ICCV 2025 Tutorial
Learning Deep Low-Dimensional Models from High-Dimensional Data: Theory to Practice

Sam Buchanan, Yi Ma, **Qing Qu**, Liyue Shen, Peihao Wang, Zhihui Zhu

October 23, 2025

EECS, University of Michigan

This Tutorial: The Outline

Session I: Introduction of Basic Low-D Models

Session II: Understanding Low-D Structures in Representations

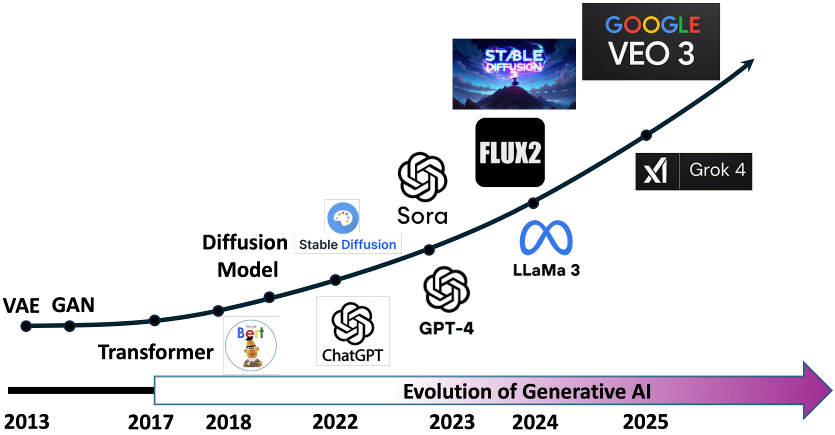
Session III: Understanding Low-D Structures in Generative Models

- **Lecture III-1: Low-Dimensional Models for Understanding Generalization in Diffusion Models**
- Lecture III-2: Exploring Low-Dimensional Structures for Controlling Diffusion Models

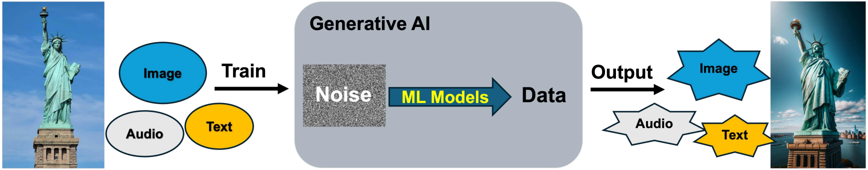
Session IV: Designing Deep Networks for Pursuing Low-D Structures

- Lecture IV-1: ReduNet: A White-box Deep Network from the Principle of Maximizing Rate Reduction
- Lecture IV-2: White-Box Transformers via Sparse Rate Reduction

The Emergence and Revolution of Generative AI

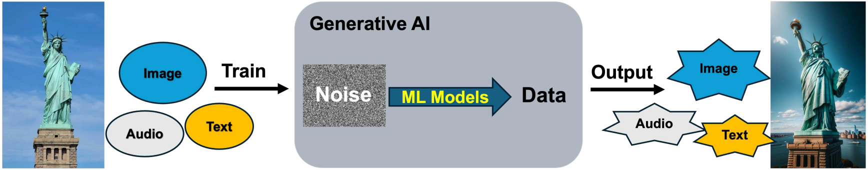


The Family of Generative Models[1]

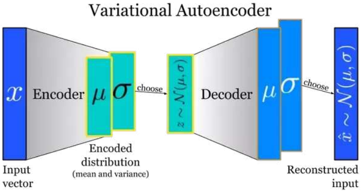


[1]Images credited to Prof. Mengdi Wang

The Family of Generative Models[1]



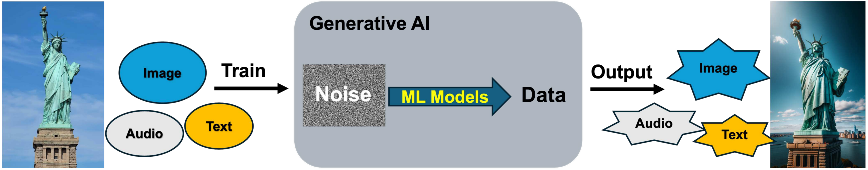
Generative models in the past:



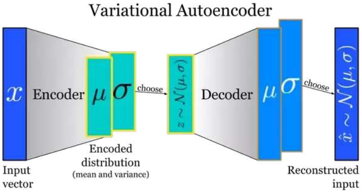
(a) VAE (Kingma & Wellings, 2013): poor generation quality.

[1] Images credited to Prof. Mengdi Wang

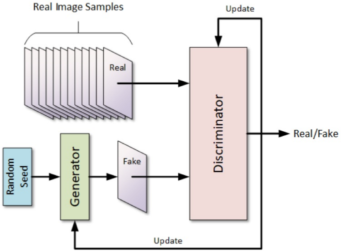
The Family of Generative Models[1]



Generative models in the past:



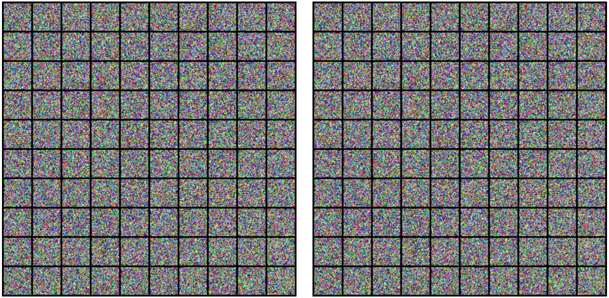
(a) VAE (Kingma & Wellings, 2013): poor generation quality.



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

[1] Images credited to Prof. Mengdi Wang

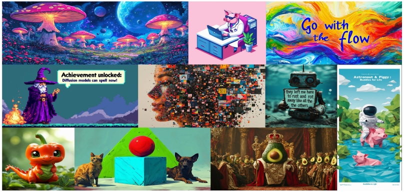
A Revolution by Diffusion Models[2]



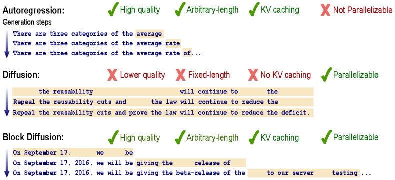
(Sohl-Dickstein et al. 2015, Song and Ermon 2019, Ho et al. 2020)

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

Many Commercial Applications



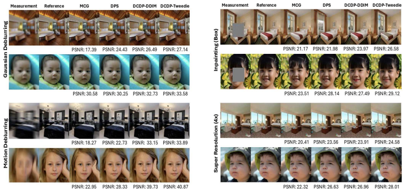
Text-to-Image Generation
(Stable Diffusion 3.5, Stability AI)



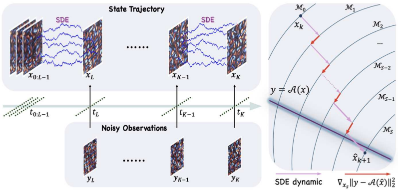
Diffusion LLM
(Block Diffusion, Arriola et al.'25)

Video Generation
(VEO, Google)

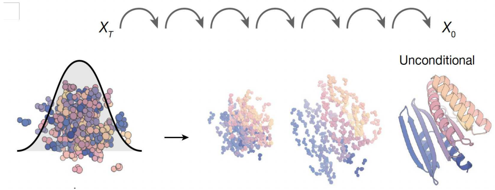
Revolutionizing Scientific Discovery



Inverse Problems
(DCDP, Li et al.'24)

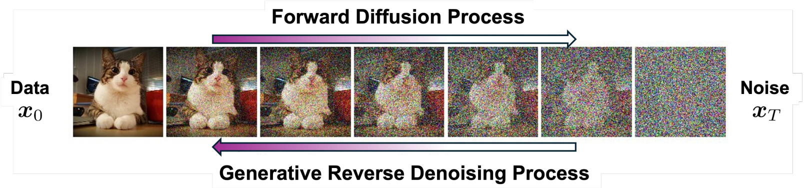


Data Assimilation
(FlowDAS, Chen et al.'25)



Protein Design
(RFDiffusion, Watson et al.'23)

What are Diffusion Models?



