Progressive Growing of Video Tokenizers for Temporally Compact Latent Spaces

Aniruddha Mahapatra[1] Long Mai[1] David Bourgin[1] Yitian Zhang[1,2] Feng Liu[1]
[1]Adobe Research [2]Northeastern University
https://progressive-video-tokenizer.github.io/Pro-MAG/

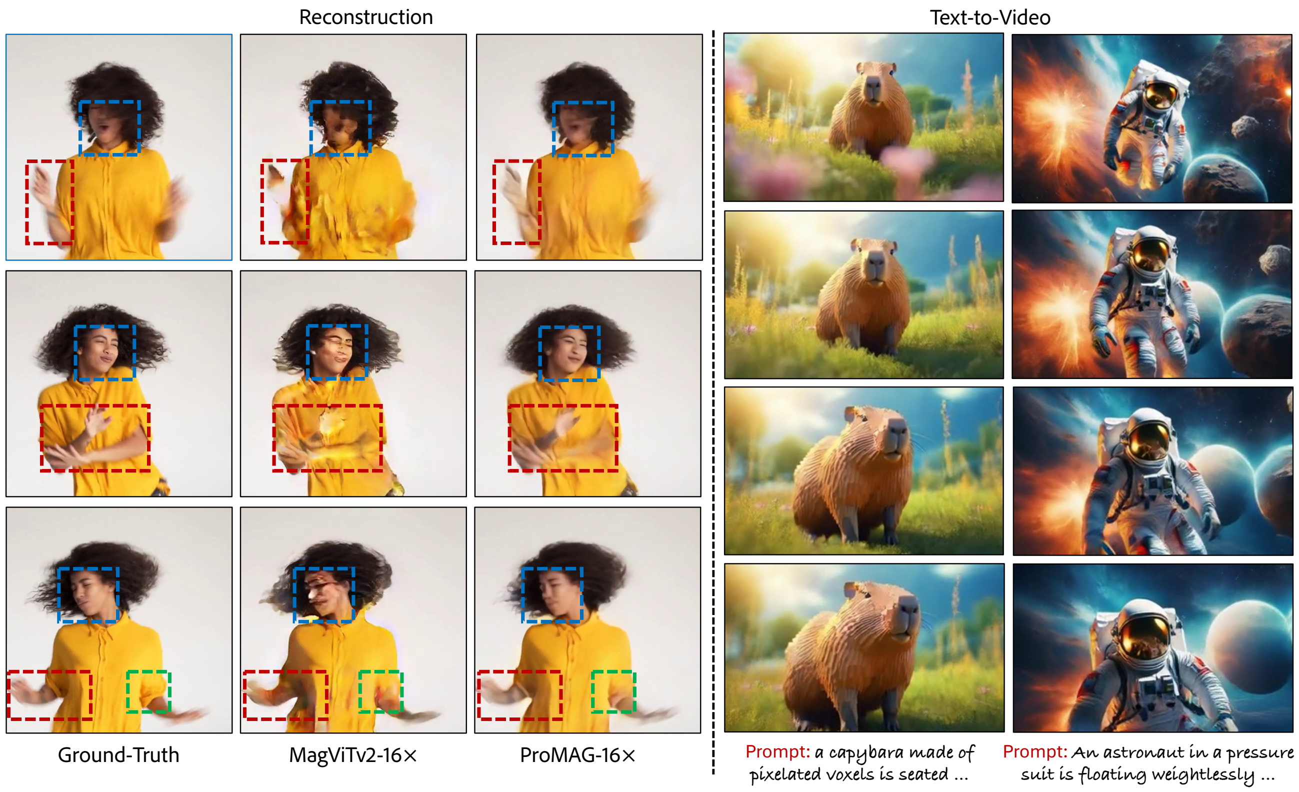

Reconstruction
Text-to-Video
Ground-Truth
MagViTv2-16×
ProMAG-16×
Prompt: a capybara made of pixelated voxels is seated ...
Prompt: An astronaut in a pressure suit is floating weightlessly ...

Figure 1. (left) Our video tokenizer ProMAG is capable of much more effectively reconstructing high motion videos even at very large (16×) temporal compression, compared to baseline MagViT-v2. Dotted boxes highlight regions where the MagViT-v2 fails catastrophically, like faces and hands, which have a lot of artifacts in the reconstructions. (right) We show that our highly compressed latent space (16×) is suitable for training high-quality text-to-video diffusion models.

Abstract

Video tokenizers are essential for latent video diffusion models, converting raw video data into spatiotemporally compressed latent spaces for efficient training. However, extending state-of-the-art video tokenizers to achieve a temporal compression ratio beyond 4× without increasing channel capacity poses significant challenges. In this work, we propose an alternative approach to enhance temporal compression. We find that the reconstruction quality of temporally subsampled videos from a low-compression encoder surpasses that of high-compression encoders applied to original videos. This indicates that high-compression models can leverage representations from lower-compression models. Building on this insight, we develop a bootstrapped high-temporal-compression model that progressively trains high-compression blocks atop well-trained lower-compression models. Our method includes a cross-level feature-mixing module to retain information from the pretrained low-compression model and guide higher-compression blocks to capture the remaining details from the full video sequence. Evaluation of video benchmarks shows that our method significantly improves reconstruction quality while increasing temporal compression compared to directly training the full model. Furthermore, the resulting compact latent space effectively trains a video diffusion model for high-quality video generation with a significantly reduced token budget.

1. Introduction

The emergence of diffusion models [16, 39] has transformed image [10, 30, 33, 35, 36] and video [5, 6, 11, 15, 17, 38, 46] generation. These models gradually transform random noise into coherent visual outputs, showcasing impressive capabilities in producing high-fidelity content [6, 9]. Latent diffusion models (LDMs)[35] have gained popularity for image and video generation[2, 36]. Unlike pixel diffusion models that operate directly on raw pixels, LDMs project pixels into a low-dimensional latent space using a variational autoencoder (VAE) [24], enabling diffusion in this compact space. This dimensionality reduction enhances computational efficiency, which is particularly crucial for video generation due to the higher dimensionality of video data compared to images.

Early latent video diffusion models (LVDMs) relied on latents extracted from each frame independently [3, 15, 38] by reusing the same VAE from image LDMs. This neglects the temporal dynamics inherent in video data, compromising the temporal consistency of the generated videos, and offers no temporal compression in the latent space. To address these limitations, the seminal work MagViT-v2 [47] pioneers a spatiotemporal video tokenizer that can jointly encode images and videos in the same latent space. Originally designed to encode video into discrete tokens, MagViT-v2 has been widely adapted to continuous tokens for use in many state-of-the-art LVDMs [14, 26, 46, 51]. In the context of this work, we use MagViT-v2 as a continuous tokenizer.

State-of-the-art video diffusion models (VDMs) utilize diffusion transformers [30] as their backbone architectures, with computational costs scaling quadratically with input token lengths. Increasing latent space compression by a factor of two in both spatial and temporal dimensions significantly enhances model efficiency. While MagViT-v2 and other video tokenizers achieve substantial spatial compression ($8\times$), their temporal compression remains limited to $4\times$. This work focuses on improving temporal compression while maintaining spatial compression at $8\times$. We found that extending MagViT-v2 by adding more temporal down/upsampling layers to boost temporal compression negatively impacts reconstruction quality. This underscores a key challenge in adapting convolutional video tokenizers for higher compression rates: training such models from scratch necessitates simultaneous optimization of all parameters for high compression, rather than employing a hierarchical approach where different components specialize in varying levels of compression.

Interestingly, we observe that $4\times$ temporal compression MagViT-v2 can accurately reconstruct videos with $4\times$ frame subsampling (i.e., low FPS), much better than a $16\times$ temporal compression model on the original (high FPS) video. Thus, we try to answer the question: *"Can we meaningfully boost the temporal compression of video tokenizers by reusing a pretrained $4\times$ model, **while keeping the number of latent channels constant**?"* In this paper, we intentionally keep the latent channel dimension fixed over the progressive growing process for two reasons. First, it helps separate the effects of channel dimensions and progressive growing on the model performance. Second, one of our main goals is to achieve good latent features not only for high reconstruction quality but also for diffusion model training. It has been observed that increasing the latent channel dimensions, while leading to a slight improvement in reconstruction, tends to make generative model training more difficult [47].

