Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Progress in generative models of images

Progress in generative models of text
Progress in generative models of text

(prompt) The best generative model is

Talktotransformer.com
Progress in generative models of text

Talktotransformer.com

(prompt) **The best generative model is** one that can learn over time and which predicts the structure and functionality of the brain as a whole.
Introduction to generative models
Introduction to generative models

\[ \mathbf{x}_i \overset{\text{i.i.d.}}{\sim} p_{\text{data}} \]

\[ i = 1, 2, \ldots, N \]
Introduction to generative models

\[ x_i \sim p_{data} \]
\[ i = 1, 2, \ldots, N \]
Introduction to generative models

\[ x_i \overset{i.i.d.}{\sim} p_{\text{data}} \quad i = 1, 2, \ldots, N \]

Models of Probability distributions

\[ \theta \in \Theta \]
Introduction to generative models

\[ \mathbf{x}_i \overset{i.i.d.}{\sim} p_{\text{data}} \]
\[ i = 1, 2, \ldots, N \]

$\theta \in \Theta$

Models of Probability distributions
Representations of probability distributions

**Implicit models**: represent the sampling process

Random Noise \( \epsilon \sim p(\epsilon) \)

Sample \( x = g(\epsilon) \)
Representations of probability distributions

**Implicit models**: represent the sampling process

Pros: flexible architecture, high sample quality.
Representations of probability distributions

**Implicit models**: represent the sampling process

Pros: flexible architecture, high sample quality.

Cons: hard to train, no likelihood, no principled model comparisons.
Representations of probability distributions

**Explicit models:** represent a probability density/mass function

- Bayesian networks (e.g., VAEs)
- MRF
- Autoregressive models
- Flow models
Representations of probability distributions

**Explicit models:** represent a probability density/mass function

- Bayesian networks (e.g., VAEs)
- MRF
- Autoregressive models
- Flow models

**Pros:** likelihoods
Representations of probability distributions

**Explicit models:** represent a probability density/mass function

- **Pros:** likelihoods
- **Cons:** need to be normalized, expressivity-tractability trade-off

Bayesian networks (e.g., VAEs)  MRF  Autoregressive models  Flow models
Representation of probability distributions

This talk: The gradient of log probability density
Representation of probability distributions

This talk: The gradient of log probability density

$$\nabla_x \log p(x)$$
Representation of probability distributions

**This talk:** The gradient of log probability density

\[ \nabla_x \log p(x) \quad \text{Score} \]
Representation of probability distributions

**This talk:** The gradient of log probability density

\[ \nabla_x \log p(x) \]

(PDF and score)
Why scores?
Why scores?

\[ p(x) \]
Why scores?

\[ p(x) \]

\[ \nabla_x \log p(x) \]
Why scores?

Energy-Based Model (EBM)

\[
p(x) = \frac{e^{-f_\theta(x)}}{Z_\theta}
\]
Why scores?

Energy-Based Model (EBM)

\[ p(x) = \frac{e^{-f_\theta(x)}}{Z_\theta} \]

\[ \nabla_x \log p(x) = -\nabla_x f_\theta(x) - \nabla_x \log Z_\theta \]

Score
Score estimation

• **Given:** i.i.d. samples \( \{x_1, x_2, \cdots, x_N\} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}(x) \)

• **Goal:** Estimating the score \( \nabla_x \log p_{\text{data}}(x) \)
Score estimation

- **Given:** i.i.d. samples \( \{x_1, x_2, \cdots, x_N\} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}(x) \)
- **Goal:** Estimating the score \( \nabla_x \log p_{\text{data}}(x) \)
- **Score Model:** A trainable vector-valued function \( s_{\theta}(x) : \mathbb{R}^D \to \mathbb{R}^D \)
Score estimation

- **Given:** i.i.d. samples \( \{x_1, x_2, \ldots, x_N\} \sim p_{\text{data}}(x) \)
- **Goal:** Estimating the score \( \nabla_x \log p_{\text{data}}(x) \)
- **Score Model:** A trainable vector-valued function \( s_\theta(x) : \mathbb{R}^D \to \mathbb{R}^D \)
- **Objective:** How to compare two vector fields of scores?

\[
\frac{1}{2} \mathbb{E}_{p_{\text{data}}(x)} \left[ \| \nabla_x \log p_{\text{data}}(x) - s_\theta(x) \|_2^2 \right]
\]

(Fisher divergence)
Score estimation

- **Given:** i.i.d. samples \( \{ \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N \} \overset{i.i.d.}{\sim} p_{\text{data}}(\mathbf{x}) 
- **Goal:** Estimating the score \( \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \)
- **Score Model:** A trainable vector-valued function \( s_{\theta}(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^D \)
- **Objective:** How to compare two vector fields of scores?
  \[
  \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[ \| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - s_{\theta}(\mathbf{x}) \|_2^2 \right]
  \]  
  (Fisher divergence)
Score estimation

- **Given:** i.i.d. samples \( \{x_1, x_2, \cdots, x_N\} \sim p_{\text{data}}(x) \)
- **Goal:** Estimating the score \( \nabla_x \log p_{\text{data}}(x) \)
- **Score Model:** A trainable vector-valued function \( s_\theta(x) : \mathbb{R}^D \rightarrow \mathbb{R}^D \)
- **Objective:** How to compare two vector fields of scores?
  \[
  \frac{1}{2} \mathbb{E}_{p_{\text{data}}(x)} \left[ \left\| \nabla_x \log p_{\text{data}}(x) - s_\theta(x) \right\|_2^2 \right]
  \]
  (Fisher divergence)

- Integration by parts \( \rightarrow \) **Score Matching** (Hyvärinen 2005)
  \[
  \mathbb{E}_{p_{\text{data}}(x)} \left[ \frac{1}{2} \left\| s_\theta(x) \right\|_2^2 + \text{trace} \left( \nabla_x s_\theta(x) \right) \right]
  \]
  (Jacobian of \( s_\theta(x) \))
Score estimation

