Deep Learning
lecture 7
Normalizing Flow & Score-Based Model

Yi Wu, IIIS

Spring 2025

Mar-31

Today’s Topic

- Normalizing Flow
  - A generative model class that has the best sampling and likelihood properties
- Score-based generative model
  - A different framework to tackle general energy-based models

Today's Topic

- Normalizing Flow
  - A generative model class that has the best sampling and likelihood properties

- Score-based generative model
  - A different framework to tackle general energy-based models

Latent Variable Model (Recap)

- $p(x, z) = p(z)p(x|z)$
  - Given z, we use $p(x|z)$ to generate x for a consistent probability distribution
  - Hard to estimate the exact likelihood $p(x) = \int_z p(x|z)p(z)$
  - Variational inference by ELBO

- Can we simply the generation process?

A Simplified Generation Process

- $p(x, z) = p(z)p(x|z)$
  - Given z, we use $p(x|z)$ to generate x for a consistent probability distribution
  - Hard to estimate the exact likelihood $p(x) = \int_z p(x|z)p(z)$
  - Variational inference by ELBO

- Can we simply the generation process?
  - We can directly apply a deterministic process $f: z \to x$
  - E.g., $z \sim N(0, I)$ and $x = f(z)$
    - $x \sim N(\mu, \sigma^2)$ is equivalent to $z \sim N(0,1), x = \mu + \sigma \cdot z$

A Simplified Generation Process

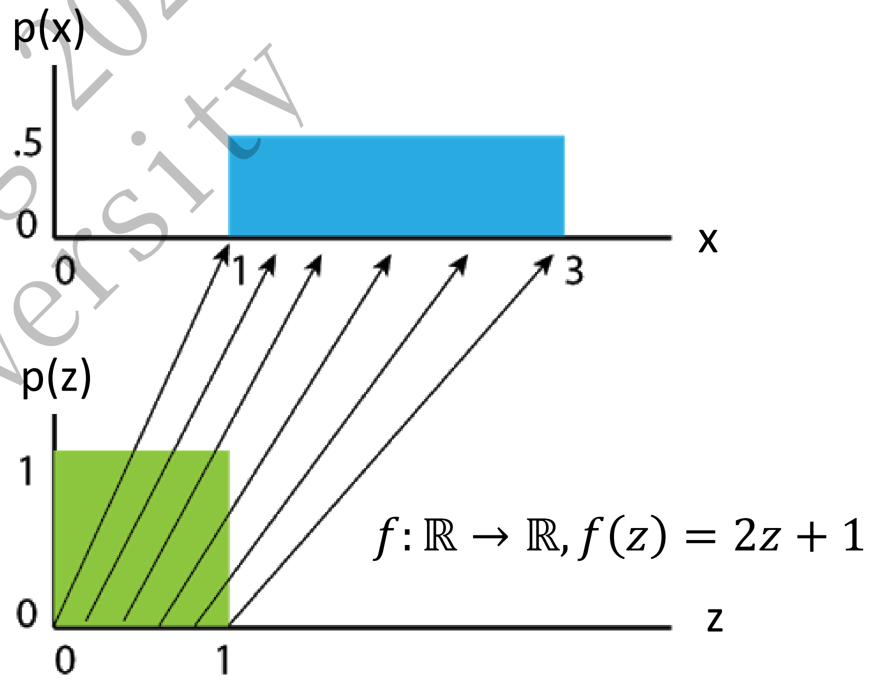
- $p(x, z) = p(z)p(x|z)$
  - Given z, we use $p(x|z)$ to generate x for a consistent probability distribution
  - Hard to estimate the exact likelihood $p(x) = \int_z p(x|z)p(z)$
  - Variational inference by ELBO

- Can we simply the generation process?
  - We can directly apply a deterministic process $f: z \to x$
  - E.g., $z \sim N(0, I)$ and $x = f(z)$

- Can we make the likelihood $p(x)$ tractable?
  - So that we can directly run MLE for training …

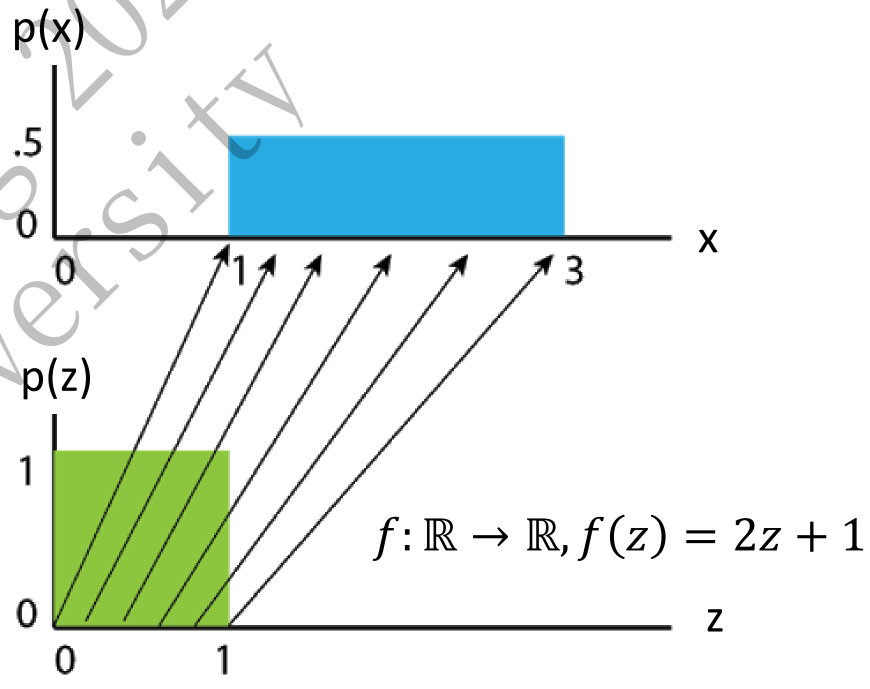
1-D Example

- Goal: design $x = f(z;\theta)$ s.t.
  - Assume z is from an "easy" distribution
  - $p(x) = p(f(z;\theta))$ has a tractable likelihood
- Uniform: $z \sim \text{unif}\,(0,1)$
  - Density $p(z) = 1$
  - $x = 2z + 1$, then $p(x) = ?$



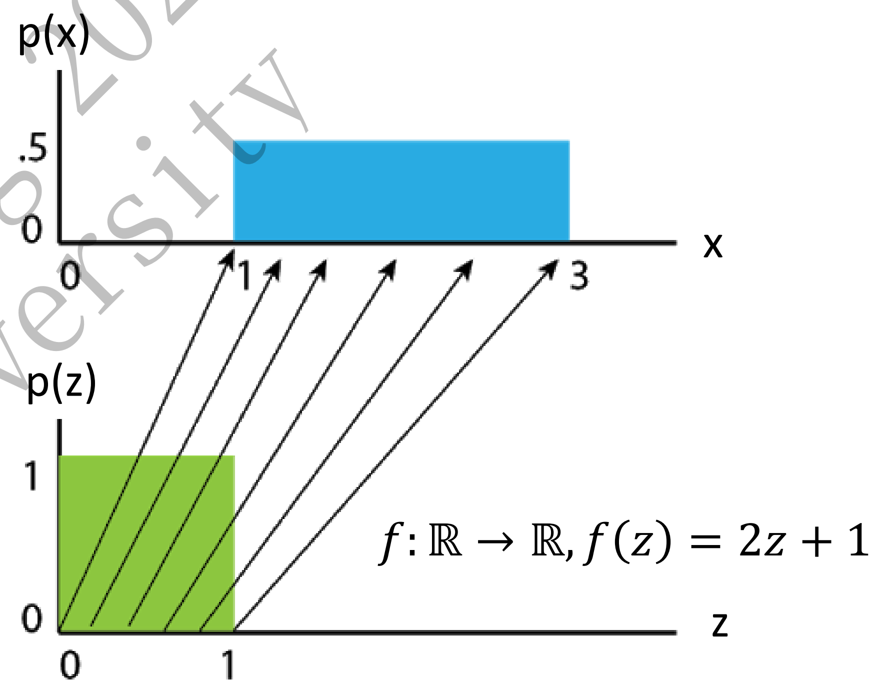
1-D Example

- Goal: design $x = f(z;\theta)$ s.t.
  - Assume z is from an "easy" distribution
  - $p(x) = p(f(z;\theta))$ has a tractable likelihood
- Uniform: $z \sim \text{unif}\,(0,1)$
  - Density $p(z) = 1$
  - $x = 2z + 1$, then $p(x) = \frac{1}{2}$
    - $x = a \cdot z + b$, then $p(x) = 1/|a|$ (for $a \neq 0$)
  - General 1-D case: $x = f(z), p(x) = ?$
    - Assume $f(z)$ is a bijection

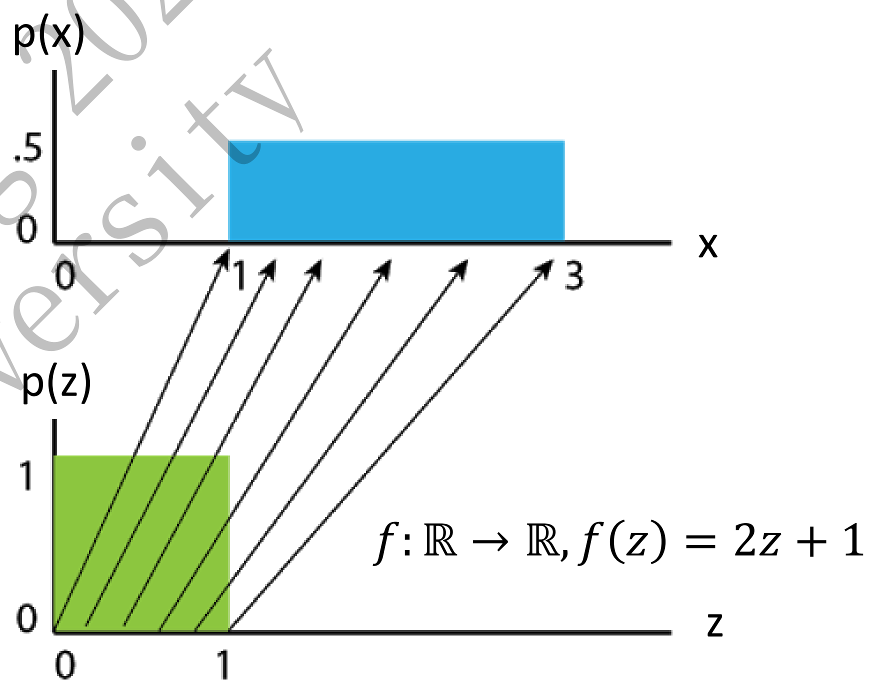


1-D Example

- Goal: design $x = f(z;\theta)$ s.t.
  - Assume z is from an "easy" distribution
  - $p(x) = p(f(z;\theta))$ has a tractable likelihood
- Uniform: $z \sim \text{unif}\,(0,1)$
  - Density $p(z) = 1$
  - $x = 2z + 1$, then $p(x) = \frac{1}{2}$
    - $x = a \cdot z + b$, then $p(x) = 1/|a|$ (for $a \neq 0$)
  - General 1-D case: $x = f(z)$, then $p(x) = p(z)\left|\frac{dz}{dx}\right|$
    - Assume $f(z)$ is a bijection
    - $p(x)dx = p(z)dz$



1-D Example



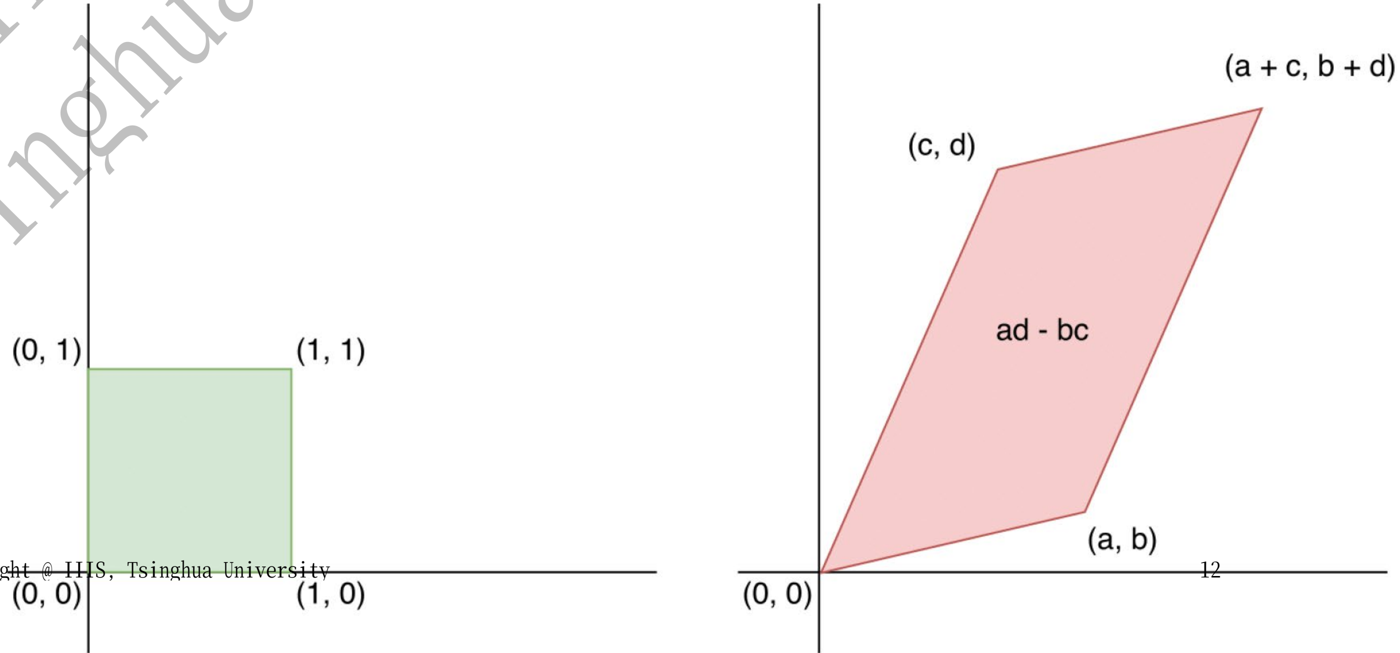
- Goal: design $x = f(z;\theta)$ s.t.
  - Assume z is from an "easy" distribution
  - $p(x) = p(f(z;\theta))$ has a tractable likelihood
- Uniform: $z \sim \text{unif}\,(0,1)$
  - Density $p(z) = 1$
  - $x = 2z + 1$, then $p(x) = \frac{1}{2}$
    - $x = a \cdot z + b$, then $p(x) = 1/|a|$ (for $a \neq 0$)
  - General 1-D case: $x = f(z)$, then $p(x) = p(z)\left|\frac{dz}{dx}\right| = |f'(z)|^{-1}p(z)$
    - Assume $f(z)$ is a bijection
    - $p(x)dx = p(z)dz$

2-D Example

- Goal: design $x = f(z; \theta)$ s.t.
  - Assume z is from an “easy” distribution
  - $p(x) = p(f(z; \theta))$ has a tractable likelihood
- Uniform: $z = [z_1, z_2] \sim \text{unif}\,([0,1] \times [0,1])$
  - Density $p(z) = 1$
  - $x = Az$, then $p(x) = ?$
    - $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

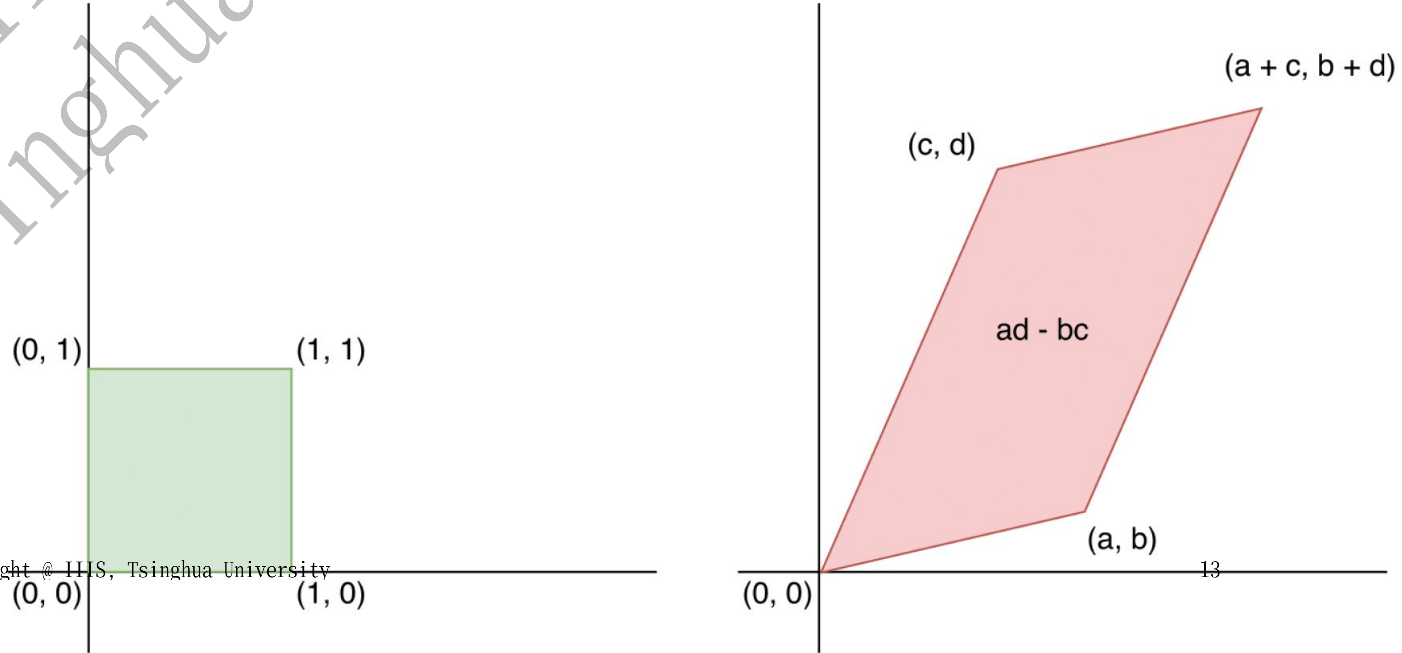
2-D Example

- Goal: design $x = f(z;\theta)$ s.t.
  - Assume z is from an "easy" distribution
  - $p(x) = p(f(z;\theta))$ has a tractable likelihood
- Uniform: $z = [z_1, z_2] \sim \text{unif}([0,1] \times [0,1])$
  - Density $p(z) = 1$
  - $x = Az$, then $p(x) = ?$
    - $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$
    - z is mapped to a parallelogram



2-D Example

- Goal: design $x = f(z;\theta)$ s.t.
  - Assume z is from an “easy” distribution
  - $p(x) = p(f(z;\theta))$ has a tractable likelihood
- Uniform: $z = [z_1, z_2] \sim \mathrm{unif}\,([0,1] \times [0,1])$
  - Density $p(z) = 1$
  - $x = Az$, then $p(x) = 1/S$
    - $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$
    - z is mapped to a parallelogram
    - $S = |ad - bc|$, the area



2-D Geometry

- The area of the parallelogram is equivalent to the determinant of A

$$\det A = \det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$

- For any linear transformation $x = Az + b$
  - The following holds (for space of any dimensions)

$$p(x) = |\det A|^{-1} \cdot p(z)$$

  - Remark: A has a full rank! (bijection)

- More general cases: the change of variable

Change of Variable

- Suppose $x = f(z)$ w.r.t. a general non-linear $f(\cdot)$
  - the linearized change in volume is determined by the Jacobian of $f(\cdot)$

$$\frac{\partial f(z)}{\partial z} = \begin{bmatrix} \frac{\partial f_1(z)}{\partial z_1} & \cdots & \frac{\partial f_1(z)}{\partial z_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_d(z)}{\partial z_1} & \cdots & \frac{\partial f_d(z)}{\partial z_d} \end{bmatrix}$$

- Given a bijection $f(z)\colon \mathbb{R}^d \rightarrow \mathbb{R}^d$
  - $z = f^{-1}(x)$

$$p(x) = p\left(f^{-1}(x)\right) \left|\det\left(\frac{\partial f^{-1}(x)}{\partial x}\right)\right| = p(z) \left|\det\left(\frac{\partial f^{-1}(x)}{\partial x}\right)\right|$$

Change of Variable

- Suppose $x = f(z)$ w.r.t. a general non-linear $f(\cdot)$
  - the linearized change in volume is determined by the Jacobian of $f(\cdot)$

$$\frac{\partial f(z)}{\partial z} = \begin{bmatrix} \frac{\partial f_1(z)}{\partial z_1} & \cdots & \frac{\partial f_1(z)}{\partial z_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_d(z)}{\partial z_1} & \cdots & \frac{\partial f_d(z)}{\partial z_d} \end{bmatrix}$$

- Given a bijection $f(z): \mathbb{R}^d \rightarrow \mathbb{R}^d$
  - $z = f^{-1}(x)$

$$p(x) = p\left(f^{-1}(x)\right)\left|\det\left(\frac{\partial f^{-1}(x)}{\partial x}\right)\right| = p(z)\left|\det\left(\frac{\partial f^{-1}(x)}{\partial x}\right)\right|$$

  - Since $\frac{\partial f^{-1}}{\partial x} = \left(\frac{\partial f}{\partial z}\right)^{-1}$ (Jacobian of invertible function)

$$p(x) = p(z)\left|\det\left(\frac{\partial f(z)}{\partial z}\right)^{-1}\right| = p(z)\left|\det\left(\frac{\partial f(z)}{\partial z}\right)\right|^{-1}$$

Change of Variable

- Suppose $x = f(z)$ w.r.t. a general non-linear $f(\cdot)$
  - the linearized change in volume is determined by the Jacobian of $f(\cdot)$

$$\frac{\partial f(z)}{\partial z} = \begin{bmatrix} \frac{\partial f_1(z)}{\partial z_1} & \cdots & \frac{\partial f_1(z)}{\partial z_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_d(z)}{\partial z_1} & \cdots & \frac{\partial f_d(z)}{\partial z_d} \end{bmatrix}$$

- Given a bijection $f(z)\colon \mathbb{R}^d \to \mathbb{R}^d$
  - $z = f^{-1}(x)$

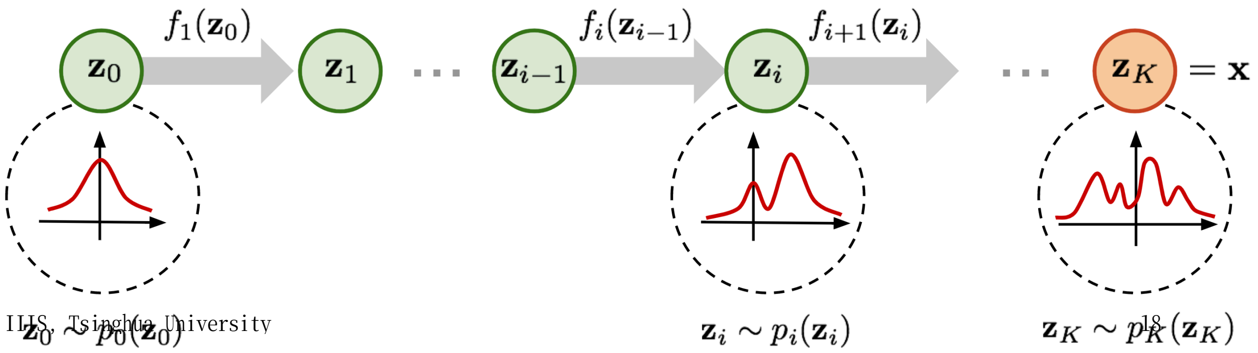
$$p(x) = p\left(f^{-1}(x)\right)\left|\det\left(\frac{\partial f^{-1}(x)}{\partial x}\right)\right| = p(z)\left|\det\left(\frac{\partial f^{-1}(x)}{\partial x}\right)\right|$$

  - Since $\frac{\partial f^{-1}}{\partial x} = \left(\frac{\partial f}{\partial z}\right)^{-1}$ (Jacobian of invertible function)

$$p(x) = p(z)\left|\det\left(\frac{\partial f(z)}{\partial z}\right)^{-1}\right| = p(z)\left|\det\left(\frac{\partial f(z)}{\partial z}\right)\right|^{-1}$$

