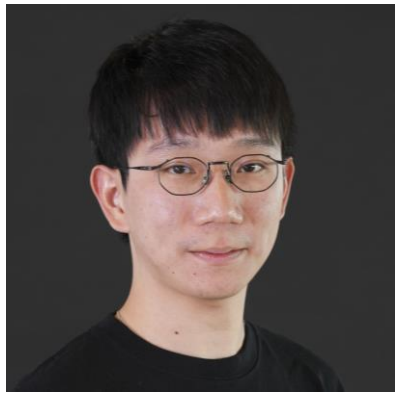
FROM WHITE NOISE TO SYMPHONY🎼: DIFFUSION MODELS FOR MUSIC AND SOUND

@ ISMIR 2024

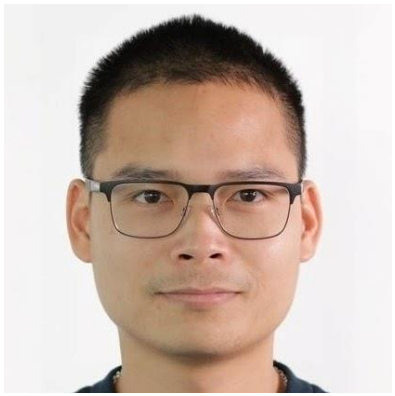
Sony AI

Stanford University

MEET OUR TEAM



Chieh-Hsin (Jesse) Lai



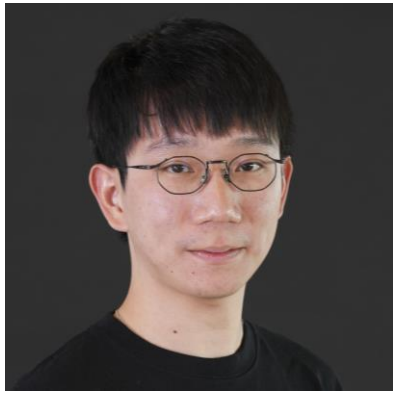
Bac Nguyen

Koichi Saito

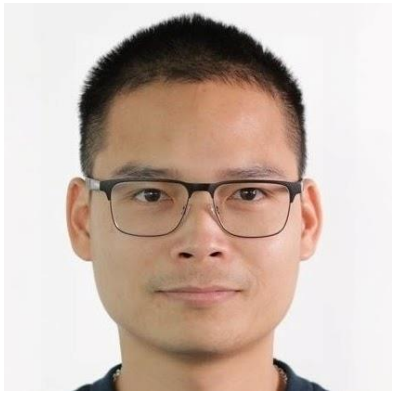
Yuki Mitsufuji

Stefano Ermon

MEET OUR SPEAKERS



Chieh-Hsin (Jesse) Lai



Bac Nguyen

Koichi Saito

Yuki Mitsufuji

AGENDA

Opening (9:00-9:10)

Session 1. (9:10-10:00) Foundation of Score-Based Diffusion Model

Session 2. (10:00-10:40) Practical Implementation and Application of Diffusion Model

Break (10:40-11:10)

Session 3. (11:10-12:00) Demo on Controllable Music Generation

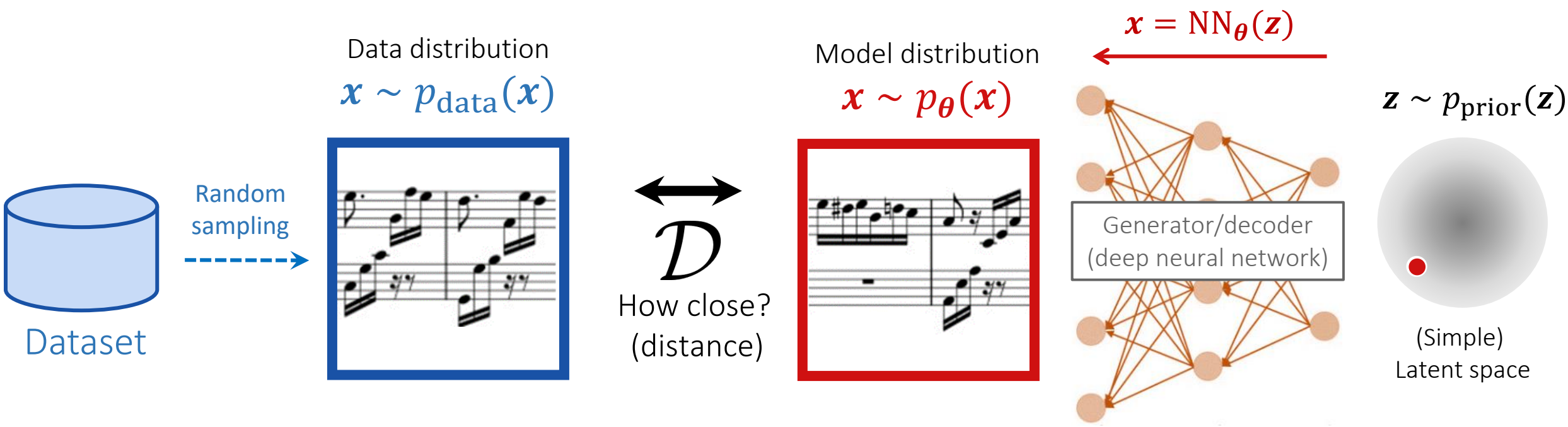
Session 4. (12:00-12:30) Future Trend

Disclaimer

For illustration, we occasionally use images.
But you can think of them as music data!

Deep Generative Model

Goal: use NN to learn target distribution explicitly/implicitly



$$\min_\theta \mathcal{D}(p_\theta(x), p_{\text{data}}(x))$$

DGM in Music Generation

Autoregressive

- ✓ Good for sequential data

Diffusion Model

- ✓ More efficient generation
- ✓ Better controllability

DGM in Music Generation

Autoregressive

Diffusion Model

User specified Text prompt:
A grand orchestral arrangement with thunderous percussion, epic brass fanfares, and soaring strings, creating a cinematic atmosphere fit for a heroic battle.

[Google Deepmind '16] WaveNet

[Meta '23] MusicGen

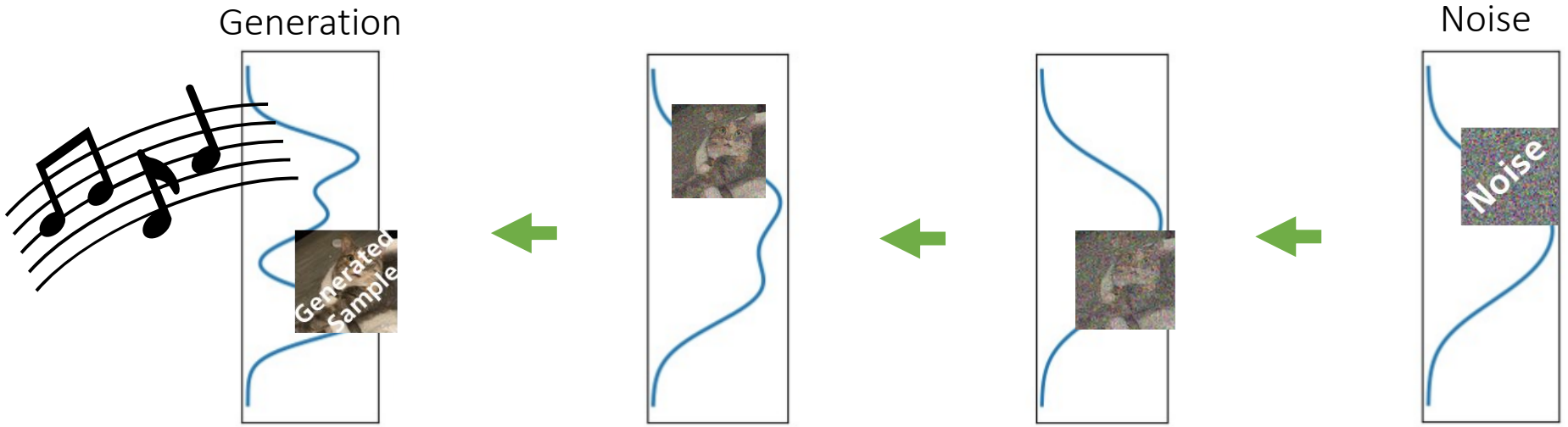
Stable Audio

Suno

Riffusion

Regime of Continuous/Discrete Diffusion Model

Continuous Data Diffusion Model:

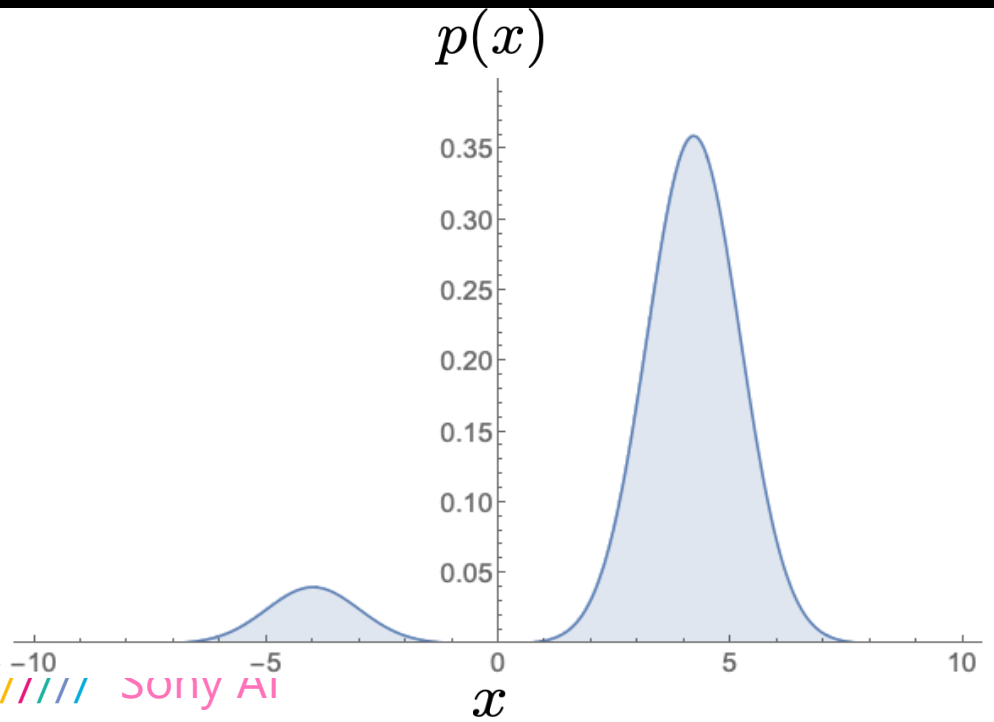


Discrete Data Diffusion Model:

DEVELOPMENT OF SCORE-BASED DIFFUSION MODEL

Using “Score” (Slope) for Generation

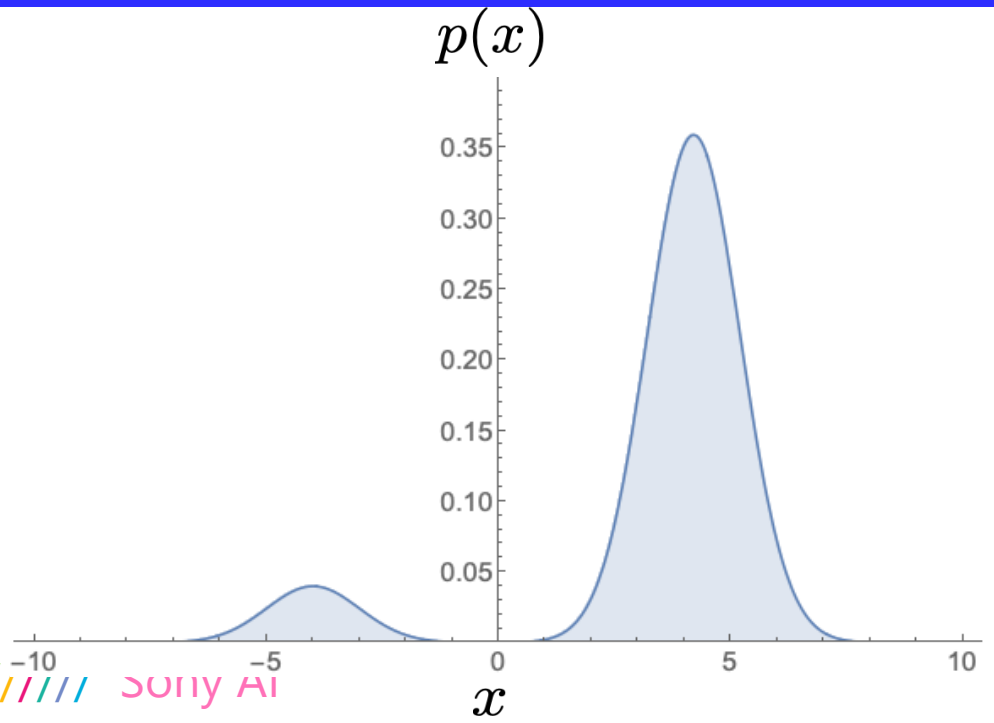
$$p_{\text{data}}(\boldsymbol{x})$$

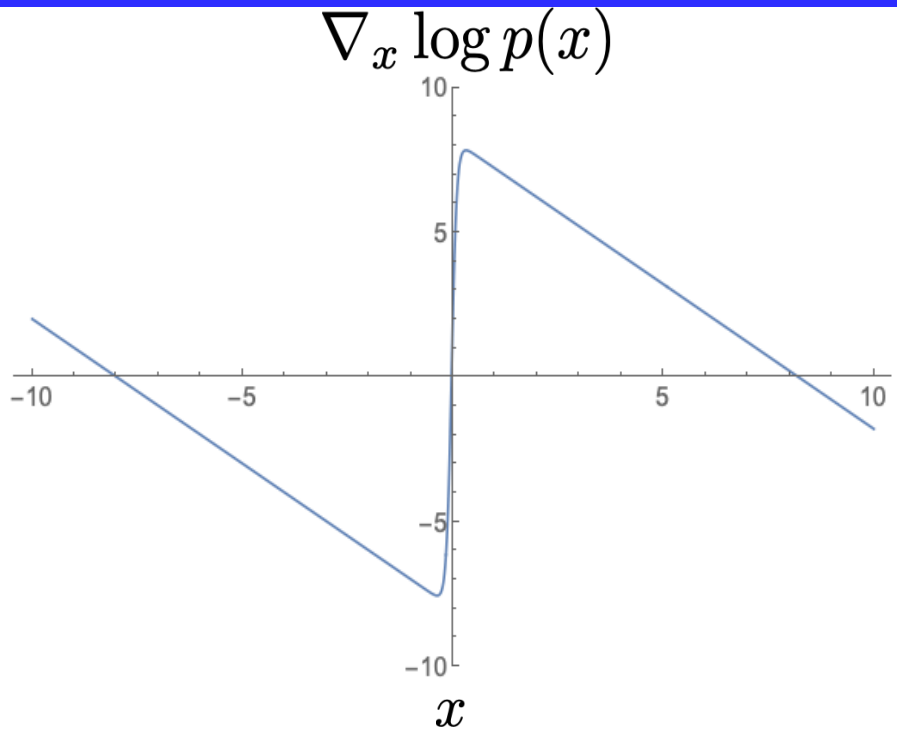


Using “Score” (Slope) for Generation

Moving toward more likely direction…

$$\nabla_x \log p_{\text{data}}(x)$$





Using “Score” (Slope) for Generation

Moving toward more likely direction…

$$\nabla_x \log p_{\text{data}}(x)$$

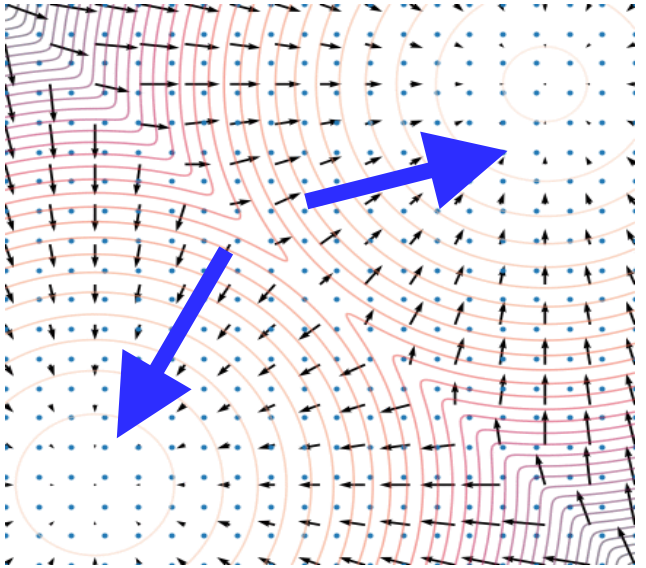
Phase I. Score Matching
Phase II. Denoising Score Matching
Phase III. Noised Conditioned Denoising Score Matching

Phase IV. Score-Based Continuous Time Diffusion Model

Phase I (2005). Score Matching

Score function:

$$\nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})$$

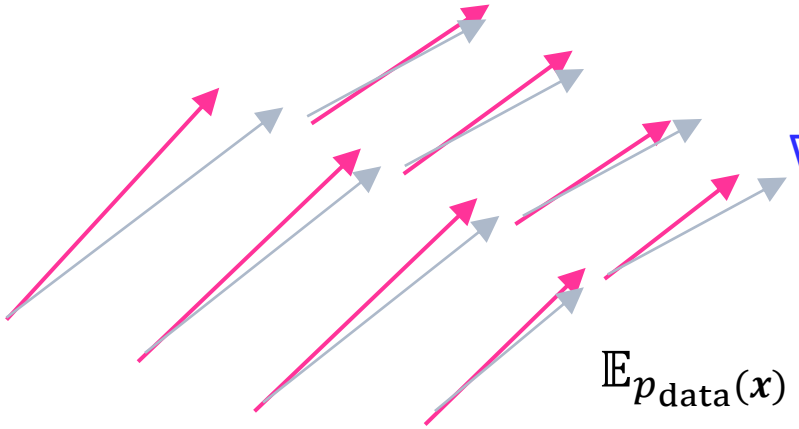


[Generation] Langevin sampling:

$$\boldsymbol{x}_{t+1} \leftarrow \boldsymbol{x}_t + \eta \nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x}) + \sqrt{2}\,\boldsymbol{\epsilon}_t, \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

$$\approx \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})$$

[Training] Score Matching:



$\nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})$

$$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\frac{1}{2}\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})\|_2^2\right] = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\frac{1}{2}\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})\|_2^2 + \text{tr}(\nabla_{\boldsymbol{x}} \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}))\right] + C$$

[Hyvärinen '05]

Phase II (2011). Denoising Score Matching

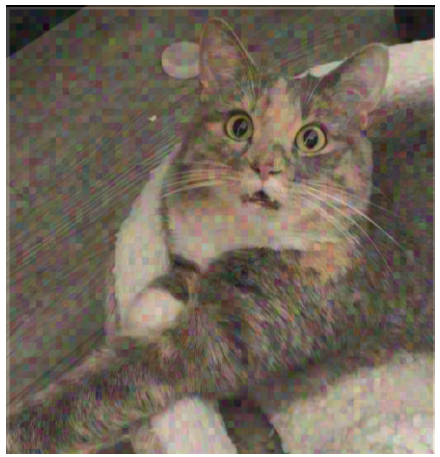
[Training] Denoising Score Matching

$$\mathbb{E}_{p(\boldsymbol{x}_{\text{perturb}}|\boldsymbol{x})}\left[\frac{1}{2}\left\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})-\left(\frac{\boldsymbol{x}-\boldsymbol{x}_{\text{perturb}}}{\sigma^2}\right)\right\|_2^2\right]$$

$\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})$

+ **Minor Noise** $\mathcal{N}(0,\sigma^2 I)$

$p(\boldsymbol{x}_{\text{perturb}}|\boldsymbol{x})$



$$\nabla \log \mathcal{N}(\boldsymbol{x}_{\text{perturb}}; \boldsymbol{x}, \sigma^2 \mathrm{I})$$

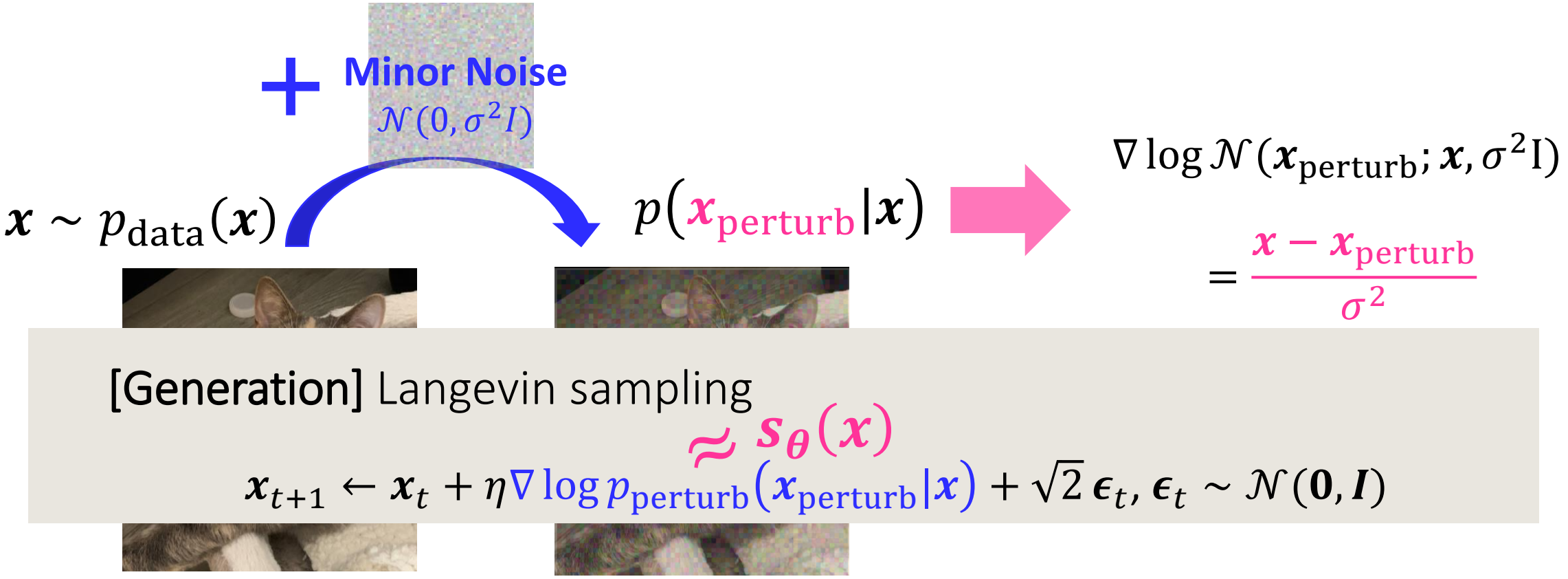
$$= \frac{\boldsymbol{x}-\boldsymbol{x}_{\text{perturb}}}{\sigma^2}$$

[Vincent '11]

Phase II (2011). Denoising Score Matching

[Training] Denoising Score Matching

$$\mathbb{E}_{p(\boldsymbol{x}_{\text{perturb}}|\boldsymbol{x})}\left[\frac{1}{2}\left\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})-\left(\frac{\boldsymbol{x}-\boldsymbol{x}_{\text{perturb}}}{\sigma^2}\right)\right\|_2^2\right]$$



[Generation] Langevin sampling

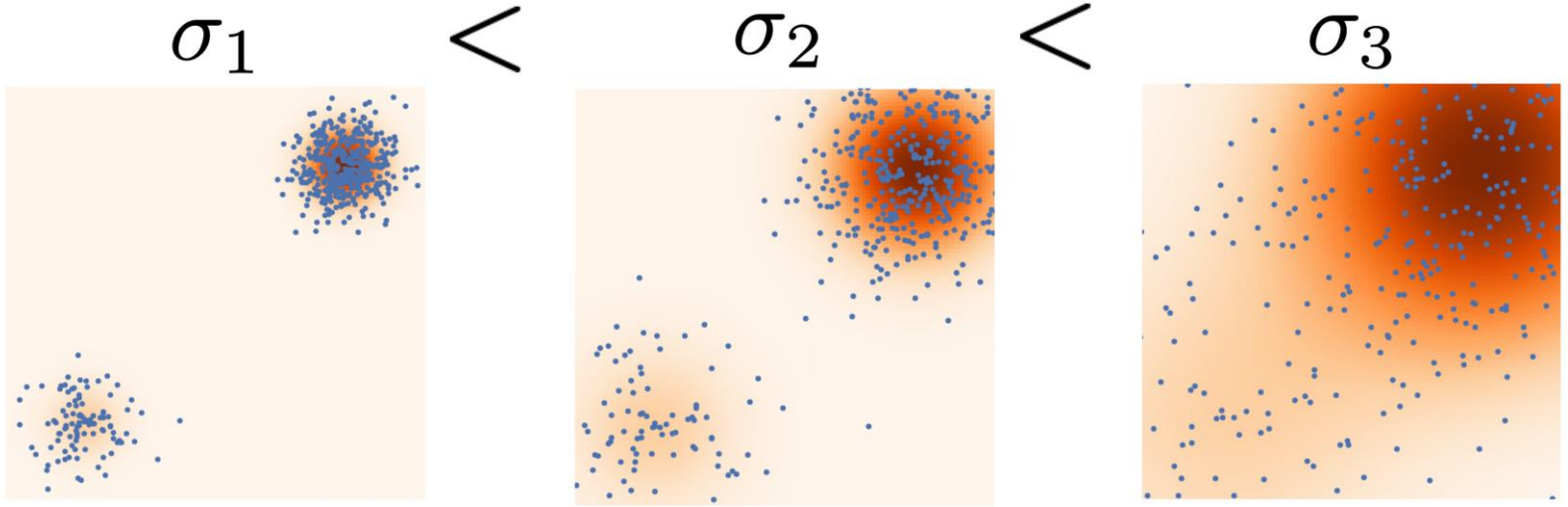
$$\boldsymbol{x}_{t+1}\leftarrow \boldsymbol{x}_t+\eta\nabla\log p_{\text{perturb}}(\boldsymbol{x}_{\text{perturb}}|\boldsymbol{x})+\sqrt{2}\,\boldsymbol{\epsilon}_t,\ \boldsymbol{\epsilon}_t\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$$

$$\approx \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})$$

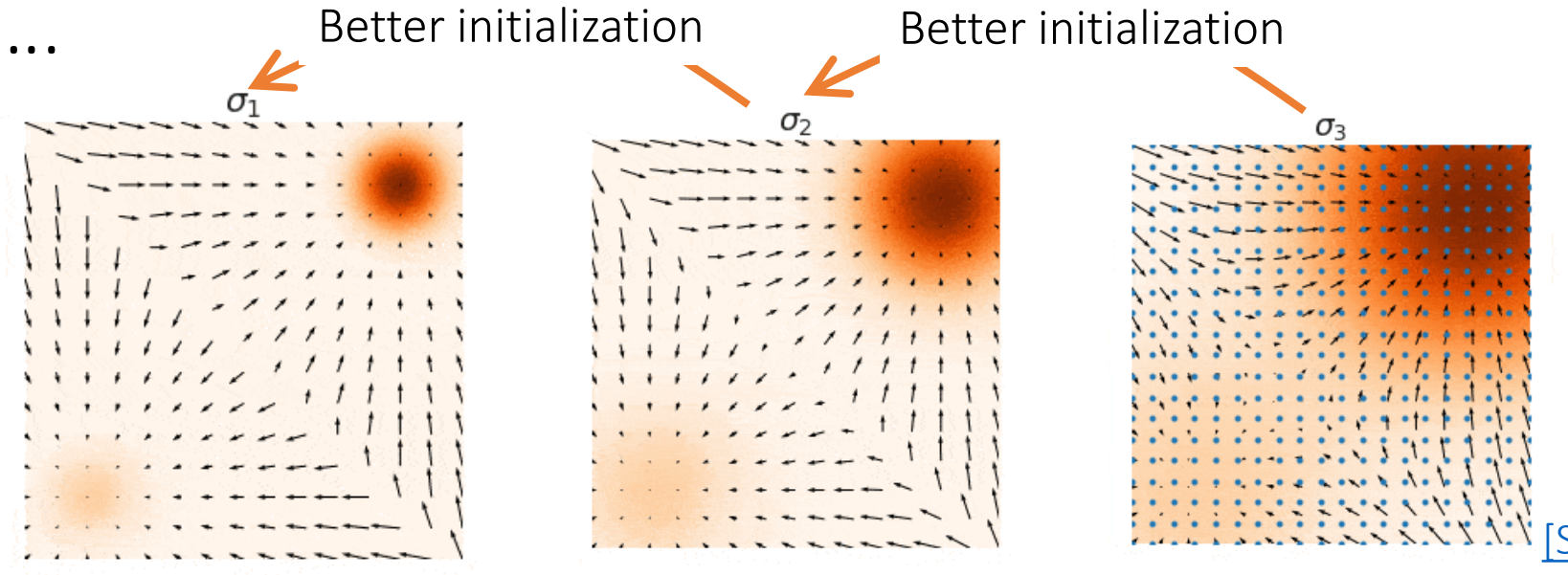
[Vincent '11]

Phase III (2019). Noised Conditioned Denoising Score Matching

Injecting incremental multiple levels of noises to data



In generation….



[Song+ NeurIPS'19]

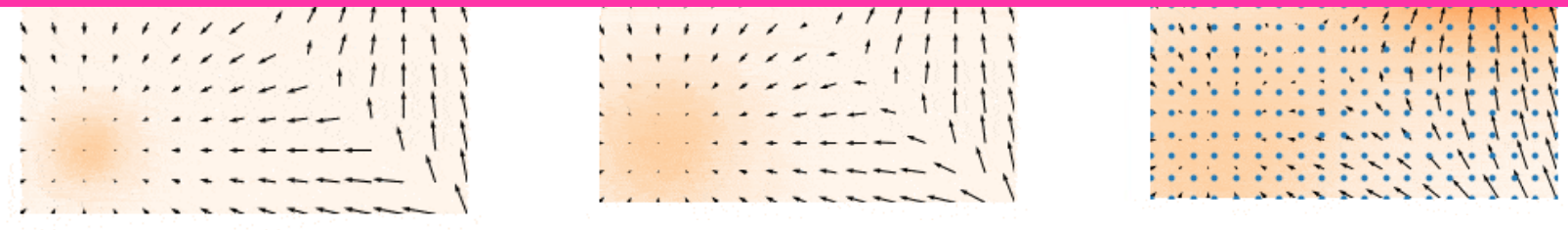
Phase III (2019). Noised Conditioned Denoising Score Matching

Injecting incremental multiple levels of noises to data

$$\sigma_1 < \sigma_2 < \sigma_3$$

$$\sum_{i=1}^{3} \omega_i \, \mathbb{E}_{p\left(x_{\text{perturb},\sigma_i}|x\right)} \left[\frac{1}{2} \left\| \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, \sigma_i) - \left(\frac{x - x_{\text{perturb},\sigma_i}}{{\sigma_i}^2} \right) \right\|_2^2 \right]$$

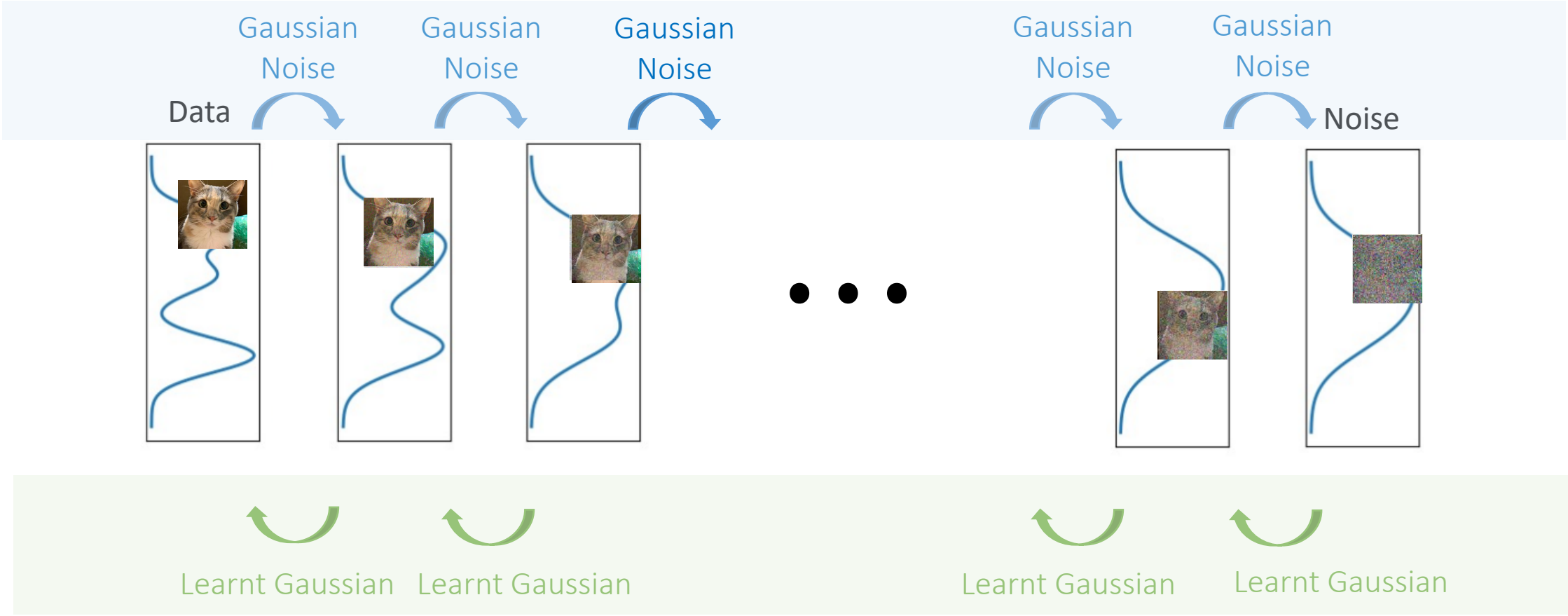
The vibe of Diffusion Model appeared…



[Song+ NeurIPS'19]

Concurrently, **D**enoising **D**iffusion **P**robabilistic **M**odel (DDPM)

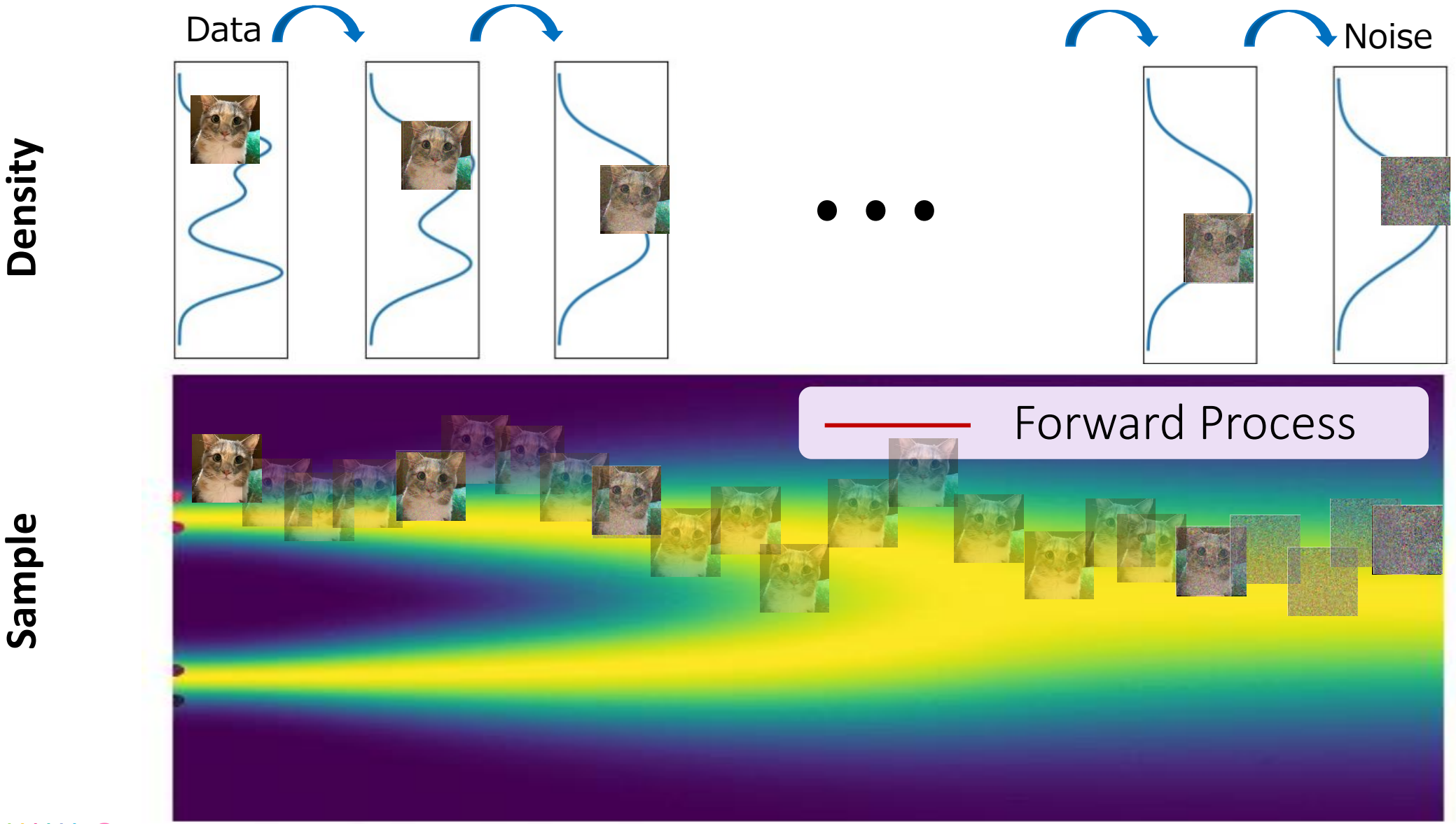
- Different forward noise schedulers



- Bayesian viewpoint for reverse process – learn reversal Gaussian

[Ho+ NeurIPS '20]

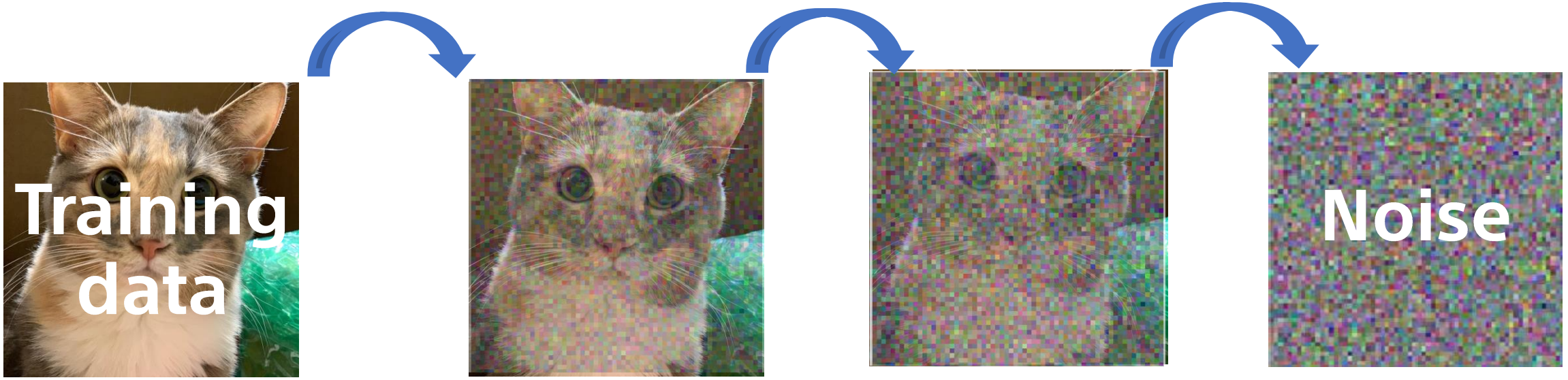
Phase IV (2020). Score-Based Continuous Time Diffusion Model



[Song+ ICLR'21]

Diffusion Model at High Level

Forward Process: Training data → white noise (simple space)



Diffusion Model at High Level

Backward Process: Learn NN to gradually denoise

Generated new image

Gradually transforming noise to a new image

Diffusion Model's **Forward** Process

Forward Process:

- Data → Noise (Common simple space)
- With user specified noise schedulers a_t and b_t

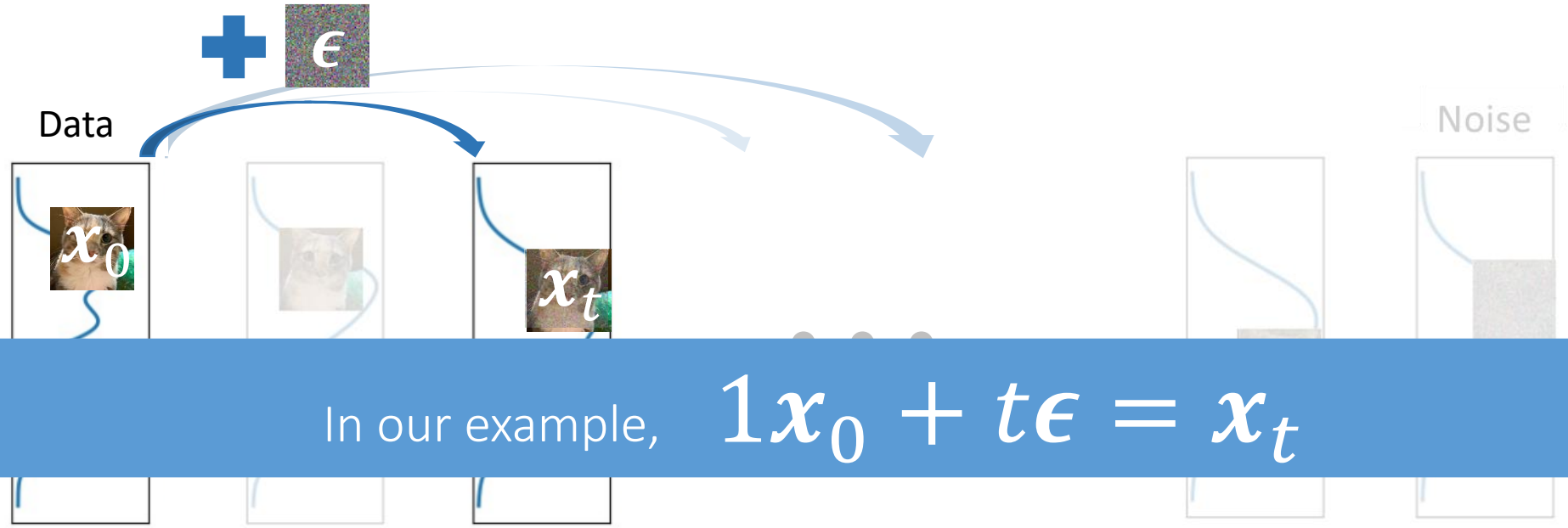
$$a_t \boldsymbol{x}_0 + b_t \boldsymbol{\epsilon} = \boldsymbol{x}_t$$

[Ho+ NeurIPS '20; Song+ ICLR'21]

Diffusion Model's **Forward** Process

Forward Process:

- Data → Noise (Common simple space)
- With user specified noise schedulers a_t and b_t

