Diffusion Models I

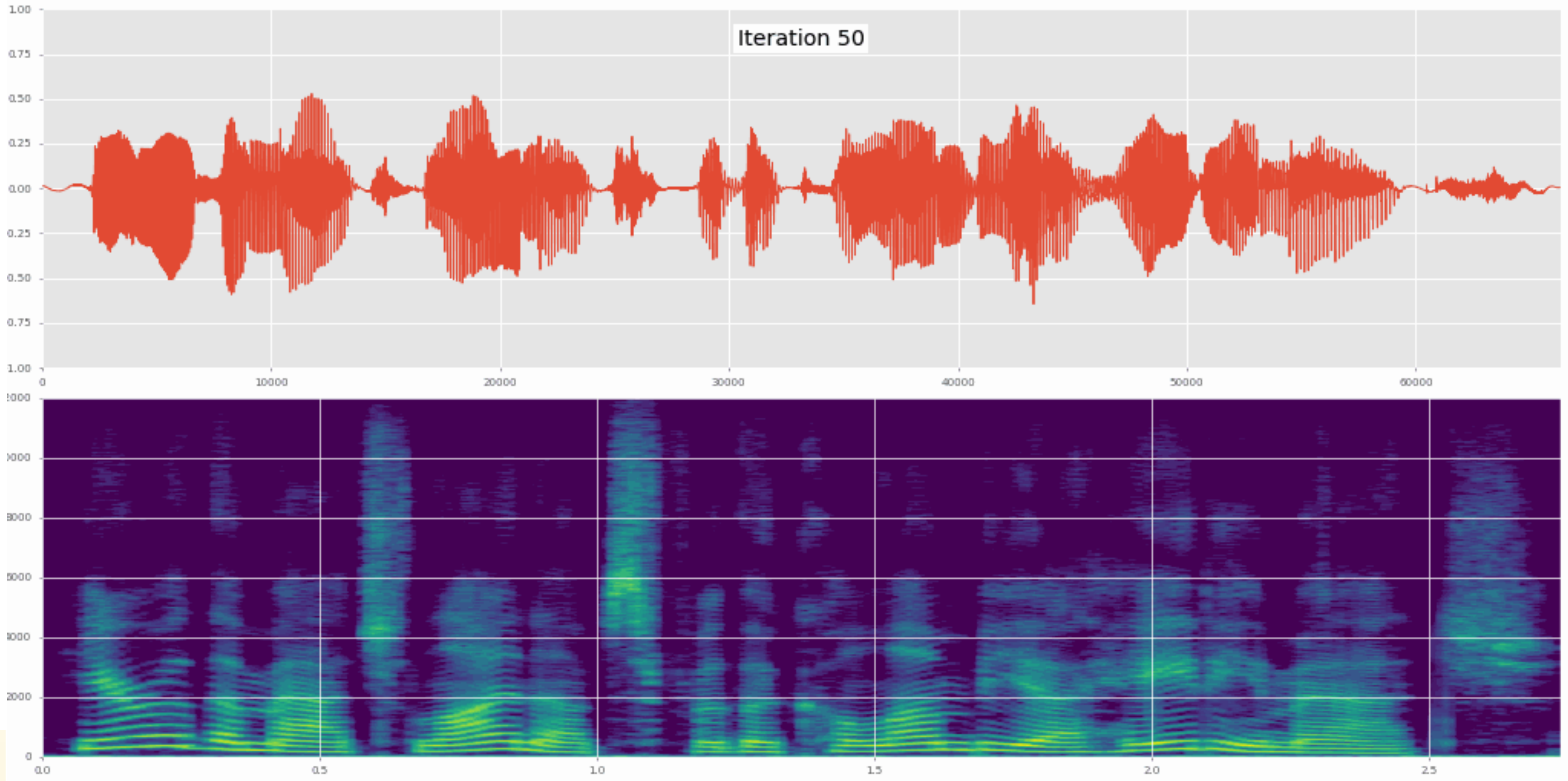
DDPMs

April 29th, 2024

Diffusion Models Beat GANs on Image Synthesis

