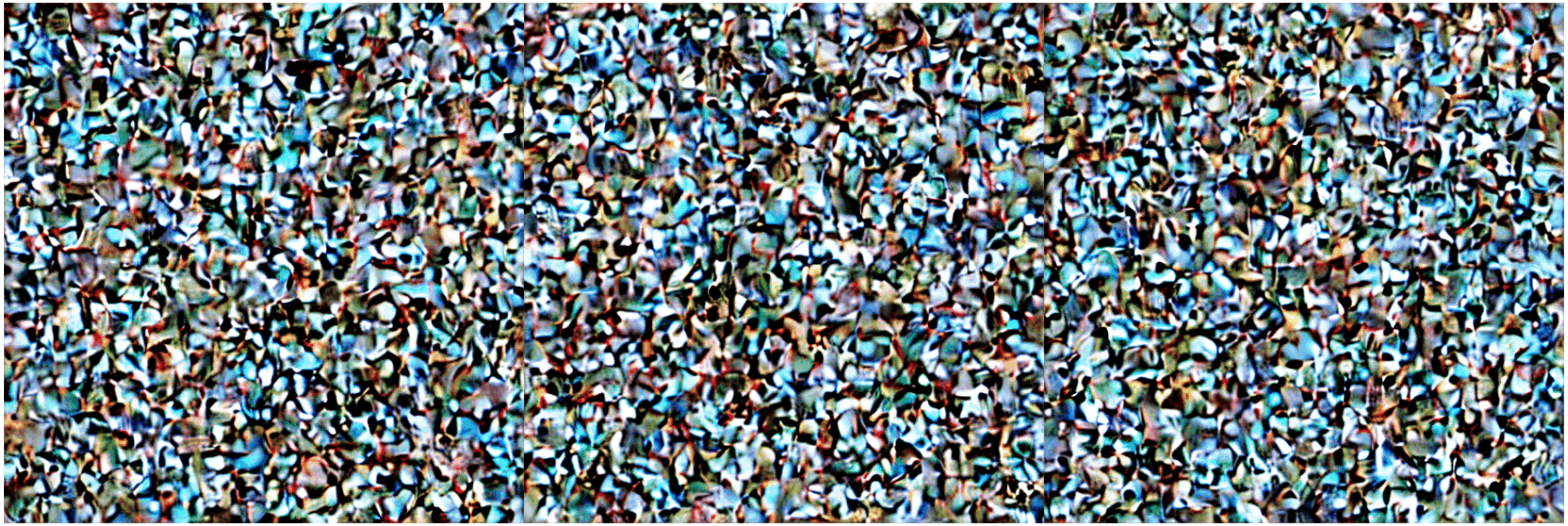
Gradient Guidance for Diffusion Models: An Optimization Perspective

NeurIPS 2024

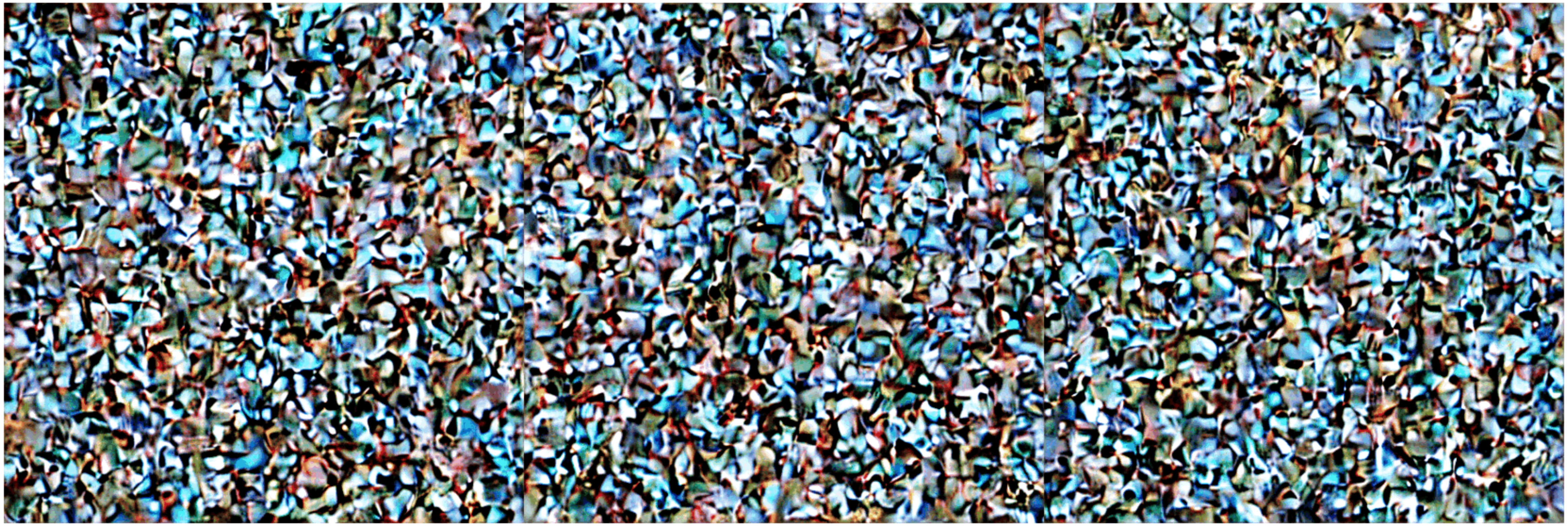
Yingqing Guo*, Hui Yuan*, Yukang Yang, Minshuo Chen, Mengdi Wang (* equal contribution)

