MILLIVID: Hierarchical Latents for Long-Range Consistency in Video Generation

Ishaan Preetam Chandratreya*
Massachusetts Institute of Technology
ishaanpc@mit.edu

David Charatan*
Massachusetts Institute of Technology
charatan@mit.edu

Basile Van Hoorick
Toyota Research Institute
basile.vanhoorick@tri.global

Sergey Zakharov
Toyota Research Institute
sergey.zakharov@tri.global

Vitor Guizilini
Toyota Research Institute
vitor.guizilini@tri.global

Phillip Isola
Massachusetts Institute of Technology
phillipi@mit.edu

Vincent Sitzmann
Massachusetts Institute of Technology
sitzmann@mit.edu

Abstract

Video generative models have become increasingly powerful, but long-range consistency remains challenging to achieve because even a few dozen frames require impractically long transformer sequence lengths. We show that this issue can be mitigated by generating video using coarse-to-fine rollout within a multi-scale token space. Our approach is simple: first, we pre-train an autoencoder that compresses each frame into a hierarchy of tokens, with levels ranging from the typical latent resolution to only a handful of tokens per frame. The coarsest levels capture the most consequential information—such as scene layout and semantics—while finer levels add high-frequency appearance and texture. Then, we train a video diffusion model to generate these tokens using coarse-to-fine rollout. By carefully controlling the level of detail at which frames are generated and used as context during each rollout step, we are able to preserve long-range consistency in geometry and object permanence while spending less compute on the long-range consistency of less perceptually relevant details. We validate this approach using a custom dataset of long Minecraft videos, where it produces substantially more consistent rollouts compared to existing baselines. Project page: davidcharatan.com/millivid.

1 Introduction

Video generative modeling has advanced rapidly in realism, scale, and generality, with growing relevance to applications such as computer graphics, robotics, and world modeling [1–6]. Yet long-range consistency—generating long videos that stay coherent from start to finish—remains an ongoing challenge. Current models are largely designed for autoregressive rollout, generating long videos chunk by chunk, with the most recent chunk serving as context for the next. This produces long videos, but sacrifices consistency: as chunks exit the context, their content is forgotten.

A straightforward solution to this problem is to increase the context length. However, this quickly becomes prohibitive, as compute costs scale quadratically with sequence length in transformers and each additional video frame adds several hundred tokens. A more efficient approach, exemplified by FramePack [7], is to allocate fewer tokens to distant context frames. This strategy rests on the observation that different visual information needs to persist over different temporal horizons. Coarse structure, such as the layout of a scene, must be faithfully preserved over long time spans, as any inconsistency is immediately noticeable. Fine structure, such as exact texture patterns, can safely be forgotten, since long-range inconsistency within minor details is less perceptible. However, we find

*Equal contribution. Order decided by coin flip.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

that this insight alone does not tell the whole story—surprisingly, FramePack largely fails to recall content that leaves the camera frame, even if that content remains in its compressed context window.

We therefore present two further insights that enable consistent long-context video generation. First, where prior work on token-efficient image and video generation has often compressed by simply downsampling or patchifying in either pixel or latent space, we find that training a hierarchical tokenizer yields compression that better preserves relevant detail. Second, we hypothesize that a model trained for predicting only short future time horizons learns to only consider the most recent context, mirroring insights for training long-horizon policies in robot behavior cloning [8–10]. We hence force the model to make predictions *far into the future*.

Based on these insights, we present MILLIVID, a diffusion-based video generative model designed to maximize long-range consistency under a fixed sequence length. Our approach has two components. First, inspired by flexible image tokenizers [11, 12], we train an autoencoder with a hierarchical latent space, in which each level represents an image using a specific number of tokens. In this latent space, coarser levels capture global structure, while finer levels add detailed appearance and texture. Second, we train a video diffusion model for coarse-to-fine generation in this latent space. Generation starts at the coarsest level, where the small number of tokens per frame allows the model to cover many frames, and then progressively refines toward finer scales. Transformer weights are shared across all scales; the same transformer with a fixed sequence length performs generation at every scale.

We evaluate our method on long videos of Minecraft gameplay, which we find to be particularly suitable for measuring long-range consistency. Compared with prior work on long-context video generation, our method produces substantially more consistent long rollouts, successfully recalling details and scene structure that baselines forget, without relying on retrieval or explicit 3D maps. Our contributions are as follows:

- We show that hierarchical autoencoding can be combined with a novel coarse-to-fine rollout strategy to achieve long-range consistency in video generation.
- We demonstrate that under a common sequence length constraint, our proposed method produces significantly better consistency than FramePack, the state-of-the-art baseline, as well as typical autoregressive rollout, without sacrificing per-frame quality.

2 Related Work

Retrieval-augmented video generation seeks to extend the effective temporal context of a video generative model by retrieving a small set of frames or chunks from the distant past and appending them to the current context window [13–18]. Our approach fundamentally differs in that we do not perform retrieval and instead rely only on a coarse-to-fine hierarchy to extend the temporal context of the transformer. Our method is orthogonal and could be combined with a retrieval mechanism for past frames that fall outside the temporal context of the coarsest level of our hierarchy.

3D Memory A particularly effective way to store information about past generations is to leverage 3D geometry. Garcin et al. [19] use unprojection and projection into a 3D voxel grid for persistent 3D scene memory. Huang et al. [20] build an incremental 3D point cloud and use camera pose to retrieve past frames most relevant to the current target frames. Wu et al. [21] maintain a geometry-grounded spatial memory as an autoregressively generated point map, fusing only static scene content to enforce consistency across revisits. We pursue a conceptually different direction that assumes neither camera poses nor an explicit 3D world.

Flexible-length tokenization learns representations whose token length can vary with the desired compression level. In images, variable-length tokenization has been shown to naturally induce a coarse-to-fine ordering that can be leveraged for coarse-to-fine image generation [11, 22–24]. SceneTok [25] demonstrates that flexible-length tokenization can similarly accelerate 3D scene generation for novel view synthesis. Flexible-length *spatiotemporal* tokenization has been explored for video to facilitate efficient generation [24, 26, 27], but has not been used to improve long-range video consistency. By contrast, we use a *per-frame* tokenizer in conjunction with a latent diffusion model to demonstrate long-range consistent video generation. Our frame tokenizer can be considered a multi-scale variant of ElasticTok [24], and is related to other recent works in adaptive tokenization of images and video [11, 12, 28, 29].

Multi-scale generation is a classical way to reduce the cost of generative modeling by generating coarse structure first and refining detail later. In images, this idea appears in hierarchical latent

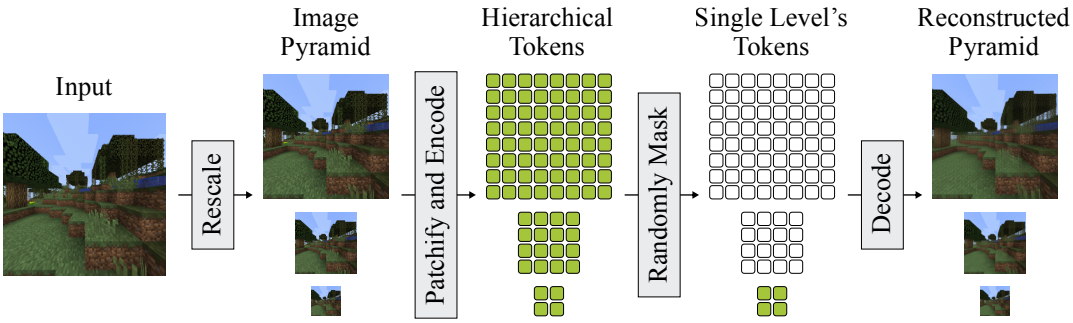


Figure 1: Our hierarchical autoencoder consists of a multi-resolution encoder-decoder pair. The encoder receives a multi-resolution image pyramid as input. It patchifies the pyramid's images using a fixed kernel size, yielding fewer tokens for coarser levels, then feeds the tokens through a transformer. During training, we zero out all but one randomly chosen level. The decoder, also a transformer, must reconstruct the entire resolution cascade based on the remaining tokens. The highest-resolution reconstruction is supervised using MSE and LPIPS, while the others are supervised only using MSE. We show three levels and a small number of tokens here for clarity; in practice, we use more.

models such as NVAE [30], cascaded diffusion models [31, 32], and more recent coarse-to-fine generators such as Edify Image and VAR [33, 34]. In video, Imagen Video and I2VGen-XL use cascaded diffusion pipelines [35, 36], while Pyramidal Flow Matching organizes generation across pyramid stages in a unified model [37]. TECO [38] achieves a long temporal context using aggressive spatial downsampling of latents before temporal attention. FAR [39] proposes separate patchification schemes for far-away and recent frames. Harvey et al. [40] train video models to predict subsets of a video's frames from other subsets. FramePack [7] is the most related and recent approach to multi-scale video modeling for long-range consistency. It allocates fewer tokens to frames that are less salient or further in the past, enabling a longer past context window, but generates full-resolution target latents. Our method is likewise coarse-to-fine, but differs from prior work in two key ways: we learn the scale space itself, and we explicitly allocate a transformer's fixed token budget across scales, generating frames coarse-to-fine, to maximize temporal context and long-range consistency.

3 Method

In this section, we present MILLIVID, a two-stage algorithm for generating videos with long-term consistency. First, in Section 3.1, we introduce an autoencoder that encodes frames into a hierarchical latent space, where each level contains a different number of tokens. Next, in Section 3.2, we introduce a video generative model that is trained within this hierarchical latent space. Given a fixed transformer sequence length, this model can flexibly alternate between generating many highly compressed frames and fewer highly detailed frames. We harness this ability by designing a sampling algorithm that leverages both regimes, first generating long, highly compressed sequences, and then gradually upsampling them to generate fine details.

3.1 Hierarchical Autoencoding

Our hierarchical autoencoder is designed to trade off between visual fidelity and token count. As shown in Figure 1, it encodes a single video frame into a hierarchy of latent representations with a fixed number of levels. At level ℓ, the latent representation uses the following number of tokens:

$$\textbf{per-frame token count } N_\ell \textbf{ at level } \ell \qquad N_\ell = \frac{H}{2^\ell} \times \frac{W}{2^\ell}$$

The finest representation is at $\ell = 0$, where the frame is encoded into a grid of $H \times W$ tokens. Each coarser level halves the previous level's resolution along both the height (H) and width (W) dimensions. Coarser levels are more compressed, while finer levels yield higher-quality reconstructions. Each level can be decoded on its own without access to the other levels' tokens.

Our encoder is a transformer that simultaneously outputs the full latent hierarchy. To encode an image, we first transform it into a multi-resolution pyramid containing one image per hierarchy level.

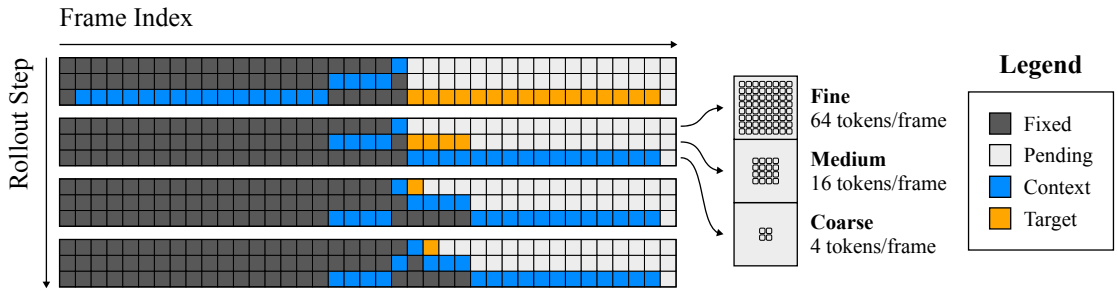


Figure 2: To generate videos, we use coarse-to-fine rollout. During each rollout step, the model sees a mixture of **context** and **target** tokens across different hierarchy levels; it does not see the **fixed** (already generated) or **pending** (not yet generated) tokens. Our model's first four rollout steps are shown on the left. Each step is represented as a 3-row grid where the rows represent hierarchy levels and each column represents a single frame. Over the first three steps, the model generates a long sequence of coarse frames, a medium-length sequence of medium frames, and a single fine frame. The full rollout sequence is shown on our accompanying project page and in Figure 10 in the appendix. We show three levels here for clarity; in practice, we use more.

