Vid Gen

Authors

Affiliations

August 16, 2024

1 Motivation

A video with f frames is denoted as $\mathbf{x} = [\boldsymbol{x}^1, \cdots, \boldsymbol{x}^f] \in \mathbb{R}^{f \times c \times h \times w}$, where $\boldsymbol{x}^i \in \mathbb{R}^{c \times h \times w}$ denotes a frame (an image), we would like to build a diffusion model for the video distribution $p(\mathbf{x})$.

1.1 Autoregressive

We can autoregressively decompose the video distribution as:

$$p(\mathbf{x}) = \prod_{i=1}^{f} p(\boldsymbol{x}^i \mid \boldsymbol{x}^{<i}) \tag{1}$$

where $\boldsymbol{x}^{<i} = [\boldsymbol{x}^1, \cdots, \boldsymbol{x}^{i-1}]$ denotes the first $i-1$ frames. In this case, we would like to cast a pretrained T2I model to represent the conditional distribution. This can be simply achieved by modeling the autoregressively conditional score:

$$\nabla_{\boldsymbol{x}^i} \log p(\boldsymbol{x}^i \mid \boldsymbol{x}^{<i}, \boldsymbol{c}), \tag{2}$$

where $\boldsymbol{c}$ denotes the text prompt.

1.2 Distributed Noisy Frame Conditioning (Deprecated)

The score function $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$ is our modeling target, where $\mathbf{x}_t$ denotes the noisy video at timestep t. The dimension of the score function is f times larger than an image diffusion model and is unaffordable.

The key observation is that the score function of the noisy video $\mathbf{x}_t = (\boldsymbol{x}_t^1, \cdots, \boldsymbol{x}_t^f)$ can be rewrote as:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \begin{bmatrix} \nabla_{\boldsymbol{x}_t^1} \log p_t(\mathbf{x}_t) \\ \vdots \\ \nabla_{\boldsymbol{x}_t^f} \log p_t(\mathbf{x}_t) \end{bmatrix} = \begin{bmatrix} \nabla_{\boldsymbol{x}_t^1} \left(\log p_t(\boldsymbol{x}_t^1 \mid \boldsymbol{x}_t^{-1}) + \log p_t(\boldsymbol{x}_t^{-1}) \right) \\ \vdots \\ \nabla_{\boldsymbol{x}_t^f} \left(\log p_t(\boldsymbol{x}_t^f \mid \boldsymbol{x}_t^{-f}) + \log p_t(\boldsymbol{x}_t^{-f}) \right) \end{bmatrix} = \begin{bmatrix} \nabla_{\boldsymbol{x}_t^1} \log p_t(\boldsymbol{x}_t^1 \mid \boldsymbol{x}_t^{-1}) \\ \vdots \\ \nabla_{\boldsymbol{x}_t^f} \log p_t(\boldsymbol{x}_t^f \mid \boldsymbol{x}_t^{-f}) \end{bmatrix}. \tag{3}$$

Thus we can **distributedly** model the video score by modeling the conditional score $\nabla_{\boldsymbol{x}_t^i} \log p(\boldsymbol{x}_t^i \mid \boldsymbol{x}_t^{-i})$, and aggregate them into $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ during sampling.

And we would like to build $\nabla_{\boldsymbol{x}_t^i} \log p(\boldsymbol{x}_t^i \mid \boldsymbol{x}_t^{-i})$ with a pretrained image diffusion model.

2 Method

2.1 Distributed Noisy Frame Conditioning (Deprecated)

Most pretrained image diffusion models, e.g. Stable Diffusion, PixArt, are all trained as a DDPM. Consider a DDPM $\epsilon_\theta(\boldsymbol{x}_t, t)$ that tries to predict the noise ϵ added to its clean version $\boldsymbol{x}_0$, its link to the score function is:

$$\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) = -\frac{\epsilon_\theta(\boldsymbol{x}_t, t)}{\sigma_t^2}, \tag{4}$$

and what we want is:

$$\nabla_{\boldsymbol{x}_t^i} \log p(\boldsymbol{x}_t^i \mid \boldsymbol{x}_t^{-i}) = -\frac{\epsilon_\theta(\boldsymbol{x}_t^i, t \mid \boldsymbol{x}_t^{-i})}{\sigma_t^2} = -\frac{\epsilon_\theta(\boldsymbol{x}_t^i, t \mid \boldsymbol{x}_t, \boldsymbol{x}_t^{-i}, i)}{\sigma_t^2} = -\frac{\epsilon_\theta(\boldsymbol{x}_t^i, t \mid \mathbf{x}_t, i)}{\sigma_t^2}, \tag{5}$$

so what we need to do is build a model to summarize $(\mathbf{x}_t, i)$ and feed it into a pretrained image diffusion model $\epsilon_\theta(\boldsymbol{x}_t, t)$ as additional conditions in addition to the text prompts.

Note that $\mathbf{x}_t$ is a noisy video, which can be very memory-costing and redundancy to be directly feeded into the image diffusion model, we want to first build a model that can summarize $\mathbf{x}_t \in \mathbb{R}^{f \times c \times h \times w}$ to a fixed number of context tokens $\mathbf{c} \in \mathbb{R}^{\ell \times d}$, where ℓ denotes the number of tokens and d denotes the dimension of each token.

References

Z. Deng, J. Shi, and J. Zhu. Neuralef: Deconstructing kernels by deep neural networks. In International Conference on Machine Learning, pages 4976–4992. PMLR, 2022.

[Deng et al., 2022]