This paper makes the following main contributions:

- We adopt the continuous-token MagViT-v2 architecture as our base model and make several modifications to make it more efficient and amenable to our method of growing the model from $4\times$ to $8\times$ and $16\times$ compression. The modifications preserve the quality of the base $4\times$ MagViT-v2. We call our model **ProMAG** (abbreviation for **Pro**gressive growing of **Mag**ViT-v2 for continuous-space tokens).
- We present our progressive model growing technique to train ProMAG for high temporal compression ratios, achieving up to $8\times$ and even $16\times$. Notably, we are the first to achieve high-quality reconstruction with a $16\times$ temporal compression model. Evaluations on video benchmarks show that ProMAG delivers significantly better reconstruction quality than simply extending existing tokenizers for higher compression. Our method also consumes $\sim 2.7\times$ less training time compared to full model training.
- We showcase the effectiveness of our highly compressed latents for text-to-video generation with DiT. Experiments on the text-to-video evaluation benchmark [19] reveal that DiT trained with our $16\times$ temporal-compression latents achieves comparable or slightly higher quality than the standard $4\times$ compression, while significantly enhancing efficiency and token length.

2. Related work

Latent video diffusion models. The pioneering LVDMs [3, 4, 11, 15, 29, 38] repurposed image VAEs to transform each frame of the video independently into the latent space. This caused temporal flickering in the generated video and limited the video length due to the limited compression achieved by image VAEs. More recent works [6, 14, 26, 46, 51] use VAEs that operate on video-level, hence mitigating both limitations of using image VAEs for video diffusion models. Building upon this, we aim to create a VAE with even higher temporal compression ($8\times$ and $16\times$) compared to the contemporary $4\times$, enabling highly efficient and longer video generation.

High compression VAEs. There have been few works recently that investigate very high compression in latent space for higher-resolution image and longer video generation. UltraPixel [34] provides an image autoencoder that can perform $24\times$ spatial compression. Very recently, DC-AE [8] provide a solution to increase the spatial compression to as large as

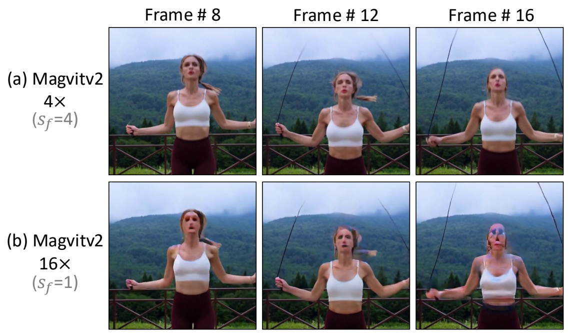


Figure 2. **Motivation.** We highlight the key motivation of our progressive growing approach. Directly training MagViT-v2 for 16× temporal compression leads to poor reconstruction quality for a 24-fps video (bottom). s_f stands for frame subsampling factor. However, we observed that the 4× temporal compression model can still accurately reconstruct a 6-fps video by feeding the same 24-fps video after subsampling frames by a factor of 4×, $s_f = 4$ (top). This observation implies that *it is not necessarily the large motion that leads to worse reconstruction, but that training many downsampling (upsampling) layers of encoder (decoder) at once makes training difficult*, as also observed in DC-AE [8].

128× spatially. However, they also have to increase channel dimension consistently to preserve reconstruction quality. MovieGen [31] and Cosmos [1] also achieve 8× temporal compression by making modifications to the tokenizer architecture. In comparison, we tackle a different problem in this work. Starting from a well-trained 4× temporal compression model, our goal is to create a method to increase the temporal compression while keeping the number of latent channels fixed to avoid making the latent space harder to learn for the diffusion model. In this work, we keep the spatial compression fixed at 8×. To the best of our knowledge, we are also the first method to achieve 16× temporal compression.

Hierarchical models. Hierarchical generation has been explored most notably using generative adversarial networks (GANs) [13] to perform high-resolution image generation in stages [20, 21, 45]. These approaches first learn a generator that operates in low resolution, followed by learning additional high-resolution blocks on top in stages. Similar ideas have also been explored in image and video generation using diffusion models [2, 10, 17, 34], where content is first generated in low resolution and then upsampled in the spatial and temporal domain using a separate model. In our work, we are inspired by ProGAN [20] and adapt the progressive learning idea to our problem of boosting the temporal compression in video tokenizers, progressively learning separate model blocks to handle different levels of compression.

3. Method

Our goal is to create a continuous space video tokenizer with high temporal compression ratios (8× or 16×) and good reconstruction quality. In Section 3.1, we discuss the modifications we make to the MagViT-v2 [47] to arrive at our

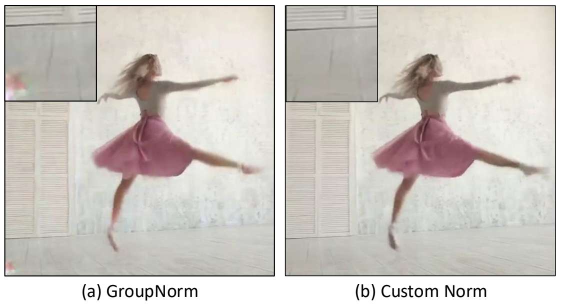


Figure 3. **'spot artifacts'.** Our progressive growing approach with `GroupNorm` leads to the 'spot' like artifacts (left) at the bottom right corner in reconstructed videos. (right) Removing the mean subtraction from group normalization eliminates the spot artifacts in reconstructed frames.

(ProMAG) base model. In Section 3.2, we analyze why the base ProMAG, capable of doing 4× temporal compression, has difficulties extending to 8× or 16× temporal compression. We discuss a method of progressively growing the ProMAG-4× model to achieve 8× and consequently 16× temporal compression in Section 3.2. Finally, in Section 3.3 we introduce our layer-wise spatial tiling technique during decoding to enable high-resolution encoding-decoding.

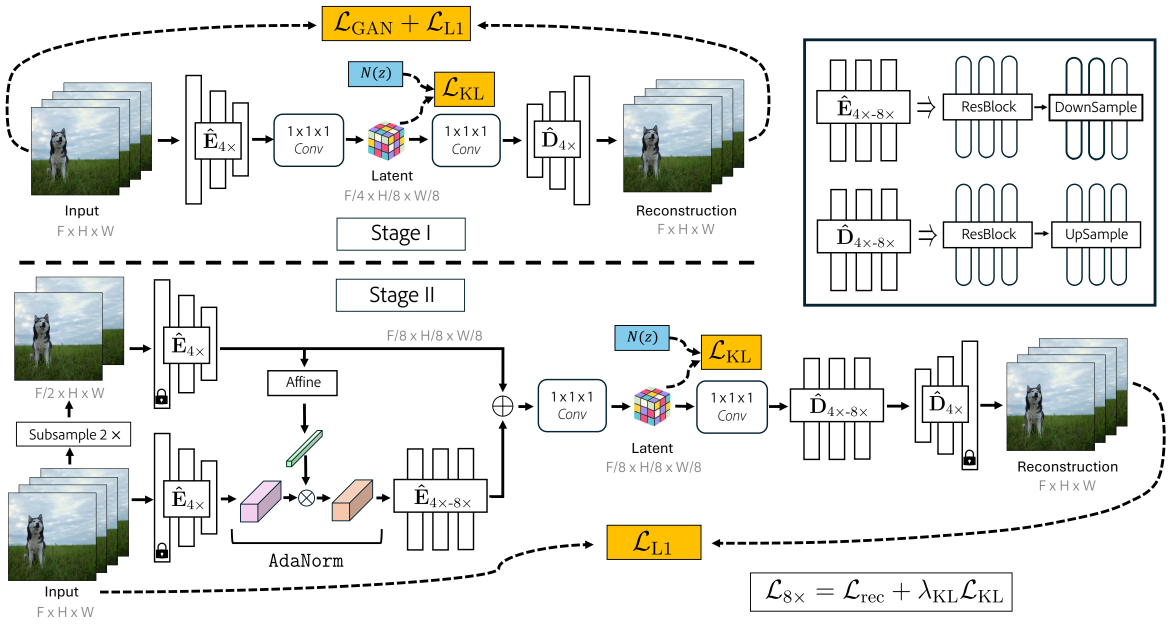
3.1. Base model: ProMAG

Following prior works [32, 46], we build our model based on the continuous-token variant of MagViT-v2 [47]. The MagViT-v2 tokenizer is composed of 3D causal convolution layers. The use of causal 3D convolution not only enables full spatiotemporal processing but also facilitates the consistent encoding of images and video data. Our model encodes an input video (or image) with $1+k \times N$ frames into 1+ N latent frames, where k is the compression ratio and $N = 0$ when encoding images. MagViT-v2, by default, can perform 4× temporal compression ($k = 4$) very reliably.

We make three key modifications to the base MagViT-v2 model to increase training and inference efficiency and to make it amenable to our method of progressive growing. We call our model ProMAG.

