	45.35	39.97	52.91	51.60	**57.49**
Appearance style	22.57	24.12	20.19	22.84	23.12	**24.79**
Temporal style	23.34	24.01	21.10	22.49	23.10	**25.52**
Overall consistency	25.63	25.60	23.08	25.39	24.48	**25.76**
Quality score	85.24	78.80	82.88	85.85	83.21	**86.02**
Semantic score	78.70	72.35	64.31	79.70	77.75	**84.98**
Total score	83.93	77.51	79.17	84.62	82.12	**85.81**

(a) VBench-T2V

Dimension	Wan2.1-14B	Open-Sora 2.0	LTX-Video 0.9.7	Wan2.1-14B + DC-Gen	Wan2.1-14B + Baseline	Wan2.1-14B + **GRACE (ours)**
Latent tokens	32.8k	8.2k	4.3k	8.2k	4.3k	4.3k
Latency (s) ↓	863.2	138.4	104.1	165.0	77.8	**77.7**
I2V						
Video-image subject fidelity	98.92	94.58	**98.97**	90.97	96.72	98.67
Video-image background fidelity	**99.51**	94.03	99.20	93.37	97.94	99.24
Camera motion	31.59	**58.85**	30.93	48.89	33.29	33.68
Quality						
Subject consistency	96.74	93.70	**97.87**	93.30	94.79	96.11
Background consistency	97.79	95.48	**98.39**	96.94	97.49	97.92
Motion smoothness	98.73	98.75	**99.52**	97.05	99.11	99.22
Dynamic degree	25.20	42.68	24.80	**59.76**	32.52	30.08
Aesthetic quality	**66.69**	57.17	64.59	61.60	62.34	65.54
Imaging quality	**71.10**	61.72	70.24	69.85	68.79	69.22
Temporal flickering[†]	98.04	97.72	**99.18**	95.75	98.43	98.23
I2V score	**95.82**	91.17	95.62	88.25	93.67	95.61
Quality score	80.01	76.96	**80.32**	80.01	79.22	80.11
Total score	87.92	84.07	**87.97**	84.13	86.44	87.86

(b) VBench-I2V

Table D.3: **Per-dimension VBench scores** at $480\times832\times81$. The best score in each row is in bold.

Dimension	Wan2.1-14B	Open-Sora 2.0	LTX-Video 0.9.7	Wan2.1-14B + DC-Gen	Wan2.1-14B + **GRACE (ours)**
Latent tokens	77.3k	19.3k	10.1k	19.3k	10.1k
Latency (s) ↓	3361.3	425.4	264.2	456.4	**215.6**
Quality					
Subject consistency	94.78	94.17	92.96	**96.72**	95.10
Background consistency	97.98	97.54	95.46	97.92	**98.48**
Temporal flickering	98.98	98.81	98.24	99.12	**99.43**
Motion smoothness	98.70	**99.18**	98.92	97.74	**99.18**
Dynamic degree	58.33	52.78	**88.89**	72.22	59.72
Aesthetic quality	68.16	58.52	59.89	**70.55**	67.85
Imaging quality	68.39	55.18	66.67	68.80	**68.81**
Semantic					
Object class	88.21	87.90	76.03	87.58	**90.27**
Multiple objects	77.29	69.97	34.53	**82.32**	78.96
Human action	97.00	**100.00**	86.00	96.00	**100.00**
Color	86.60	81.18	77.60	89.96	**90.29**
Spatial relationship	67.01	73.88	53.98	81.17	**81.51**
Scene	46.29	52.25	41.42	49.27	**55.67**
Appearance style	22.20	24.73	19.87	22.52	**24.74**
Temporal style	23.04	24.76	21.48	22.78	**25.71**
Overall consistency	25.73	**26.35**	23.72	25.78	**26.35**
Quality score	84.69	80.73	84.46	**86.08**	85.42
Semantic score	76.01	78.16	63.58	78.80	**82.04**
Total score	82.96	80.22	80.29	84.62	**84.74**

(a) VBench-T2V

Dimension	Wan2.1-14B	Open-Sora 2.0	LTX-Video 0.9.7	Wan2.1-14B + DC-Gen	Wan2.1-14B + **GRACE (ours)**
Latent tokens	77.3k	19.3k	10.1k	19.3k	10.1k
Latency (s) ↓	3396.8	425.4	274.6	550.7‡	**218.8**
I2V					
Video-image subject fidelity	98.55	96.65	**98.88**	94.62	98.44
Video-image background fidelity	99.22	97.09	98.95	96.50	**99.29**
Camera motion	34.34	**46.66**	37.35	43.25	32.11
Quality					
Subject consistency	95.03	95.17	**96.76**	94.07	96.15
Background consistency	96.69	96.71	96.88	97.62	**98.04**
Motion smoothness	98.22	98.96	99.23	97.83	**99.24**
Dynamic degree	41.87	26.42	49.59	**57.72**	26.83
Aesthetic quality	**65.30**	60.03	64.07	61.67	65.00
Imaging quality	70.44	67.61	**70.78**	70.26	69.93
Temporal flickering†	96.92	98.04	97.90	96.70	**98.55**
I2V score	95.56	93.72	**95.71**	92.04	95.44
Quality score	80.20	77.82	**81.79**	80.73	79.90
Total score	87.88	85.77	**88.75**	86.39	87.67

(b) VBench-I2V

Table D.4: **Per-dimension VBench scores** at $736\times1280\times81$, following Tab. D.3. The best score in each row is in bold. ‡Measured with spatial tiling in the VAE, without which encoding the conditioning image runs out of memory at this resolution.

D.2 HUMAN EVALUATION

Task	Aspect	vs. Wan2.1 (Wan et al., 2025) (%)			vs. DC-Gen (He et al., 2026) (%)		
		Ours	Equal	Wan2.1	Ours	Equal	DC-Gen
T2V	Visual quality	49.4	15.4	35.3	71.2	11.5	17.3
	Temporal consistency	46.8	21.2	32.1	64.7	21.2	14.1
	Text alignment	52.6	21.8	25.6	62.2	23.1	14.7
I2V	Visual quality	23.7	32.7	43.6	60.3	16.7	23.1
	Temporal consistency	29.5	25.6	44.9	64.1	11.5	24.4
	Text alignment	22.4	42.9	34.6	46.8	29.5	23.7

Table D.5: **Human evaluation.** Participants compared each pair of videos without knowing which model produced which. Each cell gives the percentage of votes preferring our model (Ours), rating both videos about equal (Equal), or preferring the baseline named in the column.

