

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.

TELECOMMUNICATIONS,
COMPUTER
ENGINEERING,
AND PHOTONICS
INSTITUTE



Sant'Anna
School of Advanced Studies – Pisa



Introduction

Diffusion Models

Conclusion

Introduction



Generative Models

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**.

*Generative model
to be learned*

*Simple 1D gaussian
distribution we know
how to sample from*

*Targeted complex 1D
distribution we don't know
how to sample from*

$$G(\text{)} = \text{ }$$

$$G(\text{)} = \text{ }$$

*Generative model
to be learned*

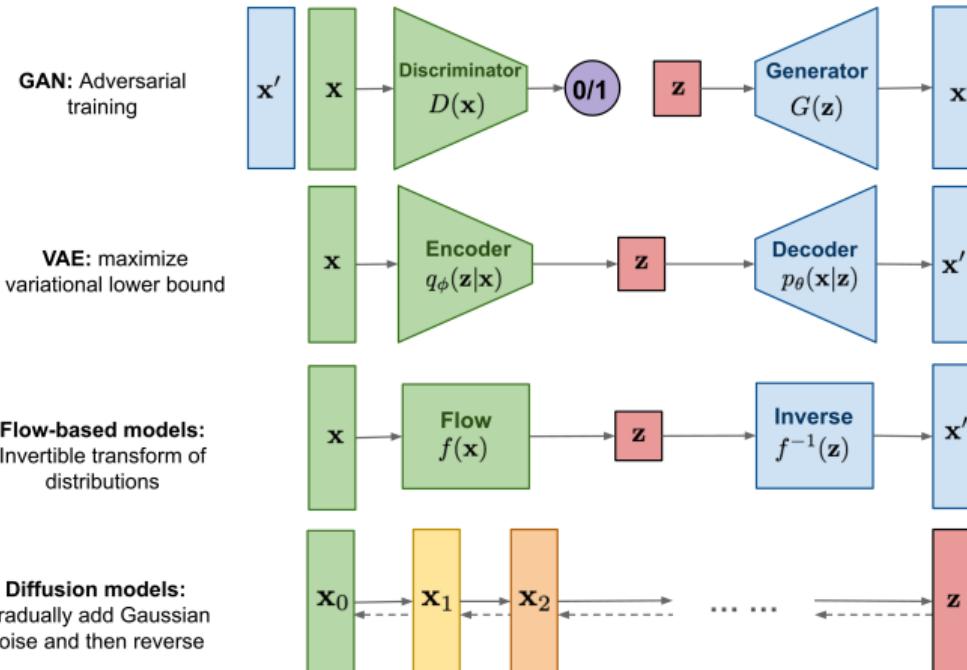
*High dimension data
point from simple
noise distribution*

*High dimension data
point from complex
image distribution*

General Schema of generative models.

¹Credits: <https://towardsdatascience.com/understanding-diffusion-probabilistic-models-dpms-1940329d6048>

Generative Models



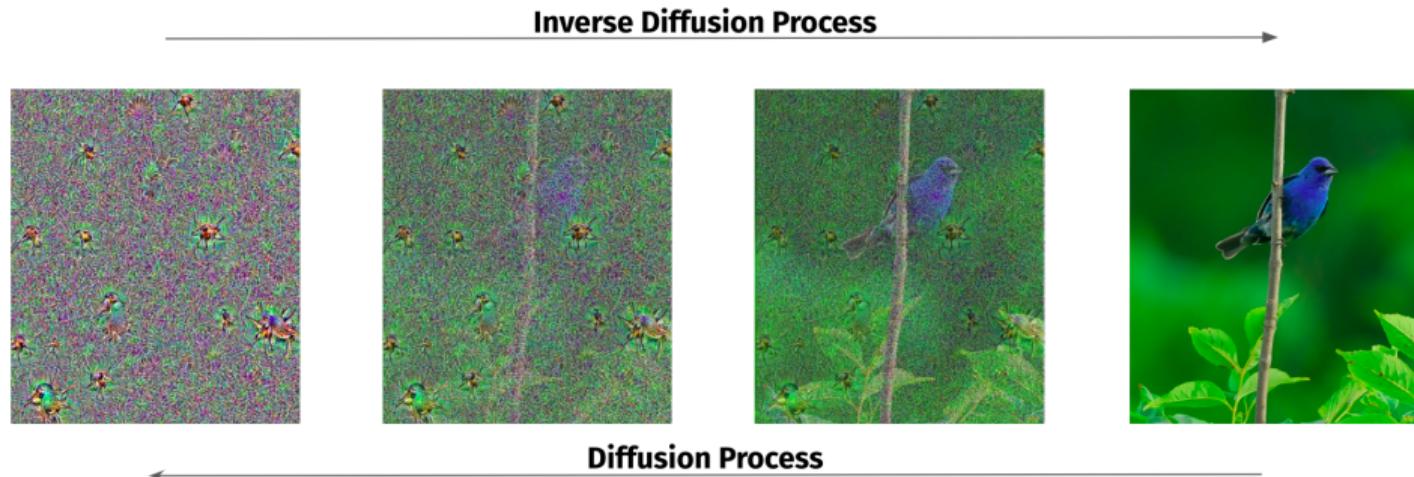
Overview of different generative models¹