ECE @ Princeton University

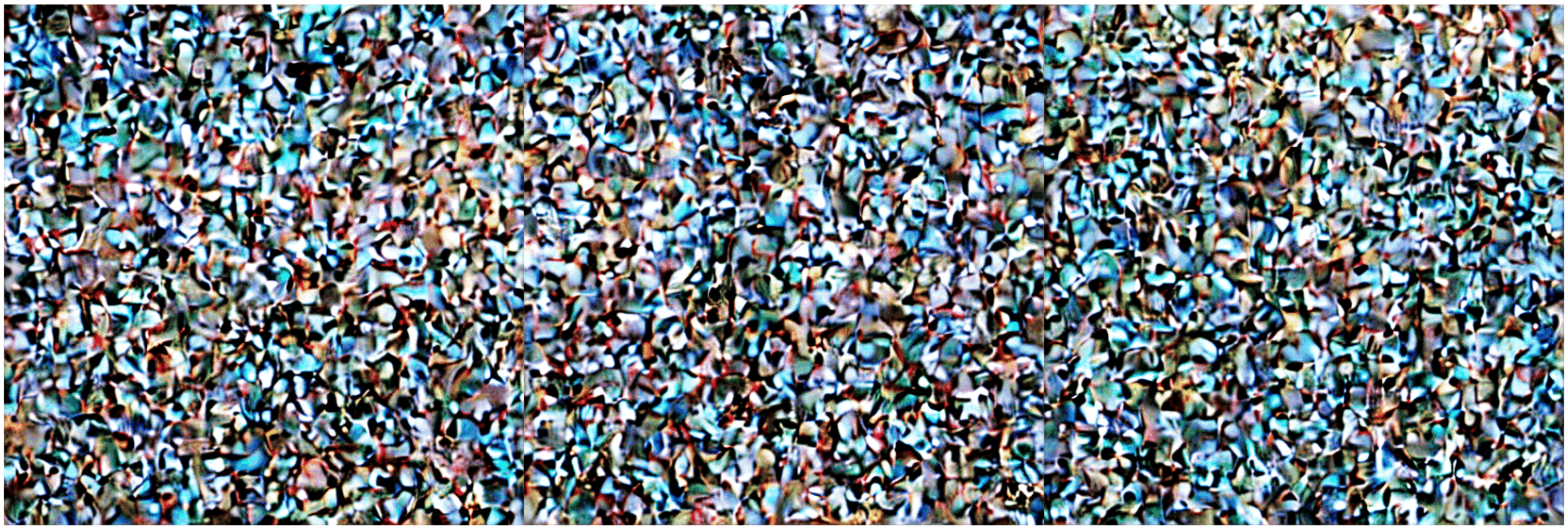
Diffusion model



Diffusion model



Diffusion model

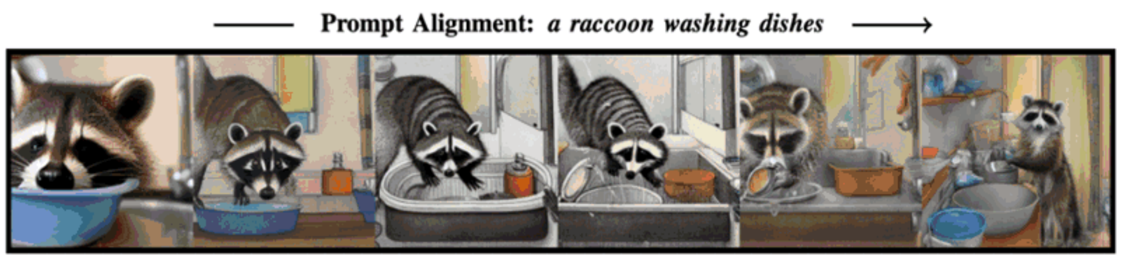


Diffusion model

Diffusion model

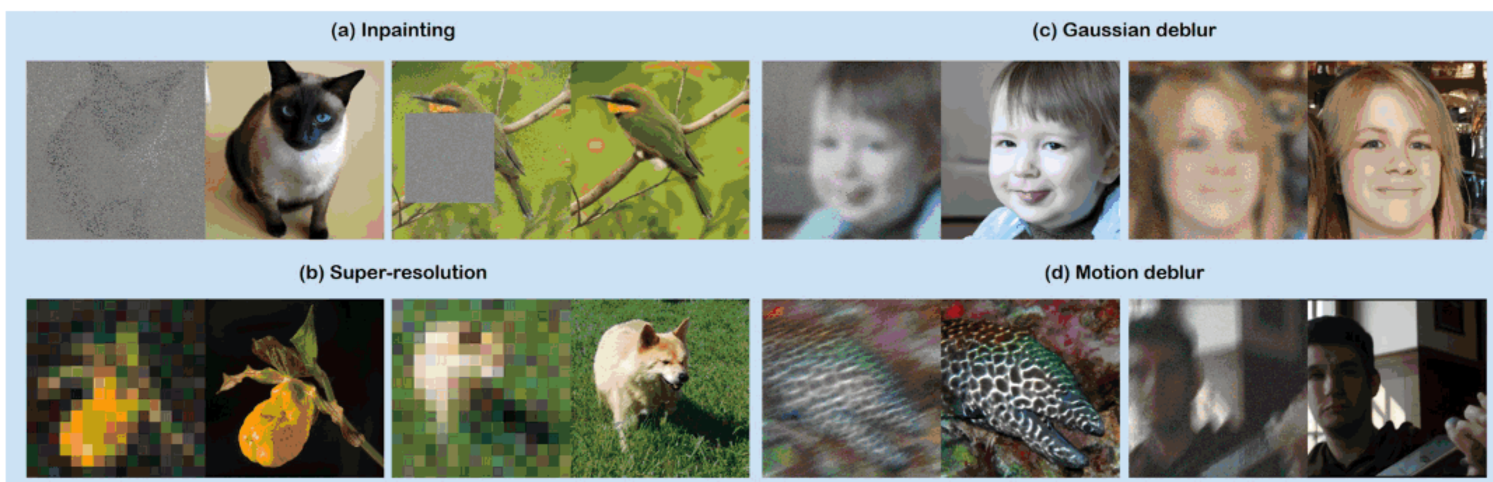
Adapting Pre-trained Model to Downstream Tasks

