Diffusion Models and an application to DALL-E2

Deep Learning and Neural Networks: Advanced Topics

Fabio Brau

March 13, 2024

Scuola Superiore Sant'Anna, Pisa.

Introduction

Diffusion Models

Conclusion

Introduction

Goal

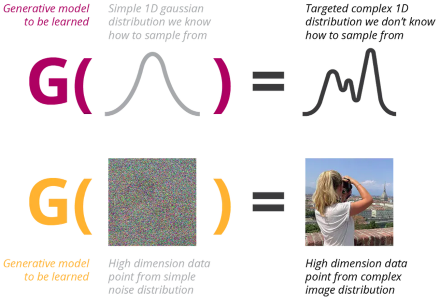
Given a set of examples (x_i) sampled from some distribution $\mathcal{X}$, create a new examples $\tilde{x}$ in the same distribution.

Method

Search for a G such that if z sampled from an easy distribution (e.g., $\mathcal{N}(0, \sigma I)$), then $G(z)$ is in the desired distribution.

Issues

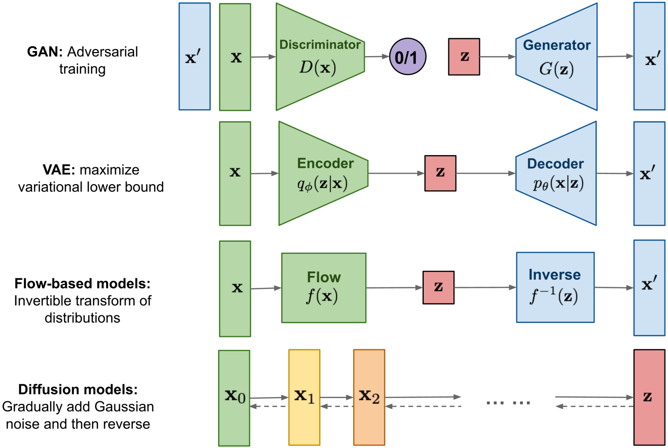
$\mathcal{X}$ is either **unkown** and **not explicit**.



General Schema of generative models.

[1] **Credits:** https://towardsdatascience.com/understanding-diffusion-probabilistic-models-dpms-1940329d6048

Generative Models

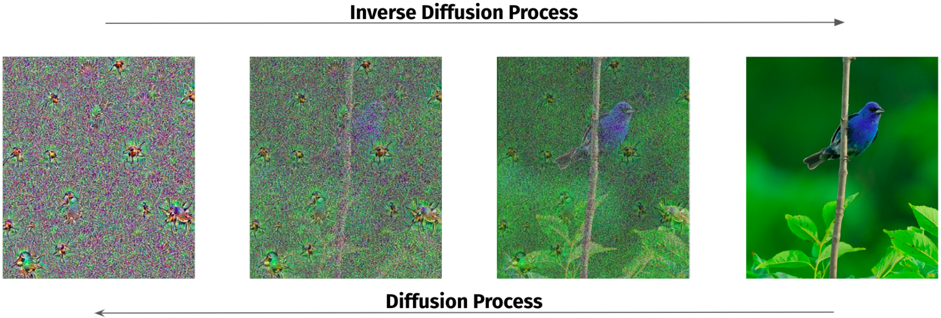


Overview of different generative models[1]

[1] **Credits** https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion Models

Diffusion models are generative models that learns a inverse diffusion process

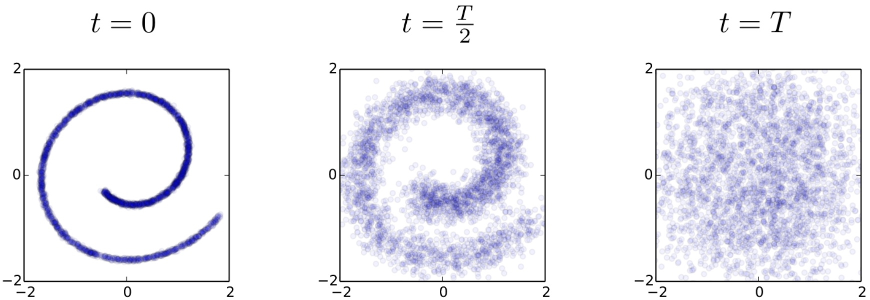


Timeline

2015) *Deep Unsupervised Learning with Non-equilibrium Thermodynamics*. Sohl-Dickstein et al. ICML.

2020) *Denoising Diffusion Probabilistic Models*. Ho et al. NeurIPS.

2021) *Score-Based Generative Modeling Through SDE*. Song et al. ICLR.



Diffusion process as a **Markov Chain** with **Continuous State Space** and **Discrete Time**.[2]

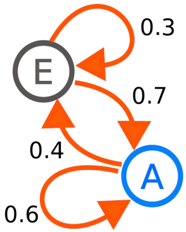
[2] Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics".

Informal Definition

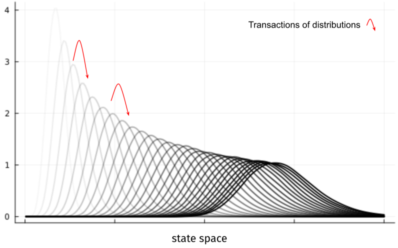
A sequence of random variables $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(t)}, \cdots$, such that

- $\mathbf{x}^{(t)} \in S$, where S **State Space**
- The future $\mathbf{x}^{(t+1)}$ depends on the present $\mathbf{x}^{(t)}$ but not on the past $\mathbf{x}^{(t-1)}$

Discrete State Space S



Continuous State Space S



Definition (Discrete Space Markov Chain)
Let $S = \{s_0, \cdots, s_n, \cdots\}$ be a discrete state space.
A sequence $\{\mathbf{x}^{(t)}\}_t \subseteq S$ is a Markov chain if there exists a transition matrix $P = (p_{ij})$ such that

(Markov Property)

$$\mathbb{P}(\mathbf{x}^{(t+1)} = s_j | \mathbf{x}^{(0)} = s_{i_0}, \cdots, \mathbf{x}^{(t)} = s_{i_t}) \\ = \\ \mathbb{P}(\mathbf{x}^{(t+1)} = s_j | \mathbf{x}^{(t)} = s_{i_t})$$

(Homogeneous Property)

$$\mathbb{P}\left(\mathbf{x}^{(t+1)} = s_j | \mathbf{x}^{(t)} = s_i\right) = p_{ij}$$

