Rolling Diffusion Models

David Ruhe [1 2 *] **Jonathan Heek** [1] **Tim Salimans** [1] **Emiel Hoogeboom** [1]

Abstract

Diffusion models have recently been increasingly applied to temporal data such as video, fluid mechanics simulations, or climate data. These methods generally treat subsequent frames equally regarding the amount of noise in the diffusion process. This paper explores *Rolling Diffusion*: a new approach that uses a sliding window denoising process. It ensures that the diffusion process progressively corrupts through time by assigning more noise to frames that appear later in a sequence, reflecting greater uncertainty about the future as the generation process unfolds. Empirically, we show that when the temporal dynamics are complex, Rolling Diffusion is superior to standard diffusion. In particular, this result is demonstrated in a video prediction task using the Kinetics-600 video dataset and in a chaotic fluid dynamics forecasting experiment.

1. Introduction

Diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020) have significantly boosted the field of generative modeling. They provided the fundaments for large-scale text-to-image systems like DALL-E 2 (Ramesh et al., 2022), Imagen (Saharia et al., 2022), Parti (Yu et al., 2022), and Stable Diffusion (Rombach et al., 2022). Other applications of diffusion models include density estimation, text-to-speech, and image editing (Kingma et al., 2021; Gao et al., 2023; Kawar et al., 2023).

After these successes in these domains, interest in developing diffusion models for time sequences has grown. Prominent recent large-scale works include, e.g., Imagen Video (Ho et al., 2022a), Stable Diffusion Video (StabilityAI, 2023). Other impressive results for generating video data have been achieved by, e.g., (Blattmann et al., 2023; Ge et al., 2023; Harvey et al., 2022; Singer et al., 2022; Ho et al., 2022b). Applications of sequential generative modeling outside video include, e.g., fluid mechanics or weather and climate modeling (Price et al., 2023; Meng et al., 2022; Lippe et al., 2023).

What is common across many of these works is that they treat the temporal axis as an 'extra spatial dimension'. That is, they treat the video as a 3D tensor of shape $K \times H \times W$. This has several downsides. First, the memory and computational requirements can quickly grow infeasible if one wants to generate long sequences. Second, one is typically interested in being able to *roll out* the sampling process for a variable number of time steps. Therefore, an alternative angle is a fully autoregressive approach by conditioning on a sequence of input frames and simulating a single output frame, which is then concatenated to the input frames, upon which the recursion can continue. In this case, one has to traverse the entire denoising diffusion sampling chain for every single frame, which is computationally intensive. Additionally, iteratively sampling single frames leads to quick autoregressive error accumulation. A middle ground can be found by jointly generating blocks of frames. However, in this *block-autoregressive* case, a diffusion model would use the same number of denoising steps for every frame. This is suboptimal since, given a sequence of conditioning frames, the generative uncertainty about the next few is much lower than the frames further into the future. Finally, both methods sample frames only jointly with earlier frames, which is potentially a suboptimal parameterization.

In this paper, we propose a new framework called *Rolling Diffusion*, a method that explicitly corrupts data from past to future. This is achieved by reparameterizing the global diffusion time to a *local time* for each frame. It turns out that by doing this, one can (apart from boundary conditions) completely focus on a local sliding window sequential denoising process. This has several temporal inductive biases, alleviating some of the abovementioned issues.

1. In denoising diffusion models, the model output tends to contain low-frequency information in high noise regimes and includes high-frequency information only when corruptions are light. In our framework, the noise

[*]Work done as a Student Researcher at Google. [1]Google Deepmind, Amsterdam, Netherlands [2]University of Amsterdam, Netherlands. Correspondence to: David Ruhe <david.ruhe@gmail.com>, Jonathan Heek, Tim Salimans, Emiel Hoogeboom <{jheek, salimans, emielh}@google.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

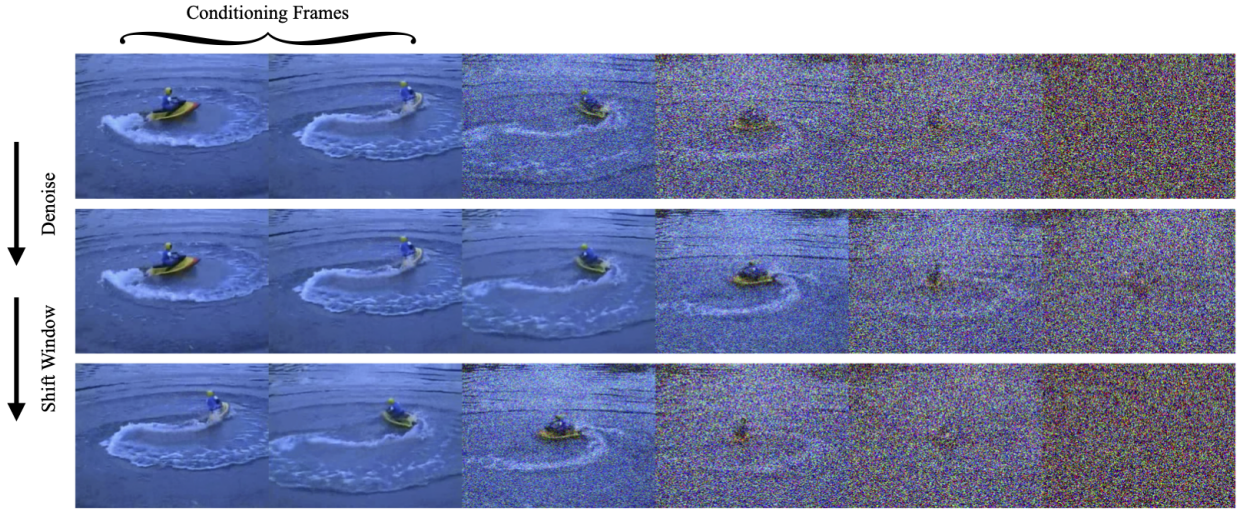

Figure 1. Overview of the Rolling Diffusion rollout sampling procedure. The input to the model contains some conditioning and a sequence of partially denoised frames. The model then denoises the frames by a small amount. After denoising, the sliding window shifts, and the fully denoised frames are concatenated with the conditioning. This process is repeats until the desired number of frames is generated. Example video taken from the Kinetics-600 dataset (Kay et al., 2017) (CC BY 4.0).

strength is higher for frames that are further from the conditioning. As such, the model only needs to predict low-frequency information (i.e., global structures) for frames further into the future; high-frequency information gets included as frames move closer to the present.

2. Each frame is generated together with both a number of preceding and succeeding frames.

3. Due to the local sliding window point of view, every frame enjoys the same inductive bias and undergoes a similar sampling procedure regardless of its absolute position in the video.

These merits are empirically demonstrated in, among others, a video prediction experiment using the Kinetics-600 video dataset (Kay et al., 2017) and in an experiment involving chaotic fluid mechanics simulations.

2. Background: Diffusion Models

2.1. Diffusion

Diffusion models consist of a process that destroys data stochastically, named the 'diffusion process', and a generative process called the denoising process. Let $\boldsymbol{z}_t \in \mathbb{R}^D$ denote a latent variable over a diffusion dimension $t \in [0, 1]$. We refer to t as the *global (diffusion) time*, which will determine the amount of noise added to the data. Given a datapoint $\boldsymbol{x} \in \mathbb{R}^D$, $\boldsymbol{x} \sim q(\boldsymbol{x})$, the diffusion process is designed so that $\boldsymbol{z}_0 \approx \boldsymbol{x}$ and $\boldsymbol{z}_1 \sim \mathcal{N}(0, 1)$ via the distribution:

$$q(\boldsymbol{z}_t|\boldsymbol{x}) := \mathcal{N}(\boldsymbol{z}_t|\alpha_t \boldsymbol{x}, \sigma_t^2 \mathbf{I}) \,, \tag{1}$$

where a_t and σ_t^2 are strictly positive scalar functions of t. We define their *signal-to-noise ratio*

$$\mathrm{SNR}(t) := \frac{\alpha_t^2}{\sigma_t^2} \tag{2}$$

to be *monotonically decreasing* in t. Finally, we let $\alpha_t^2 + \sigma_t^2 = 1$, corresponding to a *variance-preserving* process which also implies $a_t^2 \in (0, 1]$ and $\sigma_t^2 \in (0, 1]$.

Given the noising process, it can be shown (Sohl-Dickstein et al., 2015) that the *true* (i.e., optimal) denoising distribution for a single datapoint $\boldsymbol{x}$ from time t to time s ($s \leq t$) is given by

$$q(\boldsymbol{z}_s|\boldsymbol{z}_t, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}_s|\mu_{t\to s}(\boldsymbol{z}_t, \boldsymbol{x}), \sigma_{t\to s}^2 \mathbf{I}) \,, \tag{3}$$

where μ and σ^2 are *analytical* mean and variance functions of t, s, $\boldsymbol{x}$ and $\boldsymbol{z}_t$. The parameterized generative process $p_\theta(\boldsymbol{z}_s|\boldsymbol{z}_t)$ is then defined by approximating $\boldsymbol{x}$ via a neural network $f_\theta : \mathbb{R}^D \times [0, 1] \to \mathbb{R}^D$. That is, we set

$$p_\theta(\boldsymbol{z}_s|\boldsymbol{z}_t) := q(\boldsymbol{z}_s|\boldsymbol{z}_t, \boldsymbol{x} = f_\theta(\boldsymbol{z}_t, t)) \,. \tag{4}$$

The diffusion objective can be expressed as a KL-divergence between the diffusion process and the denoising process, i.e. $D_{\mathrm{KL}}(q(\boldsymbol{x}, \boldsymbol{z}_0, \ldots, \boldsymbol{z}_1) \,||\, p(\boldsymbol{x}, \boldsymbol{z}_0, \ldots, \boldsymbol{z}_1))$ which simplifies to (Kingma et al., 2021):

$$\begin{aligned}\mathcal{L}_\theta(\boldsymbol{x}) := \mathbb{E}_{t\sim U(0,1), \boldsymbol{\epsilon}\sim\mathcal{N}(0,1)} \left[a(t)||\boldsymbol{x} - f_\theta(\boldsymbol{z}_{t,\boldsymbol{\epsilon}}, t)||^2\right] \\ + \mathcal{L}_{\mathrm{prior}} + \mathcal{L}_{\mathrm{data}} \,,\end{aligned} \tag{5}$$

where $\mathcal{L}_{\mathrm{prior}}$ and $\mathcal{L}_{\mathrm{data}}$ are typically negligible. The weighting $a(t)$ can be freely specified. In practice, it was found