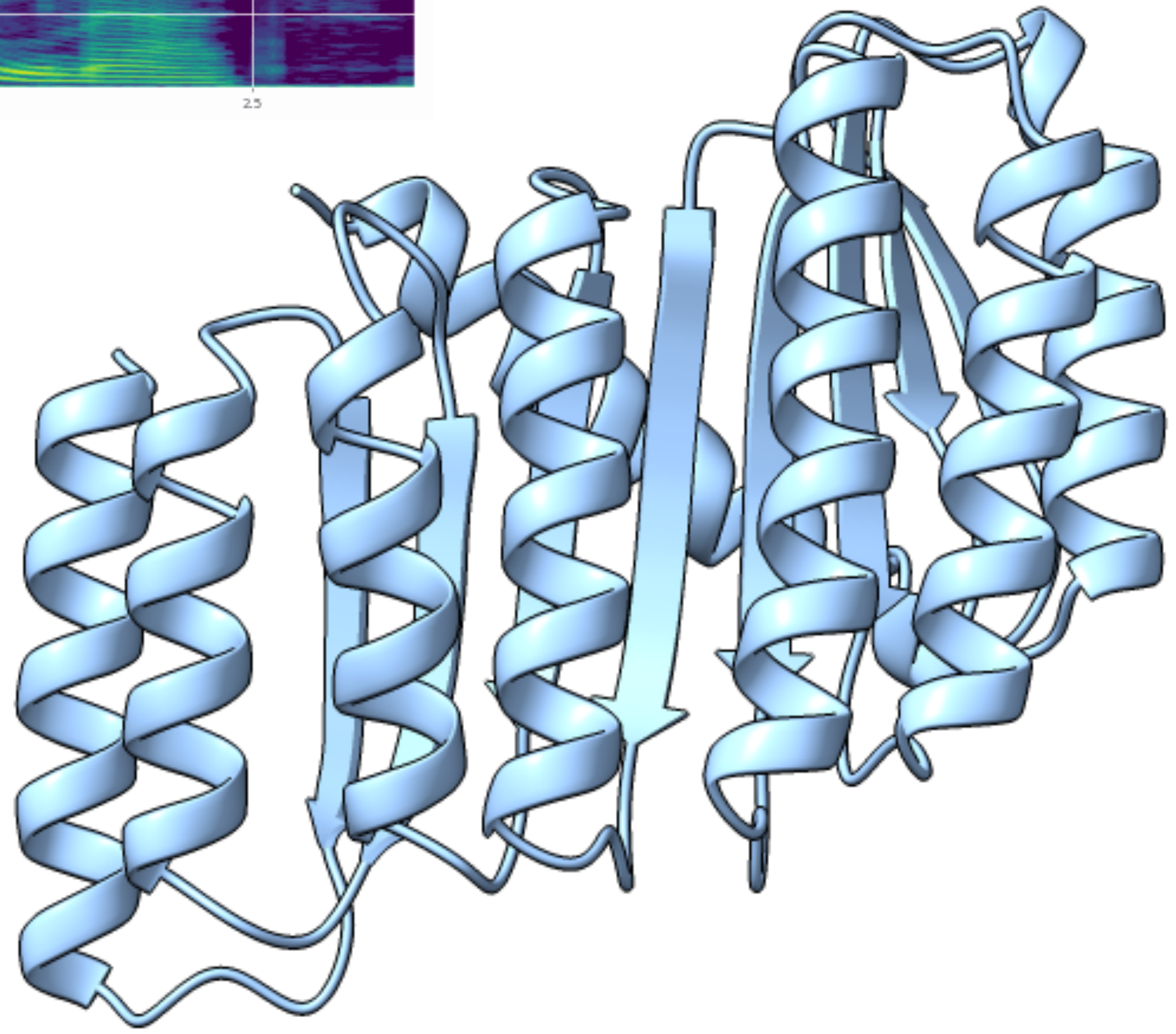
Prafulla Dhariwal*
OpenAI
prafulla@openai.com

Alex Nichol*
OpenAI
alex@openai.com



Audio generation with WaveGrad.