We conduct a blind user study comparing our method with Wan2.1 (Wan et al., 2025) and the adaptation-based DC-Gen (He et al., 2026) in both the T2V and I2V settings, using 40 prompts randomly selected per setting from VBench (Huang et al., 2023; 2024) (with their conditioning images for I2V). Each model uses its own prompt extension; participants see the original VBench prompt. Participants view two videos generated from the same prompt, ours and one baseline, in randomized A/B order, and judge which is better in visual quality (fewer unnatural colors or shapes and a better overall appearance), temporal consistency (no flicker, stutter, or objects changing over time), and text alignment (the subjects, actions, details, and style described in the prompt), with an "about equal" option for each. 39 human raters each rated 16 pairs (8 T2V, 8 I2V), so each of the 160 comparison pairs is rated by at least 3 different participants, giving 156 votes per setting, baseline, and criterion. Tab. D.5 reports the win, tie, and loss rates of our method.

Against DC-Gen, which also operates in a compressed latent space, participants prefer our model by a wide margin in both settings (60.3–71.2% vs. 14.1–24.4% for visual quality and temporal consistency). Against the uncompressed Wan2.1, our model is preferred in T2V on all three criteria (46.8–52.6% vs. 25.6–35.3%), whereas Wan2.1 is preferred in I2V (34.6–44.9% vs. 22.4–29.5%), with many votes rating the two about equal (up to 42.9% for text alignment). This pattern is consistent with the VBench results in Tab. D.3 and Tab. D.4. Figs. D.2 and D.3 show the interface used in the study.

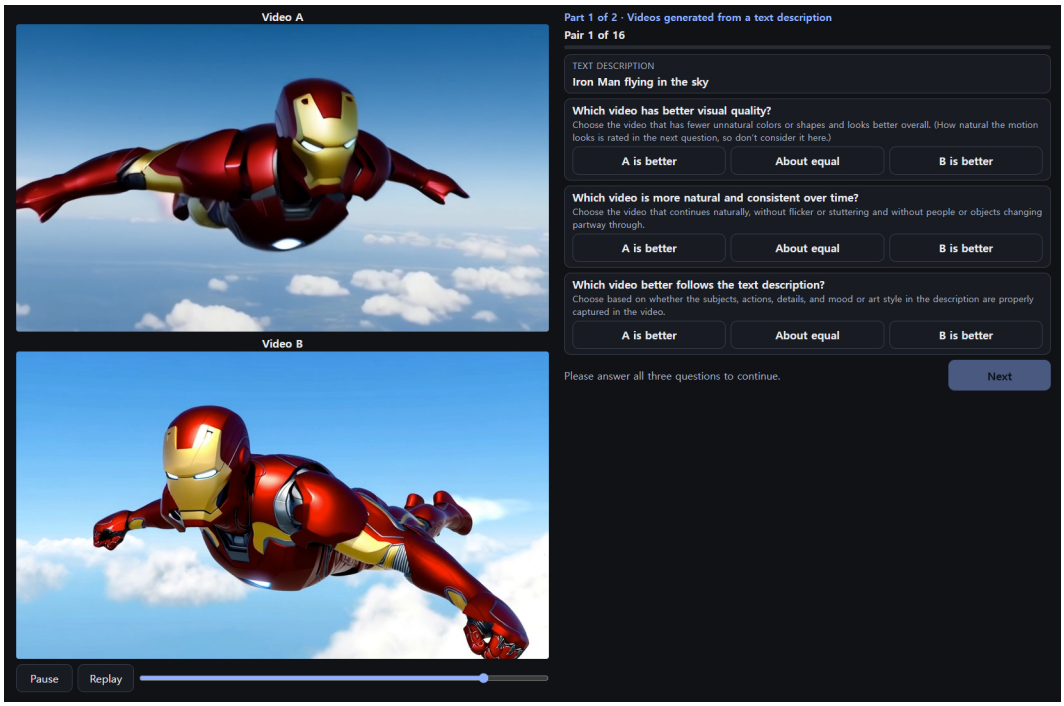


Figure D.2: **User study interface for T2V samples.**

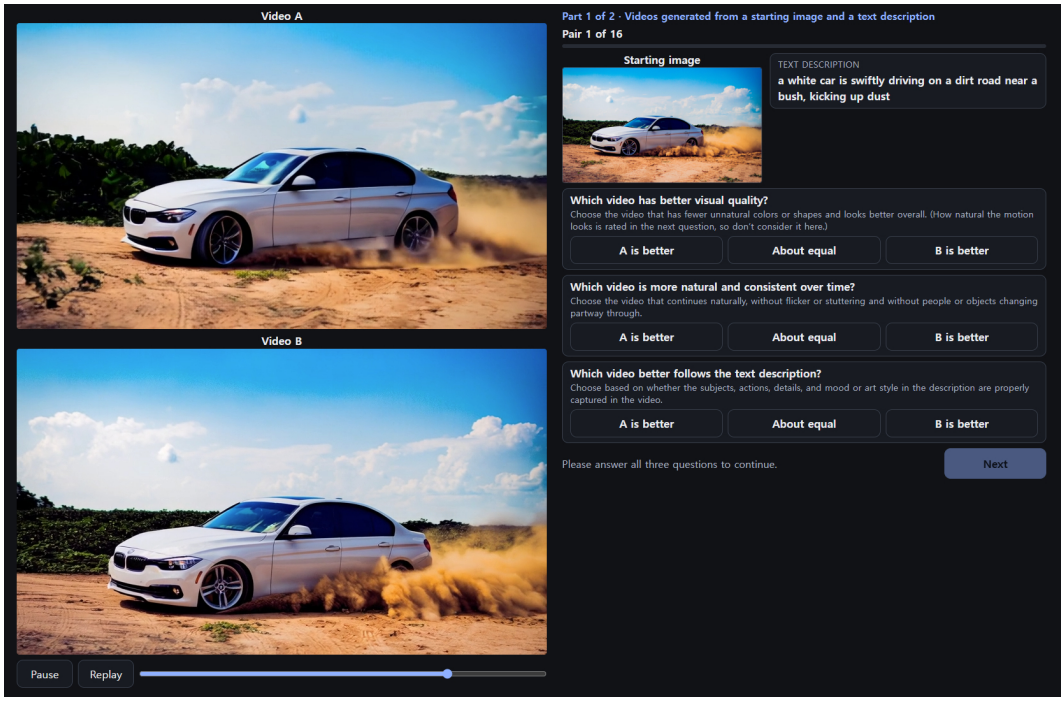


Figure D.3: **User study interface for I2V samples.**

E Additional Ablations

E.1 Autoencoder Design

We examine the design choices behind the autoencoder that are not covered in Appendix A.