¹Credits <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

Diffusion Models



Diffusion models are generative models that learn a inverse diffusion process

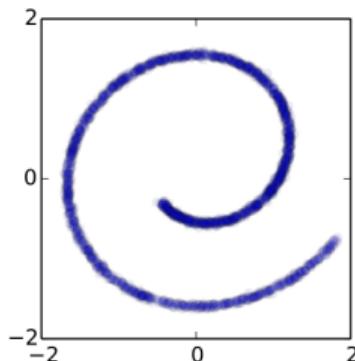


Timeline

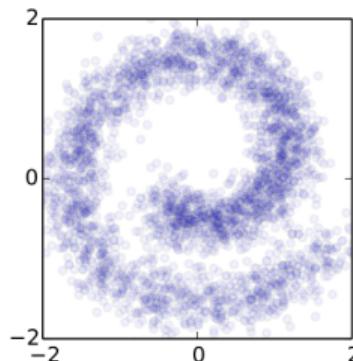
- 2015) *Deep Unsupervised Learning with Non-equilibrium Thermodynamics*. Sohl-Dickstein et al. ICML.
- 2020) *Denoising Diffusion Probabilistic Models*. Ho et al. NeurIPS.
- 2021) *Score-Based Generative Modeling Through SDE*. Song et al. ICLR.

Deep Unsupervised Learning using Non-Equilibrium Thermodynamics

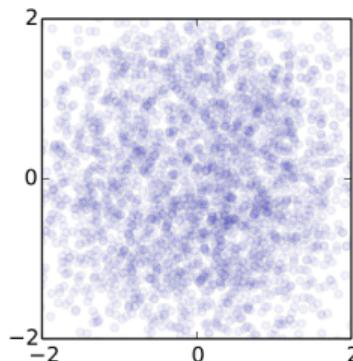
$t = 0$



$t = \frac{T}{2}$



$t = T$



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

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

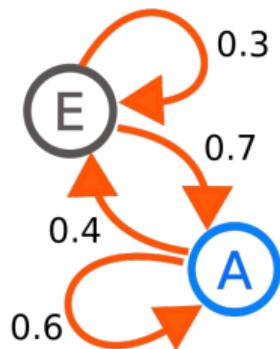
Reminder: Markov Chains with Discrete Time

Informal Definition

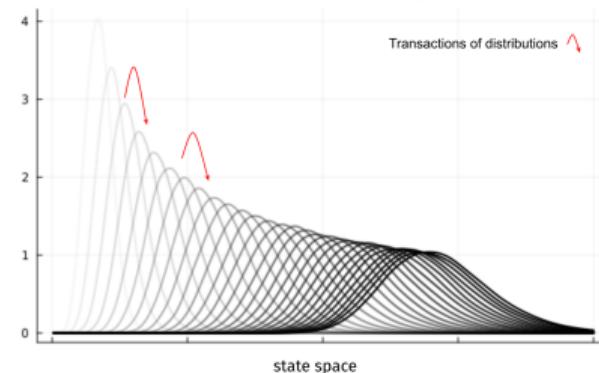
A sequence of random variables $x^{(0)}, x^{(1)}, \dots, x^{(t)}, \dots$, such that

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

Discrete State Space S



Continuous State Space S



Reminder: Discrete Time Markov Chain with Discrete State Space

Definition (Discrete Space Markov Chain)

Let $S = \{s_0, \dots, s_n, \dots\}$ 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

- $\mathbf{x}^{(t+1)}$ not dep. $\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(t-1)}$ (Markov Property).
- $\mathbb{P}(\mathbf{x}^{(t+1)} = s_j | \mathbf{x}^{(t)} = s_i) = p_{ij}$ (Homogeneous Property)

Reminder: Discrete Time Markov Chain with Discrete State Space

Definition (Discrete Space Markov Chain)

Let $S = \{s_0, \dots, s_n, \dots\}$ be a discrete state space.

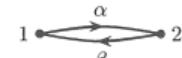
A sequence $\{\mathbf{x}^{(t)}\}_t \subseteq S$ is a Markov chain if there exists a transition matrix $P = (p_{ij})$ such that

P is a stochastic matrix!

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

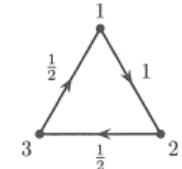
• $\mathbf{x}^{(t+1)}$ not dep. $\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(t-1)}$ (Markov Property).

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



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

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



Reminder: DTMC with Continuous State Space

Definition (DTMC Continuous state) A sequence $\{\mathbf{x}^{(t)}\} \subseteq S$ of random variables, with labeled distributions $q(\mathbf{x}^{(i)})$ (stochastic process), is a Markov Chain if there exists a parameterized map $\kappa_t : S \times S \rightarrow [0, 1]$ 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. $\mathbf{x}^{(t+1)}$ depends on $\mathbf{x}^{(t)}$ and not on previous states.