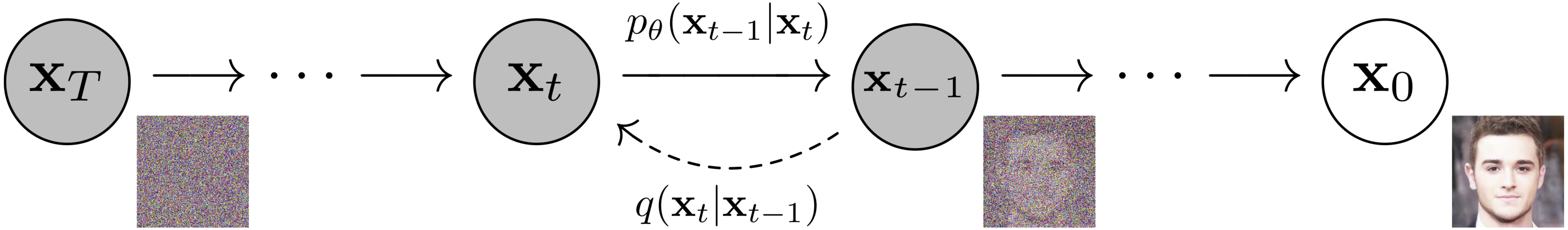
Image generated by DALL-E 3 based on prompt “cartoon penguin riding a unicycle”.



Generated protein backbone from RFDiffusion.

What are Diffusion Models?

- Diffusion models are a kind of deep generative model, i.e., a model that learns to sample from an underlying probability distribution.
- Diffusion models generate samples through a noising/denoising process:
  - During training, “noise” is added to samples from the distribution, and the model learns to predict this noise.
  - During inference, a noisy sample is first generated. The model then iteratively removes the noise until it produces a final result, which should be a sample from the desired distribution.



The noising and denoising process in DDPM

- Why do this?
  - The noisy distribution is much simple to sample from.
  - The generation process is broken down into smaller, easier steps.
  - Most of the time, it is easy to add a desired amount of noise to a sample, making training simple.

Denoising Diffusion Probabilistic Models

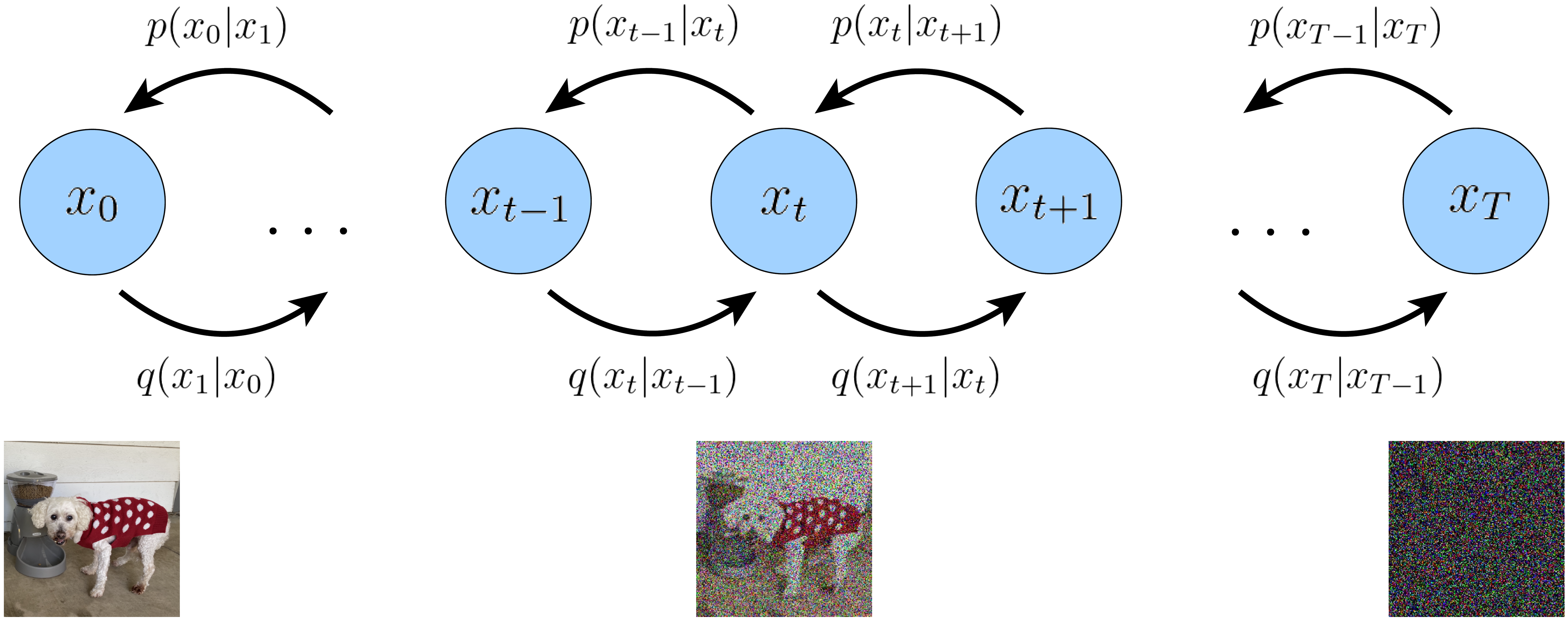
- First diffusion model described in “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”
  - Inspired by methods in thermodynamics and statistics (in particular, Annealed Importance Sampling)
- This lecture will focus on the diffusion model proposed in “Denoising Diffusion Probabilistic Models” (DDPMs)
  - Arguably the most popular form of diffusion models.
  - Simpler to understand and implement.
- Notation between papers is somewhat inconsistent, but we will be following the notation in the DDPM paper.

