Diffusion Models: DALL-E

Deep Learning and Neural Networks: Advanced Topics

Fabio Brau

March 1, 2023

Scuola Superiore Sant'Anna, Pisa.

TELECOMMUNICATIONS, COMPUTER ENGINEERING, AND PHOTONICS INSTITUTE

Sant'Anna School of Advanced Studies – Pisa

Retis Real-Time Systems Laboratory

Introduction

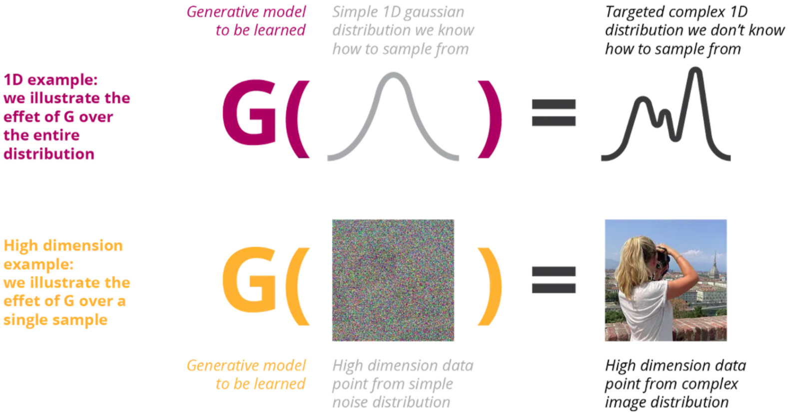
Diffusion Models

DALL-E

Broader Impacts

Introduction

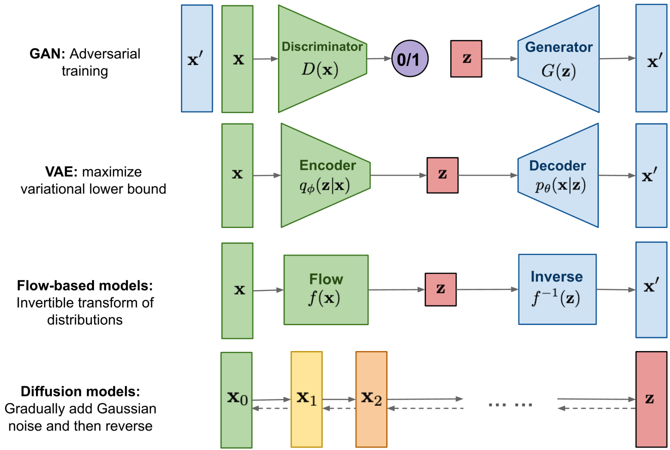
Generative Models



Schema of generative models[1]

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

Generative Models



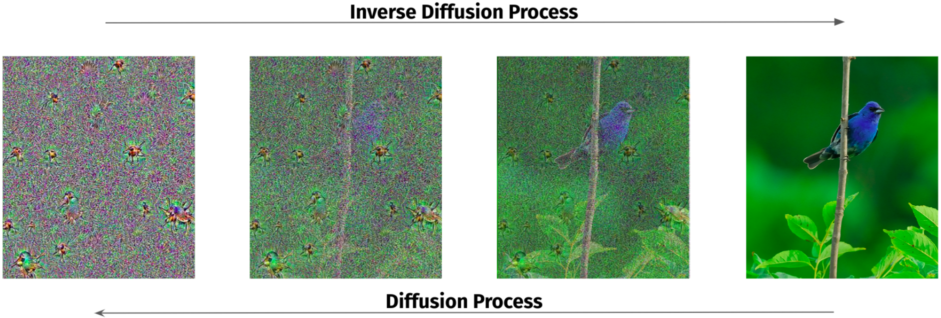
Overview of different generative models[2]

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

Diffusion Models

Overview

Diffusion models are generative models that aim at denoising data



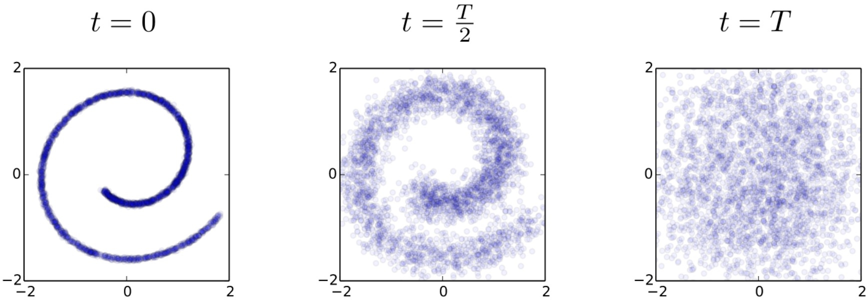
Timeline

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.

