Efficient Long Video Tokenization via Coordinate-based Patch Reconstruction

Supplementary Material

A. Implementation Details

A.1. Long video tokenization

We train CoordTok via AdamW optimizer [25] with a constant learning rate of 10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, and weight decay 0.001. We use a batch size of 256, where each sample is a randomly sampled 128-frame video. CoordTok is trained in two stages: main training and fine-tuning. In the main training stage, we reconstruct $N = 1024$ randomly sampled coordinates and update the model using ℓ_2 loss. In the fine-tuning stage, we reconstruct 16 randomly sampled frames (i.e., $N = 4096$ coordinates) and update the model using a combination of ℓ_2 loss and LPIPS loss with equal weights. To speed up training, we use mixed-precision (fp16). For the main experimental results, we train CoordTok for 1M iterations and fine-tune it for 50K iterations. For analysis and ablation studies, we train CoordTok for 200K iterations and fine-tune it for 10K iterations.

Architecture CoordTok consists of a *transformer encoder* that extracts video features from raw videos, a *cross-self encoder* that processes video features into triplane representations via cross-attention between learnable parameters and video features, and a *transformer decoder* that learns a mapping from coordinate-based representations into corresponding patches. In what follows, we describe each component in detail.

- **Transformer encoder** consists of a Conv3D patch embedding, learnable positional embedding, and transformer layers, where each transformer layer comprises self-attention and feed-forward layers.
- **Cross-self encoder** consists of plane-wise Conv2D patch embeddings, transformer layers, and plane-wise linear projectors, where each transformer layer comprises cross-attention, self-attention, and feed-forward layers.
- **Transformer decoder** consists of linear patch embedding, learnable positional embedding, transformer layers, and a linear projector, where each transformer layer comprises self-attention and feed-forward layers.

We provide the detailed architecture configurations for each model size in Table 5.

Table 5. Model configurations of CoordTok for each model size.

Model size	Module	#layers	Hidden dim.	#heads
Large	Transformer Encoder	8	1024	16
	Cross-self Encoder	24	1024	16
	Transformer Decoder	24	1024	16
Base	Transformer Encoder	8	768	12
	Cross-self Encoder	12	768	12
	Transformer Decoder	12	768	12
Small	Transformer Encoder	8	512	8
	Cross-self Encoder	8	512	8
	Transformer Decoder	8	512	8

A.2. Long video generation

We implement CoordTok-SiT-L/2 based on the original SiT implementation [27]. The inputs of SiT-L/2 are the normalized triplane representation obtained by tokenizing video clips of length 128 with CoordTok. To normalize the triplane representation, we randomly sample 2048 video clips of length 128 and calculate the mean and standard deviation for each plane. We train SiT-L/2 via AdamW optimizer [25] with a constant learning rate of 10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, and no weight decay. We use a batch size of 64. We train the model for 600K iterations and we update an EMA model with a momentum parameter 0.9999.

Architecture We use the same structure as SiT, except that our patch embedding and final projection layers are implemented separately for each plane. To train the unconditional video generation model, we assume the number of classes as 1, and we set the class dropout ratio to 0. We provide the detailed architecture configurations in Table 6.

Table 6. Model configurations of CoordTok-SiT-L/2.

	SiT-L/2, #token = 1280	SiT-L/2, #token = 3072
Input dim. ($\mathbf{z}^{xy}$)	16×16×8	32×32×8
Input dim. ($\mathbf{z}^{yt}$)	16×32×8	32×32×8
Input dim. ($\mathbf{z}^{xt}$)	16×32×8	32×32×8
# layers	24	24
Hidden dim.	1024	1024
# heads	16	16

Sampling For sampling, we use the Euler-Maruyama sampler with 250 sampling steps and a diffusion coefficient $w_t = \sigma_t$. We use the last step of the SDE sampler as 0.04.

B. Evaluation Details

B.1. Long video reconstruction

For our CoordTok, we tokenize and reconstruct 128-frame videos all at once. Specifically, we encode the video into a triplane representation and then reconstruct the video by passing all patch coordinates through the transformer decoder at once. In contrast, the baselines can only handle videos of much shorter lengths (e.g., 16 frames for PVDM-AE [66]). Therefore, to evaluate the reconstruction quality of 128-frame videos for the baselines, we split the videos into short clips and tokenize and reconstruct them. To be specific, we first split a 128-frame video into shorter clips suitable for each tokenizer. We then tokenize and reconstruct each short clip individually using the tokenizer. Finally, we concatenate all the reconstructed short clips to obtain the 128-frame video.

