Appendix

This document provides comprehensive descriptions and results of our method that could not be accommodated in the main paper due to space restriction. More generated samples from MeBT and the baselines [11, 31, 41, 54] can be found at https://sites.google.com/view/mebt-cvpr2023.

A. Experimental Details

In this section, we first describe the detailed implementation of training and inference in Section 3.3 in the main paper. Then, we provide additional descriptions of the datasets and baselines used in Section 5 in the main paper.

A.1. Training and Inference

Training For all experiments, we used the same network architecture except for the size of latent bottleneck N_L. Following the configuration of TATS [11], all models have 24 attention layers, 16 attention heads, 1024 embedding dimensions, and learnable positional embedding for all spatiotemporal positions. For the size of latent bottleneck, we used $N_L = 128$ for SkyTimelapse 16-frame videos, and $N_L = 256$ otherwise. We use AdamW optimizer [29] with $\beta_1 = 0.9$ and $\beta_2 = 0.95$. Table 5 summarizes our training configuration for all experiments.

Inference When sampling long videos, we decode N_d tokens by appending a single token to the context tokens N_d times before applying the iterative decoding to hold the consistency of the video. Then, the iterative decoding is applied with the N_d decoded context tokens. For the iterative decoding, we adopt a top-k sampling strategy when sampling the code from logits. We also adopt context temperature annealing when updating the context tokens to increase the diversity of samples following the official implementation of MaskGIT[4]. Specifically, let $p_i \in (0, 1)$ be the probability value of sampled token i, and s, S be the current decoding step and total decoding steps, respectively. Then, we update context tokens with top-N_s decoded tokens with the highest confidence scores $c_i = p_i + \text{Gumbel(0,1)} * (1 - \frac{s}{S})\tau$. For the revision, we repeated the revision N_R times with the logits scaled with a certain temperature. The specific configuration for each experiment is summarized in Table 6.

A.2. Datasets

SkyTimelapse SkyTimelapse [52] contains time-lapse videos that illustrate the dynamics of the sky such as the growth and motions of clouds. We used the training split to train the model and the validation split to measure the FVD. We train our model with videos that are longer than the training length. We utilized 2,115 videos for training the short-term models and 1,059 videos for training the long-term models.

Taichi-HD Taichi-HD is a collection of videos of a single actor performing Taichi. Taichi-HD has a total of 2,854 videos including both training and validation split. For Taichi-HD, we utilize both splits and sampled every 4 frames when training on 16-frame videos following the setting of [11, 54]. For 128-frame videos, we sampled all frames for training as the 4-frame sampling drops 2,438 videos over 2,854 videos. By sampling every frame, we could utilize 2,760 videos for training the 128-frame model.

UCF-101 UCF-101 is an action recognition dataset that contains 13,320 videos with 101 classes in total. We trained our short-term model on the training split with 9,537 16-frame videos and our long-term model with 6,469 128-frame videos.

A.3. Long-term Video Generation Baselines

MoCoGAN-HD MoCoGAN-HD [41] models videos by training a motion generator on the image latent space provided by a pre-trained image generator. The motion generator of MoCoGAN-HD is implemented with RNNs and thus has sub-quadratic complexity to the video length. Therefore, we trained MoCoGAN-HD with 128-frame videos for long-term comparison.

DIGAN DIGAN [54] considers a video as a function that outputs the RGB pixel value for the given spatiotemporal coordinate. DIGAN models videos with a motion discriminator that achieves constant complexity by discovering unnatural motions when two frames and the time gap between frames are given. As the discriminator gets two frames and the video is generated by a coordinate-based network, DIGAN has sub-quadratic complexity to the video length. Thus, we trained DIGAN with 128-frame videos for long-term comparison.

CCVS CCVS [31] is a video prediction model that utilizes optical flow and an autoregressive transformer. CCVS can generate longer videos by sliding the attention window, but due to the quadratic complexity of the autoregressive transformer, it cannot be directly trained with long videos. Hence, for the long-term comparison, we utilized the 16-frame CCVS model to predict 128-frame videos from randomly sampled real frames as an initial frame.

TATS TATS [11] models videos by adopting an autoregressive transformer and a time-agnostic 3d VQGAN. The proposed time-agnostic 3d VQGAN can decode longer

[4]https://github.com/google-research/maskgit

videos beyond the training length. Therefore, TATS-base can generate longer videos by sliding the attention window. However, due to the quadratic complexity of transformers, TATS-base cannot directly model long videos. To address this issue, the authors proposed TATS-hierarchical that generates long videos sparsely and interpolates the missing frames. We compared TATS-base by applying a sliding attention window, and TATS-hierarchical by training the hierarchical model on 128-frame videos.