- **Fine-tuning:** *Prompt alignment, Aesthetic quality*



-- Credit "Training Diffusion Models with Reinforcement Learning", K. Black et al., 2023

- **Guidance:** *Inverse problem*

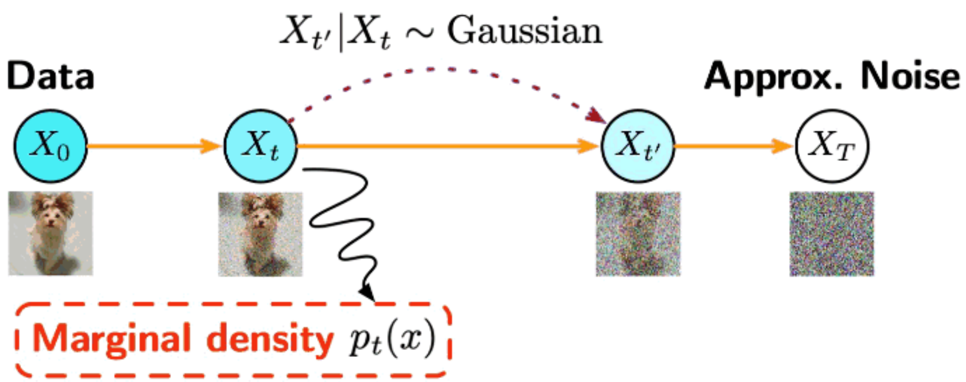


-- Credit "Diffusion Posterior Sampling for General Noisy Inverse Problems", H. Chung et al., 2022

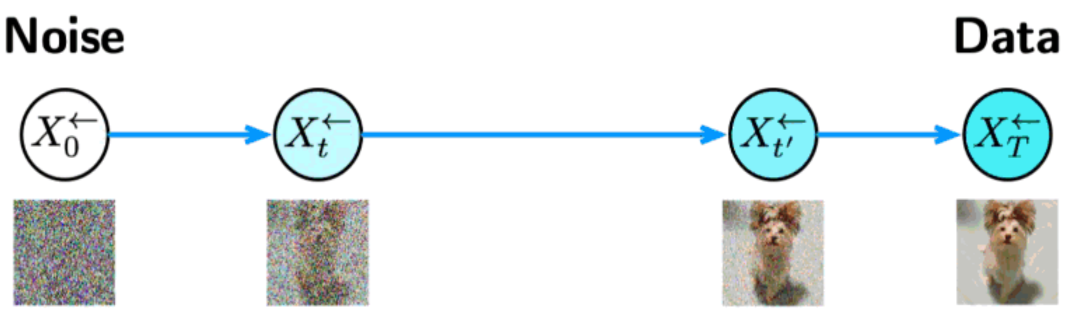
Guided Generation with Diffusion

Conditional DM is an essential way

- **Conditional Diffusion Model**



$$\mathrm{d}X_t^{\leftarrow} = \left[\frac{1}{2}X_t^{\leftarrow} + \nabla \log p_{T-t}(X_t^{\leftarrow})\right]\mathrm{d}t + \mathrm{d}\overline{W}_t$$



To include conditioning

$y = \mathrm{dog}$

$$\mathrm{d}X_t^{\leftarrow} = \left[\frac{1}{2}X_t^{\leftarrow} + \nabla \log p_{T-t}(X_t^{\leftarrow} \mid y)\right]\mathrm{d}t + \mathrm{d}\overline{W}_t$$

Conditional Denoiser

Adding Guidance to the Generation Process

- Conditioned generation

$$\mathrm{d}X_t^{\leftarrow} = \left[\frac{1}{2}X_t^{\leftarrow} + s_\theta(X_t^{\leftarrow}, y, T-t)\right]\mathrm{d}t + \mathrm{d}\overline{W}_t$$

- Conditional score decomposition

$$\nabla_{x_t} \log p_t(x_t \mid y) = \underbrace{\nabla \log p_t(x_t)}_{\text{est. by } s_\theta(x_t, t)} + \underbrace{\nabla_{x_t} \log p_t(y \mid x_t)}_{\text{to be est. by guidance}}.$$

Discrete y

Classifier guidance

Classifier

$$\mathtt{G}(x_t, t) = \nabla \log c_t(y|x_t)$$

- Adding guidance

$$\mathrm{d}X_t^{\leftarrow} = \left[\frac{1}{2}X_t^{\leftarrow} + s_\theta(X_t^{\leftarrow}, T-t) + \mathtt{G}(X_t^{\leftarrow}, t)\right]\mathrm{d}t + \mathrm{d}\overline{W}_t$$

