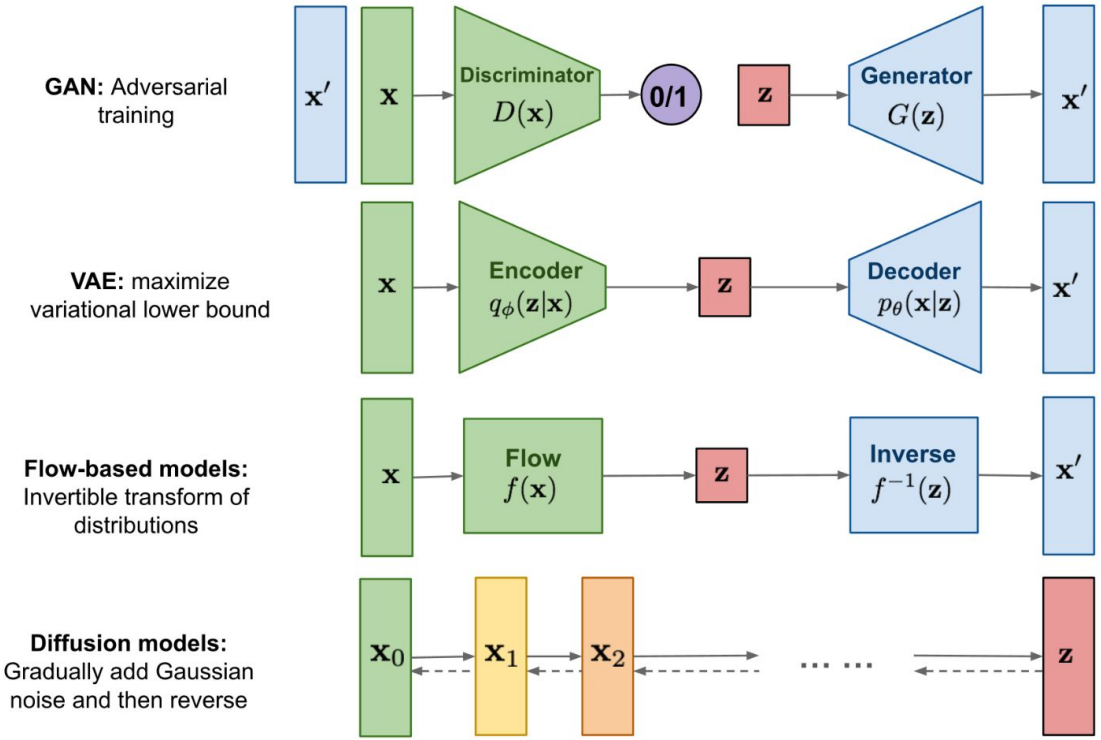
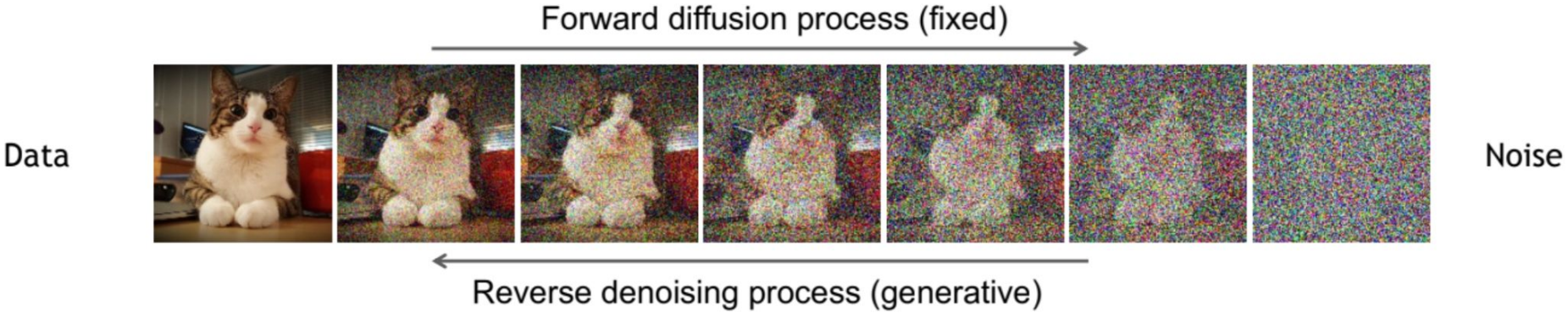
Diffusion Models for Audio Generation

Aibek Alanov,
Research scientist at AIRI and at Centre of Deep Learning and Bayesian Methods HSE University