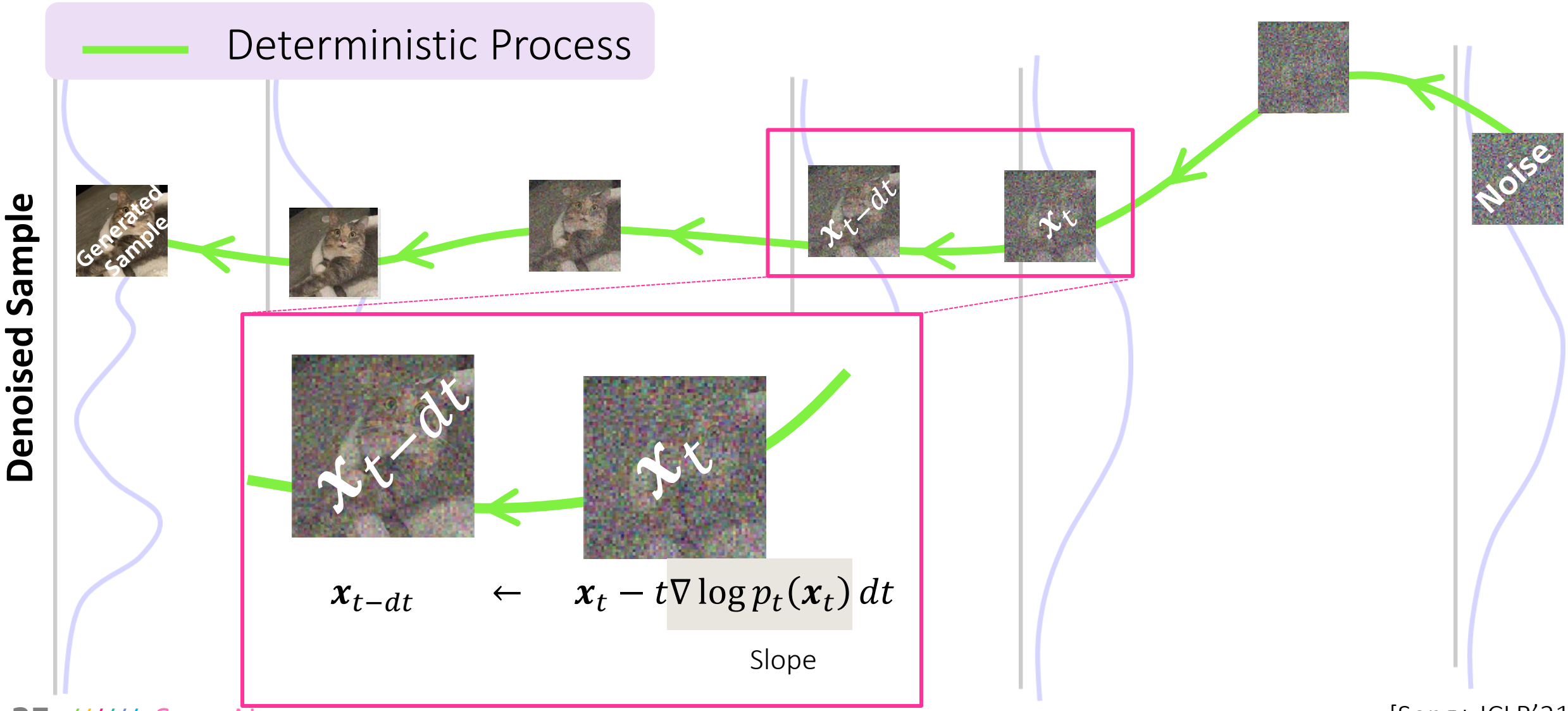


In our example, $1\boldsymbol{x}_0 + t\boldsymbol{\epsilon} = \boldsymbol{x}_t$

$$a_t\boldsymbol{x}_0 + b_t\boldsymbol{\epsilon} = \boldsymbol{x}_t$$

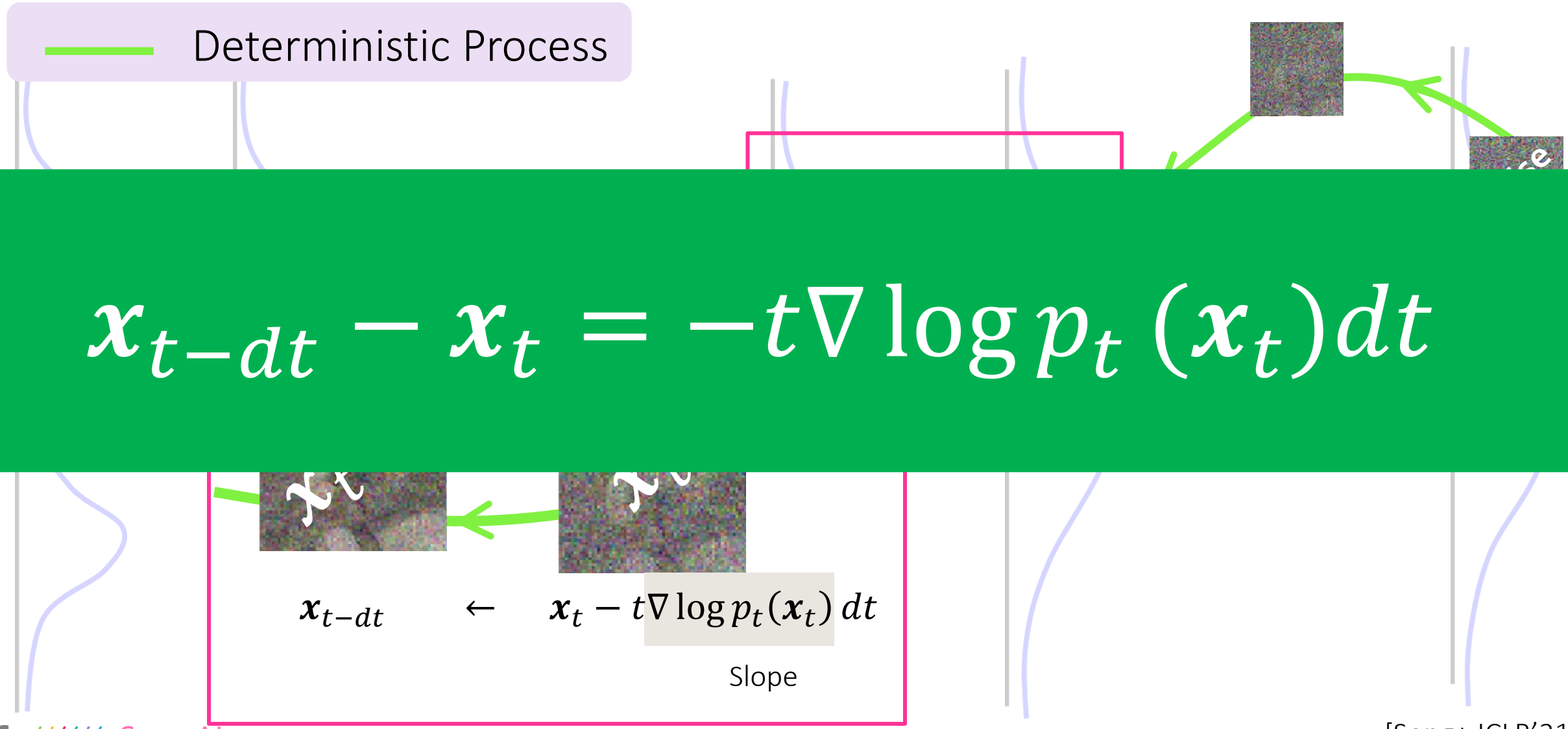
[Ho+ NeurIPS '20; Song+ ICLR'21]

Diffusion Model's **Backward Deterministic** Process (Generation)



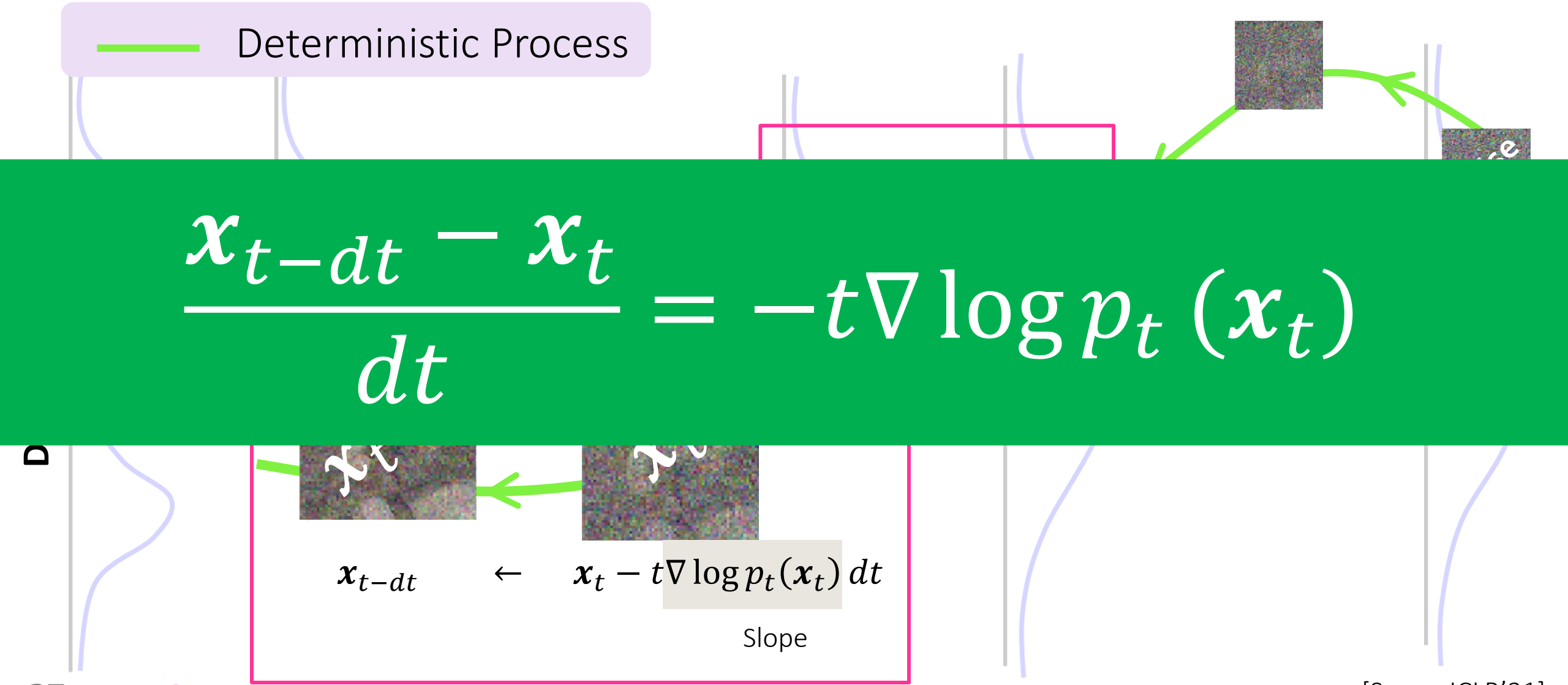
[Song+ ICLR'21]

Diffusion Model's **Backward Deterministic** Process (Generation)



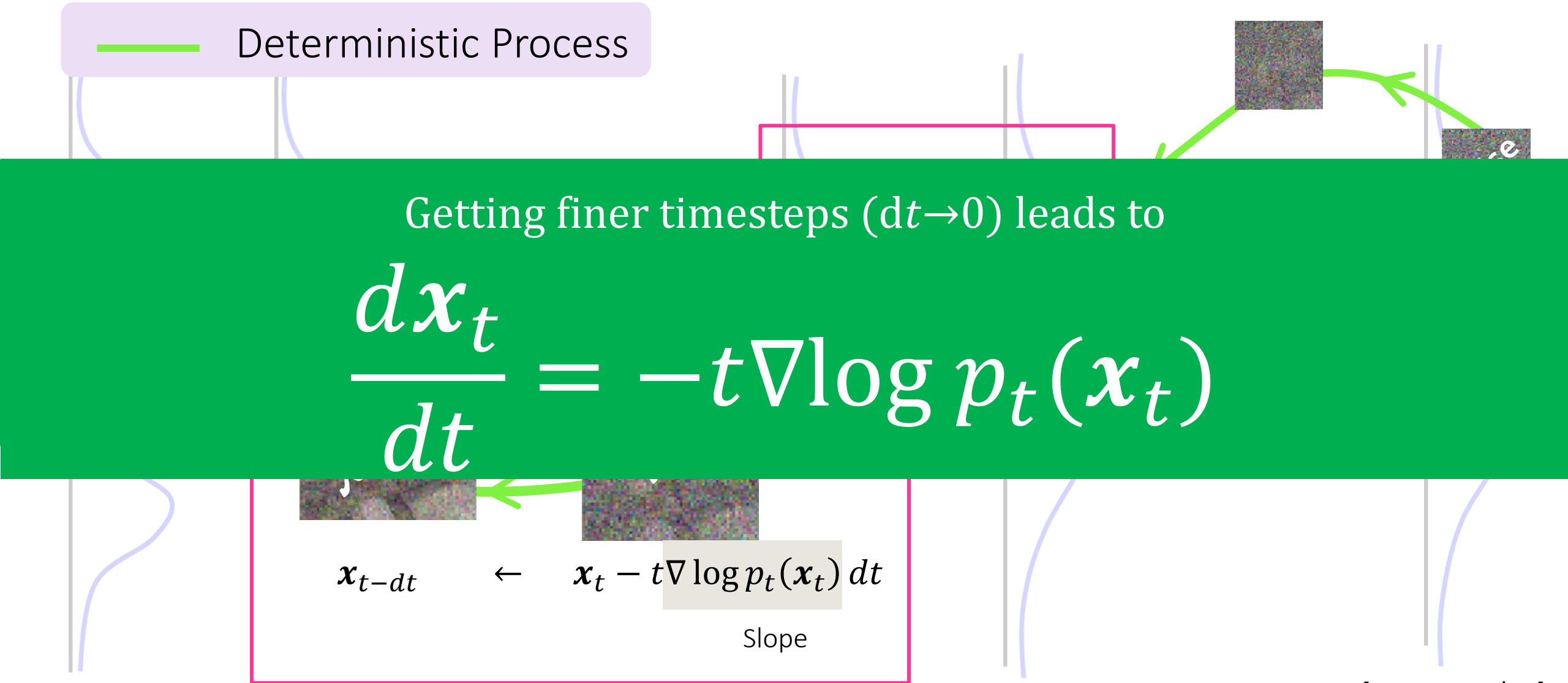
[Song+ ICLR'21]

Diffusion Model's **Backward Deterministic** Process (Generation)



[Song+ ICLR'21]

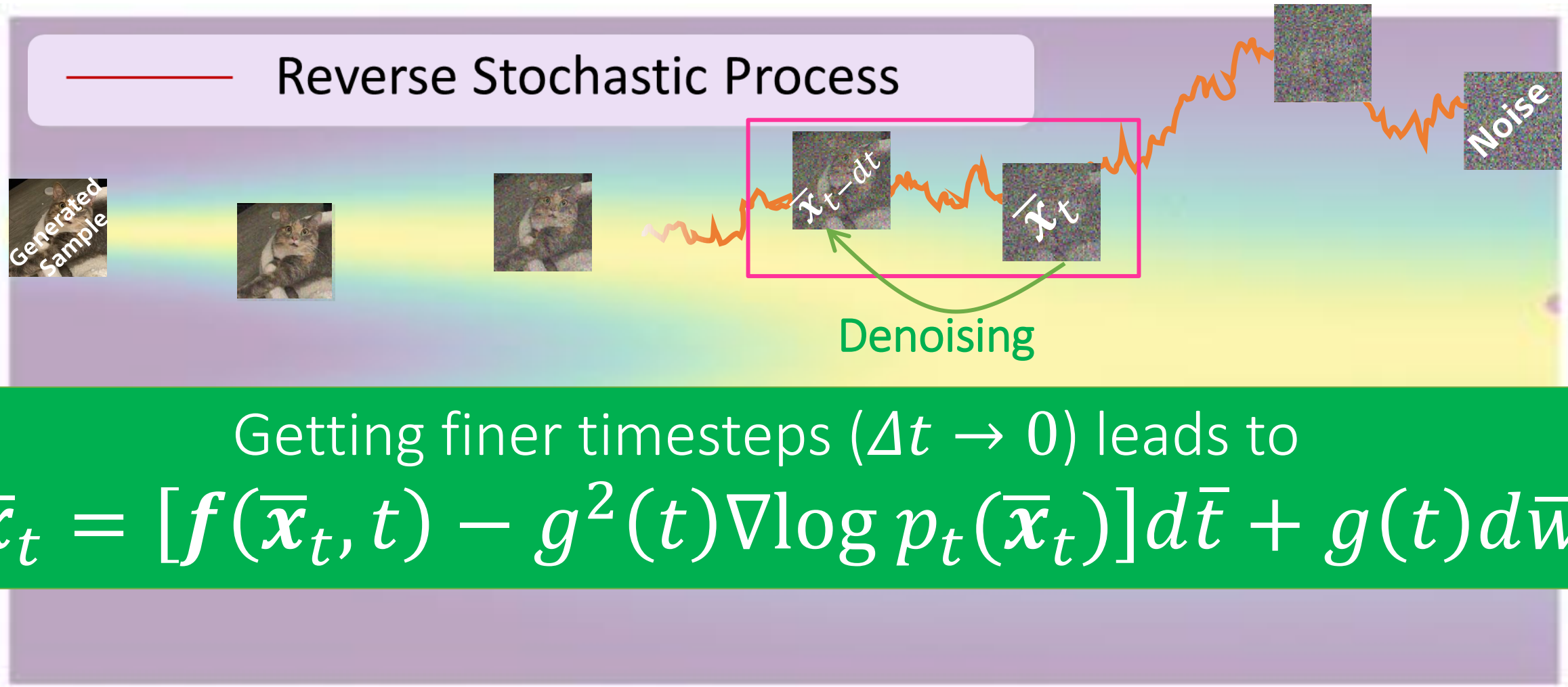
Diffusion Model's **Backward Deterministic** Process (Generation)



[Song+ ICLR'21]

Diffusion Model's **Backward Stochastic** Process (Generation)

Users predefined noise schedulers: $f(x_t, t) = 0$ & $g(t) = \sqrt{2t}$



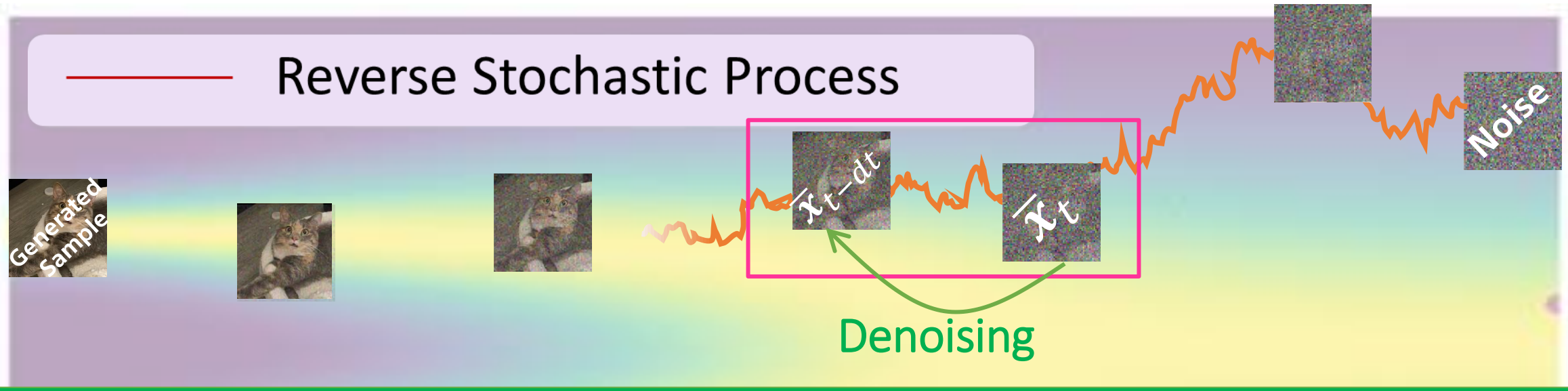
Getting finer timesteps ($\Delta t \to 0$) leads to

$$d\bar{\boldsymbol{x}}_t = [\boldsymbol{f}(\bar{\boldsymbol{x}}_t, t) - g^2(t)\nabla \log p_t(\bar{\boldsymbol{x}}_t)]d\bar{t} + g(t)d\bar{w}_t$$

[Song+ ICLR'21]

Diffusion Model's **Backward Stochastic** Process (Generation)

Users predefined noise schedulers: $f(x_t, t) = 0$ & $g(t) = \sqrt{2t}$



In our example,

$$d\bar{\boldsymbol{x}}_t = [-2t\nabla \log p_t(\bar{\boldsymbol{x}}_t)]d\bar{t} + \sqrt{2t}d\bar{w}_t$$

[Song+ ICLR'21]

Diffusion M (Generation)

Users predefine

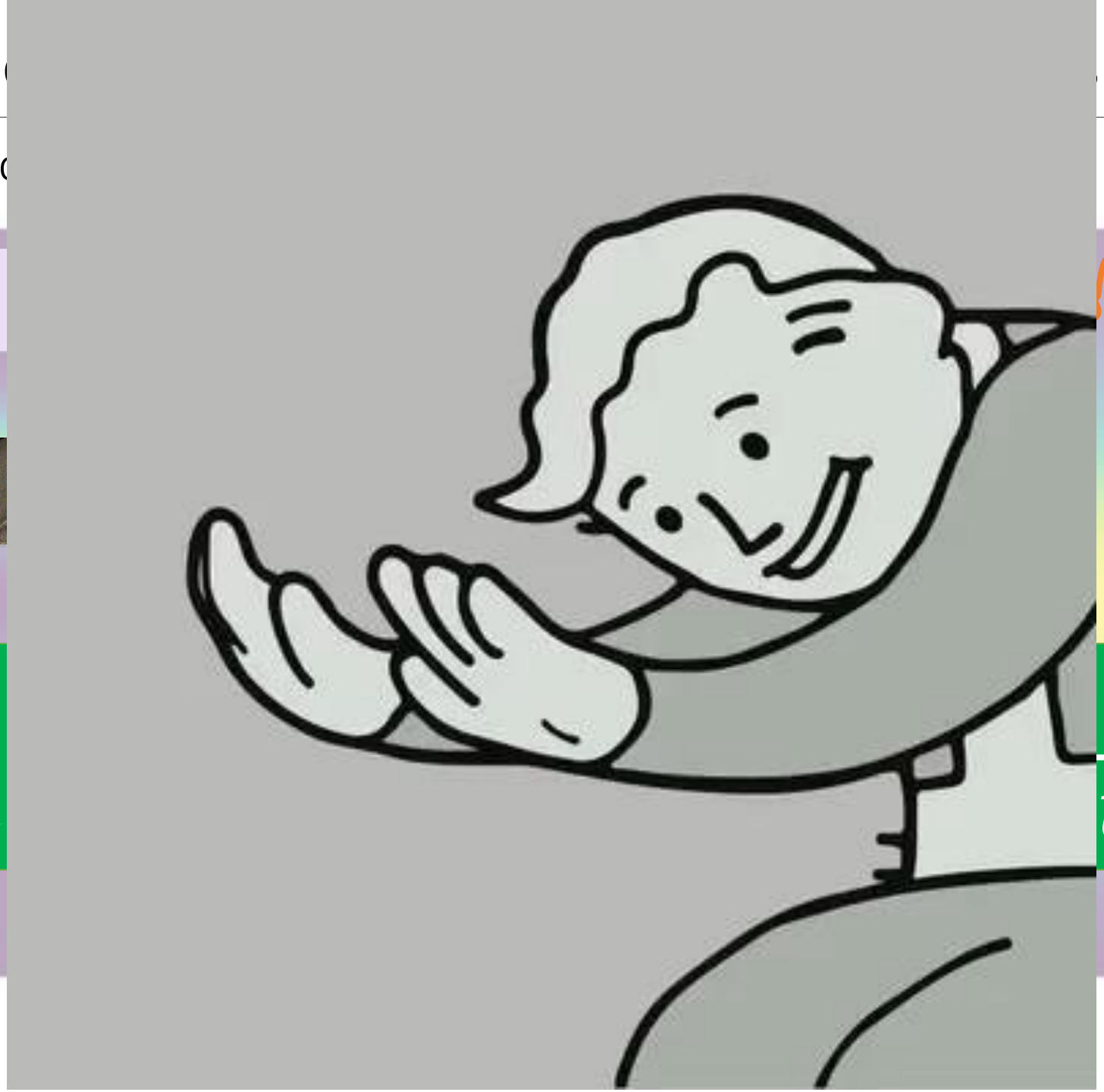
sed Sample

Generated Sample

Noise

$d\bar{\boldsymbol{x}}_t$

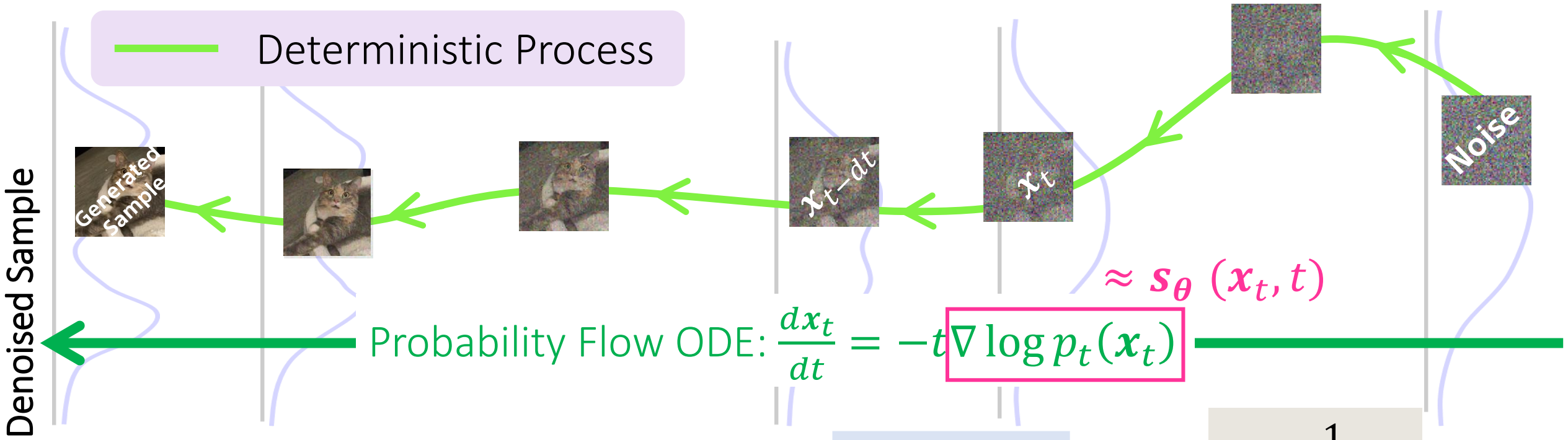
$t d\bar{w}_t$



[Song+ ICLR'21]

Score-Based Diffusion Model's Training and Sampling

Users predefined noise schedulers: $\boldsymbol{x_t} = a_t \boldsymbol{x_0} + b_t \boldsymbol{\epsilon}$ with $a_t = 1$ & $b_t = t$



Denoising score matching:

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\mathbb{E}_{t\in\mathcal{U}[0,1]}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\lambda_t\left\|\boldsymbol{s_\theta}(\boldsymbol{x}_t,t) - \nabla \log p_t\left(\boldsymbol{x}_t|\boldsymbol{x}_0\right)\right\|^2\right]$$

[Song+ ICLR'21]

Score-Based Diffusion Model's Training and Sampling

Users predefined noise schedulers: $\boldsymbol{x_t} = a_t \boldsymbol{x_0} + b_t \boldsymbol{\epsilon}$ with $a_t = 1$ & $b_t = t$

$$= -\frac{1}{t}\boldsymbol{\epsilon}$$

Denoising score matching:

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\mathbb{E}_{t\in\mathcal{U}[0,1]}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\lambda_t \left\| \boldsymbol{s_\theta}(\boldsymbol{x}_t, t) - \nabla \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)) \right\|^2\right]$$

Training DM

Initialize: $\boldsymbol{s_\theta}(\boldsymbol{x}_t, t)$

- Get data $\boldsymbol{x}_0$
- Sample time $t \sim \mathcal{U}(0,1)$ and noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Perturb $\boldsymbol{x}_t = \boldsymbol{x}_0 + t\boldsymbol{\epsilon}$
- Training with Denoising Score Matching:

$$\nabla_{\boldsymbol{\theta}} \left\| \boldsymbol{s_\theta}(\boldsymbol{x}_t, t) - \left(-\frac{1}{t}\boldsymbol{\epsilon}\right) \right\|^2$$

[Song+ ICLR'21]

Score-Based Diffusion Model's Training and Sampling

$$\nabla_{x_t} \log p(\boldsymbol{x}_t) \approx \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$$

Training DM

Initialize: $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$

- Get data $\boldsymbol{x}_0$
- Sample time $t \sim \mathcal{U}(0,1)$ and noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Perturb $\boldsymbol{x}_t = \boldsymbol{x}_0 + t\boldsymbol{\epsilon}$
- Training with Denoising Score Matching:

$$\nabla_{\boldsymbol{\theta}} \left\| \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \left(-\frac{1}{t}\boldsymbol{\epsilon}\right) \right\|^2$$

For instance, Euler method

Starting from $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, T^2\boldsymbol{I})$, any ODE solver can be used to solve

$$\frac{d\boldsymbol{x}_t}{dt} = -t\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t),$$

by calling a library, such as

`scipy.integrate.odeint`

Score-Based Diffusion Model's Training and Sampling

$$\nabla_{x_t} \log p(\boldsymbol{x}_t) \approx \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$$

Training DM

Initialize: $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$

- Get data $\boldsymbol{x}_0$
- Sample time $t \sim \mathcal{U}(0,1)$ and noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Perturb $\boldsymbol{x}_t = \boldsymbol{x}_0 + t\boldsymbol{\epsilon}$
- Training with Denoising Score Matching:

$$\nabla_{\boldsymbol{\theta}} \left\| \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \left(-\frac{1}{t}\boldsymbol{\epsilon}\right) \right\|^2$$

Sampling with DM

Request: $T = t_{N-1} > \cdots > t_1 > t_0 = 0$

Initialize: $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, T^2\boldsymbol{I})$

- For $i = N-1, \cdots, 1, 0$
  - $dt = t_{i-1} - t_i$
  - $\boldsymbol{x}_{t_{i-1}} \leftarrow \boldsymbol{x}_{t_i} - t_i \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_i}, t_i) dt$

Different (but Equivalent) Parametrizations

Given $\boldsymbol{x}_t$ at time t

Predicting score
$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$

Predict clean
$\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$

Predict noise
$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$

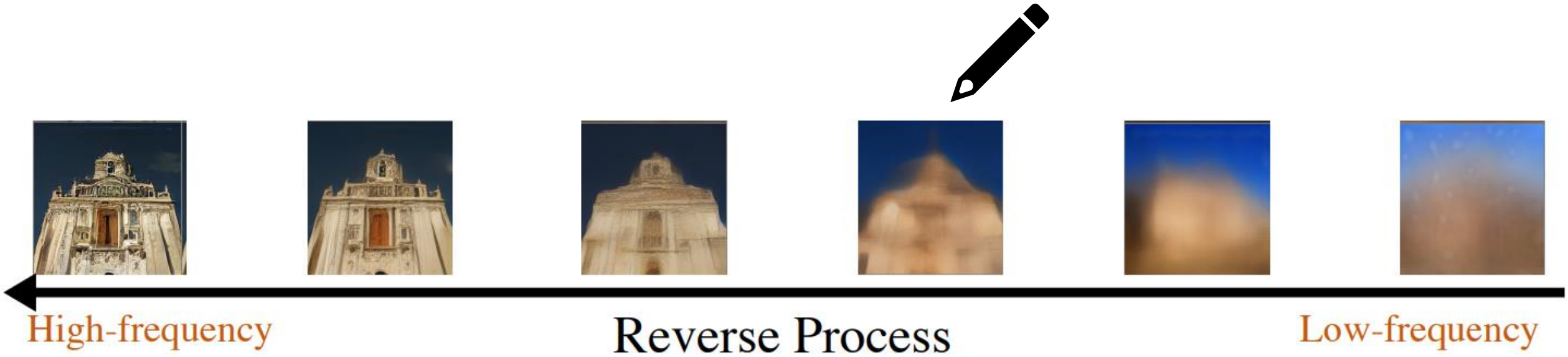
$$a_t \boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \boldsymbol{x}_t + b_t^2 \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$$

$$\frac{\boldsymbol{x}_t - b_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)}{a_t} = \boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$$

$$\left\| \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \left(\frac{a_t \boldsymbol{x}_0 - \boldsymbol{x}_t}{b_t^2} \right) \right\|^2 \cong \left\| \boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \boldsymbol{x}_0 \right\|^2$$

$$\cong \left\| \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon} \right\|^2$$

What is Happening in Generation?



Fine-grained details

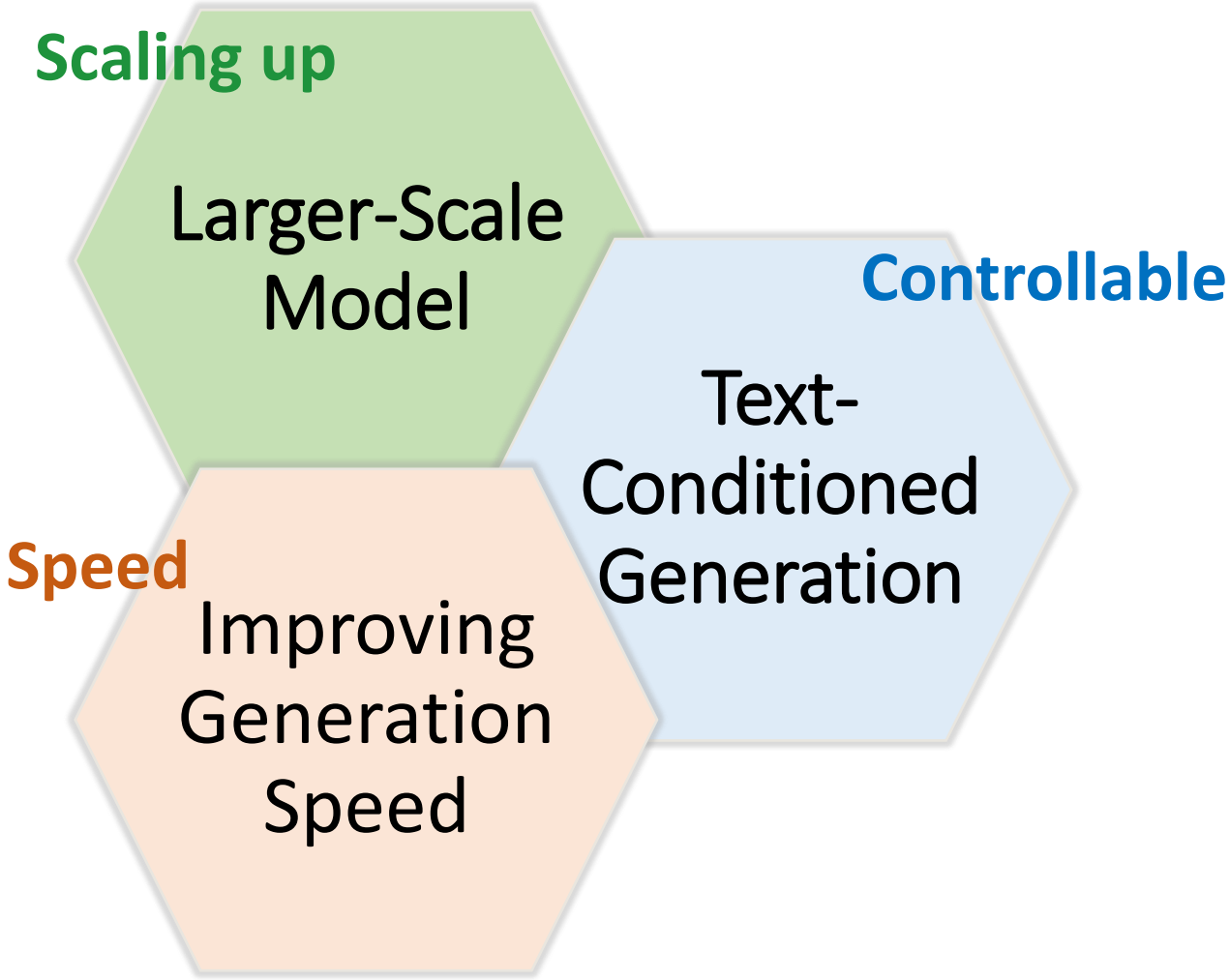
Global structure

Gradual denoising process, on the other hand,
offers flexibility and control!

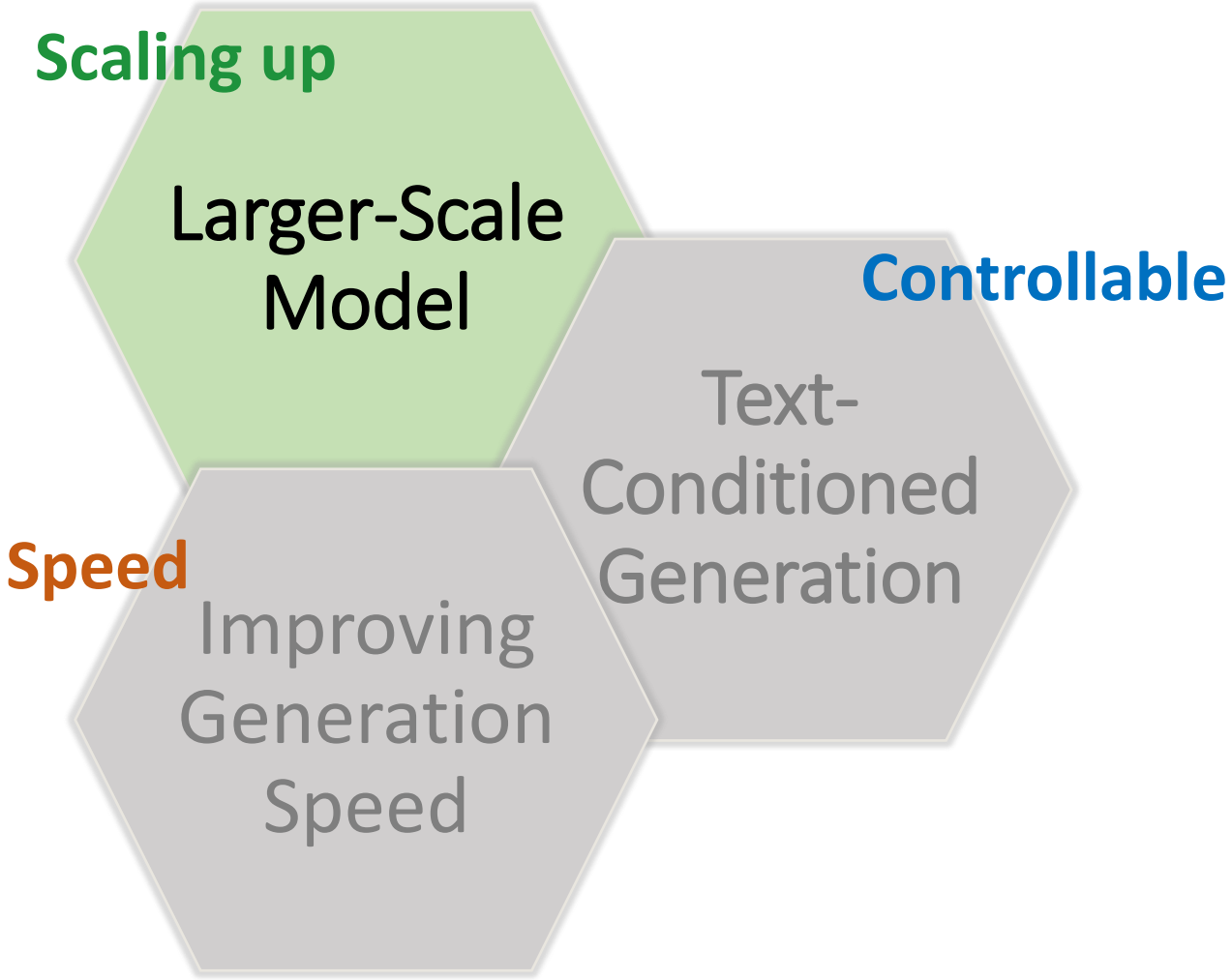
[Yang+ CVPR'23]

Making Diffusion Model Powerful

Making Diffusion Model More Powerful



Making Diffusion Model More Powerful



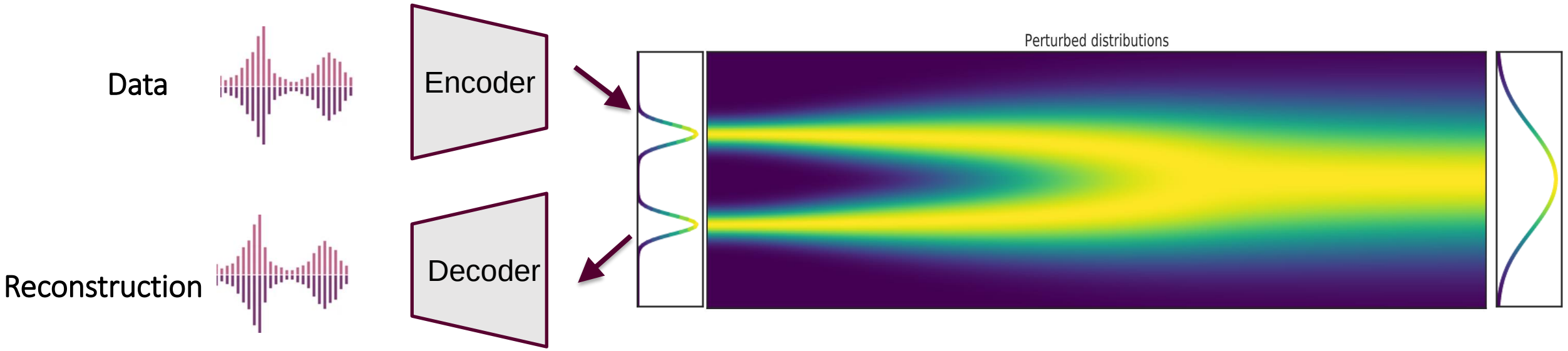
Latent Diffusion Model

Challenge: Music generation requires modeling long temporal sequences.

A solution:

Compress music data via autoencoder →

Examples: MusicLDM, Riffusion, Stable Audio etc..



Diffusion Model in Latent Space

[Vahdat+ NeurIPS'21; Rombach+ CVPR'22]

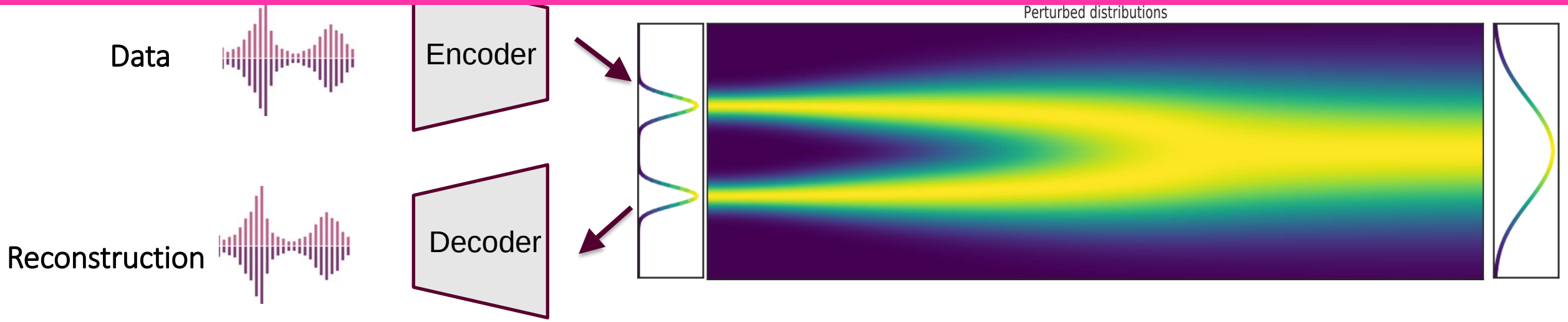
Latent Diffusion Model

Challenge: Music generation requires modeling long temporal sequences.

A solution:

Compress music data via autoencoder →

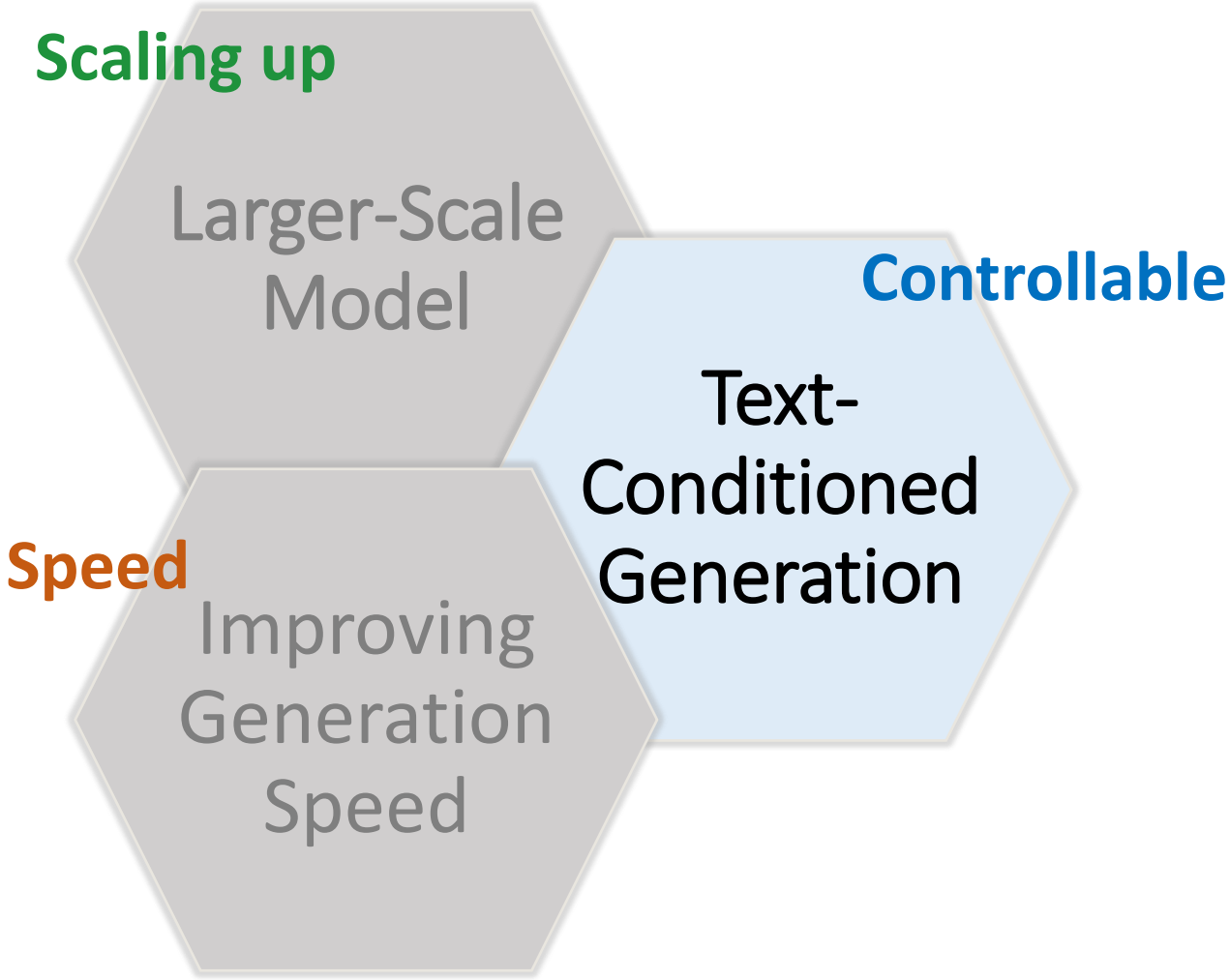
More Introduction in Session 2…



Diffusion Model in Latent Space

[Vahdat+ NeurIPS'21; Rombach+ CVPR'22]

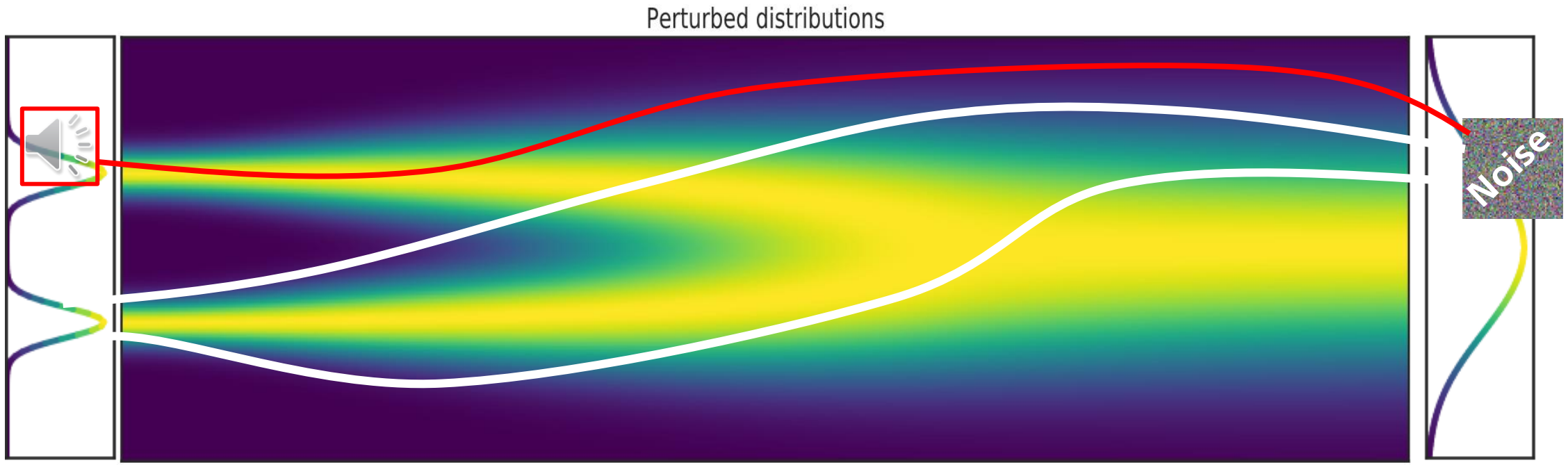
Making Diffusion Model More Powerful



Text-Conditional Generation for Controllability

Goal: Make diffusion model generates music in response to user specified text prompts

Frogs croaking and a humming with insects vocalizing.