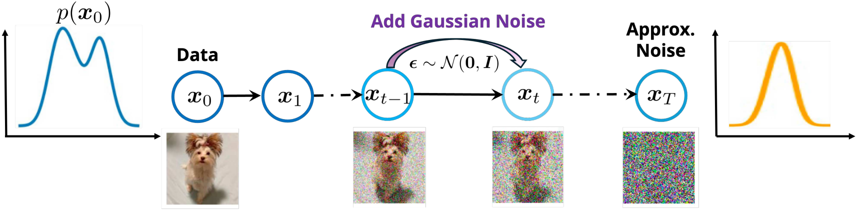
- **Forward process:** progressively adding noise to an image x_0;[3]

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

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

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

Forward Process: Progressively Adding Noise



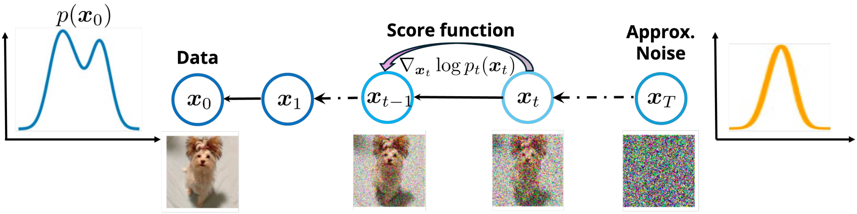
Forward stochastic differential equation (SDE):

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

- $f(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ and $g(\cdot) : \mathbb{R} \to \mathbb{R}$ are pre-defined *diffusion* and *drift* functions, respectively.[4]

[4] 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



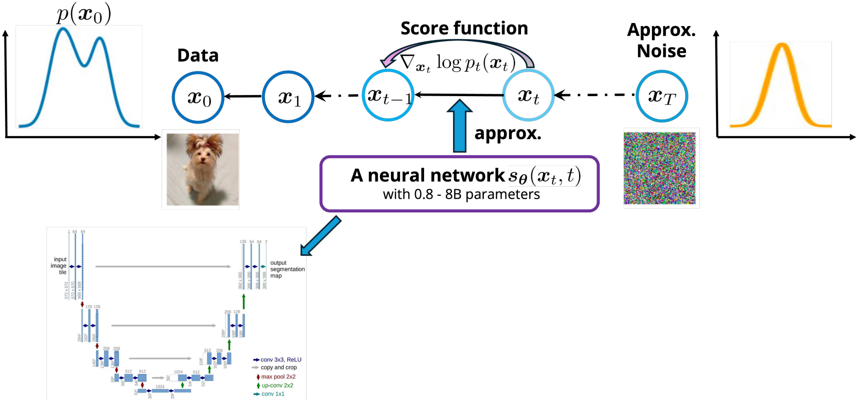
Backward probability flow **ODE** (Song et al., 2020):

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

Deterministic, **much faster** with slightly inferior sample quality.[5]

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

How to Estimate the Score Function?

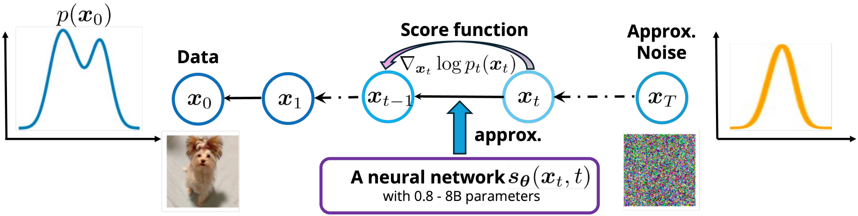


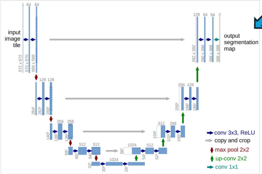
U-Net

(Ronneberger et al. 2015)

Stable Diffusion v1,v2

How to Estimate the Score Function?

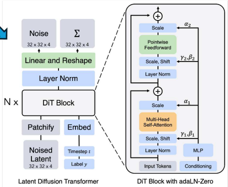




U-Net

(Ronneberger et al. 2015)

Stable Diffusion v1,v2

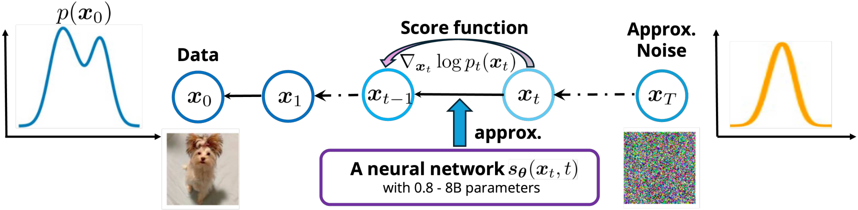


Transformer

(Peebles et al. 2023)

Sora, Stable Diffusion v3

How do We Learn the Neural Network?

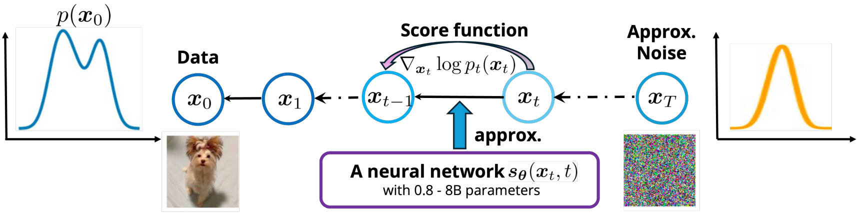


Training loss: we can learn the denoiser $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ simply by solving[6]

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

[6] 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?



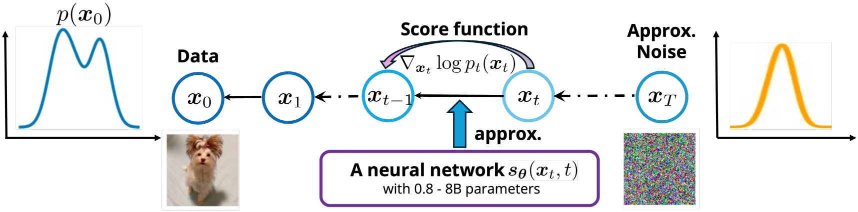
Training loss: we can learn the denoiser $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ simply by solving[6]

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

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

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

How do We Learn the Neural Network?



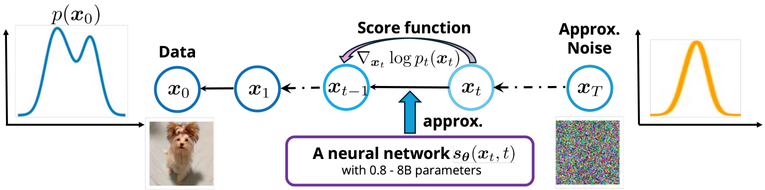
Training loss: we can learn the denoiser $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ simply by solving[7]

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

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

[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 $\boldsymbol{\theta}$.

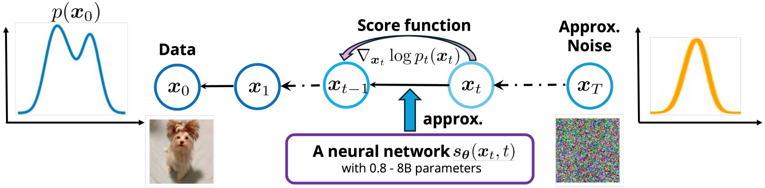
Outline



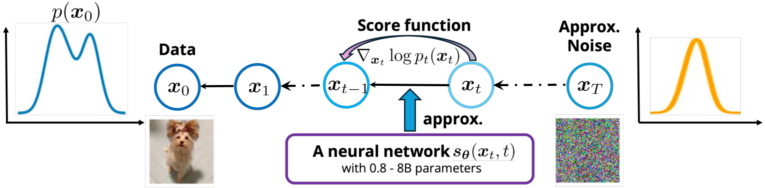
Session III: Exploring Low-D Structures in Generative Models

- **Lecture III-1: Low-Dimensional Models for Understanding Generalization in Diffusion Models**
- Lecture III-2: Exploring Low-Dimensional Structures for Controlling Diffusion Models

Generalization of Diffusion Models



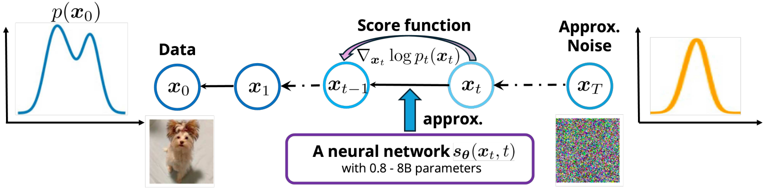
Generalization of Diffusion Models



Generalization: in practice, diffusion models are trained using **empirical loss** of $\mathcal{L}(\theta)$ with **finite** training samples $\mathcal{S} = \{x^{(i)}\}$:

- **Question I:** As diffusion models are trained to fit the training samples, why and when can they **generate new sensible** samples without curse of dimensionality?

Generalization of Diffusion Models



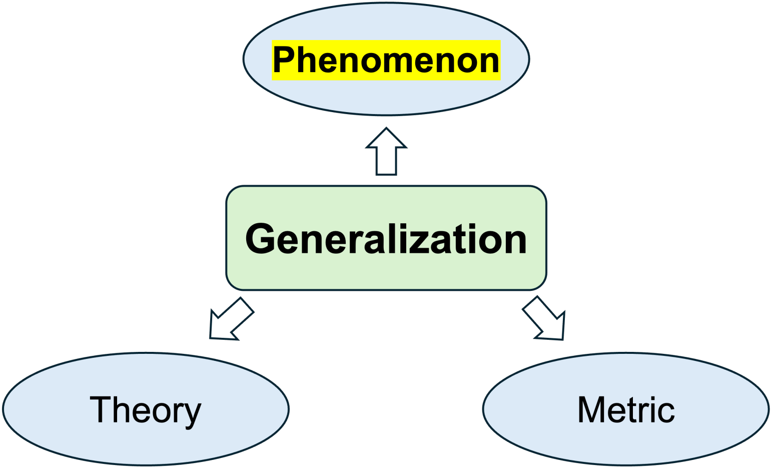
Generalization: in practice, diffusion models are trained using **empirical loss** of $\mathcal{L}(\boldsymbol{\theta})$ with **finite** training samples $\mathcal{S} = \{\boldsymbol{x}^{(i)}\}$:

- **Question I:** As diffusion models are trained to fit the training samples, why and when can they **generate new sensible** samples without curse of dimensionality?
- **Question II:** Without **ground-truth** data distribution, how can we measure generalization behavior in terms of training samples and model capacity?