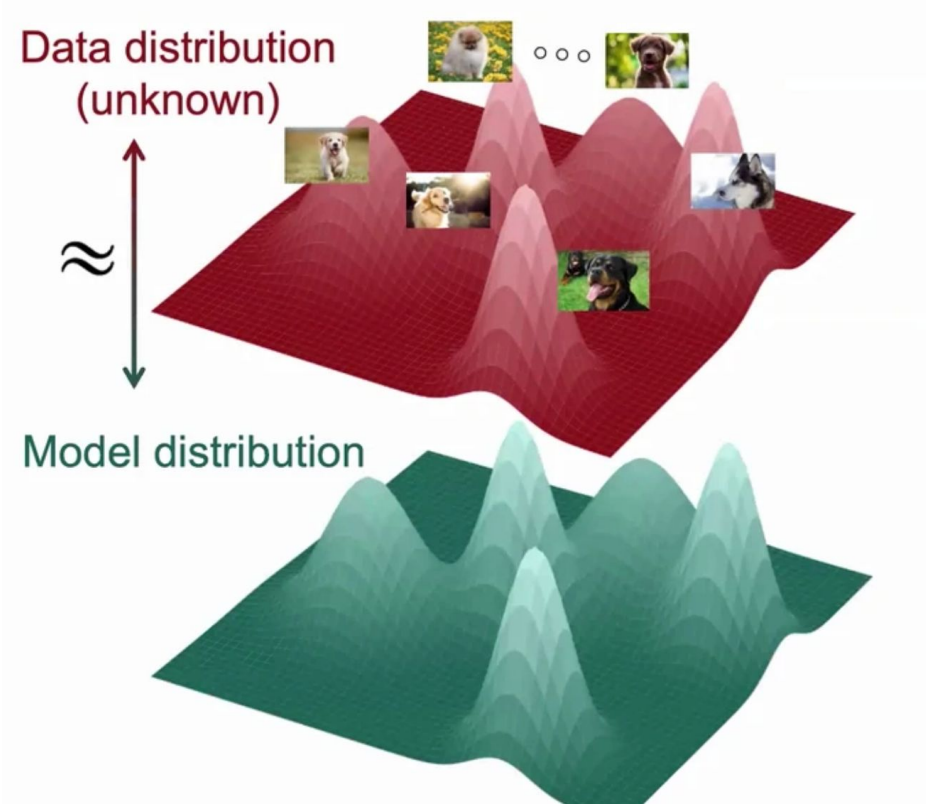
Generative models



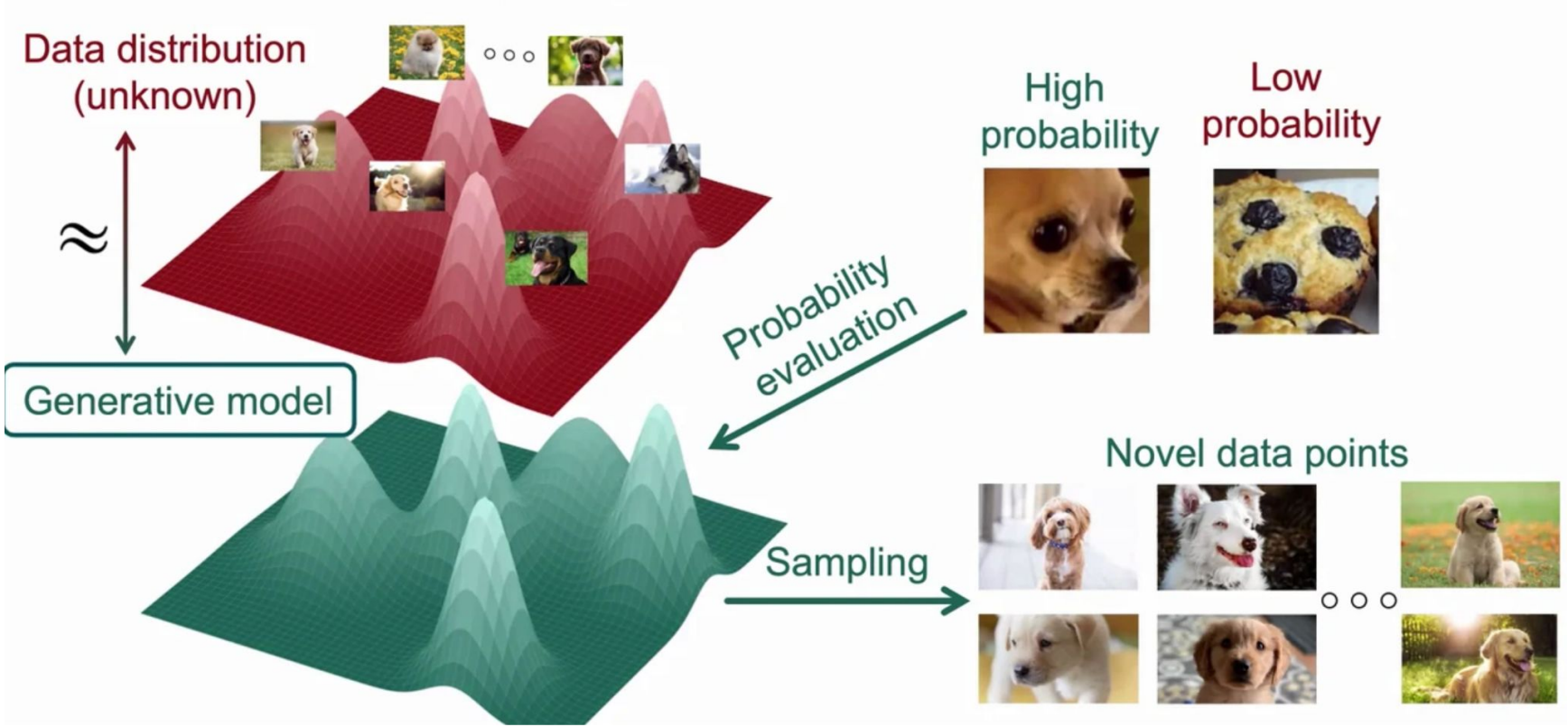
Diffusion models



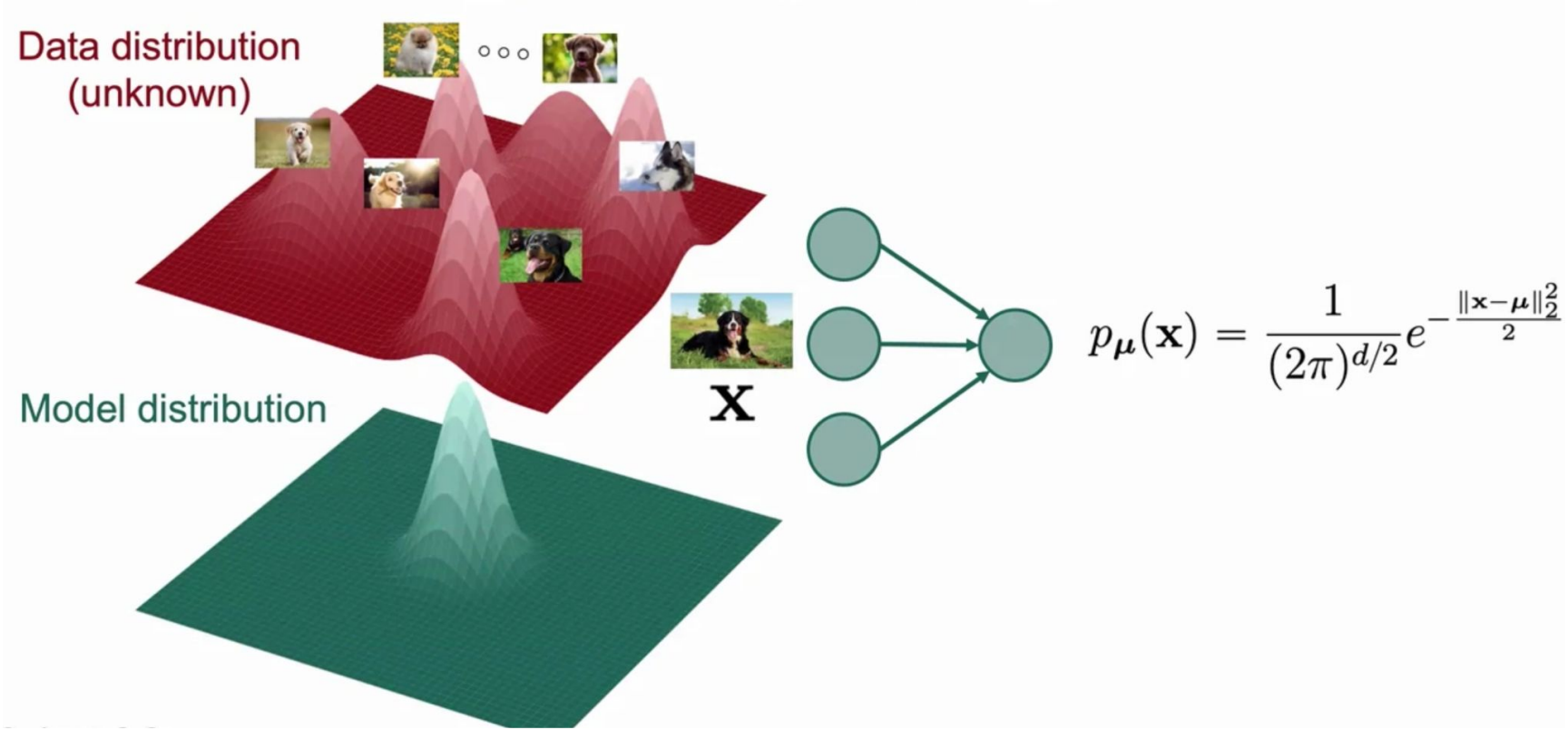
Estimating the probability distribution of data



Estimating the probability distribution of data

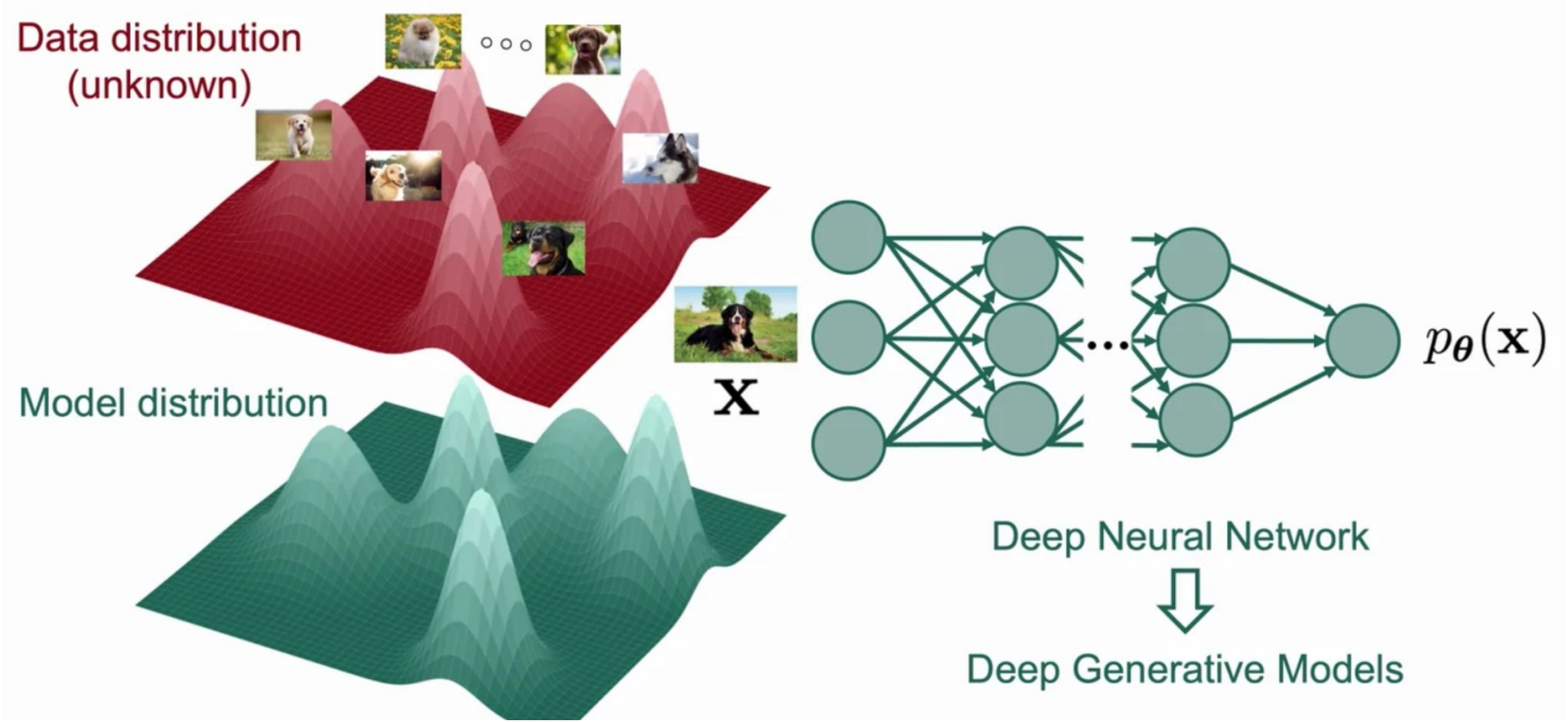


Estimating the probability distribution of data



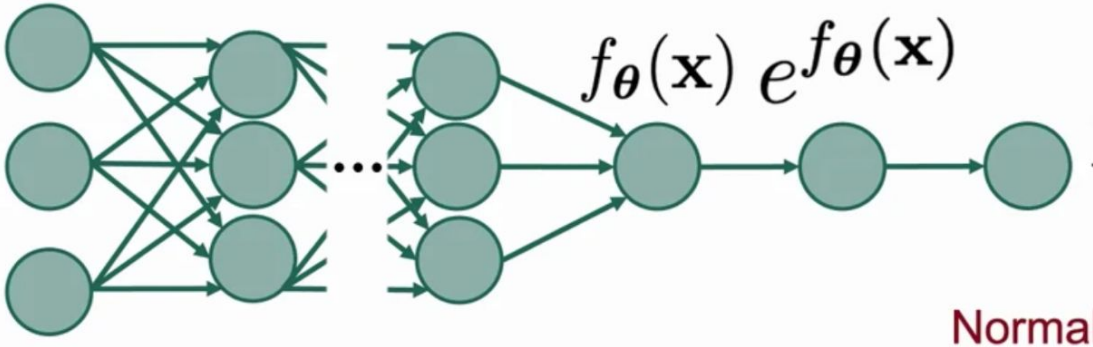
$$p_{\mu}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{\|\mathbf{x}-\mu\|_2^2}{2}}$$

Estimating the probability distribution of data



Estimating the probability distribution of data

$\mathbf{x}$

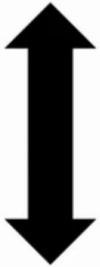


$$Z_{\boldsymbol{\theta}} = \int e^{f_{\boldsymbol{\theta}}(\mathbf{x})} \, \mathrm{d}\mathbf{x}$$