Deep Unsupervised Learning using Non-Equilibrium Thermodynamics



Diffusion process as a **Markov Chain** with **Continuous State Space** and **Discrete Time**.[3]

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

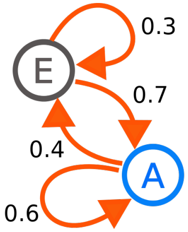
Reminder: Markov Chains with Discrete Time

Informal Definition

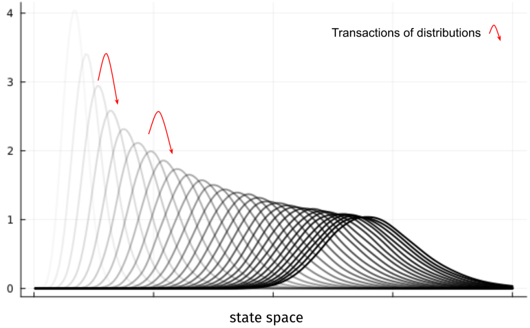
A sequence of random variables $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(t)}, \cdots$, such that:

- $\mathbf{x}^{(t)} \in S$, 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



Reminder: MCDT with Discrete State Space

Definition
A sequence $\left\{\mathbf{x}^{(t)}\right\}_{t\in\mathbb{N}} \subseteq S$, a matrix $P = \left(p_{ij}\right)$.

- Discrete state space: $S = \{s_0, \cdots, s_n, \cdots\}$
- Markov Property: $\mathbf{x}^{(t+1)}$ not dep. $\mathbf{x}^{(0)}, \cdots, \mathbf{x}^{(t-1)}$.
- Transaction Matrix: $\mathbb{P}\left(\mathbf{x}^{(t+1)} = s_j | \mathbf{x}^{(t)} = s_i\right) = p_{ij}$

Reminder: MCDT with Discrete State Space

Definition

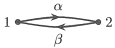
A sequence $\{\mathbf{x}^{(t)}\}_{t\in\mathbb{N}} \subseteq S$, a matrix $P = (p_{ij})$.

- Discrete state space: $S = \{s_0, \cdots, s_n, \cdots\}$
- Markov Property: $\mathbf{x}^{(t+1)}$ not dep. $\mathbf{x}^{(0)}, \cdots, \mathbf{x}^{(t-1)}$.
- Transaction Matrix: $\mathbb{P}\left(\mathbf{x}^{(t+1)} = s_j | \mathbf{x}^{(t)} = s_i\right) = p_{ij}$

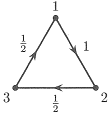
P is a stochastic matrix!

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

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



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



Reminder: DTMC with Continuous State Space

Let assume $\mathbf{x}, \mathbf{y} \in S$ where S continuous state space (e.g. $S = \mathbb{R}^d$).

Joint Distribution $p(\mathbf{x}, \mathbf{y})$

$$\mathbb{P}\left(\mathbf{x} \in A \,|\, \mathbf{y} \in B\right) = \int_A \int_B p\left(\mathbf{x}, \mathbf{y}\right)\, d\mathbf{x}\, d\mathbf{y}$$

Transactional Kernel $p(\mathbf{x} \,|\, \mathbf{y})$

$$p\left(\mathbf{x}, \mathbf{y}\right) = p(\mathbf{x} \,|\, \mathbf{y})\, p\left(\mathbf{y}\right)$$

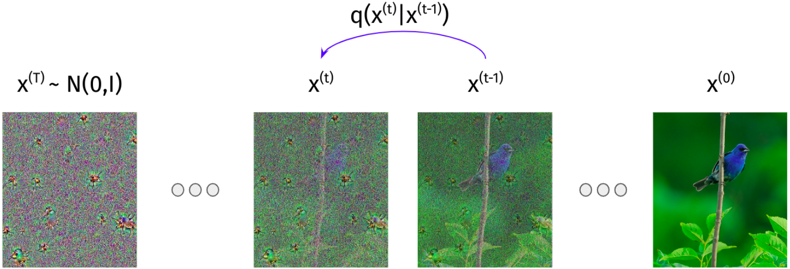
Marginal Distribution $p(\mathbf{x})$

$$p\left(\mathbf{x}\right) = \int_S p\left(\mathbf{x}, \mathbf{y}\right)\, d\mathbf{y} = \int_S p(\mathbf{x} \,|\, \mathbf{y})\, p\left(\mathbf{y}\right)\, d\mathbf{y}$$

Forward Diffusion Process

"Adding noise to data..."

