A HYPER-PARAMETERS

Symbol	Value	Description
t_x, w_x, h_x, c_x	$11, 128, 128, 3$	Video dimensions
t_p, w_p, h_p, c_p	$2, 8, 8, 3$	Patches dimensions (all frames except the first one)
t_z, w_z, h_z	$6, 16, 16$	Video tokens dimension (before linear projection)
h_z	512	Hidden size in the transformer layer
d_z	32	Embedding dimension (after linear projection)
$-$	4	Number of layers for spatial transformer
$-$	4	Number of layers for temporal transformer
$-$	2048	MLP size
$\|E\|$	8192	Codebook size
-	AdamW	Optimizer
β_1	0.9	first moment of gradient
β_2	0.99	second moment of gradient
-	1e-4	Learning rate
-	1e-4	Weight decay
-	Cosine decay	Learning rate scheduler
-	1M	Target number of training steps for learning rate scheduler
-	100K	Warmup steps
-	10	Gradient clipping magnitude
-	1028	Batch size

Table 6. Hyperparamters used for C-ViViT architecture and optimizer.

Symbol	Value	Description
$\|\mathbf{z}\|$	1536	Sequence Length
-	24	Number of layer
-	2048	Embedding dimension
-	8192	MLP dimension
-	32	Number of heads
-	AdamW	Optimizer
β_1	0.9	first moment of gradient
β_2	0.99	second moment of gradient
-	1e-4	Learning rate
-	1e-4	Weight decay
-	Cosine decay	Learning rate scheduler
-	4M	Target number of training steps for learning rate scheduler
-	10K	Warmup steps
-	10	Gradient clipping magnitude
-	512	Batch size

Table 7. Hyperparamters used for MaskGIT architecture and optimizer.

B DETAILS OF EXPERIMENTS

B.1 VIDEO QUANTIZATION

B.1.1 NETWORK ARCHITECTURE

All encoder-decoder baselines have approximately 50M parameters. The Convolutional baseline encoder architecture consists of 5 convolutional blocks with channel multipliers of $[1, 1, 2, 2, 4]$, 2 residual layers and 128 hidden units per block, and embedding dimension of 256. The ViT baseline encoder architecture consists of an image patchification step over non-overlapping 8×8 spatial patches which are linearly transformed into image tokens. Next, we follow with 8 transformer layers with 512 hidden units, 8 attention heads, 2048 mlp units, and embedding dimension of 32. C-ViViT encoder architecture patches the first frame to non-overlapping 8×8 patches, and then the rest of

the frames to non-overlapping $2 \times 8 \times 8$ spatio-temporal patches which are linearly transformed into video embeddings. Next, C-ViViT encoder architecture consists of 4 spatial and 4 temporal transformer layers with 512 hidden units, 8 attention heads, 2048 mlp hidden units, and embedding dimension of 32. The decoder architecture for all models is the same as the encoder but in reverse to put the latent embeddings back to image space. The VQ objective is trained with commitment loss of $\beta = 0.25$ and codebook size of 8192. The discriminator architecture is the StyleGAN [25] discriminator with blur resample, and channel multiplier of 1.

B.1.2 TRAINING

We train all encoder-decoder baselines and with StyleGAN [25] discriminators with a batch size of 128 using Adam optimizer [27] with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. We use a linear learning rate warmup to a peak value of 1×10^{-4} over $100,000$ steps and then decaying over the remaining $900,000$ steps with a cosine schedule, and use a decoupled weight decay [30] of 1×10^{-4} for the encoder-decoder and discriminator. To capture longer time horizons during training and better evaluate temporal coherence, we downsample the MiT dataset from 25 FPS to 6 FPS and evaluate on videos of 10 frames at spatial resolution of 128×128.

B.2 IMAGE CONDITIONAL VIDEO GENERATION

B.2.1 BAIR ROBOT PUSH C-VIVIT ARCHITECTURE

We use a similar setup as in Section B.1, but the video tokenization step is done over 4×4 spatial patches on the first image and $2 \times 4 \times 4$ spatio-temporal patches in the rest of the video. The spatial encoder consists of 8 layers and the temporal encoder consists of 6 layers.

B.2.2 KINETICS-600 C-VIVIT ARCHITECTURE

We use a similar setup as in Section B.2.1, but both the spatial encoder and temporal encoder consist of 8 layers.

B.2.3 MASKGIT ARCHITECTURE

To perform video prediction in latent space in the BAIR Robot Push and Kinetics-600 datasets, we use an unconditional transformer architecture consisting of 24 layers, 768 hidden units, 16 attention heads, dropout and attention dropout rate of 0.1, 3072 mlp hidden units.

