Diffusion Models

Volodymyr Kuleshov

Cornell Tech

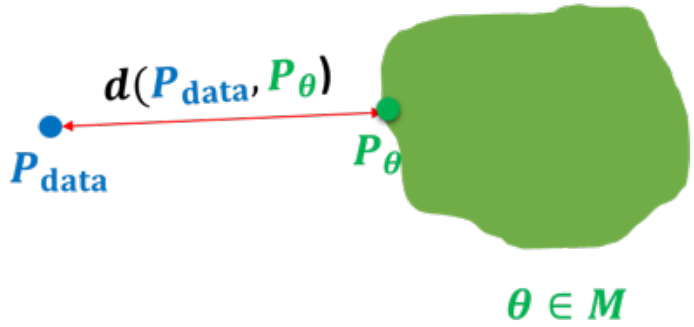
Lecture 13

Announcements

- Please make sure to have signed up for a presentation and reviewing slot **by the end of the week**.
- Please indicate which papers you will be presenting by EOD tomorrow.

Recap So Far

$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



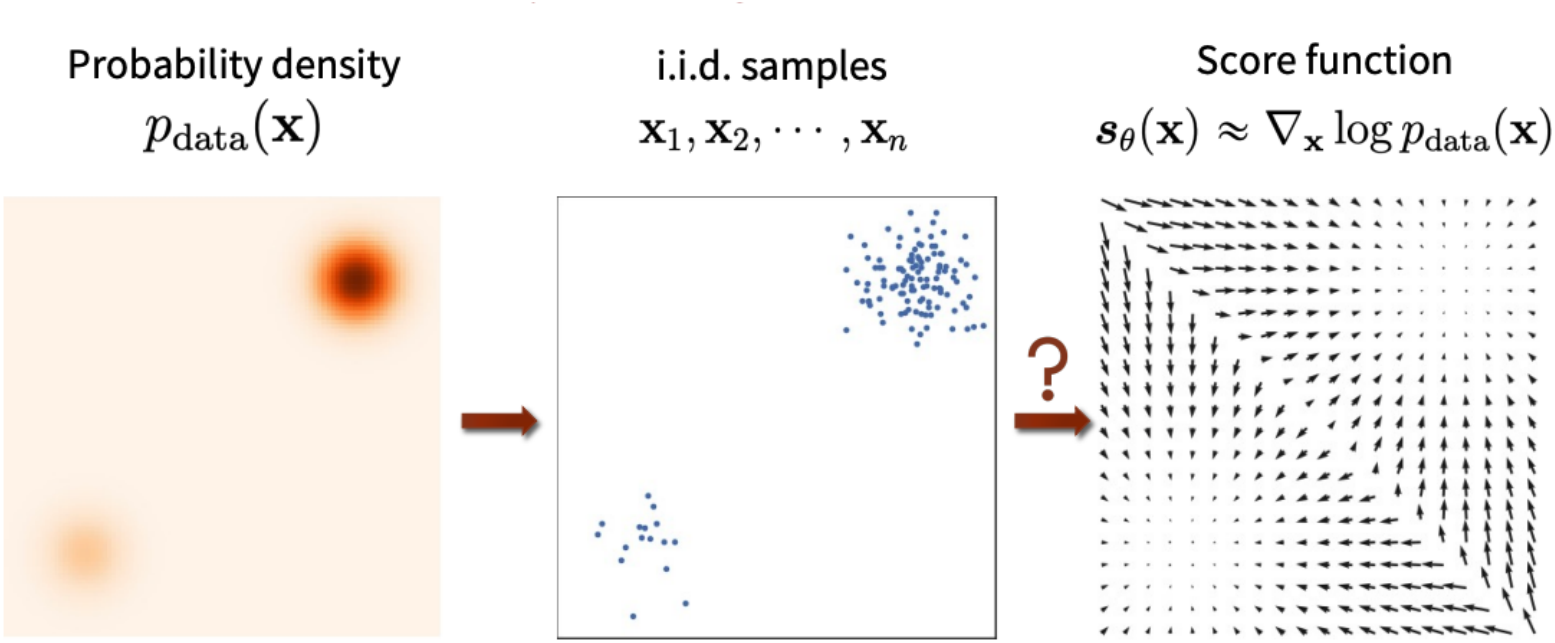
We have seen several generative model families so far:

1. Autoregressive, Latent Variable, Energy-Based Models, and GANs
   - Can perform generation, representation learning, outlier detection, etc.
   - Oftentimes need complex models (e.g., tractable Jacobians in flows)
   - In other cases are slow to train (MCMC) or unstable (GANs)
2. Score-Based Generative Models:
   - Produce high quality samples and have fast, stable training
   - No density estimation or representation learning

Can we connect score-based models and previous model families?

