

# DIFFUSION MODELS

CSE 705 SEMINAR (FALL 2024) DAY 1

Presented By:

Team Sushi (Vamshi Krishna Kyatham,  
Rahul Dasari, Xiaofeng Chen)

 University at Buffalo  
Department of Computer Science  
and Engineering  
School of Engineering and Applied Sciences





University at Buffalo

Department of Computer Science  
and Engineering

School of Engineering and Applied Sciences

# WHAT ARE GENERATIVE MODELS?



# Generative Models

Generative models try to model the probability density function of the data and sample from it or directly sample from the distribution.

Explicit models: (They model the distribution)

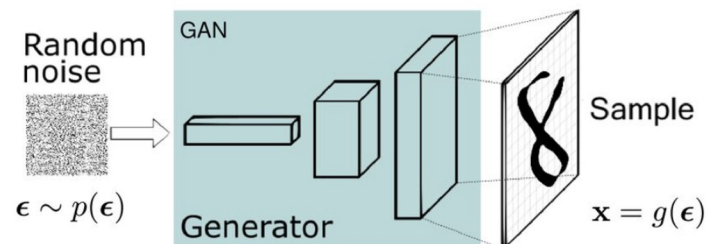
- Autogressive models
- Variational autoencoders

$$p_{\theta}(\mathbf{x}) = \prod_{i=1}^d p_{\theta}(\mathbf{x}_i | \mathbf{x}_{<i})$$

$$p_{\theta}(\mathbf{x}) = \int p(\mathbf{z})p_{\theta}(\mathbf{x} | \mathbf{z}) d\mathbf{z}$$

Implicit models: (They directly sample from the distribution)

- GANs





University at Buffalo

Department of Computer Science  
and Engineering

School of Engineering and Applied Sciences

# SCORE BASED GENERATIVE MODELS



# What is Score?

**Energy-based model:**  $p_{\theta}(\mathbf{x}) = \frac{\exp\{f_{\theta}(\mathbf{x})\}}{Z(\theta)}$  ,  $\log p_{\theta}(\mathbf{x}) = f_{\theta}(\mathbf{x}) - \log Z(\theta)$

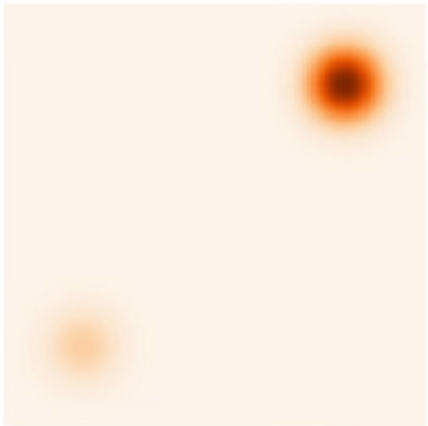
**(Stein) Score function:**

$$\begin{aligned} s_{\theta}(\mathbf{x}) &= \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}) \\ &= \nabla_{\mathbf{x}} \log \frac{e^{-f_{\theta}(\mathbf{x})}}{Z_{\theta}} \\ &= \nabla_{\mathbf{x}} \log e^{-f_{\theta}(\mathbf{x})} - \underbrace{\nabla_{\mathbf{x}} \log Z_{\theta}}_{=0} \\ &= -\nabla_{\mathbf{x}} f_{\theta}(\mathbf{x}) \end{aligned}$$

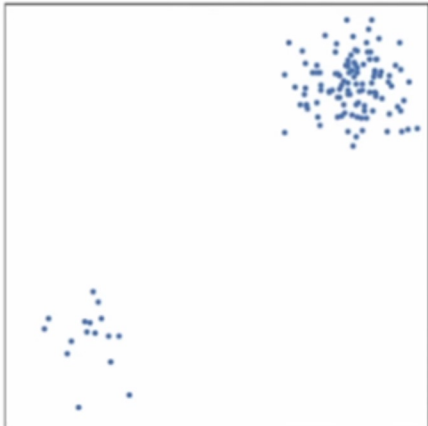
Score is essentially the field of the probability density function

# What is score

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

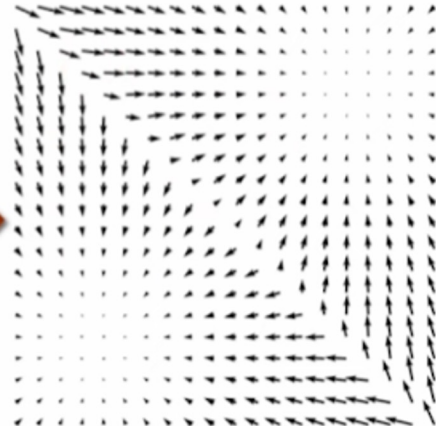


i.i.d. samples  
 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$



Score function

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



# Score Matching

## Observation

$s_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$  is independent of the partition function  $Z(\theta)$ .

Fisher divergence between  $p(\mathbf{x})$  and  $q(\mathbf{x})$ :

$$D_F(p, q) := \frac{1}{2} E_{\mathbf{x} \sim p} [\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \nabla_{\mathbf{x}} \log q(\mathbf{x})\|_2^2]$$

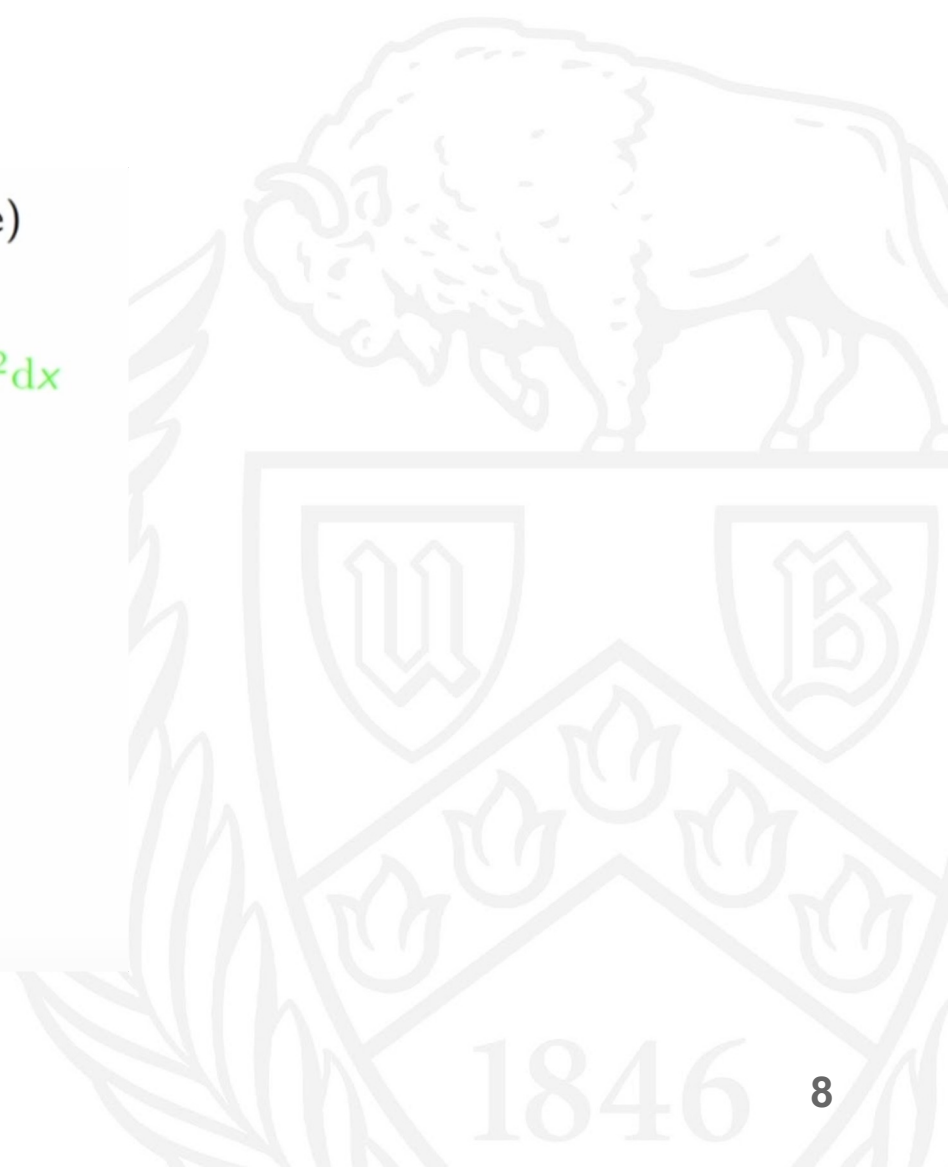
$$\begin{aligned} & \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - s_\theta(\mathbf{x})\|_2^2] \\ &= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} f_\theta(\mathbf{x})\|_2^2] \end{aligned}$$

