Diffusion networks

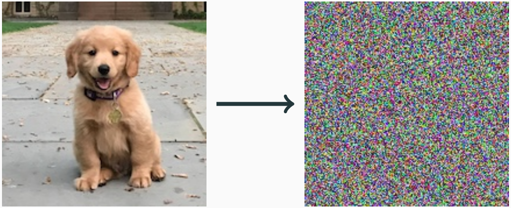
Quick maffs and foundations

Jeremi Levesque

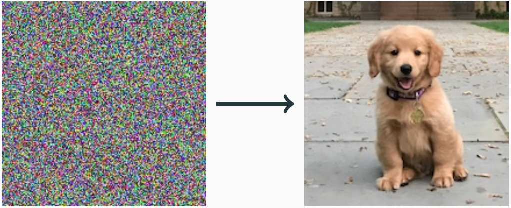
November 26, 2024

Université de Sherbrooke

Data generation is a long-standing challenge in machine learning.



Easy to do



Generative modelling

Variational Autoencoders (VAEs) are extensively used to project data into a probabilistic latent space from which we can sample to generate new data.

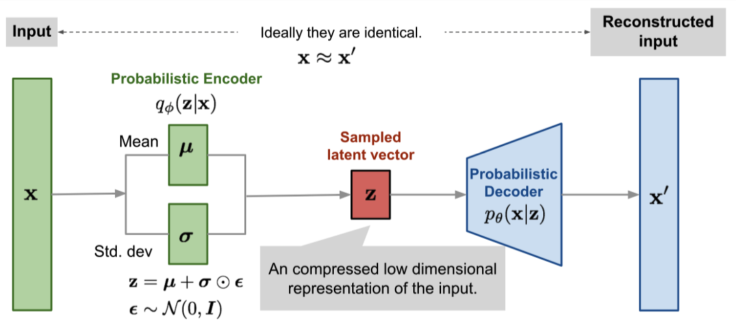


Figure 1: Weng, Lilian. (2018). From Autoencoder to Beta-VAE. https://lilianweng.github.io/posts/2018-08-12-vae/

Generative Adversarial Networks (GANs) have long been used to generate fake data:

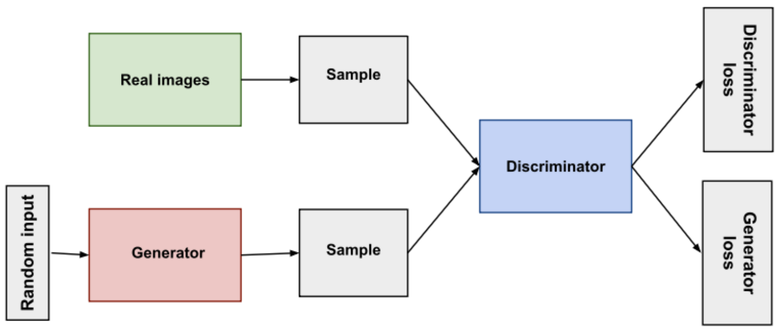


Figure 2: From Google advanced machine learning course on GANs. Consulted on Nov. 20 2024.

Outline

2 different perspectives to get somewhat similar conclusions

Score-based generation models

- Score matching
- Langevin dynamics
- Challenges
- Multiple noise perturbations
- Stochastic differential equations

Denoising diffusion probabilistic models

- Forward process
- Reverse process

Score-based generation models

Problem setting

Dataset $\{x_1, ..., x_n\}$

Real distribution (unknown): $p_d(x)$

We want to model another distribution $p_m(x; \theta)$ (θ being the model's parameters)

Objective

Real distribution (unknown): $p_d(x)$
Modelled distribution: $p_m(x;\theta)$

Find θ such that $p_m(x;\theta)$ is as close as possible to $p_d(x)$

Finding the weights

Maximum Likehood:

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \log p_m(x;\theta)$$

Since $p_m(x;\theta)$ is a **normalized** density function, we have:

$$p_m(x;\theta) = \frac{\tilde{p}_m(x;\theta)}{Z_\theta} \qquad \text{where } Z_\theta = \int \tilde{p}_m(x;\theta)dx$$

$\tilde{p}_m(x;\theta)$: unnormalized density function

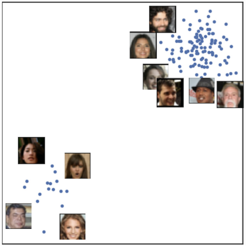
Z_θ: normalizing constant (intractable because of $\int$ on all data coming from real distribution)

Intuition for score-based generation models

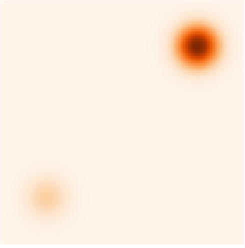
Instead of directly maximizing the likelihood, we find a θ such that the gradients of the model's log-likelihood are approx. the same as the gradients of the data distribution log-likelihood.

$$
\begin{aligned}
s_\theta(x) &= \nabla_x \log p_m(x;\theta) \\
&= \nabla_x \log\left(\frac{\tilde{p}_m(x;\theta)}{Z_\theta}\right) \\
&= \nabla_x \log \tilde{p}_m(x;\theta) - \cancel{\nabla_x \log Z_\theta} \\
&= \nabla_x \log \tilde{p}_m(x;\theta)
\end{aligned}
$$

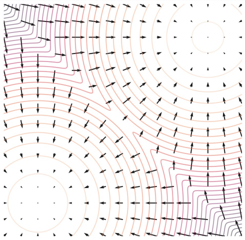
Data samples



Data density



Scores $s_\theta(x)$



To minimize the distance between the score and the score of the real data, we minimize the **Fisher divergence** between the two distributions:

Fisher divergence Doesn't require the two distributions to be normalized.

$$
\begin{aligned}
\hat{\theta}_{SM} &= \arg\min_{\theta} D_F(p_d(x)||p_m(x;\theta)) \\
&= \arg\min_{\theta} \frac{1}{2}\mathbb{E}_{p_d(x)}\left[\|\nabla_x \log p_d(x) - \nabla_x \log p_m(x;\theta)\|^2\right] \\
&= \arg\min_{\theta} \frac{1}{2}\mathbb{E}_{p_d(x)}\left[\|\nabla_x \log p_d(x) - s_\theta(x)\|^2\right]
\end{aligned}
$$

Score matching

However, we have no way of computing the ground truth score $\nabla_x \log p_d(x)$.

$$\hat{\theta}_{SM} = \arg\min_{\theta} \frac{1}{2} \mathbb{E}_{p_d(x)} \left[\left\| \underbrace{\nabla_x \log p_d(x)}_{\textit{intractable}} - s_\theta(x) \right\|^2 \right]$$

There's a derivation from [2] that allows to get to the following objective not involving the ground truth score:

$$\frac{1}{2} \mathbb{E}_{p_d(x)} \left[\left\| \nabla_x \log p_d(x) - s_\theta(x) \right\|^2 \right] \approx$$
$$\frac{1}{2} \mathbb{E}_{p_d(x)} \left[s_\theta(x)^2 \right] + \mathbb{E}_{p_d(x)} \left[\nabla_x s_\theta(x) \right]$$

We will see a similar derivation later on that is used in practice.