Estimating the score function of data

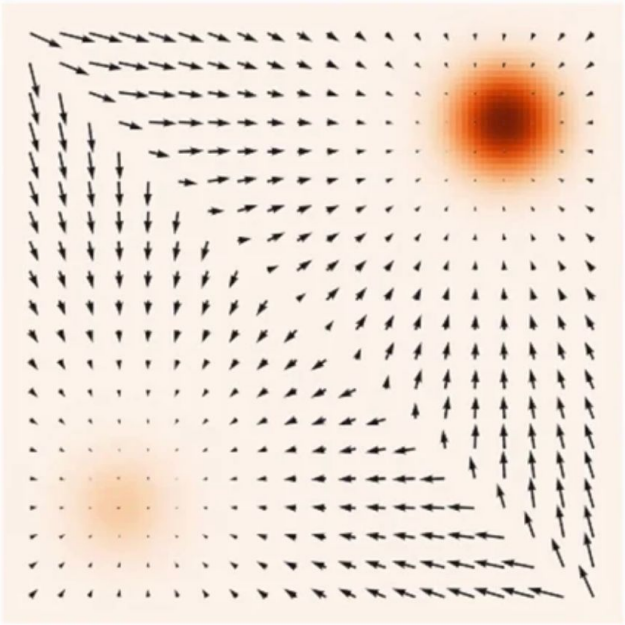
$p(\mathbf{x})$

Probability density function



$\nabla_{\mathbf{x}} \log p(\mathbf{x})$

(Stein) score function

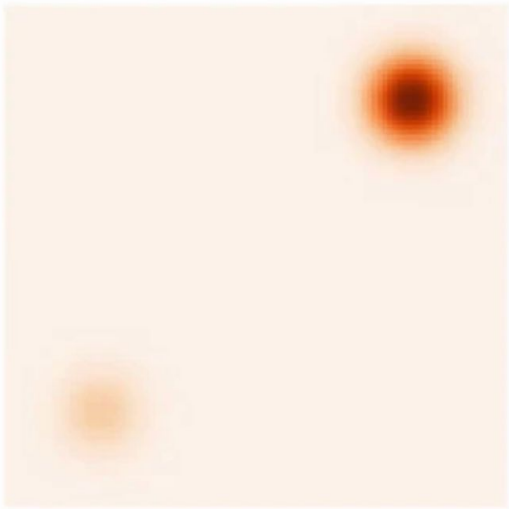


Score vs. density function

Estimating the score function of data

Training data

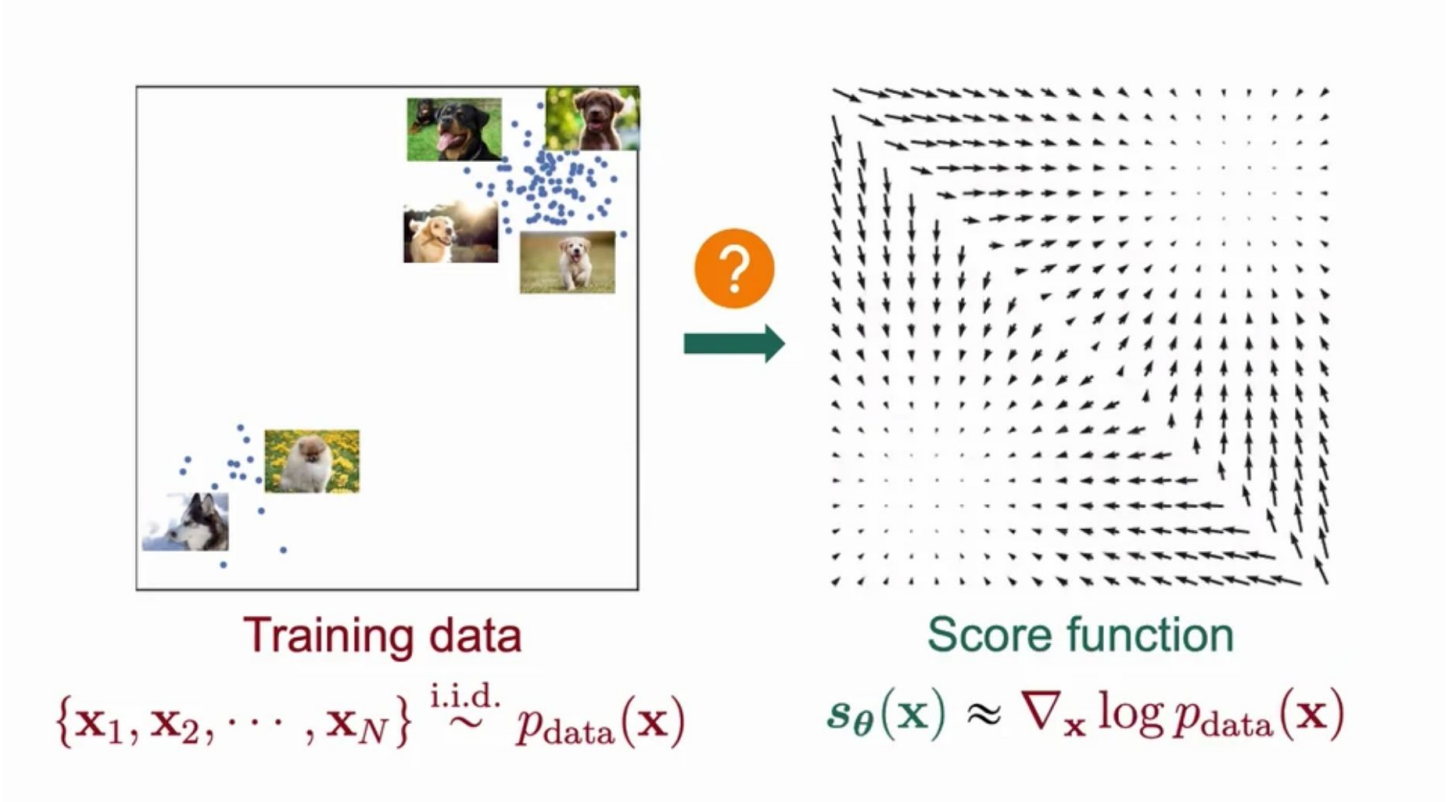
$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$$



Probability density function

$$p_{\boldsymbol{\theta}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$$

Estimating the score function of data



Estimating the score function of data

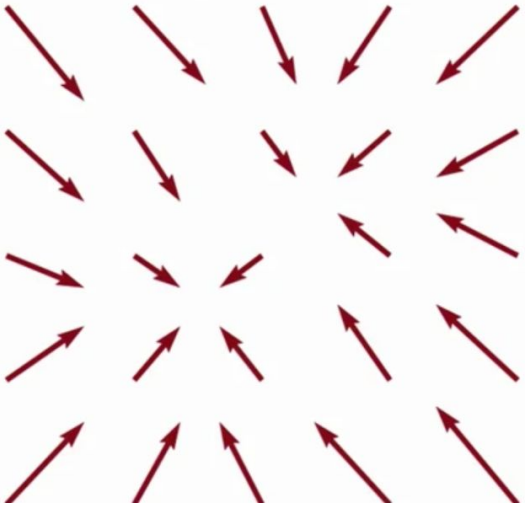
Given: $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$

Goal: $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$

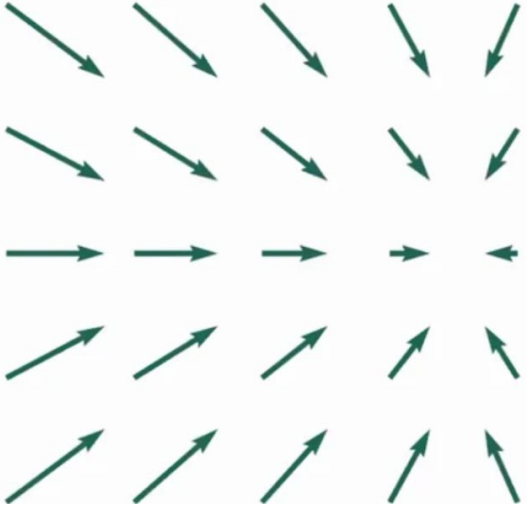
Score Model: $s_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$

Objective: How to compare two vector fields of scores?

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



$s_\theta(\mathbf{x})$



Estimating the score function of data

Given: $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$

Goal: $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$

Score Model: $\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$

Objective: How to compare two vector fields of scores?

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

(Fisher divergence)

Estimating the score function of data

$$
\begin{aligned}
&\frac{1}{2}\mathbb{E}_{p_{\text{data}}}[(\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_\theta(x))^2] \\
=&\frac{1}{2}\int p_{\text{data}}(x)(\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_\theta(x))^2 \mathrm{d}x \\
=&\underbrace{\frac{1}{2}\int p_{\text{data}}(x)(\nabla_x \log p_{\text{data}}(x))^2 \mathrm{d}x}_{\text{const}} + \frac{1}{2}\int p_{\text{data}}(x)(\nabla_x \log p_\theta(x))^2 \mathrm{d}x \\
&- \int p_{\text{data}}(x)\nabla_x \log p_\theta(x)\nabla_x \log p_{\text{data}}(x)\mathrm{d}x.
\end{aligned}
$$

Estimating the score function of data

$$
\begin{aligned}
&-\int p_{\text{data}}(x)\nabla_x \log p_\theta(x)\nabla_x \log p_{\text{data}}(x)\mathrm{d}x \\
=&-\int \nabla_x \log p_\theta(x)\nabla_x p_{\text{data}}(x)\mathrm{d}x \\
=&-p_{\text{data}}(x)\nabla_x \log p_\theta(x)\Big|_{-\infty}^{\infty}+\int p_{\text{data}}(x)\nabla_x^2 \log p_\theta(x)\mathrm{d}x \\
\overset{(i)}{=}&\mathbb{E}_{p_{\text{data}}}[\nabla_x^2 \log p_\theta(x)],
\end{aligned}
$$

Estimating the score function of data

Given: $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$

Goal: $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$

Score Model: $s_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$

Objective: How to compare two vector fields of scores?

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

(Fisher divergence)

Integration by parts (Gauss's theorem)

Score Matching [Hyvärinen 2005]:

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}\|s_\theta(\mathbf{x})\|_2^2 + \text{trace}(\underbrace{\nabla_{\mathbf{x}} s_\theta(\mathbf{x})}_{\text{Jacobian of } s_\theta(\mathbf{x})})\right]$$

Denoising score matching

$\mathbf{x}$

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

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

$\tilde{\mathbf{x}}$

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

- Matching the score of a noise-perturbed distribution

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

- Denoising score matching (Vincent 2011)

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

Scalable!

Denoising score matching

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

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

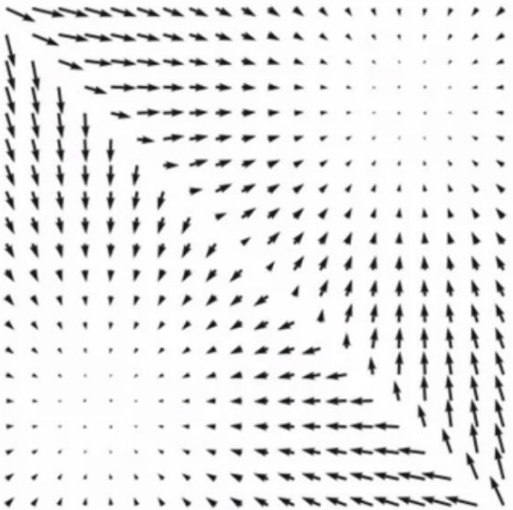
$$\frac{1}{2n}\sum_{i=1}^{n}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}_i) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}_i \mid \mathbf{x}_i)\|_2^2]$$

- If Gaussian perturbation

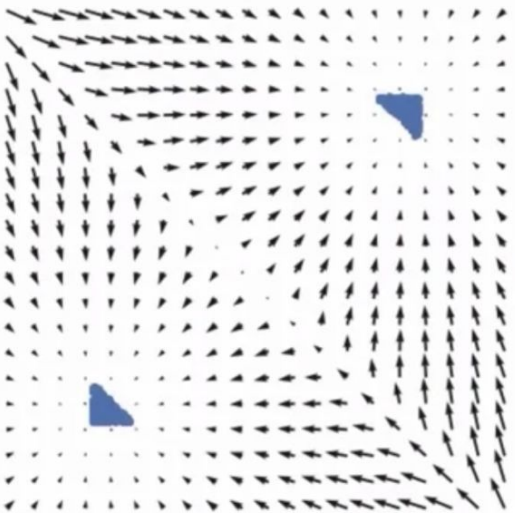
$$\frac{1}{2n}\sum_{i=1}^{n}\left[\left\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}_i) + \frac{\tilde{\mathbf{x}}_i - \mathbf{x}_i}{\sigma^2}\right\|_2^2\right]$$

