# 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 $4 \times 4$ 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.

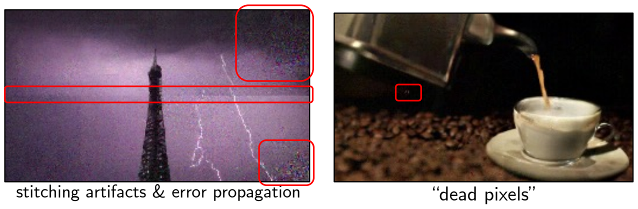


Figure 7. Illustrating the failure cases of HPDM..

## 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 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).

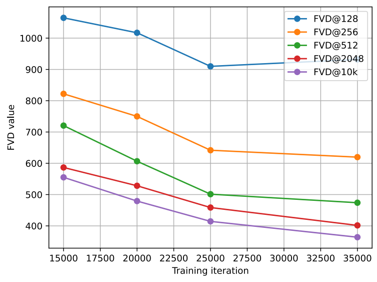


Figure 8. Using different amounts of fake videos to compute FVD [59] gives very correlated, but offset values with the main trend being "the more —- the better". We hypothesize that using more synthetic samples yields better coverage of different modes of the data distribution and decreases the influence 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.

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 inutitively, 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.

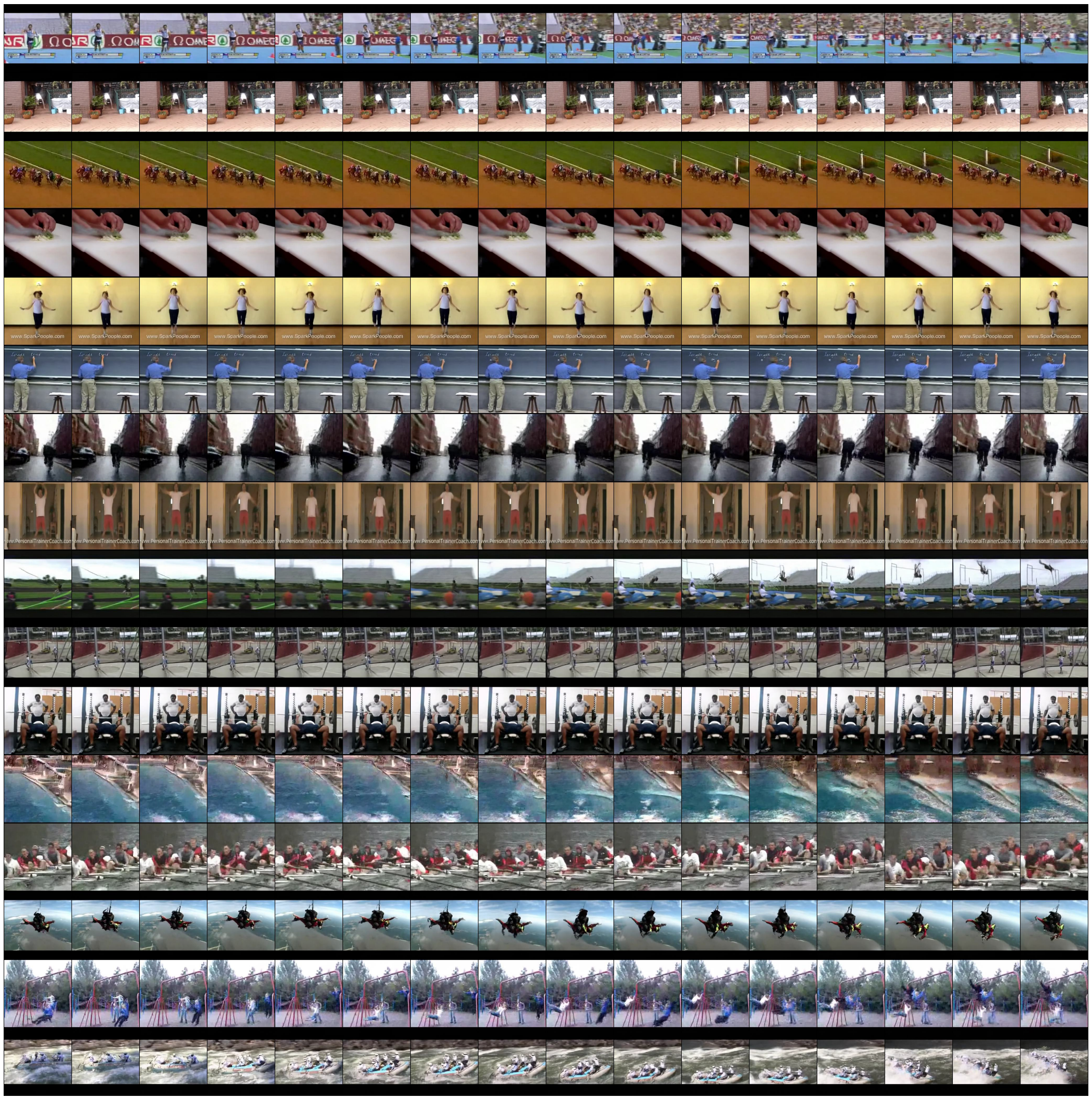
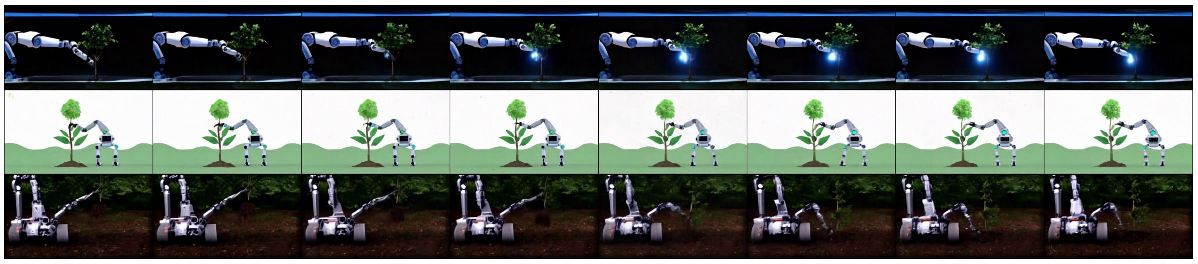
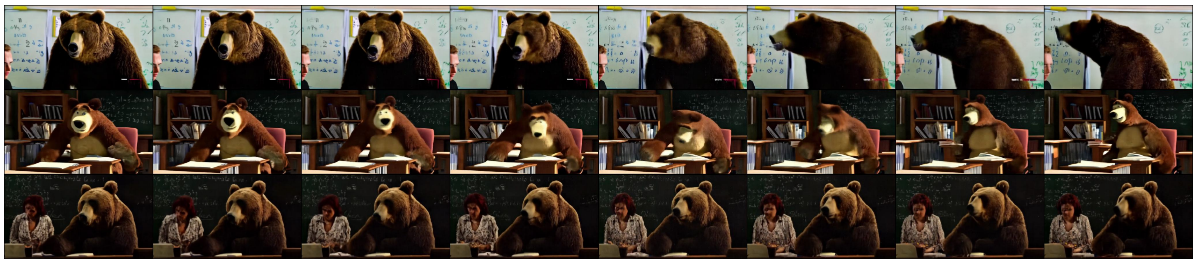
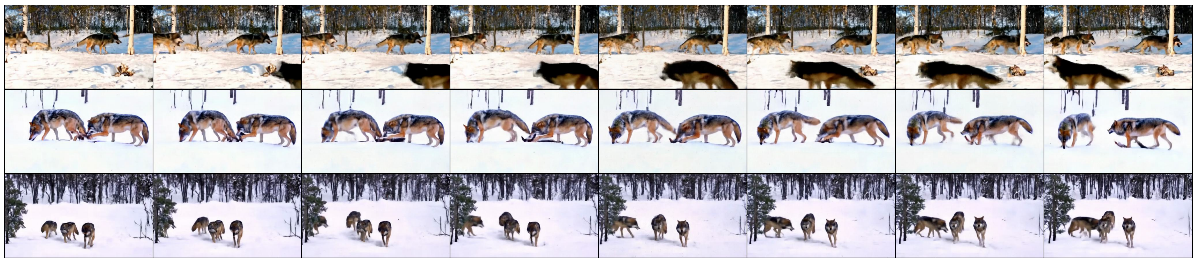


Figure 9. *Random* samples from HPDM-L on UCF-101 $64 \times 256^2$ [56] without classifier-free guidance. We display 16 frames from a 64-frame-long video with $4\times$ subsampling.