Image model initialization. Instead of training a 2D image encoder and using the weights to initialize 3D convolution kernels as in MagViT-v2 [47], we freeze the encoder of our pretrained 2D image model and use it to encode the first frame of the input video. We then train the MagViT-v2 model with 3D causal convolutions to focus only on encoding the rest of the video frames. In our experience, this makes the model training converge faster.

Efficient upsampling. MagViT-v2 encodes a 17-frame video into 5 latent frames which are decoded to 20 frames after 4× temporal upsampling. The presence of three additional frames in the reconstruction is a byproduct of the fact that the first latent frame only represents a single input frame. The MagViT-v2 decoder discards these 3 reconstruc-



$\mathcal{L}_{\text{GAN}} + \mathcal{L}_{\text{L1}}$
$N(z)$
$\mathcal{L}_{\text{KL}}$
$\hat{\text{E}}_{4\times}$
1x1x1 Conv
Latent
F/4 x H/8 x W/8
1x1x1 Conv
$\hat{\text{D}}_{4\times}$
Input
F x H x W
Reconstruction
F x H x W
Stage I
$\hat{\text{E}}_{4\times\text{-}8\times}$
ResBlock
DownSample
$\hat{\text{D}}_{4\times\text{-}8\times}$
ResBlock
UpSample
Stage II
F/2 x H x W
Subsample 2 ×
F/8 x H/8 x W/8
Affine
AdaNorm
$\hat{\text{E}}_{4\times\text{-}8\times}$
Latent
F/8 x H/8 x W/8
$\hat{\text{D}}_{4\times\text{-}8\times}$
$\hat{\text{D}}_{4\times}$
$\mathcal{L}_{\text{L1}}$
$\mathcal{L}_{8\times} = \mathcal{L}_{\text{rec}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}$

Figure 4. **Methodology.** Figure shows details of our method of progressive growing. In Stage I (top) we show a method of training our base $4\times$ video tokenizer. (bottom) we illustrate the detailed method of growing the base $4\times$ model to achieve $8\times$ temporal compression.

tion frames when generating the final output. We find this operation to be wasteful in terms of computation and becomes a greater problem with higher temporal compression. For example, in the case of $16\times$ compression on 17 frames, the decoder will output 32 frames and have to reject the first 15 frames. To mitigate this issue, in all the temporal upsampling layers of the decoder, we discard the 1^{st} frame, as it can be considered a padding frame in causal convolutions. This reduces the memory consumption significantly, especially for higher compression models, without hurting the reconstruction quality.

Removing *'spot'* artifacts. MagViT-v2 uses group normalization to normalize activations between layers. This causes *'spot'* like artifacts when trained to progressively increase the temporal compression (Figure 3(a)). This occurs because the model tends to encode important global information in these high-norm latent pixels, as also observed in [31]. To address this phenomenon, we modify the group-normalization operation by removing the mean subtraction following [22, 23]. We found this effectively resolves the *'spot'* artifacts in the reconstructed videos (Figure 3(b)).

3.2. Progressive model growing

Motivation. To increase the temporal compression of ProMAG from $4\times$ to $8\times$ or $16\times$, the standard approach is to add additional downsampling and upsampling blocks in the bottleneck layers of the encoder and decoder respectively. In our preliminary study, we found that directly training MagViT-v2 for $16\times$ temporal compression leads to poor reconstruction quality for a 24-fps video (Figure 2(b)). However, we observed that the $4\times$ temporal compression model can still accurately reconstruct a 6-fps video by feeding the same 24-fps video after subsampling frames by a factor of $4\times$ (Figure 2(a)). Since our base model ProMAG at $4\times$ temporal compression is similar to MagViT-v2, this conclusion is also applicable for ProMAG.

Since the ProMAG-$4\times$ can accurately reconstruct a 6-fps video, our strategy is to use it as guidance for the additional bottleneck downsampling and upsampling blocks to achieve $8\times$ and $16\times$ temporal compression. This can be thought of as a guided video interpolation problem. Instead of forcing the bottleneck compression layers to learn the entire information flow, we aim to induce them to reuse the information of the $4\times$ blocks, and only learn the essential information needed to synthesize the in-between frames, which is necessary to reconstruct the full video.

Progressive model growing. Building on the above observation, we develop our progressive model growing framework for boosting the temporal compression in ProMAG. We describe in detail below our framework for growing our model from $4\times$ to $8\times$ temporal compression. Growing the model from $8\times$ to $16\times$ follows the same procedure.

Key-frame embedding. A naive way to achieve $8\times$ temporal compression using the $4\times$ model, $\mathbf{E}_{4\times}$, is to subsample the input video frames $\mathbf{v}$ by a factor of two, $\mathbf{v}_{//2}$, and then encode the frames with $\mathbf{E}_{4\times}$. This can be referred to as learning the key-frame encodings z_{key} of the subsampled keyframes.

$$z_{\text{key}} = \hat{\mathbf{E}}_{4\times}(\mathbf{v}_{//2}) \tag{1}$$

where $\hat{\mathbf{E}}_{4\times}$ is the encoder blocks without the bottleneck $1 \times 1 \times 1$ layers in Figure 4.

Residual embedding. Now, with the encodings to accurately reconstruct keyframes in the video, we only need to learn the information for the remaining frames. We denote $\overset{*}{z}$ as the embedding of entire video $\mathbf{v}$ upto the $4\times$ temporal compression blocks, $\hat{\mathbf{E}}_{4\times}$. To reach $8\times$ compression, we compress $\overset{*}{z}$ through the additional $2\times$ downsampling block. However, we need to ensure that the downsampling blocks $\hat{\mathbf{E}}_{4\times\text{-}8\times}$ are aware of the information already present in z_{key}, as it only needs to preserve the remaining information to avoid redundancy. For this, we use adaptive group normalization (`AdaNorm`) to condition the intermediate latent, $\overset{*}{z}$, on the key-frame embeddings z_{key} before passing it to $\hat{\mathbf{E}}_{4\times\text{-}8\times}$.

$$\begin{aligned} \overset{*}{z}_{\text{inter}} &= \texttt{AdaNorm}(z_{\text{key}}, \hat{\mathbf{E}}_{4\times}(\mathbf{v})) \\ z_{\text{inter}} &= \hat{\mathbf{E}}_{4\times\text{-}8\times}(\overset{*}{z}_{\text{inter}}) \end{aligned} \tag{2}$$

We obtain the final latent z as the linear combination of both z_{key} and z_{inter}.

$$z = \texttt{Conv}_{1\times1\times1}(z_{\text{key}} + z_{\text{inter}}) \tag{3}$$

For the decoder we only add a bottleneck upsampling block.
Training strategy. We obtain the base $4\times$ temporal compression model by training with (1) the video-level GAN loss $\mathcal{L}_{\text{GAN}}$ [13], (2) the L_1 reconstruction loss $\mathcal{L}_{\text{rec}}$, and (3) the KL-divergence loss $\mathcal{L}_{\text{KL}}$.

$$\mathcal{L}_{4\times} = \mathcal{L}_{\text{rec}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{\text{GAN}}\mathcal{L}_{\text{GAN}} \tag{4}$$

We initialize the weights for the $8\times$ temporal compression model with the $4\times$ models and freeze the encoder and decoder blocks initialized from the $4\times$ model. At this stage, we only train the newly added blocks and the $1\times1\times1$ bottleneck layer using (1) the L_1 reconstruction loss $\mathcal{L}_{\text{rec}}$, and (2) the KL-divergence loss $\mathcal{L}_{\text{KL}}$. While GAN loss can slightly improve the FVD score in the reconstructed videos, it tends to introduce visual artifacts in challenging scenarios. Therefore, we do not use it at this stage, which also makes training with progressive growing significantly faster. Details about training efficiency are provided in the supplementary.

3.3. High resolution video reconstruction

Directly using our model, ProMAG, or MagViT-v2 to encode and decode videos at high resolution (*e.g.* 540×960), results in out-of-memory (OOM) errors due to the VRAM-intensive 3D convolution layers. We found the OOM issue arises only during decoding (i.e., high-resolution videos can be encoded as-is). With this in mind, we opted to encode the entire video at once and only spatially tile the encoded latent, decoding each tile separately. However, similar to tiling in RGB space, this approach results in artifacts, as seen in Figure 5 (a). In the latent tiling case, we found the issue to be a misalignment between the full, low-resolution video encoding and decoding regimen learned during training and the

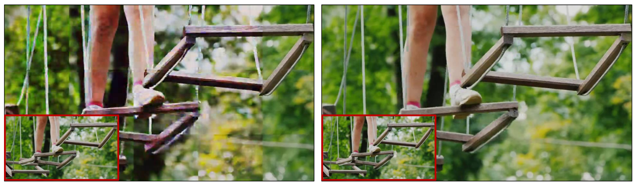

(a) Spatial tiling before Decoder
(b) Layer-wise Spatial Tiling

Figure 5. **High-Resolution Reconstruction.** (left) tilting the latent before passing through the decoder leads to artifacts in the reconstructed video. (right) layer-wise tiling resolves the artifacts. The red-bordered frame at the bottom left corner represents the ground-truth frame.

tiled decoding seen at inference. Specifically, we found that there was a meaningful difference in normalization statistics encountered during training and inference. To mitigate this discrepancy, we introduce a **layer-wise spatial tiling** technique: during decoding, we divide the input tensor to *each* `conv` layer into overlapping spatial tiles, process each tile independently, and merge the tiles back together using linear interpolation weights to blend the overlapping components. Using this approach, we can decode into high-resolution videos without any artifacts (Figure 5 (b)).