Definition (Discrete Space Markov Chain)

Let $S = \{s_0, \cdots, s_n, \cdots\}$ be a discrete state space.
A sequence $\{\mathbf{x}^{(t)}\}_t \subseteq S$ is a Markov chain if there exists a transition matrix $P = (p_{ij})$ such that

(Markov Property)

$$\mathbb{P}(\mathbf{x}^{(t+1)} = s_j | \mathbf{x}^{(0)} = s_{i_0}, \cdots, \mathbf{x}^{(t)} = s_{i_t}) = \mathbb{P}(\mathbf{x}^{(t+1)} = s_j | \mathbf{x}^{(t)} = s_{i_t})$$

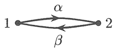
(Homogeneous Property)

$$\mathbb{P}\left(\mathbf{x}^{(t+1)} = s_j | \mathbf{x}^{(t)} = s_i\right) = p_{ij}$$

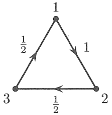
P is a stochastic matrix!

$$\forall i, \quad \sum_{j \in \mathbb{N}} p_{ij} = 1$$

$$P = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$$



$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \end{pmatrix}$$



Reminder: DTMC with Continuous State Space

Definition (DTMC Continous state) A sequence $\{(\mathbf{x}^{(t)}\} \subseteq S$ of random variables, with labeled distributions $q(\mathbf{x}^{(i)})$, is a DTMC if there exists a parameterized map $\kappa_t : S \times S \to [0, 1]$ such that

$$\mathbb{P}(x^{(t+1)} \in A \,|\, x^{(t)} = s_t) = \int_A \kappa_t(y, s_t)\, dy$$

and the Markov property holds, i.e. absence of memory.

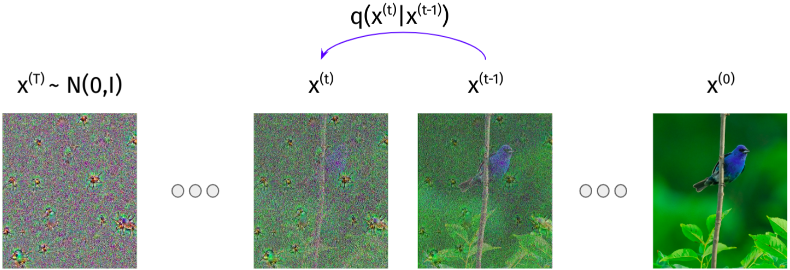
Notation

- We call $(\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(T)})$ **trajectory**
- We call $q(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}) = \kappa_t$ **transition kernel**.
- We call $q((\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(T)})) = q(\mathbf{x}^{(0 \ldots T)})$ the **joint distribution** of the whole trajectory.

Forward Diffusion Process

The forward diffusion process is a DTMC with a Gaussian transition kernel.

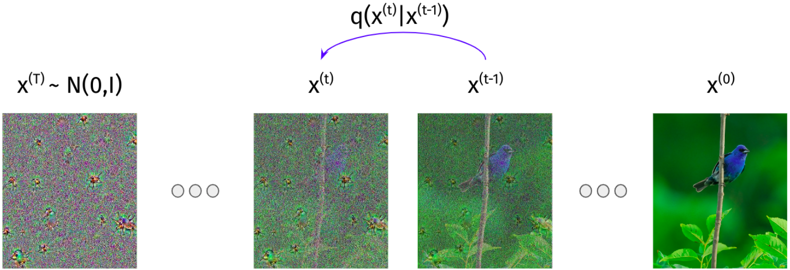
- **Data Distribution**: $\mathbf{x}^{(0)}$ with (unctractable) distribution $q(\mathbf{x}^{(0)})$;
- **Transaction Kernel**: $q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1-\beta_t}\mathbf{x}^{(t-1)}; \beta_t I\right)$
- **Diffusion rates**: $0 < \beta_1 < \cdots < \beta_T = 1$



Forward Diffusion Process

The forward diffusion process is a DTMC with a Gaussian transition kernel.

- **Data Distribution**: $\mathbf{x}^{(0)}$ with (unctractable) distribution $q(\mathbf{x}^{(0)})$; Not Analytic!!
- **Transaction Kernel**: $q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1-\beta_t}\mathbf{x}^{(t-1)}; \beta_t I\right)$
- **Diffusion rates**: $0 < \beta_1 < \cdots < \beta_T = 1$



Observation I Each term of the trajectory can be expressed as a combination of the previous state and the Gaussian noise.

$$\mathbf{x}^{(t)} = \sqrt{1-\beta_t}\,\mathbf{x}^{(t-1)} + \sqrt{\beta_t}\,\boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, I)$$

Observation II Each term can be written as a combination of the initial state and a Gaussian noise.

$$\mathbf{x}^{(t)} = \sqrt{1-\alpha_t}\,\mathbf{x}^{(0)} + \sqrt{\alpha_t}\boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, I)$$

where

$$\alpha_t = 1 - \prod_{i=0}^{t}(1-\beta_i)$$

Observation

The markov property allows breaking up the distributional Representation of the whole trajectory as a product of transitional distributions. Consiering the following interemediate rappresentation

$$q(\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(T)}) = q\left(\mathbf{x}^{(T)} \mid \mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(T-1)}\right) q\left(\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(T-1)}\right)$$

Observation

The markov property allows breaking up the distributional Representation of the whole trajectory as a product of transitional distributions. Consiering the following interemediate rappresentation

$$\begin{aligned} q(\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(T)}) &= q\left(\mathbf{x}^{(T)} \mid \mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(T-1)}\right) q\left(\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(T-1)}\right) \\ &= q\left(\mathbf{x}^{(T)} \mid \mathbf{x}^{(T-1)}\right) q\left(\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(T-1)}\right) \\ &\vdots \end{aligned} \tag{1}$$

By iterating the procedure we get the following representation of the diffusione process

Observation

The markov property allows breaking up the distributional Representation of the whole trajectory as a product of transitional distributions. Consiering the following interemediate rappresentation