Within this pyramid, the highest-resolution image matches the input frame's resolution, while the subsequent images each halve the previous image's height and width. We patchify the pyramid's images using a fixed kernel size (with shared weights) across levels to produce $\frac{H}{2^\ell} \times \frac{W}{2^\ell}$ tokens per level; add positional encodings for each token's row, column, and level index; and then feed all of the tokens through the encoder. This produces the complete latent hierarchy—the first $H \times W$ output tokens correspond to $\ell = 0$, the next $\frac{H}{2} \times \frac{W}{2}$ correspond to $\ell = 1$, and so on.

Our decoder is a transformer that reconstructs the input frame from a single level's latent tokens. It returns the input frame's reconstruction as the highest-resolution image in a multi-resolution pyramid that mirrors the encoder's input. To decode a particular level's tokens, we take the encoder's output and zero out the other levels' tokens. We then add positional encodings for row, column, and level; feed the tokens through the decoder; and unpatchify the output, mirroring the encoder's patchification.

To train the autoencoder, we randomly sample a level to decode at, then supervise on the resulting reconstruction. Regardless of which level was sampled, we supervise on the entire output pyramid using MSE and also supervise on the highest-resolution image using LPIPS [41]. The lower-resolution output images serve to accelerate and stabilize convergence; they are discarded during inference.

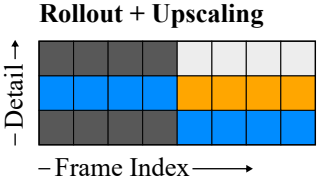
3.2 Coarse-to-Fine Video Generation

In this section, we describe our video generative model, which is a transformer-based latent diffusion model that operates within our autoencoder's hierarchical latent space.

The fundamental constraint that our model is designed around is a transformer's sequence length S. Consider that in a typical video diffusion model, a video is represented as $F \times H \times W$ tokens. For a 30-second, 20 fps video with 16×16 tokens per frame, this equals $600 \times 16 \times 16 = 153{,}600$ tokens. In all but the most compute-rich settings, this exceeds feasible values of S, precluding us from feeding the entire video to the transformer at once. Thus, we must design video models that operate under the constraint $S \ll F \times H \times W$ while still producing temporally coherent long videos.

Prior work has addressed this problem by using autoregressive rollout, which is subject to rapid forgetting. Consider the case of $S = 1024$, which is enough for two context frames and two generated frames. In this case, we can generate F frames in two-frame chunks, but only have 0.1 seconds of context. As a result, anything that even momentarily exits the frame will be forgotten.

Our solution to this problem leverages our hierarchical autoencoder's ability to control the number of tokens allocated to each frame. Broadly speaking, instead of rolling out at the full resolution, we roll out at the coarsest resolution—where many frames fit into S—then upscale the resulting frames. The most obvious upscaling strategy, shown in the inset, conditions the **generated** frames on **context** consisting of the recent past and the next-coarser level.



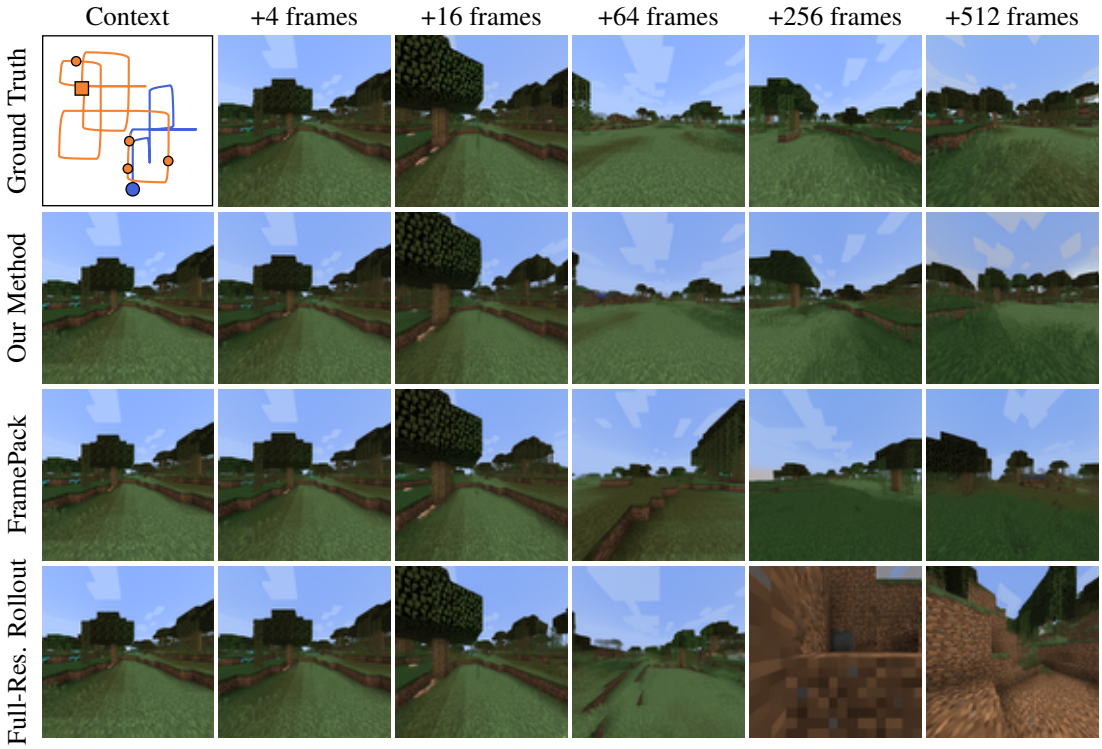


Figure 3: Unlike the baselines, our method reliably recalls the scene's structure, even when many frames have elapsed. The top-left panel shows a top-down view of the trajectory the models follow. Blue is context; orange is generated. The points at which frames are shown are marked. We encourage the reader to consult our project website for videos of similar comparisons. Note that because our method is designed to preferentially retain the most consequential information over long horizons, it does not capture minor details (e.g., exact vine and grass textures) in the rightmost columns.

Unfortunately, as we will show, this approach is doomed to generate inconsistencies in the resulting video. To illustrate why this happens, consider a video in which the camera approaches a street sign whose text is currently too small to decipher. When we play the video, the sign grows closer, and the text becomes legible. If we could instead accurately super-resolve the video, we could read the text from far away. In other words, uncertainty (about the sign's text) is resolved both with increased resolution and temporal rollout. The issue with the aforementioned strategy is that it *independently* performs super-resolution and temporal rollout. If super-resolution resolves the sign's text as "stop" and temporal rollout resolves it as "go," we are left with an awkward inconsistency.

Our method, shown in Figure 2, solves this problem by carefully considering which tokens to use as context to avoid creating inconsistencies. Like the aforementioned strategy, it alternates between generating many coarse frames and fewer fine frames at once. However, critically, its context always includes both the highest-resolution most-recent frame *and* future frames that have only been generated up to coarser levels, ensuring that situations like the stop-go discrepancy cannot occur. Consult Figure 10 for an overview of the exact rollout procedure.

Training Procedure As shown in Figure 10, our model's rollout consists of many similar but distinct steps. At training time, we randomly sample from these steps and supervise the model on its ability to denoise the **generated** tokens conditioned on clean **context** tokens; the other tokens are not shown to the model. An important property of our method is that each rollout step requires the same number of tokens (without padding). As a result, different rollout steps can trivially be stacked into the same batch for efficient training. We use a single model for all rollout steps, and positional encodings for each token's level, frame, row, and column let the model distinguish between steps. We train our model on a single 8xH200 node; it takes around 2 days to reach 192,000 steps.

Table 1: Consistency and quality metrics averaged over short (frames 1-64), medium (frames 65-256), and long horizons (frames 257-768). We measure consistency using PSNR, LPIPS, SSIM, DINOv2 class token cosine similarity, and LightGlue matches. Our LightGlue match metric counts the number of keypoint matches detected by LightGlue with confidence greater than 0.5. We measure quality using FID and FVD. In each column, the best-performing method is highlighted in **bold**. See Figure 8 in the appendix for a plot showing a fine-grained view of the information presented in this table.

Method	PSNR ↑			LPIPS ↓			SSIM ↑		
	short	med.	long	short	med.	long	short	med.	long
Our Method	**21.78**	**19.13**	**16.69**	**0.159**	**0.247**	**0.335**	**0.612**	**0.562**	**0.522**
FramePack	19.17	14.08	11.98	0.238	0.437	0.533	0.607	0.491	0.434
Full-Res. Rollout	17.58	11.95	11.02	0.290	0.538	0.630	0.583	0.439	0.407

Method	DINOv2 ↑			Matches ↑			FVD ↓			FID ↓		
	short	med.	long	short	med.	long	short	med.	long	short	med.	long
Our Method	**0.943**	**0.927**	**0.906**	**166.4**	**94.5**	**62.4**	**43.8**	**57.8**	**83.7**	**23.06**	**26.46**	**29.13**
FramePack	0.920	0.854	0.803	138.7	22.5	7.0	64.9	145.0	286.2	24.92	39.89	57.72
Full-Res. Rollout	0.906	0.824	0.754	144.6	12.7	5.0	78.1	211.6	607.1	24.69	41.06	76.96

4 Results

In this section, we validate our model's long-range consistency against FramePack [7], the most similar state-of-the-art approach to our method, as well as full-resolution autoregressive rollout.

Datasets A useful dataset for measuring long-range consistency must have four properties. First, it must contain videos that are long—several hundred frames or more. Second, it must provide enough videos to facilitate training a generative model. Third, it must contain videos that can be used to measure memory, where the same content frequently exits and later re-enters the frame. Finally, it must provide fine-grained conditioning signals (e.g., poses or actions) that can be used to steer video models towards previously-seen content.

Existing datasets fall short of these criteria. For instance, video datasets like Kinetics [42] do not provide appropriate fine-grained conditioning. Posed video datasets like RealEstate10k [43], ACID [44], and DL3DV-10k [45] contain relatively short videos or too few instances of content exiting and re-entering the frame. Visual odometry datasets like KITTI [46] and TUM RGB-D [47] contain too few sequences for training.

We address this issue by generating LOOPCRAFT, a dataset of 1024-frame videos of Minecraft gameplay. It contains 200,000 videos at 256×256 resolution, with metadata for both action and pose conditioning. Created using a dataset generation pipeline adapted from TECO's [38], it uses the MineRL simulator [48] to collect trajectories of an agent running through the world and intermittently making random 90-degree turns. The agent is slightly biased towards making pairs and quartets of turns, and as a result, it frequently returns to previously seen areas of the world. This makes the LOOPCRAFT dataset ideally suited to measuring long-range consistency.

Metrics Our metrics are designed to independently measure two aspects of video generation: consistency and quality. Consistency is a model's ability to accurately recall previously seen content; quality is the ability to generate high-quality frames during rollout, regardless of whether those frames are consistent or not. The distinction between consistency and quality has previously been studied by [49] and referred to as the drifting-forgetting tradeoff [7].

To measure consistency, we condition each video model on up to 256 frames (depending on its supported context length) of a 1024-frame ground-truth sequence, plus the ground-truth actions for all 1024 frames, then compare its 768-frame generated rollout against the ground truth. Effectively, the question we ask is: if the model sees the world and then follows a prescribed path through it, can it reproduce the world's content correctly? Our comparison uses image metrics—peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) [50], and perceptual similarity [41]—in addition to DINOv2 class token cosine similarity [51] and LightGlue keypoint match count [52].