A.4. Comparison to Sparse Attentions

This section presents more in-depth analysis and comparisons with the baselines presented in Section 5.4. Let $N(= H \times W)$ and T denote the number of tokens in spatial and temporal dimensions, respectively. Then,

1. **Axial attention** [15] alternates attention over horizontal, vertical, and temporal axes. The queries in each attention layer can only interact with tokens on the same axis. The overall complexity is $O(NT(H + W + T))$.

2. **Window attention** [12] alternates attention over spatial and temporal axes. It performs frame-wise attention over N tokens followed by spatio-temporal attention over $n \times T$ tokens where $n(= 16)$ is the size of spatial window. The complexity is $O(N^2T + nNT^2)$.

3. **Local attention** [51] computes the attention within a fixed number (n) of spatio-temporal neighborhood tokens. The complexity of local attention is $O(nNT)$.

While axial attention and window attention are asymptotically more efficient than dense attention of $O(N^2T^2)$, they still have quadratic complexity with respect to the spatial (N) or temporal dimension (T). On the other hand, local attention and MeBT with n latent codes have a linear complexity of $O(nNT)$ for both N and T. However, local attention cannot effectively model the long-term dependency of tokens due to the limited receptive fields, and its benefits are often not fully leveraged due to the lacking support in both hardware and software for the sparse operations[5]. Therefore, we only compared MeBT with axial and window attention in our ablation study in Section 5.4.

B. Empirical Analysis on Memory Complexity

We compare MeBT's empirical memory consumption over video lengths with an autoregressive transformer by measuring the training peak memory on a single A100 GPU with 80GB of VRAM with batch size 4. The results are shown in Figure 8. As shown in the figure, the training peak

[5]Most of the currently available implementations of local attention is not leveraging the sparse operators, which makes them have the same complexity as dense attention.

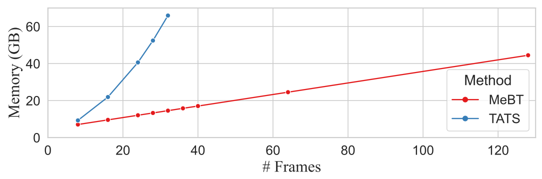

Figure 8. Empirically measured training peak memory over training video lengths. MeBT showed linear complexity to the video length, while TATS showed quadratic complexity.

memory of MeBT scaled up linearly while the autoregressive transformer [11] showed quadratic growth. In particular, with batch size 4, we couldn't train the autoregressive transformer with videos longer than 36 frames represented with 2304 tokens due to the out-of-memory error. Compared to the autoregressive transformer, our model spends about 40GB when training with 128-frame videos (8,192 tokens). It shows that MeBT is much more efficient than the transformer in long videos and capable of modeling longer videos directly.

C. Additional Qualitative Results

In this section, we extend the discussion in Section 5.3 in the main paper with additional qualitative results. The generated videos from MeBT and the baselines can be found at https://sites.google.com/view/mebt-cvpr2023.

C.1. Qualitative Comparison on SkyTimelapse

The generated videos from the baselines and MeBT are displayed in Figure 9. As shown in the figure, our model demonstrates consistent high-fidelity sky videos. To be specific, compared to TATS-base and TATS-hierarchical, our model showed better consistency on modeling the static ground. This is because our model can decode the tokens in a bidirectional manner. As autoregressive transformers follow the raster-scan ordering, the tokens that represent the ground in the previous frame and current frame are far away on the sequence. Compared to DIGAN [54] and MoCoGAN-HD [41], MeBT exhibits high-fidelity video generation while DIGAN shows unrealistic artifacts and MoCoGAN-HD collapses when generating long-term frames.

C.2. Qualitative Comparison on Taichi-HD

The generated videos on Taichi-HD are shown in Figure 10. Unlike other baselines, MeBT could model the non-linear motions in Taichi by connecting the basic actions. Specifically, DIGAN showed difficulties in generating non-linear motions and MoCoGAN-HD could not keep the background and actor consistently. The frames generated by TATS-base become brighter as the video goes longer due to the error propagation, and TATS-hierarchical showed temporally ziggy motions by adopting two separately trained transformers. The ziggy motions of TATS-hierarchical are best viewed on the website.