• **Given:** i.i.d. samples \( \{x_1, x_2, \cdots, x_N\} \sim p_{data}(x) \)

• **Goal:** Estimating the score \( \nabla_x \log p_{data}(x) \)

• **Score Model:** A trainable vector-valued function \( s_{\theta}(x) : \mathbb{R}^D \rightarrow \mathbb{R}^D \)

• **Objective:** How to compare two vector fields of scores?

\[
\frac{1}{2} \mathbb{E}_{p_{data}(x)} \left[ \| \nabla_x \log p_{data}(x) - s_{\theta}(x) \|_2^2 \right]
\]

(Fisher divergence)

• Integration by parts \( \rightarrow \) **Score Matching** (Hyvärinen 2005)

\[
\mathbb{E}_{p_{data}(x)} \left[ \frac{1}{2} \| s_{\theta}(x) \|_2^2 + \text{trace} \left( \begin{array}{c} \nabla_x s_{\theta}(x) \\ \text{Jacobian of } s_{\theta}(x) \end{array} \right) \right]
\]

\[
\approx \frac{1}{N} \sum_{i=1}^{N} \left[ \frac{1}{2} \| s_{\theta}(x_i) \|_2^2 + \text{trace} \left( \nabla_x s_{\theta}(x_i) \right) \right]
\]
Score estimation

- **Given:** i.i.d. samples \( \{x_1, x_2, \cdots, x_N\} \sim p_{\text{data}}(x) \)
- **Goal:** Estimating the score \( \nabla_x \log p_{\text{data}}(x) \)
- **Score Model:** A trainable vector-valued function \( s_\theta(x) : \mathbb{R}^D \to \mathbb{R}^D \)
- **Objective:** How to compare two vector fields of scores?

\[
\frac{1}{2} \mathbb{E}_{p_{\text{data}}(x)} \left[ \left\| \nabla_x \log p_{\text{data}}(x) - s_\theta(x) \right\|^2 \right]
\]

(Fisher divergence)

- Integration by parts \( \rightarrow \) **Score Matching** (Hyvärinen 2005)

\[
\mathbb{E}_{p_{\text{data}}(x)} \left[ \frac{1}{2} \left\| s_\theta(x) \right\|^2 + \text{trace} \left( \nabla_x s_\theta(x) \right) \right]
\]

\[
\approx \frac{1}{N} \sum_{i=1}^{N} \left[ \frac{1}{2} \left\| s_\theta(x_i) \right\|^2 + \text{trace} \left( \nabla_x s_\theta(x_i) \right) \right]
\]

Not scalable
Score Matching is not scalable

- Deep score models

\[ x_1 \to s_{\theta,1} \to s_{\theta,2} \to s_{\theta,3} \to s_\theta(x) \]
Score Matching is not scalable

- Deep score models

\[ s_\theta(x) \]

- Compute \( \| s_\theta(x) \|_2^2 \) and \( \text{trace}(\nabla_x s_\theta(x)) \)
Score Matching is not scalable

- Deep score models

\[
\mathbf{x} \\
\mathbf{x}_1 \\
\mathbf{x}_2 \\
\mathbf{x}_3 \\
\mathbf{s}_\theta,1 \\
\mathbf{s}_\theta,2 \\
\mathbf{s}_\theta,3
\]

- Compute \( \|s_\theta(\mathbf{x})\|_2^2 \) and \( \text{trace}(\nabla_x s_\theta(\mathbf{x})) \)
Score Matching is not scalable

- Deep score models

\begin{align*}
    s_\theta(x) = & s_{\theta,1}(x) \\
    s_{\theta,2}(x) = & s_{\theta,1}(x) \\
    s_{\theta,3}(x) = & s_{\theta,1}(x)
\end{align*}

- Compute $\|s_\theta(x)\|_2^2$ and $\text{trace}(\nabla_x s_\theta(x))$
Score Matching is not scalable

- Deep score models

\[ s_\theta(x) \]

- Compute \( \|s_\theta(x)\|^2_2 \) and \( \text{trace}(\nabla_x s_\theta(x)) \)
Score Matching is not scalable

- Deep score models

\[
s_\theta(x)
\]

- Compute \( \|s_\theta(x)\|_2^2 \) and \( \text{trace}(\nabla_x s_\theta(x)) \)
Score Matching is not scalable

- Deep score models

\[ s_\theta(x) \]

- Compute \( \|s_\theta(x)\|_2^2 \) and \( \text{trace}(\nabla_x s_\theta(x)) \)
Score Matching is not scalable

• Deep score models

\[
\begin{align*}
\frac{\partial s_{\theta,1}(x)}{\partial x_1} & \\
\frac{\partial s_{\theta,2}(x)}{\partial x_2} & \\
\frac{\partial s_{\theta,3}(x)}{\partial x_3} & \\
\end{align*}
\]

• Compute \(||s_{\theta}(x)||_2^2\) and trace\(\nabla_x s_{\theta}(x)\)

\[
\nabla_x s_{\theta}(x) = 
\begin{pmatrix}
\frac{\partial s_{\theta,1}(x)}{\partial x_1} & \frac{\partial s_{\theta,1}(x)}{\partial x_2} & \frac{\partial s_{\theta,1}(x)}{\partial x_3} \\
\frac{\partial s_{\theta,2}(x)}{\partial x_1} & \frac{\partial s_{\theta,2}(x)}{\partial x_2} & \frac{\partial s_{\theta,2}(x)}{\partial x_3} \\
\frac{\partial s_{\theta,3}(x)}{\partial x_1} & \frac{\partial s_{\theta,3}(x)}{\partial x_2} & \frac{\partial s_{\theta,3}(x)}{\partial x_3}
\end{pmatrix}
\]
Score Matching is not scalable

- Deep score models

\[ \frac{\partial s_{\theta,1}(x)}{\partial x_1} \]

