

# Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets

Andreas Blattmann\* Tim Dockhorn\* Sumith Kulal\* Daniel Mendelevitch  
 Maciej Kilian Dominik Lorenz Yam Levi Zion English Vikram Voleti  
 Adam Letts Varun Jampani Robin Rombach  
 Stability AI



*"A robot dj is playing the turntables, in heavy raining futuristic tokyo, rooftop, sci-fi, fantasy"*



*"An exploding cheese house"*



*"A fat rabbit wearing a purple robe walking through a fantasy landscape"*



Figure 1. **Stable Video Diffusion samples.** Top: Text-to-Video generation. Middle: (Text-to-)Image-to-Video generation. Bottom: Multi-view synthesis via Image-to-Video finetuning.

## Abstract

We present Stable Video Diffusion — a latent video diffusion model for high-resolution, state-of-the-art text-to-video and image-to-video generation. Recently, latent diffusion models trained for 2D image synthesis have been turned into generative video models by inserting temporal layers and finetuning them on small, high-quality video datasets. However, training methods in the literature vary widely, and the field has yet to agree on a unified strategy for cu-

rating video data. In this paper, we identify and evaluate three different stages for successful training of video LDMs: text-to-image pretraining, video pretraining, and high-quality video finetuning. Furthermore, we demonstrate the necessity of a well-curated pretraining dataset for generating high-quality videos and present a systematic curation process to train a strong base model, including captioning and filtering strategies. We then explore the impact of finetuning our base model on high-quality data and train a text-to-video model that is competitive with closed-source video generation. We also show that our base

\* Equal contributions.

*model provides a powerful motion representation for downstream tasks such as image-to-video generation and adaptability to camera motion-specific LoRA modules. Finally, we demonstrate that our model provides a strong multi-view 3D-prior and can serve as a base to finetune a multi-view diffusion model that jointly generates multiple views of objects in a feedforward fashion, outperforming image-based methods at a fraction of their compute budget. We release code and model weights at <https://github.com/Stability-AI/generative-models>.*

## 1. Introduction

Driven by advances in generative image modeling with diffusion models [38, 68, 71, 76], there has been significant recent progress on generative video models both in research [9, 42, 82, 95] and real-world applications [54, 74]. Broadly, these models are either trained from scratch [41] or finetuned (partially or fully) from pretrained image models with additional temporal layers inserted [9, 32, 43, 82]. Training is often carried out on a mix of image and video datasets [41].

While research around improvements in video modeling has primarily focused on the exact arrangement of the spatial and temporal layers [9, 41, 43, 82], none of the aforementioned works investigate the influence of data selection. This is surprising, especially since the significant impact of the training data distribution on generative models is undisputed [13, 105]. Moreover, for generative image modeling, it is known that pretraining on a large and diverse dataset and finetuning on a smaller but higher quality dataset significantly improves the performance [13, 71]. Since many previous approaches to video modeling have successfully drawn on techniques from the image domain [9, 42, 43], it is noteworthy that the effect of data and training strategies, i.e., the separation of video pretraining at lower resolutions and high-quality finetuning, has yet to be studied. This work directly addresses these previously uncharted territories.

We believe that the significant contribution of *data selection* is heavily underrepresented in today’s video research landscape despite being well-recognized among practitioners when training video models at scale. Thus, in contrast to previous works, we draw on simple latent video diffusion baselines [9] for which we fix architecture and training scheme and assess the effect of *data curation*. To this end, we first identify three different video training stages that we find crucial for good performance: text-to-image pre-training, *video pretraining* on a large dataset at low resolution, and high-resolution *video finetuning* on a much smaller dataset with higher-quality videos. Borrowing from large-scale image model training [13, 64, 66], we introduce a systematic approach to curate video data at scale and present an empirical study on the effect of data curation during video

pretraining. Our main findings imply that pretraining on well-curated datasets leads to significant performance improvements that persist after high-quality finetuning.

**A general motion and multi-view prior** Drawing on these findings, we apply our proposed curation scheme to a large video dataset comprising roughly 600 million samples and train a strong pretrained text-to-video base model, which provides a general motion representation. We exploit this and finetune the base model on a smaller, high-quality dataset for high-resolution downstream tasks such as text-to-video (see Figure 1, top row) and image-to-video, where we predict a sequence of frames from a single conditioning image (see Figure 1, mid rows). Human preference studies reveal that the resulting model outperforms state-of-the-art image-to-video models.

Furthermore, we also demonstrate that our model provides a strong multi-view prior and can serve as a base to finetune a multi-view diffusion model that generates multiple consistent views of an object in a feedforward manner and outperforms specialized novel view synthesis methods such as Zero123XL [14, 57] and SyncDreamer [58]. Finally, we demonstrate that our model allows for explicit motion control by specifically prompting the temporal layers with motion cues and also via training LoRA-modules [32, 45] on datasets resembling specific motions only, which can be efficiently plugged into the model. To summarize, our core contributions are threefold: (i) We present a systematic data curation workflow to turn a large uncurated video collection into a quality dataset for generative video modeling. Using this workflow, we (ii) train state-of-the-art text-to-video and image-to-video models, outperforming all prior models. Finally, we (iii) probe the strong prior of motion and 3D understanding in our models by conducting domain-specific experiments. Specifically, we provide evidence that pretrained video diffusion models can be turned into strong multi-view generators, which may help overcome the data scarcity typically observed in the 3D domain [14].

## 2. Background

Most recent works on video generation rely on diffusion models [38, 84, 87] to jointly synthesize multiple consistent frames from text- or image-conditioning. Diffusion models implement an iterative refinement process by learning to gradually denoise a sample from a normal distribution and have been successfully applied to high-resolution text-to-image [13, 64, 68, 71, 75] and video synthesis [9, 29, 41, 82, 95].

In this work, we follow this paradigm and train a latent [71, 92] video diffusion model [9, 23] on our video dataset. We provide a brief overview of related works which utilize latent video diffusion models (Video-LDMs)

in the following paragraph; a full discussion that includes approaches using GANs [10, 30] and autoregressive models [43] can be found in App. B. An introduction to diffusion models can be found in App. D.

**Latent Video Diffusion Models** Video-LDMs [9, 31, 32, 35, 97] train the main generative model in a latent space of reduced computational complexity [22, 71]. Most related works make use of a pretrained text-to-image model and insert temporal mixing layers of various forms [1, 9, 29, 31, 32] into the pretrained architecture. Ge et al. [29] additionally relies on temporally correlated noise to increase temporal consistency and ease the learning task. In this work, we follow the architecture proposed in Blattmann et al. [9] and insert temporal convolution and attention layers after every spatial convolution and attention layer. In contrast to works that only train temporal layers [9, 32] or are completely training-free [52, 114], we finetune the full model. For text-to-video synthesis in particular, most works directly condition the model on a text prompt [9, 97] or make use of an additional text-to-image prior [23, 82].

In our work, we follow the former approach and show that the resulting model is a strong general motion prior, which can easily be finetuned into an image-to-video or multi-view synthesis model. Additionally, we introduce micro-conditioning [64] on frame rate. We also employ the EDM-framework [51] and significantly shift the noise schedule towards higher noise values, which we find to be essential for high-resolution finetuning. See Section 4 for a detailed discussion of the latter.

**Data Curation** Pretraining on large-scale datasets [80] is an essential ingredient for powerful models in several tasks such as discriminative text-image [66, 105] and language [27, 63, 67] modeling. By leveraging efficient language-image representations such as CLIP [47, 66, 105], data curation has similarly been successfully applied for generative image modeling [13, 64, 80]. However, discussions on such data curation strategies have largely been missing in the video generation literature [41, 43, 82, 94], and processing and filtering strategies have been introduced in an ad-hoc manner. Among the publicly accessible video datasets, WebVid-10M [7] dataset has been a popular choice [9, 82, 115] despite being watermarked and suboptimal in size. Additionally, WebVid-10M is often used in combination with image data [80], to enable joint image-video training. However, this amplifies the difficulty of separating the effects of image and video data on the final model. To address these shortcomings, this work presents a systematic study of methods for video data curation and further introduces a general three-stage training strategy for generative video models, producing a state-of-the-art model.

### 3. Curating Data for HQ Video Synthesis

In this section, we introduce a general strategy to train a state-of-the-art video diffusion model on large datasets of videos. To this end, we (i) introduce data processing and curation methods, for which we systematically analyze the impact on the quality of the final model in Section 3.3 and Section 3.4, and (ii), identify three different training regimes for generative video modeling. In particular, these regimes consist of

- Stage I: *image pretraining*, i.e. a 2D text-to-image diffusion model [13, 64, 71].
- Stage II: *video pretraining*, which trains on large amounts of videos.
- Stage III: *video finetuning*, which refines the model on a small subset of high-quality videos at higher resolution.

We study the importance of each regime separately in Sections 3.2 to 3.4.

#### 3.1. Data Processing and Annotation

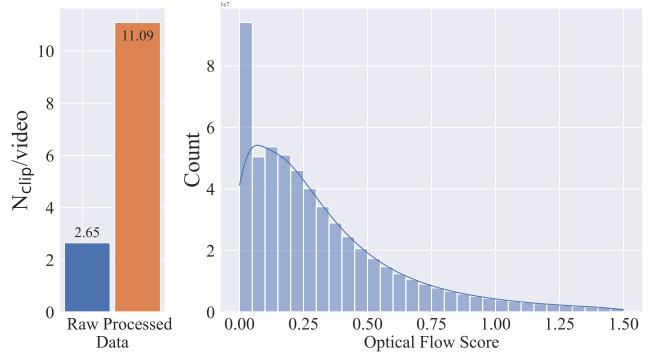


Figure 2. Our initial dataset contains many static scenes and cuts which hurts training of generative video models. *Left:* Average number of clips per video before and after our processing, revealing that our pipeline detects lots of additional cuts. *Right:* We show the distribution of average optical flow score for one of these subsets before our processing, which contains many static clips.

We collect an initial dataset of long videos which forms the base data for our *video pretraining* stage. To avoid cuts and fades leaking into synthesized videos, we apply a cut detection pipeline<sup>1</sup> in a cascaded manner at three different FPS levels. Figure 2, left, provides evidence for the need for cut detection: After applying our cut-detection pipeline, we obtain a significantly higher number ( $\sim 4\times$ ) of clips, indicating that many video clips in the unprocessed dataset contain cuts beyond those obtained from metadata.

Next, we annotate each clip with three different synthetic captioning methods: First, we use the image captioner CoCa [108] to annotate the mid-frame of each clip and use

<sup>1</sup><https://github.com/Breakthrough/PySceneDetect>

Table 1. Comparison of our dataset before and after filtering with publicly available research datasets.

	<i>LVD</i>	<i>LVD-F</i>	<i>LVD-10M</i>	<i>LVD-10M-F</i>	<i>WebVid</i>	<i>InternVid</i>
#Clips	577M	152M	9.8M	2.3M	10.7M	234M
Clip Duration (s)	11.58	10.53	12.11	10.99	18.0	11.7
Total Duration (y)	212.09	50.64	3.76	0.78	5.94	86.80
Mean #Frames	325	301	335	320	-	-
Mean Clips/Video	11.09	4.76	1.2	1.1	1.0	32.96
Motion Annotations?	✓	✓	✓	✓	✗	✗

V-BLIP [109] to obtain a video-based caption. Finally, we generate a third description of the clip via an LLM-based summarization of the first two captions.

The resulting initial dataset, which we dub *Large Video Dataset (LVD)*, consists of 580M annotated video clip pairs, forming 212 years of content.

However, further investigation reveals that the resulting dataset contains examples that can be expected to degrade the performance of our final video model, such as clips with less motion, excessive text presence, or generally low aesthetic value. We therefore additionally annotate our dataset with dense optical flow [24, 48], which we calculate at 2 FPS and with which we filter out static scenes by removing any videos whose average optical flow magnitude is below a certain threshold. Indeed, when considering the motion distribution of *LVD* (see Figure 2, right) via optical flow scores, we identify a subset of close-to-static clips therein.

Moreover, we apply optical character recognition [5] to weed out clips containing large amounts of written text. Lastly, we annotate the first, middle, and last frames of each clip with CLIP [66] embeddings from which we calculate aesthetics scores [80] as well as text-image similarities. Statistics of our dataset, including the total size and average duration of clips, are provided in Tab. 1.

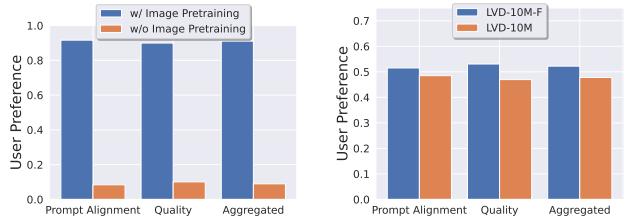
### 3.2. Stage I: Image Pretraining

We consider image pretraining as the first stage in our training pipeline. Thus, in line with concurrent work on video models [9, 41, 82], we ground our initial model on a pre-trained image diffusion model - namely *Stable Diffusion 2.1* [71] - to equip it with a strong visual representation.