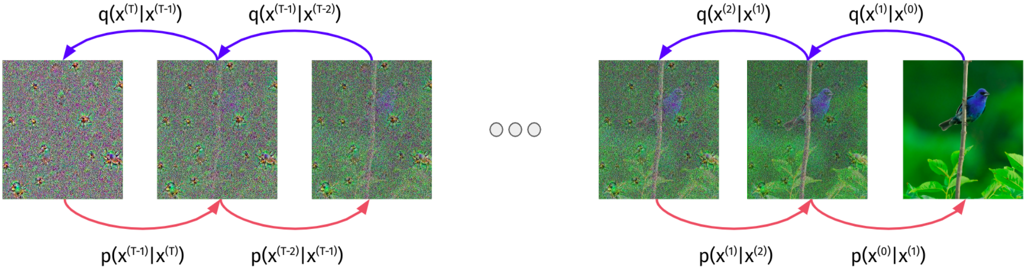
$$
\begin{aligned}
q(\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T)}) &= q\left(\mathbf{x}^{(T)} \,|\, \mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T-1)}\right) q\left(\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T-1)}\right) \\
&= q\left(\mathbf{x}^{(T)} \,|\, \mathbf{x}^{(T-1)}\right) q\left(\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T-1)}\right) \\
&\vdots
\end{aligned} \tag{1}
$$

By iterating the procedure we get the following representation of the diffusione process

Distributional Representation

$$
q(\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T)}) = q(\mathbf{x}^{(0)}) \prod_{t=1}^{T} q\left(\mathbf{x}^{(t)} \,|\, \mathbf{x}^{(t-1)}\right)
$$

Reverse Diffusion Process



Reverse Diffusion Process

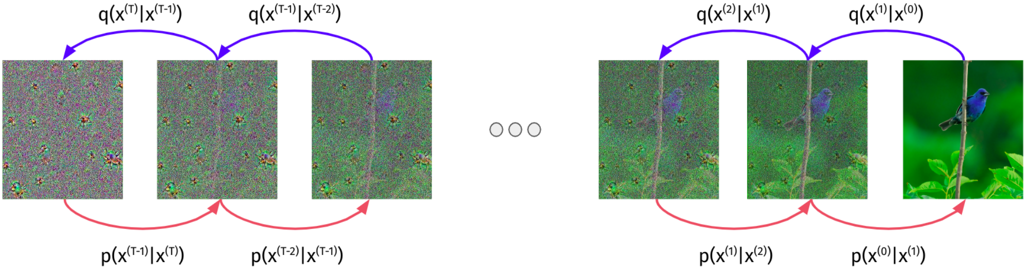
Fixed Forward Process

Initial Distribution

$q(\mathbf{x}^{(0)})$

Gaussian Transaction Kernel

$$q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1-\beta_t}\mathbf{x}^{(t-1)}; \beta_t I\right)$$



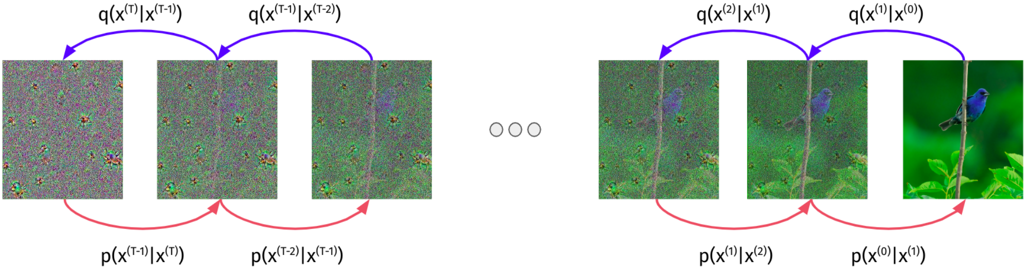
Fixed Forward Process

Initial Distribution

$q(\mathbf{x}^{(0)})$

Gaussian Transaction Kernel

$q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1-\beta_t}\mathbf{x}^{(t-1)}; \beta_t I\right)$



Learned Reverse Process

Initial Distribution

$p(\mathbf{x}^{(T)}) \sim \mathcal{N}(0, I)$

Approximation of

$q(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)})$

Gaussian Kernel with parameters

$p_\theta(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}) = \mathcal{N}\left(\mathbf{x}^{(t-1)}; \boldsymbol{\mu}_\theta\left(\mathbf{x}^{(t)}, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}^{(t)}, t\right)\right)$

Forward Diffusion Process

$q(\mathbf{x}^{(0)})$ Data Distribution

$$q(\mathbf{x}^{(0\ldots T)}) = q(\mathbf{x}^{(0)}) \prod_{t=1}^{T} q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right)$$

Reverse Diffusion Process

$$q(x^{(T)}) = \mathcal{N}(0, I)$$

$$q(\mathbf{x}^{(0\ldots T)}) = q(\mathbf{x}^{(T)}) \prod_{t=1}^{T} q\left(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}\right)$$

[3] Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics".

Forward Diffusion Process

$q(\mathbf{x}^{(0)})$ Data Distribution

$$q(\mathbf{x}^{(0\dots T)}) = q(\mathbf{x}^{(0)}) \prod_{t=1}^{T} q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right)$$

Reverse Diffusion Process

$$q(x^{(T)}) = \mathcal{N}(0, I)$$

$$q(\mathbf{x}^{(0\dots T)}) = q(\mathbf{x}^{(T)}) \prod_{t=1}^{T} q\left(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}\right)$$

Theorem. Reverse of Gaussian DP is $\approx$ Gaussian DP[3]

If $|\beta_i - \beta_{i+1}| \approx 0$, i.e. diffusion **slow enough**, then the reverse diffusion process can be approximated by an other diffusion process with a gaussin transition kernel

$$q(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}) \approx \mathcal{N}\left(\mathbf{x}^{(t-1)}; \boldsymbol{\mu}_\theta\left(\mathbf{x}^{(t)}, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}^{(t)}, t\right)\right)$$

[3] Sohl-Dickstein et al., “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”.