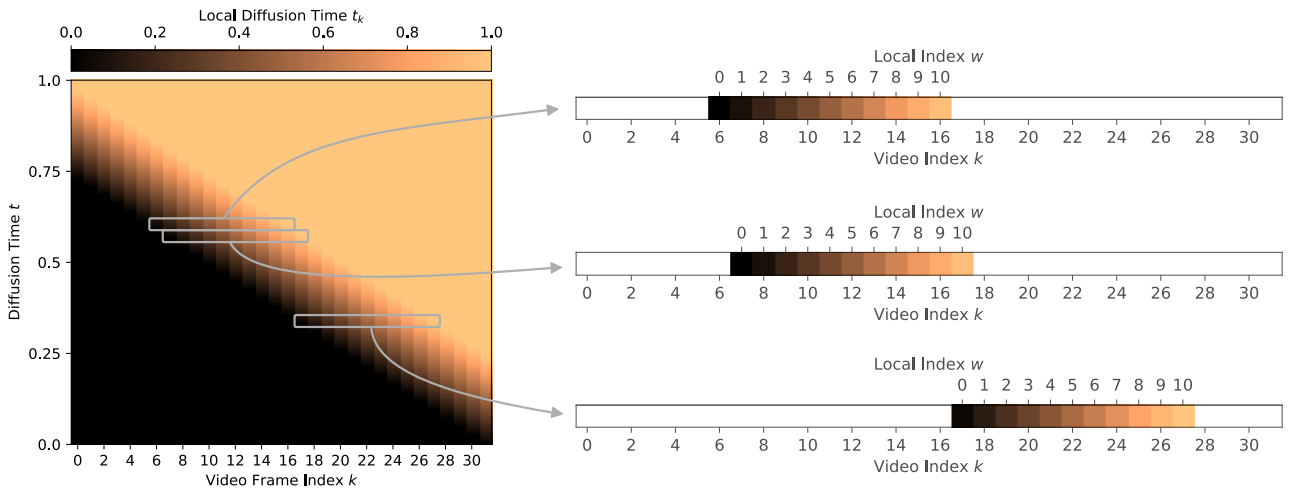

Figure 2. Left: an illustration of a global rolling diffusion process and its local time reparameterization. The global diffusion denoising time t (vertical axis) is mapped to a local time t_k for a frame k (horizontal axis). The local time is then used to compute the diffusion parameters α_{t_k} and σ_{t_k}. On the right, we show how the same *local* schedule can be applied to each sequence of frames based on the frame index w. The nontrivial part of sampling the generative process only occurs in the sliding window as it gets shifted over the sequence.

that specific weightings of the loss result in better sample quality (Ho et al., 2020). This is the case for, e.g., ϵ-*loss*, which corresponds to $a(t) = \mathrm{SNR}(t)$.

2.2. Diffusion for temporal data

If one is interested in generation of temporal data beyond typical hardware constraints, one must consider (autoregressive) conditional extension of previously generated data. I.e., given an initial sample $\boldsymbol{x}^k$ at a temporal index k, we want to sample a (faithful) conditional distribution $p(\boldsymbol{x}^{k+1}|\boldsymbol{x}^k)$. This process can then be extended to videos of arbitrary lengths. As discussed in Section 1, it is not yet clear what kinds of parameterization choices are optimal to estimate this conditional distribution. Further, no temporal inductive bias is typically baked into the denoising process.

3. Rolling Diffusion Models

We introduce *rolling diffusion models*, merging the arrow of time with the (de)noising process. To formalize this, we first have to discuss the global diffusion model. We will see that the only nontrivial parts of the global process take place locally. Defining the noise schedule locally is advantageous since the resulting model does not depend on the number of frames K and can be unrolled indefinitely.

3.1. A global perspective

Let $\boldsymbol{x} \in \mathbb{R}^{D\times K}$ be a time series datapoint where K denotes the number of frames and D the dimensionality of each frame. The core idea that allows rolling diffusion is a *reparameterization* of the diffusion (denoising) time t to a frame-dependent *local (frame-dependent) time*: i.e.,

$$t \mapsto t_k\,. \tag{6}$$

Note that we still require $t_k \in [0, 1]$ for all $k \in \{0, \dots, K-1\}$. Furthermore, we still have a monotonically decreasing signal-to-noise schedule, ensuring a well-defined diffusion process. However, we now have a different signal-to-noise schedule for each frame. In this work, we also always have $t_k \leq t_{k+1}$, i.e., the local denoising time of a given frame is smaller than the local time of the next frame. This means we add more noise to future frames: a natural temporal inductive bias. Note that this is not strictly required; one could also have a reverse-time inductive bias or a mixture. An example of such a reparameterization is shown in Figure 2 (left). We depict a map that takes a global diffusion time t (vertical axis) and a frame index k (horizontal axis), and computes a local time t_k, indicated with a color intensity.

Forward process We now redefine the forward process using the local time:

$$q(\boldsymbol{z}_t|\boldsymbol{x}) := \prod_{k=0}^{K-1} \mathcal{N}(\boldsymbol{z}_t^k|\alpha_{t_k}\boldsymbol{x}^k, \sigma_{t_k}^2\mathbf{I})\,, \tag{7}$$

where we can *reuse* the α and σ functions (now evaluated locally at t_k) from before. Here, $\boldsymbol{x}^k$ denotes the k-th frame of $\boldsymbol{x}$.

True backward process and generative process Given a tuple (s, t), $s \in [0, 1]$, $t \in [0, 1]$, $s \leq t$, we can divide the

frames $k \in \{0, \dots, K-1\}$ into three categories:

$$\text{clean}(s,t) := \{k \mid s_k = t_k = 0\}\,, \tag{8}$$

$$\text{noise}(s,t) := \{k \mid s_k = t_k = 1\}\,, \tag{9}$$

$$\text{win}(s,t) := \{k \mid s_k \in [0,1), t_k \in (s_k, 1]\}\,. \tag{10}$$

Note that here, s and t are both diffusion time-steps (corresponding to certain SNR levels), while k denotes a frame index. This categorization can be motivated using the schedule depicted in Figure 2. Given, for example, $t = 0.5$ and $s = 0.375$, we see that the first frame $k = 0$ falls in the first category. At this point in time, $\boldsymbol{z}_{t_0} = \boldsymbol{z}_{s_0}$ are identical given that $\lim_{t\to 0^+} \log \text{SNR}(t) = \infty$. On the other hand, the last frame $k = K-1$ (31 in the figure) falls in the second category, i.e., both $\boldsymbol{z}_{t_{K-1}}$ and $\boldsymbol{z}_{s_{K-1}}$ are distributed as independent standard Gaussians, given that $\lim_{t\to 1^-} \log \text{SNR}(t) = -\infty$. Finally, the frame $k = 16$ falls in the third, most interesting category: the sliding window. As such, observe that the *true* denoising process can be factorized as:

$$q(\boldsymbol{z}_s|\boldsymbol{z}_t, \boldsymbol{x}) = q(\boldsymbol{z}_s^{\text{clean}}|\boldsymbol{z}_t, \boldsymbol{x}) q(\boldsymbol{z}_s^{\text{noise}}|\boldsymbol{z}_t, \boldsymbol{x}) q(\boldsymbol{z}_s^{\text{win}}|\boldsymbol{z}_t, \boldsymbol{x}). \tag{11}$$

This is helpful because we will see that the only frames that need to be modeled are in the window. Namely, the first factor has

$$q(\boldsymbol{z}_s^{\text{clean}}|\boldsymbol{z}_t, \boldsymbol{x}) = \prod_{k \in \text{clean}(s,t)} \delta(\boldsymbol{z}_s^k | \boldsymbol{z}_t^k)\,. \tag{12}$$

In other words, if $\boldsymbol{z}_t^k$ is already noiseless, then $\boldsymbol{z}_s^k$ will also be noiseless. Regarding the second factor, we see that they are all independently normally distributed:

$$q(\boldsymbol{z}_s^{\text{noise}}|\boldsymbol{z}_t, \boldsymbol{x}) = \prod_{k \in \text{noise}(s,t)} \mathcal{N}(\boldsymbol{z}_s^k | 0, \mathbf{I}). \tag{13}$$

Simply put, in these cases $\boldsymbol{z}_s^k$ is independent noise and does not depend on data at all. Finally, the third factor has a true non-trivial denoising process:

$$q(\boldsymbol{z}_s^{\text{win}}|\boldsymbol{z}_t, \boldsymbol{x}) = \prod_{k \in \text{win}(s,t)} \mathcal{N}(\boldsymbol{z}_s^k | \mu_{t_k \to s_k}(\boldsymbol{z}_t^k, \boldsymbol{x}^k), \sigma^2_{t_k \to s_k} \mathbf{I})$$

where $\mu_{t_k \to s_k}$ and $\sigma^2_{t_k \to s_k}$ are the analytical mean and variance functions. Note that we can then optimally (w.r.t. a KL-divergence) factorize the generative process similarly:

$$p_\theta(\boldsymbol{z}_s|\boldsymbol{z}_t) := p(\boldsymbol{z}_s^{\text{clean}}|\boldsymbol{z}_t) p(\boldsymbol{z}_s^{\text{noise}}|\boldsymbol{z}_t) p_\theta(\boldsymbol{z}_s^{\text{win}}|\boldsymbol{z}_t)\,, \tag{14}$$

with $p(\boldsymbol{z}_s^{\text{clean}}|\boldsymbol{z}_t) := \prod_{k \in \text{clean}(s,t)} \delta(\boldsymbol{z}_s^k | \boldsymbol{z}_t^k)$ and $p(\boldsymbol{z}_s^{\text{noise}}|\boldsymbol{z}_t) := \prod_{k \in \text{noise}(s,t)} \mathcal{N}(\boldsymbol{z}_s^k | 0, \mathbf{I})$. The only 'interesting' parameterized part of the generative process then has

$$p_\theta(\boldsymbol{z}_s^{\text{win}}|\boldsymbol{z}_t) := \prod_{k \in \text{win}(s,t)} q(\boldsymbol{z}_s^k | \boldsymbol{z}_t, \boldsymbol{x}^k = f_\theta(\boldsymbol{z}_t, t_k)). \tag{15}$$