# Score Matching

$$\begin{aligned}
 & \frac{1}{2} E_{x \sim p_{\text{data}}} [(\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_{\theta}(x))^2] \quad (\text{Univariate case}) \\
 &= \frac{1}{2} \int p_{\text{data}}(x) [(\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_{\theta}(x))^2] dx \\
 &= \frac{1}{2} \int p_{\text{data}}(x) (\nabla_x \log p_{\text{data}}(x))^2 dx + \frac{1}{2} \int p_{\text{data}}(x) (\nabla_x \log p_{\theta}(x))^2 dx \\
 &\quad - \int p_{\text{data}}(x) \nabla_x \log p_{\text{data}}(x) \nabla_x \log p_{\theta}(x) dx
 \end{aligned}$$

Recall Integration by parts:  $\int f'g = fg - \int g'f$ .

$$\begin{aligned}
 & - \int p_{\text{data}}(x) \nabla_x \log p_{\text{data}}(x) \nabla_x \log p_{\theta}(x) dx \\
 &= - \int p_{\text{data}}(x) \frac{1}{p_{\text{data}}(x)} \nabla_x p_{\text{data}}(x) \nabla_x \log p_{\theta}(x) dx \\
 &= \underbrace{-p_{\text{data}}(x) \nabla_x \log p_{\theta}(x) \Big|_{x=-\infty}^{\infty}}_{=0} + \int p_{\text{data}}(x) \nabla_x^2 \log p_{\theta}(x) dx \\
 &= \int p_{\text{data}}(x) \nabla_x^2 \log p_{\theta}(x) dx
 \end{aligned}$$





# Score Matching

Univariate score matching

$$\begin{aligned} & \frac{1}{2} E_{x \sim p_{\text{data}}} [(\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_{\theta}(x))^2] \\ &= \frac{1}{2} \int p_{\text{data}}(x) (\nabla_x \log p_{\text{data}}(x))^2 dx + \frac{1}{2} \int p_{\text{data}}(x) (\nabla_x \log p_{\theta}(x))^2 dx \\ & \quad - \int p_{\text{data}}(x) \nabla_x \log p_{\text{data}}(x) \nabla_x \log p_{\theta}(x) dx \\ &= \underbrace{\frac{1}{2} \int p_{\text{data}}(x) (\nabla_x \log p_{\text{data}}(x))^2 dx}_{\text{const. wrt } \theta} + \frac{1}{2} \int p_{\text{data}}(x) (\nabla_x \log p_{\theta}(x))^2 dx \\ & \quad + \int p_{\text{data}}(x) \nabla_x^2 \log p_{\theta}(x) dx \\ &= E_{x \sim p_{\text{data}}} \left[ \frac{1}{2} (\nabla_x \log p_{\theta}(x))^2 + \nabla_x^2 \log p_{\theta}(x) \right] + \text{const.} \end{aligned}$$



# Score Matching for Multivariate case

$$E_{\mathbf{x} \sim p_{\text{data}}} \left[ \frac{1}{2} (\nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}))^2 + \nabla_{\mathbf{x}}^2 \log p_{\theta}(\mathbf{x}) \right] + \text{const.}$$

$$\begin{aligned} & \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x})\|_2^2] \\ &= E_{\mathbf{x} \sim p_{\text{data}}} \left[ \frac{1}{2} \|\nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x})\|_2^2 + \text{tr} \left( \underbrace{\nabla_{\mathbf{x}}^2 \log p_{\theta}(\mathbf{x})}_{\quad} \right) \right] \end{aligned}$$

$$\begin{aligned} & \frac{1}{n} \sum_{i=1}^n \left[ \frac{1}{2} \|\nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}_i)\|_2^2 + \text{tr}(\nabla_{\mathbf{x}}^2 \log p_{\theta}(\mathbf{x}_i)) \right] \\ &= \frac{1}{n} \sum_{i=1}^n \left[ \frac{1}{2} \|\nabla_{\mathbf{x}} f_{\theta}(\mathbf{x}_i)\|_2^2 + \text{trace}(\nabla_{\mathbf{x}}^2 f_{\theta}(\mathbf{x}_i)) \right] \end{aligned}$$

The derivative of the gradient simplifies to the trace of the hessian of  $P(\mathbf{x})$

# Score Matching

The objective is to maximise fisher divergence by using score matching.

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

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

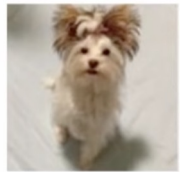
(Fisher divergence)