C.3. Qualitative Comparison on UCF-101

The generated videos on UCF-101 are displayed in Figure 11. On the complex UCF-101 dataset, DIGAN and MoCoGAN-HD failed to model complex structures such as human faces. On the other hand, transformer-based models (TATS, MeBT) could model complex structures. However, TATS-base showed unnatural artifacts due to the error propagation, and TATS-hierarchical showed inconsistent quality between the keyframes generated by the hierarchical transformer and the interpolated frames. Compared to the baselines, MeBT generates complex motion that combines both camera motion and horse riding.

Table 5. Training configuration for all experiments. Subscripts denote the length of training videos.

Dataset	SkyTimelapse$_{16}$	Taichi-HD$_{16}$	UCF-101$_{16}$	SkyTimelapse$_{128}$	Taichi-HD$_{128}$	UCF-101$_{128}$
Batch size	24	128	128	40	48	48
Learning rate	1.08e-5	3e-5	3e-5	1.8e-5	3e-5	3e-5
Training steps	400k	750k	2.6M	500k	3M	3.55M
Dropout rate	0.1	0	0	0.1	0	0

Table 6. Decoding configuration for all experiments. Subscripts denote the length of training videos.

Dataset	SkyTimelapse$_{16}$	Taichi-HD$_{16}$	UCF-101$_{16}$	SkyTimelapse$_{128}$	Taichi-HD$_{128}$	UCF-101$_{128}$
N_d	0	0	0	64	64	64
S	32	64	128	32	32	32
top-k	-	-	-	32	32	32
γ	cosine	cosine	cosine	cosine	cosine	cosine
τ	8	2	6	4	4	2
R	2	2	4	2	2	32
temperature	0.7	0.3	0.7	0.7	0.1	0.1
N_R	2	8	4	2	4	2

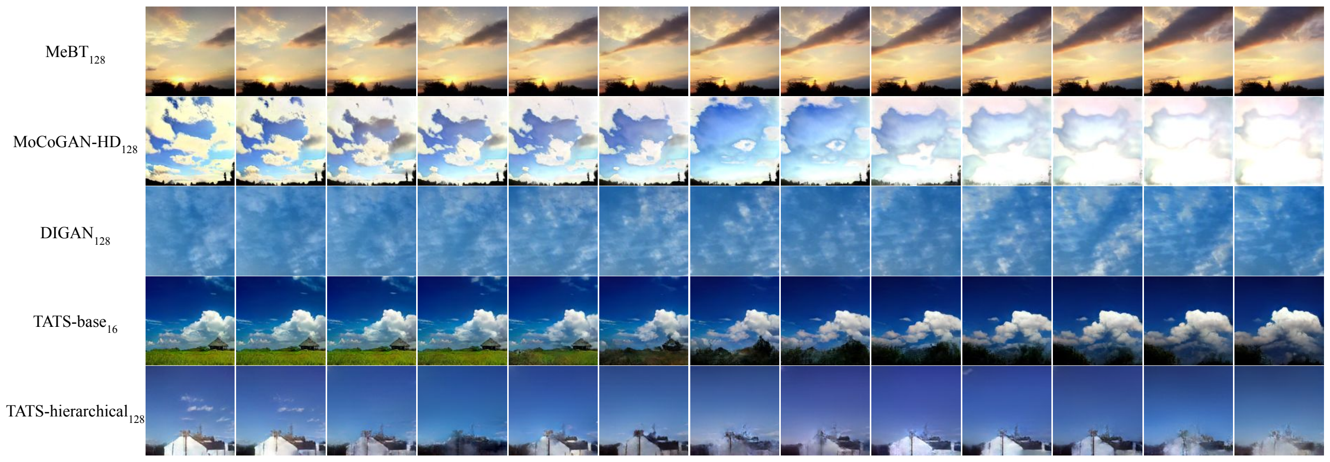

Figure 9. Generated videos on SkyTimelapse. We displayed every 10th frame in the generated video. The subscript denotes the length of training videos. More samples can be found on the website.