Adding Guidance to the Generation Process

- **Consider the problem:** we want to generate x with high y=f(x)

- Adding guidance

$$\mathrm{d}X_t^{\leftarrow} = \left[\frac{1}{2}X_t^{\leftarrow} + s_\theta(X_t^{\leftarrow}, T-t) + \mathtt{G}(X_t^{\leftarrow}, t)\right]\mathrm{d}t + \mathrm{d}\overline{W}_t$$

- Can we choose guidance to be the gradient?

$$G(X,t) \propto \nabla f$$

A toy modal: Gaussian data + Linear objective

- Gaussian Data, Linear Reward model: $y = g^\top x + \epsilon$
- Conditional score function

$$\nabla \log p_t(x_t|y) = \nabla \log p_t(x_t) + \beta(t) \left[y - g^\top \mathbb{E}[x_0|x_t]\right] \cdot g$$

A toy modal: Gaussian data + Linear objective

- Gaussian Data, Linear Reward model: $y = g^\top x + \epsilon$
- Conditional score function

$$\nabla \log p_t(x_t|y) = \nabla \log p_t(x_t) + \beta(t) \left[y - g^\top \mathbb{E}[x_0|x_t]\right] \cdot g$$

Pre-trained score

A toy modal: Gaussian data + Linear objective

- Gaussian Data, Linear Reward model: $y = g^\top x + \epsilon$
- Conditional score function

$$\nabla \log p_t(x_t|y) = \nabla \log p_t(x_t) + \beta(t)\left[y - g^\top \mathbb{E}[x_0|x_t]\right] \cdot g$$

Pre-trained score

Gradient guidance

A toy modal: Gaussian data + Linear objective

- Gaussian Data, Linear Reward model: $y = g^\top x + \epsilon$
- Conditional score function

$$\nabla \log p_t(x_t|y) = \nabla \log p_t(x_t) + \beta(t) \left[y - g^\top \mathbb{E}[x_0|x_t]\right] \cdot g$$

A toy modal: Gaussian data + Linear objective

- Gaussian Data, Linear Reward model: $y = g^\top x + \epsilon$
- Conditional score function

"Look-ahead" gap

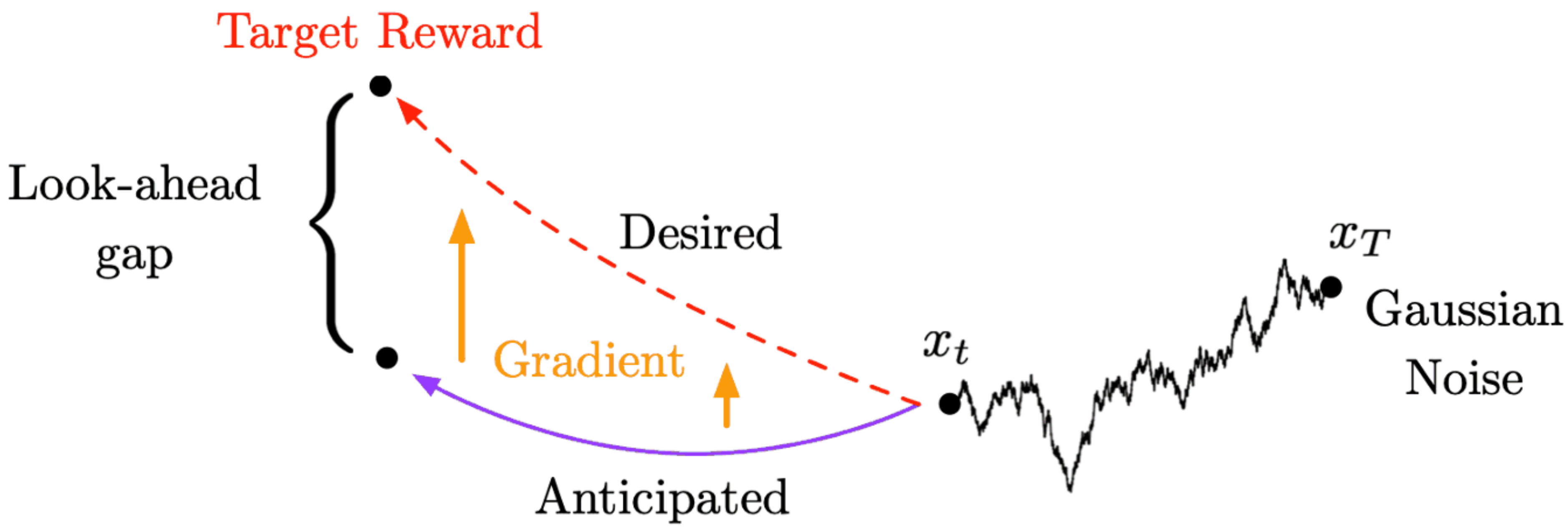
$$\nabla \log p_t(x_t|y) = \nabla \log p_t(x_t) + \beta(t) \boxed{[y - g^\top \mathbb{E}[x_0|x_t]]} \cdot g$$

A toy modal: Gaussian data + Linear objective

- Gaussian Data, Linear Reward model: $y = g^\top x + \epsilon$
- Conditional score function

"Look-ahead" gap

$$\nabla \log p_t(x_t|y) = \nabla \log p_t(x_t) + \beta(t)\left[y - g^\top \mathbb{E}[x_0|x_t]\right] \cdot g$$