In other words, we can *only focus the generative process on the frames that are in the sliding window*. Finally, note that we can choose to not condition the model on all $\boldsymbol{z}_t^k$ that have $t_k = 0$, since frames that are far in the past are likely to be independent of the current frame, and this excessive conditioning would exceed computational constraints. As such, we get

$$p_\theta(\boldsymbol{z}_s^{\text{win}}|\boldsymbol{z}_t) = p_\theta(\boldsymbol{z}_s^{\text{win}}|\boldsymbol{z}_t^{\text{clean}}, \boldsymbol{z}_t^{\text{win}}) \tag{16}$$

$$:\approx p_\theta(\boldsymbol{z}_s^{\text{win}}|\widehat{\boldsymbol{z}_t^{\text{clean}}}, \boldsymbol{z}_t^{\text{win}}) \tag{17}$$

where $\widehat{\boldsymbol{z}_t^{\text{clean}}}$ denotes a specific subset of $\boldsymbol{z}_t^{\text{clean}}$, typically including a few frames slightly before the current sliding window.

In Appendix A, we motivate, in addition to the arguments above, the following loss function:

$$\mathcal{L}_{\text{win},\theta}(\boldsymbol{x}) := \mathbb{E}_{t \sim U(0,1), \boldsymbol{\epsilon} \sim \mathcal{N}(0,1)} \left[L_{\text{win},\theta}(\boldsymbol{x}; t, \boldsymbol{\epsilon}) \right] \tag{18}$$

with

$$L_{\text{win},\theta} := \sum_{k \in \text{win}(t)} a(t_k) ||\boldsymbol{x}^k - f_\theta^k(\boldsymbol{z}_{t,\boldsymbol{\epsilon}}^{\text{win}}, \boldsymbol{z}_{t,\boldsymbol{\epsilon}}^{\text{clean}}, t)||^2\,,$$

where we suppress some arguments for notational convenience. Here, $\boldsymbol{z}_{t,\boldsymbol{\epsilon}}$ denotes a noised version of $\boldsymbol{x}$ as a function of t and $\boldsymbol{\epsilon}$ and $a(t_k)$ is a weighting function leading to, e.g., the usual 'simple' ϵ-MSE loss, v-MSE loss, or x-MSE loss.

Observe Figure 2 again. After training is completed, we can essentially sample from the generative model by traversing the image with the sliding window from the top left to the bottom right.

3.2. A local perspective

In the previous section, we discussed how rolling diffusion enables us to concentrate entirely on frames within a sliding window. Instead of using t to represent the global diffusion time, which determines the noise level for all frames, we now redefine t to determine the noise level for each frame in a smaller subsequence. Specifically, running the denoising chain from $t = 1$ to $t = 0$ will sample a sequence such that the first frame is completely denoised, but the subsequent frames still retain some noise. In contrast, the global process described earlier denoises an entire video.

Similar to before, we reparameterize t to allow for different noise levels for each frame in the sliding window. This reparameterization should be:

1. *local*, meaning we allow for sharing and reusing the parameterization across various positions of the sliding window, independent of their absolute locations.

2. *consistent under moving the window*, meaning that the noise level for the current frame w when $t = 0$ should match the noise level at $w-1$ at $t = 1$. This consistency enables seamless denoising as the window slides, ensuring that each frame is progressively denoised while shifting positions.

Let $W < K$ be the size of the sliding window, and $w \in \{0, \dots, W-1\}$ be the local indices of the frames. To satisfy the first assumption, we define the schedule in terms of the local index w (see Figure 2 (right)):

$$t \mapsto t_w^W \,. \tag{19}$$

For the second, we know t_w^W must have

$$t_w^W = g\left(\frac{w+t}{W}\right) \tag{20}$$

for some monotonically increasing (in t) function $g : [0,1] \to [0,1]$. We will sometimes suppress W for notational convenience. Note that due to the locality of the parameterization, the process can be unfolded indefinitely at test time.

A linear reparameterization In this work we typically put $g := \mathrm{id}$, i.e.,

$$t_w^{\text{lin}} = \frac{w+t}{W} \,. \tag{21}$$

See Figure 2 (right) for an illustration of how this local schedule is applied to each sequence of frames. Observe that $t_w^{\text{lin}} \in [w/W, (w+1)/W] \subseteq [0,1]$. As such, we can directly use our SNR schedule to compute the diffusion parameters for each frame.

One can extend the linear local time to include clean conditioning frames. Let n_{cln} denote the number of clean frames (chosen as a hyperparameter), then the local time for a frame w is:

$$t_w^{\text{lin}}(n_{\text{cln}}) := \text{clip}\left(\frac{w+t-n_{\text{cln}}}{W-n_{\text{cln}}}\right) \,, \tag{22}$$

where $\text{clip} : \mathbb{R} \to [0,1]$ clips value between 0 and 1.

3.3. Boundary conditions

While framing rolling diffusion purely from a local perspective is convenient for training and sampling, it introduces some complicated edge conditions. Given the linear local time reparameterization t_w^{lin}, we have that given the diffusion time t running from 1 to 0, the local times run from $(\frac{1}{W}, \frac{2}{W}, \dots \frac{W}{W})$ to $(\frac{0}{W}, \frac{1}{W}, \dots, \frac{W-1}{W})$. This means that using this setting, the signal-to-noise ratios are never minimal for all frames, meaning we cannot start sampling from a completely noisy state. Visually, in Figure 2, the rolling sampling procedure can be seen as moving the sliding window over the diagonal from top left to bottom right such that the local times of the frames remain invariant upon shifting. However, this means that placing the window at the very left edge still results in having partially denoised frames.

To account for this, we *co-train* the rolling diffusion model with an additional schedule that can handle this boundary condition:

$$t_w^{\text{init}} := \text{clip}\left(\frac{w}{W} + t\right) \,. \tag{23}$$

This *init* noise schedule can start from initial random noise and generates a video in the 'rolling state'. That is, at diffusion time $t = 1$, this will put all frames to maximum noise, and at $t = 0$ the frames will be in the rolling state. To be precise, it starts from local times $(1, 1, \dots 1)$ and denoises to $(0, \frac{1}{W}, \frac{2}{W}, \dots, \frac{W-1}{W})$, after which we can start using the previously described t_w^{lin} schedule. From a visual perspective, in Figure 2, this corresponds to placing the window at the upper left corner and moving it down *vertically*, until it reaches the stage where it can start moving diagonally.

On the interval $[0, \frac{1}{W}]$, this schedule contains the previous local schedule t_w^{lin} as a special case. To see this, note that $t_w^{\text{init}}(\frac{1}{W}) = \left(\frac{1}{W}, \frac{2}{W}, \dots, 1\right)$, which is the same as $t_w^{\text{lin}}(1)$. Similarly, one can check the case for $t_w^{\text{init}}(0)$. This means that the model could be trained solely with t_w^{init} to handle the boundaries as well as being able to roll out indefinitely. At sampling time, one then still uses the t_w^{init} schedule but restricted to $[0, \frac{1}{W}]$. The caveat, however, is that the t_w^{lin} schedule only gets selected $1/W$ of the time during training (assuming $t \sim U(0,1)$). In contrast, this schedule is used almost exclusively at test time, with the exception being at the boundary. As such, we find it beneficial to oversample the t_w^{lin} schedule during training based on a Bernoulli rate β.

3.4. Local training

We briefly discuss training details under the aforementioned local time reparameterizations. We now consider $\boldsymbol{x} \in \mathbb{R}^{D \times W}$, chunking videos into blocks of W frames. From t_w one can compute α_{t_w} and σ_{t_w} using typical SNR schedules. Let $\boldsymbol{z} \in \mathbb{R}^{D \times W}$, we have as the forward (noising) process

$$q(\boldsymbol{z}_t|\boldsymbol{x}) := \prod_{w=0}^{W} \mathcal{N}(\boldsymbol{z}_t^w | \alpha_{t_w} \boldsymbol{x}^w, \sigma_{t_w}^2 \mathbf{I}), \tag{24}$$

where $\boldsymbol{x}^w$ denotes the w-th frame of $\boldsymbol{x}$.

The training objective becomes

$$\mathcal{L}_{\text{loc},\theta}(\boldsymbol{x}) := \mathbb{E}_{t \sim U(0,1), \boldsymbol{\epsilon} \sim \mathcal{N}(0,1)} \left[L_{\text{loc},\theta}(\boldsymbol{x}; t, \boldsymbol{\epsilon}) \right] , \tag{25}$$

where we put

$$L_{\text{loc},\theta}(\boldsymbol{x}, t, \boldsymbol{\epsilon}) := \sum_{w=0}^{W} a(t_w) ||\boldsymbol{x}^w - f_\theta^w(\boldsymbol{z}_{t,\boldsymbol{\epsilon}}; t)||^2 \,. \tag{26}$$

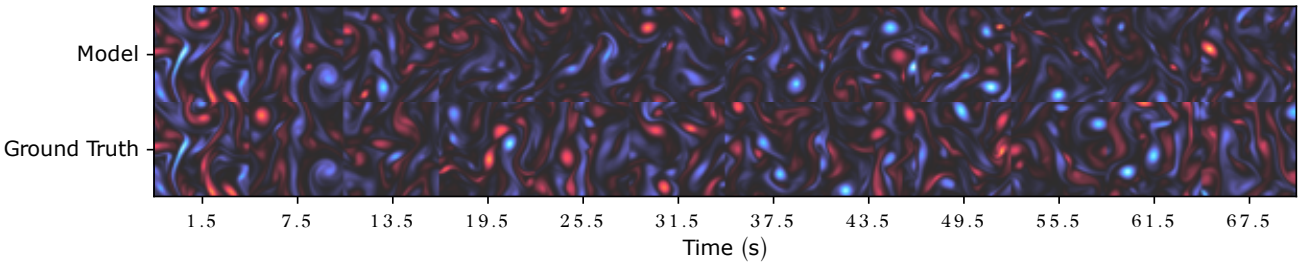

Figure 3. Sample Kolmogorov flow rollout. We observe that ground-truth structures are preserved initially, but the model diverges from the true data later on. Despite this, model is able to generate new turbulent dynamics much later on in the sequence.

