SinFusion: Training Diffusion Models on a Single Image or Video

Yaniv Nikankin* Niv Haim* Michal Irani
Weizmann Institute of Science, Rehovot, Israel

Project page: https://yanivnik.github.io/sinfusion

Figure 1. **Diverse video generation.** *Top Row* shows consecutive frames from a single training video. *Second/Third Rows* each show a set of consecutive frames generated by our single video DDPM. *For the full videos we refer the reader to our project page*.

Abstract

Diffusion models exhibited tremendous progress in image and video generation, exceeding GANs in quality and diversity. However, they are usually trained on very large datasets and are not naturally adapted to manipulate a given input image or video. In this paper we show how this can be resolved by training a diffusion model on a single input image or video. Our image/video-specific diffusion model (SinFusion) learns the appearance and dynamics of the single image or video, while utilizing the conditioning capabilities of diffusion models. It can solve a wide array of image/video-specific manipulation tasks. In particular, our model can learn from few frames the motion and dynamics of a single input video. It can then generate diverse new video samples of the same dynamic scene, extrapolate short videos into long ones (both forward and backward in time) and perform video upsampling. When trained on a single image, our model shows comparable performance and capabilities to previous single-image models in various image manipulation tasks.

*Equal contribution.

1. Introduction

Not long ago, generative adversarial networks (GANs [17]) ruled the field of generative models, with seminal works like StyleGAN [32, 33, 34], BigGAN [7] etc. [42, 66]. Diffusion models (**DMs**) [24, 53, 55] have gained the lead in the last years, surpassing GANs by image quality and diversity [12] and becoming the leading method in many vision tasks like text-to-image generation, super-resolution and many more [26, 31, 39, 40, 43, 46, 47, 54] (see surveys [8, 10]). Recent works also demonstrate the effectiveness of DMs for video and text-to-video generation [25, 27, 51, 59].

DMs are trained on massive datasets and as such, these models are very large and resource demanding. Applying their capabilities to edit or manipulate a specific input provided by the user is non-trivial and requires careful manipulation and fine-tuning [2, 16, 35, 45, 57].

In this work we propose a framework for training diffusion models on a single input image or video - *"SinFusion"*. We harness the success and high-quality of DMs at image synthesis, to single-image/video tasks. Once trained, SinFusion can generate new image/video samples with similar

appearance and dynamics to the original input and perform various editing and manipulation tasks. In the video case, SinFusion exhibits impressive generalization capabilities by coherently extrapolating an input video far into the future (or past). This is learned from very few frames (mostly 2-3 dozens, but is already apparent for fewer frames).

We demonstrate the applicability of SinFusion to single-image/video tasks. For images we show diverse generation of new images from a single image, image editing, generation from sketch and visual summary. For videos we show diverse generation of new videos (from a single video), video extrapolation (both forward and backward in time) and video upsampling. Such capabilities were not shown by previous video-specific generative models [19, 20], while existing diffusion models for video generation [27, 65] are not designed for handling a single input video.

Our framework builds on top the commonly used DDPM [24] architecture, but introduce several important modifications that are essential for allowing it to train on a single image/video.

Our main contributions are as follows:

- We present, for the first time, diffusion models trained on a single image/video, adapted to single-image/video tasks.
- A single unified framework for a large variety of single-image and single-video tasks.
- Unlike general large-scale diffusion models, SinFusion provides capabilities to edit and manipulate a real input video. These include: diverse video generation, temporal upsampling and video extrapolation (both forward/backward in time).
- SinFusion provides new capabilities and tasks which are not realizable by current single-video generative models - in particular, autoregressive video generation with impressive motion generalization capabilities.
- We propose a new set of evaluations for diverse video generation from a single video.

2. Related Work

Our work lies in the intersection of several fields: generative models trained on a single image or video, manipulation of a real input image/video, diffusion models and methods for image/video generation in general. Here we briefly mention the main achievements in each field and their relation (and difference) from our proposed approach.

Video Generation. In general, video generation is a broad field of research including many areas such as video GANs [9, 52, 56, 61], video-to-video translation [6, 63] or autoregressive prediction models [3, 5, 11, 58], to name a few. Diffusion models for video generation are fairly recent and most rely on DDPM [24] framework for image generation, extended to handle videos. RVD [65] tackles video prediction by conditioning the generative process on recurrent neural networks. RaMViD [28] and MCVD [60] train an autoregressive model conditioned on previous frames for video prediction and infilling using masking mechanisms. VDM [27] introduces unconditional video generation by modifying the Conv2D layers in the basic DDPM UNet to Conv3D, as well as autoregressive generation. Imagen-Video [25] extends VDM to text-to-video and also include spatio-temporal superresolution conditioned on upsampled versions of smaller scales. FDM [21] modifies DDPM to include temporal attention mechanism and can be conditioned on any number of previous frames. The above-mentioned methods can synthesize beautiful videos. However, none of them can modify or manipulate an existing input video provided by the user. Also, all these models are trained on large datasets, some with ridiculous amounts of compute power (VDM using 128-256 TPUs) whereas our training time is much more affordable (a few hours per video on a single GPU).

Generation from a Single Image. Generative models trained on a single image aim to generate new diverse samples, similar in appearance to the image/video on which they were trained. Most notably, SinGAN [48] and InGAN[49] trained multi-scale GANs to learn the distribution of patches in an image. They showed its applicability to diverse random generation from a single image, as well as a variety of other image synthesis applications (inpainting, style transfer, etc.). Their results are usually better suited to synthesis from a single image than models trained on large collection of data. More recently, GPNN [18] showed that most image synthesis tasks proposed by single-image GAN-based models [23, 48, 49] can be solved by classical non-parametric patch nearest-neighbour methods [13, 14, 50], and achieve outputs of higher quality while reducing generation time by orders of magnitude. However, nearest-neighbour methods have a very limited notion of generalization, and are therefore limited to tasks where it is natural to ”copy” parts of the input. In this respect, learning based methods like SinGAN [48] still offer applicability like shown in the tasks of harmonization or animation.