We empirically find that measuring consistency can be challenging for two reasons. First, it is difficult to guarantee that the content the models are asked to reproduce has been seen in the context. To

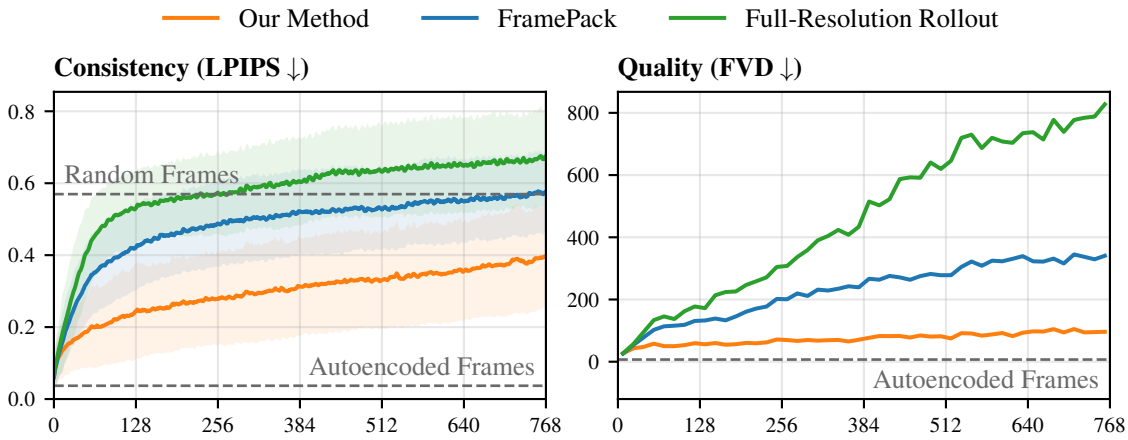


Figure 4: We separately measure consistency and quality on the LOOPCRAFT dataset. The X axis indicates the number of frames each model has generated since seeing ground-truth context. We use LPIPS to measure consistency—i.e., how well generated videos match ground-truth videos following the same trajectories. We use Fréchet Video Distance (FVD) as a measure of visual fidelity that is independent of consistency. Our method exceeds the baselines on quality and consistency. Refer to Figure 8 in the appendix for further plots of consistency and quality metrics.

alleviate this issue, we generate a 1,000-video test set whose trajectories have high overlap between the context and the remaining trajectory (details in appendix A.1). Second, even well-performing models can produce trajectories that drift slightly compared to the ground truth. This manifests as slight misalignments in otherwise consistent scenes. Several of our metrics (LPIPS, DINOv2, and keypoints) are robust to such shifts, and we empirically find that LPIPS is an especially good measure of consistency in the face of slight misalignments, as previously reported in [53].

To measure quality, we use two metrics: Fréchet Inception Distance (FID) [54] and Fréchet Video Distance (FVD) [55]. We find that these metrics roughly correlate with perceived quality.

Models We compare our model to FramePack [7] and full-resolution autoregressive rollout. Since the baselines do not use hierarchical tokenization, we train them using our autoencoder's highest-resolution latent space. All models share the same decoder, which decodes from this latent space to images. All models are action-conditioned; the action space consists of a single ternary value that indicates whether the agent is turning left, turning right, or moving forward. We describe the models below; see appendix A.2 for further training details.

MILLIVID (Our Method): We train our model with 4 latent levels. In descending order of resolution, we train our model with 3, 12, 48, and 192 frames of context, which yields a total budget of 3840 tokens. As a result, our model simultaneously denoises up to 192 frames (at the coarsest level) with up to 255 frames of context.

FramePack: In descending order of resolution, FramePack is trained with 3, 12, 48, and 192 frames of context. While FramePack uses varying patchification of full-resolution latents rather than hierarchical latents, the number of tokens per frame of context matches our model exactly. We train FramePack to denoise three high-resolution frames at once, which yields a token budget of exactly 3840 tokens. Note that the FramePack baseline does not implement the original FramePack paper's endpoint planning strategy (i.e., first generating frames that are far in the future and then generating intermediate frames), since we wish to study the effects of varying rollout strategies in isolation.

Full-Resolution Autoregressive Rollout: This model denoises full-resolution target latents conditioned on full-resolution context latents. We train this model with seven frames of context and seven denoised frames, for a token budget of 3584 tokens.

Results Our method clearly outperforms the baselines in terms of consistency and quality. See Figure 3 for qualitative results, Table 1 for quantitative results, and Figure 4 for a fine-grained view of how consistency changes as rollout progresses. An expanded version of Figure 4 is available in the appendix as Figure 8. We further urge the reader to consult our accompanying project page, which includes video results, to qualitatively judge the presented models.

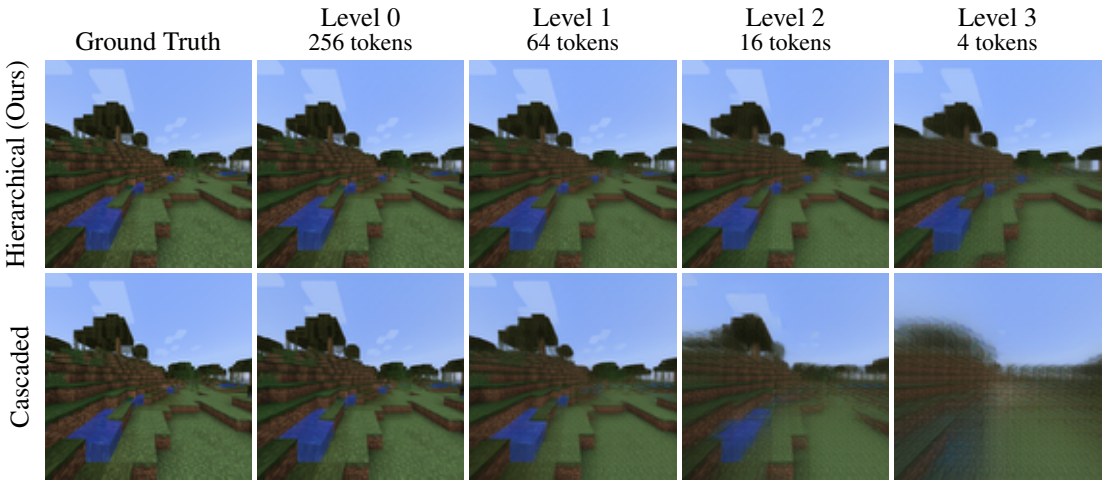


Figure 5: As our hierarchical autoencoder's token budget decreases, it discards fine-grained texture and geometry while retaining coarse scene structure (top row). Compared to an autoencoder trained to decode mean-pooled, full-resolution latents (bottom row)—the kind of latents one would find in a cascaded diffusion model—our hierarchical autoencoder produces significantly better reconstructions.

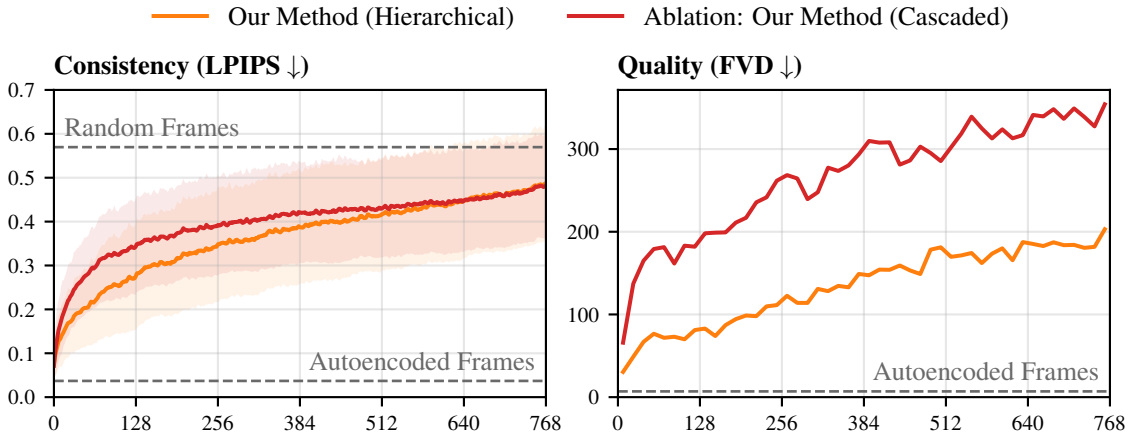


Figure 6: A cascaded variant of our method, in which the model is trained to operate on downscaled (mean-pooled) versions of our highest-resolution hierarchy level instead of using the full latent hierarchy, performs worse on both consistency and quality. For all ablations, we train models with sequence lengths of 1280, where 1/3 as many frames are seen at every level compared to the models in Figure 4. Note that the mean-pooled tokens the cascaded model sees are also visualized in Figure 5.

5 Analysis

We investigate our core design choices—hierarchical autoencoding and coarse-to-fine generation within a hierarchical latent space—against reasonable alternatives through three central questions.

Q1: What information is contained within our hierarchical latent space? To answer this question, we visualize our hierarchical autoencoder's latent space. See Figure 5 (top row) for an example of such visualization. We find that our highest-resolution latent level faithfully recreates the input image, including both structure and texture, while the most compressed latent level retains coarse scene structure and forgets exact textures. We highlight that this is significantly more desirable than simply producing a low-frequency (i.e., blurry) reproduction of the input image, since the most important scene structure remains clearly defined.

To highlight the benefits of our hierarchical latent space, we train a variant of our autoencoder in which the coarser levels are defined as downscaled (mean-pooled) versions of the finest level rather than being learned. In this case, the decoder sees a resolution cascade, exactly as a cascaded diffusion

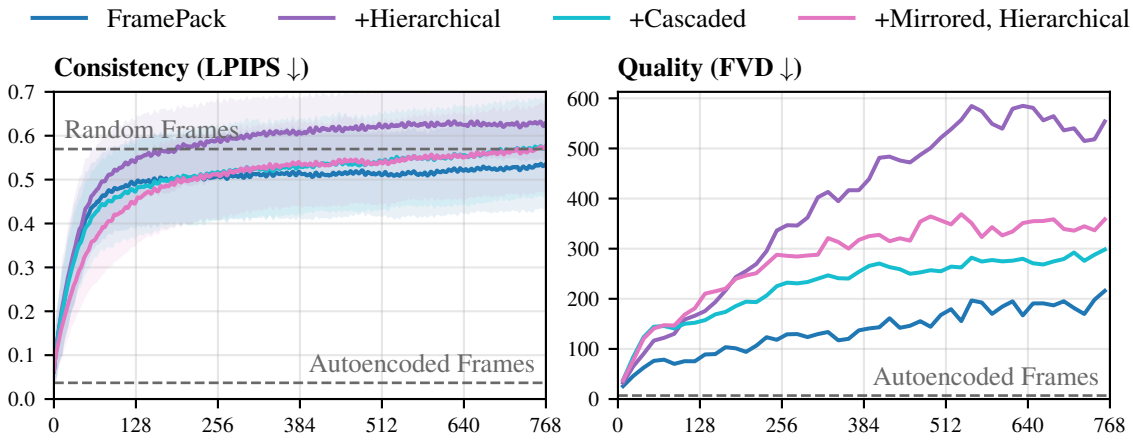


Figure 7: We plot three variants of FramePack against the original FramePack model. See Section 5 (Q3) for a description of each model. The mirrored variant performs better on consistency despite having a shorter context window; all perform worse on quality. We find that the hierarchical variant's consistency becomes worse than random frames due to rollout instability (i.e., because the model's outputs collapse, their consistency is worse than that of random in-distribution frames).

model would. Figure 5 (bottom row) shows the results of this experiment: the best the decoder can do is produce a blurry version of the input image, where most structure is lost.

Q2: How does a hierarchical model compare to an identically trained cascaded one? To answer this question, we train a cascaded variant of our method. In this version, rather than predicting our highest-resolution latents from a series of coarser hierarchical latents, the model operates on a series of downscaled (mean-pooled) versions of the highest-resolution latents. Figure 6 shows that this variant of our model is clearly worse, both in terms of consistency and quality.

