Hierarchical Patch Diffusion Models for High-Resolution Video Generation

Ivan Skorokhodov[1,2]   Willi Menapace[1,3]   Aliaksandr Siarohin[1]   Sergey Tulyakov[1]

Snap Inc.[1]   KAUST[2]   University of Trento[3]

Abstract

Diffusion models have demonstrated remarkable performance in image and video synthesis. However, scaling them to high-resolution inputs is challenging and requires restructuring the diffusion pipeline into multiple independent components, limiting scalability and complicating downstream applications. In this work, we study patch diffusion models (PDMs) — a diffusion paradigm which models the distribution of patches, rather than whole inputs, keeping up to ≈0.7% of the original pixels. This makes it very efficient during training and unlocks end-to-end optimization on high-resolution videos.

We improve PDMs in two principled ways. First, to enforce consistency between patches, we develop deep context fusion — an architectural technique that propagates the context information from low-scale to high-scale patches in a hierarchical manner. Second, to accelerate training and inference, we propose adaptive computation, which allocates more network capacity and computation towards coarse image details. The resulting model sets a new state-of-the-art FVD score of 66.32 and Inception Score of 87.68 in class-conditional video generation on UCF-101 256^2, surpassing recent methods by more than 100%. Then, we show that it can be rapidly fine-tuned from a base 36×64 low-resolution generator for high-resolution $64 \times 288 \times 512$ text-to-video synthesis. To the best of our knowledge, our model is the first diffusion-based architecture which is trained on such high resolutions entirely end-to-end. Project webpage: https://snap-research.github.io/hpdm.

1. Introduction

Recently, diffusion models (DMs) have achieved remarkable performance in image and video synthesis, greatly surpassing previous dominant generative paradigms, such as GANs [16], VAEs [33] and autoregressive models [5]. However, scaling them to high-resolution inputs broke their end-to-end nature, since training the full-scale monolithic foundational generator led to infeasible computational demands [23, 44]. Splitting the architecture into several stages
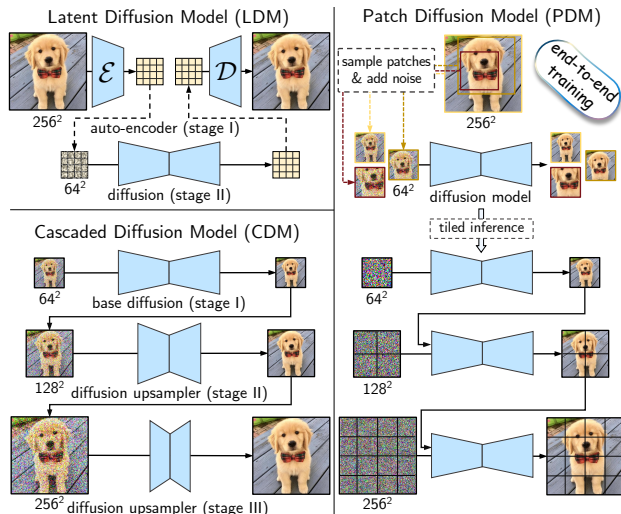

Figure 1. Comparing existing diffusion paradigms: Latent Diffusion Model (LDM) [44, 60] (upper left), Cascaded Diffusion Model (CDM) [23] (bottom left), and Patch Diffusion Model (this work) during training (upper right) and inference (bottom right). In our work, we develop *hierarchical* patch diffusion, which never operates on full-resolution inputs, but instead optimizes the lower stages of the hierarchy to produce spatially aligned context information for the later pyramid levels to enforce global consistency between patches.

satisfied the immediate practical needs, but having multiple components in the pipeline makes it harder to tune and complicates downstream tasks like editing or distillation.

For example, LDM [44] trains a diffusion model in the latent space of an autoencoder, which requires an additional extensive hyperparameters search. The original work has dedicated more than a dozen experiments to it (see Tab. 8 of [44]), and the search for its optimal design is still ongoing [3, 8, 41]. Moreover, retraining an auto-encoder requires retraining the latent generator, resulting in extra computational costs. Also, having multiple components complicates downstream applications: for example, SnapFusion [35] had to come with two unrelated sets of techniques to distill the generator and the auto-encoder separately.

Table 1. Efficiency comparison between patch-wise and full-resolution diffusion in the RIN [27] framework (which scales more gracefully with the input size than UNets [9, 45]). Memory consumption is measured in GB for the batch size of 1; speed as videos/sec for a maxed-out batch size on NVidia A100 80GB.

Method	64×256^2		64×512^2	
	Mem $\downarrow$	Speed $\uparrow$	Mem $\downarrow$	Speed $\uparrow$
Full-resolution DM	65.3	1.24	OOM	OOM
HPDM (32×128^2 patch size)	29.0	2.64	41.3	1.55
+ adaptive computation	23.4	3.58	29.9	2.49
HPDM (16×64^2 patch size)	18.1	4.25	22.1	2.78
+ adaptive computation	14.2	6.71	16.3	4.96

Cascaded DM (CDM) [23] sequentially trains several diffusion models of increasing resolution, where each next DM is conditioned on the outputs of the previous one. This framework enjoys a more independent nature of its components, where each generator is trained independently from the rest, but it has more modules in the pipeline (e.g., ImagenVideo [22] consists of 7 video generators) and more expensive inference. An end-to-end design is a highly desirable property of a diffusion generator, from the perspectives of both practical importance and conceptual elegance.

The main obstacle to moving a standard high-resolution DM onto end-to-end rails is an increased computational burden. In the past, patch-wise training proved successful for GAN training for high-resolution image [52], video (e.g., [53, 70]) and 3D (e.g., [48, 54]) synthesis, but, however, has not picked up much momentum in the diffusion space. To our knowledge, PatchDiffusion [64] and MaskDIT [73] are the only works that explore it, but none of them considers the required level of input sparsity to scale to high-resolution videos: PatchDiffusion still relies on full-resolution training for 50% of its optimization (so it is not purely patch-wise), while MaskDIT preserves ≈50% of the original input. In our work, we explore patch diffusion models while keeping just *up to 0.7% of the original pixels*. The comparison of patch-wise training and conventional paradigms is depicted in Fig. 1, and in Table 1, we show that it can achieve ×5 larger throughput and is trainable on high-resolution videos. We focus on video synthesis since, for videos, the computational burden of high resolutions is considerably more pronounced than for images: there now exist end-to-end image diffusion models that are able to train even on 1024^2 resolution (e.g., [7, 19, 26, 32]).

For our patch-wise training, we consider a hierarchy of patches instead of treating them independently [64], which means that the synthesis of high-resolution patches is conditioned on the previuosly generated low-resolution ones. It is a similar idea to cascaded DMs [23] and helps to improve the consistency between patches and simplifies noise scheduling for high resolutions [6, 26, 57]. To improve both the qualitative performance and computational efficiency

of patch diffusion, we develop two principled techniques: deep context fusion and adaptive computation.

Deep context fusion considers conditions the generation of higher-resolution patches on subsampled, positionally aligned features from the lower levels of the pyramid. It serves as an elegant way to incorporate global context information into synthesis of higher-frequency textural details and to facilitate knowledge sharing between the stages. Adaptive computation restructures the model architecture in such a way that only a subset of layers operate on high-resolution patches, while more difficult low-resolution ones go through the whole pipeline.

We apply the designed techniques to the recent attention-based RIN generator [27], and benchmark our approach on two video generation datasets: UCF-101 [56] in the 64×256^2 resolution, and our internal dataset of text/video pairs for $64 \times 288 \times 512$ (and $16 \times 576 \times 1024$) text-to-video generation. Our model achieves state-of-the-art performance on UCF-101 and demonstrates strong scalability performance for large-scale text-to-video synthesis.

2. Related work

High-level diffusion paradigms. To the best of our knowledge, one can identify two main conceptual paradigms on how to structure a high-resolution diffusion-based generator: latent diffusion models (LDM) [44] and cascaded diffusion models (CDM) [23]. For CDMs, it was shown that the cascade can be trained jointly [18], but scaling for high resolutions or videos still requires progressive training from low-resolution models to obtain competitive results [19].

Video diffusion models. The rise of diffusion models as foundational image generators [9, 43] motivated the community to explore them for video synthesis as well [24]. VDM [24] is one of the first works to demonstrate their scalability for conditional and unconditional video generation using the cascaded diffusion approach [23]. ImagenVideo [22] further pushes their results, achieving photorealistic quality. VIDM [38] designs a separate module to implicitly model motion. PVDM [71] trains a diffusion model in a spatially decomposed latent space. Make-A-Video [51] uses a vast unsupervised video collection in training a text-to-video generator by fine-tuning a text-to-image generator. PYoCo [14] and VideoFusion [37] design specialized noise structures for video generation. Numerous works explore training of a foundational video generator on limited resources by fine-tuning a publicly available StableDiffusion [41, 44] model for video synthesis (e.g., [1, 4, 20, 37, 63]). Another important line of research is the adaptation of the foundational image or video generators for downstream tasks, such as video editing (e.g. [11, 15, 30, 65, 66]) or 4D generation [50]. None of these models is end-to-end and all follow cascaded [23] or latent [44] diffusion paradigms.

