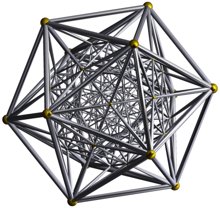
The Emergence of Generalizability and Semantic Low-Dim Subspaces in Diffusion Models

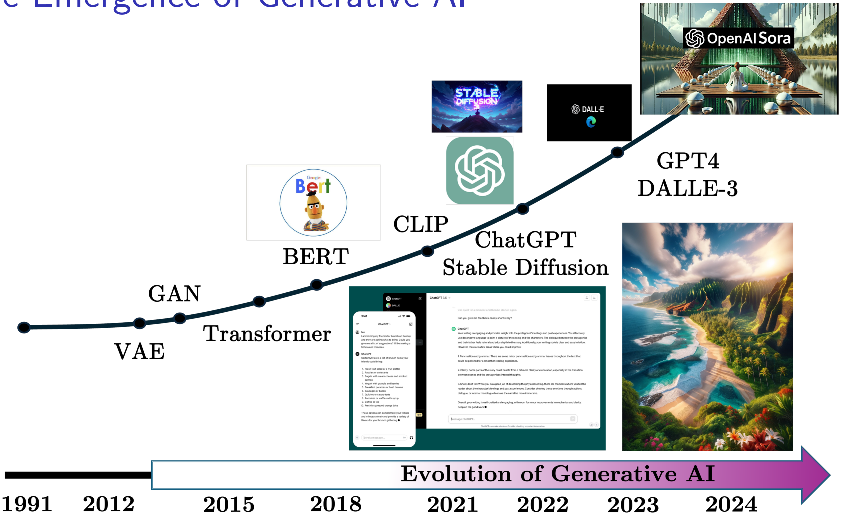
Qing Qu

EECS, University of Michigan

December 21, 2024

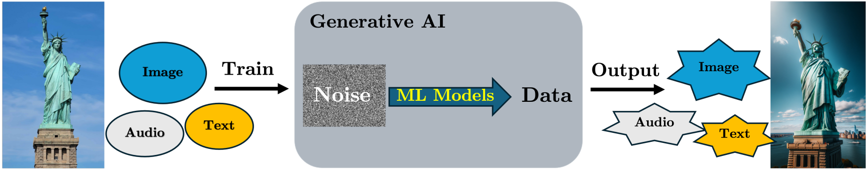


The Emergence of Generative AI[1]

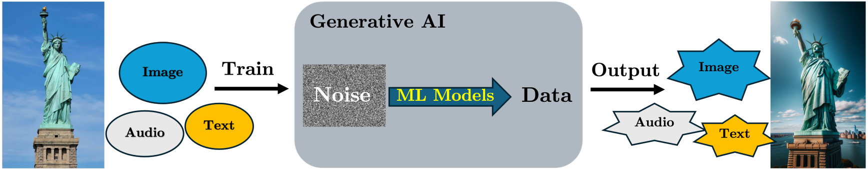


[1] Image credited to Prof. Mengdi Wang

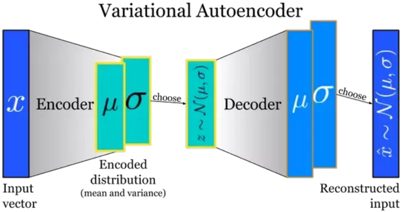
The Family of Generative Models



The Family of Generative Models

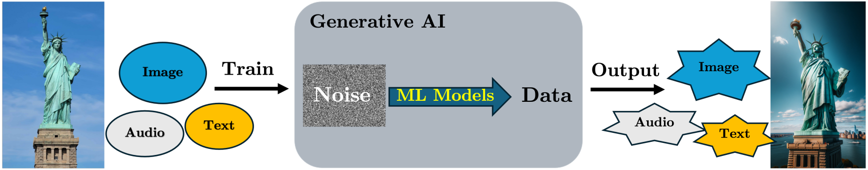


Generative models in the past:

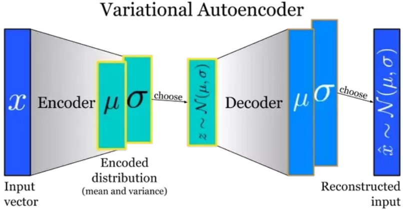


(a) VAE (Kingma & Wellings, 2013): poor generation quality.

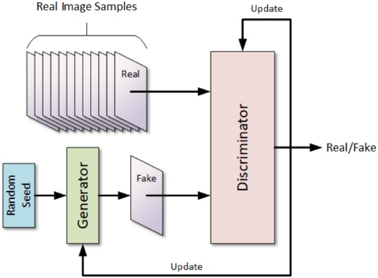
The Family of Generative Models



Generative models in the past:

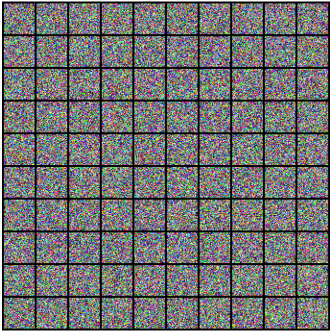


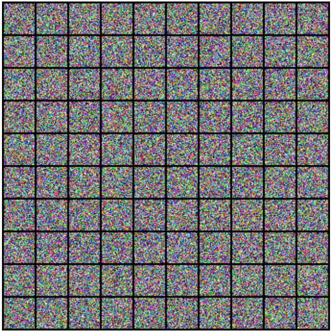
(a) VAE (Kingma & Wellings, 2013): poor generation quality.



(b) GAN (Goodfellow et al. 2014): unstable to train on large dataset.

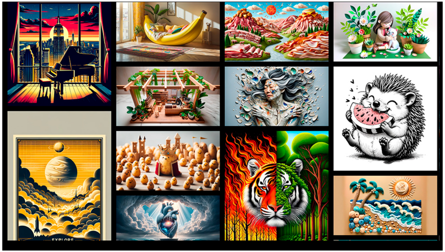
A Revolution by Diffusion Models[2]



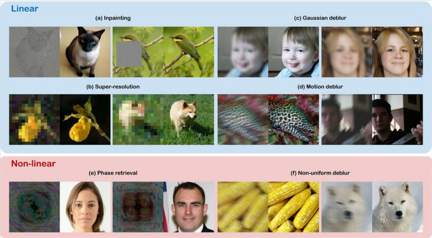


(Sohl-Dickstein et al. 2015, Song and Ermon 2019, Ho et al. 2020)

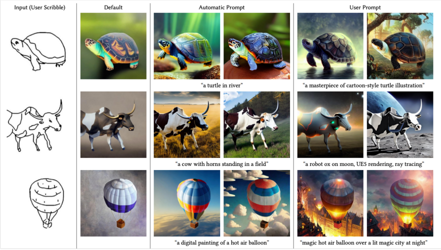
[2] https://yang-song.net/blog/2021/score/



Text-to-image Generation (DALL·E)



Solving Inverse Problem
(DPS, Chung et al.'23)

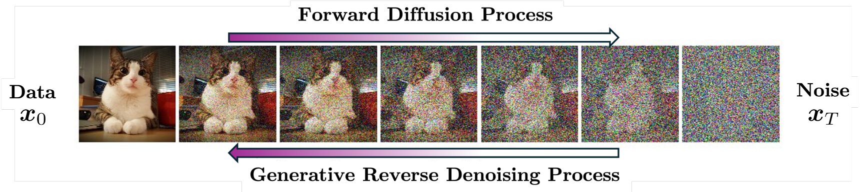


ControlNet
(Zhang et al.'23, ICCV best paper)

Video Generation: Sora - OpenAI[3]

[3] https://openai.com/sora

What are Diffusion Models?



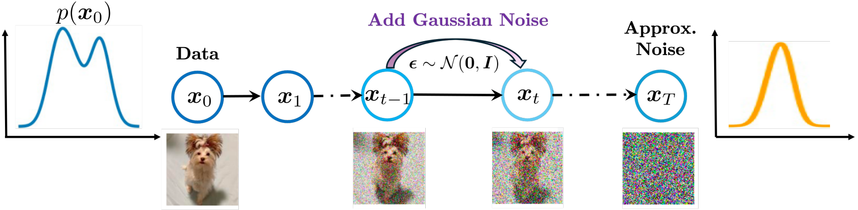
- **Forward process:** progressively adding noise to an image x_0;[4]

$$x_t = \alpha_t x_0 + \beta_t \epsilon, \ \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

- **Backward process:** starting from a random noise ϵ, progressively denoising to generate an image x_0

[4] Here, α_t and β_t are some pre-defined noise scales.

Forward Process: Progressively Adding Noise



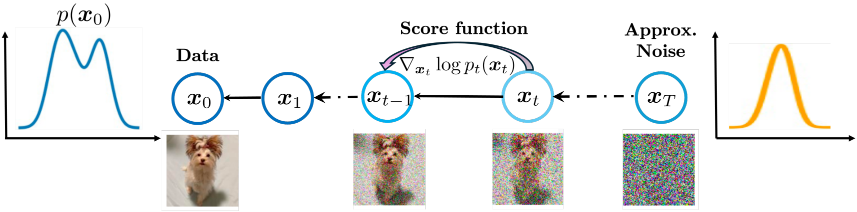
Forward stochastic differential equation (SDE):

$$\mathrm{d}\boldsymbol{x} = f(\boldsymbol{x}, t)\mathrm{d}t + g(t) \cdot \underbrace{\mathrm{d}\boldsymbol{w}}_{\textbf{Brownian}}.$$

- $f(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ and $g(\cdot) : \mathbb{R} \to \mathbb{R}$ are pre-defined *diffusion* and *drift* functions, respectively.[5]

[5] Here, $f(\boldsymbol{x}, t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\boldsymbol{x}$ and $g(t) = \frac{\mathrm{d}\beta_t^2}{\mathrm{d}t} - 2\beta_t^2 \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}$.

Generative Backward Process: Progressive Denoising

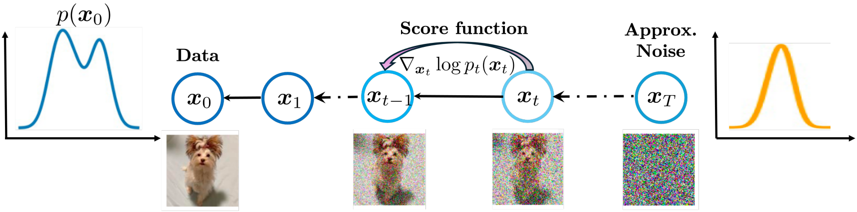


Backward probability flow **ODE** (Song et al., 2020):

$$\mathrm{d}\boldsymbol{x} = \left[f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 \cdot \underbrace{\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})}_{\textbf{score function}} \right] \mathrm{d}t.$$

[6]For example, EDM (Karras et al., 2022), DPM-solver (Lu et al., 2022).

Generative Backward Process: Progressive Denoising



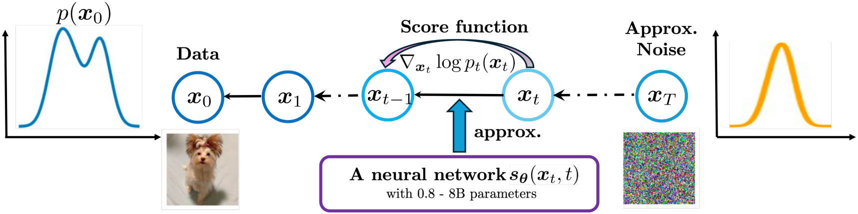
Backward probability flow **ODE** (Song et al., 2020):

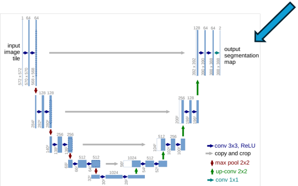
$$\mathrm{d}\boldsymbol{x} = \left[f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 \cdot \underbrace{\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})}_{\text{score function}} \right] \mathrm{d}t.$$

Deterministic, **much faster** with slightly inferior sample quality.[6]

[6] For example, EDM (Karras et al., 2022), DPM-solver (Lu et al., 2022).