4. Experiments

4.1. Reconstruction Quality

Implementation details. We train our models on our internal dataset of 300M images and 15M videos at 256×256 resolution. The $4\times$, $8\times$, and $16\times$ temporal compression models are trained with 17-frame training video clips sampled at 6 fps, 12 fps, and 24 fps, respectively. Since we follow MagViT-v2's approach of using 3D *causal convolution* layers for spatiotemporal processing, videos with $1 + k$ x N frames are compressed into $1 + N$ latent frames, where k is the compression ratio. Following this, the $4\times$ models compress videos from 17 frames to 5 latent frames, similarly, $8\times$ from 17 frames to 3, and, $16\times$ from 17 frames to 2.
Baselines. We compare our method to the following state-of-the-art methods:

- MagViT-v2 [47]. As the official codebase is unavailable, we implement the continuous space version ourselves and train with a setting similar to our method. This is our main baseline where we perform full-model training for $4\times$, $8\times$, and $16\times$ temporal compression at $z\_dim = 8$ and 16.
- For fairness of comparison, we compare with SOTA tokenizers with $z\_dim = 8$ and 16. We compare against OmniTokenizer [44], VidTok [40], Cosmos-CV (continuous version) [1], VAE in CogVideoX [46], Wan [42], Hunyuan Video [18], CV-VAE [50], and, WF-VAE [27]. All the tokenizers are continuous-space, except for OmniTokenizer, which is discrete. Additionally, all the open-sourced SOTA tokenizers operate at $4\times$ temporal compression, except Cosmos-CV, which has both $4\times$ and $8\times$

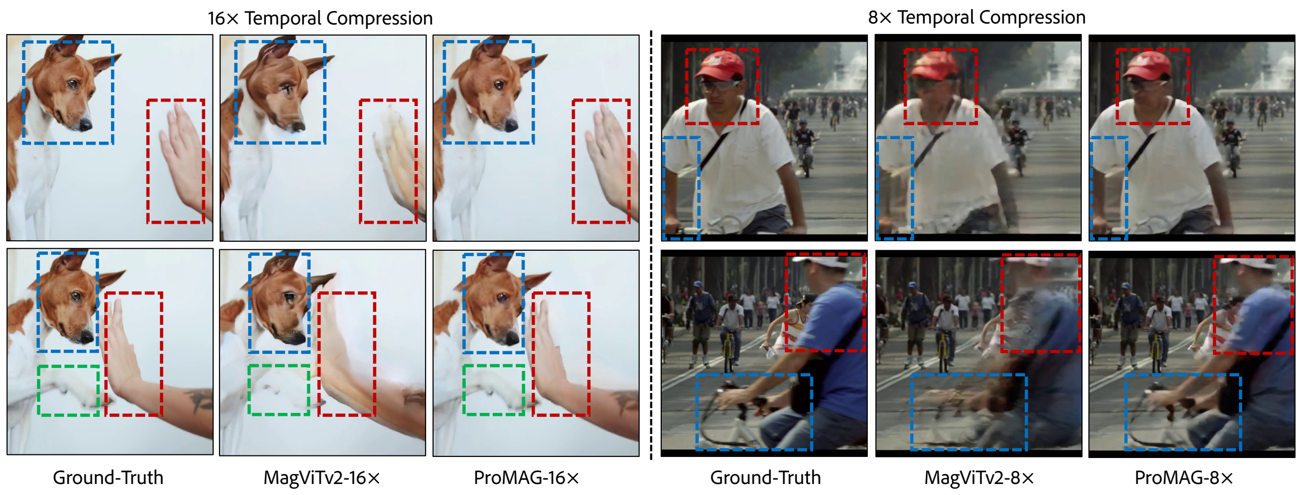


Figure 6. **Effectiveness of Progressive Growing at high temporal compression.** Video reconstruction example for (left) 16× temporal compression, (right) 8× compression. MagViT-v2 causes loss of details and produces artifacts in the reconstructed frames, like motion blur in the dog's and human's hand (left) and cyclist (right). The dotted boxes show regions of the most difference.

Method	z_dim	MCL-JCV			DAVIS		
		PSNR	LPIPS	rFVD	PSNR	LPIPS	rFVD
MagViT-v2	8	30.63	6.49	58.11	28.32	7.30	129.11
OmniTokenizer [44]		24.55	11.06	124.83	23.60	15.02	290.38
VidTok [40]		**31.20**	6.93	62.41	**28.73**	8.05	150.54
ProMAG		30.99	**6.53**	**57.91**	28.43	**7.38**	**130.56**
Cosmos-CV [1]	16	32.08	9.54	50.65	29.81	13.71	113.13
CogVideoX [46]		32.57	6.21	50.75	29.8	7.49	100.56
VidTok [40]		32.95	5.55	37.7	**31.12**	5.22	94.04
WF-VAE [27]		**33.18**	5.09	39.71	30.29	6.51	88.27
Wan [42]		32.27	5.15	39.33	30.15	5.29	98.49
Hunyuan [25]		33.13	4.95	36.21	30.4	5.02	97.56
CV-VAE [50]		32.83	6.31	46.35	30.07	7.93	101.94
ProMAG		32.94	**4.91**	**34.21**	30.77	**4.98**	**93.92**

Table 1. **Reconstruction comparison at base 4× temporal compression.** We compare our method with baselines on MCL-JCV and DAVIS under 8×8×4 compression on 512×512 resolution.

Method	Compression	z_dim	MCL-JCV			DAVIS		
			PSNR	LPIPS	rFVD	PSNR	LPIPS	rFVD
MagViT-v2	8×8×8	8	28.47	11.17	219.36	24.02	16.7	525.1
ProMAG			**30.26**	**8.36**	**96.85**	**26.48**	**11.37**	**360.54**
MagViT-v2	8×8×16	8	26.15	14.54	255.51	21.82	20.54	728.04
ProMAG			**28.31**	**11.2**	**183.4**	**23.89**	**16.9**	**608.43**
Cosmos-CV [1]	8×8×8	16	31.05	11.79	101.36	28.27	17.87	222.61
MagViT-v2			28.51	10.62	135.57	27.18	8.84	243.19
ProMAG			**32.06**	**6.94**	**63.64**	**28.70**	**8.32**	**194.79**
MagViT-v2	8×8×16	16	28.51	10.62	135.57	24.17	14.84	427.52
ProMAG			**30.02**	**9.24**	**115.25**	**25.64**	**13.89**	**320.80**

Table 2. **Reconstruction comparison at high temporal compression.** We compare our method with baselines on MCL-JCV and DAVIS under 8×8×4 compression on 512×512 resolution.

temporal compression.

We compare the above baselines against our base ProMAG-4× temporal compression. For more aggressive compression, 8× and 16×, we extend and train MagViT-v2 at the target compression ratio and compare against our method.

Evaluation benchmarks. We evaluate all methods for reconstruction quality on two standard video benchmarks, MCL-JCV [43] and DAVIS 2019 (Full Resolution) [7]. We select two different settings for reconstruction, one at square crops of 512×512 and the other at 360×640 (at native aspect ratio of 16:9). We compare the reconstruction quality on PSNR, LPIPS [49], and Fréchet video distance (FVD) [12, 41].

Effectiveness of our base 4× ProMAG. While our main goal is not to achieve the best 4× temporal compression model, we do this experiment to verify that the base 4× model we developed can serve as a strong tokenizer. From Table 1, we see that MagViT-v2 and ProMAG at $z\_dim = 8$ perform better than Omnitokenizer [44] and are comparable to SOTA tokenizer VidTok [40]. Thus, we build our base model inspired from MagViT-v2 architecture. Even at $z\_dim = 16$ our method performs better than SOTA methods like Cosmos-CV (continuous) [1], CogVideoX [46], and CV-VAE [50] across all metrics and datasets and comparable to very recent SOTA methods like VidTok [40] WF-VAE [27], Wan [42], and Hunyuan Video [25], which focus on creating high-quality 4× temporal compression video tokenizers. Thus, our base 4× tokenizer is a competitive model in terms of reconstruction quality.

