This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Generative Video Bi-flow

Chen Liu
University College London
chen.liu.21@ucl.ac.uk

Tobias Ritschel
University College London
t.ritschel@ucl.ac.uk

Abstract

We propose a novel generative video model to robustly learn temporal change as a neural ordinary differential equation (ODE) flow with a bilinear objective which combines two aspects: The first is to map from the past into future video frames directly. Previous work has mapped the noise to new frames, a more computationally expensive process. Unfortunately, starting from the previous frame, instead of noise, is more prone to drifting errors. Hence, second, we additionally learn how to remove the accumulated errors as the joint objective by adding noise during training. We demonstrate unconditional video generation in a streaming manner for various video datasets, all at competitive quality compared to a conditional diffusion baseline but with higher speed, i.e., fewer ODE solver steps.

1. Introduction

We suggest a method to generate videos. Our model is based on a neural ODE flow, modeling the pixel evolution over time. To train the model, we reformulate the flow matching objective [1, 13, 19, 22], which has been widely used to map the noise distribution to an image or video distribution. One of our key insights is to consider a video dataset as separate image datasets in the temporal dimension. We can now learn flow between the current and the next-frame distribution. To make this work, we need to improve the coverage of the training space and learn a joint flow field of denoising by using train-time noise. This enables robust inference with few ODE solver steps. Our approach can produce new video frames in a streaming fashion at an interactive rate, making it a candidate technique for interactive applications such as computer games.

So far, there are mainly two dominant ways to generate videos, which we compare to our approaches, summarized as different columns in Fig. 1. The first approach, full-sequence diffusion [16], is a straightforward extension from images to videos. With large curated video datasets and established methods such as the noise conditional score function [29], this achieves good coverage of empirical video distributions, denoted as a success in the row "High coverage" in Fig. 1. However, it is a computationally intense process in both memory and time as the diffusion problem requires numerous steps to solve from a 3D noise tensor to a complete video, denoted as a challenge in the "Short distance" row in Fig. 1. It also does not support streaming video generation or the ability to react to external input in interactive applications. We seek to improve these aspects.

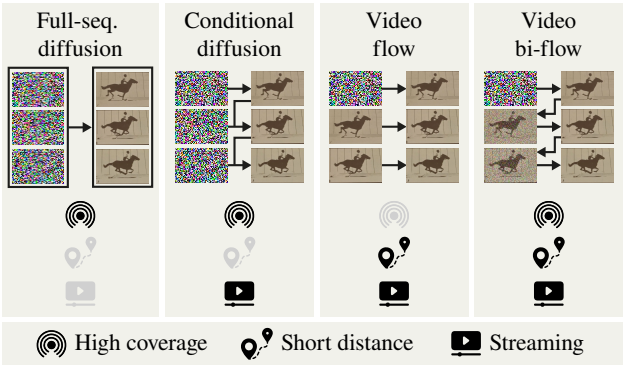

Figure 1. The two main ways to generate videos on the left and our approaches on the right. The rightmost is our final approach. "Coverage" denotes if training sees all conditions occurring at test. Higher coverage indicates more robust inference, while low coverage results in quick divergence. "Distance" is the length of the ODE solver path. The shorter distance to solve means fewer steps. "Streaming" is the ability to adapt to the condition and produce an infinite image stream using finite memory.

The second approach, conditional diffusion, is performed on frames in isolation. To help temporal consistency, the generation is conditioned on one or multiple previous frames. This allows for streaming, but still starts from noise and requires solving a large distance, as a result of which the compute demands remain high.

We introduce here a third approach, *video streaming flow*, which relates to the classic single-image animation problem, generating a plausible video sequence given a single image. The ODE flow models the gradient concerning the current frame and predicts the next frame directly from it, thus eliminating the lengthy denoising process. Such an approach is ideal, as it allows streaming, and does not need to change much between frames. Nevertheless, since the flow is trained solely to map from one frame to another

rather than mapping noisy inputs to clean frames, its coverage of the training space of temporal variations is limited. Consequently, the process becomes unstable, and the solution diverges after only a couple of frames.

Our final approach, *video bi-flow*, combines the strengths of both the second and the third approach. It infers each frame directly from the previous one, without starting from noise, but still injects sufficient noise during training to adequately cover the space of transitions and ensure stable solving. The resulting method is both computationally efficient and capable of stable streaming.

In summary, our contributions are:

- Learning of video flow to exploit correlation between frames to enable efficient streaming video generation;
- Mitigating error accumulation by a novel joint flow field, facilitating stable and long video generation;
- Experiments to systematically demonstrate that our video bi-flow exhibits superiority over the conditional diffusion both in stability and efficiency with comparable fidelity.

Our code will be released at https://github.com/ryushinn/ode-video.

2. Our Approach

2.1. Joint objective

Flow matching [19, 22] aims to regress a flow field f_θ to a pre-defined probability path between two distributions $\mathcal{X}_0$ and $\mathcal{X}_1$:

$$\arg\min_\theta \mathop{\mathbb{E}}_{\mathbf{x}_0,\mathbf{x}_1,\alpha} ||f_\theta(\mathbf{x}_\alpha,\alpha)-(\mathbf{x}_1-\mathbf{x}_0)||^2, \quad (1)$$

where $\mathbf{x}_0 \sim \mathcal{X}_0, \mathbf{x}_1 \sim \mathcal{X}_1$, $\alpha \sim \mathcal{U}(0,1)$, and $\mathbf{x}_\alpha = \mathbf{x}_0 + \alpha(\mathbf{x}_1-\mathbf{x}_0)$ is linear interpolation, such that it can drive a sample $\hat{\mathbf{x}}_0$ from $\mathcal{X}_0$ to a sample $\hat{\mathbf{x}}_1$ following $\mathcal{X}_1$:

$$\hat{\mathbf{x}}_1 = \mathcal{S}_{0\to1}(\hat{\mathbf{x}}_0, f_\theta) = \hat{\mathbf{x}}_0 + \int_0^1 f_\theta(\hat{\mathbf{x}}_s, s)\mathrm{d}s, \quad (2)$$

where $\mathcal{S}$ is an off-the-shelf ODE solver that solves f_θ with the initial value $\hat{\mathbf{x}}_0$ from 0 to 1.