How to Estimate the Score Function?



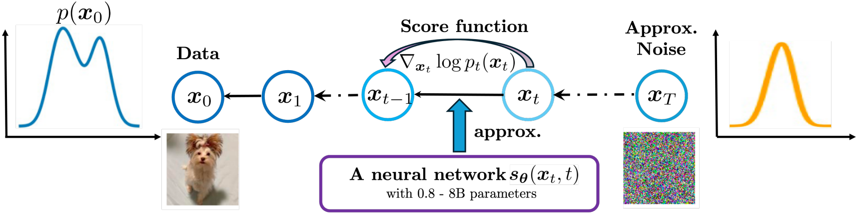


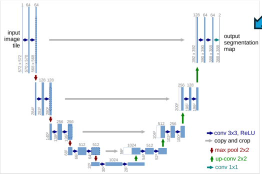
U-Net

(Ronneberger et al. 2015)

Stable Diffusion v1,v2

How to Estimate the Score Function?

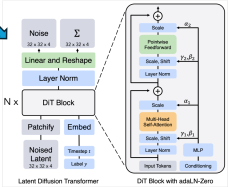




U-Net

(Ronneberger et al. 2015)

Stable Diffusion v1,v2

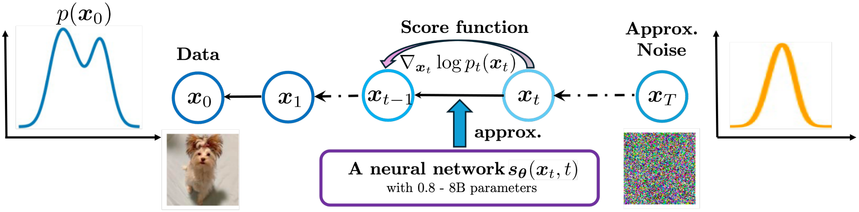


Transformer

(Peebles et al. 2023)

Sora, Stable Diffusion v3

How do We Learn the Neural Network?

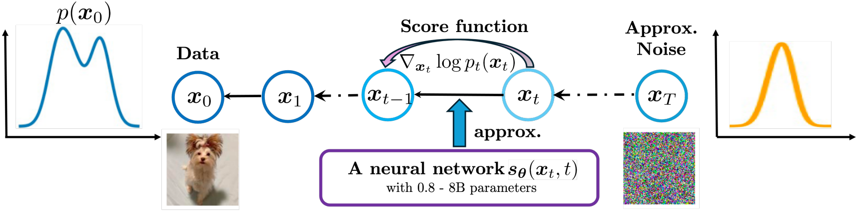


Training loss: we can learn the denoiser $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ simply by solving[7]

$$\min_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}) := \mathbb{E}_{\substack{t\sim\mathcal{U}[0,1],\, \boldsymbol{x}_0\sim p(\boldsymbol{x}_0) \\ \boldsymbol{x}_t\sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)}} \left[\beta_t^2 \|\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t) - \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\|_2^2\right]$$

[7]This can be achieved by sampling $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)$, $t \sim \mathcal{U}[0,1]$, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, to run stochastic gradient descent on $\mathcal{L}(\boldsymbol{\theta})$ to optimize the network parameters $\boldsymbol{\theta}$.

How do We Learn the Neural Network?



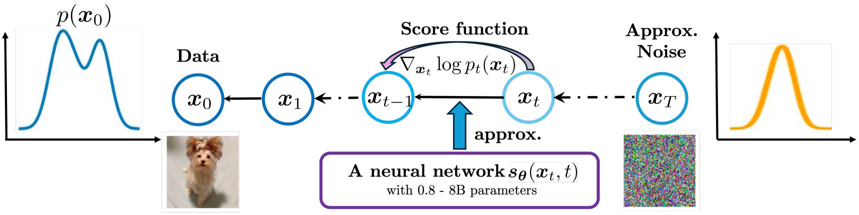
Training loss: we can learn the denoiser $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ simply by solving[7]

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) := \mathbb{E}_{\substack{t \sim \mathcal{U}[0,1],\, \boldsymbol{x}_0 \sim p(\boldsymbol{x}_0) \\ \boldsymbol{x}_t \sim p(\boldsymbol{x}_t | \boldsymbol{x}_0)}} \left[\beta_t^2 \left\| \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t) - \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \right\|_2^2 \right]$$

$$= \boxed{\mathbb{E}_{\substack{t \sim \mathcal{U}[0,1],\, \boldsymbol{x}_0 \sim p(\boldsymbol{x}_0) \\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})}} \left[\left\| \boldsymbol{\epsilon} + \beta_t \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \right\|_2^2 \right]} + \text{const.}$$

[7]This can be achieved by sampling $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)$, $t \sim \mathcal{U}[0,1]$, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, to run stochastic gradient descent on $\mathcal{L}(\boldsymbol{\theta})$ to optimize the network parameters $\boldsymbol{\theta}$.

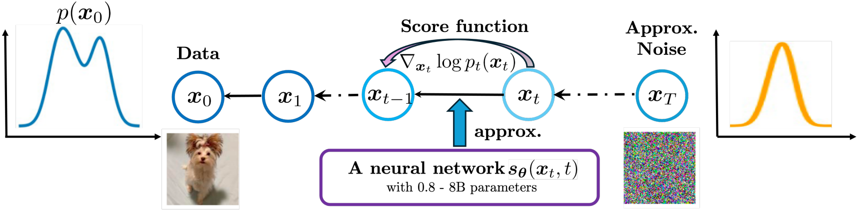
Mysteries Behind the Success of Diffusion Models



Fundamental questions to be answered:

- **Generalizability (theory):** When and why do diffusion models generate new samples?

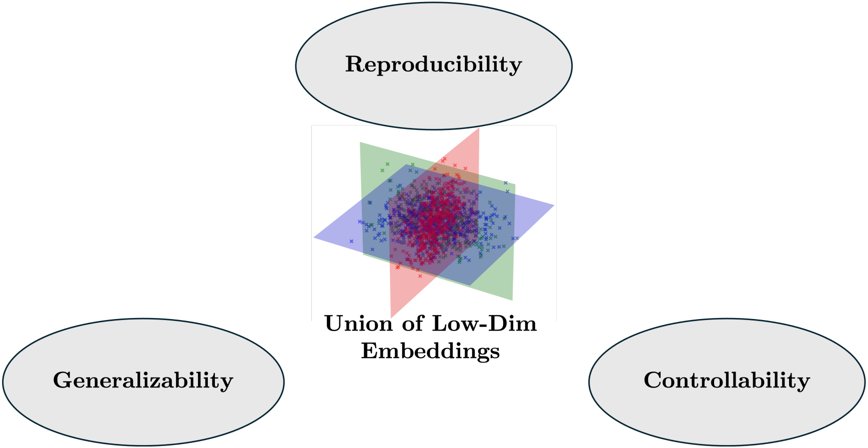
Mysteries Behind the Success of Diffusion Models



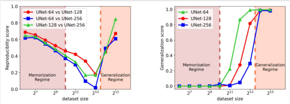
Fundamental questions to be answered:

- **Generalizability (theory):** When and why do diffusion models generate new samples?
- **Controllability (practice):** How can we control and manipulate the generated contents?

Outline

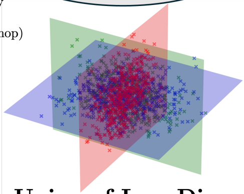

Reproducibility
Union of Low-Dim Embeddings
Generalizability
Controllability

Outline



Huijie Zhang et al. The Emergence of Reproducibility and Consistency in Diffusion Models. ICML'24.
(**Best Paper** at NeurIPS'23 Diffusion Model Workshop)

Reproducibility



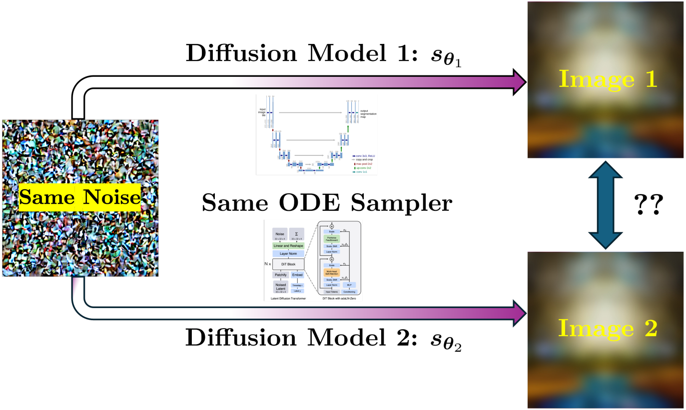
Union of Low-Dim Embeddings

Generalizability

Controllability

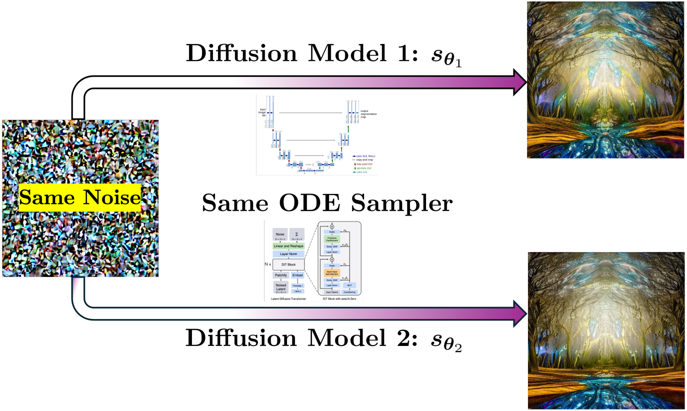
Reproducibility in Diffusion Models

Q1: *Starting from the* **same noise input**, *how are the generated data samples from various diffusion models related to each other?*



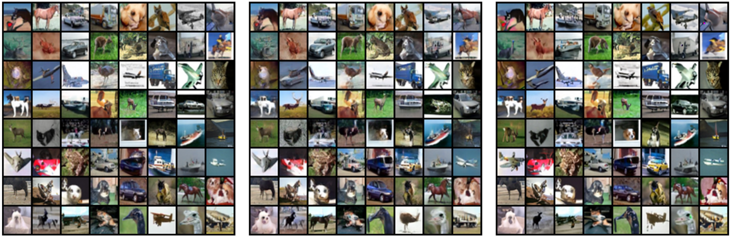
Reproducibility in Diffusion Models

Q1: *Starting from the* **same noise input**, *how are the generated data samples from various diffusion models related to each other?*



Reproducibility in Diffusion Model

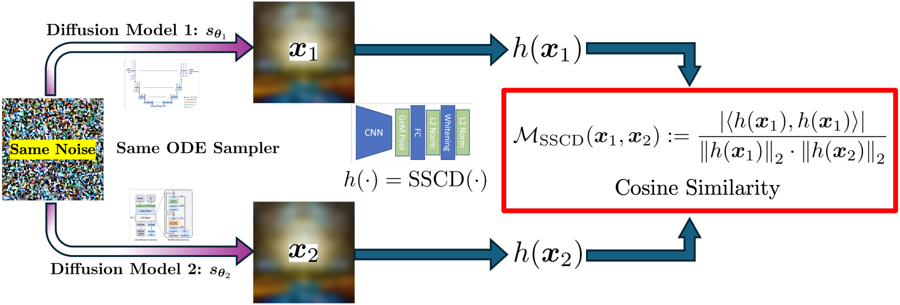
Q1: *Starting from the* **same noise input**, *how are the generated data samples from various diffusion models related to each other?*



(a) DDPMv4 (b) CT (c) U-ViT

Training on the same dataset, sampling by an ODE deterministic sampler.