Generation from a Single Video. Similar to the image domain, extensions of SinGAN [48] to generation from a single *video* were proposed [1, 19], generating diverse new videos of similar appearance and dynamics to the input video. These too, were outperformed by patch nearest-neighbour methods [20] in both output quality and speed. However, these video-based nearest-neighbour methods suffer from drawbacks similar to the image case. While

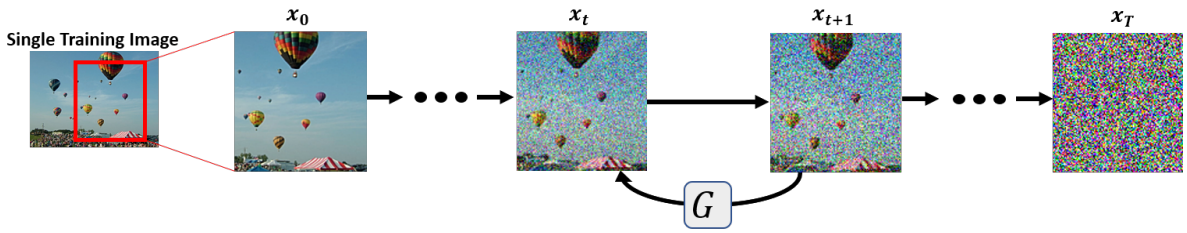


Figure 2. **Single Image DDPM.** Our single-image DDPM trains on large crops from a single image. It learns to remove noise from noisy crops, and, at inference, can generate diverse samples with similar structure and appearance to the training image.

the generated samples are of high quality and look realistic, the main reason for this is that the samples are essentially copies of parts of the original video stitched together. They fail to exhibit motion generalization capabilities. None of the above-mentioned methods can handle input videos longer than a few dozens frames. Single-video GAN based methods are limited in compute time (e.g., HP-VAE-GAN [19] takes 8 days to train on a short video of 13 frames), whereas VGPNN [20] is limited in memory (since each space-time patch in the output video searches for its nearest-neighbor space-time patch in the entire input video, at each iteration). In contrast, our method can handle any length of input video. While it can generalize well from just a few frames, it can also easily train on a long input video at a fixed and very small memory print, and at reasonable compute time (a few hours per video).

Reference Image Manipulation with Large Generative Models. One of the practical application of generative models trained on large datasets is their strong generalization capabilities for semantic image editing, often obtained via latent space interpolation [7, 33, 42]. Applying these capabilities to an existing reference image was mostly achieved by GAN "inversion" techniques [64], and very recently by fine-tuning large diffusion models [2, 16, 35, 45, 57]. However, to the best of our knowledge, there are no existing large-scale models to-date which can manipulate an existing input reference video.

3. Perliminaries: Overview of DDPM

Denoising diffusion probabilistic models (DDPM) [24, 53] are a class of generative models that can learn to convert unstructured noise to samples from a given distribution, by performing an iterative process of removing small amounts of Gaussian noise at each step. Since our method heavily relies on DDPM, we provide here a very brief overview of DDPM and it's basics.

To train a DDPM, an input image x_0 is sampled, and small portions of gaussian noise ϵ are gradually added to it in a parameter-free *forward* process, resulting in a noisy image x_t. The forward process can be written as:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{1}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$, $\beta_t \in (0,1)$ is a predefined parameter and $\epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ is the noise used to generate the noisy image x_t.

A neural network is then trained to perform the *reverse* process. In the reverse process, the noisy image x_t is given as input to the neural network, which predicts the noise ϵ that was used to generate the noisy image. The network is trained with an L_2 loss:

$$L(\theta) = \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(x_t, t\right)\right\|^2\right] . \tag{2}$$

In existing DDPM-based methods, The network is typically trained on a large dataset of images, from which x_0 is sampled. Once trained, the generation process is initiated with a random noise image $x_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$. The image is passed through the model in a series of *reverse* steps. In each timestep $t = T, ..., 1$, the neural network predicts the noise ϵ_t. This noise is then used to generate a less noisy version of the image (x_{t-1}), and the process is repeated until a possible clean image x_0 is generated.

4. Single Image DDPM

Our goal is to leverage the powerful mechanism of diffusion models to generation from a single image/video. While the main contribution of this paper is in using DDPMs for generation from a single video, we first explain how a diffusion model can be trained on a single *image* (in Sec. 5 we show how this model can be extended to *video* generation).

Given a single input image, we want our model to generate new diverse samples that are similar in appearance and structure to that of the input image, but also allow for semantically coherent variations. We build upon the common DDPM [24] framework (Sec. 3) and introduce several modifications to the training procedure and to the core network of DDPM. These are highlighted below:

Training on Large Crops. Instead of training on a large collection of images, we train a single diffusion model on

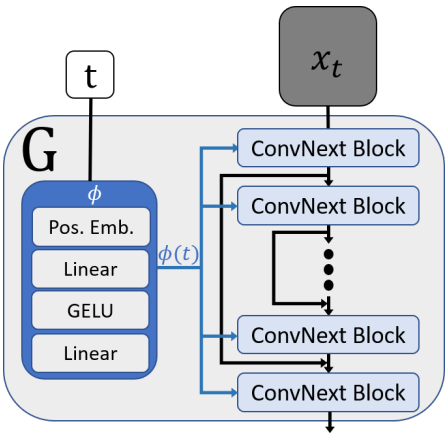


Figure 3. **Network Architecture**. Our backbone network is a fully convolutional chain of ConvNext [38] blocks with residual connections. Note that our network does not include any reduction in the spatial dimensions along the layers.

many large random crops from the input image (typically, about 95% the size of the original image, Fig. 2). We find that training on the original resolution of the image is sufficient for generating diverse image samples, even without the use of multi-scale pyramid (unlike most previous single image/video generative methods [1, 18, 19, 20, 23, 48, 49]). By training on large crops our generated outputs retain the global structure of the input image.

Network Architecture. Directly training the standard DDPM [24] on the single image or its large crops results in "overfitting", namely the model only generates the same image crops. We postulate that this phenomenon occurs because of the receptive field of the core backbone network in DDPM, which is the entire input image. To this end we modify the backbone UNet [44] network of DDPM, in order to reduce the size of its receptive field. We remove the attention layers as they have global receptive field. We also remove the downsampling and upsampling layers which cause the receptive field to grow too rapidly. Removing the attention layers has an unwanted side-effect - harming the performance of the diffusion model. Liu et al. [38] proposed a fully convolutional network that matches the attention mechanism on many vision tasks. Inspired by this idea, we replace the ResNet [22] blocks in the network with ConvNext [38] blocks. This architectural choice is meant to replace the functionality of the attention layers, while keeping a non-global receptive field. It also has the advantage of reducing computation time. The overall receptive field of our network is then determined by the number of ConvNext blocks in the network. Changing the number of ConvNext blocks allows us to control the diversity of the output samples. Please see further analysis and hyperparameter choice in Appendix A.1. The rest of our backbone network is similar to DDPM, as well as the embedding network (ϕ) which is used to incorporate the diffusion timestep t into the model (and will be later used to embed the video frame difference, see Sec. 5). See Fig. 3 for details.