First-frame conditioning. For image-to-video, $\hat{\mathcal{D}}$ receives the first frame through gated cross-attention. We drop it with probability 0.5 during training, so the decoder learns to reconstruct without it. Tab. E.6 removes it at inference as well: reconstruction stays close to the conditioned setting, and the dropout also improves the conditioned result, since the decoder is forced to use the compressed latent rather than copying from the first frame.

First frame	PSNR ↑	SSIM ↑	LPIPS ↓
×	31.97	0.928	0.038
✓	32.63	0.930	0.032

Table E.6: **Ablation on first-frame conditioning.**

E.2 Alignment Design

We examine the design choices behind $\mathcal{L}_{\text{align}}$, the alignment applied in Stage 1.

Alignment depth. We compare supervising the first 10 blocks of the DiT with supervising all 40. The two settings generate at the same level (87.86 against 87.94 on the I2V total), while supervising all 40 reconstructs worse (32.63 to 32.21 dB, with LPIPS rising from 0.032 to 0.038) and converges more slowly. The later blocks conflict with $\mathcal{L}_{\text{recon}}$ more strongly, so the encoder spends its capacity on matching them rather than on reconstruction, and we supervise the first 10.

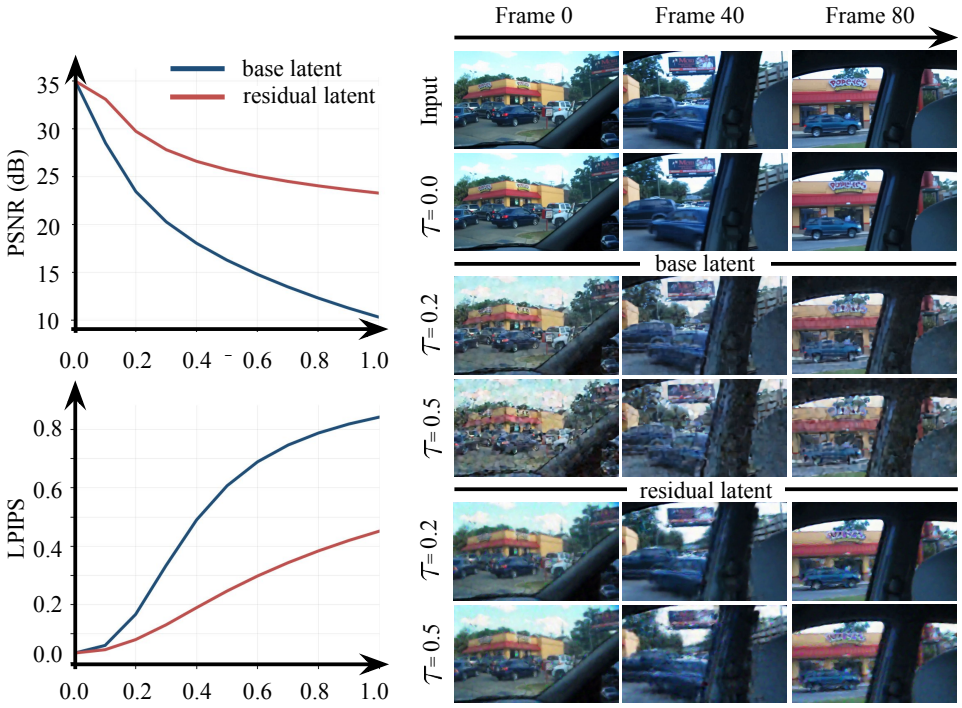
Depth	Reconstruction			VBench-I2V		
	PSNR ↑	SSIM ↑	LPIPS ↓	I2V ↑	Quality ↑	Total ↑
10	32.63	0.930	0.032	95.61	80.11	87.86
40	32.21	0.929	0.038	95.70	80.19	87.94

Table E.7: **Ablation on alignment depth.**

F Additional Analysis

F.1 Latent Analysis

In this section, we look into what the base and the residual latent each contain, first by perturbing one part while the other stays clean and then through their principal components.



(a) Quantitative comparison across τ (b) Qualitative comparison across τ

Figure F.4: **Reconstruction under latent noise.** We noise $\mathbf{z}_{\text{base}}$ or $\mathbf{z}_{\text{res}}$ at level τ, leave the other unchanged, and reconstruct. **(a)** Reconstruction quality on Panda-70M (Chen et al., 2024) at $480 \times 832 \times 81$ as τ grows, measured in PSNR and LPIPS. **(b)** Reconstructed frames at two levels of τ, shown below the input and the noise-free reconstruction, with the first block noising $\mathbf{z}_{\text{base}}$ and the second noising $\mathbf{z}_{\text{res}}$.

Per-latent analysis on reconstruction. We analyze how $\mathbf{z}_{\text{base}}$ and $\mathbf{z}_{\text{res}}$ each affect the reconstruction performance by injecting a controlled amount of noise into one of them while keeping the other clean. Fig. F.4 sweeps τ from 0 to 1 in the flow matching interpolation of Eq. 2, so that $\tau = 0$ leaves the latent unchanged and $\tau = 1$ replaces it with pure noise. The noise is scaled by the per-channel variance of the latent it is added to. We report PSNR and LPIPS on Panda-70M (Chen et al., 2024) at $480 \times 832 \times 81$ for each level.

As plotted in Fig. F.4 (a), both PSNR and LPIPS degrade steadily for either latent under noise, and faster for $\mathbf{z}_{\text{base}}$ across the whole range. Fig. F.4 (b) also shows that both $\mathbf{z}_{\text{base}}$ and $\mathbf{z}_{\text{res}}$ fail in different ways, with $\mathbf{z}_{\text{base}}$ breaking the spatial layout of the scene while $\mathbf{z}_{\text{res}}$ keeps the layout and instead blurs each frame and makes the motion appear at a lower frame rate. This is expected from the design, since the reduction in space and time leaves the overall scene in $\mathbf{z}_{\text{base}}$ while $\mathbf{z}_{\text{res}}$ carries what the reduction removes and the motion between the sampled frames. The larger drop for $\mathbf{z}_{\text{base}}$ follows as well, and it is what we intend: $\mathbf{z}_{\text{base}}$ stays in the space the DiT was pretrained on, so the latent that carries more of the reconstruction is also the one that needs the least adaptation. The burden of Stage 2 therefore falls on $\mathbf{z}_{\text{res}}$, which lies outside the space the pretrained DiT was trained on. A video built from $\mathbf{z}_{\text{base}}$ alone would show blurred frames and motion that steps between the sampled frames rather than flowing through them, precisely what $\mathbf{z}_{\text{res}}$ was trained to restore.