Forward Diffusion Process

$q(\mathbf{x}^{(0)})$ Data Distribution

$$q(\mathbf{x}^{(0\dots T)}) = q(\mathbf{x}^{(0)}) \prod_{t=1}^{T} q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right)$$

Reverse Diffusion Process

$$q(x^{(T)}) = \mathcal{N}(0, I)$$

$$q(\mathbf{x}^{(0\dots T)}) = q(\mathbf{x}^{(T)}) \prod_{t=1}^{T} q\left(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}\right)$$

Theorem. Reverse of Gaussian DP is $\approx$ Gaussian DP[3]

If $|\beta_i - \beta_{i+1}| \approx 0$, i.e. diffusion **slow enough**, then the reverse diffusion process can be approximated by an other diffusion process with a gaussin transition kernel

$$q(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}) \approx \mathcal{N}\left(\mathbf{x}^{(t-1)}; \boxed{\boldsymbol{\mu}_\theta}\left(\mathbf{x}^{(t)}, t\right), \boxed{\boldsymbol{\Sigma}_\theta}\left(\mathbf{x}^{(t)}, t\right)\right)$$

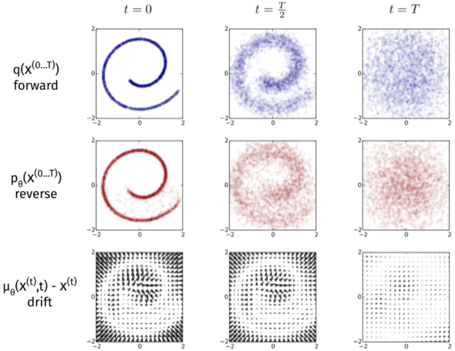
Mean $\boldsymbol{\mu}_\theta$ and covariance $\boldsymbol{\Sigma}_\theta$ are the output of a neural model !!

[3] Sohl-Dickstein et al., “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”.

Visualization of Diffusion Process: 2D dimensional case[4]

- **(Top)** Forward diffusion process that brings the swiss-roll data (left) into the $\mathcal{N}(0, I)$ gaussian distribution (right).
- **(Center)** The trained reverse trajectory from the identity-covariance gaussian distribution (right) to the reconstructed data distribution.
- **(Bottom)** The drift term for different time slices defined as follows

$$\epsilon_t = \mu_\theta(\mathbf{x}^{(t)}, t) - \mathbf{x}^{(t)}$$



[4]Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics".

Training of μ_θ and Σ_θ

Goal

Search for the best parameters θ

$$q(\mathbf{x}^{(0)}) \approx p_\theta(\mathbf{x}^{(0)})$$

where $\mathbf{x}^{(0)} \cdots , \mathbf{x}^{(T)}$ diffusion process

Estimated Reverse Process

$$p_\theta(\mathbf{x}^{(T)}) = \mathcal{N}\left(\mathbf{x}^{(T)}; 0, I\right)$$

$$p_\theta(\cdot \mid \mathbf{x}^{(t)}) = \mathcal{N}\left(\boldsymbol{\mu}_\theta\left(\mathbf{x}^{(t)}, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}^{(t)}, t\right)\right)$$

Training of $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$

Goal

Search for the best parameters θ

$$q(\mathbf{x}^{(0)}) \approx p_\theta(\mathbf{x}^{(0)})$$

where $\mathbf{x}^{(0)} \cdots , \mathbf{x}^{(T)}$ diffusion process

Method

Minimize the *Kullback–Leibler* Divergence

$$D_{KL}(q \,||\, p_\theta) := \int q(\mathbf{x}^{(0)}) \log \left(\frac{q(\mathbf{x}^{(0)})}{p_\theta(\mathbf{x}^{(0)})} \right) d\mathbf{x}^{(0)}$$

Estimated Reverse Process

$$p_\theta(\mathbf{x}^{(T)}) = \mathcal{N}\left(\mathbf{x}^{(T)}; 0, I\right)$$

$$p_\theta(\cdot \,|\, \mathbf{x}^{(t)}) = \mathcal{N}\left(\boldsymbol{\mu}_\theta\left(\mathbf{x}^{(t)}, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}^{(t)}, t\right)\right)$$

Goal

Search for the best parameters θ

$$q(\mathbf{x}^{(0)}) \approx p_\theta(\mathbf{x}^{(0)})$$

where $\mathbf{x}^{(0)} \cdots, \mathbf{x}^{(T)}$ diffusion process

Estimated Reverse Process

$$p_\theta(\mathbf{x}^{(T)}) = \mathcal{N}\left(\mathbf{x}^{(T)}; 0, I\right)$$
$$p_\theta(\cdot \,|\, \mathbf{x}^{(t)}) = \mathcal{N}\left(\boldsymbol{\mu}_\theta\left(\mathbf{x}^{(t)}, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}^{(t)}, t\right)\right)$$

Method

Minimize the *Kullback–Leibler* Divergence

$$\boxed{D_{KL}(q \,||\, p_\theta) := \int q(\mathbf{x}^{(0)}) \log\left(\frac{q(\mathbf{x}^{(0)})}{p_\theta(\mathbf{x}^{(0)})}\right) d\mathbf{x}^{(0)}}$$

Easy??

Goal

Search for the best parameters θ

$$q(\mathbf{x}^{(0)}) \approx p_\theta(\mathbf{x}^{(0)})$$

where $\mathbf{x}^{(0)} \cdots, \mathbf{x}^{(T)}$ diffusion process

Estimated Reverse Process

$$p_\theta(\mathbf{x}^{(T)}) = \mathcal{N}\left(\mathbf{x}^{(T)}; 0, I\right)$$
$$p_\theta(\cdot \,|\, \mathbf{x}^{(t)}) = \mathcal{N}\left(\boldsymbol{\mu}_\theta\left(\mathbf{x}^{(t)}, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}^{(t)}, t\right)\right)$$

Method

Minimize the *Kullback–Leibler* Divergence

$$\boxed{D_{KL}(q \,||\, p_\theta) := \int q(\mathbf{x}^{(0)}) \log\left(\frac{q(\mathbf{x}^{(0)})}{p_\theta(\mathbf{x}^{(0)})}\right) d\mathbf{x}^{(0)}}$$

Easy??

No. $q(\mathbf{x}^{(0)})$ is analytically intractable!!