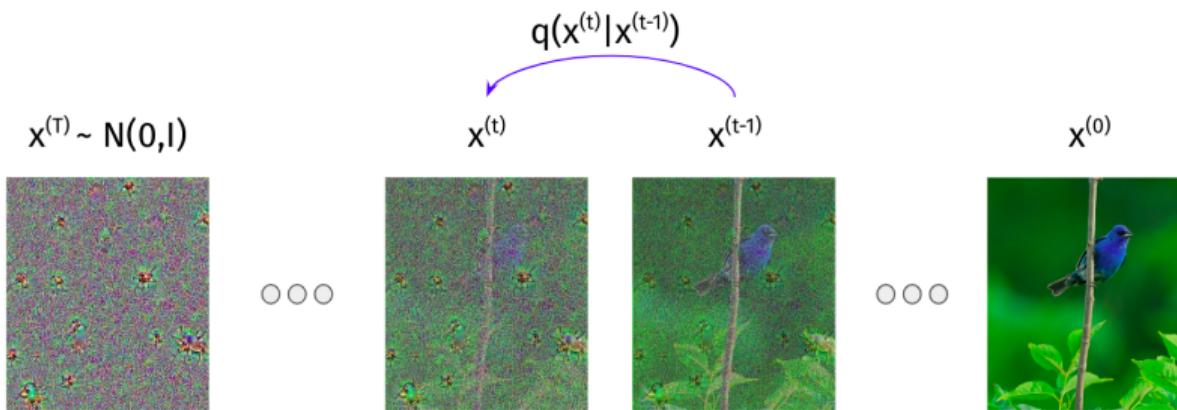
Notation

- We call $(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)})$ trajectory
- We call $q(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}) = \kappa_t$ transition kernel.
- We call $q((\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)})) = q(\mathbf{x}^{(0\dots T)})$ 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 (untractable) distribution $q(\mathbf{x}^{(0)})$;
- Transaction Kernel: $q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1 - \beta_t} \mathbf{x}^{(t-1)}; \beta_t I\right)$
- Variance Scheduler: $0 < \beta_1 < \dots < \beta_T < 1$

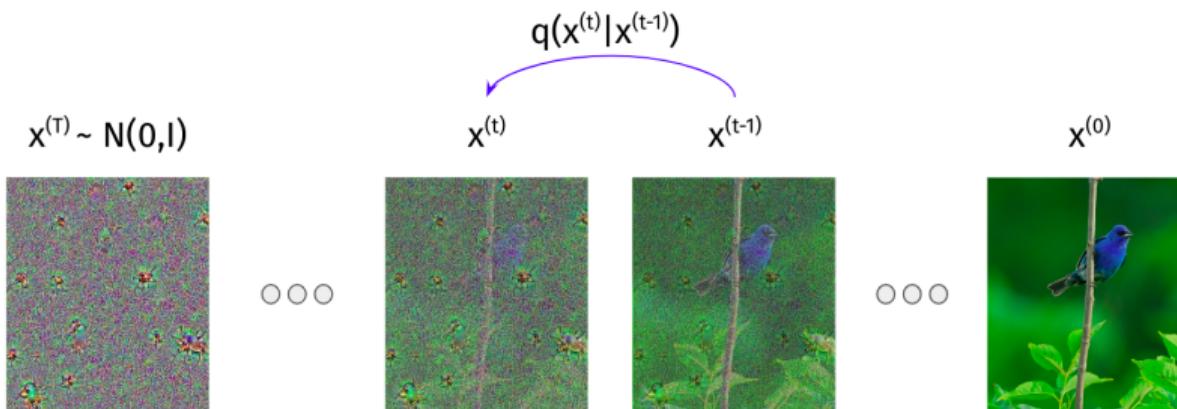


Forward Diffusion Process

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

- Data Distribution: $\mathbf{x}^{(0)}$ with (untractable) distribution $q(\mathbf{x}^{(0)})$;
- Transaction Kernel: $q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1 - \beta_t} \mathbf{x}^{(t-1)}; \beta_t I\right)$
- Variance Scheduler: $0 < \beta_1 < \dots < \beta_T < 1$

Not Analytic!!



Forward Diffusion Process: Explicit Representation

Observation 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: Many small noisy steps \approx Large Noisy step

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

where

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

Forward Diffusion Process: Distribution Representation

Markov property allows breaking up distributional Representation...

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

Forward Diffusion Process: Distribution Representation

Markov property allows breaking up distributional Representation...

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

Forward Diffusion Process: Distribution Representation

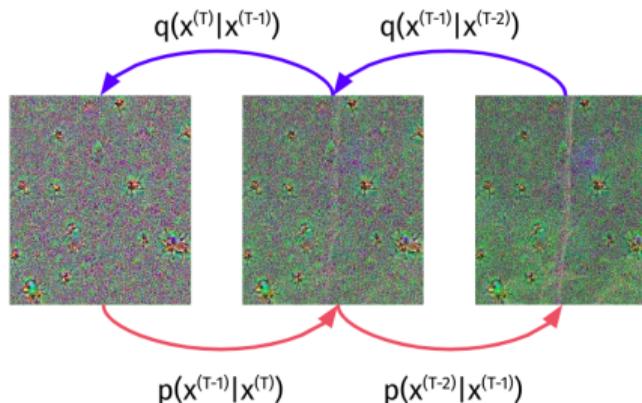
Markov property allows breaking up distributional Representation...