This method is widely applied in image generation which we will first review. We then reformulate it to video generation by mapping from the distributions of past frames to the distribution of future frames. We finally introduce the robust training of our video flow, the key to our approach, where we combine the benefits of the above two formulations with bi-linear interpolation.

Image generation To generate images, $\mathcal{X}_0$ is usually a normal distribution $\mathcal{N}$ that we can sample, and $\mathcal{X}_1$ is an empirical image distribution formed by the image data of interest. The ODE field trained by solving Eq. 1 is the expectation of all possible linear paths between random pairs of noises and images, as depicted by Fig. 2a. In inference, a new image can be generated by solving Eq. 2 with a normal-random initial condition.

Video generation A video distribution can be formalized as temporally evolving distributions of image frames. The transport describing this manifold morphing provides a mechanism for generating videos, and the key challenge is deriving such a transport from data. To this end, we define $\mathcal{X}_0$ as the image distribution of all frames in a video dataset and $\mathcal{X}_1$ as the image distribution composed by all the next frames of every image from $\mathcal{X}_0$. Let $\mathcal{P}(\mathcal{X}_0,\mathcal{X}_1)$ be the joint distribution of all these consecutive frame pairs; this gives us an empirical approximation of our target transport.

By flow matching, we can then regress a neural ODE flow to this target transport by minimizing:

$$\arg\min_\theta \mathop{\mathbb{E}}_{(\mathbf{x}_0,\mathbf{x}_1),t} ||f_\theta(\mathbf{x}_t,t)-(\mathbf{x}_1-\mathbf{x}_0)||^2, \quad (3)$$

where $t \sim \mathcal{U}(0,1)$, $(\mathbf{x}_0,\mathbf{x}_1)$ is sampled from $\mathcal{P}(\mathcal{X}_0,\mathcal{X}_1)$, and $\mathbf{x}_t = \mathbf{x}_0 + t(\mathbf{x}_1-\mathbf{x}_0)$. Fig. 2c illustrates this flow.

Specifically, we minimize to transition from one image distribution to another. Instead of using independently random data pairs from $\mathcal{X}_0$ and $\mathcal{X}_1$, we employ $\mathcal{P}$ during training, the intrinsic coupling in video data. A proper dependent coupling can result in straighter and easier-to-solve trajectories as shown by Liu et al. [22]. If solved exactly in Eq. 3, our coupling provides a valid ODE flow field driving $\mathcal{X}_0$ to $\mathcal{X}_1$. By iteratively solving the learned ODE, we can generate an entire trajectory of coherent frames as a video.

Compared to conditional diffusion models which generate the next frame by starting from noise and conditioning the last frame, we show that our video flow significantly reduces the number of steps to stream the video generation with comparable quality and consistency. Intuitively, it is because we start from an initial state closer to the solution when rolling out the next frame.

Bi-linearity for robustness Unfortunately, the video flow alone seems not to be sufficiently stable to advance in time across many frames. The original approach for images maps between noise and distribution of images, while here, we need to map between images and do so in a stable fashion, *i.e.*, iterating as many times as the video has frames. In this condition, the key problem is the accumulation of errors, and the inability to recover from these during inference, as they were never observed during training.

As in Eq. 3, $\mathbf{x}_t$ lies in the image manifold and its linear interpolations, the training of the video flow can only supervise a rather low coverage of the entire space, in contrast to the common flow matching which blends noise and image data during training. This results in that the video flow is sensitive to any accumulated errors and deviates quickly.

As a remedy, we populate the training space with noise and bi-linear interpolation and train a joint flow field called *video bi-flow* which predicts both the video direction and the denoising direction.

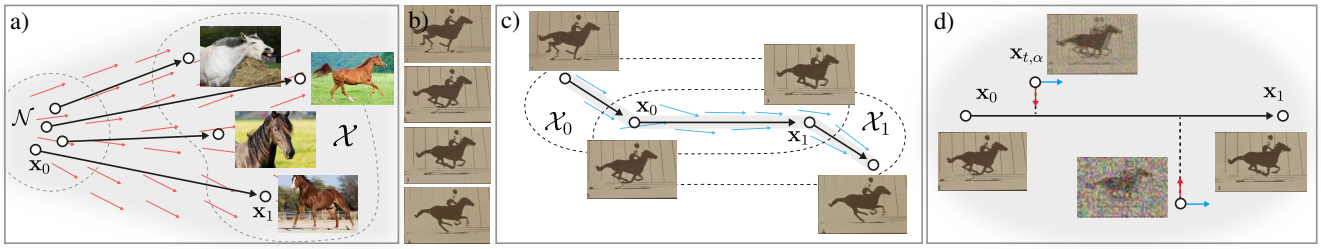

Figure 2. The main idea of our approach: We extend the flow matching shown in **a)** which learns a flow from a normal distribution to a distribution of images. Consider a sequence of four video frames in **b)**. **c)** shows the sequence and the linear interpolations between them as black lines in the respective order. Note that the horizontal axis is not time, but time flows along the line. We learn a flow field to drive these trajectories by applying flow matching between two consecutive frame distributions $\mathcal{X}_0$ and $\mathcal{X}_1$. The fitted flow field is, unfortunately, only stable in a small region around these linear connections, shown as the gray area. **d)** shows our final approach, for simplicity zooming onto a link of only two frames. We add noise to the interpolated points during training, and two random points are shown as examples. The flow field is then tasked to map these distorted points forward (blue arrows), but also denoise them back to clean interpolations (red arrows). This widens the coverage of the training frame space, as shown as the populated gray area, resulting in a flow field that can effectively recover from errors accumulating over many frames.