Loss. At each training step, the model is given a noisy image crop x_t. However, in contrast to DDPM [24], whose model predicts the added noise (as in Eq. (2)), our model predicts the clean image crop $\tilde{x}_{0,\theta}$. The loss in our single-image DDPM is:

$$L(\theta) = \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\left\|x_0 - \tilde{x}_{0,\theta}\left(x_t, t\right)\right\|^2\right] \tag{3}$$

We find that predicting the image instead of the noise leads to better results when training on a single image, both in terms of quality and training time. We attribute this difference to the simplicity of the data distribution in a single image compared to the data distribution of a large dataset of images. The full training algorithm is as follows:

Algorithm 1 Training on a single image x

repeat
  $x_0 \leftarrow Crop(x)$
  $t \sim \text{Uniform}(1, ..., T = 50)$
  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  Take gradient descent step on:
    $\nabla_\theta \left\|x_0 - \tilde{x}_{0,\theta}\left(\sqrt{\bar{\alpha}_t}\mathrm{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|^2$
until converged

Our single-image DDPM can be used for various image synthesis tasks like diverse generation (Fig. 4), generation from sketch, image editing and more (Fig. 5). For more examples, see the project page.

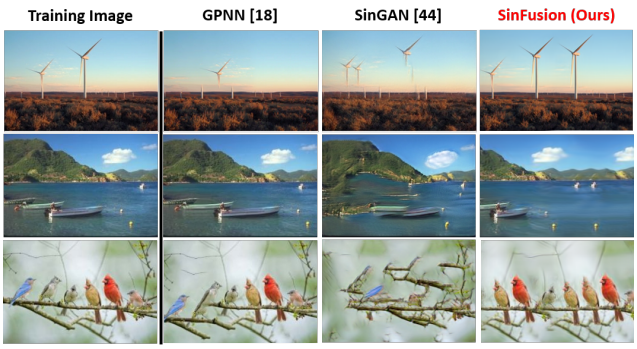


Figure 4. **Generation from a single image comparison:** (Please zoom in) Images generated by our single-image DDPM are comparable in visual quality to nearest-neighbour methods such as GPNN [18] and outperform SinGAN [48].

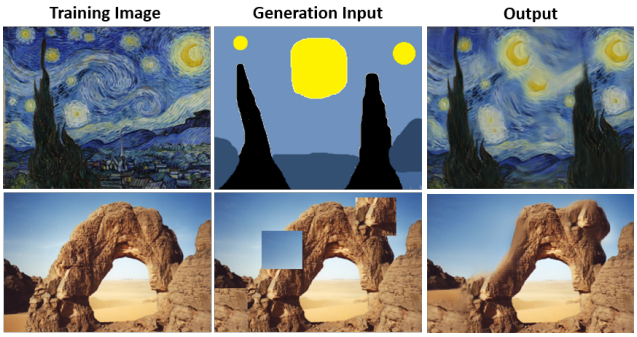


Figure 5. **Image Manipulation Tasks:** *Top* row shows generation from sketch, *Bottom* row shows image editing. Our single-image DDPM trains on crops from the training image (*left*) and receives, at inference, a noisy version of the sketch/edited image (*middle*). It denoises the input, thus projecting it to the original patch distribution, resulting in a natural looking edit (*right*).

5. Single Video DDPM

Our video generation framework consists of three single-image-DDPM models and is used for video-related applications (Sec. 6.1): diverse video generation (Fig. 1), video extrapolation (Fig. 7) and video upsampling.

Overview. Our framework is essentially an autoregressive video generator. Namely, we train the models on a given input video with frames $\{x_0^1, x_0^2, ..., x_0^N\}$, and generate new videos with frames $\{\tilde{x}_0^1, \tilde{x}_0^2, ..., \tilde{x}_0^M\}$ such that each generated new frame $\tilde{x}_0^{n+1}$ is conditioned on its previous frame $\tilde{x}_0^n$. The three models that constitute our framework are all single-image DDPM models with the same network architecture as described in Sec. 4. The models are trained *separately* and differ by the type of inputs they are given, and by their role in the overall generation framework. The inference is application-dependant and is discussed in Sec. 6.1. Here we describe the training procedure of each model:

DDPM Frame Predictor (Fig. 6a). The role of the *Predictor* model is to generate new frames, each conditioned on its previous frame. At each training iteration we sample a condition frame from the video x_0^n and a noisy version of the $(n+k)$'th frame (x_t^{n+k}), which is to be denoised. The two frames are concatenated along the channels axis before being passed to the model (as in Saharia et al. [47]). The model is also given an embedding of the temporal difference (i.e frame index difference) between the two frames ($\phi(k)$). This embedding is concatenated to the timestep embedding ($\phi(t)$) of the DDPM. At early training $k = 1$, and in following iterations it is gradually increased to be sampled at random from $k = [-3, 3]$. We find that such a curriculum learning approach improves outputs quality (even when at inference $k = 1$).

DDPM Frame Projector (Fig. 6b). The role of the *Projector* model is to "correct" frames that were generated by the *Predictor*. The Projector is a straightforward single-image-DDPM as described in Sec. 4, only it is trained on image crops from *all* the frames in the video. After learning the image structure and appearance of the video frames it is used to correct small artifacts in the generated frames, that may otherwise accumulate and destroy the video generation process. Intuitively, it "projects" patches from the generated frames back unto the original patch distribution, hence its name. The Projector is also used to generate the first frame. Frame correction is done at inference via a truncated diffusion process on the predicted frame.

DDPM Frame Interpolator (Fig. 6c). Our video-specific DDPM framework can be further trained to increase the temporal resolution of our generated videos, known also as "video upsampling" or "frame interpolation". Our DDPM frame *Interpolator* receives as input a pair of clean frames (x_0^n, x_0^{n+2}) as conditioning, and a noised version of the frame between them (x_t^{n+1}). The frames are concatenated along the channels axis, and the model is trained to predict the clean version of the interpolated frame ($\tilde{x}_0^{n+1}$). We find that this interpolation generalizes well to small motions in the video, and can be used to interpolate between every two consecutive frames, thus increasing the temporal resolution of generated videos as well as the input video.

Losses. We find that some models work better with different losses. The Projector and the Interpolator are trained with the loss in Eq. (3), while the Predictor is trained with Eq. (2), i.e. the noise is predicted instead of the output.

6. Applications & Results

In this section we present how to apply the trained single image/video DDPMs (Secs. 4 and 5) to various visual synthesis tasks. *We refer the reader to our project page*, especially for videos - results are not easily conveyed through the 2D format of a document.

6.1. Video Applications

We describe how to apply a trained Single Video DDPM (Sec. 5) to single-video tasks (see Fig. 1 and Fig. 7).