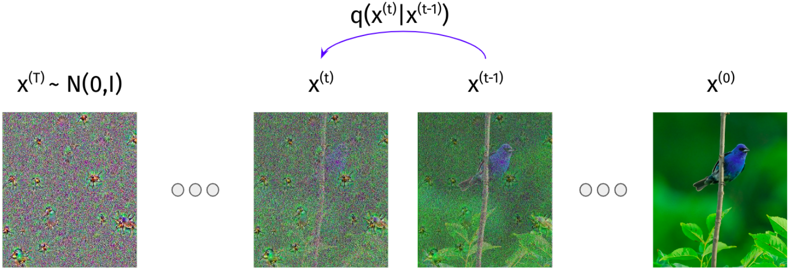
- Data Distribution: $\mathbf{x}^{(0)} \sim q$
- Transaction Kernel: $q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1-\beta_t}\mathbf{x}^{(t-1)}; \beta_t I\right)$
- Variance Scheduler: $\beta_1, \cdots, \beta_T \in (0, 1]$



Forward Diffusion Process

"Adding noise to data..."

- Data Distribution: $\mathbf{x}^{(0)} \sim q$ — Not Analytic!!
- Transaction Kernel: $q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1-\beta_t}\mathbf{x}^{(t-1)}; \beta_t I\right)$
- Variance Scheduler: $\beta_1, \cdots, \beta_T \in (0, 1]$ — $\beta_T = 1$



Forward Diffusion Process: Explicit Representation

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

Observation: Many small noisy steps $\approx$ Large Noisy step

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

where

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

Forward Diffusion Process: Distribution Representation

Markov property allows breaking up distributional Representation...

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

Forward Diffusion Process: Distribution Representation

Markov property allows breaking up distributional Representation...

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

Forward Diffusion Process: Distribution Representation

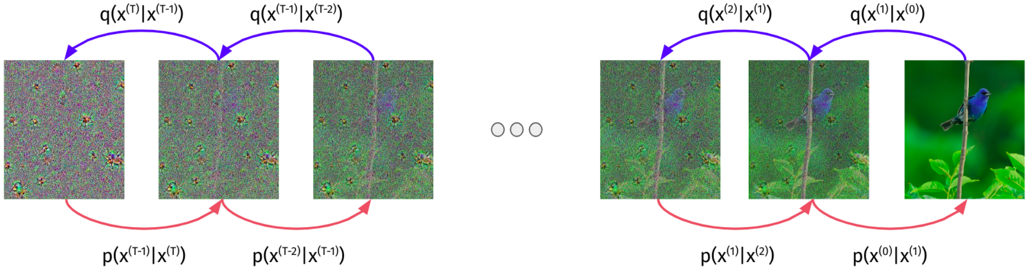
Markov property allows breaking up distributional Representation...