We denote a bi-linear interpolation

$$\mathbf{x}_{t,\alpha} = \mathbf{x}_0 + t(\mathbf{x}_1 - \mathbf{x}_0) + \alpha\mathbf{n}, \tag{4}$$

which first blends two consecutive frames by time factor t, and then blends the result with random noise $\mathbf{n} \sim \mathcal{N}$ of magnitude α. The noise here serves as a surrogate of all potential artifacts accumulated during inference, given that we have no easy access to their true distribution. Combining Eq. 1 and Eq. 3, the training objective now becomes:

$$\begin{aligned}\arg\min_{\theta} \mathop{\mathbb{E}}_{\substack{(\mathbf{x}_0,\mathbf{x}_1),\mathbf{n},\\ t,\alpha}} & [||f_\theta^{\mathrm{v}}(\mathbf{x}_{t,\alpha}, t, \alpha) - (\mathbf{x}_1 - \mathbf{x}_0)||^2 \\ & + \quad ||f_\theta^{\mathrm{n}}(\mathbf{x}_{t,\alpha}, t, \alpha) - \mathbf{n}||^2].\end{aligned} \tag{5}$$

where the target $f_\theta = (f_\theta^{\mathrm{v}}, f_\theta^{\mathrm{n}})$ is indeed a joint field of two ODE flow fields of moving forward in time and denoising, as depicted in Fig. 2d. The video field f_θ^{v}, as optimized in the first term, predicts the-next-frame trajectory under a noise level of α and the (de)noising field f_θ^{n}, as optimized in the second term, predicts the noising errors added and can drive the corrupted images to clean images or their linear mix by t. Thus, the video bi-flow subsumes the two cases of generation mentioned above as two extremes when $\alpha = 0$ or $t \in \{0, 1\}$.

Our bi-linear training objective achieves the same aim as described by Song and Ermon [29] for classic diffusion: to increase the coverage of the training space – for us video frames – therefore stabilizing the generation process.

2.2. Joint sampling patterns

In inference, our trained video bi-flow can be solved along the dimension of α as the denoiser/corrector:

$$\frac{\partial \mathbf{x}_{t,\alpha}}{\partial \alpha} = f_\theta^{\mathrm{n}}(\mathbf{x}_{t,\alpha}, t, \alpha), \tag{6}$$

or along the dimension of t as the predictor:

$$\frac{\partial \mathbf{x}_{t,\alpha}}{\partial t} = f_\theta^{\mathrm{v}}(\mathbf{x}_{t,\alpha}, t, \alpha). \tag{7}$$

In practice, we evolve one given frame with the predictor to form a video. A frame generated by a separate model or a random frame in the test set can be the first frame. To ensure stable video streaming, we additionally employ the denoiser to remove the error accumulated at each step with a joint sampling pattern, which is the characteristic curve [25] to solve the first-order partial differential equation (PDE) field f_θ in the space of (t, α), along which the PDE becomes an ODE. Below, we will review the streaming generation and introduce our joint sampling pattern.

Streaming generation We aim to sample a video of n frames $\{\hat{\mathbf{x}}^i | i = 0, 1, ..., n-1\}$ with the first frame $\hat{\mathbf{x}}^0$ given. The following frames are generated by Markovian sampling, where we solve Eq. 7 to get $\hat{\mathbf{x}}^i = \mathcal{S}_{0\to1}(\hat{\mathbf{x}}^{i-1}, f_\theta^{\mathrm{v}}(\cdot, \cdot, 0))$. Note that here we set $\alpha = 0$, *i.e.*, we have to assume the last frame is clean without reliable prior information on how many errors have been accumulated. In this pattern, the denoising flow field f_θ^{n} in Eq. 6 is never used, but we will show that it is an effective corrector to significantly mitigate the accumulated drift.

Joint sampling The joint modeling actually enables one to solve simultaneously in both the temporal and denoising dimensions. Our bilinear video flow is a first-order PDE system in the $t\alpha$-plane, where we can consider a characteristic curve (t_k, α_k) parameterized by $k \in [0, 1]$. Solving the PDE at (t_1, α_1) with an initial value at (t_0, α_0) reduces to solving an ODE along this curve. The corresponding characteristic ODE of this curve is:

$$\begin{aligned}\frac{\mathrm{d}\mathbf{x}_{t_k,\alpha_k}}{\mathrm{d}k} &= \frac{\partial \mathbf{x}}{\partial t}\frac{\mathrm{d}t}{\mathrm{d}k} + \frac{\partial \mathbf{x}}{\partial \alpha}\frac{\mathrm{d}\alpha}{\mathrm{d}k} = \\ &\underbrace{f_\theta^{\mathrm{v}}(\mathbf{x}_{t_k,\alpha_k}, t_k, \alpha_k)\frac{\mathrm{d}t_k}{\mathrm{d}k} + f_\theta^{\mathrm{n}}(\mathbf{x}_{t_k,\alpha_k}, t_k, \alpha_k)\frac{\mathrm{d}\alpha_k}{\mathrm{d}k}}_{f_\theta^{\mathrm{j}}(\mathbf{x}_{t_k,\alpha_k}, k)}.\end{aligned} \tag{8}$$