Goal: Deduce a tractable loss function

$$\boxed{D_{KL}(q \,||\, p_\theta) := \int q(\mathbf{x}^{(0)}) \log \left(\frac{q(\mathbf{x}^{(0)})}{p_\theta(\mathbf{x}^{(0)})} \right) d\mathbf{x}^{(0)}}$$

Training of μ_θ and Σ_θ

Goal: Deduce a tractable loss function

$$\boxed{D_{KL}(q \,||\, p_\theta) := \int q(\mathbf{x}^{(0)}) \log\left(\frac{q(\mathbf{x}^{(0)})}{p_\theta(\mathbf{x}^{(0)})}\right) d\mathbf{x}^{(0)}}$$

Simplification I: Minimize the Cross Entropy

$$D_{KL}\left(q(\mathbf{x}^{(0)})||p_\theta(\mathbf{x}^{)0)})\right) = \int q(\mathbf{x}^{(0)}) \log(q(\mathbf{x}^{(0)})\, d\mathbf{x}^{(0)} + \int -q(\mathbf{x}^{(0)}) \log(p_\theta(\mathbf{x}^{(0)}))\, d\mathbf{x}^{(0)}$$

Goal: Deduce a tractable loss function

$$\boxed{D_{KL}(q \,||\, p_\theta) := \int q(\mathbf{x}^{(0)}) \log\left(\frac{q(\mathbf{x}^{(0)})}{p_\theta(\mathbf{x}^{(0)})}\right) d\mathbf{x}^{(0)}}$$

Simplification I: Minimize the Cross Entropy

$$D_{KL}\left(q(\mathbf{x}^{(0)}) || p_\theta(\mathbf{x}^{)0)})\right) = \underbrace{\int q(\mathbf{x}^{(0)}) \log(q(\mathbf{x}^{(0)})\, d\mathbf{x}^{(0)}}_{-\mathbb{H}\left(q(\mathbf{x}^{(0)}\right)} + \underbrace{\int -q(\mathbf{x}^{(0)}) \log(p_\theta(\mathbf{x}^{(0)}))\, d\mathbf{x}^{(0)}}_{L_{CE}(p_\theta)}$$

Training of μ_θ and Σ_θ

Minimize the Cross Entropy Loss

$$L_{CE}(\, p_\theta(\mathbf{x}^{(0)}) := -\int q(\mathbf{x}^{(0)}) \log \left(p_\theta(\mathbf{x}^{(0)}) \right) \, d\mathbf{x}^{(0)}$$

Minimize the Cross Entropy Loss

$$L_{CE}(\, p_\theta(\mathbf{x}^{(0)}) := - \int q(\mathbf{x}^{(0)}) \log \left(p_\theta(\mathbf{x}^{(0)}) \right) \, d\mathbf{x}^{(0)}$$

Observation: Marginal Distribution

$$p_\theta(\mathbf{x}^{(0)}) = \int p_\theta(\mathbf{x}^{(0\ldots T)}) \, d\mathbf{x}^{(1\ldots T)}$$

Training of μ_θ and Σ_θ

Minimize the Cross Entropy Loss

$$\boxed{L_{CE}(\,p_\theta(\mathbf{x}^{(0)})\,) := -\int q(\mathbf{x}^{(0)}) \log\left(p_\theta(\mathbf{x}^{(0)})\right)\, d\mathbf{x}^{(0)}}$$

Observation: Marginal Distribution

$$p_\theta(\mathbf{x}^{(0)}) = \int p_\theta(\mathbf{x}^{(0\ldots T)})\, d\mathbf{x}^{(1\ldots T)}$$

Simplification II: Jensen Inequality

$$L_{CE}(p_\theta) \leq -\mathbb{E}_{q(\mathbf{x}^{(0\ldots T)})}\left[\log \frac{q(\mathbf{x}^{(1\ldots T)} \mid \mathbf{x}^{(0)})}{p_\theta(\mathbf{x}^{(0\ldots T)})}\right]$$

...after some manipulations :)

Reformulated Loss Function

$$\mathcal{L} = \mathcal{L}_T + \sum_{t=1}^{T-1} \mathcal{L}_t + \mathcal{L}_0$$

where,

$$\mathcal{L}_T = \mathbb{E}_{q(\mathbf{x}^{(0\dots T)})}\left[D_{KL}\left(q(\mathbf{x}^{(T)} \mid \mathbf{x}^{(0)}) \quad || \quad p_\theta(\mathbf{x}^{(T)})\right)\right]$$
$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}^{(0\dots T)})}\left[D_{KL}\left(q(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t+1)}, \mathbf{x}^{(0)}) \quad || \quad p_\theta(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t+1)})\right)\right]$$
$$\mathcal{L}_0 = \mathbb{E}_{q(\mathbf{x}^{(0\dots T)})}\left[-\log(p_\theta(\mathbf{x}^{(0)} \mid \mathbf{x}^{(1)}))\right]$$

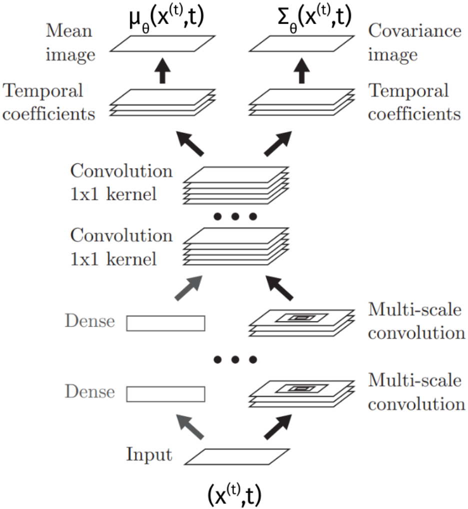
Observation

Note that in $\mathcal{L}_T$ does not depend on θ. L_t allows an explicit formulation being the forward transition gaussian and the reverse transition gaussian with parameterized mean and convariance. The term $\mathcal{L}_0$ has a closed form.

Neural Network that estimates μ_θ and Σ_θ

Description of the model

- Simple model, only convolutions, no residual layers.
- For each input $x \in \mathbb{R}^n$ and t, the model returns a vector $y = f_\theta(x) \in \mathbb{R}^{2n}$ that stores the mean and a diagonal covariance μ_θ, Σ_θ.
- The diffusions rate β_t are trained as the other paramters and fixed.