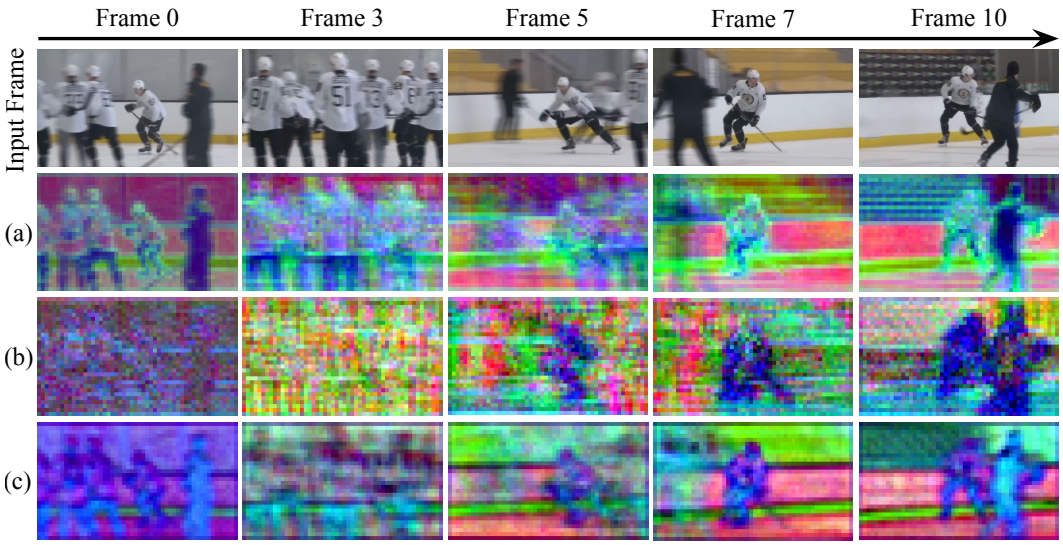


Figure F.5: **PCA visualization of the base and residual latents.** Principal Component Analysis (PCA) of each part at f16t8, encoded from a $480 \times 832 \times 81$ video, for **(a)** $\mathbf{z}_{\text{base}}$, **(b)** $\mathbf{z}_{\text{res}}$ without $\mathcal{L}_{\text{align}}$, and **(c)** $\mathbf{z}_{\text{res}}$ with $\mathcal{L}_{\text{align}}$. Since $\mathcal{E}$ is frozen, $\mathbf{z}_{\text{base}}$ is the same in both settings.

PCA visualization. Fig. F.5 visualizes the Principal Component Analysis (PCA) of $\mathbf{z}_{\text{base}}$ and $\mathbf{z}_{\text{res}}$ across video frames, encoded from a $480 \times 832 \times 81$ video. The three leading components are computed per clip and mapped to RGB, so colours are not comparable across panels. The components of $\mathbf{z}_{\text{base}}$ are semantically organized, following the objects in the frame. Without $\mathcal{L}_{\text{align}}$, noise dominates the components of $\mathbf{z}_{\text{res}}$, which show little of the scene. $\mathcal{L}_{\text{align}}$ suppresses much of that noise and brings out the objects, and $\mathbf{z}_{\text{res}}$ occasionally resolves them at an even coarser scale than $\mathbf{z}_{\text{base}}$. Note that $\mathbf{z}_{\text{base}}$ is identical in both settings, since $\mathcal{E}$ is frozen. More samples are in Appendix I.

F.2 Convergence Behavior

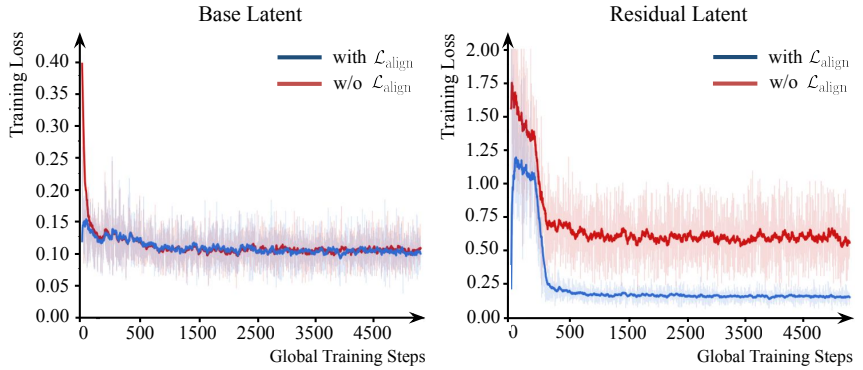


Figure F.6: **Convergence of the base and the residual.** Flow matching loss on the base channels (left) and the residual channels (right) during DiT adaptation, for the dual latent with and without $\mathcal{L}_{\text{align}}$. Each part is standardized with its own statistics.

Fig. 6 tracks the flow matching loss on the residual channels during DiT adaptation. Fig. F.6 adds the base channels of the same runs, trained under identical settings except for $\mathcal{L}_{\text{align}}$, with each part standardized by its own statistics. The base channels behave almost identically with and without alignment, since the DiT already models that space and has little to adapt. The two runs separate only on the residual, which converges faster and reaches a lower loss with alignment, so the gain in Fig. 6 does not come at the cost of the base.

G Computational Cost

		T2V		I2V	
		Wan2.1-14B	Ours	Wan2.1-14B	Ours
Training	Stage 1, full autoencoder	–	6.9	–	6.9
	Stage 1, decoder-only (+EMA)	–	1.6	–	1.6
	Stage 2	–	30	–	30
	Total (H200 GPU days)	–	38.5	–	38.5
Latency (s) ↓		851.5	**75.8**	863.2	**77.6**

Table G.8: **Computational cost.** Training in H200 GPU days, and inference at $480 \times 832 \times 81$ on a single A100 with 50 sampling steps.

H More Discussion on Related Work

Improving the latent space for generation. Beyond compression, several works improve the latent space itself so that the diffusion model learns it more easily (Zheng et al., 2025; Shin et al., 2026; Chen et al., 2025c; Wu et al., 2024). These works mostly train the generation model from scratch on the resulting latent, whereas we shape the latent so that an already trained model can be reused.

Efficient video autoencoder architectures. Compared to images, video carries an additional temporal dimension, which makes the autoencoder both harder to design and more expensive to run. Wavelet-based designs reduce this cost by replacing part of the convolutional stack with a fixed multi-resolution transform (Li et al., 2025; Cheng & Yuan, 2025; NVIDIA et al., 2025). These reduce the cost of the autoencoder itself rather than the token count the diffusion model processes.

