Make-A-Video

A Preprint

1 Overall contributions

- Extending text to image diffusion models to text to video hrough a spatiotemporally factorized diffusion model
- Joint text-image priors to bypass the need for paired text-video data

2 Method

(1) Make-A-Video consists of three main components: T2I model trained on text-image pairs

(2) spatiotemporal convolution and attention layers that extend the networks' building blocks to the temporal dimension

(3) Spatiotemporal networks that consist of both spatiotemporal layers, as well as another crucial element needed for T2V generation - a frame interpolation network for high frame rate generation

Inference is formulated as:

$$\hat{y}_t = \mathrm{SR}_h \circ \mathrm{SR}_t^l \circ \uparrow_F \circ D_t \circ P \circ (\hat{x}, C_x(x))$$

where:

$$\hat{y}_t = \text{generated video} \qquad \mathrm{SR}_h = \text{spatial network} \qquad \mathrm{SR}_t = \text{spatiotemporal network} \tag{1}$$

$$\uparrow_F = \text{frame interpolation network} \qquad D^t = \text{spatiotemporal decoder} \qquad P = \text{Prior} \tag{2}$$

$$\hat{x} = \text{BPE-encoded text} \qquad C_x = \text{CLIP text encoder} \tag{3}$$

2.1 Text-to-image model

Algorithm 1 Text-to-Image Generation Pipeline

Input: Text embeddings x_e, BPE-encoded text tokens $\hat{x}$
Step 1: Prior Network
Generate image embeddings y_e using the prior network P:

$$y_e = P(x_e, \hat{x})$$

Step 2: Decoder Network
Generate a low-resolution image $\hat{y}_l \in \mathbb{R}^{64\times64\times3}$ using decoder D:

$$\hat{y}_l = D(y_e)$$

Step 3: Super-Resolution
Increase resolution to 256×256 using SR_l:

$$\hat{y}_m = SR_l(\hat{y}_l)$$

Further upscale to 768×768 using SR_h:

$$\hat{y} = SR_h(\hat{y}_m)$$

Output: Final high-resolution image $\hat{y}$

2.2 Spatiotemporal layers

The original decoder's job is to take an image embedding y_e, which is derived from the input text, and generate a single low-resolution 64x64 RGB image(U-Net-based diffusion network)

To make this decoder work for video, it needs to understand and generate sequences over time. This is where the "adaptation"comes in, transforming D into D_t. In order to include not only height and width, but also temporal information authors suggest Pseudo-3D Convolutional Layers and Pseudo-3D Attention Layers

2.3 Pseudo-3D CONV

The suggested method is to stack 1D Conv right after every 2D Conv in order to avoid heavy computational load of 3D conv. Let h be input tensor. Then

$$Conv_{P3D}(h) = Conv_{1D}(Conv_{2D}(h) \circ T) \circ T$$

$Conv_{2D}$ is initialized from he pre-trained T2I model while $Conv_{1D}$ is initialized as the identity function, enabling a seamless transition from training spatial-only layers, to spatiotemporal layers.

2.4 Pseudo-3D Attention

A key component of text-to-image (T2I) networks is the attention layer. In addition to applying self-attention over extracted visual features, these layers incorporate textual information into multiple levels of the network, alongside other inputs such as the diffusion timestep. While 3D convolutional layers are already computationally expensive, extending attention layers to include a temporal dimension leads to prohibitive memory usage.

Inspired by the approach of Ho et al. (2022), we extend our dimension decomposition strategy to attention modules. Specifically, following each pre-trained spatial attention layer, we introduce an additional temporal attention layer, thus approximating full spatiotemporal attention as in the pseudo-3D convolution scheme.

Let h be the input tensor. We define the operator `flatten` to reshape the spatial dimensions, such that:

$$h' = \texttt{flatten}(h) \in \mathbb{R}^{B\times C\times(F\cdot H\cdot W)},$$

and its inverse as `unflatten`. The pseudo-3D attention layer is then given by:

$$\mathrm{ATTN}_{\mathrm{P3D}}(h) = \texttt{unflatten}\left(\mathrm{ATTN}_{\mathrm{1D}}\left(\mathrm{ATTN}_{\mathrm{2D}}\left(\texttt{flatten}(h)\right) \circ T\right) \circ T\right),$$

where $\circ T$ denotes conditioning on the temporal information (e.g., the diffusion timestep).

To facilitate spatiotemporal consistency during training, we initialize ATTN_{2D} using weights from a pre-trained T2I model, while ATTN_{1D} is initialized as the identity function.

2.5 Frame Rate Conditioning

Authors also suggest adding an additional conditioning parameter fps, representing the number of frames-per second in a generated video

So, the first step is usually to convert this scalar fps value into an embedding vector. This could be done using: A learned embedding layer (if fps values are treated as discrete categories or binned). A fixed sinusoidal embedding (similar to how positional encodings in Transformers or time-step embeddings in diffusion models are often created), which can represent continuous or a range of numerical values well. A small multi-layer perceptron (MLP) that transforms the scalar fps into an embedding vector.