Patch Diffusion Models. Patch-wise generation has a long history in GANs [16] and has enjoyed applications in image [36], video [53] and 3D synthesis [48]. In the context of diffusion models, there are several works that explore patch-wise inference to extend foundational text-to-image generators to higher resolutions than what they had been trained on (e.g., [2, 31, 74]). Also, a regular video diffusion model can be inferred in an autoregressive manner at the test time because it can be easily conditioned on its previous generations via classifier guidance or noise initialization [24], and this kind of synthesis can also be seen as a patch-wise generation. Later stages of CDMs can also operate in a patch-wise fashion [43], even though they have not been explicitly trained for this. These works have relevance to ours, since they design patch-wise sampling strategies with better global consistency in the resulting samples and thus could be employed for our generator as well.

The primary focus of our work is patch-wise training of diffusion models, which has been explored in several prior works. Several works (e.g., [34, 40, 62]) train a diffusion model on a single image to produce its variations [17, 49]. The closest work to ours is PatchDiffusion [64], which explores direct patch-wise diffusion training. However, to learn the consistent global image structure, their developed model operates on full-size inputs in 50% of the optimization steps, which is computationally infeasible for high-resolution videos. Our generator design, in contrast, *never* operates on full-resolution videos and instead relies on context fusion to enforce the consistency between the patches.

Apart from expensive training, diffusion models also suffer from slow inference [9], and some works explored alternative denoising paradigms (e.g., [68, 69]) to mitigate this, which is a close but orthogonal line of research.

3. Background

3.1. Diffusion Models

Given a dataset $\boldsymbol{X} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$, consisting of N samples $\boldsymbol{x}^{(n)} \in \mathbb{R}^d$ (most commonly images or videos), we seek to recover the underlying data-generating distribution $\boldsymbol{x}^{(n)} \sim p(\boldsymbol{x})$. We follow the general design of time-continuous diffusion models [29], for which a neural network $D_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}; \sigma)$ is trained to predict ground-truth dataset samples $\boldsymbol{x}$ from their noised versions $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma \boldsymbol{I})$:

$$\mathbb{E}_{\boldsymbol{x}, \boldsymbol{\varepsilon}, \sigma} \left[\|D_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}; \sigma) - \boldsymbol{x}\|_2^2 \right] \to \min_{\boldsymbol{\theta}} \quad (1)$$

In the above formula, $p(\sigma)$ controls the corruption intensity and its distribution parameters are treated as hyperparameters [6, 29]. The denoising network can serve as a score estimator [29]:

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, \sigma) \triangleq \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) \approx \frac{1}{\sigma^2}(D_{\boldsymbol{\theta}}(\boldsymbol{x}; \sigma) - \boldsymbol{x}). \quad (2)$$
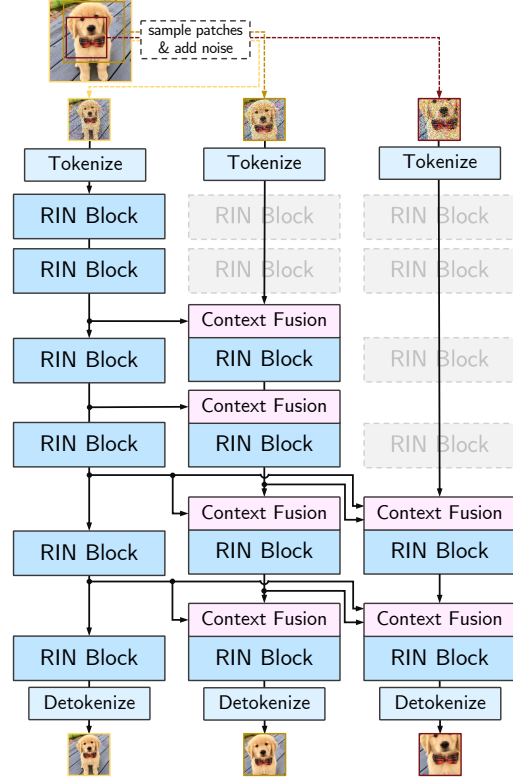

Figure 2. Architecture overview of Hierarchical Patch Diffusion Model (HPDM) for a 3-level pyramid. The model is trained to denoise all the patches jointly. During training, we use only a single patch from each pyramid level and restrict information propagation in the coarse-to-fine manner. This allows one to synthesize the whole image (or video) at a given resolution patch-by-patch using tiled inference (see Figure 1).

For large enough σ, the corrupted sample $\tilde{\boldsymbol{x}}$ is indistinguishable from pure Gaussian noise, and this allows to employ the score predictor for sampling at test-time using Langevin dynamics [55] (with $\sigma \to 0$ and $T \to \infty$):

$$\tilde{\boldsymbol{x}}_t = \tilde{\boldsymbol{x}}_{t-1} + \frac{\sigma}{2} \boldsymbol{s}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}, \sigma) + \boldsymbol{\varepsilon}_t. \quad (3)$$

3.2. Recurrent Interface Networks

For our base architecture, we chose to follow Recurrent Interface Networks (RINs) [27] for their simplicity and expressivity. A typical RIN network has a uniform structure and consists of a ViT-like [10] linear image tokenizer, followed by a sequence of identical attention-only blocks and a linear detokenizer to transform the image tokens back to the RGB pixel values. RIN blocks do not employ an expensive self-attention mechanism [61] and instead rely on linear cross-attention layers with a set of learnable latent tokens. This allows to scale gracefully with input resolution without sacrificing communication between far-away input locations. We refer the reader to the original work [27] for

additional details and provide the illustration for our RIN block in Figure 11 in Appendix C.

4. Method

Our high-level patch diffusion design is different from PatchDiffusion [64] in that our model never operates on full-resolution inputs. Instead, we consider a hierarchical cascade-like structure consisting of L stages and patch scales s_ℓ decrease exponentially: $s_\ell = 1/2^\ell$ for $\ell \in \{0, 1, ..., L\}$. Patches are always of the same resolution $r = (r_f, r_h, r_w)$, which leads to substantial memory and computational savings compared to full-resolution training. During training, we randomly sample a video from the dataset and extract a hierarchy of patch coordinates $c_0, ..., c_L$ in such a way that the ℓ-th patch is always located inside the previous $\ell' < \ell$ patches so that they provide the necessary context information. Hierarchical patch diffusion is trained to jointly denoise a combination of these patches, denoted as $\boldsymbol{P} = (p_\ell)\ell = 0^L$, and their corresponding noise levels $\boldsymbol{\sigma} = (\sigma_\ell)_{\ell=0}^L$:

$$\mathbb{E}_{\boldsymbol{p}, \boldsymbol{\varepsilon}, \sigma} \left[\|D_{\boldsymbol{\theta}}(\tilde{\boldsymbol{P}}_0; \boldsymbol{\sigma}) - \boldsymbol{P}\|_2^2 \right] \to \min_{\boldsymbol{\theta}}, \qquad (4)$$

where each patch is corrupted independently: $\tilde{\boldsymbol{P}} = (\tilde{\boldsymbol{p}}_\ell + \boldsymbol{\varepsilon}_\ell)_{\ell=0}^L, \boldsymbol{\varepsilon}_\ell \sim \mathcal{N}(0, \sigma_\ell, I)$ Restricting the information flow in the coarse-to-fine manner (see Fig. 2) allows to do inference at test-time in the cascaded diffusion fashion [23].

Below, we elaborate on three fundamental components of our method that allow a patch-wise paradigm to achieve state-of-the-art results in video generation: deep context fusion, adaptive computation and overlapped sampling.

4.1. Patch Diffusion

The training objective of patch diffusion is similar to the regular diffusion design, but instead of full-size videos (or images) $\boldsymbol{x} \in \mathbb{R}^{R_f \times R_h \times R_w}$, it uses randomly subsampled patches $\boldsymbol{p} \in \mathbb{R}^{r_f \times r_h \times r_w}$ and trains the patch-wise model $D_{\boldsymbol{\theta}}(\tilde{\boldsymbol{p}}; \sigma)$ to denoise them:

$$\mathbb{E}_{\boldsymbol{p}, \boldsymbol{\varepsilon}, \sigma} \left[\|D_{\boldsymbol{\theta}}(\tilde{\boldsymbol{p}}; \sigma) - \boldsymbol{p}\|_2^2 \right] \to \min_{\boldsymbol{\theta}}. \qquad (5)$$

Following [54], the patch extraction procedure extracts pixels using random scales $\boldsymbol{s} = (s_f, s_h, s_w), s_* \in [r_*/R_*, 1]$, and offsets $\boldsymbol{\delta} = (\delta_f, \delta_h, \delta_w), \delta_* \in [0, 1 - s]$:

$$\boldsymbol{p} = \texttt{downsample}(\texttt{crop}(\boldsymbol{x}; \boldsymbol{\delta}); \boldsymbol{r}), \qquad (6)$$

where the `crop` function slices the input signal given the pixel offsets $\boldsymbol{\delta}$, and `downsample` resizes it to the specified resolution $r_f \times r_h \times r_w$.