Design the Gradient Guidance

$$\mathsf{G}(x_t, t) = \beta(t)\left(y - g^\top \hat{\mathbb{E}}[x_0|x_t]\right) \cdot g$$

- $\beta(t)$ is a tuning parameter, akin to a step size
- y is the target reward value
- g is a gradient vector of the reward function
- $\hat{\mathbb{E}}[x_0|x_t]$ is an estimator of $\mathbb{E}[x_0|x_t]$, via the pre-trained score:

$$\hat{\mathbb{E}}[x_0|x_t] = e^{t/2}(x_t + (1 - e^{-t}) \cdot \text{pre-trained-score})$$

(Tweedie's formula (Efron, 2011))

Design the Gradient Guidance

$$\mathsf{G}(x_t, t) = \beta(t) \left(y - g^\top \hat{\mathbb{E}}[x_0 | x_t] \right) \cdot g$$

Local linearization

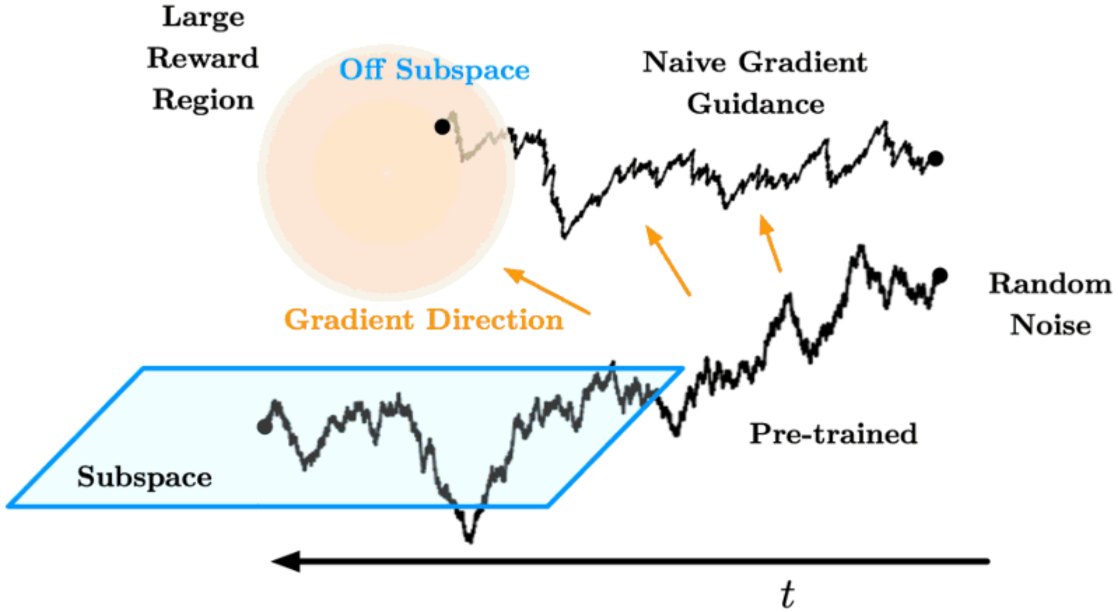
- $\beta(t)$ is a tuning parameter, akin to a step size
- y is the target reward value
- g is a gradient vector of the reward function
- $\hat{\mathbb{E}}[x_0|x_t]$ is an estimator of $\mathbb{E}[x_0|x_t]$, via the pre-trained score:

$$\hat{\mathbb{E}}[x_0|x_t] = e^{t/2}(x_t + (1 - e^{-t}) \cdot \text{pre-trained-score})$$

(Tweedie's formula (Efron, 2011))

Problematic Naive Gradient

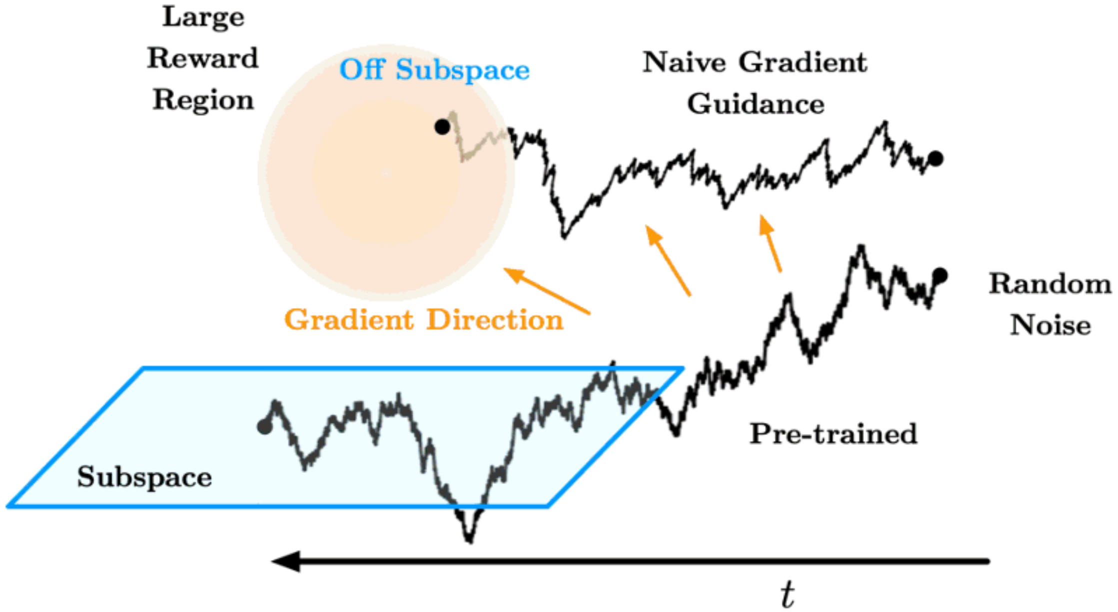
- No! Naïve gradient would **jeopardize the latent data structure**



$$G_{loss}(x_t, t) := -\beta(t) \cdot \nabla_{x_t} \left(y - g^\top \mathbb{E}[x_0 | x_t]\right)^2$$