Score Function Estimation

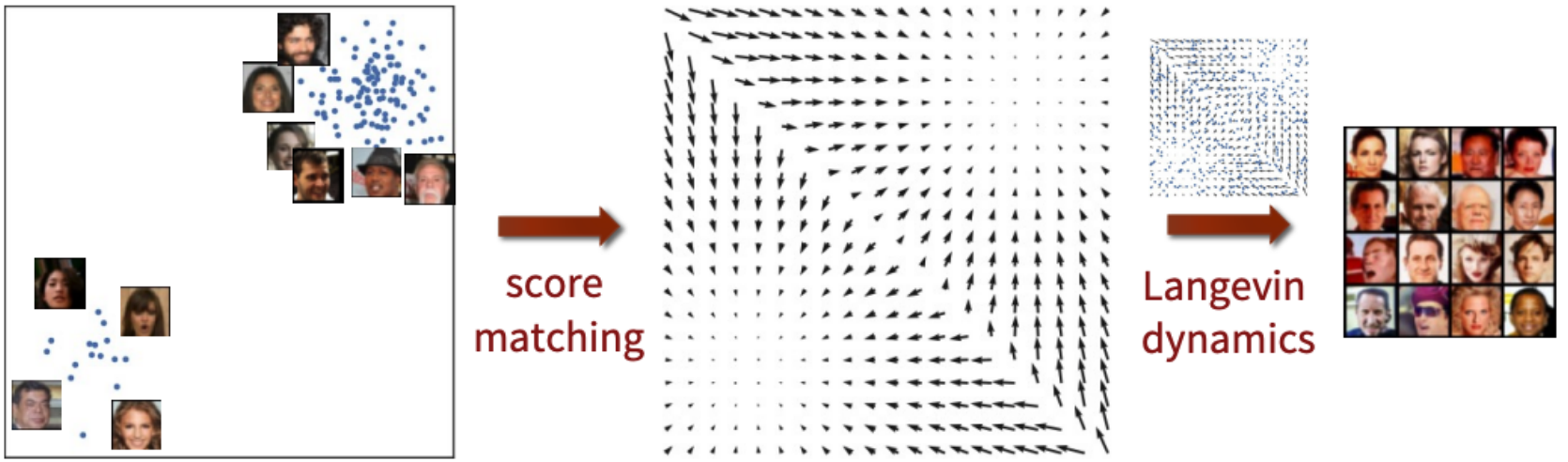
The idea of score function estimation is to learn a model of the score function $\nabla_x \log p_{\text{data}}(\mathbf{x})$ from a dataset of sample points.



- Given data $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, we want to estimate $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- Our model is a parameterized function $s_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$
- Minimizing the Fisher divergence ensures that $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \approx s_\theta(\mathbf{x})$
- We approximate it with sliced or denoising score matching

Sampling Using Score Functions

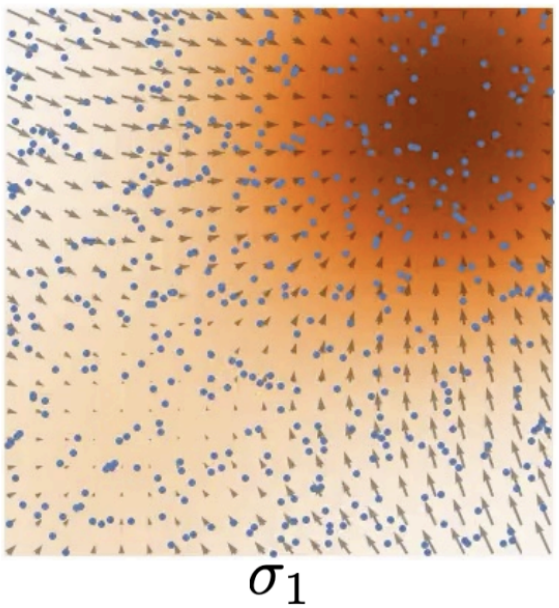
Once we learn $s_\theta(\mathbf{x}) \approx \nabla_x \log p_\theta(\mathbf{x})$ from a set of samples, we then generate new $\mathbf{x}$ via Langevin dynamics (gradient ascent on $\log p_\theta(\mathbf{x})$)

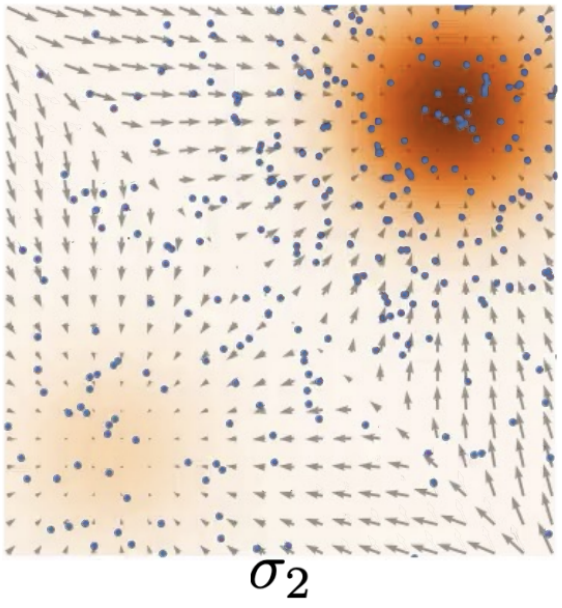


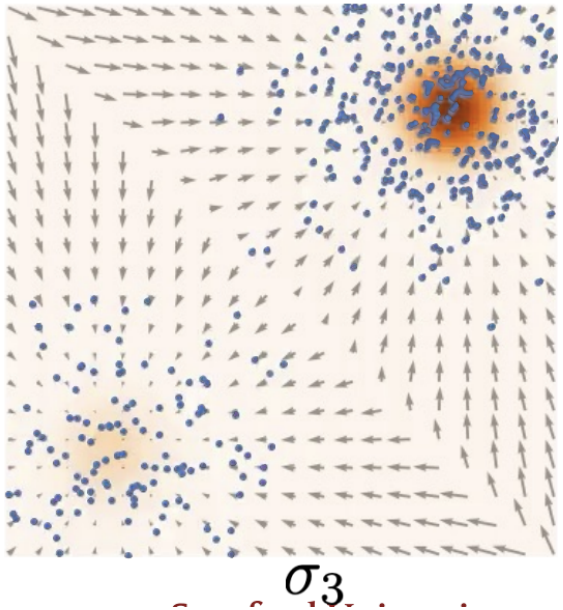
Because, it's hard to learn $s_\theta(\mathbf{x}) \approx \nabla_x \log p_\theta(\mathbf{x})$ in low-density regions, we use *annealed* Langevin dynamics with a noise-conditional network $s_\theta(\mathbf{x}, \sigma)$.