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

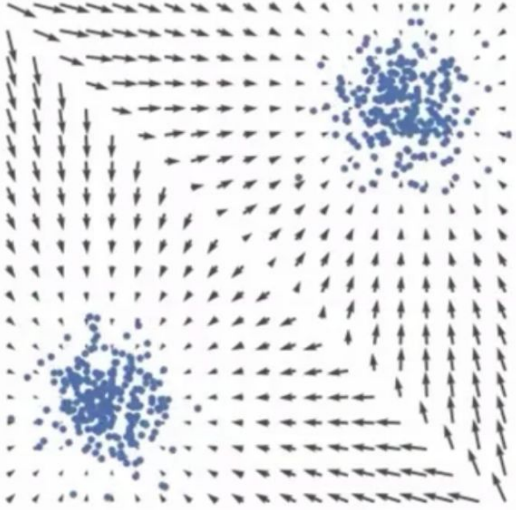
Sampling from score function



Score function



Follow the scores



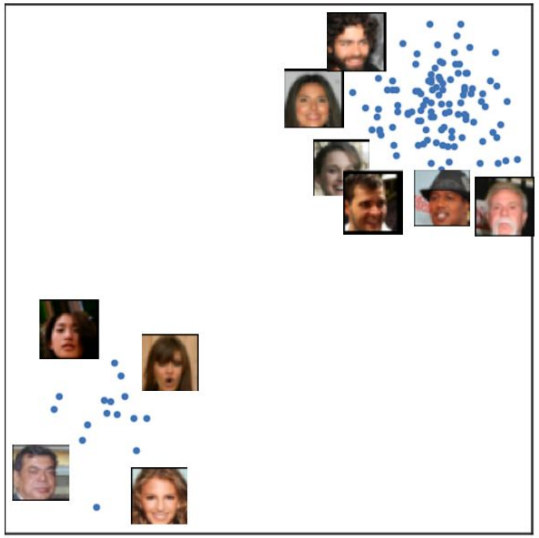
Follow the noisy scores

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\, \mathbf{z}_i$$

Langevin dynamics

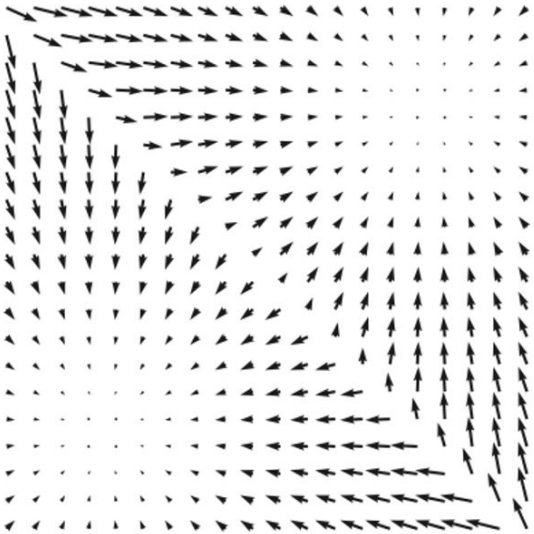
Sampling from score function



Data samples

$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p(\mathbf{x})$$

score matching



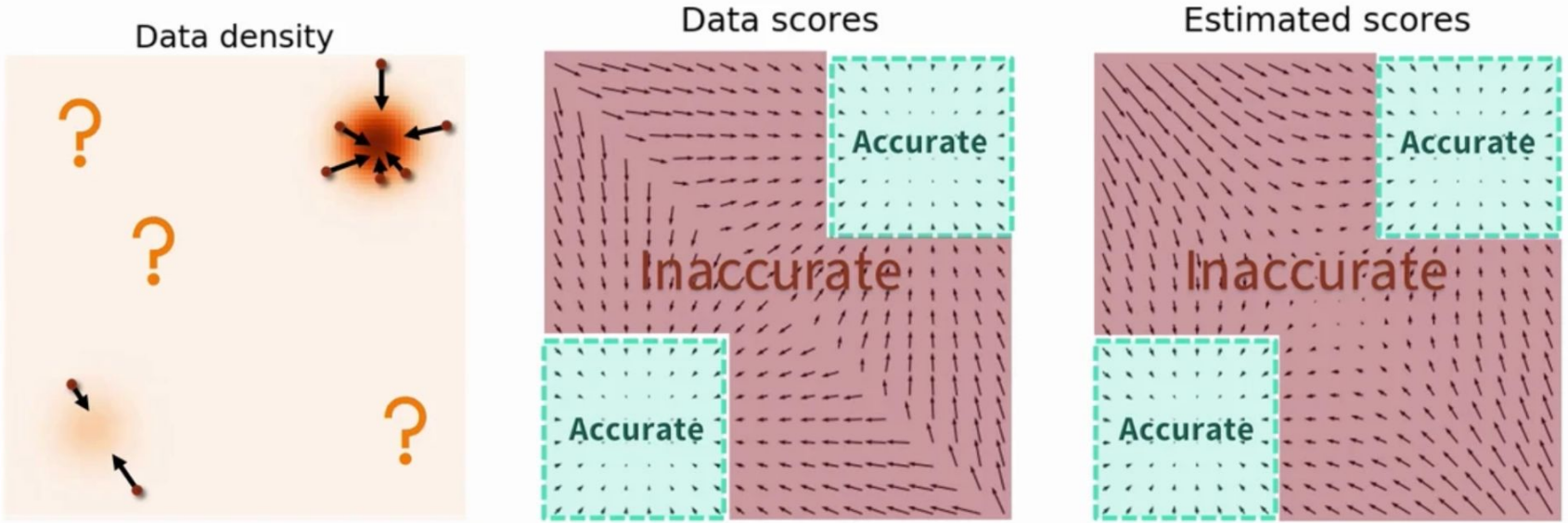
Scores

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

Langevin dynamics

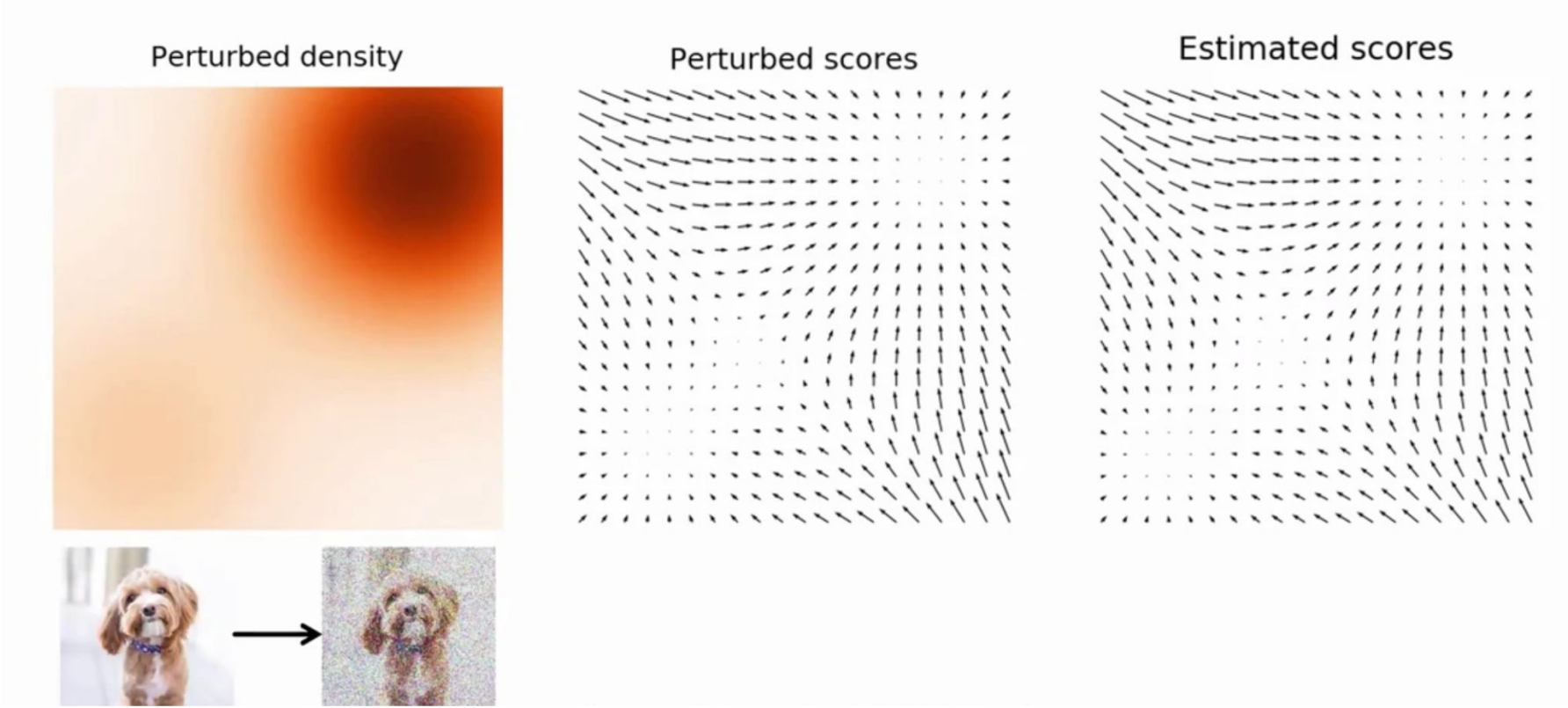
New samples

Challenge in low data regions

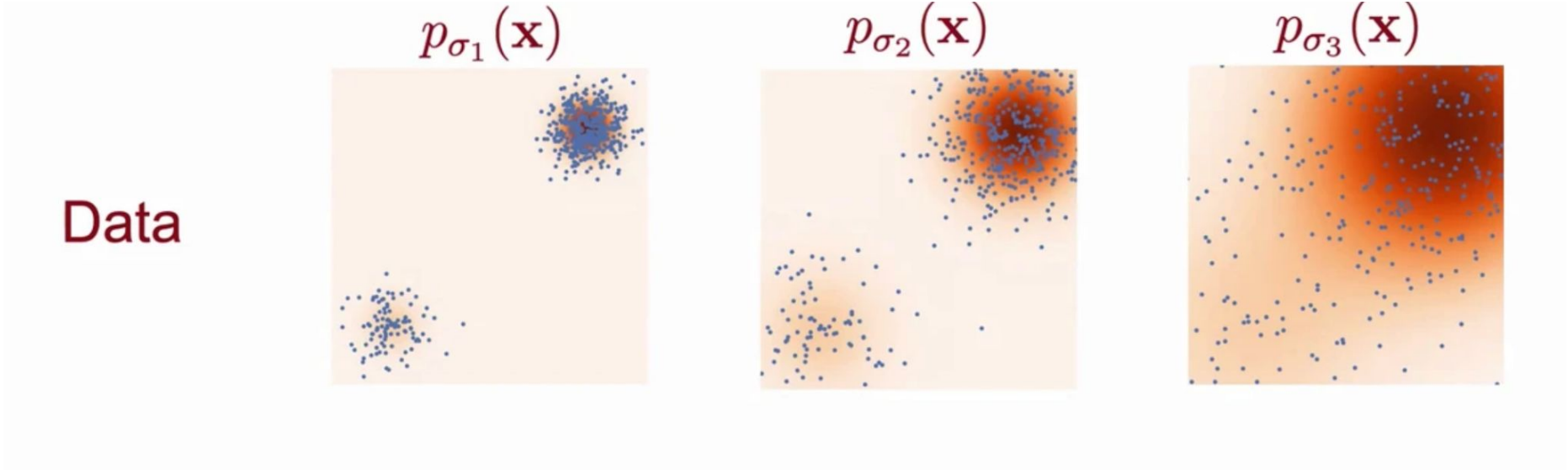


$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x})\|_2^2]$$

Challenge in low data regions



Using multiple noise levels

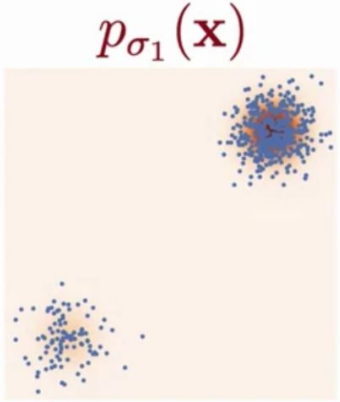


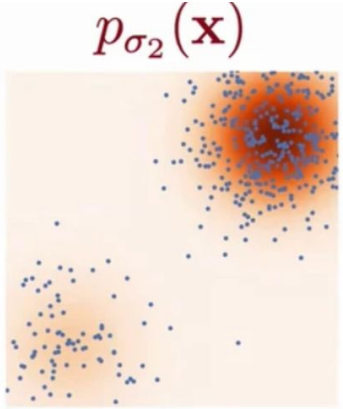
Using multiple noise levels

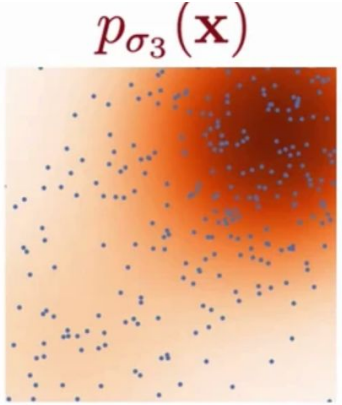
Data

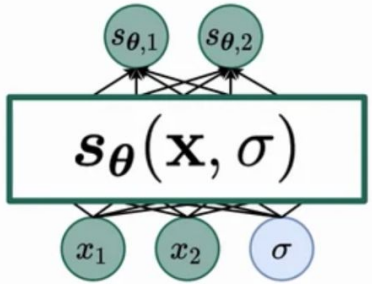
Using multiple noise levels

Data







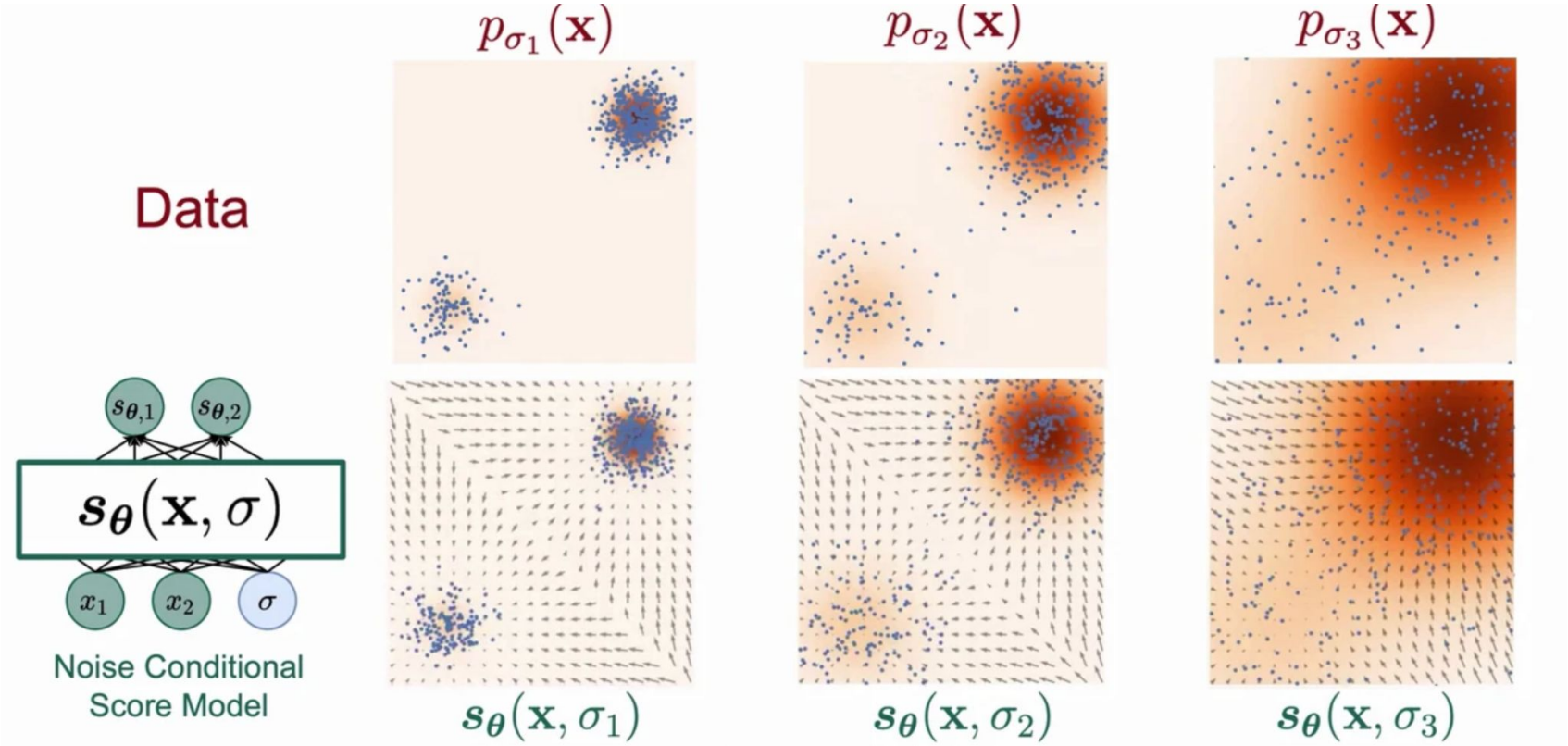


Noise Conditional Score Model

Positive weighting function

$$\frac{1}{N}\sum_{i=1}^{N}\overbrace{\lambda(\sigma_i)}\underbrace{\mathbb{E}_{p_{\sigma_i}(\mathbf{x})}[\|\nabla_{\mathbf{x}}\log p_{\sigma_i}(\mathbf{x}) - s_\theta(\mathbf{x},\sigma_i)\|_2^2]}_{\text{Score matching loss}}$$