Algorithm 1 Rolling Diffusion: Training

Require: $\mathcal{D}_{\text{tr}} := \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, $\boldsymbol{x} \in \mathbb{R}^{D \times W}$, n_{cln}, β, f_θ
repeat
  Sample $\boldsymbol{x}$ from $\mathcal{D}_{\text{tr}}$, $t \sim U(0,1)$, $y \sim B(\beta)$
  if y **then**
    Compute local time $t_w^{\text{init}}(n_{\text{cln}})$, $w = 0, \dots, W-1$
  else
    Compute local time $t_w^{\text{lin}}(n_{\text{cln}})$, $w = 0, \dots, W-1$
  end if
  Compute α_{t_w} and σ_{t_w} for all $w = 0, \dots, W-1$
  Sample $\boldsymbol{z}_t \sim q(\boldsymbol{z}_t|\boldsymbol{x})$ using Equation (24) (reparameterized from $\boldsymbol{\epsilon} \sim \mathcal{N}(0,1)$)
  Compute $\hat{\boldsymbol{x}} \leftarrow f_\theta(\boldsymbol{z}_{t,\boldsymbol{\epsilon}}; t)$
  Update θ using $L_{\text{loc},\theta}(\boldsymbol{x}; t, \boldsymbol{\epsilon})$
until Converged

Algorithm 2 Rolling Diffusion: Rollout

Require: p_θ, n_{cln}, $\boldsymbol{z}_0$ with local diffusion times $(0/W, \dots, (W-1)/W)$ (i.e., progressively noised).
Video Prediction $\hat{\boldsymbol{x}} \leftarrow \{\boldsymbol{z}_0^{n_{\text{cln}}}\}$
repeat
  Sample $\boldsymbol{z}^W \sim \mathcal{N}(0, \mathbf{I})$
  $\boldsymbol{z}_1 \leftarrow \{\boldsymbol{z}_0^1, \dots, \boldsymbol{z}_0^{W-1}, \boldsymbol{z}^W\}$
  for $t = 1, (T-1)/T, \dots, 1/T$ **do**
    Compute local times $t_w^{\text{lin}}(n_{\text{cln}})$, $w = 0, \dots, W-1$
    Sample $\boldsymbol{z}_{t-1/T} \sim p_\theta(\boldsymbol{z}_{t-1/T}|\boldsymbol{z}_t)$
  end for
  $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}} \cup \{\boldsymbol{z}_0^{n_{\text{cln}}}\}$
until Completed

The training and sampling procedures are summarized in Algorithm 1 and Algorithm 2. We summarize sampling at the boundary using t_w^{init} in Appendix B. Furthermore, we provide a visual of the rolling sampling loop in Figure 1.

4. Related Work

4.1. Video diffusion

Video diffusion has been studied and applied directly in pixel space (Ho et al., 2022a;b; Singer et al., 2022) and in latent space (Blattmann et al., 2023; Ge et al., 2023; He et al., 2022; Yu et al., 2023c), the latter typically empirically being slightly more effective. Furthermore, these videos usually extend the two-dimensional image setting to three (two spatial dimensions and one temporal dimension) without considering autoregressive extension.

Methods that explore autoregressive video generation include Yang et al. (2023); Harvey et al. (2022). Directly parameterizing the conditional distribution of future frames given past frames is preferable (Harvey et al., 2022; Tashiro et al., 2021) compared to adapting the denoising schedule of an unconditional diffusion model. Unlike previous approaches, Rolling Diffusion explicitly introduces a notion of time in the training procedure. Harvey et al. (2022) compare various conditioning schemes but do not explicitly consider a temporally adapted noise schedule.

4.2. Other time-series diffusion models

Apart from video, sequential diffusion models have also been applied to other modes of time-series data, such as audio (Kong et al., 2021), text (Li et al., 2022), but also scientifically to weather data (Price et al., 2023) or fluid mechanics (Kohl et al., 2023). Lippe et al. (2023) show that incorporating a diffusion-inspired denoising procedure can help recover high frequency information that typically gets lost when using learned numerical PDE solver emulators. Dyffusion (Cachay et al., 2023) employs a forecasting and interpolation model in a two-stage fashion. Both models are trained with MSE objectives. This means that the generative model can be interpreted as factorized Gaussians, potentially leading to blurry predictions when generating stochastic or highly chaotic data as considered in this work. Wu et al. (2023) also study autoregressive models with specialized noising schedules, focusing mostly on text generation. Finally, Zhang et al. (2023) published around the same time as the current work a paper with similar ideas as rolling diffusion. There are, however, some core differences. This work analyzes in more detail how rolling diffusion relates to a global, well-defined diffusion process, motivating why

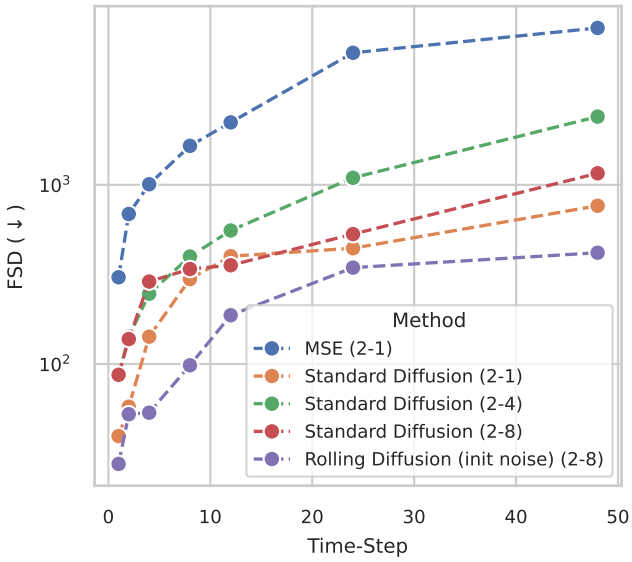

Figure 4. FSD results of the Kolmogorov Flow rollout experiment. Lower is better.

training on a sliding window is acceptable. We isolate the effect of rolling diffusion and study it in more detail, while Zhang et al. (2023) combine local noise levels with additional losses, potentially blurring the effect of the sliding window idea with the impact of these auxiliary terms. Finally, we introduce rolling diffusion schedules that deal with the boundary situations, allowing for generating sequences in an end-to-end manner.

5. Experiments

We conduct experiments using data from various domains and explore several conditioning settings. In all our experiments, we use the *Simple Diffusion* architecture (Hoogeboom et al., 2023) with equal parameters for both standard and rolling diffusion. We use two-dimensional spatial convolution blocks after which we have transformer blocks that attend both spatially and temporally in the deepest layers. Hyperparameter settings can be found in Appendix C. A note on the runtime complexity can be found in Appendix D.

5.1. Kolmogorov Flow

First, we run an experiment on simulated fluid dynamics from JaxCFD (Kochkov et al., 2021; Dresdner et al., 2022). Specifically, we use the *Kolmogorov flow*, an instance of the incompressible Navier-Stokes equations. Recently, there has been increasing interest in emulating classical numerical PDE integrators with machine learning models. Various results have shown that these have the capacity to simulate from initial conditions complex systems to high precision (e.g., Li et al. (2020)). To similar ends, generative models are of increasing interest, as they provide several benefits. First, they provide a way to directly obtain marginal distributions over a future state of a physical system, as opposed to numerically rolling out an ensemble of initial conditions. This especially has use-cases in weather or climate modeling, fluid mechanics analyses, and stochastic differential equation studies. Second, they can improve modeling high frequency information over approaches based on mean-squared error objectives.

The simulation is based on the following partial differential equation $\frac{\partial \mathbf{u}}{\partial \tau} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = \nu \nabla^2 \mathbf{u} - \frac{1}{\rho} \nabla p + \mathbf{f}$, with $\mathbf{u} : [0, \mathcal{T}] \times \mathbb{R}^2 \to \mathbb{R}^2$ is the solution, $\otimes$ the tensor product, ν the kinematic viscosity, ρ the fluid density, p the pressure field, and, finally, $\mathbf{f}$ the external forcing. We randomly sample viscosities ν between $5 \cdot 10^{-4}$ and $5 \cdot 10^{-3}$, and densities between 0.5 and 2, making the task of predicting the dynamics non-deterministic. The ground truth data is generated using a finite volume-based direct numerical simulation (DNS) with a maximal time step of 0.05s with 6 000 time-steps, corresponding to 300 seconds of simulation time, subsampled every 1.5s. Note that this is much longer than typical simulations, which only consider simulation times in the order of tens of seconds. We simulate 200 000 such samples at 64×64, corresponding to 2 terabytes of data. Due to the chaotic nature and the long simulation time, a model cannot be expected to predict the exact future state, which makes it an ideal dataset to test long temporal rollouts. Details on the simulation settings can be found in Appendix E. The model is given 2 input frames containing the horizontal and vertical velocities. Note that the long rollout lengths (up to $\pm$ 60 seconds), together with the large 1.5s strides, make this a more challenging task than the usual 'neural emulator' task, where the predictions can be extremely accurate due to the shorter time-scales and higher temporal resolutions. For example, Lippe et al. (2023); Sun et al. (2023) only go up to ± 15s with much smaller strides.