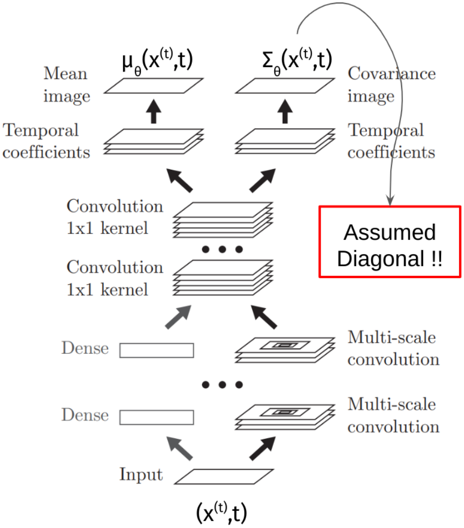


Proposed Neural Network for CIFAR10 image generation. T=1000

Neural Network that estimates μ_θ and Σ_θ

Description of the model

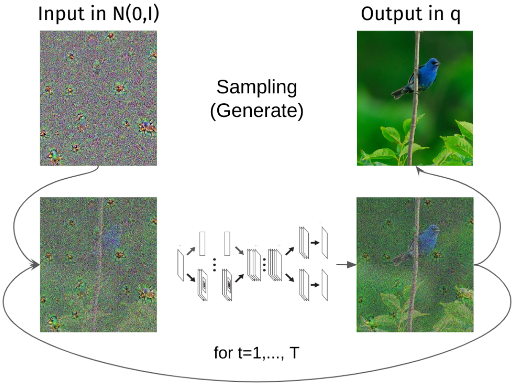
- Simple model, only convolutions, no residual layers.
- For each input $x \in \mathbb{R}^n$ and t, the model returns a vector $y = f_\theta(x) \in \mathbb{R}^{2n}$ that stores the mean and a diagonal covariance $\mu_\theta, \Sigma_\theta$.
- The diffusions rate β_t are trained as the other paramters and fixed.



Proposed Neural Network for CIFAR10 image generation. T=1000

Sampling or Generative Stage

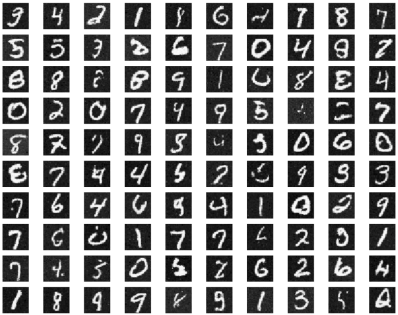
1. Set $t = T$ and pick a random image $x^{(t)} \in \mathbb{R}^n$ from $\mathcal{N}(0, I)$;
2. Deduce mean and covariance from $y = f_\theta(x^{(t)}, t)$
3. Pick a random image $x^{(t-1)}$ from $\mathcal{N}(\mu, \Sigma)$ and set $t = t - 1$;
4. Iterate 2. and 3. until $t = 0$.



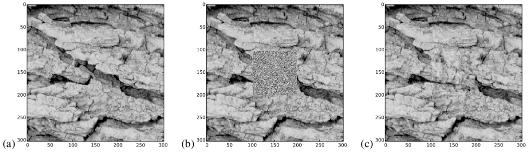
Proposed Neural Network for CIFAR10 image generation. T=1000

Experiments[5]

MNIST



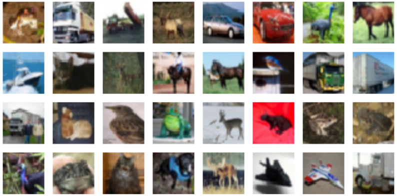
Bark Dataset



[5] Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics".

Experiments[6]

CIFAR10 (original)



CIFAR10 (generated)

[6] Sohl-Dickstein et al., “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”.

Timeline

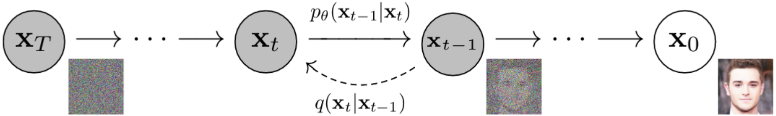
2015) ...*Non-equilibrium Thermodynamics*. Sohl-Dickstein et al. ICML. ✓

2020) *Denoising Diffusion Probabilistic Models*. Ho et al. NeurIPS.

2021) *Score-Based Generative Modeling Through SDE*. Song et al. ICLR.

Denoising Diffusion Probabilistic Model

Small technical improvements highly impact the performance...[7]



Simplification I: Diagonal Uniform Covariance Matrix

$$p_\theta(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}) = \mathcal{N}\left(\mathbf{x}^{(t-1)};\, \boldsymbol{\mu}_\theta\left(\mathbf{x}^{(t)}, t\right),\, \boldsymbol{\Sigma}_\theta\left(\mathbf{x}^{(t)}, t\right)\right)$$

where

$$\boldsymbol{\Sigma}_\theta\left(\mathbf{x}^{(t)}, t\right) = \sigma_t^2 I$$

[7] Ho, Jain, and Abbeel, "Denoising Diffusion Probabilistic Models".

Previous work aims at estimating $\boldsymbol{\mu}_\theta(\mathbf{x}^{(t)}, t)$ and $\boldsymbol{\Sigma}_\theta(\mathbf{x}^{(t)}, t)$.

Simplification II: Estimating the commited error

$$\boldsymbol{\mu}_\theta(\mathbf{x}^{(t)}) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}^{(t)} - \frac{1-\beta_t}{\sqrt{\alpha_t}} \quad \boldsymbol{\varepsilon}_\theta(\mathbf{x}^{(t)}, t)\right)$$

Previous work aims at estimating $\boldsymbol{\mu}_\theta(\mathbf{x}^{(t)}, t)$ and $\boldsymbol{\Sigma}_\theta(\mathbf{x}^{(t)}, t)$.

Simplification II: Estimating the commited error

$$\boldsymbol{\mu}_\theta(\mathbf{x}^{(t)}) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}^{(t)} - \frac{1-\beta_t}{\sqrt{\alpha_t}}\ \boxed{\boldsymbol{\varepsilon}_\theta(\mathbf{x}^{(t)}, t)}\right)$$

$\boldsymbol{\varepsilon}_\theta(\mathbf{x}^{(t)}, t)$ DNN (U-Net) with learnable parameters!!

Previous work aims at estimating $\boldsymbol{\mu}_\theta(\mathbf{x}^{(t)}, t)$ and $\boldsymbol{\Sigma}_\theta(\mathbf{x}^{(t)}, t)$.

Simplification II: Estimating the commited error

$$\boldsymbol{\mu}_\theta(\mathbf{x}^{(t)}) = \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}^{(t)} - \frac{1-\beta_t}{\sqrt{\alpha_t}} \boxed{\boldsymbol{\varepsilon}_\theta(\mathbf{x}^{(t)}, t)} \right)$$

$\boldsymbol{\varepsilon}_\theta(\mathbf{x}^{(t)}, t)$ DNN (U-Net) with learnable parameters!!