Effectiveness of progressive growing for boosting temporal compression. We compare our method, ProMAG, against MagViT-v2 on 2 different settings, one with 8 channels and the other with 16 channels in the latent dimension for both 8× and 16× temporal compression. From Table 2, ProMAG achieves better reconstruction quality than MagViT-v2 directly extended to high compression. It is interesting to note that the 16-channel version of ProMAG-8× can maintain a somewhat similar reconstruction quality to ProMAG-4×. Qualitative analysis in Figure 6, (left) we observe that at 16× compression, MagViT-v2 is unable to preserve even the coarse structural details in the face or the dog or the human hand. Even at 8× temporal compression, in regions of very high motion like the cyclist, MagViT-v2 blends

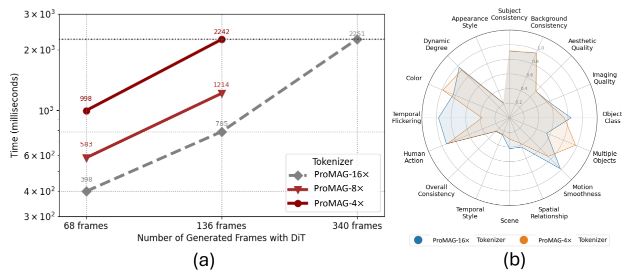


Figure 7. **Text-to-Video Evaluation.** Quantitative comparison of T2V generation model train on 4× and 16× latent space of ProMAG. (top) The graph of time taken per denoising step of the diffusion model for generating N frame video. The time is computed on H100 with Flash Attention 3 [37]. The resolution of the videos is 192 × 360 pixels. We find that it takes the same time per diffusion step to generate a 136-frame video using the 4× latent space, as generating a 340-frame video using the 16× latent space. (bottom) Quantitative analysis different dimension on VBench [19] on generated videos with 4× and 16× compressed latents.

Method	Compression	*z_dim*	MCL-JCV			DAVIS		
			PSNR	LPIPS	rFVD	PSNR	LPIPS	rFVD
ProMAG	8×8×8	8	**32.06**	6.94	63.64	**28.70**	8.32	194.79
ProMAG (w/ GAN)			31.78	**6.21**	**53.18**	28.21	**7.76**	**160.22**
ProMAG	8×8×16	16	**30.02**	9.24	115.25	**25.64**	13.89	320.80
ProMAG (w/ GAN)			29.67	**8.38**	**99.64**	25.05	**12.04**	**274.24**

Table 3. **Ablation on GAN loss during progressive growing.** We compare our method of progressive growing with and without using GAN loss.

the foreground object with the background, causing noticeable motion blur in the reconstructed videos. ProMAG at 8× temporal compression also performs better than Cosmos-CV.

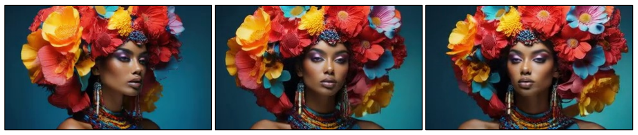
Ablation on using GAN loss for progressive growing. In this ablation study, we verify whether adding GAN loss during progressive growing is beneficial. From Table 3, we find GAN loss at this stage while slightly improving perceptual quality (LPIPS and FVD), also slightly reduces PSNR. Additionally, not using GANs while progressive growing improves training efficiency (details in supplementary). Thus, we choose not to GAN loss during progressive growing.

4.2. T2V Quality with Compressed Latent Space

Our ultimate goal is to generate a latent space that is not only compact but more importantly, good for downstream diffusion model training. We evaluate the latent spaces of our tokenizers by accessing the text-to-video generation quality of the Diffusion Transformer (DiT) model trained with the resulting latent spaces. In this section, we evaluate our 4× and 16× models. We aim to investigate if DiT training quality and convergence degrades when trained with the highly compressed latent space of 16× temporal compression compared to the commonly used 4× temporal compression.

Training and implementation details. Our text-to-video diffusion model is based on the standard DiT formulation [30], composed of multiple Transformer blocks, where we replace spatial self-attention with spatial-temporal self-attention blocks. We train our model on an internal dataset of images and videos. We first train DiT on 256p images for 200K iterations and then jointly train on images and videos for about 150K more iterations.

Prompt: *"In the aerial view of Santorini, white cycladic buildings with blue domes are distinctively seen against ..."*



Prompt: *"A woman with a flower headpiece, inspired by vray tracing, features vivid colors and playful berrypunk..."*

Figure 8. **Text-to-Video Generation Results.** We show vibrant videos generated by DiT on our ProMAG-16× latent space. This highlights that training DiT on highly compressed latent space can generate videos with accurate text coherence and realistic motion.

Evaluation metrics. We evaluate the quality of text-to-video generation using VBench [19]. Following the official guidelines, we generate videos with all the 946 provided text prompts and generate videos with 5 random seeds per prompt. We evaluate based on all 16 quality dimensions.

T2V generation quality. From Figure 7(b), we find that training DiT with 16× temporally compressed latent space does not degrade the video generation quality on almost all dimensions compared to 4× compression latent. Figure 8 shows two diverse and vibrant videos generated by DiT with 16× compressed latent. We find that the DiT can accurately follow the caption and generate very realistic motion.

Efficiency. Using a compact latent space allows DiT to operate with fewer tokens, significantly reducing GPU memory and time. In a fixed frame length case (68 frames, ∼2.8s at 24 fps), DiT with 4× compression generates 20 latents at 0.99s/timestep, while 8× compression reduces this to 12 latents at 0.58s (1.7× speedup). With 16×, only 8 latents are needed, achieving 0.39s/timestep (2.5×). In a fixed token budget of 40 latent frames, the 4× model generates a 136-frame video (5.6s), whereas the 16× model produces 340 frames (14.1s), a 2.5× increase in video length.

5. Discussion

5.1. Progressive growing vs. progressive training

We want to investigate if the effectiveness of our method comes from our specific design towards reusing pretrained high-quality, lower compression model and only learning the remaining information in the bottleneck layers, or simply just from progressive training of the model in stages by adding

Method	MCL-JCV		
	PSNR	LPIPS	rFVD
MagViT-v2	28.51	10.62	135.57
ProMAG (w/o residuals & `AdaNorm`)	30.35	8.05	88.18
ProMAG (full method)	**32.06**	**6.94**	**63.64**

Table 4. **Progressive Growing v/s Progressive training**. Comparison of reconstruction quality of our full method against the naive way of progressive training and ProMAG on the 512x512 crops of the MCL-JCV video dataset.

Method	MCL-JCV		
	PSNR	LPIPS	rFVD
ProMAG (4× w/ $s_f = 2$ + 2X Interpolation)	30.68	8.61	230.16
ProMAG (8× w/ $s_f = 1$)	**32.06**	**6.5**	**53.77**
ProMAG (4× w/ $s_f = 4 + 4\times$ Interpolation)	28.42	12.35	341.14
ProMAG (16× w/ $s_f = 1$)	**29.6**	**9.93**	**147.48**

Table 5. **Quantitaive Comparison with External Interpolation**. Reconstruction quality on MCL-JCV dataset on 512x512 crops on 8× and the other at 16× temporal compression.

more bottleneck layers (w/o skip information flow).

Progressive training. In this case, we start from a pretrained 4× model, and then add bottleneck downsampling and upsampling blocks to the encoder and decoder respectively, to grow the compression rate and only train the additional blocks. This does not include the use of subsampled frames encoding or `AdaNorm` as our full method (Figure 4). Progressive training, in essence, is similar to tokenizer in OpenSORA [51] and OpenSORA-Plan [26], which first trains a spatial-only VAE and then adds additional bottleneck 4× temporal compression blocks.

Evaluation. Table 4 shows the comparison of the reconstruction quality on MCL-MJC at 512x512 crops for 8× temporal compression. We find that progressive training (Table 4, 2nd row) by itself, improves the PSNR, LPIPS, and FVD compared to directly training MagViT-v2 from scratch for 8× compression. We hypothesize that it might be because with large compression and increasing channel multiples, training the video tokenizer at once may lead to difficulty in convergence. On top of that, our method of progressive growing further improves the reconstruction of all the metrics.

5.2. Progressive growing vs. frame subsampled encoding + external interpolation

The primary motivation for our method of progressive growing was that the 4× compression model performs very accurate reconstruction even for a 6fps video (after 4 times frame subsampling from a 24fps video).