- Underlying distribution: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- Add Gaussian noise T times to get $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$
- Forward process is a Markov chain: $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$ where $0 < \beta_i < 1$ is the variance schedule
- Can sample at arbitrary t without stepping through MC: $\alpha_t := 1 - \beta_t$ and $\overline{\alpha}_t := \prod_{s=1}^{t} \alpha_s$ then

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\overline{\alpha}_t}\mathbf{x}_0, (1 - \overline{\alpha}_t)\mathbf{I})$$

Approaches standard normal distribution: $\mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

- Reverse process: $p_\theta(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$
- When β_t are small, the reverse process can also be written as Gaussian transitions: $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$
- Determining $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$ will determine the backward process. We set $\boldsymbol{\Sigma}_\theta = \beta_t \mathbf{I}$ for simplicity.
- Training goal is to minimize the negative log likelihood



Given trained model, sample Gaussian noise and then step through reverse process MC to get a sample $\mathbf{x}_0$

- see “Understanding Diffusion Models: A Unified Perspective” for detailed derivation

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq$$

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

- First term: reconstruction term. Can be optimized separately, but ultimately will be treated along with other terms.

- Second term: measure of difference between normal distribution and explicit prior distribution (does not depend on θ)

- Third term: measure of difference between backward process and the actual marginal distributions of the forward process.

- Let's optimize the third term through gradient descent!
- KL divergence terms have exact formulas when distributions are normal.
  - Recall: $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \beta_t \mathbf{I})$; Just need other distribution.

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0) q(\mathbf{x}_{t-1} \mid \mathbf{x}_0)}{q(\mathbf{x}_t \mid \mathbf{x}_0)}$$

$$\cdots$$

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

where,

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

- Plugging into KL divergence…

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

- One could train a model off this using gradient descent, but there's a simpler formulation not dependent on x_t.

- Re-parameterize based on explicit formula for $q(\mathbf{x}_t \mid \mathbf{x}_0)$. Knowing x_0 allows easy sampling of x_t:

$$\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \overline{\alpha}_t}\boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- Then substituting the equivalent value for $\mathbf{x}_0$ gives

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1 - \overline{\alpha}_t}} \boldsymbol{\epsilon} \right)$$

- Since our model knows $\mathbf{x}_t$ at inference and needs to approximate $\tilde{\boldsymbol{\mu}}$, a good parameterization of $\boldsymbol{\mu}_\theta$ is

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

- So our model is now predicting $\boldsymbol{\epsilon}$ given $\mathbf{x}_t$ and the loss for fixed t becomes

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

- Tempting to drop the time scaling out front, so let's try it:

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

- This ends up working very well.

Ancestral Sampling

Algorithm 1 Training

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

Algorithm 2 Sampling

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

Why did we use $L_{\text{simple}}(\theta)$ instead of the log-likelihood maximizing loss?

- Empirical reason: Using $L_{\text{simple}}(\theta)$ results in better sample quality (and is easier to implement)

- Slightly more detailed reason: the simplified loss more heavily weights small times in denoising. This is important in maintaining image quality when sampling.

- Theoretical reason: this loss learns the score of perturbed distribution (reweighted based on variances).

$$\nabla \log q(\mathbf{x}_t \mid \mathbf{x}_0) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}$$

- Two interpretations of DDPMs: learning to remove noise or learning perturbed distributions.

Further Reading

- “Deep Unsupervised Learning using Nonequilibrium Thermodynamics” by Sohl-Dickstein, et al.
- “Denoising Diffusion Probabilistic Models” by Ho, et al.
- “Understanding Diffusion Models: A Unified Perspective” by Luo