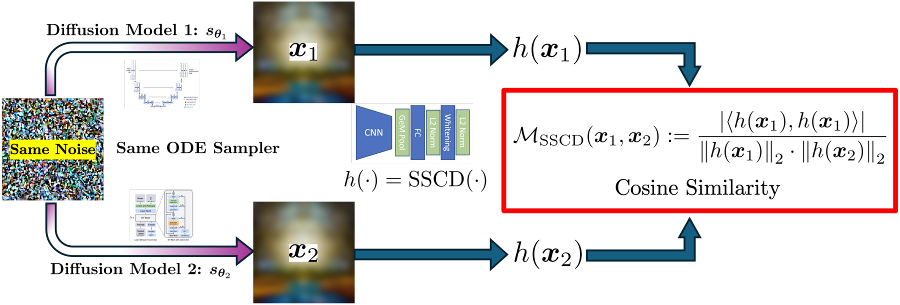
How to Measure Reproducibility Quantitatively?



Self-supervised copy detection (SSCD) similarity $\mathcal{M}_{\mathrm{SSCD}}(\cdot, \cdot)$.

- Here, $h(\cdot) = \mathrm{SSCD}(\cdot)$ represents a neural descriptor for copy detection. (Pizzi et al.'22, Somepalli et al.'23)

How to Measure Reproducibility Quantitatively?

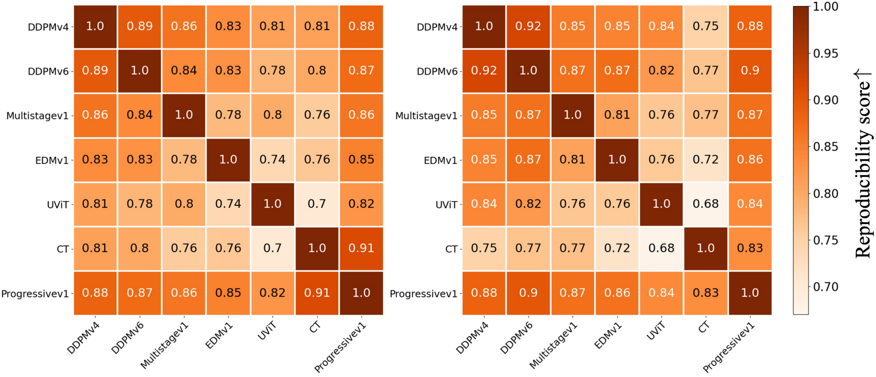


Reproducibility (RP) Score:

$$\text{RP Score} := \mathbb{P}\left(\mathcal{M}_{\mathsf{SSCD}}(\boldsymbol{x}_1, \boldsymbol{x}_2) > 0.6\right).$$

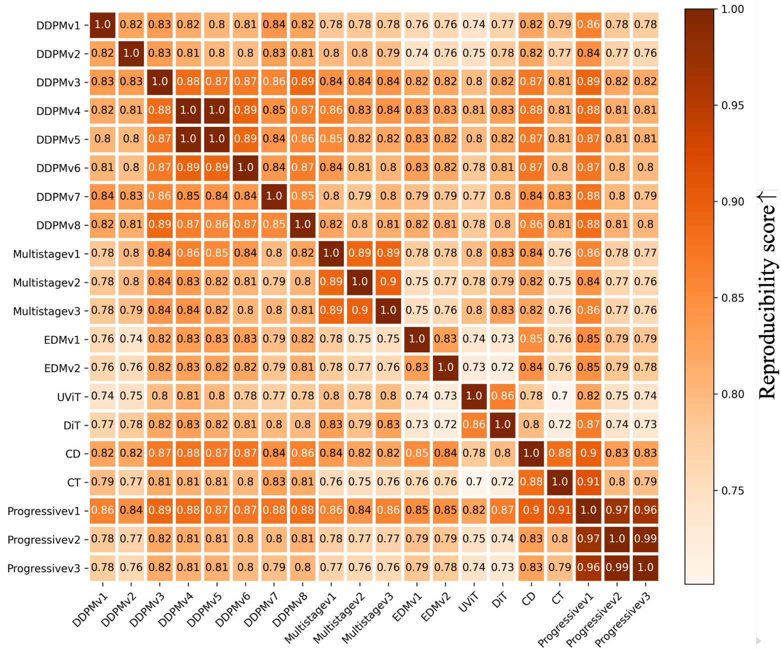
- It is a *probability measure* of the similarity between two models.
- We sample 10K random noise pairs to estimate the probability.

Quantitative Analysis of Diffusion Models (Cifar10)



Reproducibility score　　　　MAE score

- **Network architectures.** Transformer (U-ViT) vs U-Nets.
- **Training loss.** Consistency loss (CT), EDMv1, and others.
- **Sampling procedures.** DPM (DDPMv4), EDMv1, vs CT.
- **Perturbation kernels.** VP (DDPMv4), sub-VP(DDPMv6), EDMv1.


DDPMv1 1.0 0.82 0.83 0.82 0.8 0.81 0.84 0.82 0.78 0.78 0.78 0.76 0.76 0.74 0.77 0.82 0.79 0.86 0.78 0.78
DDPMv2 0.82 1.0 0.83 0.81 0.8 0.8 0.83 0.81 0.8 0.8 0.79 0.74 0.76 0.75 0.78 0.82 0.77 0.84 0.77 0.76
DDPMv3 0.83 0.83 1.0 0.88 0.87 0.87 0.86 0.89 0.84 0.84 0.84 0.82 0.82 0.8 0.82 0.87 0.81 0.89 0.82 0.82
DDPMv4 0.82 0.81 0.88 1.0 1.0 0.89 0.85 0.87 0.86 0.83 0.84 0.83 0.83 0.81 0.83 0.88 0.81 0.88 0.81 0.81
DDPMv5 0.8 0.8 0.87 1.0 1.0 0.89 0.84 0.86 0.85 0.82 0.82 0.83 0.82 0.8 0.82 0.87 0.81 0.87 0.81 0.81
DDPMv6 0.81 0.8 0.87 0.89 0.89 1.0 0.84 0.87 0.84 0.81 0.8 0.83 0.82 0.78 0.81 0.87 0.8 0.87 0.8 0.8
DDPMv7 0.84 0.83 0.86 0.85 0.84 0.84 1.0 0.85 0.8 0.79 0.8 0.79 0.79 0.77 0.8 0.84 0.83 0.88 0.8 0.79
DDPMv8 0.82 0.81 0.89 0.87 0.86 0.87 0.85 1.0 0.82 0.8 0.81 0.82 0.81 0.78 0.8 0.86 0.81 0.88 0.81 0.8
Multistagev1 0.78 0.8 0.84 0.86 0.85 0.84 0.8 0.82 1.0 0.89 0.89 0.78 0.78 0.8 0.83 0.84 0.76 0.86 0.78 0.77
Multistagev2 0.78 0.8 0.84 0.83 0.82 0.81 0.79 0.8 0.89 1.0 0.9 0.75 0.77 0.78 0.79 0.82 0.75 0.84 0.77 0.76
Multistagev3 0.78 0.79 0.84 0.84 0.82 0.8 0.8 0.81 0.89 0.9 1.0 0.75 0.76 0.8 0.83 0.82 0.76 0.86 0.77 0.76
EDMv1 0.76 0.74 0.82 0.83 0.83 0.83 0.79 0.82 0.78 0.75 0.75 1.0 0.83 0.74 0.73 0.85 0.76 0.85 0.79 0.79
EDMv2 0.76 0.76 0.82 0.83 0.82 0.82 0.79 0.81 0.78 0.77 0.76 0.83 1.0 0.73 0.72 0.84 0.76 0.85 0.79 0.78
UViT 0.74 0.75 0.8 0.81 0.8 0.78 0.77 0.78 0.8 0.78 0.8 0.74 0.73 1.0 0.86 0.78 0.7 0.82 0.75 0.74
DiT 0.77 0.78 0.82 0.83 0.82 0.81 0.8 0.8 0.83 0.79 0.83 0.73 0.72 0.86 1.0 0.8 0.72 0.87 0.74 0.73
CD 0.82 0.82 0.87 0.88 0.87 0.87 0.84 0.86 0.84 0.82 0.82 0.85 0.84 0.78 0.8 1.0 0.88 0.9 0.83 0.83
CT 0.79 0.77 0.81 0.81 0.81 0.8 0.83 0.81 0.76 0.75 0.76 0.76 0.76 0.7 0.72 0.88 1.0 0.91 0.8 0.79
Progressivev1 0.86 0.84 0.89 0.88 0.87 0.87 0.88 0.88 0.86 0.84 0.86 0.85 0.85 0.82 0.87 0.9 0.91 1.0 0.97 0.96
Progressivev2 0.78 0.77 0.82 0.81 0.81 0.8 0.8 0.81 0.78 0.77 0.77 0.79 0.79 0.75 0.74 0.83 0.8 0.97 1.0 0.99
Progressivev3 0.78 0.76 0.82 0.81 0.81 0.8 0.79 0.8 0.77 0.76 0.76 0.79 0.78 0.74 0.73 0.83 0.79 0.96 0.99 1.0
DDPMv1 DDPMv2 DDPMv3 DDPMv4 DDPMv5 DDPMv6 DDPMv7 DDPMv8 Multistagev1 Multistagev2 Multistagev3 EDMv1 EDMv2 UViT DiT CD CT Progressivev1 Progressivev2 Progressivev3
1.00
0.95
0.90
0.85
0.80
0.75
Reproducibility score ↑

Reproducibility is Rare in Other Generative Models

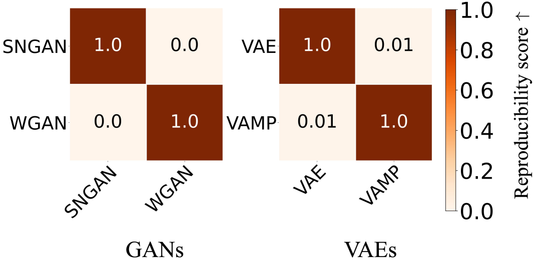


Figure: **Reproducibility for GANs and VAEs.**

- Before this work, only for VAE with a factorized prior distribution over the latent variables (Khemakhem et al. 2020).