Baseline. A very natural question would be how our method for 8× (or 16×) temporal compression would fare against the solution where we first perform 4× temporal compression on videos subsampled with a factor of 2 (or 4), followed by using external state-of-the-art frame interpolation method like EMA-VFI [48] to perform 2X (4X) frame interpolation

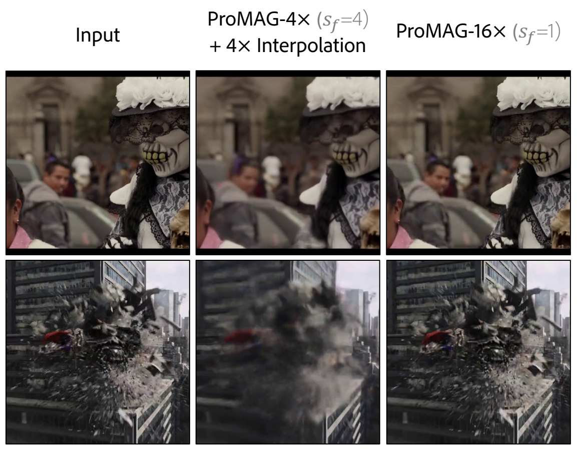


Figure 9. **Qualitative Comparison with External Interpolation.** Reconstruction comparison of our method ProMAG with 16× temporal compression on a 24fps video, against a baseline where we first encode the video at low fps (with frame subsampling $s_f = 4$), in this case 6fps, followed by using external interpolation method to generate the 4 in-between frames. (middle) The baseline with external frame interpolation produces very blurry outputs in regions of abrupt and high-intensity motion, compared to our method ProMAG-16× operating on 24fps video directly (right). This implies in some sense, that our progressive growing approach has in some sense learned a better interpolation for reconstruction.

on the reconstructed frames. We observe, in Figure 9 that, performing external interpolation in regions of very high and complex motion causes much more blurring compared to applying our ProMAG-16× directly on the original videos.

Evaluation. From Table 5, we see that our method for 8× temporal compression achieves much better reconstruction quality compared to using our 4× model on subsampled frames followed by 2X interpolation. A similar trend can be observed in the case of 16× temporal compression.

6. Conclusion and Limitations

In this work, we push the boundary to which we can perform temporal compression while keeping latent channel dimension constant. To this end, we propose a novel method of progressively growing our 4× temporal compression video tokenizer, ProMAG, to 8× and subsequently 16× temporal compression. Through extensive evaluation of reconstruction and text-to-video generation, we show that (i) our video tokenizer can perform much better reconstruction than baseline methods at very high temporal compression rates, and (ii) our high compression latent space is suitable for DiT training and provides an immense boost in terms of efficiency for video generation. However, there still exist limitations, which offer opportunities for future research. In scenarios of very high and abrupt motion, like sports videos, our high-compression tokenizers, while performing significantly better than the directly trained baselines, also suffer from temporal artifacts in reconstruction.

Acknowledgments. We thank Kuldeep Kulkarni, Cusuh Ham, and Gunjan Aggarwal for proofreading the paper. We are also grateful to Richard Zhang, Zhifei Zhang, Jianming Zhang, Gunjan Aggarwal, Kevin Duarte, Yang Zhou, Zhan Xu, and Lakshya for several fruitful discussions. This work was partly done while Yitian was an intern at Adobe.

References

[1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025. 3, 5, 6

[2] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022. 2, 3

[3] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 2

[4] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023. 2

[5] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023. 2

[6] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. 2

[7] Sergi Caelles, Jordi Pont-Tuset, Federico Perazzi, Alberto Montes, Kevis-Kokitsi Maninis, and Luc Van Gool. The 2019 davis challenge on vos: Unsupervised multi-object segmentation. *arXiv preprint arXiv:1905.00737*, 2019. 6

[8] 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. *arXiv preprint arXiv:2410.10733*, 2024. 2, 3

[9] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024. 2

[10] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. *arXiv preprint arXiv:2203.13131*, 2022. 2, 3

[11] Songwei Ge, Seungjun Nah, Guilin Liu, Tyler Poon, Andrew Tao, Bryan Catanzaro, David Jacobs, Jia-Bin Huang, Ming-Yu Liu, and Yogesh Balaji. Preserve your own correlation: A noise prior for video diffusion models. *arXiv preprint arXiv:2305.10474*, 2023. 2, 11

[12] Songwei Ge, Aniruddha Mahapatra, Gaurav Parmar, Jun-Yan Zhu, and Jia-Bin Huang. On the content bias in fréchet video distance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 6

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 3, 5

[14] Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Fei-Fei Li, Irfan Essa, Lu Jiang, and José Lezama. Photorealistic video generation with diffusion models. In *European Conference on Computer Vision*, pages 393–411. Springer, 2025. 2, 11

[15] Yingqing He, Haoxin Chen, and Menghan Xia. Videocrafter: A toolkit for text-to-video generation and editing. https://github.com/VideoCrafter/VideoCrafter, 2023. 2

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2

[17] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 2, 3, 11

[18] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *ECCV*, 2018. 5

[19] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024. 2, 7

[20] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 3

[21] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 3

[22] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 4

[23] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 4

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 2

[25] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024. 6

[26] PKU-Yuan Lab and Tuzhan AI etc. Open-sora-plan, 2024. 2, 8

[27] 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. *arXiv preprint arXiv:2411.17459*, 2024. 5, 6

[28] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 11

[29] Willi Menapace, Aliaksandr Siarohin, Ivan Skorokhodov, Ekaterina Deyneka, Tsai-Shien Chen, Anil Kag, Yuwei Fang, Aleksei Stoliar, Elisa Ricci, Jian Ren, et al. Snap video: Scaled spatiotemporal transformers for text-to-video synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7038–7048, 2024. 2

[30] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 2, 7, 11

[31] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024. 3, 4

[32] Can Qin, Congying Xia, Krithika Ramakrishnan, Michael Ryoo, Lifu Tu, Yihao Feng, Manli Shu, Honglu Zhou, Anas Awadalla, Jun Wang, et al. xgen-videosyn-1: High-fidelity text-to-video synthesis with compressed representations. *arXiv preprint arXiv:2408.12590*, 2024. 3

[33] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 2

[34] Jingjing Ren, Wenbo Li, Haoyu Chen, Renjing Pei, Bin Shao, Yong Guo, Long Peng, Fenglong Song, and Lei Zhu. Ultrapixel: Advancing ultra-high-resolution image synthesis to new peaks. *arXiv preprint arXiv:2407.02158*, 2024. 2, 3

[35] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 2, 11

[36] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022. 2

[37] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08608*, 2024. 7

[38] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022. 2, 11

[39] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 2

[40] Anni Tang, Tianyu He, Junliang Guo, Xinle Cheng, Li Song, and Jiang Bian. Vidtok: A versatile and open-source video tokenizer. *arXiv preprint arXiv:2412.13061*, 2024. 5, 6

[41] Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018. 6

[42] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 5, 6

[43] Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C-C Jay Kuo. Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset. In *ICIP*, 2016. 6

[44] Junke Wang, Yi Jiang, Zehuan Yuan, Bingyue Peng, Zuxuan Wu, and Yu-Gang Jiang. Omnitokenizer: A joint image-video tokenizer for visual generation. *Advances in Neural Information Processing Systems*, 37:28281–28295, 2025. 5, 6

[45] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018. 3

[46] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024. 2, 3, 5, 6, 11

[47] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023. 2, 3, 5, 11

[48] Guozhen Zhang, Yuhan Zhu, Haonan Wang, Youxin Chen, Gangshan Wu, and Limin Wang. Extracting motion and appearance via inter-frame attention for efficient video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5682–5692, 2023. 8

[49] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

[50] 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 mod-

els. *Advances in Neural Information Processing Systems*, 37: 12847–12871, 2025. 5, 6

[51] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, 2024. 2, 8

A. Progressive Growing - from 8× to 16× Temporal Compression

In Figure 2 of the main paper we illustrate our method of growing the temporal compression of the 4× video tokenizer to 8× temporal compression. In Figure 10, we show the details of our entire method, i.e., growing the base 4× model to 8× then ultimately to 16× temporal compression in 3 stages. Note that all the parameter weights in the subsequent are initialized from their corresponding parameters in the previous stage. The newly added parameters are initialized with random weights.

B. ProMAG Implementation Details

Model Architecture. In Table 9, we describe the details of our tokenizer architecture for 4×, 8× and 16× temporal compression models. The encoder and decoder design of our model, ProMAG follows MagViT-v2 structure. The discriminator is taken from Stable Diffusion VAE [35], where we replace the `Conv2D` with `Conv3D`. In our case, using the MagViT-v2 discriminator produced checkerboard-like artifacts in reconstunction.

