Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion

Boyuan Chen
MIT CSAIL
boyuanc@mit.edu

Diego Marti Monso*
Technical University of Munich
diego.marti@tum.de

Yilun Du
MIT CSAIL
yilundu@mit.edu

Max Simchowitz
MIT CSAIL
msimchow@mit.edu

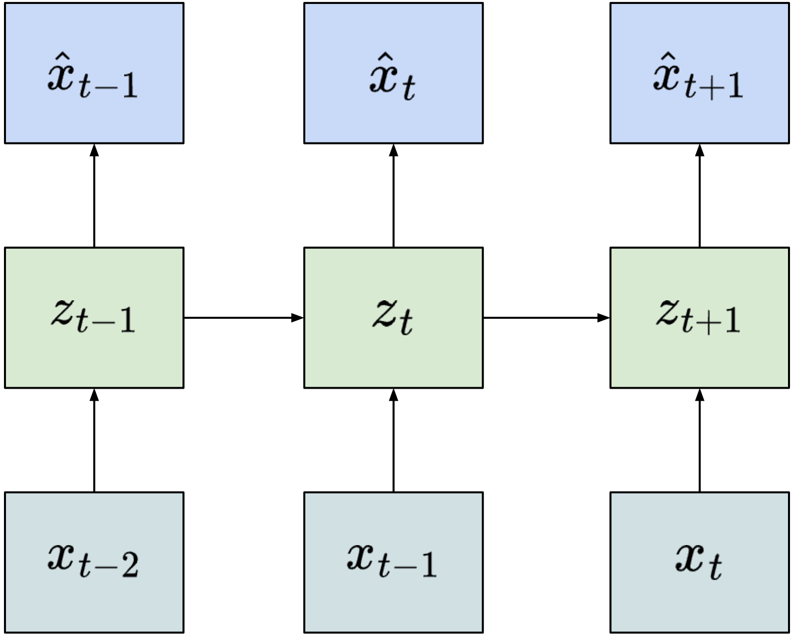
Russ Tedrake
MIT CSAIL
russt@mit.edu

Vincent Sitzmann
MIT CSAIL
sitzmann@mit.edu

NeurIPS 2024

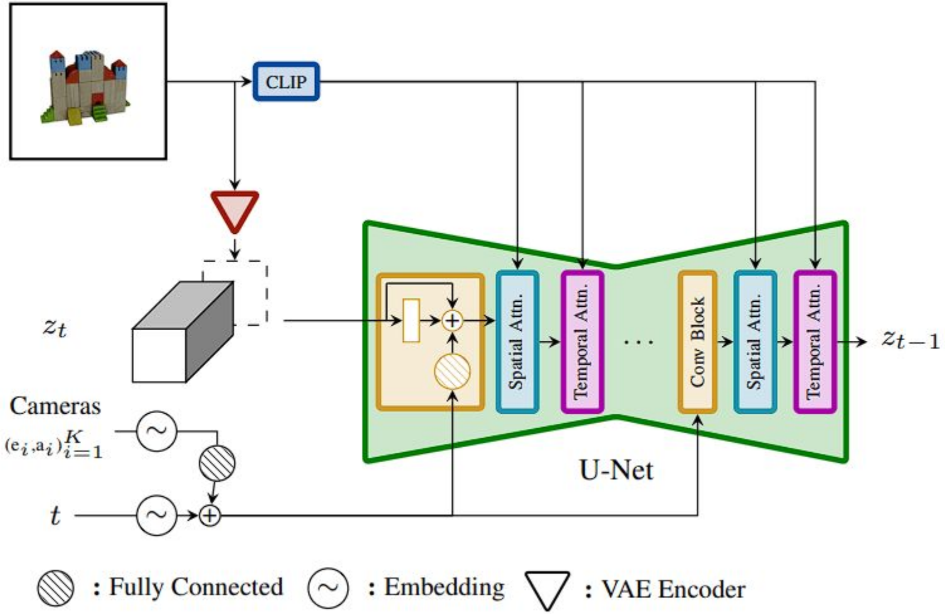
Sequence Models

Autoregressive (Next-Token) Sequence Models



Teacher Forcing → ❌ unstable long rollouts

Full Sequence Diffusion Models



Full-Sequence Diffusion → ❌ non-causal

Diffusion Forcing = Teacher Forcing + Diffusion

Unifying time-axis masking (teacher forcing) and noise-axis masking (diffusion)

