Progressive Growing of Video Tokenizers for Temporally Compact Latent Spaces - Supplementary Material

Aniruddha Mahapatra[1] Long Mai[1] David Bourgin[1] Yitian Zhang[1,2] Feng Liu[1]
[1]Adobe Research [2]Northeastern University

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 1, 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 4, 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 [6], 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 [4].
- 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 [5], 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 [8]. 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 [1–3, 7], 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×

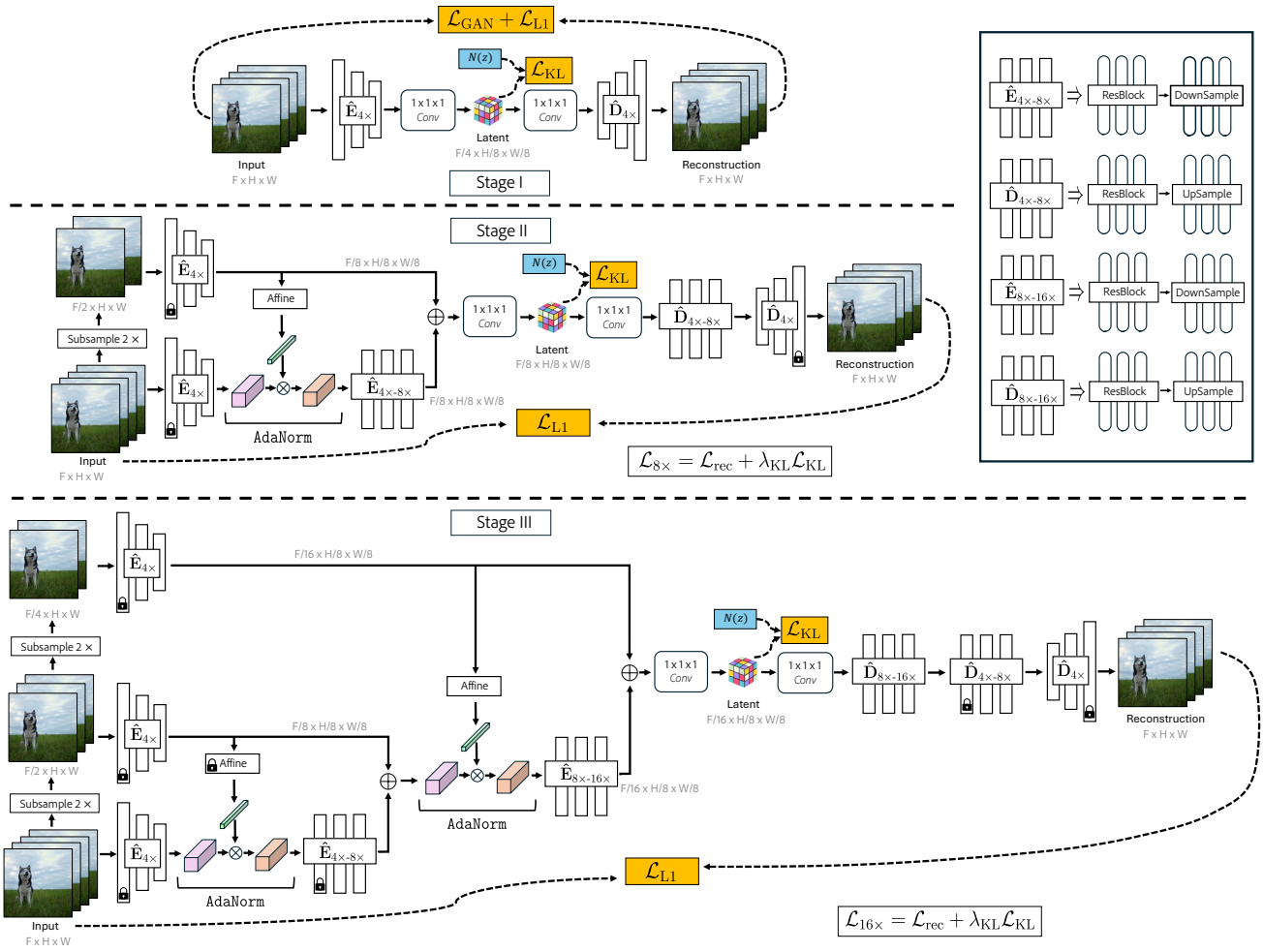

Figure 1. **Methodology.** Figure shows details of our complete method of progressive growing for extending the temporal compression to 16× from 4× compression. In Stage I (top) we show a method of training our base 4× video tokenizer. (middle) Stage II we illustrate the detailed method of growing the 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.

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 [9], 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. 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 2. 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

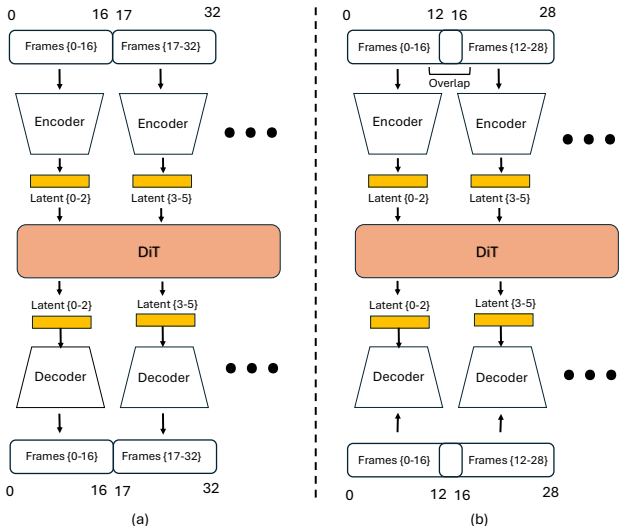

Figure 2. **Text-to-Video with overlapping frames.** (left) The conventional way of generating videos, where frames are encoded into overlapping chunks. This results in jumps like artifacts across chunks in the generated video in regions of high frequency details. (right) Generating video with chunks 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.

showing this in the supplementary videos).

E. Training Efficiency of Our Approach

Method	Compression	Time/iter	Total iters (cumulative)	Training time (cumulative)
MagViT-v2	8×8×4	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	8×8×16	1.04s	500K	6.01 days
ProMAG (8× → 16×)		0.23s	500K	2.21 days

Table 1. 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 training efficiency overhead. In contrast, it improves the training efficiency compared to training the full model directly for high compression. From 1, 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

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 2. 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 3. 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.

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 ratios. The difference is the progressive training strategy and the residual encoding with `AdaNorm` layers. From Table 2, 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 3, 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×	8×	16×
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 4. Details of the encoder and decoder configurations of our video tokenizer ProMAG, and the discriminator configuration for different temporal compression ratios 4×, 8× and 16×. We use discriminator only for training the 4× base model.

References

[1] 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. 1

[2] 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.

[3] 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. 1

[4] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 1

[5] 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. 1

[6] 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. 1

[7] 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. 1

[8] 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. 1

[9] 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