Outline

1. **Phenomenon of Generalization** via Model Reproducibility
2. **Theory of Generalization** via Low-dimensional Models
3. **Quantifying Generalization** via Probability Flow Distance
4. **Conclusion** & **Discussion**

Outline[8]



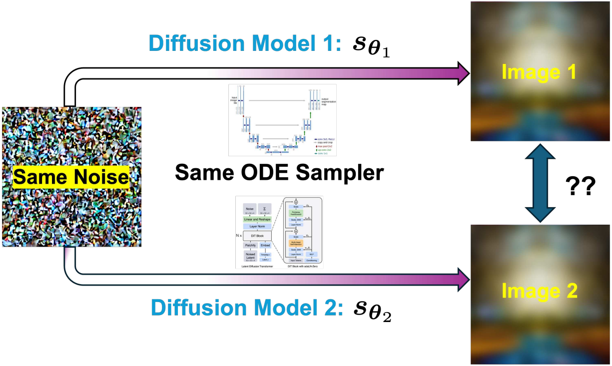
[8] H. Zhang*, J. Zhou*, Y. Lu, M. Guo, P. Wang, L. Shen, and Q. Qu. The Emergence of Reproducibility and Consistency in Diffusion Models. ICML, 2024. (NeurIPS'23 Workshop on Diffusion Models, **Best Paper Award**)

Phenomenon of Generalization via Model Reproducibility

Reproducibility in Diffusion Models

Question for Audience

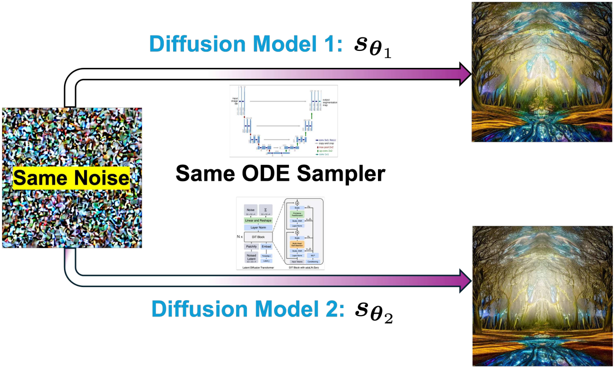
Starting from the **same noise input**, how are the generated data samples from various diffusion models related to each other?



Reproducibility in Diffusion Models

Question for Audience

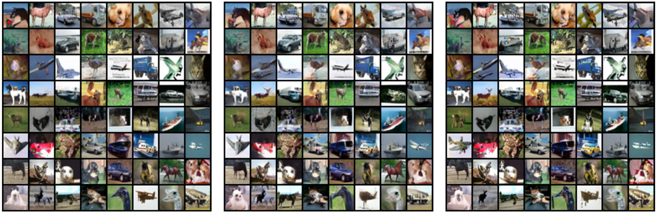
Starting from the **same noise input**, how are the generated data samples from various diffusion models related to each other?



Reproducibility in Diffusion Model

Question for Audience

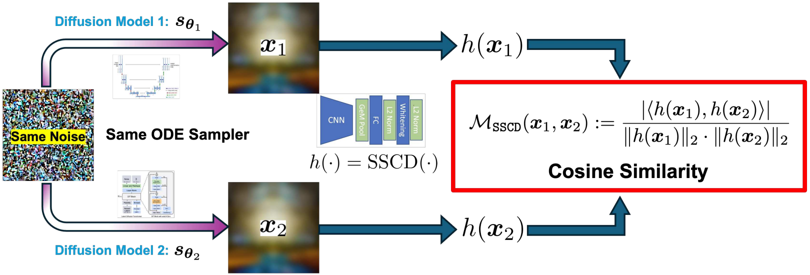
Starting from the **same noise input**, how are the generated data samples from various diffusion models related to each other?



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

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

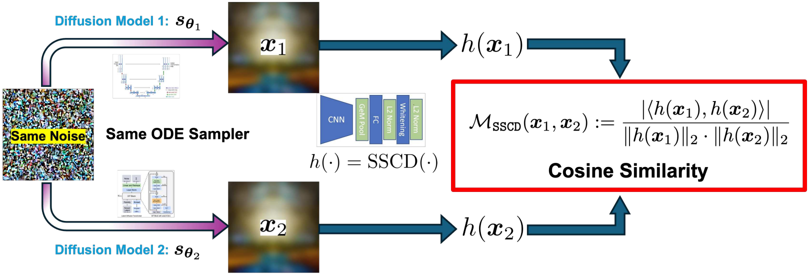
How to Measure Reproducibility Quantitatively?



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

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

How to Measure Reproducibility Quantitatively?

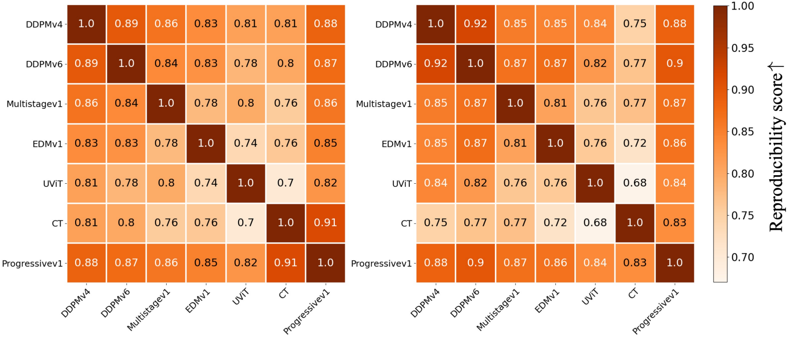


Reproducibility (RP) Score:

$$\boxed{\text{RP Score} := \mathbb{P}\left(\mathcal{M}_{\mathrm{SSCD}}(x_1, x_2) > 0.6\right).}$$

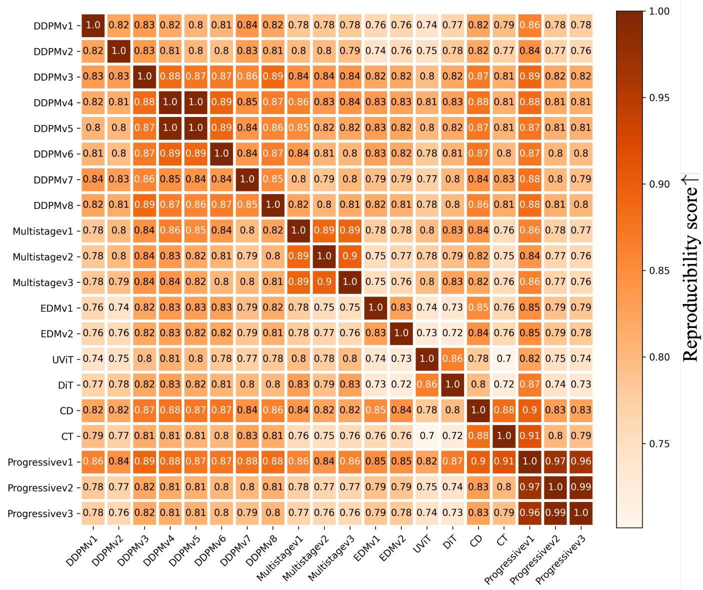
- A *probability measure* of the similarity between models.
- We sample 10^4 random noise pairs to estimate the probability.

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

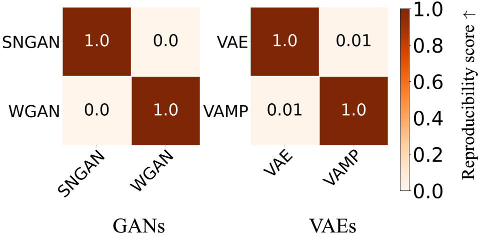


Figure 4: Reproducibility for GANs and VAEs.

- 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

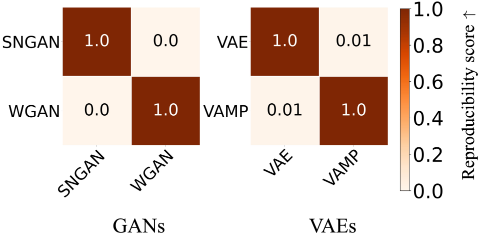
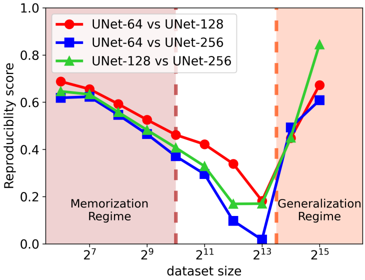


Figure 4: Reproducibility for GANs and VAEs.

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

Reproducibility Manifests in Two Different Regimes

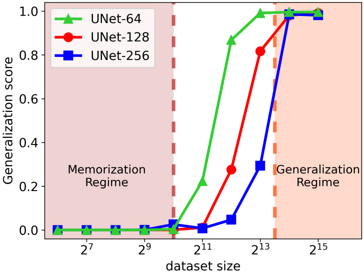


Reproducibility (RP) Score:

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

Higher implies better reproducibility between two diffusion models.

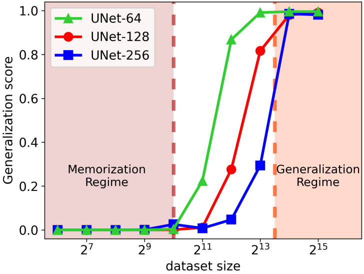
Reproducibility Manifests in Two Different Regimes



Generalization (GL) score is defined to

> measure the difference between a **newly generated sample** x and the **whole training dataset** $\mathcal{S} = \left\{ \boldsymbol{x}^{(i)} \right\}_{i=1}^{N}$.