Simple sampling

Given that $s_\theta(x) = \nabla_x p_m(x_i; \theta)$, we can follow gradient ascent to sample from the model's distribution:

Algorithm 1: Simple sampling

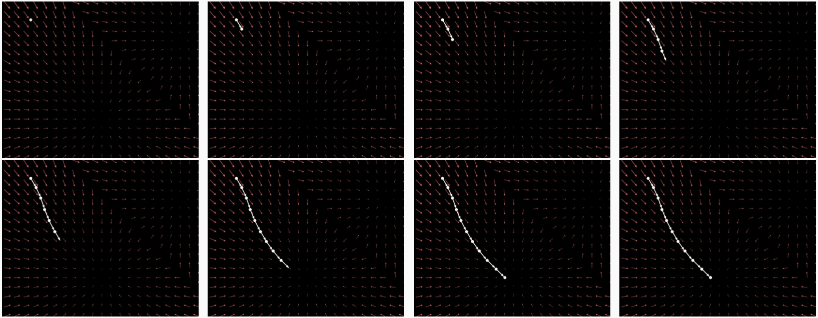
$\tilde{x}_0 \sim N(0, I)$

for $i = 1..K$ **do**

$\quad \tilde{x}_{i+1} \leftarrow \tilde{x}_i + \alpha \underbrace{\nabla_x \log p_m(\tilde{x}_i; \theta)}_{s_\theta(\tilde{x}_i)}$

Simple sampling

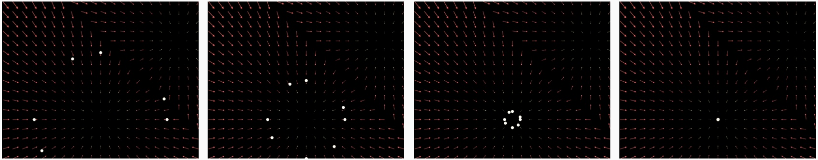
We iteratively apply $\tilde{x}_{i+1} \leftarrow \tilde{x}_i + \alpha \nabla_x \log p_m(x_i;\theta)$



Source: Outlier. (Oct 2024). Diffusion Models From Scratch — Score-Based Generative Models Explained. YouTube.

Simple sampling

This is problematic as sampling different points will always give the same results (e.g. same image).



Source: Outlier. (Oct 2024). Diffusion Models From Scratch — Score-Based Generative Models Explained. YouTube.

Langevin dynamics

It's almost the same as simple sampling, but with added noise:

Algorithm 2: Langevin dynamics

$\tilde{x}_0 \sim N(0, I)$

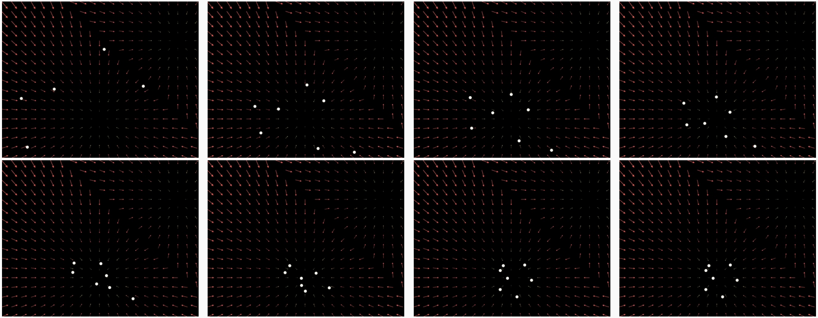
for $i = 1..K$ **do**

$\quad \epsilon \sim \mathcal{N}(0; 1)$

$\quad \tilde{x}_{i+1} \leftarrow \tilde{x}_i + \alpha \underbrace{\nabla_x \log p_m(\tilde{x}_i; \theta)}_{s_\theta(\tilde{x}_i)} + \sqrt{2\alpha}\epsilon$

Langevin dynamics

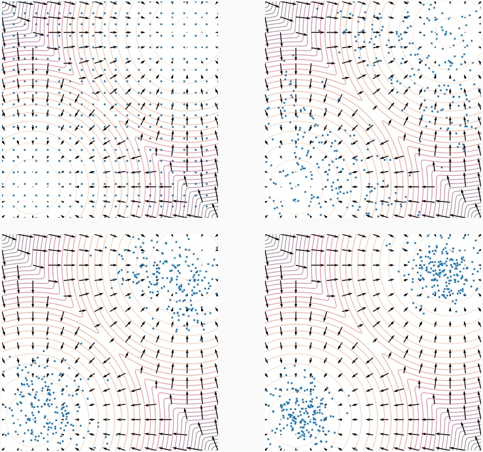
This enables us to get different data samples. The points will still converge to higher density regions, but with small variations.



Source: Outlier. (Oct 2024). Diffusion Models From Scratch — Score-Based Generative Models Explained. YouTube.

Langevin dynamics

This enables us to get different data samples. The points will still converge to higher density regions, but with small variations.



Source: Song, Yang. (May 2021). Generative Modeling by Estimating Gradients of the Data Distribution. Yang Song's blog.

Naive score-based generative modelling

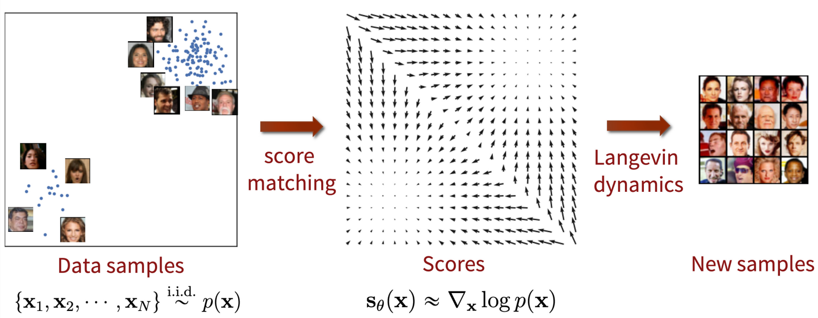


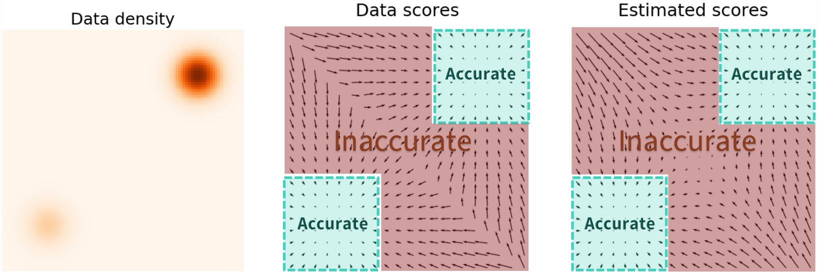
Figure 3: Score-based generative modeling with Langevin dynamics

Song, Yang. (2021). Generative Modeling by Estimating Gradients of the Data Distribution. https://yang-song.net/blog/2021/score/.

Challenges with naive score-based generative modelling

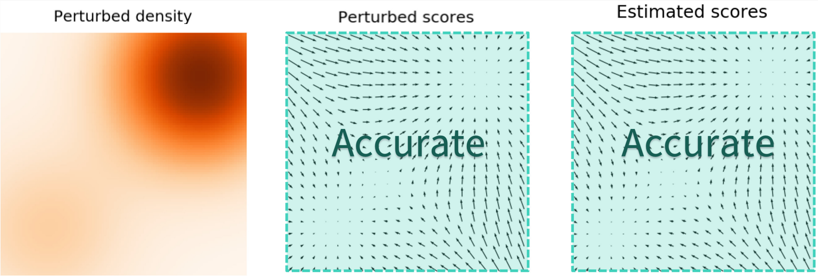
The main pitfall is that the score function is not accurate in low-density regions. When data reside into a high dimensional space, it's very unlikely that we end up in a high-density region.

Figure 4: Estimated scores are only accurate in high density regions.



Multiple noise perturbations

To bypass having low-density regions, we can add a certain **level of noise** to our data. That create more variability in the data, thus widening the high-density regions to cover more space.



With more space covered by the data, the score function will be accurate in more regions.

Multiple noise perturbations

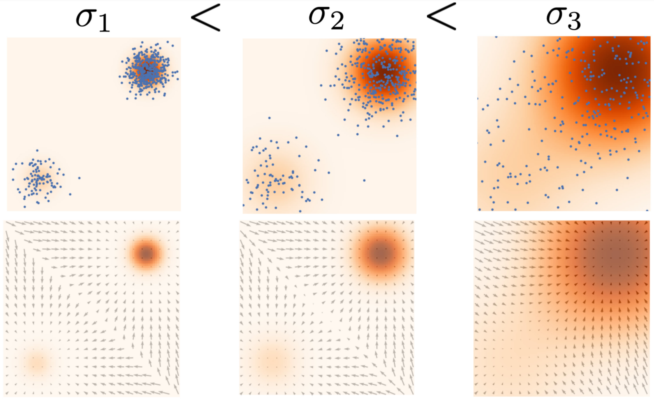
However, 2 things to consider:

1. Adding too little noise will cause a **inaccurate score function**.
2. Adding too much noise will **corrupt the data too much**.

Multiple noise perturbations

Solution: apply multiple different levels of noise to the data and learn from that **simultaneously**.

We perturb the data distribution $p_d(x)$ with each of the Gaussian noise $\mathcal{N}(0, \sigma_o^2 I)$ with $\sigma_1 < \sigma_2 < \cdots < \sigma_L$. Usually $L > 1000$.



Multiple noise perturbations

i-th noise-perturbed distribution:

$$p_{\sigma_i}(x) = \int p(y)\mathcal{N}(x; y, \sigma_i^2 I)dy$$

In reality, we simply sample from $p_{\sigma_i}(x)$ using:

$$x + \sigma_i z \text{ where } z \sim \mathcal{N}(0, I)$$

Noise Conditional Score Network

To be able to predict the score on different levels of noise, we condition the score network on the noise level i.

We train a **Noise Conditional Score Network** (NCSN):

$$s_\theta(x, i) \approx \nabla_x \log p_{\sigma_i}(x)$$

With an objective weighting all noise levels:

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

- $\lambda(i)$: $\mathbb{R} > 0$. Often $\lambda(i) = \sigma_i^2$.
- Train exactly like before with score matching and this new objective.

Denoising score matching

As before, we still can't compute the ground truth score for any noise level:

$$\sum_{i=1}^{L} \lambda(i) \mathbb{E}_{p_{\sigma_i}(x)} \left[\left\| \underbrace{\nabla_x \log p_{\sigma_i}(x)}_{\textit{intractable}} - s_\theta(x, i) \right\|^2 \right]$$

However, we can derive another form similar to the previous score matching objective:

$$\begin{aligned} &\frac{1}{2} \mathbb{E}_{p_{\sigma_i}(\tilde{x})} \left[\| \nabla_x \log p_\theta(\tilde{x}) - s_\theta(\tilde{x}, i) \|_2^2 \right] \\ &= \frac{1}{2} \int p_\theta(\tilde{x}) (\nabla_x \log p_\theta(\tilde{x}) - s_\theta(\tilde{x}))^2 d\tilde{x} \end{aligned}$$

$$= \underbrace{\cancel{\frac{1}{2}\int p_\theta(\tilde{x})(\nabla_x \log p_\sigma(\tilde{x}))^2 d\tilde{x}}}_{\text{don't involve } s_\theta(\tilde{x})} + \frac{1}{2}\int p_\theta(\tilde{x}) s_\theta(\tilde{x})^2 d\tilde{x} - \frac{1}{2}\int p_\theta(\tilde{x}) 2\nabla_x \log p_\sigma(\tilde{x}) s_\theta(\tilde{x}) d\tilde{x}$$

$$= \frac{1}{2}\int p_\theta(\tilde{x}) s_\theta(\tilde{x})^2 d\tilde{x} - \frac{1}{2}\int p_\theta(\tilde{x}) 2\nabla_x \log p_\sigma(\tilde{x}) s_\theta(\tilde{x}) d\tilde{x}$$

$$= \frac{1}{2}\int p_\theta(\tilde{x}) s_\theta(\tilde{x})^2 d\tilde{x} - \int \cancel{p_\theta(\tilde{x})} \frac{\nabla_x p_\sigma(\tilde{x})}{\cancel{p_\sigma(\tilde{x})}} s_\theta(\tilde{x}) d\tilde{x}$$

$$= \frac{1}{2}\int p_\theta(\tilde{x}) s_\theta(\tilde{x})^2 d\tilde{x} - \iint p(x) \nabla_x p_\sigma(\tilde{x}|x) s_\theta(\tilde{x}) dx d\tilde{x}$$

$$= \frac{1}{2}\int p_\theta(\tilde{x}) s_\theta(\tilde{x})^2 d\tilde{x} - \iint p(x) p_\sigma(\tilde{x}|x) \nabla_x \log p_\sigma(\tilde{x}|x) s_\theta(\tilde{x}) dx d\tilde{x}$$

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

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

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

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

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

$$= \frac{1}{2}\mathbb{E}_{x\sim p(x),\tilde{x}\sim p_\theta(\tilde{x})}\left[\|s_\theta(\tilde{x}) - \nabla_x \log p_\sigma(\tilde{x}|x)\|_2^2\right] - \underbrace{\cancel{\mathbb{E}_{x\sim p(x),\tilde{x}\sim p_\theta(\tilde{x})}\left[\|\nabla_x \log p_\sigma(\tilde{x}|x)\|_2^2\right]}}_{\text{don't involve } s_\theta(\tilde{x})}$$

$$= \frac{1}{2}\mathbb{E}_{x\sim p(x),\tilde{x}\sim p_\theta(\tilde{x})}\left[\left\| s_\theta(\tilde{x}) - \underbrace{\nabla_x \log p_\sigma(\tilde{x}|x)}_{\text{why is this better?}} \right\|_2^2\right]$$

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

$\tilde{x} = x + \epsilon$ corresponds to $p_\sigma(\tilde{x}|x)$. We have:

$$
\begin{aligned}
p_\sigma(\tilde{x}|x) &= \frac{1}{(2\pi)^{d/2}\sigma^2} e^{-\frac{1}{2\sigma^2}\|\tilde{x}-x\|^2} \\
\nabla_x \log p_\sigma(\tilde{x}|x) &= \nabla_x \log\left[\frac{1}{(2\pi)^{d/2}\sigma^2} e^{-\frac{1}{2\sigma^2}\|\tilde{x}-x\|^2}\right] \\
&= \cancel{\nabla_x \log\left[\frac{1}{(2\pi)^{d/2}\sigma^2}\right]} + \nabla_x \log\left[e^{-\frac{1}{2\sigma^2}\|\tilde{x}-x\|^2}\right] \\
&= \nabla_x \log\left[e^{-\frac{1}{2\sigma^2}\|\tilde{x}-x\|^2}\right] \\
&= \nabla_x \frac{-1}{2\sigma^2}\|\tilde{x}-x\|^2 \\
&= -\frac{2}{2\sigma^2}(\tilde{x}-x) \\
&= -\frac{1}{\sigma^2}(\tilde{x}-x) \\
&\text{Since } \tilde{x} = x + \epsilon \\
&= -\frac{1}{\sigma^2}(\cancel{x} + \epsilon - \cancel{x}) \\
&= -\frac{\epsilon}{\sigma^2}
\end{aligned}
$$

Replacing the objective we found:

$$\frac{1}{2}\mathbb{E}_{x\sim p(x),\tilde{x}\sim p_\theta(\tilde{x})}\left[\left\|s_\theta(\tilde{x}) - \underbrace{\nabla_x \log p_\sigma(\tilde{x}|x)}\right\|_2^2\right]$$

$$= \frac{1}{2}\mathbb{E}_{x\sim p(x),\tilde{x}\sim p_\theta(\tilde{x})}\left[\left\|s_\theta(\tilde{x}) + \frac{\epsilon}{\sigma^2}\right\|_2^2\right]$$

Making it **tractable** where $s_\theta(\tilde{x})$ is trying to predict $-\frac{\epsilon}{\sigma^2}$ (to remove the noise)!

Annealed Langevin Dynamics

Sampling with NCSN: Same method as Langevin dynamics, but with a twist: **each step**, the scores are predicted from a different noise level.

Algorithm 3: Annealed Langevin dynamics

$x_0 \sim N(0, I)$

for $i = L..1$ **do**

$\quad \epsilon \sim \mathcal{N}(0; 1)$

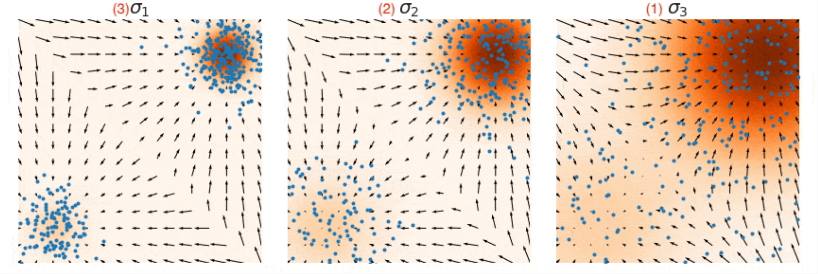
$\quad z_t \sim \mathcal{N}(0; 1)$

$\quad \alpha_i \leftarrow \epsilon \frac{\sigma_i^2}{\sigma_L^2}$

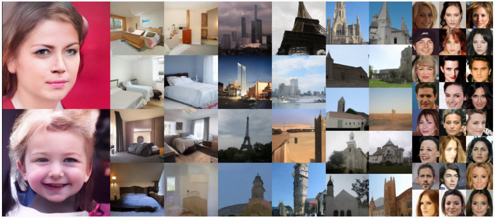
$\quad \tilde{x}_{i+1} \leftarrow \tilde{x}_i + \alpha_i \underbrace{\nabla_x \log p_m(x_i; \sigma_i, \theta)}_{s_\theta(x_i, i)} + \sqrt{2\alpha_i} z_t$

Important: noise is decreasing at each iteration (since $\sigma_L > \sigma_{L-1} > \cdots > \sigma_1$)

Annealed Langevin Dynamics

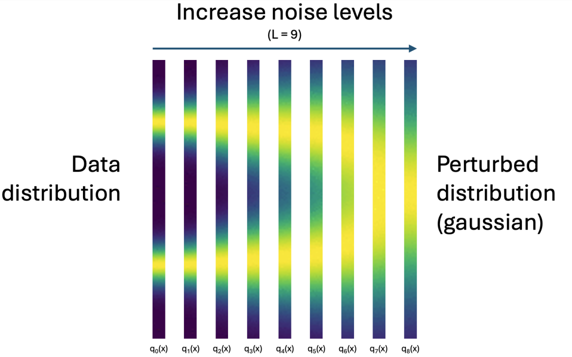


Annealed Langevin Dynamics



Stochastic differential equations

With NCSN, we have a **discrete and finite sequence of noise levels** that the model denoise data with. Most of the time, the sequence length L is fixed to at least 1000 steps.



Stochastic differential equations

In general, an Ordinary Differential Equation (ODE) has the following formulation:

$$dx = f(x, t)dt$$

which describes the evolution of a deterministic system over time.

Stochastic differential equations

Now, if the process is random, we can analyse it's evolution with Stochastic differential equations (SDEs):

$$dx = f(x_t, t)dt + g(t)dw_t$$

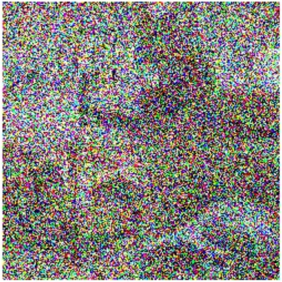
where

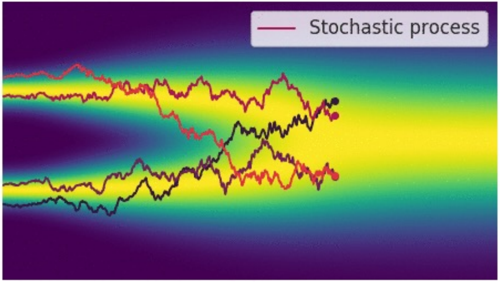
- dw_t is infinitesimal noise.
- $f(x_t, t)$ is the drift coefficient (deterministic part).

Stochastic differential equations

The **forward SDE** perturbs the data on a continuous time-scale:

$$dx = f(x_t, t)dt + g(t)dw_t$$





Stochastic differential equations

It turns out that if you do the derivation, the reverse process of an SDE in general is the following:

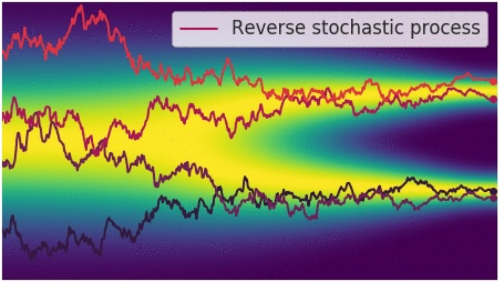
$$\begin{aligned} dx &= \left[f(x_t, t) - g^2(t) \nabla_x \log p_t(x) \right] dt + g(t) dw \\ &= \left[f(x_t, t) - g^2(t) s_\theta(x, t) \right] dt + g(t) dw \end{aligned}$$

We just have to train a **Time-dependent score-based model** $s_\theta(x, t) \approx \nabla_x \log p_t(x)$ which is basically the same as the NCSN $s_\theta(x, i) \approx \nabla_x \log p_{\sigma_i}(x)$. Instead of conditioning on a specific schedule time-step, we **condition on the continuous time** t.

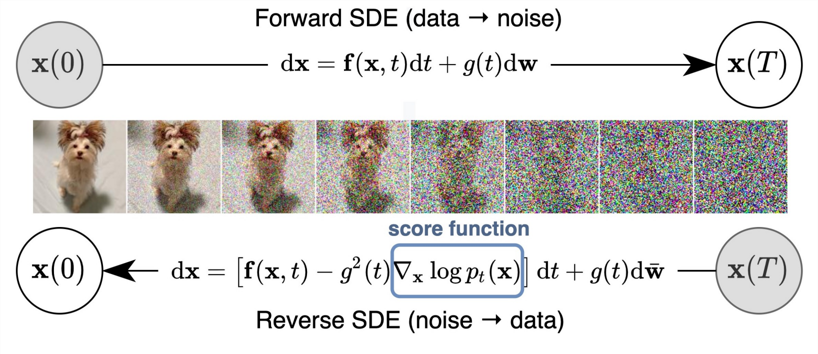
Stochastic differential equations

The reverse SDE starts from noise from a prior distribution (gaussian) to the data:

$$dx = \left[f(x_t, t) - g^2(t)s_\theta(x, t)\right] dt + g(t)dw$$



Stochastic differential equations



Stochastic differential equations

To train that **Time-dependent score-based model**, we use the following objective:

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

- $\mathcal{U}(0, T)$ a uniform distribution over timesteps
- Weighting function $\lambda(t) \propto \frac{1}{\mathbb{E}\left[\left\| \nabla_{x(t)} \log p(x(t) | x(0)) \right\|_2^2 \right]}$

Stochastic differential equations

We now have the reverse SDE:

$$dx = \left[f(x_t, t) - g^2(t) s_\theta(x, t) \right] dt + g(t) dw$$

How do we solve it? (Solving means to get from $t = T$ (noise) to $t = 0$ (data))

1. Train the time-dependent score-based model $s_\theta(x, t)$
2. Use **any** numerical SDE solver to solve the SDE (e.g. Euler-Maruyama).

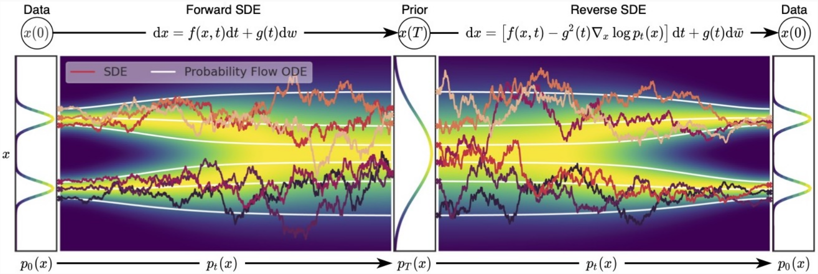
The Euler-Maruyama discretizes the SDE into discrete time-steps (similar to Langevin dynamics).

Algorithm 4: Example SDE solver: Euler-Maruyama

$\Delta t \approx 0$ (very small);
$t \leftarrow T$;
$z_t \sim \mathcal{N}(0, I)$;
repeat
    $\Delta x \leftarrow \left[f(x,t) - g^2(t) s_\theta(x,t)\right] \Delta t + g(t)\sqrt{|\Delta t|} z_t$;
    $x \leftarrow x + \Delta x$;
    $t \leftarrow t + \Delta t$;
until $t \approx 0$;

Benefits:

- We can solve using an **arbitrary number of denoising steps** with the same model.
- The SDE solver is **independent of the model**.



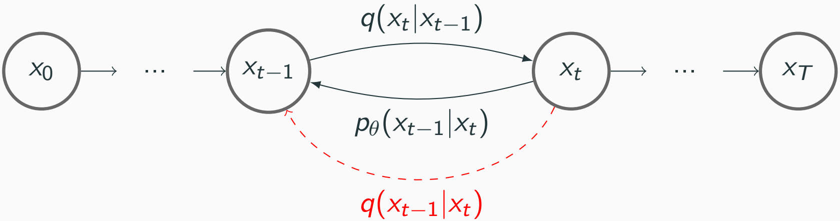
Data
Forward SDE
Prior
Reverse SDE
Data
$x(0)$
$\mathrm{d}x = f(x,t)\mathrm{d}t + g(t)\mathrm{d}w$
$x(T)$
$\mathrm{d}x = \left[f(x,t) - g^2(t)\nabla_x \log p_t(x)\right]\mathrm{d}t + g(t)\mathrm{d}\bar{w}$
$x(0)$
SDE
Probability Flow ODE
x
$p_0(x)$
$p_t(x)$
$p_T(x)$
$p_t(x)$
$p_0(x)$

This formulation of score-based models bring:

- Exact likelihood computation (with probability flow ODE).
- Better sampling methods (but still much slower than GANs).
- Link the score-based models to DDPM.

Denoising diffusion probabilistic models

Diffusion models at high-level



The real $q(x_{t-1}|x_t)$ reverse function is unknown.
We approximate it with a learned function $p_\theta(x_{t-1}|x_t)$.

Forward process

The forward process gradually adds gaussian noise to the data **during training**.

1. Sample a data point x_0 from the real data $x_0 \sim q_d(x)$
2. Add gaussian noise with variance β_t to x_{t-1} to produce a **new latent variable** $q(x_t|x_{t-1})$ where

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Forward process

Problem: Calculating x_t from x_0 iteratively is expensive (especially if it's an image with $t > 1000$).

Solution: Reparametrization trick!

From $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ to $z = \mu + \sigma \odot \epsilon$

Forward process

Solution: Reparametrization trick!

Let $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{t=1}^{T} \alpha_i$

$$
\begin{aligned}
x_t &= \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t I) \\
&= \sqrt{1-\beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon_{t-1} \\
&= \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t} \epsilon_{t-1} \\
&= \sqrt{\alpha_t} (\sqrt{\alpha_{t-1}} x_{t-2} + \sqrt{1-\alpha_{t-1}} \epsilon_{t-2}) + \sqrt{1-\alpha_t} \epsilon_{t-1} \\
&= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{\alpha_t (1-\alpha_{t-1})} \epsilon_{t-2} + \sqrt{1-\alpha_t} \epsilon_{t-1} \\
&= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{1-\alpha_t \alpha_{t-1}} \overline{\epsilon}_{t-2} \\
&= \ldots \text{ replace } x_{t-2} \text{ and so on } \ldots \\
&= \textcolor{red}{\sqrt{\overline{\alpha}_t}} x_0 + \textcolor{blue}{\sqrt{1-\overline{\alpha}_t}} \epsilon_t \\
q(x_t|x_0) &= \mathcal{N}(x_t; \textcolor{red}{\sqrt{\overline{\alpha}_t}} x_0, \textcolor{blue}{(1-\overline{\alpha}_t)} I)
\end{aligned}
$$

Forward process

- Recall: $\alpha_t = 1 - \beta_t$ and β_t is an hyperparameter.
- We can precompute $\overline{\alpha}_t$ (not expensive).
- With $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)I)$ we can sample x_t from x_0 **in one step**.

How to parametrize β_t?

The choice of the noise schedule can be arbitrary as long as there's a **near-linear change in the middle** and very **subtle changes around t=0 and t=T**.

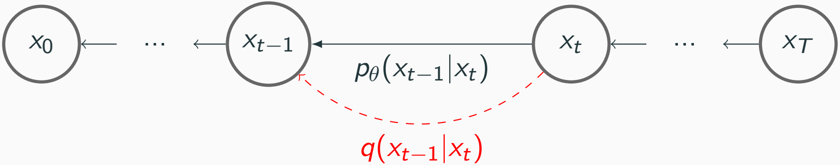
- Linear schedule (perturbs image too quickly initially).
  E.g. from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$.
- Cosine schedule (showed better results)

$$\beta_t = \text{clip}(1 - \frac{\overline{\alpha}_t}{\overline{\alpha}_{t-1}}, 0.999) \text{ with } \overline{\alpha}_t = \frac{f(t)}{f(0)}$$

$$\text{and } f(t) = \cos\left(\frac{t/T + s}{1 + s}\frac{\pi}{2}\right)^2$$

Reverse process

If we could sample from $q(x_{t-1}|x_t)$ (intractable), we could recreate an image from a randomly sampled x_T from a gaussian.



If β_t is small enough, $q(x_{t-1}|x_t)$ is gaussian. So we can approximate it with:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

- $\mu_\theta(x_t, t)$ is learned.
- In practice (for DDPM), $\Sigma_\theta(x_t, t) = \sigma_t^2 I$ which comes from a fixed variance schedule (not learned).

Reverse process

Objective: minimize $-\log p_\theta(x_0)$
Issue: $p_\theta(x_0)$ is intractable as it depends on all previous timesteps $x_0, x_1, ..., x_{T-1}, x_T$.

Variational Lower Bound

Instead of minimizing $-\log p_\theta(x_0)$, we minimize the variational lower bound:

$$-\log(p_\theta(x_0)) \leq -\log p_\theta(x_0) + D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0))$$

- We add the KL (always ≥ 0) since we are minimizing.
- Still depends on $-\log p_\theta(x_0)$! **We have to reformulate.**

Variational Lower Bound

$$
\begin{aligned}
&= -\log(p_\theta(x_0)) \\
&\leq -\log p_\theta(x_0) + D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0)) \\
&= -\log p_\theta(x_0) + \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)}\right] && \text{(Def. KL)} \\
&= -\log p_\theta(x_0) + \mathbb{E}_q\left[\log \frac{q(x_{1:T}|x_0)}{\frac{p_\theta(x_{0:T})}{p_\theta(x_0)}}\right] && \text{(Bayes)} \\
&= \cancel{-\log p_\theta(x_0)} + \mathbb{E}_q\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} + \cancel{\log p_\theta(x_0)}\right] \\
L_{VLB} &= \mathbb{E}_q\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]
\end{aligned}
$$

$q(x_{1:T}|x_0)$ is the forward process, we can compute it.

$p_\theta(x_{0:T})$ is not analytically computable.

Variational Lower Bound

We have to reformulate L_{VLB} even more to make it computable:

$$
\begin{aligned}
L_{VLB} &= \mathbb{E}_q\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right] \\
&= \mathbb{E}_q\left[\log \frac{\prod_{t=1}^{T} q(x_t|x_{t-1})}{p_\theta(x_T)\prod_{t-1}^{T} p_\theta(x_{t-1}|x_t)}\right] && \text{(Def. } q \text{ and} \\
&= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=1}^{T} \log \frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}\right] && \text{(Log prop.)} \\
&= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=2}^{T} \log \frac{\color{red}{q(x_t|x_{t-1})}}{p_\theta(x_{t-1}|x_t)} + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right] && \text{(First term)}
\end{aligned}
$$

Details

$$
\color{red}{q(x_t|x_{t-1})} \overset{\text{bayes}}{=} \frac{q(x_{t-1}|x_t)q(x_t)}{q(x_{t-1})} \implies \frac{q(x_{t-1}|x_t, x_0)q(x_t|x_0)}{q(x_{t-1}|x_0)}
$$

Condition on x_0 to lower the variance.

$$= \mathbb{E}_q \left[-\log p_\theta(x_T) + \sum_{t=2}^{T} \log \frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)} + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} \right]$$

$$= \mathbb{E}_q \left[-\log p_\theta(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)} + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} \right]$$

$$= \mathbb{E}_q \left[-\log p_\theta(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} + \sum_{t=2}^{T} \log \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)} + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} \right]$$

Additional simplification

$$\sum_{t=2}^{T} \log \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)} = \log \prod_{t=2}^{T} \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)} = \log \frac{\cancel{q(x_2|x_0)}\,\cancel{q(x_3|x_0)}\,q(x_4|x_0)\ldots}{q(x_1|x_0)\,\cancel{q(x_2|x_0)}\,\cancel{q(x_3|x_0)}\ldots}$$

$$= \log \frac{q(x_T|x_0)}{q(x_1|x_0)}$$

$$= \mathbb{E}_q \left[-\log p_\theta(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} + \sum_{t=2}^{T} \log \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)} + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} \right]$$

$$= \mathbb{E}_q \left[-\log p_\theta(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} + \log \frac{q(x_T|x_0)}{q(x_1|x_0)} + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} \right]$$

$$= \mathbb{E}_q \left[-\log p_\theta(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} + \log \frac{q(x_T|x_0)\cancel{q(x_1|x_0)}}{p_\theta(x_0|x_1)\cancel{q(x_1|x_0)}} \right]$$

$$= \mathbb{E}_q \left[-\log p_\theta(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} + \log \frac{q(x_T|x_0)}{p_\theta(x_0|x_1)} \right]$$

$$= \mathbb{E}_q \left[-\log p_\theta(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} + \log q(x_T|x_0) - \log p_\theta(x_0|x_1) \right]$$

$$= \mathbb{E}_q \left[\log \frac{q(x_T|x_0)}{p_\theta(x_T)} + \sum_{t=2}^{T} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} - \log p_\theta(x_0|x_1) \right]$$

$$= \mathbb{E}_q [\underbrace{D_{KL}(q(x_T|x_0)||p_\theta(x_T))}_{(1)} + \underbrace{\sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))}_{(2)} \underbrace{- \log p_\theta(x_0|x_1)}_{(3)}]$$

(1) $D_{KL}(q(x_T|x_0)||p_\theta(x_T))$

$D_{KL}(q(x_T|x_0)||p_\theta(x_T))$

- $q(x_T|x_0)$: no learnable parameter **AND** converges to isotropic gaussian.
- $p_\theta(x_T)$: random noise sampled from an isotropic gaussian.

We can ignore this term (1)!

We are left with:

$$L_{VLB} = \mathbb{E}_q[\underbrace{\sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))}_{(2)} \underbrace{- \log p_\theta(x_0|x_1)}_{(3)}]$$

(2) $\sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$

Recall that estimating $q(x_{t-1}|x_t)$ is intractable. However, by conditioning on x_0, the term in the above equation can be computed:

$$\text{Let } q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta} I)$$

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

Replace each term by their gaussian form w/o the coeffcient.

$$\propto \exp\left[\frac{-1}{2}\left(\frac{(x_t - \sqrt{\alpha_t}x_{t-1})^2}{\sqrt{\beta_t}^2} + \frac{(x_{t-1} - \sqrt{\overline{\alpha}_{t-1}}x_0)^2}{\sqrt{1-\overline{\alpha}_{t-1}}^2} - \frac{(x_t - \sqrt{\overline{\alpha}_t}x_0)^2}{\sqrt{1-\overline{\alpha}_t}^2}\right)\right]$$

$$= \exp\left[\frac{-1}{2}\left(\frac{(x_t - \sqrt{\alpha_t}x_{t-1})^2}{\beta_t} + \frac{(x_{t-1} - \sqrt{\overline{\alpha}_{t-1}}x_0)^2}{1-\overline{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\overline{\alpha}_t}x_0)^2}{1-\overline{\alpha}}\right)\right]$$

$$= \exp\left[\frac{-1}{2}\left(\frac{(x_t - \sqrt{\alpha_t}x_{t-1})^2}{\beta_t} + \frac{(x_{t-1} - \sqrt{\overline{\alpha}_{t-1}}x_0)^2}{1-\overline{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\overline{\alpha}_t}x_0)^2}{1-\overline{\alpha}}\right)\right]$$

$$= \exp\left[\frac{-1}{2}\left(\frac{x_t^2 - 2\sqrt{\alpha_t}x_t x_{t-1} + \alpha_t x_{t-1}^2}{\beta_t} + \frac{x_{t-1}^2 - 2\sqrt{\overline{\alpha}_{t-1}}x_{t-1}x_0 + \overline{\alpha}_{t-1}x_0^2}{1-\overline{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\overline{\alpha}_t}x_0)^2}{1-\overline{\alpha}}\right)\right]$$

$$= \exp\left[\frac{-1}{2}\left(\frac{x_t^2}{\beta_t} - \frac{2\sqrt{\alpha_t}x_t x_{t-1}}{\beta_t} + \frac{\alpha_t x_{t-1}^2}{\beta_t} + \frac{x_{t-1}^2}{1-\overline{\alpha}_{t-1}} - \frac{2\sqrt{\overline{\alpha}_{t-1}}x_{t-1}x_0}{1-\overline{\alpha}_{t-1}} + \frac{\overline{\alpha}_{t-1}x_0^2}{1-\overline{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\overline{\alpha}_t}x_0)^2}{1-\overline{\alpha}}\right)\right]$$

Discard/regroup terms in black that don't depend on x_{t-1}

$$= \exp\left[\frac{-1}{2}\left(-\frac{2\sqrt{\alpha_t}x_t x_{t-1}}{\beta_t} + \frac{\alpha_t x_{t-1}^2}{\beta_t} + \frac{x_{t-1}^2}{1-\overline{\alpha}_{t-1}} - \frac{2\sqrt{\overline{\alpha}_{t-1}}x_{t-1}x_0}{1-\overline{\alpha}_{t-1}} + C(x_t, x_0)\right)\right]$$

We factorize x_{t-1}^2 and x_{t-1}

$$= \exp\left[\frac{-1}{2}\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\overline{\alpha}_{t-1}}\right)x_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}x_t}{\beta_t} + \frac{\sqrt{\overline{\alpha}_{t-1}}x_0}{1-\overline{\alpha}_{t-1}}\right)x_{t-1} + C(x_t, x_0)\right]$$

$$\exp\left[\frac{-1}{2}\left(\frac{\alpha_t}{\beta_t}+\frac{1}{1-\overline{\alpha}_{t-1}}\right)x_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}x_t}{\beta_t}+\frac{\sqrt{\overline{\alpha}_{t-1}}x_0}{1-\overline{\alpha}_{t-1}}\right)x_{t-1} + C(x_t, x_0)\right]$$

Following the gaussian distribution form (omitting coefs) of $\mathcal{N}(x_{t-1}, \tilde{\mu}(x_t, x_0), \tilde{\beta}I)$

$$\begin{aligned}
\mathcal{N}(x_{t-1}, \tilde{\mu}(x_t, x_0), \tilde{\beta}I) &= \exp\left[\frac{-1}{2}\frac{(x_{t-1}-\tilde{\mu}(x_t, x_0))^2}{\tilde{\beta}}\right] \\
&= \exp\left[\frac{-1}{2}\frac{x_{t-1}^2 - 2x_{t-1}\tilde{\mu}(x_t, x_0) + \tilde{\mu}(x_t, x_0)^2}{\tilde{\beta}}\right] \\
&= \exp\left[\frac{-1}{2}\left(\frac{x_{t-1}^2}{\tilde{\beta}} - \frac{2x_{t-1}\tilde{\mu}(x_t, x_0)}{\tilde{\beta}} + C(x_t, x_0)\right)\right] \\
&= \exp\left[\frac{-1}{2}\left(\frac{1}{\tilde{\beta}}x_{t-1}^2 - \frac{2\tilde{\mu}(x_t, x_0)}{\tilde{\beta}}x_{t-1} + C(x_t, x_0)\right)\right]
\end{aligned}$$

By plugging the formulation at the top into the gaussian form, we can find $\tilde{\mu}(x_t, x_0)$ and $\tilde{\beta}$.

Recall that $\beta_t = 1 - \alpha_t$. We begin by finding $\tilde{\beta}$:

$$
\begin{aligned}
\frac{1}{\tilde{\beta}} &= \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\overline{\alpha}_{t-1}}\right) \\
\tilde{\beta} &= \frac{1}{\frac{\alpha_t}{\beta_t} + \frac{1}{1-\overline{\alpha}_{t-1}}} \\
&= \frac{1}{\frac{\alpha_t - \alpha_t\overline{\alpha}_{t-1} + \beta_t}{\beta_t(1-\overline{\alpha}_{t-1})}} \\
&= \frac{\beta_t(1-\overline{\alpha}_{t-1})}{\alpha_t - \alpha_t\overline{\alpha}_{t-1} + \beta_t} \\
&= \frac{1-\overline{\alpha}_{t-1}}{\alpha_t - \overline{\alpha}_t + \beta_t}\beta_t \\
&= \frac{1-\overline{\alpha}_{t-1}}{\cancel{\alpha_t} - \overline{\alpha}_t + 1 - \cancel{\alpha_t}}\beta_t \\
&= \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t
\end{aligned}
$$

Now, for $\tilde{\mu}(x_t, x_0)$, we have:

$$\begin{aligned}
\frac{2\tilde{\mu}(x_t, x_0)}{\tilde{\beta}_t} &= 2\left(\frac{\sqrt{\alpha_t}}{\beta_t}x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}}{1-\overline{\alpha}_{t-1}}x_0\right) \\
\tilde{\mu}(x_t, x_0) &= \left(\frac{\sqrt{\alpha_t}}{\beta_t}x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}}{1-\overline{\alpha}_{t-1}}x_0\right)\tilde{\beta}_t \\
&= \left(\frac{\sqrt{\alpha_t}}{\beta_t}x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}}{1-\overline{\alpha}_{t-1}}x_0\right)\frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t \\
&= \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{(1-\overline{\alpha}_t)}x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{(1-\overline{\alpha}_t)}x_0
\end{aligned}$$

Reparametrization (again)

Using the reparametrization trick of the forward process, we can reparametrize x_0 as a function of x_t:

$$x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon_t \implies x_0 = \frac{1}{\sqrt{\overline{\alpha}_t}}(x_t - \sqrt{1-\overline{\alpha}_t}\epsilon_t)$$

$$= \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{(1-\overline{\alpha}_t)} x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{(1-\overline{\alpha}_t)} x_0$$

By substituing it into the equation, we get:

$$= \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{(1-\overline{\alpha}_t)} x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{(1-\overline{\alpha}_t)} \frac{1}{\sqrt{\overline{\alpha}_t}}(x_t - \sqrt{1-\overline{\alpha}_t}\epsilon_t)$$

... don't understand how to solve this part ...

$$\tilde{\mu}(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_t\right)$$

To recap

To recap, we were trying to make sense of the second term (2) in our variational lower bound:

$$L_{VLB} = \mathbb{E}_q[\underbrace{\sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))}_{(2)} \underbrace{- \log p_\theta(x_0|x_1)}_{(3)}]$$

$q(x_{t-1}|x_t, x_0)$ is our 'target' gaussian that we want p_θ to approximate. We just found a way to calculate it.

$$\begin{aligned} q(x_{t-1}|x_t, x_0) &= \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta} I) \\ &= \mathcal{N}(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_t\right), \tilde{\beta} I) \end{aligned}$$

To recap

To recap, we were trying to make sense of the second term (2) in our variational lower bound:

$$L_{VLB} = \mathbb{E}_q[\underbrace{\sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))}_{(2)} \underbrace{- \log p_\theta(x_0|x_1)}_{(3)}]$$

$p_\theta(x_{t-1}|x_t)$ is the distribution we're trying to model.

$$\begin{aligned} p_\theta(x_{t-1}|x_t) &= \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \\ &= \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I) \end{aligned}$$

In practice (for DDPM), $\sigma_t^2 = \beta_t$ or $\sigma_t^2 = \tilde{\beta}_t$ gives similar results.

N.B.: In a later improved version, [4] proposed to parametrize and **learn the reverse process variance schedule** instead of having it fixed. They proposed an interpolation between β_t and $\tilde{\beta}_t$ by learning a mixing parameter θ:

$$\Sigma_\theta(x_t, t) = \exp(\theta \log \beta_t + (1-\theta)\log \tilde{\beta}_t)$$

and as we found earlier, $\tilde{\beta}_t = \dfrac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$ which means

$$= \exp(\theta \log \beta_t + (1-\theta)\log \left[\frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t\right])$$

But for now, we'll stick with a reverse process of **fixed variance schedule**.

Training loss

Since the variance of $p_\theta(x_{t-1}|x_t)$ is fixed, authors chose to simply minimize the distance between $\mu_\theta(x_t, t)$ and $\tilde{\mu}(x_t, t)$:

$$L_t = \mathbb{E}_{x_0,\epsilon}\left[\frac{1}{2\left\|\sum_\theta(x_t, t)\right\|_2^2}\left\|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\right\|^2\right]$$

However, we know that

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_t\right)$$

And **we also have access to x_t, a_t and $\overline{\alpha}_t$ during the training of** μ_θ. We can then reparametrize μ_θ to only predict the noise ϵ_t instead:

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

The training loss then becomes

$$
\begin{aligned}
L_t &= \mathbb{E}_{x_0,\epsilon}\left[\frac{1}{2\left\|\sum_\theta\right\|_2^2}\left\|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\right\|^2\right] \\
&= \mathbb{E}_{x_0,\epsilon}\left[\frac{1}{2\left\|\sum_\theta\right\|_2^2}\left\|\frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_t) - \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t, t))\right\|^2\right] \\
&= \mathbb{E}_{x_0,\epsilon}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\overline{\alpha}_t)\left\|\sum_\theta\right\|_2^2}\left\|\epsilon_t - \epsilon_\theta(x_t, t)\right\|^2\right]
\end{aligned}
$$

Empirically, ignoring the weighting term provides a more stable training for the diffusion model:

$$
\begin{aligned}
L_t &= \mathbb{E}_{x_0,\epsilon}\left[\left\|\epsilon_t - \epsilon_\theta(x_t, t)\right\|^2\right] \\
&= \mathbb{E}_{x_0,\epsilon}\left[\left\|\epsilon_t - \epsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon_t, t)\right\|^2\right]
\end{aligned}
$$

Algorithm 5: Training

repeat

  Sample $x_0 \sim q(x_0)$;

  Sample $t \sim \text{Uniform}(\{1, \ldots, T\})$;

  Sample $\epsilon \sim \mathcal{N}(0, I)$;

  Take gradient descent step on $\nabla_\theta \left\| \epsilon_t - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_t, t \right) \right\|^2$;

until *converged*;

Algorithm 6: Sampling

Sample $x_T \sim \mathcal{N}(0, I)$

for $t = T, \ldots, 1$ **do**

  if $t > 1$ **then**

    Sample $z \sim \mathcal{N}(0, I)$

  else

    $z = 0$

  $x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z$

return x_0

(3) $-\log p_\theta(x_0|x_1)$

As mentionned before, the overall objective we're trying to minimize is the following:

$$L_{VLB} = \mathbb{E}_q[\underbrace{\sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))}_{(2)} \underbrace{- \log p_\theta(x_0|x_1)}_{(3)}]$$

We know how to minimize (2) with L_t. But, what about (3)? We can't forget about it, otherwise we won't know how to get the image from x_1 to x_0 (the final denoising step).

(3) $-\log p_\theta(x_0|x_1)$

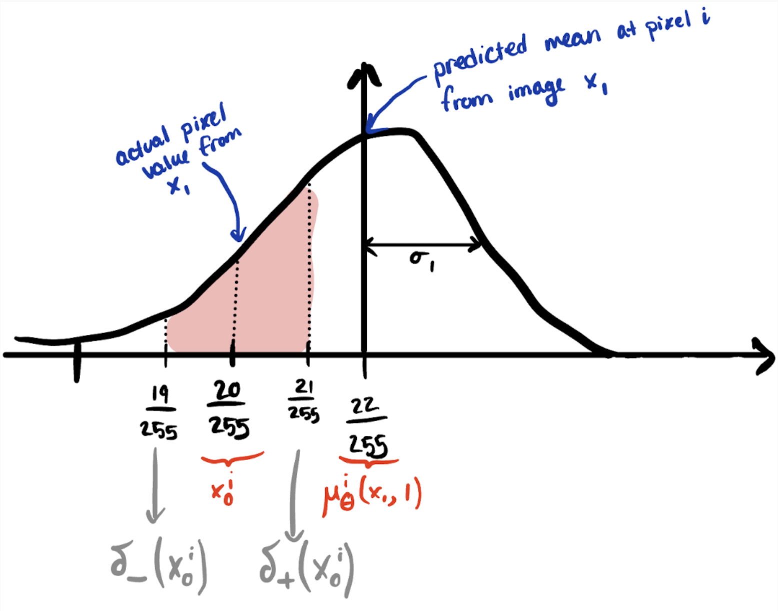
The final denoising step is modelled by another **independent discrete decoder**, with D being the data dimension (e.g. nb of pixels in an image):

$$p_\theta(x_0|x_1) = \prod_{i=1}^{D} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x; \mu_\theta^i(x_1, 1), \sigma_1^2) dx$$

$$\delta_-(x) = \begin{cases} \infty & \text{if } x = 1 \\ x + \frac{1}{255} & \text{if } x < 1 \end{cases} \qquad \delta_+(x) = \begin{cases} -\infty & \text{if } x = -1 \\ x - \frac{1}{255} & \text{if } x > -1 \end{cases}$$

In summary:

1. Calc. the distribution for the i-th pixel of x_0 given the image x_1.
2. Calc. the likelihood of the pixel value x_0^i given the distribution.
3. Multiply the likelihood of all pixels to get the final likelihood.


predicted mean at pixel i from image x1
actual pixel value from x1
σ1
19/255
20/255
21/255
22/255
x0i
μθi(x1, 1)
δ−(x0i)
δ+(x0i)

Recap on the training objective

$$L_{VLB} = \mathbb{E}_q[\underbrace{\sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))}_{(2)} \underbrace{- \log p_\theta(x_0|x_1)}_{(3)}]$$

We now have a way to compute (2) and (3), completing the training objective of the foundation of diffusion models:

- (2): Train an independent decoder: $p_\theta(x_0|x_1) = \prod_{i=1}^{D} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x; \mu_\theta^i(x_1, 1), \sigma_1^2) dx$
- (3): Train the diffusion model with the loss: $L_t = \mathbb{E}_{x_0,\epsilon}\left[\left\|\epsilon_t - \epsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon_t, t)\right\|^2\right]$

Conclusion

We have seen 2 different perspectives on generative modeling using diffusion:

1. **Score-based generative models**: which consists of
    1. Modelling the "score" which is the gradients of the log-likelihood of the data $\nabla_x \log p(x)$ using score matching.
    2. Using the gradients with Langevin dynamics to sample new data.
    3. Learning from multiple levels of noise perturbation to improve the accuracy of scores in the data space.
    4. Learning from an infinite number of noise levels using SDEs.

Conclusion

We have see 2 different perspectives on generative modeling using diffusion:

2. **Denoising Diffusion Probabilistic Models (DDPM)**: which consists of

1. A forward process iteratively adds noise to the data.
2. A reverse process modelled by a model $p_\theta(x_{t-1}|x_t)$, that can generate data (images) from a prior distribution.
3. Training the reverse process using a variational lower bound.

From here

Most subsequent developments on diffusion models focus on improving the methods presented above with:

1. Faster and more efficient sampling.
2. Handling different structures of data.
3. More accurate likelihood and density estimation.

Some keywords I haven't covered in this presentation:

- Conditional generation (with/without guidance from classifiers)
- Scale up resolution.
- Latent diffusion models.
- Speed up sampling process.
- Diffusion models for video generation.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. **"Denoising diffusion probabilistic models".** In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[2] Aapo Hyvärinen and Peter Dayan. **"Estimation of non-normalized statistical models by score matching.".** In: *Journal of Machine Learning Research* 6.4 (2005).

[3] Tero Karras et al. **"Elucidating the design space of diffusion-based generative models".** In: *Advances in neural information processing systems* 35 (2022), pp. 26565–26577.

[4] Alexander Quinn Nichol and Prafulla Dhariwal. **"Improved denoising diffusion probabilistic models".** In: (2021), pp. 8162–8171.

[5] Jiaming Song, Chenlin Meng, and Stefano Ermon. **"Denoising diffusion implicit models".** In: *arXiv preprint arXiv:2010.02502* (2020).

[6] Yang Song and Stefano Ermon. **"Generative modeling by estimating gradients of the data distribution".** In: *Advances in neural information processing systems* 32 (2019).

[7] Yang Song and Stefano Ermon. **"Improved techniques for training score-based generative models".** In: *Advances in neural information processing systems* 33 (2020), pp. 12438–12448.

[8] Yang Song et al. **"Score-based generative modeling through stochastic differential equations".** In: *arXiv preprint arXiv:2011.13456* (2020).

[9] Lilian Weng. **"What are diffusion models?"** In: *lilianweng.github.io* (July 2021). URL: `https://lilianweng.github.io/posts/2021-07-11-diffusion-models/`.

[10] Ling Yang et al. **"Diffusion models: A comprehensive survey of methods and applications".** In: *ACM Computing Surveys* 56.4 (2023), pp. 1–39.