Classifier-Free Guidance (CFG) for Conditional Generation

Given a text condition: $\boldsymbol{c}_{\text{text}}$.

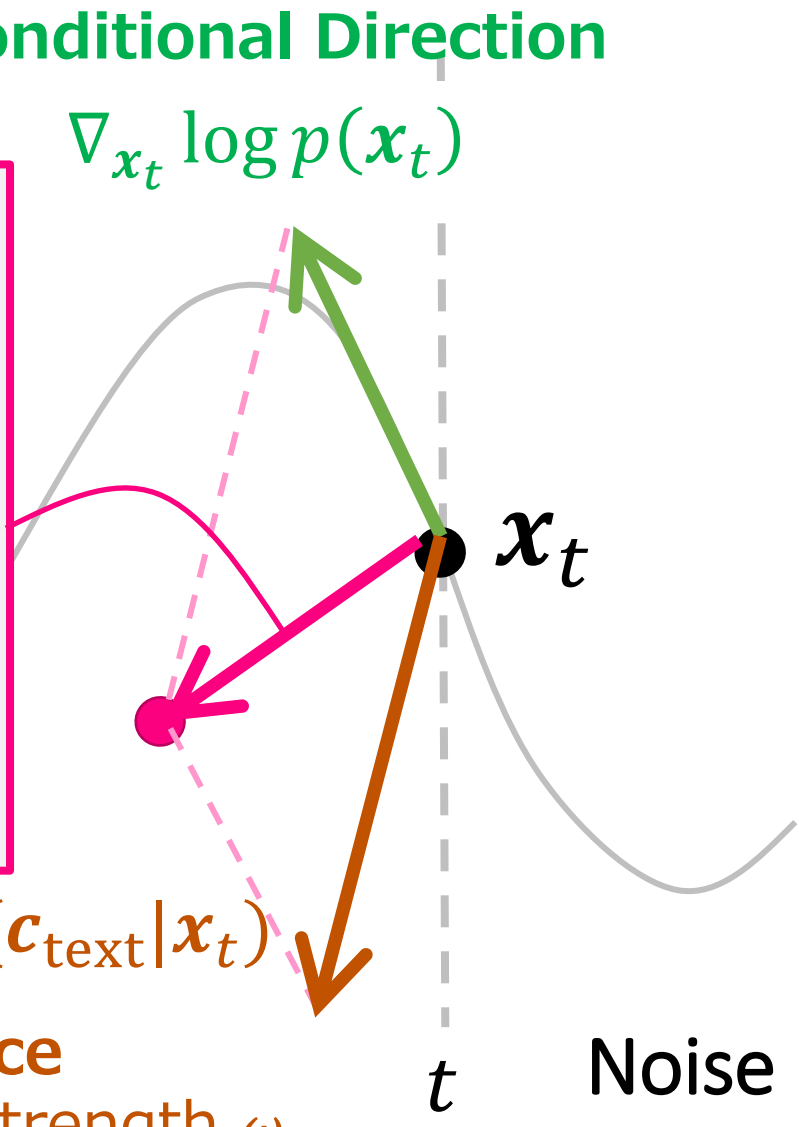
$$\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t) + \omega \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{c}_{\text{text}}|\boldsymbol{x}_t)$$

Bayes' rule

$$= (1-\omega)\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t) + \omega \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{c}_{\text{text}})$$

$$\approx \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, \text{null}) \qquad \approx \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, \boldsymbol{c}_{\text{text}})$$

$$= (1-\omega)\text{Uncon. Direction} + \omega\text{Con. Direction}$$



[Ho+ '22 CFG]

Training and Generation with CFG

Joint training of (un-) conditional DM

Require: κ ratio of Null condition

Initialize: $s_\theta(x_t, t, c_{\text{text}})$

- Get paired data (x, c_{text})
- $c_{\text{text}} \leftarrow$ Null with ratio κ
- Sample time $t \sim \mathcal{U}(0,1)$ and noise $\epsilon \sim \mathcal{N}(0, I)$
- Perturb $x_t = x_0 + t\epsilon$
- Training with Denoising Score Matching:

$$\nabla_\theta \left\| s_\theta(x_t, t, c_{\text{text}}) - \left(-\frac{1}{t}\epsilon\right) \right\|^2$$

Training DM

Initialize: $s_\theta(x_t, t)$

- Get data x_0
- Sample time $t \sim \mathcal{U}(0,1)$ and noise $\epsilon \sim \mathcal{N}(0, I)$
- Perturb $x_t = x_0 + t\epsilon$
- Training with Denoising Score Matching:

$$\nabla_\theta \left\| s_\theta(x_t, t) - \left(-\frac{1}{t}\epsilon\right) \right\|^2$$

Training and Generation with CFG

Joint training of (un-) conditional DM

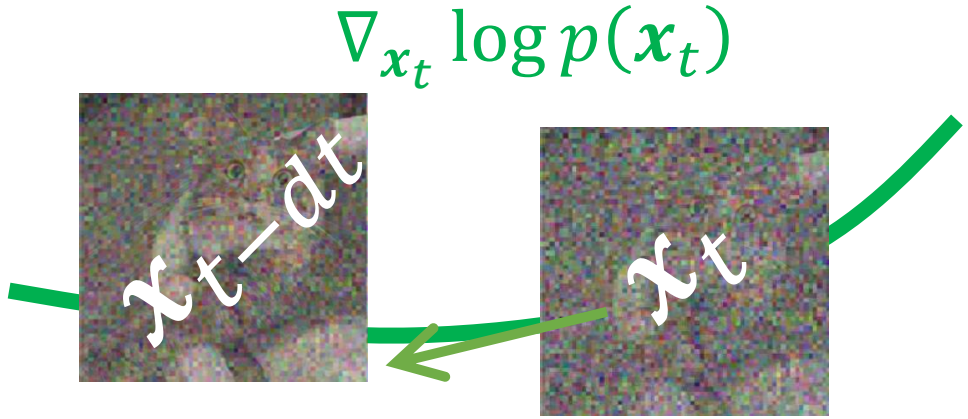
Require: κ ratio of Null condition

Initialize: $s_\theta(x_t, t, c_{\text{text}})$

- Get paired data (x, c_{text})
- $c_{\text{text}} \leftarrow$ Null with ratio κ
- Sample time $t \sim \mathcal{U}(0,1)$ and noise $\epsilon \sim \mathcal{N}(0, I)$
- Perturb $x_t = x_0 + t\epsilon$
- Training with Denoising Score Matching:

$$\nabla_\theta \left\| s_\theta(x_t, t, c_{\text{text}}) - \left(-\frac{1}{t}\epsilon\right) \right\|^2$$

Sampling with CFG



$$x_{t-dt} \leftarrow x_t - t\nabla \log p_t(x_t)\, dt$$

Training and Generation with CFG

Joint training of (un-) conditional DM

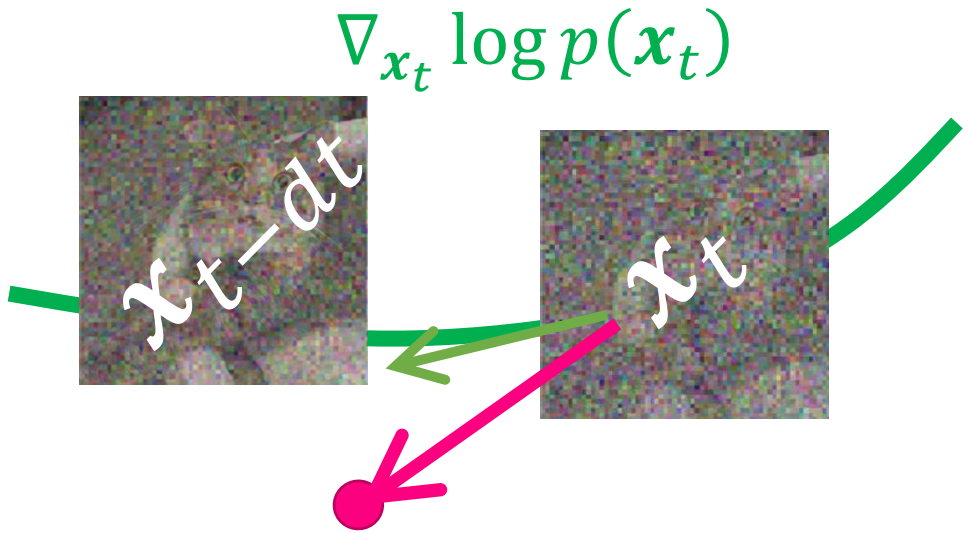
Require: κ ratio of Null condition

Initialize: $s_\theta(x_t, t, c_{\text{text}})$

- Get paired data (x, c_{text})
- $c_{\text{text}} \leftarrow$ Null with ratio κ
- Sample time $t \sim \mathcal{U}(0,1)$ and noise $\epsilon \sim \mathcal{N}(0, I)$
- Perturb $x_t = x_0 + t\epsilon$
- Training with Denoising Score Matching:

$$\nabla_\theta \left\| s_\theta(x_t, t, c_{\text{text}}) - \left(-\frac{1}{t}\epsilon\right) \right\|^2$$

Sampling with CFG

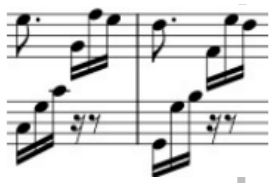


$$x_{t-dt}^{\text{CFG}} \leftarrow x_t - t\,\boxed{\phantom{(1-\omega)s_\theta(x_t,t,\text{Null})+\omega s_\theta(x_t,t,c_{\text{text}})}}\,dt$$

$$(1-\omega)s_\theta(x_t, t, \text{Null}) + \omega s_\theta(x_t, t, c_{\text{text}})$$

Training and Generation with CFG

Given a text condition: $\boldsymbol{c}_{\text{text}}$.



Clean

0

Unconditional Direction

$\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$

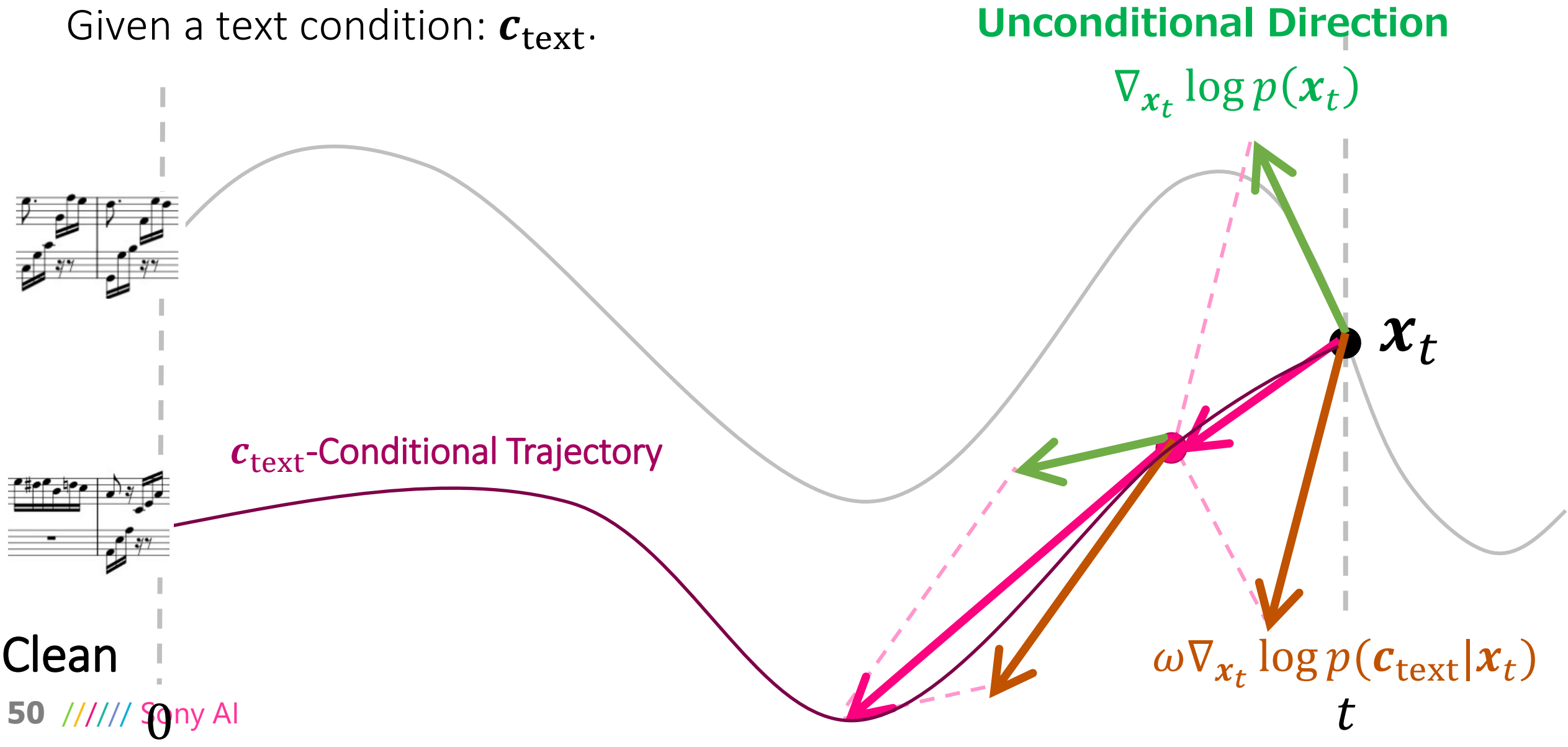
$\boldsymbol{x}_t$

$\omega \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{c}_{\text{text}} | \boldsymbol{x}_t)$

t

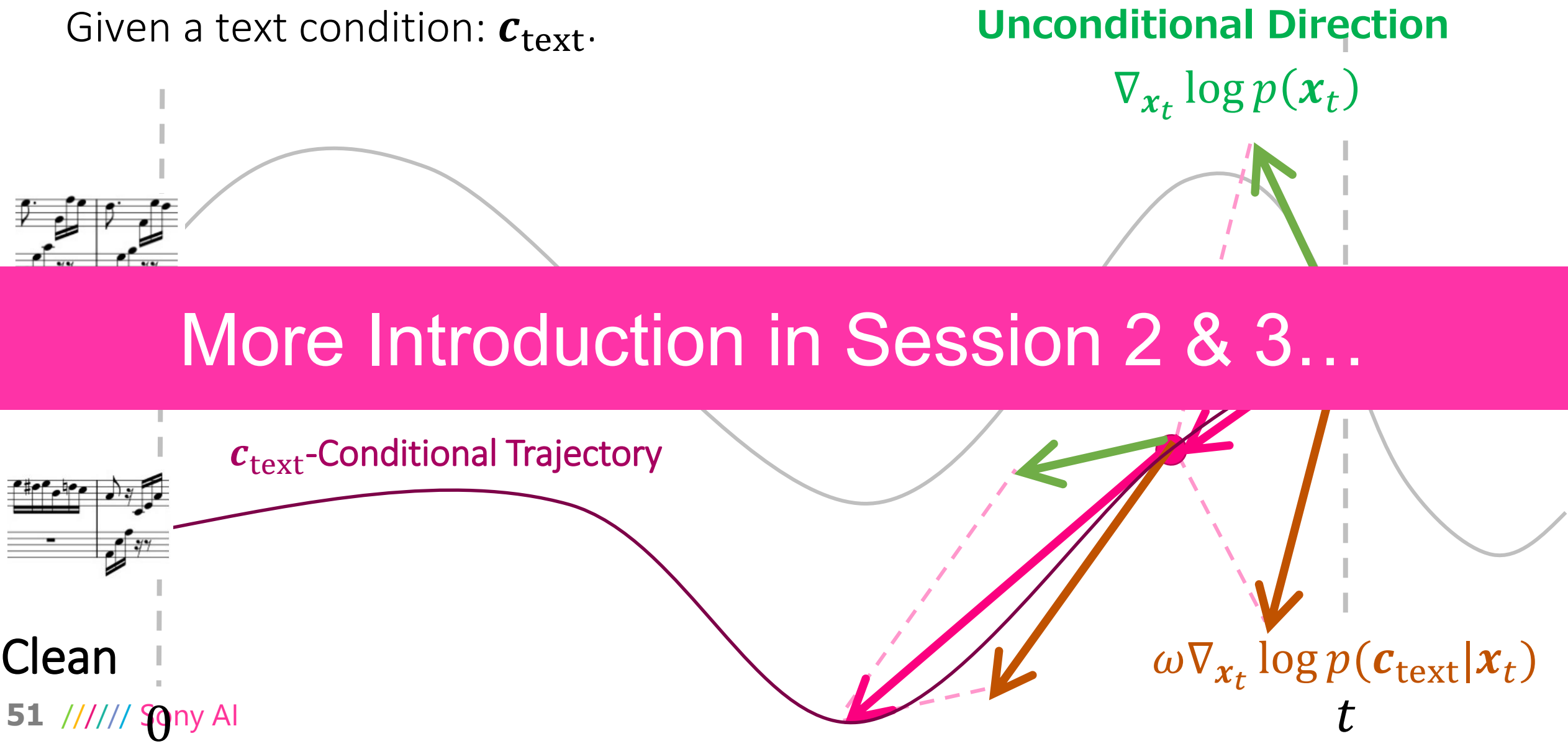
Training and Generation with CFG

Given a text condition: $\boldsymbol{c}_{\text{text}}$.



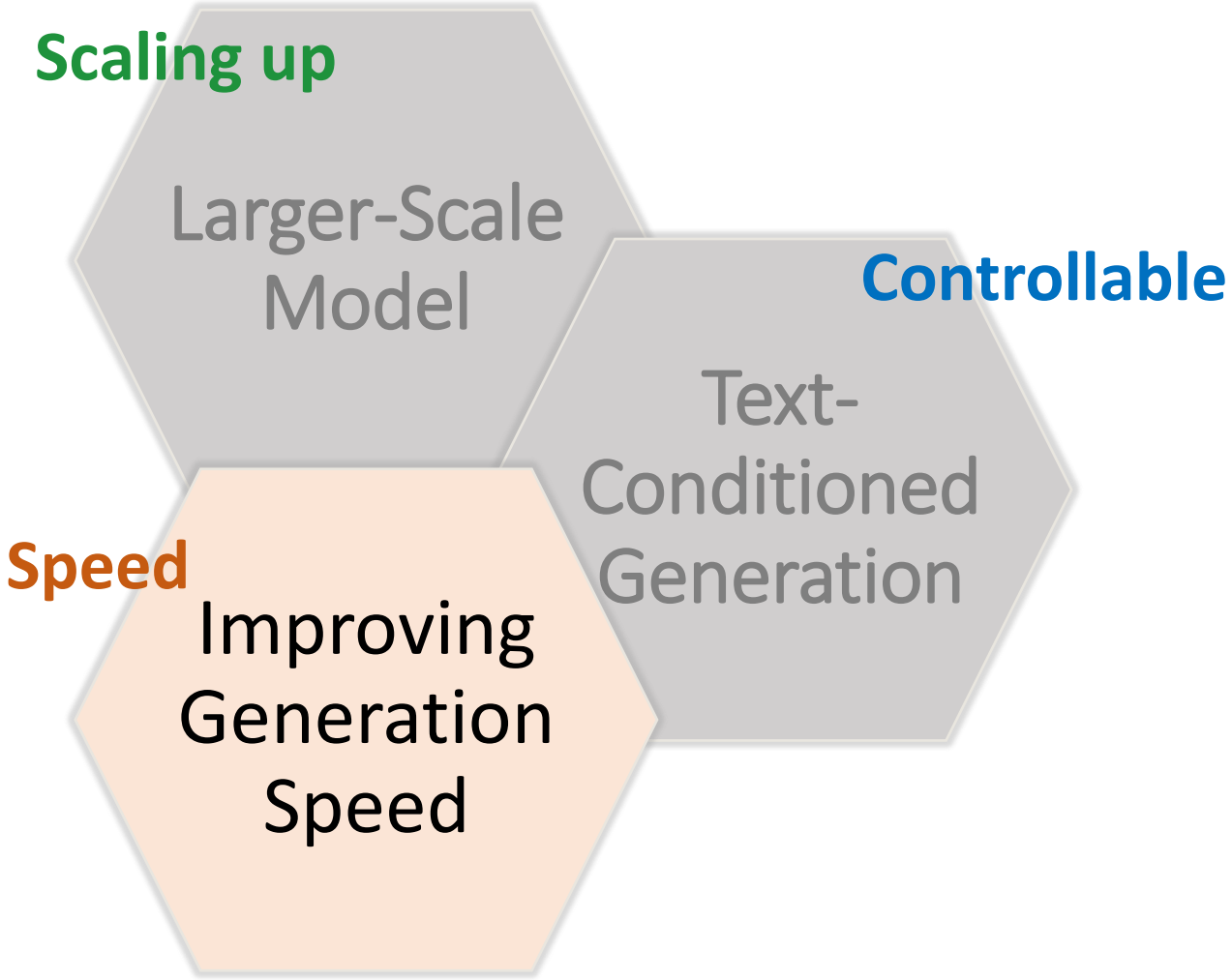
Training and Generation with CFG

Given a text condition: $\boldsymbol{c}_{\text{text}}$.



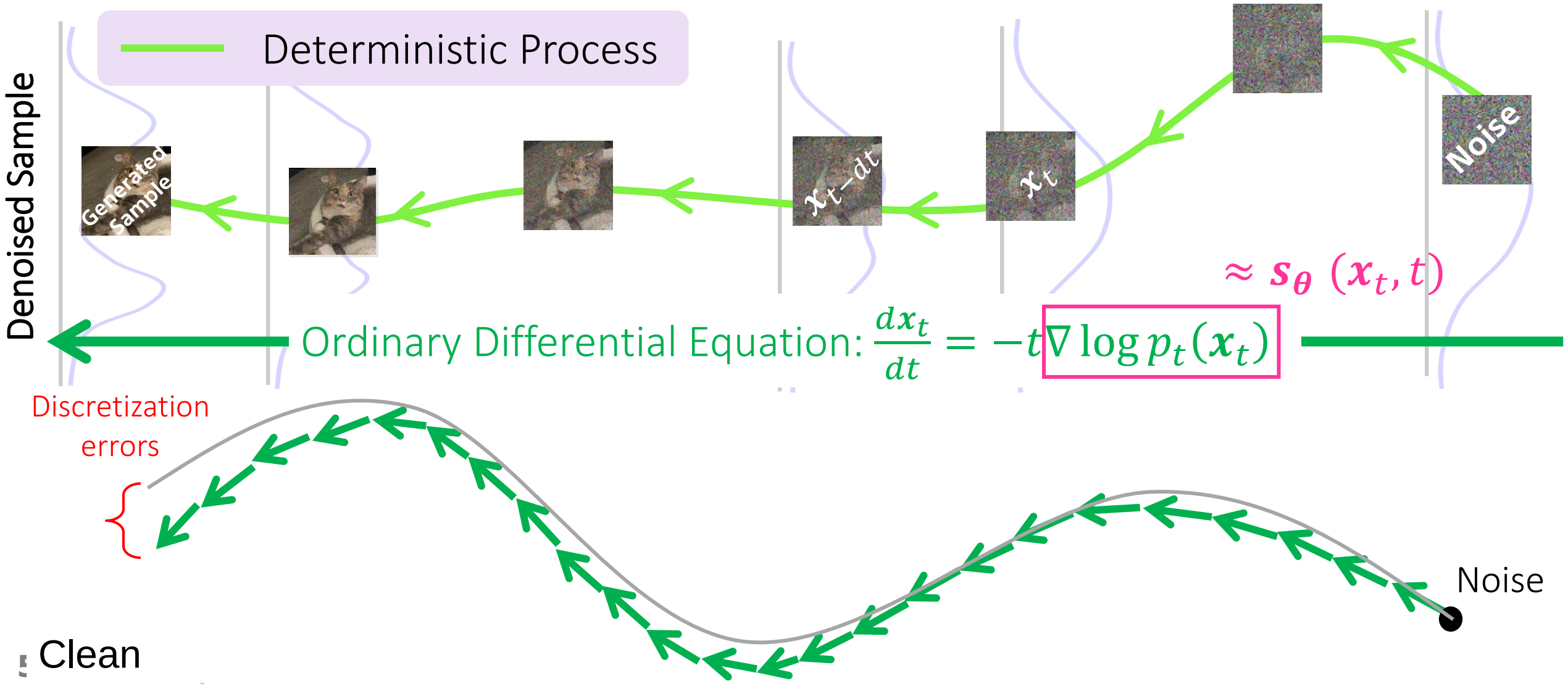
More Introduction in Session 2 & 3…

Making Diffusion Model More Powerful



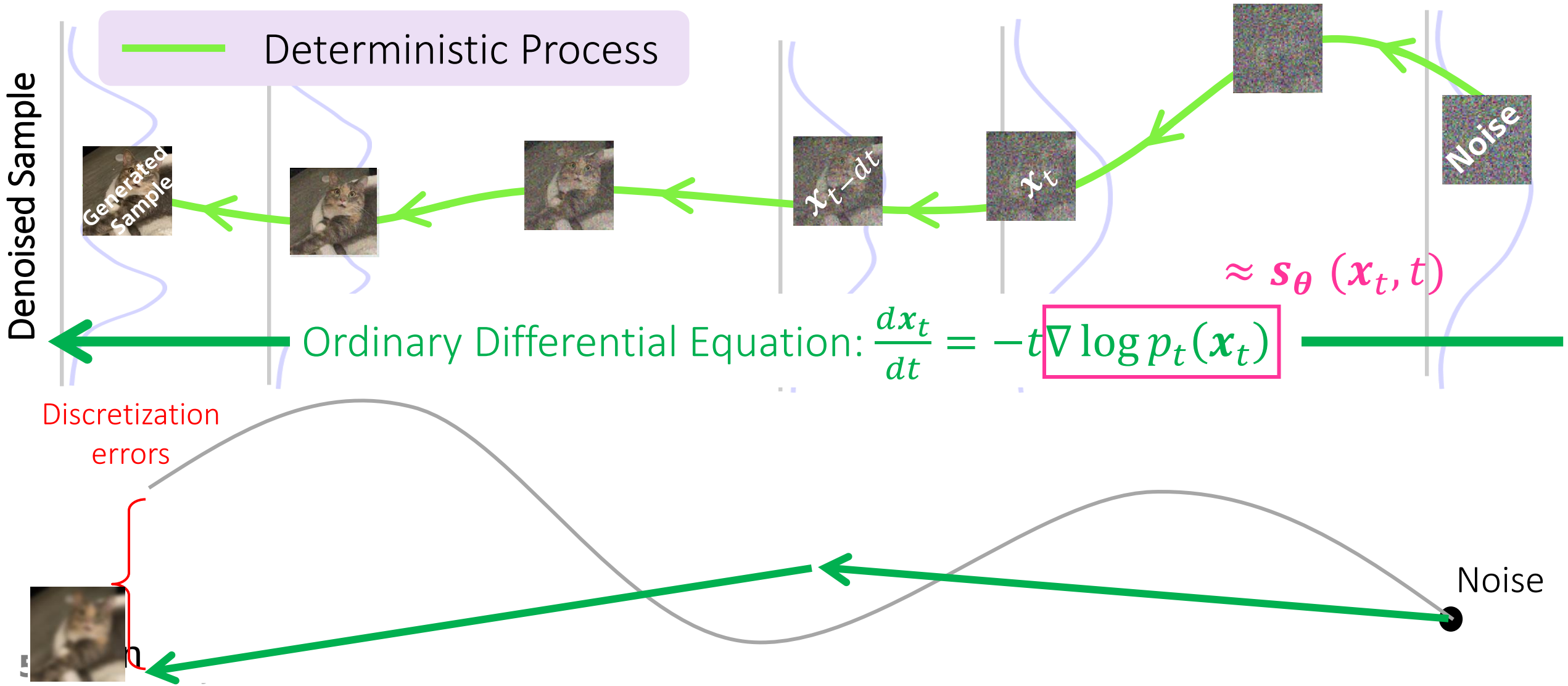
Score-Based Diffusion Model's Training and Sampling

Users predefined noise schedulers: $\boldsymbol{x}_t = a_t \boldsymbol{x}_0 + b_t \boldsymbol{\epsilon} : a_t = 1 \;\&\; b_t = t$



Score-Based Diffusion Model's Training and Sampling

Users predefined noise schedulers: $\boldsymbol{x_t} = a_t \boldsymbol{x_0} + b_t \boldsymbol{\epsilon} : a_t = 1 \;\&\; b_t = t$



Score-Based Diffusion Model's Training and Sampling

Users predefined noise schedulers: $\boldsymbol{x}_t = a_t \boldsymbol{x}_0 + b_t \boldsymbol{\epsilon} : a_t = 1 \;\&\; b_t = t$

Deterministic Process

Trade-off Between Generation Speed and Quality

$\approx \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$

Ordinary Differential Equation: $\frac{d\boldsymbol{x}_t}{dt} = -t \nabla \log p_t(\boldsymbol{x}_t)$

Discretization errors

Noise

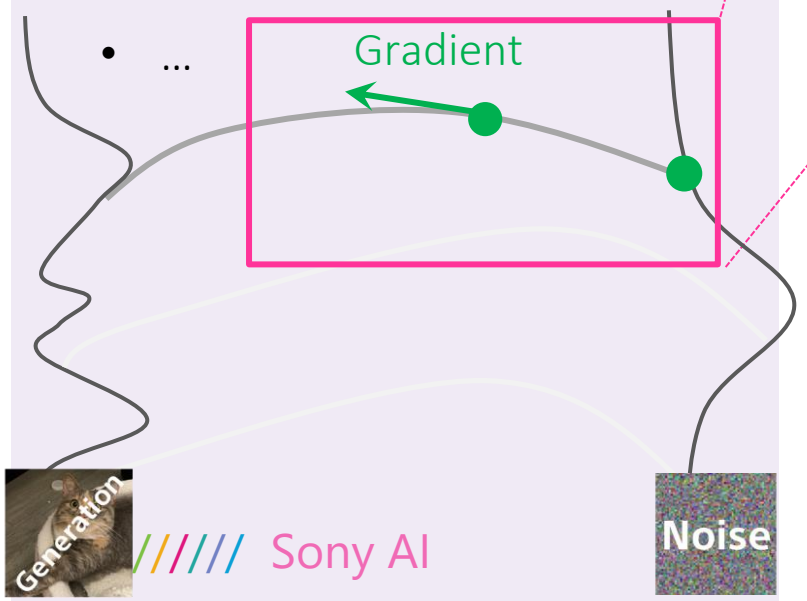
Fast Sampling in Diffusion Model

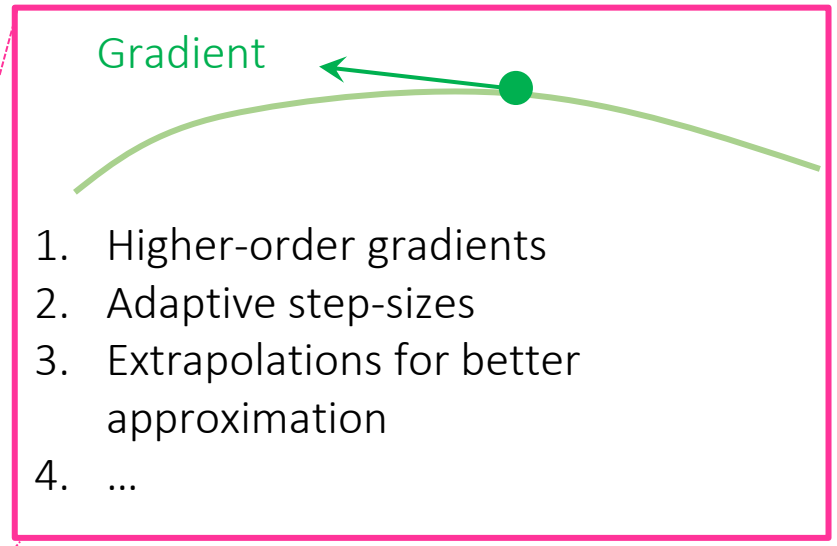
PF-ODE (2022)

Training-free

Sophisticated solvers (2021-2022+)

- [ICLR'21] DDIM
- [NeurIPS'22] DPM; DPM++
- [ICLR'23] DEIS
- …





1. Higher-order gradients
2. Adaptive step-sizes
3. Extrapolations for better approximation
4. …

////// Sony AI

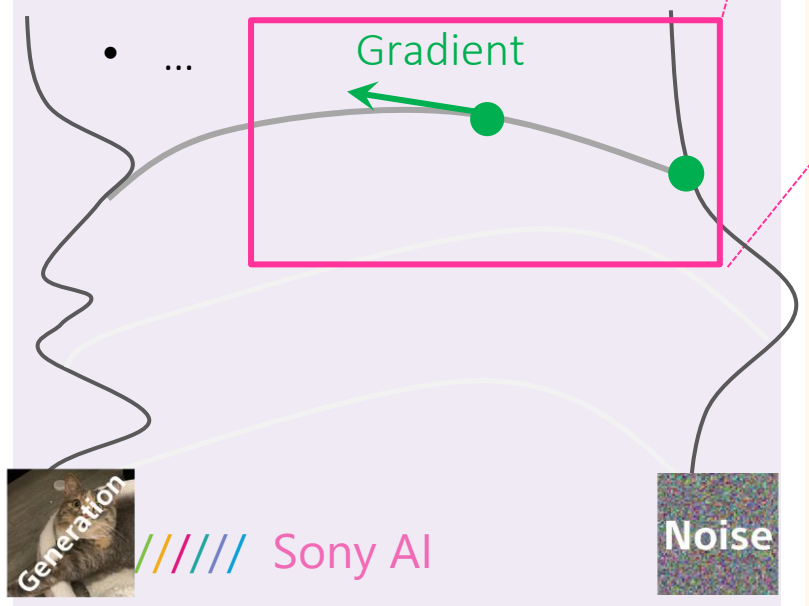
Fast Sampling in Diffusion Model

PF-ODE (2022)

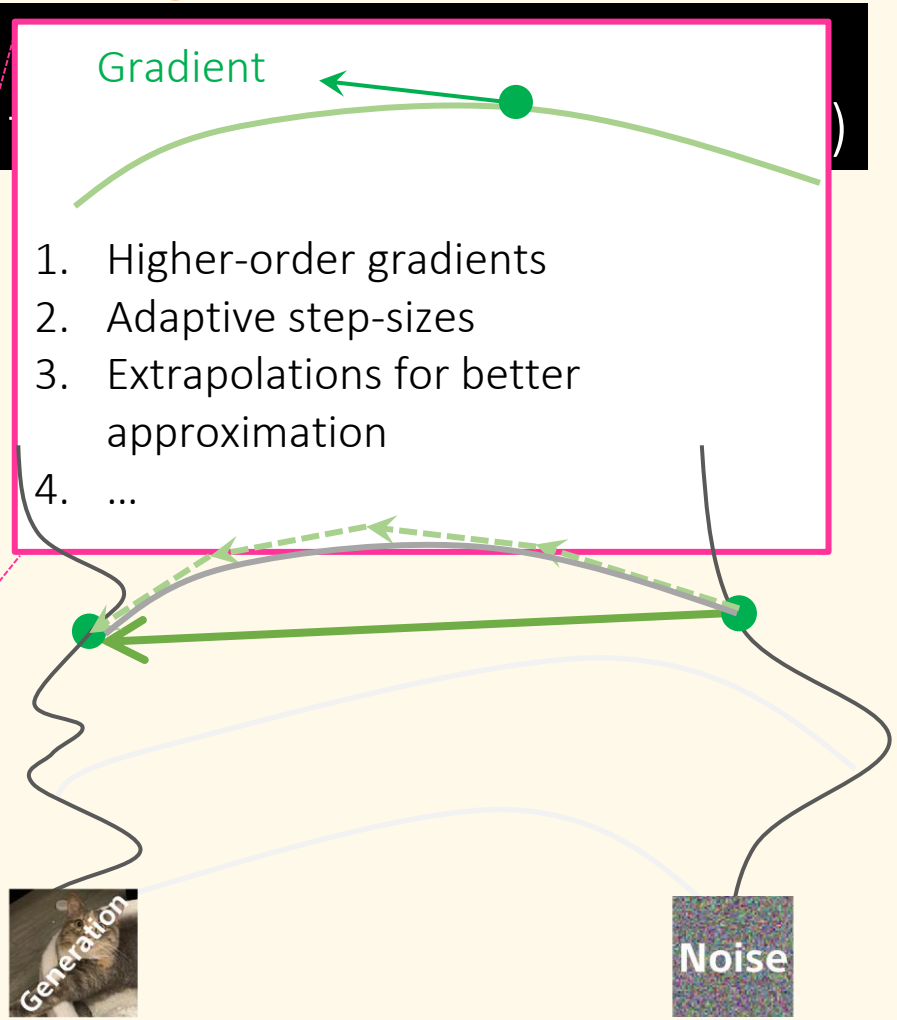
Training-free

Sophisticated solvers (2021-2022+)

- [ICLR'21] DDIM
- [NeurIPS'22] DPM; DPM++
- [ICLR'23] DEIS
- …

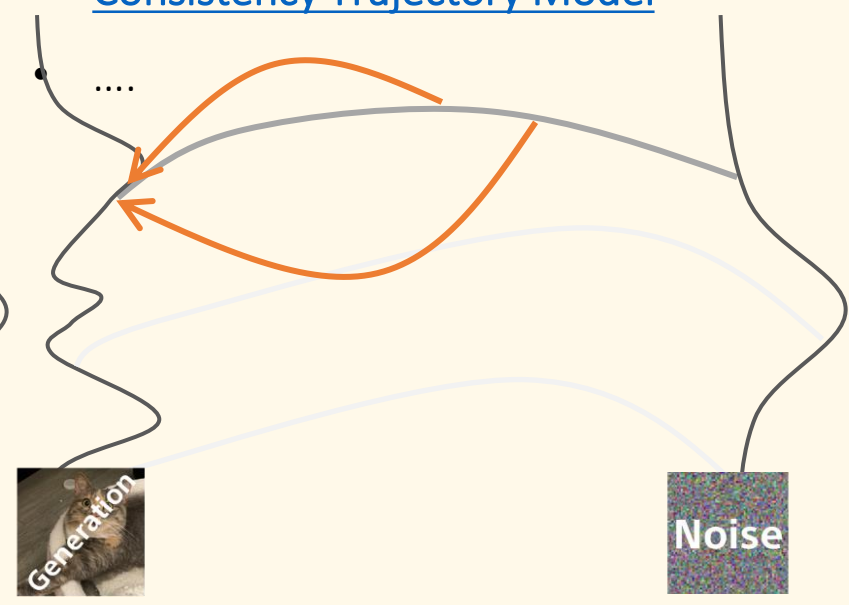


Training-based

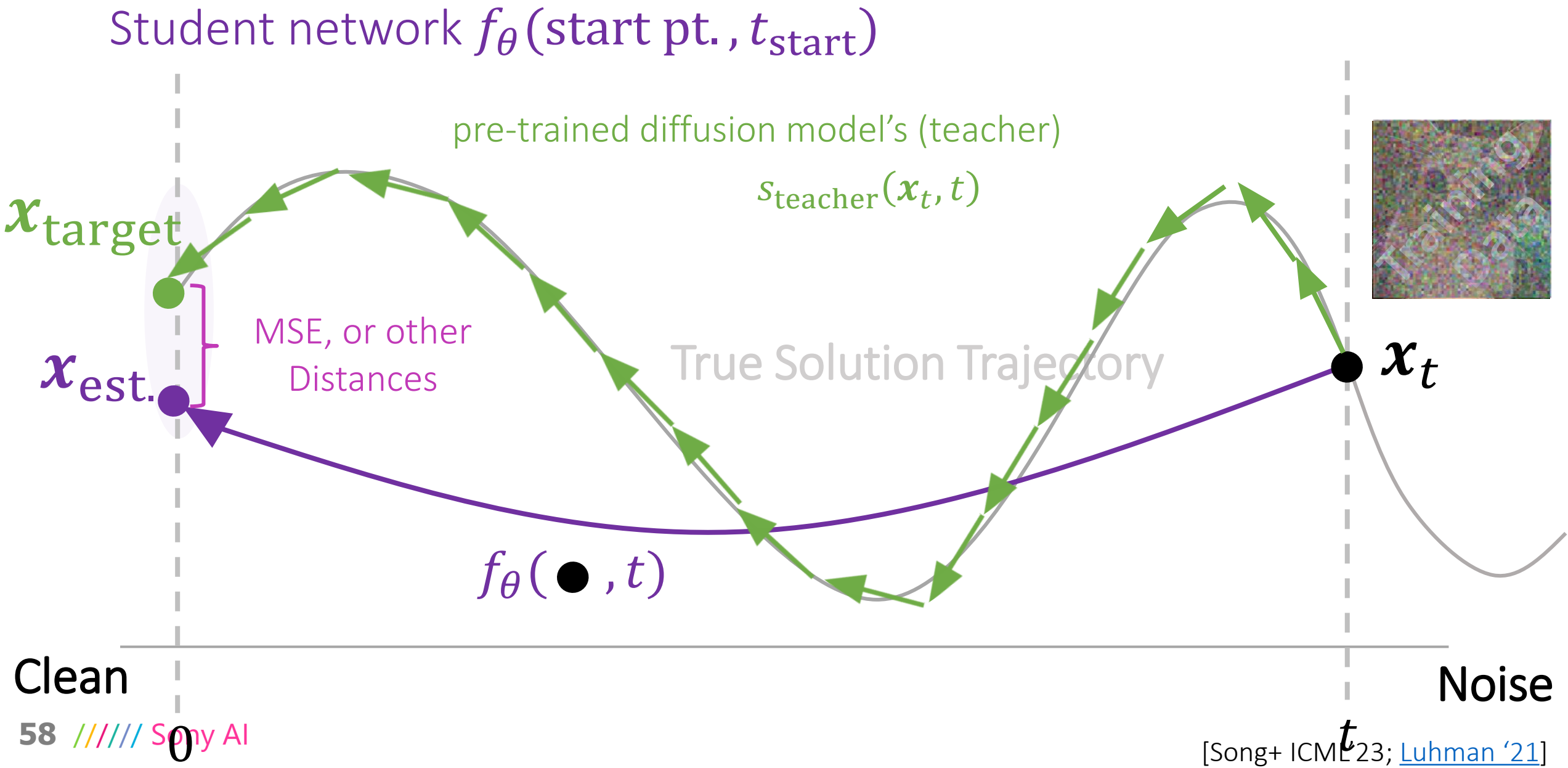


Distillation from DM (2023-)

- [ICML'23 Song+] Consistency Model
- [ICLR'24 Kim&Lai+] Consistency Trajectory Model
- ….



Distillation-Based Method's Training



[Song+ ICML'23; Luhman '21]

Distillation-Based Method's Training

Student network $f_\theta(\text{start pt.}, t_{\text{start}})$

pre-trained diffusion model's (teacher)

$s_{\text{teacher}}(x_t, t)$

x_{target}

$x_{\text{est.}}$

MSE, or other Distances

True Solution Trajectory

x_t

It learns any-to-0 jump on the trajectory

Clean

Noise

0

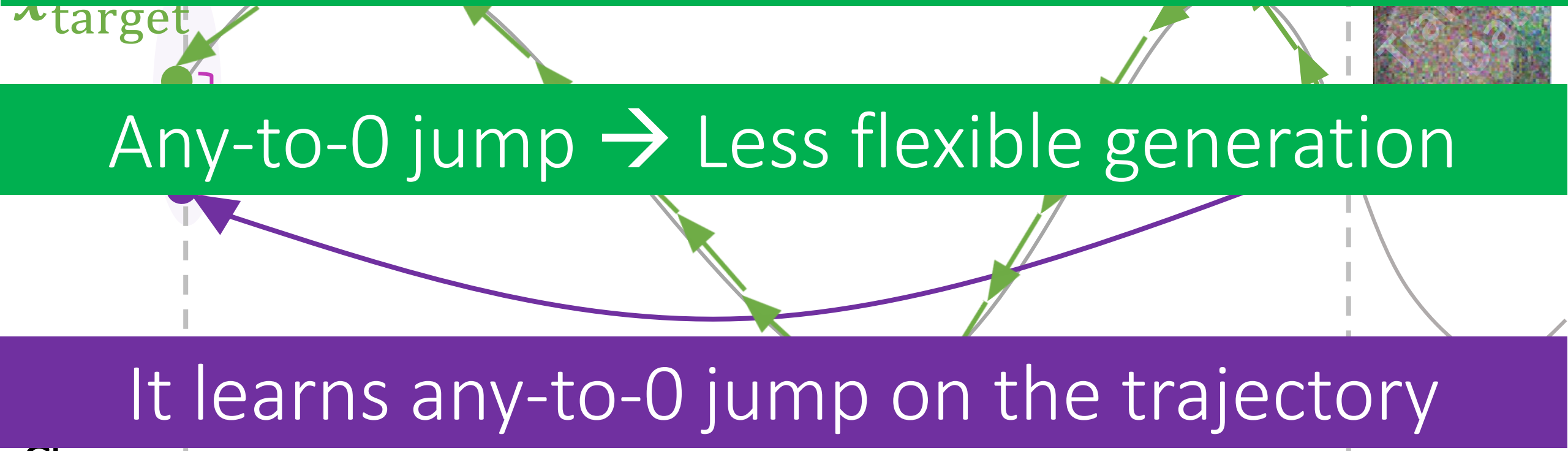
t

[Song+ ICML'23; Luhman '21]

Distillation-Based Method's Training

Student network $f_\theta(\text{start pt.}, t_{\text{start}})$

Solving ODE in Training is Expensive!