Problematic Naive Gradient

- No! Naïve gradient would **jeopardize the latent data structure**

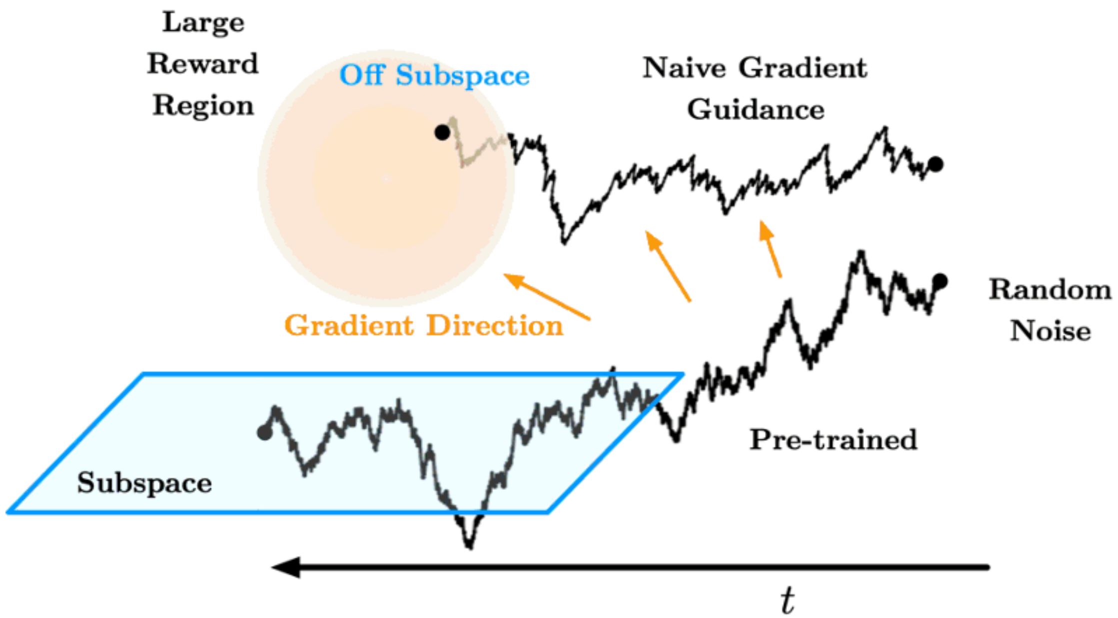


- Let's use the gradient of a lookahead loss

$$G_{loss}(x_t, t) := -\beta(t) \cdot \nabla_{x_t} \left(y - g^\top \mathbb{E}[x_0|x_t]\right)^2$$

Problematic Naive Gradient

- No! Naïve gradient would **jeopardize the latent data structure**



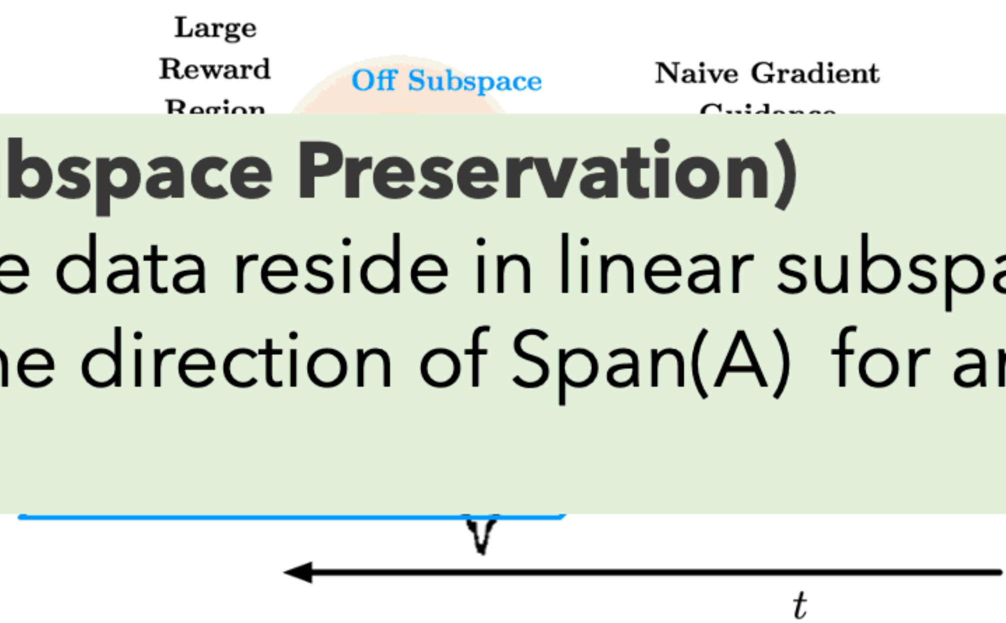
- Let's use the gradient of a lookahead loss