3.1 Noising as partial masking

Recall that *masking* is the practice of occluding a subset of data, such as patches of an image [27] or timesteps in a sequence [15, 49], and training a model to recover unmasked portions. Without loss of generality, we can view any collection of tokens, sequential or not, as an ordered set indexed by t. Training next-token prediction with teacher forcing can then be interpreted as masking each token $\mathbf{x}_t$ at time t and making predictions from the past $\mathbf{x}_{1:t-1}$. Restricted to sequences, we refer to all these practices as *masking along the time axis*. We can also view full-sequence forward diffusion, i.e., gradually adding noise to the data $\mathbf{x}^0_{1:T} \equiv \mathbf{x}_{1:T}$, as a form of *partial masking*, which we refer to as *masking along the noise axis*. Indeed, after K steps of noising, $\mathbf{x}^K_{1:T}$ is (approximately) pure white noise without information about the original data.

We establish a unified view along both axes of masking (see Fig. 2). We denote $\mathbf{x}_{1:T}$ for a sequence of tokens, where the subscript indicates the time axis. As above, $\mathbf{x}_t^{k_t}$ denotes $\mathbf{x}_t$ at noise level k_t under the forward diffusion process (2.1); $\mathbf{x}_t^0 = \mathbf{x}$ is the unnoised token, and $\mathbf{x}_t^K$ is white noise $\mathcal{N}(0, \mathbf{I})$. Thus, $(\mathbf{x}_t^{k_t})_{1 \le t \le T}$ denotes a sequence of noisy observations where each token has a *different* noise level k_t, which can be seen as the degree of *partial masking* applied to each token through noising.

Diffusion Forcing = Teacher Forcing + Diffusion

Unifying time-axis masking (teacher forcing) and noise-axis masking (diffusion)

3.2 Diffusion Forcing: different noise levels for different tokens

Diffusion Forcing (DF) is a framework for training and sampling arbitrary sequence lengths of noisy tokens $(\mathbf{x}_t^{k_t})_{1\leq t\leq T}$, where critically, *the noise level k_t of each token can vary by time step*. In this paper, we focus on time series data, and thus instantiate Diffusion Forcing with causal architectures (where $\mathbf{x}_t^{k_t}$ depends only on past noisy tokens), which we call *Causal Diffusion Forcing* (CDF). For simplicity, we focus on a minimal implementation with a vanilla Recurrent Neural Network (RNN) [11]. Potential transformer implementation of Diffusion Forcing is also possible but we defer its discussion to Appendix B.1.

The RNN with weights θ maintains latents $\mathbf{z}_t$ capturing the influence of past tokens, and these evolve via dynamics $\mathbf{z}_t \sim p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_t^{k_t}, k_t)$ with a recurrent layer. When an incoming noisy observation $\mathbf{x}_t^{k_t}$ is made, the hidden state is updated in a Markovian fashion $\mathbf{z}_t \sim p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_t^{k_t}, k_t)$[2]. When $k_t = 0$, this is the posterior update in Bayes filtering; whereas when $k_t = K$ (and $\mathbf{x}_t^K$ is pure noise and thus uninformative), this is equivalent to modeling the "prior distribution" $p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ in Bayes filtering. Given latent $\mathbf{z}_t$, an observation model $p_\theta(\mathbf{x}_t^0|\mathbf{z}_t)$ predicts $\mathbf{x}_t$.

Diffusion Forcing = Teacher Forcing + Diffusion

Unifying time-axis masking (teacher forcing) and noise-axis masking (diffusion)