For evaluating the quality of reconstructed videos, we follow the setup of MAGVIT [63]. We randomly sample 10000 video clips of length 128, and then measure the reconstruction quality using the metrics as follows:

- **rFVD** [48] measures the feature distance between the distributions of real and reconstructed videos. It uses the I3D network [3] to extract features, and it computes the distance based on the assumption that both feature distributions are multivariate Gaussian. Specifically, we compute the rFVD score on video clips of length 128.
- **PSNR** measures the similarity between pixel values of real and reconstructed images using the mean squared error. For videos, we compute the PSNR score for each frame and then average these frame-wise PSNR scores.
- **LPIPS** [68] measures the perceptual similarity between real and reconstructed images by computing the feature distance using a pre-trained VGG network [36]. It aggregates the distance of features extracted from various layers. For videos, we compute the LPIPS score for each frame and then average these frame-wise LPIPS scores.
- **SSIM** [55] measures the structural similarity between real and reconstructed images by comparing luminance, contrast, and structural information. For videos, we compute the SSIM score for each frame and then average these frame-wise SSIM scores.

B.2. Long video generation

For sname-SiT-L/2, we generate the tokens corresponding to a 128-frame video all at once and then decode these tokens using CoordTok. In contrast, baselines iteratively generate 128-frame videos. For instance, PVDM and HVDM generate the next 16-frame video conditioned on the previously generated 16-frame video clip.

For evaluating the quality of generated videos, we strictly follow the setup of StyleGAN-V [39] that calculates the FVD scores [48] between the distribution of real and generated videos. To be specific, we use 2048 video clips of length 128 for each distribution, where the real videos are sampled from the dataset used to train generation models (*i.e.*, the UCF-101 dataset [42]).

B.3. Analysis

- **Dynamics magnitude** To measure how dynamic each video is, we use the pixel value differences between consecutive frames. To be specific, we compute the dynamics magnitude for each pair of consecutive frames, calculate the mean of these values, and then take the logarithm. Here, dynamics magnitude of two frames f^1 and f^2 of resolution $H \times W$ can be defined as follows:

$$d(f^1, f^2) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} d_2(f^1_{hw}, f^2_{hw}), \qquad (3)$$

  where f^i_{hw} denotes the RGB values at coordinates (h, w) of frame f^i and d_2 denotes ℓ_2-distance of RGB pixel values. In Figure 7a, we standardize the video dynamics score into a range of 0 to 100.
- **Frequency magnitude** To measure the frequency magnitude, we use the metric proposed in Yan et al. [60] that utilizes a Sobel edge detection filter. To be specific, to get the frequency magnitude, we apply both horizontal and vertical Sobel filters to each frame to compute the gradient magnitude at each pixel. We then calculate the average of these magnitudes across all pixels.

C. Baselines

C.1. Long video reconstruction

We describe the main idea of baseline methods that we used for the evaluation. We also provide the shape of tokens of baselines in Table 7.

- **MaskGiT-AE** [5] uses 2D VQ-GAN [8] that encodes an image into a 2D discrete tokens.
- **TATS-AE** [11] introduces 3D-VQGAN that compresses a 16-frame video clip both temporally and spatially into 3D discrete tokens.
- **MAGVIT-AE-L** [63] also introduces 3D-VQGAN but improves architecture design (*e.g.*, uses deeper 3D discriminator rather than two shallow discriminators for 2D and 3D separately, uses group normalization [57] and Swish activation [32]) and scales up the model size.
- **PVDM-AE** [66] encodes a 16-frame video clip into factorized triplane representations.
- **LARP** [52] encodes videos into 1D arrays by utilizing a next-token prediction model as a prior model.
- **OmniTokenizer-DV** [53] introduces image-video joint VQGAN that compresses a 17-frame video clip into 3D discrete tokens with more advanced architecture design (*e.g.*, uses both 2D and 3D patch embedding layers to support both image and video tokenization, uses transformer backbone with causal attention layers).
- **OmniTokenizer-CV** [53] uses the same architecture design as OmniTokenizer-DV, but replaces the VQ loss with KL loss so that it compresses a 17-frame video clip into 3D continuous latent vectors.