Use pretrained score

$$G_{loss}(x_t, t) := -\beta(t) \cdot \nabla_{x_t} \left(y - g^\top \mathbb{E}[x_0|x_t]\right)^2$$

Problematic Naive Gradient

- No! Naïve gradient would **jeopardize the latent data structure**



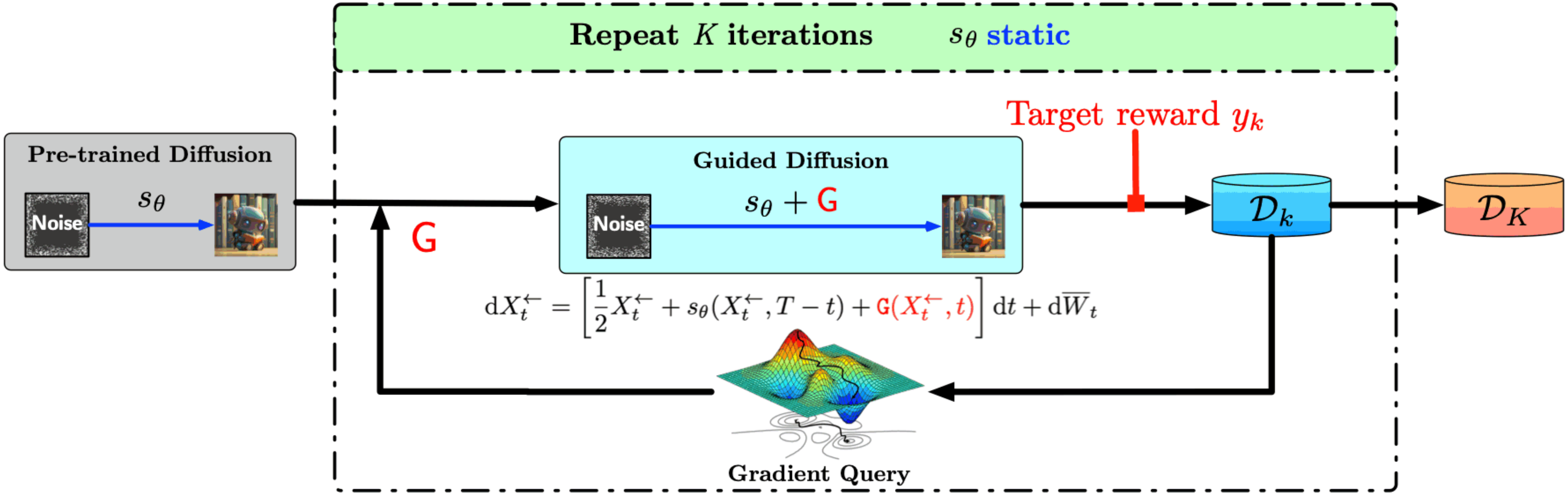
Theorem(Subspace Preservation)
Assuming the data reside in linear subspace x=Au, Gloss aligns with the direction of Span(A) for any data distribution.

- Let's use the gradient of a lookahead loss

Use pretrained score

$$G_{loss}(x_t, t) := -\beta(t) \cdot \nabla_{x_t} \left(y - g^\top \mathbb{E}[x_0 | x_t]\right)^2$$

Gradient-Guided Diffusion for Generative Optimization



- Use a pretrained score network. Only add guidance to the denoising process via gradient evaluations

Convergence to Regularized Optima

Theorem

Suppose the reward function is concave and *L*-smooth. Consider linear pre-trained score. The last batch satisfies

$$\mathbb{E}\left[f(x_\lambda^*) - f(\bar{z}_K)\right] = \lambda \left(\frac{L}{\lambda}\right)^K \cdot \mathcal{O}(D) \qquad (\mathcal{O}(d) \text{ for subspace data})$$

where $\lambda = \mathcal{O}(L)$ and x_λ^* is the maximizer of

$$x_\lambda^* = \arg\max_{x \in \mathbb{R}^D} \; f(x) - \frac{\lambda}{2}\|x - \bar{x}_0\|^2_{\bar{\Sigma}_0^{-1}}$$

with $\bar{x}_0$ and $\bar{\Sigma}_0$ the mean and covariance of the pre-training data.

Convergence to Regularized Optima

Theorem

Suppose the reward function is concave and *L*-smooth. Consider linear pre-trained score. The last batch satisfies

$$\mathbb{E}\left[f(x_\lambda^*) - f(\bar{z}_K)\right] = \lambda \left(\frac{L}{\lambda}\right)^K \cdot \mathcal{O}(D) \qquad (\mathcal{O}(d) \text{ for subspace data})$$

where $\lambda = \mathcal{O}(L)$ and x_λ^* is the maximizer of

$$x_\lambda^* = \arg\max_{x \in \mathbb{R}^D} \; f(x) - \frac{\lambda}{2}\|x - \bar{x}_0\|^2_{\bar{\Sigma}_0^{-1}}$$