Q3: Does FramePack benefit from hierarchical latents, and do models that can access many frames of context inherently give better consistency? We analyze three variants of FramePack: the original model, whose adaptivity stems from varying patchification kernel sizes; a variant trained using our hierarchical latent space; and a cascaded variant that uses the same cascaded latent space as in Figure 6. We also modify FramePack to allocate half of its capacity to the target, such that it not only has coarse context, but also coarse targets that exactly mirror the context's token layout. This "mirrored" FramePack model predicts coarse, long-range targets at training time, but discards them at sampling time, where it behaves like a FramePack model with about half the usual context length. See Figure 7 for the results of these experiments. We find that the hierarchical and cascaded variants perform worse across both consistency and quality. Surprisingly, despite having a shorter context window, the mirrored variant performs better than FramePack on consistency at certain time horizons. We hypothesize that this is because, despite having a long context window, the original FramePack model only predicts a short distance into the future, where it can mostly rely on information from the most recent context frame. By being asked to predict further into the future, the mirrored model may be forced to examine the distant past for information, improving its overall consistency.

6 Conclusion

In this paper, we observe that reasoning over long visual sequences necessarily means forgetting *something*: given a fixed sequence length, we have to decide on what past and present information to fill it with. We suggest that in long video generation, we should compromise on the long-range consistency of fine detail, trading it off against an increased temporal horizon for coarse structure. We hence learn a coarse-to-fine hierarchy by training an adaptive-length frame tokenizer, and then fill our token budget preferentially with many past coarse frames and only a few recent fine frames.

Our study suggests that such multi-scale video generative modeling can yield dramatic gains in long-range consistency, recalling 3D scene structure over hundreds of frames where a conventional video generative model can only afford a handful of context frames. Beyond its practical potential, we hope that our work will encourage future exploration of multi-scale video generative modeling.

Limitations and Future Work Since our method relies on a custom-trained hierarchical autoencoder, it is not designed to be fine-tuned from standard video diffusion models. As a result, fine-tuning a large, pre-trained model like Wan [56] or Hunyuan Video [57] for use with MilliVid's rollout strategy is not straightforward. We hope that future work will explore doing so—for example, one could distill these methods' original autoencoders to be hierarchical (e.g., by aligning a hierarchical autoencoder's finest latent space to the original latent space), then fine-tune the associated latent diffusion model to be hierarchical. Additionally, compared to FramePack, our model requires about 33% more rollout steps (see A.4 in the appendix). However, we believe this is a worthwhile tradeoff for the dramatically increased consistency our method offers. Moreover, our method does not address the quadratic scaling of transformer attention, unlike methods that leverage RNNs or state-space models [58, 59]. And finally, since MILLIVID allocates fewer tokens to the distant past, it is unable to recall fine details (e.g., text and logos) over long horizons.

7 Acknowledgements

We thank Andrew Song and Hannah Schlueter for their feedback during the process of writing and editing the paper. This work was supported by the Toyota Research Institute (TRI) University 3.0 (URP) program, the National Science Foundation under Grant No. 2211259, by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) under 140D0423C0075, by the Amazon Science Hub, by the MIT-Google Program for Computing Innovation, by AMD via the MIT AI Hardware Program, and by a 2025 MIT Office of Research Computing and Data Seed Grant. Ishaan Chandratreya is supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 2141064. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any other entity.

References

[1] Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models, 2025. URL https://arxiv.org/abs/2509.24527. 1

[2] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems*, volume 36, pages 9156–9172, 2023.

[3] Boyang Wang, Nikhil Sridhar, Chao Feng, Mark Van der Merwe, Adam Fishman, Nima Fazeli, and Jeong Joon Park. This&that: Language-gesture controlled video generation for robot planning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12842–12849. IEEE, 2025.

[4] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

[5] Boyuan Chen, Tianyuan Zhang, Haoran Geng, Kiwhan Song, Caiyi Zhang, Peihao Li, William T. Freeman, Jitendra Malik, Pieter Abbeel, Russ Tedrake, Vincent Sitzmann, and Yilun Du. Large video planner enables generalizable robot control, 2025. URL https://arxiv.org/abs/2512.15840.

[6] Ryan Po, David Junhao Zhang, Amir Hertz, Gordon Wetzstein, Neal Wadhwa, and Nataniel Ruiz. Multigen: Level-design for editable multiplayer worlds in diffusion game engines. *arXiv preprint arXiv:2603.06679*, 2026. 1

[7] Lvmin Zhang, Shengqu Cai, Muyang Li, Gordon Wetzstein, and Maneesh Agrawala. Frame context packing and drift prevention in next-frame-prediction video diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. 1, 3, 6, 7, 16, 17

[8] Marcel Torne Villasevil, Andy Tang, Yuejiang Liu, and Chelsea Finn. Learning long-context diffusion policies via past-token prediction. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 1744–1755. PMLR, 2025. URL https://proceedings.mlr.press/v305/villasevil25a.html. 2

[9] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[10] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016. 2

[11] Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. FlexTok: Resampling images into 1D token sequences of flexible length. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 2241–2292. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/bachmann25a.html. 2

[12] Shivam Duggal, Phillip Isola, Antonio Torralba, and William T. Freeman. Adaptive length image tokenization via recurrent allocation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=mb2ryuZ3wz. 2

[13] Taiye Chen, Xun Hu, Zihan Ding, and Chi Jin. Learning world models for interactive video generation. *arXiv preprint arXiv:2505.21996*, 2025. 2

[14] Shengqu Cai, Ceyuan Yang, Lvmin Zhang, Yuwei Guo, Junfei Xiao, Ziyan Yang, Yinghao Xu, Zhenheng Yang, Alan Yuille, Leonidas Guibas, et al. Mixture of contexts for long video generation. *arXiv preprint arXiv:2508.21058*, 2025.

[15] Jiwen Yu, Jianhong Bai, Yiran Qin, Quande Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Xihui Liu. Context as memory: Scene-consistent interactive long video generation with memory retrieval. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–11, 2025.

[16] Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. Worldmem: Long-term consistent world simulation with memory. *arXiv preprint arXiv:2504.12369*, 2025. 15

[17] Zile Wang, Zexiang Liu, Jiaxing Li, Kaichen Huang, Baixin Xu, Fei Kang, Mengyin An, Peiyu Wang, Biao Jiang, Yichen Wei, Yidan Xietian, Jiangbo Pei, Liang Hu, Boyi Jiang, Hua Xue, Zidong Wang, Haofeng Sun, Wei Li, Wanli Ouyang, Xianglong He, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-game 3.0: Real-time and streaming interactive world model with long-horizon memory, 2026. URL https://arxiv.org/abs/2604.08995. 16

[18] Chonghyuk Song, Michal Stary, Boyuan Chen, George Kopanas, and Vincent Sitzmann. Generative view stitching. *arXiv preprint arXiv:2510.24718*, 2025. 2, 15

[19] Samuel Garcin, Thomas Walker, Steven McDonagh, Tim Pearce, Hakan Bilen, Tianyu He, Kaixin Wang, and Jiang Bian. Beyond pixel histories: World models with persistent 3d state. *arXiv preprint arXiv:2603.03482*, 2026. 2

[20] Junchao Huang, Xinting Hu, Boyao Han, Shaoshuai Shi, Zhuotao Tian, Tianyu He, and Li Jiang. Memory forcing: Spatio-temporal memory for consistent scene generation on minecraft. *arXiv preprint arXiv:2510.03198*, 2025. 2

[21] Tong Wu, Shuai Yang, Ryan Po, Yinghao Xu, Ziwei Liu, Dahua Lin, and Gordon Wetzstein. Video world models with long-term spatial memory. *arXiv preprint arXiv:2506.05284*, 2025. 2

[22] Xin Wen, Bingchen Zhao, Ismail Elezi, Jiankang Deng, and Xiaojuan Qi. " principal components" enable a new language of images. In *ICCV*, pages 16641–16651, 2025. 2

[23] Yiheng Liu, Liao Qu, Huichao Zhang, Xu Wang, Yi Jiang, Yiming Gao, Hu Ye, Xian Li, Shuai Wang, Daniel K Du, et al. Detailflow: 1d coarse-to-fine autoregressive image generation via next-detail prediction. *arXiv preprint arXiv:2505.21473*, 2025.

[24] Wilson Yan, Volodymyr Mnih, Aleksandra Faust, Matei Zaharia, Pieter Abbeel, and Hao Liu. Elastictok: Adaptive tokenization for image and video. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 38036–38056, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5e6cec2a9520708381fe520246018e8b-Paper-Conference.pdf. 2

[25] Mohammad Asim, Christopher Wewer, and Jan Eric Lenssen. Scenetok: A compressed, diffusable token space for 3d scenes. *CVPR*, 2026. 2

[26] Andrei Atanov, Jesse Allardice, Roman Bachmann, Oğuzhan Fatih Kar, R. Devon Hjelm, David Griffiths, Peter Fu, Afshin Dehghan, and Amir Zamir. Videoflextok: Flexible-length coarse-to-fine video tokenization. *arXiv preprint arXiv:2604.12887*, 2026. 2

[27] Tianwei Xiong, Jun Hao Liew, Zilong Huang, Zhijie Lin, Jiashi Feng, and Xihui Liu. Evatok: Adaptive length video tokenization for efficient visual autoregressive generation. *arXiv preprint arXiv:2603.12267*, 2026. 2

[28] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 30233–30249. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c32319f4868da7613d78af9993100e42-Paper-Conference.pdf. 2

[29] Shivam Duggal, Sanghyun Byun, William T. Freeman, Antonio Torralba, and Phillip Isola. Single-pass adaptive image tokenization for minimum program search. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=HkTOnCUQ1z. 2

[30] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020. 3

[31] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23 (47):1–33, 2022. 3

[32] Soumik Mukhopadhyay, Prateksha Udhayanan, and Abhinav Shrivastava. Scale space diffusion. *arXiv preprint arXiv:2603.08709*, 2026. 3

[33] Yuval Atzmon, Maciej Bala, Yogesh Balaji, Tiffany Cai, Yin Cui, Jiaojiao Fan, Yunhao Ge, Siddharth Gururani, Jacob Huffman, Ronald Isaac, et al. Edify image: High-quality image generation with pixel space laplacian diffusion models. *arXiv preprint arXiv:2411.07126*, 2024. 3

[34] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024. 3

[35] 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. 3

[36] Shiwei* Zhang, Jiayu* Wang, Yingya* Zhang, Kang Zhao, Hangjie Yuan, Zhiwu Qing, Xiang Wang, Deli Zhao, and Jingren Zhou. I2vgen-xl: High-quality image-to-video synthesis via cascaded diffusion models. *arXiv preprint arXiv:2311.04145*, 2023. 3

[37] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. *arXiv preprint arXiv:2410.05954*, 2024. 3

[38] Wilson Yan, Danijar Hafner, Stephen James, and Pieter Abbeel. Temporally consistent transformers for video generation, 2023. URL https://arxiv.org/abs/2210.02396. 3, 6

[39] Yuchao Gu, Weijia Mao, and Mike Zheng Shou. Long-context autoregressive video modeling with next-frame prediction. *arXiv preprint arXiv:2503.19325*, 2025. 3

[40] William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27953–27965. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b2fe1ee8d936ac08dd26f2ff58986c8f-Paper-Conference.pdf. 3

[41] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep networks as a perceptual metric. In *CVPR*, 2018. 4, 6

[42] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 6

[43] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 37, 2018. URL https://arxiv.org/abs/1805.09817. 6, 19, 26

[44] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021. 6

[45] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22160–22169, 2024. 6

[46] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 6

[47] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. 6

[48] William H Guss, Cayden Codel, Katja Hofmann, Brandon Houghton, Noboru Kuno, Stephanie Milani, Sharada Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, et al. The minerl competition on sample efficient reinforcement learning using human priors. *arXiv preprint arXiv:1904.10079*, 2, 2019. 6

[49] Jing Wang, Fengzhuo Zhang, Xiaoli Li, Vincent Y. F. Tan, Tianyu Pang, Chao Du, Aixin Sun, and Zhuoran Yang. Error analyses of auto-regressive video diffusion models: A unified framework, 2025. URL https://arxiv.org/abs/2503.10704. 6

[50] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6

[51] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 6

[52] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In *ICCV*, 2023. 6

[53] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 7

[54] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf. 7

[55] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Fvd: A new metric for video generation. 2019. 7