B.2.4 TRAINING AND INFERENCE

As described in Table 7, we train C-ViViT with the same optimizer setup as in Sec B.1, but we do not downsample the FPS of any of the datasets in this section for fair comparison with the video prediction baselines. We train MaskGIT on the video tokens extracted using C-ViViT in an unconditional setting, that is, we do not assume frames or text inputs to be given. During training, we use the Adam [27] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. We use a linear learning rate warmup up to a peak value of 1×10^{-4} over $10,000$ steps, and constant learning rate schedule for $\sim 2M$ steps. At inference time, we initialize MaskGIT given a number of input frames, and predict the rest of the frames depending on the dataset on which we evaluate.

B.3 TEXT CONDITIONAL VIDEO GENERATION

B.3.1 ARCHITECTURE

In our text conditional video generation, we use the same C-ViViT architecture and training described in Section B.1. To train MaskGIT, we include a text conditioning in the form of T5X embeddings [41] which are used as input through the use of cross attention with the video tokens. We reduce the number of parameters of our base model for fairness in the quantitative comparisons against NUWA. The MaskGIT architecture used against NUWA consists of 20 transformer layers with 1536 hidden units, 24 attention heads, and 6144 MLP hidden units, resulting in 0.9B parameters similar to NUWA. For the main experiments in this paper, we use a larger architecture that

consists of consists of 24 transformer layers with 2048 hidden units, 32 attention heads, and 8192 mlp hidden units, resulting in 1.8B parameters.

B.3.2 TRAINING AND INFERENCE

For all our text-conditional video generation, we use the training parameters Table 7.

B.3.3 INFERENCE PARAMETERS AGAINST NUWA

We use $\lambda = 0.1$, 12 MaskGIT iterations, and temperature of 4.0.

B.3.4 INFERENCE PARAMETERS FOR ABLATION OF IMAGE AND VIDEO DATA FOR TRAINING.

We use $\lambda = 6$, 24 MaskGIT iterations, and temperature of 4.0.

B.3.5 INFERENCE PARAMETERS FOR ALL VIDEOS IN THE PAPER.

We use $\lambda = 12$, 48 MaskGIT iterations, and temperature of 8.0.

Figure 5. Another example of story conditional video generation. Full videos are available at phenaki.github.io.

C OTHER EXPERIMENTS

C.1 CLASS CONDITIONAL VIDEO GENERATION

In order to compare Phenaki with more previous work, we train a smaller version of the model with 345M parameters on UCF-101 [47] while conditioning it on the class of each video. As it can be

Table 8. Quantitative results of video generation on class conditional UCF-101 [47]. We report FVD numbers of a smaller version of Phenaki with 345M parameters The numbers for other method after 540K training steps. The FVD numbers for previous methods are taken from [16].

Method	FVD↓
TGANv2 [44]	1209
CogVideo [22]	626
TATS [16]	332
Phenaki (Ours)	250

Table 9. Training and inference speed of Phenaki.

Size	Condition	Training				MaskGIT Iterations	Inference Time
		C-ViViT		MaskGiT			
		Number of TPUs	Training Time	Number of TPUs	Training Time		
1.8B	Text	64	81 hours	512	126 Hours	24	4 FPS
345M	Class	64	65 hours	64	49 Hours	128	2 FPS

seen in Table 8, Phenaki outperforms all of the previous work. We use a temperature of 6.0, 128 MaskGIT iterations during sampling, and no classifier free guidance during training or evaluation for simplicity in our experiments.

D Computational Cost

As mentioned in the paper, training Phenaki has two main stages. First, training C-ViViT which takes $\sim$81 hours on 64 TPUs and second training MaskGIT which takes $\sim$126 hours on 512 TPUs. Our 1.8B parameter text-conditioned model runs 24 MaskGIT iterations during sampling and generates video at $\sim$4 frames per second. In addition, we provide a summary of the computational cost for the 345M parameter class-conditioned model used in Section C which runs 128 MaskGIT iterations during sampling and generates videos at $\sim$2 frames per second. A summary of these numbers can be seen in Table 9.

E Stylization from image datasets

To highlight the stylization learned from images in our model, we provide comparisons of generations from the model trained only with video data and with a combination of video and image datasets in our website phenaki.github.io/style_videos.html.

F Long video generation

To highlight the long term generation in our model, we provide 5 minute videos given the same prompt during the entire generation in our website phenaki.github.io/five_min_videos.html.