- Compute \( \|s_{\theta}(x)\|_2^2 \) and \( \text{trace}(\nabla_x s_{\theta}(x)) \)

\[ s_{\theta,2}(x) \]

\[ \nabla_x s_{\theta}(x) = \begin{pmatrix} \frac{\partial s_{\theta,1}(x)}{\partial x_1} & \frac{\partial s_{\theta,1}(x)}{\partial x_2} & \frac{\partial s_{\theta,1}(x)}{\partial x_3} \\ \frac{\partial s_{\theta,2}(x)}{\partial x_1} & \frac{\partial s_{\theta,2}(x)}{\partial x_2} & \frac{\partial s_{\theta,2}(x)}{\partial x_3} \\ \frac{\partial s_{\theta,3}(x)}{\partial x_1} & \frac{\partial s_{\theta,3}(x)}{\partial x_2} & \frac{\partial s_{\theta,3}(x)}{\partial x_3} \end{pmatrix} \]
Score Matching is not scalable

- Deep score models

- Compute $\|s_\theta(x)\|_2^2$ and $\text{trace}(\nabla_x s_\theta(x))$

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

$$\nabla_x s_\theta(x) = \begin{pmatrix}
\frac{\partial s_{\theta,1}(x)}{\partial x_1} & \frac{\partial s_{\theta,1}(x)}{\partial x_2} & \frac{\partial s_{\theta,1}(x)}{\partial x_3} \\
\frac{\partial s_{\theta,2}(x)}{\partial x_1} & \frac{\partial s_{\theta,2}(x)}{\partial x_2} & \frac{\partial s_{\theta,2}(x)}{\partial x_3} \\
\frac{\partial s_{\theta,3}(x)}{\partial x_1} & \frac{\partial s_{\theta,3}(x)}{\partial x_2} & \frac{\partial s_{\theta,3}(x)}{\partial x_3}
\end{pmatrix}$$
Score Matching is not scalable

- Deep score models

\[ \nabla_x s_\theta(x) = \begin{pmatrix}
\frac{\partial s_{\theta,1}(x)}{\partial x_1} & \frac{\partial s_{\theta,1}(x)}{\partial x_2} & \frac{\partial s_{\theta,1}(x)}{\partial x_3} \\
\frac{\partial s_{\theta,2}(x)}{\partial x_1} & \frac{\partial s_{\theta,2}(x)}{\partial x_2} & \frac{\partial s_{\theta,2}(x)}{\partial x_3} \\
\frac{\partial s_{\theta,3}(x)}{\partial x_1} & \frac{\partial s_{\theta,3}(x)}{\partial x_2} & \frac{\partial s_{\theta,3}(x)}{\partial x_3}
\end{pmatrix} \]
Score Matching is not scalable

- Deep score models

\[ s_{\theta}(x) \]

- Compute \( \|s_{\theta}(x)\|_2^2 \) and \( \text{trace}(\nabla_x s_{\theta}(x)) \)

\[
\begin{align*}
\frac{\partial s_{\theta,1}(x)}{\partial x_1} \\
\frac{\partial s_{\theta,2}(x)}{\partial x_2} \\
\frac{\partial s_{\theta,3}(x)}{\partial x_3}
\end{align*}
\]

\[
\nabla_x s_{\theta}(x) = \begin{pmatrix}
\frac{\partial s_{\theta,1}(x)}{\partial x_1} & \frac{\partial s_{\theta,1}(x)}{\partial x_2} & \frac{\partial s_{\theta,1}(x)}{\partial x_3} \\
\frac{\partial s_{\theta,2}(x)}{\partial x_1} & \frac{\partial s_{\theta,2}(x)}{\partial x_2} & \frac{\partial s_{\theta,2}(x)}{\partial x_3} \\
\frac{\partial s_{\theta,3}(x)}{\partial x_1} & \frac{\partial s_{\theta,3}(x)}{\partial x_2} & \frac{\partial s_{\theta,3}(x)}{\partial x_3}
\end{pmatrix}
\]

Stanford University
Score Matching is not scalable

- Deep score models

- Compute $\|s_\theta(x)\|_2^2$ and $\text{trace} (\nabla_x s_\theta(x))$

\[
\nabla_x s_\theta(x) = \begin{pmatrix}
\frac{\partial s_{\theta,1}(x)}{\partial x_1} & \frac{\partial s_{\theta,1}(x)}{\partial x_2} & \frac{\partial s_{\theta,1}(x)}{\partial x_3} \\
\frac{\partial s_{\theta,2}(x)}{\partial x_1} & \frac{\partial s_{\theta,2}(x)}{\partial x_2} & \frac{\partial s_{\theta,2}(x)}{\partial x_3} \\
\frac{\partial s_{\theta,3}(x)}{\partial x_1} & \frac{\partial s_{\theta,3}(x)}{\partial x_2} & \frac{\partial s_{\theta,3}(x)}{\partial x_3}
\end{pmatrix}
\]

$O(D)$ Backprops!

Stanford University
Score Matching is not scalable

- Deep score models

\[ s_{\theta}(x) \]

- Compute \( \|s_{\theta}(x)\|_2^2 \) and trace(\( \nabla_x s_{\theta}(x) \))

\[ \nabla_x s_{\theta}(x) = \begin{bmatrix} \frac{\partial s_{\theta,1}(x)}{\partial x_1} & \frac{\partial s_{\theta,1}(x)}{\partial x_2} & \frac{\partial s_{\theta,1}(x)}{\partial x_3} \\ \frac{\partial s_{\theta,2}(x)}{\partial x_1} & \frac{\partial s_{\theta,2}(x)}{\partial x_2} & \frac{\partial s_{\theta,2}(x)}{\partial x_3} \\ \frac{\partial s_{\theta,3}(x)}{\partial x_1} & \frac{\partial s_{\theta,3}(x)}{\partial x_2} & \frac{\partial s_{\theta,3}(x)}{\partial x_3} \end{bmatrix} \]

O(D) Backprops!