- **Score matching:**

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

Computing the hessian is expensive

# Denoising Score Matching



$\mathbf{x}$

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

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



$\tilde{\mathbf{x}}$

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

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

yet again we try to compute the loss

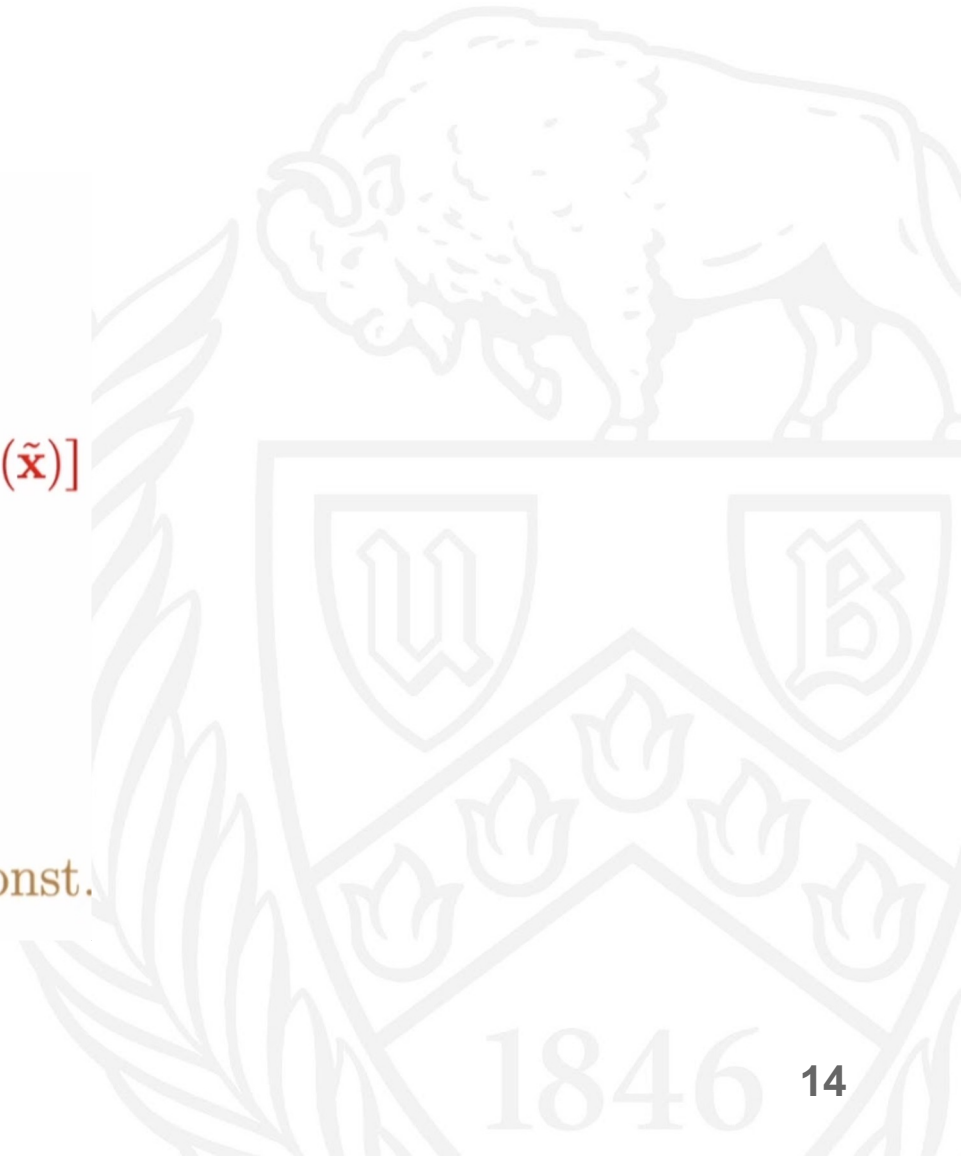
# Denoising Score Matching

$$\begin{aligned}
 & - \int q_{\sigma}(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log q_{\sigma}(\tilde{\mathbf{x}})^{\top} \mathbf{s}_{\theta}(\tilde{\mathbf{x}}) d\tilde{\mathbf{x}} \\
 = & - \int q_{\sigma}(\tilde{\mathbf{x}}) \frac{1}{q_{\sigma}(\tilde{\mathbf{x}})} \nabla_{\tilde{\mathbf{x}}} q_{\sigma}(\tilde{\mathbf{x}})^{\top} \mathbf{s}_{\theta}(\tilde{\mathbf{x}}) d\tilde{\mathbf{x}} \\
 = & - \int \nabla_{\tilde{\mathbf{x}}} q_{\sigma}(\tilde{\mathbf{x}})^{\top} \mathbf{s}_{\theta}(\tilde{\mathbf{x}}) d\tilde{\mathbf{x}} \\
 = & - \int \nabla_{\tilde{\mathbf{x}}} \left( \int p_{\text{data}}(\mathbf{x}) q_{\sigma}(\tilde{\mathbf{x}} | \mathbf{x}) d\mathbf{x} \right)^{\top} \mathbf{s}_{\theta}(\tilde{\mathbf{x}}) d\tilde{\mathbf{x}} \\
 = & - \int \left( \int p_{\text{data}}(\mathbf{x}) \nabla_{\tilde{\mathbf{x}}} q_{\sigma}(\tilde{\mathbf{x}} | \mathbf{x}) d\mathbf{x} \right)^{\top} \mathbf{s}_{\theta}(\tilde{\mathbf{x}}) d\tilde{\mathbf{x}} \\
 = & - \int \left( \int p_{\text{data}}(\mathbf{x}) q_{\sigma}(\tilde{\mathbf{x}} | \mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log q_{\sigma}(\tilde{\mathbf{x}} | \mathbf{x}) d\mathbf{x} \right)^{\top} \mathbf{s}_{\theta}(\tilde{\mathbf{x}}) d\tilde{\mathbf{x}} \\
 = & - \iint p_{\text{data}}(\mathbf{x}) q_{\sigma}(\tilde{\mathbf{x}} | \mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log q_{\sigma}(\tilde{\mathbf{x}} | \mathbf{x})^{\top} \mathbf{s}_{\theta}(\tilde{\mathbf{x}}) d\mathbf{x} d\tilde{\mathbf{x}}
 \end{aligned}$$



# Denoising Score Matching

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



# Denoising Score Matching

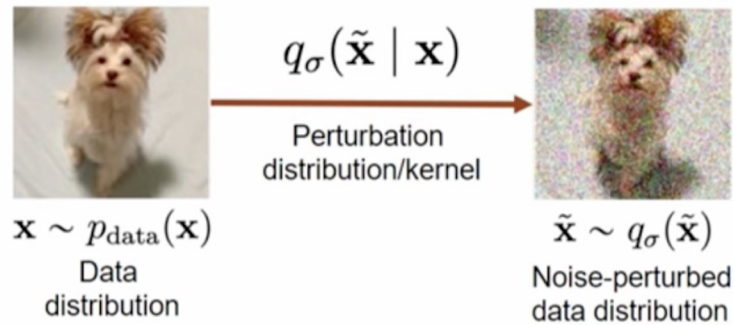
$$J_{DSM_\sigma}(\theta) = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} \left[ \frac{1}{2} \left\| s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) \right\|_2^2 \right]$$

$$\begin{aligned} \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) &= \nabla_{\tilde{\mathbf{x}}} \log \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 \cdot \mathbf{I}) \\ &= \nabla_{\tilde{\mathbf{x}}} \log \frac{\exp(-\frac{1}{2}(\tilde{\mathbf{x}} - \mathbf{x})^T \cdot (\sigma^2 \cdot \mathbf{I})^{-1} \cdot (\tilde{\mathbf{x}} - \mathbf{x}))}{\sqrt{(2\pi)^d |\sigma^2 \cdot \mathbf{I}|}} \\ &= \nabla_{\tilde{\mathbf{x}}} \log \exp(-\frac{1}{2}(\tilde{\mathbf{x}} - \mathbf{x})^T \cdot (\sigma^2 \cdot \mathbf{I})^{-1} \cdot (\tilde{\mathbf{x}} - \mathbf{x})) \\ &\quad - \underbrace{\nabla_{\tilde{\mathbf{x}}} \log \sqrt{(2\pi)^d |\sigma^2 \cdot \mathbf{I}|}}_{=0} \\ &= -\frac{1}{2\sigma^2} \nabla_{\tilde{\mathbf{x}}} (\tilde{\mathbf{x}} - \mathbf{x})^T \cdot \mathbf{I} \cdot (\tilde{\mathbf{x}} - \mathbf{x}) \\ &= -\frac{1}{2\sigma^2} \nabla_{\tilde{\mathbf{x}}} (\tilde{\mathbf{x}} - \mathbf{x})^2 \\ &= -\frac{(\tilde{\mathbf{x}} - \mathbf{x})}{\sigma^2} = \frac{(\mathbf{x} - \tilde{\mathbf{x}})}{\sigma^2}. \end{aligned}$$

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



# Denoising Score Matching



$$\begin{aligned}
 & \frac{1}{2} E_{\tilde{\mathbf{x}} \sim p_{\text{data}}} [\| \mathbf{s}_{\theta}(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_{\sigma}(\tilde{\mathbf{x}}) \|_2^2] \\
 &= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x})} [\| \mathbf{s}_{\theta}(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_{\sigma}(\tilde{\mathbf{x}} | \mathbf{x}) \|_2^2] + \text{const} \\
 &= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \left\| \mathbf{s}_{\theta}(\mathbf{x} + \sigma \mathbf{z}) + \frac{\mathbf{z}}{\sigma} \right\|_2^2 \right] + \text{const}.
 \end{aligned}$$