Thereafter, we can achieve the correction and evolution jointly by solving Eq. 8:

$$\hat{\mathbf{x}}^i = \mathcal{S}_{0\to 1}(\hat{\mathbf{x}}^{i-1} + \epsilon\mathbf{n}, f^{\mathrm{j}}_\theta), \tag{9}$$

where we set as the initial value the last generated frame with noise added, and solve along a curve from $(t_0, \alpha_0) = (0, \epsilon)$ to $(t_1, \alpha_1) = (1, 0)$, *i.e.*, advancing to the next frame while applying denoising. The added noise simulates the drifting error over time such that we can explicitly remove it, yielding a more stable video trajectory compared to the naïve streaming generation. The noise level ϵ is a hyperparameter that we can tune to trade off video consistency for frame quality during inference, and we study its different levels in the evaluation.

We only consider the linear curve straightly connecting two endpoints so $(\frac{\mathrm{d}t_k}{\mathrm{d}k}, \frac{\mathrm{d}\alpha_k}{\mathrm{d}k}) = (1, -\epsilon)$. We find experimentally that it works better than two alternative nonlinear curves, the solve-then-denoise curve ($(t_{0.5}, \alpha_{0.5}) = (1, \epsilon)$) and the denoise-then-solve curve ($(t_{0.5}, \alpha_{0.5}) = (0, 0)$).

3. Experiments

3.1. Protocol

Dataset We extensively validate our video bi-flow in six video datasets with varying levels of variations: `MINERL` (Minecraft gameply videos of an agent [27]), `MAZES` (videos of navigating a maze [27]), `CARLA` (car driving videos [11]), `SKY` (time-lapse videos of sky [32]), `BIKING` (first-person view mountain biking footage [6]), and `RIDING` (horse riding videos [6]). We resize the videos to 128^2 for training, except for `MINERL` and `MAZES`, for which we use their original resolution of 64^2. We use the default train-test split for all datasets: all methods are trained in the train set and then used to generate videos given random frames from the test set as the starting points. We adopt a random 80-20 split for those datasets without a default split. We only experiment with unconditional video generation.

Metrics We quantify our results in terms of quality and compute speed. The quality metric is Fréchet video distance (FVD) [30]. We measure FVD on 128 samples of 512 frames generated by each method against the test set. To evaluate the quality change over time, the FVD is measured with a sliding window of 32 frames in a stride of 16 frames. The compute speed is simply the wall-clock time used to generate one frame. It is measured in the unit of the number of necessary steps[1] to solve the trained flow by an adaptive ODE solver under the same accuracy.

We show more numerical analysis with frame-based metrics in the supplemental.

[1]Precisely, it is the number of ODE network evaluations.

Methods Our baseline method is a conditional diffusion model generating a frame conditioned on the previous frame, as referred to as CONDIFF[2], which we compare to our video bi-flow (BI-FLOW) and its ablation: training our proposed video streaming flow without the bilinear extension (FLOW).

We train all methods with the same network architecture using the same setting and hyper-parameters, only differing in the losses and the throughput. Specifically, CONDIFF is trained by the conditional version of Eq. 1 with the last frame concatenated to noise as the condition. Thus, CONDIFF yields a throughput of $6 \to 3$, *i.e.*, taking a six-channel input and predicting the three-channel denoising gradient. FLOW is trained by Eq. 3 with the throughput of $3 \to 3$. BI-FLOW is trained by Eq. 5, our bi-linearity loss, to enable fast and stable video streaming. As a joint field, BI-FLOW has a throughput of $3 \to 6$: each half of the six-channel output is the partial derivative along time or (de)noising. We use a 2D UNet without any dedicated temporal modeling.

In inference, we solve the learned flows using an adaptive Heun ODE solver with an absolute and relative tolerance of 10^{-2}. For CONDIFF, we iteratively use the last frame as the condition and solve the trajectory from random noise to the next frame. In contrast, FLOW and BI-FLOW solve the trajectory from the last frame to the next directly, while BI-FLOW further employs joint sampling (Eq. 9)

We conducted all experiments on the Nvidia RTX 4090 GPU. More implementation details can be found in the supplemental material.

3.2. Evaluation

Quantitative The main quantitative results are seen in Tab. 1. FLOW requires the lowest number of steps per frame, but its solving trajectory rapidly diverges due to a lack of error mitigation, resulting in the highest FVD. In contrast, CONDIFF can achieve decent FVDs, albeit at the expense of a substantially increased step count, as it always starts from noise when it rolls out the next frame. As illustrated in the scatter plots, these two methods lie in the top-left and bottom-right corners, respectively. BI-FLOW, however, generates videos robustly with denoising flow as the corrector. Compared to CONDIFF, BI-FLOW consistently attains superior—at least comparable—FVDs across all datasets while using fewer necessary steps through the fast video flow predictor. Notably, BI-FLOW occupies the ideal bottom-left quadrant in all scatter plots, indicating its ability to synthesize high-quality videos with minimal computational overhead.

In plots of Tab. 1, we also evaluate four different noise levels ϵ during the joint sampling, ranging from 0.0 to 0.3, annotated with four gradations of blue to reflect the increase

[2]With a slight abuse of terminology, it is in fact a conditional flow model, *i.e.*, a deterministic diffusion model.