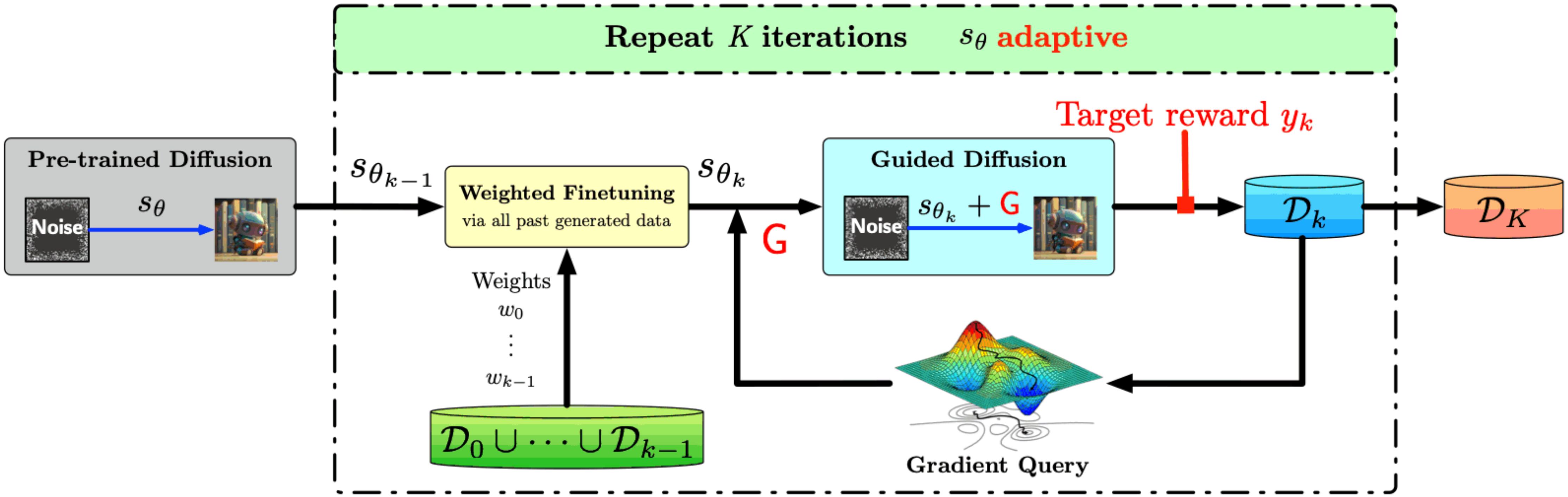
with $\bar{x}_0$ and $\bar{\Sigma}_0$ the mean and covariance of the pre-training data.

Effective gradient-guided diffusion for optimization:

✓ **Linear** convergence to a regularized maximum

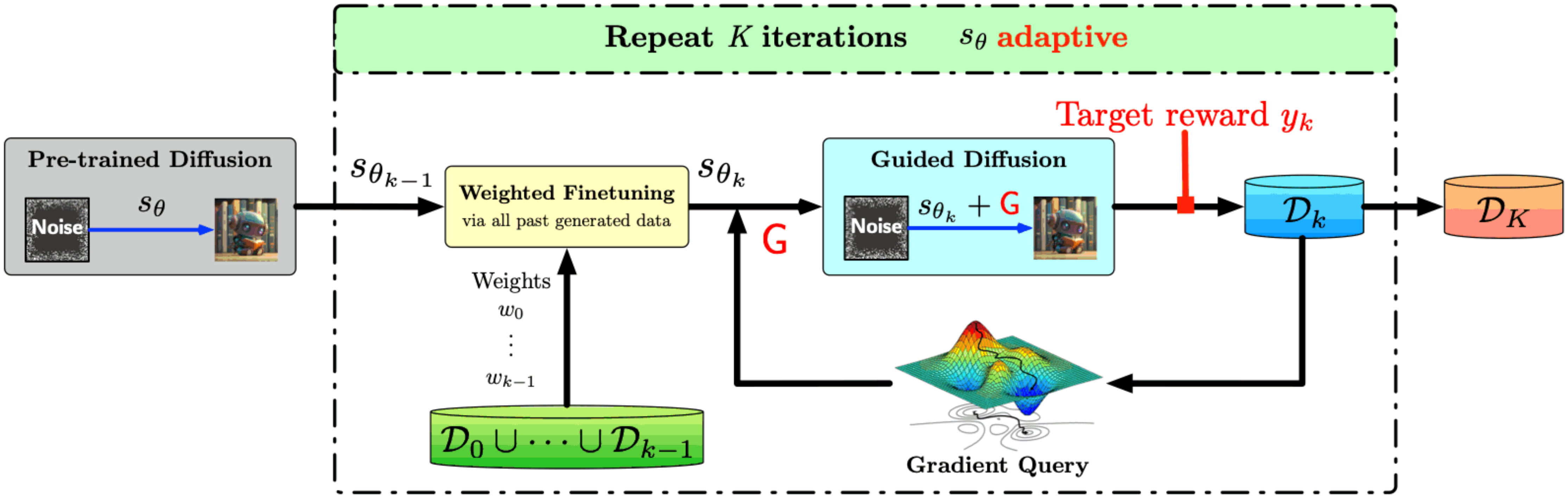
✓ Pre-training induces the **regularization**

Adaptive Finetuning via Self-Play for Finding Global Optima



Adaptive Finetuning via Self-Play for Finding Global Optima

➢ Update pretrained score with self-generated data



Adaptive Finetuning via Self-Play for Finding Global Optima

➢ Update pretrained score with self-generated data

Theorem

Assuming concave f and linear score network. Convergence to global maximum

$$\mathbb{E}\left[f(x^*) - f(z_K)\right] = \mathcal{O}\left(\frac{DL\log K}{K}\right) \qquad \left(\mathcal{O}\left(\frac{dL\log K}{K}\right) \text{ for subspace data}\right)$$

Thank You!