Using multiple noise levels



Learning NCSNs via score matching

$$\ell(\boldsymbol{\theta}; \sigma) \triangleq \frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathcal{N}(\mathbf{x},\sigma^2 I)}\left[\left\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2}\right\|_2^2\right]$$

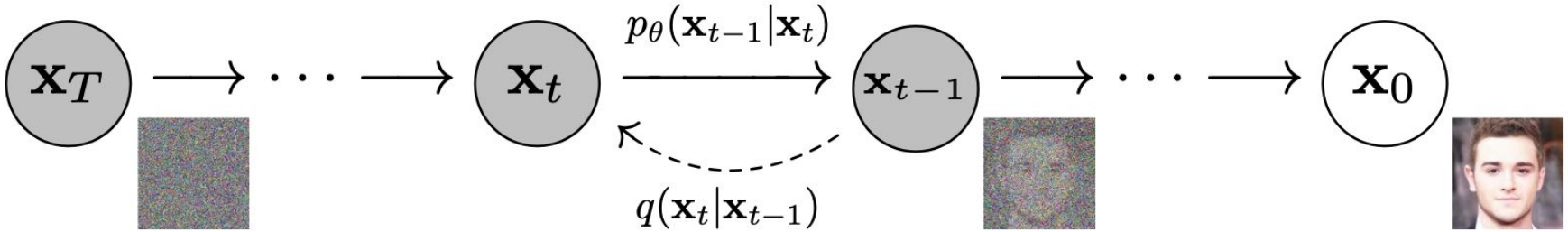
$$\mathcal{L}(\boldsymbol{\theta}; \{\sigma_i\}_{i=1}^L) \triangleq \frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)\ell(\boldsymbol{\theta}; \sigma_i)$$

Algorithm 1 Annealed Langevin dynamics.

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

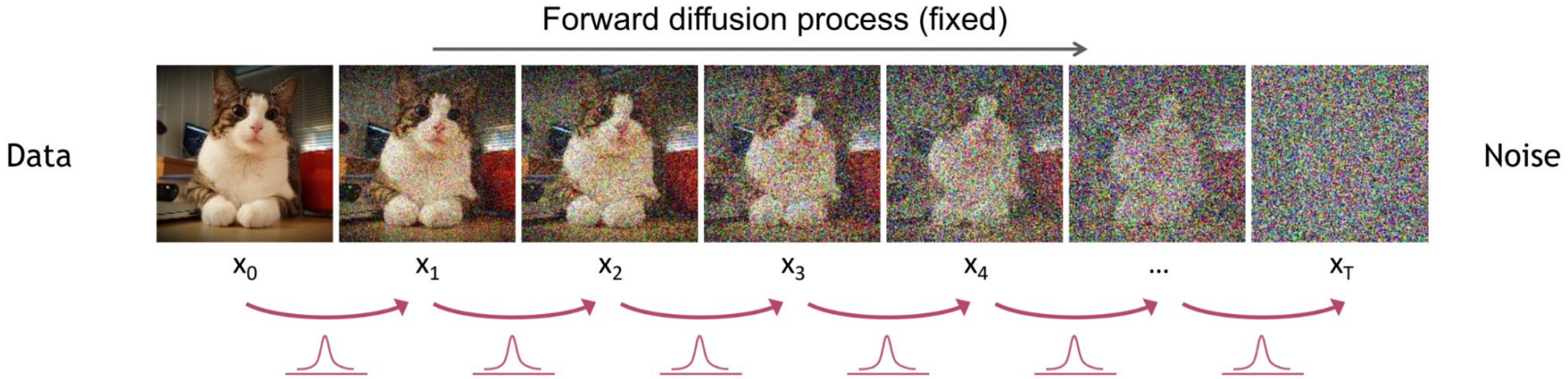
Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to L **do**
  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2/\sigma_L^2$ ▷ α_i is the step size.
  for $t \leftarrow 1$ to T **do**
    Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{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

Denoising Diffusion Probabilistic Models

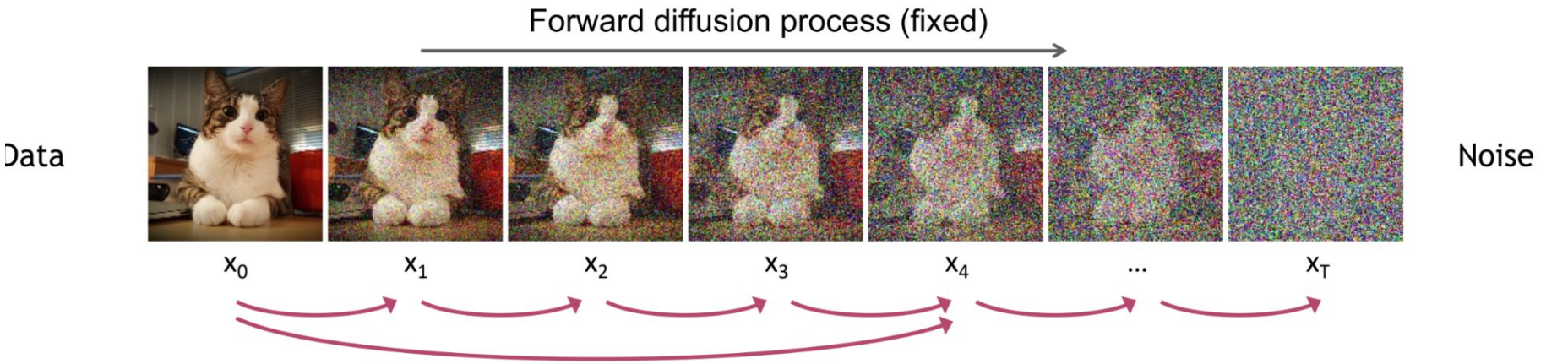


Denoising Diffusion Probabilistic Models

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



Denoising Diffusion Probabilistic Models

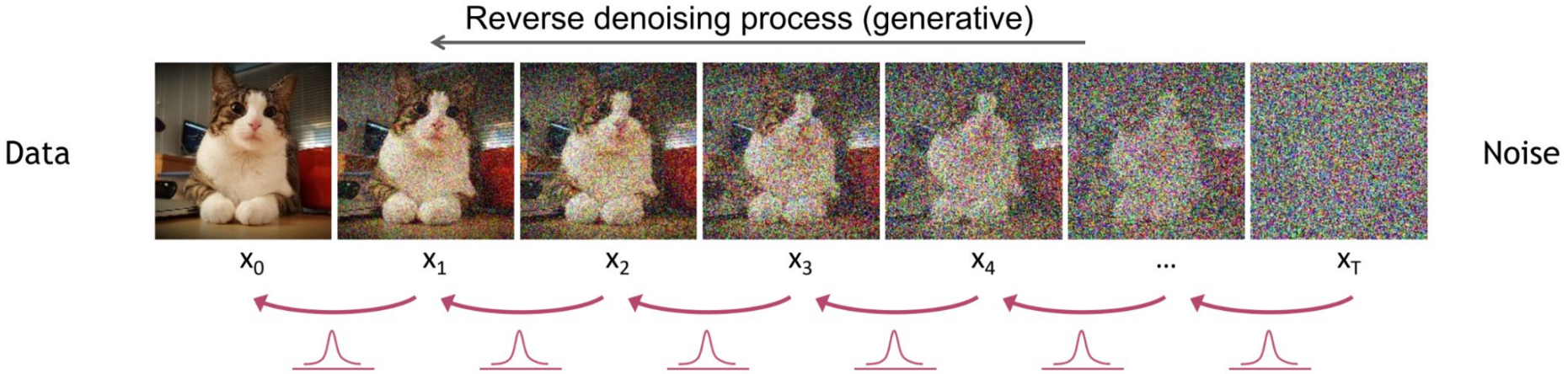


Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Denoising Diffusion Probabilistic Models