Reproducibility is Rare in Other Generative Models

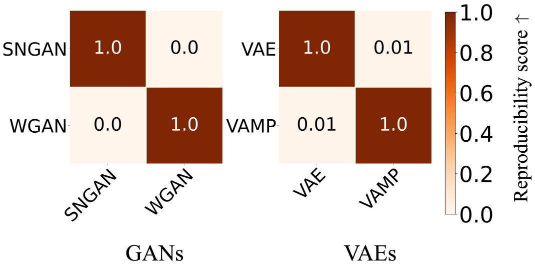
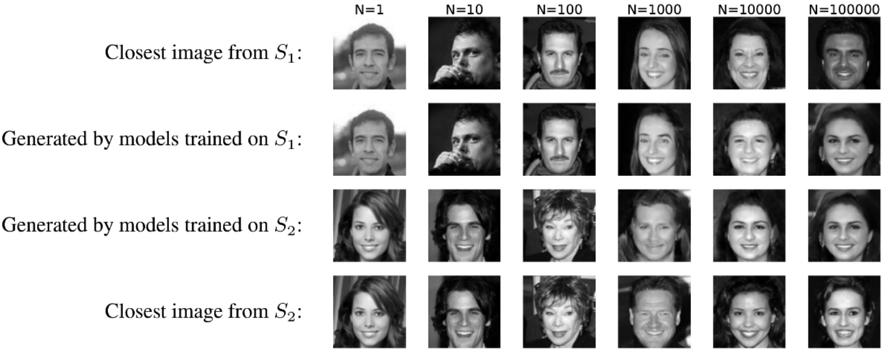


Figure: **Reproducibility for GANs and VAEs.**

- Before this work, only for VAE with a factorized prior distribution over the latent variables (Khemakhem et al. 2020).
- **Prevalent phenomenon in diffusion model!**

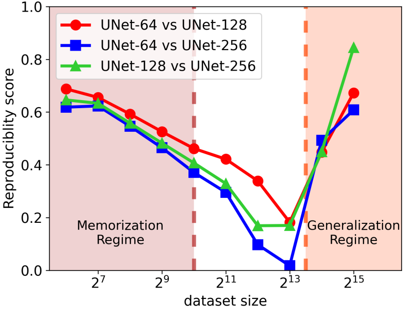
Complementary Results from Concurrent Work[8]

Non-overlapping training data from the same distribution: The same model trained from two exclusive subsets of the same training dataset



[8]Z Kadkhodaie, et al.'24 "Generalization in diffusion models arises from geometry-adaptive harmonic representation." (**ICLR'24 Outstanding Paper Award**)

Reproducibility Manifest in Two Different Regimes

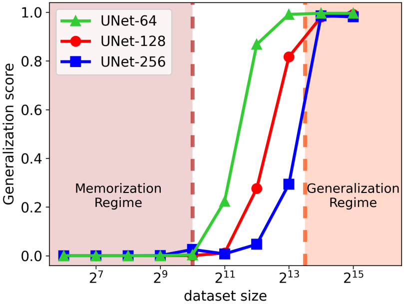


Reproducibility (RP) Score:

$$\text{RP Score} := \mathbb{P}\left(\mathcal{M}_{\text{SSCD}}(\boldsymbol{x}_1, \boldsymbol{x}_2) > 0.6\right).$$

Higher implies better reproducibility between two diffusion models.

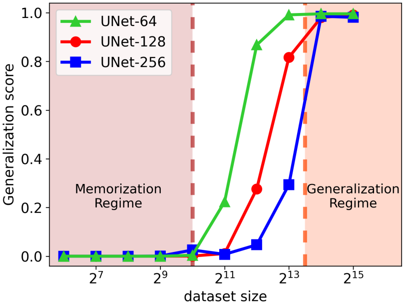
Reproducibility Manifests in Two Different Regimes



Generalization (GL) score is defined to

> measure the difference between a **newly generated sample** x and the **whole training dataset** $\{y_i\}_{i=1}^N$.

Reproducibility Manifests in Two Different Regimes

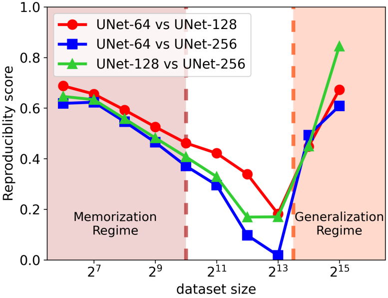


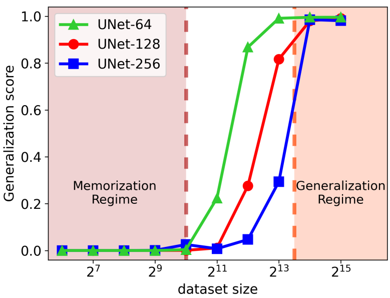
Generalization (GL) score (or perhaps memorization score?)

$$\text{GL Score} := 1 - \mathbb{P}\left(\max_{i \in [N]} \left[\mathcal{M}_{\text{SSCD}}(\boldsymbol{x}, \boldsymbol{y}_i)\right] > 0.6\right),$$

is also a *probability* measure. Higher implies better generalizability.

From "Memorization" to "Generalization"[9]



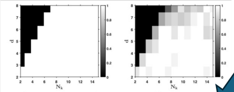


Reproducibility manifests in **two distinct regimes**, with a **strong** correlation with model's **generalizability**.

[9]Thanks AWS and Amazon Research Awards for computing resources

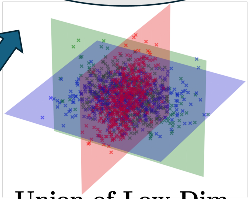
Outline

Reproducibility



(c) Subspace clustering (d) Diffusion model

Wang et al. Diffusion Models Learn Low-Dim Distributions via Subspace Clustering, 2024.

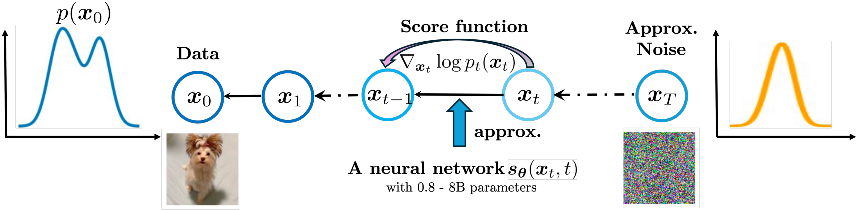


Union of Low-Dim Embeddings

Generalizability

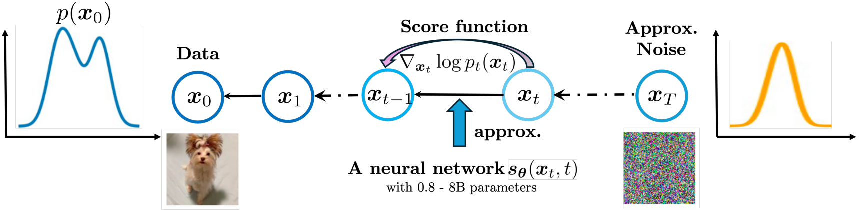
Controllability

Why Does Reproducibility Manifest in Distinct Regimes?



Backward ODE sampler: $\mathrm{d}\boldsymbol{x} = \left[f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 \cdot \underbrace{\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})}_{\text{score function}} \right] \mathrm{d}t.$

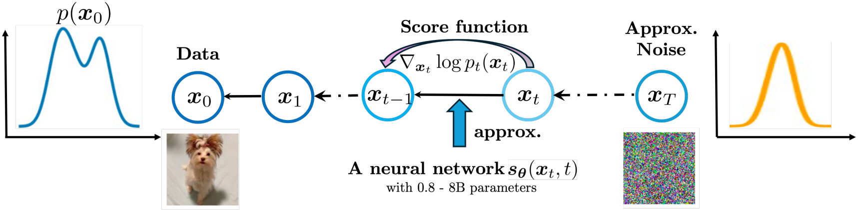
Why Does Reproducibility Manifest in Distinct Regimes?



Backward ODE sampler: $\mathrm{d}\boldsymbol{x} = \left[f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 \cdot \underbrace{\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})}_{\text{score function}} \right] \mathrm{d}t.$

- How well does diffusion model $s_{\boldsymbol{\theta}}$ approximate the score function $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$?

Why Does Reproducibility Manifest in Distinct Regimes?



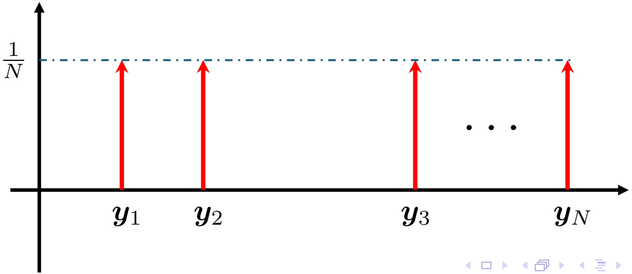
Backward ODE sampler: $\mathrm{d}\boldsymbol{x} = \left[f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 \cdot \underbrace{\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})}_{\text{score function}} \right] \mathrm{d}t.$

- How well does diffusion model $s_{\boldsymbol{\theta}}$ approximate the score function $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$?
- What distribution $p(\boldsymbol{x}_0)$ are we learning the score function for? (depending on training data size vs. model capacity)

Learning Empirical Distribution in Memorization Regime

Data Assumption: Given a training dataset $\mathcal{S} = \{\boldsymbol{y}_i\}_{i=1}^N$ of N-samples, the empirical distribution $p_{\text{emp}}(\boldsymbol{x})$ of $\mathcal{S}$ can be characterized by the **multi-delta distribution**:

$$p_{\text{emp}}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x} - \boldsymbol{y}_i).$$



Interpolation/Extrapolation of True Data Distribution

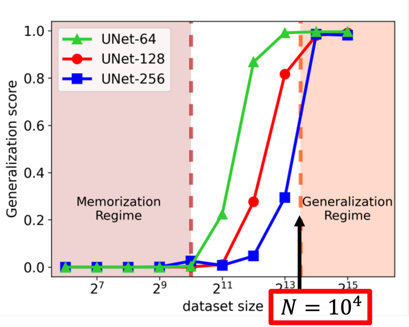
The curse of dimensionality: for image dataset (e.g., CelebA, Cifar),

$$p_{\mathsf{emp}}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x}-\boldsymbol{y}_i) \;\approx\; p_{\mathsf{data}}(\boldsymbol{x}),$$

to be ε-close, we could need at least $N \geq (L/\varepsilon)^d$ samples![10]

[10] We can draw this conclusion by a simple covering argument, the image dimension $d = 32 \times 32 = 1024$ for Cifar. See also recent work by Li et al., 2024.

Interpolation/Extrapolation of True Data Distribution

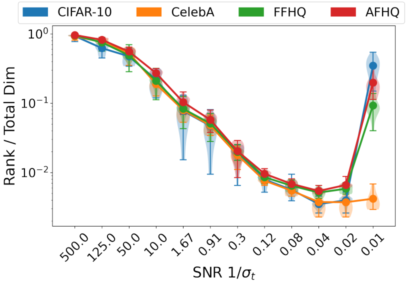


The curse of dimensionality: for image dataset (e.g., CelebA, Cifar),

$$p_{\text{emp}}(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{y}_i) \;\approx\; p_{\text{data}}(\boldsymbol{x}),$$

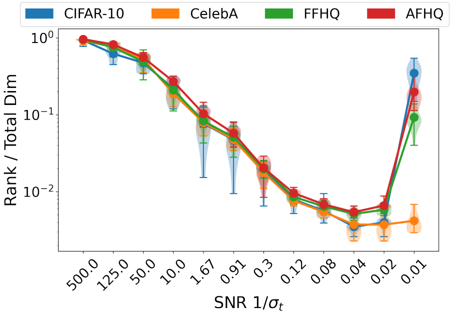
where we need an **extremely large** number of samples $N \geq (L/\varepsilon)^d$!

Intrinsic Low-Dimensionality of the Model



Evaluating the **rank ratio** of the Jacobian $\boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) = \nabla_{\boldsymbol{x}_t}\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t)$.