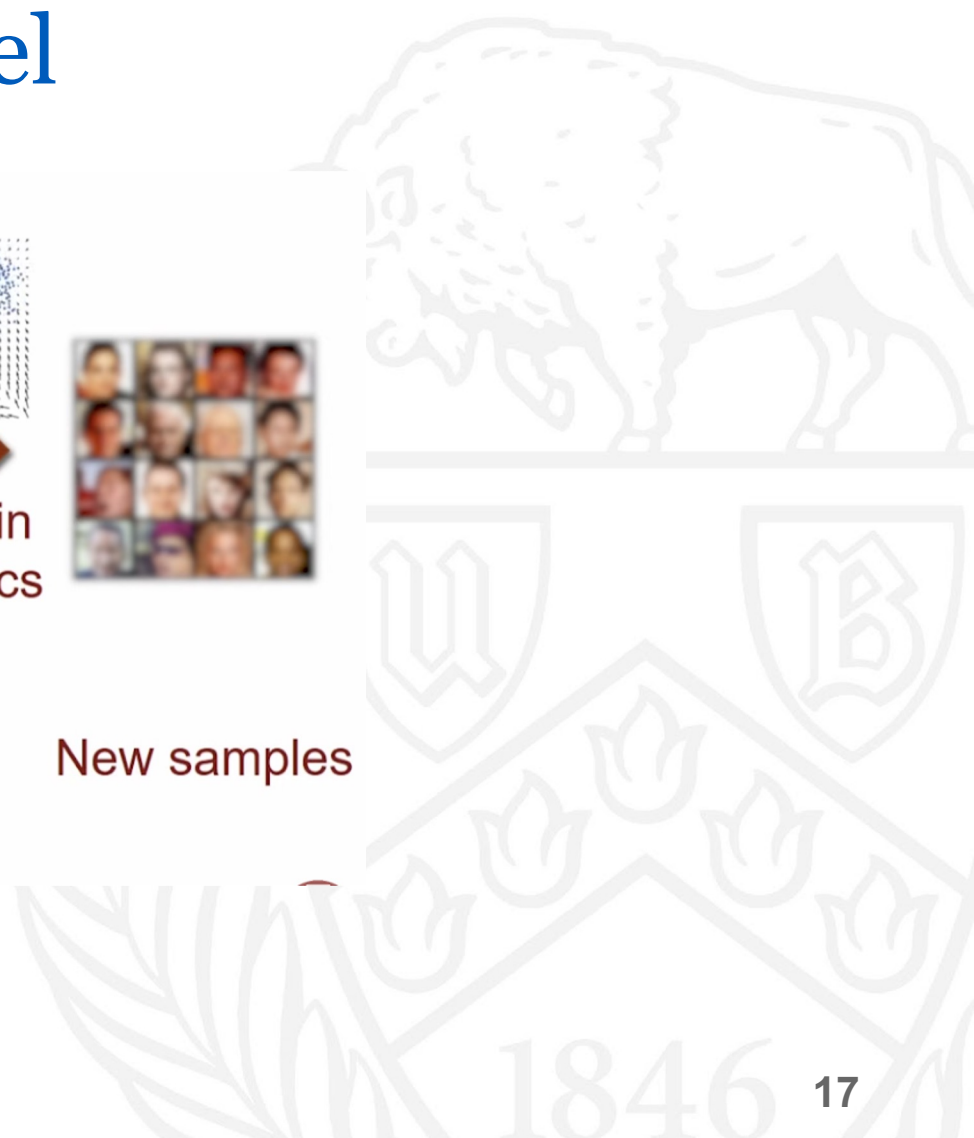
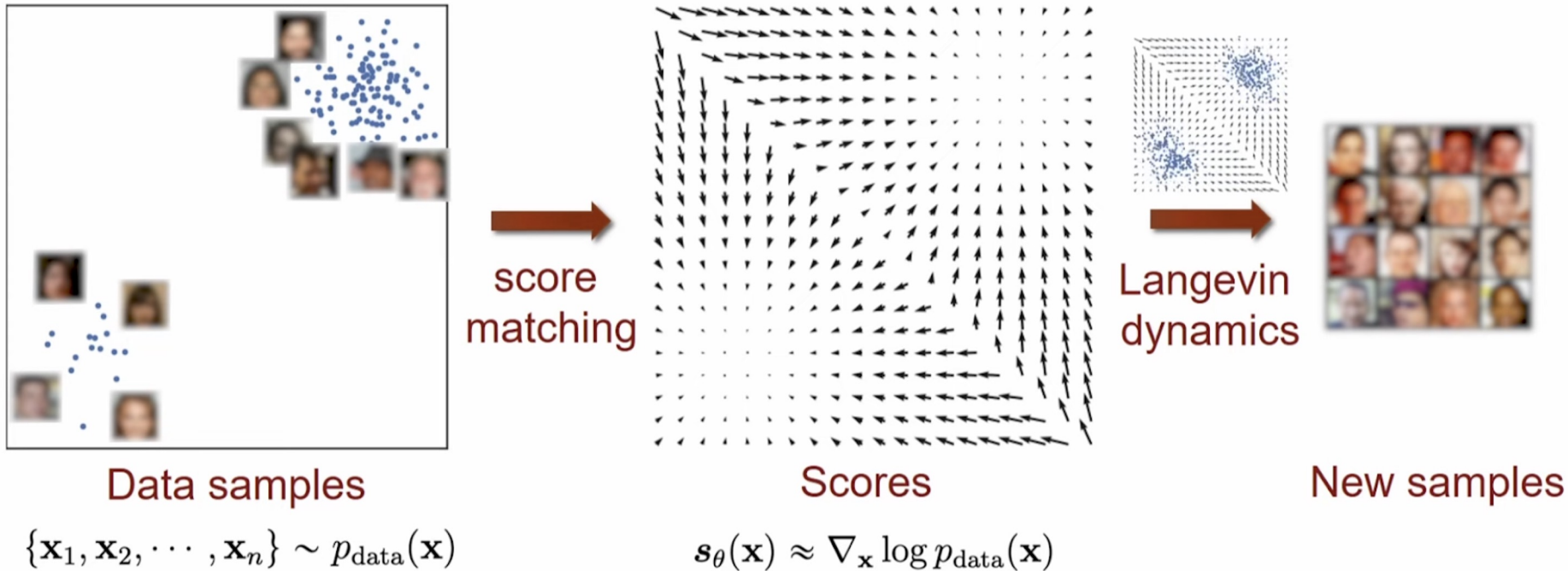
$\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$   
 Data distribution

$\tilde{\mathbf{x}} \sim q_{\sigma}(\tilde{\mathbf{x}})$   
 Noise-perturbed data distribution

This formulation avoids computing the hessian, hence its computationally efficient



# Sampling from a score based model



# Sampling from a score based model

Langevin dynamics can produce samples from a probability density  $p(\mathbf{x})$  using only the score function  $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ . Given a fixed step size  $\epsilon > 0$ , and an initial value  $\tilde{\mathbf{x}}_0 \sim \pi(\mathbf{x})$  with  $\pi$  being a prior distribution, the Langevin method recursively computes the following

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2} \nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon} \mathbf{z}_t \quad \text{where } \mathbf{z}_t \sim \mathcal{N}(0, I).$$

similar to gradient ascent, except that we add a noise

# Challenges of score based Generative Modeling

Now we analyze more closely the idea of score-based generative modeling. We argue that there are obstacles that prevent a naïve application of this idea.

- > Manifold hypothesis
- > Low density regions

# The manifold hypothesis

The manifold hypothesis states that data in the real world tend to concentrate on low dimensional manifolds embedded in a high dimensional space.

Under the manifold hypothesis, score-based generative models will face two key difficulties.

- > First, since the score  $\nabla \log p(x)$  is a gradient taken in a higher dimensional space, it is undefined when  $x$  is confined to a low dimensional manifold.
- > Second, the score matching objective provides a consistent score estimator only when the support of the data distribution is the whole space and will be inconsistent when the data reside on a low-dimensional manifold.

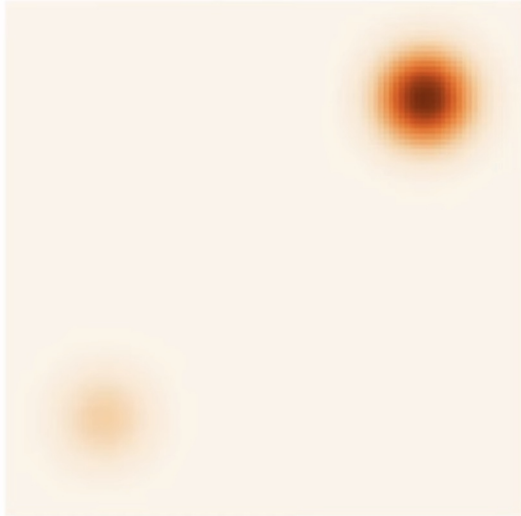
# Low data density regions

The scarcity of data in low density regions can cause difficulties for both score estimation with score matching and sampling with Langevin dynamics.

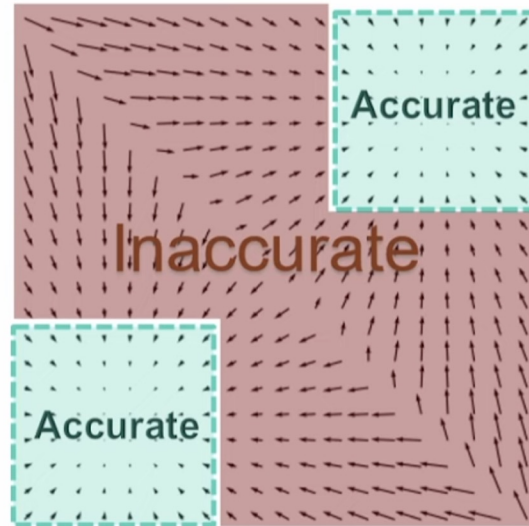
- > In regions of low data density, score matching may not have enough evidence to estimate score functions accurately, due to the lack of data samples.
- > When two modes of the data distribution are separated by low density regions, Langevin dynamics will not be able to correctly recover the relative weights of these two modes in reasonable time, and therefore might not converge to the true distribution.