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$



Denoising Diffusion Probabilistic Models

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] = \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] =: L$$

$$\mathbb{E}_q\Bigg[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Bigg]$$

Denoising Diffusion Probabilistic Models

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}),$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \| \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) \|^2 \right] + C$$

Denoising Diffusion Probabilistic Models

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \| \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) \|^2 \right] + C$$

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \tilde{\boldsymbol{\mu}}_t \left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t)) \right)$$

$$\mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t) \right\|^2 \right]$$

Denoising Diffusion Probabilistic Models

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

Algorithm 1 Training

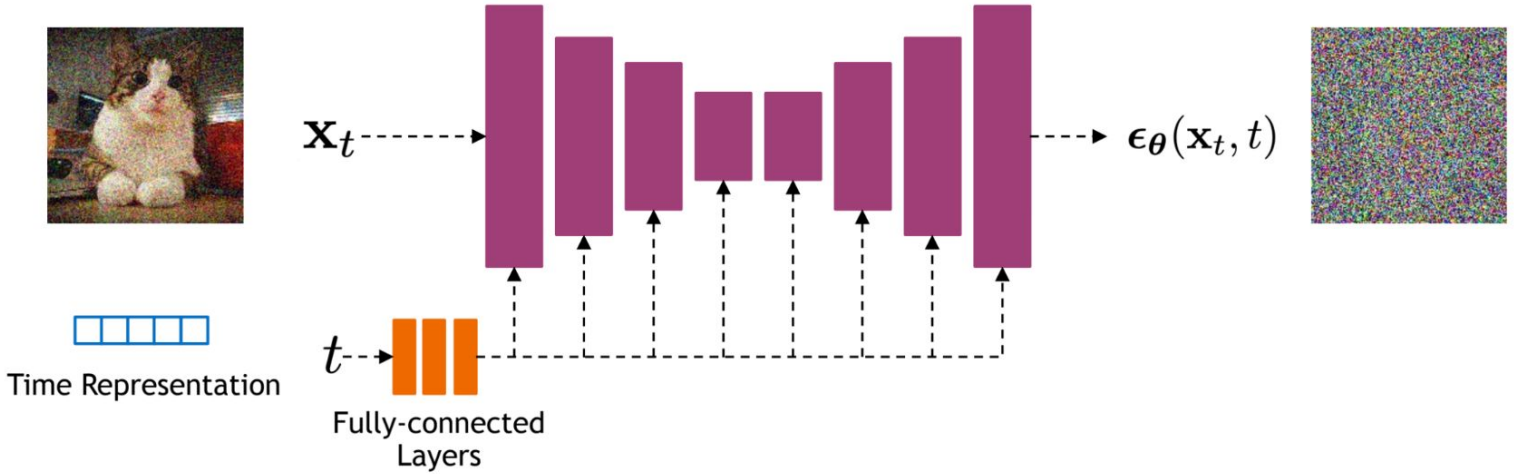
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

Algorithm 2 Sampling

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

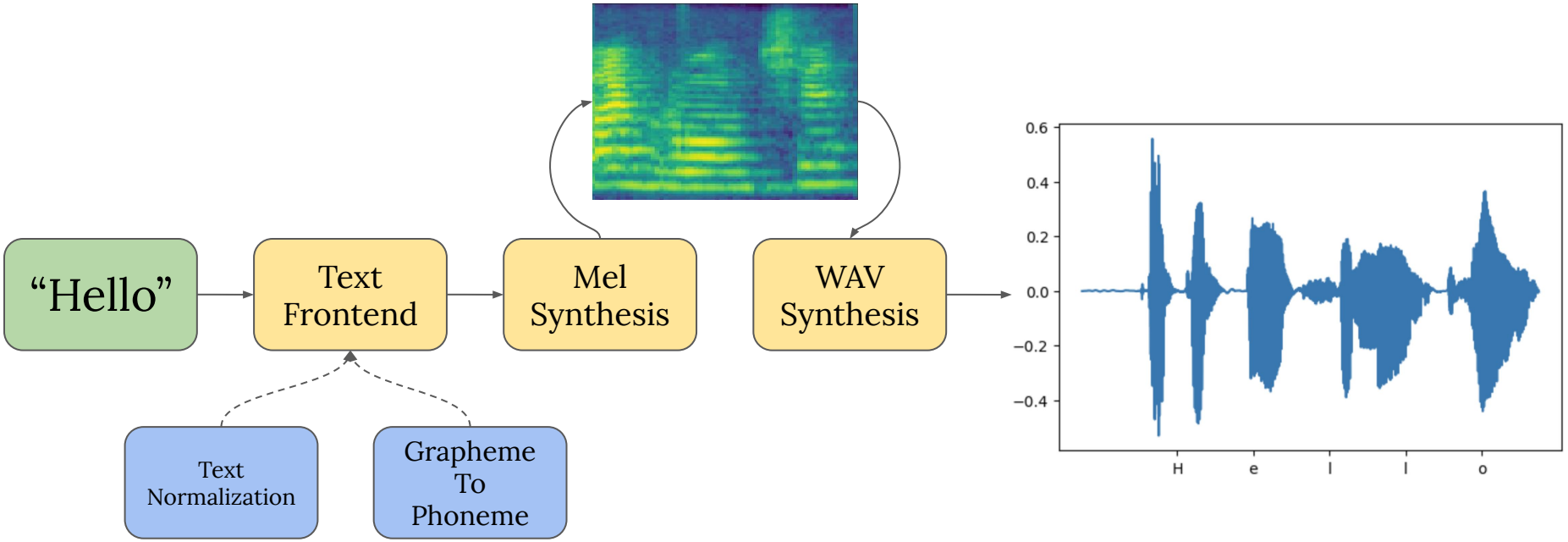
Network architectures

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$

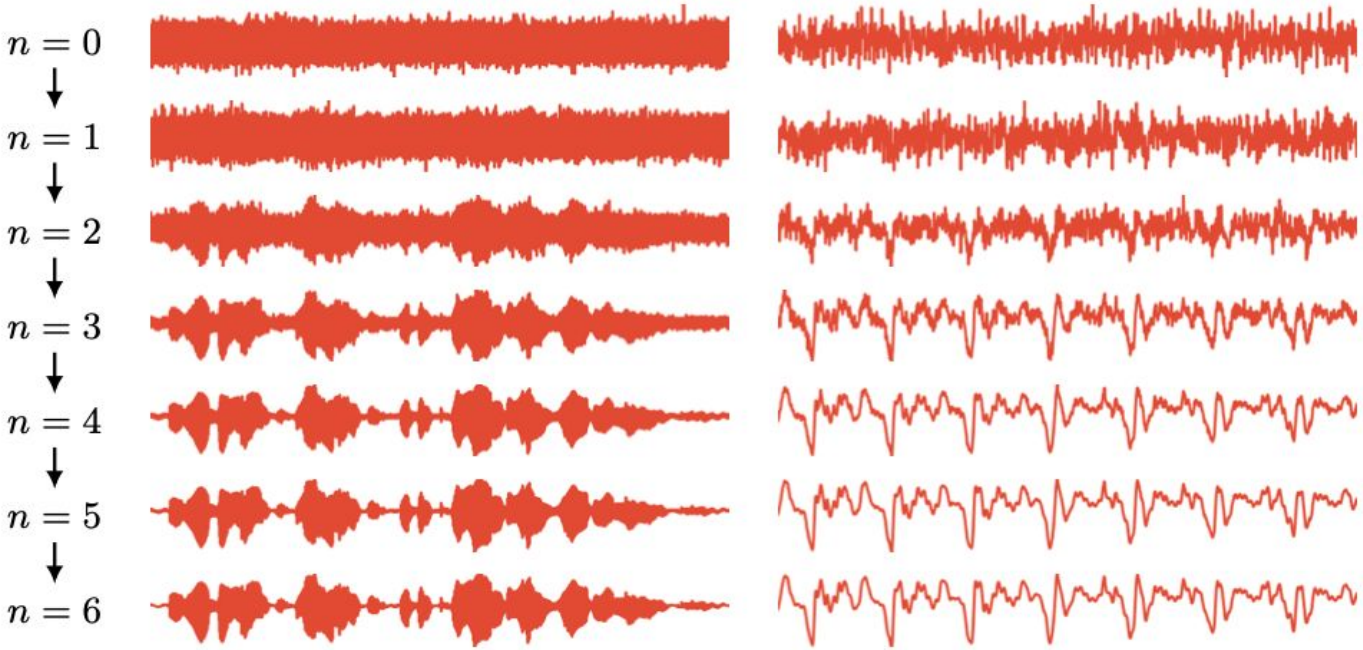


Time representation: sinusoidal positional embeddings or random Fourier features.

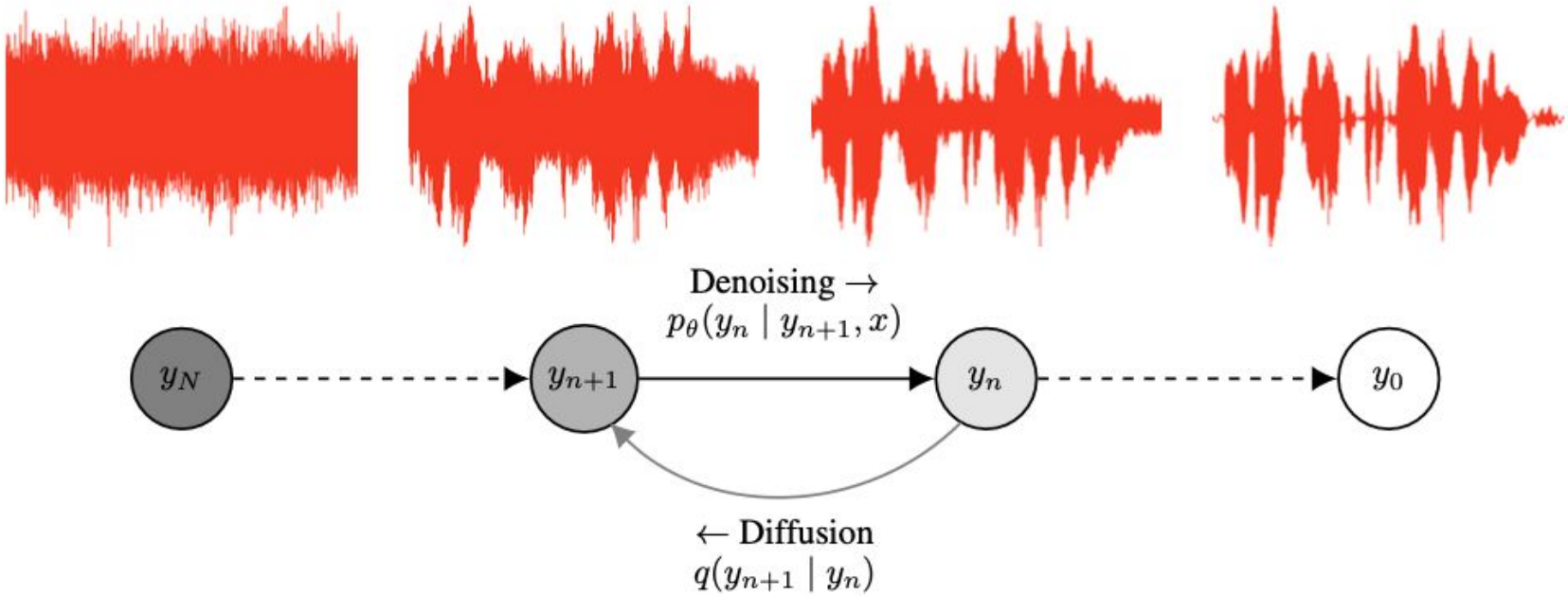
TTS recap


“Hello”
Text Frontend
Mel Synthesis
WAV Synthesis
Text Normalization
Grapheme To Phoneme
0.6
0.4
0.2
0.0
−0.2
−0.4
H
e
l
l
o