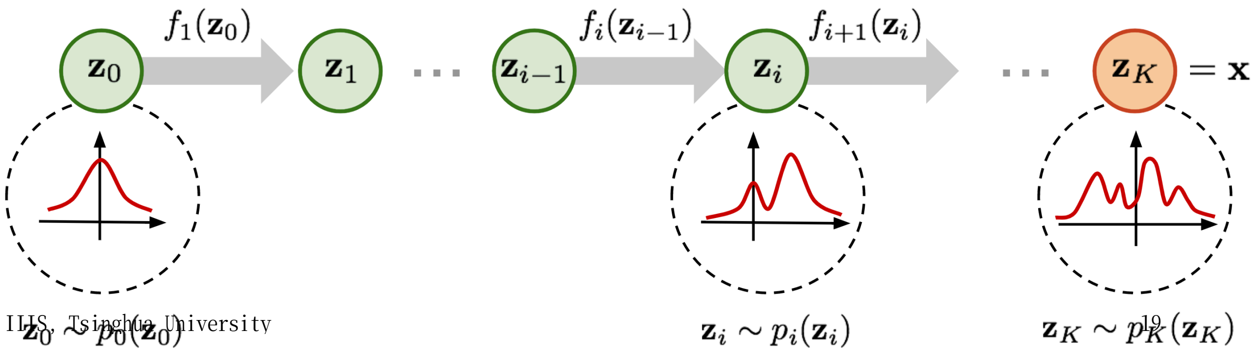
Normalizing Flow

- Idea
  - Sample z_0 from an “easy” distribution, i.e., a standard Gaussian
  - Apply K bijections $z_i = f_i(z_{i-1})\ 1 \le i \le K$
  - The final sample $x = f_K(z_K)$ has tractable density
- Normalizing Flow
  - $z_0 \sim N(0, I), z_i = f_i(z_{i-1}), x = z_K$ where $x, z_i \in \mathbb{R}^d$ & f_i is invertible
  - Every revertible function produces a normalized density function
  - $p(z_i) = p(z_{i-1}) \left|\det\left(\frac{\partial f_i}{\partial z_{i-1}}\right)\right|^{-1}$



Normalizing Flow

- Idea
  - Sample z_0 from an "easy" distribution, i.e., a standard Gaussian
  - Apply K bijections $z_i = f_i(z_{i-1})$ $1 \le i \le K$
  - The final sample $x = f_K(z_K)$ has tractable density
- Normalizing Flow
  - $z_0 \sim N(0, I), z_i = f_i(z_{i-1}), x = z_K$ where $x, z_i \in \mathbb{R}^d$ & f_i is invertible
  - Every revertible function produces a normalized density function
  - $p(z_i) = p(z_{i-1}) \left|\det\left(\frac{\partial f_i}{\partial z_{i-1}}\right)\right|^{-1}$



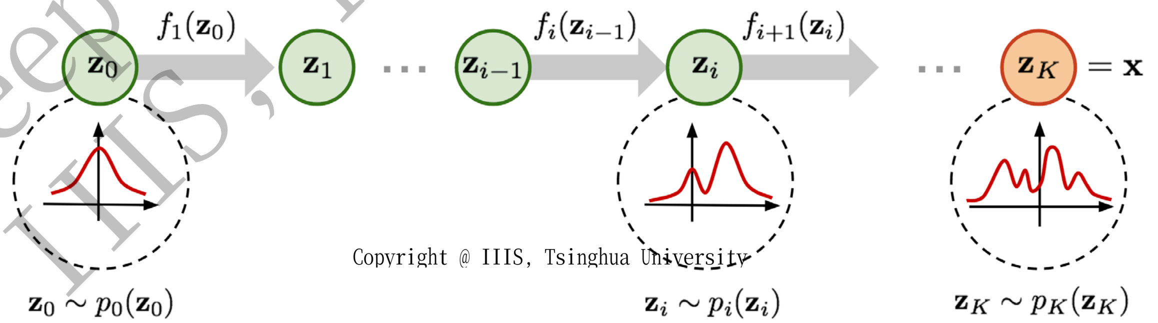
Normalizing Flow

- Generation is trivial
  - Sample z_0, then apply the transformations
- Log-Likelihood

$$\log p(x) = \log p(z_{K-1}) - \log \left|\det\left(\frac{\partial f_K}{\partial z_{K-1}}\right)\right|$$

$$\cdots$$

$$\log p(x) = \log p(z_0) - \sum_i \log \left|\det\left(\frac{\partial f_i}{\partial z_{i-1}}\right)\right|$$



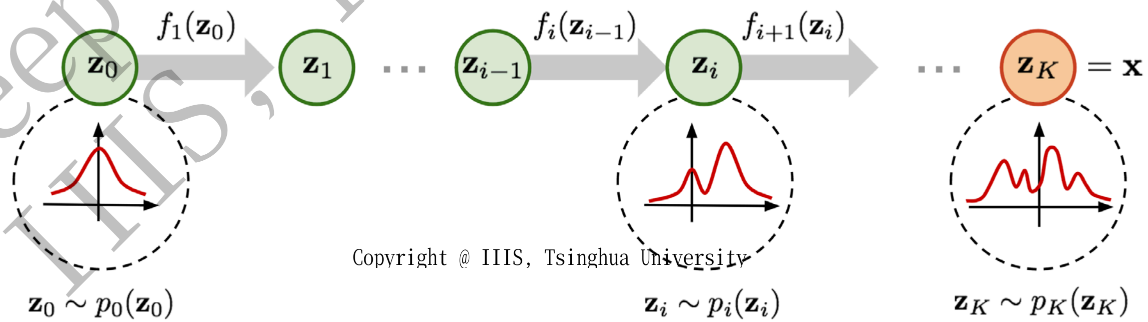
Normalizing Flow

- Generation is trivial
  - Sample z_0, then apply the transformations
- Log-Likelihood

$$\log p(x) = \log p(z_{K-1}) - \log \left|\det\left(\frac{\partial f_K}{\partial z_{K-1}}\right)\right|$$

…

$$\log p(x) = \underbrace{\log p(z_0)}_{\text{Gaussian density}} - \sum_i \log \left|\det\left(\frac{\partial f_i}{\partial z_{i-1}}\right)\right|$$



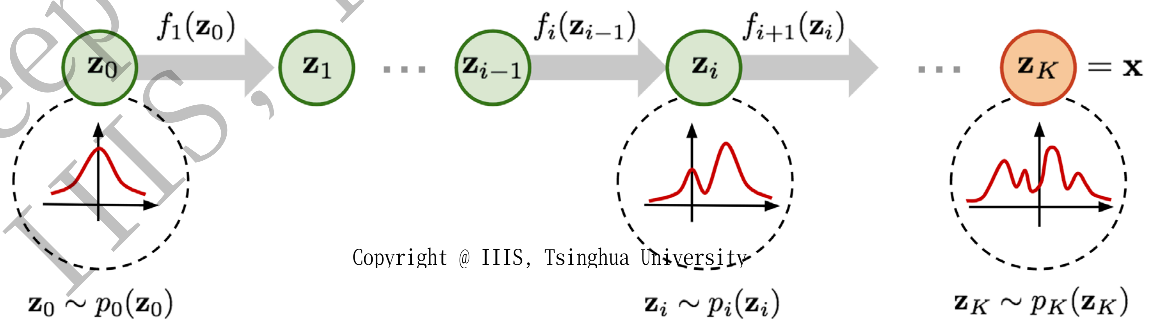
Normalizing Flow

- Generation is trivial
  - Sample z_0, then apply the transformations
- Log-Likelihood

$$\log p(x) = \log p(z_{K-1}) - \log \left|\det\left(\frac{\partial f_K}{\partial z_{K-1}}\right)\right|$$

$$\cdots$$

$$\log p(x) = \log p(z_0) - \sum_i \log \left|\det\left(\frac{\partial f_i}{\partial z_{i-1}}\right)\right| \qquad O(d^3)!!!$$



Normalizing Flow

- Naïve flow model requires extremely expensive computation
  - Determinant of a $d \times d$ matrix
- Idea
  - Design a good bijection $f_i(z)$ such that the determinant is easy to compute
- Technical Keys
  - Bijection
    - Randomly constructed matrices are typically full-rank
  - Structured Jacobian
    - Desired Jacobian structures for fast determinant computation

Triangular Jacobian

- Given $x = (x_1, \dots, x_d) = f(z) = (f_1(z), \dots, f_d(z))$

$$J = \frac{\partial f}{\partial z} = \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_d}{\partial z_1} & \cdots & \frac{\partial f_d}{\partial z_d} \end{bmatrix}$$

- Suppose $x_i = f_i(z)$ only depends on $z_{\le i}$, then

$$\det J = \det \left| \frac{\partial f}{\partial z} \right| = \det \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{\partial f_d}{\partial z_1} & \cdots & \frac{\partial f_d}{\partial z_d} \end{bmatrix} = \det \operatorname{diag}(J) = \prod_i \frac{\partial f_i}{\partial z_i}$$

NICE

- Nonlinear Independent Components Estimation (Dinh et. al, 2014)
  - $z = f(x)$
    - Notational convention for MLE learning
  - we partition x into two disjoint subsets $x_{1:m}$ and $x_{m+1:d}$ for any $1 \leq m \leq d$
  - Forward pass $x \rightarrow z$ (inference)
    - $z_{1:m} = x_{1:m}$ (identity)
    - $z_{m+1:d} = x_{m+1:d} + \mu_\theta(x_{1:m})$ (μ_θ is a neural network)
  - Backward pass $z \rightarrow x$ (sampling)
    - $x_{1:m} = z_{1:m}$ (identity)
    - $x_{m+1:d} = z_{m+1:d} - \mu_\theta(z_{1:m})$

$$J = \frac{\partial f}{\partial x} = \begin{bmatrix} I_m & 0 \\ \frac{\partial \mu}{\partial x_{1:m}} & I_{d-m} \end{bmatrix}$$

- Volume preserving transformation
  - $\det J = 1$

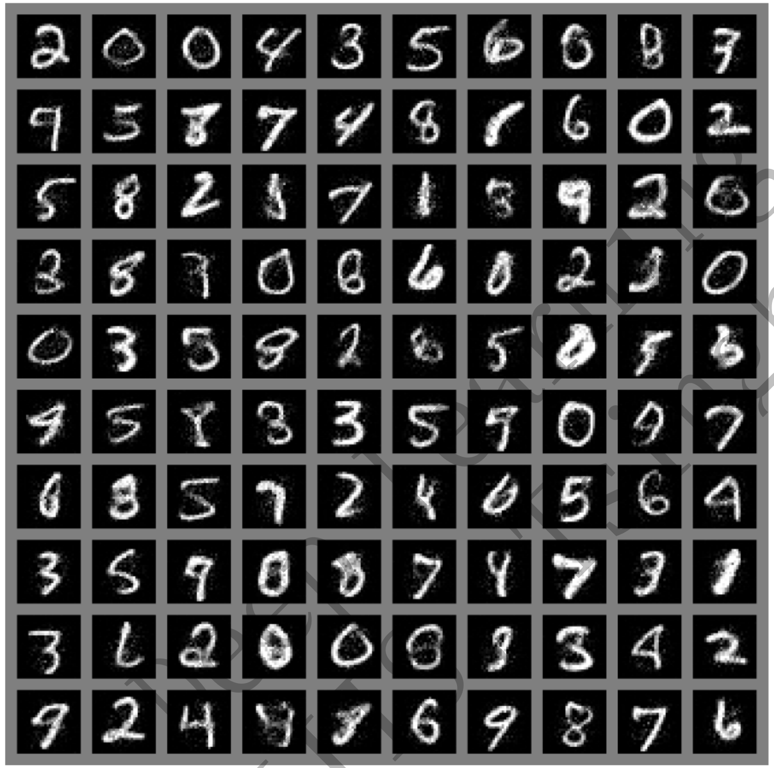
NICE

- Coupling layers are introduced to ensure all dimensions are covered
  - Reverse (or randomly shuffle) the partition before each transformation layer
- First layer of NICE uses a re-scaling layer
  - $z_i = S_{ii} x_i$
  - Ensure non-unit volume transformation
  - Jacobian of forward pass

$$J = \mathrm{diag}(S)$$
$$\det J = \prod_i S_{ii}$$

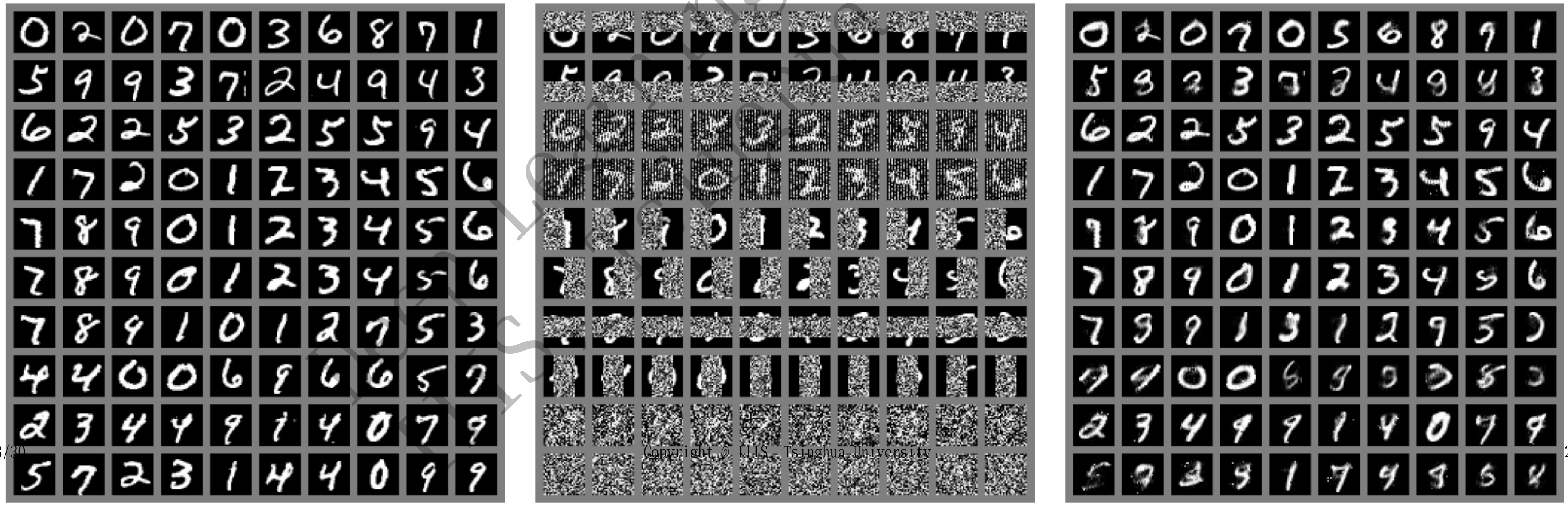
NICE

- Generation Results



NICE

- Inpainting
  - $x = (x_v, x_h)$
  - We have tractable likelihood function $p(x_v, x_h)$!
    - Gradient ascent (stochastic gradient MCMC if you want samples)



Real-NVP

- NICE: most layers maintain an *unchanged* volume
- Non-volume preserving extension of NICE (Dinh et al, 2016)
  - Two partitions over z: $z_{1:m}$ and $z_{m+1:d}$ for any $1 \leq m \leq d$
  - Forward pass $x \to z$ (inference)
    - $z_{1:m} = x_{1:m}$ (identity)
    - $z_{m+1:d} = x_{m+1:d} \cdot \exp(\alpha_\theta(x_{1:m})) + \mu_\theta(x_{1:m})$ (μ_θ & α_θ are neural networks)
  - Backward pass $z \to x$ (sampling)
    - $x_{1:m} = z_{1:m}$ (identity)
    - $x_{m+1:d} = \left(z_{m+1:d} - \mu_\theta(z_{1:m})\right) \cdot \exp(-\alpha_\theta(x_{1:m}))$
  - Non-volume preserving transformation

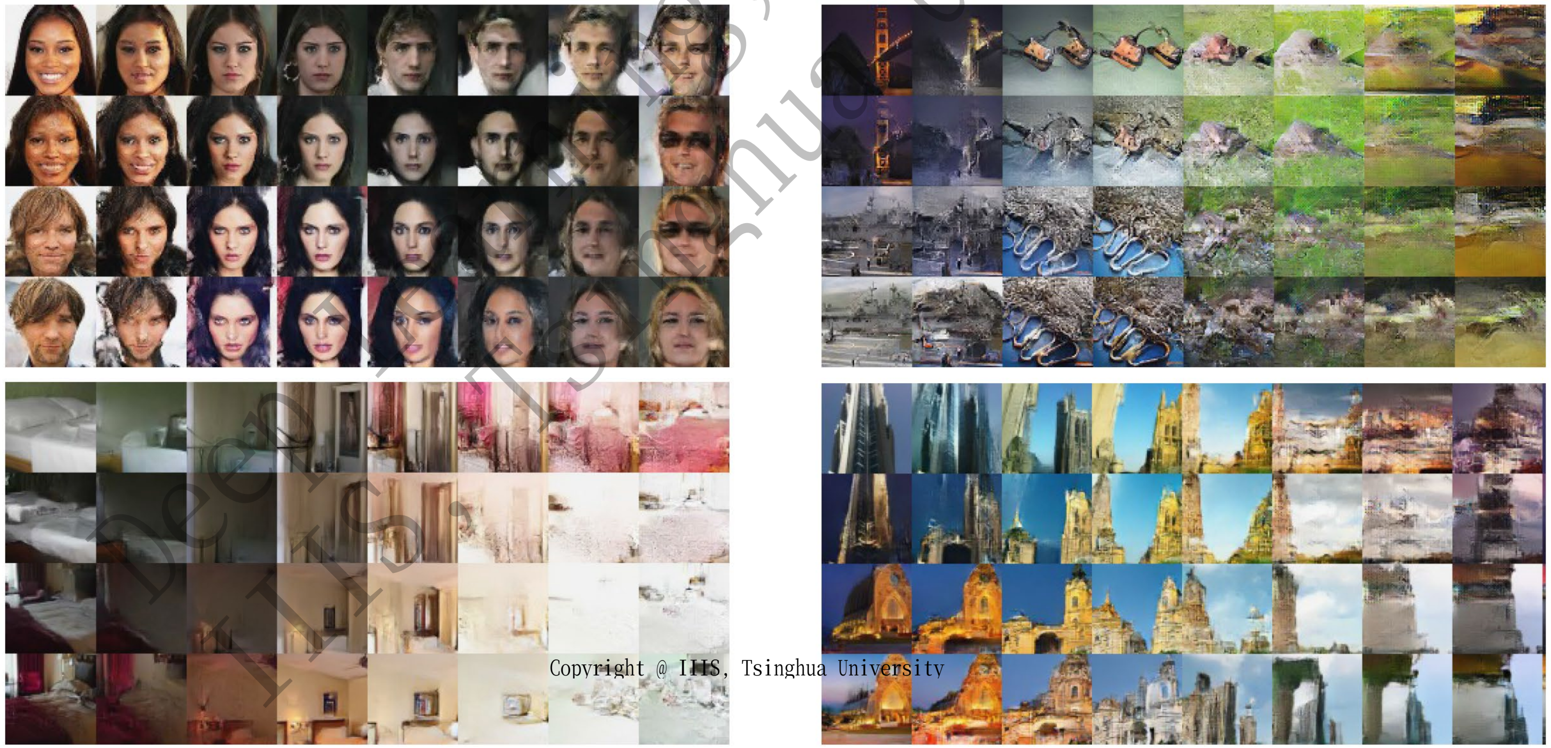
$$\det J = \prod_{i=m+1}^{d} \exp(\alpha_\theta(x_{1:m})_i)$$

Real-NVP

- Generation Results
  - Left: training data; Right: generated samples

Real-NVP

- Explore the latent space
    - 4 samples selected: z^0, z^1, z^2, z^3, two interpolation parameters α, β
    - $z = \cos(\alpha)\left(\cos(\beta)z^1 + \sin(\beta)z^2\right) + \sin(\alpha)\left(\cos(\beta)\, z^3 + \sin(\beta)z^4\right)$



Real-NVP

- Fun Fact

Accepted as a workshop contribution at ICLR 2015

NICE: Non-linear Independent Components Estimation

Laurent Dinh David Krueger Yoshua Bengio*
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Published as a conference paper at ICLR 2017

Density estimation using Real NVP

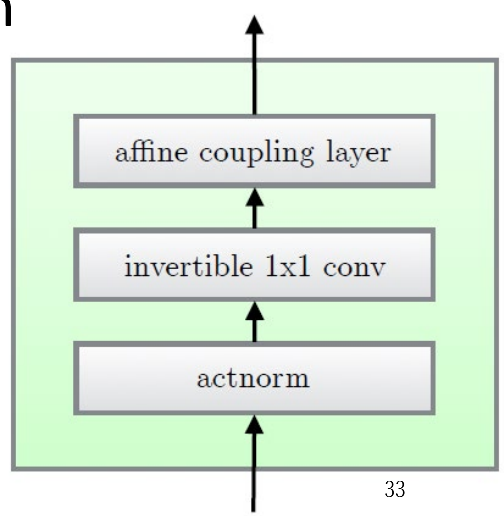
Laurent Dinh*
Montreal Institute for Learning Algorithms
University of Montreal
Montreal, QC H3T1J4

Jascha Sohl-Dickstein
Google Brain

Samy Bengio
Google Brain

GLOW

- Limited expressiveness of previous coupling layers
  - But a general non-linear transformation can be too expensive…
- Generative Flow with Invertible 1x1 Convolutions (Kingma et al. 2018)
  - Input: $x = h \times w \times c$ (height, width, channel) (assume c is small)
  - Key idea: introduce 1x1 convolutions when channel size is small
  - 1x1 conv: a linear transformation for each feature map location
    - Forward mapping: $z_{ij:} = W x_{ij:} + b$
    - Inverse mapping: simply compute the inverse matrix of W
  - Computation $O(c^3)$
    - $\log|\det J| = h \cdot w \cdot \log|\det W|$
  - Also use normalization layer to stabilizing training
    - Architecture details can be found in the paper



GLOW

- Generation Results

Normalizing Flow: Summary

- Key Ideas
  - Generation by iteratively transforming a simple distribution
  - Invertible transformation for tractable likelihood
    - Enable straightforward MLE learning
  - Design principle
    - Apply non-linear transformations with easy-to-compute Jacobian determinants
- Pros & Cons
  - Easy sampling & training via deterministic transformation from a simple distribution
  - Most restricted network structure (trade expressiveness for tractability)
    - Architecture, dimensionality, etc.
    - Most suitable for the use cases where tractability is a must

Today's Topic

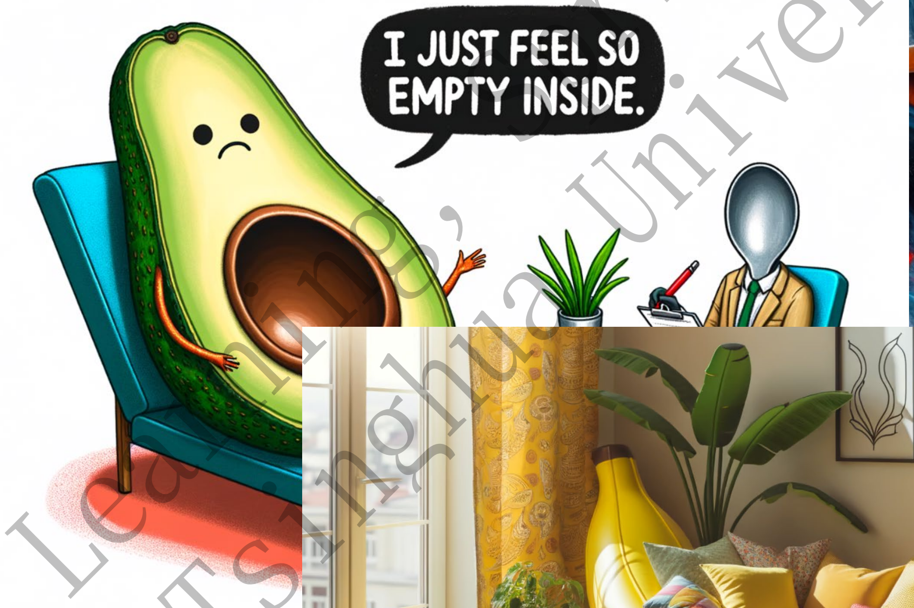
- Normalizing Flow
  - A generative model class that has the best sampling and likelihood properties
- Score-based generative model
  - A different framework to tackle general energy-based models

Today’s Topic

- Normalizing Flow
  - A generative model class that has the best sampling and likelihood properties

- Score-based generative model
  - A different framework to tackle general energy-based models
  - The model class that has the best generation quality
  - It is also called the ***diffusion model***

Why Diffusion Model?



Dall-E 3

Why Diffusion Model?