Training Hyperparameters. We provide additional detailed training hyper-parameters for our models as listed below:

- Video input: 17 frames, frame stride 1, 256 × 256 resolution.
- Reconstruction loss weight: 1.0.
- Generator loss type: Hinge Loss [28].
- Generator adversarial loss weight: 0.1 for 4× and 0.0 for 8× and 16× (Note: we do not use discriminator to progressively grow the model to 8× and 16× temporal compression.)
- Discriminator gradient penalty: 0.0 r1 weight.
- VGG Perceptual loss weight: 1.0.
- KL-Divergence loss weight: 1e-12.
- Tokenizer Learning rate: 0.0001.
- Tokenizer Optimizer Params: Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.99$.
- Tokenizer weight decay: 0.0001
- Discriminator Learning rate: 0.0001.
- Discriminator Optimizer Params: Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.99$.
- Discriminator weight decay: 0.0001
- Training iterations: 300K for 4× model, 100K for 8× model (on top of 4× model), 100K for 16× model (on top of 8× model).
- Global Batch size: 32.

C. Text-to-Video Implementation Details

Model Architecture. Our text-to-video diffusion model is based on the standard DiT [30], composed of multiple Transformer blocks, where we replace spatial self-attention with spatial-temporal self- attention blocks. The model architecture is similar to the diffusion transformer in CogVideoX [46]. Following them, we also keep the number of model parameters to be around 5B.

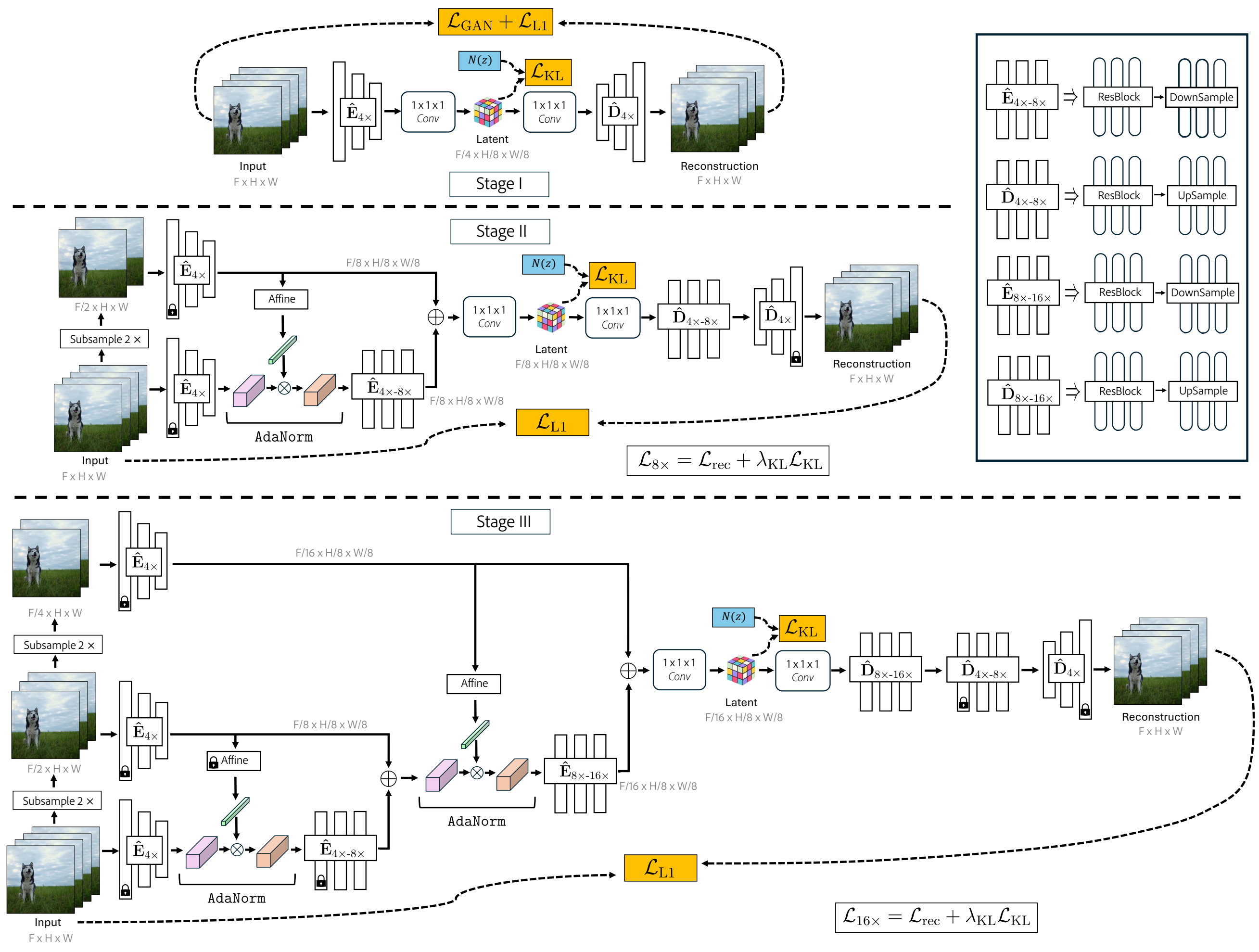
Dataset Details. We train on our internal dataset of 300M images and 1M videos. Images contain different aspect ratios, while videos are of 192x360 resolution. We train using a relatively small scale and low resolution since the computation cost for training text-to-video models is very enormous. Additionally, our main focus in training the text-to-video model is to verify that our extremely compressed latent space (16× temporal compression) is compatible with DiT training and can achieve a similar quality to 4× compressed latent space. Our goal is not to compete with state-of-the-art text-to-video models.

Training Details. Following standard practices [11, 14, 17, 38], we train our text-to-video diffusion model in 2 stages. In the first stage (image pertaining), we train the model only on images. In the second stage (joint training), we train the model jointly on both images and videos. We train the first stage for about 200K iterations and the second stage for around 150K additional iterations. We train our models on 8 nodes of H100 (64 GPUs in total).

Training on Longer Videos. In Section 4.2 (Efficiency) in the main paper, we discuss that due to the highly compact nature of our 16× latent space, we can use the same token budget for 340 frames (= 20 latent frames for 16× temporal compression), which is the same as 136 frames with 4× temporal compression. To verify the generation quality of very long videos 340 frames (or 14.1s) video at 24fps, we also train text-to-video model for this. More specifically, we only finetune our text-to-video model trained on 16× temporally compressed latent space for an additional 10K iterations on video training of 340 frames.

D. Resolving Jump Issues: Overlapping Frame Reconstruction and Text-to-Video Generation

Following MagViT-v2 [47], train ProMAG on 17 frame video, for all 4×, 8× and 16× temporal compression ratio. We found that since we train the model on 17 frames, we can only do encoding and decoding with 17 (or less) frames. We found the similar case is true with MagViT-v2. We found that reconstruction of more than 17 frames causes deterioration in the reconstruction results beyond 17 frames, like blurring, or in some cases checkerboard-like artifacts.



$\mathcal{L}_{\text{GAN}} + \mathcal{L}_{\text{L1}}$
$N(z)$
$\mathcal{L}_{\text{KL}}$
Input
F x H x W
$\hat{E}_{4\times}$
1x1x1 Conv
Latent
F/4 x H/8 x W/8
$\hat{D}_{4\times}$
Reconstruction
F x H x W
Stage I
Stage II
F/2 x H x W
Subsample 2 ×
F/8 x H/8 x W/8
Affine
AdaNorm
$\hat{E}_{4\times\text{-}8\times}$
$\hat{D}_{4\times\text{-}8\times}$
$\mathcal{L}_{\text{L1}}$
$\mathcal{L}_{8\times} = \mathcal{L}_{\text{rec}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}$
ResBlock
DownSample
UpSample
$\hat{E}_{8\times\text{-}16\times}$
$\hat{D}_{8\times\text{-}16\times}$
Stage III
F/4 x H x W
F/16 x H/8 x W/8
$\mathcal{L}_{16\times} = \mathcal{L}_{\text{rec}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}$

Figure 10. **Methodology.** Figure shows details of our complete method of progressive growing for extending the temporal compression to 16× from 4× compression. In Stage I (top) we show a method of training our base 4× video tokenizer. (middle) Stage II we illustrate the detailed method of growing the base 4× model to achieve 8× temporal compression. (bottom) Stage III we extend the 8× temporal compression model to achieve 16× temporal compression. $\mathcal{N}(z)$ represent a standard normal distribution.