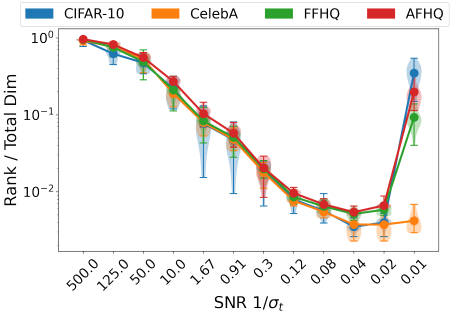
Intrinsic Low-Dimensionality of the Model



- Denoising autoencoder (DAE) formulation:

$$\min_{\boldsymbol{\theta}} \ \ell(\boldsymbol{\theta}) := \sum_{i=1}^{N} \int_0^1 \lambda_t \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_n)} \left[\left\| \boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \boldsymbol{x}^{(i)} \right\|^2 \right] \mathrm{d}t$$

Intrinsic Low-Dimensionality of the Model



- Denoising autoencoder (DAE) formulation:

$$\min_{\boldsymbol{\theta}} \ \ell(\boldsymbol{\theta}) := \sum_{i=1}^{N} \int_0^1 \lambda_t \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_n)} \left[\left\| \boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \boldsymbol{x}^{(i)} \right\|^2 \right] \mathrm{d}t$$

- Tweedie's formula, $\boldsymbol{x}_t = \alpha_t \boldsymbol{x}^{(i)} + \beta_t \boldsymbol{\epsilon}$:

$$\underbrace{\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)}_{\text{neural networks, like U-Net}} \approx \underbrace{\mathbb{E}[\boldsymbol{x}_0 | \boldsymbol{x}_t]}_{\text{posterior mean}} = (\boldsymbol{x}_t + \beta_t^2 \underbrace{\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})}_{\text{score function}})/\alpha_t.$$

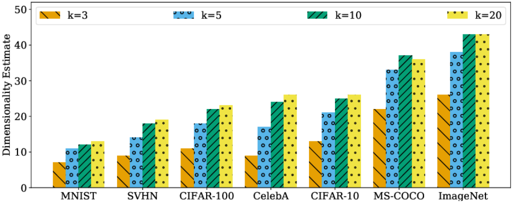
The Intrinsic Low-Dimensionality of Data[11]

The low-dim of model reflects the intrinsic dimension of our data:

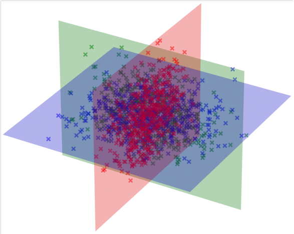
[11] Image credit: P. Pope et al., ICLR'2021.

The Intrinsic Low-Dimensionality of Data[11]

The low-dim of model reflects the intrinsic dimension of our data:



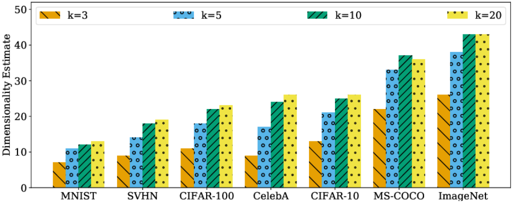
Intrinsic dimension of image datasets.



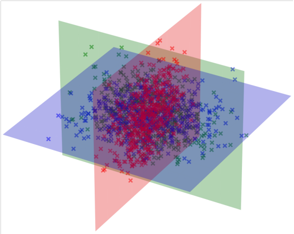
[11] Image credit: P. Pope et al., ICLR'2021.

The Intrinsic Low-Dimensionality of Data[11]

The low-dim of model reflects the intrinsic dimension of our data:



Intrinsic dimension of image datasets.



The blessing of dimensionality: the intrinsic dimension r of image data is **much lower** than the ambient dimension d, i.e., $r \ll d$.

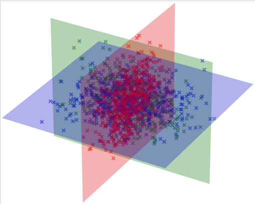
[11] Image credit: P. Pope et al., ICLR'2021.

Study Generalization under Low-Dimensional Models[12]

Data Assumption: mixture of low-rank Gaussian (MoLRG)

$$p_{\text{data}}(\boldsymbol{x}) = \frac{1}{K} \sum_{i \in [K]} \mathcal{N}(\boldsymbol{x}; \boldsymbol{0}, \boldsymbol{\Sigma}_i) \text{ with } \boldsymbol{\Sigma}_i = \boldsymbol{U}_i \boldsymbol{U}_i^\top,$$

where K is the number of clusters, and $\boldsymbol{U}_i \in \mathbb{R}^{d \times r}$ is the low-rank basis for the ith cluster with $r \ll d$, with $\boldsymbol{U}_i \perp \boldsymbol{U}_j (i \neq j)$.



[12] Chen et al. Score Approximation, Estimation and Distribution Recovery of Diffusion Models on Low-Dimensional Data. ICML'23.

Study Generalization under Low-Dimensional Models[13]

Data Assumption: mixture of low-rank Gaussian (MoLRG)

$$p_{\text{data}}(\boldsymbol{x}) = \frac{1}{K} \sum_{i \in [K]} \mathcal{N}\left(\boldsymbol{x}; \mathbf{0}, \boldsymbol{\Sigma}_i\right) \text{ with } \boldsymbol{\Sigma}_i = \boldsymbol{U}_i \boldsymbol{U}_i^\top,$$

where K is the number of clusters, and $\boldsymbol{U}_i \in \mathbb{R}^{d \times r}$ is the low-rank basis for the ith cluster with $r \ll d$, with $\boldsymbol{U}_i \perp \boldsymbol{U}_j (i \neq j)$.

Lemma. Suppose that $p_{\text{data}} \sim \texttt{MoLRG}$. For all $t > 0$,

$$\mathbb{E}\left[\boldsymbol{x}_0 | \boldsymbol{x}_t\right] = \frac{\alpha_t}{\alpha_t^2 + \beta_t^2} \sum_{k=1}^{K} w_k \boldsymbol{U}_k^\star \boldsymbol{U}_k^{\star\top} \boldsymbol{x}_t,$$

where $w_k = \frac{\pi_k \exp\left(\phi_t \|\boldsymbol{U}_k^{\star\top} \boldsymbol{x}_t\|^2\right)}{\sum_{k=1}^{K} \pi_k \exp\left(\phi_t \|\boldsymbol{U}_k^{\star\top} \boldsymbol{x}_t\|^2\right)}$ and $\phi_t := \alpha_t^2 / (2\beta_t^2(\alpha_t^2 + \beta_t^2))$.

[13] As shown by Wang et al.'23, the learned data distribution can be approx. by MoG.

A Simple Case Study: Single Low-rank Gaussian $K = 1$

Theorem (Equivalence to PCA)

Suppose that

- *The distribution* $p(\boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_0; \mathbf{0}, \boldsymbol{U}_g \boldsymbol{U}_g^\top\right)$ *with* $\boldsymbol{U}_g \in \mathcal{O}^{d \times r}$*;*
- *For each* $t \in [0, 1]$ *, we parameterize the denoiser* $\boldsymbol{x}_{\boldsymbol{U}}(\boldsymbol{x}_t, t)$ *as*

$$\boldsymbol{x}_{\boldsymbol{U}}(\boldsymbol{x}_t, t) = \frac{\alpha_t}{\alpha_t^2 + \beta_t^2} \cdot \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_t$$

Let $\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \cdots & \boldsymbol{y}_N \end{bmatrix}$ *be the training data matrix. Then we have*

A Simple Case Study: Single Low-rank Gaussian $K = 1$

Theorem (Equivalence to PCA)

Suppose that

- *The distribution* $p(\boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_0; \mathbf{0}, \boldsymbol{U}_g \boldsymbol{U}_g^\top\right)$ *with* $\boldsymbol{U}_g \in \mathcal{O}^{d\times r}$*;*
- *For each* $t \in [0,1]$ *, we parameterize the denoiser* $\boldsymbol{x}_{\boldsymbol{U}}(\boldsymbol{x}_t, t)$ *as*

$$\boldsymbol{x}_{\boldsymbol{U}}(\boldsymbol{x}_t, t) = \frac{\alpha_t}{\alpha_t^2 + \beta_t^2} \cdot \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_t$$

Let $\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \cdots & \boldsymbol{y}_N \end{bmatrix}$ *be the training data matrix. Then we have*

- *The training loss can be reduced to the loss of the* **PCA problem***:*

$$\max_{\boldsymbol{U}} \|\boldsymbol{U}^\top \boldsymbol{Y}\|_F^2, \quad \text{s.t.} \quad \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_r.$$

A Simple Case Study: Single Low-rank Gaussian $K = 1$

Theorem (Equivalence to PCA)

Suppose that

- *The distribution* $p(\boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_0; \mathbf{0}, \boldsymbol{U}_g \boldsymbol{U}_g^\top\right)$ *with* $\boldsymbol{U}_g \in \mathcal{O}^{d \times r}$*;*
- *For each* $t \in [0, 1]$ *, we parameterize the denoiser* $\boldsymbol{x}_{\boldsymbol{U}}(\boldsymbol{x}_t, t)$ *as*

$$\boldsymbol{x}_{\boldsymbol{U}}(\boldsymbol{x}_t, t) = \frac{\alpha_t}{\alpha_t^2 + \beta_t^2} \cdot \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_t$$

Let $\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \cdots & \boldsymbol{y}_N \end{bmatrix}$ *be the training data matrix. Then we have*

- *The training loss can be reduced to the loss of the* **PCA problem***:*

$$\max_{\boldsymbol{U}} \|\boldsymbol{U}^\top \boldsymbol{Y}\|_F^2, \quad \text{s.t.} \quad \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_r.$$

- *Thus, it holds for the global solution* $\boldsymbol{U}_\star$ *w.h.p. that*
  (i) If $N \geq r$*, we have* $\|\boldsymbol{U}_\star \boldsymbol{U}_\star^\top - \boldsymbol{U}_g \boldsymbol{U}_g^\top\|_F < \delta$*;*
  (ii) If $N < r$*, we have* $\|\boldsymbol{U}_\star \boldsymbol{U}_\star^\top - \boldsymbol{U}_g \boldsymbol{U}_g^\top\|_F \geq \sqrt{r - N} - \delta$*.*

Study of Multiple Low-Dim Subspaces $K > 1$

Theorem (Equivalence to Subspace Clustering)

Suppose that

- *Suppose that* $p(\boldsymbol{x}_0)$ *is* `MoLRG` *with* $K > 1$*;*
- *If we parameterize the DAE network*

$$\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{\alpha_t}{\alpha_t^2 + \beta_t^2} \sum_{k=1}^{K} w_k(\boldsymbol{\theta}; \boldsymbol{x}_t) \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{x}_t.$$

Study of Multiple Low-Dim Subspaces $K > 1$

Theorem (Equivalence to Subspace Clustering)

Suppose that

- *Suppose that $p(\boldsymbol{x}_0)$ is* `MoLRG` *with $K > 1$;*
- *If we parameterize the DAE network*

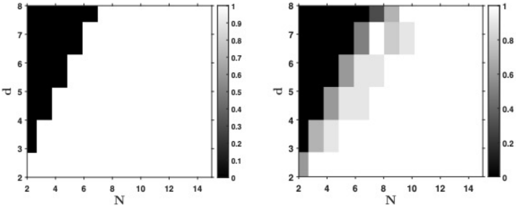
$$\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{\alpha_t}{\alpha_t^2 + \beta_t^2} \sum_{k=1}^{K} w_k(\boldsymbol{\theta}; \boldsymbol{x}_t) \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{x}_t.$$

Then the DAE training problem is equivalent to **subspace clustering**

$$\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{k=1}^{K} \sum_{i \in C_k(\boldsymbol{\theta})} \|\boldsymbol{U}_k^\top \boldsymbol{x}^{(i)}\|^2 \quad \text{s.t.} \quad \left[\boldsymbol{U}_1, \dots, \boldsymbol{U}_K\right] \in \mathcal{O}^{n \times dK},$$

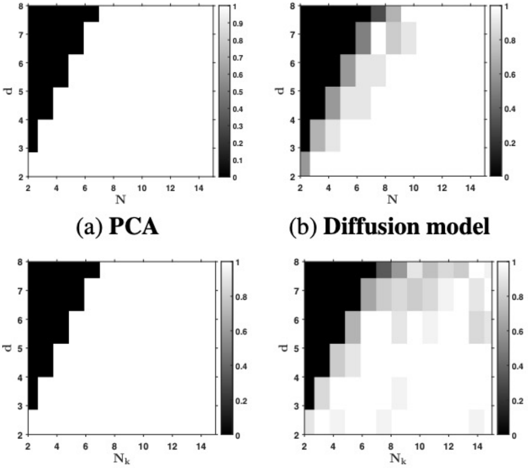
where $C_k(\boldsymbol{\theta}) := \left\{ i \in [N] : \|\boldsymbol{U}_k^\top \boldsymbol{x}^{(i)}\| \geq \|\boldsymbol{U}_l^\top \boldsymbol{x}^{(i)}\|, \ \forall l \neq k \right\}$ for $k \in [K]$.