Simplification III: Training on random istants t

$$\mathcal{L}_{simple} = \mathbb{E}_{t, \mathbf{x}^{(0)}, \boldsymbol{\varepsilon}} \left[\left\| \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta \left(\sqrt{1-\alpha_t}\mathbf{x}^{(0)} + \sqrt{\alpha_t}\boldsymbol{\varepsilon}, t \right) \right\| \right]$$

where

$$t \sim \mathcal{U}\{1, \ldots, T\}, \quad \mathbf{x}^{(0)} \sim q, \quad \varepsilon \sim \mathcal{N}(0, I)$$

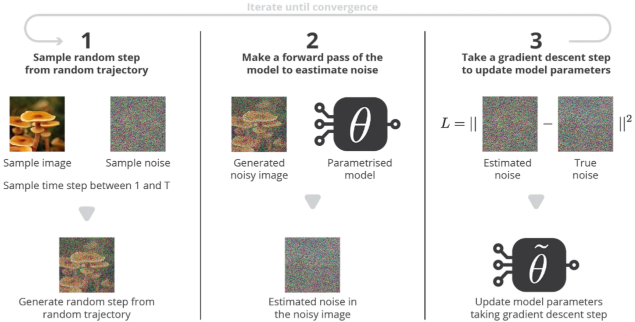
Training and Sampling Procedure

Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  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 $t = T, \dots, 1$ **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 $\mathbf{x}_0$



[8]

[8] **Credits** https://towardsdatascience.com/understanding-diffusion-probabilistic-models-dpms-1940329d6048

Experiments: Sample Quality

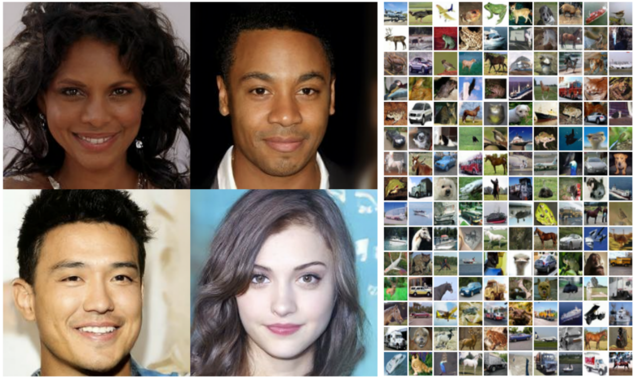


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

Objective	IS	FID
$\tilde{\boldsymbol{\mu}}$ prediction (baseline)		
L, learned diagonal $\boldsymbol{\Sigma}$	7.28 ± 0.10	23.69
L, fixed isotropic $\boldsymbol{\Sigma}$	8.06 ± 0.09	13.22
$\lVert \tilde{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}_\theta \rVert^2$	–	–
$\boldsymbol{\epsilon}$ prediction (ours)		
L, learned diagonal $\boldsymbol{\Sigma}$	–	–
L, fixed isotropic $\boldsymbol{\Sigma}$	7.67 ± 0.13	13.51
$\lVert \tilde{\boldsymbol{\epsilon}} - \boldsymbol{\epsilon}_\theta \rVert^2$ (L_{simple})	$\mathbf{9.46 \pm 0.11}$	$\mathbf{3.17}$

Metrics for CIFAR10

Note.

1. Low FID (Frechet Implicit Distance) $\Rightarrow$ high quality
2. Training improved

Experiments: Diffusion vs GAN/VAE

"Diffusion models get comparable result to Generative Adversarial Networks and Variational Autoencoders"

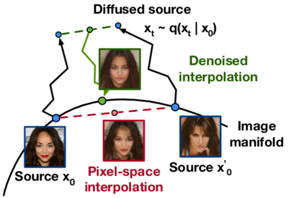
Table 1: CIFAR10 results. NLL measured in bits/dim.

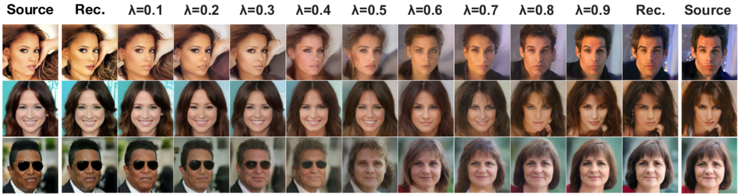
Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [55]	8.87 ± 0.12	25.32	
SNGAN [39]	8.22 ± 0.05	21.7	
SNGAN-DDLS [4]	9.09 ± 0.10	15.42	
StyleGAN2 + ADA (v1) [29]	$\mathbf{9.74} \pm 0.05$	3.26	
Ours (L, fixed isotropic $\mathbf{\Sigma}$)	7.67 ± 0.13	13.51	≤ 3.70 (3.69)
Ours (L_{simple})	9.46 ± 0.11	**3.17**	≤ 3.75 (3.72)

CIFAR10 results. NLL measured in bits/dim[9]

[9] Ho, Jain, and Abbeel, "Denoising Diffusion Probabilistic Models".

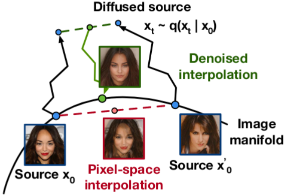
Experiments: Images Interpolation

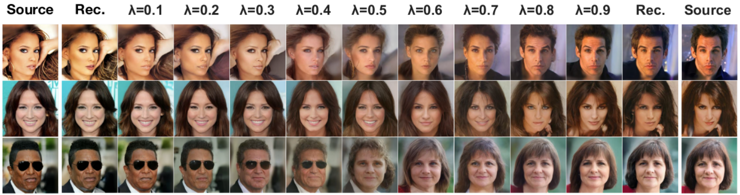




$$\mathbf{x}_{\lambda}^{(T)} := \lambda \mathbf{x}_{\mathtt{source_r}}^{(T)} + (1-\lambda)\mathbf{x}_{\mathtt{source_l}}^{(T)}, \quad \lambda \in [0,1]$$

Experiments: Images Interpolation





$$\mathbf{x}_\lambda^{(T)} := \lambda \mathbf{x}^{(T)}_{\texttt{source}_\texttt{r}} + (1-\lambda)\mathbf{x}^{(T)}_{\texttt{source}_\texttt{l}}, \quad \lambda \in [0,1]$$

$$\mathbf{x}_\lambda^{(0)} \sim p_\theta(\mathbf{x}^{(T)}), \quad \text{by diffusion}$$

Timeline

2015) *...Non-equilibrium Thermodynamics*. Sohl-Dickstein et al. ICML. ✓

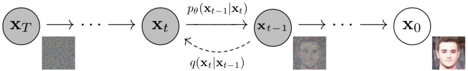
2020) *Denoising Diffusion Probabilistic Models*. Ho et al. NeurIPS.✓

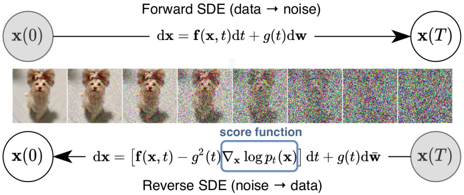
2021) *Score-Based Generative Modeling Through SDE*. Song et al. ICLR.