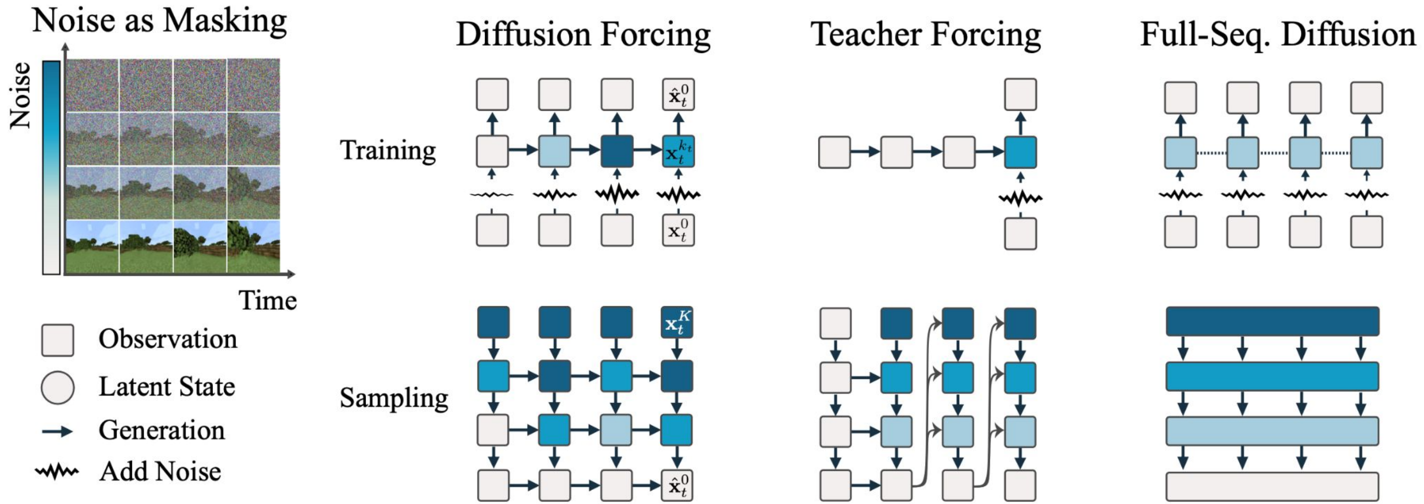


Figure 2: **Method Overview.** Diffusion Forcing trains causal sequence neural networks (such as an RNN or a masked transformer) to denoise flexible-length sequences where each frame of the sequence can have a *different* noise level. In contrast, next-token prediction models, common in language modeling, are trained to predict a single next token from a *ground-truth* sequence (teacher forcing [65]), and full-sequence diffusion, common in video generation, train non-causal architectures to denoise all frames in a sequence at once with the *same* noise level. Diffusion Forcing thus *interleaves* the time axis of the sequence and the noise axis of diffusion, unifying strengths of both alternatives and enabling completely new capabilities (see Secs. 3.2,3.4).

Training and Sampling

Algorithm 1 Diffusion Forcing Training

loop
  Sample tajectory of observations $(\mathbf{x}_1, ..., \mathbf{x}_T)$.
  for $t = 1, ..., T$ **do**
    Sample independent noise level $k_t \in \{0, 1, ..., K\}$
    $\mathbf{x}_t^{k_t} = \text{ForwardDiffuse}(\mathbf{x}_t, k_t)$
    Define $\epsilon_t = \frac{\mathbf{x}_t^{k_t} - \sqrt{\bar{\alpha}_{k_t}}\mathbf{x}_t}{\sqrt{1-\bar{\alpha}_{k_t}}}$
    Update $\mathbf{z}_t \sim p_\theta(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x}_t^{k_t}, k_t)$.
    Set $\hat{\epsilon}_t = \epsilon_\theta(\mathbf{z}_{t-1}, \mathbf{x}_t^{k_t}, k_t)$
  end for
  $L = \text{MSELoss}([\hat{\epsilon}_1, ..., \hat{\epsilon}_n], [\epsilon_1, ..., \epsilon_n])$
  Backprop with L and update θ
end loop

Algorithm 2 DF Sampling with Guidance

Input: Model θ, scheduling matrix $\mathcal{K}$, initial latent $\mathbf{z}_0$, guidance cost $c(\cdot)$.
Initialize $\mathbf{x}_1, \ldots, \mathbf{x}_T \sim \mathcal{N}(0, \sigma_K^2 I)$.
for row $m = M - 1, ..., 0$ **do**
  for $t = 1, \ldots, T$ **do**
    $\mathbf{z}_t^{\text{new}} \sim p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t, \mathcal{K}_{m+1,t})$.
    $k \leftarrow \mathcal{K}_{m,t}$, $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I})$.
    $\mathbf{x}_t^{\text{new}} \leftarrow \frac{1}{\sqrt{\alpha_k}}(\mathbf{x}_t - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta(\mathbf{z}_t^{\text{new}}, \mathbf{x}_t, k)) + \sigma_k \mathbf{w}$
    Update $\mathbf{z}_t \leftarrow \mathbf{z}_t^{\text{new}}$.
  end for
  $\mathbf{x}_{1:H} \leftarrow \text{AddGuidance}(\mathbf{x}_{1:H}^{\text{new}}, \nabla_{\mathbf{x}} \log c(\mathbf{x}_{1:H}^{\text{new}}))$
end for
Return $\mathbf{x}_{1:T}$.