Stable Diffusion Model (SD3)

Why Diffu

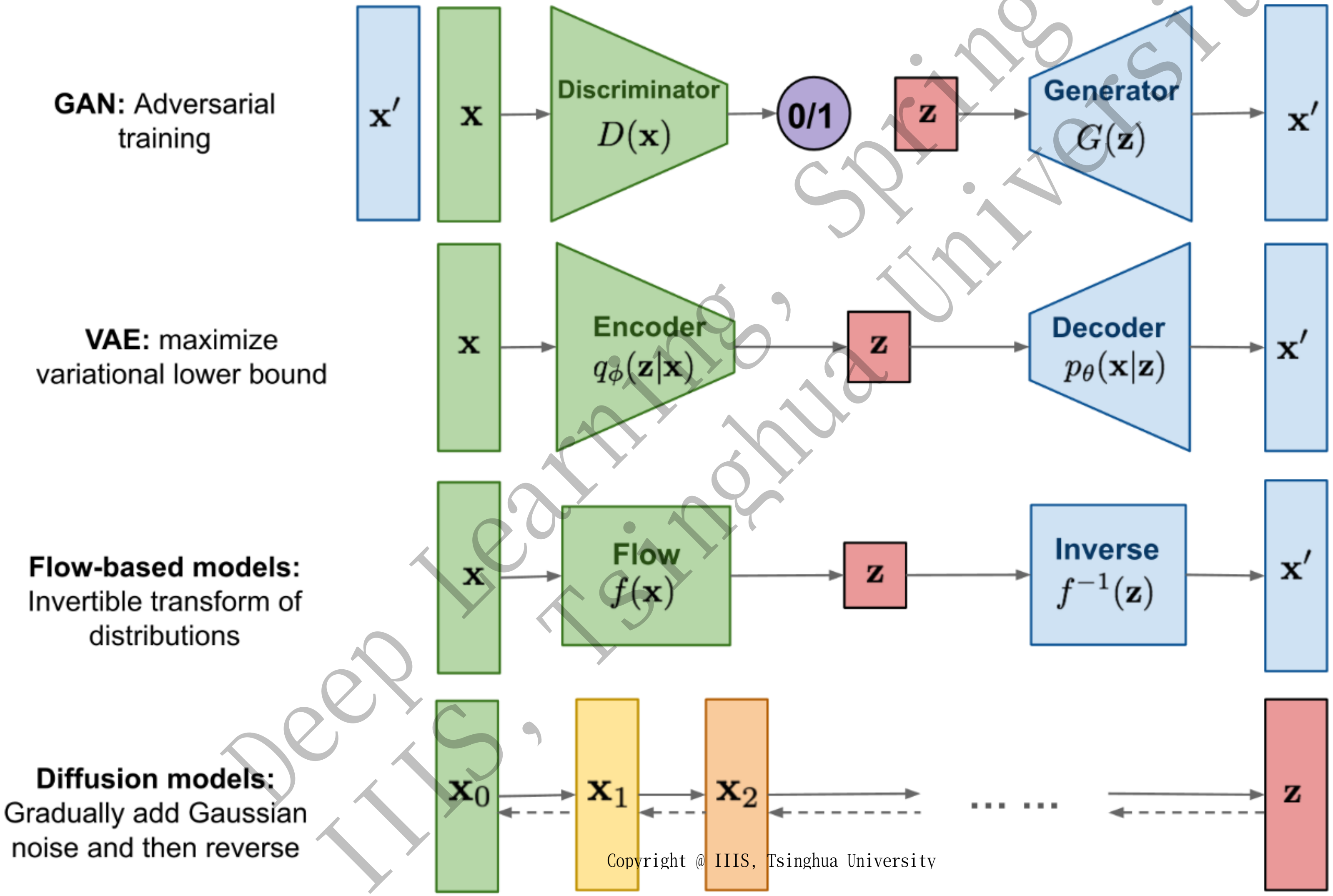
Midjourney

Why Diffusion Model?

- AI-generated photo wins the award
  - Jason Allen’s A.I.-generated work, “Théâtre D’opéra Spatial,” took first place in the digital category at the Colorado State Fair.

- The trend of AIGC
  - AI Generated Content
- Foundation: Diffusion Model

What is Diffusion Model

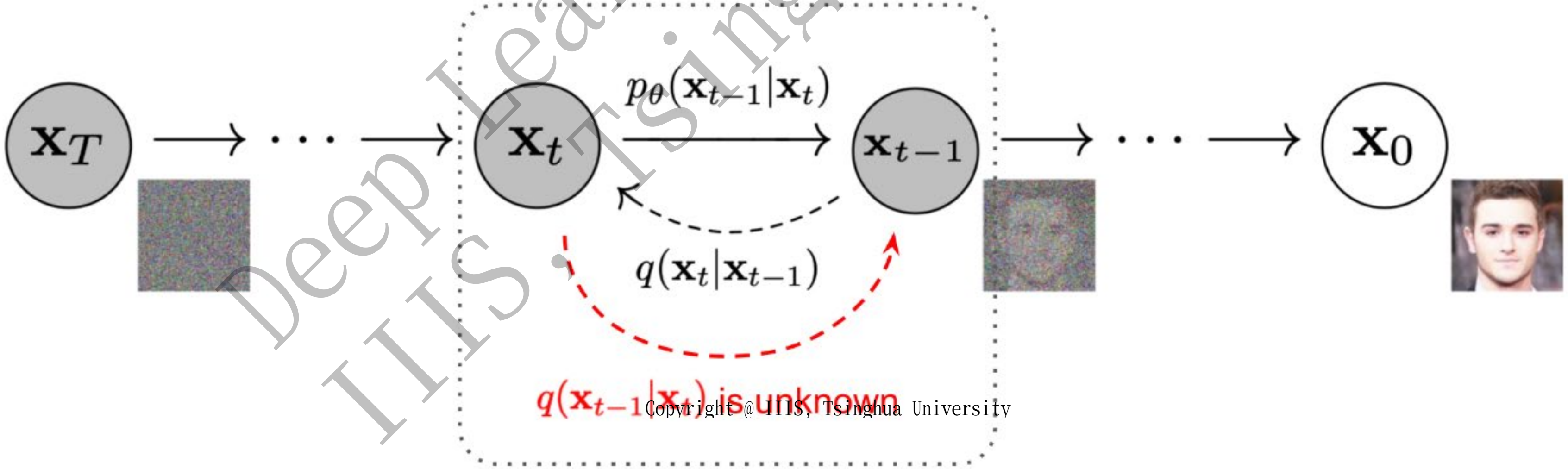


What is Diffusion Model

- Formal Definition
  - $x_0 \sim q_{data}(x)$
  - Forward diffusion process: continuously adding Gaussian noise to data

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

  - Sampling process: gradually recover the data from isomorphic Gaussian noise



Diffusion Model

- Milestone works
  - The story
    - https://www.quantamagazine.org/the-physics-principle-that-inspired-modern-ai-art-20230105/
  - Deep Unsupervised Learning using Nonequilibrium Thermodynamics (ICML 2015)
    - The original diffusion model
  - Generative Modeling by Estimating Gradients of the Data Distribution (Yang Song, et al., NIPS 2019)
    - Score-based model, foundation of modern diffusion model
  - Denoising Diffusion Probabilistic Models (Jonathan Ho, et al., NIPS 2020)
    - DDPM: the first working diffusion model

Diffusion Model

- Milestone works
  - The story
    - https://www.quantamagazine.org/the-physics-principle-that-inspired-modern-ai-art-20230105/
  - Deep Unsupervised Learning using Nonequilibrium Thermodynamics (ICML 2015)
    - The original diffusion model
  - Generative Modeling by Estimating Gradients of the Data Distribution (Yang Song, et al., NIPS 2019)
    - Score-based model, foundation of modern diffusion model
  - Denoising Diffusion Probabilistic Models (Jonathan Ho, et al., NIPS 2020)
    - DDPM: the first working diffusion model

Diffusion Model

- Denoising Diffusion Probabilistic Models (Jonathan Ho, et al., NIPS 2020)
  - Learning $\epsilon_\theta(x,t)$; $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$
  - Original diffusion model loss function from ICML15 (your homework ☺)
    - $L_t = E_{x_0,\epsilon}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)|\Sigma_\theta|_2^2}\left|\epsilon_t - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t\right)\right|^2\right]$
  - DDPM simplified objective: $T = 1000, \alpha_t = 1 - \beta_t, \beta_t \sim [10^{-4}, 0.02]$

Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1,\ldots,T\})$
  $\epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
  Take gradient descent step on
    $\nabla_\theta \left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2$
until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
for $t = T,\ldots,1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t,t)\right) + \sigma_t\mathbf{z}$
end for
return $\mathbf{x}_0$

Why does it work?

Diffusion Model

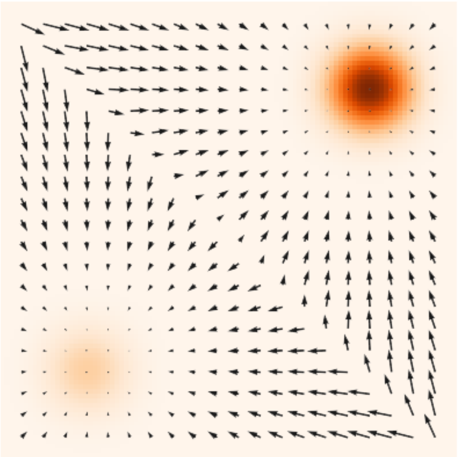
- Milestone works
  - The story
    - https://www.quantamagazine.org/the-physics-principle-that-inspired-modern-ai-art-20230105/
  - Deep Unsupervised Learning using Nonequilibrium Thermodynamics (ICML 2015)
    - The original diffusion model
  - Generative Modeling by Estimating Gradients of the Data Distribution (Yang Song, et al., NIPS 2019)
    - **Score-based model, foundation of modern diffusion model**
  - Denoising Diffusion Probabilistic Models (Jonathan Ho, et al., NIPS 2020)
    - DDPM: the first working diffusion model

Diffusion Model

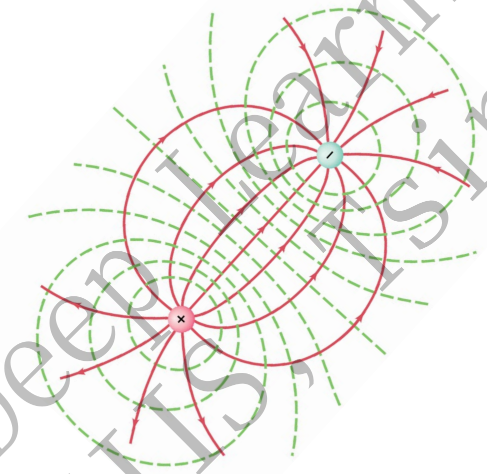
- Milestone works
  - The story
    - https://www.quantamagazine.org/the-physics-principle-that-inspired-modern-ai-art-20230105/
  - Deep Unsupervised Learning using Nonequilibrium Thermodynamics (ICML 2015)
    - The original diffusion model
  - Generative Modeling by Estimating Gradients of the Data Distribution (Yang Song, et al., NIPS 2019)
    - **Score-based model, foundation of modern diffusion model**
  - Denoising Diffusion Probabilistic Models (Jonathan Ho, et al., NIPS 2020)
    - DDPM: the first working diffusion model
    - **A simplified training objective directly connected to score-based model**

Score-Based Model

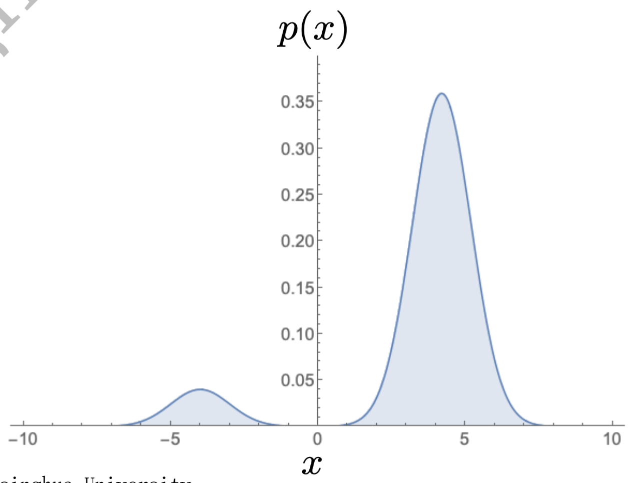
- How to represent a distribution $p(x)$?
  - When the pdf is differentiable, we can compute the gradient of a probability density.
  - Score function: $s(x) = \nabla_x \log p(x)$

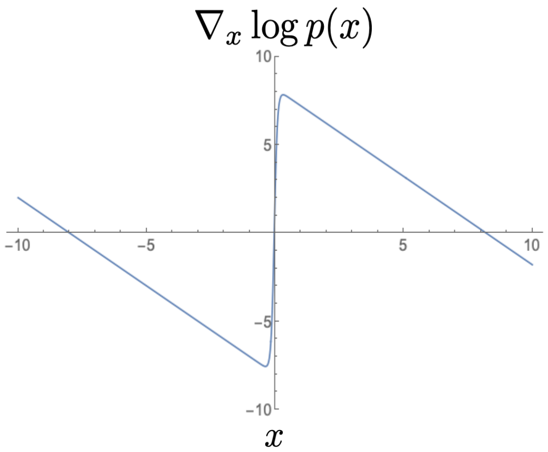


(pdf and score)



(Electrical potentials and fields)



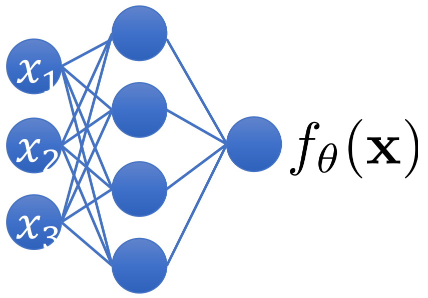


Score-Based Model

- Energy-based model (recap)
  - Energy function $f_\theta(x)$
  - Partition function $Z(\theta)$

$$f_\theta(\mathbf{x}) \in \mathbb{R}$$

$$p_\theta(\mathbf{x}) = \frac{e^{f_\theta(\mathbf{x})}}{Z(\theta)}$$



- Learning EBMs for p_{data}
  - MLE with Contrastive Divergence for $x_{\text{train}} \sim p_{data}$

$$\max_\theta f_\theta(\mathbf{x}_{\text{train}}) - \log Z(\theta)$$

$$\nabla_\theta f_\theta(\mathbf{x}_{\text{train}}) - \nabla_\theta \log Z(\theta) \approx \nabla_\theta f_\theta(\mathbf{x}_{\text{train}}) - \nabla_\theta f_\theta(\mathbf{x}_{\text{sample}})$$

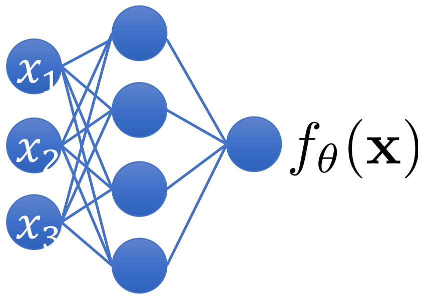
  - Monte-Carlo sampling for negative samples $\mathbf{x}_{\text{sample}} \sim p_\theta(\mathbf{x})$

Score-Based Model

- Energy-based model (recap)
  - Energy function $f_\theta(x)$
  - Partition function $Z(\theta)$

$$f_\theta(\mathbf{x}) \in \mathbb{R}$$

$$p_\theta(\mathbf{x}) = \frac{e^{f_\theta(\mathbf{x})}}{Z(\theta)}$$



- Learning EBMs for p_{data}
  - An alternative objective: Score matching
    - fisher divergence $F(p||q) = \frac{1}{2}E_{x\sim p}[|\nabla_x p(x) - \nabla_x q(x)|_2^2]$
  - Score matching by minimizing fisher divergence

$$\frac{1}{2}E_{x\sim p_{data}}[|\nabla_x \log p_{data}(x) - \nabla_x \log p_\theta(x)|_2^2\ ]$$

$$= E_{x\sim p_{data}}\left[\frac{1}{2}|\nabla_x \log p_\theta(x)|_2^2 + \mathrm{tr}(\nabla_x^2 \log p_\theta(x))\right] + Const$$

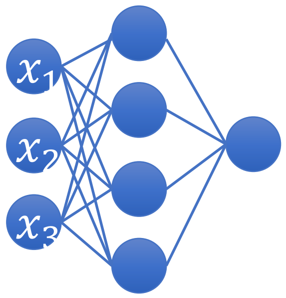
Your homework ☺

Score-Based Model

- Energy-based model (recap)
  - Energy function $f_\theta(x)$
  - Partition function $Z(\theta)$

$$f_\theta(\mathbf{x}) \in \mathbb{R}$$

$$p_\theta(\mathbf{x}) = \frac{e^{f_\theta(\mathbf{x})}}{Z(\theta)}$$



- Learning EBMs for p_{data}
  - An alternative objective: Score matching
    - fisher divergence $F(p||q) = \frac{1}{2} E_{x\sim p}[|\nabla_x p(x) - \nabla_x q(x)|_2^2]$
  - Score matching by minimizing fisher divergence

$$\frac{1}{2} E_{x\sim p_{data}}[|\nabla_x \log p_{data}(x) - \nabla_x \log p_\theta(x)|_2^2]$$

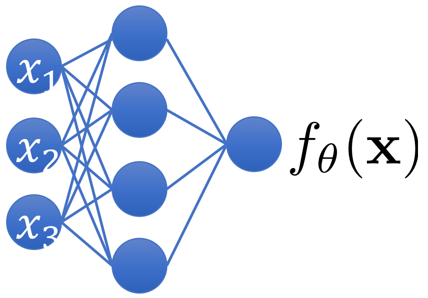
$$= E_{x\sim p_{data}}\left[\frac{1}{2}|\nabla_x \log p_\theta(x)|_2^2 + \mathrm{tr}(\nabla_x^2 \log p_\theta(x))\right] + Const$$

Score-Based Model

- Energy-based model (recap)
  - Energy function $f_\theta(x)$
  - Partition function $Z(\theta)$

$$f_\theta(\mathbf{x}) \in \mathbb{R}$$

$$p_\theta(\mathbf{x}) = \frac{e^{f_\theta(\mathbf{x})}}{Z(\theta)}$$



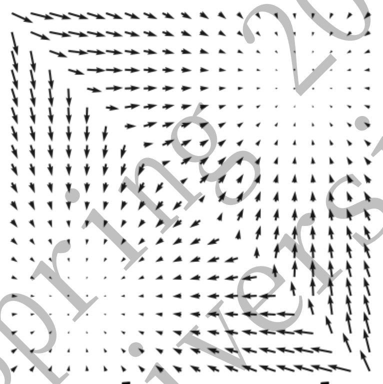
- Learning EBMs for p_{data}
  - An alternative objective: Score matching
    - fisher divergence $F(p||q) = \frac{1}{2} E_{x\sim p}[|\nabla_x p(x) - \nabla_x q(x)|_2^2]$
  - Score matching by minimizing fisher divergence

$$\frac{1}{2} E_{x\sim p_{data}}[|\nabla_x \log p_{data}(x) - \nabla_x \log p_\theta(x)|_2^2\ ]$$

$$= E_{x\sim p_{data}}\left[\frac{1}{2}|\nabla_x f_\theta(x)|_2^2 + \mathrm{tr}(\nabla_x^2 f_\theta(x))\right] + Const$$

No Partition function any more

Score-Based Model



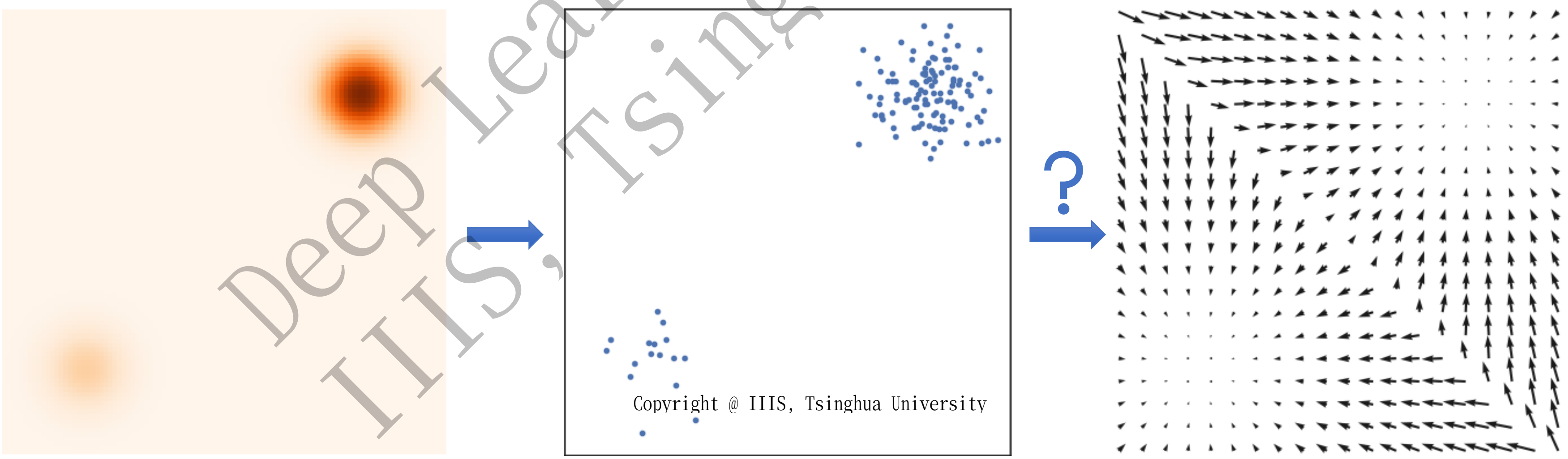
$\mathbb{R}^d \longrightarrow \mathbb{R}^d$

- Score-based model is beyond EBM
  - $s_\theta(x) \approx \nabla_x \log p_{data}(x)$
  - Learning?

Probability density $p_{\text{data}}(\mathbf{x})$

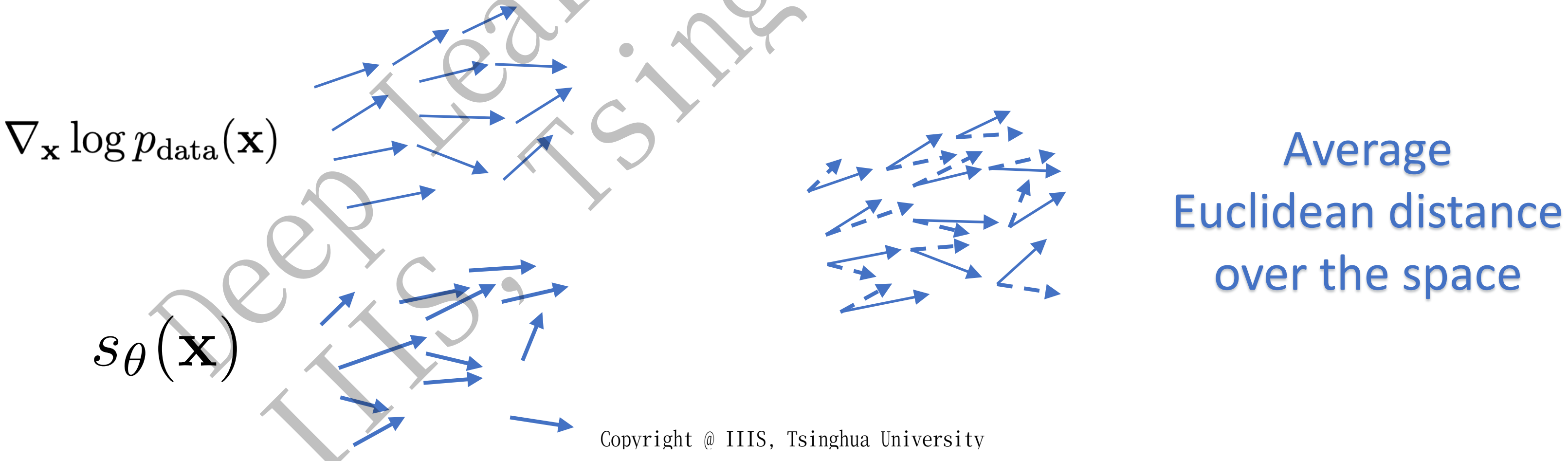
i.i.d. samples $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$

Score function $\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$



Score-Based Model

- Score estimation formulation
  - Given: i.i.d. samples $\{x_1, x_2, \dots, x_n\} \sim p_{data}(x)$
  - Task: Estimating the score $\nabla_x \log p_{data}(x)$
  - Score model: A learnable vector-valued function $s_\theta(x): \mathbb{R}^d \to \mathbb{R}^d$
  - Goal: $s_\theta(x) \approx \nabla_x p_{data}(x)$



Score-Based Model

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}[\|\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x})-\boldsymbol{s}_\theta(\mathbf{x})\|_2^2]$$

(Fisher divergence)

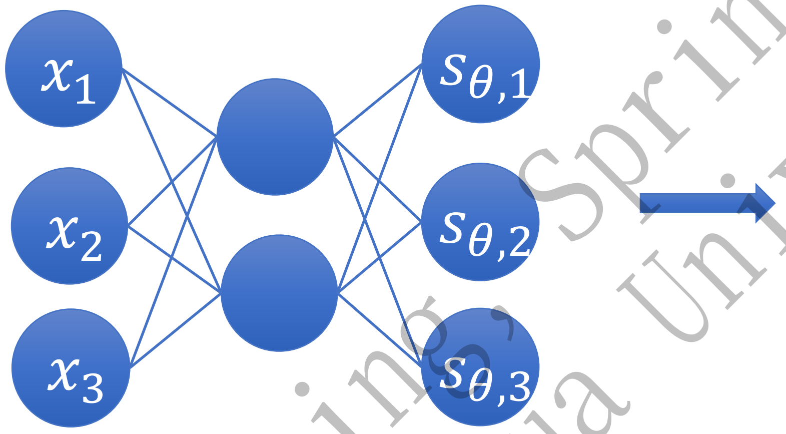
- **Score matching:**

$$E_{\mathbf{x}\sim p_{\text{data}}}\left[\frac{1}{2}\|\boldsymbol{s}_\theta(\mathbf{x})\|_2^2+\operatorname{tr}(\underbrace{\nabla_{\mathbf{x}}\boldsymbol{s}_\theta(\mathbf{x})}_{\text{Jacobian of } \boldsymbol{s}_\theta(\mathbf{x})})\right]$$

- **Requirements:**
  - The score model must be efficient to evaluate.
  - How to have a proper model for the score function?