Phase Transitions on MoLRG with Parameterized Networks



(a) **PCA** (b) **Diffusion model**

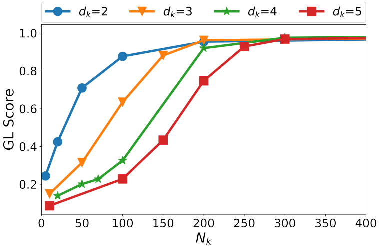
Phase Transitions on MoLRG with Parameterized Networks



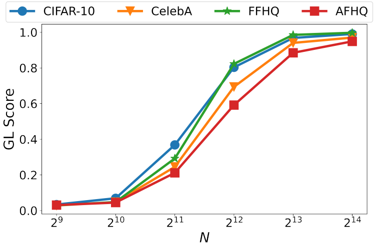
(a) **PCA** (b) **Diffusion model**

(c) **Subspace clustering** (d) **Diffusion model**

Phase Transition from Memorization to Generalization



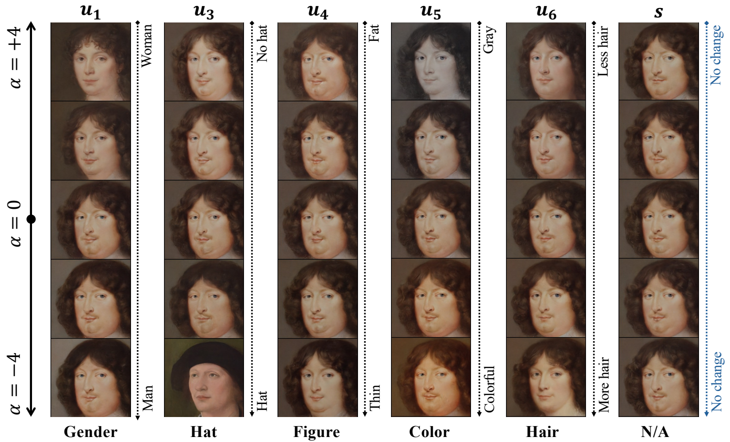
(a) MoLRG distribution



(b) **Real image data distribution**

Phase transition for diffusion models trained with U-Net.

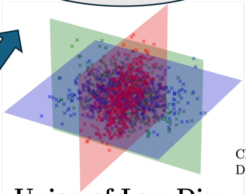
Semantic Meanings of the Low-Dimensional Basis



Semantic meanings of the eigenvectors U of the Jacobian $J_{\theta}(x_t, t)$.

Outline

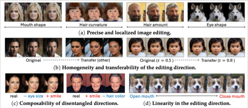
Reproducibility



Union of Low-Dim Embeddings

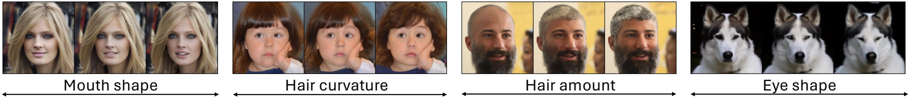
Generalizability

Controllability



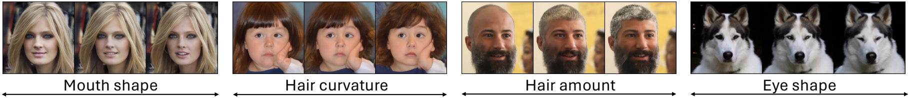
Chen et al. Exploring Low-Dimensional Subspaces in Diffusion Models for Controllable Image Editing, 2024.

LOw-rank COntrollable Image Editing (LOCO Edit)

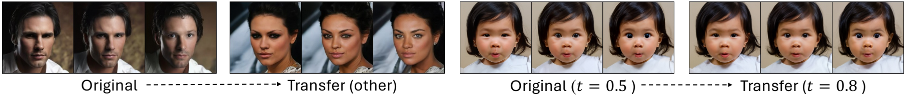


(a) **Precise and Localized**

LOw-rank **CO**ntrollable Image Editing (LOCO Edit)

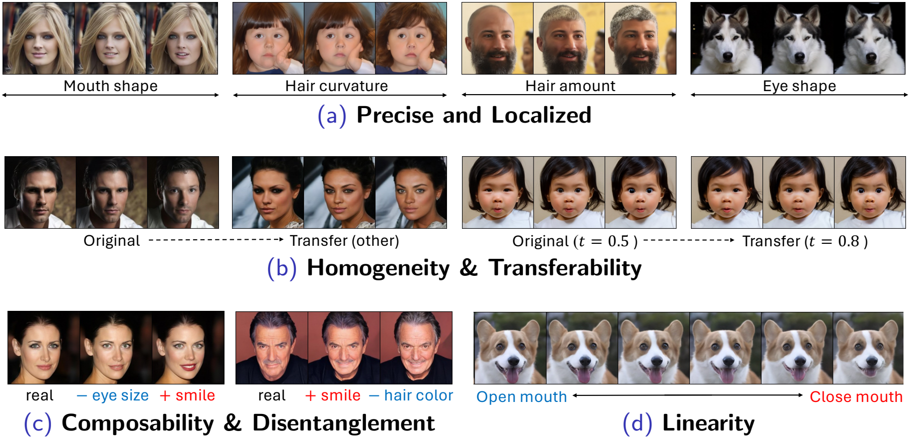


(a) **Precise and Localized**



(b) **Homogeneity & Transferability**

LOw-rank COntrollable Image Editing (LOCO Edit)



Editing in Text-to-image Diffusion Models

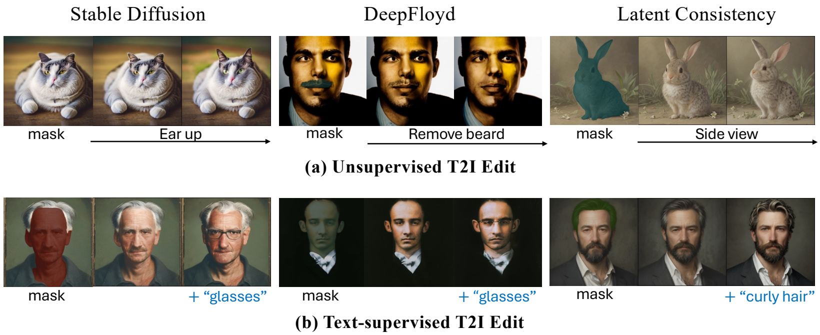


Figure: **T-LOCO Edit on T2I diffusion models.**

How does LOCO Edit Work?

Consider a unconditional diffusion model $\boldsymbol{s}_{\boldsymbol{\theta}}$:

- **Posterior mean predictor (PMP)** for the image $\boldsymbol{x}_0$:

$$\boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t;t) := \frac{\boldsymbol{x}_t + (1-\alpha_t)\,\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)}{\sqrt{\alpha_t}} \approx \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t],$$

How does LOCO Edit Work?

Consider a unconditional diffusion model $\boldsymbol{s_\theta}$:

- **Posterior mean predictor (PMP)** for the image $\boldsymbol{x}_0$:

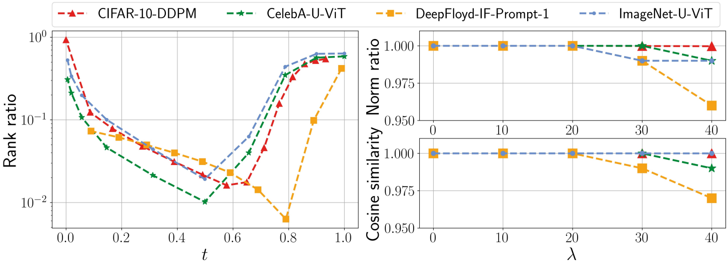
$$\boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t; t) \coloneqq \frac{\boldsymbol{x}_t + (1 - \alpha_t)\, \boldsymbol{s_\theta}(\boldsymbol{x}_t, t)}{\sqrt{\alpha_t}} \approx \mathbb{E}[\boldsymbol{x}_0 | \boldsymbol{x}_t],$$

- The **1st order Taylor expansion** of $\boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t + \lambda \Delta \boldsymbol{x})$ at $\boldsymbol{x}_t$:

$$\boxed{\boldsymbol{l_\theta}(\boldsymbol{x}_t; \lambda \Delta \boldsymbol{x}) \ \coloneqq \ \boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) + \lambda \boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) \cdot \Delta \boldsymbol{x},}$$

where $\boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) = \nabla_{\boldsymbol{x}_t} \boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t)$ is the Jacobian of $\boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t)$

How does LOCO Edit Work?



(a) **Low-rankness of the Jacobian** (b) **Local linearity of PMP**

Two key properties:

- **Local linearity** of the PMP $\boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) \approx \boldsymbol{l}_{\boldsymbol{\theta}}(\boldsymbol{x}_t; \lambda\Delta\boldsymbol{x})$.
- **Low-rankness** of the Jacobian $\boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$;

How does LOCO Edit Work?

$$\boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$$

- **Local linearity** of the PMP with $\Delta \boldsymbol{x} = \boldsymbol{v}_i$, one column of $\boldsymbol{V}$:

$$\begin{aligned} \boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t + \lambda \boldsymbol{v}_i) &\approx \boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) + \lambda \boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t)\boldsymbol{v}_i \\ &= \boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) + \lambda \sum_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^\top \boldsymbol{v}_i \\ &= \hat{\boldsymbol{x}}_{0,t} + \lambda \sigma_i \boldsymbol{u}_i. \end{aligned}$$

How does LOCO Edit Work?

$$\boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$$

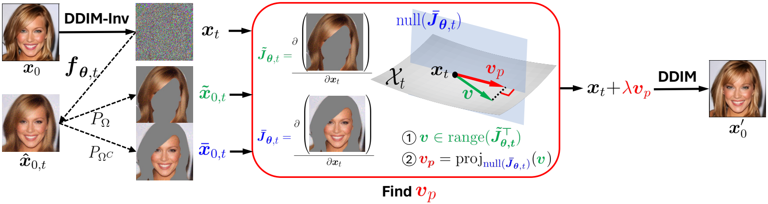
- **Local linearity** of the PMP with $\Delta \boldsymbol{x} = \boldsymbol{v}_i$, one column of $\boldsymbol{V}$:

$$\begin{aligned} \boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t + \lambda \boldsymbol{v}_i) &\approx \boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) + \lambda \boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t)\boldsymbol{v}_i \\ &= \boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) + \lambda \sum_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^\top \boldsymbol{v}_i \\ &= \hat{\boldsymbol{x}}_{0,t} + \lambda \sigma_i \boldsymbol{u}_i. \end{aligned}$$

- **Low rankness of the Jacobian** $\boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t)$ (e.g., $t = 0.7$):
  - $\boldsymbol{V}$ can be computed efficiently via generalized power method!