Any-to-0 jump → Less flexible generation

It learns any-to-0 jump on the trajectory

Clean

Noise

0

t

[Song+ ICML'23; Luhman '21]

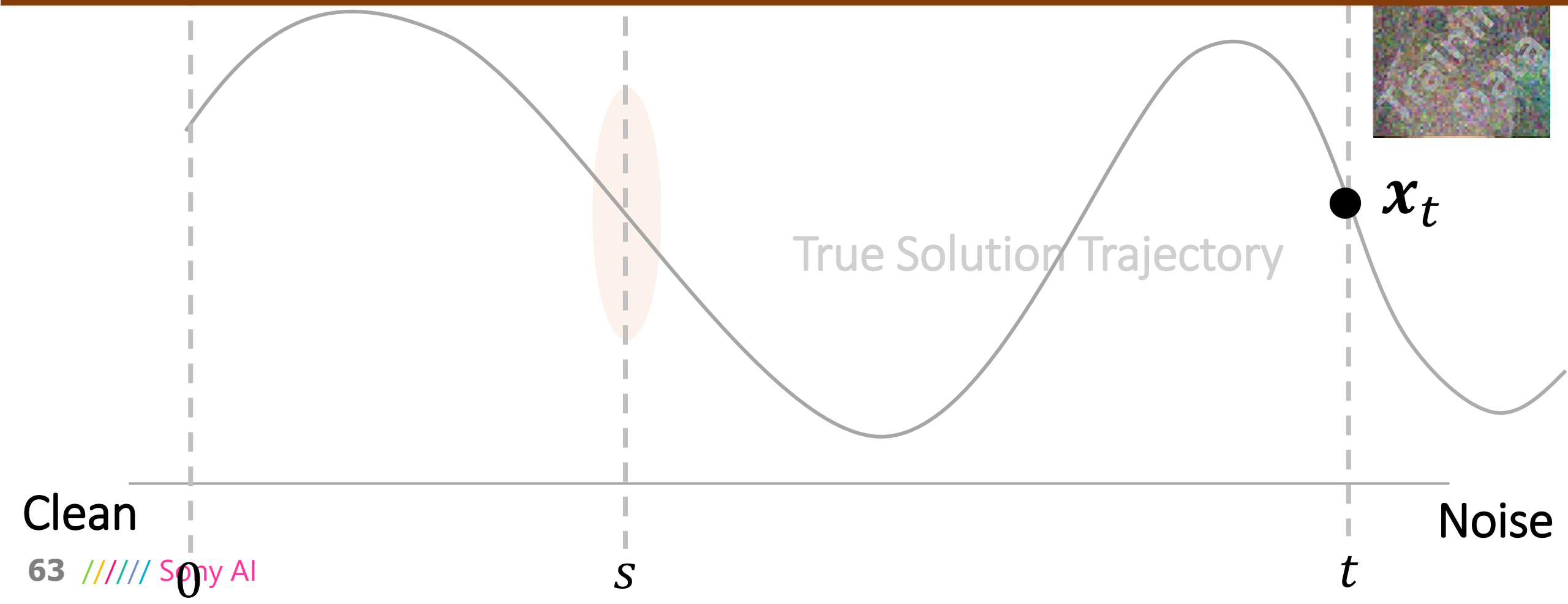
CONSISTENCY TRAJECTORY MODEL (CTM)

- ✓Efficient Training and Sampling
- ✓Unifying Diffusion Model and Distillation-Based Method for *Controllable Generation*

CTM’s Training and Mechanism

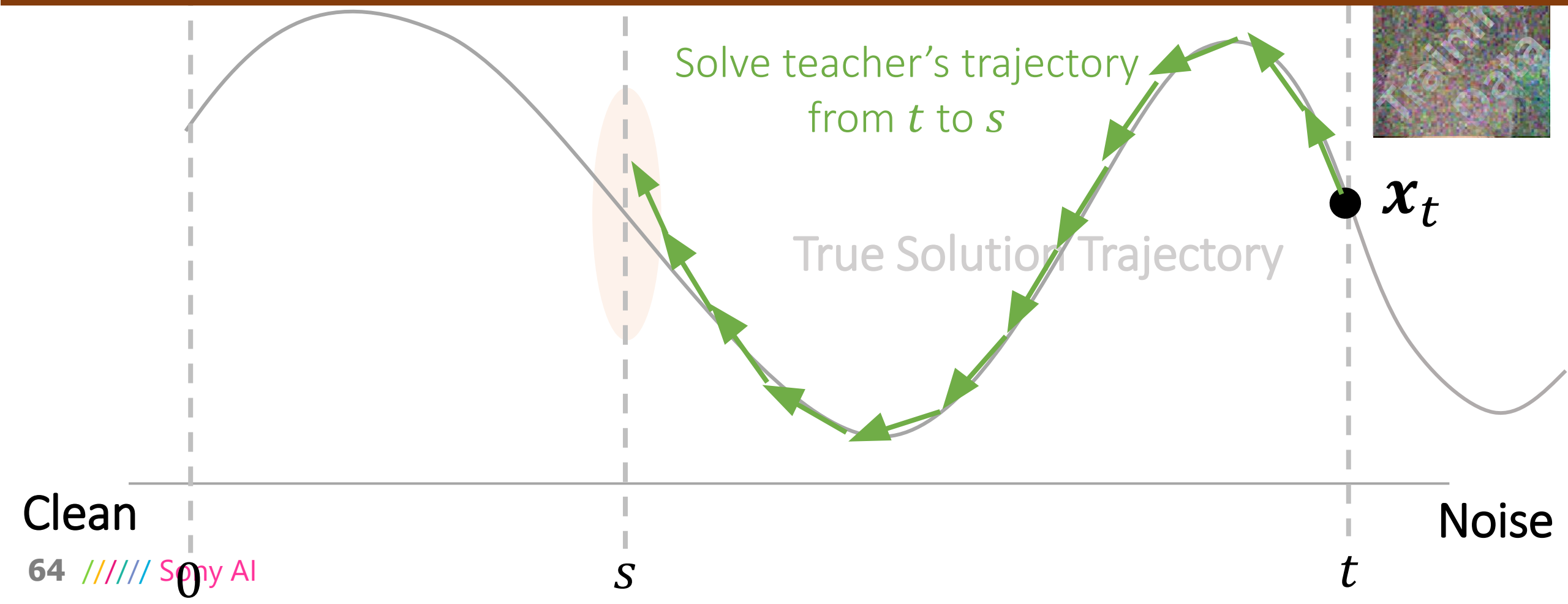
Consistency Trajectory Model (CTM)'s Training

$$\boldsymbol{x}_t - \int_t^s \tau \nabla \log p_\tau(\boldsymbol{x}_\tau) d\tau$$



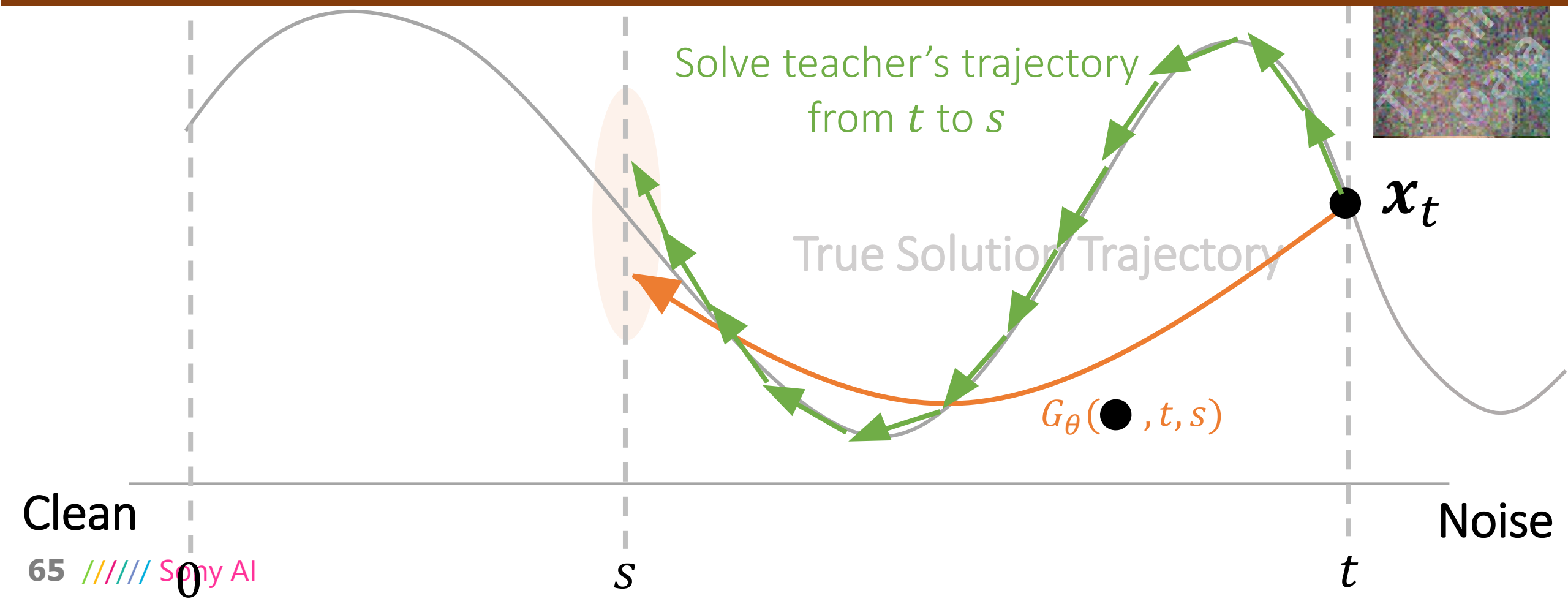
Consistency Trajectory Model (CTM)'s Training

$$x_t - \int_t^s \tau s_{\text{teacher}}(x_\tau, \tau) d\tau$$



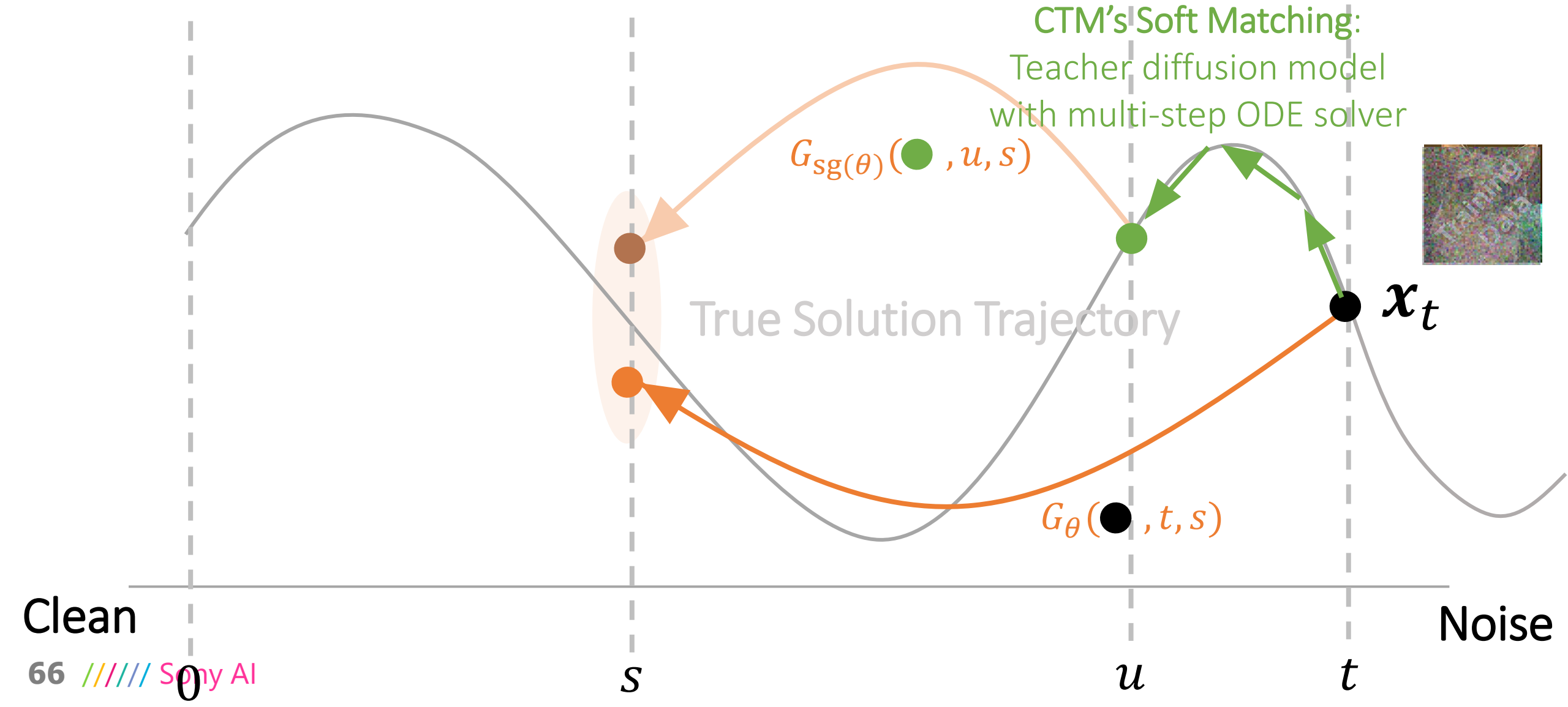
Consistency Trajectory Model (CTM)'s Training

$$G_\theta(\boldsymbol{x}_t, t, s) \approx \boldsymbol{x}_t - \int_t^s \tau \boldsymbol{s}_{\text{teacher}}(\boldsymbol{x}_\tau, \tau) d\tau$$



Consistency Trajectory Model (CTM)'s Training

Student network (CTM): $G_\theta(\text{start pt.}, t_{\text{start}}, t_{\text{end}})$



Consistency Trajectory Model (CTM)'s Training

Student network (CTM): $G_\theta(\text{start pt.}, t_{\text{start}}, t_{\text{end}})$

CTM's Soft Matching:
Teacher diffusion model
with multi-step ODE solver

$G_{\text{sg}(\theta)}(\bullet, s, 0)$

$G_{\text{sg}(\theta)}(\bullet, u, s)$

$\boldsymbol{x}_{\text{target}}$

$\boldsymbol{x}_{\text{est.}}$

Any distances, or
even GAN

$G_{\text{sg}(\theta)}(\bullet, s, 0)$

True Solution Trajectory

$\boldsymbol{x}_t$

$G_\theta(\bullet, t, s)$

Clean

Noise

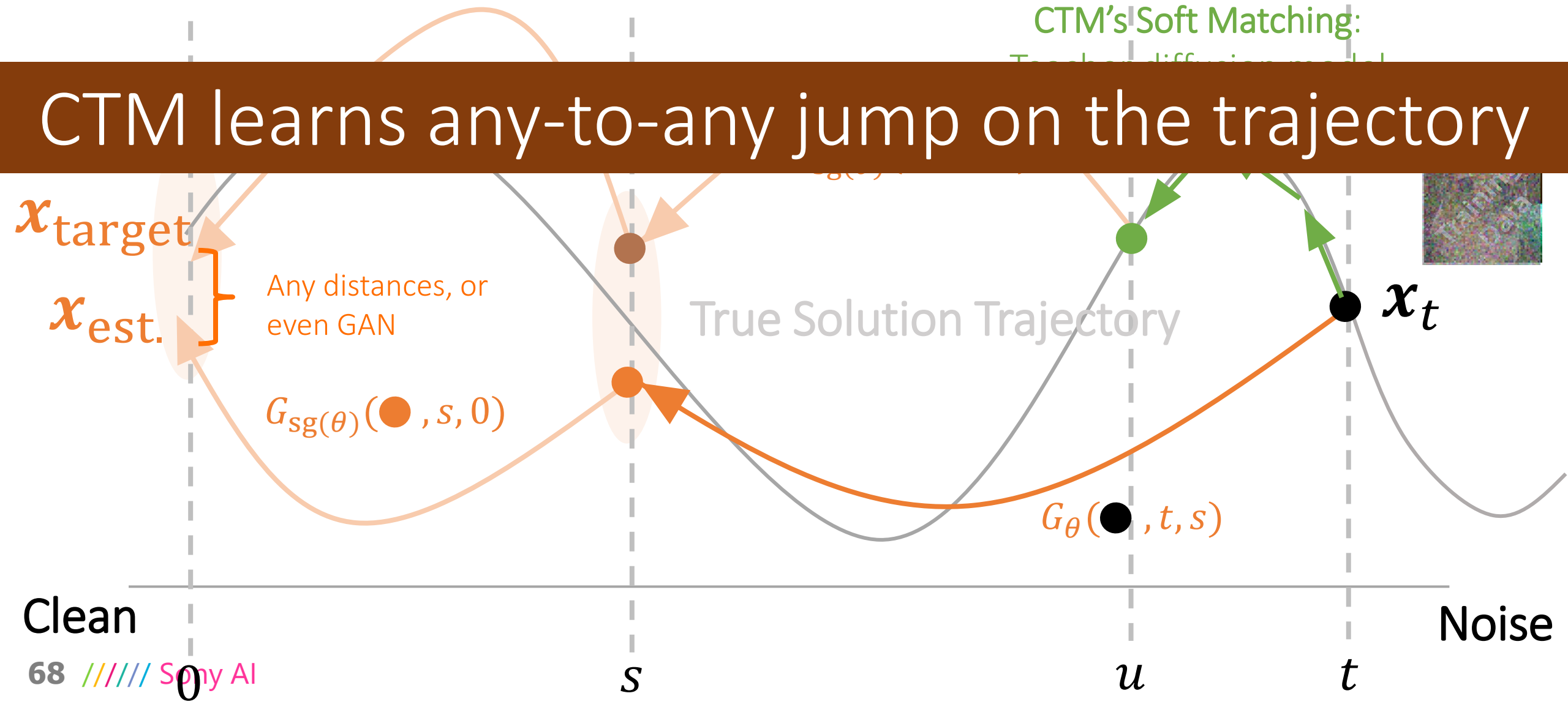
0 s u t

Consistency Trajectory Model (CTM)'s Training

Student network (CTM): $G_\theta(\text{start pt.}, t_{\text{start}}, t_{\text{end}})$

CTM's Soft Matching:

CTM learns any-to-any jump on the trajectory



Consistency Trajectory Model (CTM)'s Training

Student network (CTM): $G_\theta(\text{start pt.}, t_{\text{start}}, t_{\text{end}})$

CTM's Soft Matching:

CTM learns any-to-any jump on the trajectory

x_{target}

$x_{\text{est.}}$

Any distances, or even GAN

True Solution Trajectory

x_t

$\mathcal{L}_{\text{CTM}} + \mathcal{L}_{\text{DSM}}$

$G_\theta(\bullet, t, s)$

Clean

Noise

0 s u t

Consistency Trajectory Model (CTM)'s Training

Student network (CTM): $G_\theta(\text{start pt.}, t_{\text{start}}, t_{\text{end}})$

CTM's Soft Matching:

CTM learns any-to-any jump on the trajectory

x_{target}

$x_{\text{est.}}$

Any distances, or even GAN

True Solution Trajectory

x_t

$G_\theta(\bullet, t, s)$

$$d(x_{\text{target}}, x_{\text{est.}}) + d(G_\theta(x_t, t, s \approx t), x_{\text{data}})$$

Clean

Noise

0 s u t

CTM’s Flexible Sampling

CTM's Sampling Flexibility

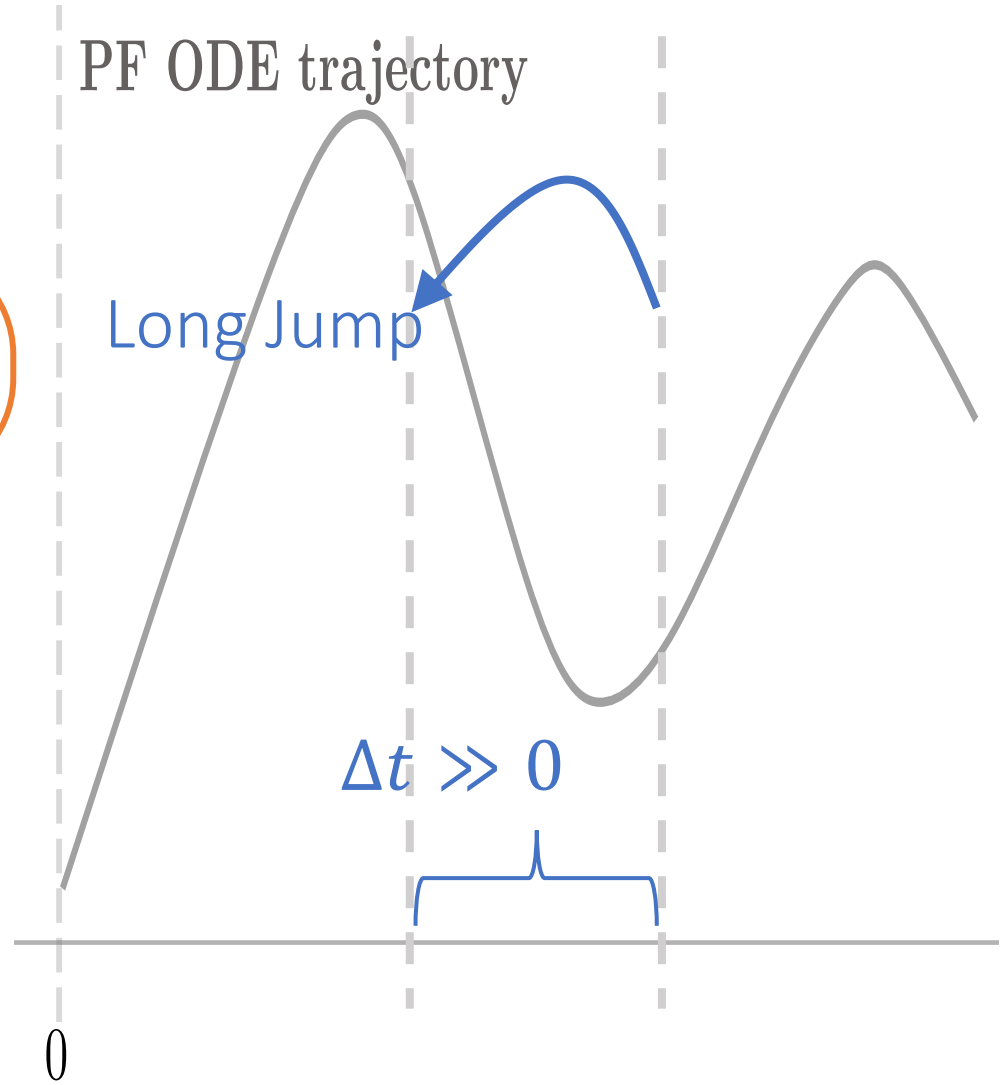
CTM enables both long "jumps" along the solution trajectory and score evaluation!

$$G_\theta(\boldsymbol{x}_t, t, t-\Delta t)$$

Long jump ($\Delta t \gg 0$)

CTM enables long jump along the trajectory

→ Allow our new sampling method



CTM's Sampling Flexibility

CTM enables both long "jumps" along the solution trajectory and score evaluation!

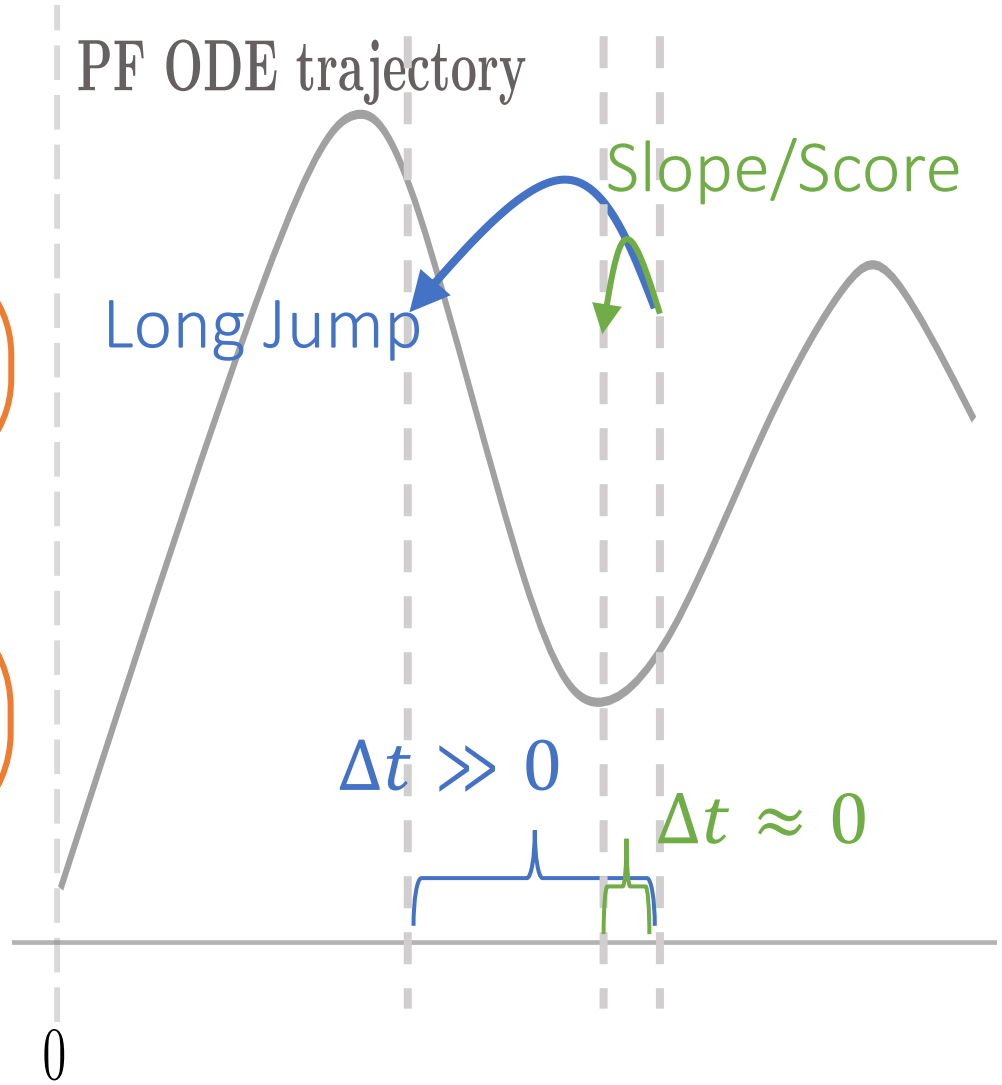
$$G_\theta(\boldsymbol{x}_t, t, t-\Delta t)$$

Long jump ($\Delta t \gg 0$)

CTM enables long jump along the trajectory
→ Allow our new sampling method

Slope (infinitesimal jump $\Delta t \approx 0$)

CTM enables score evaluations
→ Allow score-based sampling & likelihood computation



One-step Generation with CTM

Solution Trajectory

$$G_\theta(\ \text{Noise}\ , t_{\text{start}} = T,\ t_{\text{end}} = 0)$$

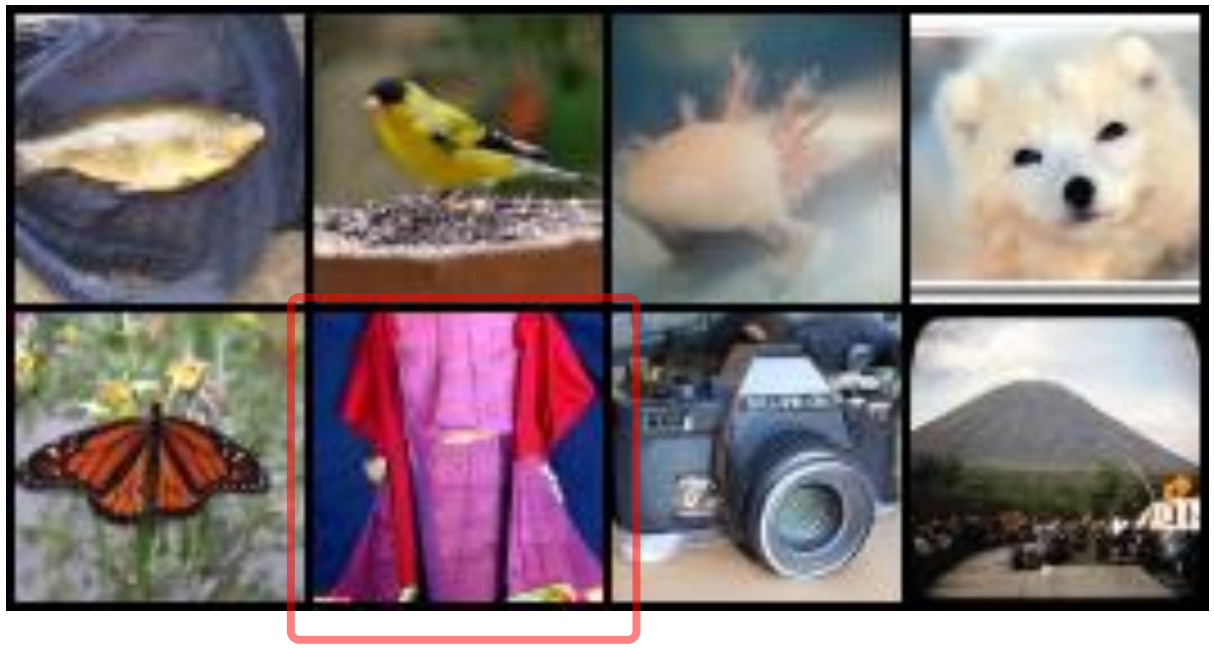
Clean

Noise

0

T

SOTA (beats DM) with One-Step



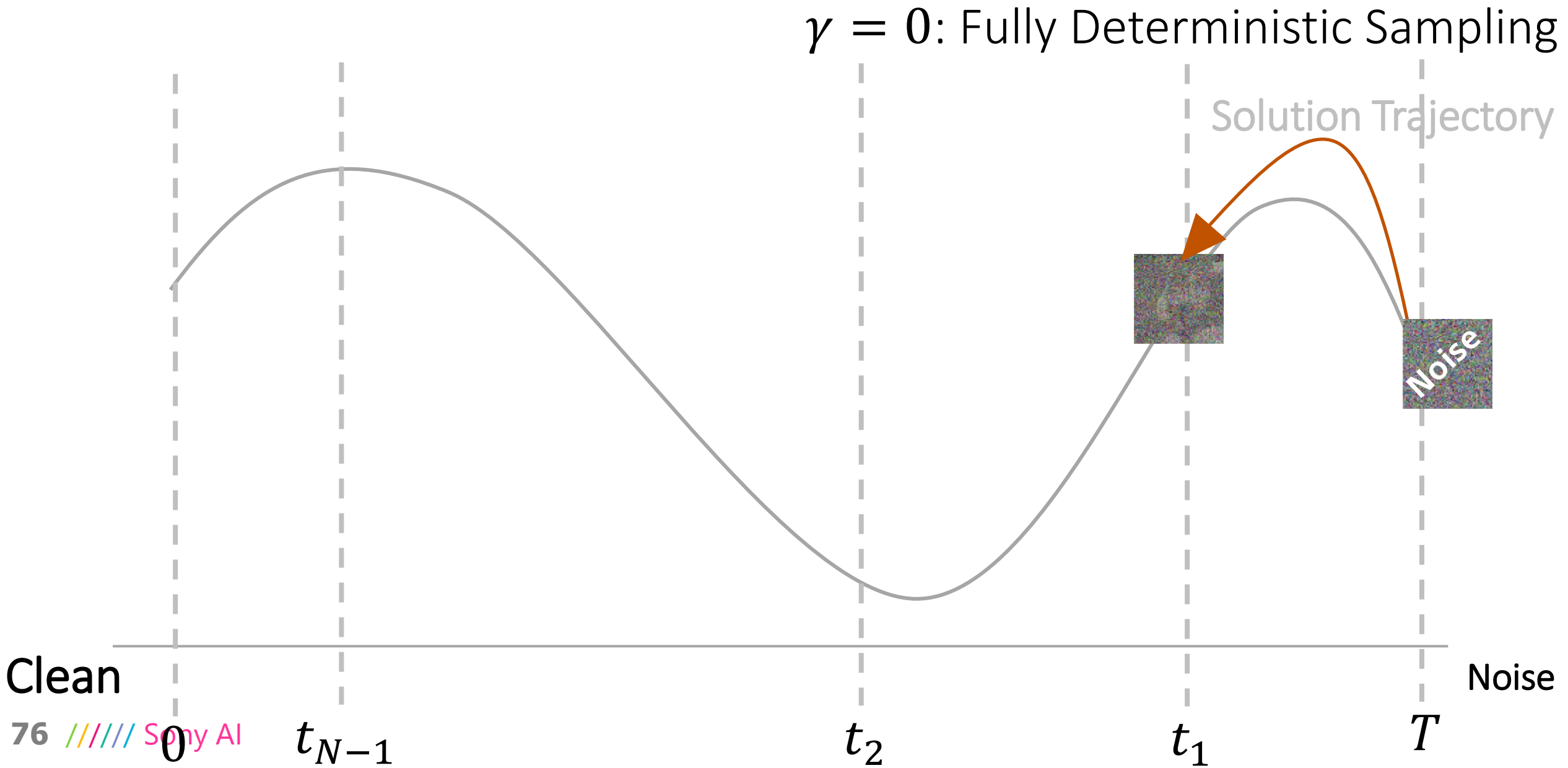
Teacher: Sampling steps = 79

Takes minutes

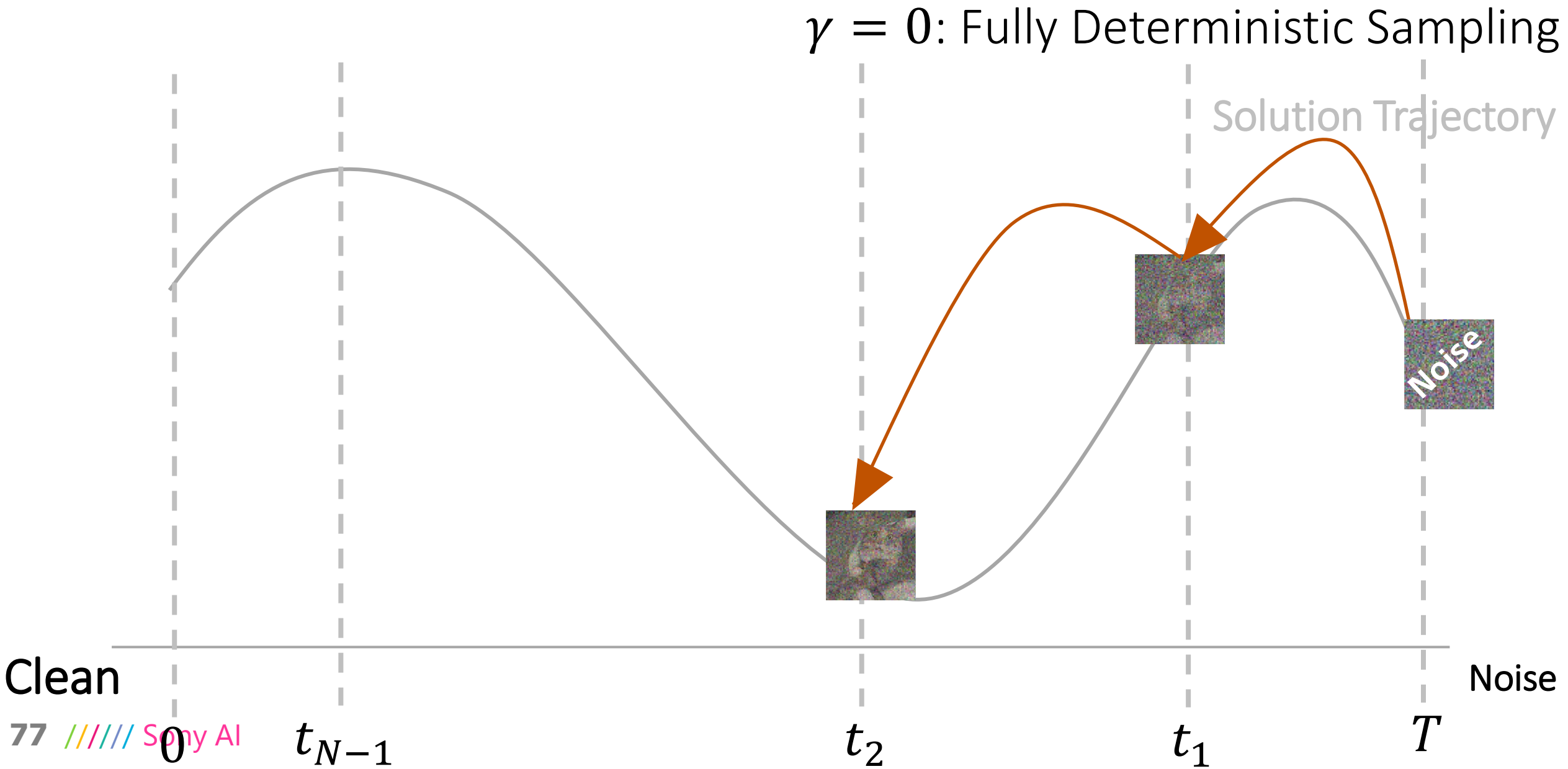
Student (CTM): Sampling step = 1

Within a second

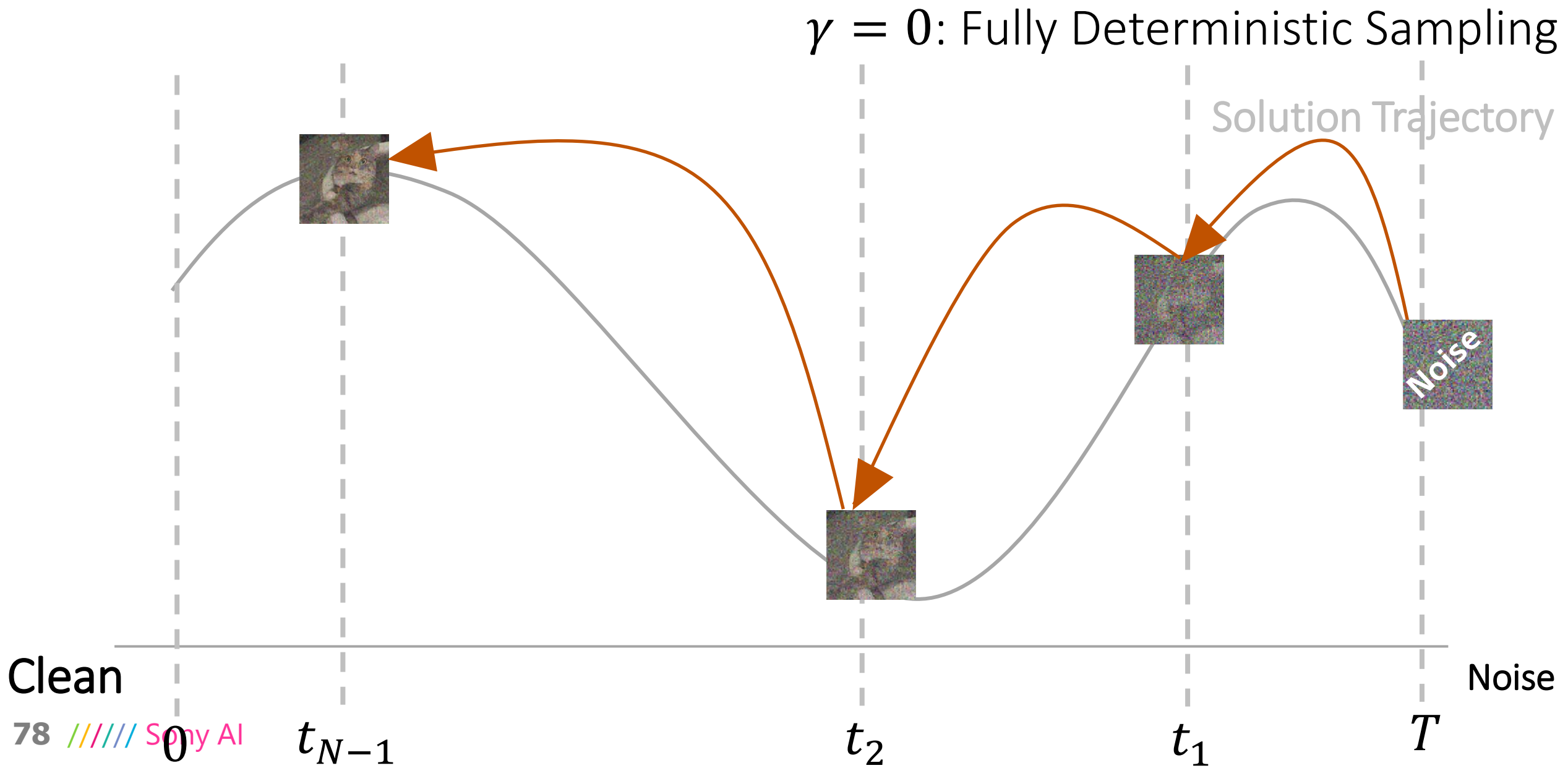
Multistep Generation with CTM: γ-sampling



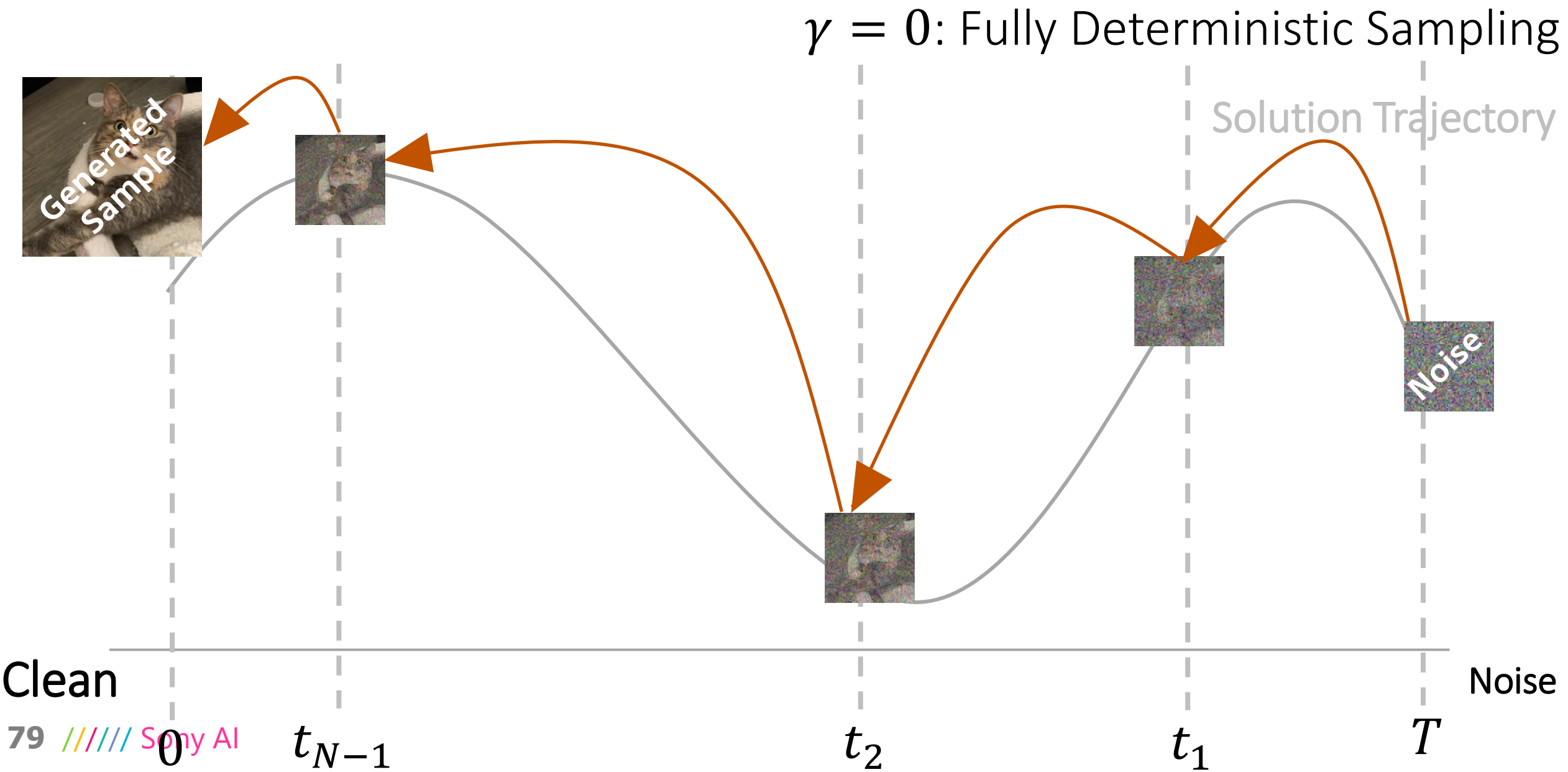
Multistep Generation with CTM: γ-sampling



Multistep Generation with CTM: γ-sampling

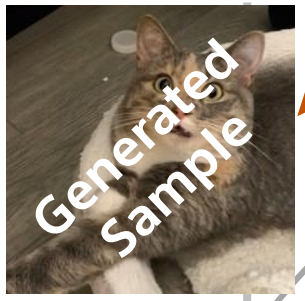


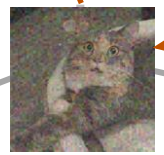
Multistep Generation with CTM: γ-sampling



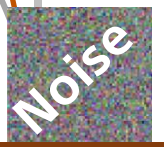
Multistep Generation with CTM: γ-sampling