# Noise Conditional Score Networks

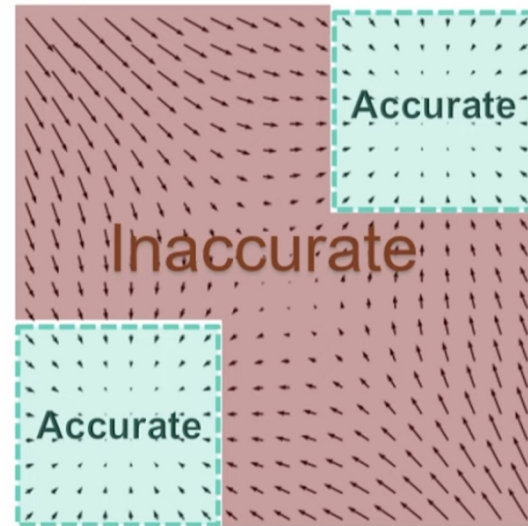
Data density



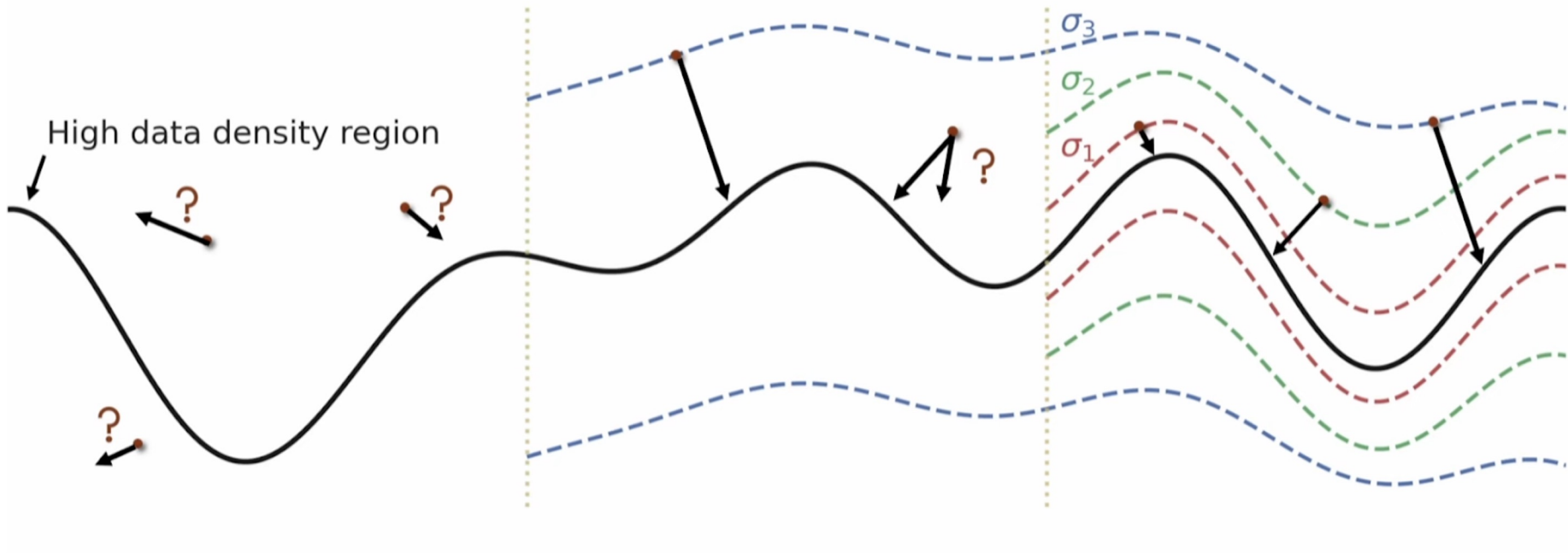
Data scores



Estimated scores

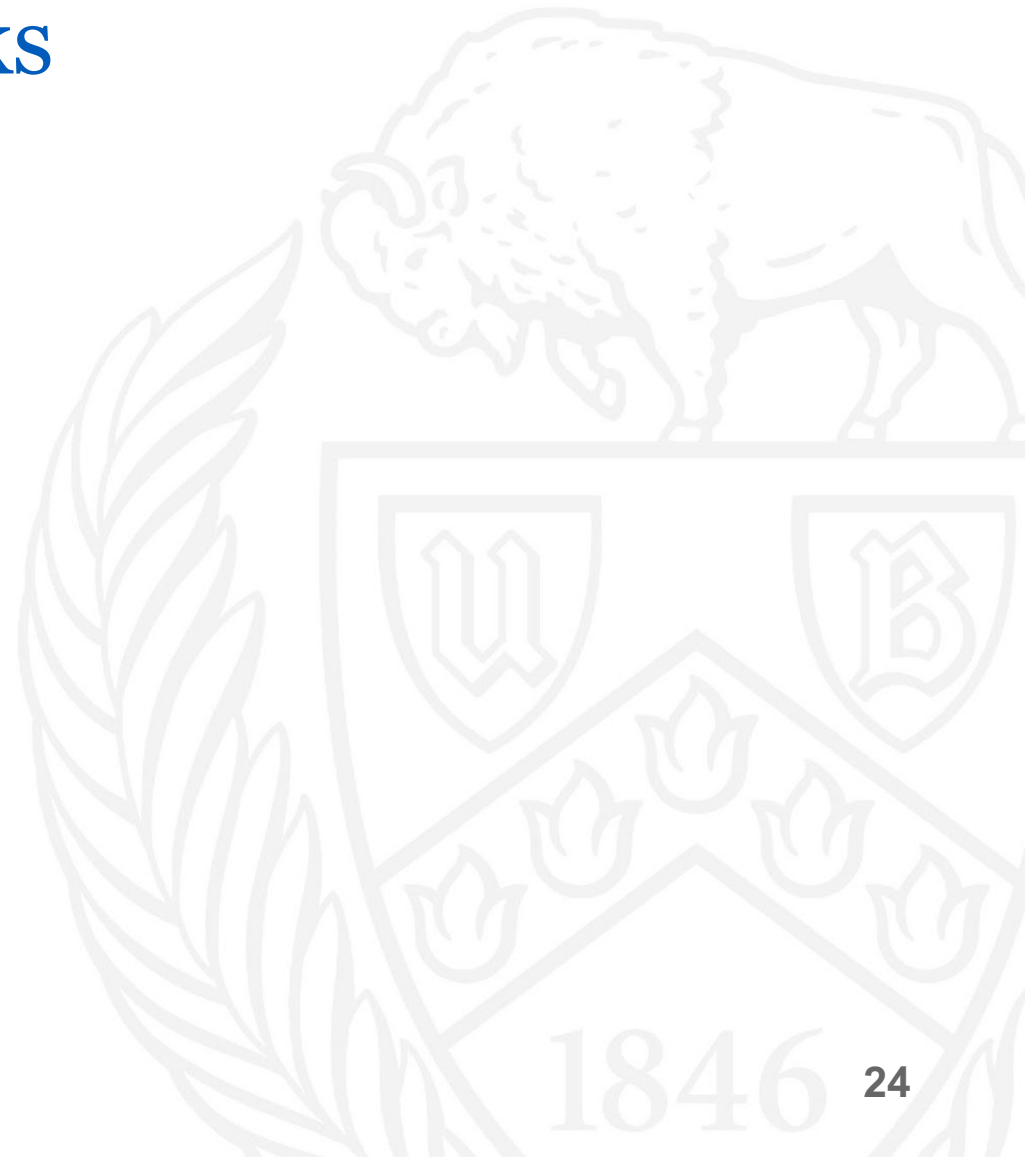


# Noise Conditional Score Networks



# Noise Conditional Score Networks

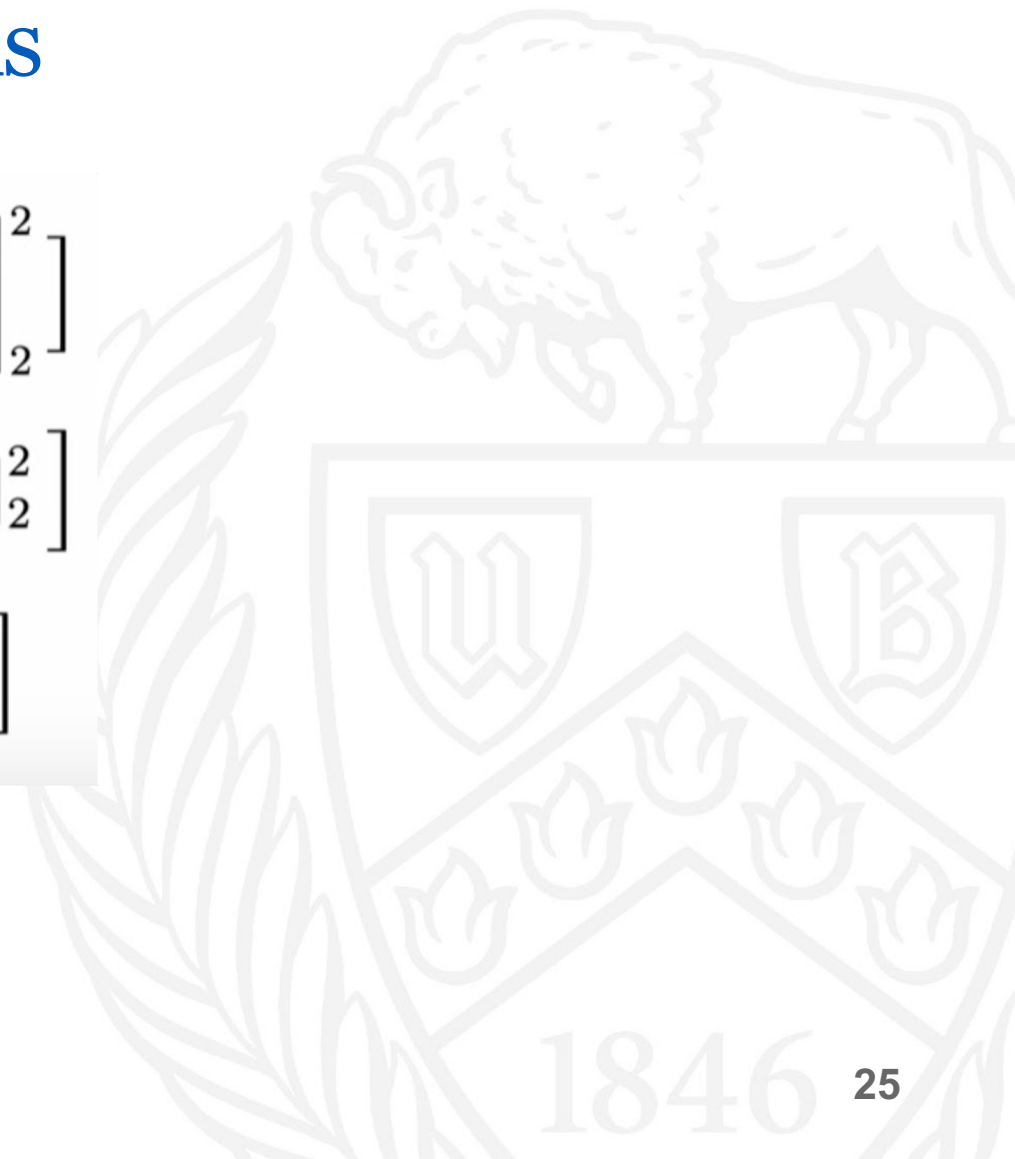
$$\begin{aligned} & \frac{1}{L} \sum_{i=1}^L \lambda(\sigma_i) E_{q_{\sigma_i}(\mathbf{x})} [\|\nabla_{\mathbf{x}} \log q_{\sigma_i}(\mathbf{x}) - \mathbf{s}_{\theta}(\mathbf{x}, \sigma_i)\|_2^2] \\ &= \frac{1}{L} \sum_{i=1}^L \lambda(\sigma_i) E_{\mathbf{x} \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0},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



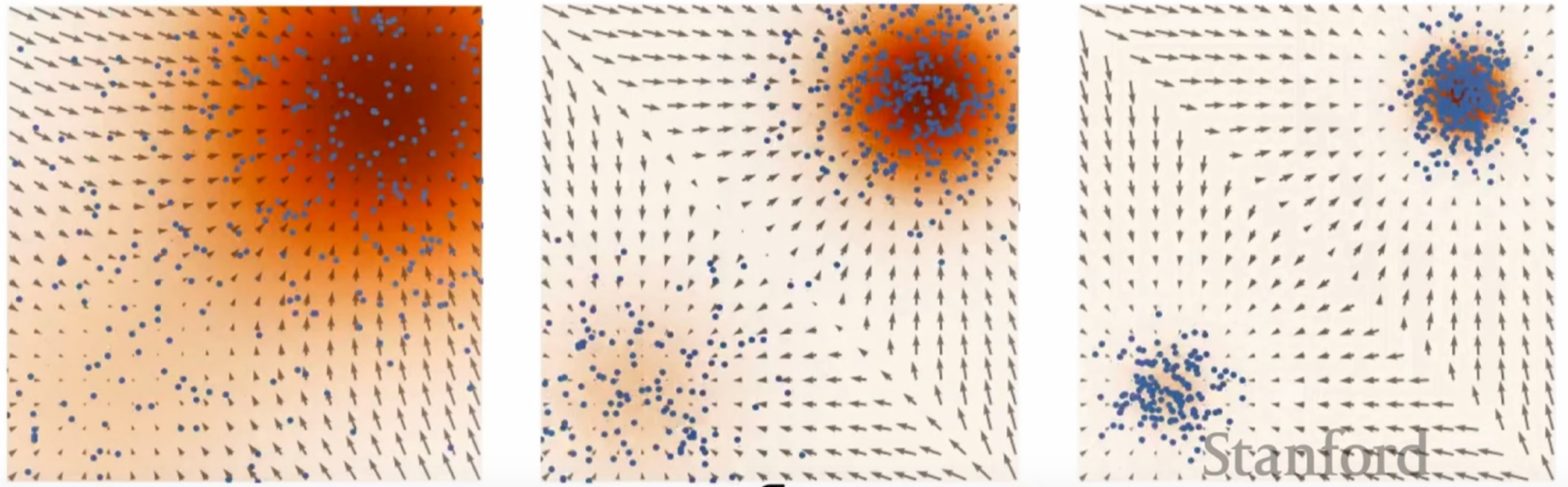


# Noise Conditional Score Networks

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

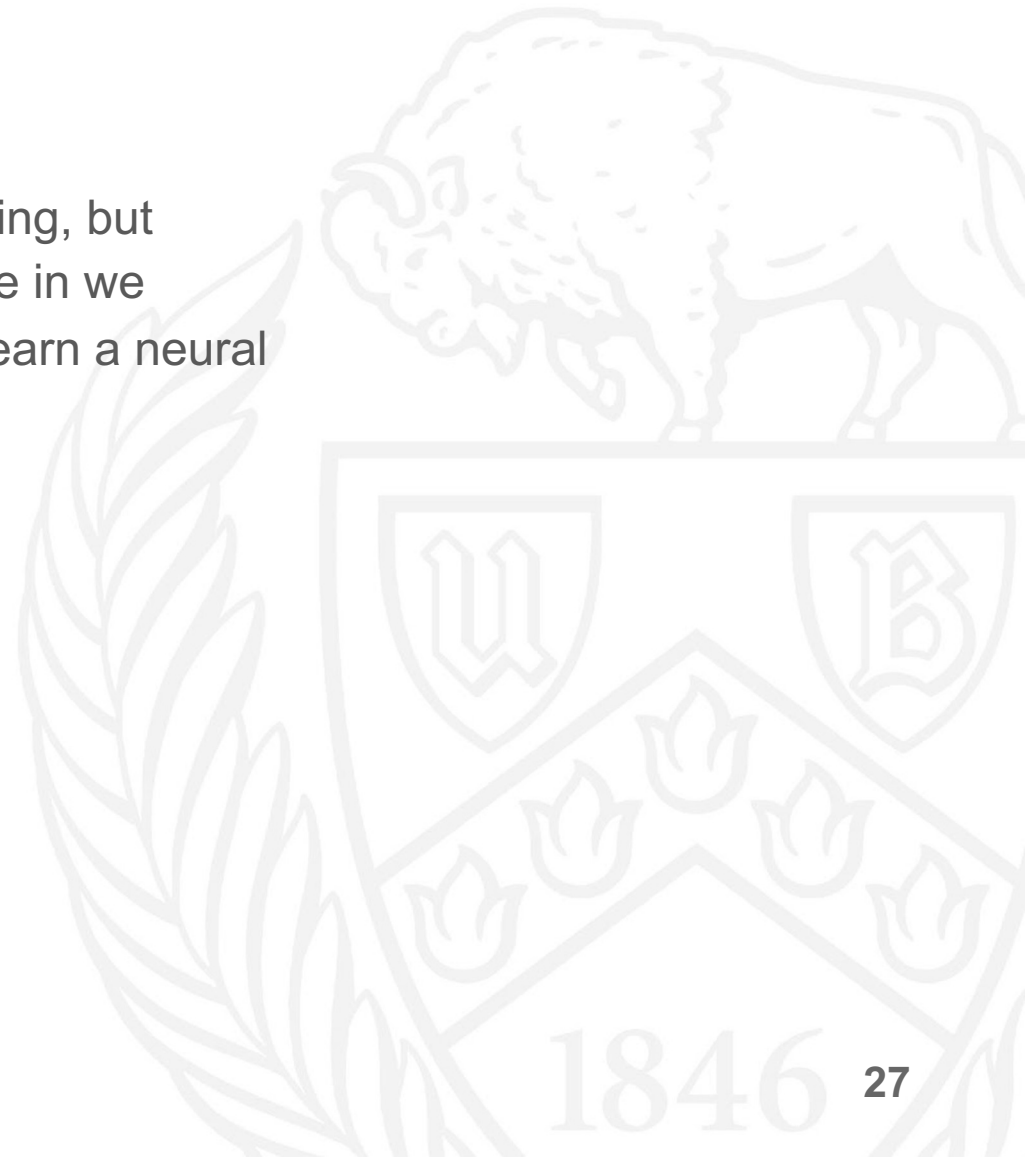


# Annealed Langevin Dynamics



# Next Up

Denoising Diffusion models. These models don't use score matching, but instead model the noising and denoising as a markov chain, where in we gradually add noise to an image until it becomes pure noise and learn a neural network that denoises the image.





# DENOISING DIFFUSION PROBABILISTIC MODEL (DDPM)



# Denoising Diffusion Probabilistic Model (DDPM)

## General Idea:

The essential idea is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data.

In short, we apply a lot of noise to the image and then have some intelligence which learns something out of it (a Neural Network) to remove the noise.