WaveGrad - Vocoder



WaveGrad - Vocoder



WaveGrad - Vocoder

$$q(y_{1:N} \mid y_0) \coloneqq \prod_{n=1}^{N} q(y_n \mid y_{n-1}), \qquad q(y_n \mid y_{n-1}) \coloneqq \mathcal{N}\left(y_n; \sqrt{(1-\beta_n)}\, y_{n-1}, \beta_n I\right),$$

$$y_n = \sqrt{\bar{\alpha}_n}\, y_0 + \sqrt{(1-\bar{\alpha}_n)}\, \epsilon$$

$$p_\theta(y_0 \mid x) \coloneqq \int p_\theta(y_{0:N} \mid x)\, \mathrm{d}y_{1:N}$$

$$\mathbb{E}_{n,\epsilon}\left[C_n \left\| \epsilon_\theta\left(\sqrt{\bar{\alpha}_n}\, y_0 + \sqrt{1-\bar{\alpha}_n}\, \epsilon, x, n\right) - \epsilon \right\|_2^2\right]$$

WaveGrad - Vocoder

$$\mathbb{E}_{\bar{\alpha},\epsilon}\left[\left\|\epsilon_\theta\left(\sqrt{\bar{\alpha}}\,y_0+\sqrt{1-\bar{\alpha}}\,\epsilon,x,\sqrt{\bar{\alpha}}\right)-\epsilon\right\|_1\right],$$

$$l_0=1,\qquad l_s=\sqrt{\textstyle\prod_{i=1}^{s}(1-\beta_i)}.$$

We first sample a segment $s\sim U(\{1,\dots,S\})$, which provides a segment (l_{s-1},l_s), and then sample from this segment uniformly to give $\sqrt{\bar{\alpha}}$.

WaveGrad - Vocoder

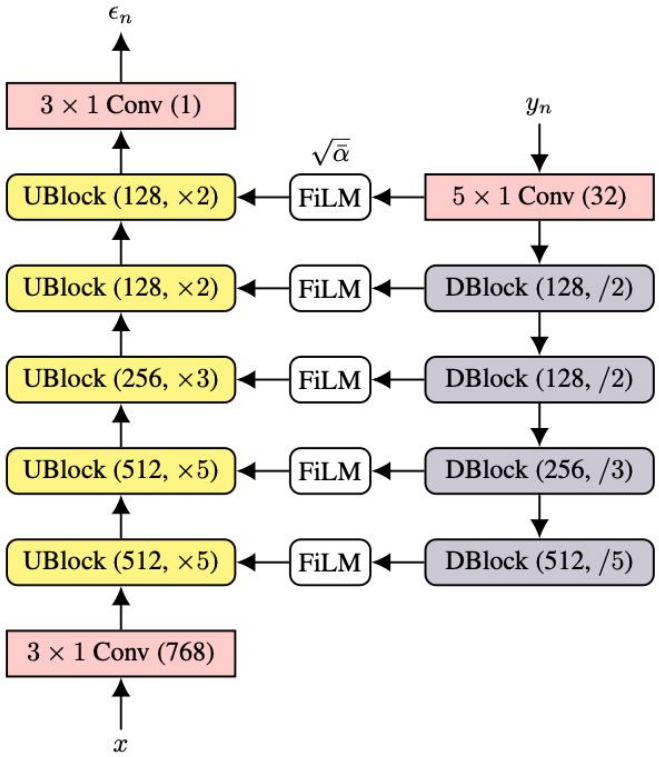
Algorithm 1 Training. WaveGrad directly conditions on the continuous noise level $\sqrt{\bar{\alpha}}$. l is from a predefined noise schedule.

repeat
  $y_0 \sim q(y_0)$
  $s \sim \text{Uniform}(\{1, \dots, S\})$
  $\sqrt{\bar{\alpha}} \sim \text{Uniform}(l_{s-1}, l_s)$
  $\epsilon \sim \mathcal{N}(0, I)$
  Take gradient descent step on
    $\nabla_\theta \left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}}\, y_0 + \sqrt{1-\bar{\alpha}}\, \epsilon, x, \sqrt{\bar{\alpha}}) \right\|_1$
until converged

Algorithm 2 Sampling. WaveGrad generates samples following a gradient-based sampler similar to Langevin dynamics.

$y_N \sim \mathcal{N}(0, I)$
for $n = N, \dots, 1$ **do**
  $z \sim \mathcal{N}(0, I)$
  $y_{n-1} = \frac{\left(y_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\, \epsilon_\theta(y_n, x, \sqrt{\bar{\alpha}_n})\right)}{\sqrt{\alpha_n}}$
  if $n > 1$, $y_{n-1} = y_{n-1} + \sigma_n z$
end for
return y_0

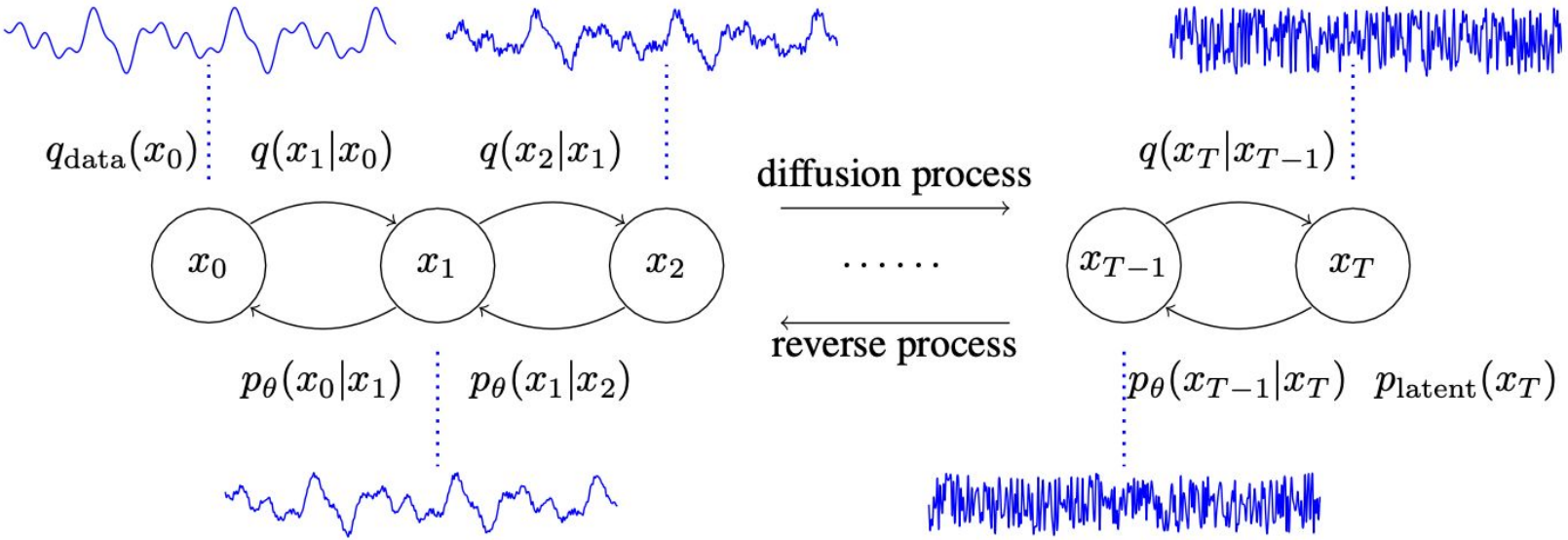
WaveGrad - Vocoder



WaveGrad - Vocoder

Model	**MOS (↑)**
WaveRNN	4.49 ± 0.04
Parallel WaveGAN	3.92 ± 0.05
MelGAN	3.95 ± 0.06
Multi-band MelGAN	4.10 ± 0.05
GAN-TTS	4.34 ± 0.04
WaveGrad	
Base (6 iterations, continuous noise levels)	4.41 ± 0.03
Base (1,000 iterations, discrete indices)	4.47 ± 0.04
Large (1,000 iterations, discrete indices)	4.51 ± 0.04
Ground Truth	4.58 ± 0.05

DiffWave - Vocoder



DiffWave - Vocoder

$$q(x_1, \cdots, x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}),$$

$$p_{\text{latent}}(x_T) = \mathcal{N}(0, I), \text{ and } p_\theta(x_0, \cdots, x_{T-1}|x_T) = \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t),$$

$$\min_\theta L_{\text{unweighted}}(\theta) = \mathbb{E}_{x_0,\epsilon,t} \left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\ t)\right\|_2^2$$

DiffWave - Vocoder

Algorithm 1 Training

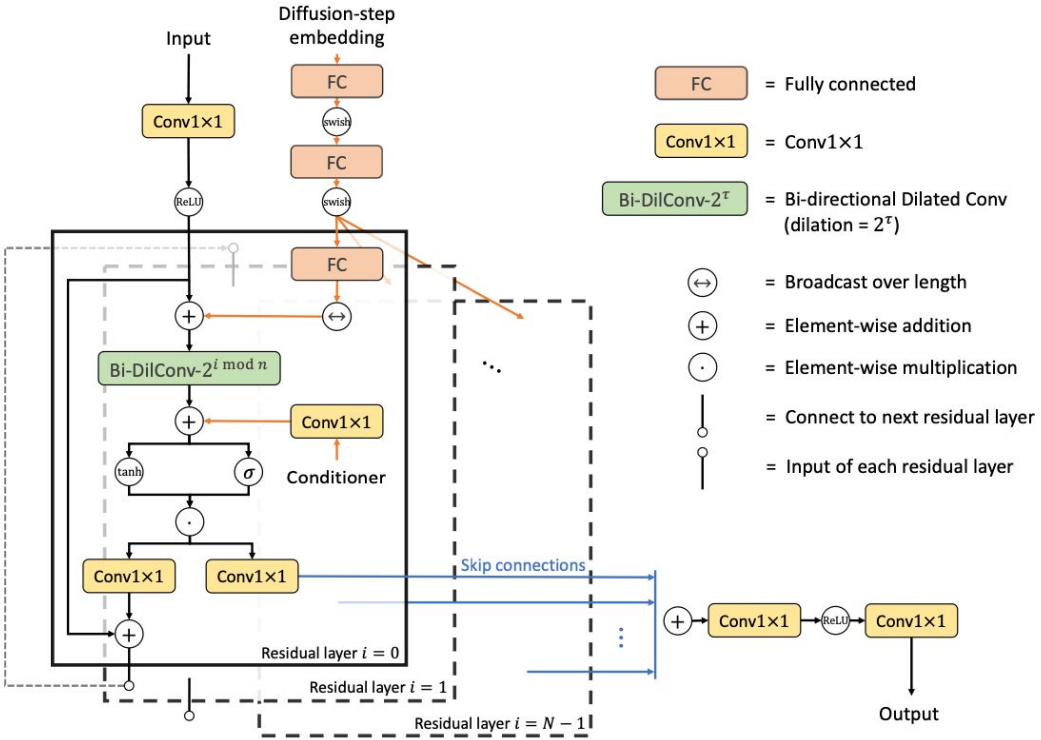
for $i = 1, 2, \cdots, N_{\text{iter}}$ **do**
  Sample $x_0 \sim q_{\text{data}}$, $\epsilon \sim \mathcal{N}(0, I)$, and
    $t \sim \text{Uniform}(\{1, \cdots, T\})$
  Take gradient step on
    $\nabla_\theta \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon,\ t)\|_2^2$
    according to Eq. (7)
end for