Score-Based Model

- Deep neural networks as more expressive score models

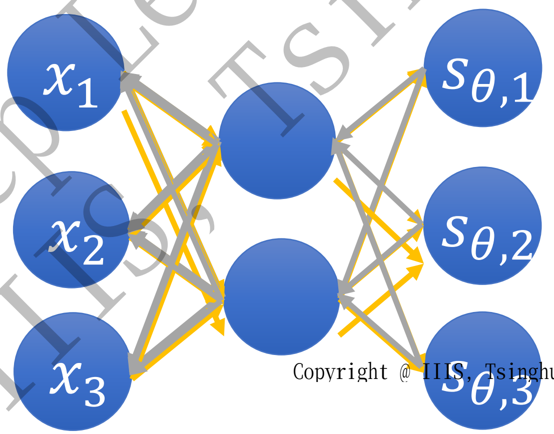


Score Matching is not Scalable due to the Jacobian!

- Compute $\|s_\theta(\mathbf{x})\|_2^2$ and $\mathrm{tr}(\nabla_{\mathbf{x}} \cancel{s_\theta}(\mathbf{x}))$

$O(d)$ Backprops!

$$\frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_1} \quad \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_2} \quad \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_3}$$



$$s_{\theta,1}(\mathbf{x}) \quad s_{\theta,2}(\mathbf{x}) \quad s_{\theta,3}(\mathbf{x})$$

$$\nabla_{\mathbf{x}} s_\theta(\mathbf{x}) = \begin{pmatrix} \boxed{\frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_1}} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_1} & \boxed{\frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_2}} & \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_2} & \boxed{\frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_3}} \end{pmatrix}$$

Denoising score matching

- Denoising score matching (Vincent 2011): matching the score of a noise-perturbed distribution

$\mathbf{x}$

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

$$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$$

$$q_\sigma(\tilde{\mathbf{x}})$$

$\tilde{\mathbf{x}}$

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) - \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2]$$

$$= \text{const.} + \frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2] - \int q_\sigma(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\, \mathrm{d}\tilde{\mathbf{x}}$$

$$= \text{const.} + \frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2] - E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}})]$$

Your homework ☺

$$= \text{const.} + \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]$$

$$- \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]$$

$$= \text{const.} + \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2] + \text{const.}$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2] + \text{const.}$$

Denoising score matching

- Estimate the score of a noise-perturbed distribution

$$
\begin{aligned}
&\frac{1}{2}E_{\tilde{\mathbf{x}}\sim p_{\text{data}}}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2] \\
=&\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x}),\tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2] + \text{const.}
\end{aligned}
$$

- $\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$ is easy to compute
  - $q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2\boldsymbol{I})$
  - $\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = -\frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2}$
- **Pros:** efficient to optimize even for very high dimensional data, and useful for optimal denoising.
- **Con:** cannot estimate the score of clean data (noise-free)

Denoising score matching

- Sample a minibatch of datapoints $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- Sample a minibatch of perturbed datapoints $\{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \cdots, \tilde{\mathbf{x}}_n\} \sim q_\sigma(\tilde{\mathbf{x}})$

$$\tilde{\mathbf{x}}_i \sim q_\sigma(\tilde{\mathbf{x}}_i \mid \mathbf{x}_i)$$

- Estimate the denoising score matching loss with empirical means

$$\frac{1}{2n}\sum_{i=1}^{n}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}_i) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}_i \mid \mathbf{x}_i)\|_2^2]$$

- If Gaussian perturbation

$$\frac{1}{2n}\sum_{i=1}^{n}\left[\left\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}_i) + \frac{\tilde{\mathbf{x}}_i - \mathbf{x}_i}{\sigma^2}\right\|_2^2\right]$$

- Stochastic gradient descent
- Need to choose a very small σ!

Pitfall of denoising score matching

- The loss variance will increase drastically as $\sigma \to 0$!
- Denoising score matching loss for Gaussian perturbations

$$\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}E_{\tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\left[\left\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})+\frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma^2}\right\|_2^2\right]$$

$$=\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}E_{\mathbf{z}\sim\mathcal{N}(0,\boldsymbol{I})}\left[\left\|\boldsymbol{s}_\theta(\mathbf{x}+\sigma\mathbf{z})+\frac{\mathbf{z}}{\sigma}\right\|_2^2\right] \qquad \text{(reparameterization trick)}$$

$$=\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}E_{\mathbf{z}\sim\mathcal{N}(0,\boldsymbol{I})}\left[\|\boldsymbol{s}_\theta(\mathbf{x}+\sigma\mathbf{z})\|_2^2+2\boldsymbol{s}_\theta(\mathbf{x}+\sigma\mathbf{z})^\mathsf{T}\frac{\mathbf{z}}{\sigma}+\frac{\|\mathbf{z}\|_2^2}{\sigma^2}\right]$$

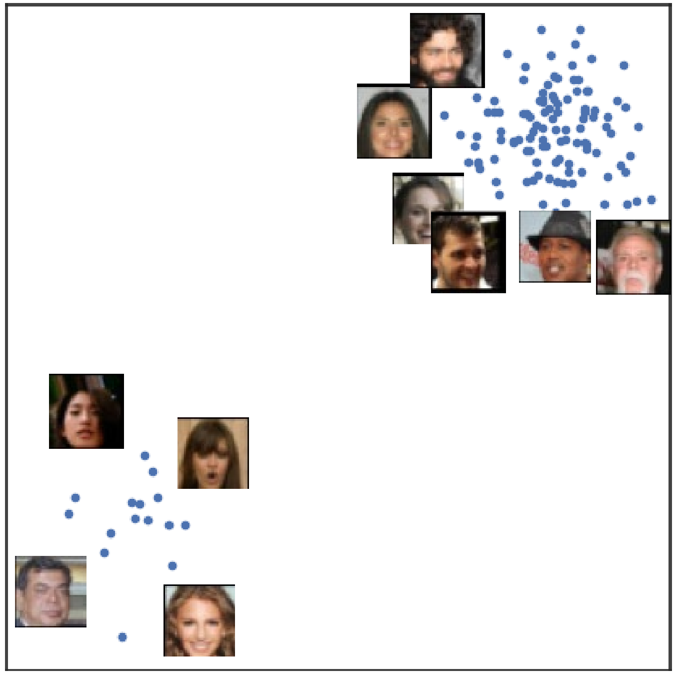
- If we choose very small $\sigma \to 0$

$$\mathrm{Var}\left(\frac{\mathbf{z}}{\sigma}\right)\to\infty$$

$$\mathrm{Var}\left(\frac{\|\mathbf{z}\|_2^2}{\sigma^2}\right)\to\infty$$

We need to tune σ carefully!

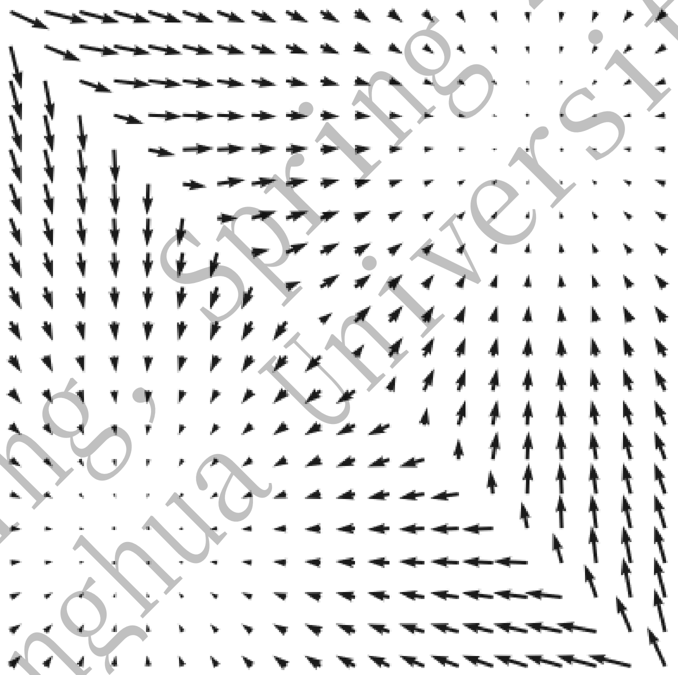
Score-based generative modeling



Data samples

$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$$

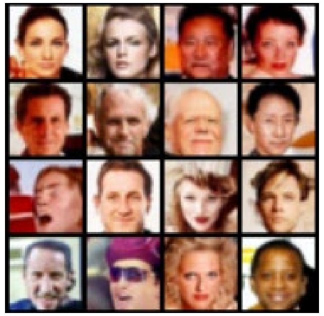
Score Matching



Scores

$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

?



New samples

Langevin dynamics sampling (Recap)

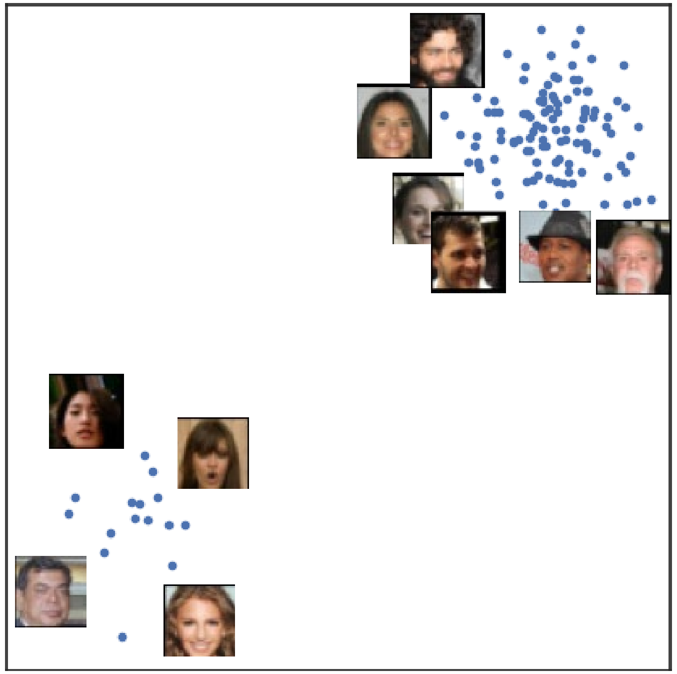
- Sample from $p(\mathbf{x})$ using only the score $\nabla_{\mathbf{x}} \log p(\mathbf{x})$
- Initialize $\mathbf{x}^0 \sim \pi(\mathbf{x})$
- Repeat for $t \leftarrow 1, 2, \cdots, T$

$$\mathbf{z}^t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

$$\mathbf{x}^t \leftarrow \mathbf{x}^{t-1} + \frac{\epsilon}{2} \nabla_{\mathbf{x}} \log p(\mathbf{x}^{t-1}) + \sqrt{\epsilon}\, \mathbf{z}^t$$

- If $\epsilon \rightarrow 0$ and $T \rightarrow \infty$, we are guaranteed to have $\mathbf{x}^T \sim p(\mathbf{x})$
- Langevin dynamics + score estimation $\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$

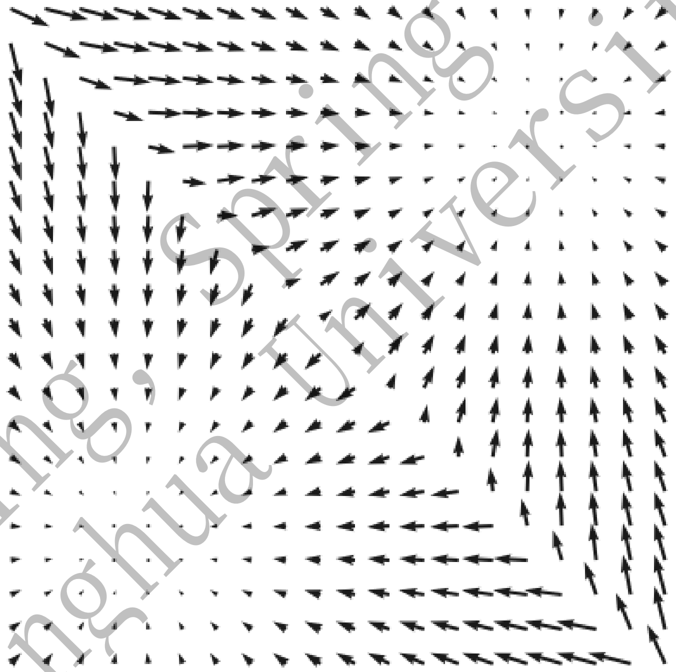
Score-based generative modeling



Data samples

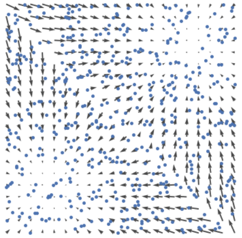
$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$$

score matching

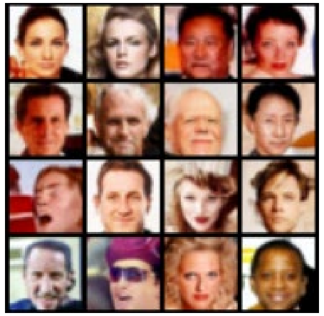


Scores

$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

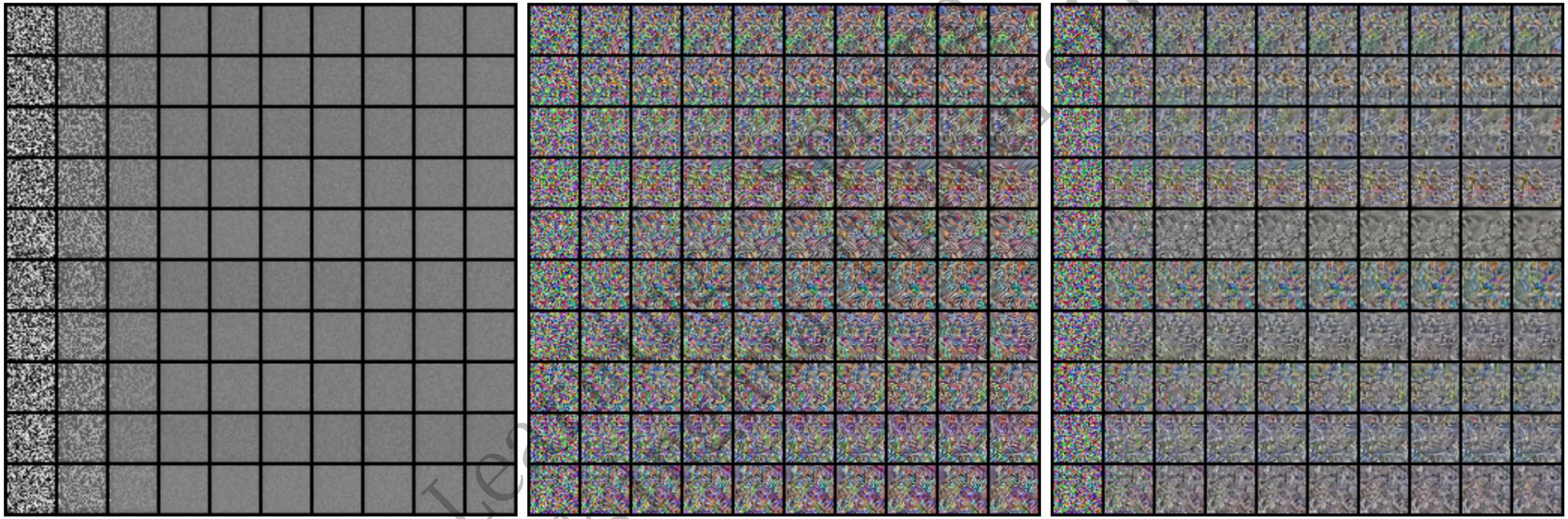


Langevin dynamics



New samples

Score-based generative modeling: empirical results

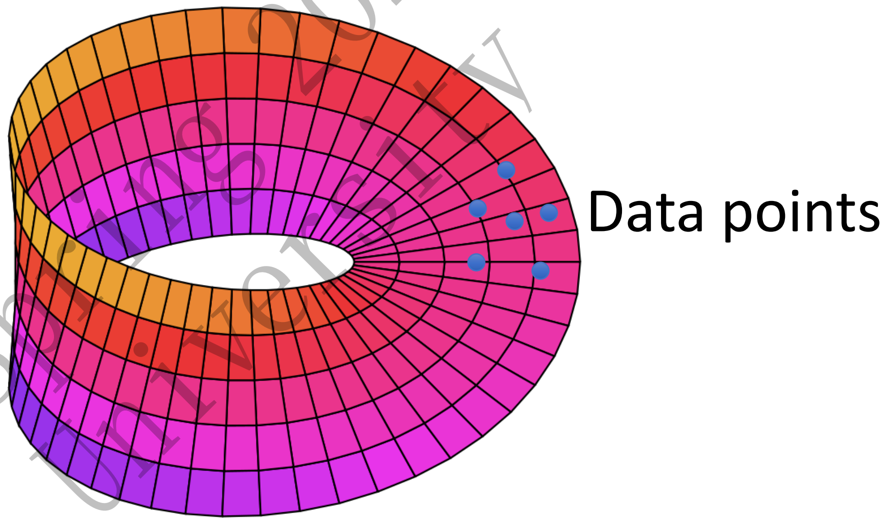


(a) MNIST (b) CelebA (c) CIFAR-10

Langevin sampling process

Why does it fail in practice???

Pitfall 1: manifold hypothesis



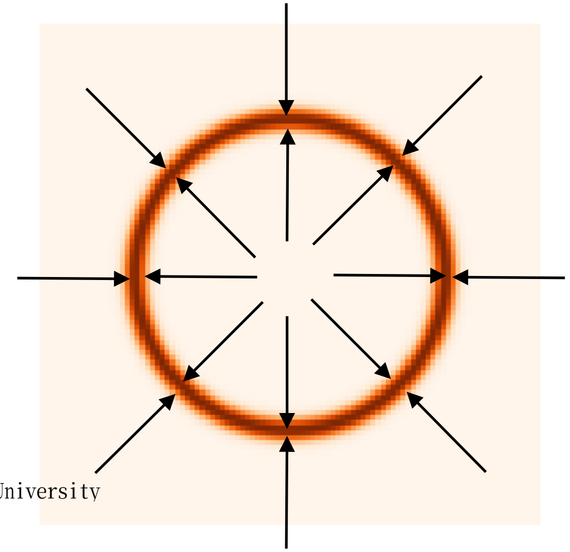
- Manifold hypothesis.

$$\nabla_{\mathbf{x}} \log \cancel{p}_{\text{data}}(\mathbf{x})$$

- Data score is undefined.

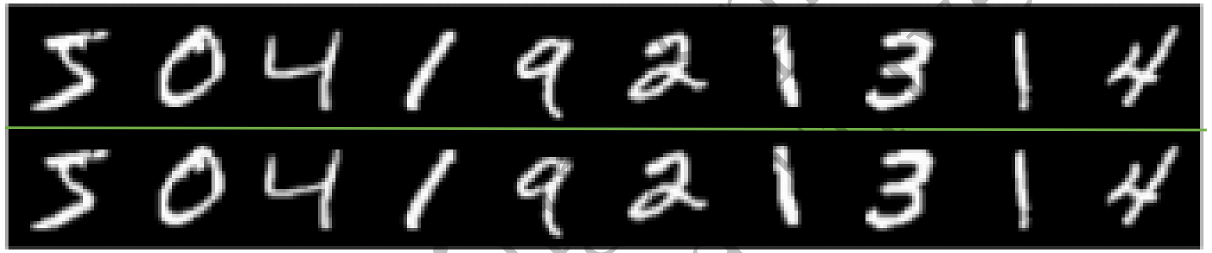
- Example
  - The data distribution is a ring
  - What is the score function like?

  - What about the real-world data?



Pitfall 1: manifold hypothesis

- A toy example for the manifold hypothesis
  - Fitting the data with a low-dimensional linear manifold (PCA)



784
Dim
595

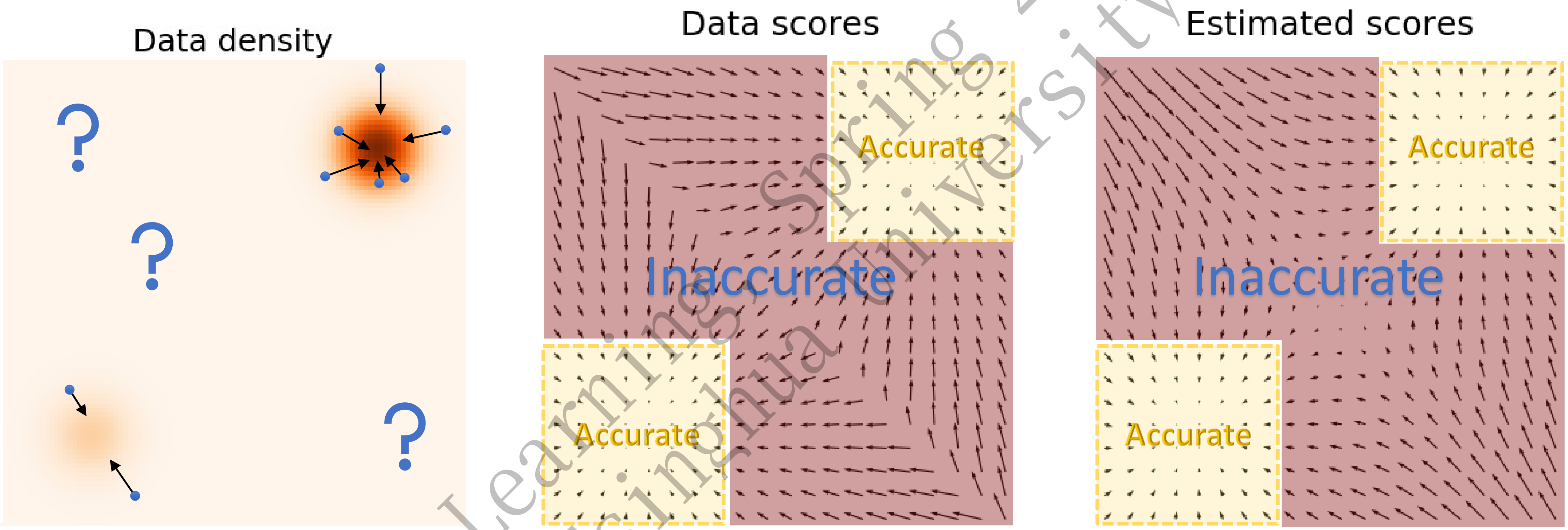
3072
Dim
2165

Real-world data have a small intrinsic dimension!

Pitfall 2: challenge in low data density regions

- Let’s assume a well defined score function over the entire space

Pitfall 2: challenge in low data density regions



$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})\|_2^2]$$

Langevin MCMC will have trouble exploring low density regions

Song and Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution." NeurIPS 2019.

Pitfall 3: slow mixing of Langevin dynamics between data modes

- Let’s further assume that we have learned accurate score functions!