I Additional Qualitative Results

This section collects the qualitative samples referenced throughout the paper: image-to-video and text-to-video generation at 480×832 (Figs. I.7–I.10), generation at 736×1280 (Figs. I.11 and I.12), and further latent visualizations (Fig. I.15).

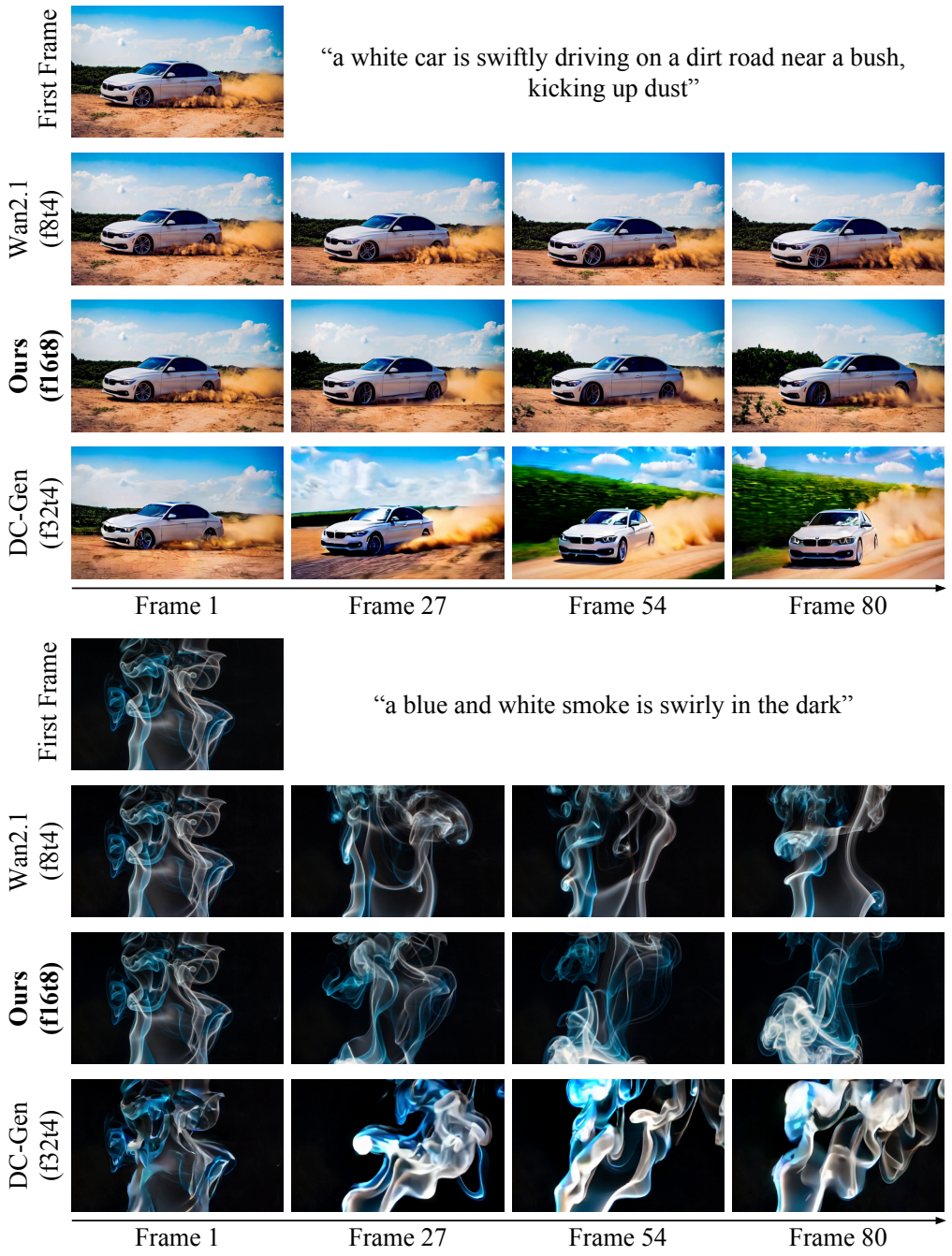


Figure I.7: **Additional image-to-video results.** Samples from VBench-I2V Huang et al. (2024) at 480 × 832 × 81.