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

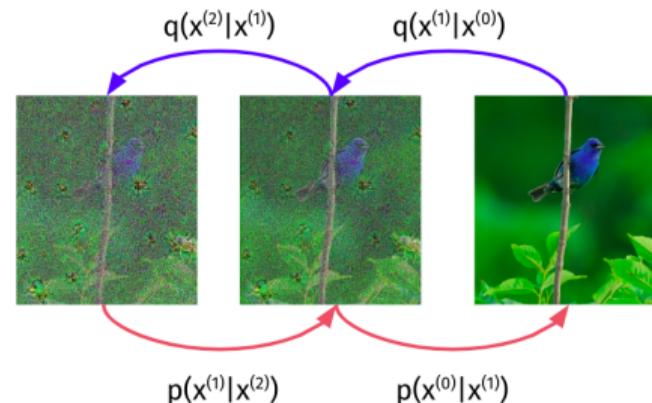
Distributional Representation

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

Reverse Diffusion Process



○ ○ ○

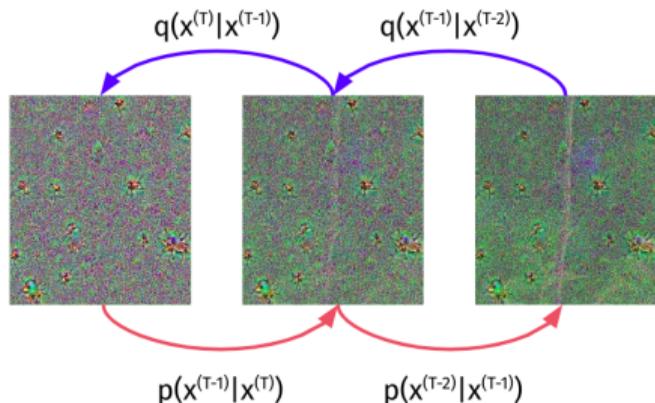


Reverse Diffusion Process

Fixed Forward Process

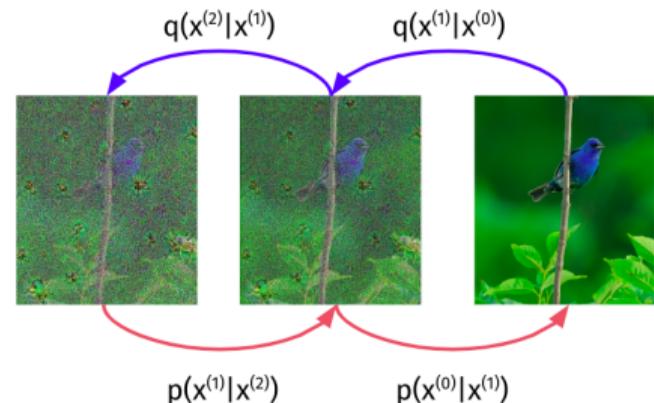
Initial Distribution

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



Gaussian Transaction Kernel

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

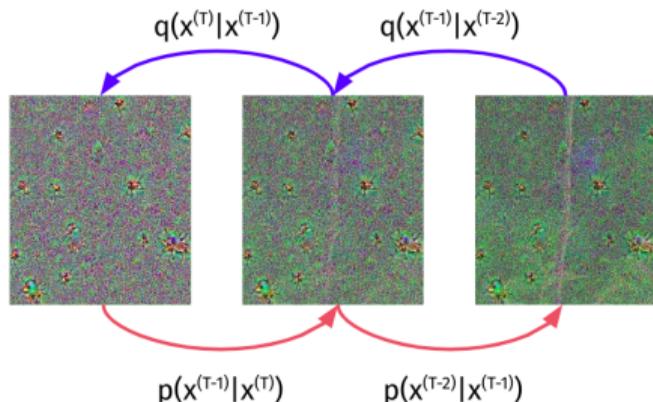


Reverse Diffusion Process

Fixed Forward Process

Initial Distribution

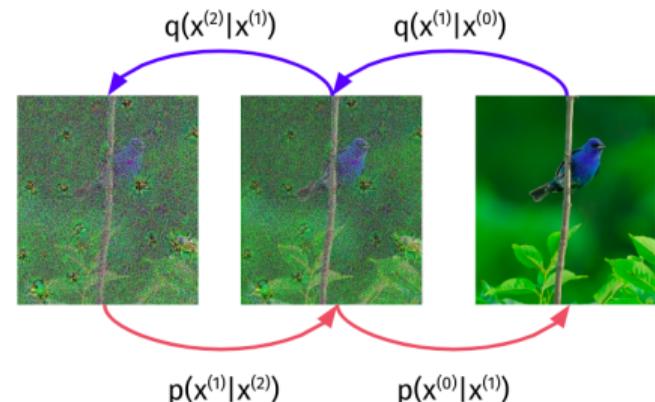
$$q(\mathbf{x}^{(0)})$$



Gaussian Transaction Kernel

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

○ ○ ○



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

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

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

³Sohl-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 DP³

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

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

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

Reverse Diffusion Process

Forward Diffusion Process

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

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

Reverse Diffusion Process

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

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

Theorem. Reverse of Gaussian DP is \approx Gaussian DP³

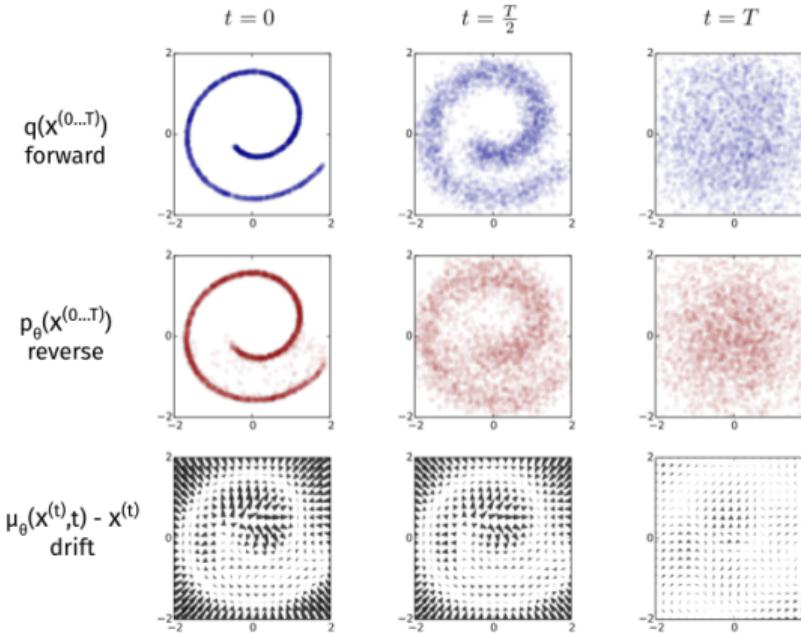
If $|\beta_i - \beta_{i+1}| \approx 0$, i.e. diffusion **slow enough**, then

$$q(\mathbf{x}^{(t-1)} | \mathbf{x}^{(t)}) \approx \mathcal{N}\left(\mathbf{x}^{(t-1)}; \mu_\theta(\mathbf{x}^{(t)}, t), \Sigma_\theta(\mathbf{x}^{(t)}, t)\right)$$

Mean μ_θ and covariance Σ_θ have to be learned!!

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

Visualization of Diffusion Process: 2D dimensional case



4

⁴Sohl-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 $\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T)}$ diffusion process

Estimated Reverse Process

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

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

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 $\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T)}$ 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