To analyze the effects of image pretraining, we train and compare two identical video models as detailed in App. D on a 10M subset of *LVD*; one with and one without pre-trained spatial weights. We compare these models using a human preference study (see App. E for details) in Figure 3a, which clearly shows that the image-pretrained model is preferred in both quality and prompt-following.

### 3.3. Stage II: Curating a Video Pretraining Dataset

**A systematic approach to video data curation.** For multimodal image modeling, data curation is a key element of many powerful discriminative [66, 105] and generative [13, 40, 69] models. However, since there are no



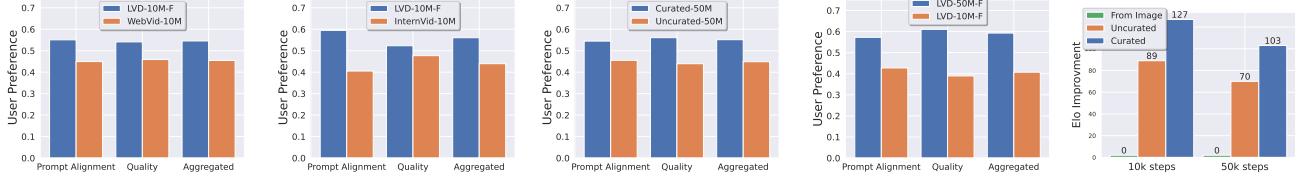
(a) Initializing spatial layers from pretrained image models clearly outperforms a similar one with randomly initialized spatial weights as shown in Figure 3a. Figure 3b emphasizes the importance of data curation for pretraining, since training on a curated subset of *LVD-10M* with the filtering threshold proposed in Section 3.3 improves upon training on the entire uncurated *LVD-10M*.

Figure 3. Effects of image-only pretraining and data curation on video-pretraining on *LVD-10M*: A video model with spatial layers initialized from a pretrained image model clearly outperforms a similar one with randomly initialized spatial weights as shown in Figure 3a. Figure 3b emphasizes the importance of data curation for pretraining, since training on a curated subset of *LVD-10M* with the filtering threshold proposed in Section 3.3 improves upon training on the entire uncurated *LVD-10M*.

equally powerful off-the-shelf representations available in the video domain to filter out unwanted examples, we rely on human preferences as a signal to create a suitable pre-training dataset. Specifically, we curate subsets of *LVD* using different methods described below and then consider the human-preference-based ranking of latent video diffusion models trained on these datasets.

More specifically, for each type of annotation introduced in Section 3.1 (*i.e.*, CLIP scores, aesthetic scores, OCR detection rates, synthetic captions, optical flow scores), we start from an unfiltered, randomly sampled 9.8M-sized subset of *LVD*, *LVD-10M*, and systematically remove the bottom 12.5, 25 and 50% of examples. Note that for the synthetic captions, we cannot filter in this sense. Instead, we assess Elo rankings [21] for the different captioning methods from Section 3.1. To keep the number of total subsets tractable, we apply this scheme separately to each type of annotation. We train models with the same training hyperparameters on each of these filtered subsets and compare the results of all models within the same class of annotation with an Elo ranking [21] for human preference votes. Based on these votes, we consequently select the best-performing filtering threshold for each annotation type. The details of this study are presented and discussed in App. E. Applying this filtering approach to *LVD* results in a final pretraining dataset of 152M training examples, which we refer to as *LVD-F*, *cf.* Tab. 1.

**Curated training data improves performance.** In this section, we demonstrate that the data curation approach described above improves the training of our video diffusion models. To show this, we apply the filtering strategy described above to *LVD-10M* and obtain a four times smaller subset, *LVD-10M-F*. Next, we use it to train a baseline model that follows our standard architecture and train-



(a) User preference for *LVD-10M-F* and *WebVid-10M* [7]. (b) User preference for *LVD-10M-F* and *InternVid-10M* [100]. (c) User preference at 50M samples scales. (d) User preference on scaling datasets. (e) Relative ELO progression over time during Stage III.

Figure 4. *Summarized findings of Sections 3.3 and 3.4*: Pretraining on curated datasets consistently boosts performance of generative video models during *video pretraining* at small (Figures 4a and 4b) and larger scales (Figures 4c and 4d). Remarkably, this performance improvement persists even after 50k steps of *video finetuning* on high quality data (Figure 4e).

ing schedule and evaluate the preference scores for visual quality and prompt-video alignment compared to a model trained on uncurated *LVD-10M*.

We visualize the results in Figure 3b, where we can see the benefits of filtering: In both categories, the model trained on the much smaller *LVD-10M-F* is preferred. To further show the efficacy of our curation approach, we compare the model trained on *LVD-10M-F* with similar video models trained on *WebVid-10M* [7], which is the most recognized research licensed dataset, and *InternVid-10M* [100], which is specifically filtered for high aesthetics. Although *LVD-10M-F* is also four times smaller than these datasets, the corresponding model is preferred by human evaluators in both spatiotemporal quality and prompt alignment as shown in Figure 4b.

**Data curation helps at scale.** To verify that our data curation strategy from above also works on larger, more practically relevant datasets, we repeat the experiment above and train a video diffusion model on a filtered subset with 50M examples and a non-curated one of the same size. We conduct a human preference study and summarize the results of this study in Figure 4c, where we can see that the advantages of data curation also come into play with larger amounts of data. Finally, we show that dataset size is also a crucial factor when training on curated data in Figure 4d, where a model trained on 50M curated samples is superior to a model trained on *LVD-10M-F* for the same number of steps.

### 3.4. Stage III: High-Quality Finetuning

In the previous section, we demonstrated the beneficial effects of systematic data curation for *video pretraining*. However, since we are primarily interested in optimizing the performance after *video finetuning*, we now investigate how these differences after Stage II translate to the final performance after Stage III. Here, we draw on training techniques from latent image diffusion modeling [13, 64] and increase the resolution of the training examples. Moreover, we use a small finetuning dataset comprising 250K pre-captioned video clips of high visual fidelity.

To analyze the influence of *video pretraining* on this last

stage, we finetune three identical models, which only differ in their initialization. We initialize the weights of the first with a pretrained image model and skip *video pretraining*, a common choice among many recent video modeling approaches [9, 82]. The remaining two models are initialized with the weights of the latent video models from the previous section, specifically, the ones trained on 50M curated and uncurated video clips. We finetune all models for 50K steps and assess human preference rankings early during finetuning (10K steps) and at the end to measure how performance differences progress in the course of finetuning. We show the obtained results in Figure 4e, where we plot the Elo improvements of user preference relative to the model ranked last, which is the one initialized from an image model. Moreover, the finetuning resumed from curated pretrained weights ranks consistently higher than the one initialized from video weights after uncurated training.

Given these results, we conclude that i) the separation of video model training in *video pretraining* and *video finetuning* is beneficial for the final model performance after finetuning and that ii) *video pretraining* should ideally occur on a large scale, curated dataset, since performance differences after pretraining persist after finetuning.

## 4. Training Video Models at Scale

In this section, we borrow takeaways from Section 3 and present results of training state-of-the-art video models at scale. We first use the optimal data strategy inferred from ablations to train a powerful base model at  $320 \times 576$  in App. D.2. We then perform finetuning to yield several strong state-of-the-art models for different tasks such as text-to-video in Section 4.2, image-to-video in Section 4.3 and frame interpolation in Section 4.4. Finally, we demonstrate that our video-pretraining can serve as a strong implicit 3D prior, by tuning our image-to-video models on multi-view generation in Section 4.5 and outperform concurrent work, in particular Zero123XL [14, 57] and SyncDreamer [58] in terms of multi-view consistency.

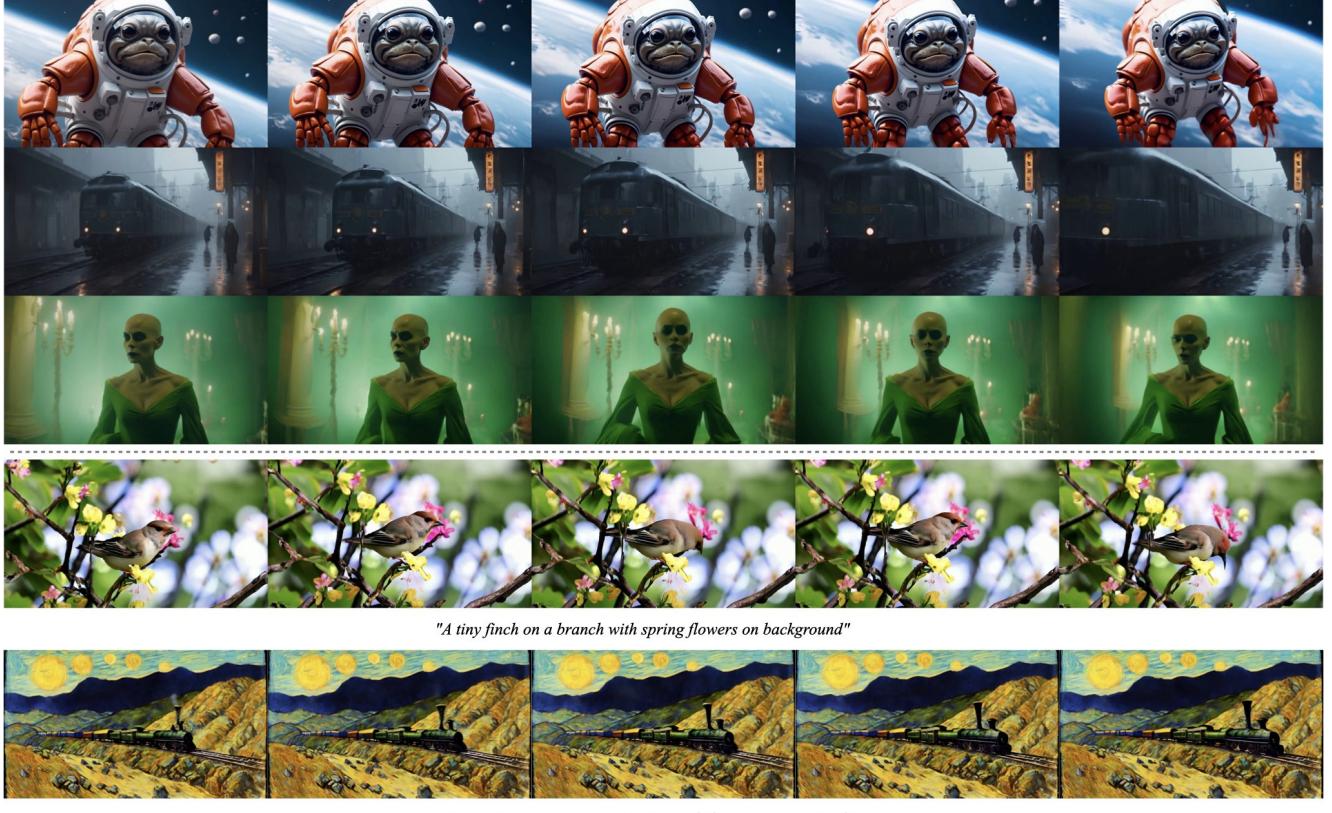


Figure 5. Samples at  $576 \times 1024$ . Top: Image-to-video samples (conditioned on leftmost frame). Bottom: Text-to-video samples.

Table 2. UCF-101 zero-shot text-to-video generation. Comparing our base model to baselines (numbers from literature).

Method	FVD ( $\downarrow$ )
CogVideo (ZH) [43]	751.34
CogVideo (EN) [43]	701.59
Make-A-Video [82]	367.23
Video LDM [9]	550.61
MagicVideo [115]	655.00
PYOCO [29]	355.20
SVD (ours)	<b>242.02</b>

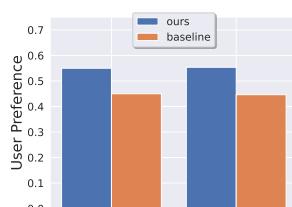


Figure 6. Our 25 frame Image-to-Video model is preferred by human voters over GEN-2 [74] and PikaLabs [54].

#### 4.1. Pretrained Base Model

As discussed in Section 3.2, our video model is based on *Stable Diffusion 2.1* [71] (SD 2.1). Recent works [44] show that it is crucial to adopt the noise schedule when training image diffusion models, shifting towards more noise for higher-resolution images. As a first step, we finetune the fixed discrete noise schedule from our image model towards continuous noise [87] using the network preconditioning proposed in Karras et al. [51] for images of size  $256 \times 384$ . After inserting temporal layers, we then train the model on *LVD-F* on 14 frames at resolution  $256 \times 384$ . We use the

standard EDM noise schedule [51] for 150k iterations and batch size 1536. Next, we finetune the model to generate 14  $320 \times 576$  frames for 100k iterations using batch size 768. We find that it is important to shift the noise schedule towards more noise for this training stage, confirming results by Hoogeboom et al. [44] for image models. For further training details, see App. D. We refer to this model as our *base model* which can be easily finetuned for a variety of tasks as we show in the following sections. The base model has learned a powerful motion representation, for example, it significantly outperforms all baselines for zero-shot text-to-video generation on UCF-101 [88] (Tab. 2). Evaluation details can be found in App. E.