$\gamma = 0$: Fully Deterministic Sampling





Solution Trajectory



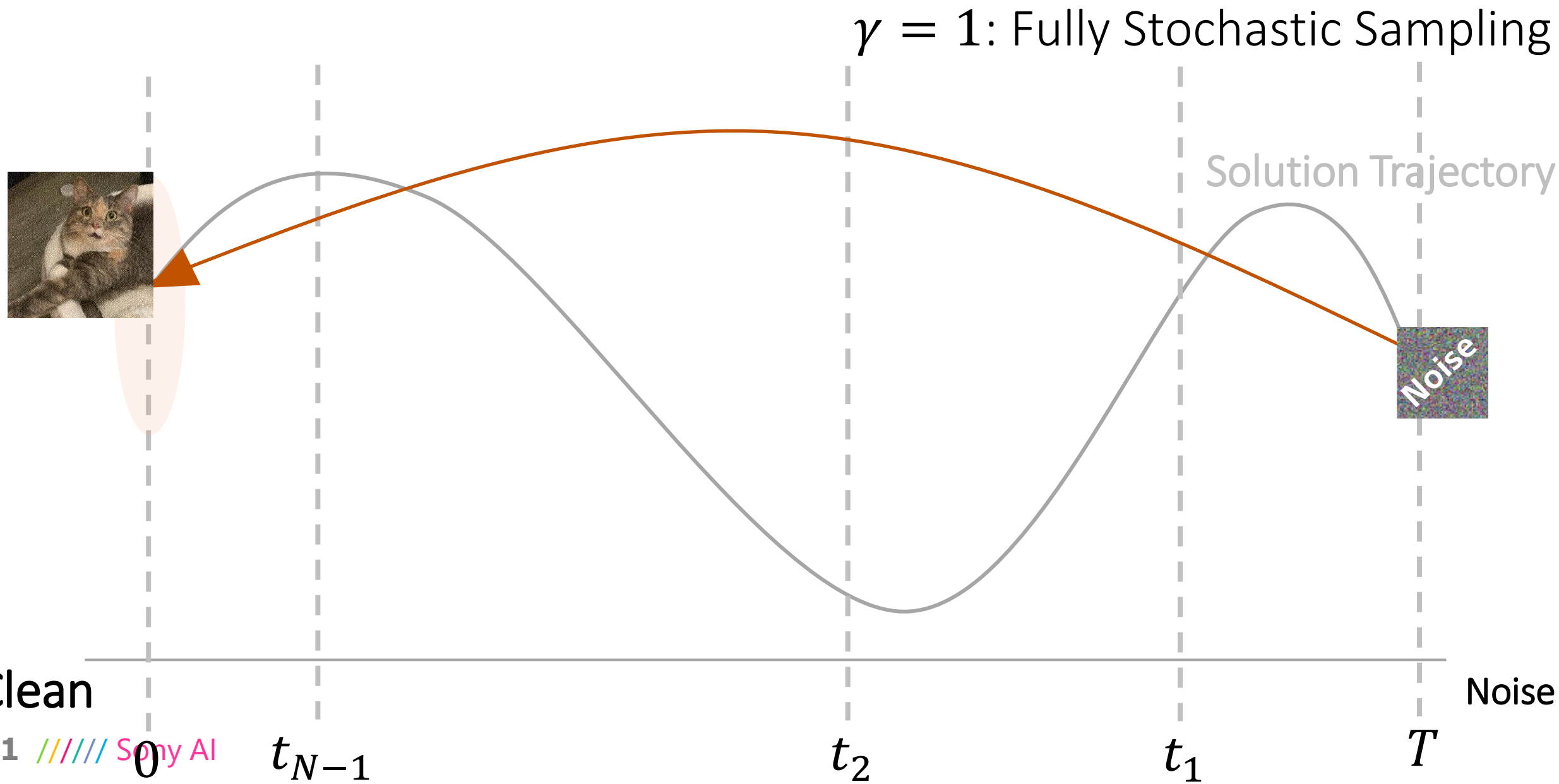
$\mathcal{O}(T)$

Clean

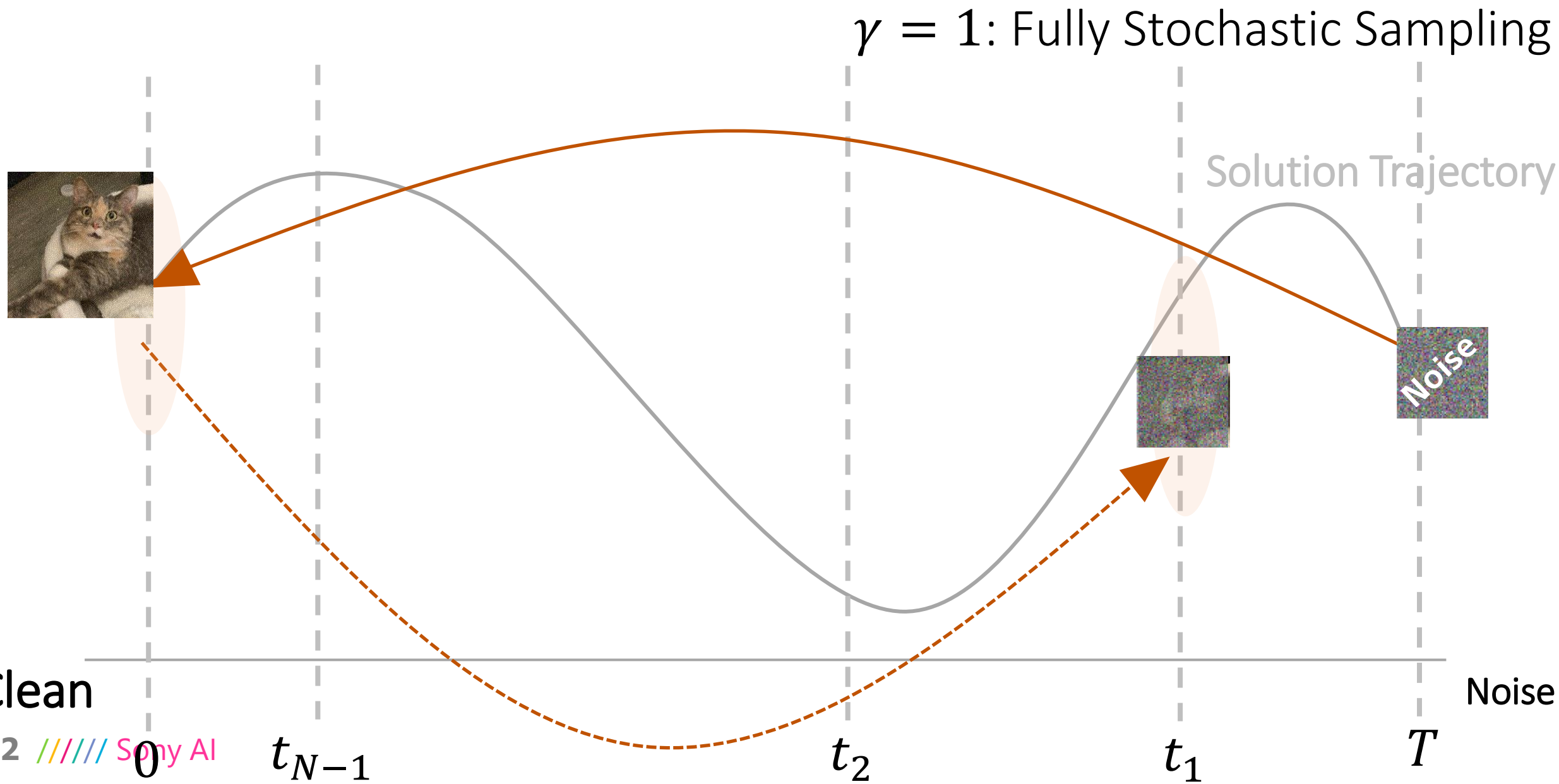
Noise

0 t_{N-1} t_2 t_1 T

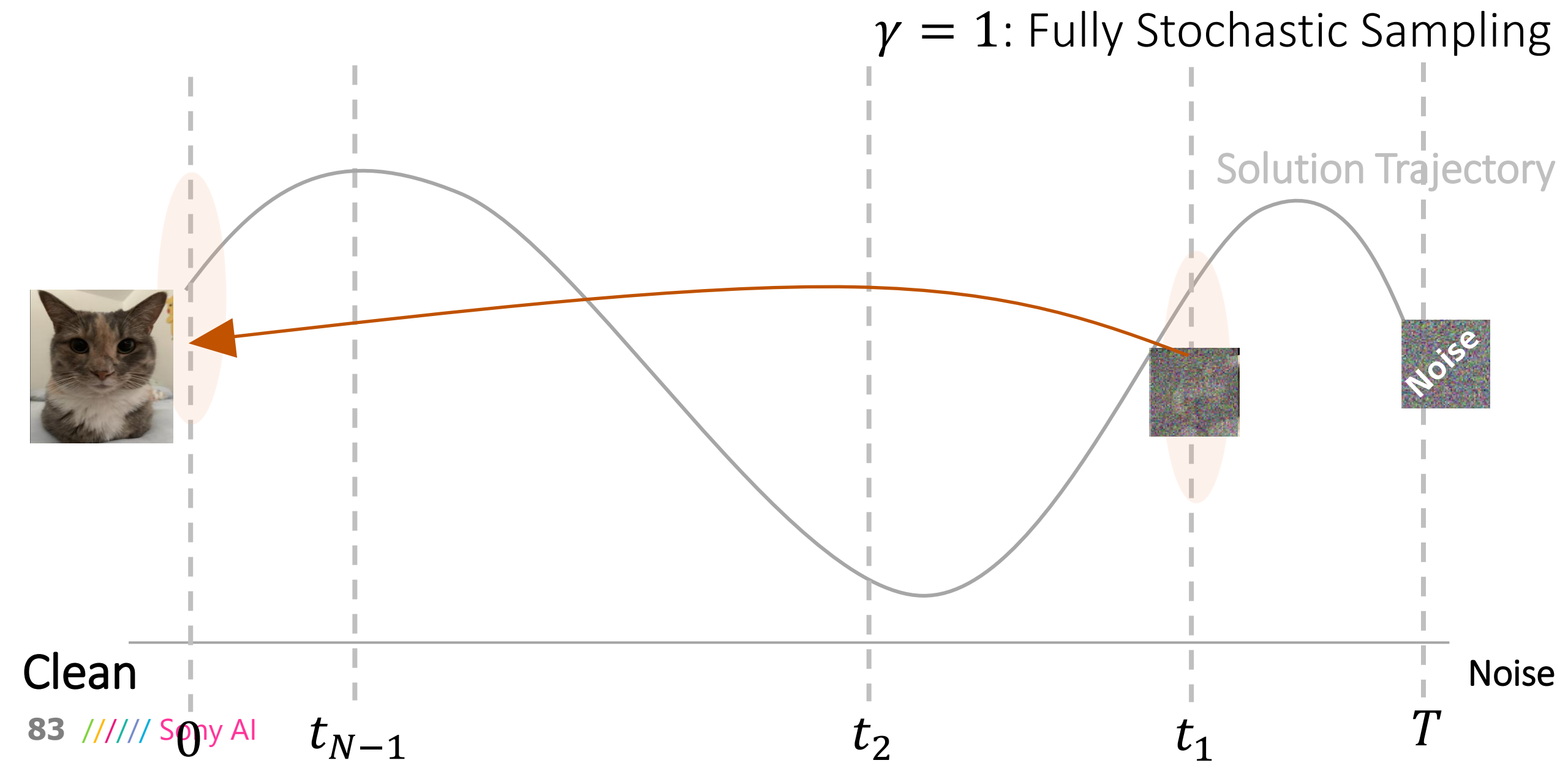
Multistep Generation with CTM: γ-sampling



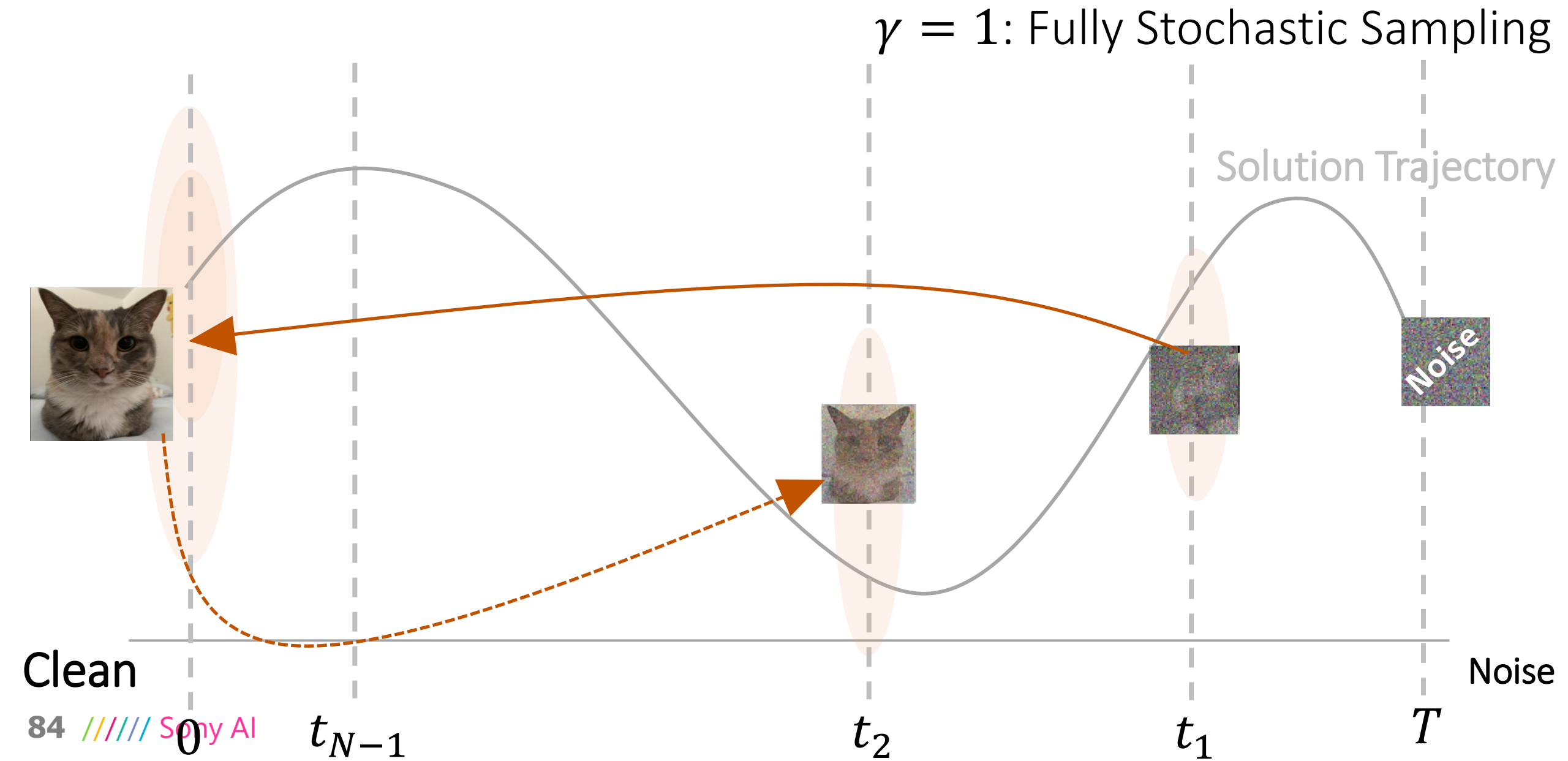
Multistep Generation with CTM: γ-sampling



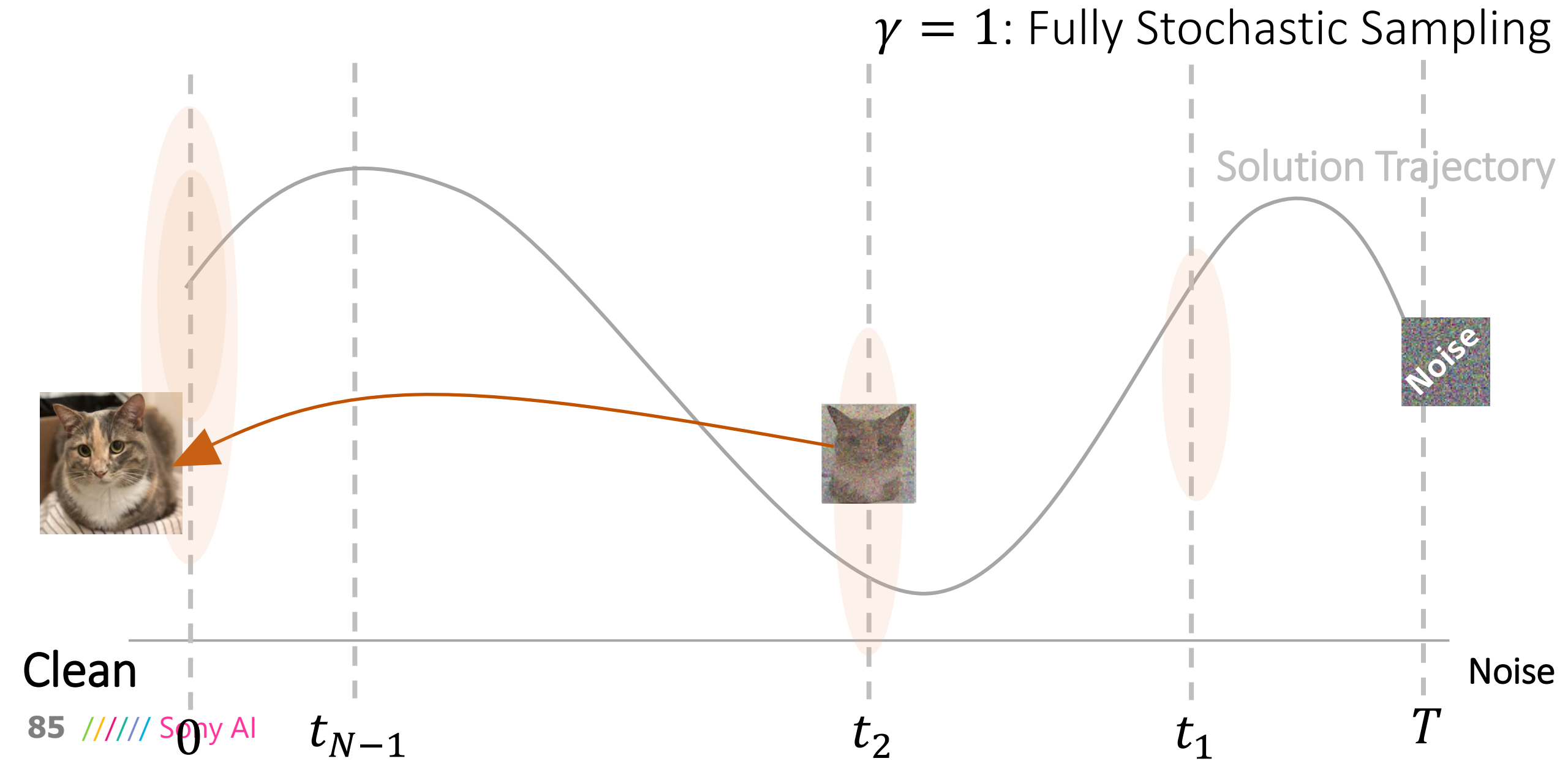
Multistep Generation with CTM: γ-sampling



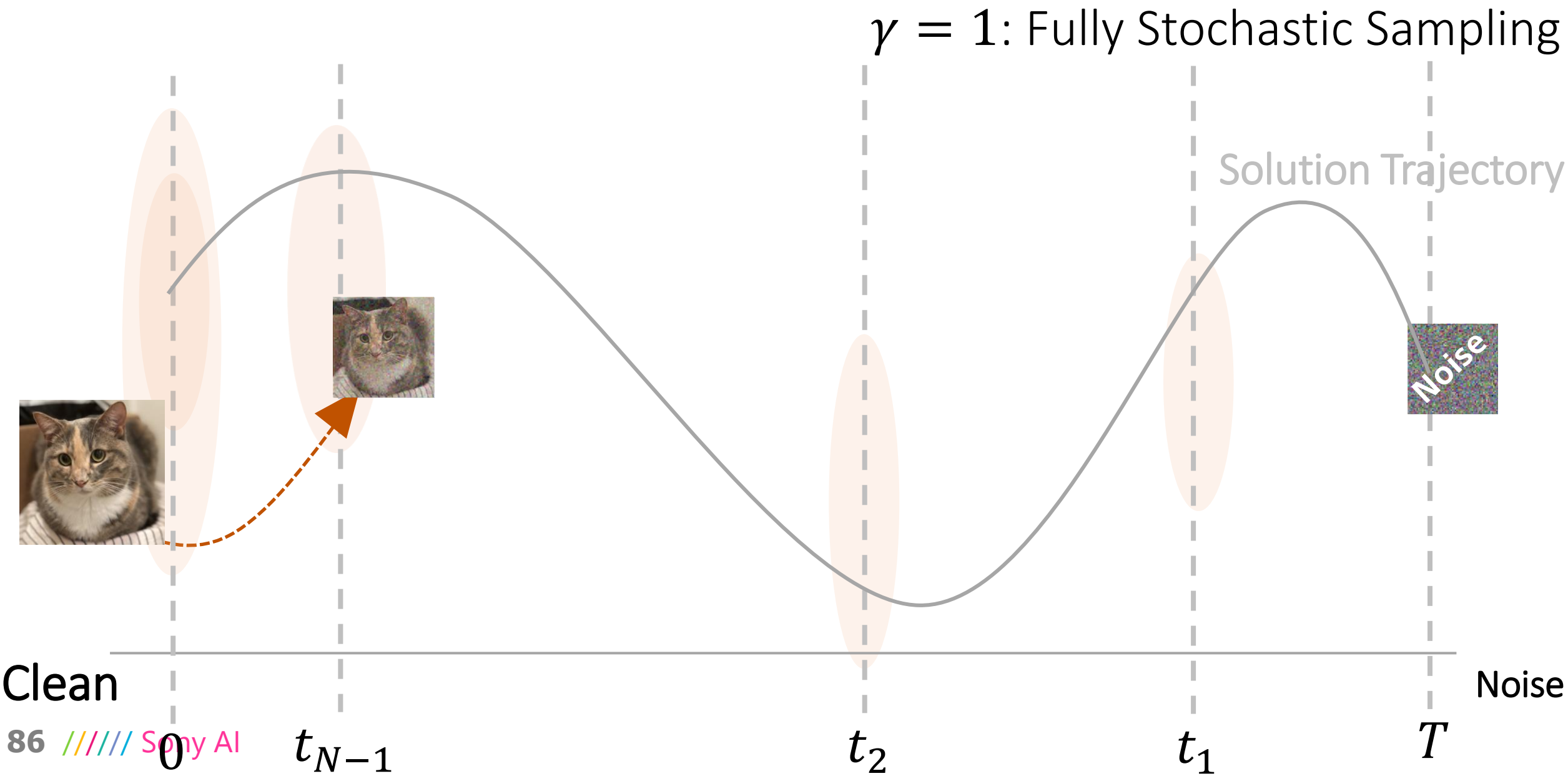
Multistep Generation with CTM: γ-sampling



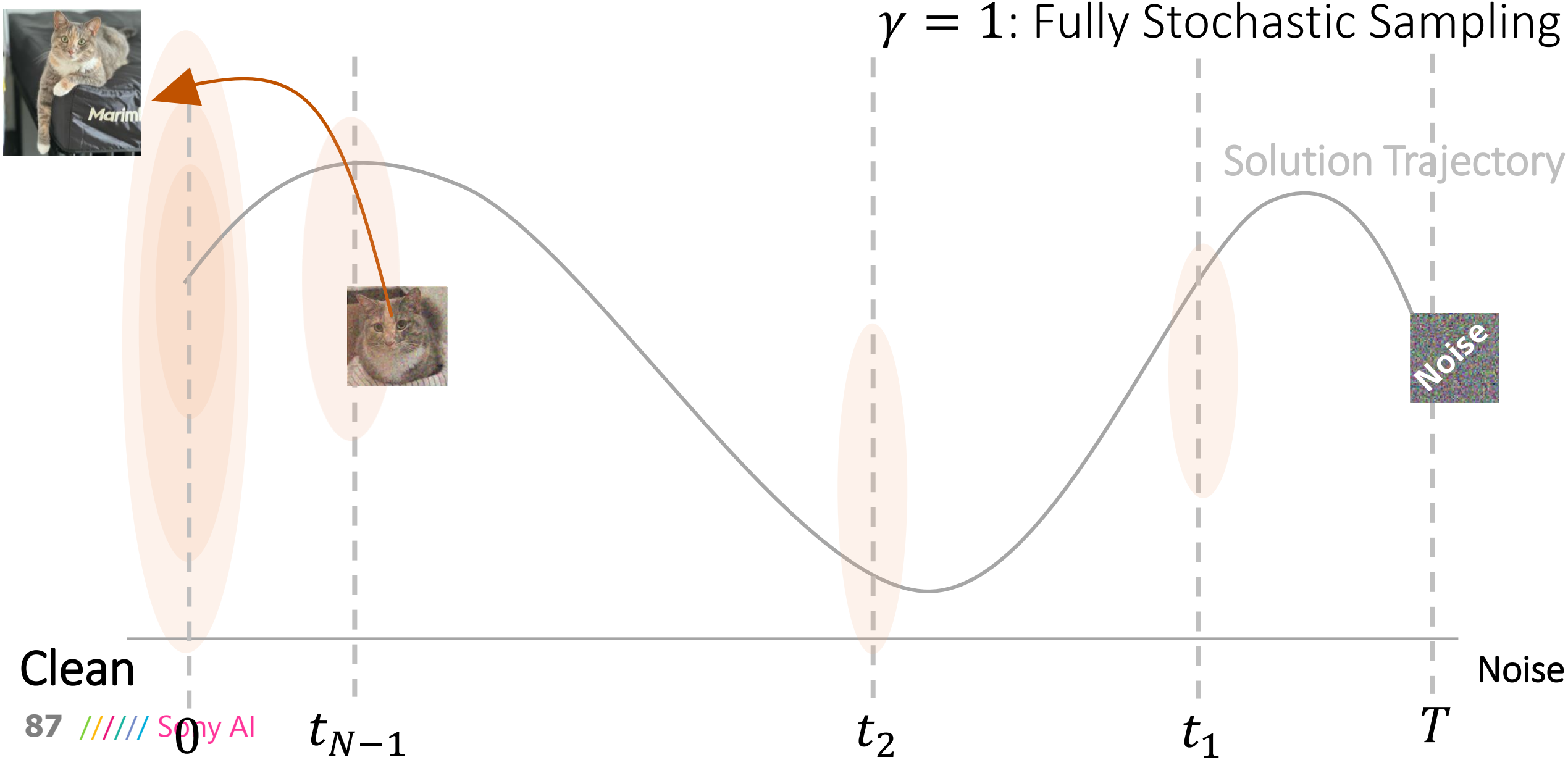
Multistep Generation with CTM: γ-sampling



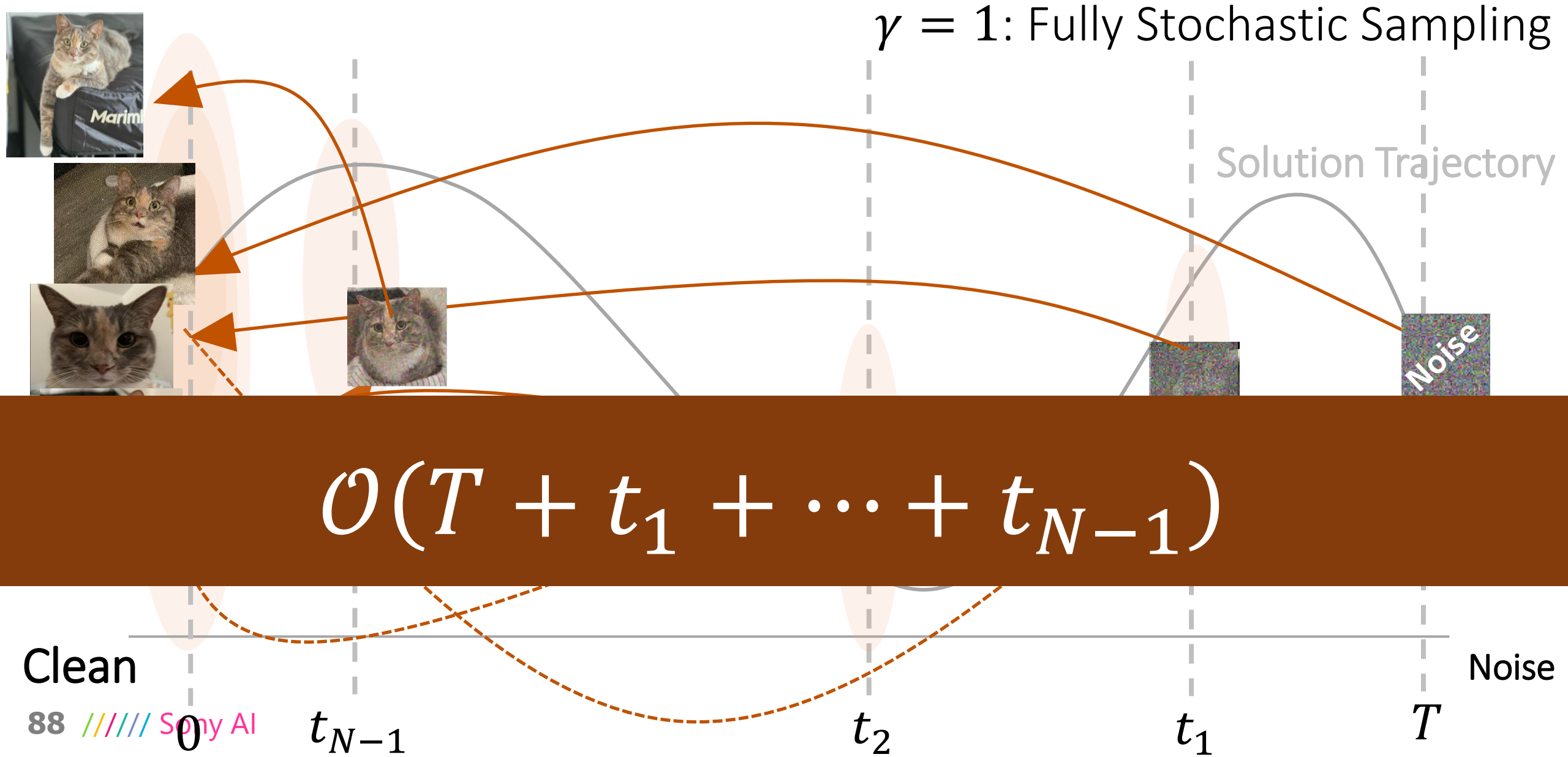
Multistep Generation with CTM: γ-sampling



Multistep Generation with CTM: γ-sampling

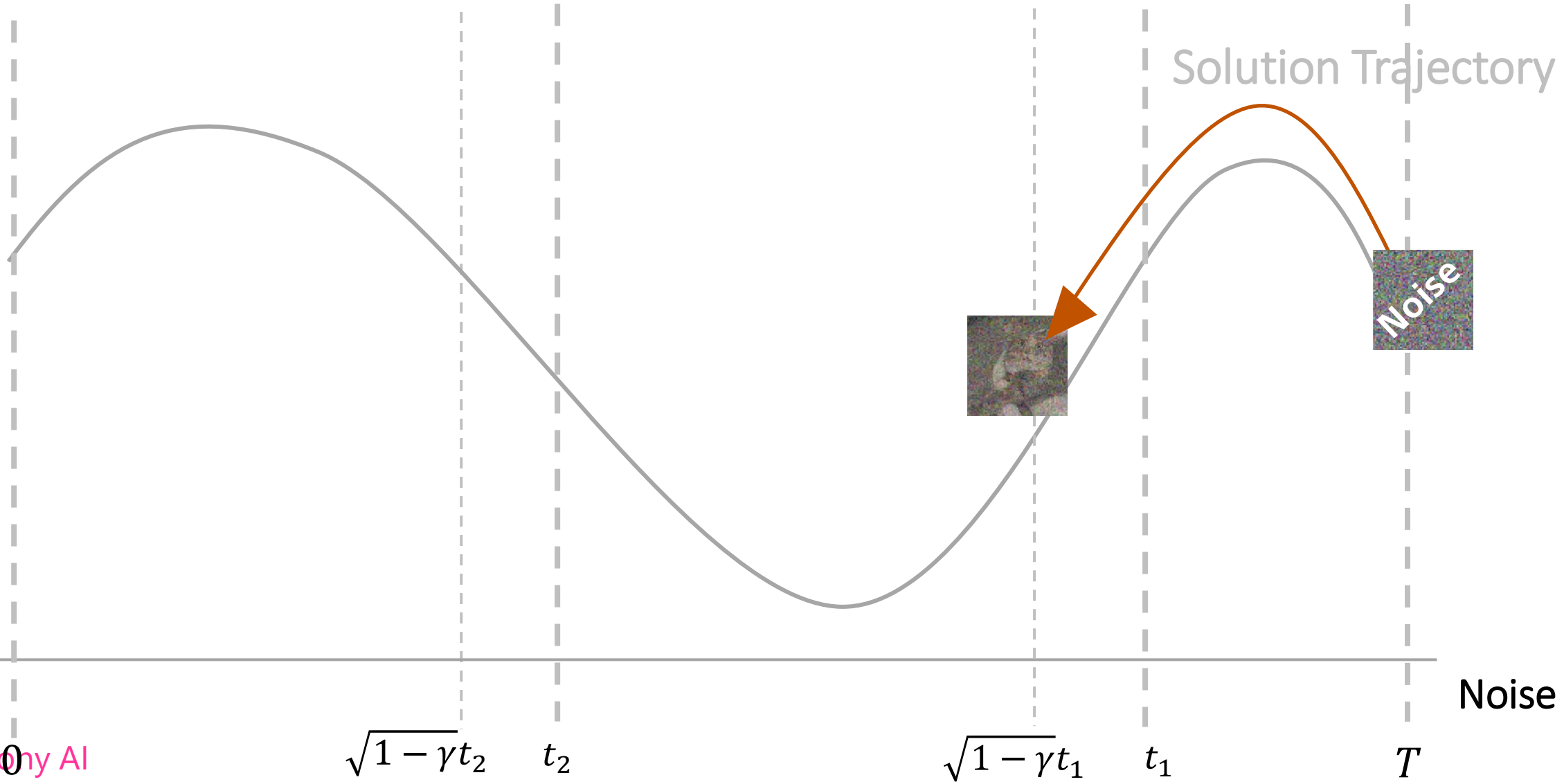


Multistep Generation with CTM: γ-sampling



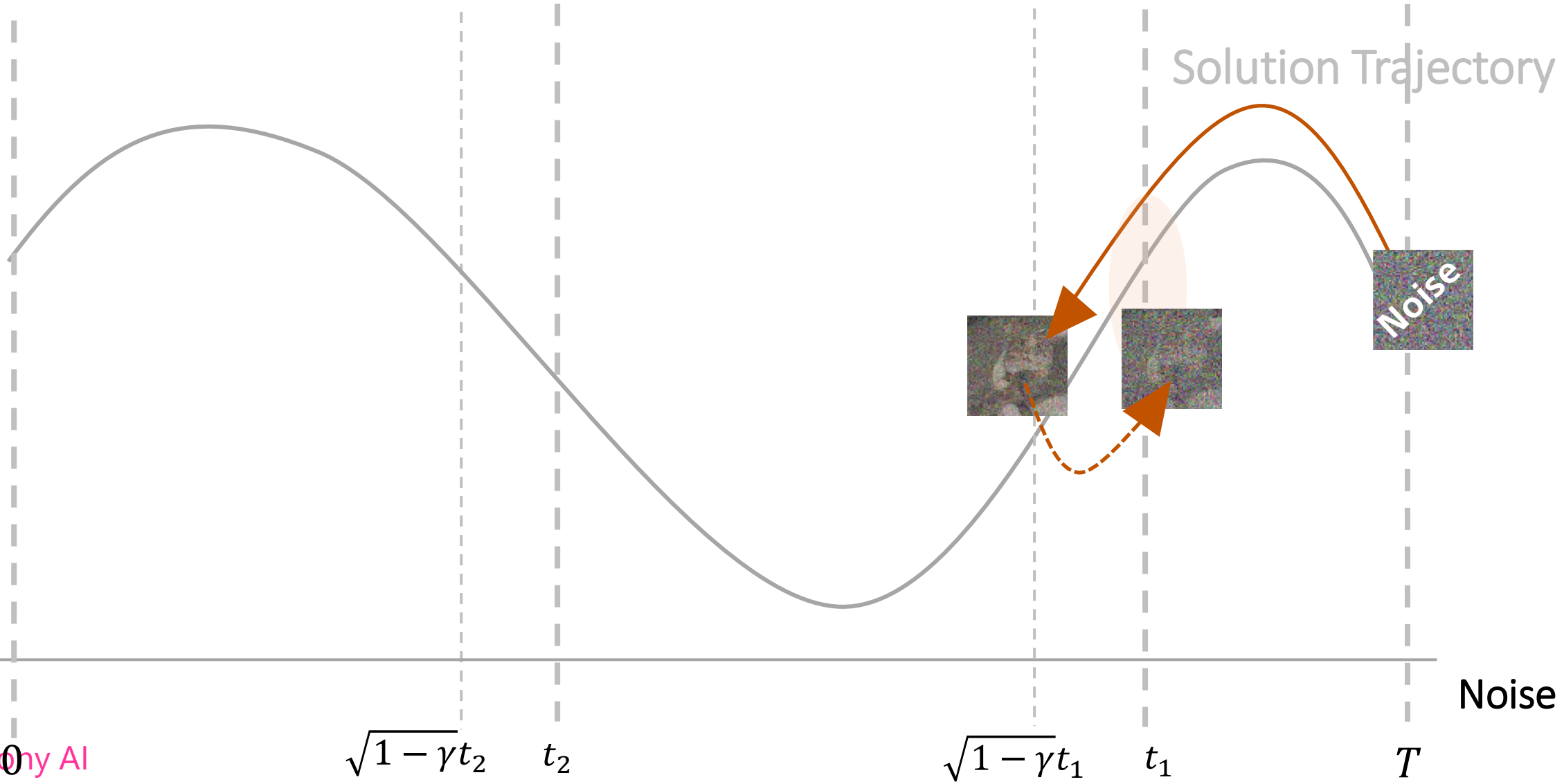
Multistep Generation with CTM: γ-sampling

$\gamma \in (0,1)$: Stochastic Sampling



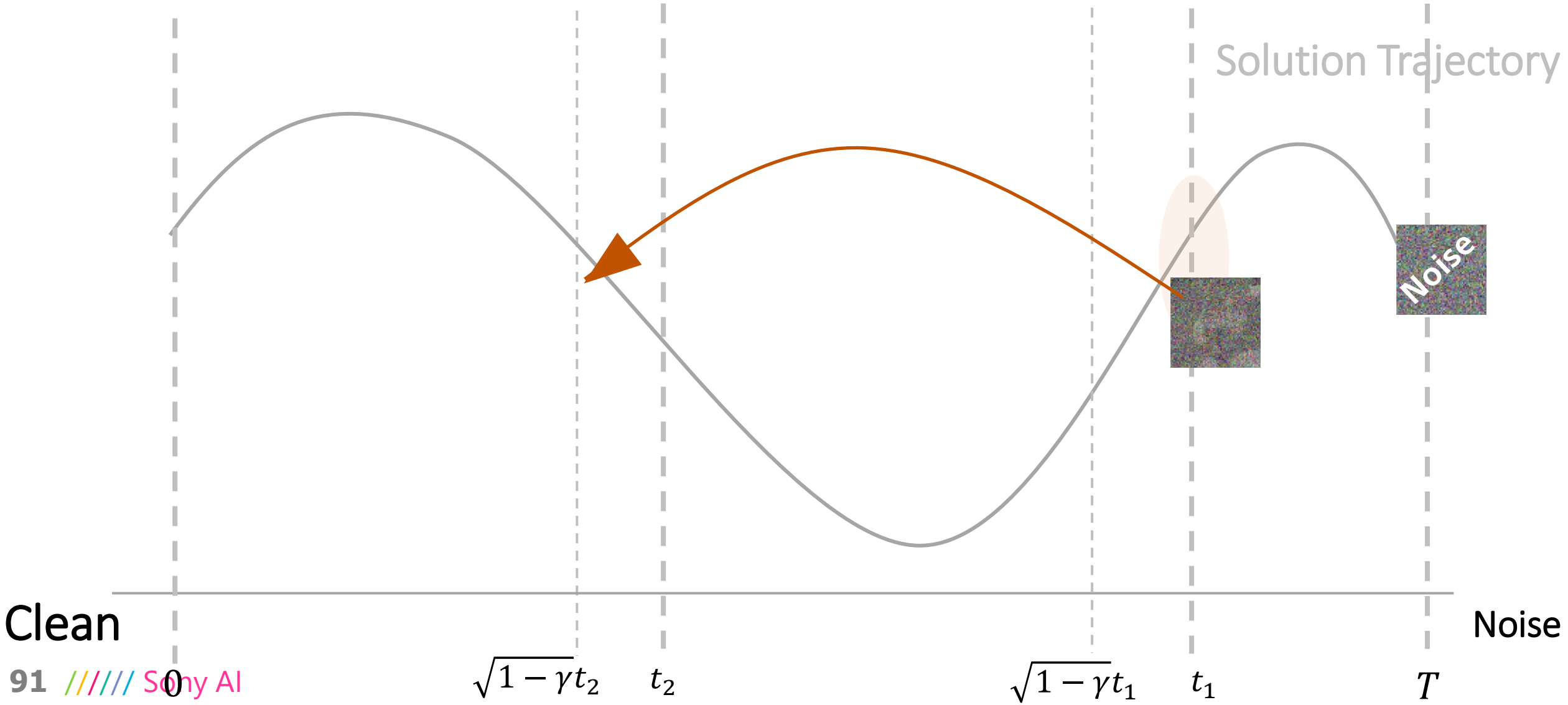
Multistep Generation with CTM: γ-sampling

$\gamma \in (0,1)$: Stochastic Sampling



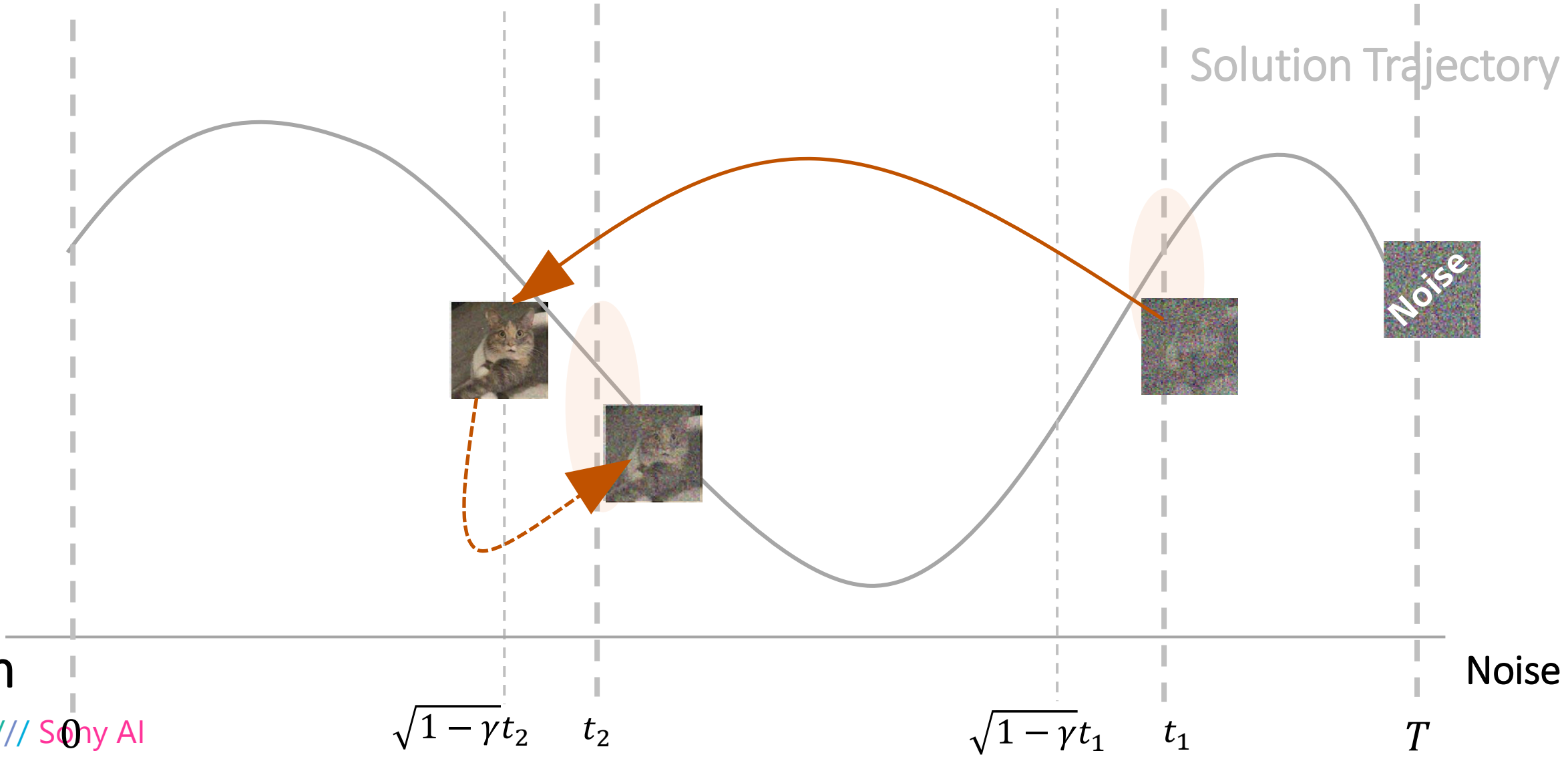
Multistep Generation with CTM: γ-sampling

$\gamma \in (0,1)$: Stochastic Sampling



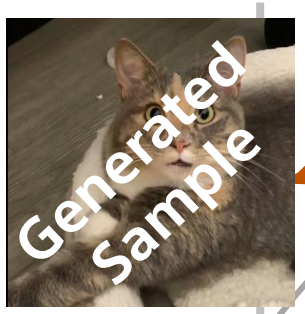
Multistep Generation with CTM: γ-sampling