Reproducibility Manifests in Two Different Regimes

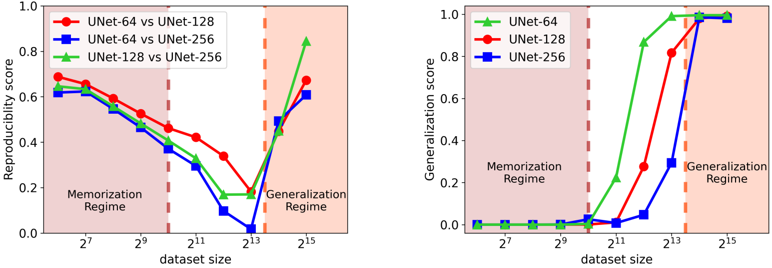


Generalization (GL) score (or perhaps memorization score?)

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

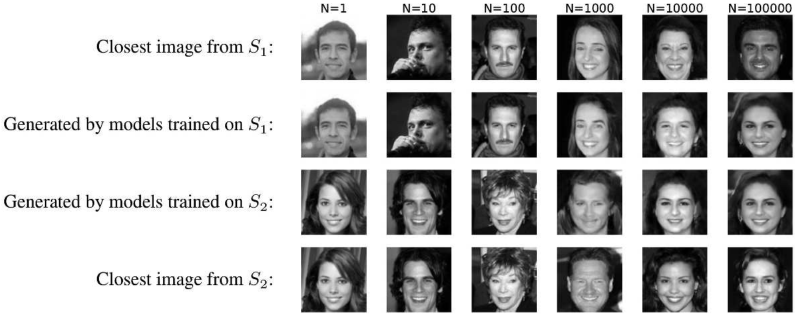
is also a *probability* measure. Higher implies better generalizability.

From “Memorization” to “Generalization”



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

Complementary Results from Concurrent Work[9]

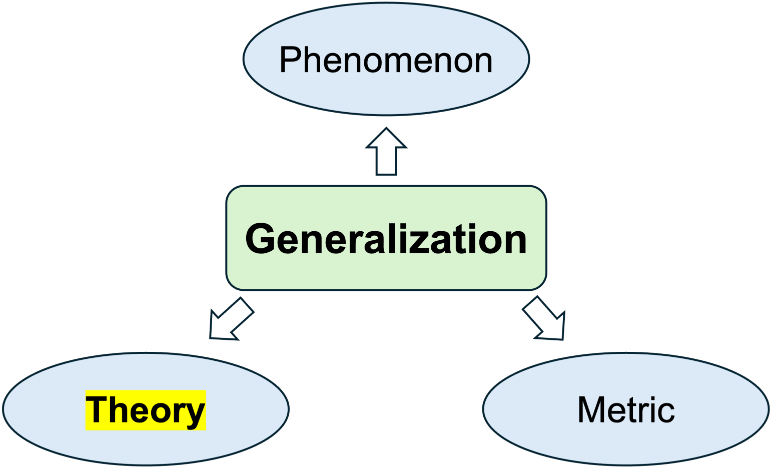


Non-overlapping subsets from the same distribution:

- the same model is trained on disjoint parts $\mathcal{S}_1$, $\mathcal{S}_2$ of the dataset
- the image is generated from the same initial noise

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

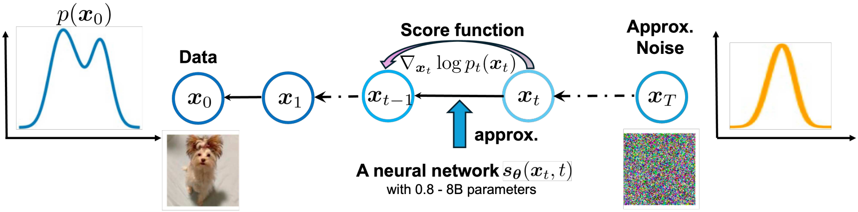
Outline[10]



[10] 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.

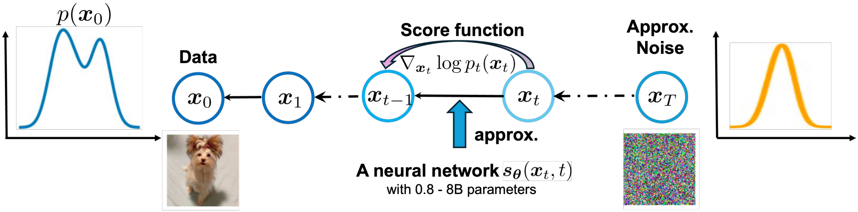
Theory of Generalization via Low-dimensional Models

Why Does Reproducibility Manifest in Distinct Regimes?



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

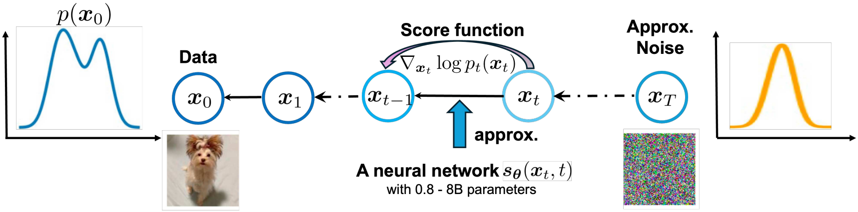
Why Does Reproducibility Manifest in Distinct Regimes?



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

- **Score approximation.** Reproducibility implies that $s_{\boldsymbol{\theta}}$ can well approximate the score function $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$.

Why Does Reproducibility Manifest in Distinct Regimes?



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

- **Score approximation.** Reproducibility implies that $s_{\boldsymbol{\theta}}$ can well approximate the score function $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$.
- **Learning different scores in distinct regimes.** But for which $p(\boldsymbol{x}_0)$ are we learning the score function?

Learning Empirical Distribution in Memorization Regime

Definition

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

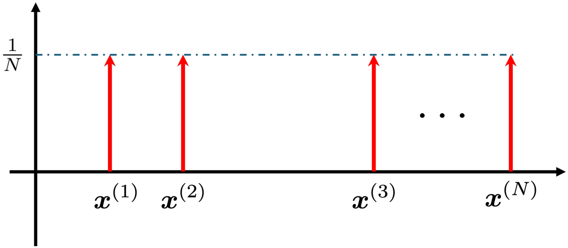
$$p_{\mathsf{emp}}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x} - \boldsymbol{x}^{(i)}).$$

Learning Empirical Distribution in Memorization Regime

Definition

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

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



Interpolation/Extrapolation of True Data Distribution

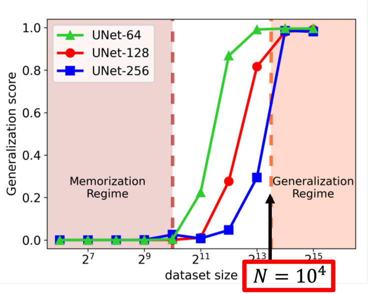
The curse of dimensionality: for image dataset (e.g., CelebA, Cifar),

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

to be ε-close,[11] we need **extremely large** $N \geq (L/\varepsilon)^d$!

[11] 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),

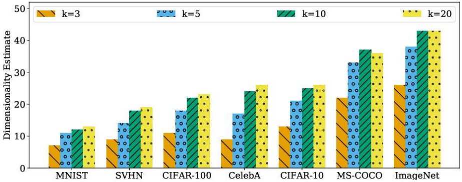
$$p_{\text{emp}}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x}-\boldsymbol{x}^{(i)}) \;\approx\; p_{\text{data}}(\boldsymbol{x}),$$

to be ε-close,[11] we need **extremely large** $N \geq (L/\varepsilon)^d$!

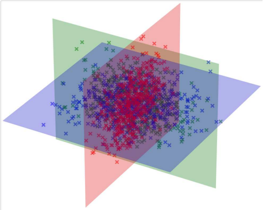
[11] We can draw this conclusion by a simple covering argument, the image dimension $d = 32 \times 32 = 1024$ for Cifar. See also recent work by Li et al., 2024.

The Intrinsic Low-Dimensionality of Data[12]

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



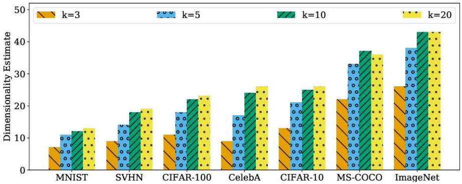
Intrinsic dimension of image datasets.



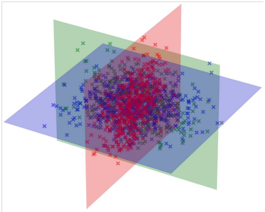
[12] Image credit: P. Pope et al., ICLR'2021.

The Intrinsic Low-Dimensionality of Data[12]

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



Intrinsic dimension of image datasets.