Since we consider a hierarchical structure, during training, we use fixed scales for each ℓ-th level $s_*^{(\ell)} = r_*^\ell / R_*$, but randomly sample offsets $\delta_*^{(\ell)} \sim U[0, 1 - s_*^{(\ell)}]$. For a level
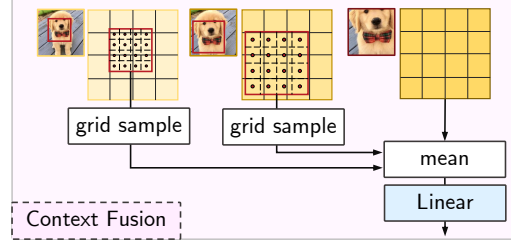

Figure 3. **Deep Context Fusion**. At each pyramid level, we grid-sample the features of a lower-resolution patch and concatenate them to the activations tensor of the current level. In this way, the information propagates in the coarse-to-fine manner and provides richer context than pixel-space concatenation of cascaded DMs (see Tab. 3).

$\ell > 1$, we sample its corresponding offset $\delta_*^{(\ell)}$ in each $*$-th dimension in such a way, that the resulting patch is always located inside the patch from the previous pyramid level, as visualized in Fig. 2. For brevity, we will omit the level superscript in the subsequent exposition for patch parameters.

Setting patch resolutions r_f, r_h, r_w lower than original ones R_f, R_h, R_w leads to drastic improvements in computational efficiency, but worsens the global consistency of the generated samples. In [64], the authors use variable-resolution training, including 50% of optimization steps performed on full-size inputs to improve the consistency. The downside of such a strategy is that it undermines computational efficiency: for a large enough video, the model cannot fit into GPU memory even for a batch size of 1. Instead, in our work, we demonstrate that consistent generation can be achieved with *deep context fusion*: conditioning higher resolution generation on the activations from previously generated stages.

4.2. Deep Context Fusion

The main struggle of patch-wise models is preserving the consistency between the patches, since each patch is modeled independently from the rest, conditioned on the previous pyramid stage. Cascaded DMs [23] provide the conditioning to later stages by simply concatenating an upsampled low-resolution video channel-wise [23] to the current latent. While it can provide the global context information when the model operates on a full-resolution input, for patch-wise models, this leads to drastic context cutouts, which, as we demonstrate in our experiments, severely worsens the performance. Also, it limits the knowledge sharing between lower and higher stages of the cascade. To address this issue, we introduce *deep context fusion (DCF)*, a context fusion strategy that conditions the higher stages of the pyramid on spatially aligned, globally pooled features from the previous stages.

For this, before each RIN block of our model, we pool

the global context information from previous stages into its inputs. For this, we use the patch coordinates to grid-sample the activations with trilinear interpolation from all previous pyramid stages, average them, and concatenate to the current-stage features.

More precisely, for a given patch b-th block inputs $a_\ell^{b-1} \in \mathbb{R}^{d \times r'_f \times r'_h \times r'_w}$ with coordinates $c_\ell = (s, \delta_f, \delta_h, \delta_w) \in \mathbb{R}^4$ at the ℓ-th pyramid level; $\ell - 1$ context patches' activations $(a_k^{b-1})_{k=1}^{\ell-1}$ with respective coordinates $(c_k)_{k=1}^{\ell-1}$, we compute the context $\text{ctx}_\ell \in \mathbb{R}^{d \times r'_f \times r'_h \times r'_w}$ as:

$$\text{ctx}_\ell^b = \frac{1}{\ell-1} \sum_{k=1}^{\ell-1} \texttt{grid\_sample}_{\text{3D}}[a_{\ell-1}^{b-1}, \hat{c}_\ell], \quad (7)$$

where $\texttt{grid\_sample}_{\text{3D}}$ is a function that extracts the features with trilinear interpolation via the coordinates queries, $\hat{c}_\ell$ are the recomputed patch coordinates (for $k < \ell$) calculated as:

$$\hat{c}_\ell(c_\ell, c_k) = [s_\ell/s_k; (\delta_\ell - \delta_k)/s_k]. \quad (8)$$

We fuse this context information via simple channel-wise concatenation together with the coordinates information c_ℓ which we found to be slightly improving the consistency:

$$\texttt{fuse}[a_\ell^{b-1}, c_\ell; (a_k^{b-1}, c_k)_{k=1}^{\ell-1}] = \texttt{concat}[a_\ell, \text{ctx}_\ell, c_\ell]. \quad (9)$$

Deep context fusion is illustrated in Fig. 3.

To keep the dimensionalities the same across the network, we then project the resulted tensor $\texttt{fuse}[\cdot] \in \mathbb{R}^{(2d+3) \times r'_f \times r'_h \times r'_w}$ with a learnable linear transformation. We considered other aggregation strategies, like concatenating all the levels's features or averaging, but the former one blows up the dimensionalities, making the training expensive, while the latter one was leading to poor performance in our preliminary experiments.

An additional advantage of DCF compared to shallow context fusion of regular cascaded DMs is that the gradient can flow from the small-scale patch denoising loss to the lower levels of the hierarchy, pushing the earlier cascade stages to learn such features that are more useful to the later ones. We found that this is indeed helpful in practice and improves the overall performance additionally by $\approx 5\%$.

4.3. Adaptive Computation

Naturally, generating high-resolution details is considered to be easier than synthesizing the low-resolution structure [12]. In this way, allocating the same amount of network capacity on high-resolution patches can be excessive, that is why we propose to use only some of the computational blocks when running the last stages of the pyramid.

We name this strategy *adaptive computation*[1] and demonstrate that it improves our model's efficiency by $\approx 60\%$ without compromising the performance (see Tab. 3). The uniform RIN's structure [27] (i.e., all the blocks are identical and have the same input/output resolutions) allows us to implement this easily: one simply skips some of the earlier blocks when processing the high-resolution activations. The high-level pseudo-code is provided in Listing 1.

```
def adaptive_computation(
    blocks: List[RINBlock],
    x: Tensor,
    num_levels_per_block: List[int]
) -> Tensor:
    # `x` has the shape: [B, L, D, F, H, W]
    for blk_idx, blk in enumerate(blocks):
        nlvl: int = num_levels_per_block[blk_idx]
        x[:, :nlvl] = blk(x[:, :nlvl])
```

Listing 1. Pseudo-code for adaptive computation (Sec. 4.3)

Adaptive computation involves two design choices: 1) whether to skip earlier or later blocks in the networks for higher resolutions, and 2) how to distribute the computation assignments among the blocks per each pyramid stage. We chose to allocate the later blocks to perform full computation to make the low-level context information go through more processing before being propagated to the higher stages. For the block allocations, we observed that simply increasing the computation assignments linearly with the block index worked well in practice.

4.4. Tiled Inference

Sampling from HPDM is different from regular diffusion sampling, since it is patch-wise and we never operate on full-resolution inputs. During inference, we generate pyramid levels one-by-one, starting from $r_t \times r_h \times r_w$ video (corresponding to a patch of scale $s = 1$), then using to generate the video of resolution $2r_t \times 2r_h \times 2r_w$ (corresponding to patch scale $s = 1/2$), and so on until we produce the final video of full resolution $R_f \times R_h \times R_w$. We visualize this hierarchical tiled inference process in Fig. 1 (bottom right).

Each next stage of the pyramid uses the generated video from the previous stage through the deep context fusion technique described in Sec. 4.2. DCF provides strong global context conditioning, but it is sometimes not enough to enforce local consistency between two neighboring patches. To mitigate this, we employ the MultiDiffusion [2] strategy and simply average-overlap the score predictions $s_\theta(\hat{p}, \sigma)$ during the denoising process. More concretely, to generate a complete video $x \in \mathbb{R}^{R_f \times R_h \times R_w}$, we first generate $(2R_f-1) \times (2R_h-1) \times (2R_w-1)$ patches with 50% of the coordinates overlapping between two neighboring patches. Then, we run the reverse diffusion process for each patch

---

[1]Our notion of adaptive computation is different from the original RIN's one, where it is used to describe the model's ability to distribute its computational capacity differently between different parts of an input [27].

Figure 4. Provided samples from PVDM [71] (left) and random samples from HPDM-L (right) for the same classes on UCF 256^2. More samples are provided in Appendix B.

and average the overlapping regions of the corresponding score predictions. The importance of overlapped inference is illustrated in Fig. 6 and Tab. 4.

4.5. Miscellaneous techniques

The core ideas that enable our work have been described above, but from the implementation and engineering standpoints, there are several other techniques that played an important role in bolstering the performance and would be of interest to a practitioner aiming to reproduce our results. Additional details and failed experiments can be found in Appendix A and D, respectively.

Integer patch coordinates. We noticed that sampling a patch on the L-th cascade level at integer coordinates allows to prevent blurry artifacts in generated videos: they appear due to oversmoothness effects of trilinear interpolation.

Noise Schedule Each stage of the pyramid operates on different frequency signals, and higher levels of the pyramid have stronger correlations between patch pixels. Inspired by [6], we found it helpful to use exponentially smaller input noise scaling with each increase in pyramid level.