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

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

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

Method

Minimize the Kullback–Leibler Divergence

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

Easy??

Estimated Reverse Process

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

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

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

Method

Minimize the Kullback–Leibler Divergence

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

Easy??

Estimated Reverse Process

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

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

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



Training of μ_θ and Σ_θ

Goal: Deduce a tractable loss function

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

Training of μ_θ and Σ_θ

Goal: Deduce a tractable loss function

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

Simplification I: Minimize the Cross Entropy

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

Training of μ_θ and Σ_θ

Goal: Deduce a tractable loss function

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

Simplification I: Minimize the Cross Entropy

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

Training of μ_θ and Σ_θ

Minimize the Cross Entropy Loss

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

Training of μ_θ and Σ_θ

Minimize the Cross Entropy Loss

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

Observation: Marginal Distribution

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

Training of μ_θ and Σ_θ

Minimize the Cross Entropy Loss

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

Observation: Marginal Distribution

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

Simplification II: Jensen Inequality

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

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,

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

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

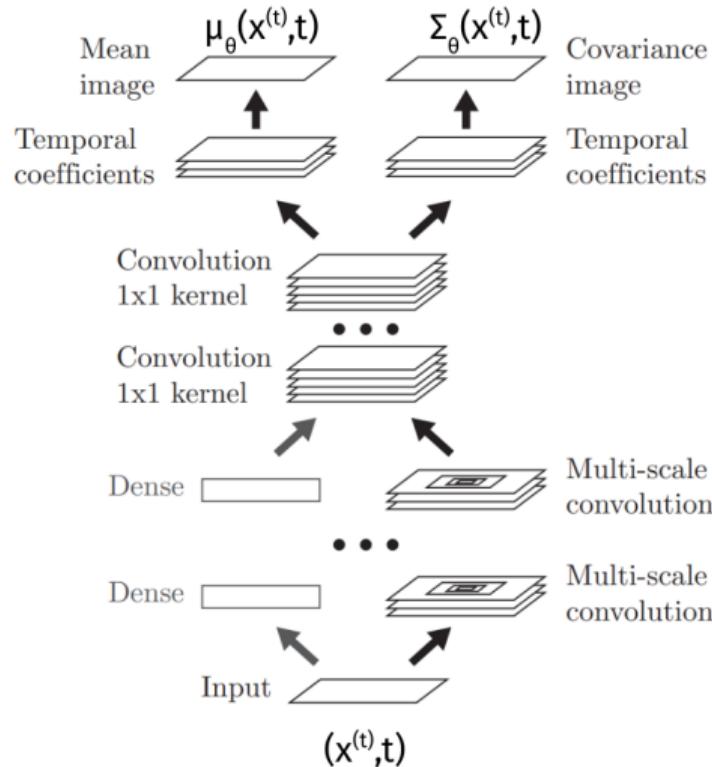
$$\mathcal{L}_0 = \mathbb{E}_{q(\mathbf{x}^{(0\dots T)})} \left[p_\theta(\mathbf{x}^{(0)} | \mathbf{x}^{(1)}) \right]$$

Note.

\mathcal{L}_T is constant.