#### 4.2. High-Resolution Text-to-Video Model

We finetune the base text-to-video model on a high-quality video dataset of  $\sim 1M$  samples. Samples in the dataset generally contain lots of object motion, steady camera motion, and well-aligned captions, and are of high visual quality altogether. We finetune our base model for 50k iterations at resolution  $576 \times 1024$  (again shifting the noise schedule towards more noise) using batch size 768. Samples in Figure 5, more can be found in App. E.



Figure 7. Applying three camera motion LoRAs (*horizontal*, *zooming*, *static*) to the same conditioning frame (on the left).

### 4.3. High Resolution Image-to-Video Model

Besides text-to-video, we finetune our base model for image-to-video generation, where the video model receives a still input image as a conditioning. Accordingly, we replace text embeddings that are fed into the base model with the CLIP image embedding of the conditioning. Additionally, we concatenate a noise-augmented [39] version of the conditioning frame channel-wise to the input of the UNet [73]. We do not use any masking techniques and simply copy the frame across the time axis. We finetune two models, one predicting 14 frames and another one predicting 25 frames; implementation and training details can be found in App. D. We occasionally found that standard vanilla classifier-free guidance [36] can lead to artifacts: too little guidance may result in inconsistency with the conditioning frame while too much guidance can result in oversaturation. Instead of using a constant guidance scale, we found it helpful to linearly increase the guidance scale across the frame axis (from small to high). Details can be found in App. D. Samples in Figure 5, more can be found in App. E.

In Section 4.5 we compare our model with state-of-the-art, closed-source video generative models, in particular GEN-2 [23, 74] and PikaLabs [54], and show that our model is preferred in terms of visual quality by human voters. Details on the experiment, as well as many more image-to-video samples, can be found in App. E.

#### 4.3.1 Camera Motion LoRA

To facilitate controlled camera motion in image-to-video generation, we train a variety of *camera motion LoRAs* within the temporal attention blocks of our model [32]; see App. D for exact implementation details. We train these additional parameters on a small dataset with rich camera-motion metadata. In particular, we use three subsets of the data for which the camera motion is categorized as “hor-

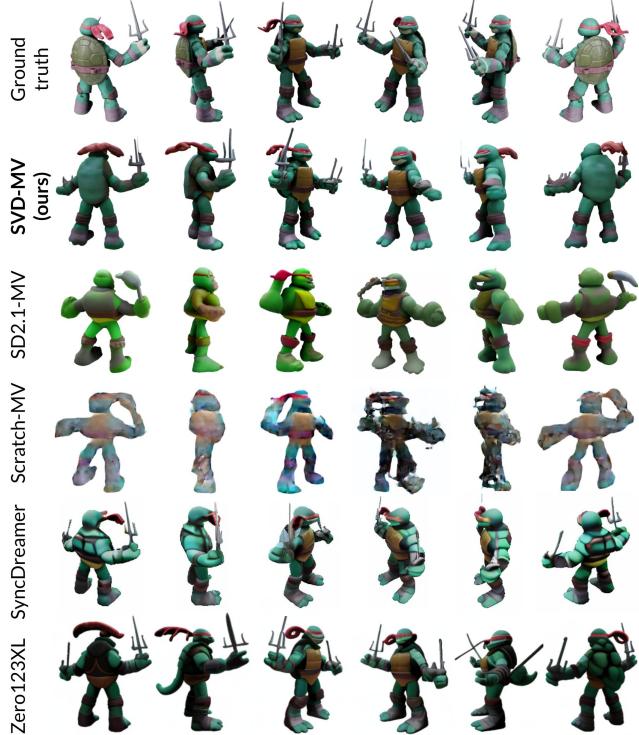


Figure 8. Generated multi-view frames of a GSO test object using our SVD-MV model (i.e. SVD finetuned for Multi-View generation), SD2.1-MV [72], Scratch-MV, SyncDreamer [58], and Zero123XL [14].

izontally moving”, “zooming”, and “static”. In Figure 7 we show samples of the three models for identical conditioning frames; more samples can be found in App. E.

### 4.4. Frame Interpolation

To obtain smooth videos at high frame rates, we finetune our high-resolution text-to-video model into a frame interpolation model. We follow Blattmann et al. [9] and concatenate the left and right frames to the input of the UNet via *masking*. The model learns to predict three frames within the two conditioning frames, effectively increasing the frame rate by four. Surprisingly, we found that a very small number of iterations ( $\approx 10k$ ) suffices to get a good model. Details and samples can be found in App. D and App. E, respectively.

### 4.5. Multi-View Generation

To obtain multiple novel views of an object simultaneously, we finetune our image-to-video SVD model on multi-view datasets [14, 15, 111].

**Datasets.** We finetuned our SVD model on two datasets, where the SVD model takes a single image and outputs a sequence of multi-view images: (i) A subset of Objaverse [15] consisting of 150K curated and CC-licensed synthetic 3D objects from the original dataset [15]. For each object, we rendered 360° orbital videos of 21 frames with

randomly sampled HDRI environment map and elevation angles between  $[-5^\circ, 30^\circ]$ . We evaluate the resulting models on an unseen test dataset consisting of 50 sampled objects from Google Scanned Objects (GSO) dataset [20], and (ii) MVImgNet [111] consisting of casually captured multi-view videos of general household objects. We split the videos into  $\sim 200$ K train and 900 test videos. We rotate the frames captured in portrait mode to landscape orientation.

The Objaverse-trained model is additionally conditioned on the elevation angle of the input image, and outputs orbital videos at that elevation angle. The MVImgNet-trained models are not conditioned on pose and can choose an arbitrary camera path in their generations. For details on the pose conditioning mechanism, see App. E.

**Models.** We refer to our finetuned Multi-View model as SVD-MV. We perform an ablation study on the importance of the video prior of SVD for multi-view generation. To this effect, we compare the results from SVD-MV i.e. from a video prior to those finetuned from an image prior i.e. the text-to-image model SD2.1 (SD2.1-MV), and that trained without a prior i.e. from random initialization (Scratch-MV). In addition, we compare with the current state-of-the-art multiview generation models of Zero123 [57], Zero123XL [14], and SyncDreamer [58].

**Metrics.** We use the standard metrics of Peak Signal-to-Noise Ratio (PSNR), LPIPS [112], and CLIP [66] Similarity scores (CLIP-S) between the corresponding pairs of ground truth and generated frames on 50 GSO test objects.

**Training.** We train all our models for 12k steps ( $\sim 16$  hours) with 8 80GB A100 GPUs using a total batch size of 16, with a learning rate of 1e-5.

**Results.** Figure 9(a) shows the average metrics on the GSO test dataset. The higher performance of SVD-MV compared to SD2.1-MV and Scratch-MV clearly demonstrates the advantage of the learned video prior in the SVD model for multi-view generation. In addition, as in the case of other models finetuned from SVD, we found that a very small number of iterations ( $\approx 12k$ ) suffices to get a good model. Moreover, SVD-MV is competitive w.r.t state-of-the-art techniques with lesser training time (12k iterations in 16 hours), whereas existing models are typically trained for much longer (for example, SyncDreamer was trained for four days specifically on Objaverse). Figure 9(b) shows convergence of different finetuned models. After only 1k iterations, SVD-MV has much better CLIP-S and PSNR scores than its image-prior and no-prior counterparts.

Figure 8 shows a qualitative comparison of multi-view generation results on a GSO test object and Figure 10 on an MVImgNet test object. As can be seen, our generated frames are multi-view consistent and realistic. More details on the experiments, as well as more multi-view generation samples, can be found in App. E.

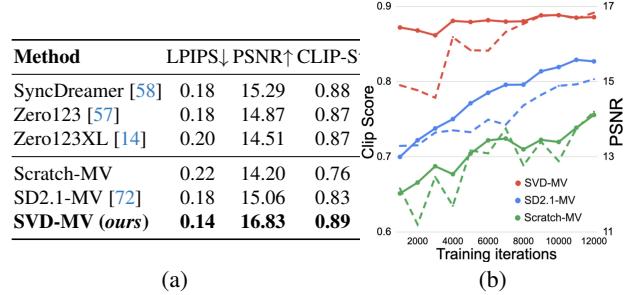


Figure 9. (a) Multi-view generation metrics on Google Scanned Objects (GSO) test dataset. SVD-MV outperforms image-prior (SD2.1-MV) and no-prior (Scratch-MV) variants, as well other state-of-the-art techniques. (b) Training progress of multi-view generation models with CLIP-S (solid, left axis) and PSNR (dotted, right axis) computed on GSO test dataset. SVD-MV shows better metrics consistently from the start of finetuning.



Figure 10. Generated novel multi-view frames for MVImgNet dataset using our SVD-MV model, SD2.1-MV [72], Scratch-MV.

## 5. Conclusion

We present *Stable Video Diffusion* (SVD), a latent video diffusion model for high-resolution, state-of-the-art text-to-video and image-to-video synthesis. To construct its pre-training dataset, we conduct a systematic data selection and scaling study, and propose a method to curate vast amounts of video data and turn large and noisy video collection into suitable datasets for generative video models. Furthermore, we introduce three distinct stages of video model training which we separately analyze to assess their impact on the final model performance. *Stable Video Diffusion* provides a powerful video representation from which we finetune video models for state-of-the-art image-to-video synthesis and other highly relevant applications such as LoRAs for camera control. Finally we provide a pioneering study on multi-view finetuning of video diffusion models and show that SVD constitutes a strong 3D prior, which obtains state-of-the-art results in multi-view synthesis while using only a

fraction of the compute of previous methods.

We hope these findings will be broadly useful in the generative video modeling literature. A discussion on our work’s broader impact and limitations can be found in App. A.

## Acknowledgements

Special thanks to Emad Mostaque for his excellent support on this project. Many thanks go to our colleagues Jonas Müller, Axel Sauer, Dustin Podell and Rahim Entezari for fruitful discussions and comments. Finally, we thank Harry Saini and the one and only Richard Vencu for maintaining and optimizing our data and computing infrastructure.

## References

- [1] Jie An, Songyang Zhang, Harry Yang, Sonal Gupta, Jia-Bin Huang, Jiebo Luo, and Xi Yin. Latent-shift: Latent diffusion with temporal shift for efficient text-to-video generation. *arXiv preprint arXiv:2304.08477*, 2023. 3
- [2] Titas Auciukevičius, Zexiang Xu, Matthew Fisher, Paul Henderson, Hakan Bilen, Niloy J Mitra, and Paul Guerrero. Renderdiffusion: Image diffusion for 3d reconstruction, inpainting and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12608–12618, 2023. 15
- [3] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A general language assistant as a laboratory for alignment, 2021. 22
- [4] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. In *International Conference on Learning Representations*, 2018. 15
- [5] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9365–9374, 2019. 4, 18
- [6] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. 22
- [7] Max Bain, Arsha Nagrani, Gü̈l Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval, 2022. 3, 5, 15
- [8] Andreas Blattmann, Timo Milbich, Michael Dorkenwald, and Björn Ommer. ipoke: Poking a still image for controlled stochastic video synthesis. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, 2021. 15
- [9] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models. *arXiv:2304.08818*, 2023. 2, 3, 4, 5, 6, 7, 15, 19, 20, 23, 25
- [10] Tim Brooks, Janne Hellsten, Miika Aittala, Ting-Chun Wang, Timo Aila, Jaakko Lehtinen, Ming-Yu Liu, Alexei A Efros, and Tero Karras. Generating long videos of dynamic scenes. In *NeurIPS*, 2022. 3, 15, 24
- [11] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 15, 24
- [12] Lluis Castrejon, Nicolas Ballas, and Aaron Courville. Improved conditional vrnns for video prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 15
- [13] Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam Tsai, Jiliang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, Matthew Yu, Abhishek Kadian, Filip Radenovic, Dhruv Mahajan, Kunpeng Li, Yue Zhao, Vladan Petrovic, Mitesh Kumar Singh, Simran Motwani, Yi Wen, Yiwen Song, Roshan Sumbaly, Vignesh Ramathan, Zijian He, Peter Vajda, and Devi Parikh. Emu: Enhancing image generation models using photogenic needles in a haystack, 2023. 2, 3, 4, 5
- [14] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-XL: A universe of 10m+ 3d objects. *arXiv preprint arXiv:2307.05663*, 2023. 2, 5, 7, 8
- [15] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 7
- [16] Congyue Deng, Chiyu Jiang, Charles R Qi, Xinchen Yan, Yin Zhou, Leonidas Guibas, Dragomir Anguelov, et al. Nerdi: Single-view nerf synthesis with language-guided diffusion as general image priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20637–20647, 2023. 15
- [17] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018*, 2018. 15
- [18] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. *arXiv:2105.05233*, 2021. 25