[56] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314. 10

[57] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024. 10

[58] Ryan Po, Yotam Nitzan, Richard Zhang, Berlin Chen, Tri Dao, Eli Shechtman, Gordon Wetzstein, and Xun Huang. Long-context state-space video world models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8733–8744, October 2025. 10

[59] Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 2024. 10, 16

[60] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703. 15

[61] 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. 15

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762. 15, 16

[63] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021. URL https://arxiv.org/abs/2111.06377. 15, 16

[64] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 15

[65] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/. 15, 16

[66] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. URL http://arxiv.org/abs/1711.05101. 15, 16

[67] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015. URL http://arxiv.org/abs/1503.03585. 15

[68] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 15

[69] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *CoRR*, abs/2202.00512, 2022. URL https://arxiv.org/abs/2202.00512. 15

[70] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pages 13213–13232. PMLR, 2023. 15

[71] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598. 15

[72] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 16, 17

[73] Haoxin Chen, Menghan Xia, Yingqing He, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Jinbo Xing, Yaofang Liu, Qifeng Chen, Xintao Wang, et al. Videocrafter1: Open diffusion models for high-quality video generation. *arXiv preprint arXiv:2310.19512*, 2023. 16

[74] Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-guided video diffusion. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=j8Vr3E3vhy. 19, 20

A Appendix

A.1 Test Set Generation

Our test set consists of 1,000 videos whose trajectories feature high overlap between the context (i.e., the first 256 frames) and the remaining 768 frames. We use a near-identical script to generate the training and test sets, but restrict the test set's trajectories to ones that are likely to feature high overlap. Specifically, for each test video we generate, we first sample 100 complete action sequences. We then compute these sequences' expected top-down trajectories (assuming movement on a flat plane where there are no collisions with trees, caves, hillsides, etc.). For each of these expected trajectories, we compute the Chamfer distance between the context's XY points and the remaining XY points. We then sample uniformly from the 10 sequences with the lowest Chamfer distances to select a single action sequence that is ultimately rendered. Using this procedure, we render 10,000 test set candidates. We then choose the 1,000 videos with the highest distance covered by the agent. This ensures that our test set videos are unlikely to contain trajectories where the agent becomes "stuck" by falling into a cave or repeatedly running into a tree. We observe that this procedure does a reasonable job of producing interesting videos with good context overlap.

We considered using camera frustum overlap [16, 18] in order to measure context overlap, but ultimately decided against doing so because we found that frustum overlap behaves counterintuitively for trajectories (like ours) that feature many loops. When there are many loops, the context's viewing frustums tend to cover almost every angle, making frustum overlap a meaningless metric.

More broadly, we argue that while it is desirable to ensure good overlap between the context frames and the remaining frames, failing to achieve perfect overlap is not catastrophic. In cases where the remaining frames have little visual overlap with the context, even a near-perfect generative model can do no better than producing plausible hallucinations. Consequently, the resulting metrics will primarily reflect generation quality rather than consistency (i.e., in the best case, the metrics will be similar to those computed with respect to random ground-truth frames). However, because every method is evaluated on the same test set, this effect applies roughly equally across methods: it compresses the absolute values of consistency metrics, effectively making them appear worse, but preserves their relative ordering. The absolute consistency numbers are therefore harder to interpret in isolation, but cross-method comparisons remain meaningful. Thus, although we try our best to ensure good overlap, it is not strictly necessary to do so.

A.2 Implementation and Training Details

Our models are implemented in PyTorch [60] and use bfloat16 automatic mixed precision (AMP).

Hierarchical Autoencoder Training Setup Our hierarchical autoencoder processes each frame in a video independently. It has a ViT-B [61] transformer architecture for both its encoder and decoder. Each input token to both the transformer encoder and decoder gets height and width 2D sin/cos positional embeddings [62, 63]. Each token also has a learned position embedding corresponding to its level in the coarse-to-fine hierarchy. In all our experiments, we set $L = 4$. The decoder is further conditioned on the level embedding using AdaLN [64].

Our model operates on images with maximum resolution $(256, 256)$ with a patch size of $(16, 16)$. Each output token has 32 channels. Our hierarchical autoencoder is trained using the Muon [65] optimizer for linear layers and AdamW [66] for the remaining parameters. We use a learning rate of $1e{-}3$, weight decay of 0.01, and gradient norm clipping at 1.0, with a linear warmup over $2{,}000$ steps. The model is trained for $128{,}000$ steps at a per-device batch size of 224 for ~ 1 day on 8 H200 GPUs, with an effective batch size of $224 * 8 = 1792$.

Generative Model Training Setup We instantiate our generative model as a diffusion model [67, 68] with a velocity-field parameterization (v-prediction) [69]. For the noising schedule, we adopt the cosine schedule from Simple Diffusion [70], and we empirically find that a shifted variant with shift = 1.0 performs well across all the prediction and upsampling tasks (our schedule is level-independent). At training time, we randomly drop out all context tokens for a batch element with probability 0.1 to experiment with classifier-free guidance (CFG) [71] at inference time.

The diffusion model is parameterized using the DiT-B [64] architecture, and is trained jointly on latents from all four levels produced by the hierarchical autoencoder. In the transformer, every token

gets a 3D sin/cos position embedding [62, 63] describing its frame, height and width indices. Each token also gets a learned level embedding. Conditioning on categorical actions (left, right, forward) is provided through a simple learned embedding layer. Both the level embedding and the diffusion timestep (which is also encoded via sin/cos positional embedding) are passed to the transformer via AdaLN conditioning.

All generative models are trained using the Muon [65] optimizer for linear layers and AdamW [66] for the remaining parameters. We use a learning rate of $1e{-3}$, weight decay of 0.01, and gradient norm clipping at 1.0, with a linear warmup over $2{,}000$ steps. Each model is trained with a per-device batch size of 48 on 8 H200 GPUs, with an effective batch size of $48 * 8 = 384$.

Inference Details At sampling time we use deterministic DDIM sampling [72] with 50 denoising steps. We use the same shifted schedule as at training time to decide where to evaluate the 50 denoising steps. We do not apply CFG to the samples used for evaluation, although we find that sampling quality can be marginally improved by using CFG with guidance scale $\gamma = 1.5$.

Examples in our test set contain 1024 total frames. We always start evaluating the models at a fixed frame (frame 256), with each model having access to a different number of maximum context frames before 256, depending on the particular choices for how each model allocates its context budget. We then keep rolling out for inference until frame 1024 is generated, appending additional actions where needed. All comparison metrics are run for the same indices of generated frames > 256, comparing to the ground truth.

Model-Specific Details Each model is trained with a maximum sequence length S. We list the model-specific implementation details below. We choose to keep S as close as possible between models to ensure fair comparison. However, due to details in the nature of the inference strategy, there will be a small discrepancy in the exact values of S for each model.

For the models reported in Figure 4 and Table 1:

Our model (MILLIVID): The model is trained with $S = 3840$. This S is split between the 3072 tokens reserved for context (which include enough budget for $[3, 12, 48, 192]$ frames at levels 0-3 respectively) and 768 tokens reserved for prediction in any given step. At inference time, the model sees a maximum of $3 + 12 + 48 + 192 = 255$ frames into the past. Note that in all of our experiments, each coarser level contains four times as many frames as the preceding level, since this ensures that each level contains the same number of tokens (and, therefore, that all rollout configurations can be stacked without extra padding; see Figure 10).

FramePack [7]: The model is trained with $S = 3840$. Like for MILLIVID, 3072 of these tokens are reserved for context budget (enough budget for $[3, 12, 48, 192]$ context frames at levels 0-3 respectively). The model predicts 3 frames into the future at the finest level, which accounts for the remaining $3 * 256 = 768$ tokens of the sequence length. At inference time, the model sees a maximum of $3 + 12 + 48 + 192 = 255$ frames into the past.

Full-Resolution Autoregressive Rollout: The model is trained with $S = 3584$, which is enough for 14 frames at our finest token level. Following prior works [17, 59, 73], we allocate half the budget to context frames and half the budget to predicted frames. At inference time, the model sees a maximum of 7 frames into the past.

Our main models are trained for $192{,}000$ steps, which takes ~ 2 days on 8 H200 GPUs. For ablations and comparisons reported in Figure 6 and Figure 7, we opted to train smaller models with a third of the sequence length budget S. These smaller models are trained for $256{,}000$ steps on 8 H200 GPUs with a batch size of 48 per device (for an effective batch size of $8 * 48 = 384$). This takes ~ 1 day per model.

A.3 Ablating the number of levels in MILLIVID

MILLIVID uses a model with 4 levels in its rollout strategy. We ablate this choice with fewer levels in our coarse-to-fine rollout. In Table 2, we include details for how these ablations differ from the default $L = 4$ case in parameters. Beyond these parameters, the models are trained identically. We evaluate the models on LPIPS and FVD. Results are presented in Table 3.

We find that the L = 3 model trades off medium- and long-range consistency for slightly improved short-range consistency, while performing similarly on FVD. The L = 2 model performs worse than

Table 2: Configuration of the level ablation models.

Parameter	$L = 2$	$L = 3$	$L = 4$
Sequence length	3840	4096	3840
Number of level-0 context frames	5	4	3
Number of level-1 context frames	20	16	12
Number of level-2 context frames	0	64	48
Number of level-3 context frames	0	0	192
Number of denoised tokens	1280	1024	768

Table 3: Level Ablation Results

Method	LPIPS ↓ short	LPIPS ↓ med.	LPIPS ↓ long	FVD ↓ short	FVD ↓ med.	FVD ↓ long
4 Levels	0.159	**0.247**	**0.335**	43.8	**57.8**	83.7
3 Levels	**0.147**	0.260	0.381	**38.6**	58.0	**77.5**
2 Levels	0.238	0.464	0.534	65.8	132.9	176.4

both, likely because it lacks the regularization that an initial long, coarse rollout provides. The L = 4 method included as the main method in the paper represents the model with the best overall performance.

A.4 Sampling Speed

At inference time, the total number of sampling steps differs between the different methods. We analyze the number of function evaluations needed to run each method and provide the wall clock time needed to run batched inference for 8 samples with 50 DDIM [72] steps per rollout step.

Full-resolution autoregressive rollout looks at context and predicts tokens only at the highest resolution. For some fixed constant k_b, which depends on context and prediction length, it takes $F/k_b = O(F)$ steps to generate F frames. For the sampling specification described above, it takes approximately 11 minutes to sample a batch of 8 videos (of 768 frames each).

FramePack [7] also looks at context and predicts a small number of frames at the finest resolution. As such, for some fixed constant k_p which depends on the context and prediction length, it too takes $F/k_p = O(F)$ steps. For the sampling specification described above, it takes about 30 minutes to sample a batch of 8 videos (of 768 frames each).

Our method predicts at the coarsest resolution first, and then follows an upsampling strategy level-by-level to predict finer tokens. As a result, it must not only roll out for $O(F)$ steps at the finest level, but also roll out at the coarser levels. Fortunately, the number of rollout steps taken at the coarser levels decreases exponentially with every level. Thus, for our model, for some constant k_m, it takes $1/k_m(F + F/4 + F/16 + F/64 + \ldots) = O(F)$ steps to generate F frames. For the sampling specification described above, it takes about 30 minutes to sample a batch of 8 videos (of 768 frames each). Note that our model requires about 33% more sampling steps than the equivalent FramePack model, so an optimized FramePack model should require about 25% less time than ours.

A.5 Adapting MILLIVID's Latent Space to Custom Properties

Remembering fine details from the distant past is extremely challenging, and MILLIVID is designed around the assumption that doing so is generally not needed. MILLIVID is able to be more consistent with the far past because frames from the far past are still represented in the model's memory with a smaller number of latent tokens. Consequently, if a feature of the data is not represented in a coarse token (such as a small object or a fine texture), it will not be kept in memory and MILLIVID's video model will be inconsistent with respect to it.

In certain applications (such as with logos, characters, or text), it may be known *a priori* what the most important features of data are that the video model should be consistent with respect to. In these cases, it would be desirable if consistency with respect to a specific feature could be explicitly enforced.