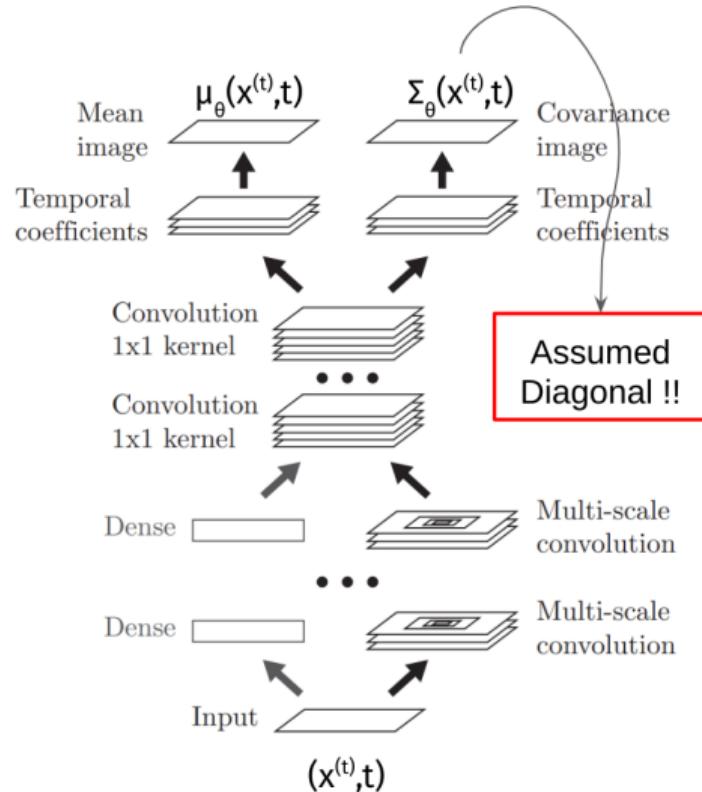
$\mathcal{L}_0, \mathcal{L}_t$ explicit.

Neural Network that estimate μ_θ and Σ_θ



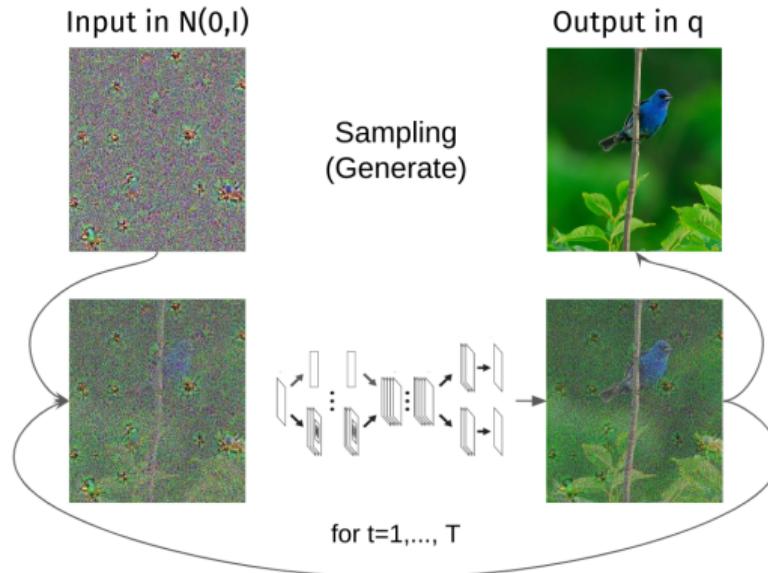
Proposed Neural Network for CIFAR10 image generation. T=1000

Neural Network that estimate μ_θ and Σ_θ

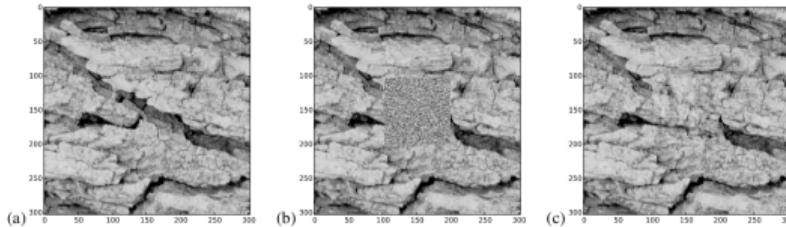


Proposed Neural Network for CIFAR10 image generation. T=1000

Sampling or Generative Stage



Experiments⁵



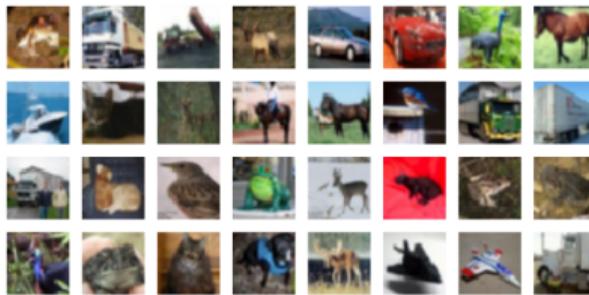
MNIST

3	4	2	1	8	6	7	7	8	7
5	5	3	2	6	7	0	4	8	2
8	8	2	8	9	1	6	8	E	4
0	2	0	7	4	9	5	4	2	7
8	2	1	9	3	6	5	0	6	0
E	7	4	4	5	2	6	9	3	3
7	6	4	6	9	4	1	0	2	9
7	0	0	1	7	7	4	2	0	1
7	4	3	0	5	2	6	2	6	4
1	8	9	9	8	9	1	3	5	4

⁵Sohl-Dickstein et al., “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”.