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

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

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, $\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_0 + \beta_t \boldsymbol{\epsilon}$:

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

Intrinsic Low-Dimensionality of the Model

- Denoising autoencoder (DAE) formulation:

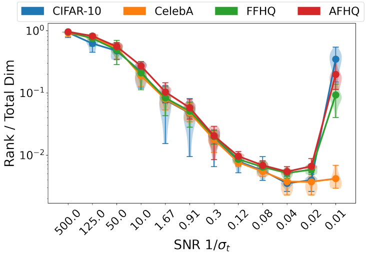
$$\min_{\boldsymbol{\theta}} \ \ell(\boldsymbol{\theta}) := \sum_{i=1}^{N} \int_0^1 \lambda_t \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_n)} \left[\left\| \boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \boldsymbol{x}^{(i)} \right\|^2 \right] \mathrm{d}t$$

- Tweedie's formula, $\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_0 + \beta_t \boldsymbol{\epsilon}$:

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

- Evaluate 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:

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

- Tweedie's formula, $\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_0 + \beta_t \boldsymbol{\epsilon}$:

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

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

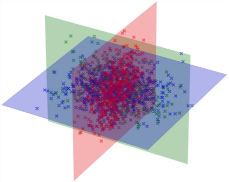
Study Generalization under Low-Dimensional Models[13]

Data Assumption

Mixture of low-rank Gaussian (MoLRG):

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

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 $\boldsymbol{U}_i \perp \boldsymbol{U}_j (i \neq j)$.



[13]Chen et al. Score Approximation, Estimation and Distribution Recovery of Diffusion Models on Low-Dimensional Data. *ICML*, 2023.

Study Generalization under Low-Dimensional Models

Data Assumption

Mixture of low-rank Gaussian (MoLRG):

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

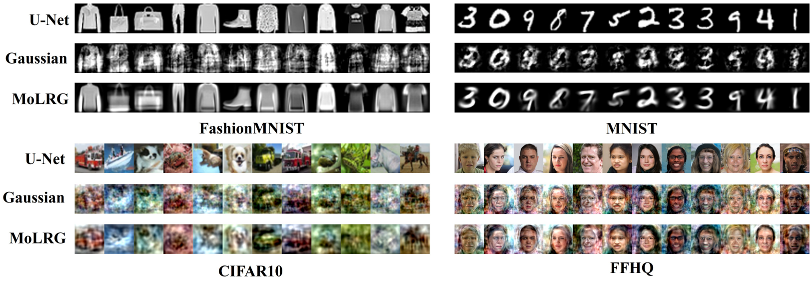
where K is the number of clusters, and $\boldsymbol{U}_i \in \mathbb{R}^{d \times r}$ is the low-rank basis for the ith cluster with $r \ll d$, with $\boldsymbol{U}_i \perp \boldsymbol{U}_j (i \neq j)$.

Lemma 1. Suppose that $p_{\text{data}}(\boldsymbol{x}_0) \sim$ 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 $\phi_t := \alpha_t^2 / (2\beta_t^2(\alpha_t^2 + \beta_t^2))$.

MoLRG Serves as a Good Approximation of Image Distributions



DAE-generated images with different parameterizations:

- **1st row:** U-Net parameterization;
- **2nd row:** optimal denoiser of a single Gaussian;
- **3rd row:** optimal denoiser of MoLRG (see Lemma 1).

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

Theorem (Equivalence to PCA)

Suppose that

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

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

Let $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}^{(i)} & \cdots & \boldsymbol{x}^{(N)} \end{bmatrix}$ be the training data matrix, then

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

Theorem (Equivalence to PCA)

Suppose that

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

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

Let $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}^{(i)} & \cdots & \boldsymbol{x}^{(N)} \end{bmatrix}$ be the training data matrix, then

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

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

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 $\boldsymbol{x}_{\boldsymbol{U}}(\boldsymbol{x}_t, t)$:

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

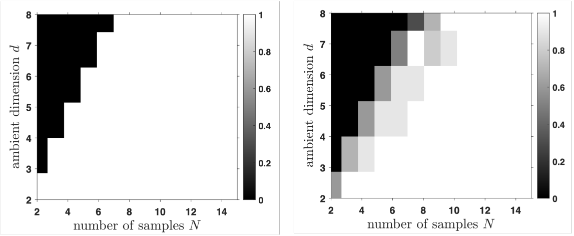
Let $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}^{(i)} & \cdots & \boldsymbol{x}^{(N)} \end{bmatrix}$ be the training data matrix, then

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

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

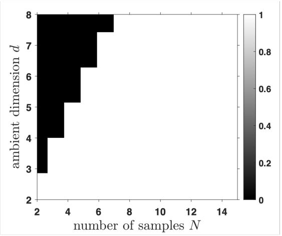
- Thus, it holds for the global solution $\boldsymbol{U}_\star$ w.h.p. that
  (i) If $N \geq r$, we have $\|\boldsymbol{U}_\star \boldsymbol{U}_\star^\top - \boldsymbol{U}_g \boldsymbol{U}_g^\top\|_F < \delta$;
  (ii) If $N < r$, we have $\|\boldsymbol{U}_\star \boldsymbol{U}_\star^\top - \boldsymbol{U}_g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

Phase Transitions on `MoLRG` with Parameterized Networks

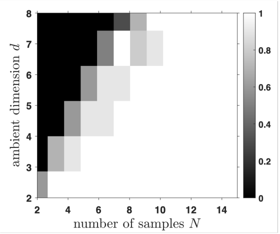


(a) PCA **(b) DM (K=1)**

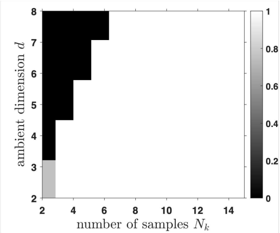
Phase Transitions on `MoLRG` with Parameterized Networks



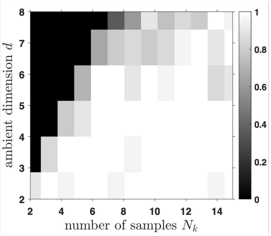
(a) PCA



(b) DM (K=1)



(c) Subspace Clustering



(d) DM (K=2)

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

Theorem (Equivalence to Subspace Clustering)

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

- $p(\boldsymbol{x}_0)$ 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 training problem is equivalent to **subspace clustering**

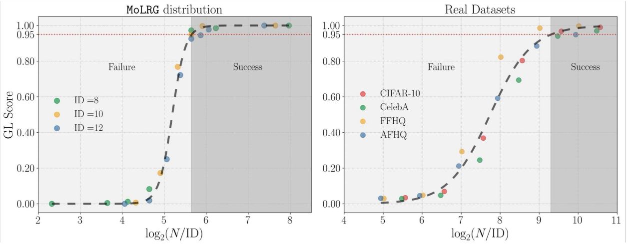
$$\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k(\boldsymbol{\theta})} \|\boldsymbol{U}_k^\top \boldsymbol{x}^{(i)}\|^2 \quad \text{s.t.} \quad \left[\boldsymbol{U}_1, \ldots, \boldsymbol{U}_K\right] \in \mathcal{O}^{n \times dK},$$

with $\mathcal{C}_k(\boldsymbol{\theta}) := \left\{ i \in [N] : \|\boldsymbol{U}_k^\top \boldsymbol{x}^{(i)}\| \geq \|\boldsymbol{U}_l^\top \boldsymbol{x}^{(i)}\|, \ \forall l \neq k \right\}$ for $k \in [K]$. The sample complexity is **linear** between N and d.

Phase Transition of Generalization: $N \geq c \cdot \mathrm{ID}$

Training U-Net model with fixed capacity on synthetic & real dataset:

Fitting N vs. ID

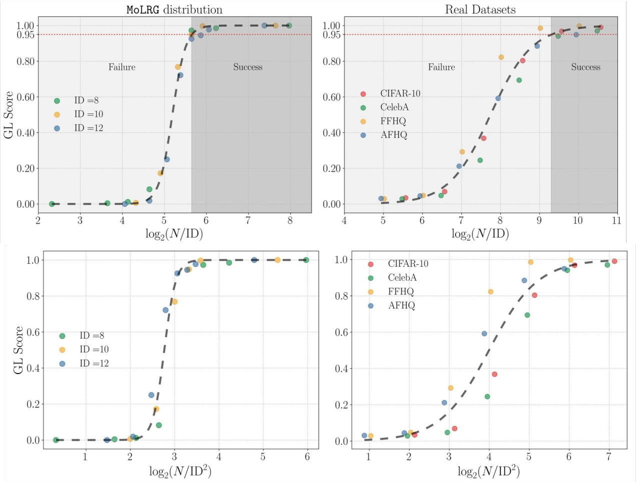


Phase Transition of Generalization: $N \geq c \cdot \mathrm{ID}$

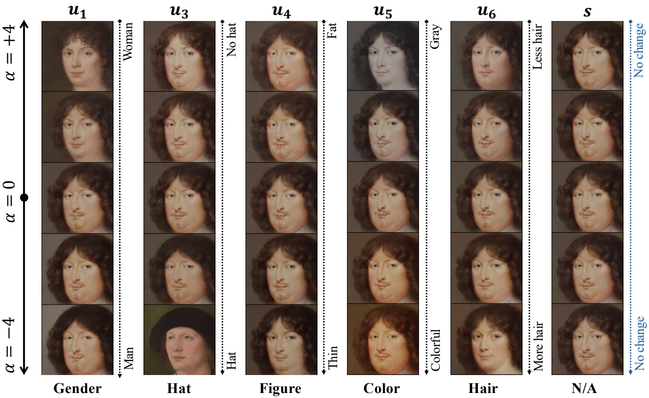
Training U-Net model with fixed capacity on synthetic & real dataset:

Fitting N vs. ID

Fitting N vs. ID^2

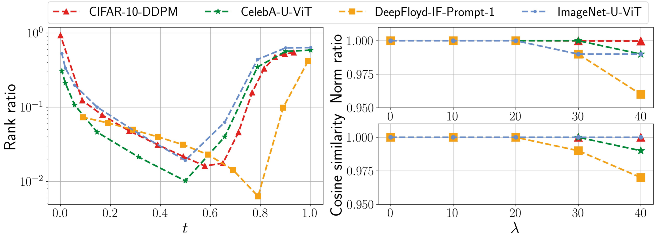


Low-Dimensional Bases are Concept Steering Vectors



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

Inductive Bias Towards "Simple" Solutions[14]



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

The trained network via Adam tends to have simple structures:

- **Low-rankness** of the Jacobian:

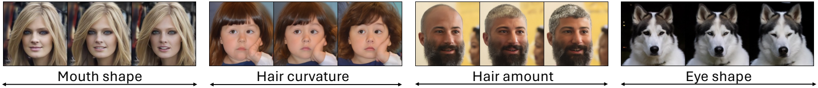
$$\boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{u}_i^\top.$$

- **Local linearity** of the DAE:

$$\boldsymbol{x}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t + \lambda \Delta \boldsymbol{x}) \approx \boldsymbol{x}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) + \lambda \boldsymbol{J}_{\boldsymbol{\theta},t}(\boldsymbol{x}_t) \cdot \Delta \boldsymbol{x}$$

[14] X. Li, Y. Dai, Q. Qu. Understanding Generalizability of Diffusion Models Requires Rethinking the Hidden Gaussian Structure. *NeurIPS*, 2024.

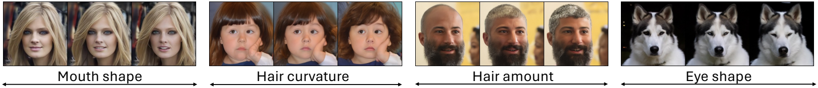
LOw-rank COntrollable Image Editing (LOCO Edit)[15]



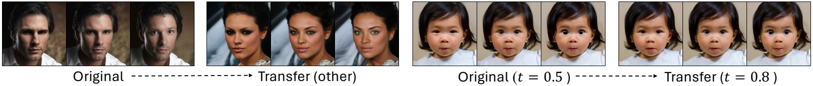
(a) Precise and Localized

[15] 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.

LOw-rank COntrollable Image Editing (LOCO Edit)[15]



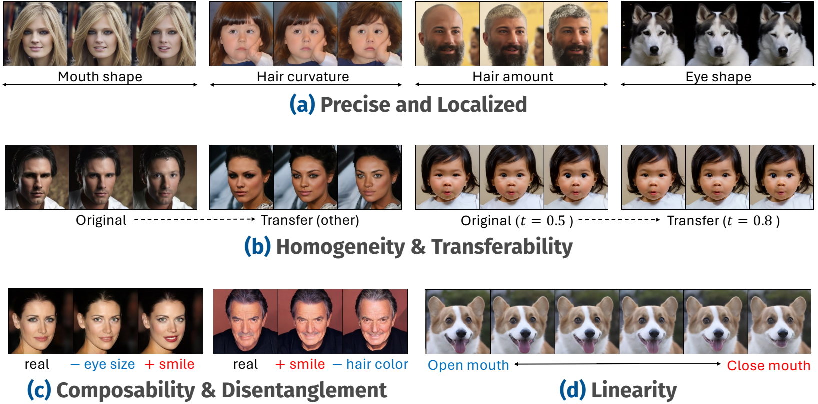
(a) Precise and Localized



(b) Homogeneity & Transferability

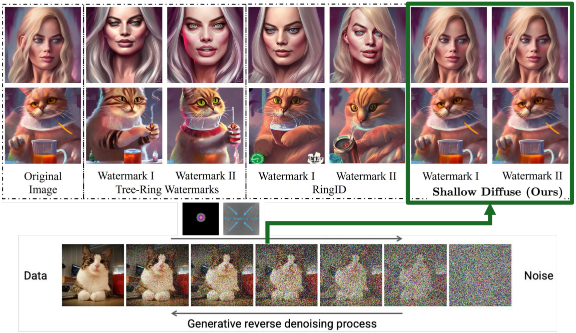
[15] S. Chen*, H. Zhang*, M. Guo, Y. Lu, P. Wang, and Q. Qu. Exploring Low-Dimensional Subspaces in Diffusion Models for Controllable Image Editing. NeurIPS, 2024.

LOw-rank COntrollable Image Editing (LOCO Edit)[15]



[15] 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.

Robust and Invisible Watermarking via Shallow Diffuse[16]

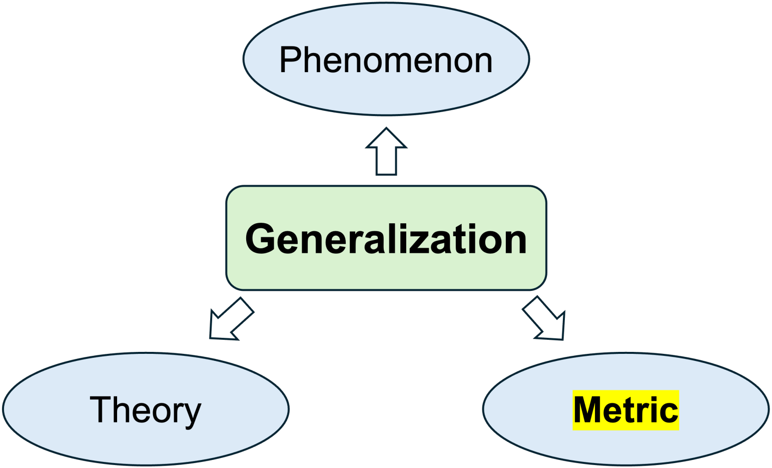


Key idea: Inject the watermark Δx in the **Null Space** of $J_{\theta,t}(x_t)$:

$$x_{\theta,t}(x_t^{\mathcal{W}}) = x_{\theta,t}(x_t) + \boxed{\underbrace{\lambda J_{\theta,t}(x_t) \cdot \Delta x}_{\approx \mathbf{0}}} \approx x_{\theta,t}(x_t)$$

[16] W. Li, H. Zhang, Q. Qu. Shallow Diffuse: Robust and Invisible Watermarking through Low-Dimensional Subspaces in Diffusion Models. NeurIPS'25, (**spotlight, top 3%**)

Outline[17]



[17] H. Zhang, Z. Huang, S. Chen, J. Zhou, Z. Zhang, P. Wang, Q. Qu. Understanding Generalization in Diffusion Models via Probability Flow Distance. *Arxiv Preprint arXiv:2505.20123*, 2025.

Quantifying Generalization via Probability Flow Distance

How to Quantify Generalization for GenAI Models?

In practice, measuring generalization errors is hard **without the groundtruth** distribution $p_{\text{data}}(\boldsymbol{x})$. Previously, we defined GL Score:

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

- It measures the **dissimilarity** between generated x and the whole training dataset $\mathcal{S} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$.

How to Quantify Generalization for GenAI Models?

In practice, measuring generalization errors is hard **without the groundtruth** distribution $p_{\text{data}}(\boldsymbol{x})$. Previously, we defined GL Score:

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

- It measures the **dissimilarity** between generated $\boldsymbol{x}$ and the whole training dataset $\mathcal{S} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$.
- However, it **alone cannot** correctly quantify generalization: A **randomly drawn** sample $\boldsymbol{x}$ can be misspecified!

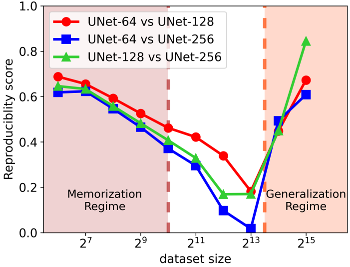
How to Quantify Generalization for GenAI Models?

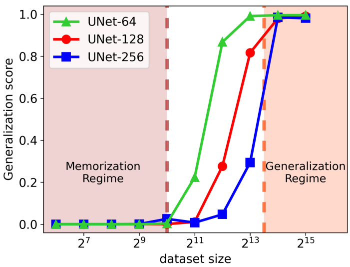
In practice, measuring generalization errors is hard **without the groundtruth** distribution $p_{\mathrm{data}}(\boldsymbol{x})$. Previously, we defined GL Score:

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

- It measures the **dissimilarity** between generated x and the whole training dataset $\mathcal{S} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$.
- However, it **alone cannot** correctly quantify generalization: A **randomly drawn** sample x can be misspecified!
- Instead, we can predict generalization by **combining** GL Score and RP Score.

Reproducibility vs. Generalization





Reproducibility (RP) Score:

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

Higher implies better reproducibility between two diffusion models.

Quantify Generalization Error $s_{\boldsymbol{\theta}} \approx \nabla \log p_{\mathrm{data}}$?

Without p_{data}, we study generalization under **teacher-student** setup:

Quantify Generalization Error $s_{\boldsymbol{\theta}} \approx \nabla \log p_{\text{data}}$?

Without p_{data}, we study generalization under **teacher-student** setup:

- **Teacher model $s_{\boldsymbol{\theta}_\star}$:** a large pre-trained model to generate training dataset $\mathcal{S} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$; serve as a proxy of p_{data}.

Quantify Generalization Error $s_{\boldsymbol{\theta}} \approx \nabla \log p_{\text{data}}$?