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

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



Learned Reverse Process

Reverse Diffusion Process

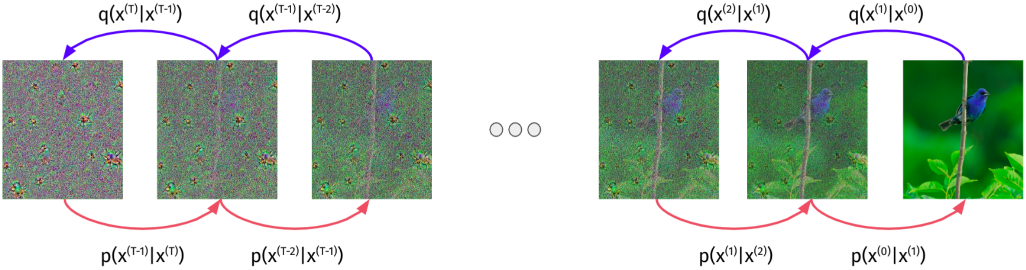
Fixed Forward Process

Initial Distribution

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

Gaussian Transaction Kernel

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



Learned Reverse Process

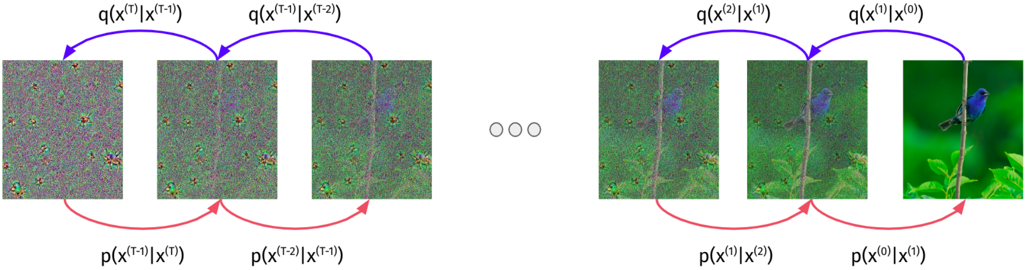
Fixed Forward Process

Initial Distribution

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

Gaussian Transaction Kernel

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



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

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

Learned Reverse Process

Reverse Diffusion Process

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

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

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

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

Reverse Diffusion Process

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[4]

If $|\beta_i - \beta_{i+1}| \approx 0$, i.e. diffusion **slow enough**, then

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

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

Reverse Diffusion Process

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[4]

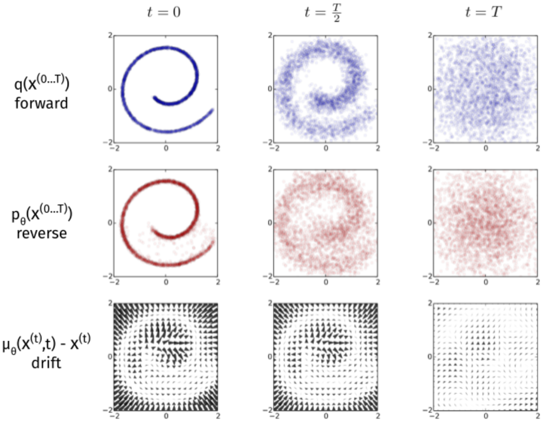
If $|\beta_i - \beta_{i+1}| \approx 0$, i.e. diffusion **slow enough**, then

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

Mean $\boldsymbol{\mu}_\theta$ and covariance $\boldsymbol{\Sigma}_\theta$ have to be learned!!

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

Visualization of Diffusion Process: 2D dimensional case



[5]

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

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

Aim

Seach 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 $\boldsymbol{\Sigma}_\theta$

Aim

Seach 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

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

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

Aim

Seach 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??

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

Aim

Seach 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

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

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!!

Training of μ_θ and Σ_θ

Aim: 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 $\boldsymbol{\mu}_\theta$ and Σ_θ

Aim: 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

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

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

Aim: 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

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

Training of μ_θ and Σ_θ

Minimize the Cross Entropy Loss

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

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

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

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

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

Training of μ_θ and Σ_θ

...after some algebraic steps :)

Reformulated Loss Function

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

where,

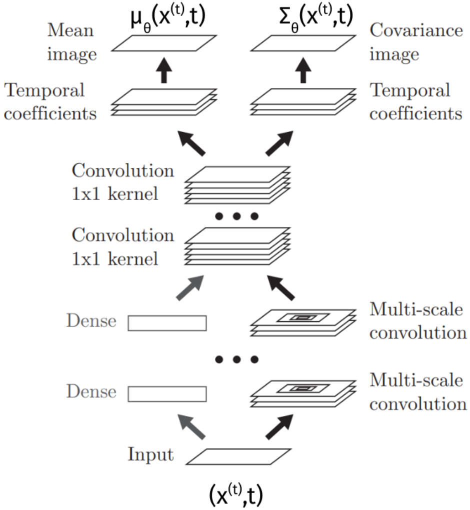
$$\mathcal{L}_T = \mathbb{E}_{q(\mathbf{x}^{(0\ldots T)})}\left[D_{KL}\left(q(\mathbf{x}^{(T)} \mid \mathbf{x}^{(0)}) \quad || \quad p_\theta(\mathbf{x}^{(T)})\right)\right]$$

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

$$\mathcal{L}_0 = \mathbb{E}_{q(\mathbf{x}^{(0\ldots T)})}\left[p_\theta(\mathbf{x}^{(0)} \mid \mathbf{x}^{(1)})\right]$$