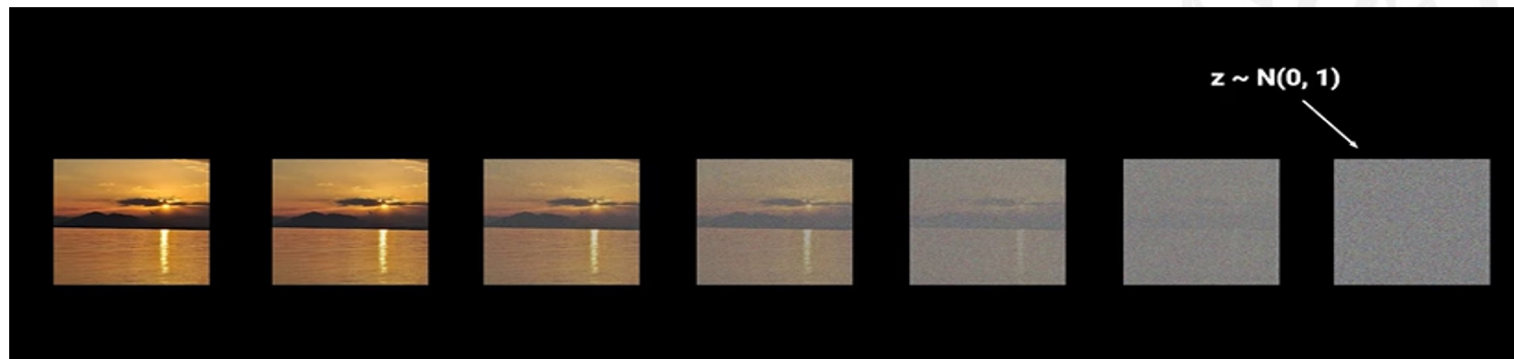
Now, the idea is if the Neural Network learns to remove the noise properly, we can start off with a completely random noise and let the model remove the noise until we have a new image which could occur from our training data.

# Understanding of DDPM

So as discussed in the previous slide, We have two main processes:

1. Forward Process: We start with the original image and step by step add more noise to the image. If we repeat this for sufficient number of steps, the image will become pure noise.  
How will be add the noise?

The paper uses Normal Distribution to sample the noise. The pure noise image will follow the Normal Distribution.



So, we can see that going from an image to noise is fairly very simple.

But wait, what about the reverse process where you go from just noise to an image? Here comes the actual problem..

## 2. Reverse Process:

It involves a Neural Network learning to remove the noise from an image step by step. This way we can give the Neural Network a random noisy image sampled from the Normal Distribution and let the model gradually remove the noise step by step until we have a clear looking image.

Ok, so why are we going step by step and not generate an image from the noisy image instantly?

Going from noisy image to a clear looking image instantly is NOT a tractable solution and can lead to much worse outcomes. You need Neural Network to learn something.. Isn't it?

Now the point goes to.. How does this Neural Network look like? What will be the input and output of this neural network?

First of all, let's see what this Neural Network can predict.

The Neural Network can predict:

1. Mean of the noise at each time step
2. Predicting the original image directly
3. Predicting the noise in the image directly

We have already seen that predicting the original image is not a tractable solution. So we can eliminate that.

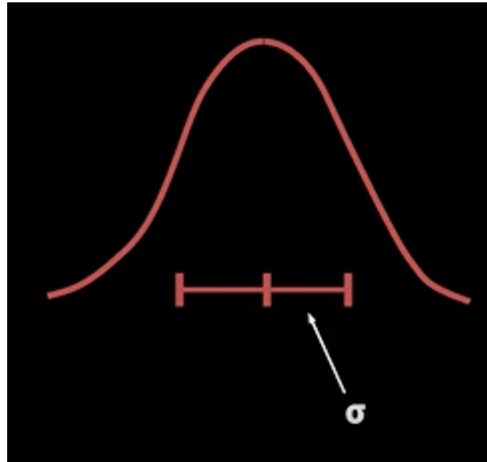
What would you select?



Well, both are mostly same. We just parameterize them differently. We will see this in further slides.

Most of the researchers chose third option i.e., Noise of the image and predict the noise directly.

Well, you all may be wondering why we are predicting the mean and not variance in the first option as the Normal Distribution needs mean and variance.



The authors of the paper decided to fix the variance and thus there is no need to predict it since it is always readily available. I will show you all in the Math Derivation part what they fixed it to.

Just as an improvement...

Can we think of learning the variance too because it may lead to the improvements in the log likelihood (loss of the Neural Network). We will see that later.

Note: We don't employ same amount of noise at each time step in the forward process. How do we regulate that?

**Schedule** - Scales the mean and the variance. This ensures that the variance doesn't explode as we add more and more noise.

This paper we are discussing employed **Linear Schedule**

Using linear schedule, the transformation looks something like this



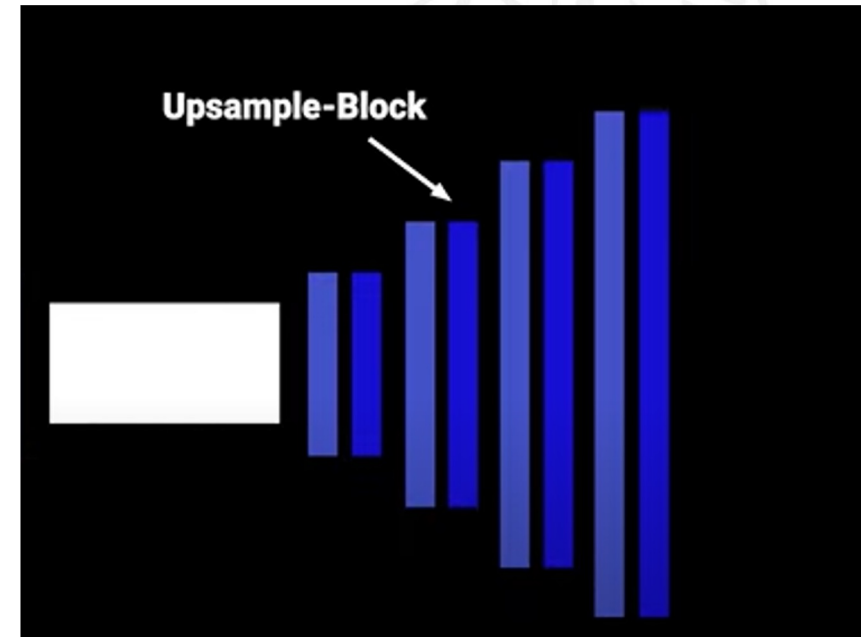
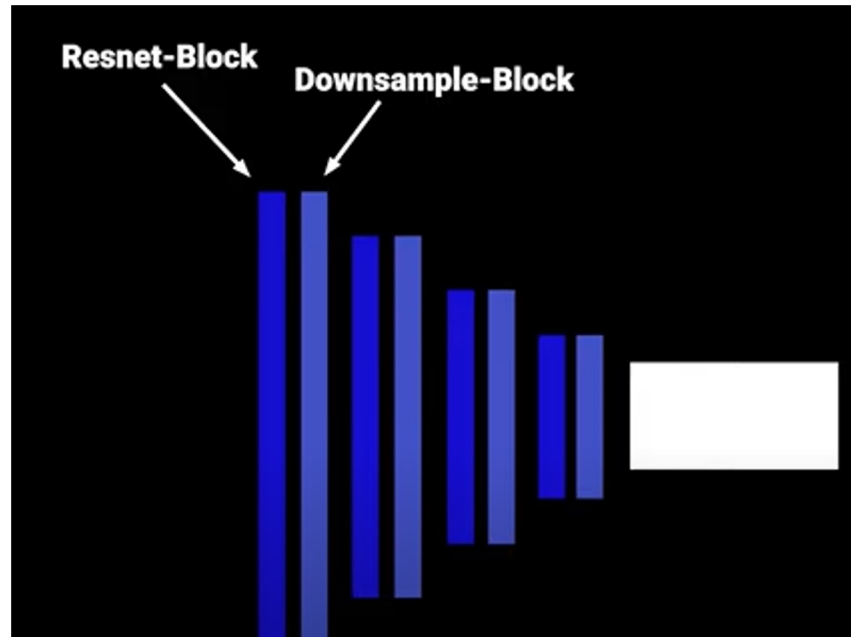
We will further discuss more about this and the scope of the improvement.