Cached inference During inference, we do not need to recompute all the activations for the previous pyramid stages, which makes it possible to cache them, which works even more gracefully. Caching block features allowed to speed up the inference by $\approx 40\%$. However, for the large model, caching needs to be implemented with CPU offloading to prevent GPU out-of-memory errors.

4.6. Implementation details

We use RINs [27] instead of U-Nets [9, 45] as the backbone since its uniform structure is conceptually simpler and aligns well with adaptive computation. We use v-prediction parametrization [47] with extra input scaling [6]. Following RINs, we train our model with the LAMB optimizer [67], with the cosine learning rate schedule and the maximum LR of 0.005. Our model has 6 RIN blocks, and we distribute the load for adaptive computation as [1, 1, 2, 2, 3, 4]: e.g., the 1-st and 2-nd blocks only compute the first pyramid level, the 3-rd and 4-rd ones — first two levels of the pyramid, and so on. Not using adaptive computation is equivalent to having

a load of [4, 4, 4, 4, 4, 4], which is almost twice as expensive. We use 768 latent tokens of 1024/3072 dimensionality with $1 \times 4 \times 4$ pixel tokenization for class-conditional/text-conditional experiments, respectively. To encode the textual information, we rely on T5 language model [42] and use its T5-11B variant. Further implementation details can be found in Appx C.

5. Experiments

Datasets. In our work, we consider two datasets: 1) UCF101 [56] (for exploration and ablations) and 2) our internal video dataset to train a large-scale text-to-video model. UCF101 is a popular academic benchmark for unconditional and class-conditional video generation consisting of videos of the 240×320 resolution with 25 FPS and has an average video length of ≈ 7 seconds. Our internal dataset consists of $\approx 25M$ high-quality text/video pairs in the style of stock footage with manual human annotations and $\approx 70M$ of low-quality in-the-wild videos with automatically generated captions. Additionally, for text-to-video experiments, we used an internal dataset of $\approx 150M$ high-quality text/image pairs for extra supervision [51].

Evaluation. Following prior work [22, 24, 25, 53], we evaluate the model with two main video quality metrics: Frechet Video Distance (FVD) [59], and Inception Score (IS) [46]. For FVD and IS, we report their values based on 10,000 generated videos. But for ablations, we use FVD@512 instead for efficiency purposes: an FVD variant computed on just 512 generated videos. We noticed that it correlates well with the traditional FVD, but with just a fixed offset. Apart from that, we report the training throughput for various designs of our network and also provide the samples from our model for qualitative assessment.

5.1. Video generation on UCF-101

We train HPDM in three variants: HPDM-S HPDM-M, and HPDM-L, which differ in the amount of parameters, batch size and training iterations used. The hyperparameters for them are provided in Tab. 8. For ablations, we train all the models for 50K steps with the batch size of 512. UCF mod-

"A panda bear driving a car."   "A dog wearing a Superhero outfit with red cape flying through the sky."   "A robot dj is playing the turntable, in heavy raining futuristic tokyo rooftop cyberpunk night, sci-fi, fantasy, intricate, elegant, neon lightlhighly detailed, concept art, soft light, smooth, sharp focus, illustration."

ImagenVideo   Make-A-Video   PYoCo

Ours   Ours   Ours

Figure 5. HPDM-T2Vis able to efficiently fine-tune from the standard low-resolution generator to high-resolution $64 \times 288 \times 512$ text-to-video generation when fine-tuned from a low-resolution 36×64 diffusion for just 15,000 training steps.

els are trained for the final video resolution of 64×256^2 with the pyramid $16 \times 64^2 \rightarrow 32 \times 128^2 \rightarrow 64 \times 256^2$.

Main results. Our patch-wise model is trained on UCF-101 [56] for 64×256^2 generation entirely end-to-end with the hierarchical patch sampling procedure described in Sec. 4. In Tab. 5, we compare these results with recent state-of-the-art methods: MoCoGAN-HD [58], StyleGAN-V [53], TATS [13], VIDM [38], DIGAN [70], PVDM [71]. While our model is trained to synthesize 64 frames, we report quantitative results for 16 generated frames, since it is a much more popular benchmark in the literature (for this, we simply subsample 16 frames out of the generated 64). Our model substantially outperforms all previously reported results for this benchmark (i.e., for the 16×256^2 resolution and without pretraining) by a striking margin of more than 100%. To our knowledge, these are the best reported FVD and IS scores for the 16×256^2 resolution on UCF. Make-A-Video [51] reports FVD of 81.25 and IS of 82.55 when fine-tuned from a large-scale text-to-video generator.

Ablations. We consider two lines of ablations: ablating core architectural decisions and benchmakring various inference strategies, since the latter also crucially influences the final performance. For the training components, we first analyze the influence of deep context fusion. For this, we launch an experiment with "shallow context fusion", where we concatenate only the RGB pixels (non-averaged, only from the patch of the previous pyramid level) as the context information. As one can see from the results in Tab. 3 (first row), this strategy produces considerably worse results (though the training becomes $\approx 10\%$ faster).

The next ablation is whether the low-level pyramid stages indeed learn such features that are more useful for later pyramid stages, when they are directly supervised with the denoising loss of small-scale patches through the context aggregation procedure. For this ablation, we detach the context variable ctx from the autograd graph. The results are presented in Tab. 3 (second row). One can observe that the performance can be better for earlier pyramid stages, but the late stage suffers: this demonstrates that the lowest stage indeed learns to encode the global context in a way that is more accessible for later levels of the cascade, but by

Table 2. Comparison with the recent state-of-the-art methods on UCF-101 [56] 16×256^2 class-conditional video generation (note that our model is trained 64×256^2 videos). *Note that Make-A-Video [51] was pretrained on a large-scale text-to-video dataset.

Method	FVD ↓	IS ↑	Venue
DIGAN [70]	1630.2	29.71	ICLR'22
MoCoGAN-HD [58]	700	33.95	ICLR'21
StyleGAN-V [53]	1431.0	23.94	CVPR'22
TATS [51]	635	57.63	ECCV'22
VIDM [38]	294.7	-	AAAI'23
PVDM [71]	343.6	74.4	CVPR'23
Make-A-Video* [51]	81.25	82.55	ICLR'23
HPDM-S	344.5	73.73	
HPDM-M	143.1	84.29	CVPR'24
HPDM-L	66.32	87.68	

sacrificing a part of its capacity due to this.

One of the key techniques we used in our model is adaptive computation, and in Tab. 3 (third row), we demonstrate how the model performs without it. While it allows to obtain slightly better results, it decreases the training speed by almost twice. The cost of the later pyramid stages becomes even more critical during inference time, when sampling high-resolution videos.

Finally, we verify the existing observation of the community that positional encoding in patch-wise models help in producing more spatially consistent samples [36, 52]. This can be seen from the worse FVD@512 scores in Tab. 3 (4th row) when no coordinates information is input to the model in context fusion (Eq. (9)).

5.2. Text-to-video generation

Training setup. To explore the scalability of the patch-wise paradigm, we launched a large-scale experiment for HPDM with ≈ 4B parameters on a text/video dataset consisting of ≈ 95M samples. Since training a foundational model incurs extreme financial costs, we instead found it financially less risky to fine-tune it from a low-resolution generator. For this, we used the base SnapVideo [39] model, which oper-

Table 3. Ablating architectural components in terms of FVD scores and training speed measured as the videos/sec throughout on a single NVidia A100 80GB GPU.

Setup	FVD@512 16×64^2	FVD@512 32×128^2	FVD@512 64×256^2	Training speed ↑
Shallow fusion	298.9	411.9	467.0	4.91
Context detach	290.6	375.0	397.3	4.4
No adapt. computation	319.3	391.5	373.9	2.73
No coordinates	305.3	400.7	389.5	4.47
Default model	287.6	376.6	378.2	4.4

Table 4. FVD@512 for various overlapped inference strategies.

Inference strategy	32×128^2	64×256^2
No overlapping	385.40	475.05
50% w-overlapping	367.10	452.79
50% h-overlapping	383.15	467.36
50% h/w-overlapping	382.25	456.10
50% f-overlapping	380.63	460.74
50% f/w-overlapping	398.77	492.84
50% f/h-overlapping	360.46	436.81
50% $f/h/w$-overlapping	381.85	467.37

no overlapping   spatial overlapping   full overlapping

Figure 6. Effect of the overlapped inference [2] on the consistency between the patches. Surprisingly, even without the full-resolution training [64] and patch overlapping, our deep context fusion strategy manages to preserve strong consistency in the generated sample. See Tab. 4 for quantitative analysis.

ates on 36×64 resolution videos. Our patch-wise variant, HPDM-T2V, was trained for the final output resolution of $64 \times 288 \times 512$ with the pyramid $8 \times 36 \times 64 \rightarrow 16 \times 72 \times 128 \rightarrow 32 \times 144 \times 256 \rightarrow 64 \times 288 \times 512$ (4 pyramid levels in total). This 4-level pyramid structure results in just $4 \cdot (1/8)^3 \approx 0.7\%$ of the original video pixels seen in each optimization step. The base 36×64 generator was trained for 500,000 iterations, and we fine-tuned HPDM-T2V for 15,000 more steps (3% of the base generator training steps) with a batch size of 4096. We also fine-tune another model, HPDM-T2V-1K, a $16 \times 576 \times 1024$ text-to-video generator with a patch resolution of $16 \times 72 \times 128$. It is initialized from the base 36×64 SnapVideo diffusion model, but fine-tuned for 100,000 iterations. Longer fine-tuning was required for it since its input resolution was chosen to be larger than that of the base generator to make it

Table 5. Zero-shot performance on UCF-101. HPDM-T2V achieves competitive performance when fine-tuned from the base low-resolution 36×64 generator for just 15,000 training steps.