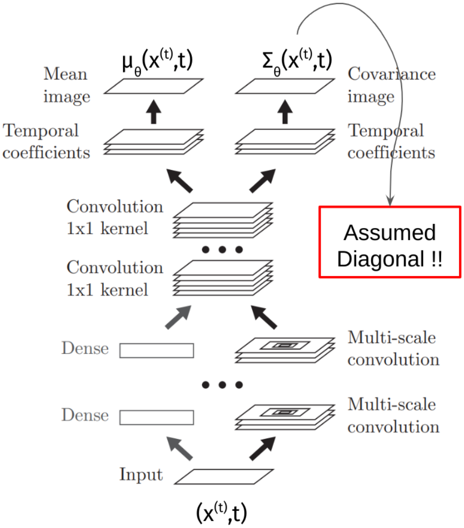
Note.
$\mathcal{L}_T$ is constant.
$\mathcal{L}_0$, $\mathcal{L}_t$ explicit.

Neural Network that estimate μ_θ and Σ_θ



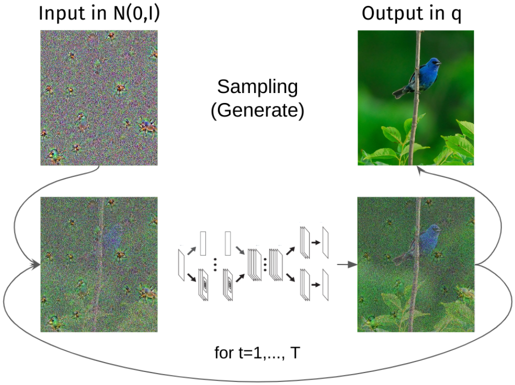
Prposed Neural Network for CIFAR10 image generation. T=1000

Neural Network that estimate μ_θ and Σ_θ



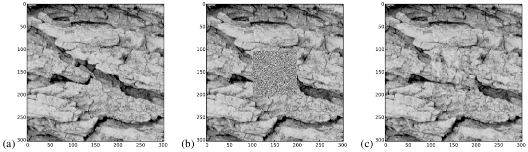
Prposed Neural Network for CIFAR10 image generation. T=1000

Sampling or Generative Stage

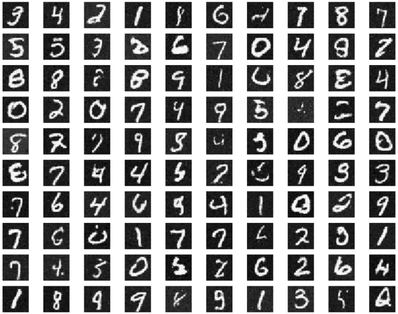


Experiments

Bark Dataset



MNIST

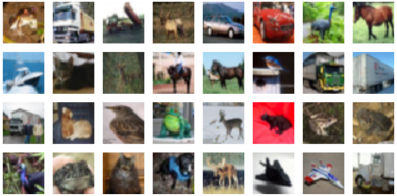


6

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

Experiments

CIFAR10 (original)



CIFAR10 (generated)

.[7]

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

Timeline

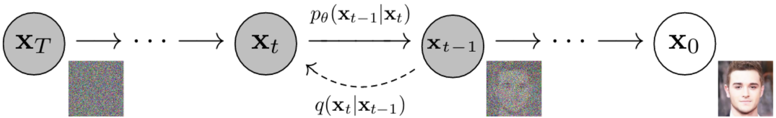
2015) *...Non-equilibrium Thermodynamics.* Sohl-Dickstein et al. ICML ✓

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...[8]



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

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

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

Denoising Diffusion Probabilistic Model

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

Simplification II: Estimating the commited error

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

Denoising Diffusion Probabilistic Model

Previous work aim 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!!

Denoising Diffusion Probabilistic Model

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

Simplification II: Estimating the commited error

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

$\varepsilon_\theta(\mathbf{x}^{(t)}, t)$ 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

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

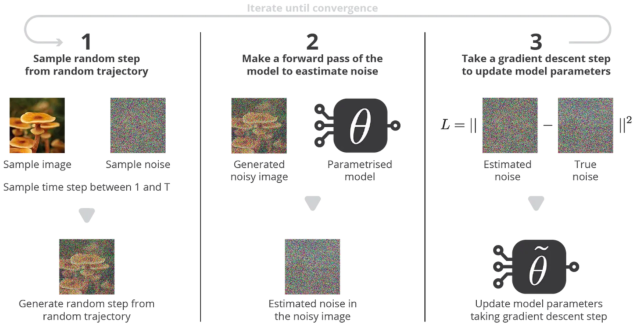
Training and Sampling Procedure

Algorithm 1 Training

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

Algorithm 2 Sampling

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



[9]