Without p_{data}, we study generalization under **teacher-student** setup:

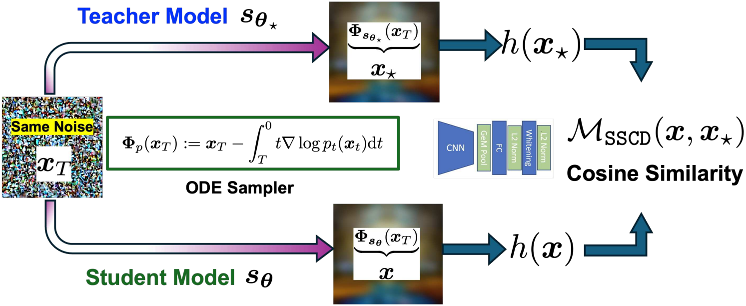
- **Teacher model $s_{\boldsymbol{\theta}_\star}$:** a large pre-trained model to generate training dataset $\mathcal{S} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$; serve as a proxy of p_{data}.
- **Student model $s_{\boldsymbol{\theta}}$**: a smaller model trained on $\mathcal{S}$.

Quantify Generalization Error $s_{\theta} \approx \nabla \log p_{\text{data}}$?

Without p_{data}, we study generalization under **teacher-student** setup:

- **Teacher model $s_{\theta_\star}$:** a large pre-trained model to generate training dataset $\mathcal{S} = \{x^{(i)}\}_{i=1}^{N}$; serve as a proxy of p_{data}.
- **Student model** s_{θ}: a smaller model trained on $\mathcal{S}$.
- **Evaluation:** We collect noise $\mathcal{T} = \{x_T^{(i)}\}_{i=1}^{M}$ distinct from training set $\mathcal{S}$ to compare reproducibility between s_{θ} and $s_{\theta_\star}$!

Quantify Generalization Error $s_\theta \approx \nabla \log p_{\text{data}}$?



Without p_{data}, we study generalization under **teacher-student** setup:

- **Teacher model** $s_{\theta_\star}$**:** a large pre-trained model to generate training dataset $\mathcal{S} = \{x^{(i)}\}_{i=1}^N$; serve as a proxy of p_{data}.
- **Student model** s_θ: a smaller model trained on $\mathcal{S}$.
- **Evaluation:** We collect noise $\mathcal{T} = \{x_T^{(i)}\}_{i=1}^M$ distinct from training set $\mathcal{S}$ to compare reproducibility between s_θ and $s_{\theta_\star}$!

Quantifying Generalization via Probability Flow Distance (PFD)

Definition (Probability Flow Distance)

Let $p_{\boldsymbol{\theta}}$ denote the distribution induced by $s_{\boldsymbol{\theta}}$, and similarly $p_{\boldsymbol{\theta}_\star}$ for $s_{\boldsymbol{\theta}_\star}$. We measure the distance between $p_{\boldsymbol{\theta}}$ and $p_{\boldsymbol{\theta}_\star}$ by

$$\mathtt{PFD}\left(p_{\boldsymbol{\theta}}, p_{\boldsymbol{\theta}_\star}\right) := \left(\mathbb{E}_{\boldsymbol{x}_T}\left[\left\|h \circ \boldsymbol{\Phi}_{p_{\boldsymbol{\theta}}}\left(\boldsymbol{x}_T\right) - h \circ \boldsymbol{\Phi}_{p_{\boldsymbol{\theta}_\star}}\left(\boldsymbol{x}_T\right)\right\|_2^2\right]\right)^{\frac{1}{2}},$$

where $\boldsymbol{x}_T \sim \mathcal{N}(0, T^2\boldsymbol{I})$, and $\boldsymbol{\Phi}_p\left(\boldsymbol{x}_T\right) = \boldsymbol{x}_T - \int_T^0 t\nabla \log p_t(\boldsymbol{x}_t)\mathrm{d}t$.

Quantifying Generalization via Probability Flow Distance (PFD)

Definition (Probability Flow Distance)

Let $p_{\boldsymbol{\theta}}$ denote the distribution induced by $s_{\boldsymbol{\theta}}$, and similarly $p_{\boldsymbol{\theta}_\star}$ for $s_{\boldsymbol{\theta}_\star}$. We measure the distance between $p_{\boldsymbol{\theta}}$ and $p_{\boldsymbol{\theta}_\star}$ by

$$\texttt{PFD}\left(p_{\boldsymbol{\theta}}, p_{\boldsymbol{\theta}_\star}\right) := \left(\mathbb{E}_{\boldsymbol{x}_T}\left[\left\|h \circ \boldsymbol{\Phi}_{p_{\boldsymbol{\theta}}}\left(\boldsymbol{x}_T\right) - h \circ \boldsymbol{\Phi}_{p_{\boldsymbol{\theta}_\star}}\left(\boldsymbol{x}_T\right)\right\|_2^2\right]\right)^{\frac{1}{2}},$$

where $\boldsymbol{x}_T \sim \mathcal{N}(0, T^2\boldsymbol{I})$, and $\boldsymbol{\Phi}_p\left(\boldsymbol{x}_T\right) = \boldsymbol{x}_T - \int_T^0 t\nabla \log p_t(\boldsymbol{x}_t)\mathrm{d}t$.

- In practice, the expectation can be well approximated by empirical mean over finite samples $\{\boldsymbol{x}_T^{(i)}\}_{i=1}^M \overset{i.i.d.}{\sim} \mathcal{N}(0, T^2\boldsymbol{I}_n)$.

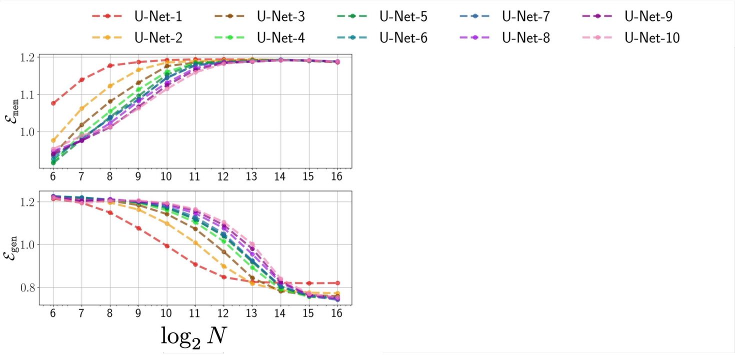
Finding I: Empirical Scaling Behavior with $\log_2(N/\sqrt{|\boldsymbol{\theta}|})$

Generalization and Memorization Errors

Suppose $\boldsymbol{s}_{\boldsymbol{\theta}}$ is trained on $\mathcal{S} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N} \overset{i.i.d.}{\sim} p_{\boldsymbol{\theta}_\star}$, and let $p_{\texttt{emp}}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x} - \boldsymbol{x}^{(i)})$. We define

$$\mathcal{E}_{\text{gen}}(\boldsymbol{\theta}) \coloneqq \texttt{PFD}(p_{\boldsymbol{\theta}}, p_{\boldsymbol{\theta}_\star}), \quad \mathcal{E}_{\text{mem}}(\boldsymbol{\theta}) \coloneqq \texttt{PFD}(p_{\boldsymbol{\theta}}, p_{\texttt{emp}}).$$

Finding I: Empirical Scaling Behavior with $\log_2(N/\sqrt{|\theta|})$

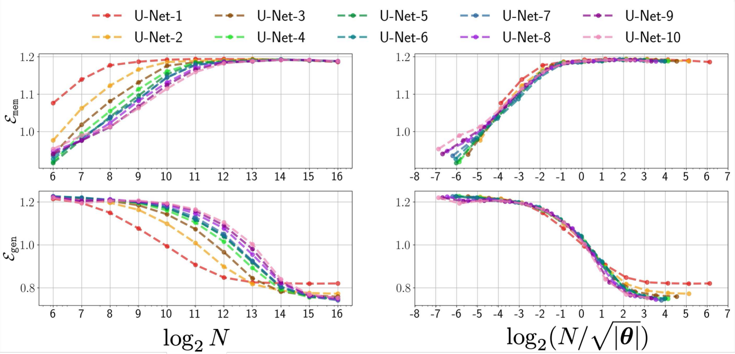


Generalization and Memorization Errors

Suppose s_{θ} is trained on $\mathcal{S} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N} \overset{i.i.d.}{\sim} p_{\theta_\star}$, and let $p_{\mathtt{emp}}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x} - \boldsymbol{x}^{(i)})$. We define

$$\mathcal{E}_{\text{gen}}(\boldsymbol{\theta}) := \mathtt{PFD}\left(p_{\boldsymbol{\theta}}, p_{\boldsymbol{\theta}_\star}\right), \quad \mathcal{E}_{\text{mem}}(\boldsymbol{\theta}) := \mathtt{PFD}\left(p_{\boldsymbol{\theta}}, p_{\mathtt{emp}}\right).$$

Finding I: Empirical Scaling Behavior with $\log_2(N/\sqrt{|\boldsymbol{\theta}|})$

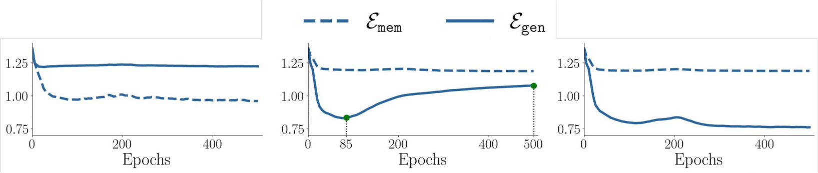


Generalization and Memorization Errors

Suppose $s_{\boldsymbol{\theta}}$ is trained on $\mathcal{S} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N} \overset{i.i.d.}{\sim} p_{\boldsymbol{\theta}_\star}$, and let $p_{\mathrm{emp}}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x}-\boldsymbol{x}^{(i)})$. We define