In such cases, MILLIVID could be augmented by shaping the latent space of the hierarchical autoencoder to explicitly decide what features are captured in the coarse latents. By default, MILLIVID 's autoencoder uses MSE + LPIPS losses, where a particular detail's importance is roughly proportional to its size: large details are assumed to be important, while small ones are assumed to be unimportant. A directed reweighting in this loss to prioritize the reconstruction of specific small objects would incentivize the model to better represent these objects inside the coarsest latents.

To show that this is possible, we fine-tune MILLIVID 's autoencoder to specifically remember flowers, which act as a stand-in for small but important details we want to remember over long time horizons. MILLIVID 's original autoencoder mostly ignores flowers in its most compressed hierarchy level because of their small size. As a result, MILLIVID 's video model cannot remember most flowers over long ranges. To make the autoencoder encode information about flowers into the most compressed hierarchy level, we simply up-weight the MSE loss for pixels that correspond to flowers and their immediate outlines. We do this by identifying flowers with a simple segmentation mask and upweighting their reconstruction error.

Segmentation details Flower pixels are segmented by color thresholding in HSV space. Three different Minecraft flower types are detected separately: red poppies, yellow dandelions and cyan blue orchids (we optimize HSV values to identify the flowers reliably in our dataset). Each per-color mask is dilated by 2 pixels so that petal outlines are included. The union of the segmentation masks corresponding to these flowers is the overall segmentation mask of the image.

Autoencoder fine-tuning We adopt MILLIVID's autoencoder and fine-tune it for 40,000 steps with an additional "flower" loss. The flower segmentation mask is patchified exactly like the image, so that at coarser levels each patch pixel holds its fractional flower coverage. Considering fractional flower coverage $m_i \in [0, 1]$, squared reconstruction error e_i^2 at pixel-channel i and compression level ℓ_i, the additional loss is

$$\mathcal{L}_{\text{flower}} = \frac{\sum_i e_i^2 \, m_i \, 4^{\ell_i}}{3 \sum_i m_i \, 4^{\ell_i}} \tag{1}$$

Because $\mathcal{L}_{\text{flower}}$ is normalized by flower area rather than image area, it contributes to the total loss with the same overall weight as the full-image MSE: flowers as a class receive as much gradient as the entire image, and this incentivizes the coarse latents to preserve them in their reconstructions.

We test if fine-tuning the autoencoder was successful by measuring the mean intersection over union (mIoU) of flowers across reconstructions from all compression levels. The fine-tuned model improves mIoU on reconstruction at all levels. Our results are shown in Table 4.

Diffusion fine-tuning and evaluation We finetune our 1280-token model (ablation scale) for 14,000 steps. The evaluation protocol and training details are kept exactly the same as for the base model. We evaluate the finetuned model against the base model to test recall of flowers explicitly, incorporating a variety of metrics to ensure that flowers are actually being recalled better and not just hallucinated.

Flower IoU: This metric is the per-frame intersection over union between generated and ground-truth flower masks, averaged over frames whose ground truth contains flowers. This is highly unforgiving, since a shift of even a few pixels sends this metric to zero due to flowers' small size.

Instance Recall (@8px): This is an instance level metric that captures "what percent of ground-truth flowers still appear." First, ground-truth flower pixels are grouped into connected instances. A test frame is considered to have remembered an instance if the generated frame contains ≥ 1 flower pixel within 8 pixels of it. The final number indicates what percent of ground-truth instances are recalled in this manner. This is a coarser "perceptual" metric capturing if the flower is remembered.

Pixel Recall (@8px): This measures the fraction of ground-truth flower pixels covered by the generated flower mask after dilating it by 8 pixels. We report this as the more appropriate metric for accommodating small pixel shift in a video model. This supplements Instance Recall with "are the flowers roughly the right size?".

Pixel Precision (@8px): This is the fraction of generated flower pixels that lie within 8 pixels of a same color ground-truth pixel. We include this to confirm that the fine-tuned model does not hallucinate too many flowers and therefore have erroneously high recall.

Our results are presented in Table 5. Fine-tuning improves on all metrics, but more so on the coarser metrics (corroborating evidence in the rest of the paper). Further, we also qualitatively observe that

Table 4: mIoU for each latent level of the original and fine-tuned autoencoder, from finest (level 0) to coarsest (level 3).

Latent level	Original	Fine-tuned
Level 0 (finest)	0.43	**0.66**
Level 1	0.35	**0.62**
Level 2	0.25	**0.54**
Level 3 (coarsest)	0.13	**0.35**

Table 5: Flower preservation metrics averaged over short (frames 1-64), medium (frames 65-256), and long horizons (frames 257-768), computed on 1000 generated videos from the held-out test set.

Method	Instance Rec. @8px ↑			Pixel Rec. @8px ↑			Pixel Prec. @8px ↑			Flower IoU ↑		
	short	med.	long	short	med.	long	short	med.	long	short	med.	long
Ours	**0.466**	**0.324**	**0.172**	**0.585**	**0.313**	**0.138**	**0.780**	**0.408**	**0.165**	**0.1051**	**0.0324**	**0.0092**
Baseline	0.228	0.078	0.015	0.405	0.111	0.013	0.624	0.228	0.048	0.0513	0.0070	0.0007

the model improves on recalling flowers at the right locations; we visualize some examples in Figure 12.

This experiment is presented as a "proof-of-concept" for adapting MILLIVID's latent space to specific applications where it may be known which details are important to remember.

A.6 Validating MILLIVID on Real Data

As outlined in Section 4's paragraph on datasets, existing real-world datasets are poorly suited for measuring long-range consistency because they lack at least one of the following: (1) enough frames per video, (2) enough videos in the dataset, (3) content that exits and re-enters the frame, or (4) fine-grained conditioning signal that can steer the model towards previously seen content. However, to provide evidence of MILLIVID's applicability to real-world data, we run additional experiments on the RealEstate10k [43] dataset.

RealEstate10k has only 10000 clips with short videos of slow moving cameras which rarely return to previously seen content. Further, past methods that train on RealEstate10k have required frame skips of 10 or more [74] to train successful video models. This means that further downsampling of already short video clips is needed for the dataset to be usable. These factors mean that this dataset is not well suited to quantitative evaluation of consistency and stability.

As a result, our goal with this experiment is solely to make sure that our training pipeline can be replicated on the real world and that the method is compatible with generating videos with the qualitative properties of real video.

Autoencoder We train MILLIVID's coarse-to-fine autoencoder on frames from RealEstate10k at 256x256 resolution, with a maximum latent resolution of 32x32 (8x8 patches). Results from the autoencoder can be seen in Figure 13. We find that the autoencoder's finest level produces detailed, high-fidelity reconstructions, and that the coarser levels contain sharp object boundaries while blurring out interior textures. For instance, boundaries for windows, doors, and table/chair legs are sharp, even at the coarsest level, while finer details (exact carpet patterns, countertop textures, fine branch structure in trees, etc.) are reconstructed in finer levels.

Video model Next, we train a camera-conditioned MILLIVID model in the latent space of the autoencoder. After accounting for frame skip, we are left with a dataset of frame sequences long enough to train a two-level MILLIVID model (1 frame of fine context and 4 frames of coarse context). We find that the model is capable of producing sharp videos. We plot random eval set videos sampled from our model in Figure 14. We also include results from a full-resolution rollout baseline which looks at 2 frames of context and generates 2 frames at once.

Training and Inference details Both our model and the baseline are size-B transformers on normalized latents from the autoencoder. The video model is a diffusion model trained with a cosine noise schedule (using a noise-schedule shift of 0.3), conditioning dropout with probability 0.1 and an EMA of the weights (decay 0.9999). Both models are trained with Muon/AdamW at learning rate 10^{-3}

with a batch size 288 on 8 H200 GPUs. During training, clips are subsampled with a random frame gap drawn uniformly from $[10, c]$, where the cap c ramps from 10 to the max possible for the given model over the first 50,000 steps (matching the skip curriculum of history-guided video diffusion [74]). The MILLIVID model is trained to 144,000 steps (about 1.5 days) while the full-resolution baseline is trained longer to 206,000 steps (about 2 days) on 8xH200.

At evaluation time, models receive up to 8 ground-truth context frames at a fixed test-time frame skip of 10 and autoregressively predict the next 16 frames. Inference uses 50 diffusion sampling steps and guidance scale 1.25, with EMA weights.

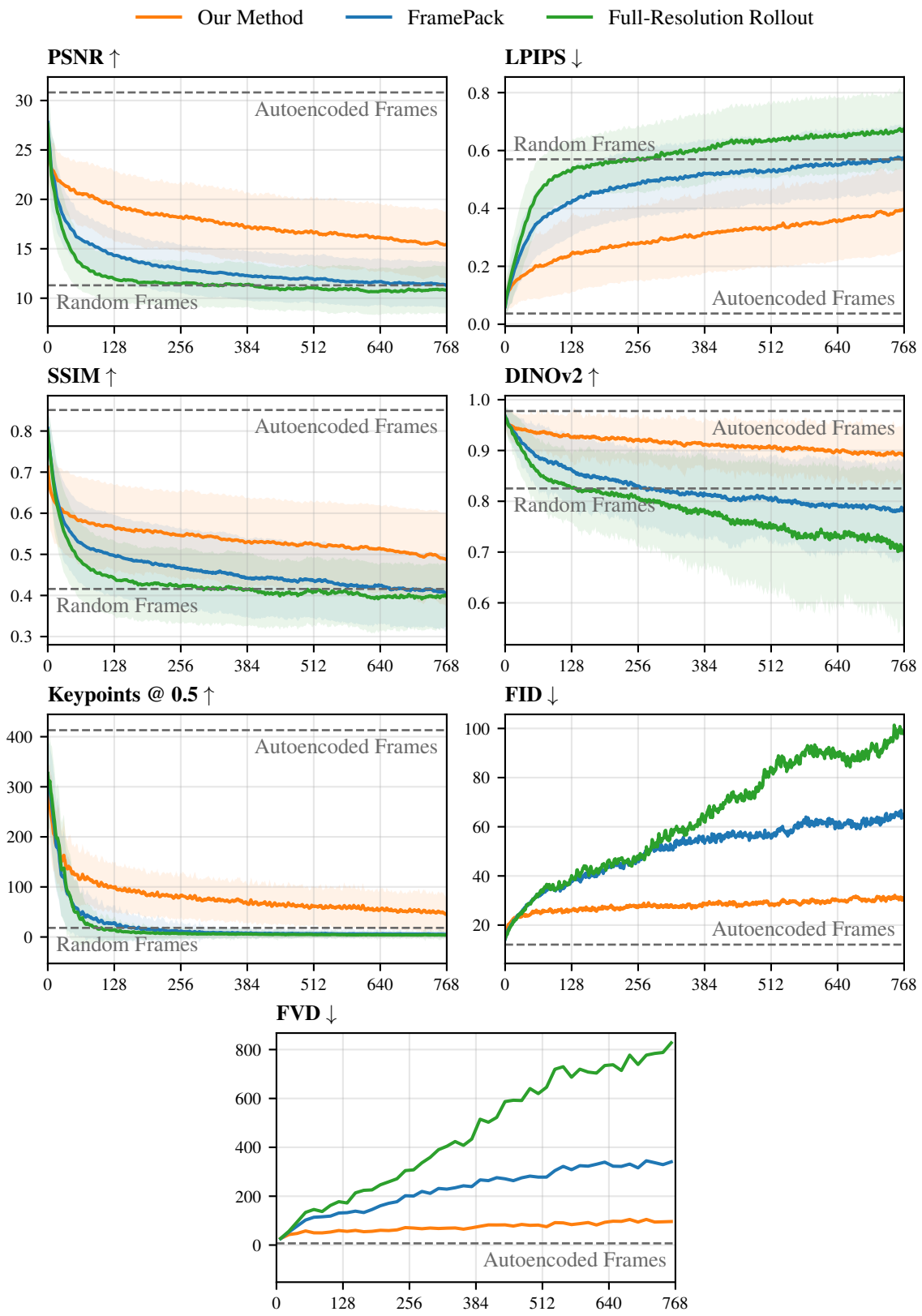


Figure 8: **Full metrics:** All metrics plotted for our method and the baselines. Our method not only produces better consistency (as measured by PSNR, LPIPS, SSIM, DINOv2 class token cosine similarity, and the number of keypoints with ≥ 0.5 confidence detected by LightGlue), but also produces significantly better per-frame quality as measured by FID and FVD, with less exposure bias.