- [19] Michael Dorkenwald, Timo Milbich, Andreas Blattmann, Robin Rombach, Konstantinos G. Derpanis, and Björn Ommer. Stochastic image-to-video synthesis using cinns. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 2021. 15
- [20] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022. 8
- [21] Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Pub., New York, 1978. 4, 22
- [22] Patrick Esser, Robin Rombach, and Björn Ommer. Tampering transformers for high-resolution image synthesis. *arXiv preprint arXiv:2012.09841*, 2020. 3
- [23] Patrick Esser, Johnathan Chiu, Parmida Atighchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models, 2023. 2, 3, 7
- [24] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. pages 363–370, 2003. 4, 17
- [25] Gereon Fox, Ayush Tewari, Mohamed Elgharib, and Christian Theobalt. Stylevideogan: A temporal generative model using a pretrained stylegan. In *British Machine Vision Conference (BMVC)*, 2021. 15
- [26] Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamplier, and Patrick Gallinari. Stochastic latent residual video prediction. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. 15
- [27] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020. 3
- [28] Songwei Ge, Thomas Hayes, Harry Yang, Xi Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. Long video generation with time-agnostic vqgan and time-sensitive transformer. In *Computer Vision – ECCV 2022*, pages 102–118, Cham, 2022. Springer Nature Switzerland. 15
- [29] Songwei Ge, Seungjun Nah, Guilin Liu, Tyler Poon, Andrew Tao, Bryan Catanzaro, David Jacobs, Jia-Bin Huang, Ming-Yu Liu, and Yogesh Balaji. Preserve your own correlation: A noise prior for video diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22930–22941, 2023. 2, 3, 6, 15
- [30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 3
- [31] Jiaxi Gu, Shicong Wang, Haoyu Zhao, Tianyi Lu, Xing Zhang, Zuxuan Wu, Songcen Xu, Wei Zhang, Yu-Gang Jiang, and Hang Xu. Reuse and diffuse: Iterative denoising for text-to-video generation. *arXiv preprint arXiv:2309.03549*, 2023. 3
- [32] Yuwei Guo, Ceyuan Yang, Anyi Rao, Yaohui Wang, Yu Qiao, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023. 2, 3, 7, 15, 20
- [33] Sonam Gupta, Arti Keshari, and Sukhendu Das. Rv-gan: Recurrent gan for unconditional video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2024–2033, 2022. 15
- [34] Nicholas Guttenberg and CrossLabs. Diffusion with offset noise, 2023. 19, 23
- [35] Yingqing He, Tianyu Yang, Yong Zhang, Ying Shan, and Qifeng Chen. Latent video diffusion models for high-fidelity long video generation, 2023. 3
- [36] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 7, 19, 20
- [37] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. *arXiv:2207.12598*, 2022. 19, 23
- [38] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020. 2, 25
- [39] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *arXiv preprint arXiv:2106.15282*, 2021. 7, 15, 20
- [40] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, and Tim Salimans. Imagen Video: High Definition Video Generation with Diffusion Models. *arXiv:2210.02303*, 2022. 4
- [41] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 2, 3, 4, 15, 20
- [42] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022. 2, 15
- [43] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers, 2022. 2, 3, 6, 15
- [44] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. *arXiv preprint arXiv:2301.11093*, 2023. 6
- [45] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 2
- [46] Aapo Hyvärinen and Peter Dayan. Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, 6(4), 2005. 18
- [47] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave,

- Vaishaal Shankar, Hongseok Namkoong, John Miller, Hanneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 3
- [48] Itseez. Open source computer vision library. <https://github.com/itseez/opencv>, 2015. 4, 17
- [49] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions, 2023. 15
- [50] Emmanuel Kahembwe and Subramanian Ramamoorthy. Lower dimensional kernels for video discriminators. *Neural Networks*, 132:506–520, 2020. 15
- [51] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. *arXiv:2206.00364*, 2022. 3, 6, 18, 19
- [52] Levon Khachatryan, Andranik Moysalyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators, 2023. 3
- [53] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021. 19
- [54] Pika Labs. Pika labs, <https://www.pika.art/>, 2023. 2, 6, 7, 24
- [55] Alex X. Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018. 15
- [56] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common Diffusion Noise Schedules and Sample Steps are Flawed. *arXiv:2305.08891*, 2023. 19
- [57] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. 2, 5, 8, 15
- [58] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syndreamer: Generating multiview-consistent images from a single-view image. *arXiv preprint arXiv:2309.03453*, 2023. 2, 5, 7, 8, 15, 16
- [59] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 19, 20, 23
- [60] Pauline Luc, Aidan Clark, Sander Dieleman, Diego de Las Casas, Yotam Doron, Albin Cassirer, and Karen Simonyan. Transformation-based adversarial video prediction on large-scale data. *ArXiv*, 2020. 15
- [61] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models, 2023. 15
- [62] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts, 2022. 15
- [63] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidi, Baptiste Pannier, Ebtesam Almazrouei, and Julien Lauvay. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023. 3
- [64] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis. *arXiv:2307.01952*, 2023. 2, 3, 5, 24
- [65] Giovanni Puccetti, Maciej Kilian, and Romain Beaumont. Training contrastive captioners. *LAION blog*, 2023. 17
- [66] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arXiv:2103.00020*, 2021. 2, 3, 4, 8, 18
- [67] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019. 3
- [68] Aditya Ramesh. How dall-e 2 works, 2022. 2
- [69] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 4
- [70] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. *arXiv:2204.06125*, 2022. 16
- [71] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. *arXiv preprint arXiv:2112.10752*, 2021. 2, 3, 4, 6, 15, 19, 20
- [72] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *arXiv preprint arXiv:2112.10752*, 2021. 7, 8
- [73] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597*, 2015. 7, 20, 23
- [74] RunwayML. Gen-2 by runway, <https://research.runwayml.com/gen2>, 2023. 2, 6, 7, 24
- [75] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*, 2021. 2
- [76] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022. 2
- [77] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *ICCV*, 2017. 15
- [78] Masaki Saito, Shunta Saito, Masanori Koyama, and Sosuke Kobayashi. Train sparsely, generate densely: Memory-efficient unsupervised training of high-resolution temporal gan. *International Journal of Computer Vision*, 2020. 15

- [79] Tim Salimans and Jonathan Ho. Progressive Distillation for Fast Sampling of Diffusion Models. *arXiv preprint arXiv:2202.00512*, 2022. [15](#), [19](#)
- [80] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022. [3](#), [4](#), [16](#), [18](#)
- [81] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. [15](#)
- [82] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-A-Video: Text-to-Video Generation without Text-Video Data. *arXiv:2209.14792*, 2022. [2](#), [3](#), [4](#), [5](#), [6](#), [15](#), [20](#)
- [83] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhosiny. Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3626–3636, 2022. [15](#)
- [84] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. *arXiv:1503.03585*, 2015. [2](#)
- [85] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Understanding and mitigating copying in diffusion models, 2023. [16](#), [17](#)
- [86] Yang Song and Stefano Ermon. Improved Techniques for Training Score-Based Generative Models. *arXiv:2006.09011*, 2020. [23](#)
- [87] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. *arXiv:2011.13456*, 2020. [2](#), [6](#), [18](#)
- [88] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. [6](#), [15](#), [24](#)
- [89] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision-ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020. [17](#)
- [90] Yu Tian, Jian Ren, Menglei Chai, Kyle Olszewski, Xi Peng, Dimitris N. Metaxas, and Sergey Tulyakov. A good image generator is what you need for high-resolution video synthesis. In *International Conference on Learning Representations*, 2021. [15](#)
- [91] Suranya Tomar. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006. [17](#)
- [92] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In *Advances in Neural Information Processing Systems*, 2021. [2](#)
- [93] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *ICLR*, 2017. [15](#)
- [94] Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description. *arXiv:2210.02399*, 2022. [3](#), [15](#)
- [95] Vikram Voleti, Alexia Jolicoeur-Martineau, and Christopher Pal. Mcvd: Masked conditional video diffusion for prediction, generation, and interpolation. In *(NeurIPS) Advances in Neural Information Processing Systems*, 2022. [2](#)
- [96] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016. [15](#)
- [97] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. **Modelscope text-to-video technical report**. *arXiv preprint arXiv:2308.06571*, 2023. [3](#), [15](#)
- [98] Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. G3an: Disentangling appearance and motion for video generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. [15](#)
- [99] Yaohui Wang, Xinyuan Chen, Xin Ma, Shangchen Zhou, Ziqi Huang, Yi Wang, Ceyuan Yang, Yinan He, Jiashuo Yu, Peiqing Yang, et al. Lavie: High-quality video generation with cascaded latent diffusion models. *arXiv preprint arXiv:2309.15103*, 2023. [15](#)
- [100] Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinyuan Chen, Yaohui Wang, Ping Luo, Ziwei Liu, Yali Wang, Limin Wang, and Yu Qiao. Internvid: A large-scale video-text dataset for multimodal understanding and generation, 2023. [5](#), [16](#)
- [101] Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models, 2022. [15](#)
- [102] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. In *International Conference on Learning Representations*, 2020. [15](#)
- [103] Chenfei Wu, Lun Huang, Qianxi Zhang, Binyang Li, Lei Ji, Fan Yang, Guillermo Sapiro, and Nan Duan. Godiva: Generating open-domain videos from natural descriptions. *arXiv:2104.14806*, 2021.
- [104] Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Dixin Jiang, and Nan Duan. Nüwa: Visual synthesis pre-training for neural visual world creation. In *European Conference on Computer Vision*, pages 720–736. Springer, 2022. [15](#)
- [105] Hu Xu, Saining Xie, Xiaoqing Ellen Tan, Po-Yao Huang, Russell Howes, Vasu Sharma, Shang-Wen Li, Gargi Ghosh, Luke Zettlemoyer, and Christoph Feichtenhofer. Demystifying clip data, 2023. [2](#), [3](#), [4](#)
- [106] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language.

In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 15

- [107] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers, 2021. 15
- [108] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models, 2022. 3, 23
- [109] Keunwoo Peter Yu. Videoblip. <https://github.com/yukw777/VideoBLIP>, 2023. If you use VideoBLIP, please cite it as below. 4, 17
- [110] Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin. Generating videos with dynamics-aware implicit generative adversarial networks. In *International Conference on Learning Representations*, 2022. 15
- [111] Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Chenming Zhu, Zhangyang Xiong, Tianyou Liang, et al. Mvimgnet: A large-scale dataset of multi-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9150–9161, 2023. 7, 8
- [112] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. 8
- [113] Shiwei Zhang, Jiayu Wang, Yingya Zhang, Kang Zhao, Hangjie Yuan, Zhiwu Qin, Xiang Wang, Deli Zhao, and Jingren Zhou. I2vgan-xl: High-quality image-to-video synthesis via cascaded diffusion models. *arXiv preprint arXiv:2311.04145*, 2023. 15
- [114] Yabo Zhang, Yuxiang Wei, Dongsheng Jiang, Xiaopeng Zhang, Wangmeng Zuo, and Qi Tian. Controlvideo: Training-free controllable text-to-video generation, 2023. 3
- [115] Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. Magicvideo: Efficient video generation with latent diffusion models. *arXiv preprint arXiv:2211.11018*, 2022. 3, 6, 15
- [116] Zhizhuo Zhou and Shubham Tulsiani. Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12588–12597, 2023. 15

## Contents

<b>1. Introduction</b>	<b>2</b>
<b>2. Background</b>	<b>2</b>
<b>3. Curating Data for HQ Video Synthesis</b>	<b>3</b>
3.1. Data Processing and Annotation . . . . .	3
3.2. Stage I: Image Pretraining . . . . .	4
3.3. Stage II: Curating a Video Pretraining Dataset . . . . .	4
3.4. Stage III: High-Quality Finetuning . . . . .	5
<b>4. Training Video Models at Scale</b>	<b>5</b>
4.1. Pretrained Base Model . . . . .	6
4.2. High-Resolution Text-to-Video Model . . . . .	6
4.3. High Resolution Image-to-Video Model . . . . .	7
4.3.1 Camera Motion LoRA . . . . .	7
4.4. Frame Interpolation . . . . .	7
4.5. Multi-View Generation . . . . .	7
<b>5. Conclusion</b>	<b>8</b>
<b>A Broader Impact and Limitations</b>	<b>15</b>
<b>B Related Work</b>	<b>15</b>
<b>C Data Processing</b>	<b>16</b>
<b>D Model and Implementation Details</b>	<b>18</b>
D.1. Diffusion Models . . . . .	18
D.2 Base Model Training and Architecture . . . . .	19
D.3 High-Resolution Text-to-Video Model . . . . .	20
D.4 High-Resolution Image-to-Video Model . . . . .	20
D.4.1 Linearly Increasing Guidance . . . . .	20
D.4.2 Camera Motion LoRA . . . . .	20
D.5. Interpolation Model Details . . . . .	20
D.6 Multi-view generation . . . . .	21
<b>E Experiment Details</b>	<b>21</b>
E.1. Details on Human Preference Assessment . . . . .	21
E.1.1 Experimental Setup . . . . .	22
E.1.2 Elo Score Calculation . . . . .	22
E.2. Details on Experiments from Section 3 . . . . .	23
E.2.1 Architectural Details . . . . .	23
E.2.2 Calibrating Filtering Thresholds . . . . .	23
E.2.3 Finetuning Experiments . . . . .	24
E.3. Human Eval vs SOTA . . . . .	24
E.4. UCF101 FVD . . . . .	24
E.5. Additional Samples . . . . .	24
E.5.1 Additional Text-to-Video Samples . . . . .	24
E.5.2 Additional Image-to-Video Samples . . . . .	24
E.5.3 Additional Camera Motion LoRA Samples . . . . .	24
E.5.4 Temporal Prompting via Temporal Cross-Attention Layers . . . . .	25
E.5.5 Additional Samples on Multi-View Synthesis . . . . .	25

## Appendix

### A. Broader Impact and Limitations

**Broader Impact:** Generative models for different modalities promise to revolutionize the landscape of media creation and use. While exploring their creative applications, reducing the potential to use them for creating misinformation and harm are crucial aspects before real-world deployment. Furthermore, risk analyses need to highlight and evaluate the differences between the various existing model types, such as interpolation, text-to-video, animation, and long-form generation. Before these models are used in practice, a thorough investigation of the models themselves, their intended uses, safety aspects, associated risks, and potential biases is essential.