Experiments⁶

CIFAR10 (original)



CIFAR10 (generated)



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

Timeline

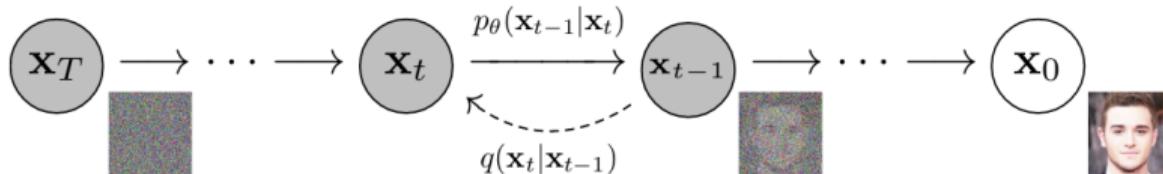
2015) ...*Non-equilibrium Thermodynamics*. Sohl-Dickstein et al. ICML. ✓

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

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

where

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

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

Previous work aims at estimating $\mu_\theta(\mathbf{x}^{(t)}, t)$ and $\Sigma_\theta(\mathbf{x}^{(t)}, t)$.

Simplification II: Estimating the committed error

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

Previous work aims at estimating $\mu_\theta(\mathbf{x}^{(t)}, t)$ and $\Sigma_\theta(\mathbf{x}^{(t)}, t)$.

Simplification II: Estimating the committed error

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

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

Denoising Diffusion Probabilistic Model

Previous work aims at estimating $\mu_\theta(\mathbf{x}^{(t)}, t)$ and $\Sigma_\theta(\mathbf{x}^{(t)}, t)$.

Simplification II: Estimating the committed error

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

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

Simplification III: Training on random instants t

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

where

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

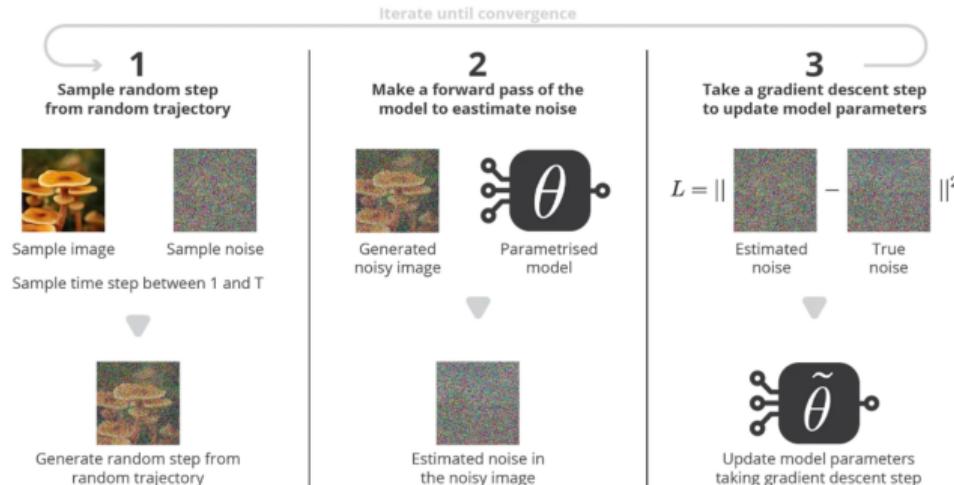
Training and Sampling Procedure

Algorithm 1 Training

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

Algorithm 2 Sampling

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



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 Σ	7.28 ± 0.10	23.69
L , fixed isotropic Σ	8.06 ± 0.09	13.22
$\ \tilde{\mu} - \tilde{\mu}_\theta\ ^2$	-	-
ϵ prediction (ours)		
L , learned diagonal Σ	-	-
L , fixed isotropic Σ	7.67 ± 0.13	13.51
$\ \tilde{\epsilon} - \epsilon_\theta\ ^2$ (L_{simple})	9.46 ± 0.11	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”

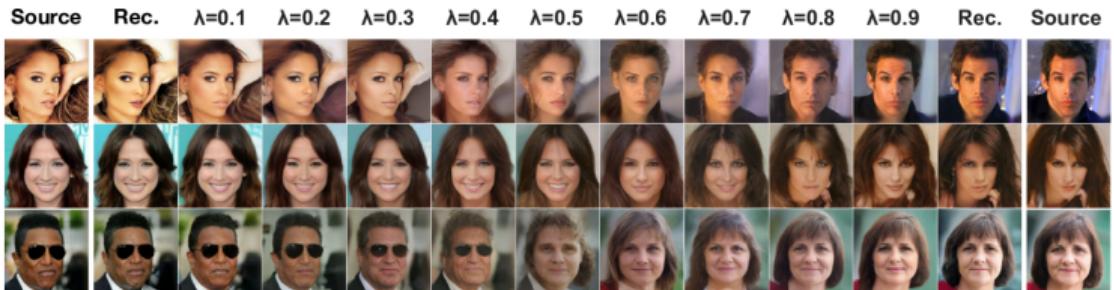
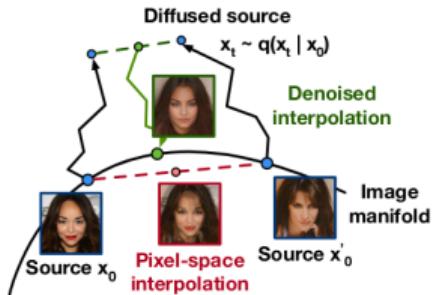
Table 1: CIFAR10 results. NLL measured in bits/dim.