Method	Resolution	FVD↓	IS↑
CogVideo [25]	128×128	701.6	25.27
Make-A-Video	256×256	367.2	33.00
MagicVideo [75]	256×256	655	-
LVDM [21]	256×256	641.8	-
Video LDM [4]	N/A	550.6	33.45
VideoFactory [63]	256×256	410.0	-
PYoCo [14]	256×256	355.2	47.46
HPDM-T2V	72×128	299.3	20.53
HPDM-T2V	144×256	383.3	21.15
HPDM-T2V	288×512	481.9	23.77
HPDM-T2V-1K	576×1024	447.5	24.51

have 4 levels in the pyramid instead of 5. Apart from videos, following prior works (e.g., [24, 51]), we utilize joint image/video training. For image training with RINs, following SnapVideo [39], we simply repeat the image along the time axis to convert it into a still video.

Results. We test the results quantitatively by reporting zero-shot performance on UCF-101 [56] in terms of FVD and IS in Tab. 5, and also qualitatively by providing visual comparisons with existing foundational generators in Fig. 5. Although trained for just 15,000 steps, HPDM-T2V yields promising results and has a comparable generation quality to modern foundational text-to-video models (ImageVideo [22], Make-A-Video [51], and PYoCo [14]) on some text prompts (see Fig. 5).

We provide more qualitative results on the project webpage: https://snap-research.github.io/hpdm.

6. Conclusion

In this work, we developed the hierarchical patch diffusion model for high-resolution video synthesis, which efficiently trains in the end-to-end manner directly in the pixel space, and is amenable to swift fine-tuning from a base low-resolution diffusion model. We showed state-of-the-art video generation performance on UCF-101, outperforming the recent methods by $\approx 100\%$ in terms of FVD, and promising scalability results for text-to-video generation. The techniques we developed hold significant potential for application across various patch-wise generative paradigms, including GANs, VAEs, autoregressive models, and beyond. In future work, we intend to investigate better context conditioning, sampling strategies with stronger dependence enforcement, and also other tokenization/detokenization transformations to mitigate dead pixels artifacts.

References

[1] Zeroscope. https : / / huggingface . co / cerspense/zeroscope_v2_576w, 2022. Accessed: 2023-11-01. 2

[2] Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. Multidiffusion: Fusing diffusion paths for controlled image generation. *arXiv preprint arXiv:2302.08113*, 2023. 3, 5, 8

[3] James Betker, Gabriel Goh, Li Jing, Tim Brooks†, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo†, Wesam Manassra, Prafulla Dhariwal, Casey Chu, Yunxin Jiao†, and Aditya Ramesh. Improving image generation with better captions. https://cdn.openai.com/papers/dall-e-3.pdf, 2023. Accessed: 2023-11-14. 1

[4] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023. 2, 8

[5] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12299–12310, 2021. 1

[6] Ting Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023. 2, 3, 6

[7] Katherine Crowson, Stefan Andreas Baumann, Alex Birch, Tanishq Mathew Abraham, Daniel Z Kaplan, and Enrico Shippole. Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers. *arXiv preprint arXiv:2401.11605*, 2024. 2

[8] Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, et al. Emu: Enhancing image generation models using photogenic needles in a haystack. *arXiv preprint arXiv:2309.15807*, 2023. 1

[9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 2, 3, 6, 1

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3, 1

[11] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7346–7356, 2023. 2

[12] Rinon Gal, Dana Cohen Hochberg, Amit Bermano, and Daniel Cohen-Or. Swagan: A style-based wavelet-driven generative model. *ACM Transactions on Graphics (TOG)*, 40(4):1–11, 2021. 5

[13] Songwei Ge, Thomas Hayes, Harry Yang, Xi Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. Long video generation with time-agnostic vqgan and time-sensitive transformer. In *European Conference on Computer Vision*, pages 102–118. Springer, 2022. 7, 2

[14] Songwei Ge, Seungjun Nah, Guilin Liu, Tyler Poon, Andrew Tao, Bryan Catanzaro, David Jacobs, Jia-Bin Huang, Ming-Yu Liu, and Yogesh Balaji. Preserve your own correlation: A noise prior for video diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22930–22941, 2023. 2, 8

[15] Michal Geyer, Omer Bar-Tal, Shai Bagon, and Tali Dekel. Tokenflow: Consistent diffusion features for consistent video editing. *arXiv preprint arxiv:2307.10373*, 2023. 2

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Adv. Neural Inform. Process. Syst.*, 2014. 1, 3

[17] Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, and Michal Irani. Drop the gan: In defense of patches nearest neighbors as single image generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13460–13469, 2022. 3

[18] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Miguel Angel Bautista, and Josh Susskind. f-dm: A multi-stage diffusion model via progressive signal transformation. *arXiv preprint arXiv:2210.04955*, 2022. 2

[19] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Josh Susskind, and Navdeep Jaitly. Matryoshka diffusion models. *arXiv preprint arXiv:2310.15111*, 2023. 2, 1

[20] Yuwei Guo, Ceyuan Yang, Anyi Rao, Yaohui Wang, Yu Qiao, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023. 2

[21] Yingqing He, Tianyu Yang, Yong Zhang, Ying Shan, and Qifeng Chen. Latent video diffusion models for high-fidelity video generation with arbitrary lengths. *arXiv preprint arXiv:2211.13221*, 2022. 8, 2

[22] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 2, 6, 8

[23] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *The Journal of Machine Learning Research*, 23(1):2249–2281, 2022. 1, 2, 4

[24] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022. 2, 3, 6, 8

[25] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022. 6, 8, 2

[26] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. *arXiv preprint arXiv:2301.11093*, 2023. 2

[27] Allan Jabri, David Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. *arXiv preprint arXiv:2212.11972*, 2022. 2, 3, 5, 6, 1

[28] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10124–10134, 2023. 1

[29] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. 3, 1, 2

[30] Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. *arXiv preprint arXiv:2303.13439*, 2023. 2

[31] Subin Kim, Kyungmin Lee, June Suk Choi, Jongheon Jeong, Kihyuk Sohn, and Jinwoo Shin. Collaborative score distillation for consistent visual synthesis. *arXiv preprint arXiv:2307.04787*, 2023. 3

[32] Diederik P Kingma and Ruiqi Gao. Understanding the diffusion objective as a weighted integral of elbos. *arXiv preprint arXiv:2303.00848*, 2023. 2

[33] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[34] Vladimir Kulikov, Shahar Yadin, Matan Kleiner, and Tomer Michaeli. Sinddm: A single image denoising diffusion model. In *International Conference on Machine Learning*, pages 17920–17930. PMLR, 2023. 3

[35] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *arXiv preprint arXiv:2306.00980*, 2023. 1

[36] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, and Hwann-Tzong Chen. Coco-gan: Generation by parts via conditional coordinating. In *ICCV*, 2019. 3, 7

[37] Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liang Wang, Yujun Shen, Deli Zhao, Jingren Zhou, and Tieniu Tan. Videofusion: Decomposed diffusion models for high-quality video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[38] Kangfu Mei and Vishal Patel. Vidm: Video implicit diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9117–9125, 2023. 2, 7

[39] Willi Menapace, Aliaksandr Siarohin, Ivan Skorokhodov, Ekaterina Deyneka, Tsai-Shien Chen, Anil Kag, Yuwei Fang, Aleksei Stoliar, Elisa Ricci, Jian Ren, et al. Snap video: Scaled spatiotemporal transformers for text-to-video synthesis. *arXiv preprint arXiv:2402.14797*, 2024. 7, 8

[40] Yaniv Nikankin, Niv Haim, and Michal Irani. Sinfusion: Training diffusion models on a single image or video. *arXiv preprint arXiv:2211.11743*, 2022. 3

[41] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 1, 2

[42] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020. 6

[43] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 2, 3, 1

[44] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 1, 2

[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 2, 6

[46] Masaki Saito, Shunta Saito, Masanori Koyama, and Sosuke Kobayashi. Train sparsely, generate densely: Memory-efficient unsupervised training of high-resolution temporal gan. *International Journal of Computer Vision*, 2020. 6, 2

[47] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. 6

[48] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Adv. Neural Inform. Process. Syst.*, 2020. 2, 3