- We may still have issue when $p(x)$ is a multi-modal distribution

Pitfall 3: slow mixing of Langevin dynamics between data modes

- Suppose the data distribution has two disajoint modes:

$$p_{\text{data}}(\mathbf{x}) = \pi p_1(\mathbf{x}) + (1-\pi)p_2(\mathbf{x})$$

$$\mathcal{A} \cap \mathcal{B} = \varnothing \qquad p_{\text{data}}(\mathbf{x}) = \begin{cases} \pi p_1(\mathbf{x}), & \mathbf{x} \in \mathcal{A} \\ (1-\pi)p_2(\mathbf{x}), & \mathbf{x} \in \mathcal{B} \end{cases}$$

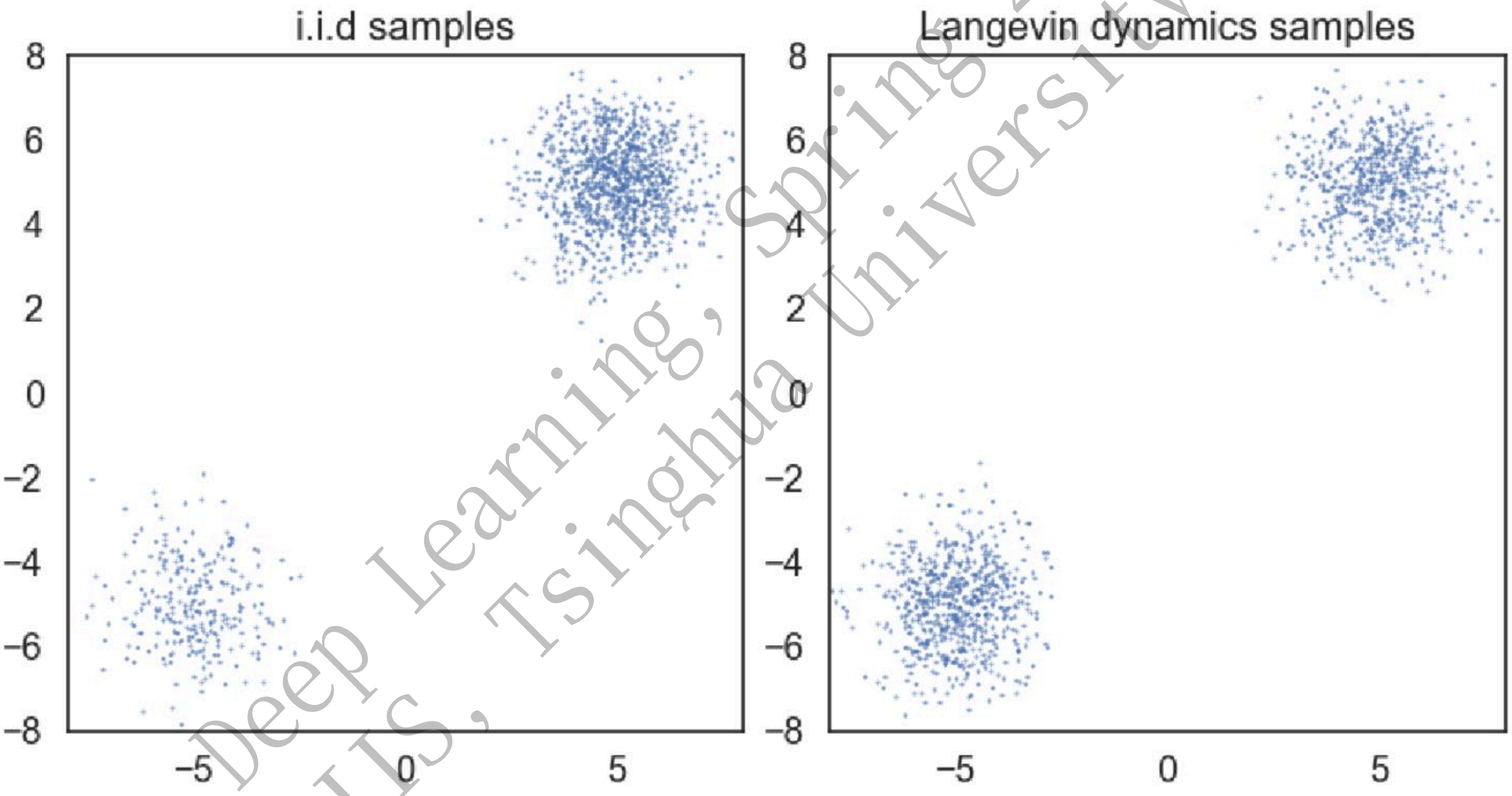
- Data score function:

$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) = \begin{cases} \nabla_{\mathbf{x}}[\log \pi + \log p_1(\mathbf{x})], & \mathbf{x} \in \mathcal{A} \\ \nabla_{\mathbf{x}}[\log(1-\pi) + \log p_2(\mathbf{x})], & \mathbf{x} \in \mathcal{B} \end{cases}$$

$$= \begin{cases} \nabla_{\mathbf{x}} \log p_1(\mathbf{x}), & \mathbf{x} \in \mathcal{A} \\ \nabla_{\mathbf{x}} \log p_2(\mathbf{x}), & \mathbf{x} \in \mathcal{B} \end{cases}$$

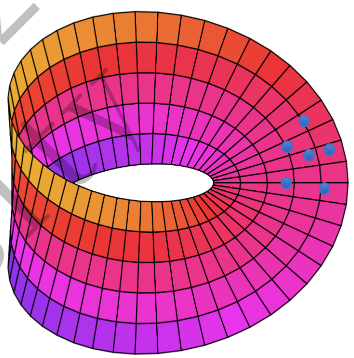
- The score function has no dependence on the mode weighting π at all!
- Langevin sampling will not reflect π

Pitfall 3: slow mixing of Langevin dynamics between data modes



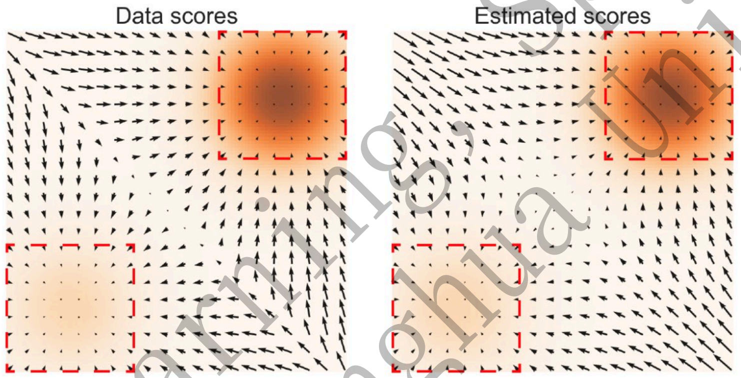
Pitfalls

- Manifold hypothesis. Data score is undefined.

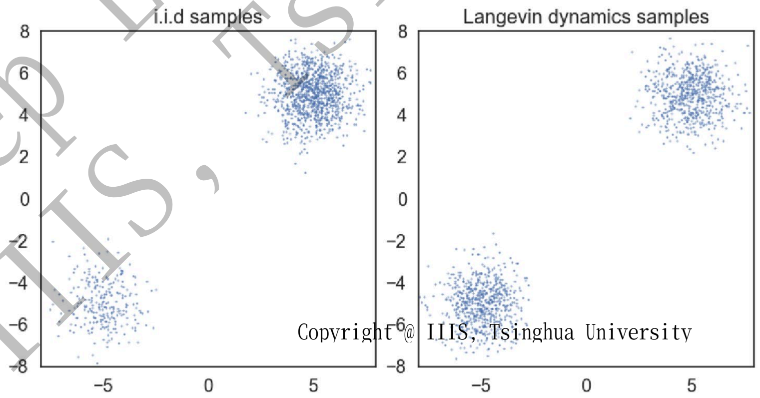


$$\nabla_{\mathbf{x}} \log \cancel{p}_{\text{data}}(\mathbf{x})$$

- Score matching fails in low data density regions



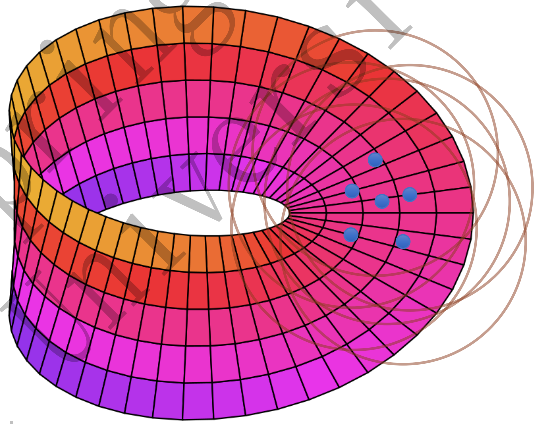
- Langevin dynamics fail to weight different modes correctly



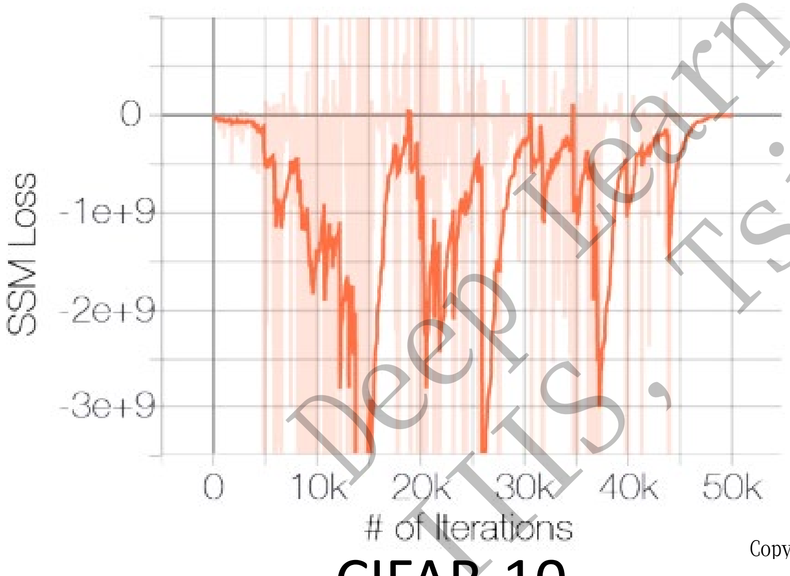
Gaussian perturbation

- The solution to all pitfalls: **Gaussian perturbation!**

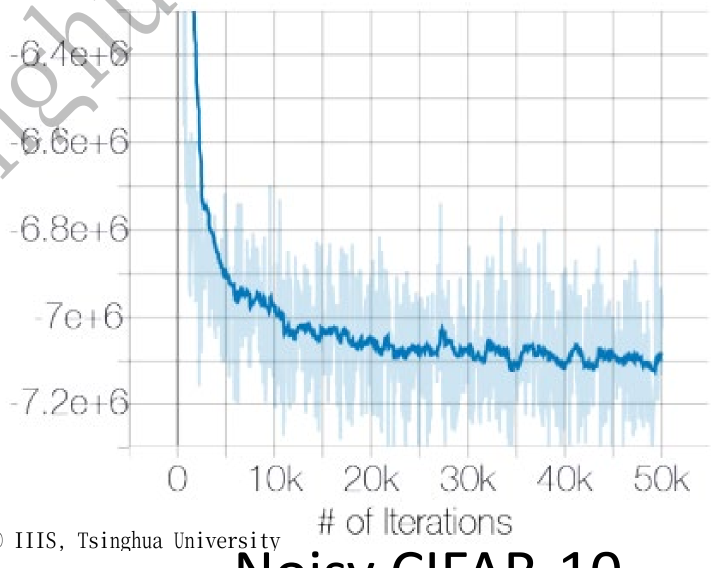
- Manifold + noise



- Score matching on noisy data.



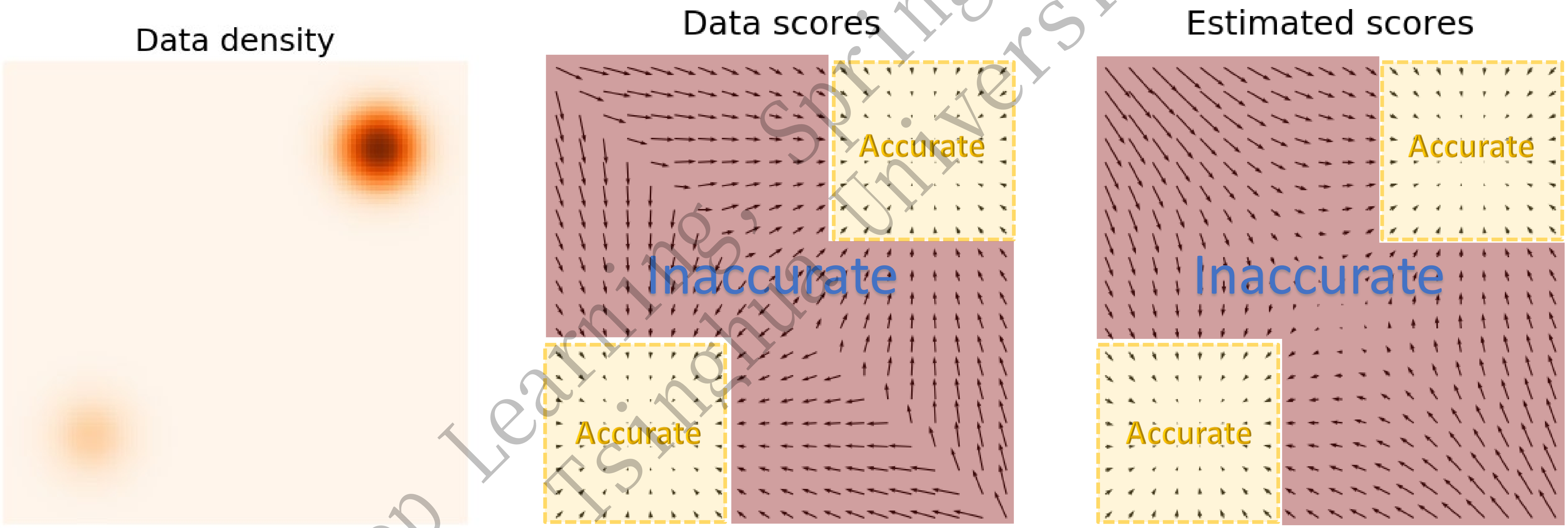
CIFAR-10



Noisy CIFAR-10

$$\mathcal{N}(0; 0.0001)$$

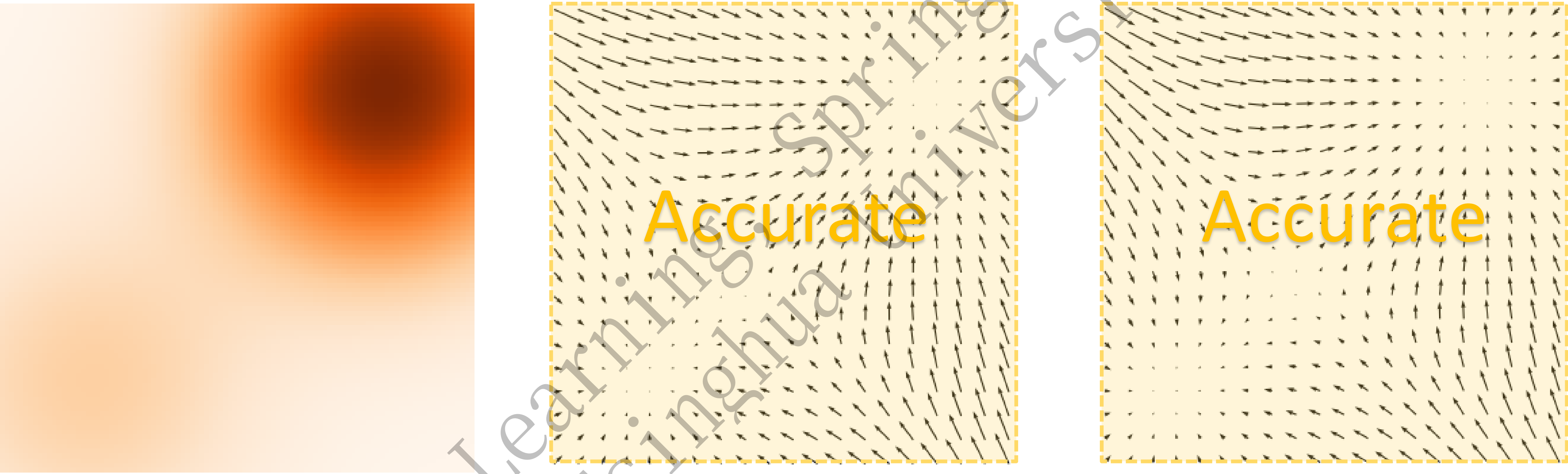
Challenge in low data density regions



Song and Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution." NeurIPS 2019.

Adding noise to data

Perturbed density Perturbed scores Estimated scores



Provide useful directional information for Langevin MCMC.

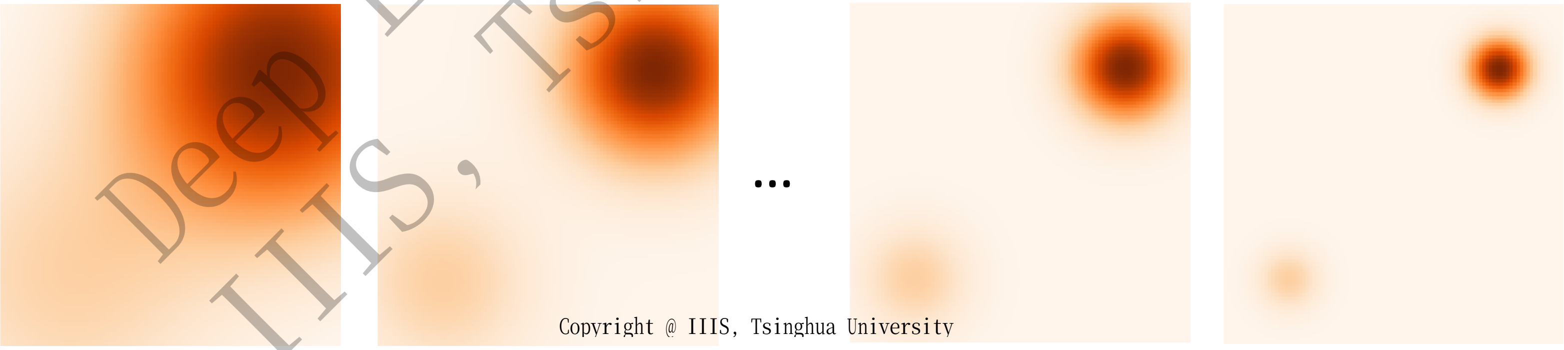
Song and Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution." NeurIPS 2019.

Multi-scale Noise Perturbation

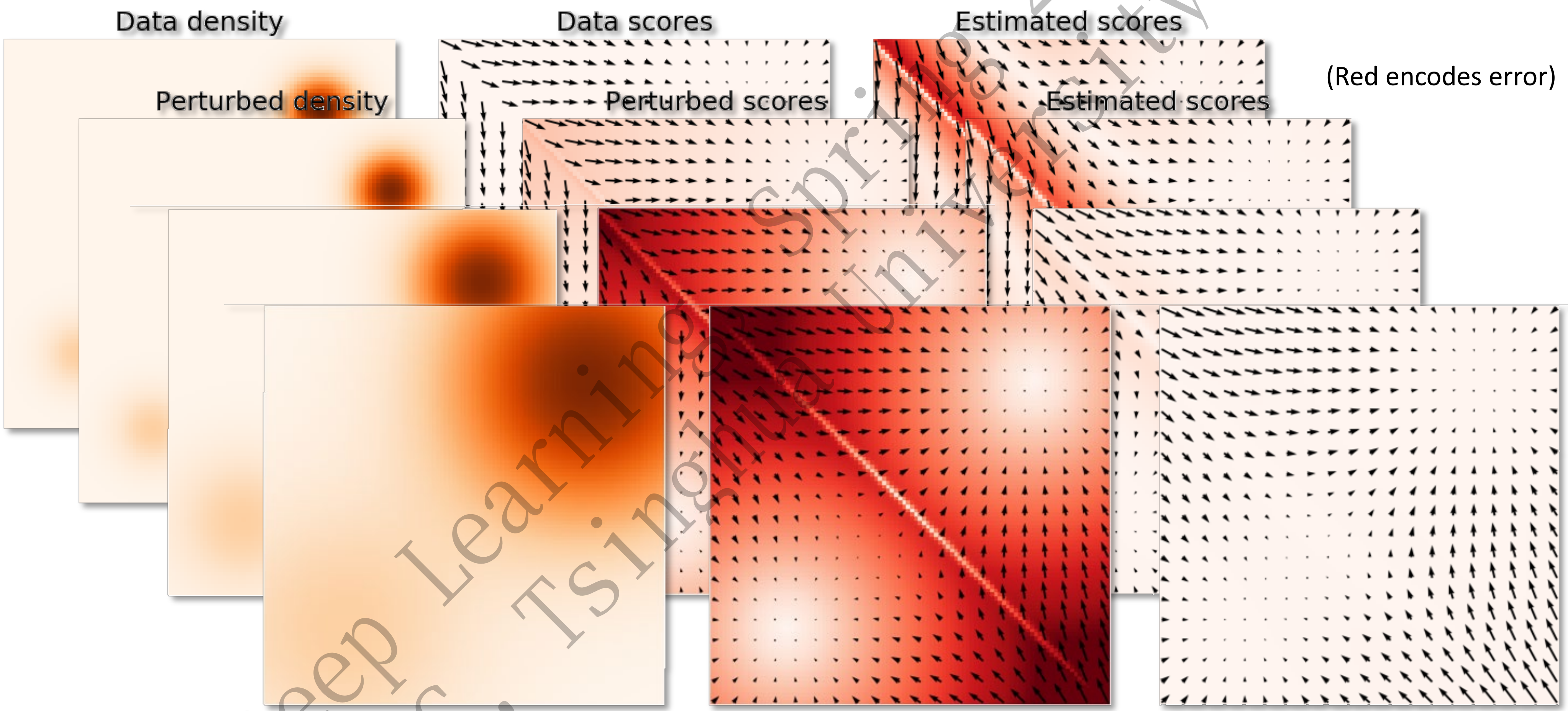
- Trade-off

- Multi-scale noise perturbations.

$$\sigma_1 > \sigma_2 > \cdots > \sigma_{L-1} > \sigma_L$$



Trading off Data Quality and Estimation Accuracy

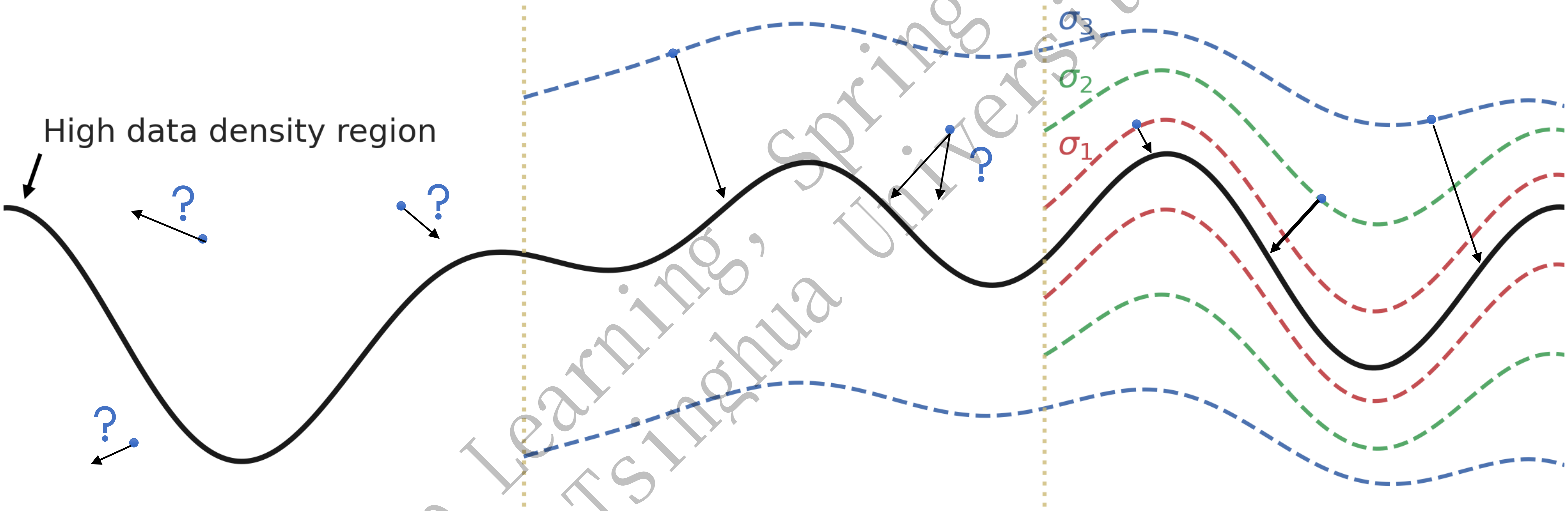


(Red encodes error)

Worse data quality!

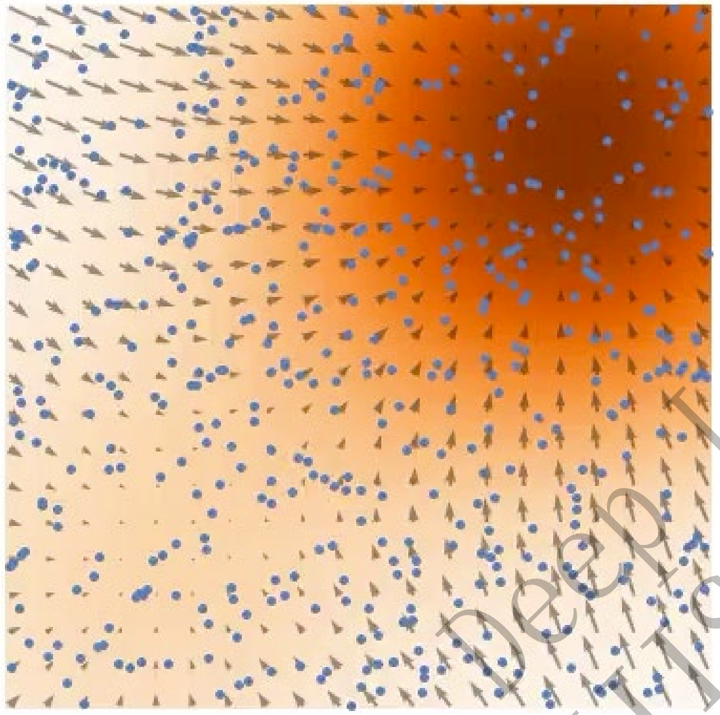
Better score estimation!