**Limitations:** While our approach excels at short video generation, it comes with some fundamental shortcomings w.r.t. long video synthesis: Although a latent approach provides efficiency benefits, generating multiple keyframes at once is expensive both during training but also inference, and future work on long video synthesis should either try a cascade of very coarse frame generation or build dedicated tokenizers for video generation. Furthermore, videos generated with our approach sometimes suffer from too little generated motion. Lastly, video diffusion models are typically slow to sample and have high VRAM requirements, and our model is no exception. Diffusion distillation methods [41, 61, 79] are promising candidates for faster synthesis.

### B. Related Work

**Video Synthesis.** Many approaches based on various models such as variational RNNs [4, 12, 17, 26, 55], normalizing flows [8, 19], autoregressive transformers [28, 33, 43, 102–104, 107], and GANs [10, 25, 50, 60, 77, 78, 83, 90, 93, 96, 98, 110] have tackled video synthesis. Most of these works, however, have generated videos either on low-resolution [4, 8, 12, 17, 19, 26, 55, 60, 90, 93, 96, 110] or on comparably small and noisy datasets [11, 88, 106] which were originally proposed to train discriminative models.

Driven by increasing amounts of available compute resources and datasets better suited for generative modeling such as WebVid-10M [7], more competitive approaches have been proposed recently, mainly based on well-scalable, explicit likelihood-based approaches such as diffusion [41, 42, 82] and autoregressive models [94]. Motivated by a lack of available clean video data, all these approaches are leveraging joint image-video training [9, 41, 82, 115] and most methods are grounding their models on pretrained image models [9, 82, 115]. Another commonality between these and most subsequent approaches to (text-to-)video synthesis [29, 97, 99] is the usage of dedicated expert models to generate the actual visual content at a coarse frame rate and to temporally upscale this low-fps video to temporally smooth final outputs at 24–32 fps [9, 41, 82]. Similar to the image domain, diffusion-based approaches can be mainly separated into cascaded approaches [41] following [29, 39] and latent diffusion models [9, 113, 115] translating the approach of Rombach et al. [71] to the video domain. While most of these works aim at learning general motion representation and are consequently trained on large and diverse datasets, another well-recognized branch of diffusion-based video synthesis tackles personalized video generation based on finetuning of pretrained text-to-image models on more narrow datasets tailored to a specific domain [32] or application, partly including non-deep motion priors [113]. Finally, many recent works tackle the task of image-to-video synthesis, where the start frame is already given, and the model has to generate the consecutive frames [32, 97, 113]. Importantly, as shown in our work (see Figure 1) when combined with off-the-shelf text-to-image models, image-to-video models can be used to obtain a full text-(to-image)-to-video pipeline.

**Multi-View Generation** Motivated by their success in 2D image generation, diffusion models have also been used for multi-view generation. Early promising diffusion-based results [2, 16, 49, 62, 101, 116] have mainly been restricted by lacking availability of useful real-world multi-view training data. To address this, more recent works such as Zero-123 [57], MVDream [81], and SyncDreamer [58] propose techniques to adapt and finetune pretrained image generation models such as Stable Diffusion (SD) for multi-view generation, thereby leveraging image priors from SD. One issue with Zero-123 [57] is that the generated multi-views can be inconsistent with respect to each other as they are generated independently with pose-conditioning. Some follow-up works try to address this view-consistency problem by jointly synthesizing the multi-view images. MVDream [81] proposes to jointly generate four views of an object using a shared attention module across images. SyncDreamer [58] proposes to estimate a 3D voxel structure in parallel to the multi-view image diffusion process to maintain consistency across the generated views.

Despite rapid progress in multi-view generation research, these approaches rely on single image generation models such as SD. We believe that our video generative model is a better candidate for the multi-view generation as multi-view images form a specific form of video where the camera is moving around an object. As a result, it is much easier to adapt a video-

generative model for multi-view generation compared to adapting an image-generative model. In addition, the temporal attention layers in our video model naturally assist in the generation of consistent multi-views of an object without needing any explicit 3D structures like in [58].

## C. Data Processing

In this section, we provide more details about our processing pipeline including their outputs on a few public video examples for demonstration purposes.

**Motivation** We start from a large collection of raw video data which is not useful for generative text-video (pre)training [70, 100] because of the following adverse properties: First, in contrast to discriminative approaches to video modeling, generative video models are sensitive to motion inconsistencies such as cuts of which usually many are contained in raw and unprocessed video data, *cf.* Figure 2, left. Moreover, our initial data collection is biased towards still videos as indicated by the peak at zero motion in Figure 2, right. Since generative models trained on this data would obviously learn to generate videos containing cuts and still scenes, this emphasizes the need for cut detection and motion annotations to ensure temporal quality. Another critical ingredient for training generative text-video models are captions - ideally more than one per video [85] - which are well-aligned with the video content. The last essential component for generative video training which we are considering here is the high visual quality of the training examples.

The design of our processing pipeline addresses the above points. Thus, to ensure temporal quality, we detect cuts with a cascaded approach directly after download, clip the videos accordingly, and estimate optical flow for each resulting video clip. After that, we apply three synthetic captioners to every clip and further extract frame-level CLIP similarities to all of these text prompts to be able to filter out outliers. Finally, visual quality at the frame level is assessed by using a CLIP-embeddings-based aesthetics score [80]. We describe each step in more detail in what follows.

Source Video	Cut Detected?	
	w/o cascade	w/ cascade (ours)
	✓	✓
	✓	✓
	✗	✓
	✗	✓

Figure 11. Comparing a common cut detector with our cascaded approach, shows the benefits of our cascaded method: While normal single-fps cut detection can only detect sudden changes in scene, more continuous transitions tend to remain undetected, what is in contrast with our approach which reliably also detects the latter transitions.

**Cascaded Cut Detection.** Similar to previous work [100], we use PySceneDetect<sup>2</sup> to detect cuts in our base video clips. However, as qualitatively shown in Figure 11 we observe many fade-ins and fade-outs between consecutive scenes, which are not detected when running the cut detector at a unique threshold and only native fps. Thus, in contrast to previous work, we apply a cascade of 3 cut detectors which are operating at different frame rates and different thresholds to detect both sudden changes and slow ones such as fades.

<sup>2</sup><https://github.com/Breakthrough/PySceneDetect>

**Keyframe-Aware Clipping.** We clip the videos using FFmpeg [91] directly after cut detection by extracting the timestamps of the keyframes in the source videos and snapping detected cuts onto the closest keyframe timestamp, which does not cross the detected cut. This allows us to quickly extract clips without cuts via seeking and isn't prohibitively slow at scale like inserting new keyframes in each video.

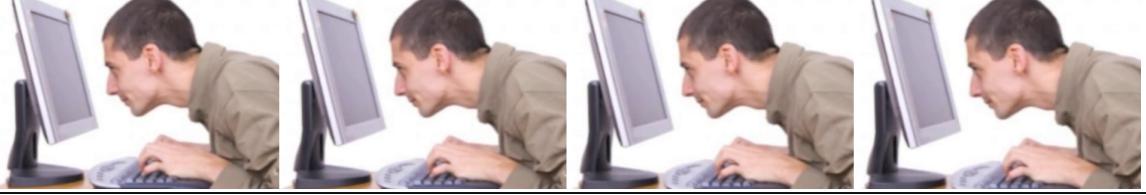
Source Video	Optical Flow Score
	0.043

Figure 12. Examples for a static video. Since such static scenes can have a negative impact on generative video-text (pre-)training, we filter them out.

**Optical Flow.** As motivated in Section 3.1 and Figure 2 it is crucial to provide means for filtering out static scenes. To enable this, we extract dense optical flow maps at 2fps using the OpenCV [48] implementation of the Farnebäck algorithm [24]. To further keep storage size tractable we spatially downscale the flow maps such that the shortest side is at 16px resolution. By averaging these maps over time and spatial coordinates, we further obtain a global motion score for each clip, which we use to filter out static scenes by using a threshold for the minimum required motion, which is chosen as detailed on App. E.2.2. Since this only yields rough approximate, for the final Stage III finetuning, we compute more accurate dense optical flow maps using RAFT [89] at  $800 \times 450$  resolution. The motion scores are then computed similarly. Since the high-quality finetuning data is relatively much smaller than the pretraining dataset, this makes the RAFT-based flow computation tractable.

Source Video	Caption CoCa	Caption VBLIP	Caption LLM
	there is a piece of wood on the floor next to a tape measure .	a person is using a ruler to measure a piece of wood	A person is using a ruler to measure a piece of wood on the floor next to a tape measure.
	two men sitting on a rock near a river . one is holding a stick and the other is holding a pole .	two people are fishing in a river	Two men are fishing in a river. One is holding a stick and the other is holding a pole.

Figure 13. Comparison of various synthetic captioners. We observe that CoCa often captures good spatial details, whereas VBLIP tends to capture temporal details. We use an LLM to combine these two, and experiment with all three types of synthetic captions.

**Synthetic Captioning.** At a million-sample scale, it is not feasible to hand-annotate data points with prompts. Hence we resort to synthetic captioning to extract captions. However in light of recent insights on the importance of caption diversity [85] and taking potential failure cases of these synthetic captioning models into consideration, we extract *three* captions per clip by using i) the image-only captioning model CoCa [65], which describes spatial aspects well, ii) - to also capture temporal aspects - the video-captioner VideoBLIP [109] and iii) to combine these two captions and like that, overcome potential flaws in each of them, a lightweight LLM. Examples of the resulting captions are shown in Figure 13.

**Caption similarities and Aesthetics.** Extracting CLIP [66] image and text representations have proven to be very helpful for data curation in the image domain since computing the cosine similarity between the two allows for assessment of text-image alignment for a given example [80] and thus to filter out examples with erroneous captions. Moreover, it is possible to extract scores for visual aesthetics [80]. Although CLIP is only able to process images, and this consequently is only possible on a single frame level we opt to extract both CLIP-based i) text-image similarities and ii) aesthetics scores of the first, center, and last frames of each video clip. As shown in Section 3.3 and App. E.2.2, using training text-video models on data curated by using these scores improves i) text following abilities and ii) visual quality of the generated samples compared to models trained on unfiltered data.

**Text Detection.** In early experiments, we noticed that models trained on earlier versions of *LVD-F* obtained a tendency to generate videos with excessive amounts of written text depicted which is arguably not a desired feat for a text-to-video model. To this end, we applied the off-the-shelf text-detector CRAFT [5] to annotate the start, middle, and end frames of each clip in our dataset with bounding box information on all written text. Using this information, we filtered out all clips with a total area of detected bounding boxes larger than 7% to construct the final *LVD-F*.



Figure 14. An example of a video with lots of unwanted text. We apply text-detection and annotate bounding boxes around text, and then compute the ratio between the area of all the boxes and the size of the frame.