Evaluation Typical PDE-emulation tasks directly compare models to ground truth using data-space RMSE scores. However, in our uncertain setting we are more interested in how well the generated distribution matches the target distribution. The ground-truth simulator provides no inherent uncertainty estimation, and running ensembles from various initial parameters is not straightforward due to a lack of proposal distribution for these initial settings. Instead, we propose a method similar to *Fréchet Inception Distance* (FID) or FVD. We make use of the fact that spatial frequency intensities provide a good summary statistic (Sun et al., 2023; Dresdner et al., 2022; Kochkov et al., 2021). Let $\boldsymbol{f}_{\boldsymbol{x}} \in \mathbb{R}^F$ denote a vector of spatial-spectral magnitudes computed using a (two-dimensional) *Discrete Fourier Transform* (DFT) from a variable $\boldsymbol{x}$. Then, $\mathbf{F}_{\mathcal{D}} \in N \times F$ denotes all the

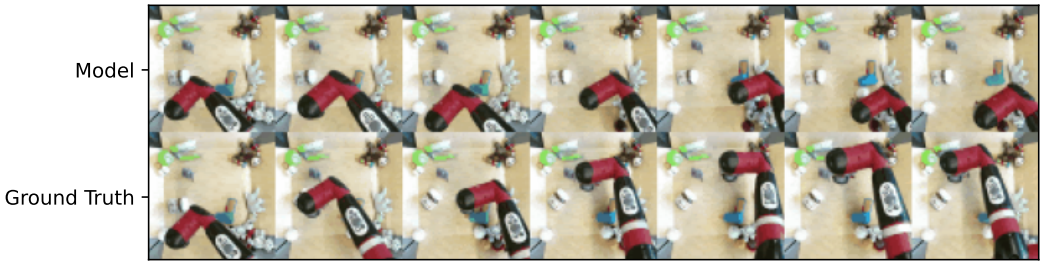

Figure 5. Top: Rolling Diffusion rollout on the BAIR Robot Pushing dataset. Bottom: ground-truth.

frequencies of a (test-time) dataset $\mathcal{D} := \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. Let $\mathbf{F}_\theta$ denote the Fourier magnitudes of N samples from a generative model. We now compute the Frećhet distance between the generated samples and the true data by setting

$$\begin{aligned} &\mathrm{FSD}(\mathcal{D}, \theta) \\ &:= \|\bar{\boldsymbol{f}}_{\mathcal{D}} - \bar{\boldsymbol{f}}_\theta\|^2 + \mathrm{tr}\Big(\boldsymbol{\Sigma}_{\mathcal{D}} + \boldsymbol{\Sigma}_\theta - 2(\boldsymbol{\Sigma}_{\mathcal{D}}\boldsymbol{\Sigma}_\theta)^{1/2}\Big) \quad (27) \end{aligned}$$

where $\bar{\boldsymbol{f}}$ and $\boldsymbol{\Sigma}$ denote the mean and covariance of the frequencies, respectively. We call this metric the *Fréchet Spectral Distance* (FSD).

We provide an example rollout in Figure 3, where we plot the *vorticity* of the velocity field. Quantitatively, we present in Figure 4 the FSD results derived from the horizontal velocity fields of the fluid. Note that the standard diffusion (n_{cln}-1) baselines can be seen as a adaptations of PDE Refiner (Lippe et al., 2023) and (Kohl et al., 2023) for our task. Regarding rolling diffusion, we use the $t_w^{\text{init}}(n_{\text{cln}})$ reparameterization with $n_{\text{cln}} = 2$, and use $t_w^{\text{lin}}(n_{\text{cln}})$ for long rollouts. It is clear that an autoregressive MSE-based model, as typically used in the literature, is not suitable for this task. For standard diffusion, we iteratively generate $W - n_{\text{cln}}$ frames, after which we concatenate these to the conditioning and continue the rollout. Rolling diffusion always shifts the window by one, sampling using the process described before. Rolling diffusion consistently outperforms standard diffusion methods, regardless of conditioning settings and window sizes $(n_{\text{cln}}, W - n_{\text{cln}})$. Additional qualitive and numerical results can be found in Appendix F.

5.2. BAIR Robot Pushing Dataset

The Berkeley AI Research (BAIR) robot pushing dataset (Ebert et al., 2017) is a standard benchmark for video prediction. It contains 44 000 videos at 64×64 of a robot arm pushing objects around. Following previous methods, we condition in on 1 frame and predict the next 15. We evaluate, consistently with previous works, using the *Frechét Video Distance* (FVD) (Unterthiner et al., 2019). For FVD, we use the I3D network (Carreira & Zisserman, 2017) by comparing 100×256 model samples against the 256 examples in the evaluation set.

Method	FVD (↓)
DVD-GAN (Clark et al., 2019)	109.8
VideoGPT (Yan et al., 2021)	103.3
TrIVD-GAN-FP	103.3
Transframer (Nash et al., 2022)	100
CCVS (Le Moing et al., 2021)	99
VideoTransformer (Weissenborn et al., 2019)	94
FitVid (Babaeizadeh et al., 2021)	93.6
NUWA (Wu et al., 2022)	86.9
Video Diffusion (Ho et al., 2022b)	66.9
Standard Diffusion (Ours)	**59.7**
Rolling Diffusion (Ours)	**59.6**

Table 1. Results of the BAIR Robot Pushing baseline experiment.

Regarding rolling diffusion, we use the $t_w^{\text{init}}(n_{\text{cln}})$ reparameterization to sample the $W = 16$ ($n_{\text{cln}} = 1$) frames to a partially denoised state, and then use $t_w^{\text{lin}}(n_{\text{cln}})$ to rollout and complete the sampling. Note that standard diffusion samples all 15 frames at once and might be at an advantage since we do not consider autoregressive extension.

The results are shown in Table 1. We observe that both standard diffusion and rolling diffusion using the same (Simple Diffusion) architecture outperform previous methods. Additionally, we see that there is no significant difference between the standard and rolling framework in this setting. This is because the sampled sequences are, in both cases, indistinguishable from the true data Figure 5.

5.3. Kinetics-600

Finally, we evaluate video prediction on the Kinetics-600 benchmark (Kay et al., 2017; Carreira et al., 2018). It contains approximately 400 000 training videos depicting 600 different activities rescaled to 64×64. We run two experiments using this dataset. The first is a baseline experiment in a setting equal to previously published works. The next one specifically tests Rolling Diffusion's ability to autoregressively rollout for long sequences.

Baseline We compare against previous methods using 5 input frames and 11 output frames, and show the results

Figure 6. Top: Rolling Diffusion rollout on the Kinetics-600 dataset. Bottom: ground-truth. License: CC BY 4.0.

Method	FVD (↓)
Phenaki (Villegas et al., 2022)	36.4
TrIVD-GAN-FP (Luc et al., 2020)	25.7†
Video Diffusion (Ho et al., 2022b)	16.2
RIN (Jabri et al., 2022)	10.8
MAGVIT (Yu et al., 2023a)	9.9†
MAGVITv2 (Yu et al., 2023b)	4.3†
W.A.L.T.-L (Gupta et al., 2023)	**3.3**†
Rolling Diffusion (Ours)	5.2
Standard Diffusion (Ours)	**3.9**

Table 2. FVD results of the Kinetics-600 baseline task (stride 1). Two-stage methods are indicated with '†'.

	Method	Cond-Gen	FVD (↓)
stride=8	Standard Diffusion	(5-11)	58.1
steps=24	Rolling $\beta = 0.1$	(5-11)	**39.8**
stride=1	Standard Diffusion	(15-1)	1369
steps=64	Standard Diffusion	(8-8)	157.1
	Standard Diffusion	(5-11)	**123.7**
	Rolling $\beta = 0.9$	(5-11)	211.2

Table 3. Kinetics-600 (Rollout) with 8192 FVD samples, 100 sampling steps per 11 frames. Trained for 300k iterations.

in Section 5.3 The evaluation metric is again FVD. We note that many of the current SOTA methods are *two-stage*, meaning that they use an autoencoder and run diffusion in latent space. While empirically compelling (Rombach et al., 2022), this makes it hard to isolate the effect of the diffusion model itself. Note that it is not always clear whether the autoencoder parameters are included in the parameter count for two-stage methods, or on what data they are pretrained. Running diffusion using the standard diffusion U-ViT architecture achieves an FVD of 3.9, which is comparable to the best two-stage methods. Rolling diffusion has a strong disadvantage in this case: (1) the baseline generates all frames at once, thereby not suffering from autoregressive errors; and (2) with a stride of 1, there is very little dynamics in the 16 frames, mostly suitable for a standard diffusion model. Still, rolling diffusion achieves a competitive FVD of 5.2.

Rollout Next, we compare the models' capabilities to autoregressively rollout and show the results in Table 3. All settings use a window size of $W = 16$ frames, exploring several n_{cln} settings, denoted with 'Cond-Gen'. During training, we mix t_w^{lin} with a slightly adjusted version of t_w^{lin} (see Appendix G) at an oversampling rate of β.

We analyze two settings, one with a stride (also known as frame-skip or frame-step) of 1, rolling out for 64 steps, and another setting with a stride of 8 rolling out for 24 steps, effectively predicting ahead up to the 192th frame. In the first setting, standard diffusion performs better, quite possibly due to the invariability of the data. We oversample the linear rolling schedule with a rate of $\beta = 0.9$ to account for the high number of test-time steps. Rolling diffusion consistently wins in the second setting, which is much more dynamic. Note also that single-frame diffusion significantly underperforms here and that larger block autoregression is favorable. See an example rollout in Figure 6. Additionally, we compare to TECO (Yan et al., 2023), which uses slightly different input-output settings, in Appendix F. From Appendix H, we get an indication of the effect of the oversampling rate on the performance of rolling diffusion. In this case, slightly oversampling t_w^{lin} yields the best result.

From these (and previous) results we draw the conclusion that rolling diffusion is particularly effective in dynamic settings, where the data is highly variable.

6. Conclusion

We presented *Rolling Diffusion Models*, a new DDPM framework that progressively noises (and denoises) data through time. Validating our method on video and fluid mechanics data, we observed that rolling diffusion's natural inductive bias gets most effectively exploited when the data is highly dynamic. In this setting, Rolling Diffusion outperforms various existing methods. This allows for exciting future directions in, e.g., video, audio, and weather or climate modeling.