$\gamma \in (0,1)$: Stochastic Sampling



Multistep Generation with CTM: γ-sampling

$\gamma \in (0,1)$: Stochastic Sampling



Solution Trajectory

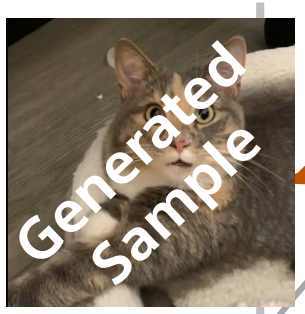
Clean

Noise

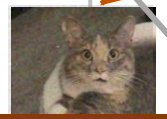
0 $\sqrt{1-\gamma}t_2$ t_2 $\sqrt{1-\gamma}t_1$ t_1 T

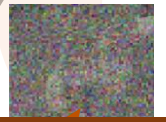
Multistep Generation with CTM: γ-sampling

$\gamma \in (0,1)$: Stochastic Sampling



Solution Trajectory





$$\mathcal{O}\left(\sum_{n=0}^{N-1}\sqrt{t_n-\sqrt{1-\gamma^2}t_{n+1}}\right)$$

0 $\sqrt{1-\gamma}t_2$ t_2 $\sqrt{1-\gamma}t_1$ t_1 T

CTM is a General Recipe to Many Modalities

SoundCTM: Uniting Score-based and Consistency Models for Text-to-Sound Generation

Koichi Saito
Sony AI
NY, USA
koichi.saito@sony.com

Dongjun Kim
Stanford University
CA, USA

Takashi Shibuya
Sony AI
Tokyo, Japan

Chieh-Hsin Lai
Sony AI
Tokyo, Japan

Zhi Zhong
Sony Group Corporation
Tokyo, Japan

Yuhta Takida
Sony AI
Tokyo, Japan

Yuki Mitsufuji
Sony AI, Sony Group Corporation
NY, USA

DITTO-2: Distilled Diffusion Inference-Time T-Optimization for Music Generation

Zachary Novack[1] **Julian McAuley**[1] **Taylor Berg-Kirkpatrick**[1] **Nicholas Bryan**[2]

[1] University of California – San Diego
[2] Adobe Research

znovack@ucsd.edu, njb@ieee.org

SoundCTM's Generation with γ-sampling

SoundCTM: Uniting Score-based and Consistency Models for Text-to-Sound Generation

Koichi Saito[1], Dongjun Kim[2], Takashi Shibuya[1], Chieh-Hsin Lai[1],
Zhi Zhong[3], Yuhta Takida[1], Yuki Mitsufuji[1,3],
[1]Sony AI, [2]Stanford University, [3]Sony Group Corporation

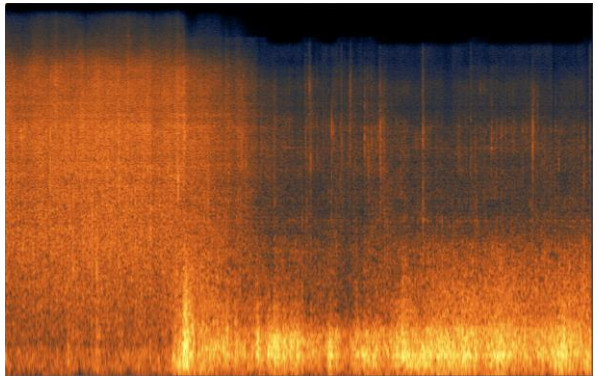
Paper PDF | arXiv | Code | Hugging Face

[Saito+ '24] SoundCTM

SoundCTM's Generation with γ-sampling

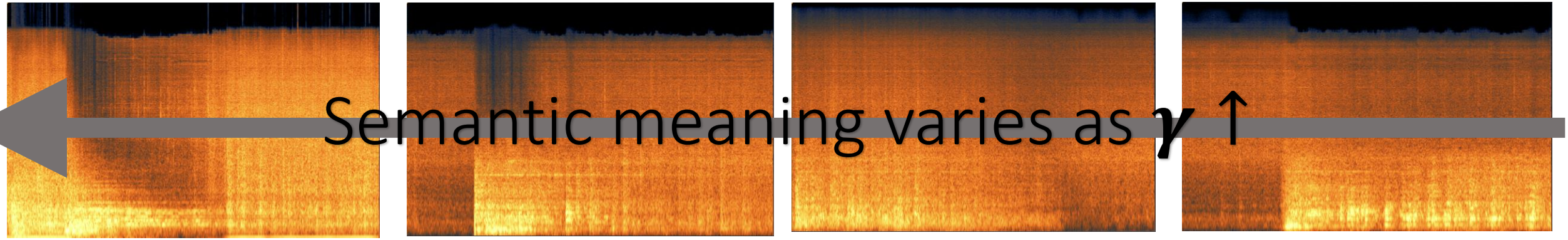
User specified Text prompt: *Thunderclaps, and hard rain falls and splashes on surfaces.*

One step



Semantically similar

16 steps

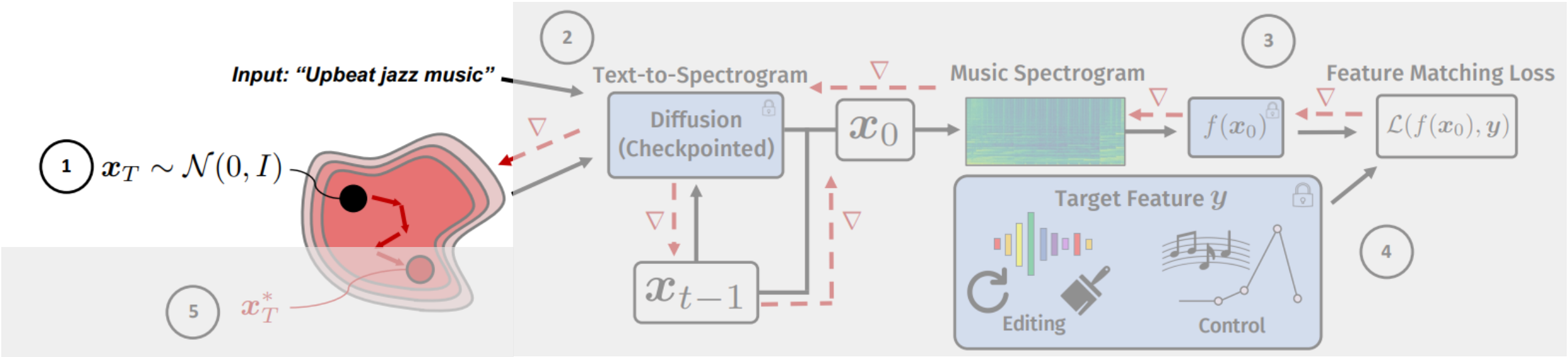


Semantic meaning varies as $\gamma \uparrow$

γ=1 γ=0.5 γ=0.25 γ=0

[Saito+ '24] SoundCTM

DITTO-2: CTM-Based for Controllable Text-to-Music



DITTO:

Diffusion Inference-Time T-Optimization for Music Generation

Zachary Novack[1,2*] Julian McAuley[1] Taylor Berg-Kirkpatrick[1] Nicholas J. Bryan[2]

[1]University of California, San Diego

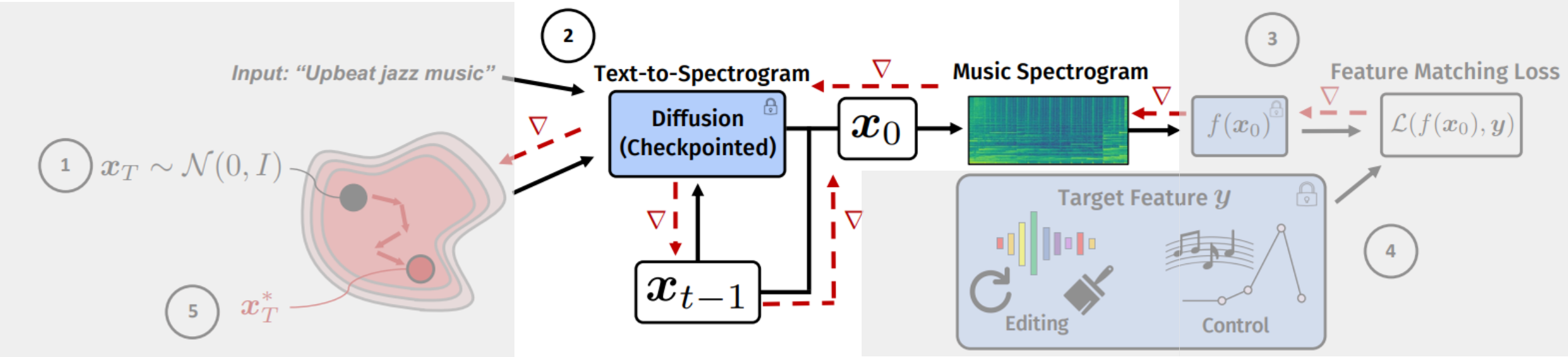
[2]Adobe Research

[*]Work done during an internship at Adobe Research

Paper | Video | HF paper

[Novack+ '24 DITTO; DITTO-2]

DITTO-2: CTM-Based for Controllable Text-to-Music



DITTO:

Diffusion Inference-Time T-Optimization for Music Generation

Zachary Novack[1,2*] Julian McAuley[1] Taylor Berg-Kirkpatrick[1] Nicholas J. Bryan[2]

[1]University of California, San Diego

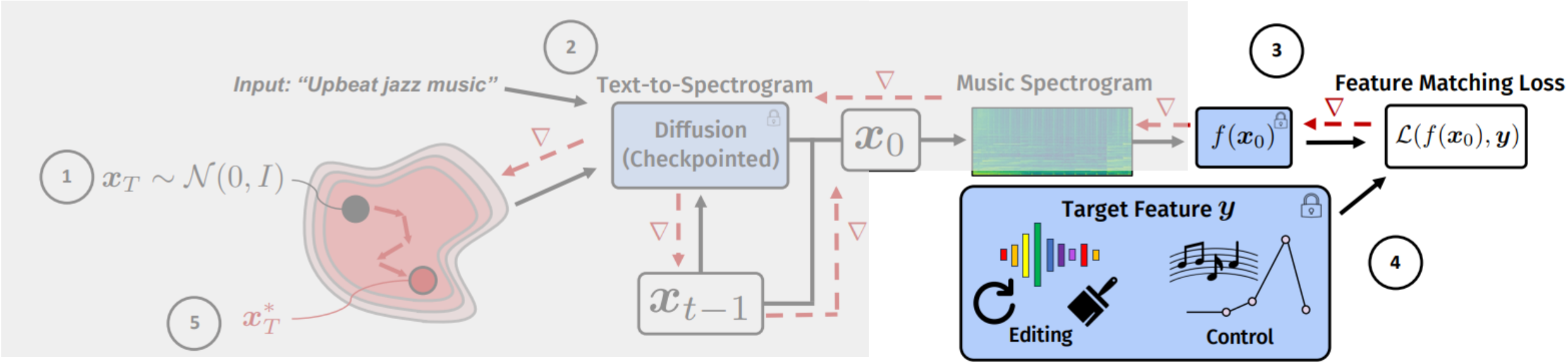
[2]Adobe Research

[*]Work done during an internship at Adobe Research

Paper | Video | HF paper

[Novack+ '24 DITTO; DITTO-2]

DITTO-2: CTM-Based for Controllable Text-to-Music



DITTO:

Diffusion Inference-Time T-Optimization for Music Generation

Zachary Novack[1,2*] Julian McAuley[1] Taylor Berg-Kirkpatrick[1] Nicholas J. Bryan[2]

[1]University of California, San Diego

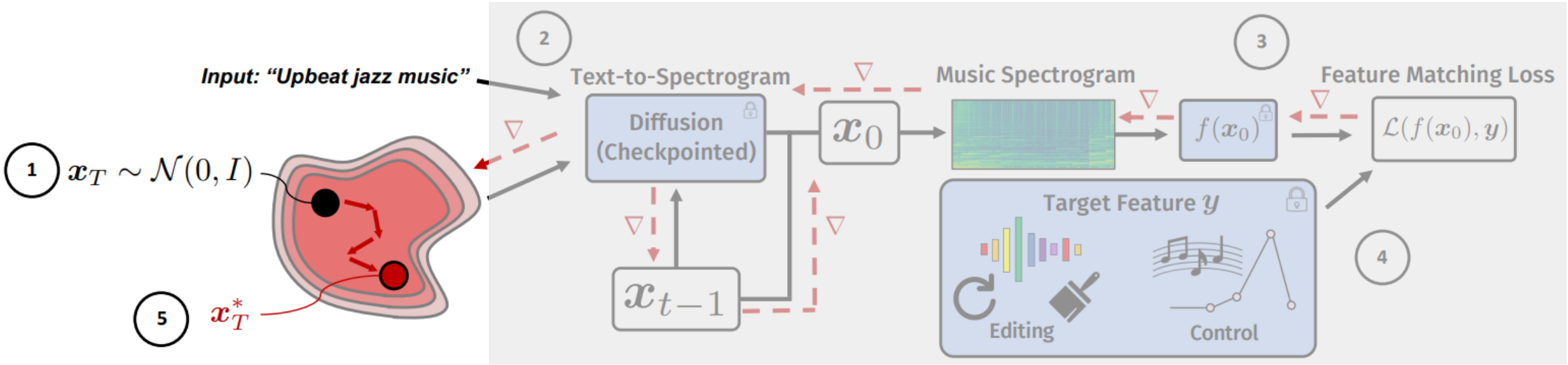
[2]Adobe Research

[*]Work done during an internship at Adobe Research

Paper | Video | HF paper

[Novack+ '24 DITTO; DITTO-2]

DITTO-2: CTM-Based for Controllable Text-to-Music



DITTO:

Diffusion Inference-Time T-Optimization for Music Generation

Zachary Novack[1,2*] Julian McAuley[1] Taylor Berg-Kirkpatrick[1] Nicholas J. Bryan[2]

[1]University of California, San Diego

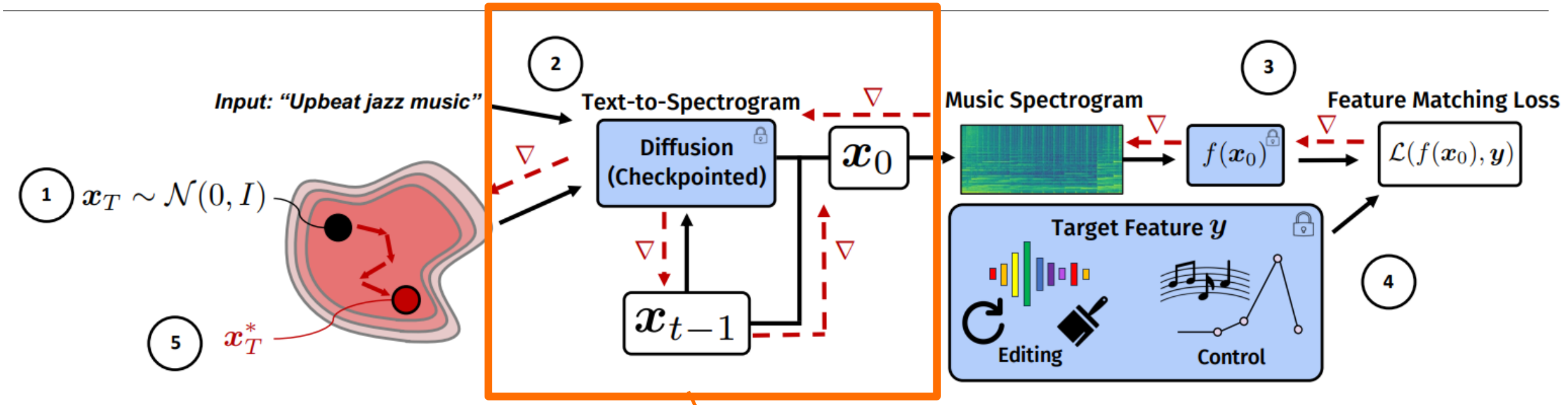
[2]Adobe Research

[*]Work done during an internship at Adobe Research

Paper | Video | HF paper

[Novack+ '24 DITTO; DITTO-2]

DITTO-2: CTM-Based for Controllable Text-to-Music



DITTO-2:

Distilled Diffusion Inference-Time T-Optimization for Music Generation

Zachary Novack[1] Julian McAuley[1] Taylor Berg-Kirkpatrick[1] Nicholas J. Bryan[2]

[1]University of California, San Diego

[2]Adobe Research

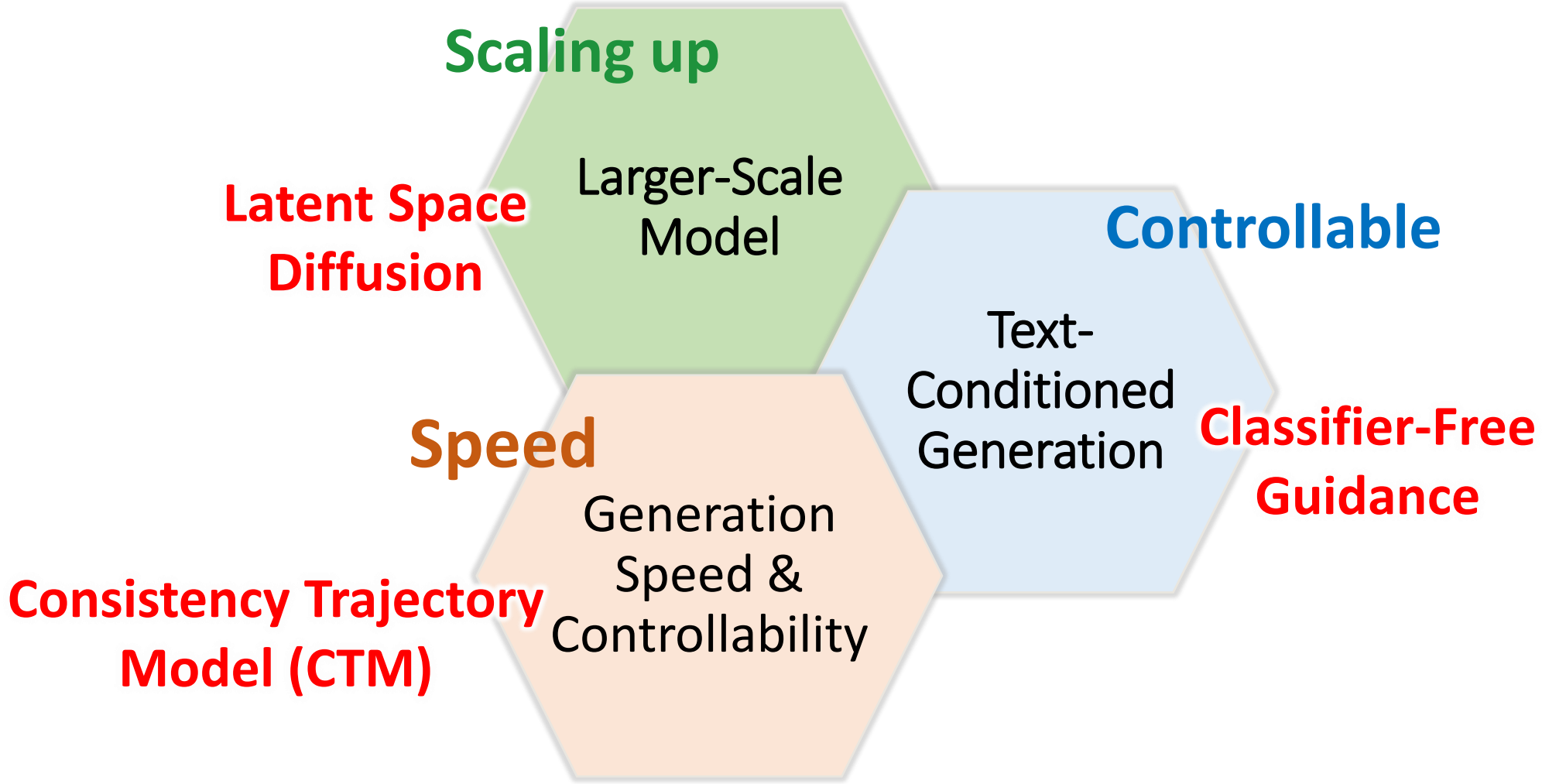
Paper | HF paper

△ Diffusion Multi-steps Generation

✔ CTM's γ-sampling

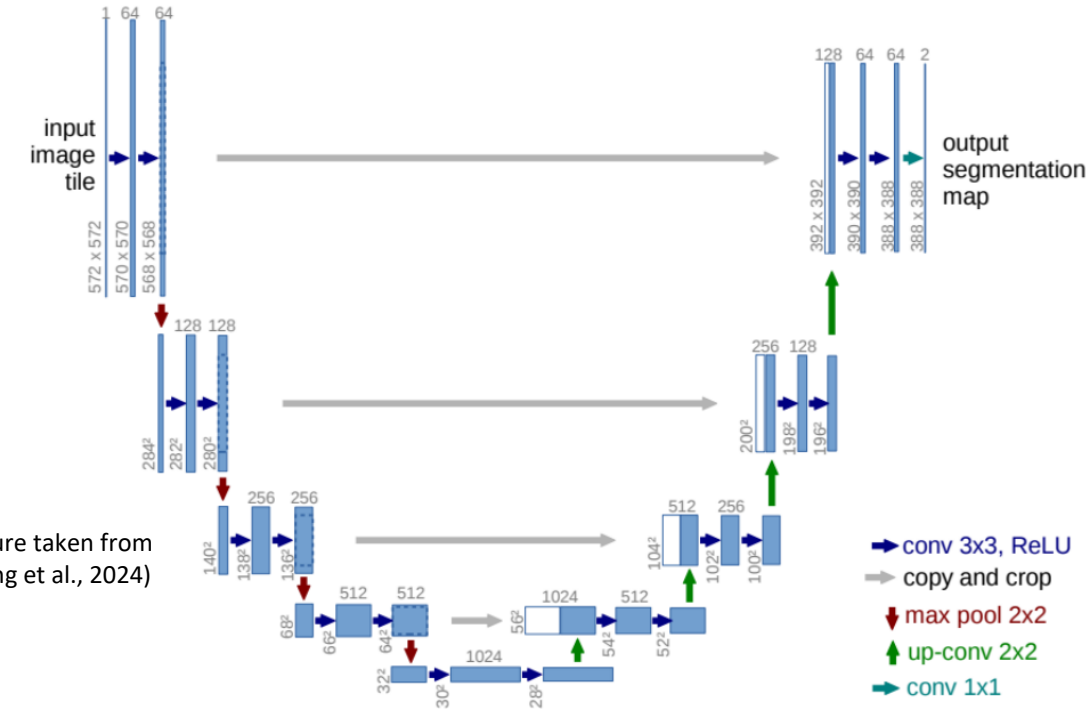
[Novack+ '24 DITTO; DITTO-2]

Making Diffusion Model More Powerful



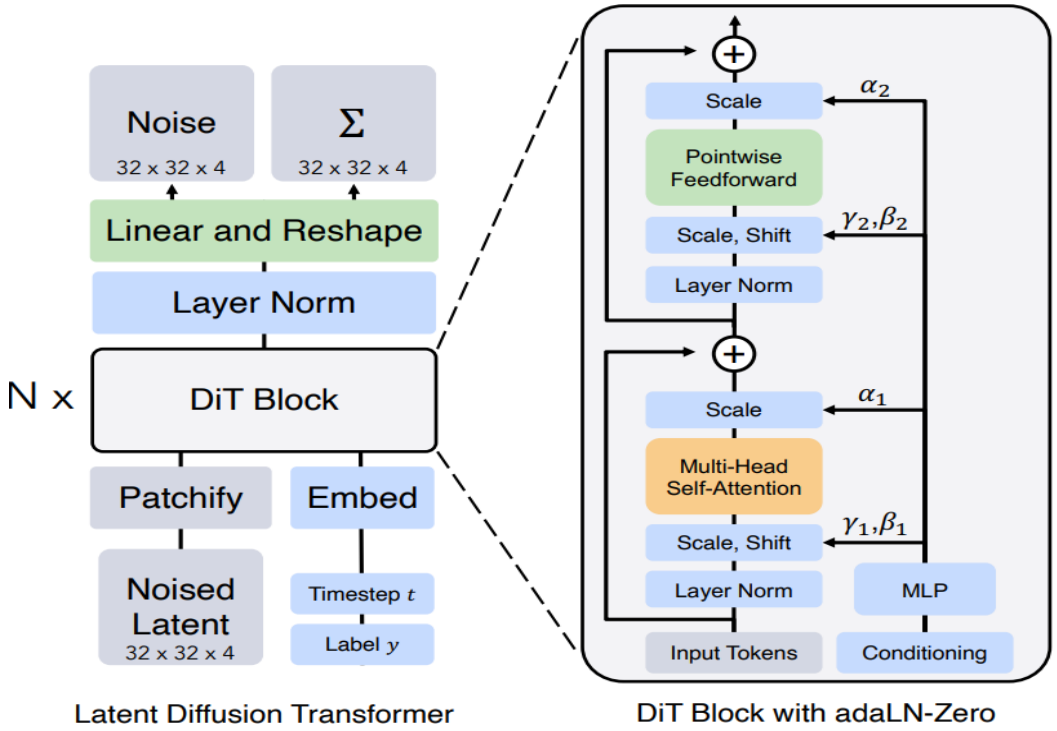
PRACTICAL IMPLEMENTATIONS

Network Architectures



(Figure taken from Zhang et al., 2024)

UNet



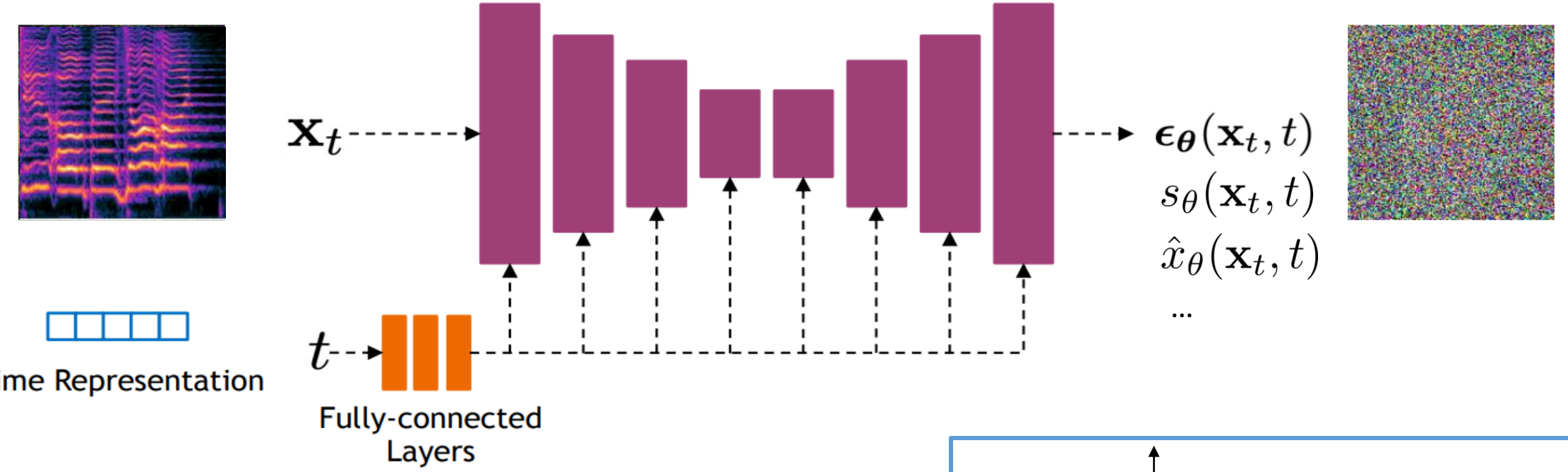
Diffusion Transformer (DiT)

- Downsampling and Upsampling stacks
- ResNet blocks and self-attention layers

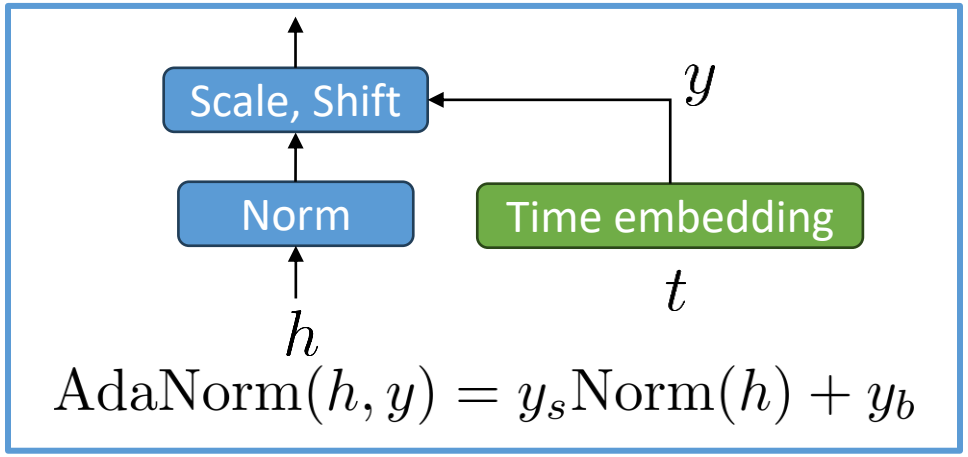
- Transformer architecture
- AdaLN-Zero blocks

[Peebles and Xie ICCV 2023] Scalable Diffusion Models with Transformers
[Nichol and Dhariwal, ICML 2021] Improved Denoising Diffusion Probabilistic Models

Timestep Embedding Conditioning

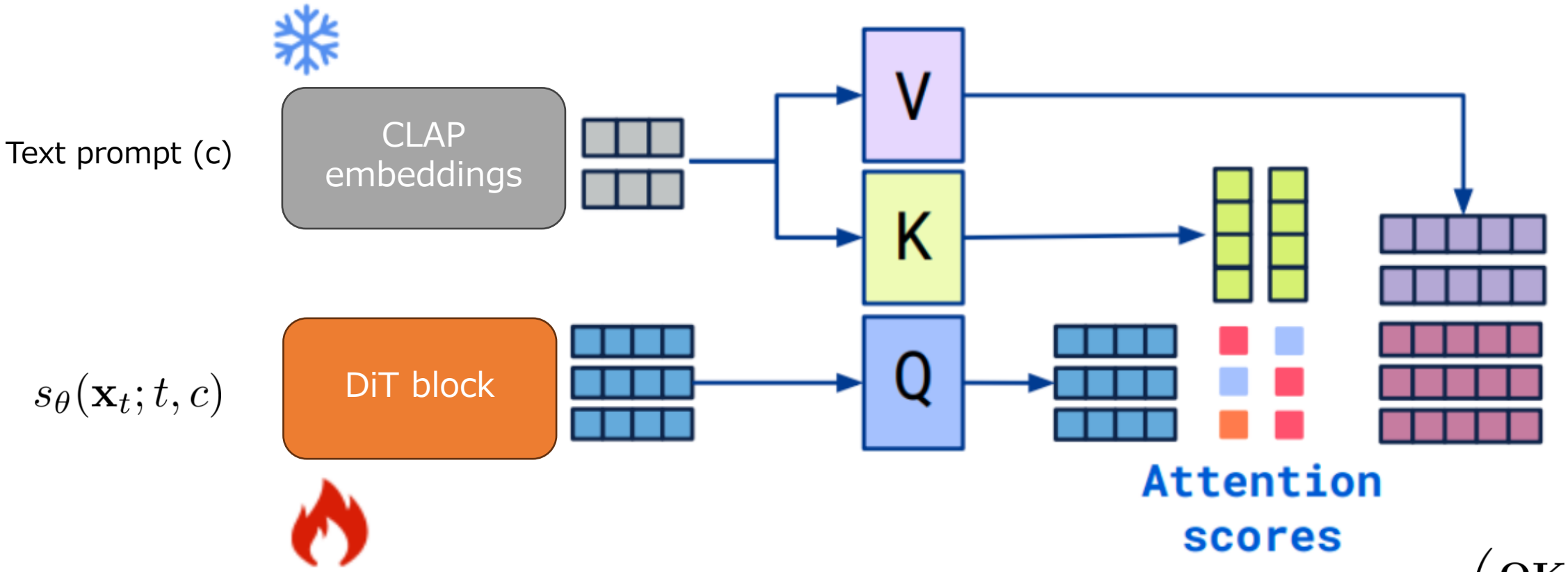


- Time embeddings are sinusoidal positional embeddings or random Fourier features
- Time embeddings can be concatenated, added or through adaptative normalization layers



[Kreis et al., CVPR 2022] Denoising Diffusion-based Generative Modeling: Foundations and Applications

Cross-Attention Conditioning



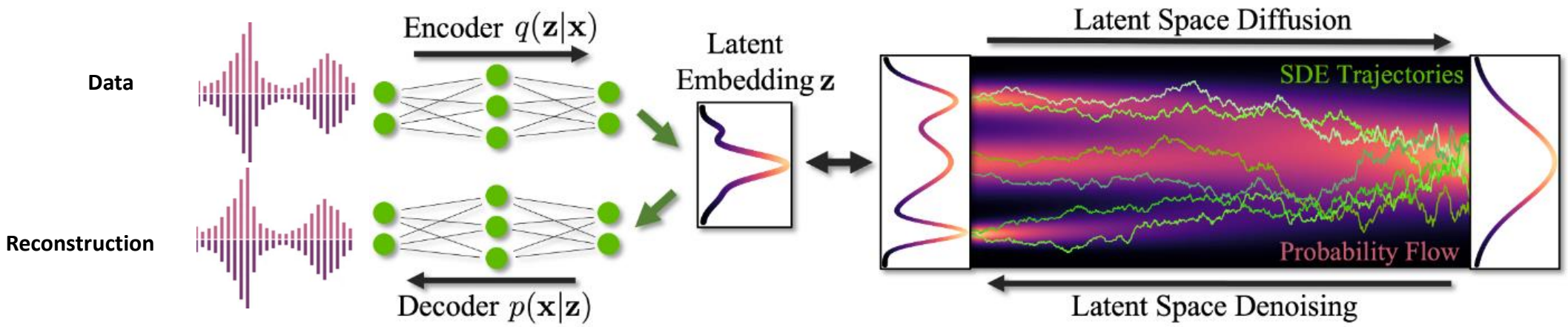
$$\text{Cross\_Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}$$

- Text prompts are embedded with CLAP, T5, ⋯
- Using cross attention to condition the text prompt

Latent Diffusion Models

- Music generation requires modelling long-range sequences
  - Using the full frequency spectrum (48 kHz vs 16 kHz)
- Music contains harmonies and melodies from different instruments, create complex structures
  - Human listeners are highly sensitive to disharmony

Compress the audio with a pre-trained auto-encoder



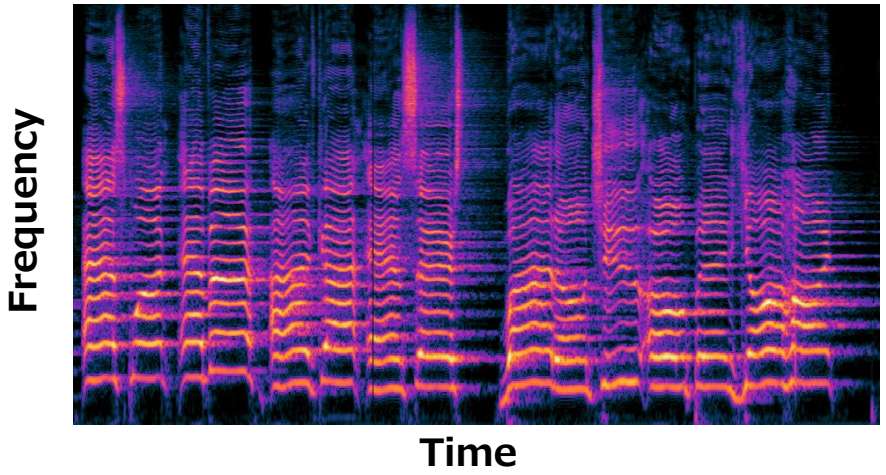
Latent diffusion models

[Kreis et al., NeurIPS 2023] Latent Diffusion Models: Is the Generative AI Revolution Happening in Latent Space?

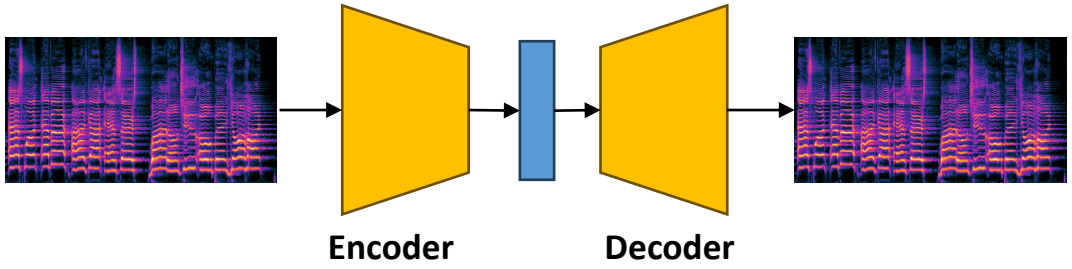
Encoder-Decoder

➢ Audio spectrograms

- Can be seen as an image, but requires special consideration: temporal structure, data augmentation, frequencies are non-locally distributed, …)

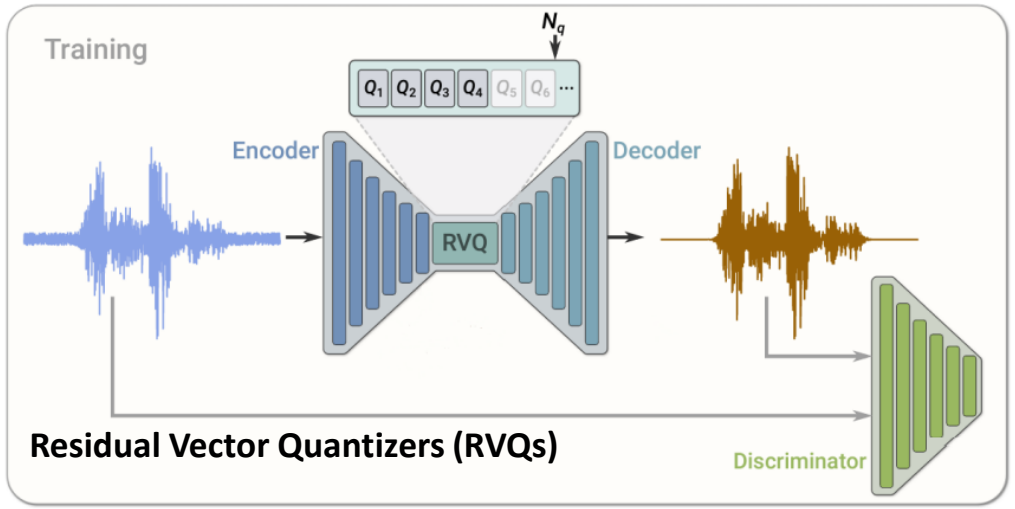


➢ Variational Auto-Encoders (VAE)



➢ Neural Audio Codecs and RVQs

- SoundStream, EncoDec, DAC, …



90x compression

[Kumar et al., NeurIPS 2023] High-Fidelity Audio Compression with Improved RVQGAN
[Zeghidour et al., Trans. Audio, 2021] SoundStream: An End-to-End Neural Audio Codec
[Défossez et al., TMLR 2024] High Fidelity Neural Audio Compression

APPLICATIONS

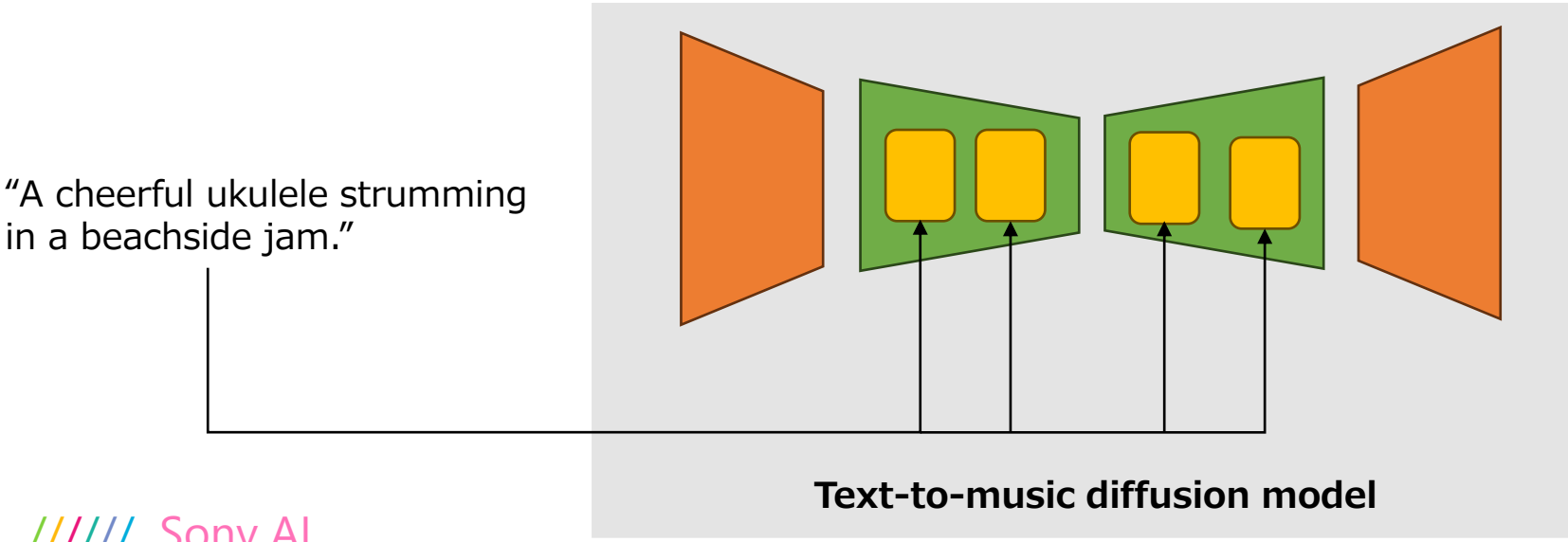
- Controllable generation
- Inverse problem

CONTROLLABLE GENERATION

- **Text-to-Music Generation**
- **Finetuning**
- **Inference-Time Optimization**
- **Inference-Time Guidance**

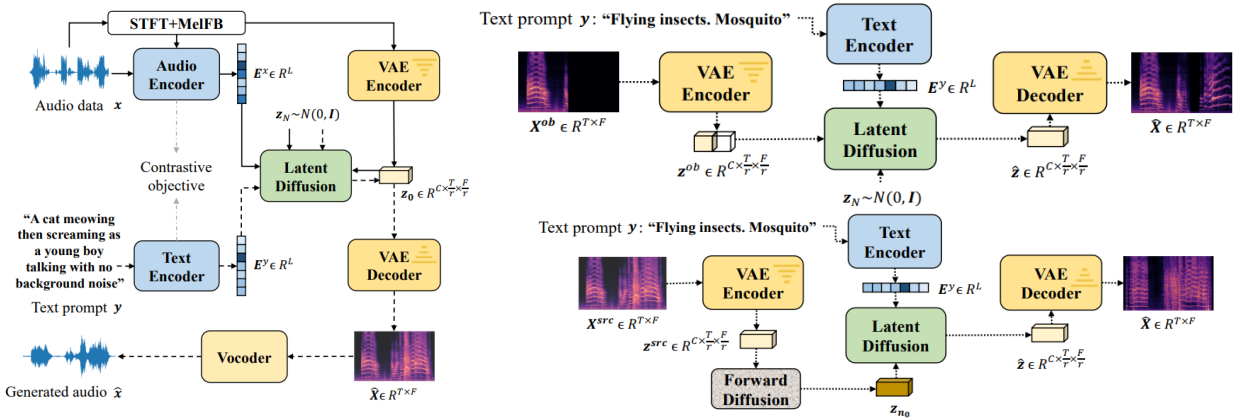
Text-to-Music Generation (I)

- **Problem**
  - A generative model that generates music in response to text prompts
- **Applications**
  - Music composition and soundtrack creation
  - Assistance for musicians
  - Music editing and personalized
- **Challenges**
  - Modelling long sequence
  - Sound quality
  - A large amount of supervised data

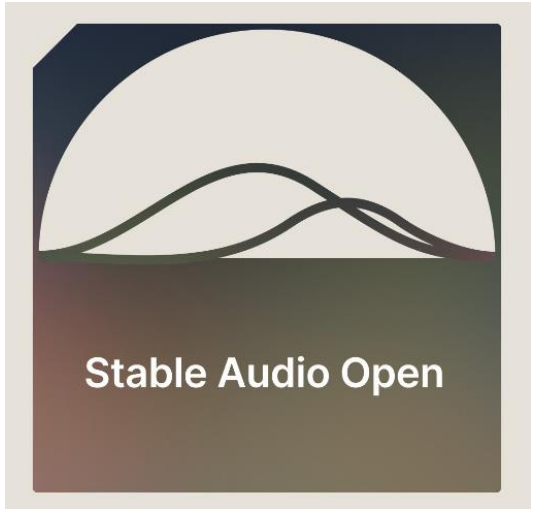


Text-to-Music Generation (II)

AudioLDM: Text-to-Audio Generation with Latent Diffusion Models



https://github.com/haoheliu/AudioLDM

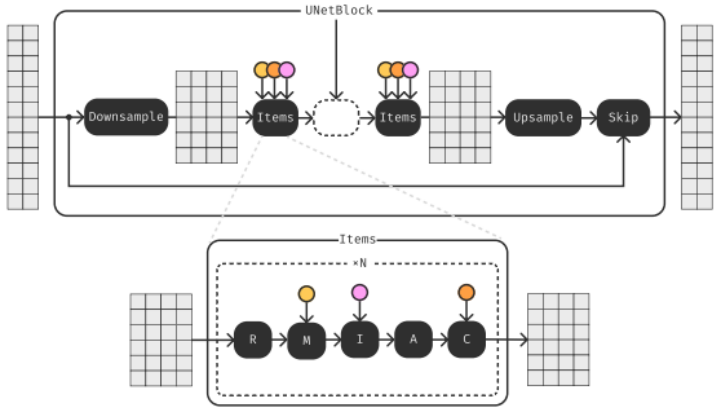


https://github.com/Stability-AI/stable-audio-tools

Riffusion

MUSIC CREATION WITH AI

***Moûsai*: Efficient Text-to-Music Diffusion Models**



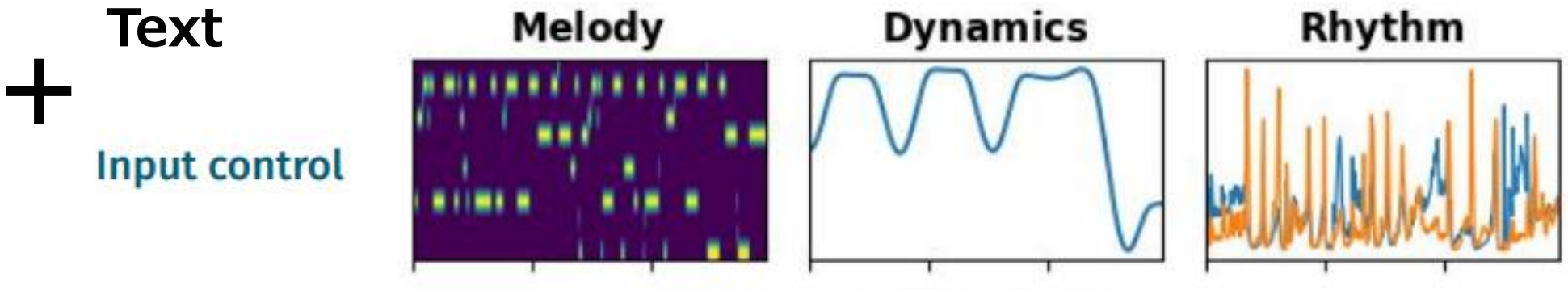
https://github.com/archinetai/audio-diffusion-pytorch

Text-to-Music Generation (III)

[Liu et al., Trans. Audio 2024] AudioLDM 2: Learning Holistic Audio Generation with Self-supervised Pretraining

Text-to-Music Generation (IV)

- Text-to-Music generation models can produce high-fidelity music audio, but they **lack fine-grained controllability**
  - Text descriptions are insufficient for detailed controls
  - Suitable for global attributes like genre, mood, tempo, …
  - Not suitable for fine-grained controls over time-varying attributes



How to enable control with additional information alongside the text description?