Table 7. Token shapes of video tokenization baselines

Method	Input shape	Token shape
MaskGiT-AE [5]	128×128×3	8×8
TATS-AE [11]	16×128×128×3	4×16×16
MAGVIT-AE-L [63]	16×128×128×3	4×16×16
PVDM-AE [66]	16×128×128×3	(16×16) × 3
LARP [52]	16×128×128×3	1024
OmniTokenizer [53]	(1+16)×128×128×3	(1+4)×16×16

C.2. Long video generation

We describe the main idea of baseline methods that we used for the evaluation.

- **MoCoGAN** [47] proposes a video generative adversarial network (GAN; [12]) that has a separate content generator and an autoregressive motion generator for generating videos.
- **MoCoGAN-HD** [46] also proposes a video GAN with motion-content decomposition but uses a strong pre-trained image generator (StyleGAN2 [20]) for a high-resolution image synthesis.
- **DIGAN** [65] interprets videos as implicit neural representation (INR; [38]) and trains GANs to generate such INR parameters.
- **StyleGAN-V** [39] also introduces an INR-based video GAN with a computation-efficient discriminator.
- **PVDM-L** [66] proposes a latent video diffusion model that generates videos in a projected triplane latent space.
- **HVDM** [21] proposes a latent video diffusion model that generates videos with 2D triplane and 3D wavelet representation.
- **Latte-L/2** [10] proposes a latent video diffusion transformer that generates video by processing latent vectors with alternating spatial and temporal attention layers.

D. Additional Analysis

Computational costs We provide the GPU memory usage during training in Figure 1a, and FLOPs during training in Figure 9. We find that our decoder design allows the efficient long video tokenization in terms of both GPU memory and FLOPs.

Analysis on the number of tokens We provide the reconstruction quality of CoordTok with 1280 and 3072 tokens in Table 8. Although there is no significant difference in the reconstruction quality between CoordTok with token sizes of 1280 and 3072, training SiT-L/2 with the 1280 tokens results in substantially better generation quality (see Section 3.3).

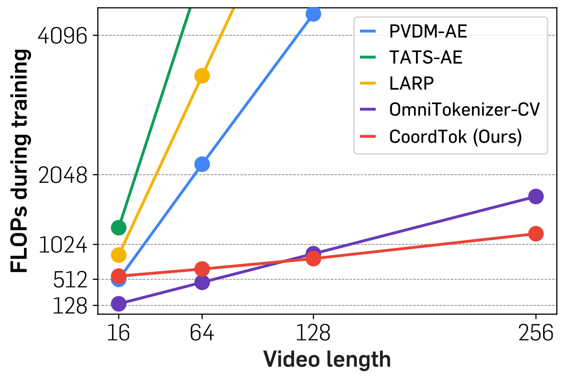

Figure 9. FLOPs when training video tokenizers on 128×128 resolution videos with varying lengths.

Table 8. Reconstruction quality of CoordTok with varying number of token sizes, evaluated on 128-frame videos. ↓ and ↑ denotes whether lower or higher values are better, respectively.

#tokens	rFVD↓	PSNR↑	LPIPS↓	SSIM↑
1280	102.9	28.6	0.066	0.892
3072	**100.5**	**28.7**	**0.065**	**0.894**

Analysis on the effect of LPIPS fine-tuning In Table 9, we investigate the effect of the additional fine-tuning phase, where we train CoordTok with both ℓ_2 loss and LPIPS loss [68] for 50K iterations after training CoordTok with ℓ_2 loss for 1M iterations. We find that fine-tuning phase improves the perceptual quality (*i.e.*, rFVD score: 188.3 $\rightarrow$ 102.9, and LPIPS score: 0.141 $\rightarrow$ 0.066), but degrades the pixel-level reconstruction quality (*i.e.*, PSNR: 30.3 $\rightarrow$ 28.6, and SSIM: 0.905 $\rightarrow$ 0.892).

Table 9. Effect of LPIPS fine-tuning phase for CoordTok. ↓ and ↑ denotes whether lower or higher values are better, respectively.