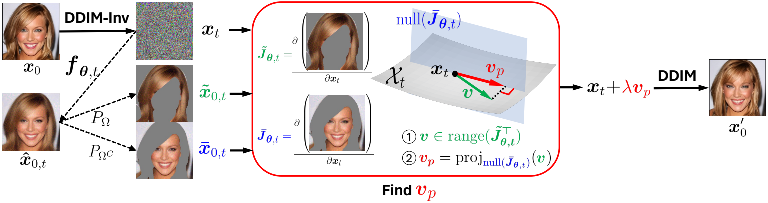
Overview of LOCO Edit

- Illustration of LOCO Edit for unconditional diffusion models:

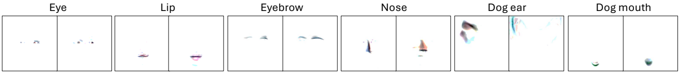


Overview of LOCO Edit

- Illustration of LOCO Edit for unconditional diffusion models:

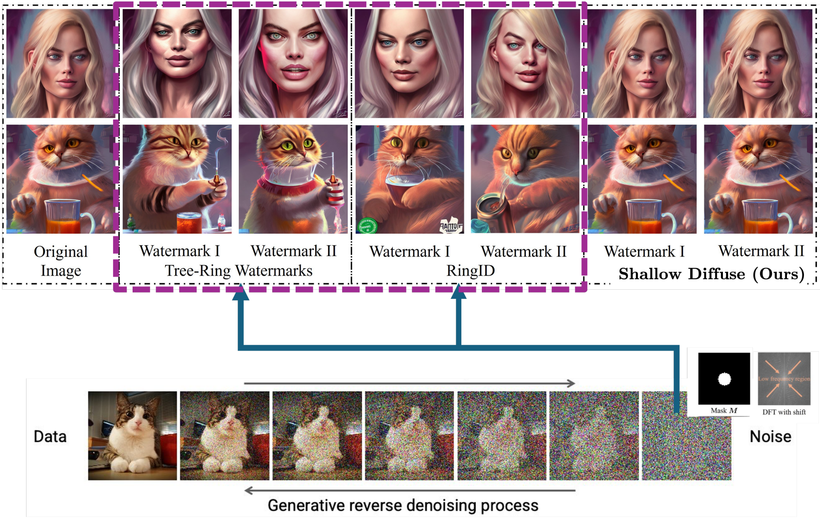


- Visualizing **editing directions** identified via LOCO Edit:

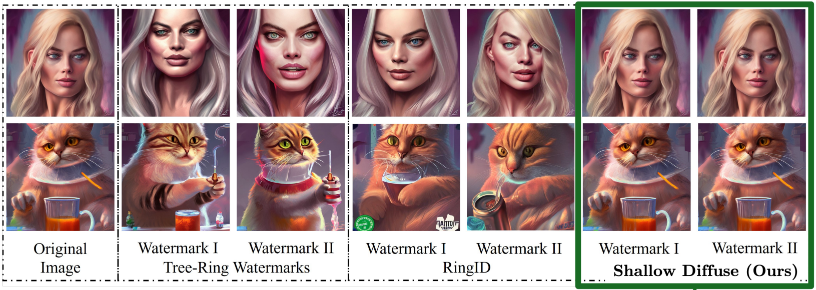


Visual Comparison with Existing Methods

Shallow Diffuse: Robust and Invisible Watermarking

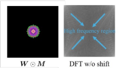


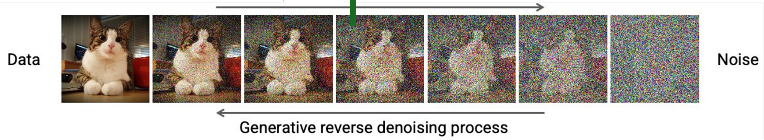
Shallow Diffuse: Robust and Invisible Watermarking



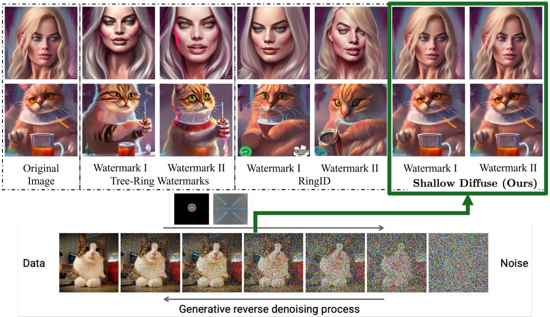
$$x_t^{\mathcal{W}} = x_t + \lambda \Delta x$$

$$\underbrace{J_{\theta,t}(x_t) \cdot \Delta x}_{\approx 0}$$





Shallow Diffuse: Robust and Invisible Watermarking



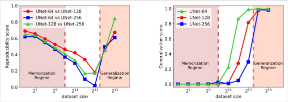
Key idea: Inject the watermark $\Delta \boldsymbol{x}$ in the **Null Space** of $\boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t)$:

$$\boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t^{\mathcal{W}}) = \boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) + \boxed{\underbrace{\lambda \boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) \cdot \Delta \boldsymbol{x}}_{\approx \boldsymbol{0}}} \approx \boldsymbol{f}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t)$$

Shallow Diffuse: Robust and Invisible Watermarking

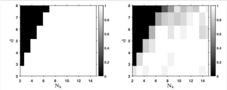
Method	CLIP-Score ↑	FID ↓	Watermarking Robustness (AUC ↑/TPR@1%FPR↑)					
			Clean	JPEG	G.Blur	G.Noise	Color Jitter	Average
Non-diffusion Method								
DwtDct	**0.3298**	25.73	0.97/0.85	0.64/0.00	0.78/0.00	0.44/0.02	0.53/0.09	0.60/0.03
DwtDctSvd	0.3291	26.00	**1.00/1.00**	0.80/0.08	0.99/0.80	0.97/0.84	0.50/0.09	0.82/0.45
RivaGAN	0.3252	**24.60**	1.00/0.99	**0.98/0.76**	**0.97/0.72**	**1.00/0.99**	**0.96/0.77**	**0.98/0.81**
Diffusion Method								
Stable Diffusion w/o WM	0.3286	25.56	-	-	-	-	-	-
Stable Signature	0.3622	30.86	**1.00/1.00**	0.99/0.76	0.57/0.00	0.71/0.14	0.96/0.87	0.81/0.46
Tree-Ring Watermarks	0.3310	25.82	**1.00/1.00**	0.99/0.97	0.98/0.98	0.94/0.50	0.96/0.67	0.97/0.80
RingID	0.3285	27.13	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	1.00/0.99	0.99/0.98	1.00/0.99
Gaussian Shading	**0.3631**	26.17	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**
Shallow Diffuse (ours)	0.3285	**25.58**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**

Conclusion



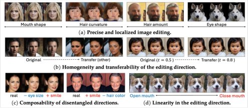
Huijie Zhang et al. The Emergence of Reproducibility and Consistency in Diffusion Models. ICML'24. (**Best Paper** at NeurIPS'23 Diffusion Model Workshop)

Reproducibility

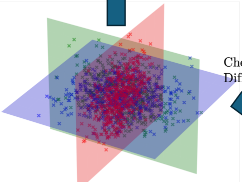


(c) **Subspace clustering** (d) **Diffusion model**

Wang et al. Diffusion Models Learn Low-Dim Distributions via Subspace Clustering, 2024.



Chen et al. Exploring Low-Dimensional Subspaces in Diffusion Models for Controllable Image Editing, 2024.



Union of Low-Dim Embeddings

Generalizability

Controllability

Conclusion

- Diffusion models exhibit **unique reproducibility** which manifests in two distinct data regimes: **memorization vs. generalization**.
- Diffusion models can learn **low-dimensional data distribution** without the curse of dimensionality.
- Diffusion models can be controlled through manipulating the low-dimensional **semantic subspaces**.

Conclusion

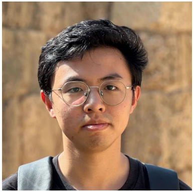
- Diffusion models exhibit **unique reproducibility** which manifests in two distinct data regimes: **memorization vs. generalization**.
- Diffusion models can learn **low-dimensional data distribution** without the curse of dimensionality.
- Diffusion models can be controlled through manipulating the low-dimensional **semantic subspaces**.

- **Theory: fundamental questions on generalization.**
- **Practice: many potential applications of our findings:**
  - More efficient training;
  - Interpretable & controllable data generation;
  - Model safety, privacy, and robustness;

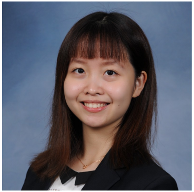
Major References

1. H. Zhang*, J. Zhou*, Y. Lu, M. Guo, P. Wang, L. Shen, and Q. Qu. The Emergence of Reproducibility and Consistency in Diffusion Models. ICML, 2024. (NeurIPS'23 Workshop on Diffusion Models, **Best Paper Award**)
2. P. Wang*, H. Zhang*, Z. Zhang, S. Chen, Y. Ma, and Q. Qu. Diffusion Model Learns Low-Dimensional Distributions via Subspace Clustering. Arxiv Preprint arXiv:2409.02426, 2024.
3. S. Chen*, H. Zhang*, M. Guo, Y. Lu, P. Wang, and Q. Qu. Exploring Low-Dimensional Subspaces in Diffusion Models for Controllable Image Editing. NeurIPS, 2024.
   https://chicychen.github.io/LOCO/index.html
4. X. Li, Y. Dai, Q. Qu. Understanding Generalizability of Diffusion Models Requires Rethinking the Hidden Gaussian Structure. NeurIPS, 2024.
5. W. Li, H. Zhang, Q. Qu. Shallow Diffuse: Robust and Invisible Watermarking through Low-Dimensional Subspaces in Diffusion Models. Arxiv Preprint arXiv:2410.21088, 2024.

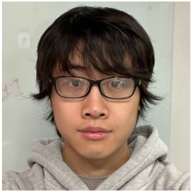
Acknowledgement


Huijie Zhang


Peng Wang

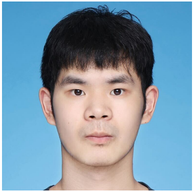

Siyi Chen


Zekai Zhang


Xiang Li


Minzhe Guo


Yifu Lu


Jinfan Zhou


Liyue Shen


Yi Ma

Acknowledgement

MIDAS
MICHIGAN INSTITUTE FOR DATA SCIENCE
UNIVERSITY OF MICHIGAN

Office of Naval Research
ONR
Science & Technology

aws

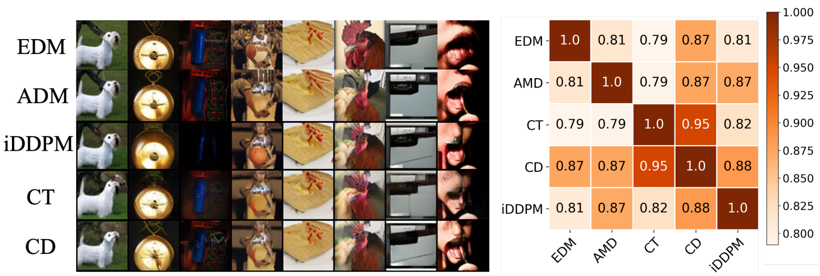
MICHIGAN INSTITUTE FOR COMPUTATIONAL DISCOVERY
UNIVERSITY OF MICHIGAN

NSF

KLA

Thank You! Questions?