Score-Based Generative Modeling Through SDE (no details)

Continuous Time

Discrete Time





[10]

Stochastic Process

Continous sequence of $\mathbf{x}$ indexed by $t \in [0, T]$, where $\mathbf{x}(t)$ has distribution $p_t(\mathbf{x})$;

[10] Song et al., "Score-based generative modeling through stochastic differential equations".

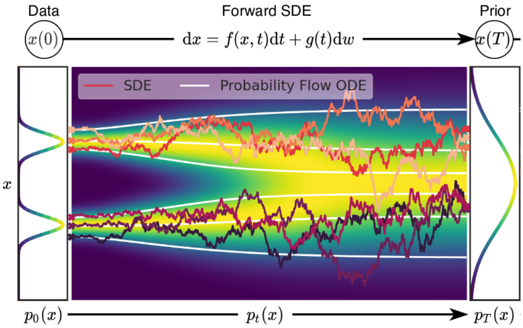
Continous Diffusion Process

$$d\mathbf{x}(t) = f(\mathbf{x}, t)dt + g(t)\, d\mathbf{w}(t)$$

1. $f(\mathbf{x}, t)$ drift coefficient.
2. $g(t)$ diffusion coefficient.
3. $\mathbf{w}(t)$, Weiner Process

Weiner Process

$$\mathbf{w}(t) - \mathbf{w}(s) \sim \mathcal{N}(0, (t-s)I)$$



Reverse Process explicity given by $\nabla_{\mathbf{x}} p_t(\mathbf{x})$

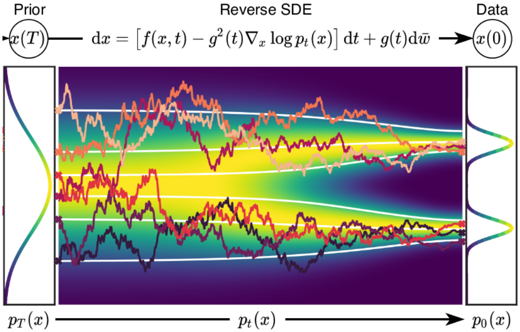
Theorem

Reverse Process is still a diffusion process

$$dx(t) = \bar{f}(\mathbf{x}, t)\, dt + g(t)\, d\bar{\mathbf{w}}(t)$$

where

- $\bar{f}(\mathbf{x}, t) = f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} log(p_t(\mathbf{x}))$
- $\bar{\mathbf{w}}(t) = \mathbf{w}(T - t)$, reverse Weiner Process



Conclusion

Broader Impacts

1. Diffusion models, as generative models, can be used for malicious porpouse. Fake images can become less detectable.
2. Diffusion models reflects the biases in the dataset with which they are trained. Hence, using generated images for training other models can produce a **fade-in** effect.

[11] Ho, Jain, and Abbeel, "Denoising Diffusion Probabilistic Models".

1. Diffusion models, as generative models, can be used for malicious porpouse. Fake images can become less detectable.
2. Diffusion models reflects the biases in the dataset with which they are trained. Hence, using generated images for training other models can produce a **fade-in** effect.

"If samples from generative models trained on these datasets proliferate throughout the internet, then these biases will only be reinforced further.[11]"

[11] Ho, Jain, and Abbeel, "Denoising Diffusion Probabilistic Models".

Summary

1. Diffusion Generative Models (from 2015 to 2022)

2. Overview on CLIP model for representation learning.

3. The Diffusion models inside DALL-E2

Thanks for the attention

Fabio Brau

Scuola Superiore Sant'Anna, Pisa

fabio.brau@santannapisa.it

retis.santannapisa.it/~f.brau

linkedin.com/in/fabio-brau

TELECOMMUNICATIONS, COMPUTER ENGINEERING, AND PHOTONICS INSTITUTE

Sant'Anna School of Advanced Studies – Pisa

ECCELLENZA MIUR 2018-2022

ROBOTICS & AI

Sant'Anna Scuola Universitaria Superiore Pisa

Retis Real-Time Systems Laboratory

Proof Details

Proof of Explicit Representation of Forward Diffusion Process

Let us proceeding by induction by assuming $\mathbf{x}^{(t)} = \sqrt{1-\alpha_t}\,\mathbf{x}^{(0)} + \sqrt{\alpha_t}\,\boldsymbol{\varepsilon}$ where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, I)$ and where $\alpha_t = 1 - \prod_{i=0}^{t}(1-\beta_i)$.

$$
\begin{aligned}
\mathbf{x}^{(t+1)} &= \sqrt{1-\beta_{t+1}}\,\mathbf{x}^{(t)} + \sqrt{\beta_{t+1}}\,\boldsymbol{\varepsilon}_{t+1} \\
&= \sqrt{1-\beta_{t+1}}\,\left(\sqrt{1-\alpha_t}\,\mathbf{x}^{(0)} + \sqrt{\alpha_t}\,\boldsymbol{\varepsilon}\right) + \sqrt{\beta_{t+1}}\,\boldsymbol{\varepsilon}_{t+1} \\
&= \sqrt{\left(\prod_{i=0}^{t+1}(1-\beta_i)\right)}\mathbf{x}^{(0)} + \sqrt{(1-\beta_{t+1})\alpha_t + \beta_{t+1}}\,\tilde{\boldsymbol{\varepsilon}}
\end{aligned}
\tag{2}
$$

where the last term of the summation is obtained by observing that, since $\sqrt{(1-\beta_{t+1})\alpha_t}\,\boldsymbol{\varepsilon}$ and $\sqrt{\beta_{t+1}}\,\boldsymbol{\varepsilon}_{t+1}$ are independent, then the variance of their sum (that still has a gaussian distribution) is given by $(1-\beta_{t+1})\alpha_t + \beta_{t+1}$.