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

Experiments: Sample Quality

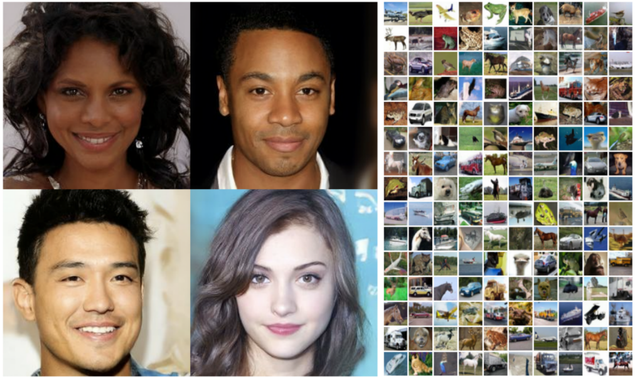


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

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

Metrics for CIFAR10

Note.

1. High 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"

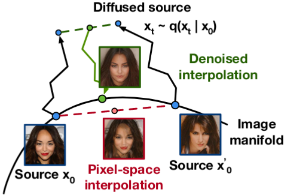
Table 1: CIFAR10 results. NLL measured in bits/dim.

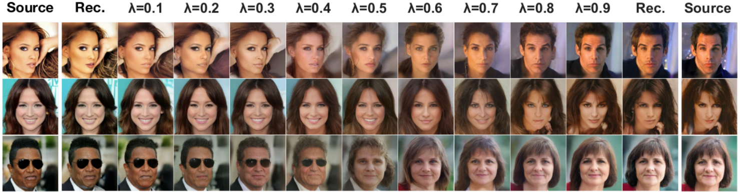
Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	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[10]

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

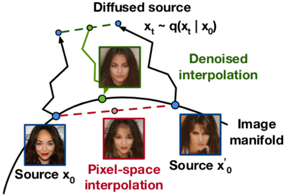
Experiments: Images Interpolation

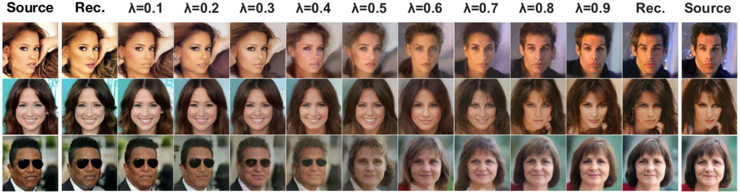




$$\mathbf{x}_{\lambda}^{(T)} := \lambda \mathbf{x}_{\mathtt{source_r}}^{(T)} + (1-\lambda)\mathbf{x}_{\mathtt{source_l}}^{(T)}, \quad \lambda \in [0,1]$$

Experiments: Images Interpolation





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

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

Timeline

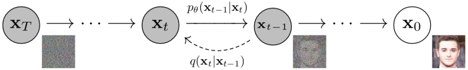
2015) *...Non-equilibrium Thermodynamics*. Sohl-Dickstein et al. ICML. ✓

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

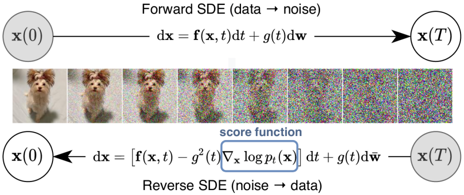
2021) *Score-Based Generative Modeling Through SDE*. Song et al. ICLR.

Score-Based Generative Modeling Through SDE (no details)

Discrete Time Markov Chain with **Continuous** State Space



Continuous Time Markov Chain with **Continuous** State Space



[11]

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

DALL-E

Samples

Timeline

Broader Impacts

Image Generation

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

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

Image Generation

1. Diffusion models, as generative models, can be used for malicius porpouse. Fake images can become less detectable.
2. Diffusion models relfect the biases in the dataset in which they are trianed. 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.[12]*"*

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

CLIP Model

"We also found discrepancies across gender and race for people categorized into the 'crime' and 'non-human' categories..."[13]

[13] Radford et al., "Learning Transferable Visual Models From Natural Language Supervision".

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 $(1-\beta_{t+1})\alpha_t + \beta_{t+1}$.