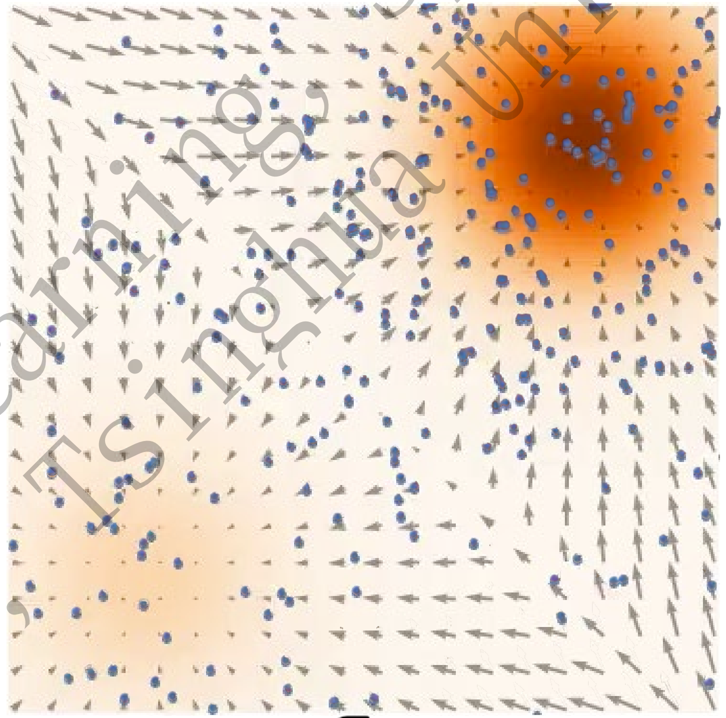
Using multiple noise scales

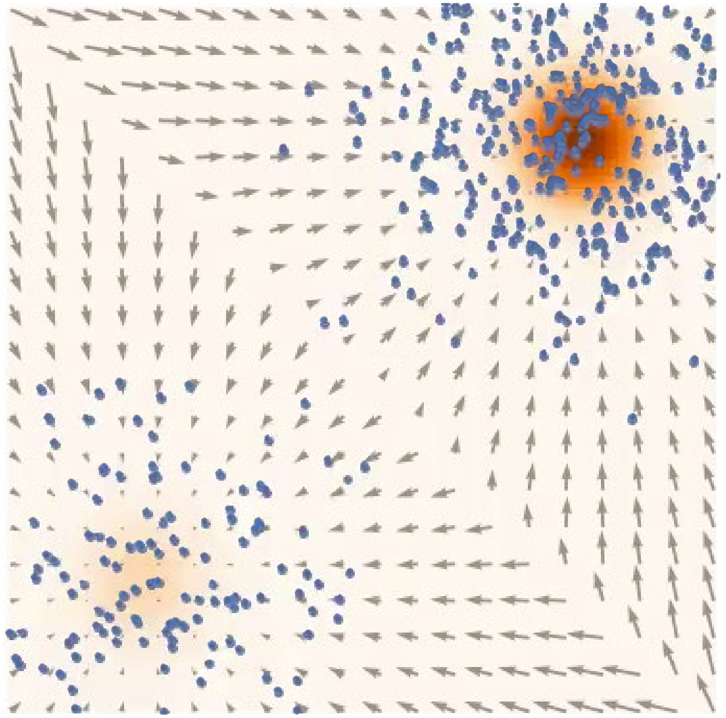


Annealed Langevin Dynamics: Joint Scores to Samples

- Sample using $\sigma_1, \sigma_2, \cdots, \sigma_L$ sequentially with Langevin dynamics.
- Anneal down the noise level.
- Samples used as initialization for the next level.


σ_1


σ_2


σ_3

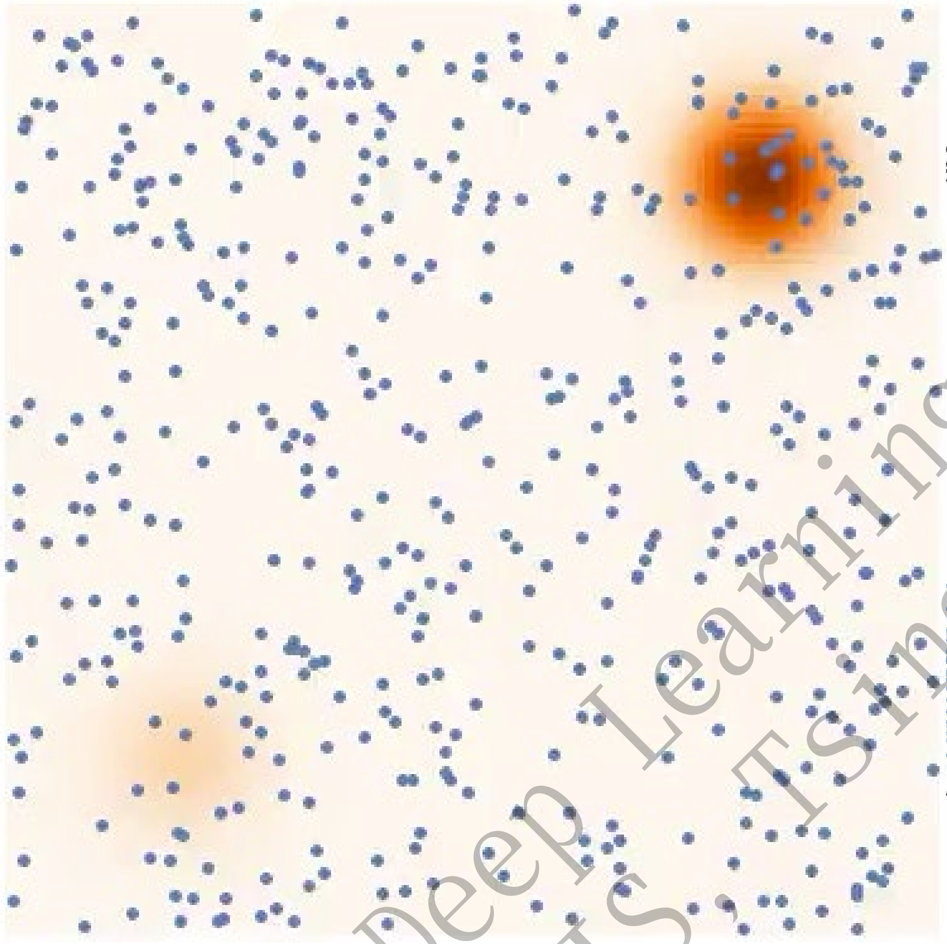
Annealed Langevin dynamics

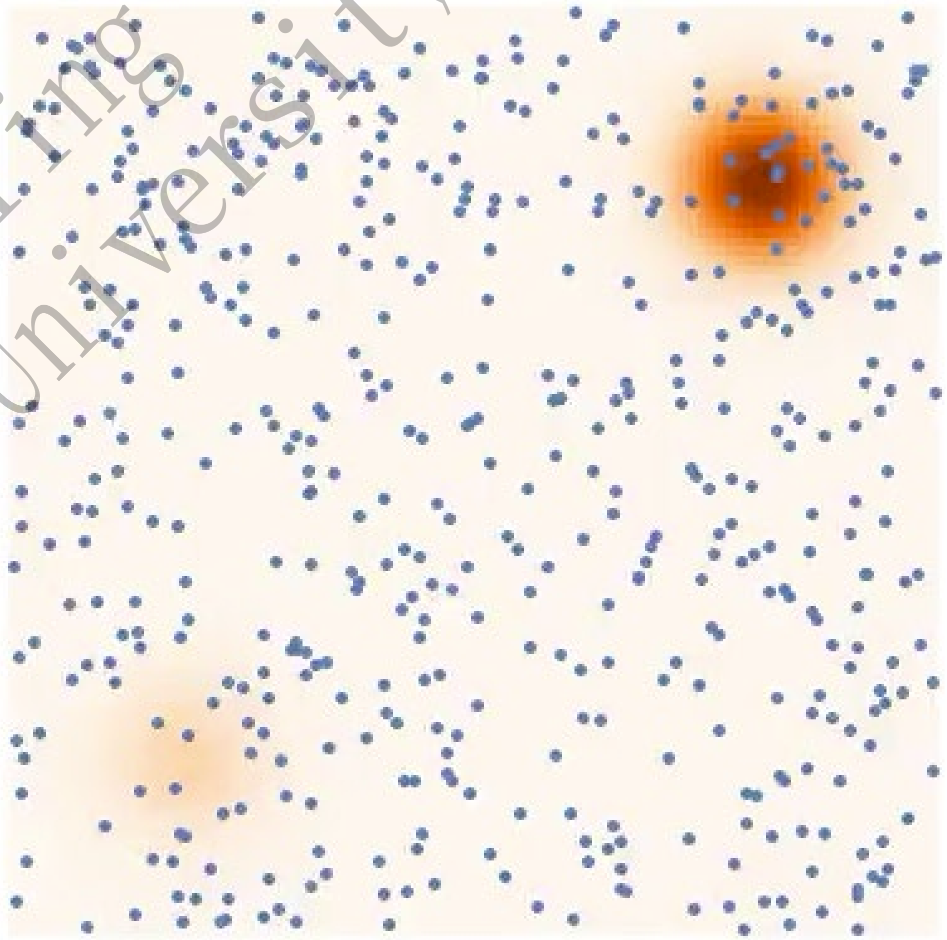
Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T.$

Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to L **do**
    $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ $\triangleright$ α_i is the step size.
    for $t \leftarrow 1$ to T **do**
        Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
        $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \dfrac{\alpha_i}{2} \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\, \mathbf{z}_t$
    end for
    $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$

Comparison to the vanilla Langevin dynamics

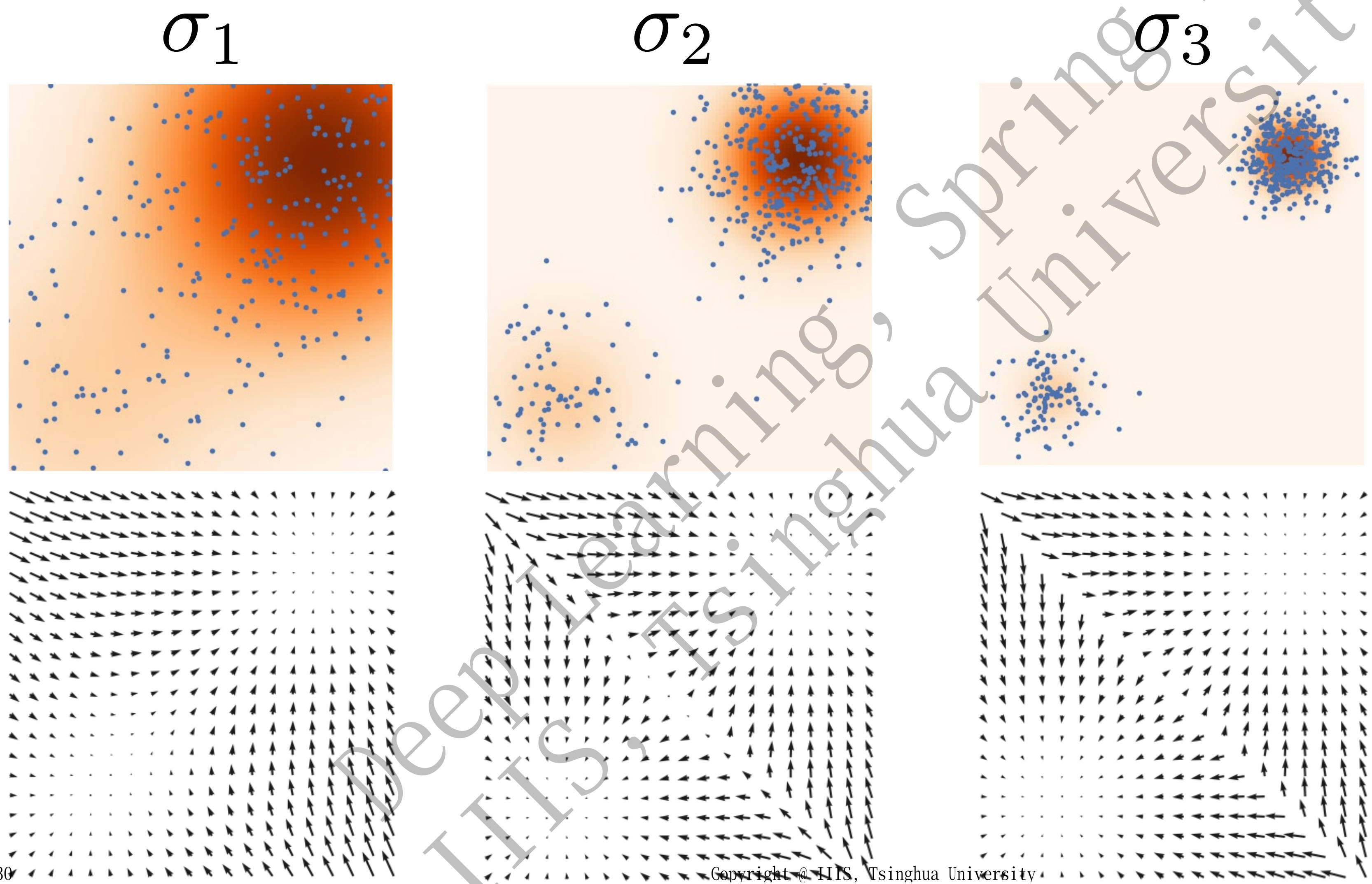


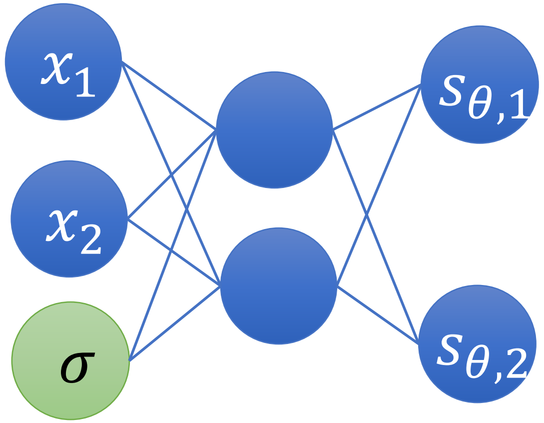


Langevin dynamics

Annealed Langevin dynamics

Joint Score Estimation via Noise Conditional Score Networks





Noise Conditional Score Network (NCSN)

Training noise conditional score networks

- Weighted combination of denoising score matching losses
  - Given the noise levels $\sigma_1 \dots \sigma_L$

$$\frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)E_{q_{\sigma_i}(\mathbf{x})}[\|\nabla_{\mathbf{x}}\log q_{\sigma_i}(\mathbf{x})-\boldsymbol{s}_\theta(\mathbf{x},\sigma_i)\|_2^2]$$

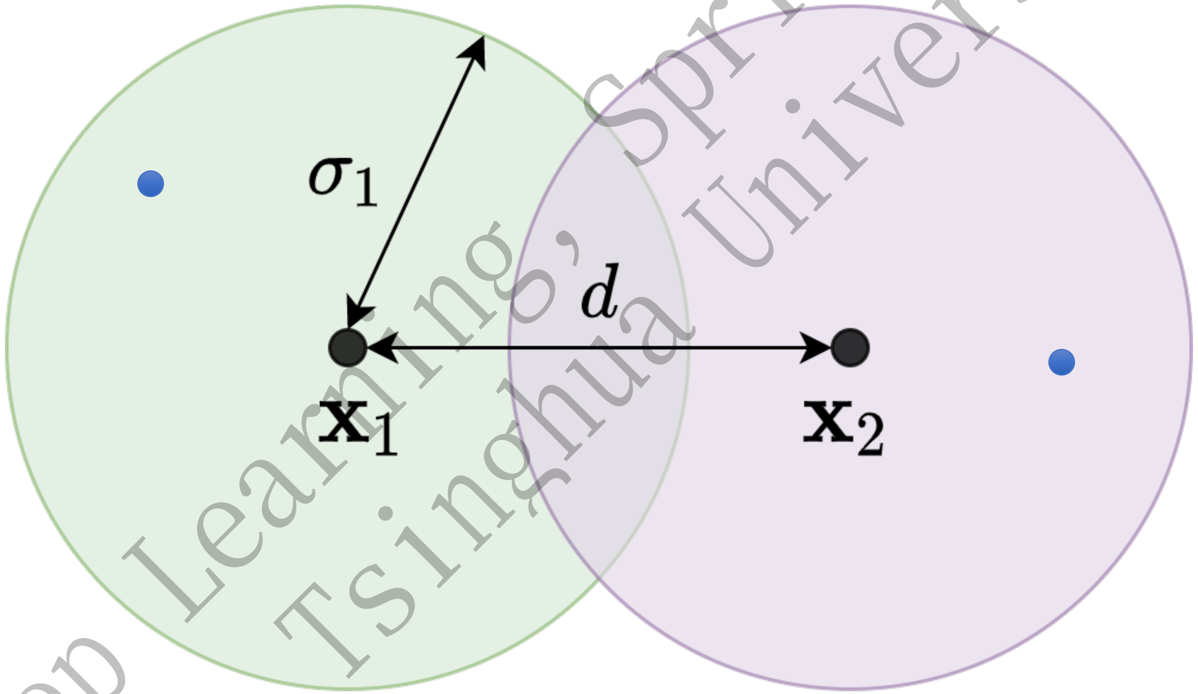
$$=\frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}[\|\nabla_{\tilde{\mathbf{x}}}\log q_{\sigma_i}(\tilde{\mathbf{x}}\mid\mathbf{x})-\boldsymbol{s}_\theta(\tilde{\mathbf{x}},\sigma_i)\|_2^2]+\text{const.}$$

$$=\frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left\|\boldsymbol{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\frac{\mathbf{z}}{\sigma_i}\right\|_2^2\right]+\text{const.}$$

Choosing noise scales

- Maximum noise scale

$$\sigma_1 \approx \text{maximum pairwise distance between datapoints}$$



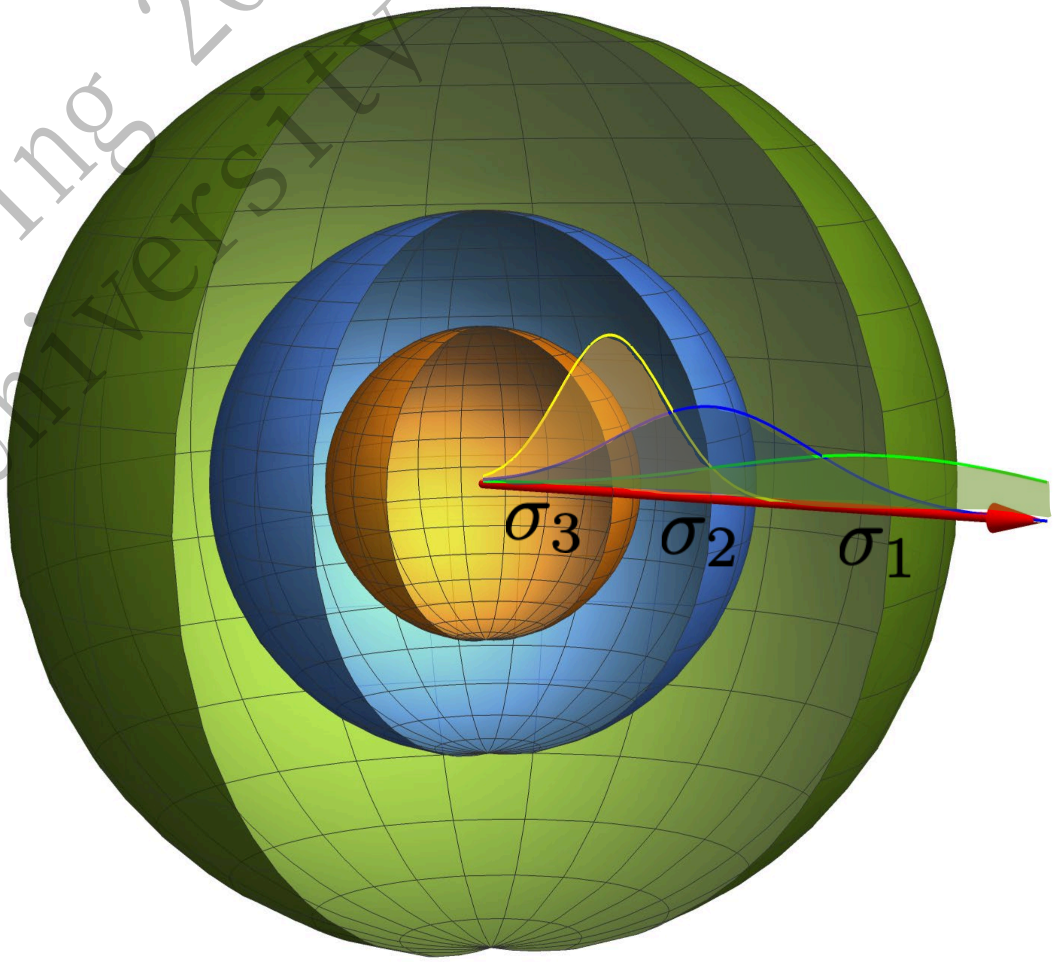
- Minimum noise scale: σ_L should be sufficiently small to control the noise in final samples.

Choosing noise scales

- **Key intuition:** adjacent noise scales should have sufficient overlap to facilitate transitioning across noise scales in annealed Langevin dynamics.
- A geometric progression with sufficient length.

$$\sigma_1 > \sigma_2 > \sigma_3 > \cdots > \sigma_{L-1} > \sigma_L$$

$$\frac{\sigma_1}{\sigma_2} = \frac{\sigma_2}{\sigma_3} = \cdots = \frac{\sigma_{L-1}}{\sigma_L}$$



Choosing the weighting function

- Weighted combination of denoising score matching losses

$$\frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left\|\boldsymbol{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\frac{\mathbf{z}}{\sigma_i}\right\|_2^2\right]$$

- How to choose the weighting function $\lambda:\mathbb{R}_{>0}\to\mathbb{R}_{>0}$?
- **Goal:** balancing different score matching losses in the sum → $\lambda(\sigma_i)=\sigma_i^2$

$$\begin{aligned}&\frac{1}{L}\sum_{i=1}^{L}\sigma_i^2E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left\|\boldsymbol{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\frac{\mathbf{z}}{\sigma_i}\right\|_2^2\right]\\ =&\frac{1}{L}\sum_{i=1}^{L}E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\|\sigma_i\boldsymbol{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\mathbf{z}\|_2^2\right]\\ =&\frac{1}{L}\sum_{i=1}^{L}E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\|\boldsymbol{\epsilon}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\mathbf{z}\|_2^2\right]\end{aligned}$$

$[\ \boldsymbol{\epsilon}_\theta(\cdot,\sigma_i):=\sigma_i\boldsymbol{s}_\theta(\cdot,\sigma_i)\ ]$

Training noise conditional score networks

- Sample a mini-batch of datapoints $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}$
- Sample a mini-batch of noise scale indices

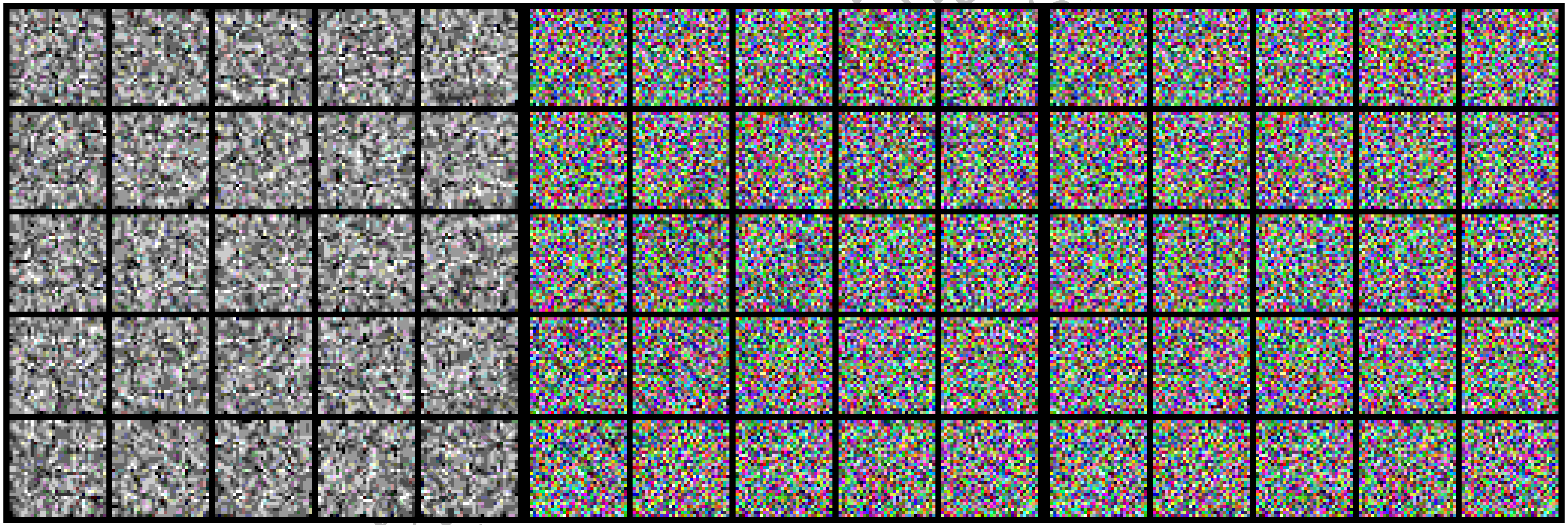
$$\{i_1, i_2, \cdots, i_n\} \sim \mathcal{U}\{1, 2, \cdots, L\}$$

- Sample a mini-batch of Gaussian noise $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Estimate the weighted mixture of score matching losses

$$\frac{1}{n}\sum_{k=1}^{n}\left[\left\|\sigma_{i_k}\boldsymbol{s}_\theta(\mathbf{x}_k + \sigma_{i_k}\mathbf{z}_k, \sigma_{i_k}) + \mathbf{z}_k\right\|_2^2\right]$$

- Stochastic gradient descent
- As efficient as training one single non-conditional score-based model

Experiments: Sampling



Experiments: Sampling

Model	Inception	FID
CIFAR-10 Unconditional		
PixelCNN [59]	4.60	65.93
PixelIQN [42]	5.29	49.46
EBM [12]	6.02	40.58
WGAN-GP [18]	$7.86 \pm .07$	36.4
MoLM [45]	$7.90 \pm .10$	**18.9**
SNGAN [36]	$8.22 \pm .05$	21.7
ProgressiveGAN [25]	$8.80 \pm .05$	-
NCSN (Ours)	$\mathbf{8.87} \pm .12$	25.32