[49] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, 2019. 3

[50] Liao Shen, Xingyi Li, Huiqiang Sun, Juewen Peng, Ke Xian, Zhiguo Cao, and Guosheng Lin. Make-it-4d: Synthesizing a consistent long-term dynamic scene video from a single image. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 8167–8175, 2023. 2

[51] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022. 2, 6, 7, 8

[52] Ivan Skorokhodov, Grigorii Sotnikov, and Mohamed Elhoseiny. Aligning latent and image spaces to connect the unconnectable. *arXiv preprint arXiv:2104.06954*, 2021. 2, 7

[53] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2. In *CVPR*, 2022. 2, 3, 6, 7, 1

[54] Ivan Skorokhodov, Sergey Tulyakov, Yiqun Wang, and Peter Wonka. Epigraf: Rethinking training of 3d gans. In *Adv. Neural Inform. Process. Syst.*, 2022. 2, 4

[55] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 3

[56] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 2, 6, 7, 8, 1, 3

[57] Jiayan Teng, Wendi Zheng, Ming Ding, Wenyi Hong, Jianqiao Wangni, Zhuoyi Yang, and Jie Tang. Relay diffusion: Unifying diffusion process across resolutions for image synthesis. *arXiv preprint arXiv:2309.03350*, 2023. 2

[58] Yu Tian, Jian Ren, Menglei Chai, Kyle Olszewski, Xi Peng, Dimitris N. Metaxas, and Sergey Tulyakov. A good image generator is what you need for high-resolution video synthesis. In *ICLR*, 2021. 7

[59] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018. 6, 1

[60] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. 2021. *arXiv preprint arXiv:2106.05931*, 2021. 1

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[62] Weilun Wang, Jianmin Bao, Wengang Zhou, Dongdong Chen, Dong Chen, Lu Yuan, and Houqiang Li. Sindiffusion: Learning a diffusion model from a single natural image. *arXiv preprint arXiv:2211.12445*, 2022. 3

[63] Wenjing Wang, Huan Yang, Zixi Tuo, Huiguo He, Junchen Zhu, Jianlong Fu, and Jiaying Liu. Videofactory: Swap attention in spatiotemporal diffusions for text-to-video generation. *arXiv preprint arXiv:2305.10874*, 2023. 2, 8

[64] Zhendong Wang, Yifan Jiang, Huangjie Zheng, Peihao Wang, Pengcheng He, Zhangyang Wang, Weizhu Chen, and Mingyuan Zhou. Patch diffusion: Faster and more data-efficient training of diffusion models. *arXiv preprint arXiv:2304.12526*, 2023. 2, 3, 4, 8

[65] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7623–7633, 2023. 2

[66] Shuai Yang, Yifan Zhou, Ziwei Liu, and Chen Change Loy. Rerender a video: Zero-shot text-guided video-to-video translation. *arXiv preprint arXiv:2306.07954*, 2023. 2

[67] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019. 6

[68] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, et al. Magvit: Masked generative video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10459–10469, 2023. 3

[69] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023. 3

[70] Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin. Generating videos with dynamics-aware implicit generative adversarial networks. In *ICLR*, 2022. 2, 7

[71] Sihyun Yu, Kihyuk Sohn, Subin Kim, and Jinwoo Shin. Video probabilistic diffusion models in projected latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 2, 6, 7, 1

[72] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023. 1

[73] Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. *Transactions on Machine Learning Research*, 2024. 2

[74] Qingping Zheng, Yuanfan Guo, Jiankang Deng, Jianhua Han, Ying Li, Songcen Xu, and Hang Xu. Any-size-diffusion: Toward efficient text-driven synthesis for any-size hd images. *arXiv preprint arXiv:2308.16582*, 2023. 3

[75] Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. Magicvideo: Efficient video generation with latent diffusion models. *arXiv preprint arXiv:2211.11018*, 2022. 8, 2

Hierarchical Patch Diffusion Models for High-Resolution Video Generation

Supplementary Material

A. Limitations

Although our model provides considerable improvements in video generation quality and enjoys a convenient end-to-end design, it still suffers from some limitations.

Stitching artifacts. Despite using overlapped inference, our model occasionally exhibits stitching artifacts. We illustrate these issues in Fig. 7 (left). Inference strategies with stronger spatial communication, like classifier guidance [9], should be employed to mitigate them.

Error propagation. Since our model generally follows the cascaded pipeline [19, 23, 28, 43] (with the difference that we train jointly and more efficiently), it suffers from the typical cascade drawback: the errors made in earlier stages of the pyramid are propagated to the next. The error propagation artifacts are illustrated in Fig. 7 (left).

Dead pixels. By "dead pixels" artifacts we imply failures of the ViT [10]-like pixel tokenization/detokenization procedure, where the model sometimes produces broken 4×4 patches. They are illustrated in Fig. 7. These artifacts are unique to RINs [27] and we have not experienced them in our preliminary experiments with UNets [9, 29]. However, since they do not appear catastrophically often, we chose to continue to experiment with RINs.

Slow inference. Patch-wise inference requires more function evaluations at test time, which slows down the inference process. For our exponentially growing pyramid starting at $8 \times 36 \times 64$ and ending at $64 \times 288 \times 512$, with full (i.e., maximal) overlapping, we need to produce $(2 \cdot \frac{64}{8} - 1) \times (2 \cdot \frac{288}{36} - 1) \times (2\frac{512}{64} - 1) = 3375$ patches for a single reverse diffusion step (see Sec. 4.4 for calculation details). Adaptive computation with caching greatly accelerates this process, but it is still heavy.
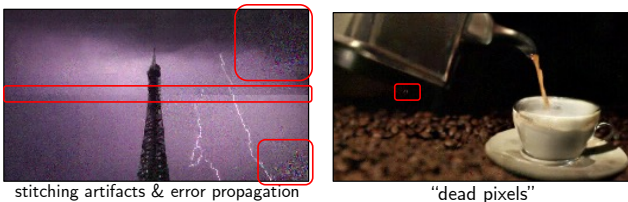

Figure 7. Illustrating the failure cases of HPDM..

B. Additional results

There are multiple incosistencies in quantitative evaluation of video generators that are inconsistent between previous projects [53, 71]. For FVD [59] on UCF101 (the most popular metric for it), there are differences in the amounts of
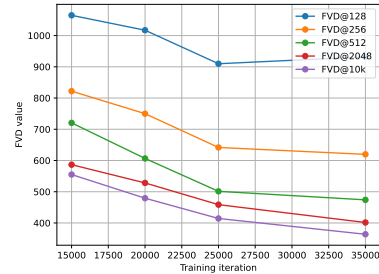

Figure 8. Using different amounts of fake videos to compute FVD [59] gives very correlated, but offset values with the main trend being "the more —- the better". We hypothesize that using more synthetic samples yields better coverage of different modes of the data distribution and decreases the influence of outliers. These FVD scores are computed for different training steps of HPDM-S. Using too few videos leads to undiscriminative results only closer to convergence.

fake/real videos used to compute the statistics, FPS values, resolutions, and real data subsets ("train" or "train + test"). To account for these differences, in Tab. 6, we release a comprehensive set of metrics for easier assessment of our models' performance in comparison with the prior work. Apart from that, it also includes additional models, HPDM-S and HPDM-M, and also the results for the fixed version of our text-to-video HPDM model (after the main deadline, we noticed that our FSDP-based [72] training was not updating some of the EMA parameters properly, which was the cause of gaussian jitter artifacts in Fig. 7).

To compute real data FVD statistics, we always use the train set of UCF-101 (around 9.5k videos in total). We train the models with the default 25FPS resolution. Our models are trained for 64 frames, and to compute the results for 16 frames, we simply take the first 16 frames out of the sequence.

Additional results are also provided on the project webpage: https://snap-research.github.io/hpdm.

C. Implementation details

In this section, we provide additional implementation details for our model. We train our model in a patch-wise fashion with the patch resolution of $16 \times 64 \times 64$ for UCF-101 [56] and $8 \times 36 \times 64$ for text-to-video generation. After the main deadline, we continued training our model on UCF for several more training steps, and also trained two smaller versions for fewer steps. We denote the smaller versions as HPDM-S and HPDM-M, while the larger one is denoted as HPDM-L. They differ in the amount of train-

Table 6. Additional FVD evaluation results for class-conditional UCF-101 video generation. "Pre-trained" denotes whether the model was pre-trained on an external dataset. "#samples" is the amount of fake videos used to compute the fake data statistics. In Fig. 8, we also demonstrated that FVD scores computed for different amount of samples are well-correlated with one another. For IS, we cannot compute it for 64-frames-long videos due to the design of C3D model [46, 53].