Table 1. The **table** reports the main quantitative results of our and other approaches (rows) for different video datasets (columns). Numbers here are averaged over time; The **scatter plots** below show how quality and speed relate: an ideal method would have a low step count and a low FVD to reside in the bottom-left corner. We also plot BI-FLOW with varying levels of the noise ϵ applied in sampling, as 0.0, 0.1, 0.2, and 0.3. These variants form the Pareto front of our BI-FLOW connected by dots, depicting the spectrum of performance we can alternatively favor based on the application. Note that it is our proposed joint flow field that enables this trade-off dimension, independent of the ODE solver. The noise level yielding the best FVD for each dataset is marked with *a star* in the scatter plot. This variant is reported in the table and also henceforth reported by default in other evaluations; The **line charts** are the linear trend lines of FVD vs. Time, plotted by computing the FVDs in a sliding window on the entire video. Here "Time" denotes the frame index. A flat trend line indicates a method can effectively mitigate the accumulated errors and maintain video quality over time. A positive slope means error accumulation.

	SKY		BIKING		RIDING		CARLA		MAZES		MINERL	
	FVD ↓	Steps ↓	FVD ↓	Steps ↓	FVD ↓	Steps ↓	FVD ↓	Steps ↓	FVD ↓	Steps ↓	FVD ↓	Steps ↓
FLOW	2144	**7.84**	14010	**10.38**	9154	**7.43**	4913	**7.78**	5051	**10.02**	3259	**10.20**
BI-FLOW	**526**	14.57	**3753**	25.63	**1552**	13.73	**345**	23.51	605	29.62	**834**	25.73
CONDIFF	1271	46.18	11465	33.15	4668	37.73	947	53.77	**424**	45.31	1149	41.61

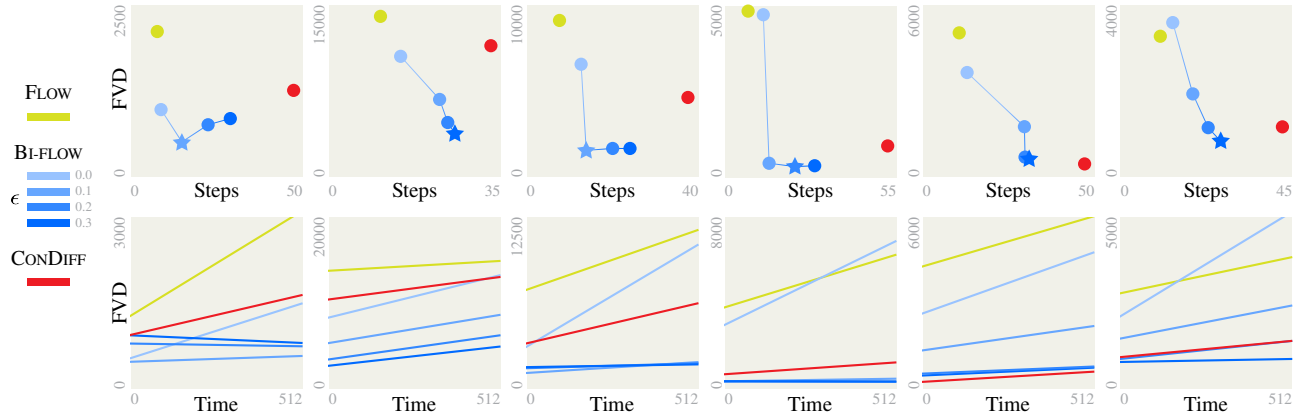

Table 2. FVD reported for comparisons between methods under the same number of steps.

	SKY	BIKING	RIDING	CARLA	MAZES	MINERL
FLOW	2109	11158	8984	4931	4928	3170
BI-FLOW	**491**	**3271**	**1621**	**307**	**620**	**823**
CONDIFF	994	10253	6698	829	655	2286

in noise levels. We find that increasing ϵ requires slightly more solving steps while taming error accumulation, as evidenced by reduced slopes in the line charts: BI-FLOW with $\epsilon = 0.0$ (*i.e.*, using the naïve streaming generation) exhibits a sharp FVD rise akin to FLOW and CONDIFF, but presents nearly flat trend lines with higher ϵ. Thus, it is necessary to adopt an appropriate ϵ for stable long video generation.

However, increasing ϵ does not necessarily decrease the average FVD over time, despite enhancing long-term stability, as it is also a trade-off between the intra-frame quality and the inter-frame consistency. Although higher ϵ enables stronger denoising corrections that improve per-frame fidelity and suppress error propagation, it paradoxically degrades temporal coherence due to the additional noise introduced. Therefore, depending on the emphasis on steps, stability, or the video quality, the optimal ϵ can vary vastly across datasets. Here, we report the noise level yielding the best FVD by default for each dataset since it usually achieves a good balance among these three aspects with only half of steps compared to CONDIFF on average.

Besides the results of equal-accuracy by an adaptive ODE solver, we also conduct equal-time experiments by an Euler ODE solver with a fixed step size of 0.05, that is all methods operate under the same budget of 20 steps, as shown in Tab. 2. BI-FLOW consistently outperforms CONDIFF in this scenario.

Qualitative We show frames generated by BI-FLOW in Fig. 3. Using the denoising flow, our method can produce decent-quality long videos without noticeable degradation over time. Notably, for SKY dataset comprising videos with only 32 frames, our method can even robustly stream videos significantly longer than the training data, *i.e.*, achieving video extrapolation. We also qualitatively compare our methods and baselines in Fig. 4, which demonstrates the effectiveness of our joint flow design in removing accumulated errors.