High Resolution Image Generation

Comparison between NCSN and DDPM

- NCSN
  - Learning: $\frac{1}{L}\sum_{i=1}^{L} E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\,\|\boldsymbol{\epsilon}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\mathbf{z}\|_2^2\right]$ $\quad$ $[\ \boldsymbol{\epsilon}_\theta(\cdot,\sigma_i) := \sigma_i \boldsymbol{s}_\theta(\cdot,\sigma_i)\ ]$
  - Inference: $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1},\sigma_i) + \sqrt{\alpha_i}\,\mathbf{z}_t$
- DDPM
  - NCSN with a few enhancements for training stabilities
    - More discussions can be found in the original paper (your homework ☺)

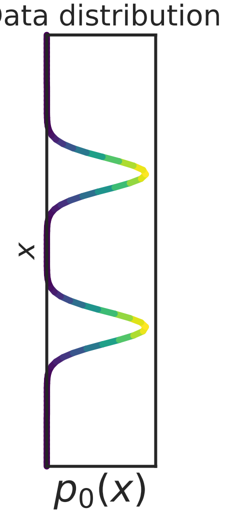
Algorithm 1 Training

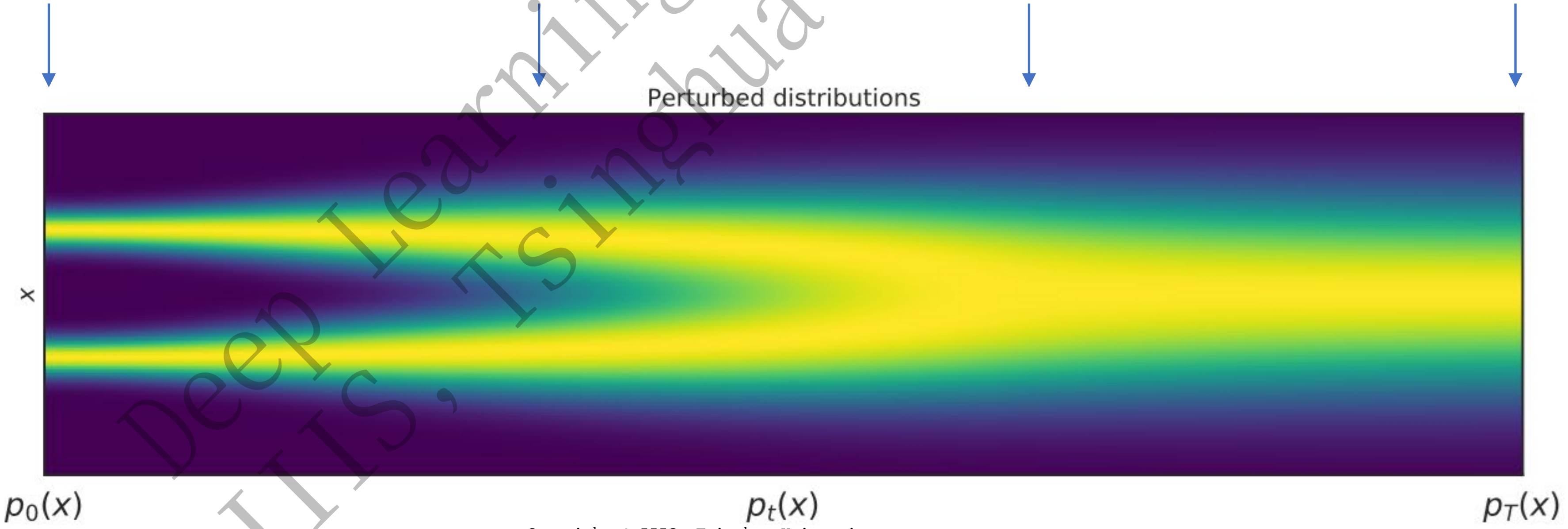
repeat
$\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
$\quad t \sim \text{Uniform}(\{1,\dots,T\})$
$\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
$\quad$ Take gradient descent step on
$$\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$$
until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
for $t = T,\dots,1$ **do**
$\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
$\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right) + \sigma_t\mathbf{z}$
end for
return $\mathbf{x}_0$

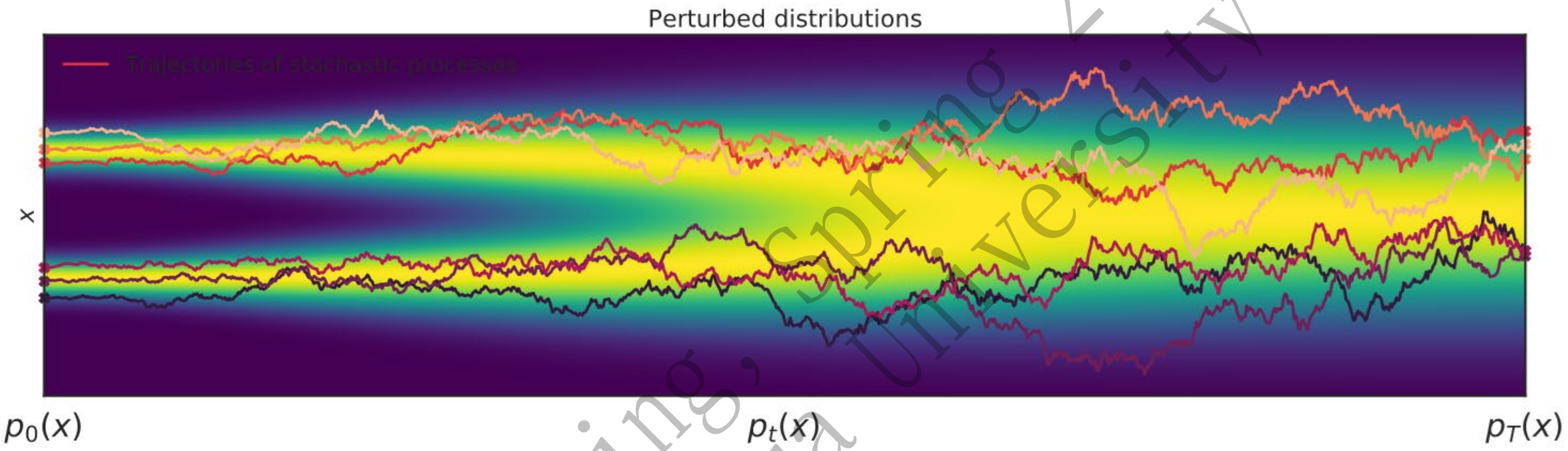
Using an infinite number of noise scales





$t \in [0, T]$: continuous index of perturbed distributions

Compact representation of infinite distributions



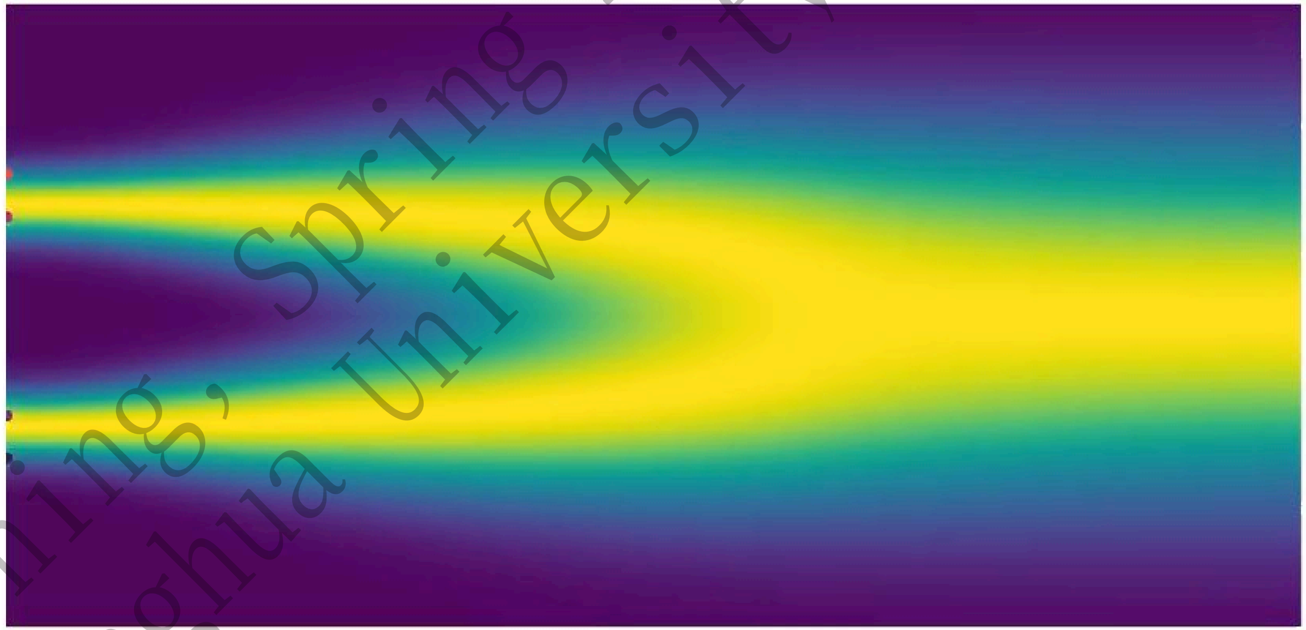
- Stochastic process $\{\mathbf{x}(t)\}_{t=0}^{T}$ → Marginal probability densities $\{p_t(\mathbf{x})\}_{t=0}^{T}$
- Stochastic differential equation: $\mathrm{d}\mathbf{x} = \boxed{\mathbf{f}(\mathbf{x}, t)\mathrm{d}t} + \sigma(t)\boxed{\mathrm{d}\mathbf{w}}$

Deterministic drift

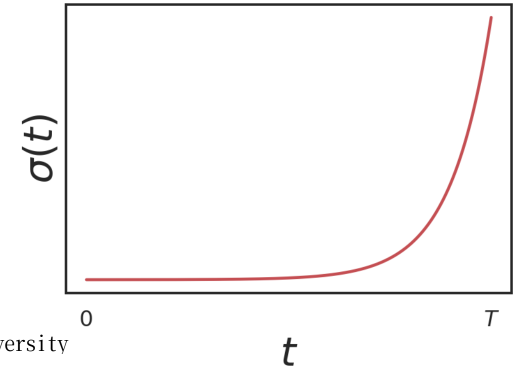
Infinitesimal white noise

Score-based generative modeling via SDEs

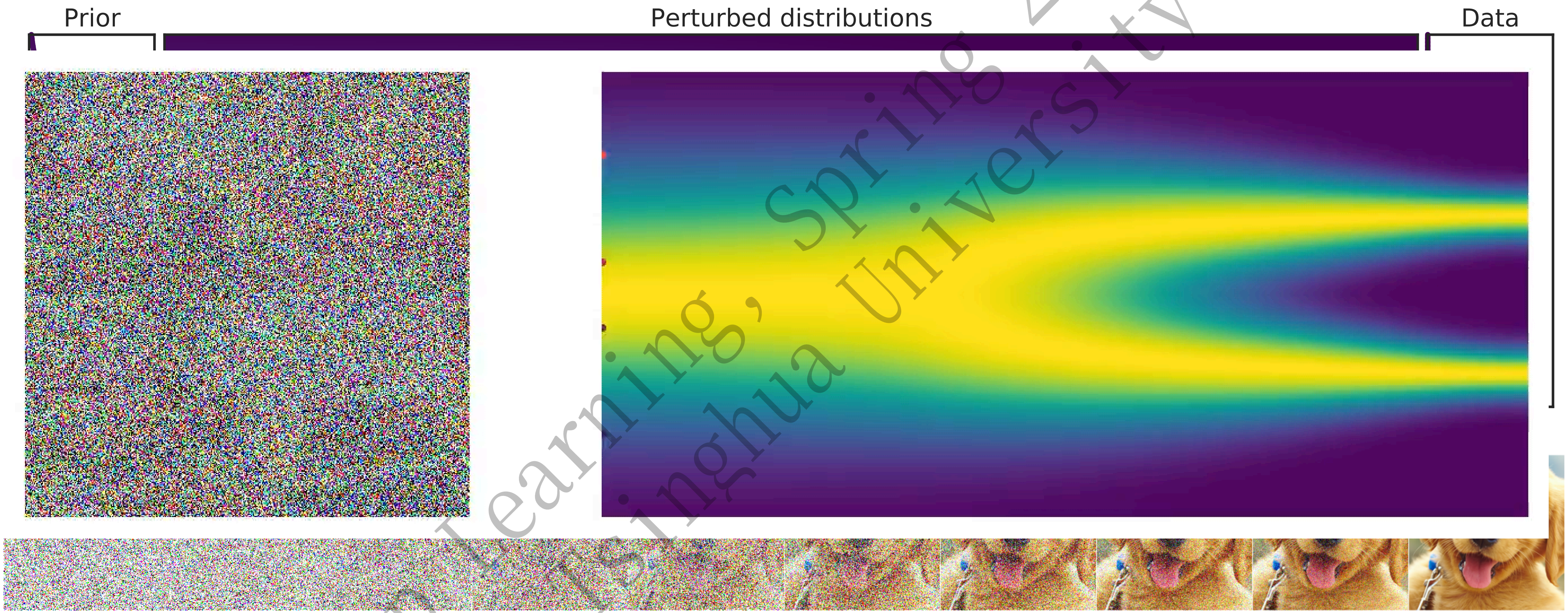
Data



$$\mathrm{d}\mathbf{x} = \sigma(t)\mathrm{d}\mathbf{w}$$



Score-based generative modeling via SDEs



$$\mathrm{d}\mathbf{x} = \sigma(t)\mathrm{d}\mathbf{w} \xrightarrow{\text{Time reversal}} \mathrm{d}\mathbf{x} = -\sigma^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\mathrm{d}t + \sigma(t)\mathrm{d}\bar{\mathbf{w}}$$

$\{p_t(\mathbf{x})\}_{t=0}^{T}$

Score function!

Score-based generative modeling via SDEs

- Time-dependent score-based model

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

- Training:

$$\mathbb{E}_{t \in \mathcal{U}(0,T)}[\lambda(t)\mathbb{E}_{p_t(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2]]$$

- Reverse-time SDE

$$\mathrm{d}\mathbf{x} = -\sigma^2(t)\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\mathrm{d}t + \sigma(t)\mathrm{d}\bar{\mathbf{w}}$$

- Sampling: Euler-Maruyama

$$\mathbf{x} \leftarrow \mathbf{x} - \sigma(t)^2 \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\Delta t + \sigma(t)\ \mathbf{z} \quad (\mathbf{z} \sim \mathcal{N}(\mathbf{0}, |\Delta t|\ \mathbf{I}))$$

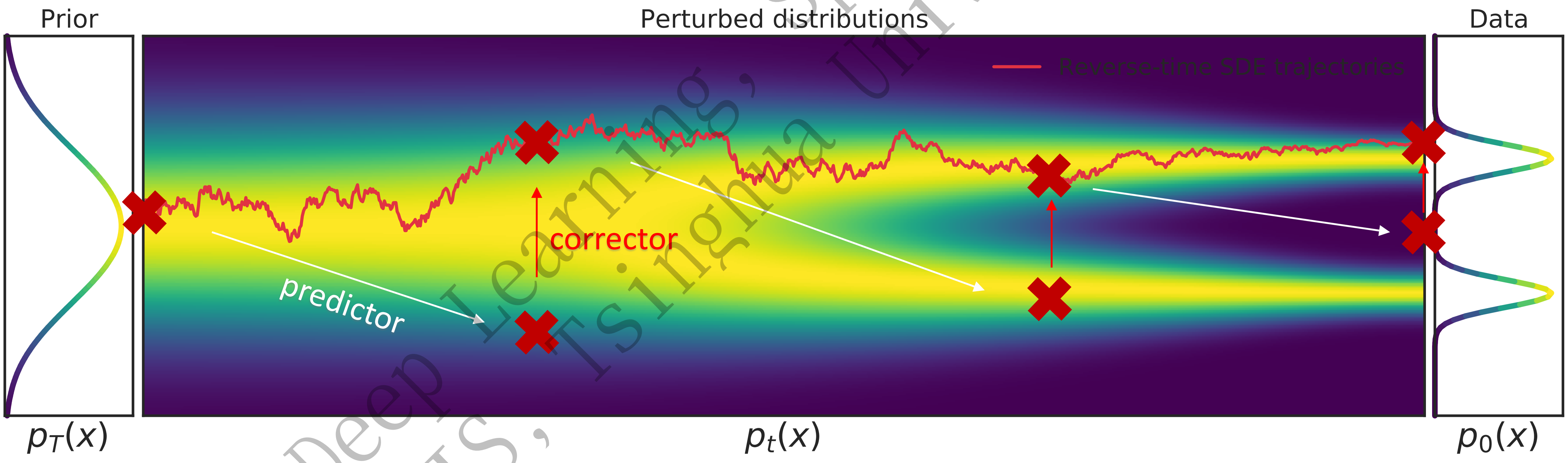
$$t \leftarrow t + \Delta t$$

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.

Faster sampling?

Predictor-Corrector sampling methods

- Predictor-Corrector sampling.
  - **Predictor:** Numerical SDE solver
  - **Corrector:** Score-based MCMC



Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. "Score-Based Generative Modeling through Stochastic Differential Equations." ICLR 2021.

Results on predictor-corrector sampling

Model	FID↓	IS↑
Conditional		
BigGAN (Brock et al., 2018)	14.73	9.22
StyleGAN2-ADA (Karras et al., 2020a)	**2.42**	**10.14**
Unconditional		
StyleGAN2-ADA (Karras et al., 2020a)	2.92	9.83
NCSN (Song & Ermon, 2019)	25.32	$8.87 \pm .12$
NCSNv2 (Song & Ermon, 2020)	10.87	$8.40 \pm .07$
DDPM (Ho et al., 2020)	3.17	$9.46 \pm .11$
DDPM++	2.78	9.64
DDPM++ cont. (VP)	2.55	9.58
DDPM++ cont. (sub-VP)	2.61	9.56
DDPM++ cont. (deep, VP)	2.41	9.68
DDPM++ cont. (deep, sub-VP)	2.41	9.57
NCSN++	2.45	9.73
NCSN++ cont. (VE)	2.38	9.83
NCSN++ cont. (deep, VE)	**2.20**	**9.89**

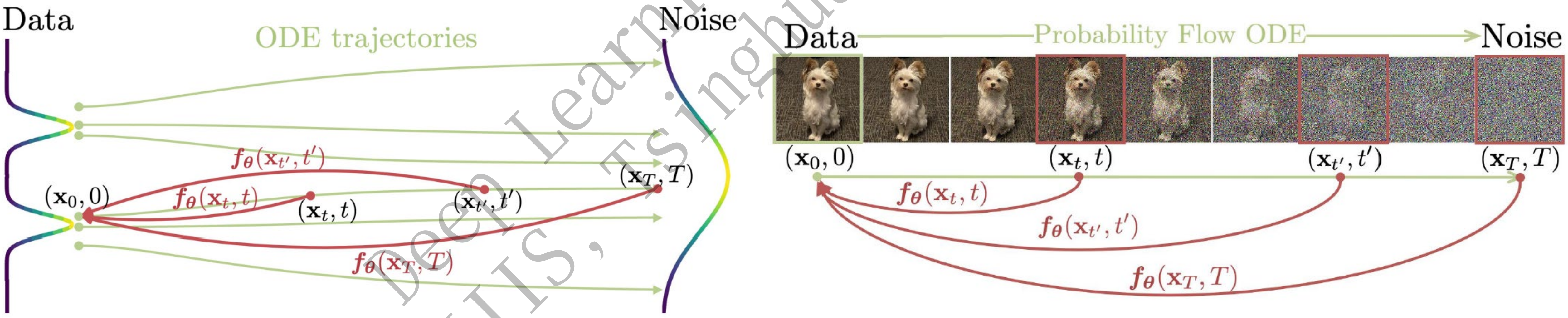
Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. "Score-Based Generative Modeling through Stochastic Differential Equations." ICLR 2021.

High-Fidelity Generation for 1024x1024 Images

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. "Score-Based Generative Modeling through Stochastic Differential Equations." ICLR 2021.

Can we further accelerate inference?

- ODE solver still requires iterative computation
  - Can we make it faster?
- Consistency Models: use a neural network for ODE prediction!
  - Given a smooth ODE, learn $f_\theta(x_t, t) \to x_0$ to map to trajectory origin.
  - The mapping over the same trajectory should be consistent
  - Training by distillation (more details can be found in the CM paper)



Song, Dhariwal, Chen, Sutskever. "Consistency Models." ICML 2023.

Can we further accelerate inference?

- Consistency Models: use a neural network for ODE prediction!
  - Given a smooth ODE, learn $f_\theta(x_t, t) \rightarrow x_0$ to map to trajectory origin.

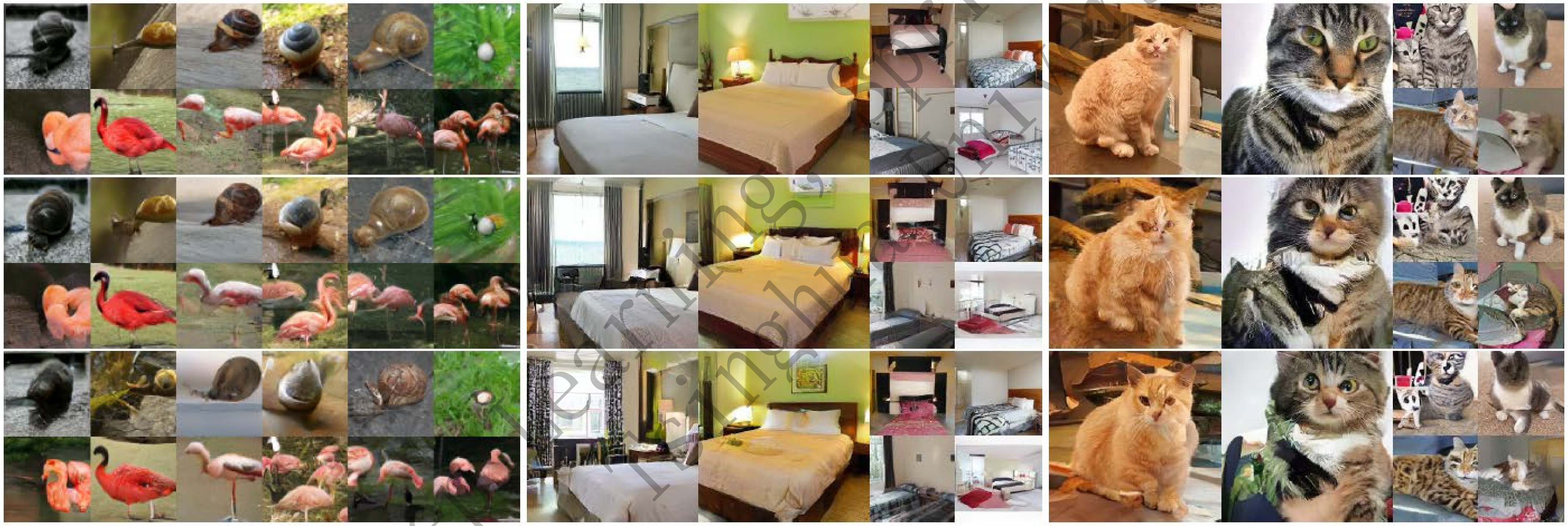
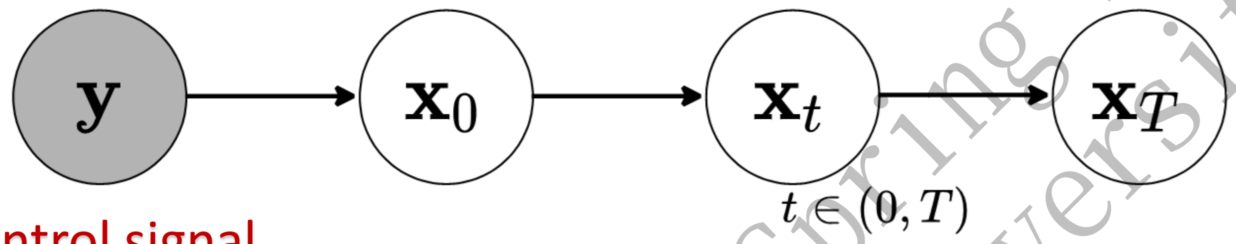


Figure 5: Samples generated by EDM (*top*), CT + single-step generation (*middle*), and CT + 2-step generation (*Bottom*). All corresponding images are generated from the same initial noise.

Song, Dhariwal, Chen, Sutskever. “Consistency Models.” ICML 2023.

Controllable Generation



Control signal

- Conditional reverse-time SDE via unconditional scores

$$\mathrm{d}\mathbf{x} = -\sigma^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \mathbf{y})}\mathrm{d}t + \sigma(t)\mathrm{d}\bar{\mathbf{w}}$$

$$\mathrm{d}\mathbf{x} = -\sigma^2(t)[\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})} + \boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \mathbf{x})}]\mathrm{d}t + \sigma(t)\mathrm{d}\bar{\mathbf{w}}$$

unconditional score,
Trained w/o y

Trained separately or
specified with domain knowledge

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. "Score-Based Generative Modeling through Stochastic Differential Equations." ICLR 2021.