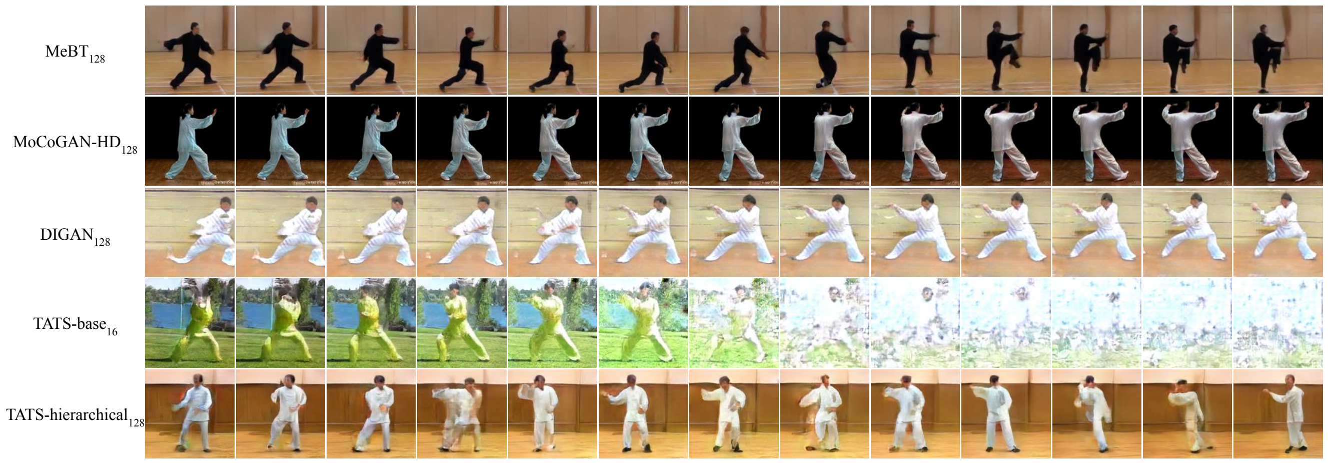

Figure 10. Generated videos on Taichi-HD. We displayed every 10th frame in the generated video. The subscript denotes the length of training videos. More samples can be found on the website.

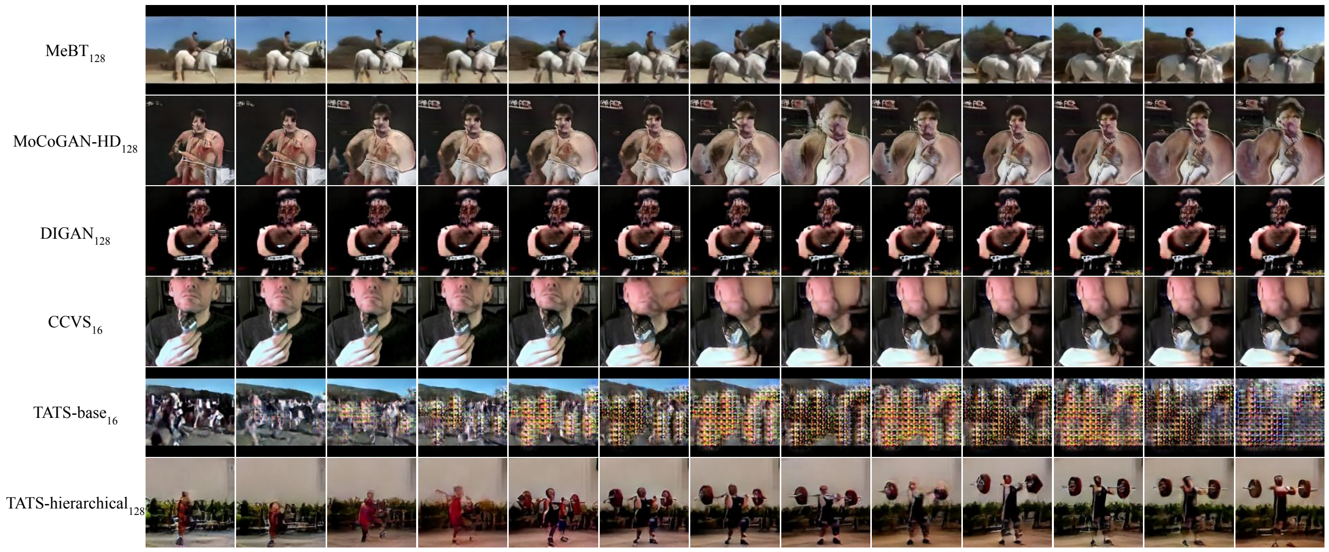

Figure 11. Generated videos on UCF-101. We displayed every 10th frame in the generated video. The subscript denotes the length of training videos. More samples can be found on the website.