[Wu et al., Trans. Audio 2024] Multiple Time-varying Controls for Music Generation

Controllable Generation (I)

- Learning conditional controls in an end-to-end way is challenging
  - Limited training data for a specific condition
  - Training cost is high

long audio files. Finally, we have in total 3,302,553 ten-seconds audio samples for model training. It should be noted that even if some datasets, e.g., AudioCaps and BBC SFX, have text captions for the audio, we do not utilize them during the training of LDMs. We only use the audio samples for training. We resample all the datasets into 16kHz sampling rate and mono format, and all samples are padded to 10 seconds.

AudioLDM

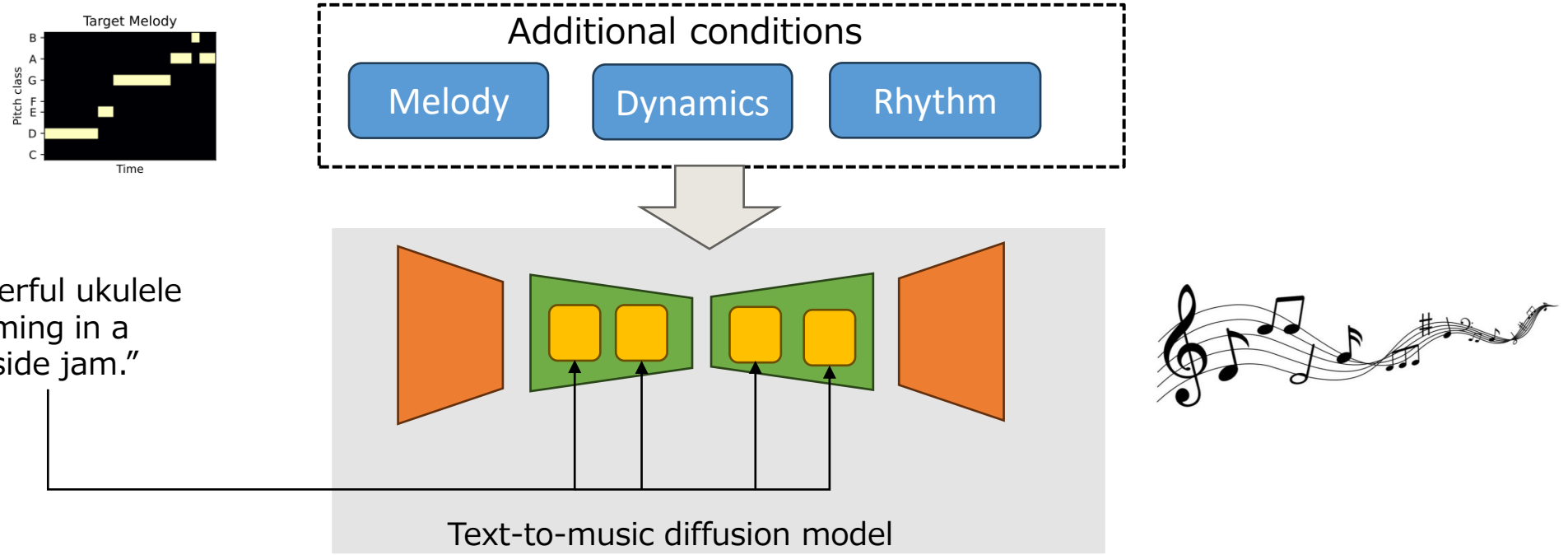
Stability AI used a cluster of 4,000 Nvidia A100 GPUs running in AWS to train Stable Diffusion over the course of a month. CompVis, the machine vision and learning research group at Ludwig Maximilian University of Munich, oversaw the training, while Stability AI donated the compute power.

Stable Diffusion was trained on LAION-5B

- **Goal:** Leveraging the pretrained diffusion models to model **new types of conditions** while ensuring high-fidelity outputs

$$p(\mathbf{x}|\mathbf{c}_{\text{text}}, \mathbf{y})$$

Controllable Generation (II)

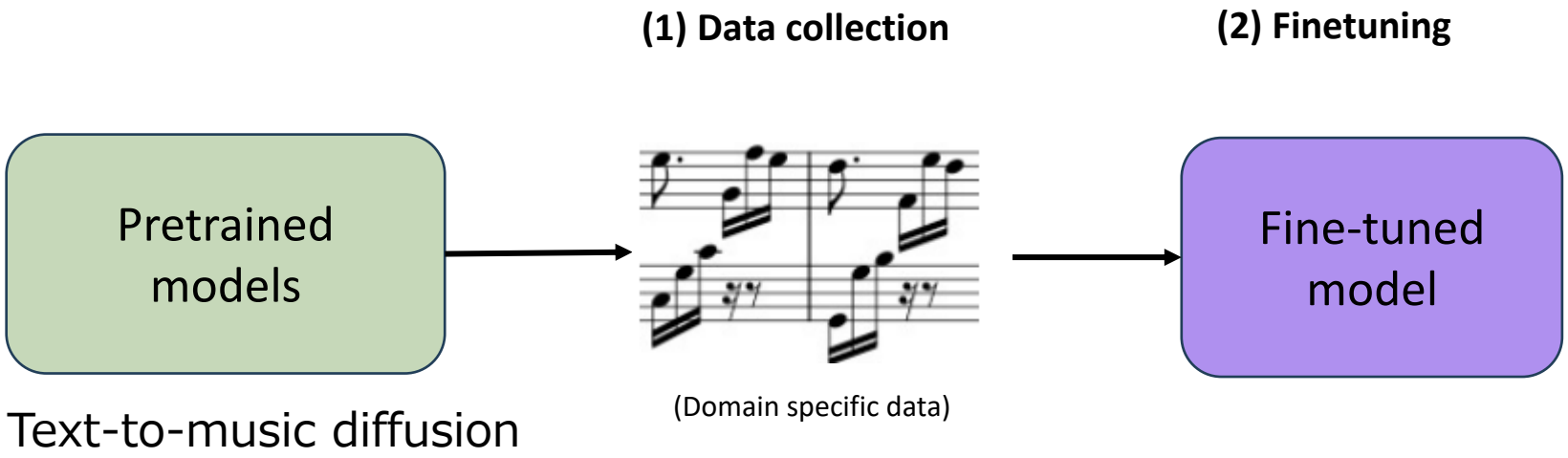


Training-based

Training-free

Finetuning	**Inference-time optimization**	**Inference-time guidance**
• MusicControlNet • …	• DITTO • DITTO-2 • …	• MusicMagus • CQT-Diff • …

Finetuning (I)



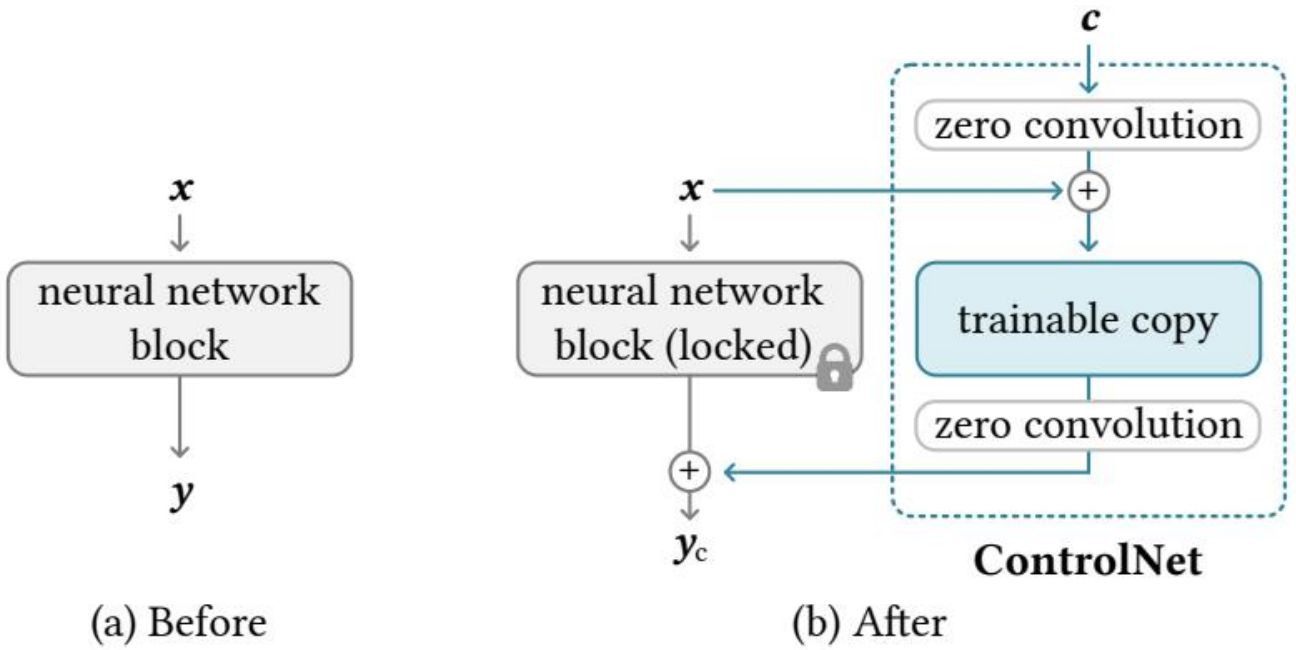
Adjusting the parameters of a pre-trained text-to-music models to a specific conditioning

- **Challenges**
  - Limitation of finetuning data causes **overfitting** and **catastrophic forgetting**

Finetuning (II)

➢ ControlNet

- Adding conditioning controls to large, pretrained diffusion models
- Reusing the deep and robust encoding layers
- “Zero convolutions” ensure that no harmful noise could affect the finetuning

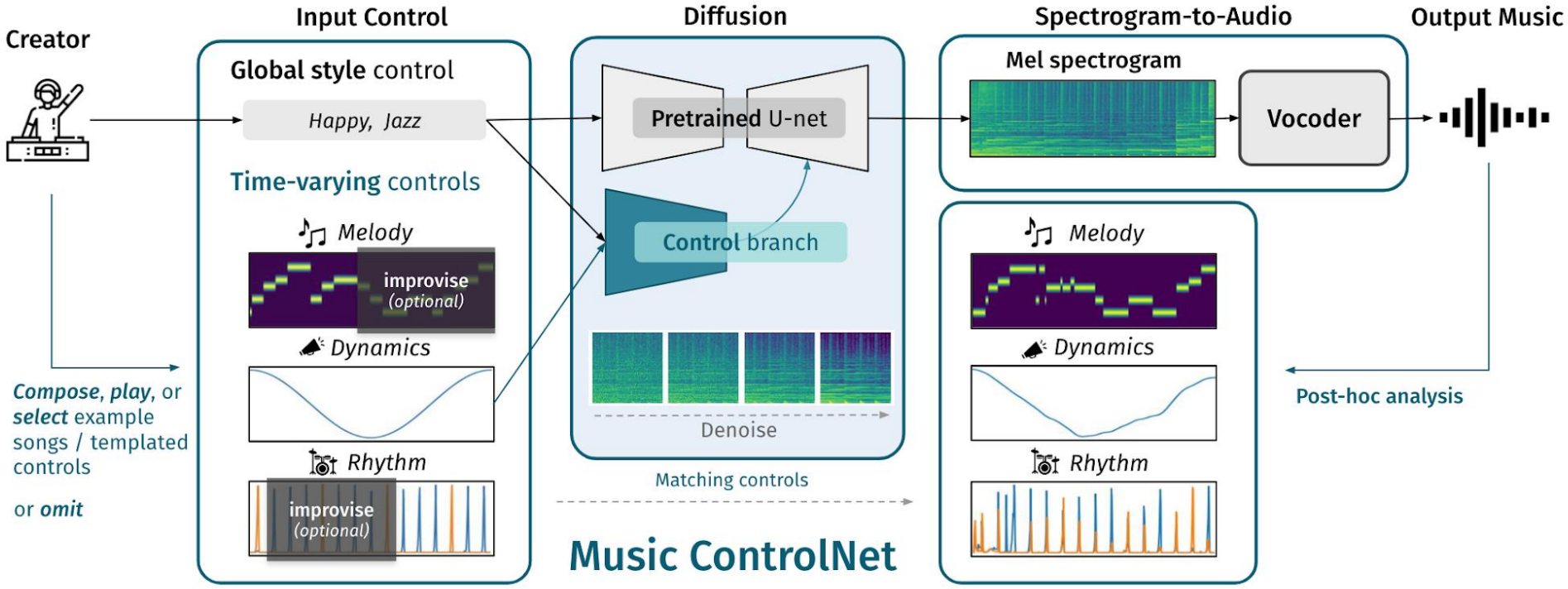


$$\boldsymbol{y}_c = \mathcal{F}(\boldsymbol{x};\Theta) + \mathcal{Z}(\mathcal{F}(\boldsymbol{x} + \mathcal{Z}(\boldsymbol{c};\Theta_{z1});\Theta_c);\Theta_{z2})$$

[Zhang et al., ICCV 2023] Adding Conditional Control to Text-to-Image Diffusion Models

Finetuning (III)

- **MusicControlNet:** Enables fine-grained control generation over
  - Positions of beats in time
  - Dynamics
  - Melody
  - Rhythm



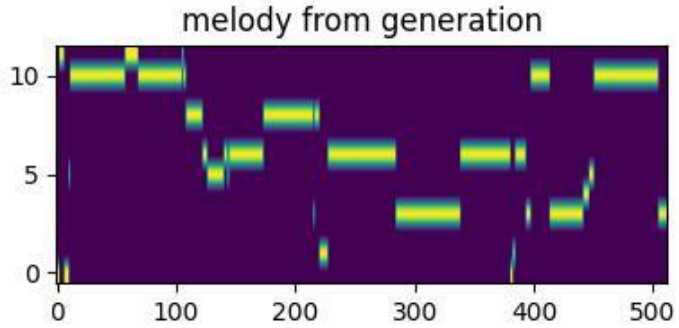
[Wu et al., Trans. Audio 2024] Multiple Time-varying Controls for Music Generation

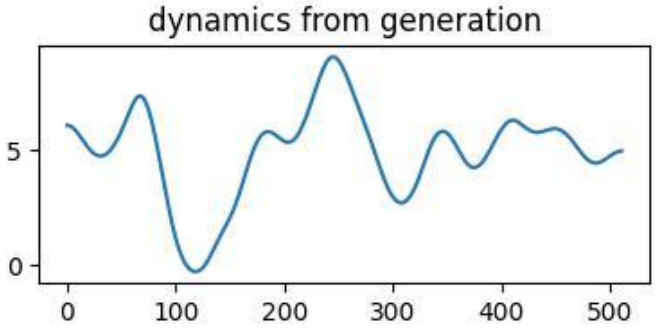
Finetuning (IV)

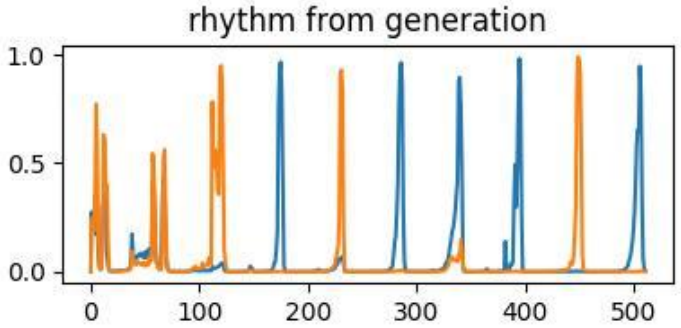
- **MusicControlNet:** Enables fine-grained control generation over

Text prompt: happy, jazz

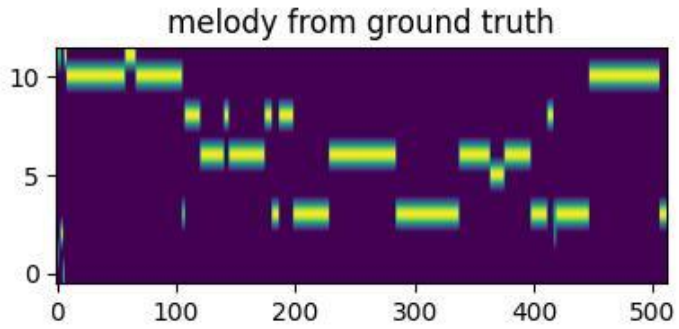
Generated

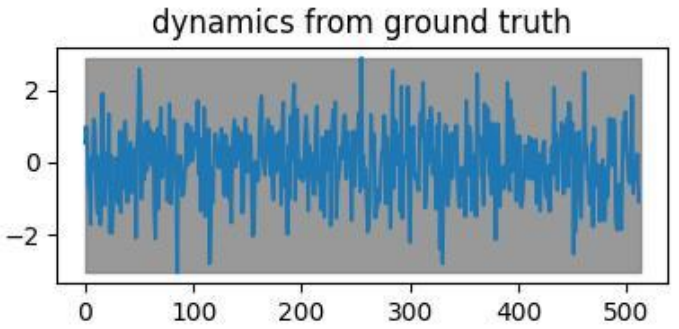


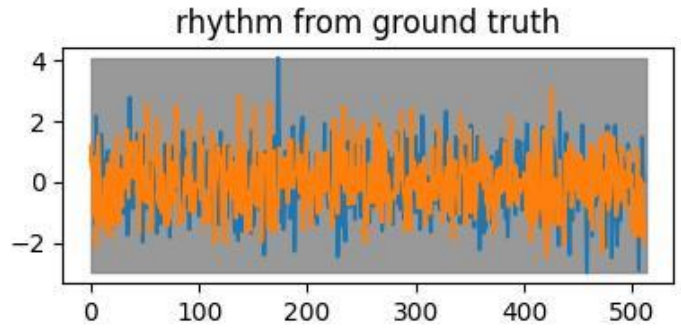




Reference







Melody Control

[Wu et al., Trans. Audio 2024] Multiple Time-varying Controls for Music Generation

Inference-Time Optimization (I)

➢ **Formulate the control task as an optimization problem**

- Searching for the noise such that the model output follows the target control $\mathcal{A}$

$$\arg\min_{\mathbf{x}_T \in \mathcal{N}(0,\mathbf{I})} \quad \mathcal{L}(\mathcal{A}(\mathbf{x}_0), \mathbf{y})$$

$$\mathbf{x}_0 = \mathrm{Sampler}_T(\epsilon_\theta, \mathbf{x}_T, \mathbf{c}_{\mathrm{text}})$$

$\mathbf{x}_T$ is optimized rather than model weights

➢ **Challenges**

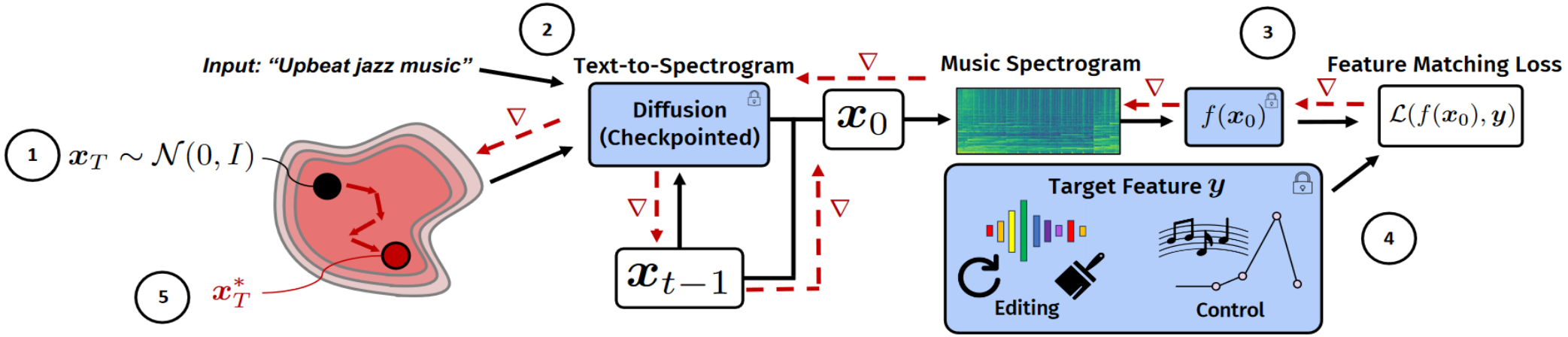
- Diffusion sampling process consists of multiple steps
- Solving the optimization problem requires huge memory

[Novack et al., ICML 2024] DITTO: Diffusion inference-time t-optimization for music generation

Inference-Time Optimization (II)

➢ **DITTO**

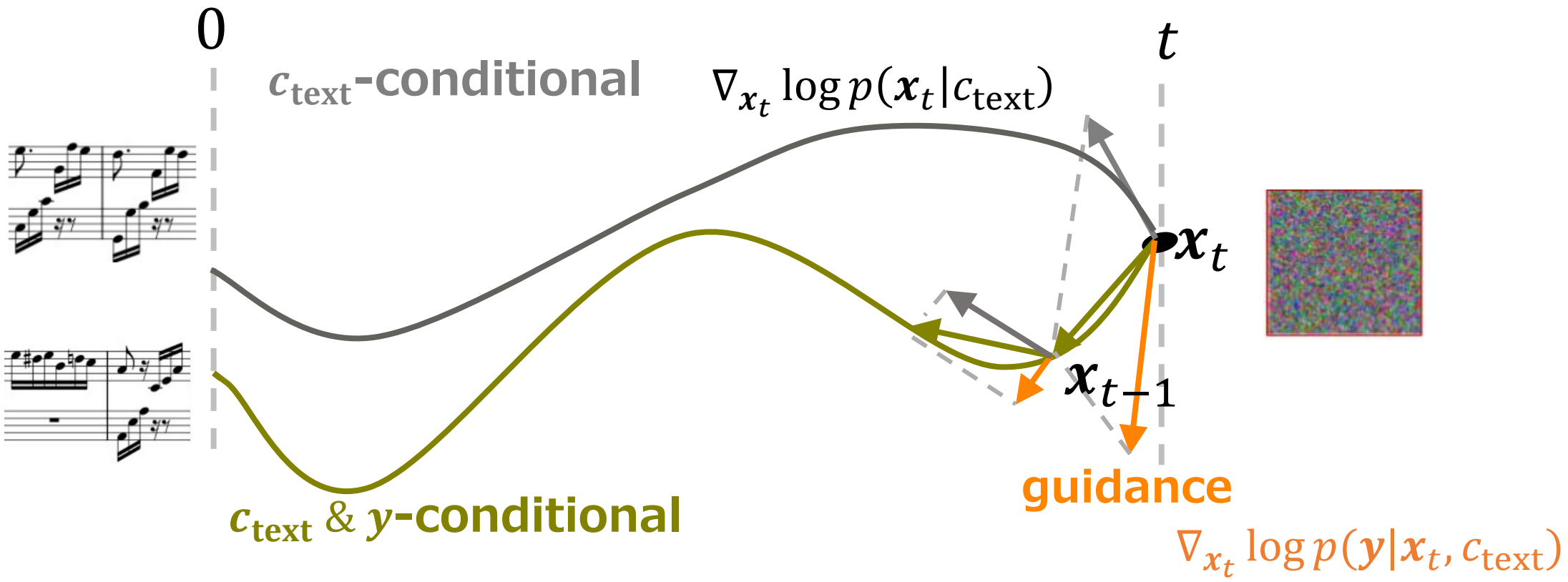
- Gradient Checkpointing to circumvent the large memory usage
- Store the intermediate noisy diffusion tensors



➢ Provide controllability over intensity, melody, musical structure, and looping

[Novack et al., ICML 2024] DITTO: Diffusion inference-time t-optimization for music generation

Inference-Time Guidance (I)

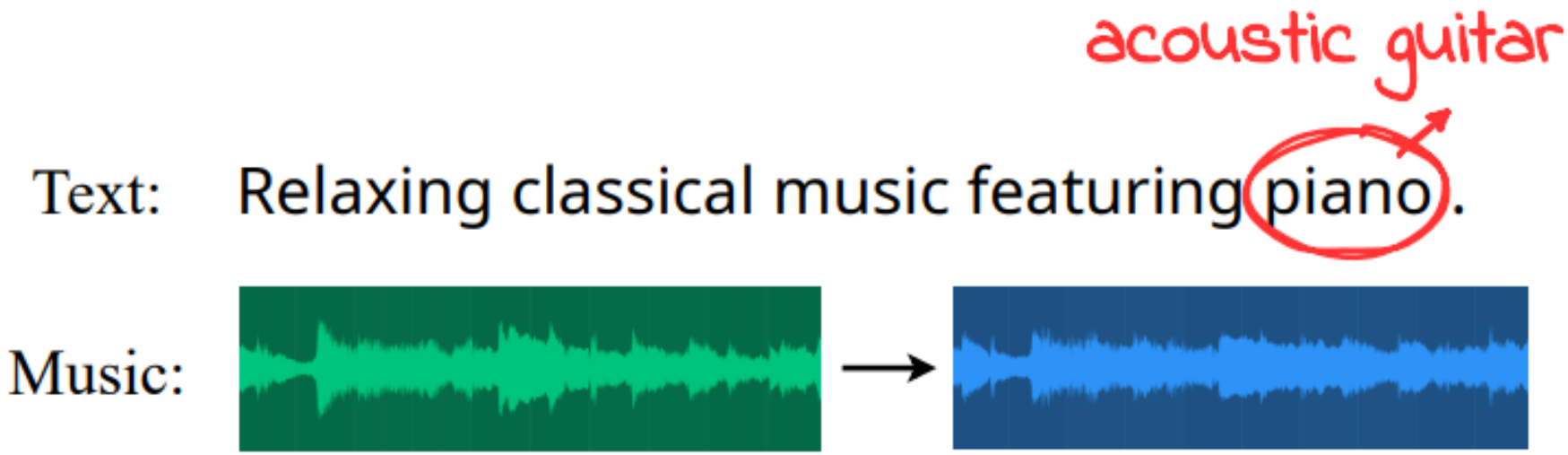


$$\nabla_{x_t} \log p(x_t|c_{\text{text}}) + \nabla_{x_t} \log p(y|x_t, c_{\text{text}})$$

- **Challenges**
  - Require an approximation of the guidance during sampling

Inference-Time Guidance (II)

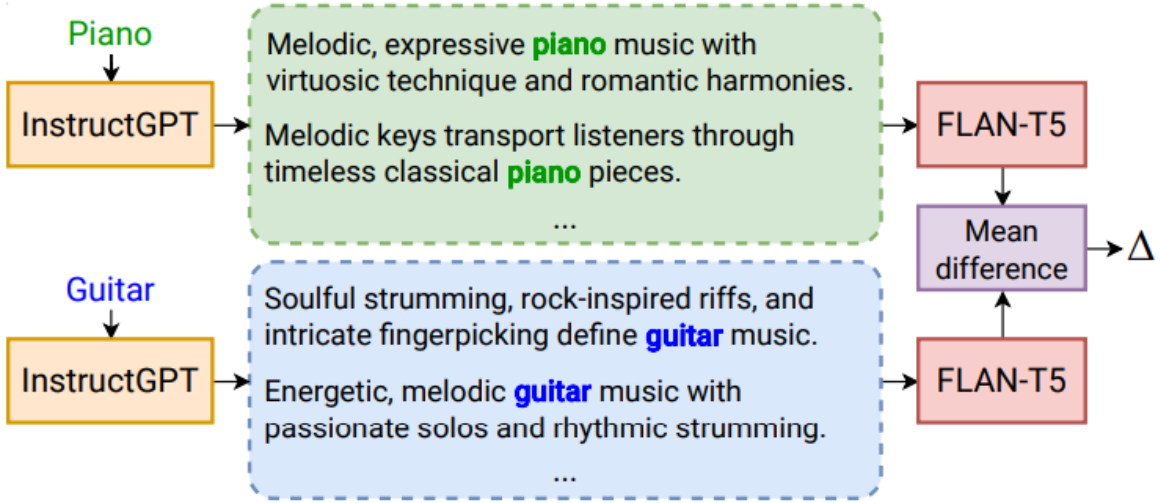
- **MusicMagus**
  - Focuses on modifications such as adding effects or changing instruments
    - E.g., `add reverb to this soundtrack`, `transfer the timbre of the specific notes`, etc.
  - It can perform zero-shot editing without additional training pairs



[Zhang et al., IJCAI 2024] MusicMagus: Zero-Shot Text-to-Music Editing via Diffusion Models

Inference-Time Guidance (II)

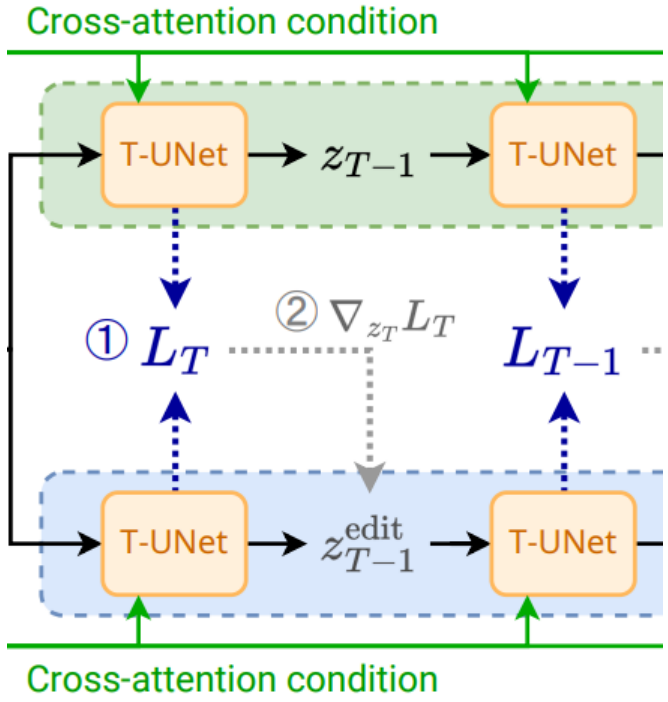
- Finding editing direction



$$E^{\text{edit}} = \{E_{\text{T5}} + \Delta, E'_{\text{GPT}}\}$$

Example: Piano to violin

- Adding constraints over cross-attention



$$L_t = \|M_t^{\text{edit}} - M_t^{\text{origin}}\|_2$$

$$\epsilon_\theta^{\text{edit}} = \epsilon_\theta(z_t - \alpha \nabla_{z_t} L_t, E^{\text{edit}}, t)$$

[Zhang et al., IJCAI 2024] MusicMagus: Zero-Shot Text-to-Music Editing via Diffusion Models

Takeaways

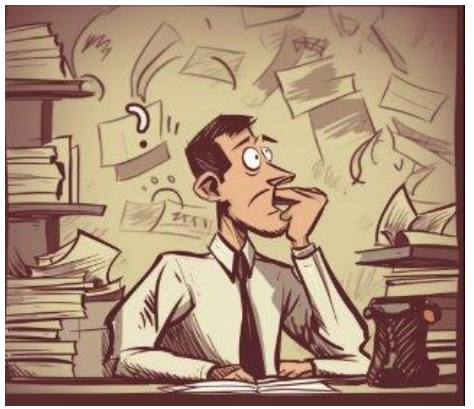
Training-based | Training-free

Finetuning	Inference-time optimization	Inference-time guidance
✓ Arbitrary controls ✓ Fast inference	✓ Zero training ✓ Applicable to many tasks	✓ Zero training ✓ Applicable to many tasks
- Large-scale training data - Only applicable to specific problems	- Slow inference - Differentiable control	- Limited fine-grained controllability

➢ Which method should I use to enable controllable generation for my problem?

- Do I have enough budgets (data + compute + skills)? → Finetuning
- I'm working on small data and want quick results → Inference-time guidance
- I'm willing to pay some cost (slow at inference) because I care about quality

  → Inference-time optimization



INVERSE PROBLEM

- **General Inverse Problem**
- **Linear Inverse Problem**
- **Linear Blind Inverse Problem**

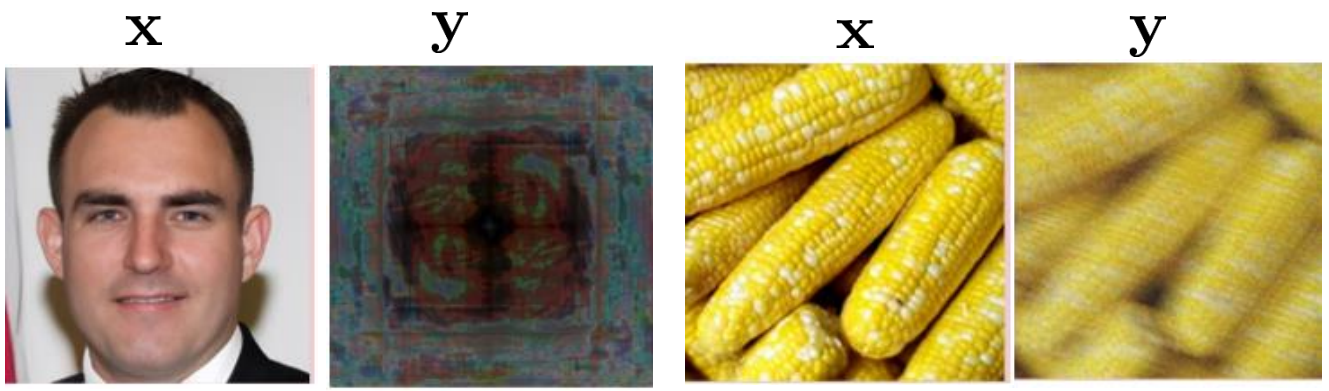
Motivation

- Finding $\mathbf{x}$ given $\mathbf{y}$

$$p(\mathbf{x}|\mathbf{y})$$

$$\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{z}$$

Applications



Phase retrieval deblurring

Ill-posed: There are infinitely many solutions …

$\mathbf{z} \sim \mathcal{N}(0, \sigma_{\mathbf{y}}^2 \mathbf{I})$ Gaussian noise with known variance

$\mathcal{A}$ Measurement operator

- **How can we use diffusion models for solving general inverse problems?**
  - Training is expensive
  - It is data-specific and not applicable to many inverse problems

Challenge

- **How to sample with diffusion model?**

$$\mathbf{x}_{t-dt} \leftarrow \mathbf{x}_t - t\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y})dt$$

Using Bayesian theorem

$$p(\mathbf{x}_t|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{y})}$$

$$\log p(\mathbf{x}_t|\mathbf{y}) = \log p(\mathbf{y}|\mathbf{x}_t) + \log p(\mathbf{x}_t) - \log p(\mathbf{y})$$

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$$

Don't know **Diffusion model**

General Inverse Problem (I)

$$\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{z}$$

Given:

$\mathcal{A}: \mathbb{R}^n \to \mathbb{R}^m$ A known measurement operator

$\mathbf{y} \in \mathbb{R}^m$ Measurement

$\mathbf{z} \sim \mathcal{N}(0, \sigma_{\mathbf{y}}^2 \mathbf{I})$ Gaussian noise with known variance

Find:

$\mathbf{x} \in \mathbb{R}^n$ The true signal

How to estimate?

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

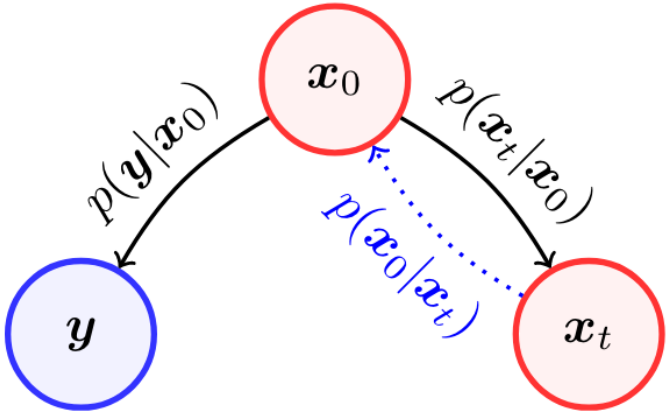
General Inverse Problem (II)

- Diffusion Posterior Sampling (DPS)

$$p(\mathbf{y}|\mathbf{x}_t) = \int p(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_t) p(\mathbf{x}_0|\mathbf{x}_t) d\mathbf{x}_0$$

$$= \int p(\mathbf{y}|\mathbf{x}_0) p(\mathbf{x}_0|\mathbf{x}_t) d\mathbf{x}_0$$

$$= \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)}[p(\mathbf{y}|\mathbf{x}_0)] \simeq p(\mathbf{y}|\hat{\mathbf{x}}_0) \quad \text{where} \quad \hat{\mathbf{x}}_0 := \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$$



$\log p(\mathbf{y}|\mathbf{x}_0)$ **is known**

$\log p(\mathbf{y}|\mathbf{x}_t)$ is **unknown**

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) \simeq \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_0)$$

$$-\frac{1}{\sigma_{\mathbf{y}}^2} \nabla_{\mathbf{x}_t} \|\mathbf{y} - \mathcal{A}(\hat{\mathbf{x}}_0(\mathbf{x}_t))\|_2^2$$

[Chung et al., ICLR 2023] Diffusion Posterior Sampling for General Noisy Inverse Problems

Linear Inverse Problem (I)

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z}$$

Given:

$\mathbf{H} \in \mathbb{R}^{m \times n}$ A known linear degradation matrix

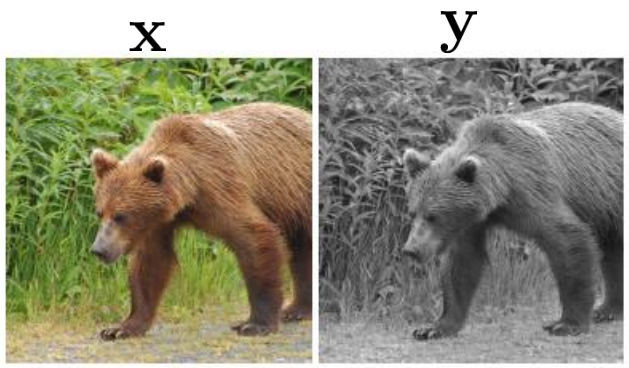
$\mathbf{y} \in \mathbb{R}^{m}$ Measurement

$\mathbf{z} \sim \mathcal{N}(0, \sigma_{\mathbf{y}}^2 \mathbf{I})$ Gaussian noise with known variance

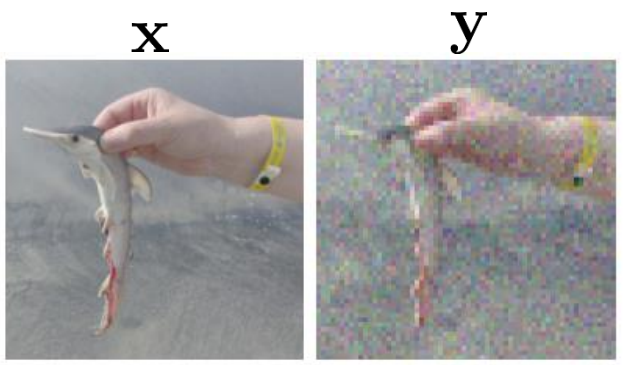
Find:

$\mathbf{x} \in \mathbb{R}^{n}$ The true signal

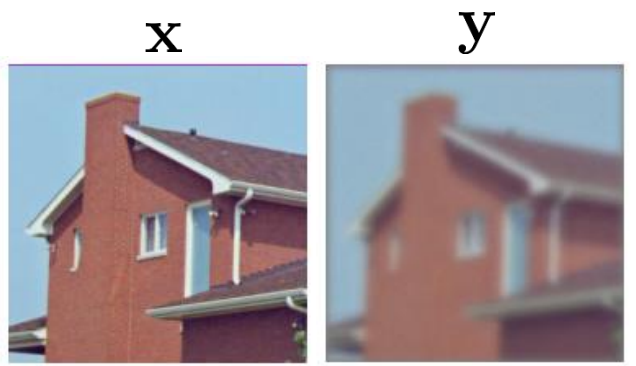
Ill-posed: There are infinitely many solutions …



colorization



super-resolution



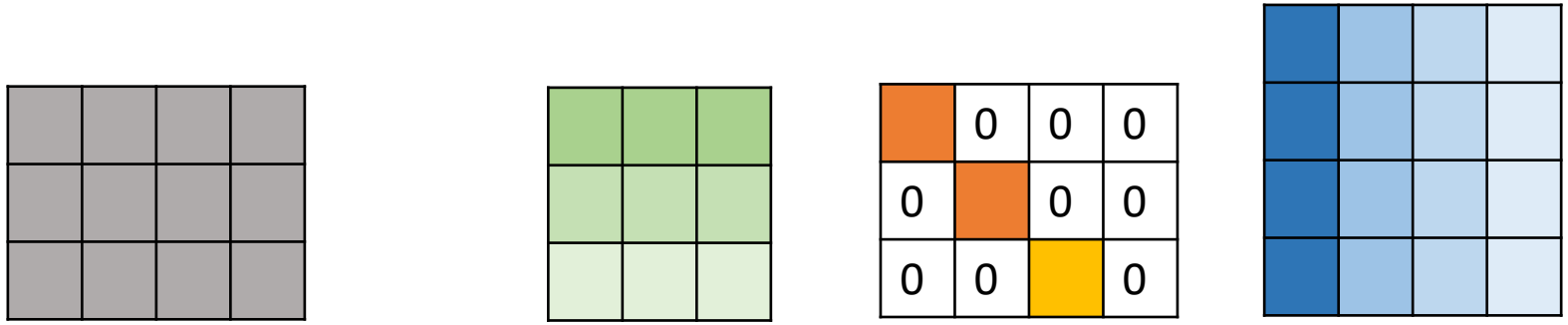
deblurring

[Kawar et al., NeurIPS 2022] Denoising Diffusion Restoration Models

Linear Inverse Problem (II)

- Denoising Diffusion Restoration Models (DDRM)

$\mathbf{H}$ can be diagonalized



$$\mathbf{H} = \mathbf{U} \times \mathbf{\Sigma} \times \mathbf{V}^T$$

$$\mathbf{U}^\top \mathbf{y} = \mathbf{\Sigma}(\mathbf{V}^\top \mathbf{x}) + \mathbf{U}^\top \mathbf{z}$$

- Linear inverse problem can be transformed into “denoising and inpainting” in spectral space

[Kawar et al., NeurIPS 2022] Denoising Diffusion Restoration Models

Linear Blind Inverse Problem (I)

$$\mathbf{y} = \mathbf{H}_{\varphi}(\mathbf{x}) + \mathbf{z}$$

Given:

$\mathbf{H}_{\varphi} : \mathbb{R}^n \to \mathbb{R}^m$ An unknown linear degradation matrix

$\mathbf{y} \in \mathbb{R}^m$ Measurement

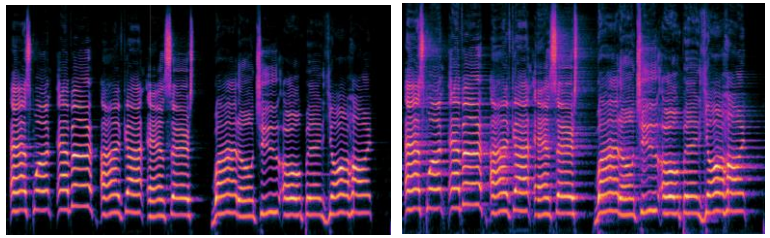
$\mathbf{z} \sim \mathcal{N}(0, \sigma_{\mathbf{y}}^2 \mathbf{I})$ Gaussian noise with known variance

Find:

$\mathbf{x} \in \mathbb{R}^n$ The true signal

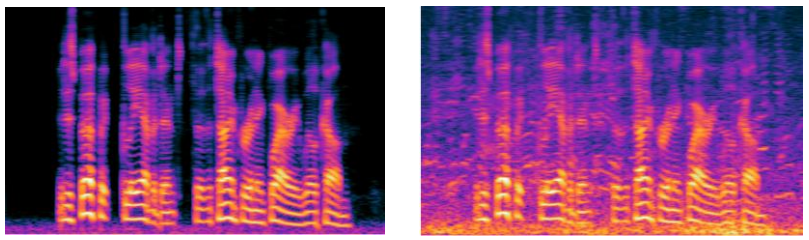
Ill-posed: There are infinitely many solutions …

$\mathbf{x}$ $\mathbf{y}$



Vocal dereverberation

$\mathbf{x}$ $\mathbf{y}$



Speech enhancement

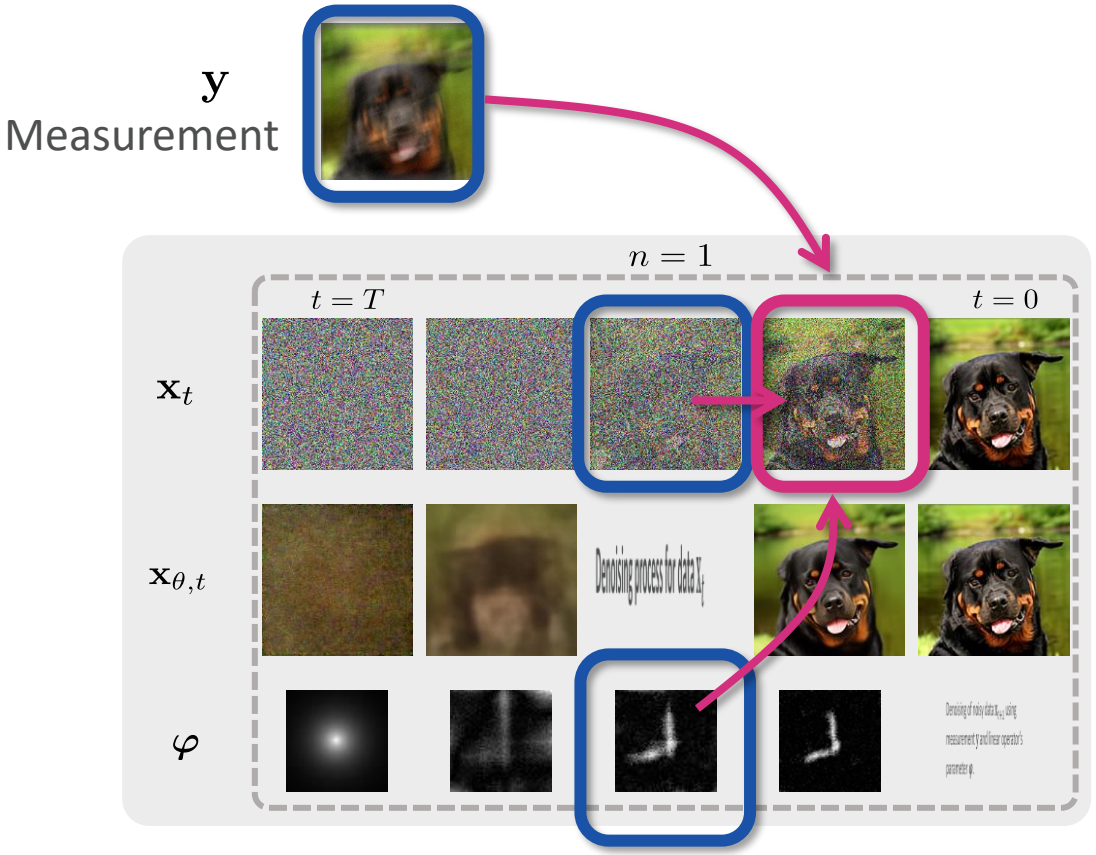
$\mathbf{x}$ $\mathbf{y}$

deblurring

[Murata et al., ICML 2023] GibbsDDRM: A Partially Collapsed Gibbs Sampler for Solving Blind Inverse Problems with Denoising Diffusion Restoration

Linear Blind Inverse Problem (II)

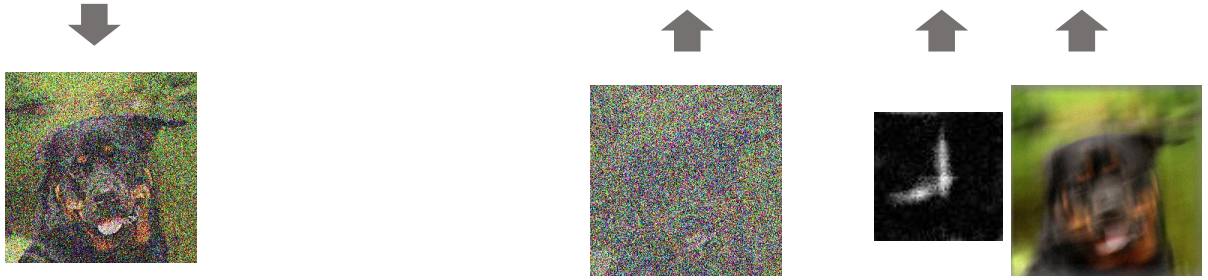
- Simultaneous estimation of data (sampling) and linear operator parameters



Denoising process for data $\mathbf{x}_t$

Denoising of noisy data $\mathbf{x}_{t+1}$ using measurement $\mathbf{y}$ and linear operator's parameter $\boldsymbol{\varphi}$.