Stanford University
Score Matching is not scalable

- Deep score models

Compute $\|s_\theta(x)\|_2^2$ and $\text{trace} (\nabla_x s_\theta(x))$

\[
\nabla_x s_\theta(x) = \begin{pmatrix}
\frac{\partial s_{\theta,1}(x)}{\partial x_1} \\
\frac{\partial s_{\theta,2}(x)}{\partial x_2} \\
\frac{\partial s_{\theta,3}(x)}{\partial x_3}
\end{pmatrix}
\]

Score Matching is not Scalable!
Sliced score matching

• **Intuition:** one dimensional problems should be easier
Sliced score matching

• **Intuition:** one dimensional problems should be easier
• **Idea:** project onto random directions

Sliced score matching

• **Intuition**: one dimensional problems should be easier
• **Idea**: project onto random directions
• **Randomized objective**: Sliced Fisher Divergence

\[
\frac{1}{2} \mathbb{E}_{p_{\nu}} \mathbb{E}_{p_{\text{data}}(x)} [(v^T \nabla_x \log p_{\text{data}}(x) - v^T s_\theta(x))^2]
\]

Sliced score matching

- **Intuition:** one dimensional problems should be easier
- **Idea:** project onto random directions
- **Randomized objective:** Sliced Fisher Divergence

\[
\frac{1}{2} \mathbb{E}_{p \sim v} \mathbb{E}_{p_{\text{data}}(x)} \left[ (v^T \nabla_x \log p_{\text{data}}(x) - v^T s_{\theta}(x))^2 \right]
\]

- Integration by parts \(\rightarrow\) **Sliced Score Matching:**

\[
\mathbb{E}_{p \sim v} \mathbb{E}_{p_{\text{data}}(x)} \left[ v^T \nabla_x s_{\theta}(x) v + \frac{1}{2} (v^T s_{\theta}(x))^2 \right]
\]

Sliced score matching

• **Intuition:** one dimensional problems should be easier
• **Idea:** project onto random directions
• **Randomized objective:** Sliced Fisher Divergence

\[
\frac{1}{2} \mathbb{E}_{p_{\nu}} \mathbb{E}_{p_{\text{data}}}(x)[(v^T \nabla_x \log p_{\text{data}}(x) - v^T s_{\theta}(x))^2]
\]

• Integration by parts → **Sliced Score Matching:**

\[
\mathbb{E}_{p_{\nu}} \mathbb{E}_{p_{\text{data}}}(x) \left[ v^T \nabla_x s_{\theta}(x)v + \frac{1}{2}(v^T s_{\theta}(x))^2 \right]
\]

**Scalable!**

Computing Jacobian-vector products is scalable

\[ v^T \nabla_x s_\theta(x)v = v^T \nabla_x (v^T s_\theta(x)) \]
Computing Jacobian-vector products is scalable

\[ \mathbf{v}^T \nabla_x s_\theta(x) \mathbf{v} = \mathbf{v}^T \nabla_x (\mathbf{v}^T s_\theta(x)) \]

Computing Jacobian-vector products is scalable

\[ v^T \nabla_x s_\theta(x) v = v^T \nabla_x (v^T s_\theta(x)) \]
Computing Jacobian-vector products is scalable

$$v^T \nabla_x s_\theta(x) v = v^T \nabla_x (v^T s_\theta(x))$$

Computing Jacobian-vector products is scalable

\[ v^T \nabla_x s_\theta(x) v = v^T \nabla_x (v^T s_\theta(x)) \]

Computing Jacobian-vector products is scalable

\[ v^T \nabla_x s_\theta(x)v = v^T \nabla_x (v^Ts_\theta(x)) \]

Computing Jacobian-vector products is scalable

\[ \mathbf{v}^T \nabla_x s_\theta(x) \mathbf{v} = \mathbf{v}^T \nabla_x (\mathbf{v}^T s_\theta(x)) \]

Computing Jacobian-vector products is scalable

\[ \mathbf{v}^T \nabla_x s_\theta(\mathbf{x}) \mathbf{v} = \mathbf{v}^T \nabla_x (\mathbf{v}^T s_\theta(\mathbf{x})) \]

Computing Jacobian-vector products is scalable

\[ \mathbf{v}^\text{T} \nabla_x \mathbf{s}_\theta(\mathbf{x}) \mathbf{v} = \mathbf{v}^\text{T} \nabla_x (\mathbf{v}^\text{T} \mathbf{s}_\theta(\mathbf{x})) \]

One Backprop! Sliced Score Matching is scalable

---

Results: Sliced Score Matching for EBMs

Sliced score matching methods | Other Baselines | Score Matching

Efficiency

Results: Sliced Score Matching for EBMs

Efficiency

Results: Sliced Score Matching for EBMs

Results: Sliced Score Matching for EBMs

Efficiency

Performance on density estimation

Score-based generative modeling

Data samples

\[ \{x_1, x_2, \cdots, x_N\} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}(x) \]

Scores

\[ s_{\theta}(x) \approx \nabla_x \log p_{\text{data}}(x) \]
Score-based generative modeling

Data samples
\( \{x_1, x_2, \ldots, x_N\} \) i.i.d. \( p_{\text{data}}(x) \)

Scores
\( s_{\theta}(x) \approx \nabla_x \log p_{\text{data}}(x) \)

New samples

Stanford University
From scores to samples: Langevin MCMC

Scores

$s_\theta(x)$
From scores to samples: Langevin MCMC

\[ s_\theta(x) \]

Scores

\[ \tilde{x}_{t+1} \leftarrow \tilde{x}_t + \frac{\epsilon}{2} s_\theta(\tilde{x}_t) \]

Follow the scores
From scores to samples: Langevin MCMC

\[ s_\theta(x) \]

Scores

Follow the scores

\[ \tilde{x}_{t+1} \leftarrow \tilde{x}_t + \frac{\epsilon}{2} s_\theta(\tilde{x}_t) \]
From scores to samples: Langevin MCMC