Impact Statement

Sequential generative models, including diffusion models, have a significant societal impact with applications in video generation and scientific research by enabling fast, highly detailed sampling. While they offer the upside of creating more accurate and compelling synthesis in fields ranging from climate modeling to medical imaging, there are notable downsides regarding content authenticity and originality of digital media.

References

Babaeizadeh, M., Saffar, M. T., Nair, S., Levine, S., Finn, C., and Erhan, D. Fitvid: Overfitting in pixel-level video prediction. *arXiv preprint arXiv:2106.13195*, 2021.

Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S. W., Fidler, S., and Kreis, K. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22563–22575, 2023.

Cachay, S. R., Zhao, B., James, H., and Yu, R. Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *arXiv preprint arXiv:2306.01984*, 2023.

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., and Zisserman, A. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

Clark, A., Donahue, J., and Simonyan, K. Adversarial video generation on complex datasets. *arXiv preprint arXiv:1907.06571*, 2019.

Dresdner, G., Kochkov, D., Norgaard, P., Zepeda-Núñez, L., Smith, J. A., Brenner, M. P., and Hoyer, S. Learning to correct spectral methods for simulating turbulent flows. In *arXiv*, 2022. doi: 10.48550/ARXIV.2207.00556. URL https://arxiv.org/abs/2207.00556.

Ebert, F., Finn, C., Lee, A. X., and Levine, S. Self-supervised visual planning with temporal skip connections. *CoRL*, 12:16, 2017.

Gao, Y., Morioka, N., Zhang, Y., and Chen, N. E3 tts: Easy end-to-end diffusion-based text to speech. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8. IEEE, 2023.

Ge, S., Nah, S., Liu, G., Poon, T., Tao, A., Catanzaro, B., Jacobs, D., Huang, J.-B., Liu, M.-Y., and Balaji, Y. Preserve your own correlation: A noise prior for video diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22930–22941, 2023.

Gupta, A., Yu, L., Sohn, K., Gu, X., Hahn, M., Fei-Fei, L., Essa, I., Jiang, L., and Lezama, J. Photorealistic video generation with diffusion models. *arXiv preprint arXiv:2312.06662*, 2023.

Harvey, W., Naderiparizi, S., Masrani, V., Weilbach, C., and Wood, F. Flexible diffusion modeling of long videos. *Advances in Neural Information Processing Systems*, 35: 27953–27965, 2022.

He, Y., Yang, T., Zhang, Y., Shan, Y., and Chen, Q. Latent video diffusion models for high-fidelity video generation with arbitrary lengths. *arXiv preprint arXiv:2211.13221*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*, 2020.

Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J., et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022a.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. *arXiv:2204.03458*, 2022b.

Hoogeboom, E., Heek, J., and Salimans, T. simple diffusion: End-to-end diffusion for high resolution images. *arXiv preprint arXiv:2301.11093*, 2023.

Jabri, A., Fleet, D. J., and Chen, T. Scalable adaptive computation for iterative generation. *CoRR*, abs/2212.11972, 2022.

Kawar, B., Zada, S., Lang, O., Tov, O., Chang, H., Dekel, T., Mosseri, I., and Irani, M. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6007–6017, 2023.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Kingma, D. P. and Gao, R. Understanding diffusion objectives as the elbo with simple data augmentation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Kingma, D. P., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *CoRR*, abs/2107.00630, 2021.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021. ISSN 0027-8424. doi: 10.1073/pnas.2101784118. URL https://www.pnas.org/content/118/21/e2101784118.

Kohl, G., Chen, L.-W., and Thuerey, N. Turbulent flow simulation using autoregressive conditional diffusion models. *arXiv preprint arXiv:2309.01745*, 2023.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. DiffWave: A versatile diffusion model for audio synthesis. In *9th International Conference on Learning Representations, ICLR*, 2021.

Le Moing, G., Ponce, J., and Schmid, C. Ccvs: context-aware controllable video synthesis. *Advances in Neural Information Processing Systems*, 34:14042–14055, 2021.

Li, X., Thickstun, J., Gulrajani, I., Liang, P. S., and Hashimoto, T. B. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Lippe, P., Veeling, B. S., Perdikaris, P., Turner, R. E., and Brandstetter, J. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *arXiv preprint arXiv:2308.05732*, 2023.

Luc, P., Clark, A., Dieleman, S., Casas, D. d. L., Doron, Y., Cassirer, A., and Simonyan, K. Transformation-based adversarial video prediction on large-scale data. *arXiv preprint arXiv:2003.04035*, 2020.

Meng, C., Gao, R., Kingma, D. P., Ermon, S., Ho, J., and Salimans, T. On distillation of guided diffusion models. *CoRR*, abs/2210.03142, 2022.

Nash, C., Carreira, J., Walker, J., Barr, I., Jaegle, A., Malinowski, M., and Battaglia, P. Transframer: Arbitrary frame prediction with generative models. *arXiv preprint arXiv:2203.09494*, 2022.

Price, I., Sanchez-Gonzalez, A., Alet, F., Ewalds, T., El-Kadi, A., Stott, J., Mohamed, S., Battaglia, P., Lam, R., and Willson, M. Gencast: Diffusion-based ensemble forecasting for medium-range weather. *arXiv preprint arXiv:2312.15796*, 2023.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with CLIP latents. *CoRR*, abs/2204.06125, 2022.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 10674–10685. IEEE, 2022.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. Photorealistic text-to-image diffusion models with deep language understanding. *CoRR*, abs/2205.11487, 2022.

Singer, U., Polyak, A., Hayes, T., Yin, X., An, J., Zhang, S., Hu, Q., Yang, H., Ashual, O., Gafni, O., Parikh, D., Gupta, S., and Taigman, Y. Make-a-video: Text-to-video generation without text-video data. *CoRR*, abs/2209.14792, 2022.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2015.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*, 2019.

StabilityAI. Introducing stable video diffusion. In *Stability AI*, Nov 2023. Accessed: 2024-01-25.

Sun, Z., Yang, Y., and Yoo, S. A neural pde solver with temporal stencil modeling. *arXiv preprint arXiv:2302.08105*, 2023.

Tashiro, Y., Song, J., Song, Y., and Ermon, S. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021.

Unterthiner, T., van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., and Gelly, S. Fvd: A new metric for video generation. In *arXiv*, 2019.

Villegas, R., Babaeizadeh, M., Kindermans, P.-J., Moraldo, H., Zhang, H., Saffar, M. T., Castro, S., Kunze, J., and Erhan, D. Phenaki: Variable length video generation from open domain textual description. *arXiv preprint arXiv:2210.02399*, 2022.

Weissenborn, D., Täckström, O., and Uszkoreit, J. Scaling autoregressive video models. *arXiv preprint arXiv:1906.02634*, 2019.

Wu, C., Liang, J., Ji, L., Yang, F., Fang, Y., Jiang, D., and Duan, N. Nüwa: Visual synthesis pre-training for neural visual world creation. In *European conference on computer vision*, pp. 720–736. Springer, 2022.

Wu, T., Fan, Z., Liu, X., Gong, Y., Shen, Y., Jiao, J., Zheng, H.-T., Li, J., Wei, Z., Guo, J., et al. Ar-diffusion: Auto-regressive diffusion model for text generation. *arXiv preprint arXiv:2305.09515*, 2023.

Yan, W., Zhang, Y., Abbeel, P., and Srinivas, A. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

Yan, W., Hafner, D., James, S., and Abbeel, P. Temporally consistent transformers for video generation. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML*, 2023.

Yang, R., Srivastava, P., and Mandt, S. Diffusion probabilistic modeling for video generation. *Entropy*, 25(10):1469, 2023.

Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., Hutchinson, B., Han, W., Parekh, Z., Li, X., Zhang, H., Baldridge, J., and Wu, Y. Scaling autoregressive models for content-rich text-to-image generation. *CoRR*, abs/2206.10789, 2022.

Yu, L., Cheng, Y., Sohn, K., Lezama, J., Zhang, H., Chang, H., Hauptmann, A. G., Yang, M.-H., Hao, Y., Essa, I., et al. Magvit: Masked generative video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10459–10469, 2023a.

Yu, L., Lezama, J., Gundavarapu, N. B., Versari, L., Sohn, K., Minnen, D., Cheng, Y., Gupta, A., Gu, X., Hauptmann, A. G., et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023b.

Yu, S., Sohn, K., Kim, S., and Shin, J. Video probabilistic diffusion models in projected latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18456–18466, 2023c.

Zhang, Z., Liu, R., Aberman, K., and Hanocka, R. Tedi: Temporally-entangled diffusion for long-term motion synthesis. *arXiv preprint arXiv:2307.15042*, 2023.

A. Rolling Diffusion Objective

Standard Diffusion For completeness, we briefly review the diffusion objective. Let $T \in \mathbb{N}$ be a finite number of diffusion steps, and $i \in \{0, \dots, T\}$ be a diffusion time-step. Kingma et al. (2021) show that the discrete-time D_{KL} between $q(\boldsymbol{x}, \boldsymbol{z}_0, \dots, \boldsymbol{z}_T)$ and $p(\boldsymbol{x}, \boldsymbol{z}_0, \dots, \boldsymbol{z}_T)$ can be decomposed as

$$D_{\mathrm{KL}}\left(q(\boldsymbol{x}, \boldsymbol{z}_0, \dots, \boldsymbol{z}_T) || p(\boldsymbol{x}, \boldsymbol{z}_0, \dots, \boldsymbol{z}_T)\right) = \mathbb{E}_{q(\boldsymbol{x}, \boldsymbol{z}_0, \dots, \boldsymbol{z}_T)}\left[\log q(\boldsymbol{x}, \boldsymbol{z}_0, \dots, \boldsymbol{z}_T) - \log p(\boldsymbol{x}, \boldsymbol{z}_0, \dots, \boldsymbol{z}_T)\right] \tag{28}$$

$$= c + \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{z}_T|\boldsymbol{x}) || p(\boldsymbol{z}_T))}_{\text{Prior Loss}} + \underbrace{\mathbb{E}_{q(\boldsymbol{z}_0|\boldsymbol{x})}\left[-\log p(\boldsymbol{x}|\boldsymbol{z}_0)\right]}_{\text{Reconstruction Loss}} + \mathcal{L}_D, \tag{29}$$

where c is a data entropy term. The prior and reconstruction loss terms are typically negligible. $\mathcal{L}_D$ is the *diffusion loss*, which is defined as

$$\mathcal{L}_D = \sum_{i=1}^{T} \mathbb{E}_{q(\boldsymbol{z}_{t_i}|\boldsymbol{x})}\left[D_{\mathrm{KL}}(q(\boldsymbol{z}_{s_i}|\boldsymbol{z}_{t_i}, \boldsymbol{x}) || p(\boldsymbol{z}_{s_i}|\boldsymbol{z}_{t_i}))\right] , \tag{30}$$