Controllable Generation: class-conditional generation

- $\mathbf{y}$ is the **class label**
- $p_t(\mathbf{y} \mid \mathbf{x})$ is a time-dependent classifier

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. "Score-Based Generative Modeling through Stochastic Differential Equations." ICLR 2021.

Controllable Generation: inpainting

- $\mathbf{y}$ is the **masked image**
- $p_t(\mathbf{y} \mid \mathbf{x})$ can be approximated without training

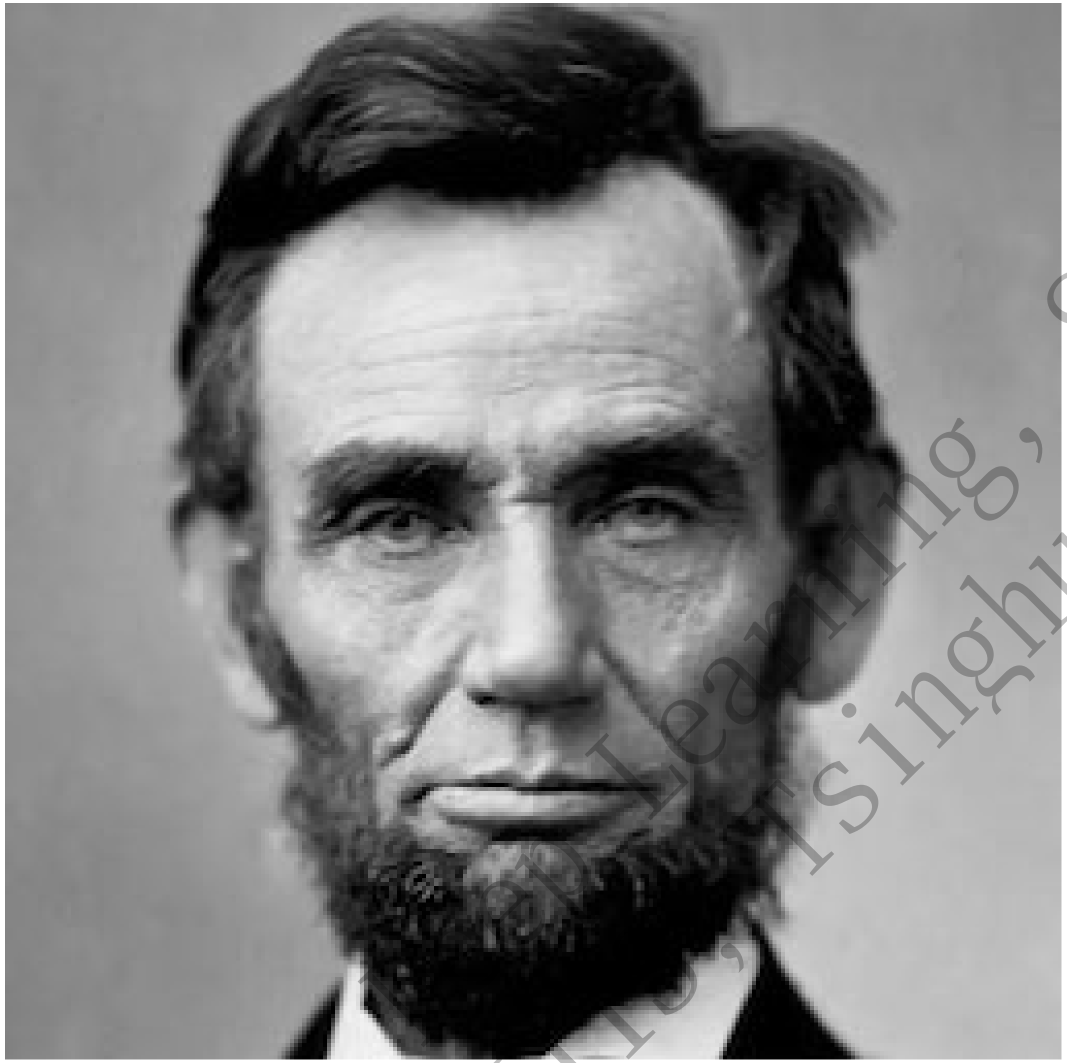
Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. "Score-Based Generative Modeling through Stochastic Differential Equations." ICLR 2021.

Controllable Generation: colorization

- $\mathbf{y}$ is the **gray-scale image**
- $p_t(\mathbf{y} \mid \mathbf{x})$ can be approximated without training

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. "Score-Based Generative Modeling through Stochastic Differential Equations." ICLR 2021.

Controllable Generation: colorization




Resolution: 1024x1024

Controllable generation: Text-guided generation

- An astronaut riding a horse in photorealistic style (Dall-E 2)

- A very attractive and natural woman, sitting on a yoka mat, breathing, eye closed, no make up, intense satisfaction, she looks like she is intensely relaxed, yoga class, sunrise, 35mm, F1: 4 (Midjourney v5)

- Cozy Scandinavian living room, there is a cat sleeping on the couch, depth of field (Midjourney v5)

More on Lecture 12

Build Your Diffusion Model

- Hugging face will be your best friend!
  - Hugging face is a platform for sharing 🤗 models
  - Example: Stable Diffusion Model https://huggingface.co/runwayml/stable-diffusion-v1-5
- How to develop your own model?
  - For example, I want to build a model to generating 二次元 images
  - Re-training can be expensive!

We wanted to know how much time (and money) it would cost to train a Stable Diffusion model from scratch using our Streaming datasets, Composer, and MosaicML platform. Our results: it would take us 79,000 A100-hours in 13 days, for a total training cost of less than $160,000. 2023年1月24日

umber f A100s	Throughput (images / second)	Days to Train on MosaicML Cloud	A100-hours	Approx. Cost on MosaicML Cloud
8	128.2	256.83	49,696	$99,000
16	254.0	130.63	50,166	$100,000
32	485.7	68.33	52,470	$105,000
64	912.2	36.38	55,876	$110,000
128	1619.4	20.5	62,987	$125,000
256*	2,589.4	12.83	78,735	$160,000

mosaicml.com
https://www.mosaicml.com › blog › training-stable-diffu...

Training Stable Diffusion from Scratch Costs <$160k - MosaicML

Build Your Diffusion Model

- LORA: Low-rank adaptation of large language models (MSR, 2021)
  - Low-rank hypothesis
    - Neural network models with over-parameterization reside in a low intrinsic dimension
    - Fine-tuning can be also performed with a “low-rank” fashion
  - Low-rank decomposition of additive weights
    - Model weights are frozen
    - A is initialized to small Gaussian noise
    - B is initialized to zero
    - https://github.com/microsoft/LoRA
  - Lora becomes extremely popular these days …
    - Some examples on next slide …

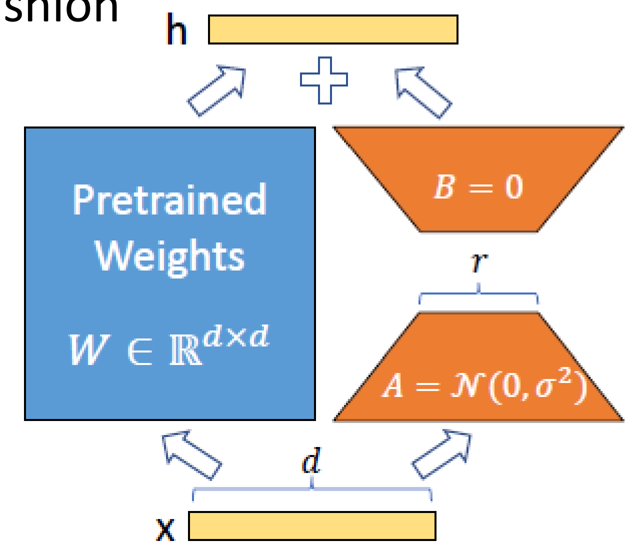
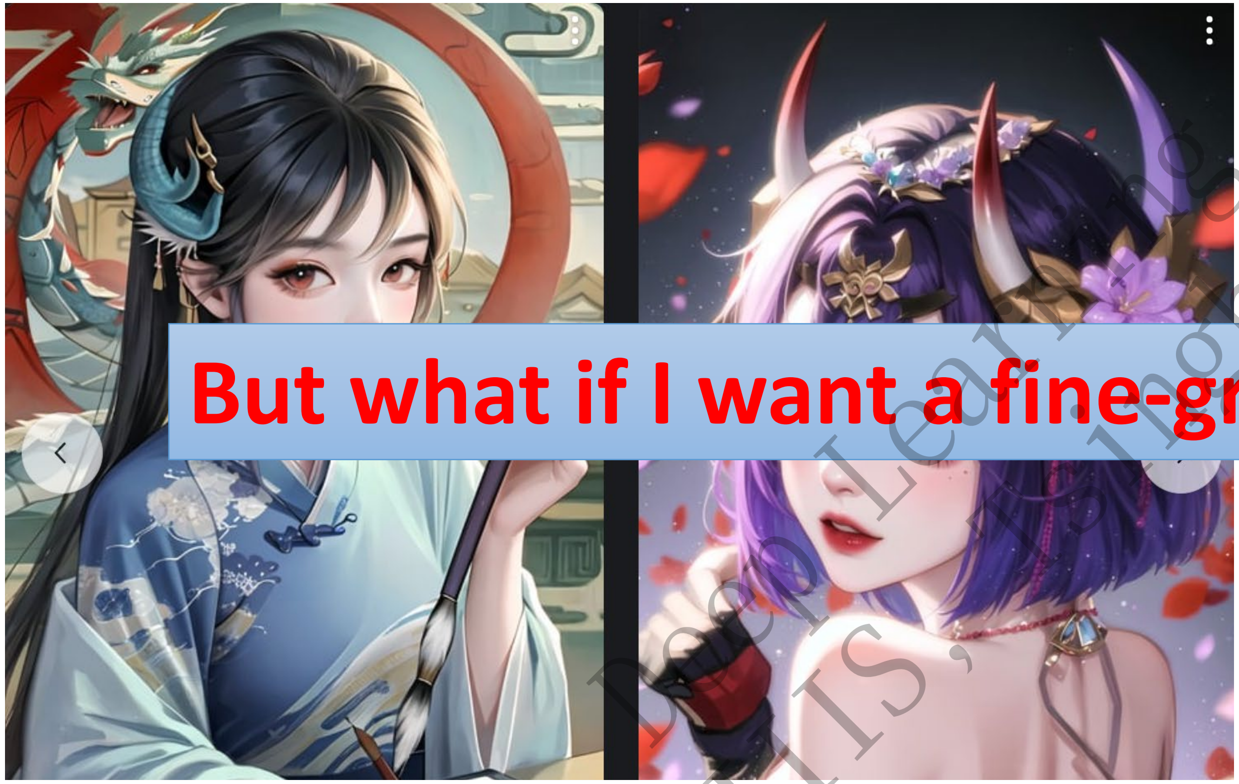
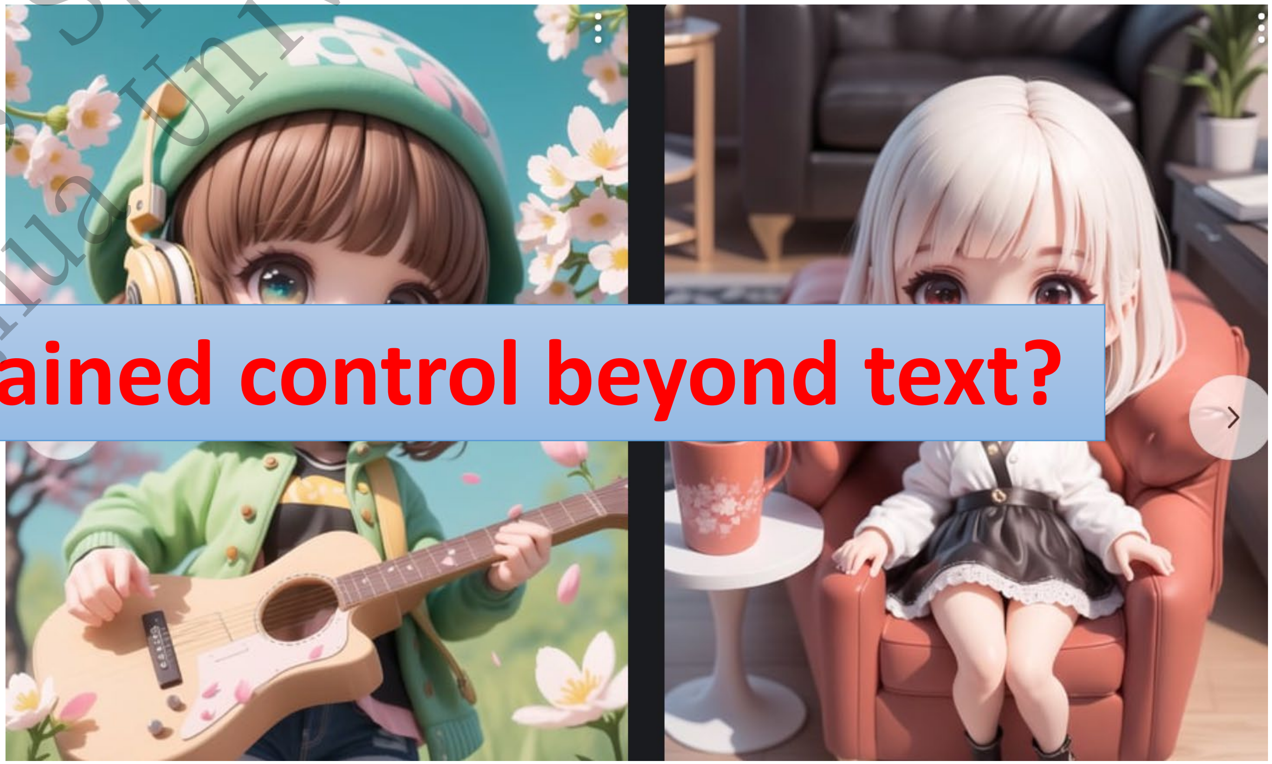


Figure 1: Our reparametrization. We only train A and B.

Build Your Diffusion Model: Examples

- Fashion Girl
- Blind Box





But what if I want a fine-grained control beyond text?

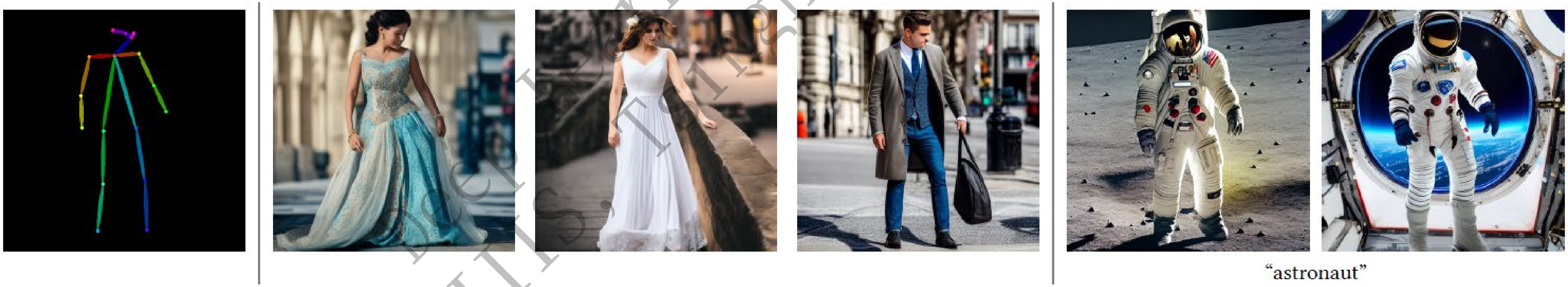
Create your own art from text, with your chosen style (Lora)!

Control Your Model

- ControlNet: Adding Conditional Control to Text-to-Image Diffusion Models (Stanford, 2023)
  - Let the model also condition on additional signal
    - Example: Canny edge detection
    - https://github.com/lllyasviel/ControlNet

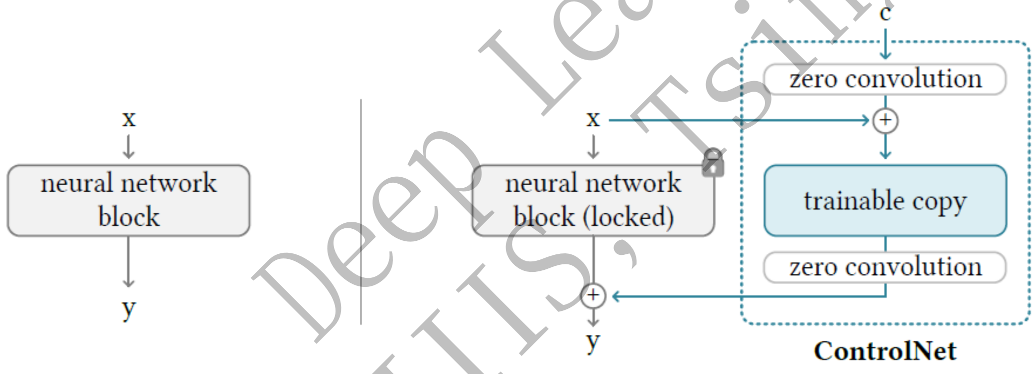
Control Your Model

- ControlNet: Adding Conditional Control to Text-to-Image Diffusion Models (Stanford, 2023)
  - Let the model also condition on additional signal
    - Example: Human Motion
    - https://github.com/lllyasviel/ControlNet

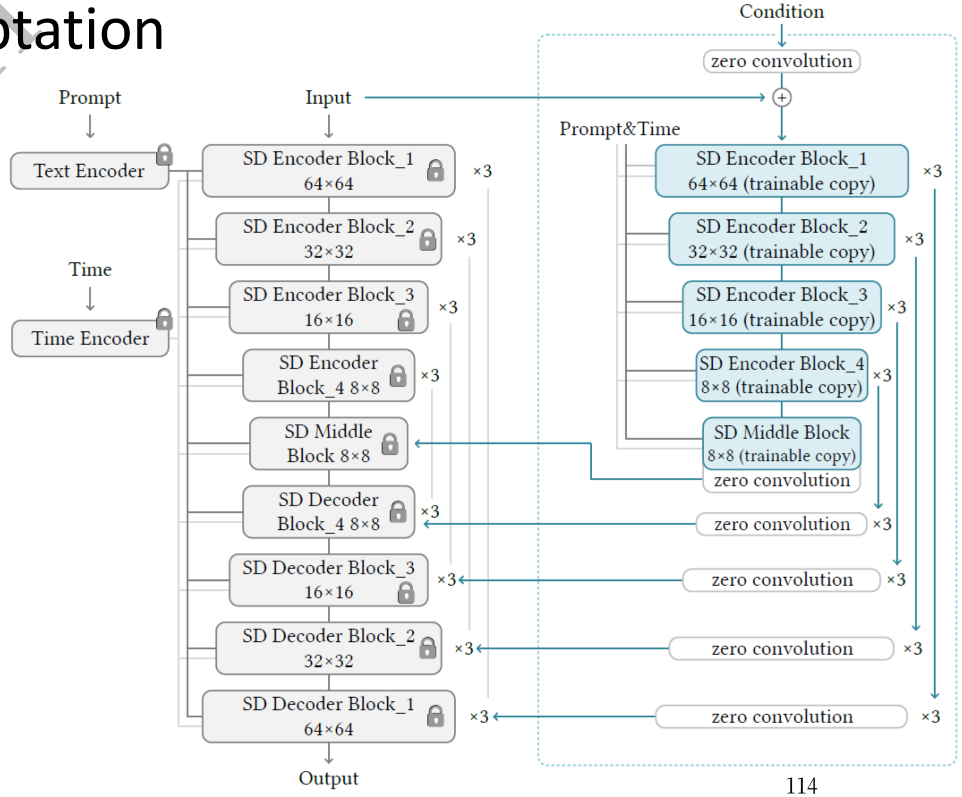


Control Your Model

- ControlNet: Adding Conditional Control to Text-to-Image Diffusion Models (Stanford, 2023)
  - Similar idea to Lora: frozen weight + small adaptation
  - Key techniques:
    - Zero-convolution: 1x1 conv-layer initializes to zero
    - Trainable copy for adaptation initialization
    - Repeated additive for conditioning



(a) Before (b) After



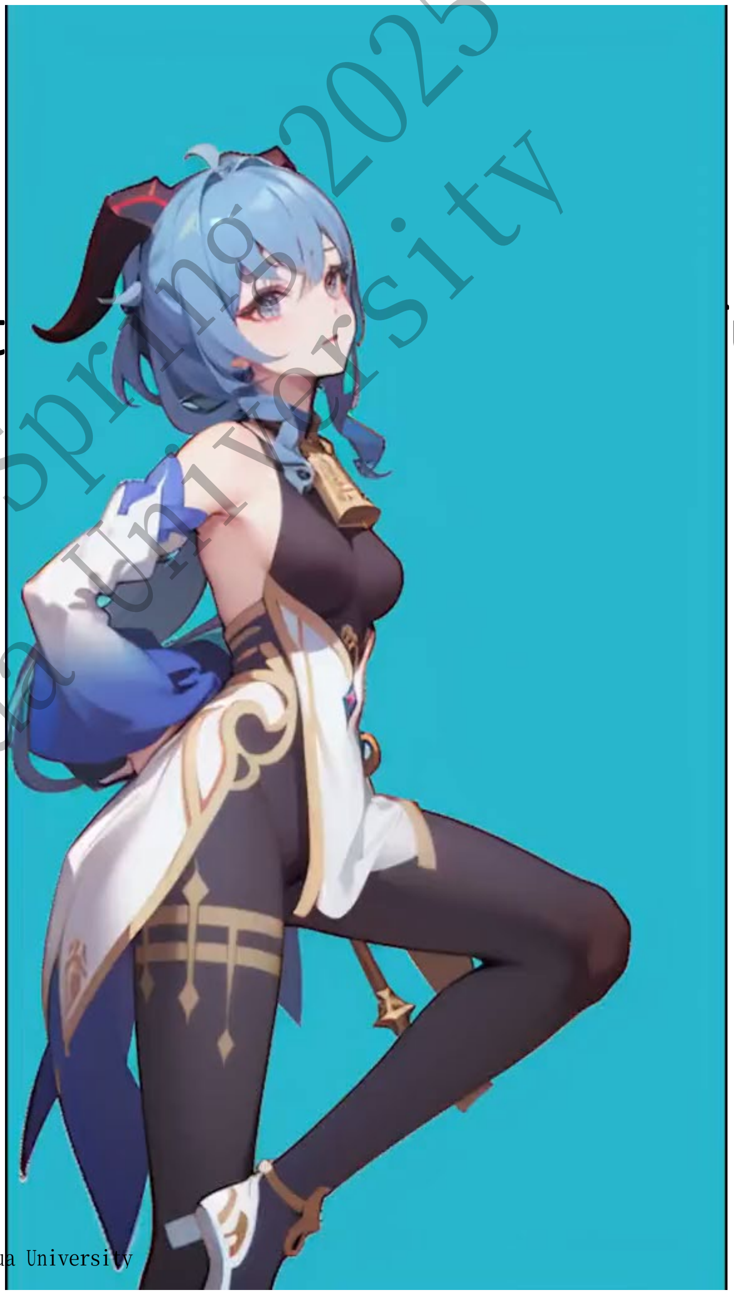
(a) Stable Diffusion (b) ControlNet

Control Your Model

- ControlNet: Adding Conditional Control to Text-to-Image Diffusion Models (Stanford, 2023)

- More ControlNet examples
  - Home designer

Control Your Model

- ControlNet: Adding Conditional Cont  fusion Models (Stanford, 2023)

- More ControlNet examples
  - Home designer
  - Even video!
    - more in future lectures…

Summary: Diffusion Model

- Advanced Generative Model
    - Diffusion Model and Score-Based Model
        - Score matching for gradients of log probability
    - Training & Inference (with condition)
        - Noise conditioned network
        - Langevin dynamics and SDE for fast sampling
        - Conditioned generation without the need of retraining
- Frontier AIGC: LORA and ControlNet
    - Hugging Face is your friend

Generative Model (Summary)

- Goal of generative model
  - Learn a distribution $p(x)$ to generate samples and unsupervised learning
- Models so far
  - Energy-based model
    - $p(x) = \frac{1}{Z}\exp(-E(x))$, powerful representation but hard to sample
  - Variational auto-encoder
    - $p(x,z) = p(z)p(x|z)$, variational inference as an approximate method
  - Generative adversarial net
    - $G(z)$ and $D(x)$, an implicit model with high generation quality and unstable training
  - Normalizing flow
    - $x = f(z)$, best mathematical properties but the most restricted representation
  - Score-based models
    - $s(x) = \nabla_x p(x;\theta)$, highest generation quality + stable training, but generation is slow

Generative Model (Summary)

- Goal of generative model
  - Learn a distribution $p(x)$ to generate samples and unsupervised learning
- Models so far
  - Energy-based model
    - $p(x) = \frac{1}{z}\exp(-E(x))$, powerful representation but hard to sample

Coming next: generating data samples beyond images!

  - Variational auto-encoder
    - $p(x,z) = p(z)p(x|z)$, variational inference as an approximate method
  - Generative adversarial net
    - $G(z)$ and $D(x)$, an implicit model with high generation quality and unstable training
  - Normalizing flow
    - $x = f(z)$, best mathematical properties but the most restricted representation
  - Score-based models
    - $s(x) = \nabla_x p(x;\theta)$, highest generation quality + stable training, but generation is slow

Thanks!