We are now clear what does the input and output of the Neural Network look like

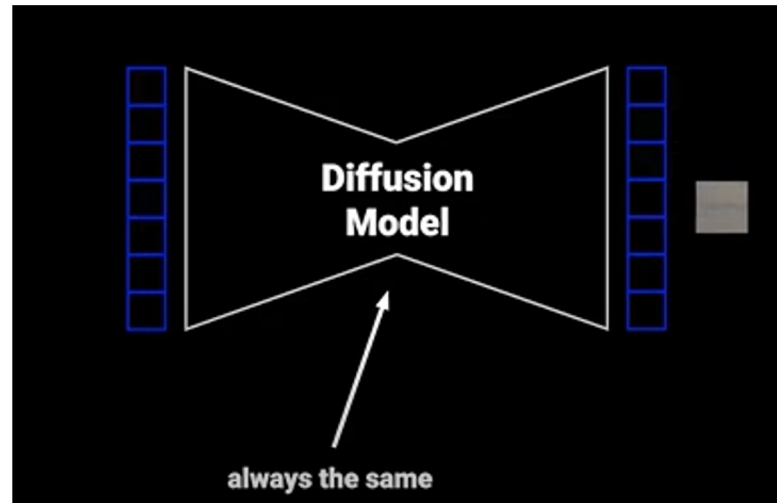
Let's have a look at the **Architecture**.

# Architecture of DDPM

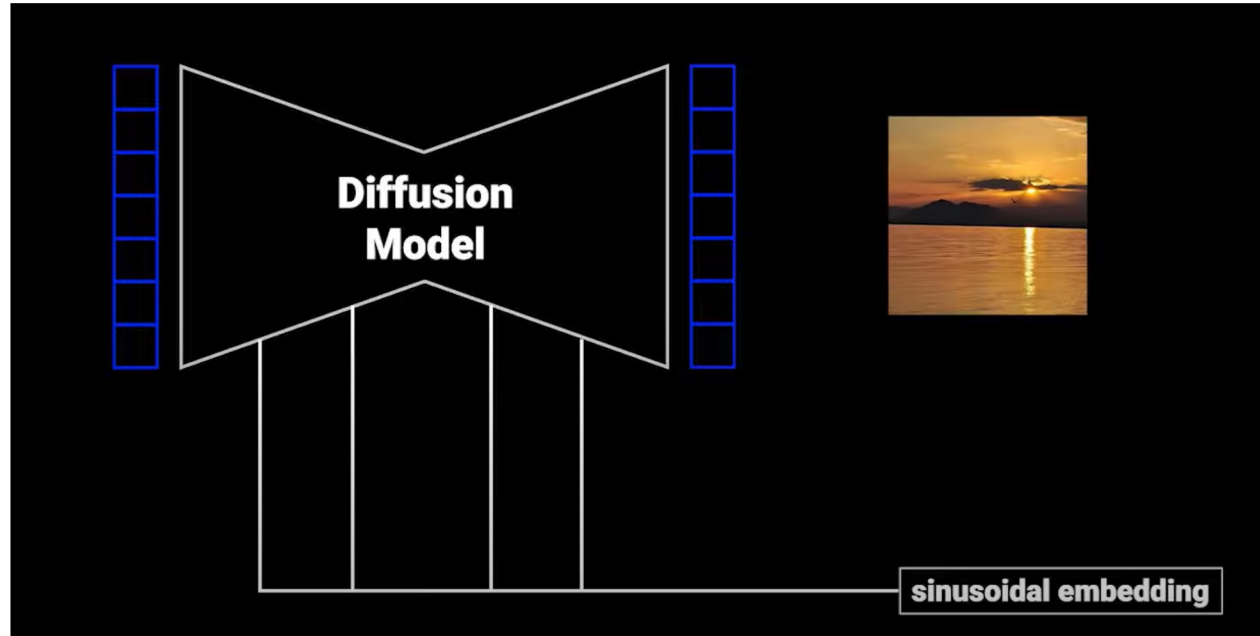
The Neural Network used for the reverse process looks like a **U-Net** Architecture.



The model will be same for each timestep.



The way of telling the model at what time step we are, is done using the sinusoidal embeddings that we learned from the **Transformer** paper.



This is passed to each residual block and it is very important because forward diffusion process is using a schedule (we discussed it a little on it which scales mean and variance). With this embedding the model can remove the noise at different levels in different amount of time intervals which effects the output a lot.

# NEXT UP

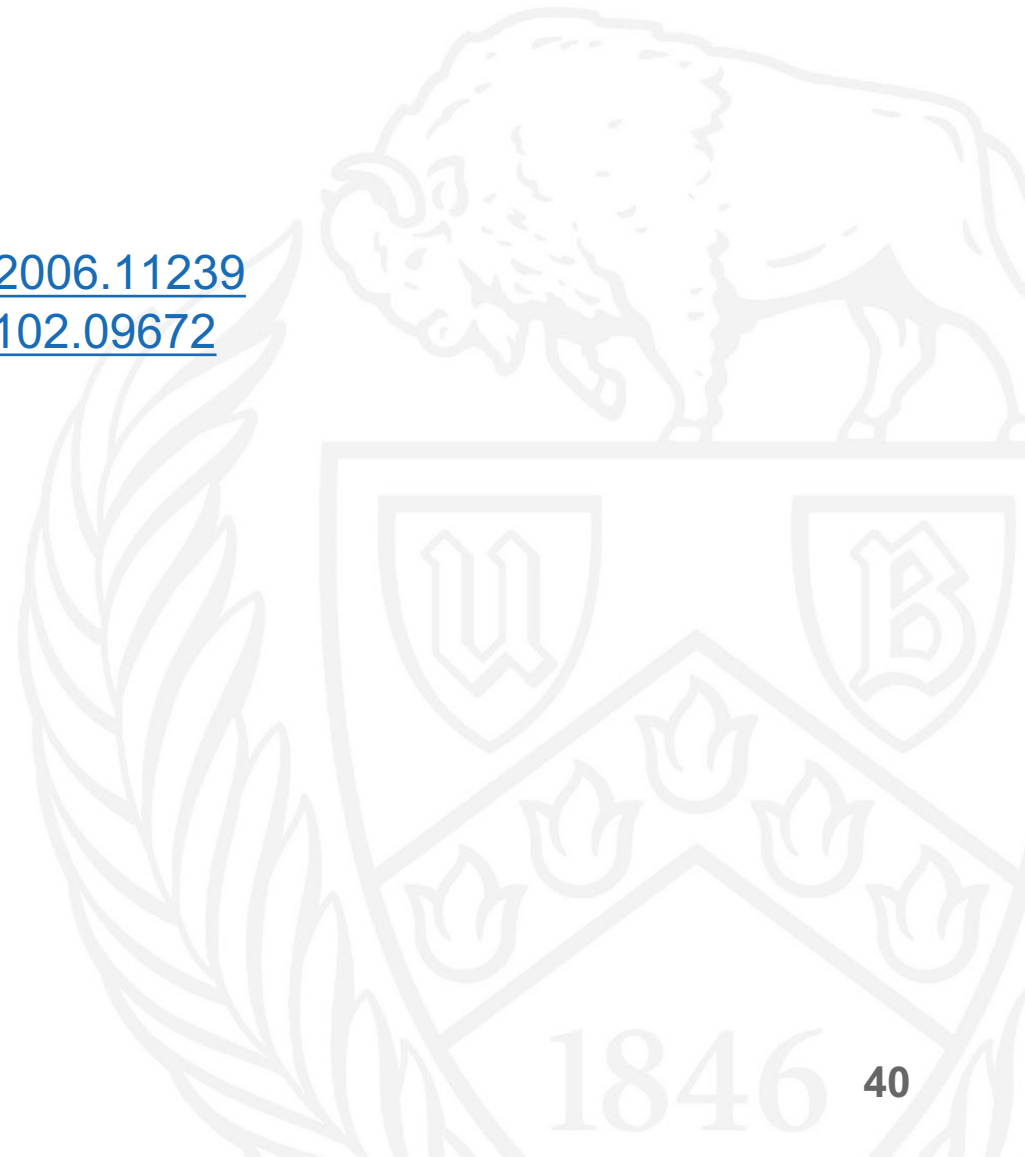
For the next presentation,

We will be going through the Math behind the working of DDPM extending some concepts we learned today and extend it to implement PyTorch based basic DDPM model.

We will be going through the improvements of the basic DDPM model like Latent Diffusion Models and some tunings to the basic DDPM along with other future scopes.

# PAPERS REFERRED

- Score Based Models - <https://arxiv.org/pdf/1907.05600v3>
- Denoising Diffusion Probabilistic Model - <https://arxiv.org/pdf/2006.11239>
- Some improvements done by OpenAI - <https://arxiv.org/pdf/2102.09672>
- Latent Diffusion Models - <https://arxiv.org/pdf/2112.10752>







**THANK YOU!**  
**SEE YOU NEXT WEEK!**