Annealed Langevin Dynamics: Intuition

We can run Langevin Dynamics with decreasing noise levels. At each new noise level, we start where we left off with the previous noise level.







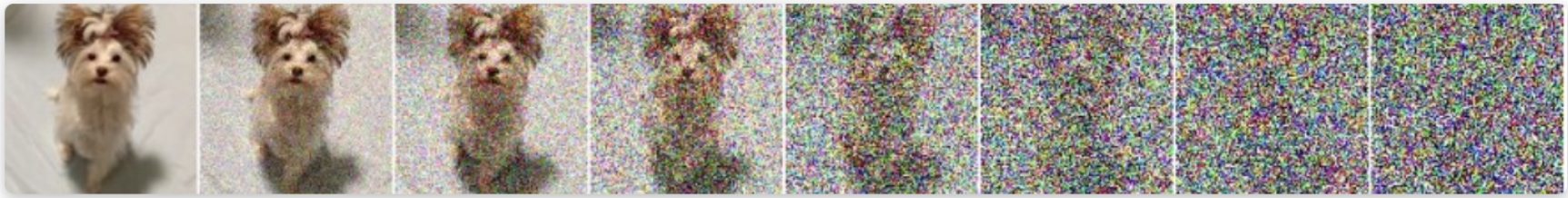
Lecture Outline

1. Diffusion Processes and Models
   - Forward Noising Process
   - Backward Denoising Process
2. Learning Diffusion Models
   - Variational Inference
   - Noise Parameterization
3. Applications
   - Sampling
   - Density Estimation
   - Latent Variable Inference
   - Latent Diffusion

Figures and contents adapted from Lily Weng, Yang Song, Stefano Ermon

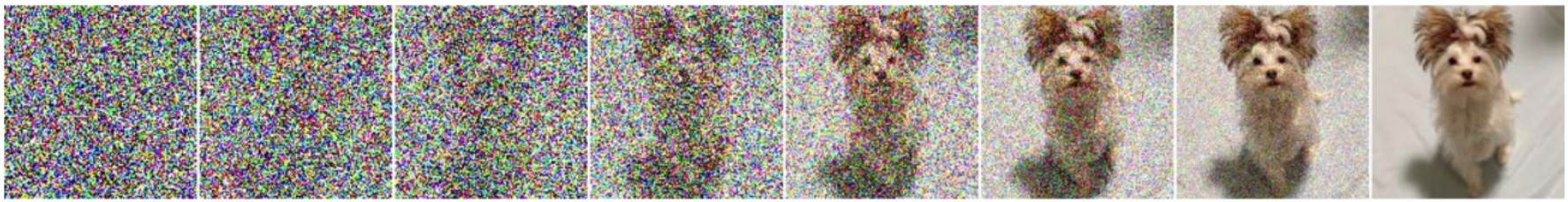
Score-Based Generative Models: Iterative Refinement

Score-based generative models learn to model distributions with increasing levels of Gaussian noise:



Perturbing an image with multiple scales of Gaussian noise.

When sampling using annealed Langevin dynamics, we undo this noise: the image gradually appears out of Gaussian noise.

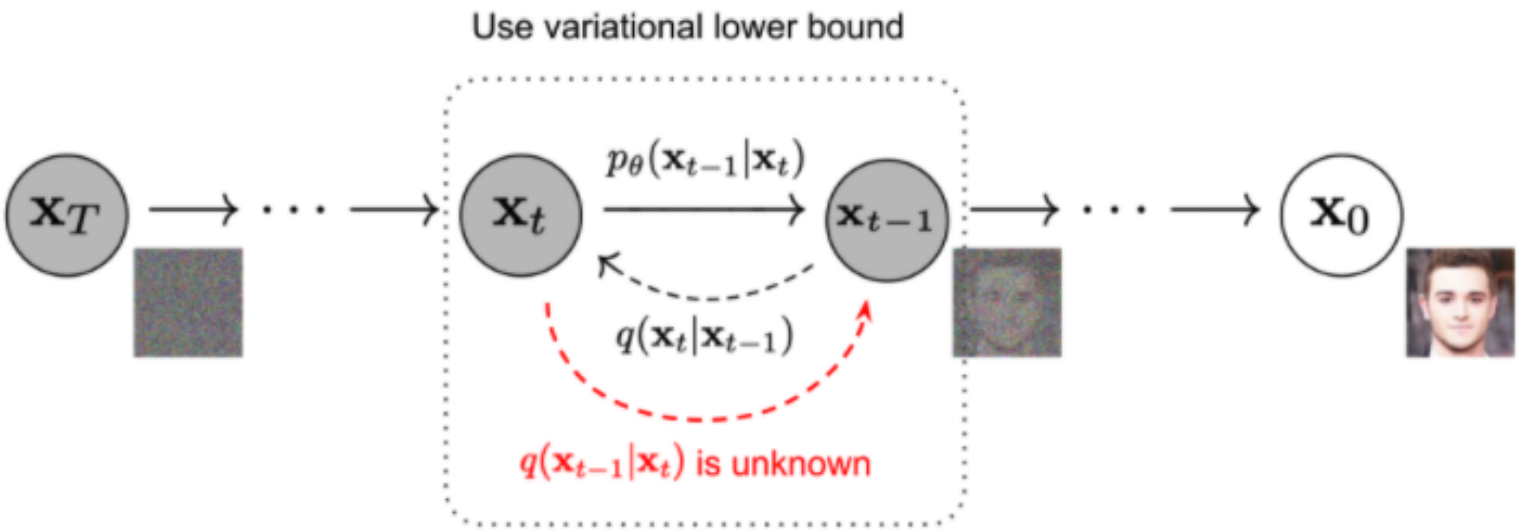


Diffusion models explicitly encode this process using the concepts of forward and backward diffusion.