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."

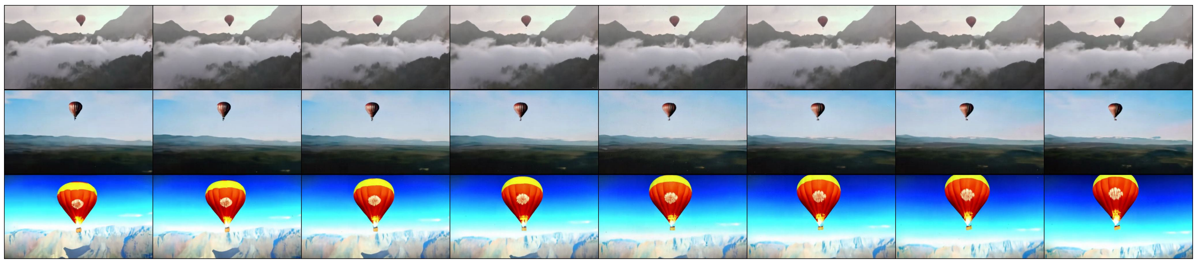

(d) "A hot air balloon floating over a mountain range."

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 $36 \times 64$ 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.

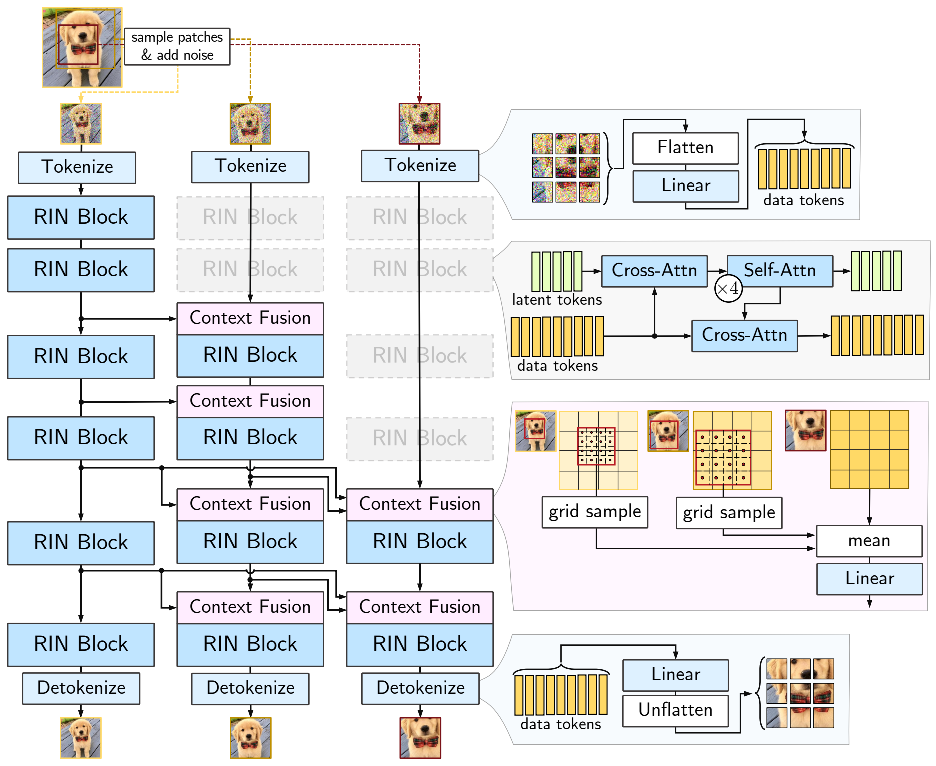


Figure 11. Full architecture illustration of HPDMwith depiction of the blocks.

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.