Diverse Video Generation: Our main application is generating diverse videos from a single given video, to any length, such that the output samples have similar appearance, structure and motion as the original input video. This can be done by combining our Predictor and Projector models. The first frame is either some frame from the original video, or a generated output image from the unconditional Projector (also see "*Diverse Image Generation*" in Sec. 6.2). The Predictor is then used to generate the next frame, conditioned on the previous generated frame. Next,

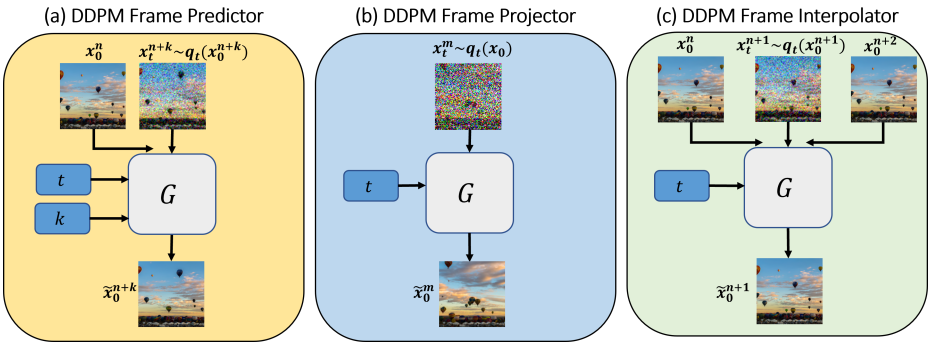


Figure 6. **Single Video DDPM** Our video framework consists of three models. The *Predictor* (left) generates new frames, conditioned on previous frames. The *Projector* (middle) generates frames from noise, and corrects small artifacts in predicted frames. The *Interpolator* (right) interpolates between adjacent conditioning frames and upsamples videos temporally. The models are used together at inference to perform various video related applications.

the predicted frame is corrected by the Projector (to remove small artifacts that may have been created). This process is repeated until the desired number of frames has been generated. Repeating this autoregressive generation process creates an arbitrary length new video. Note that the process is inherently stochastic – even if the initial frame is the same, different generated outputs will quickly diverge and create different outputs. See Fig. 1 *and project page for many more examples*.

Video Extrapolation: Given a video, we can anticipate its "future" frames, by initializing the generation process (described above) with the last frame of the input video. In Fig. 7 we show several examples for such extrapolation. Note how our method extrapolates the motion in a realistic way that keeps the appearance and dynamics of the original video. Since our Predictor is also trained backward in time (predicting the previous frame using negative k), it can generate new videos backwards in time, e.g., starting from the first frame of the video of the balloons causes them to "land", even though these motions were obviously not seen in the original video. This is a straightforward manifestation of the generalization capabilities of our framework. See Sec. 7 for more evaluations on the generalization capabilities, and *see full videos and more examples in the project page*.

Temporal Upsampling: we can increase the temporal resolution of an input video or a generated video by performing frame interpolation. Namely, generating frames in-between the original frames. This is done by using the Interpolator to interpolate between successive frames, and then correcting the appearance of the interpolated frames with the Projector. *Please see project page for results*.

Runtime. Training on a single 144×256 video of up to 1000 frames takes about 8 hours on a single V100 GPU.

6.2. Image Applications

We describe how to apply our trained single-image DDPM (Sec. 4) to image synthesis tasks (see Fig. 4, Fig. 5).

Diverse Image Generation: We can generate a new set of images with similar appearance and structure to the image on which the model was trained. This is done by sampling a noisy image $x_T \sim \mathcal{N}(0, \mathbb{I})$ and iteratively denoising using our trained model such that $x_{t-1} = G(x_t)$. Since our backbone network is fully convolutional, it can be used to generate images of any size by starting from a noisy image with the required size. In Fig. 4 We show a qualitative comparison to SinGAN [48] and GPNN [18]. Note that our results have better quality than SinGAN, similar to GPNN. *See project page for more results and a wide qualitative comparison*.

Editing & Generation from Sketch: We can edit an image by coarsely moving crops between locations in the image, then let the model "correct" the image. Or we can draw a sketch and let the model "fill in" the sketch with similar details from the image. The model is applied to the edited image / sketch by adding noise to the image, and then denoising the input image until a coherent image is achieved (see Fig. 5).

Visual Summary: We can use our model to summarize the important semantic information in an image. This is done by iteratively decreasing the size of the image, noising the downsampled image, and denoising it with the model. At each iteration the model "projects" the resized patches back to the original patch distribution. As the image shrinks, the "salient" objects (usually the larger) are left in the image, in their original resolution. *See project page for examples*.

Runtime. Training on a single 144×256 image takes about 40 minutes on a single V100 GPU.

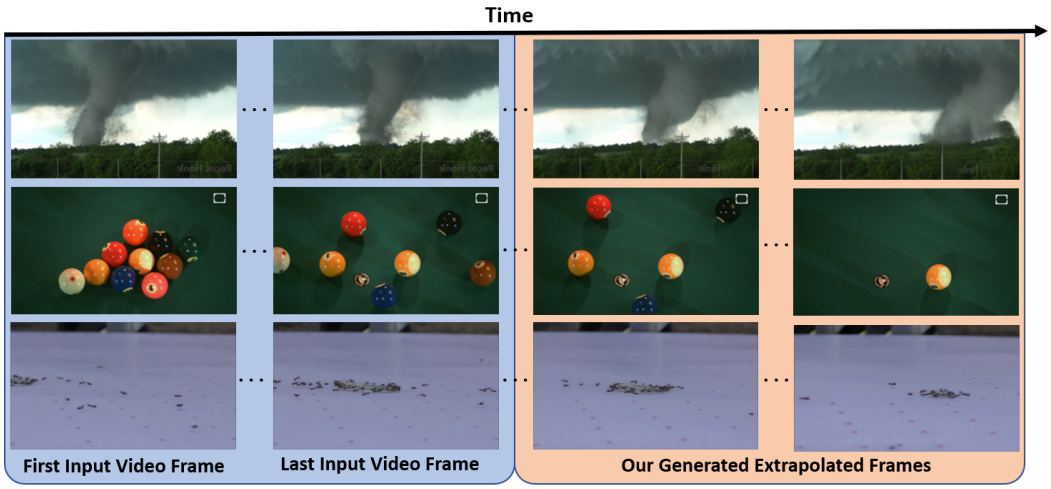


Figure 7. **Video Extrapolation:(See project page for full videos and more examples):** Our single-video generation framework can realistically extrapolate existing videos. Our models train on a single input video (*blue background*). At inference, we start the auto-regressive generation process from the *last* frame of the input video (2*nd column*), and generate a frame sequence of any desired length. The extrapolated frames (*orange background*) were never seen in the original video.

7. Evaluations & Comparisons

In this section we present quantitative evaluations to support our claim of motion generalization. We also compare to other methods for our main application of diverse video generation from a single video.

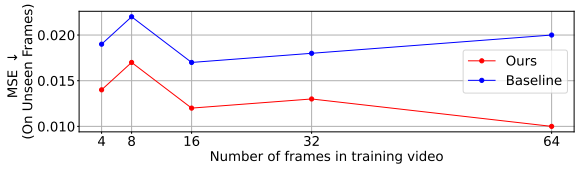
Measuring Generalization of Motion to Unseen Frames. In Fig. 7 and the project page we show qualitative results for the generalization capabilities of our framework, as evident from extrapolating a given video further into the future from the last frame, and backward into the past from the first frame. – Could this be quantified?