Diffusion Models: Intuition

The intuition behind diffusion models is to define a process for gradually turning data to noise, and learning the inverse of this process.

- Forward (or noising) process $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is defined by user and is analogous to annealing in score-based models.
- Backward process $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ undoes the noise process and is learned from data.

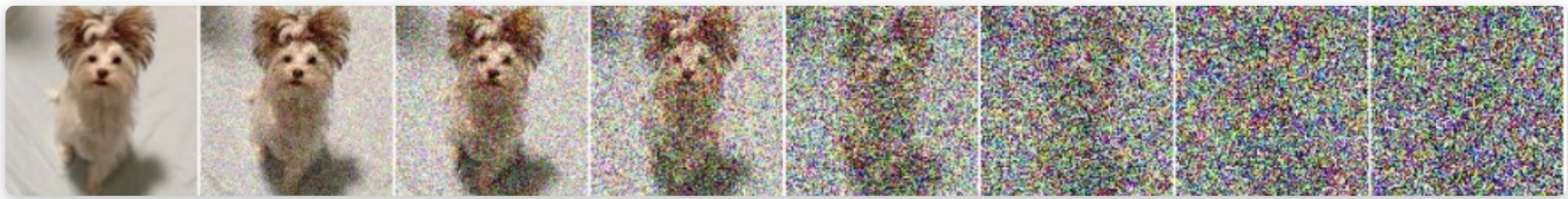


- This results in iterative & coarse-to-fine generation (very effective!)

Forward Diffusion (Noising) Process

To define a diffusion model, we start by defining the noising or forward diffusion process.

- We start with data samples $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- We run a Markov chain that gradually adds noise to the data, producing sequence of increasingly noisy samples $\mathbf{x}_1, ...\mathbf{x}_T$.



Perturbing an image with multiple scales of Gaussian noise.

- At each step t, we sample x_t from the following Markov operator:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, 1 - \alpha_t\mathbf{I})$$

  where $\alpha_t \in (0, 1)$, $\alpha_t \to 0$.
- At the end, $\mathbf{x}_T$ is a standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$

Computing Marginals and Connections to Annealing

The distribution of $q(\mathbf{x}_1, ...\mathbf{x}_T)$ has analytically computable marginals. Observe that:

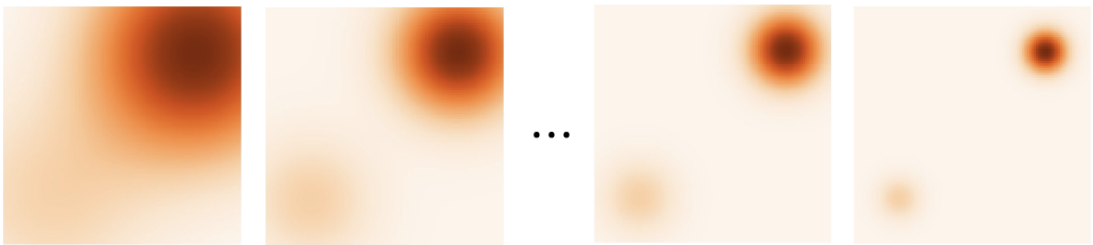
$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t && \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_t \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} && \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})
\end{aligned}$$

It follows that the marginal distribution $q(\mathbf{x}_t|\mathbf{x}_0)$ equals:

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

Thus, diffusion yields noised distributions as in score-based models:

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



Backward Diffusion (Denoising) Process

Next, we want to learn a model that undoes the noising process.

- The ideal denoising process is the inverse of the above Markov chain. At time t we sample from $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$
- We don't know this process, but we will try to learn it from data
- We define a model $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ where $\mathbf{x}_1, ...\mathbf{x}_T$ are latent variables and where

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

- To generate from this model, we sample noise $\mathbf{x}_T$, and sample from $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ until we get an image $\mathbf{x}_0$.

Next, we will look at how to learn p from q.