Thus for reconstruction (or text-to-video generation), for an N frame video, we need to process it in chunks of 17 frames at a time. This causes jumps like effect in regions of high frequency details every 17 frames. This jumps is much less noticeable in reconstruction results, but more noticeable in the text-to-video generation results. For video generation this jumping effect reduces with more training iterations but still persists slightly. To counteract this, we perform encoding in chunks with overlap of few frames (in this case of 4 frames) Figure 11. The overlapped regions in the output is blended in pixel-space with linear interpolated weights. This resolves the jumping artifacts perceptually both in reconstruction and video generation results (please refer to video results showing this in the supplementary videos).

E. Training Efficiency of Our Approach

Method	Compression	Time/iter	Total iters (cumulative)	Training time (cumulative)
MagViT-v2	$8\times8\times4$	0.42s	300K	1.45 days
ProMAG		0.42s	300K	1.45 days
MagViT-v2	8×8×8	0.92s	400K	4.25 days
ProMAG (4× → 8×)		0.42s	400K	1.94 days
MagViT-v2	$8\times8\times16$	1.04s	500K	6.01 days
ProMAG (8× → 16×)		0.23s	500K	2.21 days

Table 6. Details of the per-iteration training time and total number of iterations comparing our method of progressive-growing (ProMAG) w.r.t to full model training (MagViT-v2). All the model has latent dimension (z_dim) = 8. The training times are computed for 17 frames of 256×256 videos on H100 GPU.

Our progressive-growing method does not cause any train-

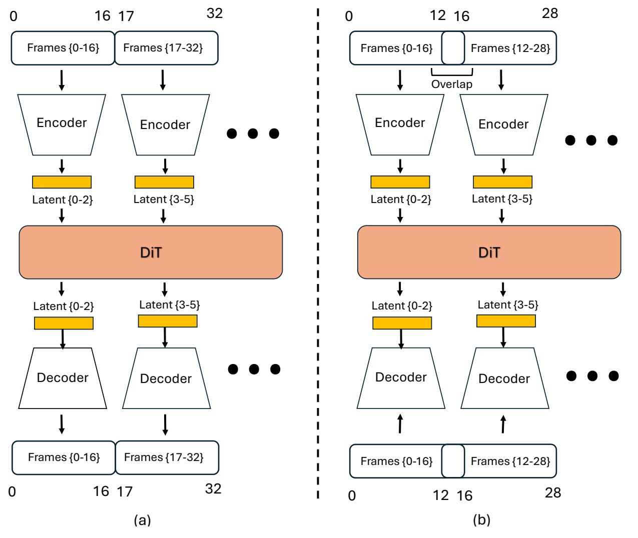


Figure 11. **Text-to-Video with overlapping frames.** (left) The conventional way of generating videos, where frames are encoded into overlapping chunks. This results in jumps like artifacts across chunks in the generated video in regions of high frequency details. (right) Generating video with chunks with overlap (here of 4 frames). The overlapped regions in the output is blended in pixel-space with linear interpolated weights. This resolves the jumping artifats.

ing efficiency overhead. In contrast, it improves the training efficiency compared to training the full model directly for high compression. From 6, we see that full model training from scratch (MagViT-v2) for 8× compression requires 2.3*s/iter* while the progressive growing strategy takes 1.05*s/iter*. Similarly, training 16× model on top of 8× model takes 0.57*s/iter*, compared to 2.6*s/iter* for full model training at 16× compression. As a result, progressive growth allows us to train faster while achieving significantly better performance with the same number of training iterations compared to the full training approach. The main reason why our approach of progressive growing has significantly lower training time per iteration is because it does not need a discriminator to grow the model from 4× to 8× or from 8× to 16× temporal compression. Additionally, most of the encoder and decoder layers are frozen. For fairness of comparison, we also train the full-model training (MagViT-v2) baseline for the same total number of iterations for a given temporal compression. Even though the full model training takes ∼3× more training time for 16× temporal compression compared to our method ProMAG, our method can still achieve a lot higher quality reconstruction.

F. Reconstruction time Number of Parameters

For fairness of comparison, our model ProMAG follows the same encoder and decoder configuration to our implementation of the baseline MagViT-v2 for respective compression

Method	Compression	Enc. Time	Dec. Time	# Params
MagViT-v2	8×8×8	0.42 s	0.47 s	793 M
ProMAG		0.61 s	0.47 s	794 M
MagViT-v2	8×8×16	0.44 s	0.83 s	984 M
ProMAG		0.73 s	0.83 s	986 M

Table 7. Details of the reconstruction time and number of parameters of our model ProMAG w.r.t the baseline MagViT-v2. All the model has latent dimension (z_dim) = 8. The times are computed for 17 frames of 512×512 videos on A100 GPU.

Method	Enc. (# Params)	Enc. (Time)	Dec. (# Params)	Dec. (Time)
CogVideoX	92.2M	0.14s	123.3M	0.32s
WF-VAE	84.5M	0.02s	232.4M	0.14s
Wan	53.3M	0.07s	73.2M	0.12s
Hunyuan	100.3M	0.13s	146.1M	0.26s
CV-VAE	69M	0.05s	112M	0.15s
ProMAG	105M	0.08s	231M	0.14s

Table 8. Details of the reconstruction time and number of parameters of encoder and decoder of our model ProMAG w.r.t the baselines which perform 4× temporal compression. All the model has latent dimension (z_dim) = 16. The times are computed for encoding and decoding 17 frames of 512×512 resolution videos on A100 GPU.

ratios. The difference is the progressive training strategy and the residual encoding with `AdaNorm` layers. From Table 7, the decoding time of our model and baseline MagViT-v2 is also the same because it follows the design. Only the encoding time of our model is 1.5× that of baseline MagViT-v2 because we need to do two forward passes through the encoder, one with the full input and the other with temporally subsampled input (subsampled by a factor of 2). Additionally, our model ProMAG has almost the same number of parameters as MagViT-v2; the only slight increase is due to the learnable parameters in `AdaNorm` blocks. Even so, for both 8× and 16× temporal compression, our model ProMAG has much better reconstruction quality than MagViT-v2 trained directly for 8× or 16× temporal compression. In Table 8, we show the number of parameters for encoder and decoder along with the encoding and decoding times between our model and other baselines with 4× temporal compression. *Note:* Our focus (or contribution) is **not** to have the fastest or the most efficient video tokenizer.

Encoder Config	$4\times$	$8\times$	$16\times$
inputs	pixels	pixels	pixels
input size	$17 \times 256 \times 256$	$17 \times 256 \times 256$	$17 \times 256 \times 256$
video fps	6	12	24
latent dimension	$5 \times 32 \times 32$	$3 \times 32 \times 32$	$2 \times 32 \times 32$
`Conv`-type	`CausalConv3D`	`CausalConv3D`	`CausalConv3D`
base channels	128	128	128
channel multipliers	1,2,4,6	1,2,4,6,6	1,2,4,6,6,6
spatial downsampling strategy	true,true,true,false	true,true,true,false,false	true,true,true,false,false,false
temporal downsampling strategy	false,false,true,true	false,false,true,true,true	false,false,true,true,true,true
downsampling strategy	strided `Conv`	strided `Conv`	strided `Conv`
number of residual blocks	2	2	2
z_channels	256	256	256
z_dim (number of channels in latent)	8 (or 16)	8 (or 16)	8 (or 16)
Decoder Config			
outputs	pixels	pixels	pixels
input (latent) dimension	$5 \times 32 \times 32$	$3 \times 32 \times 32$	$2 \times 32 \times 32$
output size	$17 \times 256 \times 256$	$17 \times 256 \times 256$	$17 \times 256 \times 256$
`Conv`-type	`CausalConv3D`	`CausalConv3D`	`CausalConv3D`
base channels	128	128	128
channel multipliers	6,4,2,1	6,6,4,2,1	6,6,6,4,2,1
spatial downsampling strategy	true,true,true,false	false,true,true,true,false	false,false,true,true,true,false
temporal downsampling strategy	false,true,true,false	true,false,true,true,false	true,true,false,true,true,false
downsampling strategy	nearest + `Conv`	nearest + `Conv`	nearest + `Conv`
number of residual blocks	3	3	3
z_channels	256	256	256
Discriminator Config			
discriminator type	patchGAN		
inputs	pixels		
input size	$17 \times 256 \times 256$		
number of layers	3	None	None
kernel size	$3 \times 4 \times 4$		
base channels	64		
`Conv`-type	`Conv3D`		

Table 9. Details of the encoder and decoder configurations of our video tokenizer ProMAG, and the discriminator configuration for different temporal compression ratios $4\times$, $8\times$ and $16\times$. We use discriminator only for training the $4\times$ base model.