Algorithm 2 Sampling

Sample $x_T \sim p_{\text{latent}} = \mathcal{N}(0, I)$
for $t = T, T-1, \cdots, 1$ **do**
  Compute $\mu_\theta(x_t, t)$ and $\sigma_\theta(x_t, t)$ using Eq. (5)
  Sample $x_{t-1} \sim p_\theta(x_{t-1}|x_t) =$
    $\mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta(x_t, t)^2 I)$
end for
return x_0

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right), \quad \text{and} \quad \sigma_\theta(x_t, t) = \tilde{\beta}_t^{\frac{1}{2}}, \tag{5}$$

DiffWave - Vocoder



DiffWave - Vocoder

Model	T	T_{infer}	**layers**	**res. channels**	**#param(↓)**	**MOS(↑)**
WaveNet	—	—	30	128	4.57M	**4.43** $\pm$ 0.10
ClariNet	—	—	60	64	2.17M	4.27 $\pm$ 0.09
WaveGlow	—	—	96	256	87.88M	4.33 $\pm$ 0.12
WaveFlow	—	—	64	64	5.91M	4.30 $\pm$ 0.11
WaveFlow	—	—	64	128	22.25M	4.40 $\pm$ 0.07
DiffWave $_{\text{BASE}}$	20	20	30	64	2.64M	4.31 $\pm$ 0.09
DiffWave $_{\text{BASE}}$	40	40	30	64	2.64M	4.35 $\pm$ 0.10
DiffWave $_{\text{BASE}}$	50	50	30	64	2.64M	**4.38** $\pm$ 0.08
DiffWave $_{\text{LARGE}}$	200	200	30	128	6.91M	**4.44** $\pm$ 0.07
DiffWave $_{\text{BASE}}$ (Fast)	50	6	30	64	2.64M	*4.37* $\pm$ 0.07
DiffWave $_{\text{LARGE}}$ (Fast)	200	6	30	128	6.91M	*4.42* $\pm$ 0.09
Ground-truth	—	—	—	—	—	4.52 $\pm$ 0.06

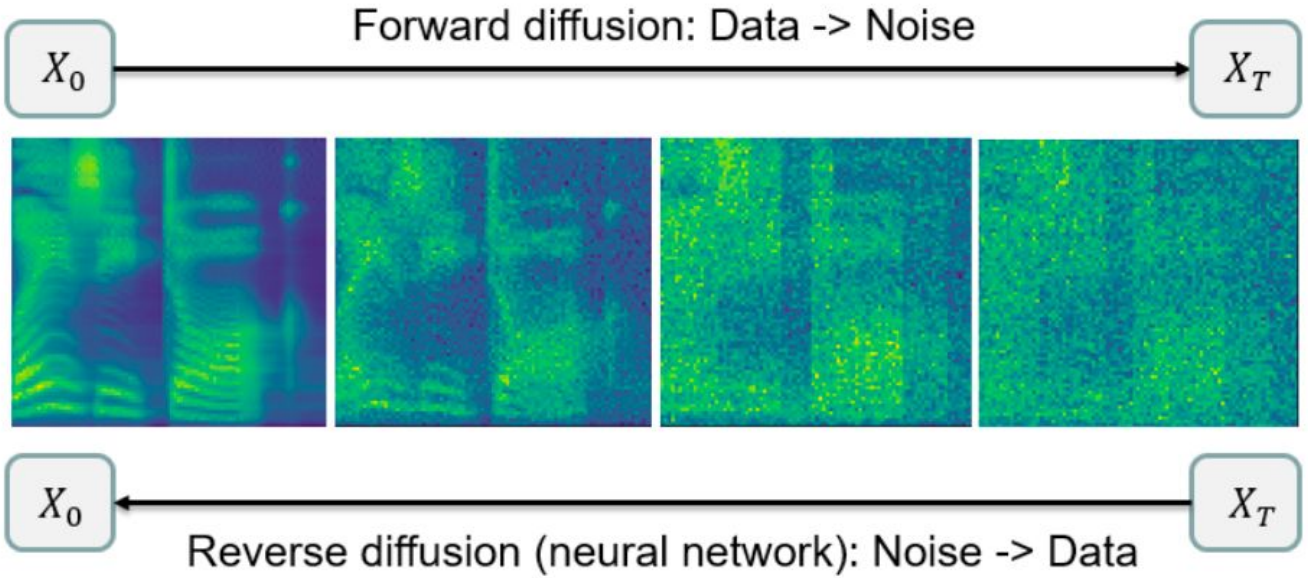
DiffWave - Vocoder

Model	**FID**(↓)	**IS**(↑)	**mIS**(↑)	**AM**(↓)	**NDB**/K(↓)	**MOS**(↑)
WaveNet-128	3.279	2.54	7.6	1.368	0.86	1.34 ± 0.29
WaveNet-256	2.947	2.84	10.0	1.260	0.86	1.43 ± 0.30
WaveGAN	1.349	4.53	36.6	0.796	0.78	2.03 ± 0.33
DiffWave	**1.287**	**5.30**	**59.4**	**0.636**	**0.74**	$\mathbf{3.39} \pm 0.32$
Trainset	0.000	8.48	281.4	0.164	0.00	—
Testset	0.011	8.47	275.2	0.166	0.10	3.72 ± 0.28

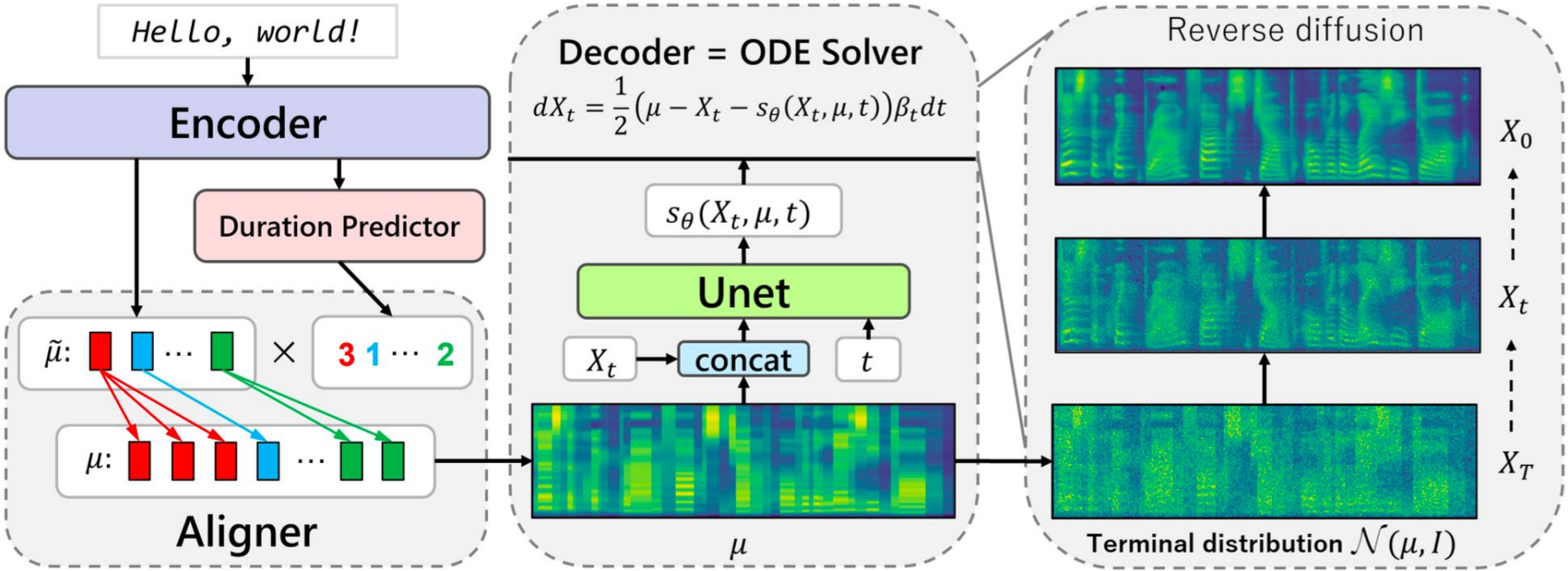
DiffWave - Vocoder

Model	**Accuracy**(↑)	**FID-class**(↓)	**IS**(↑)	**mIS**(↑)	**MOS**(↑)
WaveNet-128	56.20%	7.876±2.469	3.29	15.8	1.46 ± 0.30
WaveNet-256	60.70%	6.954±2.114	3.46	18.9	1.58 ± 0.36
DiffWave	91.20%	1.113±0.569	6.63	117.4	$\mathbf{3.50} \pm 0.31$
DiffWave (deep & thin)	**94.00%**	**0.932**±0.450	**6.92**	**133.8**	3.44 ± 0.36
Trainset	99.06%	0.000±0.000	8.48	281.4	—
Testset	98.76%	0.044±0.016	8.47	275.2	3.72 ± 0.28

GradTTS - Mel synthesis



GradTTS - Mel synthesis



GradTTS - Mel synthesis

Table 2. Model comparison.

Model	Enc params[1]	Dec params	RTF	Log-likelihood	MOS
Grad-TTS-1000	$7.2m$	$7.6m$	3.663	$\mathbf{0.174 \pm 0.001}$	$\mathbf{4.44 \pm 0.05}$
Grad-TTS-100			0.363		4.38 ± 0.06
Grad-TTS-10			0.033		4.38 ± 0.06
Grad-TTS-4			0.012		3.96 ± 0.07
Glow-TTS	$7.2m$	$21.4m$	0.008	0.082	4.11 ± 0.07
FastSpeech	$24.5m$		**0.004**	–	3.68 ± 0.09
Tacotron2	$28.2m$		0.075	–	4.32 ± 0.07
Ground Truth	–		–	–	4.53 ± 0.06

Important papers

- Generative Modeling by Estimating Gradients of the Data Distribution
- Denoising Diffusion Probabilistic Models
- WaveGrad: Estimating Gradients for Waveform Generation
- DiffWave: A Versatile Diffusion Model for Audio Synthesis
- Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech