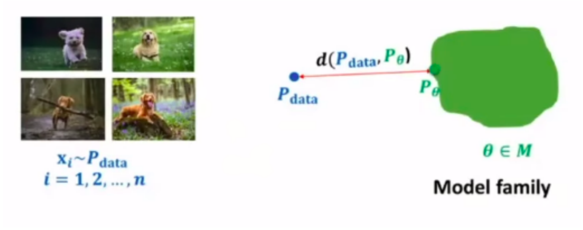
Diffusion Models

Kavindu Piyumal

March 1, 2024

Outline

Generative Models



Generative models learn a probabilistic distribution $P(x)$ over x so that we can perform tasks such as;

- **Generation**: If we sample $X_{new} \sim p(x), x_{new}$ should look like a sample from the training dataset (sampling)
- **Representation Learning**: We should be able to learn what these images have in common (features)

Any generative model consists of 3 main components:

1. **A model**: we specify it in its parametric form (e.g: Gaussian Distribution)
2. **A learning objective**: such as MLE
3. **An optimization algorithm**: such as variational inference, MCMC

Generative Model Families

There are several generative model families;

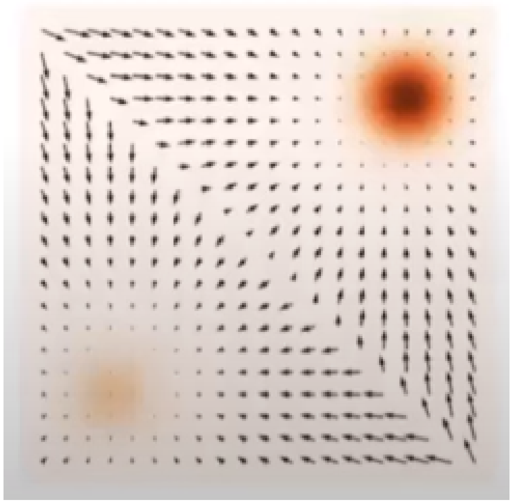
1. Autoregressive, Latent Variable Models, GANs:
   - Can perform generation, representation learning, etc.
   - Often requires complex models as we use MLE as our training objective
2. Energy-Based models:
   - Make less modeling assumptions
   - Maximum likelihood training with MCMC can be slower
3. Score-Based Generative Models:
   - Produce high quality samples and have fast, stable training
   - No density estimation or representation learning
4. Diffusion Models:
   - Produce high quality samples and stable training
   - SOTA for density estimation

Score-Based Models

Given a probability density $p_\theta(\mathbf{x})$, its (Stein) score function is defined as

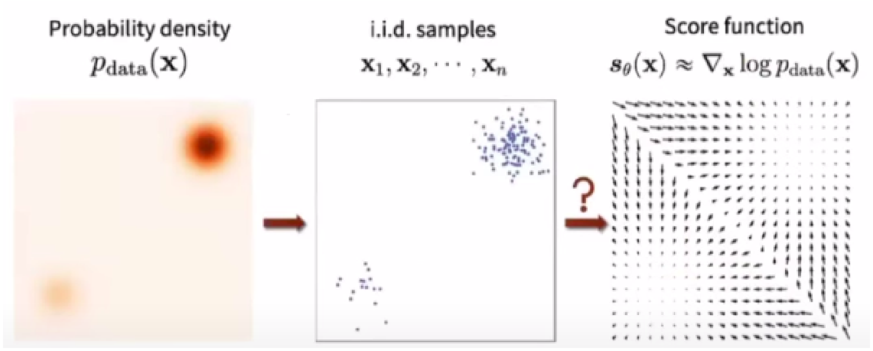
$$\nabla_{\mathbf{x}}\, log\ p_\theta(\mathbf{x})$$

- This is the gradient of the density with respect to the input.
- The score function points in the direction of increasing model probability



Score Function Estimation

The idea of score function estimation is to learn a model $s_\theta(\mathbf{x}) : \mathcal{R}^d \to \mathcal{R}^d$ of the score function from a dataset of sample points.



Probability density
$p_{\text{data}}(\mathbf{x})$
i.i.d. samples
$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$
?
Score function
$s_\theta(\mathbf{x}) \approx \nabla_\mathbf{x} \log p_{\text{data}}(\mathbf{x})$

Our objective is that $\nabla_\mathbf{x}\, log\, p_{data} \approx s_\theta(\mathbf{x})$

Score Matching

We use the Fisher divergence to compare data and model scores.

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

But the problem is that we **Do Not** know $p_{data}(\mathbf{x})$!!. However, via a change of variables trick, we can obtain an equivalent trick (Hyvarinen, 2005)

$$\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\left[\frac{1}{2}||s_\theta(\mathbf{x})||_2^2 + tr(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}))\right]$$

Note that we still have an expectation over the data coming from the data distribution, but we can use the Monte-Carlo to approximate it given a dataset.

Denoising Score Matching

But it turns out that calculating the $tr(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}))$ is computationally expensive.
One way to address this is to use something called denoising score-matching.
Suppose that we have access to a noised version $q_\sigma(\widetilde{\mathbf{x}})$ of the clean data $\mathbf{x} \sim p(\mathbf{x})$.
For example, we can add Gaussian noise to $\mathbf{x}$:

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

While it is hard to learn the score function for clean data, there is an efficient technique of learning the score function for corrupted data and reducing σ gradually to get back the cleaner version.

Denoising Score Matching Continues ...

The denoising score matching approximates the Fisher divergence between s_θ, and $q_\sigma(\widetilde{\mathbf{x}})$ as:

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

Note that $\nabla_{\widetilde{\mathbf{x}}}\, log\; q_\sigma(\widetilde{\mathbf{x}}|\mathbf{x})$ is easy to compute, e.g., $\nabla_{\widetilde{\mathbf{x}}}\, log\; q_\sigma(\widetilde{\mathbf{x}}|\mathbf{x}) = -\frac{\mathbf{x}-\widetilde{\mathbf{x}}}{\sigma^2}$ with Gaussian noise. The algorithm for score-based models can be noted below:

- Given an initial set of parameters θ_t, for $t = 1, 2, \ldots$
  1. Sample a batch of training points $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$
  2. Sample a batch of perturbed datapoints $\{\widetilde{\mathbf{x}}_\mathbf{1}, \widetilde{\mathbf{x}}_\mathbf{2}, \ldots, \widetilde{\mathbf{x}}_\mathbf{n}\} \sim q_\sigma(\widetilde{\mathbf{x}})$
  3. Compute the gradient $\nabla_\theta J(\theta)$ of the objective, where

     $$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left[\|\nabla_\theta \log q_\sigma(\widetilde{\mathbf{x}}|\mathbf{x}) - s_\theta(\widetilde{\mathbf{x}})\|_2^2\right]$$

     for some q_σ, or is an average of the two
  4. Take a gradient step $\theta_t = \theta_{t-1} - \alpha \cdot \nabla_\theta J(\theta_{t-1})$

Sample Generation: Langevin Dynamics

This is an MCMC sampling process that allows us to sample from $p_\theta(\mathbf{x})$ using only its score $\nabla_x log\ p_\theta(\mathbf{x})$

- Initialize $\mathbf{x}_0$ from a noise distribution
- For steps $t = 1, 2, \ldots, T$
  - Sample a noise variable $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - $\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\alpha_t}{2} \nabla_x\ log\ p_\theta(\mathbf{x}) + \sqrt{\alpha_t}\epsilon_t$

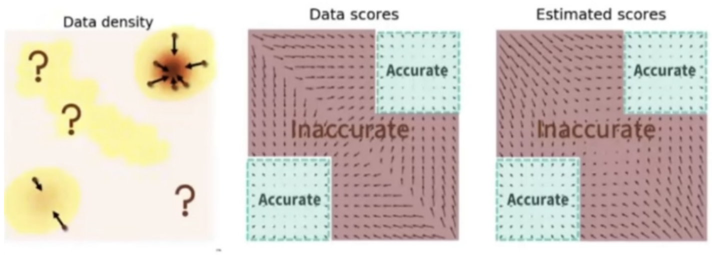
This process can be seen as gradient ascent on $log\ p_\theta(\mathbf{x})$. Note that we can use this process for sampling from a score function $s_\theta(\mathbf{x}) \approx \nabla_x log\ p_\theta(\mathbf{x})$ by plugging s_θ into the above algorithm.

Manifold Hypothesis

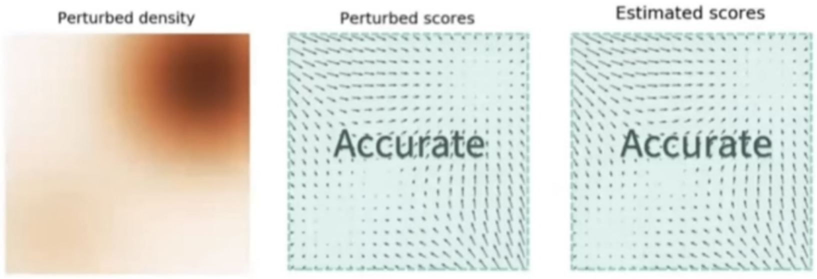
The above sampling approach faces challenges arising from the manifold structure of the data.

- The manifold approach states that most of our data lives on a low-dimensional manifold in a high-dimensional space.
- This means that the true score function will be undefined in most places of the high-dimensional space.
- This is a problem as $\nabla_{\mathbf{x}}\, log\ p_\theta(\mathbf{x})$ is not defined where $p_{data}(\mathbf{x}) = 0$
- One solution to this problem is to add Gaussian noise to the data.
- The score function is now well-defined everywhere but it's defined on a different distribution
- There is a tradeoff: smooth more and get a better score vs. smooth less and better approximate the data distribution

Learning and Sampling Scores in Low Density Regions



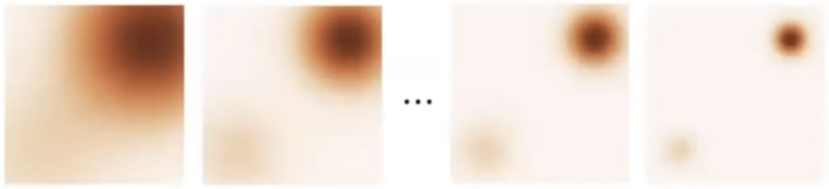
- When we start with random noise, it can come from inaccurate regions which would not allow us to reach the high-density regions.
- As a solution, we add Gaussian noise to the data to spread the high density area.
- High noise provides useful directional information for Langevin dynamics



Multi-Scale Noise

But when we perturb the density, we are no longer able to approximate the true data density. As a solution, we use multi-scale noise levels

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

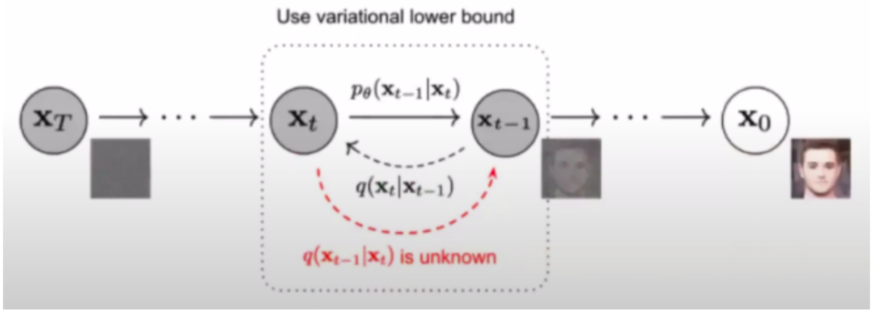


- We train using both small and large amounts of noise
- At the initial stages of sampling, we use larger noise
- We implement this strategy by training a score function model that is conditioned on the noise.
- The same model is trained for all σ and the parameters are shared across noise levels.

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 $q(\mathbf{x_t}|\mathbf{t_{t-1}})$ is defined by the 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 the data.
- The results in iterative & coarse-to-fine generation.

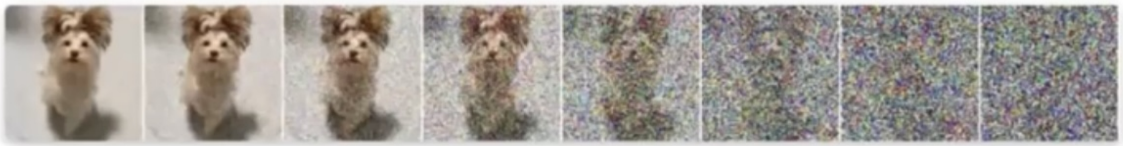

Use variational lower bound
$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is unknown

Forward Process

- We start with data samples $\mathbf{x}_0 \sim q(x_0)$
- We run a Markov chain that gradually adds noise to the data, producing a sequence of increasing noise samples $\mathbf{x_1}, \ldots, \mathbf{x_T}$
- 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 $\mathcal{N}(\mathbf{0}, \mathbf{I})$


Perturbing an image with multiple scales of Gaussian noise.

Marginals and Connections to Annealing

The distribution of $q(\mathbf{x_1}, \dots, \mathbf{x_T})$ has analytically computable marginals. Observe that;

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_t \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\overline{\epsilon}_t \\
&= \dots \\
&= \sqrt{\overline{\alpha}_t}\mathbf{x}_0 = \sqrt{1-\overline{\alpha}_t\epsilon}
\end{aligned}$$

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

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

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

Backward Diffusion Process

The ideal denoising process is the reverse of the Markov chain. But we don't know this process so we have to learn it from the 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 $\mathbf{x}_1, \ldots, \mathbf{x}_T$ are latent variables and where

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

- 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 $\mathbf{x}_0$

Learning Backward Diffusion from Forward Diffusion

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 KL divergence as our objective:

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

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

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

- By optimizing this bound, we maximize $p(\mathbf{x}_0)$ and minimize $D_{KL}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)||p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0))$.

Transforming the ELBO Bound

We can show that the ELBO bound transformed into the following three components.

1. The prior loss

$$L_T = D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p_\theta(\mathbf{x}_T))$$

   compares the final x_T and is often zero by construction

2. 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" $\mathbf{x}_1$.

3. The key components are the diffusion loss terms

$$D_{KL}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)||p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0))\ for\ 1 \leq t \leq T-1$$

   which measure whether the learned backward process $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$ looks like the real backward process $q_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0)$

Parameterizing the Model

$$D_{KL}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)||p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0))\ for\ 1 \leq t \leq T-1$$

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

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

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

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

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

Noise Parameterization

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

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t), \widetilde{\beta}_t\mathbf{I})$$

One can show that $\widetilde{\mu}_t := \widetilde{\mu}(\mathbf{x}_t, \mathbf{x}_0)$ equals using Tweedie's Formula;

$$\begin{aligned}\widetilde{\mu}_t &= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\nabla \log p(\mathbf{x}_t)\right) \text{ (This resembles the score function)}\\ &= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_t\right)\end{aligned}$$

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

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\widetilde{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

Where $\epsilon_\theta(\mathbf{x}_t, t)$ is a model of the noise ϵ_t. With this parameterization, minimizing L_t means predicting and removing the noise ϵ_t from a corrupted $\mathbf{x}_t$.

Learning the Noise

Specifically, using the above parameterization, L_t reduces to:

$$\begin{aligned} L_t &= \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[C_1 \cdot ||\widetilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] \\ &= \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[C_2 \cdot ||\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t)||^2\right] \\ &= \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[C_2 \cdot ||\epsilon_t - \epsilon_\theta(\sqrt{\widetilde{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\widetilde{\alpha}_t}\epsilon_t, t)||^2\right] \end{aligned}$$

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

- Sample a data point $\mathbf{x}_0$
- Sample a time step t uniformly from $1, 2, \ldots, T$
- Sample noise $\epsilon \sim \mathcal{N}(0, I)$. Generate noisy $\mathbf{x}_t = \sqrt{\widetilde{\alpha}}\mathbf{x}_0 + \sqrt{1-\widetilde{\alpha}_t}\epsilon_t$
- Take the gradient step on $||\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t)||^2$

and we repeat until convergence

Ancestral Sampling

Given a trained model $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we can generate x_0 by performing ancestral sampling:

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

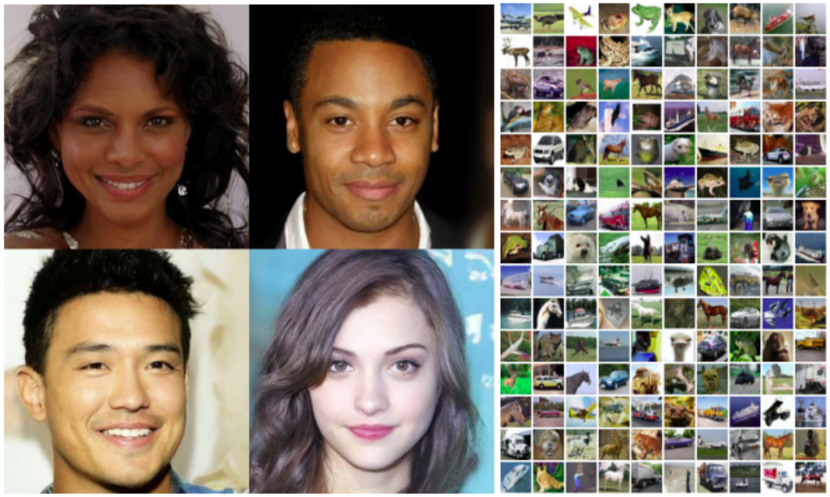


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

Diffusion Models: Pros and Cons

Pros:

1. Very high-quality samples with a stable objective
2. Effective controllable generation
3. State of the art for density estimation

Cons:

1. Sampling is still quite slow
2. Works less well on discrete data

Variations of Diffusion Models

Say that we want to control the generation process, i.e., rather than sampling from the marginal, we want to sample from the conditional. Using Bayes' rule for score functions, we can get an approximation for this.

$$
\begin{aligned}
p(x|y) &= \frac{p(x)p(y|x)}{p(y)} \\
\nabla_x \log p(x|y) &= \nabla_x \log p(x) + \nabla_x \log p(y|x) - \nabla_x \log p(y) \\
&= \nabla_x \underbrace{\log p(x)}_{\text{Unconditional score}} + \nabla_x \underbrace{\log p(y|x)}_{\text{Forward model}}
\end{aligned}
$$

Now we can plug in different forward models for the same score model to get different variations of the base diffusion model.

- Text-conditioned image generation
- Image inpainting
- Image colorization etc.

References I

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[2] Volodymyr Kuleshov. *Deep Generative Models*. YouTube video. 2024. URL: https://youtube.com/playlist?list=PL2UML_KCiC0UPzjW9BjO-IW6dqliu9O4B&si=GcwH7jFDA_Cm3ww_.

[3] Yang Song and Stefano Ermon. “Improved Techniques for Training Score-Based Generative Models”. In: *CoRR* abs/2006.09011 (2020). arXiv: 2006.09011. URL: https://arxiv.org/abs/2006.09011.

[4] Yang Song et al. “Score-Based Generative Modeling through Stochastic Differential Equations”. In: *CoRR* abs/2011.13456 (2020). arXiv: 2011.13456. URL: https://arxiv.org/abs/2011.13456.