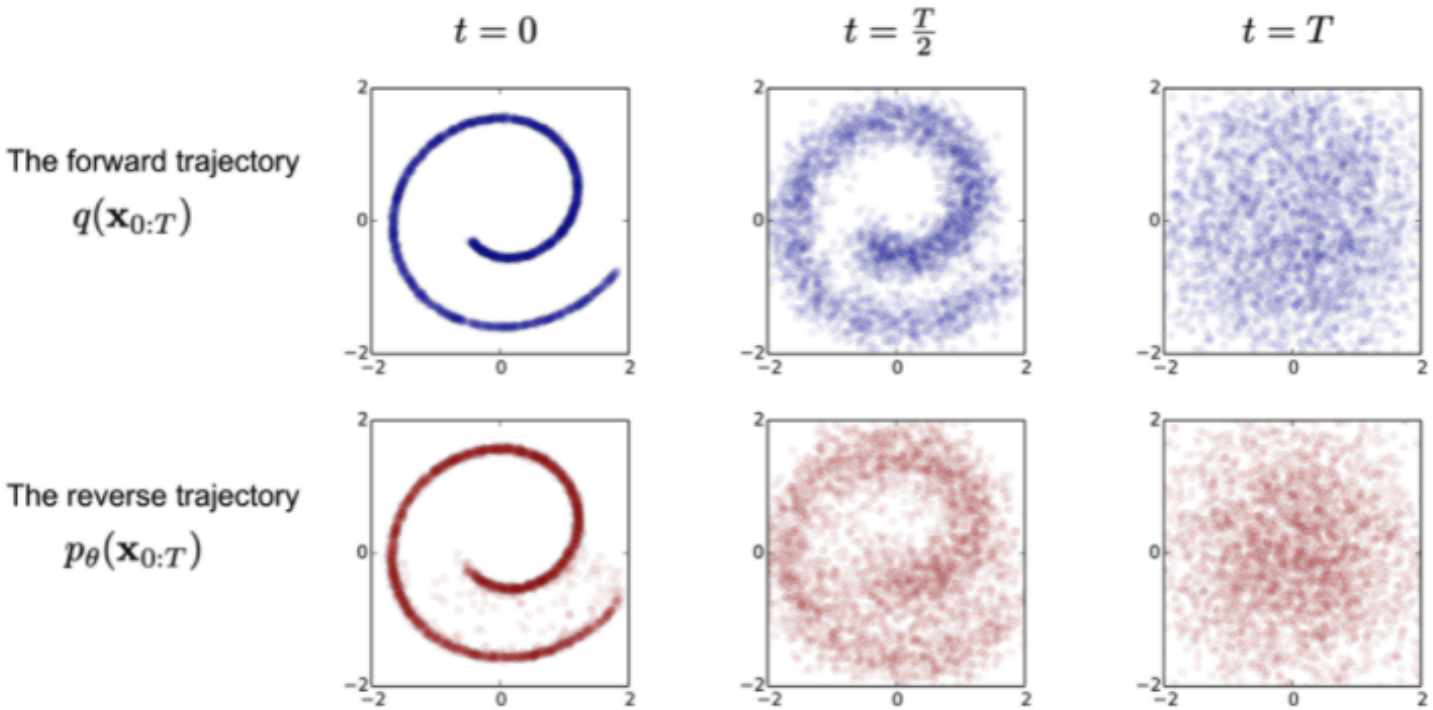
Lecture Outline

1. Diffusion Processes and Models
   - Forward Noising Process
   - Backward Denoising Process
2. Learning Diffusion Models
   - Variational Inference
   - Noise Parameterization
3. Applications
   - Sampling
   - Density Estimation
   - Latent Variable Inference
   - Latent Diffusion

Figures and contents adapted from Lily Weng, Yang Song, Stefano Ermon

Learning Backward Diffusion from Forward Diffusion

Here is an illustration of forward (top) and backward (bottom) diffusion processes on a 2D spiral dataset:



Our goal is to learn $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ from $q(\mathbf{x}_t|\mathbf{x}_{t-1})$. We will use as our objective the KL divergence:

$$D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0).$$

Learning Diffusion Models via Variational Inference

Our approach will be to apply variational inference and optimize an evidence lower bound on the likelihood $\log p(\mathbf{x}_0)$ of a data point $\mathbf{x}_0$:

$$\begin{aligned}
\log p_\theta(\mathbf{x}_0) &\geq \log p_\theta(\mathbf{x}_0) - D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= \log p_\theta(\mathbf{x}_0) - \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\right] \\
&= \log p_\theta(\mathbf{x}_0) - \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\right] \\
&= -\mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right]
\end{aligned}$$

This looks a lot like the learning objective of a VAE:

- By optimizing this bound, we maximize $p(\mathbf{x}_0)$ and minimize $D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)$.
- Unlike in VAEs, q is hand-crafted and fixed, we only optimize over p_θ.

Transforming the ELBO Bound

Both p and q have special (Markov) structure that we can use to transform the ELBO into a sum of simpler terms:

$$L_{\text{VLB}} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] = \mathbb{E}_q\Big[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big]$$

$$= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big]$$

$$= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \Big(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}\Big) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big]$$

$$= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big]$$

$$= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\Big]$$

$$= \mathbb{E}_q[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^T \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]$$

A More Convenient ELBO

The objective $L_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0$ contains three types of terms:

- The prior loss
$$L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))$$
compares the final $\mathbf{x}_T$ and is often zero by construction
- The reconstruction term
$$L_0 = -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)$$
is the probability of the true $\mathbf{x}_0$ given the model's "best guess" $\mathbf{x}_1$.
- The key components are the diffusion loss terms
$$L_t = D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1$$
which measure whether the learned backward process $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$ looks like the real backward process $q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0)$.

Parameterizing the Model

Let's focus on the diffusion loss terms

$$L_t = D_{\mathrm{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1.$$

Recall that $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$ where $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$.
Using our definition of $q(\mathbf{x}_{t+1}|\mathbf{x}_t)$ and Bayes' rule, we can show that:

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

For some $\tilde{\mu}, \tilde{\beta}$ (see next slide). Next, we choose a $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ of the form:

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

where $\mu_\theta(\mathbf{x}_t)$ is parameterized by a neural net. Minimizing L_t means fitting $\mu_\theta(\mathbf{x}_t)$ to $\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0)$, e.g., using gradient descent on θ.

Noise Parameterization

The above approach can be further improved by leveraging the structure of

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

One can show that $\tilde{\boldsymbol{\mu}}_t := \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0)$ equals

$$\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big),$$

where $\boldsymbol{\epsilon}_t$ is the noise added to $\mathbf{x}_0$ to obtain $\mathbf{x}_t$. We can set our model to

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

where $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ is a model of the noise $\boldsymbol{\epsilon}_t$.

With this parameterization, minimizing L_t means predicting and removing the noise $\boldsymbol{\epsilon}_t$ from a corrupted $\mathbf{x}_t$.