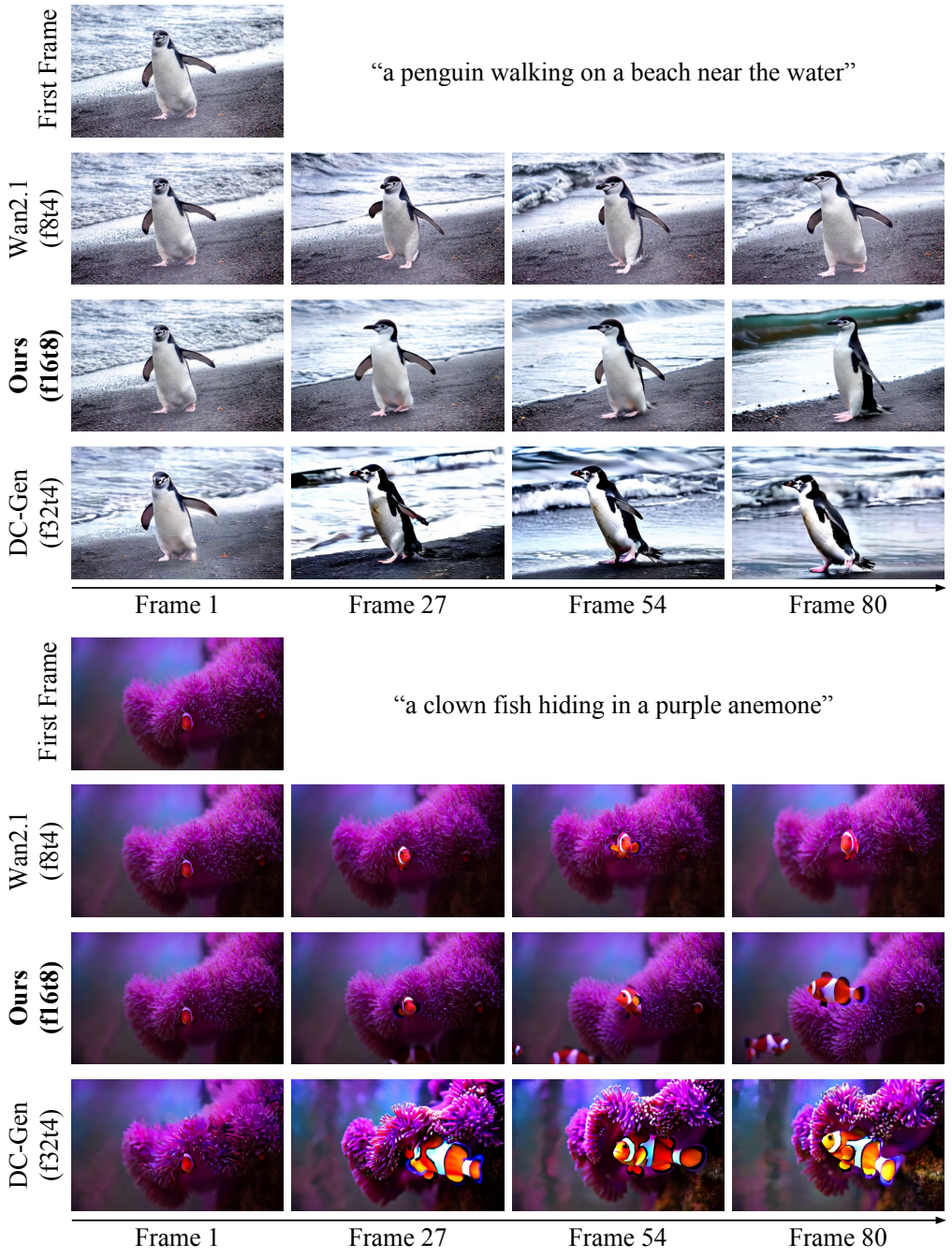


Figure I.8: **Additional image-to-video results.** Samples from VBench-I2V Huang et al. (2024) at $480 \times 832 \times 81$.

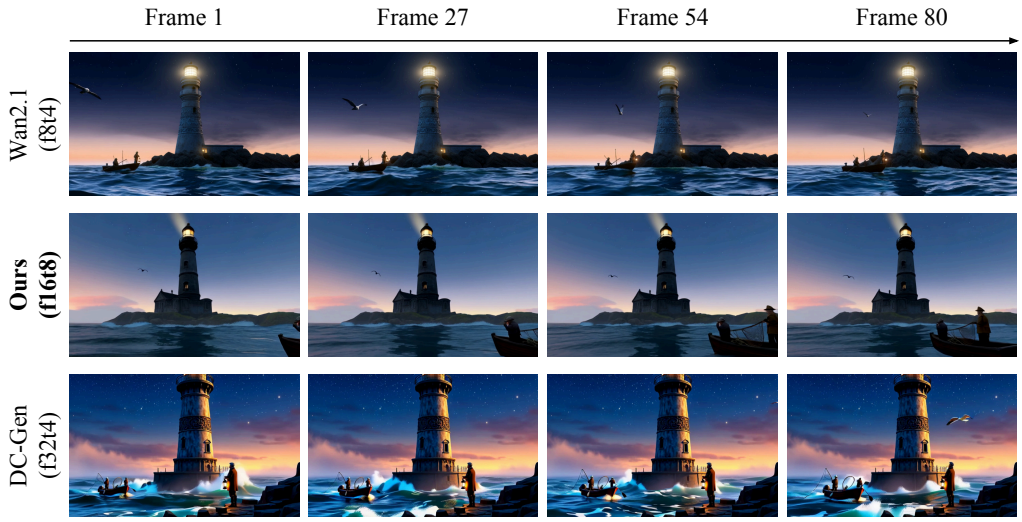


Figure I.9: **Additional text-to-video results.** Samples from VBench Huang et al. (2023) at $480 \times 832 \times 81$.

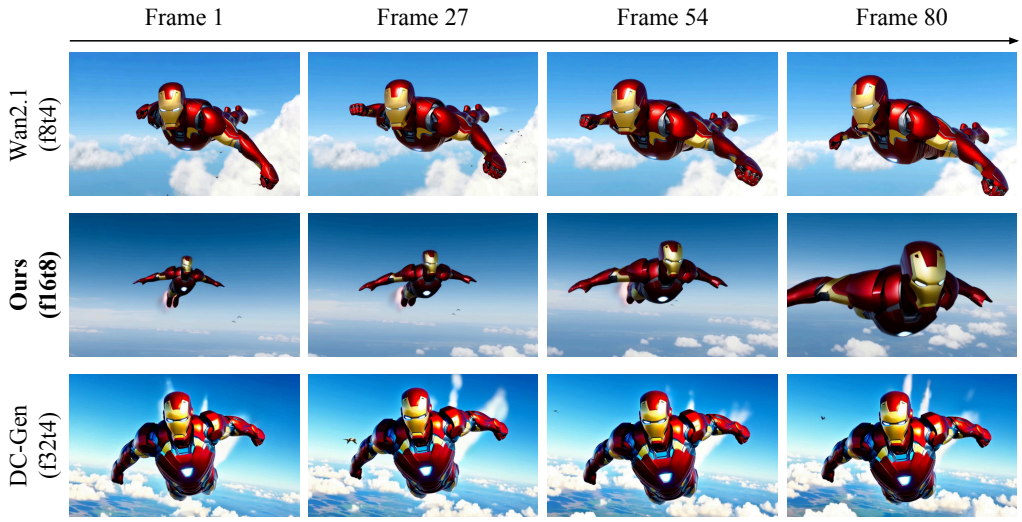


Figure I.10: **Additional text-to-video results.** Samples from VBench Huang et al. (2023) at $480 \times 832 \times 81$.



Figure I.11: **Generation at** $736 \times 1280 \times 81$**.** GRACE is not trained at this resolution.

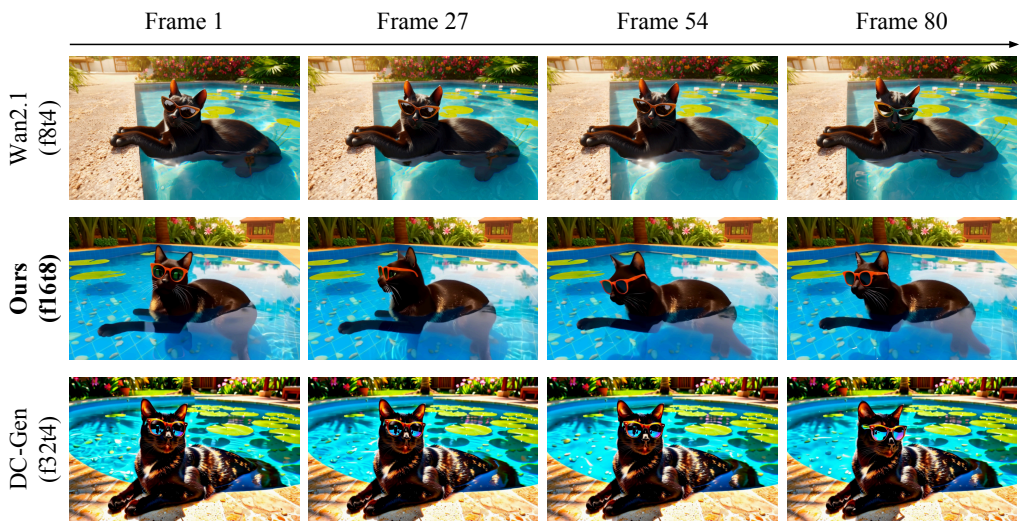


Figure I.12: **Generation at** $736 \times 1280 \times 81$**.** GRACE is not trained at this resolution.