Scores

Follow the scores

Follow noisy scores: Langevin MCMC

\[ \tilde{x}_{t+1} \leftarrow \tilde{x}_t + \frac{\epsilon}{2} s_\theta(\tilde{x}_t) \]

\[ z_t \sim \mathcal{N}(0, I) \]

\[ \tilde{x}_{t+1} \leftarrow \tilde{x}_t + \frac{\epsilon}{2} s_\theta(\tilde{x}_t) + \sqrt{\epsilon} z_t \]

Stanford University
From scores to samples: Langevin MCMC

Scores

Follow the scores

Follow noisy scores: Langevin MCMC

\[ s_\theta(x) \]

\[ \tilde{x}_{t+1} \leftarrow \tilde{x}_t + \frac{\epsilon}{2} s_\theta(\tilde{x}_t) \]

\[ z_t \sim \mathcal{N}(0, I) \]

\[ \tilde{x}_{t+1} \leftarrow \tilde{x}_t + \frac{\epsilon}{2} s_\theta(\tilde{x}_t) + \sqrt{\epsilon} z_t \]

Stanford University
Challenge in low data density regions

Challenge in low data density regions

**Data density**

**Data scores**

- Accurate

**Estimated scores**

- Accurate

Challenge in low data density regions

Data density

Data scores

Estimated scores

Accurate

Inaccurate

Challenge in low data density regions

\[ \frac{1}{2} \mathbb{E}_{p_{data}(x)} \left[ \| \nabla_x \log p_{data}(x) - s_\theta(x) \|_2^2 \right] \]

Challenge in low data density regions

\[ \frac{1}{2} \mathbb{E}_{p_{\text{data}}(x)} \left[ \| \nabla_x \log p_{\text{data}}(x) - s_\theta(x) \|^2 \right] \]

Challenge in low data density regions

\[ \frac{1}{2} \mathbb{E}_{p_{\text{data}}(x)} \left[ \| \nabla_x \log p_{\text{data}}(x) - s_\theta(x) \|_2^2 \right] \]

Langevin MCMC will have trouble exploring low density regions

Adding noise to data

Adding noise to data

Adding noise to data

**Perturbed density**

**Perturbed scores**

**Estimated scores**

Provide useful directional information for Langevin MCMC.

Using multiple noise scales
Using multiple noise scales

High data density region
Using multiple noise scales
Using multiple noise scales

\[ \sigma_1 < \sigma_2 < \sigma_3 \]

Data

Using multiple noise scales

Data

$\sigma_1 < \sigma_2 < \sigma_3$

Noise Conditional Score Networks (NCSN)

Using multiple noise scales

\[ \sum_{i=1}^{N} \lambda(\sigma_i) \mathbb{E}_{p_{\sigma_i}(x)} [ \| \nabla_x \log p_{\sigma_i}(x) - s_\theta(x, \sigma_i) \|_2^2 ] \]

Using multiple noise scales

\( \sigma_1 \)  \( < \)  \( \sigma_2 \)  \( < \)  \( \sigma_3 \)

Data

\( s_{\theta,1} \)  \( s_{\theta,2} \)  \( \vdots \)

\( x_1 \)  \( x_2 \)  \( \sigma \)

Noise Conditional Score Networks (NCSN)

Using multiple noise scales

Using multiple noise scales

$\sigma_1 < \sigma_2 < \sigma_3$

Data

$\mathbf{x}_1, \mathbf{x}_2, \sigma$

$\mathbf{s}_{\theta,1}, \mathbf{s}_{\theta,2}, \ldots$

Noise Conditional Score Networks (NCSN)

Using multiple noise scales

\[
\sigma_1 < \sigma_2 < \sigma_3
\]

Data

Noise Conditional Score Networks (NCSN)

Using multiple noise scales

Data

Noise Conditional Score Networks (NCSN)

Using multiple noise scales

\[ \sigma_1 < \sigma_2 < \sigma_3 \]

Data

\( s_{\theta,1} \quad s_{\theta,2} \quad \ldots \quad x_1 \quad x_2 \quad \sigma \)

Noise Conditional Score Networks (NCSN)


Stanford University
Using multiple noise scales

Data Noise Conditional Score Networks (NCSN)

\[ \sigma_1 \prec \sigma_2 \prec \sigma_3 \]

Using multiple noise scales

Data

Noise Conditional Score Networks (NCSN)

Using multiple noise scales

\[
\sigma_1 < \sigma_2 < \sigma_3
\]

Data

\[s_{\theta,1}, s_{\theta,2}, \ldots, x_1, x_2, \sigma\]

Noise Conditional Score Networks (NCSN)

Annealed Langevin dynamics

Sampling in the real world

High resolution image generation

Using an infinite number of noise scales
Using an infinite number of noise scales
Using an infinite number of noise scales
Using an infinite number of noise scales
Using an infinite number of noise scales
Using an infinite number of noise scales
Using an infinite number of noise scales

$t \in [0, T] : \text{continuous index of perturbed distributions}$
Compact representation of infinite distributions
Compact representation of infinite distributions

- Stochastic process \( \{x(t)\}_{t=0}^{T} \) → Marginal probability densities \( \{p_t(x)\}_{t=0}^{T} \)
Compact representation of infinite distributions

- Stochastic process \( \{x(t)\}_{t=0}^{T} \) \(\rightarrow\) Marginal probability densities \( \{p_t(x)\}_{t=0}^{T} \)

- Stochastic differential equation: \( \, \text{d}x = f(x, t)\, \text{d}t + \sigma(t)\, \text{d}w \)
Compact representation of infinite distributions

- Stochastic process \( \{x(t)\}_t^{T} \) \( \rightarrow \) Marginal probability densities \( \{p_t(x)\}_t^{T} \)

- Stochastic differential equation: \( \text{d}x = \begin{bmatrix} f(x, t) \text{d}t \\ \sigma(t) \text{d}w \end{bmatrix} \)

  Deterministic drift
Compact representation of infinite distributions

- Stochastic process \( \{x(t)\}_{t=0}^{T} \) \( \rightarrow \) Marginal probability densities \( \{p_t(x)\}_{t=0}^{T} \)