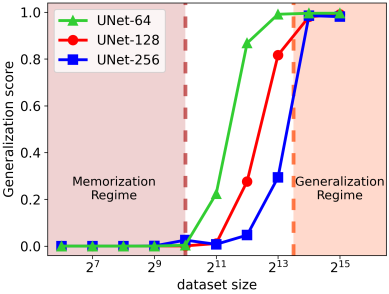
Reproducibility of Class Conditional Diffusion Models



(a) Visualization. (b) RP_{cond} Score

Figure: **Tested on ImageNet-1k dataset** for pre-trained diffusion models.

Model Capacity vs. Training Data Size

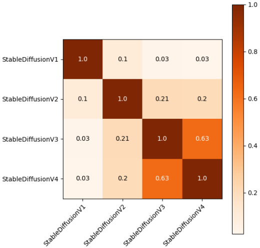


M: Memorization regime, T: Transition regime, G: Generalization regime

	2^6 (0.2M)	2^7 (0.4M)	2^8 (0.8M)	2^9 (1.6M)	2^10 (3.1M)	2^11 (6.3M)	2^12 (12.6M)	2^13 (25.2M)	2^14 (50.3M)	2^15 (100.6M)
UNet64 (26.7M)	M	M	M	M	M	T	T	G	G	G
UNet128 (106M)	M	M	M	M	M	M	T	T	G	G
UNet256 (426M)	M	M	M	M	M	M	T	T	G	G

Reproducibility of Text2Image Stable Diffusion Models

(a) Visualization of stable diffusion with same prompts. Each column has the same initial noise.



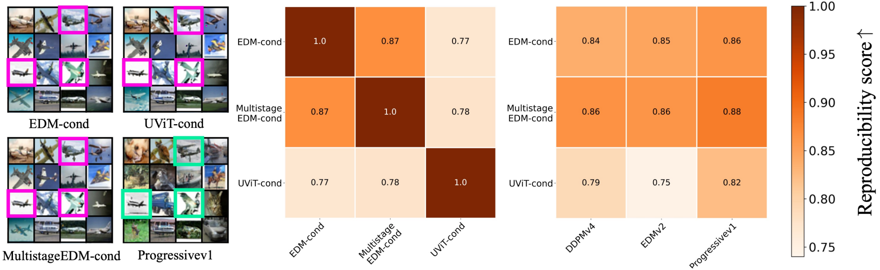
(b) Reproducibility score

Only V1-3 and V1-4 have exactly the same training dataset. Details of their relationships are on StableDiffusion's Github Page.

Conditional Diffusion Models

- Enable conditional generation, e.g. class condition, text-to-image, image-to-image translation [2].
- Change the denoiser from $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ to $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, c)$ for $c \in \mathcal{C}$. The set $\mathcal{C}$ could be the class label, text, image, and so on.
- Utilize $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, c)$ for both training and sampling.

Reproducibility of Class Conditional Diffusion Models



(a) Visualization (b) Conditional diffusion models (c) Conditional and unconditional diffusion models

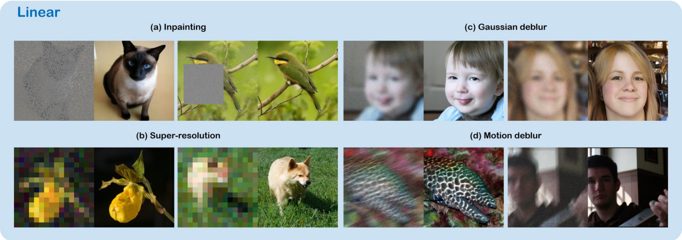
RP$_{cond}$ Score $:= \mathbb{P}\left(\mathcal{M}_{\text{SSCD}}(\boldsymbol{x}_1^c, \boldsymbol{x}_2^c) > 0.6 \mid c \in \mathcal{C}\right)$, $(\boldsymbol{x}_1^c, \boldsymbol{x}_2^c)$ are generated by two conditional models from the same initial noise and conditioned on the class $c \in \mathcal{C}$

RP$_{between}$ Score $:= \mathbb{P}\left(\max_{c \in \mathcal{C}}[\mathcal{M}_{\text{SSCD}}(\boldsymbol{x}_1, \boldsymbol{x}_2^c)] > 0.6\right)$, for an unconditional generation $\boldsymbol{x}_1$ and conditional generation $\boldsymbol{x}_2^c$ starting from the same noise.

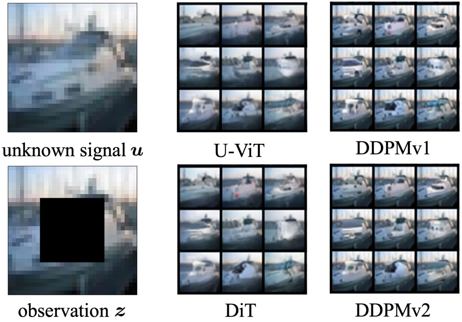
Model reproducibility of conditional models persists and is linked with unconditional counterparts.

Diffusion Models for Solving Inverse Problems

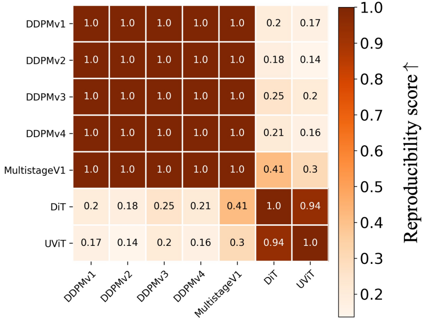
- **Inverse problem:** reconstruct an unknown signal $\boldsymbol{u}$ from the measurements $\boldsymbol{z}$ of the form $\boldsymbol{z} = \mathcal{A}(\boldsymbol{u}) + \boldsymbol{\eta}$, where $\mathcal{A}$ denotes some (given) sensing operator and $\boldsymbol{\eta}$ is the noise.
- **Sampling:** Enable conditional generation with only pre-trained unconditional denoiser $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$:
  $\boldsymbol{x}_t, \hat{\boldsymbol{x}}_0 \leftarrow \text{DeterministicSampler}(\boldsymbol{\epsilon}_{\boldsymbol{\theta}}, \boldsymbol{x}_{t+1}, t+1,)$
  $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_t - \xi_t \nabla_{\boldsymbol{x}_t} ||\boldsymbol{z} - \mathcal{A}\left(\hat{\boldsymbol{x}}_0\right)||_2^2$
- Diffusion Posterior Sampling (DPS) [1]



Reproducibility of diffusion model for inverse problem



(a) Visualization



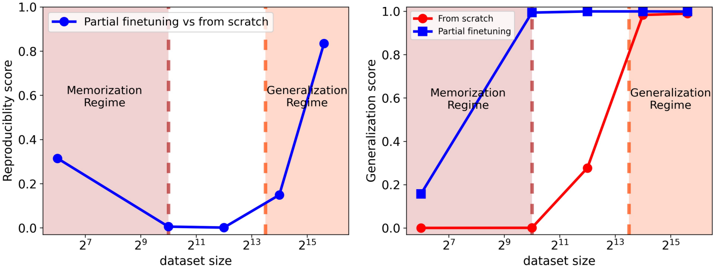
(b) Reproducibility Score

Model reproducibility largely holds only within the same type of network architectures.

Fine-tuning Diffusion Models

- For pre-trained large diffusion model (e.g. stable diffusion), we often fine-tune only a small portion of the parameters (e.g. attention layer, text embedding) on few-shot images.
- Obtain incredible generalizability, e.g. DreamBooth [3].

Reproducibility of diffusion model fine-tuning



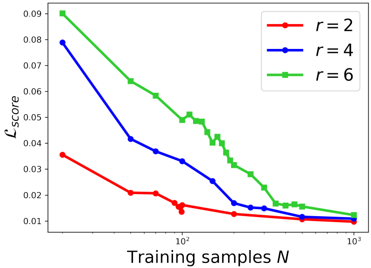
(a) Reproducibilty (b) Generalizability

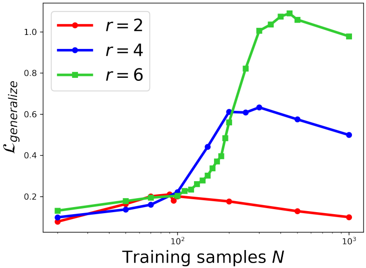
Pretrained on CIFAR-100, fine-tuning on CIFAR-10. Only fine-tuning the attention layer.

Partial fine-tuning reduces reproducibility but improves generalizability in “memorization regime”.

Study Generalization under Low-Dimensional Models

We learn $\boldsymbol{s}_{\boldsymbol{\theta}}$ with U-Nets. We set $C = 2$ and $d = 48$, varying N and r.





We measure the score distance and generalization as[14]

$$\mathcal{L}_{\text{score}} := \mathbb{E}_{\substack{t\sim\mathcal{U}(0,1),\boldsymbol{x}_0\sim p(\boldsymbol{x}_0)\\ \boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}}\left[\left\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t) - \boldsymbol{s}_{\texttt{MoLRG}}(\boldsymbol{x}_t,t)\right\|_2\right],$$

$$\mathcal{L}_{\text{generalize}} := \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{0},\beta_t\boldsymbol{I})}\left[\min_{i\in[N]}||\mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}) - \boldsymbol{y}_i||_2\right].$$

[14] Here, $\mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ is the mapping from the noise space $\mathcal{E}$ to the image space $\mathcal{I}$ induced by the learned score $\boldsymbol{s}_{\boldsymbol{\theta}}$ and ODE sampler.

Quantitative results of LOCO Edit

Method Name	Pullback	$\partial \epsilon_t / \partial \boldsymbol{x}_t$	NoiseCLR	Asyrp	BlendedDiffusion	**LOCO (Ours)**
Local Edit Success Rate↑	0.32	0.37	0.32	0.47	**0.55**	**0.80**
LPIPS↓	0.16	0.13	0.14	0.22	**0.03**	**0.08**
SSIM↑	0.60	0.66	0.68	0.68	**0.94**	**0.71**
Transfer Success Rate↑	0.14	0.24	**0.66**	0.58	Can't Transfer	**0.91**
Transfer Edit Time↓	4s	**2s**	5s	3s	Can't Transfer	**2s**
#Images for Learning	**1**	**1**	100	100	**1**	**1**
Learning Time↓	**8s**	**44s**	1 day	475s	120s	79s
One-step Edit?	✓	✓	✗	✗	✗	✓
No Additional Supervision?	✓	✓	✓	✗	✗	✓
Theoretically Grounded?	✗	✗	✗	✗	✗	✓
Localized Edit?	✗	✗	✗	✗	✓	✓

Table: **Comparisons with existing methods.** Our LOCO Edit excels in localized editing, transferability and efficiency, with other intriguing properties such as one-step edit, supervision-free, and theoretically grounded.

H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye.
Diffusion posterior sampling for general noisy inverse problems.
arXiv preprint arXiv:2209.14687, 2022.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer.
High-resolution image synthesis with latent diffusion models.
In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman.
Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.