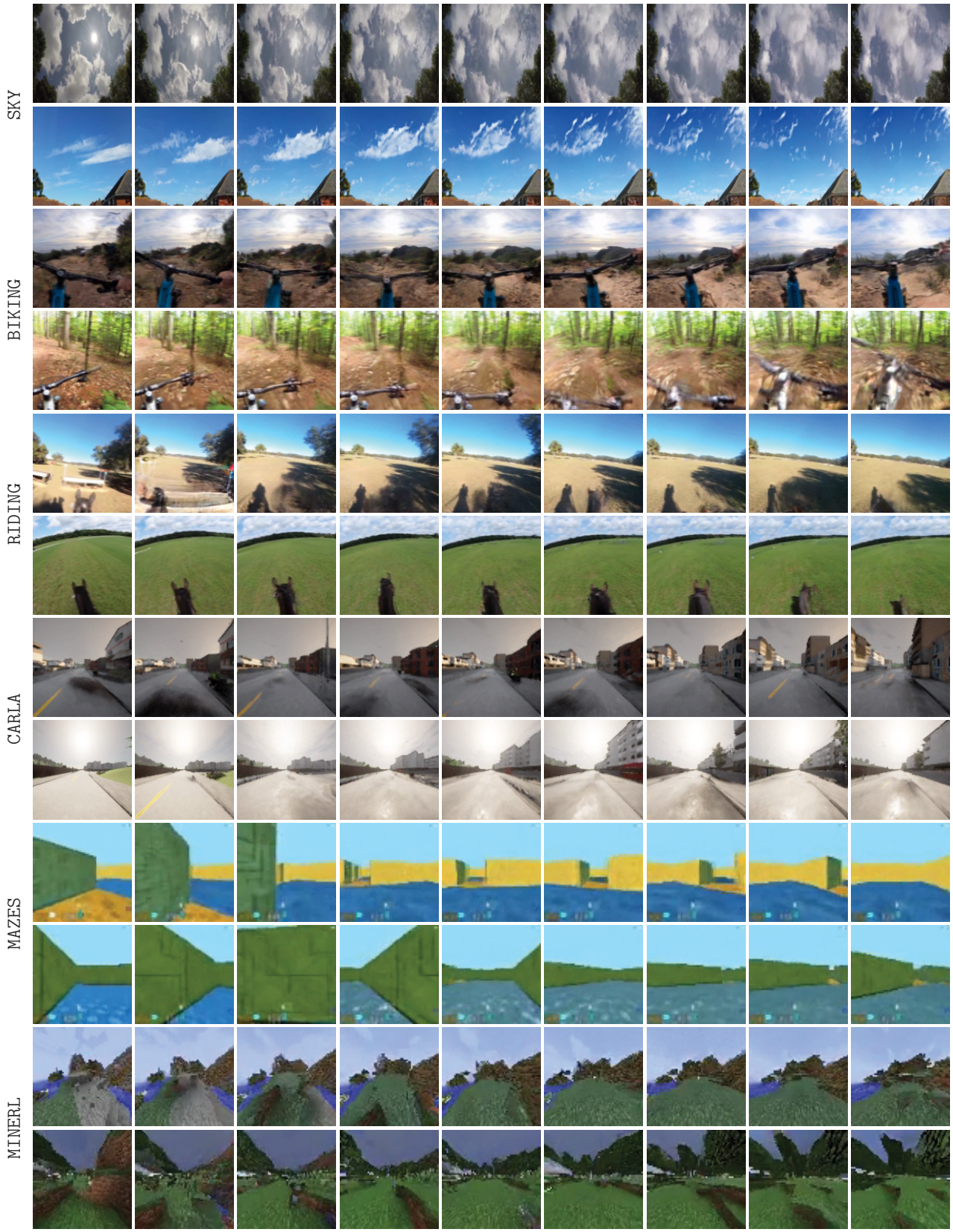

Figure 3. Videos samples generated by BI-FLOW. Time goes from left to right.

Figure 4. Five frames generated by different methods for SKY. These five frames uniformly span 128 frames in total, four times longer than 32-frame training data. Our BI-FLOW maintains the best quality throughout, while other methods induce obvious artifacts near the end of the video.

We present more videos for all methods and datasets in the supplemental.

4. Related Works

4.1. Video diffusion models

Diffusion models [14] initially applied its success to the video domain by directly expanding states to 3D tensors and adopting (factorized) 3D convolutions and attentions [4, 16]. To resolve the high compute demand associated with large 3D tensors, hierarchical synthesis and latent modeling are usually essential [2, 3, 10, 17, 18], extending not only across spatial dimensions [26] but also temporally [15], especially when long-duration video generation is a primary objective [6, 11, 12]. Nevertheless, hierarchical video synthesis is not suitable for interactive applications since users have to wait until an entire full sequence of video has been generated. Alternatively, generating videos auto-regressively in the temporal dimension has been investigated in prior works [11, 16] as another conditioning option. One of the most recent works is GameNGen [31] which runs a diffusion model auto-regressively to generate videos, conditioned on the user inputs and all the past generated frames up to the memory budget, and simulates an infinite environment for the video game "Doom".

We refer to this paradigm as conditional diffusion, irrespective of the conditioning mechanism employed. They all diffuse from noise. Our method, in contrast, addresses the inherent lengthy sampling process of diffusion models by commencing our video flow directly from the previous frame rather than random noise, thereby yielding faster sampling.

LVDM [4] and GameNGen [31] also adopt noise to perturb conditions during training for robust inference. Diffusion forcing [7] presents a similar idea that their diffusion model can condition on any noisy version of previous frames, ranging from prior (pure noise as a condition) to posterior (a clean image as a condition). They achieve this by training a separate recurrent network to update the state token based on noisy observations, on which the image diffusion model will condition. Our method not only augments the input with noise but also learns to remove the added noise explicitly as one branch of the joint flow field. The learned joint flow field builds a first-order PDE in the 2D space of time and denoising, which supports flexible sampling paths, such as the efficient and robust joint sampling pattern (Sec. 2.2) and backward sampling (Sec. 5).