- Stochastic differential equation: \( dx = \begin{cases} f(x, t) dt + \sigma(t) dw \end{cases} \)

Deterministic drift

Infinitesimal white noise

Stanford University
Score-based generative modeling via SDEs
Score-based generative modeling via SDEs
Score-based generative modeling via SDEs

\[ \frac{dx}{dt} = \sigma(t)dw \]
Score-based generative modeling via SDEs

\[ \text{dx} = \sigma(t)\text{dw} \]
Score-based generative modeling via SDEs
Score-based generative modeling via SDEs

Prior $p_T(x)$

Perturbed distributions

Data $p_0(x)$

Reverse-time SDE trajectories
Score-based generative modeling via SDEs

\[ \text{dx} = \sigma(t) \text{d}w \]

Time reversal

\[ \{ p_t(x) \}_{t=0}^T \]

\[ \text{dx} = -\sigma^2(t) \nabla_x \log p_t(x) \text{d}t + \sigma(t) \text{d}\tilde{w} \]
Score-based generative modeling via SDEs

\[ \text{dx} = \sigma(t) \text{d}w \]

Time reversal
\[
\{p_t(x)\}_{t=0}^{T}
\]

\[ \text{dx} = -\sigma^2(t) \nabla_x \log p_t(x) \text{d}t + \sigma(t) \text{d}\tilde{w} \]

Score function!
Score-based generative modeling via SDEs

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

- Time-dependent score-based model

\[ s_\theta(x, t) \approx \nabla_x \log p_t(x) \]
Score-based generative modeling via SDEs

- Time-dependent score-based model
  \[ s_\theta(x, t) \approx \nabla_x \log p_t(x) \]

- Training:
  \[ \mathbb{E}_{t \in \mathcal{U}(0,T)}[\lambda(t) \mathbb{E}_{p_t(x)}[\| \nabla_x \log p_t(x) - s_\theta(x, t) \|^2_2]] \]

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

• Time-dependent score-based model
  \[ s_\theta(x, t) \approx \nabla_x \log p_t(x) \]

• Training:
  \[ \mathbb{E}_{t \in \mathcal{U}(0,T)}[\lambda(t) \mathbb{E}_{p_t(x)}[\| \nabla_x \log p_t(x) - s_\theta(x, t) \|^2_2]] \]

• Reverse-time SDE
  \[ dx = -\sigma^2(t)s_\theta(x, t)dt + \sigma(t)d\tilde{w} \]
Score-based generative modeling via SDEs

- Time-dependent score-based model
  \[ s_\theta(x, t) \approx \nabla_x \log p_t(x) \]

- Training:
  \[
  \mathbb{E}_{t \in \mathcal{U}(0,T)}[\lambda(t)\mathbb{E}_{p_t(x)}[\| \nabla_x \log p_t(x) - s_\theta(x, t) \|^2_2]]
  \]

- Reverse-time SDE
  \[
  dx = -\sigma^2(t)s_\theta(x, t)dt + \sigma(t)d\bar{w}
  \]

- Euler-Maruyama (analogue to Euler for ODEs)
  \[
  x \leftarrow x - \sigma(t)^2 s_\theta(x, t)\Delta t + \sigma(t) z \quad (z \sim \mathcal{N}(0, |\Delta t| I))
  \]
  \[
  t \leftarrow t + \Delta t
  \]

*Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. "Score-Based Generative Modeling through Stochastic Differential Equations." ICLR 2021.*
Mixture of score matching subsumes MLE

\[ \mathbb{E}_{t \in U(0,T)}[\lambda(t) \mathbb{E}_{p_t(x)}[\|\nabla_x \log p_t(x) - \nabla_x \log q_{t,\theta}(x, t)\|^2]] \]

\[ = \frac{2}{T} \text{KL}(p_0(x) \| q_{0,\theta}(x)) \]

For certain choice of the weighting function \( \lambda(t) \)

Durkan* and Song*, “On Maximum Likelihood Training of Score-Based Generative Models”, arXiv 2101.09258
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.*
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.*
Predictor-Corrector sampling methods

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

**Figure:**
- Prior
- Perturbed distributions
  - Reverse-time SDE trajectories
- Data
  - $p_T(x)$
  - $p_t(x)$
  - $p_0(x)$

**Reference:**
Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.
Predictor-Corrector sampling methods

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

---

Predictor-Corrector sampling methods

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

**Figure:**

- Prior
- Perturbed distributions
- Data
  - Red line: Reverse-time SDE trajectories

**Equations:**

- \( p_T(x) \)
- \( p_t(x) \)
- \( p_0(x) \)

**References:**

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

<table>
<thead>
<tr>
<th>Model</th>
<th>FID $\downarrow$</th>
<th>IS $\uparrow$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Conditional</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BigGAN (Brock et al., 2018)</td>
<td>14.73</td>
<td>9.22</td>
</tr>
<tr>
<td>StyleGAN2-ADA (Karras et al., 2020a)</td>
<td><strong>2.42</strong></td>
<td><strong>10.14</strong></td>
</tr>
<tr>
<td><strong>Unconditional</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StyleGAN2-ADA (Karras et al., 2020a)</td>
<td>2.92</td>
<td>9.83</td>
</tr>
<tr>
<td>NCSN (Song &amp; Ermon, 2019)</td>
<td>25.32</td>
<td>8.87 ± .12</td>
</tr>
<tr>
<td>NCSNv2 (Song &amp; Ermon, 2020)</td>
<td>10.87</td>
<td>8.40 ± .07</td>
</tr>
<tr>
<td>DDPM (Ho et al., 2020)</td>
<td>3.17</td>
<td>9.46 ± .11</td>
</tr>
<tr>
<td>DDPM++</td>
<td>2.78</td>
<td>9.64</td>
</tr>
<tr>
<td>DDPM++ cont. (VP)</td>
<td>2.55</td>
<td>9.58</td>
</tr>
<tr>
<td>DDPM++ cont. (sub-VP)</td>
<td>2.61</td>
<td>9.56</td>
</tr>
<tr>
<td>DDPM++ cont. (deep, VP)</td>
<td>2.41</td>
<td>9.68</td>
</tr>
<tr>
<td>DDPM++ cont. (deep, sub-VP)</td>
<td>2.41</td>
<td>9.57</td>
</tr>
<tr>
<td>NCSN++</td>
<td>2.45</td>
<td>9.73</td>
</tr>
<tr>
<td>NCSN++ cont. (VE)</td>
<td>2.38</td>
<td>9.83</td>
</tr>
<tr>
<td>NCSN++ cont. (deep, VE)</td>
<td><strong>2.20</strong></td>
<td><strong>9.89</strong></td>
</tr>
</tbody>
</table>

*Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.*
High-Fidelity Generation for 1024x1024 Images

Probability flow ODE: turning the SDE to ODE

• Probability flow ODE (ordinary differential equation)
Probability flow ODE: turning the SDE to ODE

- Probability flow ODE (ordinary differential equation)

\[ dx = \sigma(t)dw \]
Probability flow ODE: turning the SDE to ODE

- Probability flow ODE (ordinary differential equation)

\[
\begin{align*}
\frac{dx}{dt} &= \sigma(t) dw \\
\frac{dx}{dt} &= -\frac{1}{2} \sigma(t)^2 \nabla_x \log p_t(x) dt
\end{align*}
\]
Probability flow ODE: turning the SDE to ODE

- Probability flow ODE (ordinary differential equation)

\[
\frac{dx}{dt} = \sigma(t)dw
\]

\[
\{p_t(x)\}_{t=0}^{T}
\]

\[
\sigma(t) = -\frac{1}{2} \sigma(t)^2 \nabla_x \log p_t(x) dt
\]

Score function

\[
\approx s_\theta(x, t)
\]
Probability flow ODE: turning the SDE to ODE

- Probability flow ODE (ordinary differential equation)

\[
\frac{dx}{dt} = \sigma(t)dw \quad \Rightarrow \quad \frac{dx}{dt} = -\frac{1}{2}\sigma(t)^2 \nabla_x \log p_t(x) dt
\]

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

Score function
\(\approx s_\theta(x, t)\)

---

**Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole.** “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.
Probability flow ODE: turning the SDE to ODE

- Probability flow ODE (ordinary differential equation)

\[ \frac{dx}{dt} = \sigma(t) dw \]

\[ \frac{dx}{dt} = -\frac{1}{2} \sigma(t)^2 \nabla_x \log p_t(x) dt \]

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

Score function

\( \sim s_\theta(x, t) \)

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.
Probability flow ODE: turning the SDE to ODE

- Probability flow ODE (ordinary differential equation)

\[
dx = \sigma(t) dw
\]

\[
\{ p_t(x) \}_{t=0}^T
\]

\[
dx = -\frac{1}{2} \sigma(t)^2 \nabla_x \log p_t(x) dt
\]

---

**Song**, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.
Solving Reverse-ODE for Sampling

• More efficient samplers via black-box ODE solvers

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

• **More efficient samplers** via black-box ODE solvers

![Image with NFE values: NFE=14, NFE=86, NFE=548]

NFE = Number of score Function Evaluations

• Predictor-corrector:
  • > 1000 NFE
• ODE
  • ≈ 100 NFE

*Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.*
Solving Reverse-ODE for Sampling

- **More efficient samplers** via black-box ODE solvers

  - **Exact likelihood** though models are trained with score matching.

  \[
  \log p_0(x) = \log p_T(x) - \frac{1}{2} \int_0^T \sigma(t)^2 \text{trace}(\nabla_x s_{\theta}(x, t)) dt
  \]

- NFE = Number of score Function Evaluations
  - Predictor-corrector:
    - > 1000 NFE
    - ODE
    - \( \approx 100 \) NFE

**Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole.** “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.
Solving Reverse-ODE for Sampling

<table>
<thead>
<tr>
<th>Model</th>
<th>NLL Test ↓</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealNVP (Dinh et al., 2016)</td>
<td>3.49</td>
<td>-</td>
</tr>
<tr>
<td>iResNet (Behrmann et al., 2019)</td>
<td>3.45</td>
<td>-</td>
</tr>
<tr>
<td>Glow (Kingma &amp; Dhariwal, 2018)</td>
<td>3.35</td>
<td>-</td>
</tr>
<tr>
<td>MintNet (Song et al., 2019b)</td>
<td>3.32</td>
<td>-</td>
</tr>
<tr>
<td>Residual Flow (Chen et al., 2019)</td>
<td>3.28</td>
<td>46.37</td>
</tr>
<tr>
<td>FFJORD (Grathwohl et al., 2018)</td>
<td>3.40</td>
<td>-</td>
</tr>
<tr>
<td>Flow++ (Ho et al., 2019)</td>
<td>3.29</td>
<td>-</td>
</tr>
<tr>
<td>DDPM (L) (Ho et al., 2020)</td>
<td>$\leq 3.70^*$</td>
<td>13.51</td>
</tr>
<tr>
<td>DDPM (L_{simple}) (Ho et al., 2020)</td>
<td>$\leq 3.75^*$</td>
<td>3.17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>NLL Test ↓</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDPM</td>
<td>3.28</td>
<td>3.37</td>
</tr>
<tr>
<td>DDPM cont. (VP)</td>
<td>3.21</td>
<td>3.69</td>
</tr>
<tr>
<td>DDPM cont. (sub-VP)</td>
<td>3.05</td>
<td>3.56</td>
</tr>
<tr>
<td>DDPM++ cont. (VP)</td>
<td>3.16</td>
<td>3.93</td>
</tr>
<tr>
<td>DDPM++ cont. (sub-VP)</td>
<td>3.02</td>
<td>3.16</td>
</tr>
<tr>
<td>DDPM++ cont. (deep, VP)</td>
<td>3.13</td>
<td>3.08</td>
</tr>
<tr>
<td>DDPM++ cont. (deep, sub-VP)</td>
<td>2.99</td>
<td>2.92</td>
</tr>
</tbody>
</table>

models trained with score matching
black-box ODE Solvers for sampling

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.
Probability flow ODE: latent space manipulation

Interpolation

Temperature scaling

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.
Probability flow ODE: uniquely identifiable encoding

- Uniquely **identifiable** encoding

Flow models, VAE, etc

Score-based models via probability flow ODE

---

**Song**, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.
Probability flow ODE: uniquely identifiable encoding

- Uniquely **identifiable** encoding

\[
\mathbf{dx} = -\frac{1}{2} \sigma(t)^2 \nabla_x \log p_t(x) dt
\]

- No trainable parameters in the probability flow ODE!