Further, when $T \to \infty$ we get *continuous* analog of Equation (30) (Kingma et al., 2021). In practice, instead of using the resulting KL objective, one typically uses a weighted loss, which in some cases still corresponds to an importance-weighted KL (Kingma & Gao, 2023).

$$\mathcal{L}_w := \frac{1}{2}\mathbb{E}_{t\sim U(0,1), \boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}\left[w(\lambda_t) \cdot -\frac{d\lambda_t}{dt}\|\hat{\boldsymbol{\epsilon}}_\theta(\boldsymbol{z}_t; \lambda_t) - \boldsymbol{\epsilon}\|^2\right] , \tag{31}$$

where $\lambda_t := \log \mathrm{SNR}(t)$. Note that in the main paper we directly write the weighting factor expression as $a(t)$ for ease of notation. The weighting function is often changed to improve image quality, for instance by being $-1/\frac{d\lambda_t}{dt}$ so that the objective becomes the simple ϵ-loss.

Rolling Diffusion Objective In rolling diffusion, the signal-to-noise schedule is unchanged but one has to account for the local time reparameterization. Recall that $t_k := t_k(t)$ denotes the local time reparameterization. Using similar derivations as (Kingma et al., 2021), where we can factorize the KL divergence also over frame indices, we get the rolling diffusion loss:

$$\mathcal{L}_\infty = \mathbb{E}_{t\sim U(0,1), \boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}\left[\sum_{k=1}^{K} w(\lambda(t_k)) \cdot -\frac{d\lambda(t_k)}{dt_k}\frac{dt_k}{dt} \cdot \|\boldsymbol{\epsilon}^k - \hat{\boldsymbol{\epsilon}}_\theta^k(\boldsymbol{z}_{\boldsymbol{\epsilon},t}; t)\|^2\right] , \tag{32}$$

where we again can apply a custom weighting factor.

Recall the frame categorization of the main paper, i.e.,

$$\text{clean}(s, t) := \{k \mid s_k = t_k = 0\} , \tag{33}$$

$$\text{noise}(s, t) := \{k \mid s_k = t_k = 1\} , \tag{34}$$

$$\text{win}(s, t) := \{k \mid s_k \in [0, 1), t_k \in (s_k, 1]\} , \tag{35}$$

we can see that $t'_k(t) = 0$ for $k \in \text{clean}(t - dt, t)$ and $k \in \text{noise}(t - dt, t)$ and thus the objective only has non-zero loss over the window:

$$\mathcal{L}_\infty = \mathbb{E}_{t\sim U(0,1), \boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}\left[\sum_{k\in\text{win}(t)} w(\lambda(t_k)) \cdot -\frac{d\lambda(t_k)}{dt_k}\frac{dt_k}{dt} \cdot \|\boldsymbol{\epsilon}^k - \hat{\boldsymbol{\epsilon}}_\theta^k(\boldsymbol{z}_{\boldsymbol{\epsilon},t}; t)\|^2\right] . \tag{36}$$

Since the weighting function w is arbitrary, we can choose it such that the entire prefactor vanishes, resulting in the typical ϵ-loss. However, the variational bound tells us that we should pay no price for any error made on the noise or clean frames. Again, in the main paper we replace all the prefactors with $a(t_k)$ for notational convenience.

B. Algorithms

Algorithm 3 Rolling Diffusion: sampling at the boundary.

Require: $\boldsymbol{x} \in \mathbb{R}^{D \times n_{\text{cln}}}$, W, T, t_w^{init}, p_θ
Sample $\boldsymbol{z}_1 \sim \mathcal{N}(0, \mathbf{I})$, $\boldsymbol{z}_1 \in \mathbb{R}^{D \times (W - n_{\text{cln}})}$
$\boldsymbol{z}_1 \leftarrow \text{concat}(\boldsymbol{x}, \boldsymbol{z}_1)$
for $t = 1, (T-1)/T, \ldots, 1/T$ **do**
  Compute local times $t_w^{\text{init}}(n_{\text{cln}})$, $w = 0, \ldots, W-1$
  Sample $\boldsymbol{z}_{t-1/T} \sim p_\theta(\boldsymbol{z}_{t-1/T} | \boldsymbol{z}_t)$
end for
Return $\boldsymbol{z}_0$, which now has local times $(0/W, 1/W, \ldots, (W-1)/W)$

C. Hyperparameters

In this section we denote the hyperparameters for the different experiments. Throughout the experiments we use U-ViTs which are essentially U-Nets with MLP Blocks instead of convolutional layers when self-attention is used in a block. In the PDE experiments relatively small architectures are used. For BAIR, we used a larger architecture, increasing both the channel count and the number of blocks. For K600 we used even larger architectures, because this dataset turned out to be the most difficult to fit.

Parameter	Value
Blocks	[3 + 3, 3 + 3, 3 + 3, 8]
Channels	[128, 256, 512, 1024]
Block Type	[Conv2D, Conv2D, Transformer (axial), Transformer]
Head Dim	128
Dropout	[0, 0.1, 0.1, 0.1]
Downsample	(1, 2, 2)
Model parametrization	v
Loss	ϵ-loss (x-loss with SNR + 1 weighting)
Number of Steps	100 000 (rollout experiments) / 570 000 (standard experiments)
EMA decay	0.9999
learning rate	1e-4

Table 4. Hyperparameters used for the Kolmogorov flow experiment.

Parameter	Value
Blocks	[4 + 4, 4 + 4, 4 + 4, 8]
Channels	[256, 512, 1024, 2048]
Block Type	[Conv2D, Conv2D, Transformer (axial), Transformer]
Head Dim	128
Dropout	0.1
Downsample	(1, 2, 2)
Model parametrization	v
Loss	ϵ-loss (x-loss with SNR + 1 weighting)
Number of Steps	660 000
EMA decay	0.9999
learning rate	1e-4

Table 5. Hyperparameters used for the BAIR robot pushing experiment.

Parameter	Value
Blocks	[4 + 4, 4 + 4, 5 + 5, 8]
Channels	[256, 512, 2048, 4096]
Block Type	[Conv2D, Conv2D, Transformer (axial), Transformer]
Head Dim	128
Dropout	[0, 0, 0.1, 0.1]
Downsample	(1, 2, 2)
Model parametrization	v
Loss	ϵ-loss (x-loss with SNR + 1 weighting)
Number of Steps	300 000 (rollout experiments) / 570 000 (standard experiments)
EMA decay	0.9999
learning rate	1e-4

Table 6. Hyperparameters used for the Kinetics-600 experiments.

D. Runtime Complexity

Rolling diffusion models do not have an inherent runtime advantage over (batch) autoregressive diffusion models, as both have similar parameter sizes. For fair comparison, one can fix a number of *evaluations-per-frame* for all models. For example, we allow 32 model evaluations per frame. An autoregressive model would sample a sequence frame by frame, each taking 32 evaluations. A batch-autoregressive model samples, e.g., 8 frames jointly, allowing for 256 model evaluations for this subsequence. In rolling diffusion using a window size of 8, the model gets evaluated 4 times before sliding the window. Upon sampling completion, every frame will be sampled using 32 evaluations.

Given this fixed number of model evaluations per frame, we show in this work that using a rolling sampling strategy can benefit time series generation. However, we note that there are some slight inefficiencies during training. Note that in rolling diffusion, some data is always revealed to the model due to the partial noising schedule. This means that if there is a high overlap between the frames, the model can directly copy the global structure from early frames to the later frames, achieving relatively good reconstruction loss. We believe that because of this, the model gets a less strong signal for learning an "image or video prior", as it can heavily rely on the conditioning signal. Further experimentation to alleviate this issue is required, perhaps by adjusting the rolling noise schedule or fine-tuning a standard diffusion model using the rolling diffusion loss.

E. Simulation Details

The parameters used to generate the Kolmogorov Flow data are shown in Table 7. It is important to note that to introduce uncertainty into an otherwise deterministic system, we vary the viscosity and density parameters, which must then be inferred from the data. This would also make a standard solver very difficult to use in such a setting. Additionally, due to the chaotic nature of the system, it is not deterministically predictable up to arbitrary precision. We use the 'simple turbulence forcing' option, which combines a driving force with a damping term such that we simulate turbulence in two dimensions.

Table 7. Parameters for Kolmogorov Flow Simulation using JaxCFD

Parameter	Value
Size	256
Viscosity	Uniform random in $[5.0 \times 10^{-4}, 5.0 \times 10^{-3}]$
Density	Uniform random in $[2^{-1}, 2^{1}]$
Maximum Velocity	7.0
CFL Safety Factor	0.5
Max Δt	0.05
Outer Steps	6000
Grid	256×256, domain $[0, 2\pi] \times [0, 2\pi]$
Initial Velocity	Filtered velocity field, 3 iterations
Forcing	Simple turbulence, magnitude=2.5, linear coefficient=-0.1, wavenumber=4
Total Simulations	200,000

F. Additional Results

F.1. Kolmogorov Flow

We show in Table 8 the MSE and FSE errors at various time-steps. Note that the MSE model is always optimal in terms of MSE loss, which is as expected. However, in terms of matching the frequency distribution, as measured by FSD, standard diffusion, and in particular rolling diffusion are optimal. Averaging ensembles of the rolling diffusion model does not improve the FSD score, but does improve the MSE score.