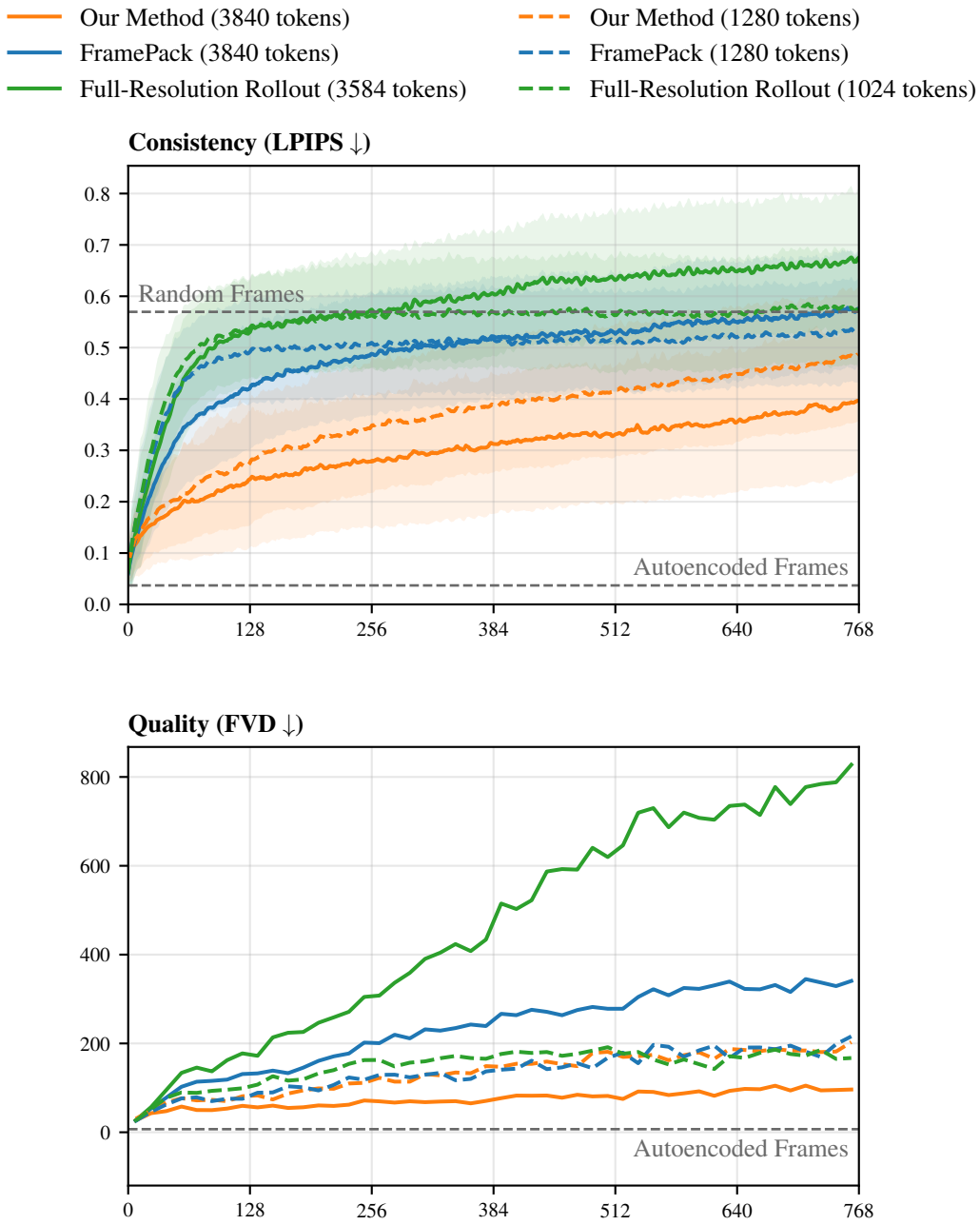


Figure 9: **Scaling properties:** A comparison of our method and the baselines at two distinct scales. Given 3840 tokens, both our method and FramePack hold 255 frames of context. Meanwhile, given 3584 tokens, the autoregressive high-resolution rollout model holds 7 frames of context. Given 1280 tokens, both our method and FramePack hold 85 frames of context; the high-resolution rollout model holds 2 frames of context. The models with longer sequence length were trained for 192,000 steps, while the ones with shorter sequence length were trained for 256,000 steps. Longer sequence length improves consistency for both our model and FramePack. However, it causes the high-resolution rollout model to become unstable, and so its consistency falls below the consistency of random ground-truth frames. With a lower sequence length, all models have roughly the same per-frame quality. However, increasing sequence length appears to increase exposure bias (as indicated by increasing FVD) for the baselines while *reducing* it for our model. This suggests that our model may have more favorable scaling properties than the baselines.

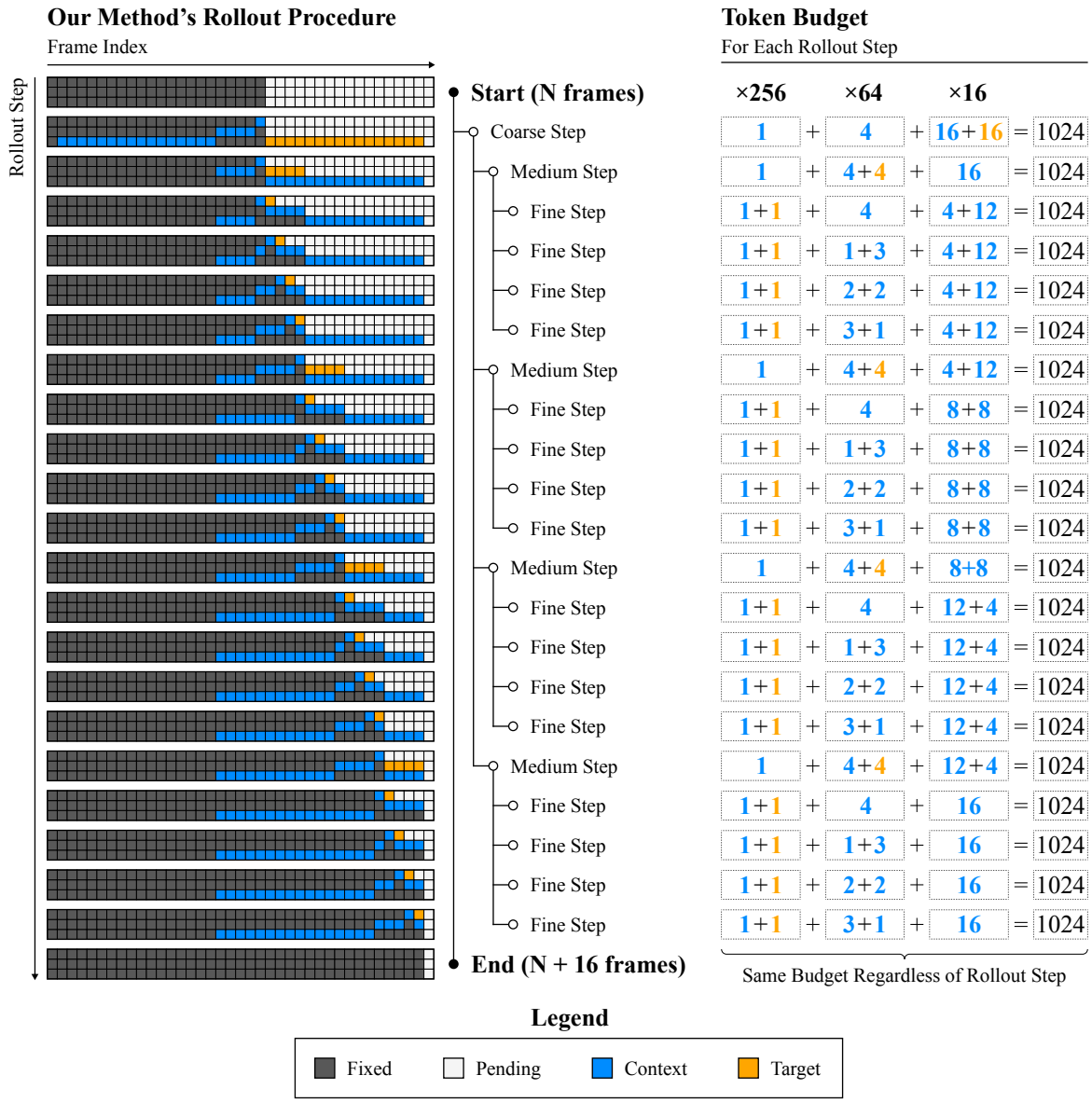


Figure 10: **Our model's rollout procedure:** On the left, see our model's full rollout sequence in the case of 3 hierarchy levels. We show the initial state, containing 22 frames; the final state, containing 38 frames; and 21 rollout steps, each of which is represented by a 3-row grid. During each rollout step, the model sees a mixture of **context** and **target** tokens across different hierarchy levels; it does not see the **fixed** (already generated) or pending (not yet generated) tokens. In each step, the rows represent hierarchy levels—fine (top), medium (middle), and coarse (bottom)—with 256, 64, and 16 tokens per frame respectively. Depending on the step, the model generates long sequences of coarse frames, medium-length sequences of medium-compression frames, or short sequences of fine frames. The rollout procedure is repeated to generate longer videos. Right side: During each rollout step, the transformer sees the exact same number of tokens. Note that while this figure shows the case of 3 levels and 256 tokens at the finest level, our approach scales to arbitrary numbers of levels and different per-image token counts. Given an increased sequence length budget, one can extend the context length by either (1) multiplying the number of frames at each level by some integer value or (2) increasing the number of levels. For a more intuitive view of the algorithm, we highly encourage readers to watch the animated version of the rollout sequence on our project website.