**Flow models, VAE, etc**

**Score-based models via probability flow ODE**

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.
Probability flow ODE: uniquely identifiable encoding

- Uniquely **identifiable** encoding

---

**Model 1**

**Model 2**

---

Flow models, VAE, etc

Score-based models via probability flow ODE

---

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

\[ X_0 \rightarrow X_t \rightarrow X_T \]

\[ t \in (0, T) \]

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

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

- Conditional reverse-time SDE via unconditional scores

\[
dx = -\sigma^2(t) \nabla_x \log p_t(x \mid y) dt + \sigma(t) d\bar{w}
\]

Controllable Generation

- Conditional reverse-time SDE via unconditional scores

\[ \text{dx} = -\sigma^2(t) \nabla_x \log p_t(x \mid y) dt + \sigma(t) d\bar{w} \]
Controllable Generation

- Conditional reverse-time SDE via unconditional scores

\[ \begin{align*}
    dx &= -\sigma^2(t) \nabla_x \log p_t(x \mid y) dt + \sigma(t) d\tilde{w} \\
    dx &= -\sigma^2(t) \left[ \nabla_x \log p_t(x) + \nabla_x \log p_t(y \mid x) \right] dt + \sigma(t) d\tilde{w}
\end{align*} \]

Controllable Generation

- Conditional reverse-time SDE via unconditional scores

\[
\text{dx} = -\sigma^2(t) \nabla_x \log p_t(x \mid y) \text{dt} + \sigma(t) \text{d}\tilde{w}
\]

\[
\text{dx} = -\sigma^2(t) \left[ \nabla_x \log p_t(x) + \nabla_x \log p_t(y \mid x) \right] \text{dt} + \sigma(t) \text{d}\tilde{w}
\]

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

- Conditional reverse-time SDE via unconditional scores

\[ dx = -\sigma^2(t) \left[ \nabla_x \log p_t(x \mid y) \right] dt + \sigma(t)d\tilde{w} \]

\[ dx = -\sigma^2(t) \left[ \nabla_x \log p_t(x) \right] dt + \sigma(t)d\tilde{w} \]

\[ dx = -\sigma^2(t) \left[ \nabla_x \log p_t(y \mid x) \right] dt + \sigma(t)d\tilde{w} \]

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.
Controllable Generation: class-conditional generation

- $y$ is the **class label**
- $p_t(y \mid x)$ 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

- $y$ is the masked image
- $p_t(y \mid x)$ can be approximated without training
Controllable Generation: colorization

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

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

- Discrete data, such as text generation
- Theoretical understanding on sample quality
- Faster sampling

Improvements

- Semi-supervised learning
- Inverse problems
- Unrestricted adversarial attacks
- Outlier detection

Applications
Conclusion
Conclusion

• Gradients of distributions (scores) can be estimated easily
Conclusion

• Gradients of distributions (scores) can be estimated easily
• Flexible architecture choices — no need to be normalized/invertible
Conclusion

- **Gradients of distributions (scores) can be estimated easily**
  - **Flexible architecture choices** — no need to be normalized/invertible
  - **Stable training** — no minimax optimization
Conclusion

• Gradients of distributions (scores) can be estimated easily
  • Flexible architecture choices — no need to be normalized/invertible
  • Stable training — no minimax optimization

• Better or comparable sample quality to GANs
Conclusion

• Gradients of distributions (scores) can be estimated easily
  • Flexible architecture choices — no need to be normalized/invertible
  • Stable training — no minimax optimization

• Better or comparable sample quality to GANs
  • State-of-the-art performance on CIFAR-10 and others
Conclusion

• Gradients of distributions (scores) can be estimated easily
  • **Flexible architecture choices** — no need to be normalized/invertible
  • **Stable training** — no minimax optimization

• Better or comparable sample quality to GANs
  • State-of-the-art performance on CIFAR-10 and others
  • Scalable to resolution of 1024x1024 for image generation
Conclusion

- Gradients of distributions (scores) can be estimated easily
  - Flexible architecture choices — no need to be normalized/invertible
  - Stable training — no minimax optimization

- Better or comparable sample quality to GANs
  - State-of-the-art performance on CIFAR-10 and others
  - Scalable to resolution of 1024x1024 for image generation

- Exact likelihood computation
Conclusion

• Gradients of distributions (scores) can be estimated easily
  • Flexible architecture choices — no need to be normalized/invertible
  • Stable training — no minimax optimization

• Better or comparable sample quality to GANs
  • State-of-the-art performance on CIFAR-10 and others
  • Scalable to resolution of 1024x1024 for image generation

• Exact likelihood computation
  • Competitive likelihood on CIFAR-10
Conclusion

• Gradients of distributions (scores) can be estimated easily
  • Flexible architecture choices — no need to be normalized/invertible
  • Stable training — no minimax optimization

• Better or comparable sample quality to GANs
  • State-of-the-art performance on CIFAR-10 and others
  • Scalable to resolution of 1024x1024 for image generation

• Exact likelihood computation
  • Competitive likelihood on CIFAR-10
  • Equivalence to Neural ODEs, plus uniquely identifiable encoding
Collaborators

Stefano Ermon
Ben Poole
Jascha Sohl-Dickstein
Durk Kingma

Sahaj Garg
Jiaxin Shi
Abhishek Kumar