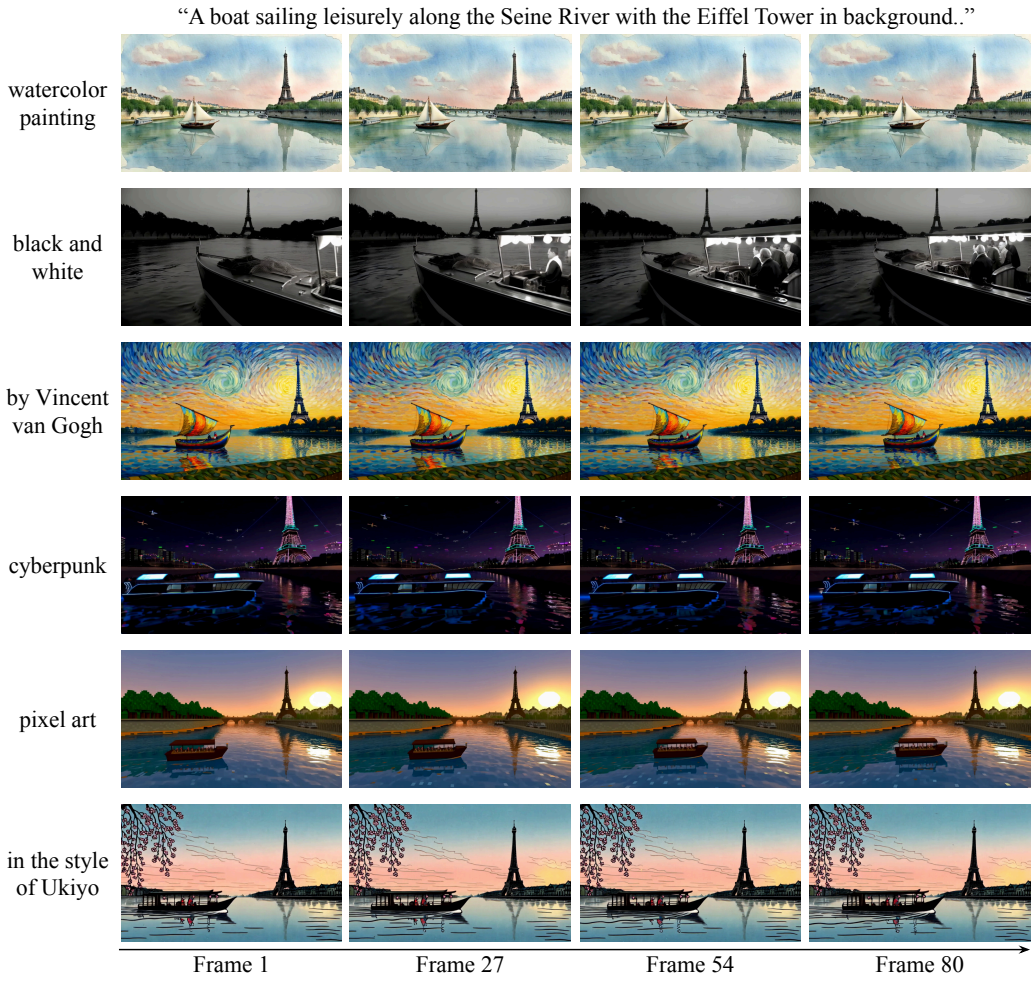


Figure I.13: **Generation in various styles.** Generated at $736 \times 1280 \times 81$.