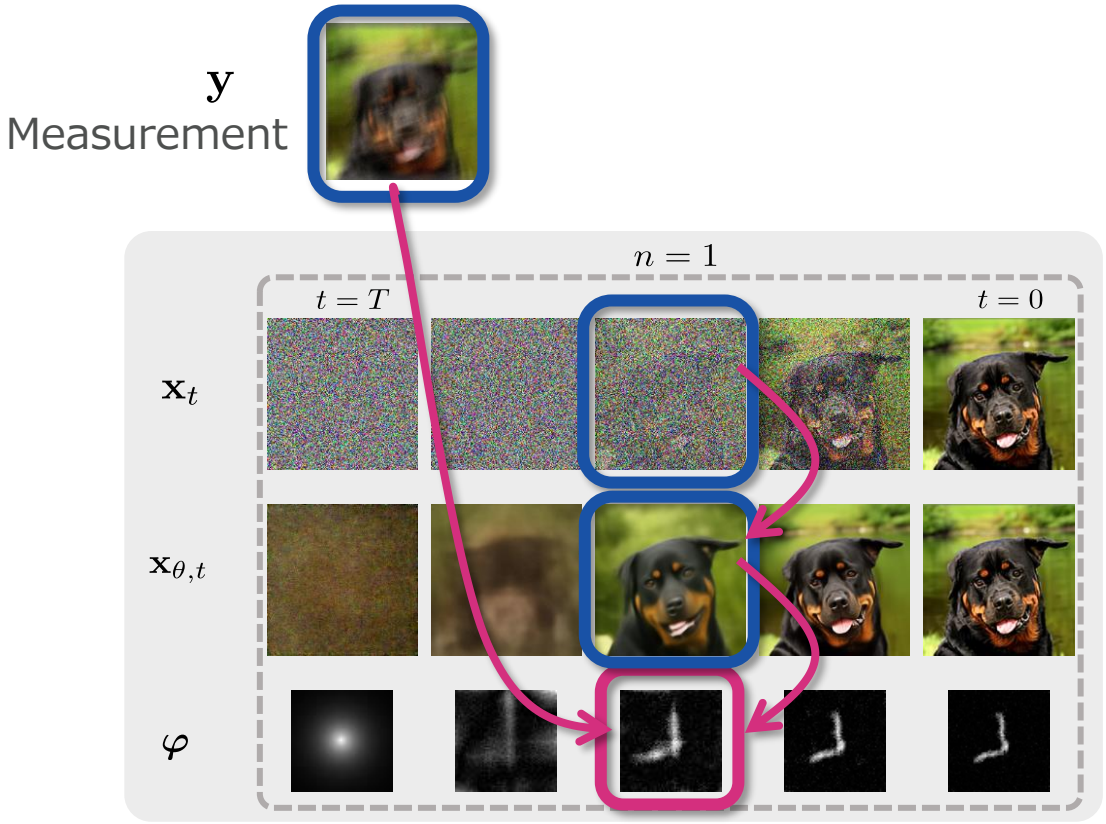
$$\mathbf{x}_t \sim p_\theta\left(\mathbf{x}_t \middle| \mathbf{x}_{t+1}, \boldsymbol{\varphi}, \mathbf{y}\right)$$



[Murata et al., ICML 2023] GibbsDDRM: A Partially Collapsed Gibbs Sampler for Solving Blind Inverse Problems with Denoising Diffusion Restoration

Linear Blind Inverse Problem (III)

- Simultaneous estimation of data (sampling) and linear operator parameters



Sampling of linear operator's parameter φ

Parameter φ is estimated using measurement $\mathbf{y}$ and clean data estimate $\mathbf{x}_{\theta,t}$

$$\nabla_{\varphi} \log p(\varphi | \mathbf{x}_{t:T}, \mathbf{y}) \simeq -\frac{1}{2\sigma_{\mathbf{y}}^2} \nabla_{\varphi} \|\mathbf{y} - \mathbf{H}_{\varphi} \mathbf{x}_{\theta,t}\|_2^2 + \nabla_{\varphi} \log p(\varphi)$$

Prior on φ

Measurement $\mathbf{y}$

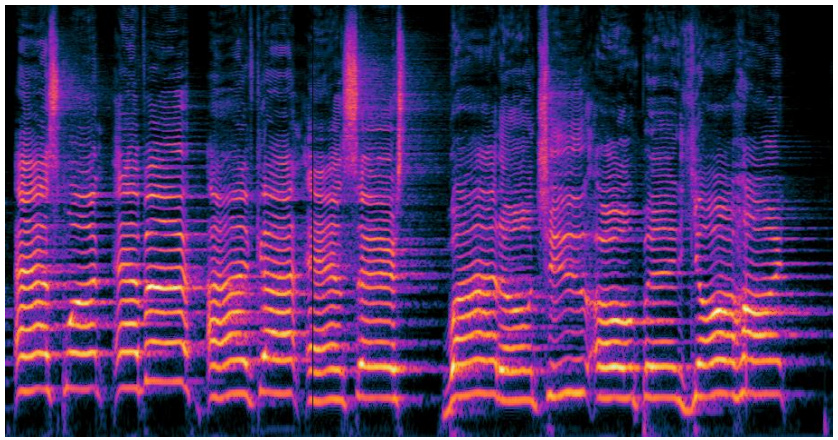
Estimation of clean data $\mathbf{x}_{\theta,t}$

[Murata et al., ICML 2023] GibbsDDRM: A Partially Collapsed Gibbs Sampler for Solving Blind Inverse Problems with Denoising Diffusion Restoration

Application: Vocal dereverberation

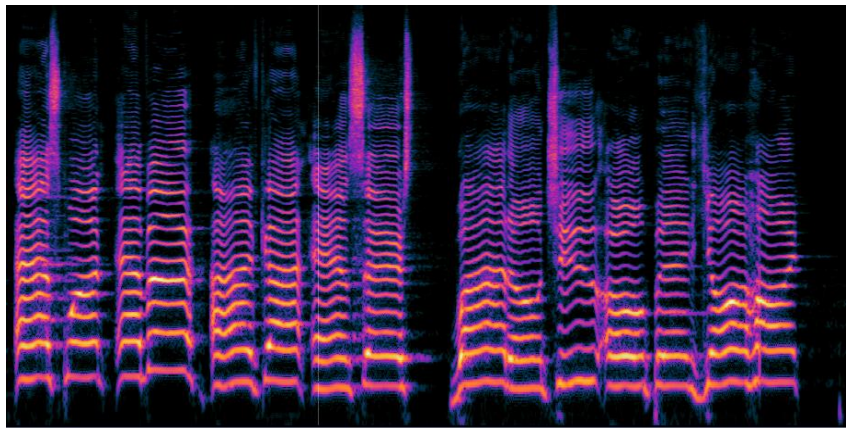
- Task**:** Restore the **dry vocal** from **the wet (reverberant) vocal**
- Pre-trained model: **Diffusion model trained on dry vocal data**
- Application: **Remix** of recorded contents

Input



(Reverberant vocal)

Output



(Clean vocal retrieved by DiffDereverb)

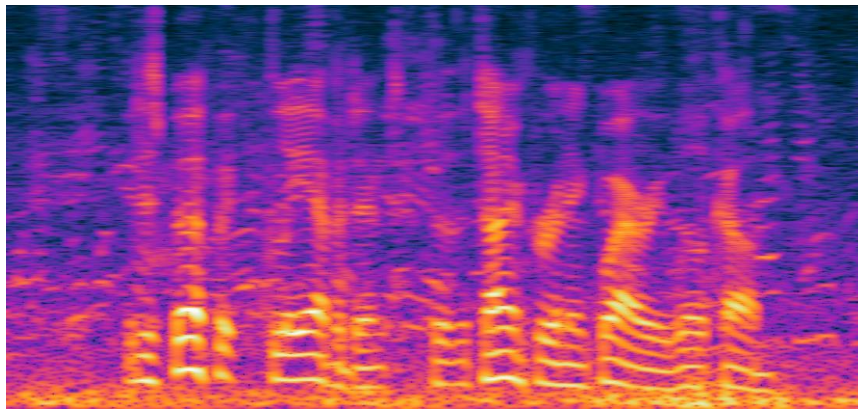
Audio samples

[Saito et al., ICASSP23] "Unsupervised vocal dereverberation with diffusion-based generative models"

Application: Speech Enhancement

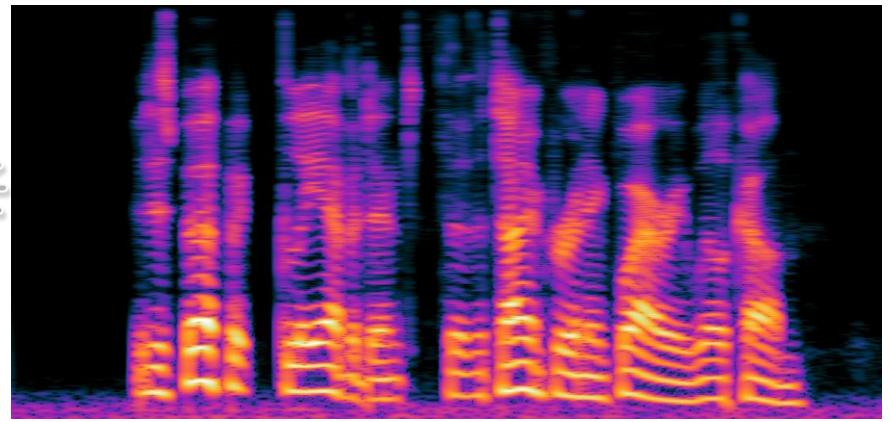
- Task: Extract **clean speech** from the **noisy speech**
- Pre-trained model: Diffusion model trained on clean speech data
- Application: Tele-communication, video post-production

Input



(Noisy speech)

Output



(Clean speech retrieved by Diffiner)

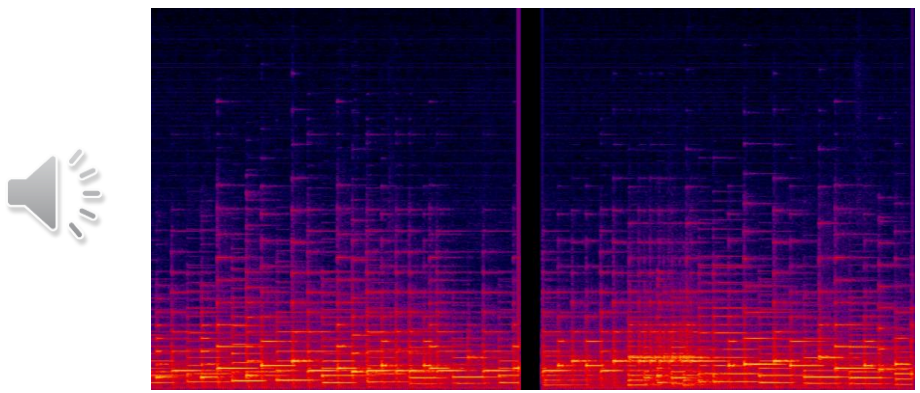
[Sawata et al., Interspeech'23] "A versatile diffusion-based generative refiner for speech enhancement"

Application: Audio Inpainting

- Task: Inpainting audio from the **masked** input
- Pre-trained model: Diffusion model trained on clean audio data
- Application: Audio editing

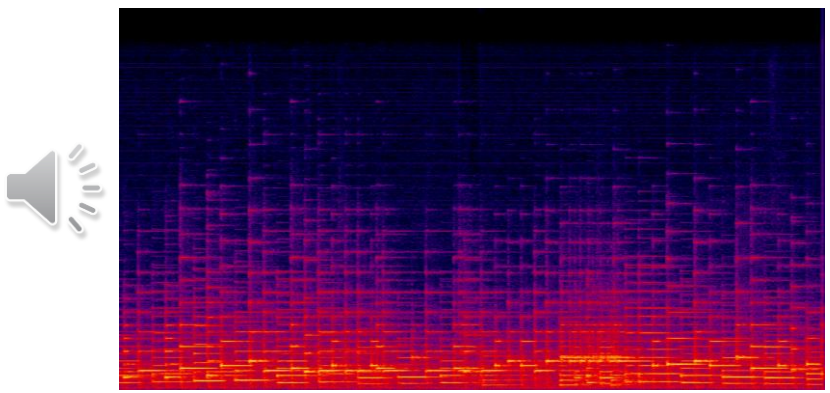
$$\mathcal{A}(\mathbf{x}) = (1 - \mathbb{1}_{[t_{\text{start}}, t_{\text{end}}]})\mathbf{x}$$

Input



Masked audio

Output



generation

[Moliner et al., ICASSP 2023] Solving audio inverse problems with a diffusion model

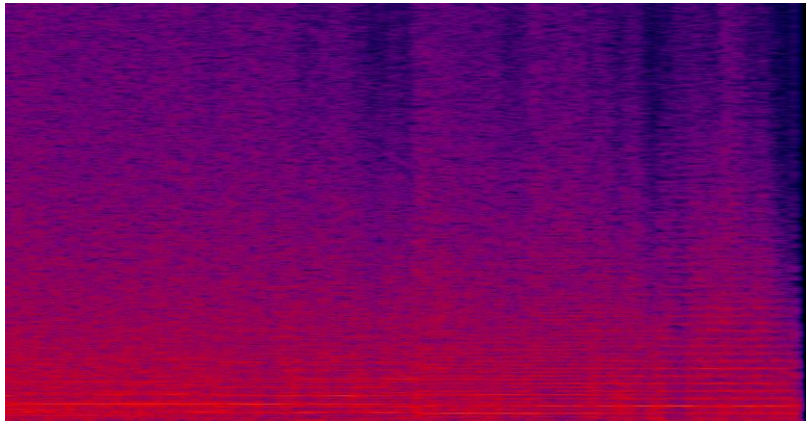
Application: Audio Declipping

- Task: Cancelling the distortion in audio signal that has been hard-clipped
- Pre-trained model: Diffusion model trained on clean audio data
- Application: Audio editing

$$\mathcal{A}(\mathbf{x}) = \frac{|\mathbf{x} + c| - |\mathbf{x} - c|}{2}$$

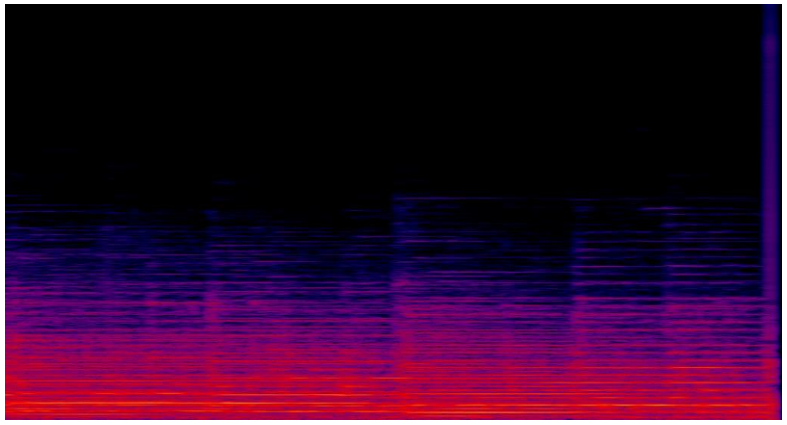
Original audio

Input



Clipped audio

Output



generation

[Hernandez-Olivan, et al., ICASSP 2024] Vrdmg: Vocal restoration via diffusion posterior sampling with multiple guidance

[Moliner et al., ICASSP 2023] Solving audio inverse problems with a diffusion model

NEW TREND

New Trend

1. Audio-Visual Generation / Sounding Video Generation

2. Stem-based Music Generation

3. Memorization

Audio-Visual Generation / Sounding Video Generation

2023 → 2024 → 2025

Image Generation
DALLE3

DALL·E 3 · An expressive oil painting of a chocolate chip cookie being dipped in a glass of milk, depicted as an... +

Video Generation
Sora

Sounding Video Generation ?

Unified vs Composable Modeling

1) Unified Modeling

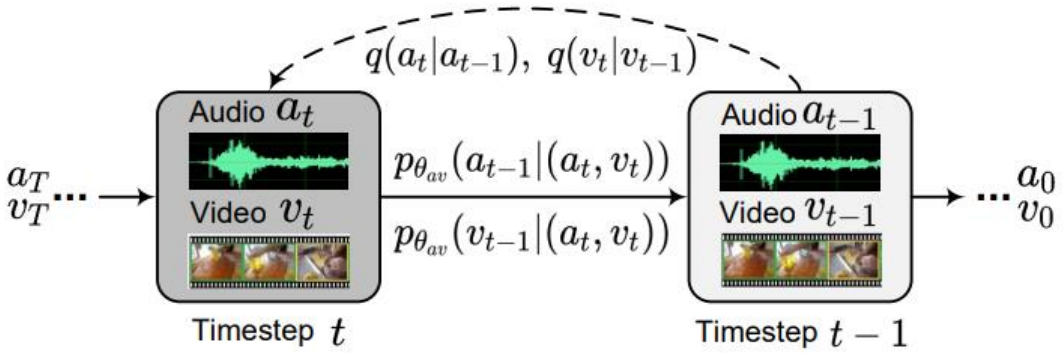
(e.g., MM-Diffusion [Ruan+ CVPR2023])

[Pros]

Multimodal alignment can be explicitly learned -> **better in alignment**

[Cons]

A lot of pair data are needed



2) Composable Modeling

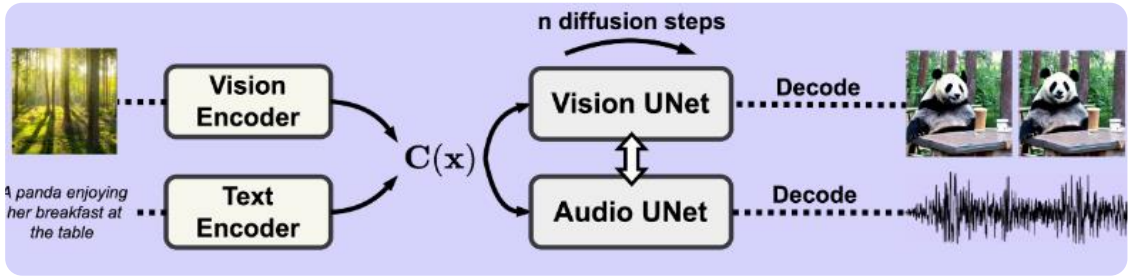
(e.g., CoDi [Tang+ NeurIPS2023])

[Pros]

Off-the-shelf single-modal pretrained models can be incorporated in the framework -> **better in quality**

[Cons]

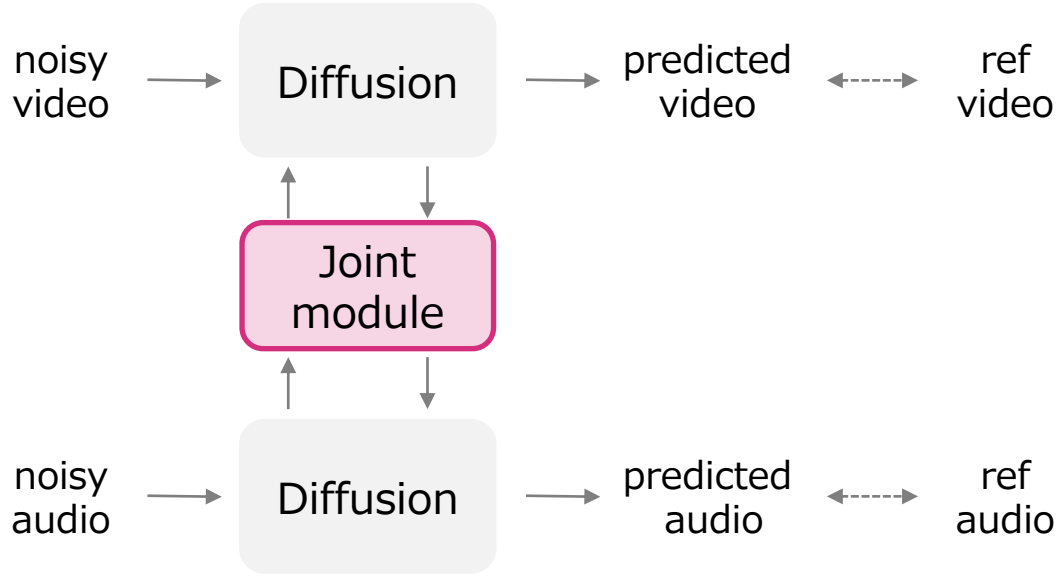
Bridging modalities need to be learned w/ pair data



Composable Modeling

Finetuning

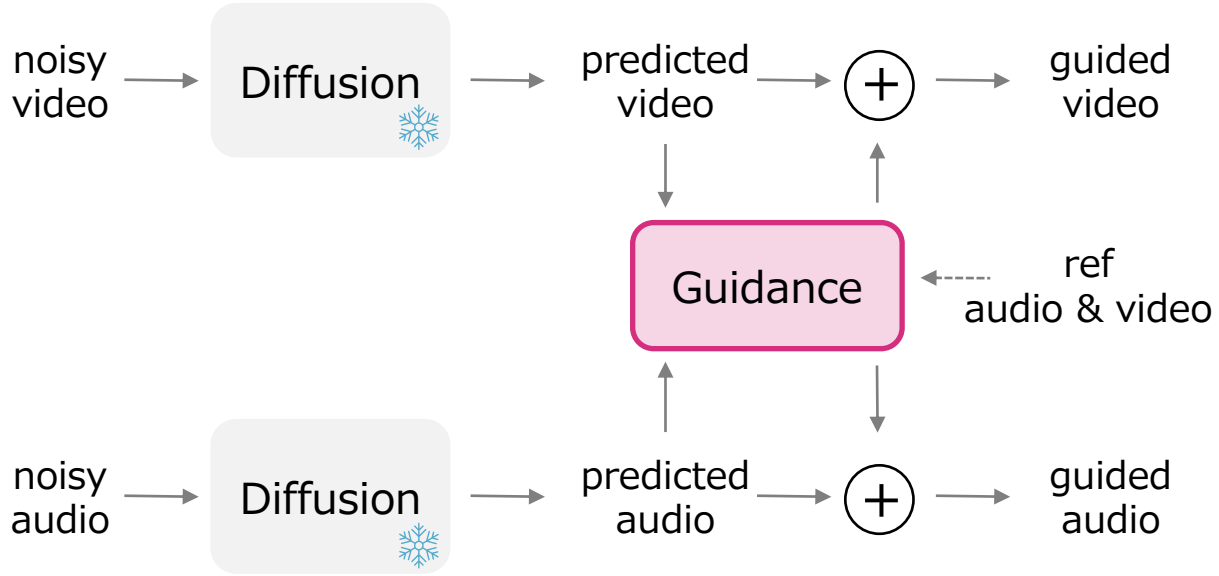
[Tang+ NeurIPS2023][Ishii+ ECCV24 AVGenL]



- Finetune pretrained model to predict joint distribution p(x, y)
- Depends on model architectures

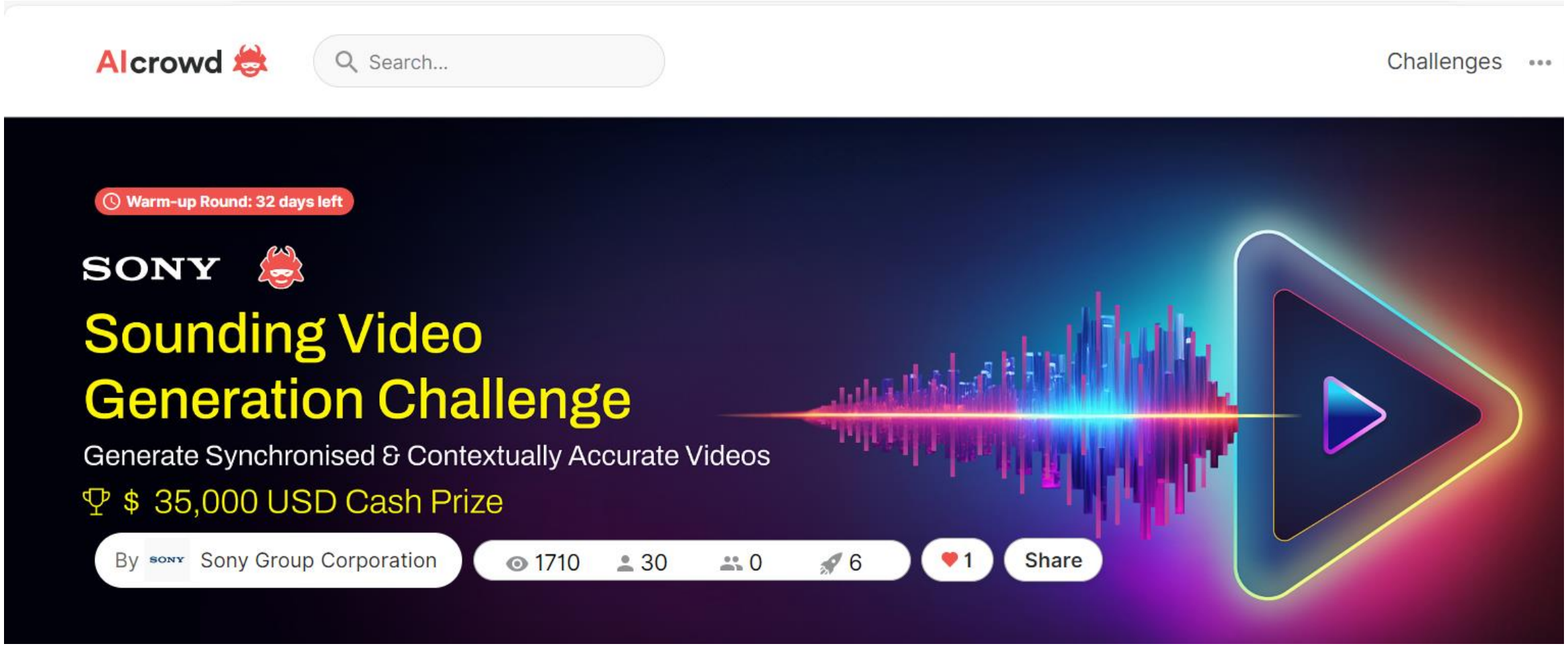
Guidance

[Xing+ CVPR2024][Hayakawa+ 2024]



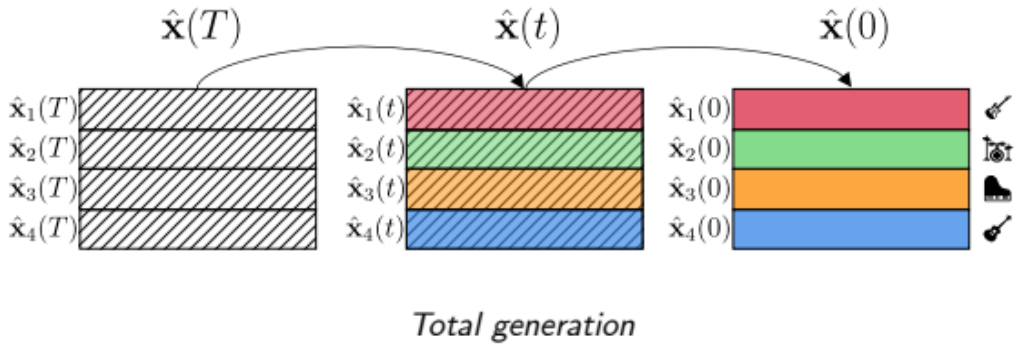
- Refine prediction from pretrained model towards joint distribution p(x, y)
- Model architecture agnostic

Academic Challenge on Sounding Video Generation [URL]

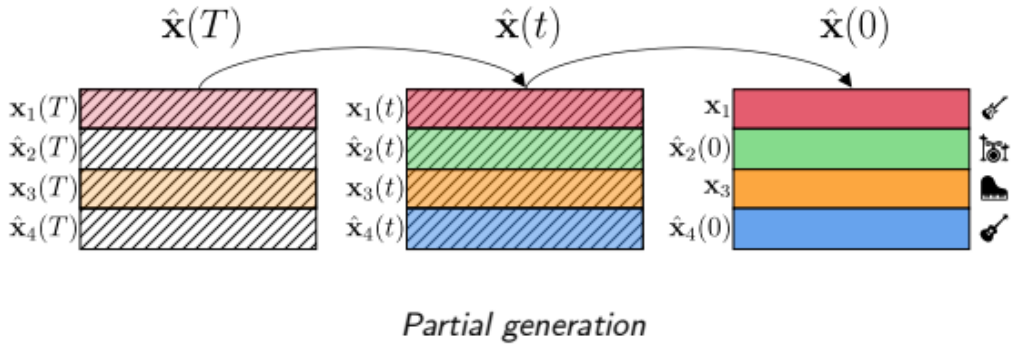


Multi-Source Diffusion [Mariani+ ICLR2024]

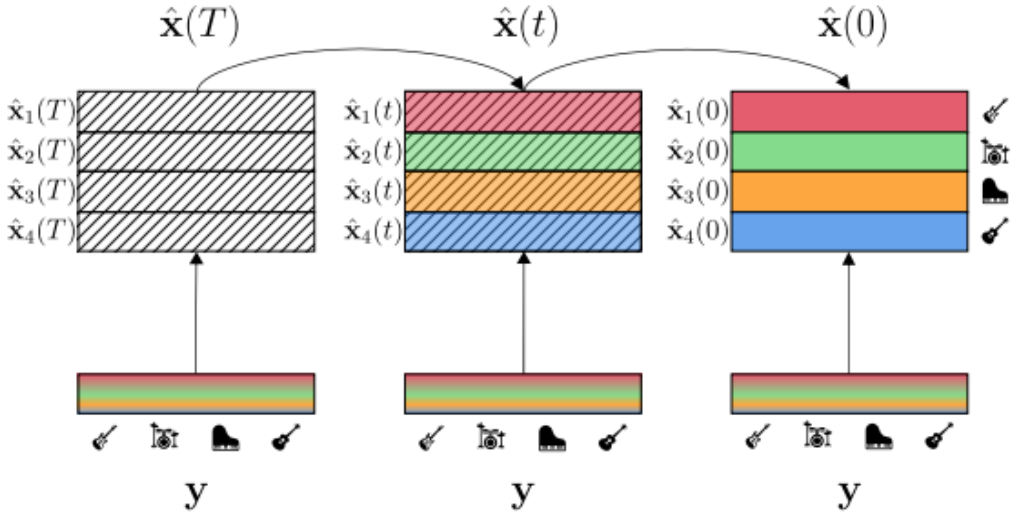
The model can achieve total generation, partial generation, and source separation



Total generation



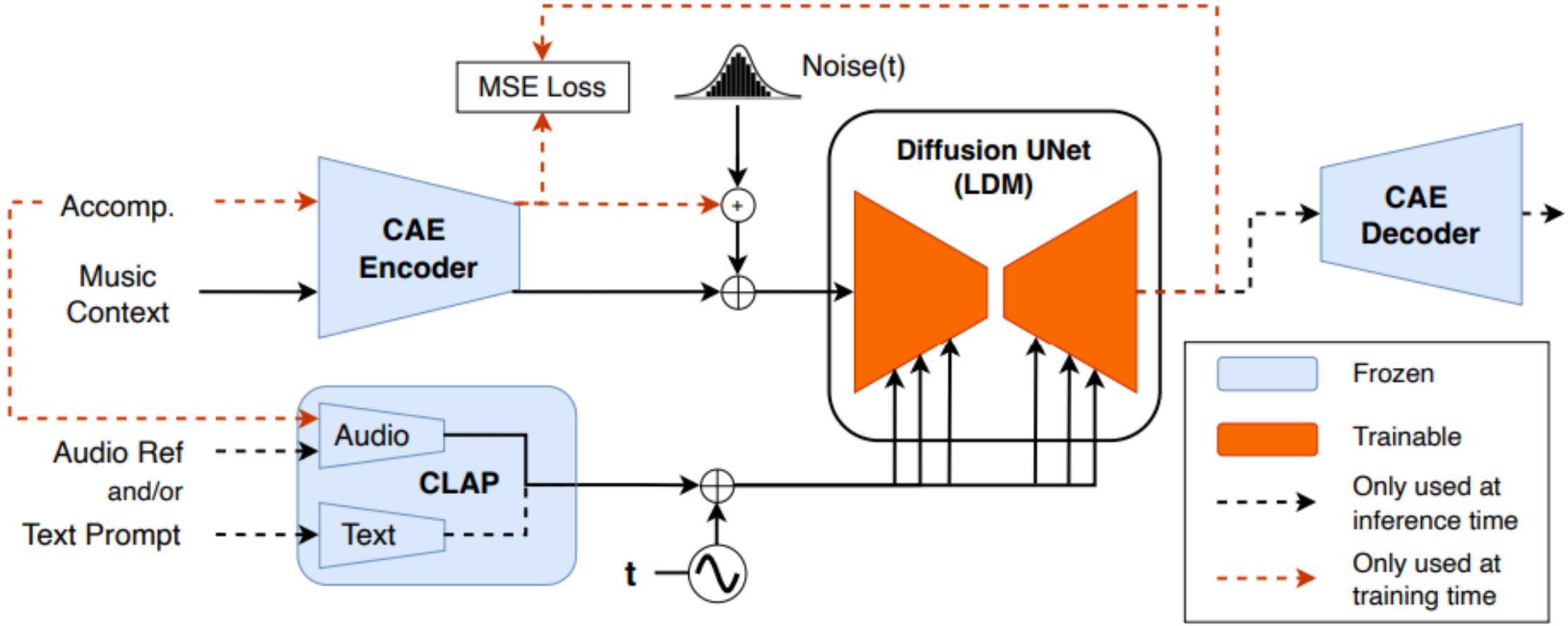
Partial generation



Source separation

Single Instrument Accompaniment Generation [Nistal+ ISMIR2024]

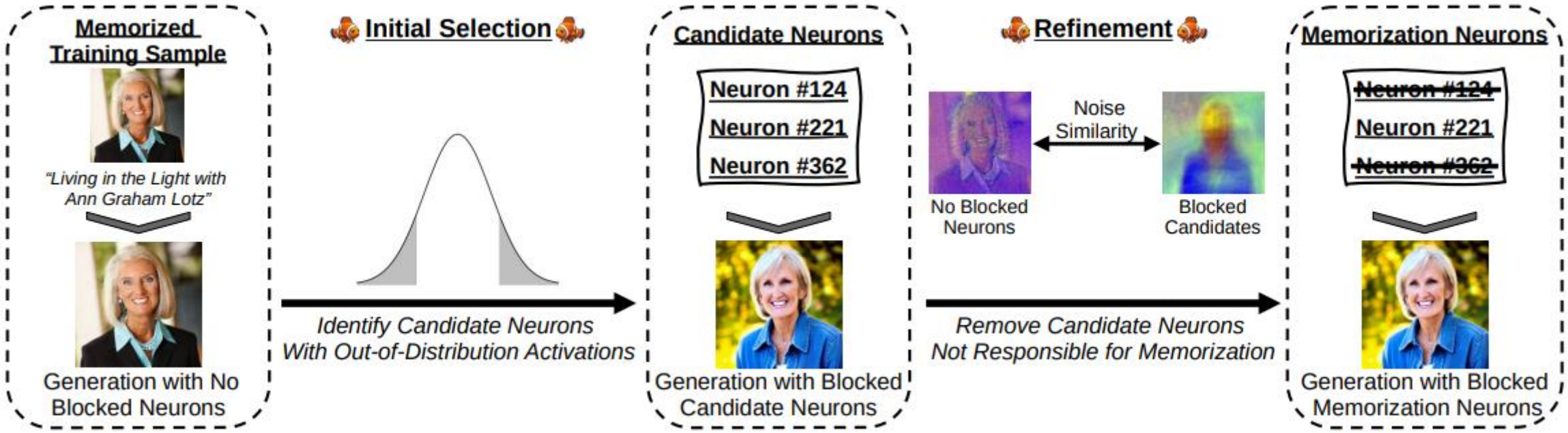
Diff-A-Riff: LDM designed to generate high-quality instrumental accompaniments adaptable to any musical context



Memorization of Diffusion Models [Hintersdorf+ NeurIPS2024]

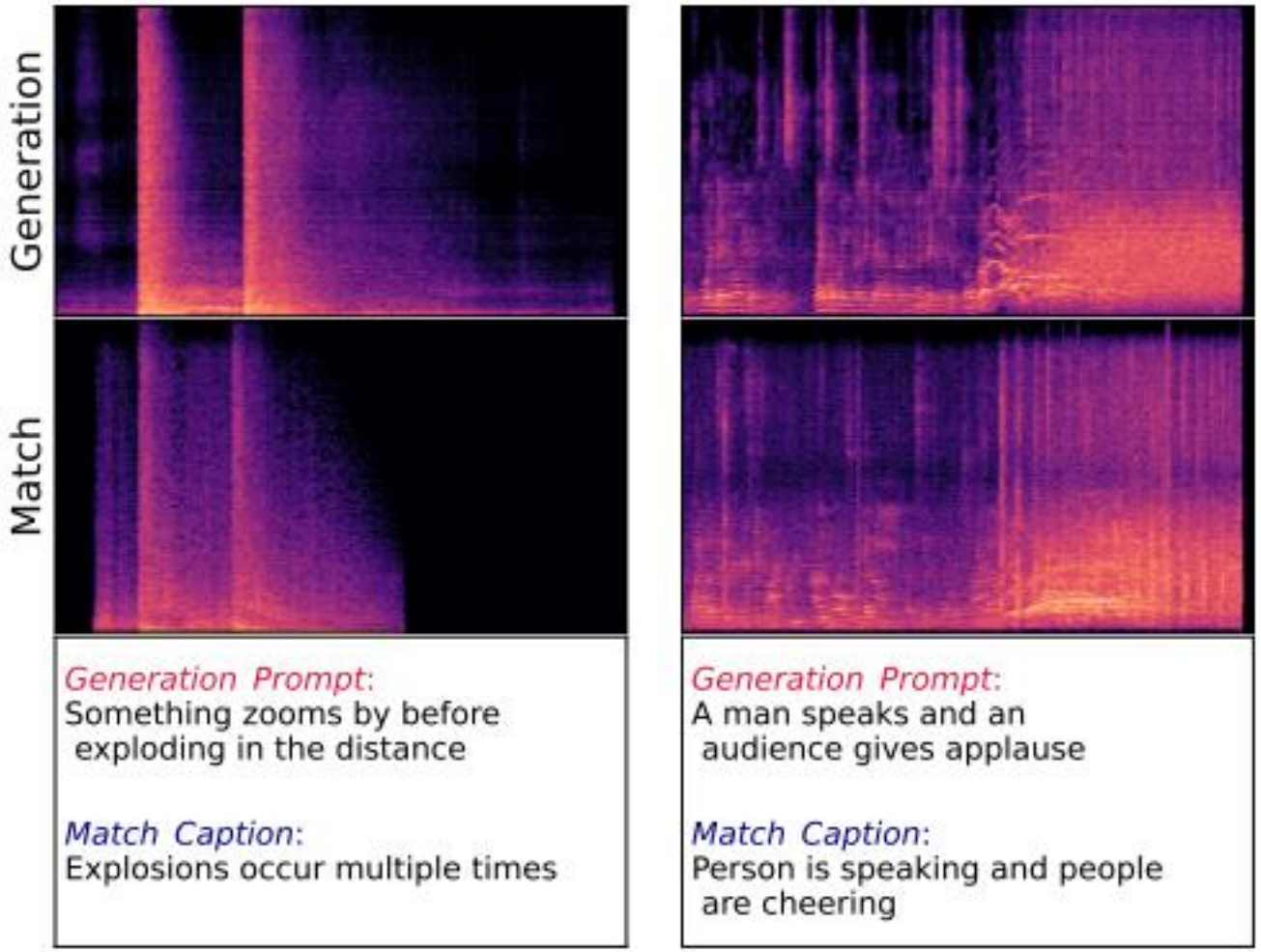
Single neurons are responsible for memorizing training samples

- ✓ Verbatim memorization: replicating the training image exactly
- ✓ Template memorization: reproducing the general composition of the training image while having some non-semantic variations at fixed image positions



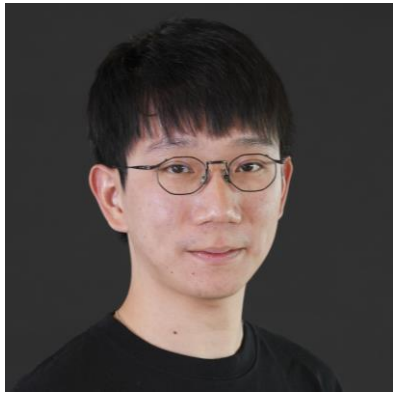
Memorization Problem in Audio Diffusion [Bralios+ ICASSP2024]

Samples generated using TANGO and their top matches in the AudioCaps



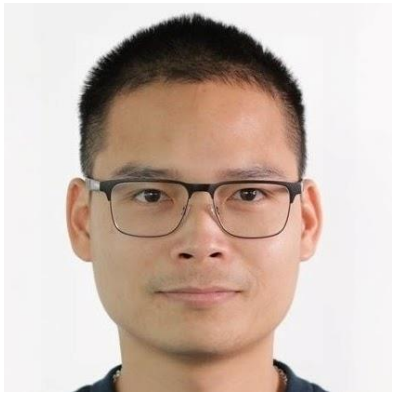
More work from
CreativeAI @Sony

MEET OUR SPEAKERS



Chieh-Hsin (Jesse) Lai

@JCJesseLai



Bac Nguyen

@nguyencongbacbk

Koichi Saito

@Koichi__Saito

Yuki Mitsufuji

@mittu1204

THANK YOU