## D. Model and Implementation Details

### D.1. Diffusion Models

In this section, we give a concise summary of DMs. We make use of the continuous-time DM framework [51, 87]. Let  $p_{\text{data}}(\mathbf{x}_0)$  denote the data distribution and let  $p(\mathbf{x}; \sigma)$  be the distribution obtained by adding i.i.d.  $\sigma^2$ -variance Gaussian noise to the data. Note that for sufficiently large  $\sigma_{\max}$ ,  $p(\mathbf{x}; \sigma_{\max}^2) \approx \mathcal{N}(\mathbf{0}, \sigma_{\max}^2)$ . DM uses this fact and, starting from high variance Gaussian noise  $\mathbf{x}_M \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2)$ , sequentially denoise towards  $\sigma_0 = 0$ . In practice, this iterative refinement process can be implemented through the numerical simulation of the *Probability Flow* ordinary differential equation (ODE) [87]

$$d\mathbf{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t)) dt, \quad (1)$$

where  $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma)$  is the *score function* [46]. DM training reduces to learning a model  $s_{\theta}(\mathbf{x}; \sigma)$  for the score function  $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma)$ . The model can, for example, be parameterized as  $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma) \approx s_{\theta}(\mathbf{x}; \sigma) = (D_{\theta}(\mathbf{x}; \sigma) - \mathbf{x})/\sigma^2$  [51], where  $D_{\theta}$  is a learnable *denoiser* that tries to predict the clean  $\mathbf{x}_0$ . The denoiser  $D_{\theta}$  is trained via *denoising score matching* (DSM)

$$\mathbb{E}_{(\mathbf{x}_0, \mathbf{c}) \sim p_{\text{data}}(\mathbf{x}_0, \mathbf{c}), (\sigma, \mathbf{n}) \sim p(\sigma, \mathbf{n})} [\lambda_{\sigma} \|D_{\theta}(\mathbf{x}_0 + \mathbf{n}; \sigma, \mathbf{c}) - \mathbf{x}_0\|_2^2], \quad (2)$$

where  $p(\sigma, \mathbf{n}) = p(\sigma) \mathcal{N}(\mathbf{n}; \mathbf{0}, \sigma^2)$ ,  $p(\sigma)$  can be a probability distribution or density over noise levels  $\sigma$ . It is both possible to use a discrete set or a continuous range of noise levels. In this work, we use both options, which we further specify in App. D.2.

$\lambda_{\sigma}: \mathbb{R}_+ \rightarrow \mathbb{R}_+$  is a weighting function, and  $\mathbf{c}$  is an arbitrary conditioning signal. In this work, we follow the EDM-preconditioning framework [51], parameterizing the learnable denoiser  $D_{\theta}$  as

$$D_{\theta}(\mathbf{x}; \sigma) = c_{\text{skip}}(\sigma)\mathbf{x} + c_{\text{out}}(\sigma)F_{\theta}(c_{\text{in}}(\sigma)\mathbf{x}; c_{\text{noise}}(\sigma)), \quad (3)$$

where  $F_\theta$  is the network to be trained.

**Classifier-free guidance.** Classifier-free guidance [37] is a method used to guide the iterative refinement process of a DM towards a conditioning signal  $\mathbf{c}$ . The main idea is to mix the predictions of a conditional and an unconditional model

$$D^w(\mathbf{x}; \sigma, \mathbf{c}) = wD(\mathbf{x}; \sigma, \mathbf{c}) - (w - 1)D(\mathbf{x}; \sigma), \quad (4)$$

where  $w \geq 0$  is the *guidance strength*. The unconditional model can be trained jointly alongside the conditional model in a single network by randomly replacing the conditional signal  $\mathbf{c}$  with a null embedding in Eq. (2), e.g., 10% of the time [37]. In this work, we use classifier-free guidance, for example, to guide video generation toward text conditioning.

## D.2. Base Model Training and Architecture

As discussed in , we start the publicly available *Stable Diffusion* 2.1 [71] (SD 2.1) model. In the EDM-framework [51], SD 2.1 has the following preconditioning functions:

$$c_{\text{skip}}^{\text{SD2.1}}(\sigma) = 1, \quad (5)$$

$$c_{\text{out}}^{\text{SD2.1}}(\sigma) = -\sigma, \quad (6)$$

$$c_{\text{in}}^{\text{SD2.1}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + 1}}, \quad (7)$$

$$c_{\text{noise}}^{\text{SD2.1}}(\sigma) = \arg \min_{j \in [1000]} (\sigma - \sigma_j), \quad (8)$$

$$(9)$$

where  $\sigma_{j+1} > \sigma_j$ . The distribution over noise levels  $p(\sigma)$  used for the original SD 2.1. training is a uniform distribution over the 1000 *discrete* noise levels  $\{\sigma_j\}_{j \in [1000]}$ . One issue with the training of SD 2.1 (and in particular its noise distribution  $p(\sigma)$ ) is that even for the maximum discrete noise level  $\sigma_{1000}$  the *signal-to-noise ratio* [53] is still relatively high which results in issues when, for example, generating very dark images [34, 56]. Guttenberg and CrossLabs [34] proposed *offset noise*, a modification of the training objective in Eq. (2) by making  $p(\mathbf{n} | \sigma)$  non-isotropic Gaussian. In this work, we instead opt to modify the preconditioning functions and distribution over training noise levels altogether.

**Image model finetuning.** We replace the above preconditioning functions with

$$c_{\text{skip}}(\sigma) = (\sigma^2 + 1)^{-1}, \quad (10)$$

$$c_{\text{out}}(\sigma) = \frac{-\sigma}{\sqrt{\sigma^2 + 1}}, \quad (11)$$

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + 1}}, \quad (12)$$

$$c_{\text{noise}}(\sigma) = 0.25 \log \sigma, \quad (13)$$

$$(14)$$

which can be recovered in the EDM framework [51] by setting  $\sigma_{\text{data}} = 1$ ; the preconditioning functions were originally proposed in [79]. We also use the noise distribution and weighting function proposed in Karras et al. [51], namely  $\log \sigma \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$  and  $\lambda(\sigma) = (1 + \sigma^2)\sigma^{-2}$ , with  $P_{\text{mean}} = -1.2$  and  $P_{\text{std}} = 1$ . We then finetune the neural network backbone  $F_\theta$  of SD2.1 for 31k iterations using this setup. For the first 1k iterations, we freeze all parameters of  $F_\theta$  except for the time-embedding layer and train on SD2.1’s original training resolution of  $512 \times 512$ . This allows the model to adapt to the new preconditioning functions without unnecessarily modifying the internal representations of  $F_\theta$  too much. Afterward, we train all layers of  $F_\theta$  for another 30k iterations on images of size  $256 \times 384$ , which is the resolution used in the initial stage of video pretraining.

**Video pretraining.** We use the resulting model as the image backbone of our video model. We then insert temporal convolution and attention layers. In particular, we follow the exact setup from [9], inserting a total of 656M new parameters into the UNet bumping its total size (spatial and temporal layers) to 1521M parameters. We then train the resulting UNet on 14 frames on resolution  $256 \times 384$  for 150k iters using AdamW [59] with learning rate  $10^{-4}$  and a batch size of 1536. We train the model for classifier-free guidance [36] and drop out the text-conditioning 15% of the time. Afterward, we increase the spatial resolution to  $320 \times 576$  and train for an additional 100k iterations, using the same settings as for the lower-resolution training except for a reduced batch size of 768 and a shift of the noise distribution towards more noise, in

particular, we increase  $P_{\text{mean}} = 0$ . During training, the base model and the high-resolution Text/Image-to-Video models are all conditioned on the input video’s frame rate and motion score. This allows us to vary the amount of motion in a generated video at inference time.

### D.3. High-Resolution Text-to-Video Model

We finetune our base model on a high-quality dataset of  $\sim 1\text{M}$  samples at resolution  $576 \times 1024$ . We train for  $50k$  iterations at a batch size of 768, learning rate  $3 \times 10^{-5}$ , and set  $P_{\text{mean}} = 0.5$  and  $P_{\text{std}} = 1.4$ . Additionally, we track an exponential moving average of the weights at a decay rate of 0.9999. The final checkpoint is chosen using a combination of visual inspection and human evaluation.

### D.4. High-Resolution Image-to-Video Model

We can finetune our base text-to-video model for the image-to-video task. In particular, during training, we use one additional frame on which the model is conditioned. We do not use text-conditioning but rather replace text embeddings fed into the base model with the CLIP image embedding of the conditioning frame. Additionally, we concatenate a noise-augmented [39] version of the conditioning frame channel-wise to the input of the UNet [73]. In particular, we add a small amount of noise of strength  $\log \sigma \sim \mathcal{N}(-3.0, 0.5^2)$  to the conditioning frame and then feed it through the standard SD 2.1 encoder. The mean of the encoder distribution is then concatenated to the input of the UNet (copied across the time axis). Initially, we finetune our base model for the image-to-video task on the base resolution ( $320 \times 576$ ) for  $50k$  iterations using a batch size of 768 and learning rate  $3 \times 10^{-5}$ . Since the conditioning signal is very strong, we again shift the noise distribution towards more noise, i.e.,  $P_{\text{mean}} = 0.7$  and  $P_{\text{std}} = 1.6$ . Afterwards, we fintune the base image-to-video model on a high-quality dataset of  $\sim 1\text{M}$  samples at  $576 \times 1024$  resolution. We train two versions: one to generate 14 frames and one to generate 25 frames. We train both models for  $50k$  iterations at a batch size of 768, learning rate  $3 \times 10^{-5}$ , and set  $P_{\text{mean}} = 1.0$  and  $P_{\text{std}} = 1.6$ . Additionally, we track an exponential moving average of the weights at a decay rate of 0.9999. The final checkpoints are chosen using a combination of visual inspection and human evaluation.

#### D.4.1 Linearly Increasing Guidance

We occasionally found that standard vanilla classifier-free guidance [36] (see Eq. (4)) can lead to artifacts: too little guidance may result in inconsistency with the conditioning frame while too much guidance can result in oversaturation. Instead of using a constant guidance scale, we found it helpful to linearly increase the guidance scale across the frame axis (from small to high). A PyTorch implementation of this novel technique can be found in Figure 15.

#### D.4.2 Camera Motion LoRA

To facilitate controlled camera motion in image-to-video generation, we train a variety of *camera motion LoRAs* within the temporal attention blocks of our model [32]. In particular, we train low-rank matrices of rank 16 for 5k iterations. Additional samples can be found in Figure 20.

### D.5. Interpolation Model Details

Similar to the text-to-video and image-to-video models, we finetune our interpolation model starting from the base text-to-video model, *cf.* App. D.2. To enable interpolation, we reduce the number of output frames from 14 to 5, of which we use the first and last as conditioning frames, which we feed to the UNet [73] backbone of our model via the concat-conditioning-mechanism [71]. To this end, we embed these frames into the latent space of our autoencoder, resulting in two image encodings  $z_s, z_e \in \mathbb{R}^{c \times h \times w}$ , where  $c = 4, h = 52, w = 128$ . To form a latent frame sequence that is of the same shape as the noise input of the UNet, *i.e.*  $\mathbb{R}^{5 \times c \times h \times w}$ , we use a learned mask embedding  $z_m \in \mathbb{R}^{c \times h \times w}$  and form a latent sequence  $z = \{z_s, z_m, z_m, z_m, z_e\} \in \mathbb{R}^{5 \times c \times h \times w}$ . We concatenate this sequence channel-wise with the noise input and additionally with a binary mask where 1 indicates the presence of a conditioning frame and 0 that of a mask embedding. The final input for the UNet is thus of shape  $(5, 9, 52, 128)$ . In line with previous work [9, 41, 82], we use noise augmentation for the two conditioning frames, which we apply in the latent space. Moreover, we replace the CLIP text representation for the crossattention conditioning with the corresponding CLIP image representation of the start frame and end frame, which we concatenate to form a conditioning sequence of length 2.

We train the model on our high-quality dataset at spatial resolution  $576 \times 1024$  using AdamW [59] with a learning rate of  $10^{-4}$  in combination with exponential moving averaging at decay rate 0.9999 and use a shifted noise schedule with

```

1 import torch
2 from einops import rearrange, repeat
3
4
5 def append_dims(x: torch.Tensor, target_dims: int) -> torch.Tensor:
6     """Appends dimensions to the end of a tensor until it has target_dims dimensions."""
7     dims_to_append = target_dims - x.ndim
8     if dims_to_append < 0:
9         raise ValueError(
10             f"input has {x.ndim} dims but target_dims is {target_dims}, which is less"
11         )
12     return x[(..., ) + (None,) * dims_to_append]
13
14
15 class LinearPredictionGuider:
16     def __init__(self,
17                  max_scale: float,
18                  num_frames: int,
19                  min_scale: float = 1.0,
20                  ) :
21         self.min_scale = min_scale
22         self.max_scale = max_scale
23         self.num_frames = num_frames
24         self.scale = torch.linspace(min_scale, max_scale, num_frames).unsqueeze(0)
25
26     def __call__(self, x: torch.Tensor, sigma: float) -> torch.Tensor:
27         x_u, x_c = x.chunk(2)
28
29         x_u = rearrange(x_u, "(b t) ... -> b t ...", t=self.num_frames)
30         x_c = rearrange(x_c, "(b t) ... -> b t ...", t=self.num_frames)
31         scale = repeat(self.scale, "1 t -> b t", b=x_u.shape[0])
32         scale = append_dims(scale, x_u.ndim).to(x_u.device)
33
34         return rearrange(x_u + scale * (x_c - x_u), "b t ... -> (b t) ...")
35

```

Figure 15. PyTorch code for our novel *linearly increasing guidance* technique.

$P_{\text{mean}} = 1$  and  $P_{\text{std}} = 1.2$ . Surprisingly, we find this model, which we train with a comparably small batch size of 256, to converge extremely fast and to yield consistent and smooth outputs after only 10k iterations. We take this as another evidence of the usefulness of the learned motion representation our base text-to-video model has learned.