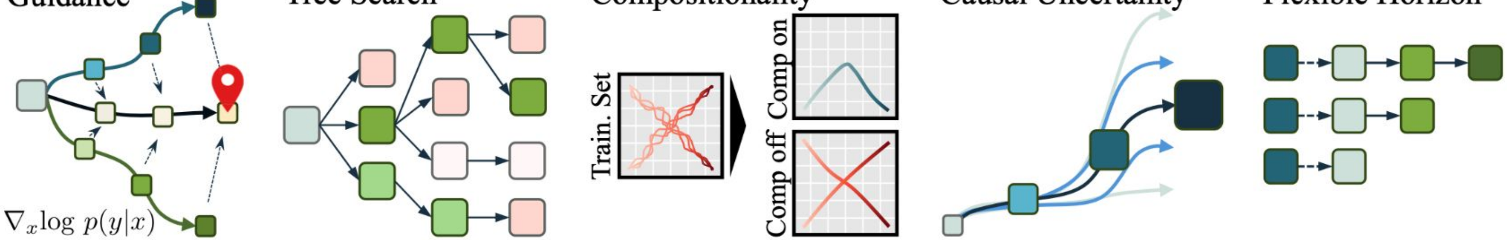
Key Ingredient – 2D Noise Schedule Grid

Sampling. Diffusion Forcing sampling is depicted in Algorithm 2 and is defined by prescribing a noise schedule on a 2D $M \times T$ grid $\mathcal{K} \in [K]^{M \times T}$; columns correspond to time step t and rows indexed by m determine noise-level. $\mathcal{K}_{m,t}$ represents the desired noise level of the time-step t token for row m. To generate a whole sequence of length T, initialize the tokens $\mathbf{x}_{1:T}$ to be white noise, corresponding to noise level $k = K$. We iterate down the grid row-by-row, denoising left-to-right across columns to the noise levels prescribed by $\mathcal{K}$. By the last row $m = 0$, the tokens are clean, i.e. their noise level is $\mathcal{K}_{0,t} \equiv 0$. Appendix D.5 discusses corner cases of this scheme; the hyperparameters $(\alpha_k, \bar{\alpha}_k, \sigma_k)$ are set to their standard values [30]. The matrix $\mathcal{K}$ specifies how fast each token gets denoised at every step of sequence diffusion. Since Diffusion Forcing is trained to denoise tokens of all sequences of noise levels, $\mathcal{K}$ can be designed to flexibly achieve different behaviors without re-training the model.

$$\mathcal{K}^{\text{pyramid}} = \begin{bmatrix} K & K & K & \dots & K \\ K-1 & K & K & \dots & K \\ K-2 & K-1 & K & \dots & K \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & 3 & \dots & H \\ 0 & 1 & 2 & \dots & H-1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

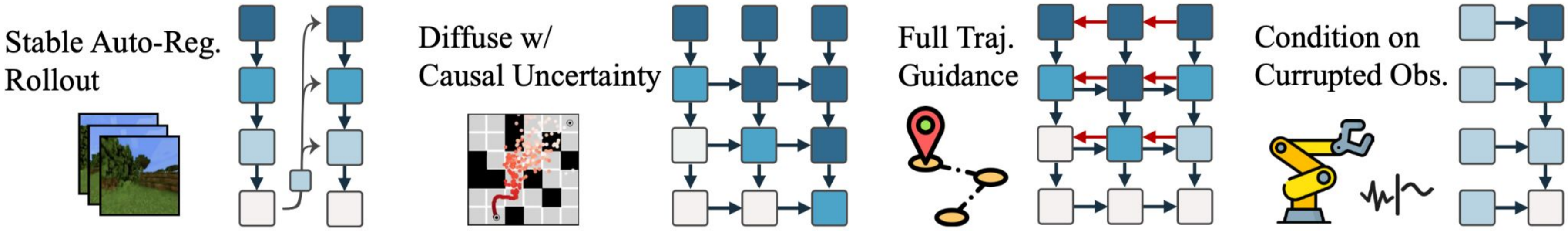
Properties of Diffusion Forcing

Standard diffusion: the same noise for all tokens.
Diffusion Forcing: each token t gets its own noise k□, independent and variable. This enables partial denoising across time and uncertainty axes simultaneously.