We present a few experiments, designed to measure how well our video framework generalizes to unseen frames. To this end we measure the performance of our framework on *frame-prediction from a single video*. Namely, the framework is trained on a smaller "slice" of the original video. Then, at inference, we sample 100 frames from the video, that were *not* part of the slice used at training. On each of these frames we use our framework to generate the next frame, then measure its distance from the ground-truth frame using MSE. the total score is the averaged MSE.

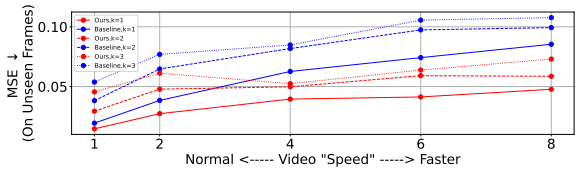
We devise a simple baseline for this task. The baseline returns the current frame as the "prediction" to next frame. Since most of the videos on which single-video models are trained, consist of large areas of static background (see Sec. 8), this choice makes for a reasonable baseline (as if there was no motion in the video).

We test our framework and baseline on two benchmarks. The first measures the performance of our framework on a varying number of training frames in the input video. The results are shown in Fig. 8a where each dot corresponds to averaging the score of 5 different runs (each trained on another slice of the video). As seen, our framework is consistently better than the baseline. Also, as expected, the generalization increases (lower MSE) with the size of the training set, while the naive baseline stays the same. Also see that our framework generalizes quite well to next-frame prediction with as low as 4 frames in the training set.

The second benchmark measures how well our framework generalizes on videos with faster motions. To this end we sub-sample the original video in intervals of increasing size , making motions in the sub-sampled video move faster. This way we can synthesize videos (all of which have 32 frames) with larger speeds from the same video, making the results consistent with the first benchmark. As shown in Fig. 8b, our framework is consistently better than the baseline. Also larger speeds increase the gap between our framework and the baseline, further validating our hypothesis for motion generalization. See more details in Appendix A.3.



(a)



(b)

Figure 8. **Evaluating Prediction to *Unseen Frames***. See text.

Dataset	Method	NNFDIV ↑	NNFDIST ↓	SVFID↓
SinGAN-GIF	VGPNN[20]	0.20	0.28	0.0058
	SinGAN-GIF [1]	0.40	1.10	0.0119
	SinFusion (Ours)	0.30	0.45	0.0090
HP-VAE-GAN	VGPNN[20]	0.22	0.14	0.0072
	HP-VAE-GAN [19]	0.31	0.39	0.0081
	SinFusion (Ours)	0.35	0.26	0.0107

Table 1. **Diverse Video Generation Comparison**.

Comparison to Single-Video Generation. We compare our main application of diverse video generation to existing single-video methods – HP-VAE-GAN [19], SinGAN-GIF [1] and VGPNN [20]. Since none of these methods is autoregressive, the evaluations above cannot be directly applied to them. However, a qualitative observation hints that the motions in most of their generated outputs are somewhat copied from the input video. Measuring motion "novelty" in generated videos is hard, mainly because of the inherent entanglement between motion and appearance, namely, we want to consider only novel motions with similar appearance to the original motions, and penalize novel motion with unrelated appearance (e.g., random noise is novel in the sense that it is not in the input but also has unrelated appearance). SinGAN [48] proposed a diversity measure that is easy to "fool" (see Appendix A.2).

We introduce a nearest-neighbor-field (**NNF**) based diversity measure. The NNF is computed by searching for each $(3, 3, 3)$ spatio-temporal patch in the generated video, its nearest neighbour patch in the real video. Simple generated videos (e.g. a translated or shuffled version of the input), will have a rather constant NNF, while more complex generated videos will have complex NNFs. An NNF of a single frame can be viewed as a color map by converting the spatio-temporal direction to the nearest neighbour using a color wheel [4]. In Fig. 9 we show how such an NNF color map looks in a frame generated by VGPNN [20], compared to a frame generated by our framework. The former is very simple, and corresponds to copying large chunks from a single input frame, whereas ours is more complex, which corresponds to a diverse generation from the input video (*see full videos in the project page*).

Inspired by *Kolmogorov complexity* [37] (simpler objects have simpler "description"), which can be easily bounded by any compression algorithm, we use ZLIB [15] to compress the NNF, and record the compression ratio. This gives a diversity measure (in $[0, 1]$) that we term *NNFDIV*. We can measure the RGB-similarity (*NNFDIST*) by recording the MSE of each generated patch to its nearest-neighbor. In Tab. 1 we report the results of these metrics on two diverse video generation datasets (Appendix B.1). As seen in the table, in both cases SinFusion has the best trade-off in terms of diversity and similarity. We also report SVFID [19] that measures similarity. while VGPNN is expected to have the best score (since it is copying chunks of frames), our score is between SinganGIF and HP-VAE-GAN. We attribute this to patch outliers in the generations (as apparent from the results), however similarity is not our main goal but the generation of novel motions.

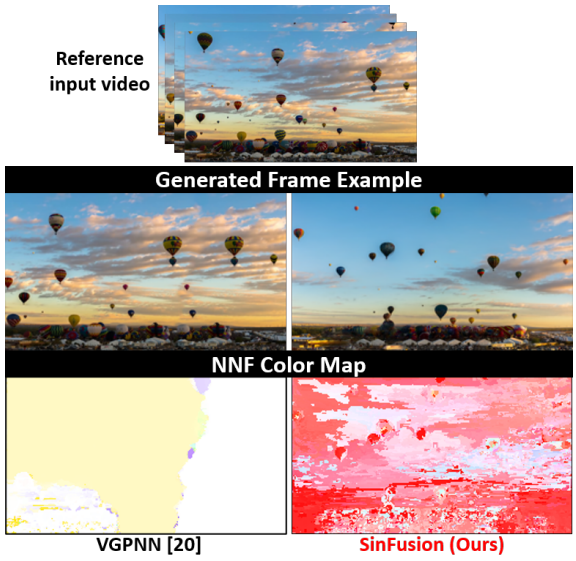


Figure 9. **Nearest-Neighbour Field (NNF) Color Map.** By computing the patch-NNF between generated video and the input video we see that VGPNN [20] outputs are actually copies of chunks, while ours generate novel patches.

Further Experiments and Ablations can be found in Appendix B and Appendix C (e.g., comparison to VDM [27]).

8. Limitations

As all methods for generation from a single video, our method is limited to videos with little to no camera motion. Also, when objects have relatively complicated motion (e.g., with many moving parts) our framework can have difficulties learning and discerning between different parts of the complex movements. Some of the above may be mitigated by including suitable priors.

9. Conclusions

We propose SinFusion, a DDPM framework trained on a single video or image. Our unified framework can be applied for a variety of tasks. Our main application, generation from a single video, exhibits generalization capabilities that were not shown neither by previous single-video methods, nor by DDPMs for video generation.