Recent works share our observation that the last frame is a better starting point for solving the next frame. RIVER [9] accelerates the sampling process by truncated denoising, which solves the denoising ODE with the noisy last frame as the initial value and a reduced integration range. SVP [24], primarily for smoother results rather than acceleration, performs a weighted sum of the current state, the last frame, and noise at an intermediate denoising step. They can be considered as special cases of conditional diffusion with modified inference procedures. Our fundamental difference from them is the ODE of time, which we show is a much more effective flow to generate the next frame with fewer steps and deserves dedicated modeling.

4.2. Diffusion bridges and coupled data

A recent family of methods named diffusion bridges focuses on generalizing the diffusion models to any two arbitrary distributions, rather than relying on the non-informative Gaussian as the prior distribution [5, 21, 28, 33]. By leveraging the correlation between distributions, especially the coupled data pairs, *e.g.*, in the image-to-image translation task, the bridge models exhibit superior performance compared to the traditional diffusions. The same idea has also been adopted and explored in flow models. Heitz et al. [13] investigate some data-to-data applications such as colorization and super-resolution. Liu et al. [22] show that we can iteratively straighten the transport to achieve faster sampling by a new round of training based on the last learned coupling, or, in the first iteration, the independent coupling.

Inspired by the potential of coupled data in these successes, we propose leveraging the inherent inter-frame correlation in video data, an aspect that has yet to be sufficiently explored, and demonstrate that it represents a more efficient and favorable transport flow toward the subsequent frame.

4.3. Neural differential equations for videos

Chen et al. [8] propose neural ODE with the adjoint gradient method. Flow matching [19] is a simulation-free training method for neural ODE to learn an ODE flow that transforms one distribution to another distribution. As revealed by recent works [13, 22] flow matching is a deterministic diffusion model with linear interpolation as the forward diffusion process.

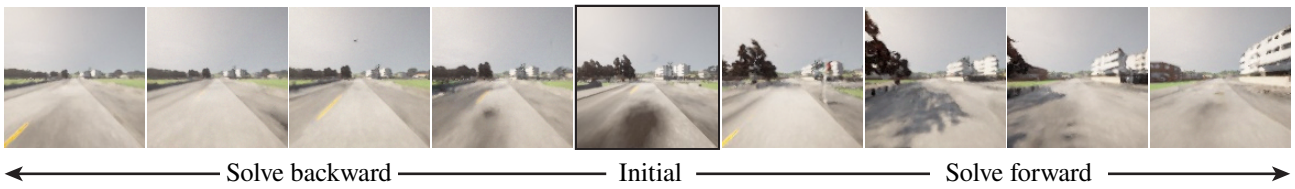

Figure 5. The given initial frame is in the middle. Frames on either side are solved with our video bi-flow forward or backward, respectively.

From the perspective of the diffusion model, only the solution at the end time-point matters, *i.e.*, the generated sample. This is distinctly different from how general neural ODEs are used, where the solution is the solving trajectory itself. Our work is inspired by the recent work of dynamic appearance ODEs [20], the solution trajectory of which is the generated video. It indicates that the ODE flow continuously models the video signal. However, since they simulate the ODEs to optimize the textural statistics loss over the trajectory, the training efficiency is limited and the target video must be textures that have spatially stationary visual statistics. Instead, we train our video ODEs based on the reformulated flow matching, which supports general videos and computationally efficient training.

5. Discussion

Solve backward The ODE flow field is invertible. By instantiating the video trajectory as an ODE flow, it enables us to solve the trajectory either forward or backward in time, *i.e.*, we can discover not only the future but also the past given one frame. Note that the diffusion model does not naturally possess this bidirectional capability: For instance, CONDIFF requires an additional flag to condition the solving direction explicitly and swap training data correspondingly to support an analogous functionality [23].

Specifically, we can solve backward in time by solving the reverted flow as:

$$\hat{\mathbf{x}}^{i-1} = \mathcal{S}_{1\to 0}(\hat{\mathbf{x}}^i + \epsilon\mathbf{n}, f_\theta^{\mathrm{j}}), \tag{10}$$

where $(t_0, \alpha_0) = (0, 0)$ and $(t_1, \alpha_1) = (1, \epsilon)$. A video by solving our BI-FLOW backward is shown in Fig. 5, with the same quality as forward-solved videos except that it is generated in the opposite direction.

Be continuous in time With a video ODE flow, we can actually model a continuous video trajectory through integration. Currently, the intermediate states in our solution trajectory are simply linear mixes of the two consecutive frames as it is trained to do so, but to our best knowledge, we are the first method to achieve general video generation in decent quality as solving the ODE flow in the same manner of solving initial value problems.

Rather than linear interpolation, any non-linear interpolation in flow matching is possible and actively explored in the community now [1, 22]. It is a promising future direction to obtain infinite temporal resolution in our video flow that we drop in a perceptually interpolating curve during training such that a state at any time point in the solved trajectory is a valid frame.

Limitations One of the main limitations of our method is the context size. Admittedly, our method cannot achieve state-of-the-art quality in video generation, which we mainly attribute to our Markovian sampling where each subsequent frame is generated solely based on the preceding frame. The context size of only one results in artifacts and incoherent generation, *e.g.*, clouds moving in different directions after a few frames and the background gradually morphing to another scene. The small context also complicates flow training with more crossing and ambiguous data involved. A promising direction to resolve this is to straightforwardly apply autoregressive sampling to our method. We leave this for future work.

Another limitation is that we need to manually select the noise level applied in the inference. This relates to the absence of a reliable estimate of how much the frame has corrupted. Although we can quickly trial and see to tune this parameter after training, developing an adaptive strategy to add an adequate but minimal amount of noise could be highly beneficial. For example, we can apply different levels of noise per frame.