## D.6. Multi-view generation

We finetuned the high-resolution image-to-video model on our specific rendering of the Objaverse dataset. We render 21 frames per orbit of an object in the dataset at  $576 \times 576$  resolution and finetune the 25-frame Image-to-Video model to generate these 21 frames. We feed one view of the object as the image condition. In addition, we feed the elevation of the camera as conditioning to the model. We first pass the elevation through a timestep embedding layer that embeds the sine and cosine of the elevation angle at various frequencies and concatenates them into a vector. This vector is finally concatenated to the overall vector condition of the UNet.

We trained for  $12k$  iterations with a total batch size of 16 across 8 A100 GPUs of 80GB VRAM at a learning rate of  $1 \times 10^{-5}$ .

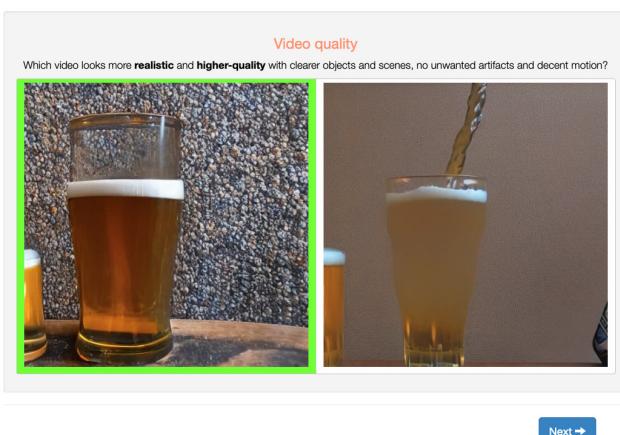
## E. Experiment Details

### E.1. Details on Human Preference Assessment

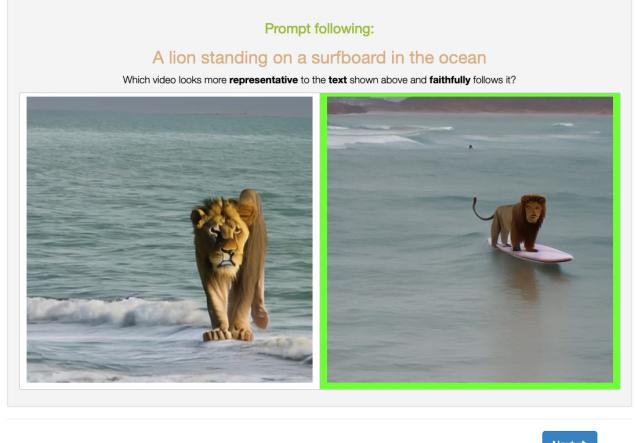
For most of the evaluation conducted in this paper, we employ human evaluation as we observed it to contain the most reliable signal. For text-to-video tasks and all ablations conducted for the base model, we generate video samples from a list of 64 test prompts. We then employ human annotators to collect preference data on two axes: i) visual quality and ii) prompt following. More details on how the study was conducted App. E.1.1 and the rankings computed App. E.1.2 are listed below.

### E.1.1 Experimental Setup

Given all models in one ablation axis (*e.g.* four models of varying aesthetic or motion scores), we compare each prompt for each pair of models (1v1). For every such comparison, we collect on average three votes per task from different annotators, i.e., three each for visual quality and prompt following, respectively. Performing a complete assessment between all pairwise comparisons gives us robust and reliable signals on model performance trends and the effect of varying thresholds. Sample interfaces that the annotators interact with are shown in Figure 16. The order of prompts and the order between models are fully randomized. Frequent attention checks are in place to ensure data quality.



(a) Sample instructions for evaluating visual quality of videos.



(b) Sample instructions for evaluating the prompt following of videos.

Figure 16. Our human evaluation framework, as seen by the annotators. The prompt & task order and model choices are fully randomized.

### E.1.2 Elo Score Calculation

To calculate rankings when comparing more than two models based on 1v1 comparisons as outlined in App. E.1.1, we use Elo Scores (higher-is-better) [21], which were originally proposed as a scoring method for chess players but have more recently also been applied to compare instruction-tuned generative LLMs [3, 6]. For a set of competing players with initial ratings  $R_{\text{init}}$  participating in a series of zero-sum games, the Elo rating system updates the ratings of the two players involved in a particular game based on the expected and actual outcome of that game. Before the game with two players with ratings  $R_1$  and  $R_2$ , the expected outcome for the two players is calculated as

$$E_1 = \frac{1}{1 + 10^{\frac{R_2 - R_1}{400}}}, \quad (15)$$

$$E_2 = \frac{1}{1 + 10^{\frac{R_1 - R_2}{400}}}. \quad (16)$$

After observing the result of the game, the ratings  $R_i$  are updated via the rule

$$R'_i = R_i + K \cdot (S_i - E_i), \quad i \in \{1, 2\} \quad (17)$$

where  $S_i$  indicates the outcome of the match for player  $i$ . In our case, we have  $S_i = 1$  if player  $i$  wins and  $S_i = 0$  if player  $i$  loses. The constant  $K$  can be seen as weight emphasizing more recent games. We choose  $K = 1$  and bootstrap the final Elo ranking for a given series of comparisons based on 1000 individual Elo ranking calculations in a randomly shuffled order. Before comparing the models, we choose the start rating for every model as  $R_{\text{init}} = 1000$ .

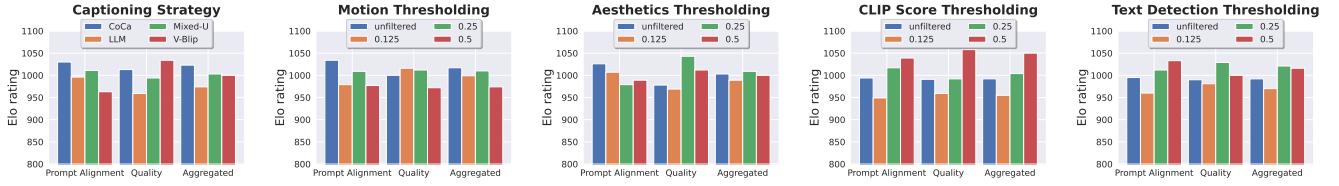


Figure 17. Results of the dedicated experiments conducted to identify most useful filtering thresholds for each ablation axis. For of these ablation studies we train four identical models using the architecture detailed in App. E.2.2 on different subset of *LVD-10M*, which we create by systematically increasing the thresholds which corresponds to filter out more and more examples.

## E.2. Details on Experiments from Section 3

### E.2.1 Architectural Details

Architecturally, all models trained for the presented analysis in Section 3 are identical. To insert create a temporal UNet [73] based on an existing spatial model, we follow Blattmann et al. [9] and add temporal convolution and (cross-)attention layers after each corresponding spatial layer. As a base 2D-UNet, we use the architecture from *Stable Diffusion 2.1*, whose weights we further use to initialize the spatial layers for all runs except the second one presented in Figure 3a, where we intentionally skip this initialization to create a baseline for demonstrating the effect of image-pretraining. Unlike Blattmann et al. [9], we train all layers, including the spatial ones, and do not freeze the spatial layers after initialization. All models are trained with the AdamW [59] optimizer with a learning rate of  $1.e - 4$  and a batch size of 256. Moreover, in contrast to our models from Section 4, we do not translate the noise process to continuous time but use the standard linear schedule used in *Stable Diffusion 2.1*, including offset noise [34], in combination with the v-parameterization [37]. We omit the text-conditioning in 10% of the cases to enable classifier-free guidance [37] during inference. To generate samples for the evaluations, we use 50 steps of the deterministic DDIM sampler [86] with a classifier guidance scale of 12 for all models.

### E.2.2 Calibrating Filtering Thresholds

Here, we present the outcomes of our study on filtering thresholds presented in Section 3.3. As stated there, we conduct experiments for the optimal filtering threshold for each type of annotation while not filtering for any other types. The only difference here is our assessment of the most suitable captioning method, where we simply compare all used captioning methods. We train each model on videos consisting of 8 frames at resolution  $256 \times 256$  for exactly 40k steps with a batch size of 256, roughly corresponding to 10M training examples seen during training. For evaluation, we create samples based on 64 pre-selected prompts for each model and conduct a human preference study as detailed in App. E.1. Figure 17 shows the ranking results of these human preference studies for each annotation axis for spatiotemporal sample quality and prompt following. Additionally, we show an averaged ‘aggregated’ score.

For *captioning*, we see that - surprisingly - the captions generated by the simple clip-based image captioning method CoCa of Yu et al. [108] clearly have the most beneficial influence on the model. However, since recent research recommends using more than one caption per training example, we sample one of the three distinct captions during training. We nonetheless reflect the outcome of this experiment by shifting the captioning sampling distribution towards CoCa captions by using  $p_{CoCa} = 0.5$ ;  $p_{v-BLIP} = 0.25$ ;  $p_{LLM} = 0.25$ .

For *motion filtering*, we choose to filter out 25% of the most static examples. However, the aggregated preference score of the model trained with this filtering method does not rank as high in human preference as the non-filtered score. The rationale behind this is that non-filtered ranks best primarily because it ranks best in the category ‘prompt following’ which is less important than the ‘quality’ category when assessing the effect of motion filtering. Thus, we choose the 25% threshold, as mentioned above, since it achieves both competitive performances in ‘prompt following’ and ‘quality’.

For *aesthetics filtering*, where, as for motion thresholding, the ‘quality’ category is more important than the ‘prompt following’-category, we choose to filter out the 25 % with the lowest aesthetics score, while for *CLIP-score thresholding* we omit even 50% since the model trained with the corresponding threshold is performing best. Finally, we filter out the 25% of samples with the largest text area covering the videos since it ranks highest both in the ‘quality’ category and on average.

Using these filtering methods, we reduce the size of *LVD* by more than a factor of 3, cf. Tab. 1, but obtain a much cleaner dataset as shown in Section 3. For the remaining experiments in Section 3.3, we use the identical architecture and hyperparameters as stated above. We only vary the dataset as detailed in Section 3.3.

### E.2.3 Finetuning Experiments

For the finetuning experiments shown in Section 3.4, we again follow the architecture, training hyperparameters, and sampling procedure stated at the beginning of this section. The only notable differences are the exchange of the dataset and the increase in resolution from the pretraining resolution  $256 \times 256$  to  $512 \times 512$  while still generating videos consisting of 8 frames. We train all models presented in this section for 50k steps.

## E.3. Human Eval vs SOTA

For comparison of our image-to-video model with state-of-the-art models like Gen-2 [74] and Pika [54], we randomly choose 64 conditioning images generated from a  $1024 \times 576$  finetune of SDXL [64]. We employ the same framework as in App. E.1.1 to evaluate and compare the visual quality generated samples with other models.

For Gen-2, we sample the image-to-video model from the web UI. We fixed the same seed of 23, used the default motion value of 5 (on a scale of 10), and turned on the “Interpolate” and “Remove watermark” features. This results in 4-second samples at  $1408 \times 768$ . We then resize the shorter side to yield  $1056 \times 576$  and perform a center-crop to match our resolution of  $1024 \times 576$ . For our model, we sample our 25-frame image-to-video finetune to give 28 frames and also interpolate using our interpolation model to yield samples of 3.89 seconds at 28 FPS. We crop the Gen-2 samples to 3.89 seconds to avoid biasing the annotators.

For Pika, we sample the image-to-video model from the Discord bot. We fixed the same seed of 23, used the motion value of 2 (on a scale of 0-4), and specified a 16:9 aspect ratio. This results in 3-second samples at  $1024 \times 576$ , which matches our resolution. For our model, we sample our 25-frame image-to-video finetune to give 28 frames and also interpolate using our interpolation model to yield samples of 3.89 seconds at 28 FPS. We crop our samples to 3 seconds to match Pika and avoid biasing the annotators. Since Pika samples have a small “Pika Labs” watermark in the bottom right, we pad that region with black pixels for both Pika and our samples to also avoid bias.

## E.4. UCF101 FVD

This section describes the zero-shot UCF101 FVD computation of our base text-to-video model. The UCF101 dataset [88] consists of 13,320 video clips, which are classified into 101 action categories. All videos are of frame rate 25 FPS and resolution  $240 \times 320$ . To compute FVD, we generate 13,320 videos (16 frames at 25 FPS, classifier-free guidance with scale  $w = 7$ ) using the same distribution of action categories, that is, for example, 140 videos of “TableTennisShot”, 105 videos of “PlayingPiano”, etc. We condition the model directly on the action category (“TableTennisShot”, “PlayingPiano”, etc.) and do not use any text modification. Our samples are generated at our model’s native resolution  $320 \times 576$  (16 frames), and we downsample to  $240 \times 432$  using bilinear interpolation with antialiasing, followed by a center crop to  $240 \times 320$ . We extract features using a pretrained I3D action classification model [11], in particular we are using a torchscript<sup>3</sup> provided by Brooks et al. [10].

## E.5. Additional Samples

Here, we show additional samples for the models introduced in App. D.2 and Secs. 4.2, 4.3 and 4.5.

### E.5.1 Additional Text-to-Video Samples

In Figure 18, we show additional samples from our text-to-video model introduced in Section 4.2.

### E.5.2 Additional Image-to-Video Samples

In Figure 19, we show additional samples from our image-to-video model introduced in Section 4.3.