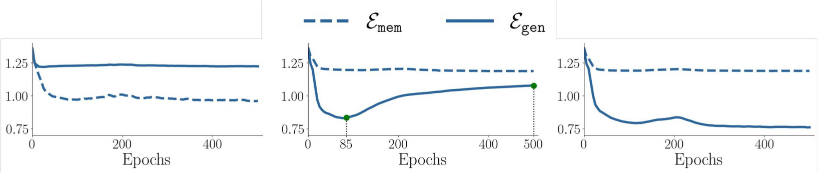
$$\mathcal{E}_{\mathrm{gen}}(\boldsymbol{\theta}) := \mathtt{PFD}\left(p_{\boldsymbol{\theta}}, p_{\boldsymbol{\theta}_\star}\right), \quad \mathcal{E}_{\mathrm{mem}}(\boldsymbol{\theta}) := \mathtt{PFD}\left(p_{\boldsymbol{\theta}}, p_{\mathtt{emp}}\right).$$

Finding II: Epoch-wise Double Descent vs. Early Learning



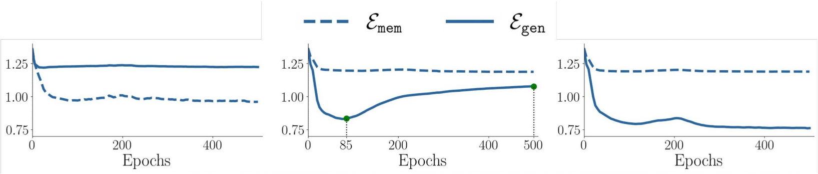
- **Generalization regime (right):** This reveals a clear **epoch-wise double descent** of the generalization error.

Finding II: Epoch-wise Double Descent vs. Early Learning



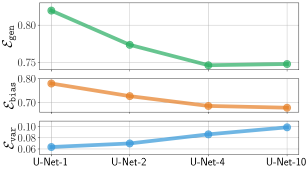
- **Generalization regime (right):** This reveals a clear **epoch-wise double descent** of the generalization error.
- **Transition regime (middle):** The **early generalization** becomes salient as the training sample size increases.

Finding II: Epoch-wise Double Descent vs. Early Learning

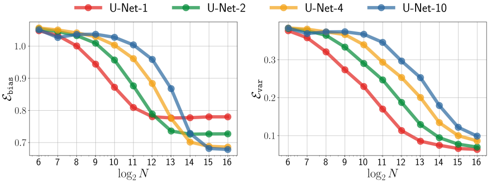


- **Generalization regime (right):** This reveals a clear **epoch-wise double descent** of the generalization error.
- **Transition regime (middle):** The **early generalization** becomes salient as the training sample size increases.

Finding III: Bias-Variance Decomposition



(a)



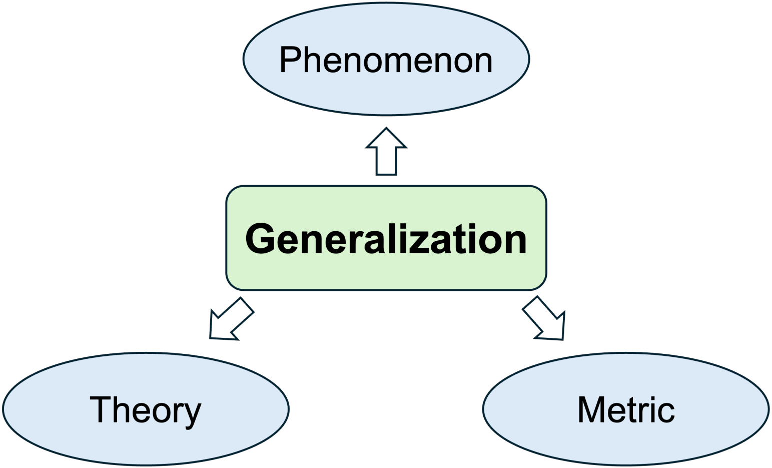
(b)

There exists a decomposition of $\mathcal{E}_{\text{gen}}$, such that

$$\mathbb{E}_{\mathcal{D}}\left[\mathcal{E}_{\text{gen}}^2\left(p_{\boldsymbol{\theta}(\mathcal{D})}\right)\right] = \mathcal{E}_{\text{bias}}^2 + \mathcal{E}_{\text{var}},$$

where $p_{\boldsymbol{\theta}(\mathcal{D})}$ denotes the distribution induced by a diffusion model $\boldsymbol{\theta}(\mathcal{D})$ trained on a given training dataset $\mathcal{D} \sim p_{\text{data}}$.

Outline



Conclusion & Discussion

Take-Home Message

- **Phenomenon:** DMs exhibit **unique reproducibility** transition from **memorization** to **generalization** regimes.

Take-Home Message

- **Phenomenon:** DMs exhibit **unique reproducibility** transition from **memorization** to **generalization** regimes.
- **Theory:** We can learn **low-dimensional data distribution** via DMs without the curse of dimensionality.

Take-Home Message

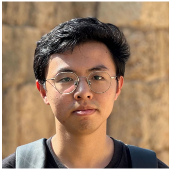
- **Phenomenon:** DMs exhibit **unique reproducibility** transition from **memorization** to **generalization** regimes.
- **Theory:** We can learn **low-dimensional data distribution** via DMs without the curse of dimensionality.
- **Metric:** We can quantify generalization via reproducibility with intriguing **generalization properties**.

Major References

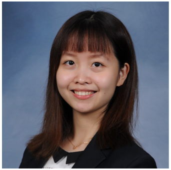
1. H. Zhang*, J. Zhou*, Y. Lu, M. Guo, P. Wang, L. Shen, and Q. Qu. The Emergence of Reproducibility and Consistency in Diffusion Models. *ICML*, 2024. (NeurIPS'23 Workshop on Diffusion Models, **Best Paper Award**)
2. P. Wang*, H. Zhang*, Z. Zhang, S. Chen, Y. Ma, and Q. Qu. Diffusion Model Learns Low-Dimensional 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.
4. X. Li, Y. Dai, Q. Qu. Understanding Generalizability of Diffusion Models Requires Rethinking the Hidden Gaussian Structure. *NeurIPS*, 2024.
5. H. Zhang*, Y. Lu*, I. Alkhouri, S. Ravishankar, D. Song, Q. Qu. Improving Efficiency of Diffusion Models via Multi-Stage Framework and Tailored Multi-Decoder Architectures. *CVPR*, 2024.

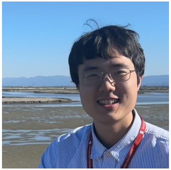
6. W. Li, H. Zhang, Q. Qu. Shallow Diffuse: Robust and Invisible Watermarking through Low-Dimensional Subspaces in Diffusion Models. *NeurIPS*, 2025 (**spotlight, top 3.2 %**).
7. H. Zhang, Z. Huang, S. Chen, J. Zhou, Z. Zhang, P. Wang, Q. Qu. Understanding Generalization in Diffusion Models via Probability Flow Distance. *Arxiv Preprint arXiv:2505.20123*, 2025.
8. X. Li*, Z. Zhang*, X. Li, S. Chen, Z. Zhu, P. Wang, and Q. Qu. Understanding Representation Dynamics of Diffusion Models via Low-Dimensional Modeling. *NeurIPS*, 2025.
9. X. Li, R. Wang, Q. Qu. Towards Understanding the Mechanisms of Classifier-Free Guidance. *NeurIPS*, 2025 (**spotlight, top 3.2 %**).
10. Lianghe Shi, Meng Wu, Huijie Zhang, Zekai Zhang, Molei Tao, Qing Qu. A Closer Look at Model Collapse: From a Generalization-to-Memorization Perspective. *Neural Information Processing Systems (NeurIPS'25)*, 2025. (**spotlight, top 3.2%**)
11. S. Chen, Y. Zhang, S. Liu, Q. Qu. The Dual Power of Interpretable Token Embeddings: Jailbreaking Attacks and Defenses for Diffusion Model Unlearning. *Arxiv Preprint arXiv:2504.21307*, 2025.

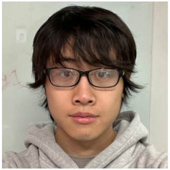
Acknowledgement


Huijie Zhang


Peng Wang

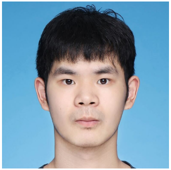

Siyi Chen

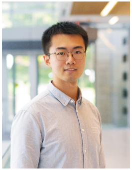

Zekai Zhang

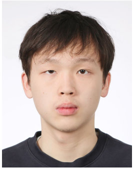

Xiang Li


Minzhe Guo

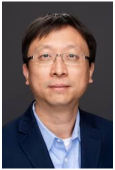

Yifu Lu


Jinfan Zhou


Lianghe Shi


Zijian Huang


Liyue Shen


Yi Ma

Acknowledgement

MICHIGAN INSTITUTE FOR
COMPUTATIONAL DISCOVERY & ENGINEERING
UNIVERSITY OF MICHIGAN

NSF National Science Foundation

amazon

DARPA

Office of Naval Research ONR Science & Technology

Google

Thank You!

SCAN ME