	FSD/MSE @						
Method	1	2	4	8	12	24	48
MSE (2-1)	304.7 / **14.64**	687.1 / **137.15**	1007 / **407.0**	1649 / **407.0**	2230 / **407.0**	5453 / **407.0**	7504 / **407.0**
MSE (2-2)	531.3 / 20.1	7205 / 148.3	5596 / 407.0	7277 / 407.0	8996 / 406.1	$1 \cdot 10^4$ / 407.0	$1 \cdot 10^4$ / 407.0
MSE (2-4)	304.7 / 21.7	6684 / 148.9	$6 \cdot 10^4$ / 378.9	$3 \cdot 10^4$ / 407.0	$2 \cdot 10^4$ / 407.0	$2 \cdot 10^4$ / 407.0	$2 \cdot 10^4$ / 407.0
Standard Diffusion (2-1)	39.59 / 20.76	57.73 / 192.6	142.0 / 710.0	297.7 / 794.6	399.4 / 773.1	442.5 / 758.3	763.6 / 732.5
Standard Diffusion (2-2)	59.15 / 47.88	86.12 / 334.9	112.6 / 766.3	241.8 / 781.5	314.7 / 755.5	403.3 / 725.0	726.3 / 695.4
Standard Diffusion (2-4)	86.19 / 49.93	141.6 / 353.6	246.6 / 753.2	397.8 / 758.0	555.6 / 726.9	1094.0 / 701.1	2401.0 / 666.3
Standard Diffusion (2-8)	87.0 / 54.0	137.7 / 399.2	288.3 / 770.1	338.7 / 725.3	355.5 / 713.6	530.6 / 705.5	1159 / 748.3
Rolling Diffusion (init noise) (2-2)	63.21 / 45.62	92.58 / 300.3	144.8 / 748.5	239.1 / 799.2	370.5 / 787.3	529.7 / 767.8	1568.7 / 735.3
Rolling Diffusion (init noise) (2-4)	29.59 / 39.72	**47.44** / 287.8	**43.39** / 738.4	**61.93** / 769.0	214.32 / 735.7	648.53 / 699.1	1238.4 / 670.0
Rolling Diffusion (init noise) (2-8)	**27.68** / 41.22	52.41 / 316.9	53.47 / 768.0	98.29 / 777.2	**187.03** / 748.8	**344.89** / 737.6	**417.59** / 719.4
Rolling 10-Ensemble (2-8)	481.6 / 36.4	8590 / 214.7	$4 \cdot 10^4$ / 429	$5 \cdot 10^4$ / 440	$5 \cdot 10^4$ / 429	$5 \cdot 10^4$ / 428	$5 \cdot 10^4$ / 429

Table 8. Kolmogorov Flow Results

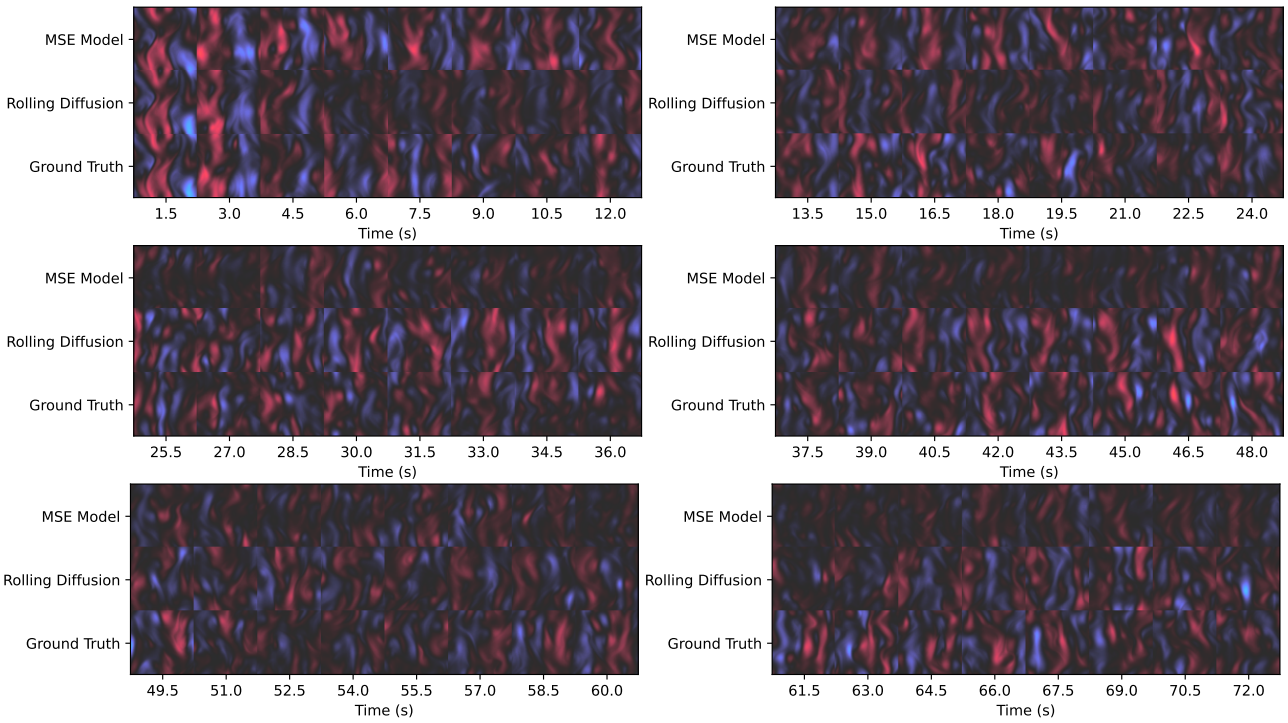

Figure 7. Example rollout of a Kolmogorov flow sample. We depict the vertical velocity field. Note how the MSE model's intensity decreases as we move further from the conditioning frames.

F.2. Kinetics-600

In Section 5.3 we compare against TECO (Yan et al., 2023) on the Kinetics-600 dataset. TECO uses 20 conditioning frames and generates 80 new frames at a resolution of 128 by 128 using 256 FVD samples.

Method	FVD (↓)
TECO	799
Rolling Diffusion (Ours)	**685**

Table 9. In TECO (Yan et al., 2023), samples are generated conditioning on 20 frames and generating 80 new frames at a resolution of 128 by 128 using 256 FVD samples. Different from other parts of the paper because of the low sample count, FVD is measured by *matching* the conditioning for the reference samples.

Furthermore, the following plot shows MSE deviations from ground-truth on Kinetics-600 data.

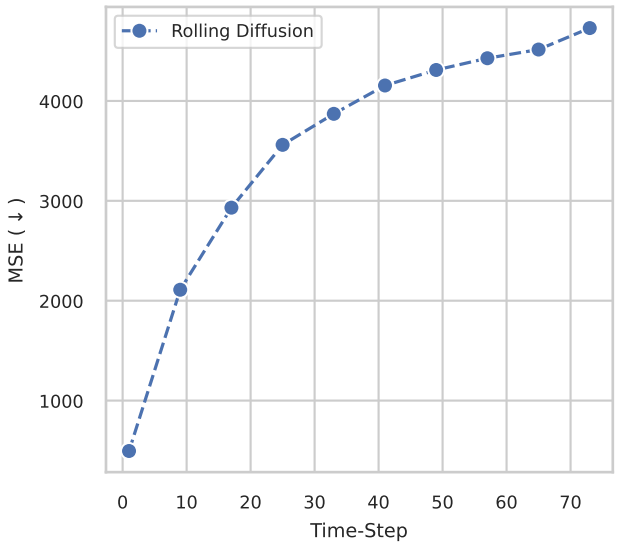

Figure 8. This figure shows MSE as a function of frame distance from the starting point on Kinetics-600 on the 20-12 setting with rollouts until frame 80 on resolution 128 $\times$ 128. As the generated frames are further from the initial 20 conditioning frames, the error between the original and the generated samples increases.

G. Rescaled Noise Schedule

For our Kinetics-600 experiments, we used a different noise schedule which can sample from complete noise towards a "rolling state", i.e., at diffusion times $(\frac{1}{W}, \frac{2}{W}, \dots \frac{W}{W})$. From there, we can roll out generation using, e.g., the linear rolling sampling schedule t_w^{lin}. The reason is that we hypothesized that the noise schedule t_w^{init} uses a clip operation, which means that will be sampling in $\text{clean}(s, t)$, which is redundant as outlined in the main paper and Appendix A.

$$t_w^{\text{init,resc}}(t) := \frac{w}{W} + t \cdot (1 - \frac{w}{W}) \tag{37}$$

Where we clearly have at $t = \frac{1}{W}$ that the local times are $\left(\frac{1}{W}, \frac{2}{W}, \dots, \frac{W}{W}\right)$, which is what we need. Note that this schedule is not directly proportional to t.

H. Hyperparameter Search for β

Task	Training regiment	Cond-Gen	FVD
stride=8	Rolling init rescaled $\beta = 0.1$	(5-11)	**39.8**
steps=24	Rolling init rescaled $\beta = 0.2$	(5-11)	44.1
	Rolling init rescaled $\beta = 0.5$	(5-11)	46.3
	Rolling init rescaled $\beta = 0.7$	(5-11)	43.0
	Rolling init rescaled $\beta = 0.8$	(5-11)	46.0
	Rolling init rescaled $\beta = 0.9$	(5-11)	47.5
	Rolling init clip $\beta = 0.0$	(5-11)	60.3
	Rolling init clip $\beta = 0.2$	(5-11)	52.0
	Rolling init clip $\beta = 0.5$	(5-11)	49.0
	Rolling init clip $\beta = 0.7$	(5-11)	48.9
stride=1	Rolling init rescaled $\beta = 0.7$	(5-11)	216
steps=64	Rolling init rescaled $\beta = 0.8$	(5-11)	227
	Rolling init rescaled $\beta = 0.9$	(5-11)	**211**
	Standard Diffusion	(15-1)	1460
	Standard Diffusion	(5-1)	1369
	Standard Diffusion	(8-8)	157.1
	Standard Diffusion	(5-8)	142.4

Table 10. Oversampling t_w^{lin} vs. t_w^{init} noise on Kinetics-600 with 8192 FVD samples. We allow 100 denoising steps per 11 frames.