### E.5.3 Additional Camera Motion LoRA Samples

In Figure 20, we show additional samples for our motion LoRA’s tuned for camera control as presented in Section 4.3.1.



Figure 18. Additional Text-to-Video samples. Captions from top to bottom: “A hiker is reaching the summit of a mountain, taking in the breathtaking panoramic view of nature.”, “A unicorn in a magical grove, extremely detailed.”, “Shoveling snow”, “A beautiful fluffy domestic hen sitting on white eggs in a brown nest, eggs are under the hen.”, and “A boat sailing leisurely along the Seine River with the Eiffel Tower in background by Vincent van Gogh”.

#### E.5.4 Temporal Prompting via Temporal Cross-Attention Layers

Our architecture follows Blattmann et al. [9], who introduced dedicated temporal cross-attention layers, which are used interleaved with the spatial cross-attention layers of the standard 2D-UNet [18, 38]. During probing our Text-to-Video model from Section 4.2, we noticed that it is possible to independently prompt the model spatially and temporally by using different text-prompts as inputs for the spatial and temporal cross-attention conditionings, see Figure 21. To achieve this, we use a dedicated spatial prompt to describe the general content of the scene to be depicted while the motion of that scene is fed to the model via a separate temporal prompt, which is the input to the temporal cross-attention layers. We provide an example of these first experiments indicating this implicit disentanglement of motion and content in Figure 21, where we show that varying the temporal prompt while fixing random seed and spatial prompt leads to spatially similar scenes that obtain global motion properties following the temporal prompt.

#### E.5.5 Additional Samples on Multi-View Synthesis

In Figures 22 to 25, we show additional visual examples for SVD-MV, trained on our renderings of Objaverse and MVIImageNet datasets as described in Section 4.5.

---

<sup>3</sup>[https://www.dropbox.com/s/ge9e5ujwgetkms/i3d\\_torchscript.pt](https://www.dropbox.com/s/ge9e5ujwgetkms/i3d_torchscript.pt) with keyword arguments `rescale=True`, `resize=True`, `return_features=True`.



Figure 19. Additional Image-to-Video samples. Leftmost frame is used for conditioning.



Figure 20. Additional Image-to-Video samples with camera motion LoRAs (conditioned on leftmost frame). The first, second, and thirs rows correspond to *horizontal*, *static*, *zooming*, respectively.

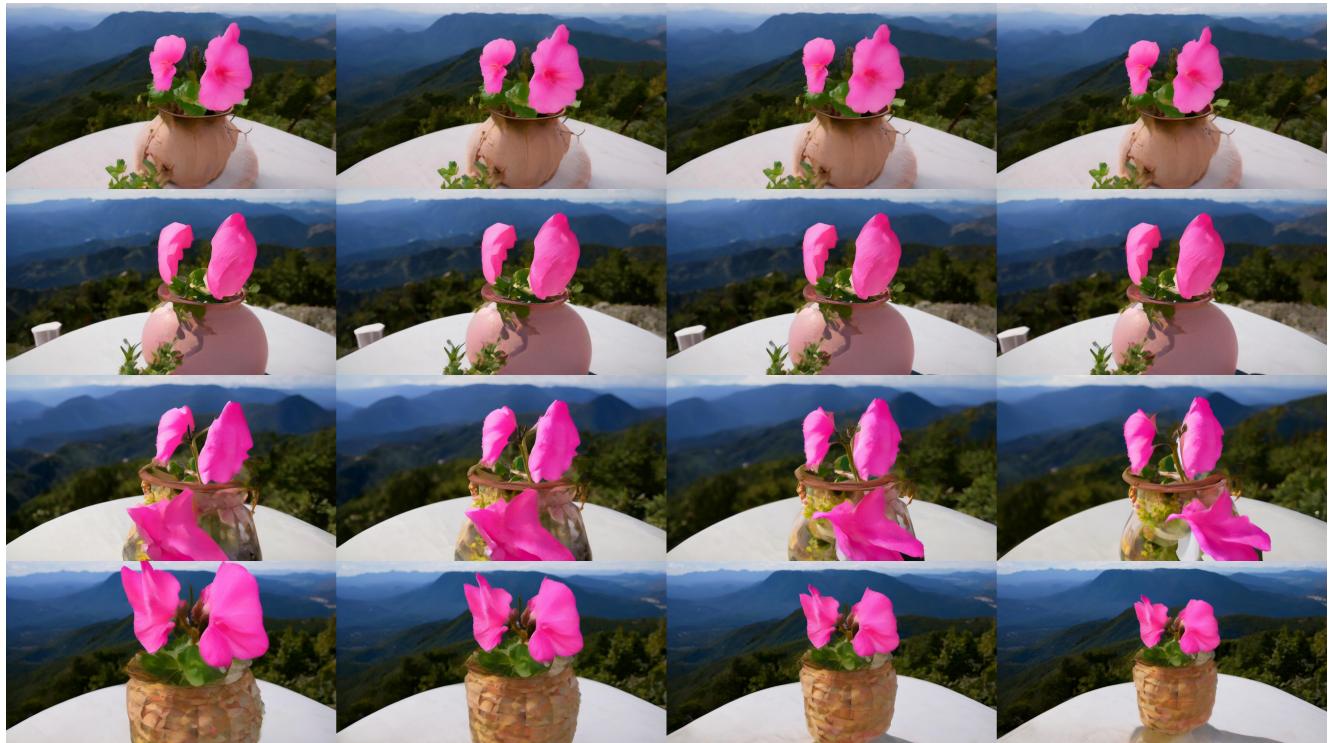


Figure 21. Text-to-video samples using the prompt “Flowers in a pot in front of a mountainside” (for spatial cross-attention). We adjust the camera control by replacing the prompt in the temporal attention using “”, “panning”, “rotating”, and “zooming” (from top to bottom). While not being trained for this inference task, the model performs surprisingly well.

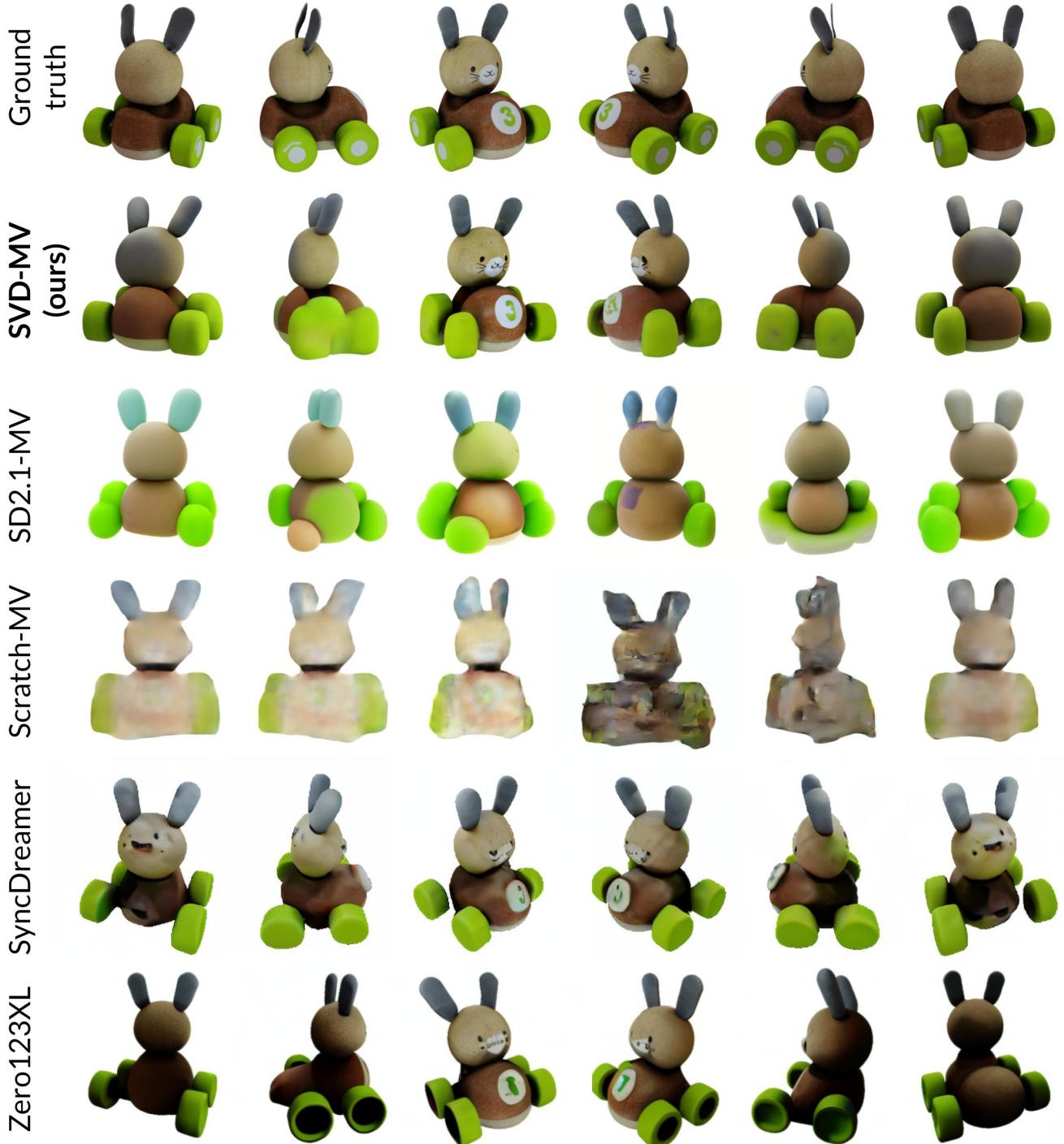


Figure 22. Additional image-to-multi-view generation samples from GSO test dataset, using our SVD-MV model trained on Objaverse, and comparison with other methods.



Figure 23. Additional image-to-multi-view generation samples from GSO test dataset, using our SVD-MV model trained on Objaverse



Figure 24. Text-to-image-to-multi-view generation samples: text to image using SDXL with the prompt "Centered 3D model of a cute anthropomorphic sunflower figure (plain background, unreal engine render 4k)", and image-to-multi-view using our SVD-MV model trained on Objaverse

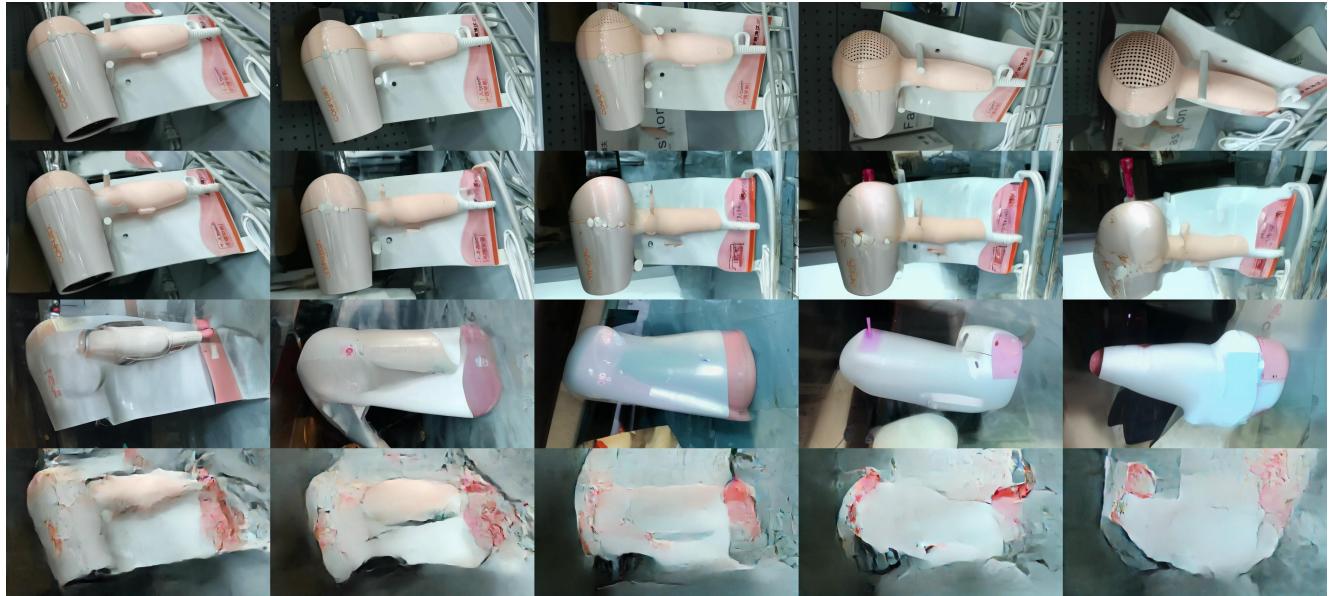


Figure 25. Additional multi-view generation samples from MVI dataset, using our SVD-MV model trained on MVImgNet, and comparison with other methods. Top row is ground truth frames, second row is sample frames from SVD-MV (ours), third row is from SD2.1-MV, bottom row is from Scratch-MV