Acknowledgements

We thank Assaf Shocher and Barack Zackay for useful discussions. This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 788535) and from the D. Dan and Betty Kahn Foundation.

References

[1] Rajat Arora and Yong Jae Lee. Singan-gif: Learning a generative video model from a single gif. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1310–1319, 2021. 2, 4, 8, 12, 13

[2] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18208–18218, 2022. 1, 3

[3] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017. 2

[4] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011. 8

[5] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015. 2

[6] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. In *Proceedings of the European conference on computer vision (ECCV)*, pages 119–135, 2018. 2

[7] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 1, 3

[8] Hanqun Cao, Cheng Tan, Zhangyang Gao, Guangyong Chen, Pheng-Ann Heng, and Stan Z Li. A survey on generative diffusion model. *arXiv preprint arXiv:2209.02646*, 2022. 1

[9] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets. *arXiv preprint arXiv:1907.06571*, 2019. 2

[10] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *arXiv preprint arXiv:2209.04747*, 2022. 1

[11] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, pages 1174–1183. PMLR, 2018. 2

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

[13] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001. 2

[14] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999. 2

[15] Jean-loup Gailly and Mark Adler. Zlib compression library. 2004. 8

[16] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022. URL https://arxiv.org/abs/2208.01618. 1, 3

[17] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014. 1

[18] Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, and Michal Irani. Drop the gan: In defense of patches nearest neighbors as single image generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13460–13469, 2022. 2, 4, 6

[19] Shir Gur, Sagie Benaim, and Lior Wolf. Hierarchical patch vae-gan: Generating diverse videos from a single sample. *arXiv preprint arXiv:2006.12226*, 2020. 2, 3, 4, 8, 12, 13

[20] Niv Haim, Ben Feinstein, Niv Granot, Assaf Shocher, Shai Bagon, Tali Dekel, and Michal Irani. Diverse generation from a single video made possible. *arXiv preprint arXiv:2109.08591*, 2021. 2, 3, 4, 8, 12, 13

[21] William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos. *arXiv preprint arXiv:2205.11495*, 2022. 2

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 14, 15

[23] Tobias Hinz, Matthew Fisher, Oliver Wang, and Stefan Wermter. Improved techniques for training single-image gans. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1300–1309, 2021. 2, 4

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1, 2, 3, 4, 13, 15

[25] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 1, 2

[26] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23:47–1, 2022. 1

[27] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022. 1, 2, 8, 13

[28] Tobias Höppe, Arash Mehrjou, Stefan Bauer, Didrik Nielsen, and Andrea Dittadi. Diffusion models for video prediction and infilling. *arXiv preprint arXiv:2206.07696*, 2022. 2

[29] Nathan Jacobs, Brian Bies, and Robert Pless. Using cloud shadows to infer scene structure and camera calibration. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1102–1109. IEEE, 2010. 15

[30] Nathan Jacobs, Austin Abrams, and Robert Pless. Two cloud-based cues for estimating scene structure and camera calibration. *IEEE transactions on pattern analysis and machine intelligence*, 35(10):2526–2538, 2013. 15

[31] Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Rémi Tachet des Combes, and Ioannis Mitliagkas. Adversarial score matching and improved sampling for image generation. *arXiv preprint arXiv:2009.05475*, 2020. 1

[32] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 1

[33] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 1, 3

[34] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020. 1

[35] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. *arXiv preprint arXiv:2210.09276*, 2022. 1, 3

[36] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 14

[37] Andrei N Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 369–376, 1963. 8

[38] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 4, 14, 15

[39] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021. 1

[40] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 1, 14

[41] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 14

[42] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 1, 3

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

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 4, 15

[45] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. 2022. 1, 3

[46] 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, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022. 1

[47] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1, 5

[48] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4570–4580, 2019. 2, 4, 6, 8, 12

[49] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. Ingan: Capturing and remapping the" dna" of a natural image. *arXiv preprint arXiv:1812.00231*, 2018. 2, 4

[50] Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. Summarizing visual data using bidirectional similarity. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 2

[51] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022. 1

[52] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. 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*, pages 3626–3636, 2022. 2

[53] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 1, 3

[54] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 1

[55] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019. 1

[56] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018. 2

[57] Dani Valevski, Matan Kalman, Yossi Matias, and Yaniv Leviathan. Unitune: Text-driven image editing by fine tuning an image generation model on a single image. *arXiv preprint arXiv:2210.09477*, 2022. 1, 3

[58] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*, 2017. 2

[59] 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 preprint arXiv:2210.02399*, 2022. 1

[60] Vikram Voleti, Alexia Jolicoeur-Martineau, and Christopher Pal. Mcvd: Masked conditional video diffusion for prediction, generation, and interpolation. *arXiv preprint arXiv:2205.09853*, 2022. 2

[61] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *arXiv preprint arXiv:1609.02612*, 2016. 2

[62] Kaisiyuan Wang, Qianyi Wu, Linsen Song, Zhuoqian Yang, Wayne Wu, Chen Qian, Ran He, Yu Qiao, and Chen Change Loy. Mead: A large-scale audio-visual dataset for emotional talking-face generation. In *European Conference on Computer Vision*, pages 700–717. Springer, 2020. 15

[63] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018. 2

[64] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 3

[65] Ruihan Yang, Prakhar Srivastava, and Stephan Mandt. Diffusion probabilistic modeling for video generation. *arXiv preprint arXiv:2203.09481*, 2022. 2

[66] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019. 1

Appendix

A. Further Evaluations

A.1. Effect of Crop-Size and Receptive-Field

Our goal is to generate outputs (image / video) that preserve global structure, are of high quality, and with large diversity. These are affected by the choice of the crop-size on which the model is trained, and the effective receptive field of the model (determined by the depth of the convolutional model and controlled via the number of ConvNext blocks in the network). As seen in Fig. A2b, the largest diversity is achieved for small crop-size and small receptive field. However, small networks fail to learn the underlying image structure and result in blurry outputs (Fig. A2a). We therefore use more blocks for the model. This reduces the diversity, but dramatically improves outputs quality (as is evident from Fig. A2a). We choose the crop-size as a trade-off to preserve global-structure but also high diversity, which means using crop-size of about 95% of the image, with network depth of 16 blocks.

A.2. SinGAN [48] Diversity Score is Easily Fooled

Quantifying the "diversity" of generated outputs from a single image or video is hard. To begin with, other than the most simple case of generating the same input image over and over again, which will have a diversity score of zero, there's no definition for "diversity" other than a subjective human factor, looking on a set of images and deciding whether it is diverse or not. SinGAN [48] proposed to calculate the standard deviation of the intensity values of each pixel over all generated samples, then average this over all the pixels, and then divide the result by the standard deviation of the intensity values of the original image. This method aligns with the simple case of generating only one sample described above, but will fail to obtain a valid score for the following simple example: given an input image, generate "new" samples by applying random translations to the image. This will generate many *different* samples, therefore not having a diversity of "zero", but it is reasonable to assume that most people will not find such a generated set as diverse. In fact, it is very easy to achieve a high "diversity score" by sampling once a random image, which will act as the input image, then generate "new samples" by random translations (in this case the proposed diversity score will converge to 1).