Learning the Noise

Specifically, using the above parameterization, L_t reduces to:

$$\begin{aligned} L_t &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[C_1 \cdot \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2\Big] \\ &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[C_2 \cdot \|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2\Big] \\ &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[C_2 \cdot \|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\Big], \end{aligned}$$

where $C_1, C_2 > 0$ hide constants. We optimize the sum of L_t's as follows:

- Sample a datapoint $\mathbf{x}_0$
- Sample a time step t uniformly from $1, 2, ..., T$.
- Sample noise $\epsilon \sim \mathcal{N}(0, I)$. Generate noisy $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t$.
- Take a gradient step on $\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2$

and we repeat until convergence.

Denoising Score Matching vs. Diffusion Models

Let $q_\sigma(\tilde{\mathbf{x}})$ be a noised version the data distribution, e.g., Gaussian noise:

$$\tilde{\mathbf{x}} = \mathbf{x} + \sigma \cdot \epsilon \text{ for } x \sim p(\mathbf{x}), \epsilon \sim \mathcal{N}(0, I)$$

Denoising score matching (Vincent, 2011) approximates the Fisher divergence between s_θ and $q_\sigma(\tilde{\mathbf{x}})$ as:

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

- When we use Gaussian noise, $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{\mathbf{x}-\tilde{\mathbf{x}}}{\sigma^2} = \frac{\epsilon}{\sigma}$.
- The score model $s_\theta(\tilde{\mathbf{x}})$ and the the denoiser $\epsilon(\tilde{\mathbf{x}})$ are related:

$$s_\theta(\tilde{\mathbf{x}}) \approx \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{\epsilon}{\sigma} \approx \frac{\epsilon(\tilde{\mathbf{x}})}{\sqrt{1-\bar{\alpha}_t}}$$

Lecture Outline

1. Diffusion Processes and Models
   - Forward Noising Process
   - Backward Denoising Process
2. Learning Diffusion Models
   - Variational Inference
   - Noise Parameterization
3. Applications
   - Sampling
   - Density Estimation
   - Latent Variable Inference
   - Latent Diffusion

Figures and contents adapted from Lily Weng, Yang Song, Stefano Ermon

Ancestral Sampling

Given a trained model $p_\theta(\mathbf{x}_{t-t}|\mathbf{x}_t)$ we can generate x_0 by performing ancestral sampling:

$$x_{t-1} \sim p_\theta(\mathbf{x}_{t-t}|\mathbf{x}_t) \text{ for } t = T, T-1, ..., 1$$

Recall that $\mathbf{x}_{t-1} = \boldsymbol{\mu}_\theta(\mathbf{x}_t) + \sigma_t \cdot \mathbf{z}$ for $\mathbf{z} \sim \mathcal{N}(0, I)$ and

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

where $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ is a model of the noise $\boldsymbol{\epsilon}_t$. Thus, we have:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big) + \sigma_t \cdot \mathbf{z}$$

for $t \in \{T, T-1, ..., 1\}$.

Ancestral Sampling vs. Langevin Dynamics

Recall that in Langevin dynamics we repeatedly perform the update:

$$\mathbf{x}_{\text{new}} = \mathbf{x} + \frac{\alpha_t}{2}\nabla_x \log p_\theta(\mathbf{x}) + \sqrt{\alpha_t}\epsilon_t.$$

Recall also that the score model $s_\theta(\tilde{\mathbf{x}})$ and the the denoiser $\boldsymbol{\epsilon}(\tilde{\mathbf{x}})$ are related:

$$s_\theta(\tilde{\mathbf{x}}) \approx \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{\epsilon}{\sigma} \approx \frac{\boldsymbol{\epsilon}(\tilde{\mathbf{x}})}{\sqrt{1-\bar{\alpha}_t}}.$$

pause Since the sampling process for a diffusion model is

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \cdot \mathbf{z}$$

it can be viewed as a form of (annealed) Langevin dynamics.

Samples from Diffusion Models

Diffusion model produce state-of-the-art samples (at least as good as GANs), but are much more faster to train.