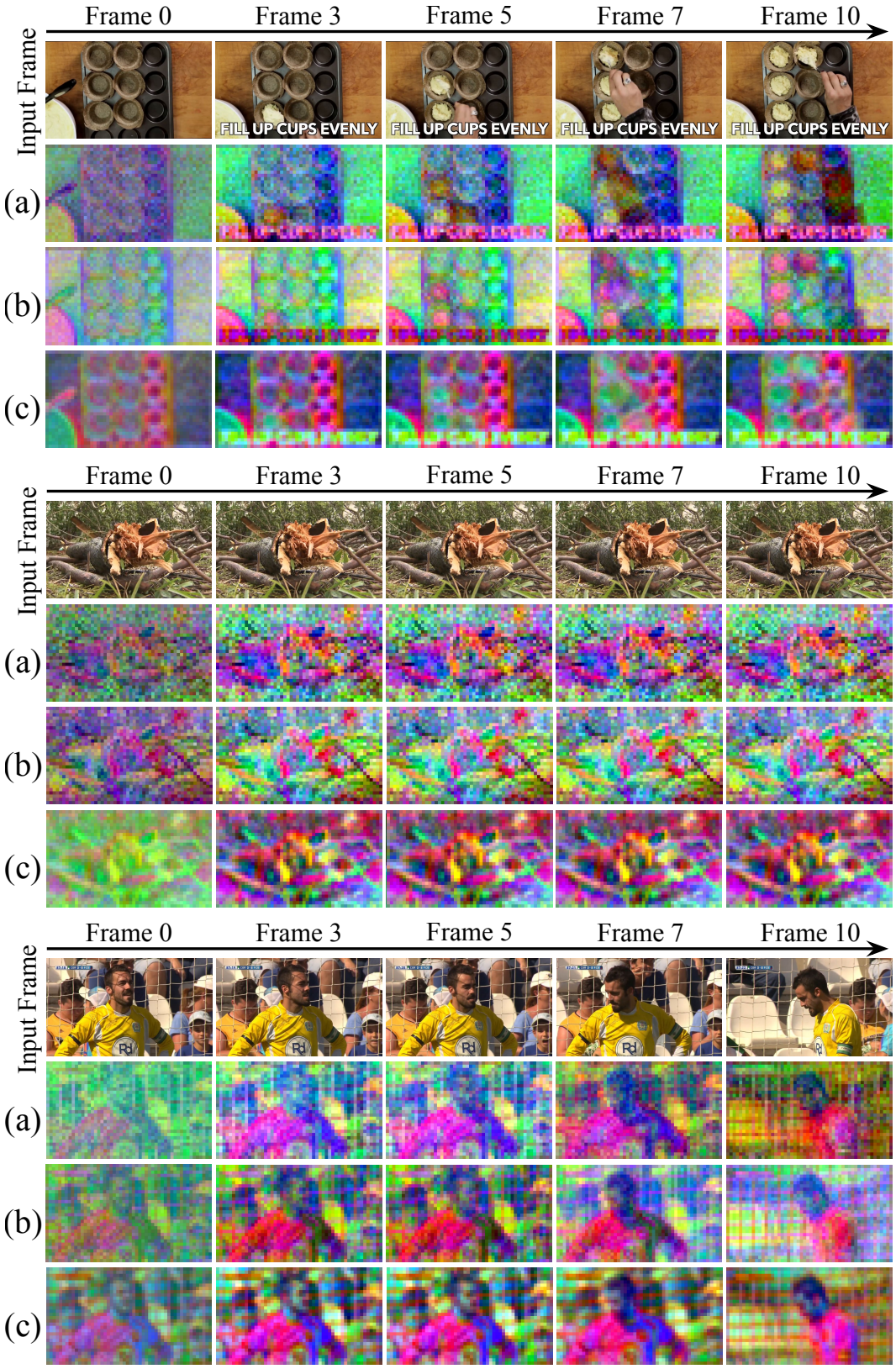


Figure I.14: **Additional latent PCA visualizations.** Principal components of the full $C+C'$ latent at f16t8, for **(a)** a single encoder, **(b)** ours without $\mathcal{L}_{\text{align}}$, and **(c)** ours, with the input frames on top. The three leading components are computed per clip and mapped to RGB, so colours are not comparable across panels.

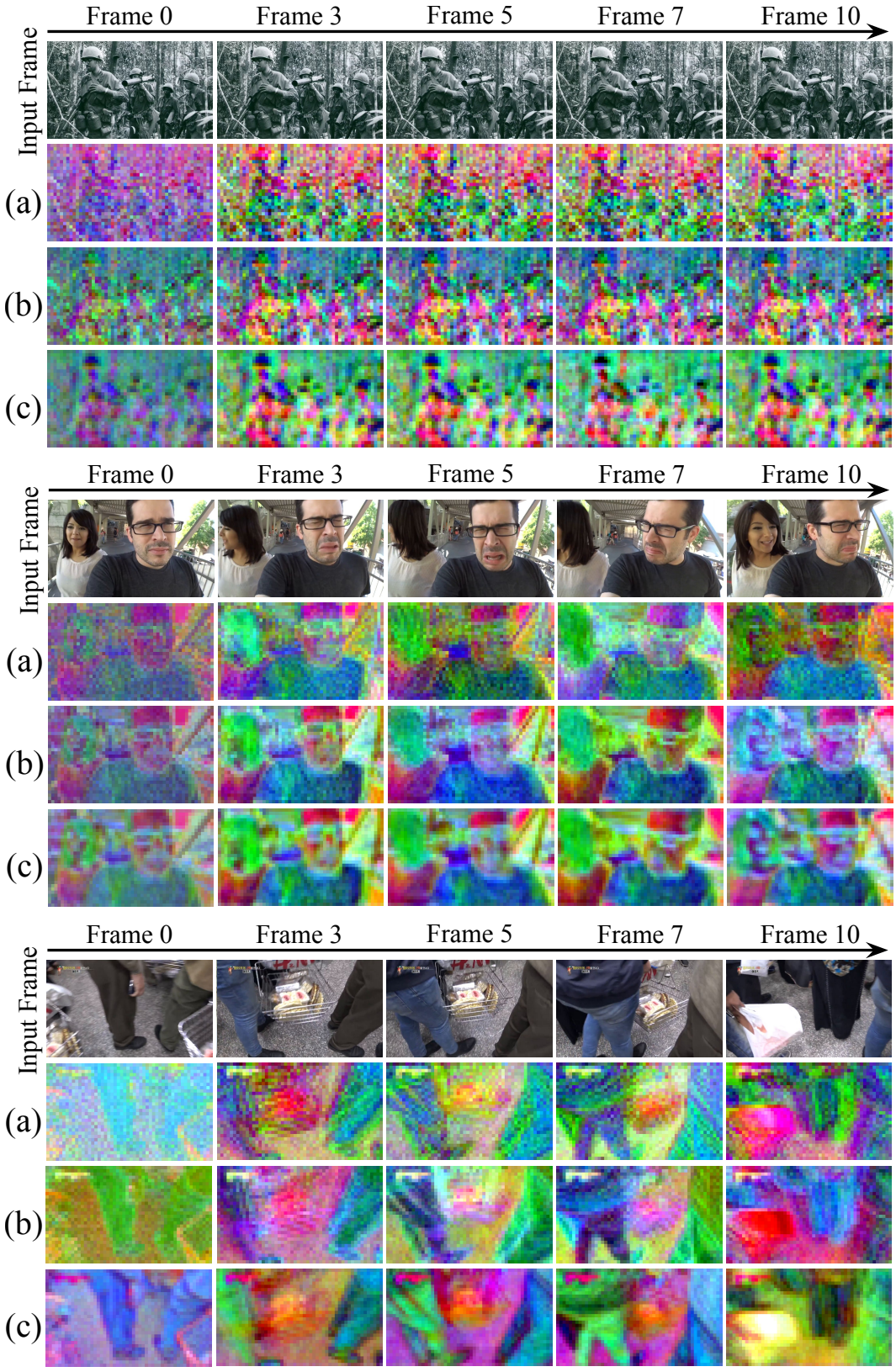


Figure I.15: **Additional latent PCA visualizations.** Principal components of the full $C+C'$ latent at f16t8, for **(a)** a single encoder, **(b)** ours without $\mathcal{L}_{\text{align}}$, and **(c)** ours, with the input frames on top. The three leading components are computed per clip and mapped to RGB, so colours are not comparable across panels.

J LIMITATIONS

Compressing the latent by $8\times$ leaves fewer tokens for each frame, and small objects suffer the most. Distant faces, text on signs, and thin structures are often lost or broken, while the overall scene and its motion stay intact. The effect shows in Figs. I.9 and I.10, and it is weaker at 736×1280, where the same compression ratio leaves more tokens per frame.