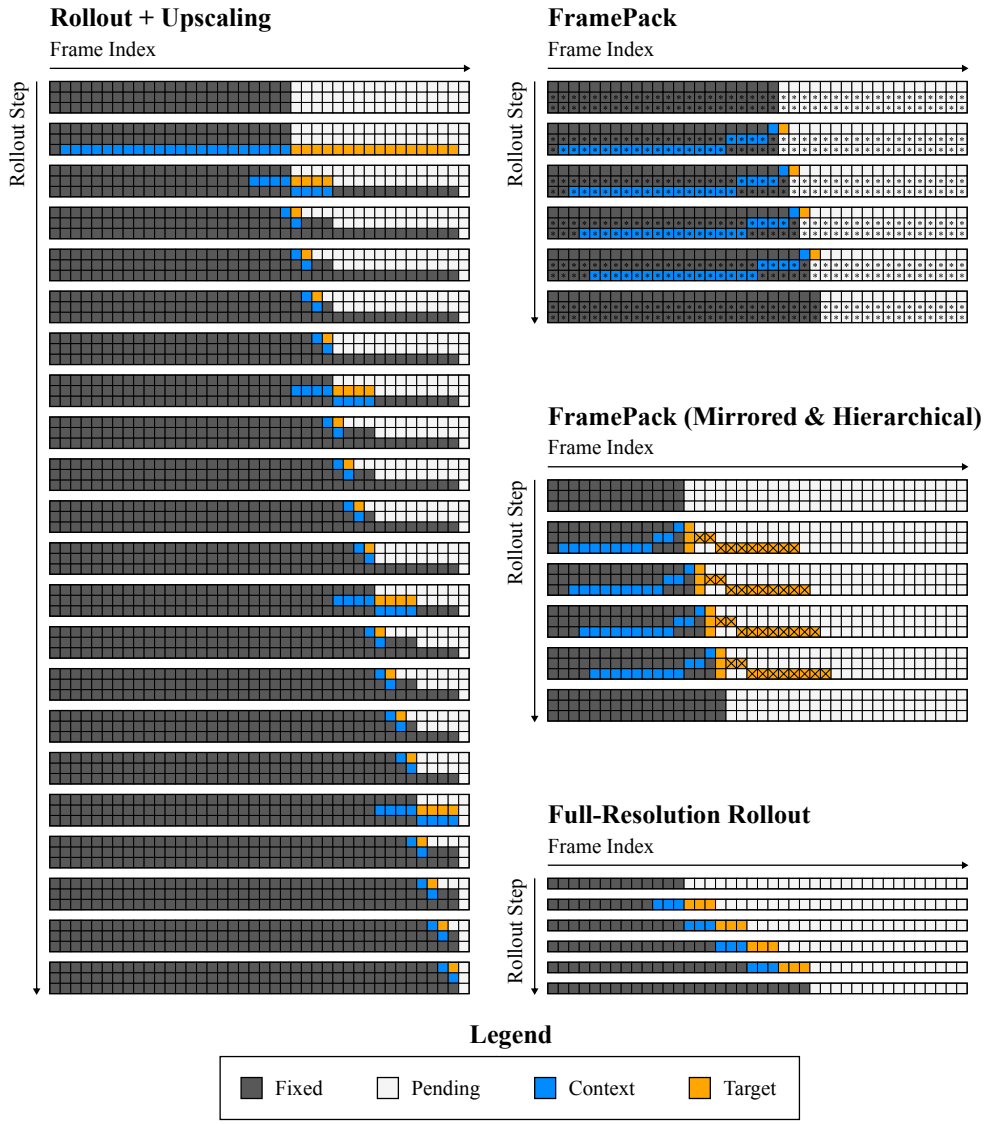


Figure 11: **Other methods:** Visualizations of the rollout procedures for the other methods mentioned in the paper. The rollout + upscaling model rolls out at the coarsest level, then combines rollout and upscaling for all subsequent levels. FramePack uses varying patchification, denoted via asterisks, to control the number of tokens per frame. It always predicts frames at their full resolution. FramePack (Mirrored & Hierarchical) is a variant of FramePack that uses hierarchical latents. It is trained to predict coarse frames far into the future. At sampling time, the coarse predictions are discarded, as indicated by the crosses drawn through them. In full-resolution rollout, there is only a single latent resolution; frames are rolled out at this resolution.

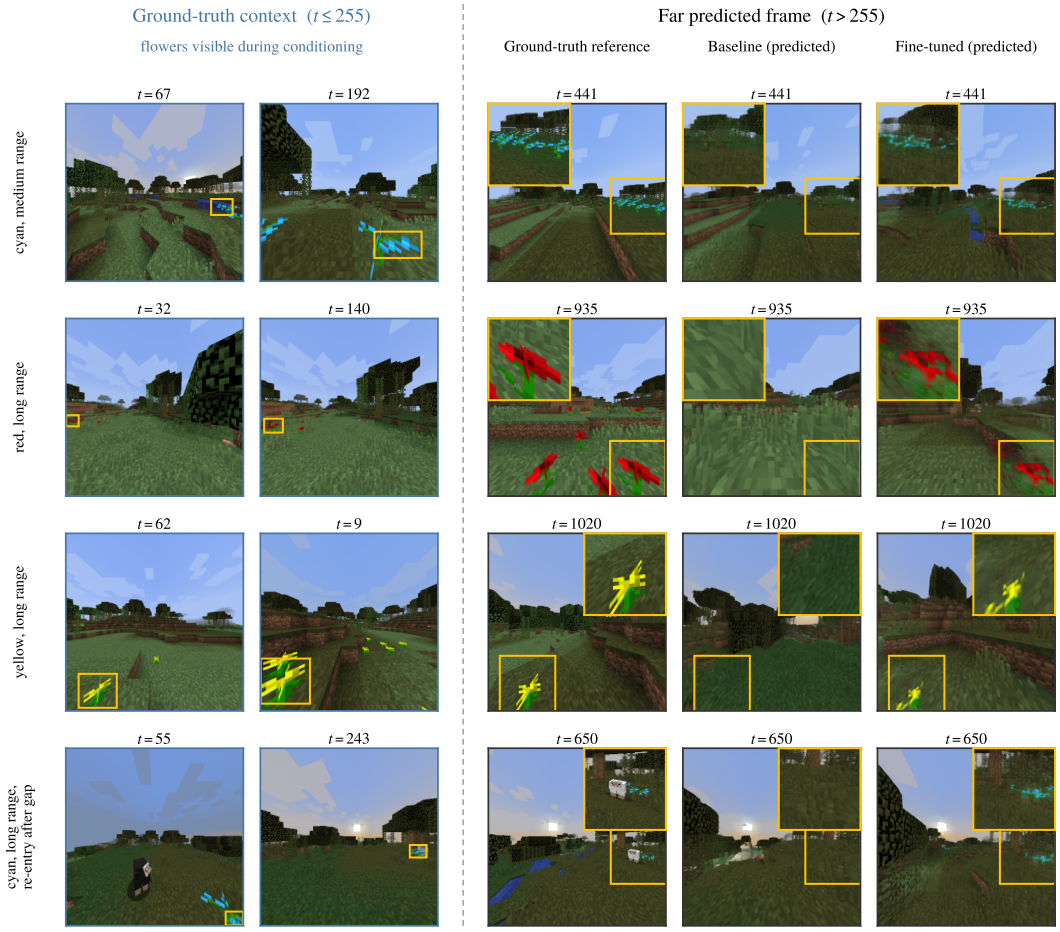


Figure 12: **Qualitative flower persistence:** We evaluate the fine-tuned model on the same held-out set as the base model. We visualize examples where we know there is an overlap between the context and predicted regions to visualize recall of flowers. The base model recalls the coarse scene structure well but forgets about the existence of flowers. The fine-tuned model shows increased flower memory, the task for which the latent space was reshaped. In this way, we show that MILLIVID can be adapted to remember specific coarse details when they are selected *a priori*.

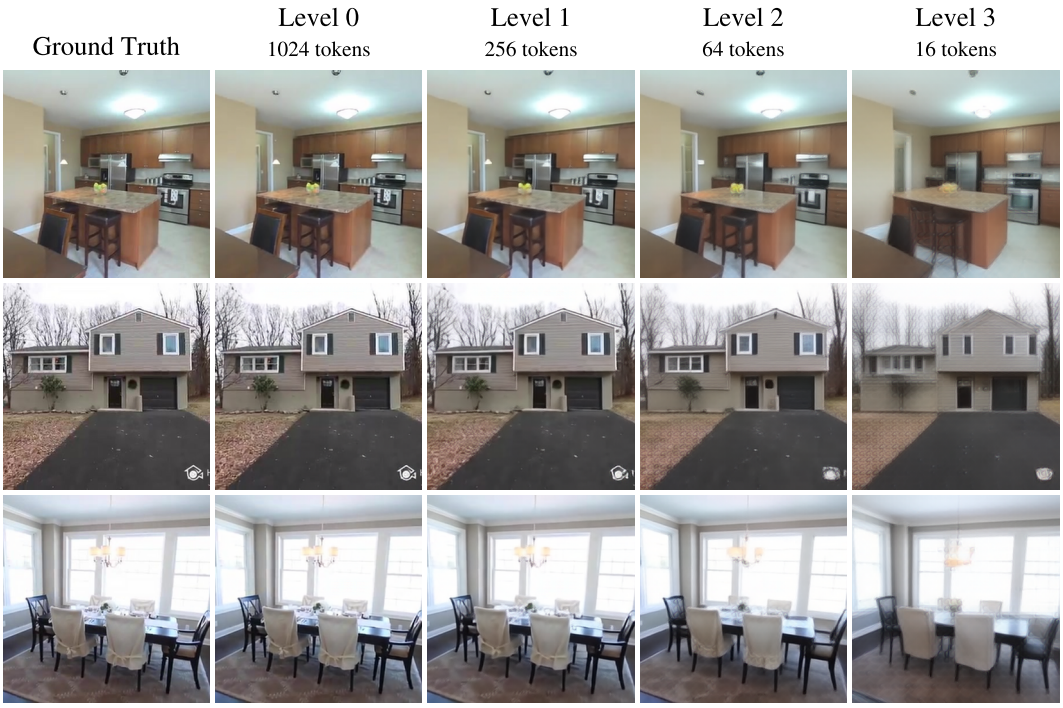


Figure 13: **RealEstate10k [43] Autoencoder:** We train our autoencoder from scratch on the RealEstate10k dataset. Coarse tokens capture the overall scene layout and object identity, and finer tokens refine textures and smaller objects. This corroborates our observations on LOOPCRAFT.

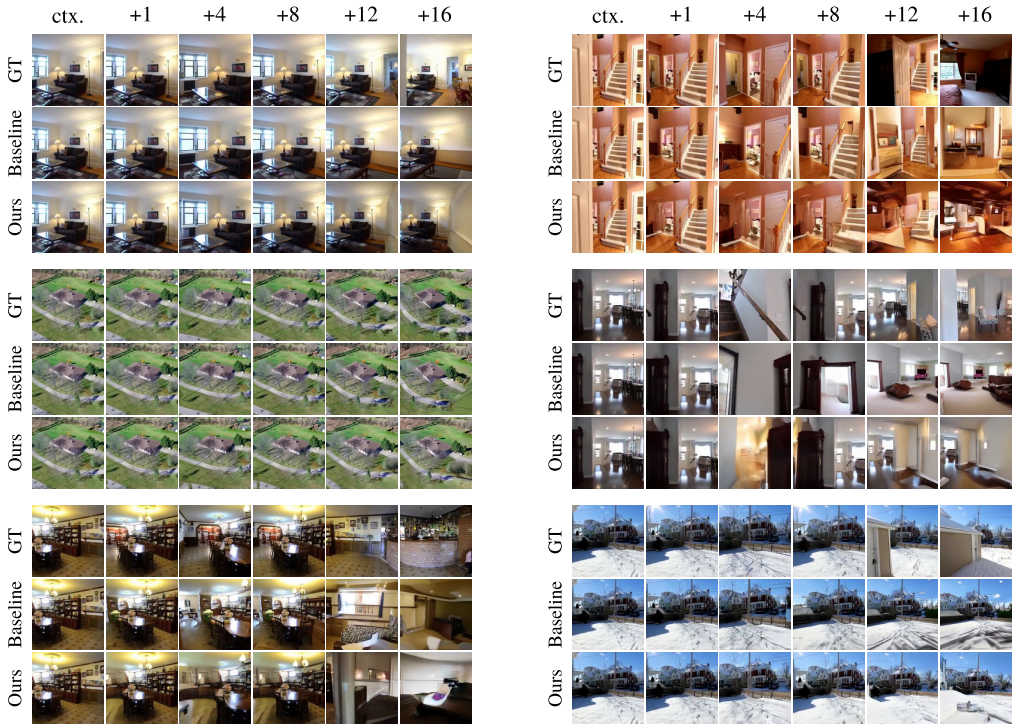


Figure 14: **RealEstate10k [43] Samples:** We present random evaluated samples from a 2-level MILLIVID model trained on RealEstate10k, showing that it is capable of producing sharp video. We show the ground truth and a full-resolution rollout baseline for reference.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims are specifically enumerated at the end of the introduction and supported by the main body of the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed after the conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: We do not present theorems or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, and we plan to release a PyTorch implementation plus pre-trained checkpoints.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We plan to release the code once it has been cleaned up slightly. We also plan to release our dataset generation code and dataset.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The appendix includes the hyperparameters, optimizer, etc., plus information about the test set.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We report standard deviations in our plots (via the shaded regions), but do not compute statistical significance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe the compute resources user to train our models in the appendix, plus sampling runtimes.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: This paper conforms to the NeurIPS Code of Ethics and does not involve sensitive data/topics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper briefly states that long-range video generation may have important applications in related research areas, e.g., robotics, but does not discuss negative societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our data and pre-trained models pose no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly credit our data source via citation.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We use a modified version of an existing data generation pipeline. We plan to release it alongside the code.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.