In Tab. A1 we add to the original Tab. 1 (colored in blue here), the SinGAN-diversity scores of all methods, plus the SinGAN-diversity score for the *tranlsated noise*, and as can be seen, it is indeed very close to 1. We also add the other scores for comparison.

As opposed to this high scores given by SinGAN-diversity, note how our NNFDIV diversity score gives a very low score for the translated noise example.

Dataset	Method	NNFDIV ↑	NNFDIST ↓	SVFID↓	DIV [48]
SinGAN-GIF	VGPNN[20]	0.20	0.28	0.0058	0.60
	SinGAN-GIF [1]	0.40	1.10	0.0119	0.86
	SinFusion (Ours)	0.30	0.45	0.0090	0.65
HP-VAE-GAN	VGPNN[20]	0.22	0.14	0.0072	0.45
	HP-VAE-GAN [19]	0.31	0.39	0.0081	0.41
	SinFusion (Ours)	0.35	0.26	0.0107	0.44
	Translated Noise	0.06	$6.8 \cdot 10^{-4}$	$1.1 \cdot 10^{-7}$	0.96

Table A1. A simple "translated noise" receives a high score of ~ 1 by SinGAN[48] proposed diversity score, while our NNFDIV manages to better handle this example, giving it a very low diversity score (blue parts are copied from Tab. 1 for comparison).

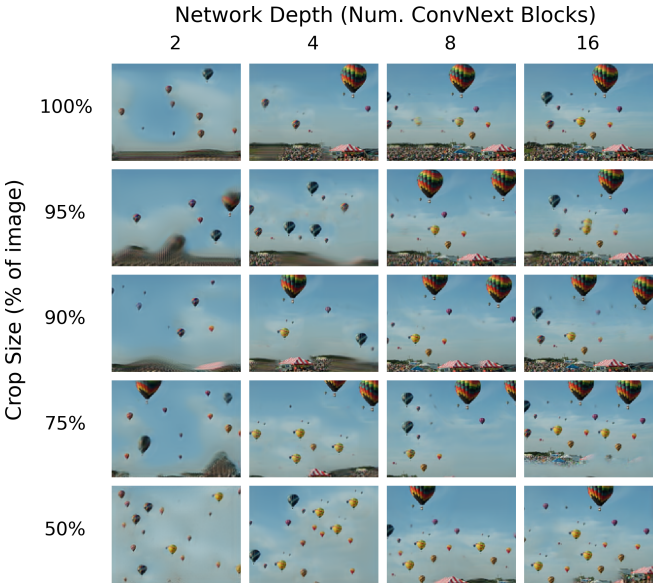
A.3. Further explaining generalization experiment

We further elaborate on the specific details of the experiment whose results appear in Fig. 8.

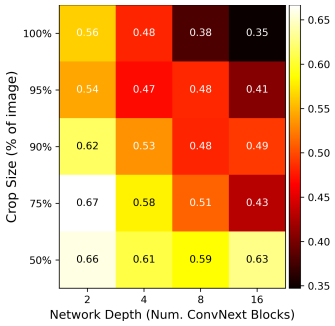
We are performing *frame prediction from a single video*. Each "run" involves training a model on n frames. After training, 100 frames (not seen during training) are chosen, and the model predicts the next frame for each of them. The total score for each run is the average MSE between predicted and ground-truth frames. Here we explain the difference between each "run":

Fig. 8a: Number of Training Frames For each number of training frames n (where $n = [4, 8, 16, 32, 64]$) we sample at random 5 starting points $i_1, ..., i_5$. For each starting point we use the n frames starting from frame i as the training samples. This is depicted in Fig. A3a – training frames (red) and test frames (green), where each test frame is used to predict the next frame. In Fig. A3b we show the schemes for runs trained on different number of frames.

Fig. 8b: Video "speed" and k In the described evaluation we fix the number of training frame to 32. For speed=1, this is similar to the same experiment shown in Fig. 8a for $n = 32$. For speed=2, for each random starting frame, we choose the next 32 frames of the $1/2$-subsampled video, e.g., if the starting point is frame number 17, the training frames will be frames number $17, 19, 21, ..., 79$. For speed=4 and starting point 17 the training frames will be frames number $17, 21, 25, ..., 141$. The parameter k is the same as in the training procedure of the DDPM-Frame-Predictor, namely, it is the frame difference between current frame and the predicted frame. That is, if $k = 1$, for each test frame number n we will predict the frame $n + 1$; if $k = 2$ we will predict $n + 2$ etc. Note that if the video is already subsampled, k is the frame difference w.r.t the subsampled video. In Fig. A3c we depict several choices for the speed (denoted by S) and k.



(a) Quality



(b) NNF Diversity (See Sec. 7)

Figure A2. Analysing the effect of Crop-Size and Network-Depth on the diversity and quality of the generated outputs

B. Comparisons

B.1. Generation from Single-Video (Tab. 1)

We run our comparisons on the data provided by the previous works on video generation from a single video: VGPNN [20], HP-VAE-GAN [19] and SinGAN-GIF [1]. We follow the same methodology used in VGPNN [20].

We compare to two datasets of videos. One provided by SinGAN-GIF [1] and the other by HP-VAE-GAN [19]. In SinGAN-GIF there are 5 videos with 8 to 15 frames, each of maximal spatial resolution 168×298. For each of the 5 input videos, each method generates 6 samples. In HP-VAE-GAN there are 10 videos each of spatial resolution 144×256. For each of the 10 input videos, each method generates 10 samples. HP-VAE-GAN and VGPNN only use the first 13 frames since their methods are limited by runtime and memory. Since learning on small amounts of data is not a goal for the task of diverse generation from a single video, and since our framework can easily learn from much more data, we train our framework on longer sequences of frames from the given input videos.