Method	Resolution	Pre-trained?	#samples	FVD↓	IS↑
DIGAN [70]	$16 \times 128 \times 128$	✗	2,048	1630.2	00.00
StyleGAN-V [53]	$16 \times 256 \times 256$	✗	2,048	1431.0	23.94
TATS [13]	$16 \times 128 \times 128$	✗	N/A	332	79.28
VIDM [38]	$16 \times 256 \times 256$	✗	2,048	294.7	-
LVDM [21]	$16 \times 256 \times 256$	✗	2,048	372	-
PVDM [71]	$16 \times 256 \times 256$	✗	2,048	343.6	-
PVDM [71]	$16 \times 256 \times 256$	✗	10,000	-	74.40
PVDM [71]	$128 \times 256 \times 256$	✗	2,048	648.4	-
VideoFusion [37]	$16 \times 128 \times 128$	✗	N/A	173	80.03
Make-A-Video* [51]	$16 \times 256 \times 256$	✓	10,000	81.25	82.55
HPDM-S	$16 \times 256 \times 256$	✗	2,048	370.50	61.50
	$16 \times 256 \times 256$	✗	10,000	344.54	73.73
	$64 \times 256 \times 256$	✗	2,048	647.48	N/A
	$64 \times 256 \times 256$	✗	10,000	578.80	N/A
HPDM-M	$16 \times 256 \times 256$	✗	2,048	178.15	69.76
	$16 \times 256 \times 256$	✗	10,000	143.06	84.29
	$64 \times 256 \times 256$	✗	2,048	324.72	N/A
	$64 \times 256 \times 256$	✗	10,000	257.65	N/A
HPDM-L	$16 \times 256 \times 256$	✗	2,048	92.00	71.16
	$16 \times 256 \times 256$	✗	10,000	66.32	87.68
	$64 \times 256 \times 256$	✗	2,048	137.52	N/A
	$64 \times 256 \times 256$	✗	10,000	101.42	N/A

Table 7. Additional zero-shot FVD evaluation results for UCF-101. For zero-shot evaluation, to the best of our knowledge, all the prior works use 10,000 generated videos to compute the I3D statistics.

Method	Resolution	FVD↓	IS↑
CogVideo [25]	$16 \times 480 \times 480$	701.6	25.27
Make-A-Video	$16 \times 256 \times 256$	367.2	33.00
MagicVideo [75]	$16 \times 256 \times 256$	655	-
LVDM [21]	$16 \times 256 \times 256$	641.8	-
Video LDM [4]	N/A	550.6	33.45
VideoFactory [63]	$16 \times 256 \times 256$	410.0	-
PYoCo [14]	$16 \times 256 \times 256$	355.2	47.46
HPDM-T2V	$16 \times 144 \times 256$	383.26	21.15
	$16 \times 256 \times 256$	728.26	23.46
	$16 \times 288 \times 512$	481.93	23.77
	$64 \times 256 \times 256$	1238.62	N/A
	$64 \times 288 \times 512$	1197.60	N/A

ing steps performed and also the latent dimensionality of

RINs [27]: 256, 512 and 1024, respectively. Our text-to-video model HPDM-T2V was fine-tuned for 15k steps and HPDM-T2V-1K for 100k steps. We provide the hyperparameters for our models in Tab. 8. For sampling, we use spatial 50% patch overlapping to compute the metrics (for performance purposes), and full overlapping for visualizations. We use stochastic sampling with second-order correction [29] for the first pyramid level. For later stages, we use Also, we disabled stochasticity for text-to-video synthesis since we have not observed it to be improving the results. We use 128 steps for the first pyramid stage, and then decrease them exponentially for later stages, dividing the number of steps by 2 with each pyramid level increase.

D. Failed experiments

In this section, we provide a list of ideas which looked promising intuitively, but didn't work out at the end — either because of some fundamental fallacies related to them, or the lack of experimentation and limited amount of time to explore them, or because of some potential implementation bugs which we have not been aware of.
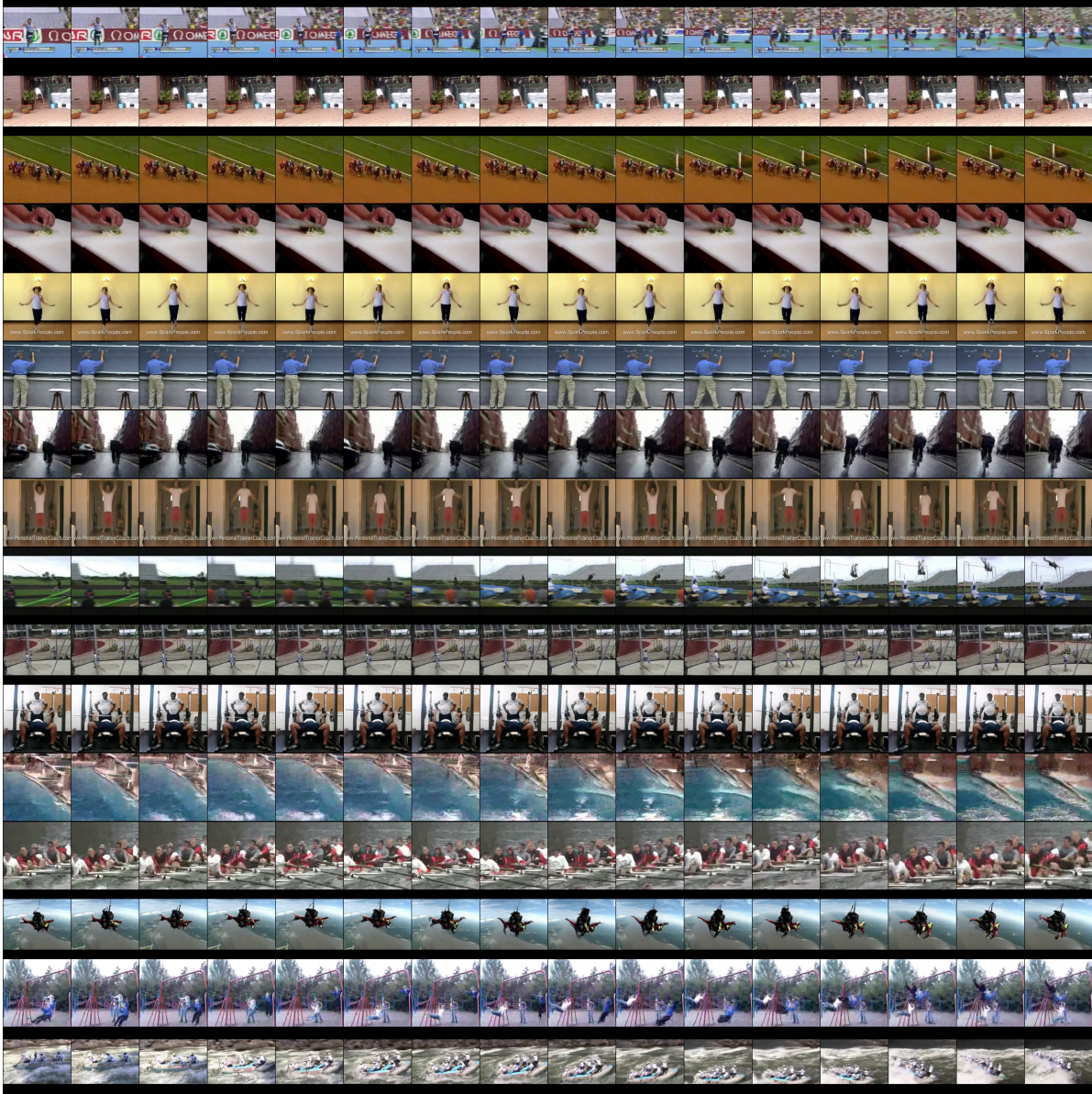
Figure 9. *Random* samples from HPDM-L on UCF-101 64×256^2 [56] without classifier-free guidance. We display 16 frames from a 64-frame-long video with $4\times$ subsampling.

1. *Cached inference has not sped up inference as much as we expected*. As described in Sec. 4.5 and Appendix C, we cache the activations from previous pyramid levels when sampling its higher stages. However, the speed-up was just $\approx 40\%$, which was not decisive. One issue is that we do not cache some activations (tokenizer activations and contexts). But the other reason is that grid-sampling is expensive. Grid sampling could be avoided by upsampling and then slicing, but this would lead to additional memory usage and will complicate the inference code.

2. *Positional encoding of the coordinates*. For some reason,

   the model started to diverge when we tried replacing raw coordinates with their sinusodial embeddings. We believe that this direction is still promising, but is underexplored.

3. *Stochastic sampling and second-order sampling for later stages*. For UCF-101, we use stochastic sampling for the first pyramid level, but disabled it for text-to-video generation. Also, second-order correction was producing grainy artifacts for later pyramid stages.

4. *Weight sharing between blocks*. To conserve GPU memory, we tried to share the weights between all the trans-
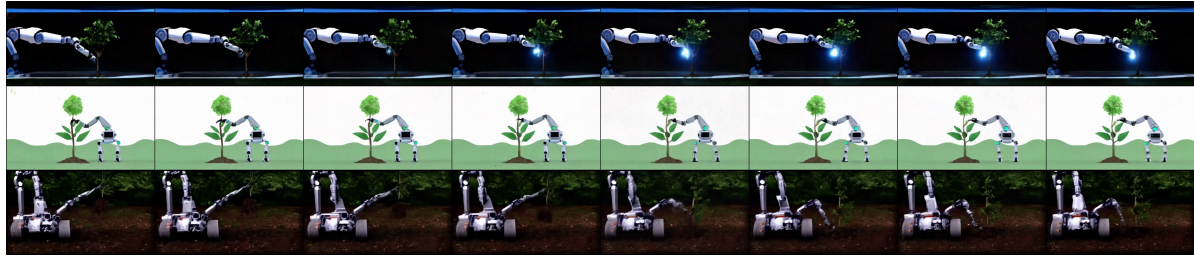
(a) "A robot planting a tree."