Phase	Iters	loss	rFVD↓	PSNR↑	LPIPS↓	SSIM↑
1	1M	ℓ_2	186.3	**30.3**	0.141	**0.905**
2	+50K	ℓ_2+LPIPS	**102.9**	28.6	**0.066**	0.892

E. Additional Quantitative Results

16-frame reconstruction quality To further evaluate the quality of reconstructed videos from tokenizers, we report the rFVD score on video clips of length 16 for the CoordTok and other tokenizers with varying number of token sizes in Figure 10. For evaluation, we use 10000 video clips of length 128, which are also used to measure the rFVD score on 128-frame videos. We split each 128-frame video into 16 non-overlapping sub-clips, and then compute the rFVD score on totally 80000 video clips of length 16.

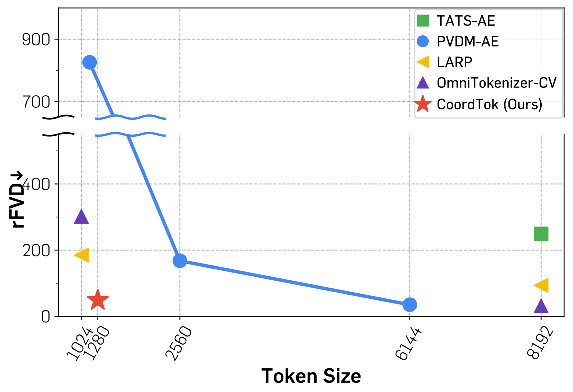

Figure 10. rFVD scores of video tokenizers, evaluated on 16-frame videos, with respect to the token size used for encoding 128-frame videos. ↓ indicates lower values are better.

F. Additional Qualitative Results

In Figure 11, we provide additional video reconstruction results from CoordTok. In addition, in Figures 12 and 13, we provide unconditional video generation results from CoordTok-SiT-L/2.

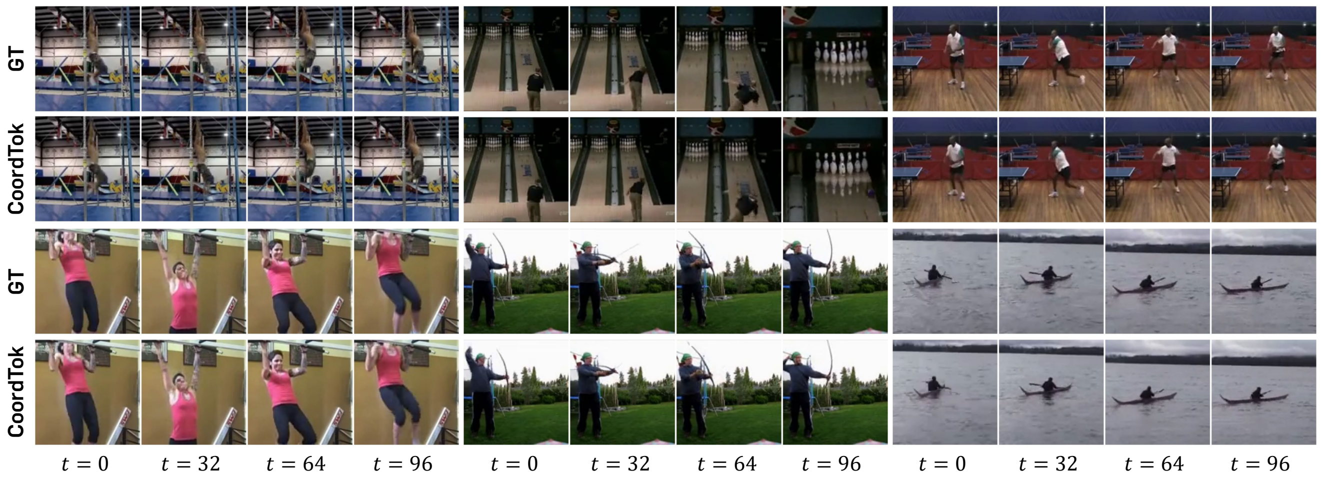

Figure 11. Additional 128-frame, 128×128 resolution video reconstruction results from CoordTok (Ours) trained on the UCF-101 dataset [42]. For each frame, we visualize the ground-truth (GT) and reconstructed pixels from CoordTok.