B.2. Comparison to VDM [27]

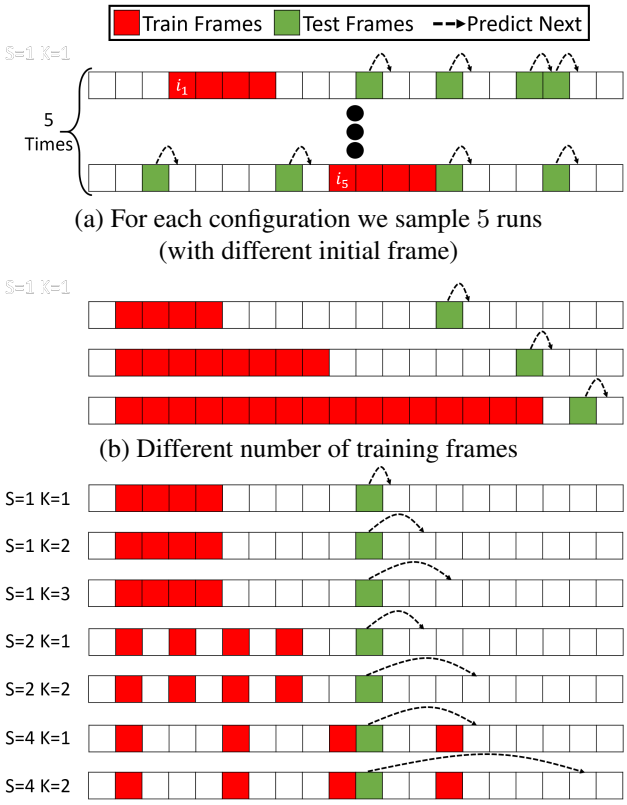
In the project page we show the results of Video-Diffusion-Models (VDM) [27] trained on a single video. Since the official implementation was not published at the time of writing, we use a third-party implementation [1]). Since VDM expects a dataset of videos, we slice a long video of 420 frames into 42 short videos of 10 frames each and let VDM train on those videos. We could only train the model on a resolution of 64×64 pixels before exceeding the memory of our GPU. We trained the model for about 150 epochs using two different learning rates (total run time was about a day on a V100 GPU). The results seems to capture the motions of the original videos, but it is difficult to evaluate since the resolution is too low compared to the original videos. The results also contain artifacts that may result from the low amount of data usually needed to train such models (without including our proposed modifications). It is qualitatively evident from the results that our framework, SinFusion, generates outputs of much higher quality when trained on a single video. For video results please visit our project page.

[1] https://github.com/lucidrains/video-diffusion-pytorch

C. Ablations

Predicting Image Instead of Noise. As opposed to the standard DDPM [24] training, our single-image-DDPM model outputs a prediction for the un-noised input crop. In Fig. A4 we show examples for our generated outputs for predicting the crop/image (top) against generated outputs for predicting noise (bottom). As shown, predicting the un-noised crop instead of the noise generates higher quality images. It also does so with much fewer training iterations.

Architectural changes. We check the importance of our proposed architectural modifications to the original DDPM [24] by reverting each change and generating several images. We compare these generated images to the images



(a) For each configuration we sample 5 runs (with different initial frame)

(b) Different number of training frames

(c) Different choices of video-speed (S) and k

Figure A3. **Frame Prediction from a Single-Video**. Depicting evaluations experiments from Sec. 7 and Fig. 8

generated by our final model. An example for these comparisons can be seen in Fig. A5.

The first comparison shows outputs of our model with upsampling and downsampling layers. The generated outputs completely overfit the training image, and have no diversity.

The second comparison shows outputs of our model with attention layers. Other than significantly increasing training time (to almost 2 hours per training image) and generation time, the added attention doesn't improve the generated image quality.

The third comparison shows outputs of our model with all ConvNext [38] blocks replaced with ResNet [22] blocks. The generated outputs suffer from more artifacts and are of lesser quality than our generated outputs.

Importance of DDPM frame Projector in diverse video generation. We show the necessity of our DDPM frame Projector model as part of the diverse video generation framework. In this ablation, we generate videos from several input videos using only the DDPM frame Predictor to generate frames, without using the Projector model to correct small artifacts in the generated frames. In all examples, it can be seen that the small artifacts, which remain uncorrected, accumulate over time and severely degrade the generation quality. For video results please visit our project page.

D. Implementation Details

Our code is implemented with PyTorch [41]. We make the following hyper-parameters choices:

- We use a batch size of 1. Each large crop contains many large "patches". Since our network is a fully convolutional network, each large "patch" is a single training example.
- We use ADAM optimizer [36] with a learning rate of 2×10^{-4}, reduced to 2×10^{-5} after $100K$ iterations.
- We set the diffusion timesteps $T = 50$. This allows for fast sampling, without sacrificing image/video quality (This trade-off is simpler in our case because of the simplicity of our learned data distribution).
- When generating diverse videos, we use the DDPM frame Projector to correct predicted frames by noising and denoising $T_{corr} = 3$ steps.
- We compared several noise schedules for the diffusion models and ended up using linear noise schedule ($\beta_0 = 2 \times 10^{-3}, \beta_T = 0.4$) for single-image DDPM and cosine noise schedule [40] for single-video DDPM.
- Our standard network architecture consists of 16 ConvNext [38] blocks, each block with a base dimension of 64 channels.

D.1. Runtimes

On a Tesla V100-PCIE-16GB, for images/videos of resolution 144×256, our model trains for about 1.5 minutes per 1000 iterations, where each iteration is running one diffusion step on a large image crop. The total amount of iterations and total runtime for each of our models are:

- Single-Image DDPM - $50K$ iterations, total runtime of 80 minutes (good results are already seen after $15K$ iterations).
- Single-Video DDPM Frame Predictor - $200K$ iterations, total runtime of 5.5 hours.
- Single-Video DDPM Frame Projector - $100K$ iterations, total runtime of 2.5 hours
- Single-Video DDPM Frame Interpolator - $50K$ iterations, total runtime of 1.5 hours.

Figure A4. **Noise vs Image prediction ablation**. *Top Row*: Input image (left;red) and generated outputs of our final model (right;purple) – predicts the un-noised image crop. *Bottom Row*: Generated outputs using the standard DDPM noise prediction.

Figure A5. **Architectural ablations**. *Top Row*: Input image (left;red) and Generated outputs of our final model (right;purple).
2nd Row: Generated outputs of our model with upsampling and downsampling layers.
3rd Row: Generated outputs of our model with added attention layers (similar to the standard DDPM [24] Unet[44] network).
4th Row: Generated outputs of our model where each ConvNext [38] block is replaced with a ResNet [22] block.

E. Videos Sources

In our project page we show results for video generation and and extrapolation for videos excerpts from the following YouTube videos (YouTube video IDs: LkrnpO5v0z8, hj6EG7x-BT8, nRxSUkZYeOE, 9ePic3dtykk, pB6XSixrCC8, ZO5lV0gh5i4, tmPqO_TGaU, bsSypB9gI0s, RZ1kK-X3QwM, FR5l48_h5Eo, 4i6VSrIYRYY, m_e7jUfvt-I, DniKM5SKe6c, rbzxxbuk3sk, W_yWqFYSggc, WA5fqO6LUUQ, -ydgKb5K_kc). The video of ants is courtesy of Aviram Gelblum and Tabea Heckenthaler from Ofer Feinerman's lab. We also use several videos from MEAD Faces Dataset [62], and Timlapse Clouds Dataset [29, 30].