	Guidance	Tree Search	Compositionality	Causal Uncertainty	Flexible Horizon
Teacher Forcing	✗	✓	✓	✓	✓
Full-Seq. Diffusion	✓	✗	✗	✗	✗
Diffusion Forcing	✓	✓	✓	✓	✓

Properties of Diffusion Forcing



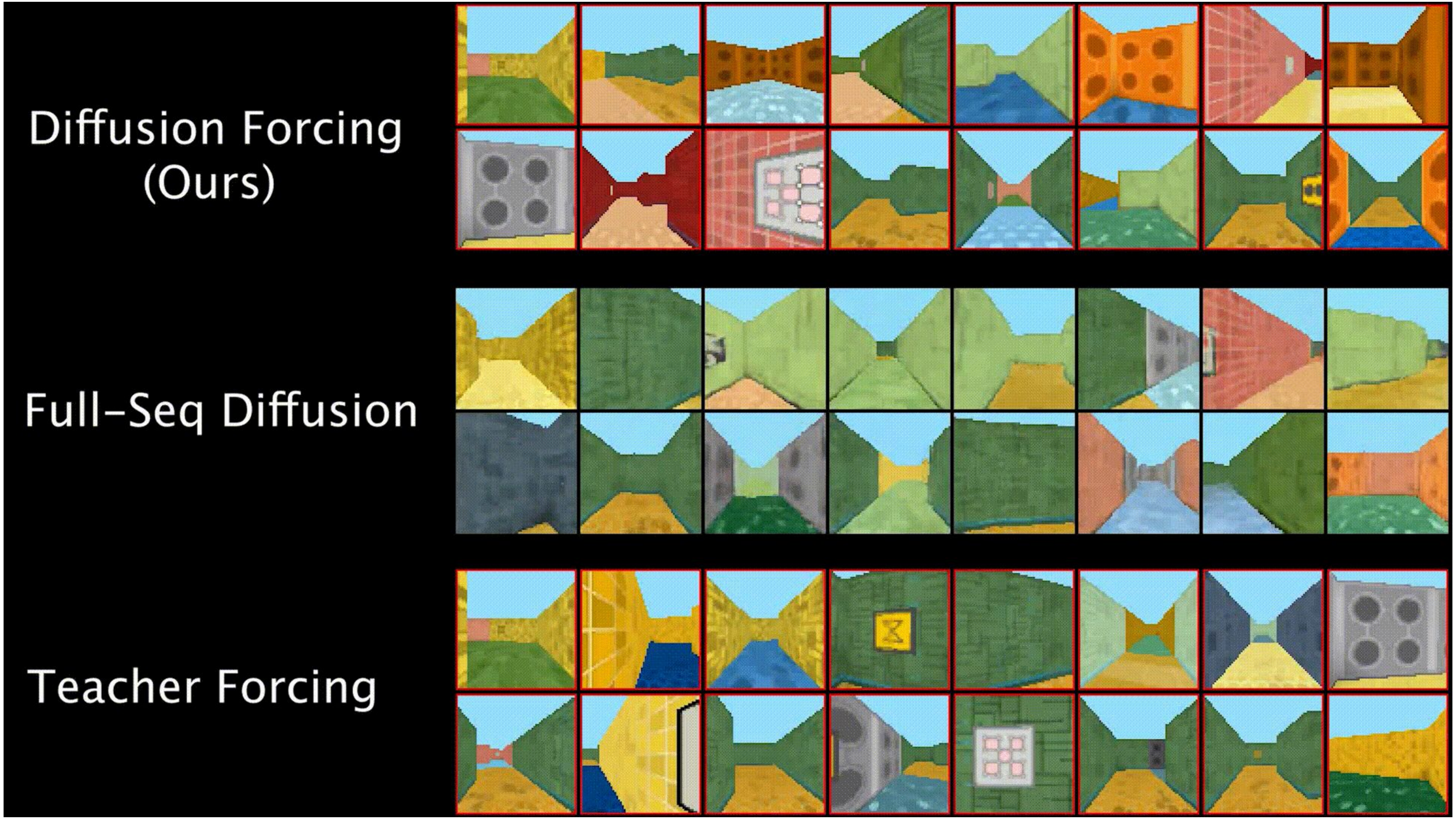
The capabilities offered by Diffusion Forcing motivate our novel framework for sequential decision making (SDM), with key applications to robotics and autonomous agents. Consider a Markov Decision Process defined by an environment with dynamics $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, observation $p(\mathbf{o}_t|\mathbf{s}_t)$ and reward $p(\mathbf{r}_t|\mathbf{s}_t, \mathbf{a}_t)$. The goal is to train a policy $\pi(\mathbf{a}_t|\mathbf{o}_{1:t})$ such that the expected cumulative reward of a trajectory $\mathbb{E}[\sum_{t=1}^{T} \mathbf{r}_t]$ is maximized. We assign tokens $\mathbf{x}_t = [\mathbf{a}_t, \mathbf{r}_t, \mathbf{o}_{t+1}]$. A trajectory is a sequence $\mathbf{x}_{1:T}$, possibly of variable length; training is conducted as in Algorithm 1. At each step t of execution, past (noise-free) tokens $\mathbf{x}_{1:t-1}$ are summarized by a latent $\mathbf{z}_{t-1}$. Conditioned on this latent, we sample, via Algorithm 2, a plan $\hat{\mathbf{x}}_{t:t+H}$, with $\hat{\mathbf{x}}_t = [\hat{\mathbf{a}}_t, \hat{\mathbf{r}}_t, \hat{\mathbf{o}}_{t+1}]^\top$ containing predicted actions, rewards and observations. H is a look-ahead window, analogous to future predictions in model predictive control [20]. After taking planned action $\hat{\mathbf{a}}_t$, the environment produces a reward $\mathbf{r}_t$ and next observation $\mathbf{o}_{t+1}$, yielding next token $\mathbf{x}_t = [\hat{\mathbf{a}}_t, \mathbf{r}_t, \mathbf{o}_{t+1}]^\top$. The latent is updated according to the posterior $p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_t, 0)$. Our framework enables functionality as both *policy* and *planner*:

Video Generation

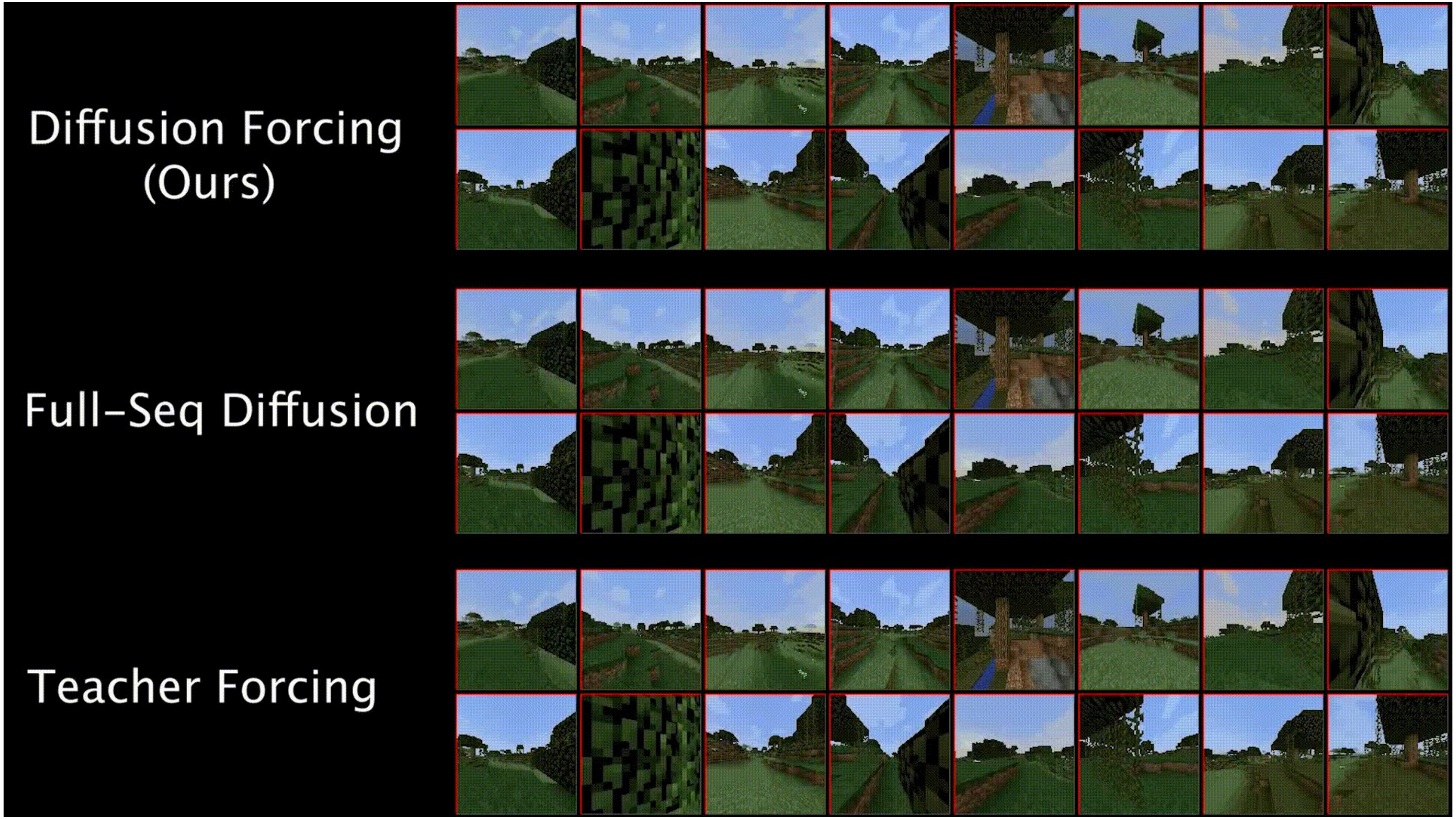



Video Generation

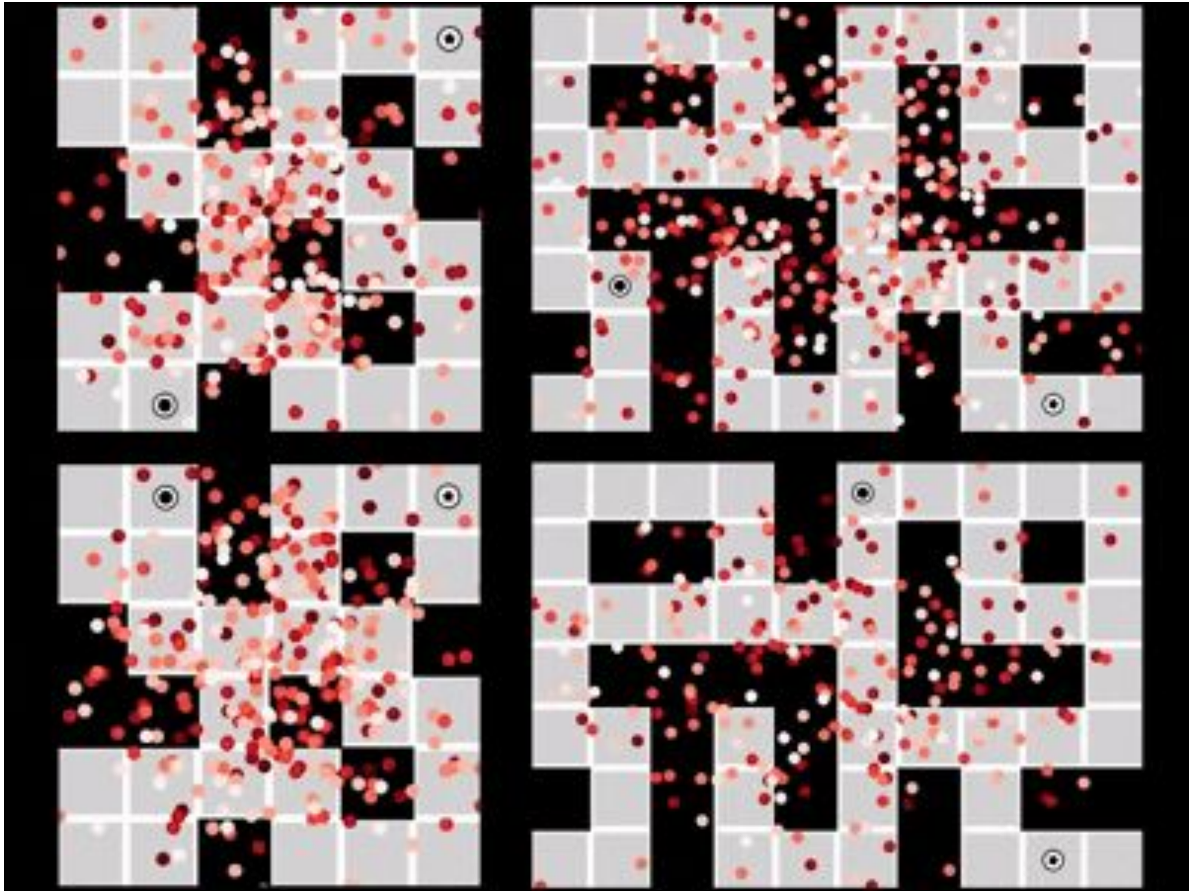
World Model Imagination



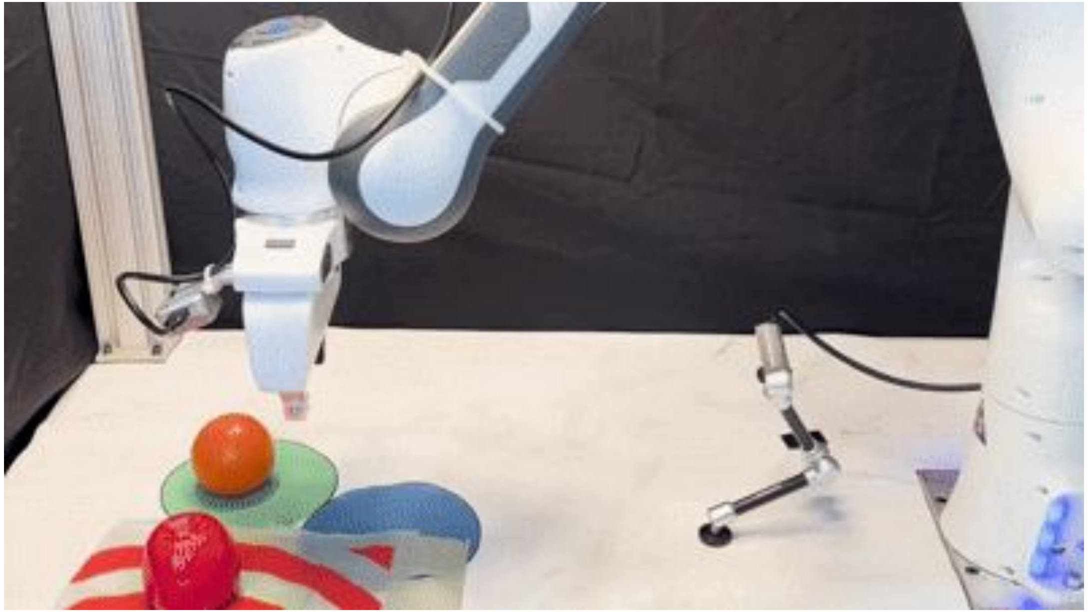
World Model Imagination



Trajectory Planning



Imitation Learning



Takeaway

Unified Framework for Causal Diffusion. Diffusion Forcing bridges the gap between *next-token prediction* and *full-sequence diffusion*.

- Each token has an **independent noise level**, enabling the model to denoise causally, *i.e.,* clean earlier tokens while keeping future ones uncertain.
- This allows DF to combine **variable-length, causal generation** (like autoregressive models) with **guided, uncertainty-aware sampling** (like diffusion models).

New Sampling Capabilities. DF introduces a **2D sampling grid** over diffusion steps (rows) and time steps (columns)

- **Stable autoregressive rollouts** beyond the training horizon by softly diffusing past tokens instead of treating them as perfect ground truth.