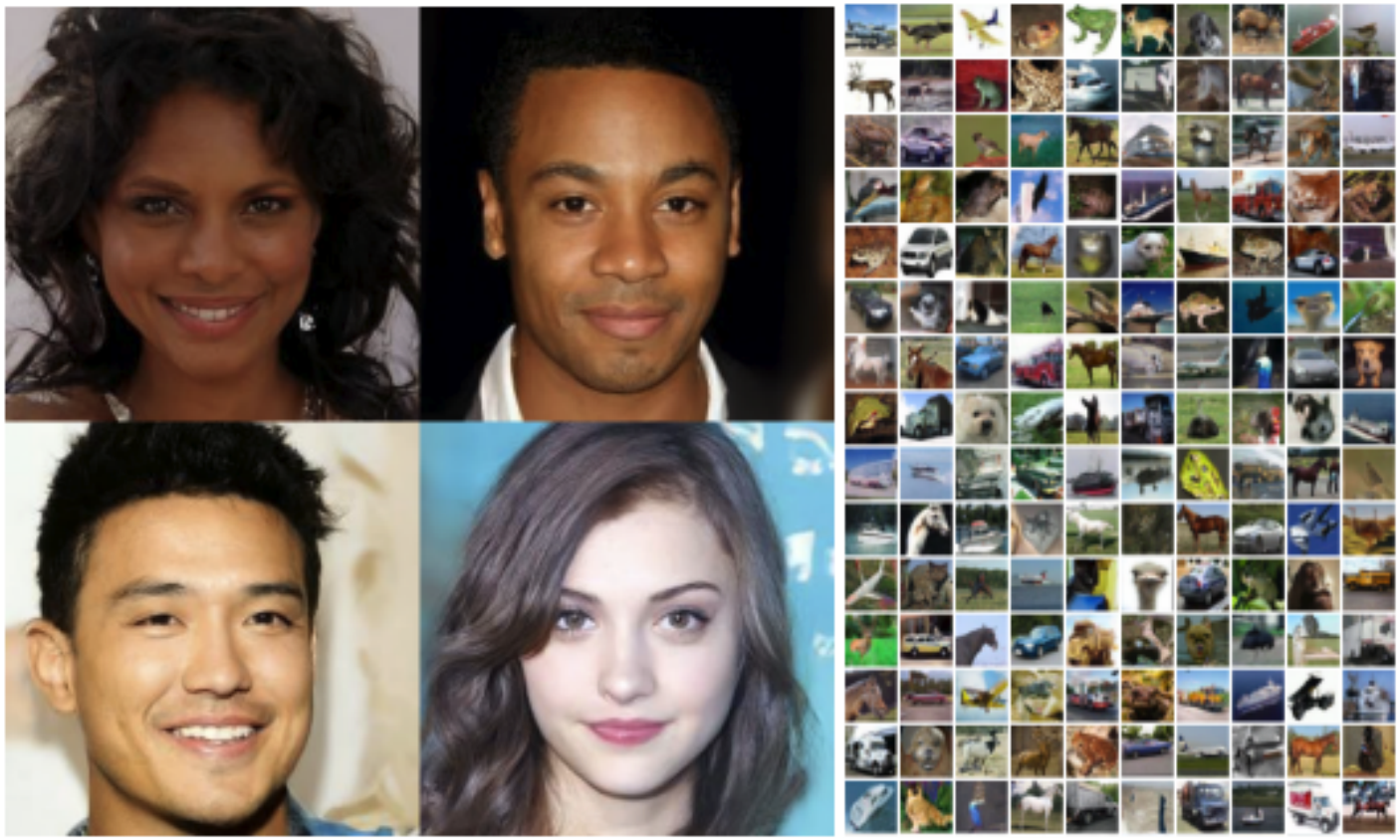


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

Their main drawback relative to GANs is slower sampling speed.

Conditional Samples from Diffusion Models

One can also condition the sampler $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$ on external data $\mathbf{y}$ at training time (e.g., the embedding of a sentence).

Diffusion models are widely used as decoders in text-to-image systems like DALLE2, Stable Diffusion, and Midjourney.

Conditional Sampling and Controllable Generation

Suppose we know a process $p(\mathbf{y}|\mathbf{x})$ mapping data $\mathbf{x}$ to auxiliary $\mathbf{y}$ (e.g., labels, noised images). We seek to model $p(\mathbf{x}|\mathbf{y})$ (a form of inverse).

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}).$$

Score-based models are naturally suited to this task:

- We can learn $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ unconditionally.
- The $\nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x})$ is already known (e.g., it's a classifier)
- Thus, we can sample from $\nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y})$ using Langevin dynamics.

Classifier-Free Guidance trains $\nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y})$ and $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ directly when $(\mathbf{x}, \mathbf{y})$ is available at train time (same model, with dropout on $\mathbf{y}$). At inference time, weigh the two. See Ho et al. 2022.

Score-Based & Diffusion Models: Conditional Samples

Here is the result of generating class-conditional samples:

Other examples include: conditioning colored images on monochrome images, noisy MRI or CT on clean data, computer art applications, etc.

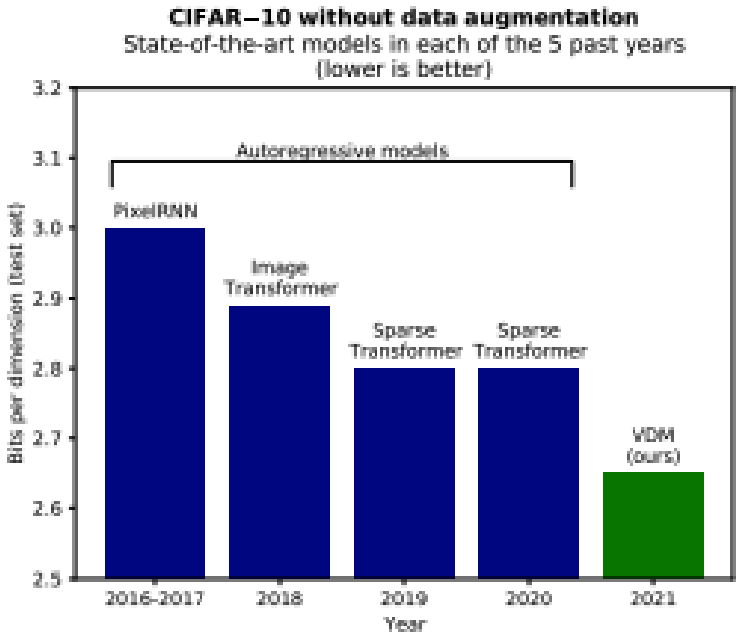
Likelihood Maximization with Diffusion Models