2.6 Frame interpolation network

This is designed to increase the number of frames in a generated video.

It consists of:

Frame Interpolation: It can insert frames between existing frames to create a smoother video.

Frame Extrapolation: It can add frames before the beginning or after the end of a video to extend its length.

This network is Finetuned D_t, during the fine-tuning for masked frame interpolation, the U-Net (within Dt) receives an additional 4 input channels (3 for RGB and 1 for mask indicating precisely which frames are known and which frames need to be predicted by the network)

Algorithm 2 Video Generation Pipeline without Paired Text–Video Training

Input:
- Text prompt x
- CLIP text encoder $\mathcal{C}_x(\cdot)$
- BPE tokenizer (producing $\hat{x}$)

Output: Generated high-resolution video $\hat{y}_t$

Stage 1: Text Processing and Initial Image Generation

[label=0.]Encode the input text:

$$e_x \leftarrow \mathcal{C}_x(x), \quad \hat{x} \leftarrow \text{BPE-encode}(x).$$

Prior Network P:

$$y_e \leftarrow P\big(e_x, \hat{x}\big)$$

(generates an image embedding y_e from text embeddings e_x and tokens $\hat{x}$) Decoder Network D:

$$\hat{y}_l \leftarrow D\big(y_e\big), \quad \hat{y}_l \in \mathbb{R}^{64\times64\times3}$$

(produces a low-resolution image from y_e)

Stage 2: Spatiotemporal Extension for Video Frame Generation

[label=0.]Modify the T2I architecture to handle temporal information:

1. • Pseudo-3D Convolutions:
   - For each pretrained 2D spatial convolutional layer Conv_{2D}, append a 1D temporal convolution Conv_{1D} initialized as identity.
   - Together they approximate a 3D convolution:

$$\text{Conv}_{\text{P3D}}(h) \;=\; \text{Conv}_{1D}\Big(\text{Conv}_{2D}(h)\Big),$$

   where Conv_{2D} is loaded from the pretrained T2I model and Conv_{1D} starts as identity.
   - Pseudo-3D Attention Layers:
   - After each pretrained 2D spatial attention Attn_{2D}, stack a 1D temporal attention Attn_{1D} initialized as identity.
   - For input tensor h, define

$$\text{Attn}_{\text{P3D}}(h) \;=\; \texttt{unflatten}\Big(\text{Attn}_{1D}\big(\text{Attn}_{2D}(\texttt{flatten}(h))\big)\Big),$$

   where `flatten` collapses spatial dimensions and `unflatten` reverses it.
2. Spatiotemporal Decoder D_t:

$$\{\hat{y}_l^{(i)}\}_{i=1}^{F} \leftarrow D_t\big(y_e\big), \quad \hat{y}_l^{(i)} \in \mathbb{R}^{64\times64\times3},\ F = 16$$

   (generates a sequence of F low-resolution frames from y_e; fine-tuned on unlabeled video)
3. Frame-Rate Conditioning:

   Introduce an additional embedding or token that encodes desired frames-per-second (fps). Use this conditioning signal during D_t's training as data augmentation.

Stage 3: Enhancing Video Quality and Frame Rate

[label=0.]Frame Interpolation Network $\uparrow F$:

$$\{\tilde{y}_l^{(j)}\}_{j=1}^{F'} = \uparrow F\Big(\{\hat{y}_l^{(i)}\}_{i=1}^{F}\Big), \quad F' = 76$$

(upsamples $F = 16$ frames to $F' = 76$ frames via masked frame interpolation; fine-tuned from D_t) Spatiotemporal Super-Resolution SR_t:

$$\{\tilde{y}_m^{(j)}\}_{j=1}^{F'} \;=\; SR_t\Big(\{\tilde{y}_l^{(j)}\}_{j=1}^{F'}\Big), \quad \tilde{y}_m^{(j)} \in \mathbb{R}^{256\times256\times3}$$

(increases resolution to 256×256 while preserving temporal consistency) Final Spatial Super-Resolution SR_h:

$$\{\hat{y}^{(j)}\}_{j=1}^{F'} \;=\; SR_h\Big(\{\tilde{y}_m^{(j)}\}_{j=1}^{F'}\Big), \quad \hat{y}^{(j)} \in \mathbb{R}^{768\times768\times3}$$

(upsamples each frame spatially to 768×768; uses identical noise initialization per frame to encourage consistent details)

Final Inference:

$$\{\hat{y}^{(j)}\}_{j=1}^{F'} \;=\; SR_h \,\circ\, SR_t \circ \uparrow F \,\circ\, D_t \,\circ\, P\,\Big(\hat{x}, \mathcal{C}_x(x)\Big)$$