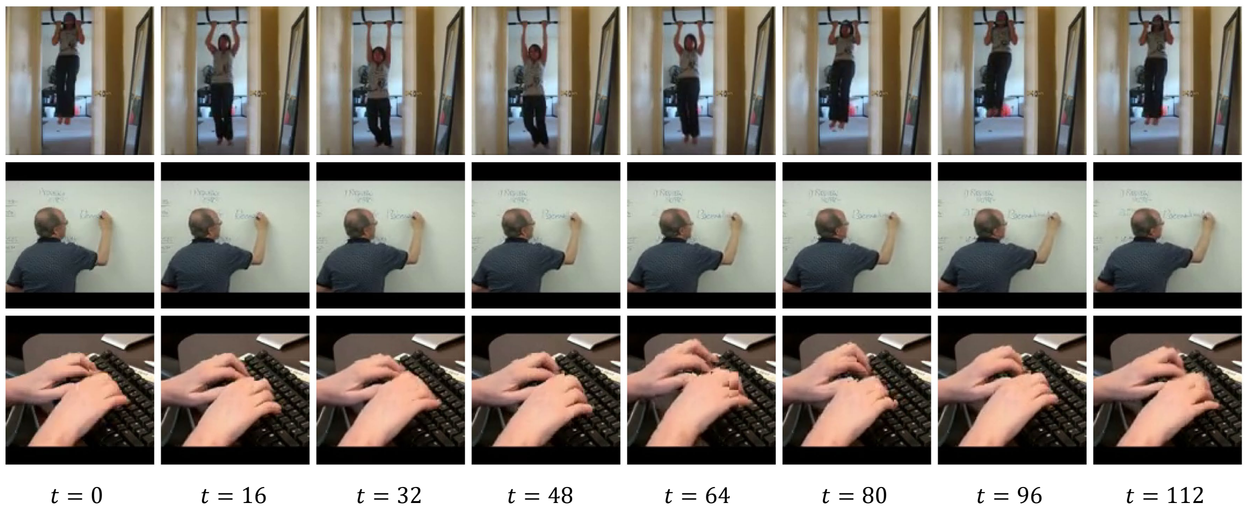

Figure 12. Unconditional 128-frame, 128×128 resolution video generation results from CoordTok-SiT-L/2 trained on 128-frame videos from the UCF-101 dataset [42].

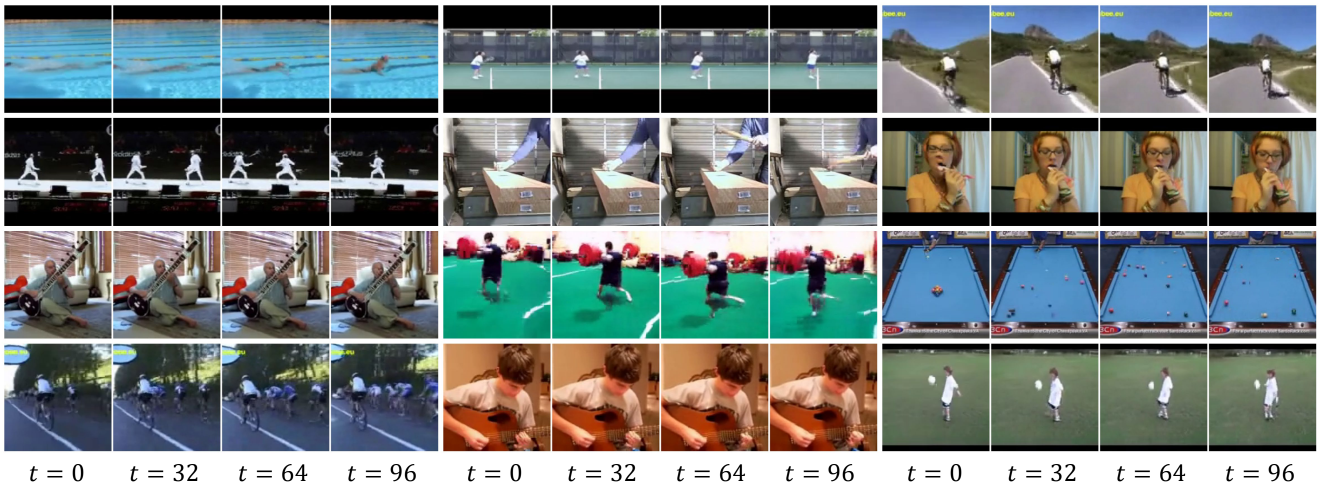

Figure 13. Unconditional 128-frame, 128×128 resolution video generation results from CoordTok-SiT-L/2 trained on 128-frame videos from the UCF-101 dataset [42].