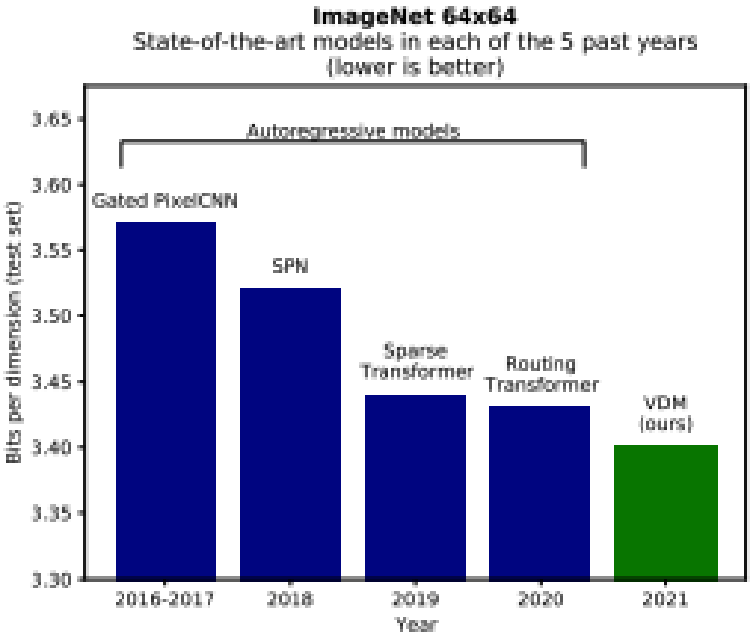
Recall that diffusion models optimize a lower bound on the log-likelihood:

$$\log p_\theta(\mathbf{x}_0) \geq -\mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] = -L_{\text{LVB}}$$

Variational diffusion models (Kingma et al. 2021) introduce:

- A finer discretization ($T \to \infty$) and a learned α_t schedule.
- Fourier features for $\mathbf{x}_t$ in $\epsilon(\mathbf{x}_t, t)$ to improve prediction of $\mathbf{x}_{t-1}$.





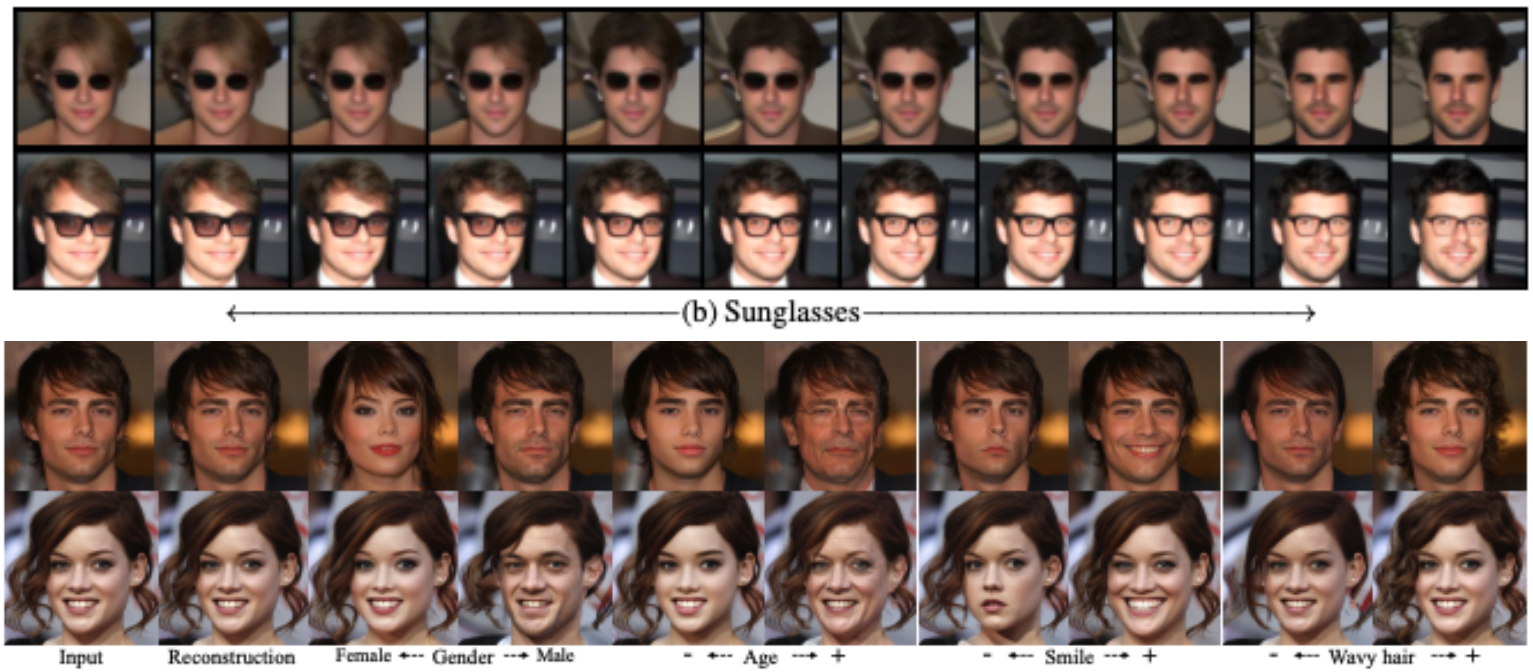
Latent Variable Inference with Diffusion Models

Diffusion models can be augmented with low-dimensional latents $\mathbf{z}$:

$$\log p_\theta(\mathbf{x}_0) \geq \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_0, \mathbf{z})}{q(\mathbf{z}|\mathbf{x}_0)}\right] \geq \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_{0:T}, \mathbf{z})}{q(\mathbf{x}_{1:T}, \mathbf{z}|\mathbf{x}_0)}\right]$$

where we applied the ELBO twice. We sample from $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z})$.

The resulting model uses $\mathbf{z}$ to encode semantic information and $\mathbf{x}_T$ to encode details (e.g., textures) (Preechakul et al., 2022; Wang et al., 2023)



Diffusion Models: Pros and Cons

Pros:

1. Very high quality samples (matching GANs) with a stable objective
2. Effective controllable generation (with and without latents)
3. Also state-of-the-art for density estimation

Cons:

1. Sampling is still quite slow (need to run MCMC chain)
2. Works less well on discrete data (active research area)