(b) "A confused grizzly bear in calculus class."
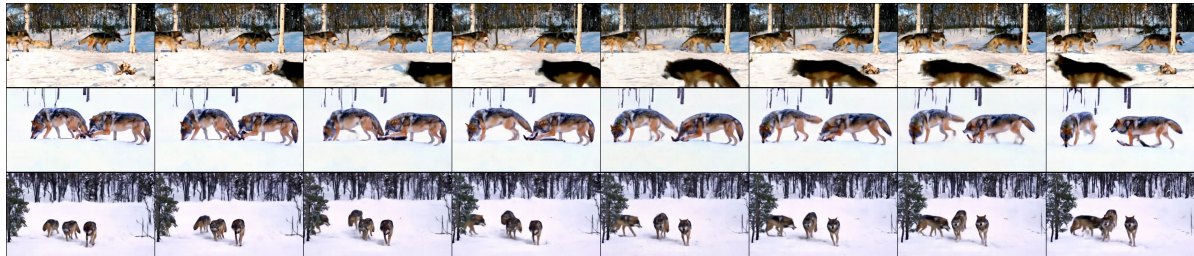
(c) "A high-definition video of a pack of wolves hunting in a snowy forest, natural behavior, dynamic angles."

(d) "A hot air balloon floating over a mountain range."

Figure 10. Text-to-video generation results for variable text prompts. Note that our text-to-video model has been fine-tuned only for 15k training steps from a 36×64 low-resolution generator. Animations and comparisons to the current SotA can be found in the supplementary.

former blocks, but that led to inferior results.

5. *Cheap high-res + expensive low-res U-Net backbone*. U-Nets were also not converging well for us in their regular design and were not giving substantial performance yields when combined with adaptive computation (only $\approx 10\%$ during training versus $\approx 50\%$ in RINs) due to the irregular amounts of blocks per resolution in their design.

6. *Random pyramid cuts*. Another strategy to make the later pyramid stages cheaper during training was to compute them only once in a while. For this, we would randomly sample the amount of pyramid stages for each mini batch per GPU. When parallelizing across many GPUs, this strategy gives enough randomness. While it decreased the training costs without severe quality degradation, it does not speed up inference and complicates logging.

7. *Mixed precision training*. It produced consistently worse convergence, either with manual mixed precision or autocast, either for FP16 and BF16.

8. *Fusing patch features for all the layers*. That strategy was not giving much quality improvement, but was tremendously expensive, which is why we gave it up.
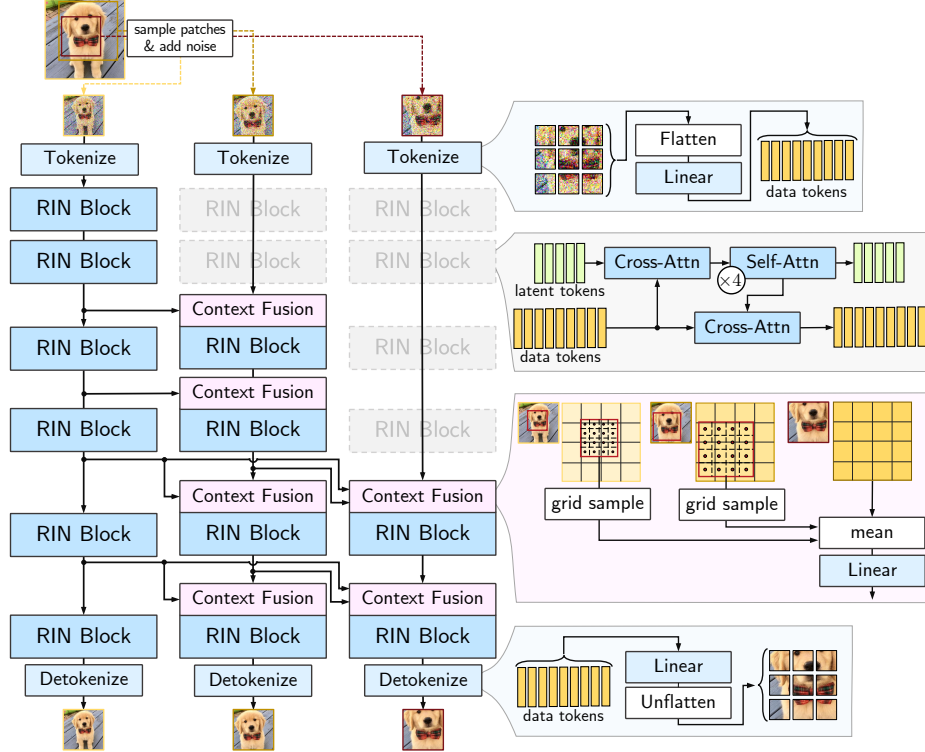
Figure 11. Full architecture illustration of HPDMwith depiction of the blocks.

Table 8. Hyperparameters for different variations of HPDM. For all the models, we used almost the same amount hyperparameters. For HPDM-T2V, we used joint video + image training which is reflected by its batch size. For HPDM-T2Vand HPDM-T2V-1K, we also used low-res pre-training by first training the lowest pyramid stage on 36×64-resolution videos for 500k steps.

Hyperparameter	HPDM-S	HPDM-M	HPDM-L	HPDM-T2V	HPDM-T2V-1K
Conditioning information	class labels	class labels	class labels	T5-11B embeddings	T5-11B embeddings
Conditioning dropout probability	0.1	0.1	0.1	0.1	0.1
Tokenization dim	1024	1024	1024	1024	1024
Tokenizer resolution	$1 \times 4 \times 4$	$1 \times 4 \times 4$	$1 \times 4 \times 4$	$1 \times 3 \times 4$	$1 \times 3 \times 4$
Latent dim	256	512	1024	3072	3072
Number of latents	768	768	768	768	768
Batch size	768	768	768	4096 + 4096	1024 + 1024
Target LR	0.005	0.005	0.005	0.005	0.005
Weight decay	0.01	0.01	0.01	0.01	0.01
Number of warm-up steps	10k	10k	10k	5k	5k
Parallelization strategy	DDP	DDP	DDP	FSDP	FSDP
Starting resolution	$16 \times 64 \times 64$	$16 \times 64 \times 64$	$16 \times 64 \times 64$	$8 \times 36 \times 64$	$16 \times 72 \times 128$
Target resolution	$64 \times 256 \times 256$	$64 \times 256 \times 256$	$64 \times 256 \times 256$	$64 \times 288 \times 512$	$16 \times 576 \times 1024$
Patch resolution	$16 \times 64 \times 64$	$16 \times 64 \times 64$	$16 \times 64 \times 64$	$8 \times 36 \times 64$	$16 \times 72 \times 128$
Number of RIN blocks [27]	6	6	6	6	6
Number of pyramid levels	3	3	3	4	4
Number of pyramid levels per block	1/1/2/2/3/3	1/1/2/2/3/3	1/1/2/2/3/3	1/2/2/3/3/4	4/4/4/4/4/4
Number of parameters	178M	321M	725M	3,934M	3,934M
Number of training steps	40k	40k	65k	15k (+ 500k)	100k (+ 500k)

E. Potential negative impact

We introduced a patch-wise diffusion-based video generation model: a new paradigm for video generation that is a step forward in the field. While our model exhibits promising capabilities, it's essential to consider its potential negative societal impacts:

- *Misinformation and Deepfakes*. While our text-to-video model underperforms compared to the largest existing ones (.e.g, [22, 51]), it demonstrates a promising direction on how to improve the existing generators further, which creates a risk of generative AI misuse in creating misleading videos or deepfakes. This can contribute to the spread of misinformation or be used for malicious purposes.
- *Intellectual Property Concerns*. The ability to generate videos can lead to challenges in copyright and intellectual property rights, especially if the technology is used to replicate or modify existing copyrighted content without permission.
- *Economic Impact*. Automation of video content generation could impact jobs in industries reliant on manual content creation, leading to economic shifts and potential job displacement.
- *Bias and Representation*. Like any AI model, ours is subject to the biases present in its training data. This can lead to issues in representation and fairness, especially if the model is used in contexts where diversity and accurate representation are crucial.

  To address the potential negative impacts, it is crucial to:
- Develop and enforce strict ethical guidelines for the use of video generation technology.
- Continuously work on improving the model to reduce biases and ensure fair representation.
- Collaborate with legal and ethical experts to understand and navigate the implications of video synthesis technology in terms of intellectual property rights. Engage with stakeholders from various sectors to assess and mitigate any economic impacts, particularly concerning job displacement.

In conclusion, while our model represents a notable advancement in video generation technology, it is imperative to approach its deployment and application with a balanced perspective, considering both its benefits and potential societal implications.