- **Causal uncertainty encoding:** near-future = low noise (certain), far-future = high noise (uncertain), yielding *long-horizon guidance* without breaking causality.

Robustness and Multi-Domain Generalization

- DF can handle **noisy or missing observations**, prompting the model to rely on its learned prior.
- Demonstrated success in **robotic imitation**, **planning**, and **time-series forecasting**, outperforming both next-token and full-sequence diffusion baselines.

Drawbacks – The “Double Long-Chain” Problem

While DF’s **per-token denoising schedule** is flexible, its **sampling process can become a double long-chain**.

During sampling, DF performs two nested loops:

1. **Diffusion chain** (rows): iterative denoising over diffusion steps.
2. **Temporal chain** (columns): sequential causal rollout over time steps.

Hence, sampling runs over a *double chain* (diffusion x temporal).

Consequences

- **Computation grows linearly in both** time steps and denoising steps. For long-horizon video or planning tasks, this becomes prohibitively expensive.
- **Gradient feedback latency:** Because later tokens are denoised much later in the schedule, the **guidance gradient** must traverse two long chains (temporal + diffusion), which can delay convergence or cause vanishing guidance signals for early tokens.
- **Error accumulation:** As tokens are re-visited multiple times in the denoising schedule, slight prediction noise can re-amplify if the noise decay rate or schedule matrix is not well-calibrated, especially in high-dimensional continuous signals.
- **Optimization complexity:** The causal dependencies and non-uniform noise levels complicate gradient propagation, making DF harder to scale to high-resolution or transformer-based architectures (acknowledged explicitly in the paper’s limitations)