6. Conclusion

We propose a new paradigm of video generation, *i.e.*, to learn an ODE flow field that drives from one frame to another. With our proposed reformulated flow matching and bi-linearity training, we train a joint flow field of temporal evolution and denoising, and in inference, we generate the video in a streaming manner while correcting the frame simultaneously. Through extensive evaluations, our method presents better efficiency and stability than the widely used conditional diffusion with comparable quality, lending itself to real-time applications such as interactive video creation.

In future work, we would like to extend our bi-flow design beyond the temporal domain to capture other correlations among image distributions, such as multi-view images that introduce a new dimension of camera angles and multi-illumination images that represent a new dimension of lighting conditions.

Acknowledgments This project was supported by Meta Reality Labs, Grant Nr. 583589. Chen is further supported by an award from The Rabin Ezra Scholarship Trust. We also thank the constructive comments from the anonymous reviewers and early discussions with Xingchang Huang.

References

[1] Michael S. Albergo and Eric Vanden-Eijnden. Building Normalizing Flows with Stochastic Interpolants. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. 1, 8

[2] Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, Yuanzhen Li, Michael Rubinstein, Tomer Michaeli, Oliver Wang, Deqing Sun, Tali Dekel, and Inbar Mosseri. Lumiere: A Space-Time Diffusion Model for Video Generation. *arXiv preprint arXiv:2401.12945*, 2024. 7

[3] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets. *arXiv preprint arXiv:2311.15127*, 2023. 7

[4] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22563–22575, 2023. 7

[5] Valentin De Bortoli, Guan-Horng Liu, Tianrong Chen, Evangelos A. Theodorou, and Weilie Nie. Augmented Bridge Matching. *arXiv preprint arXiv:2311.06978*, 2023. 7

[6] 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 *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. 4, 7

[7] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems*, pages 24081–24125. Curran Associates, Inc., 2024. 7

[8] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018. 7

[9] Aram Davtyan, Sepehr Sameni, and Paolo Favaro. Efficient video prediction via sparsely conditioned flow matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 7

[10] Rohit Girdhar, Mannat Singh, Andrew Brown, Quentin Duval, Samaneh Azadi, Sai Saketh Rambhatla, Akbar Shah, Xi Yin, Devi Parikh, and Ishan Misra. Emu Video: Factorizing Text-to-Video Generation by Explicit Image Conditioning. *arXiv preprint arXiv:2311.10709*, 2023. 7

[11] William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos. In *Advances in Neural Information Processing Systems*, pages 27953–27965. Curran Associates, Inc., 2022. 4, 7

[12] Yingqing He, Tianyu Yang, Yong Zhang, Ying Shan, and Qifeng Chen. Latent Video Diffusion Models for High-Fidelity Long Video Generation. *arXiv preprint arXiv:2211.13221*, 2023. 7

[13] Eric Heitz, Laurent Belcour, and Thomas Chambon. Iterative α -(de)Blending: A Minimalist Deterministic Diffusion Model. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–8, New York, NY, USA, 2023. Association for Computing Machinery. 1, 7

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, pages 6840–6851. Curran Associates, Inc., 2020. 7

[15] 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. 7

[16] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video Diffusion Models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022. 1, 7

[17] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. CogVideo: Large-scale pretraining for text-to-video generation via transformers. In *The Eleventh International Conference on Learning Representations*, 2022. 7

[18] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong MU, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. 7

[19] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations*, 2022. 1, 2, 7

[20] Chen Liu and Tobias Ritschel. Neural Differential Appearance Equations. *ACM Trans. Graph.*, 43(6):256:1–256:17, 2024. 8

[21] Guan-Horng Liu, Arash Vahdat, De-An Huang, Evangelos Theodorou, Weili Nie, and Anima Anandkumar. I^{2}SB: Image-to-Image Schrödinger Bridge. In *Proceedings of the 40th International Conference on Machine Learning*, pages 22042–22062. PMLR, 2023. 7

[22] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *The Eleventh International Conference on Learning Representations*, 2022. 1, 2, 7, 8

[23] Yaniv Nikankin, Niv Haim, and Michal Irani. SinFusion: Training diffusion models on a single image or video. In *Proceedings of the 40th International Conference on Machine*

Learning, pages 26199–26214. PMLR, 2023-07-23/2023-07-29. 8

[24] Mirela Ostrek and Justus Thies. Stable video portraits. In *European Conference on Computer Vision*, 2024. 7

[25] Yehuda Pinchover and Jacob Rubinstein. *An Introduction to Partial Differential Equations*. Cambridge University Press, 1 edition, 2005. 3

[26] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2022. 7

[27] Vaibhav Saxena, Jimmy Ba, and Danijar Hafner. Clockwork Variational Autoencoders. In *Advances in Neural Information Processing Systems*, pages 29246–29257. Curran Associates, Inc., 2021. 4

[28] Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion schrödinger bridge matching. In *Advances in Neural Information Processing Systems*, pages 62183–62223. Curran Associates, Inc., 2023. 7

[29] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. 1, 3

[30] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. FVD: A new metric for video generation. In *DGS@ICLR*, 2019. 4

[31] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *The Thirteenth International Conference on Learning Representations*, 2025. 7

[32] Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, and Jiebo Luo. Learning to Generate Time-Lapse Videos Using Multi-Stage Dynamic Generative Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2364–2373, 2018. 4

[33] Linqi Zhou, Aaron Lou, Samar Khanna, and Stefano Ermon. Denoising Diffusion Bridge Models. In *The Twelfth International Conference on Learning Representations*, 2023. 7