Model	IS	FID	NLL Test (Train)
Conditional			
EBM [4]	8.30	37.9	
JEM [2]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [20]	10.06	2.67	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [2]			2.80
PixellQNN [44]	5.29	49.46	
EBM [4]	6.78	38.2	
NCSNv2 [56]			31.75
NCSN [53]	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) [20]	9.74 ± 0.05	3.26	
Ours (L , fixed isotropic Σ)	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/dim⁹

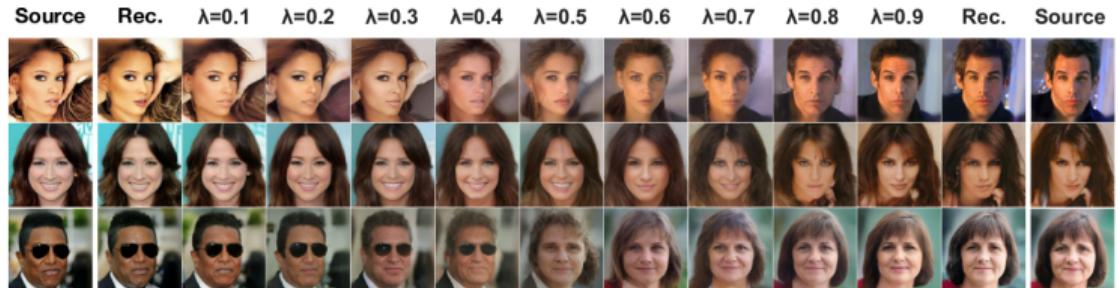
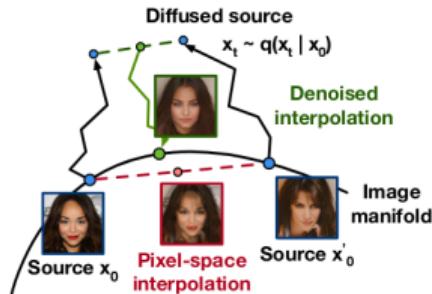
⁹ Ho, Jain, and Abbeel, “Denoising Diffusion Probabilistic Models”.

Experiments: Images Interpolation



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

Experiments: Images Interpolation



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

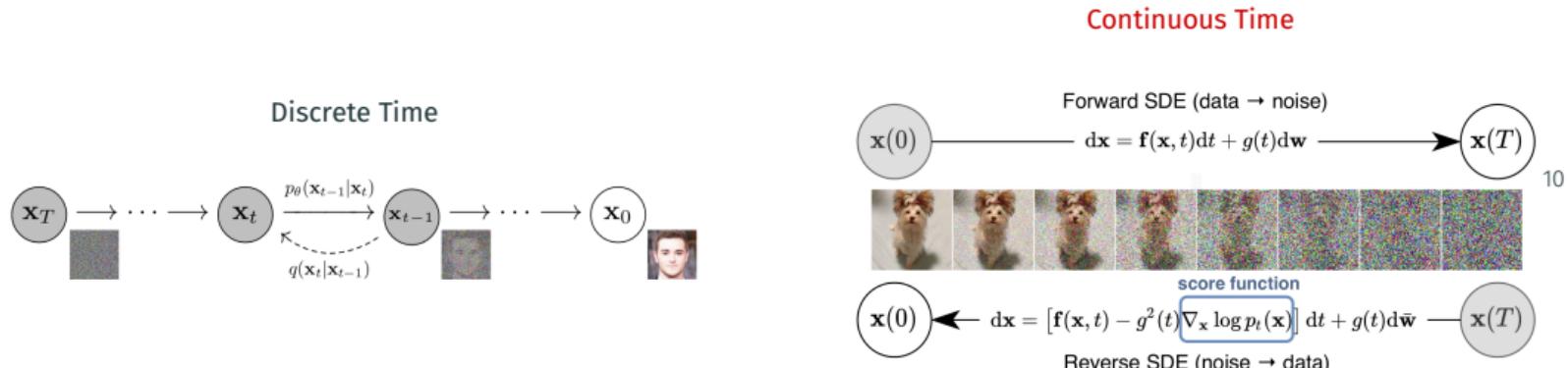
$$\mathbf{x}_\lambda^{(0)} \sim p_\theta(\mathbf{x}^{(T)}), \quad \text{by diffusion}$$

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)



Stochastic Process

Continuous 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".

Continuous Diffusion Process described by SDE

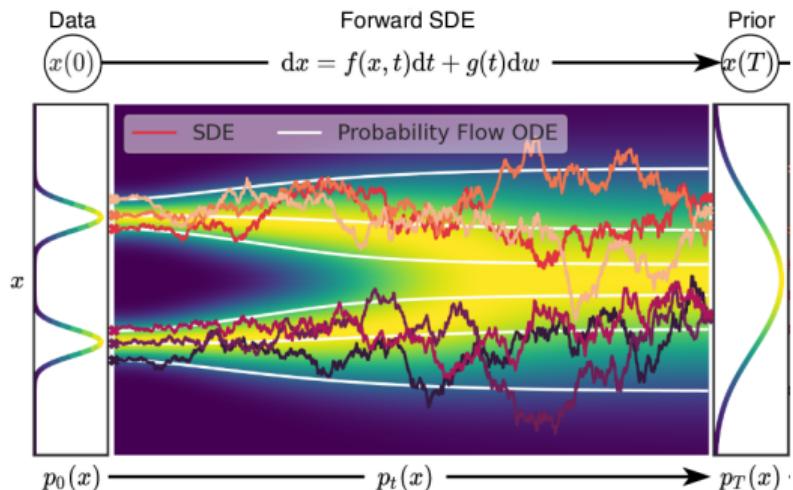
Continuous Diffusion Process

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

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 $\nabla_x p_t(x)$

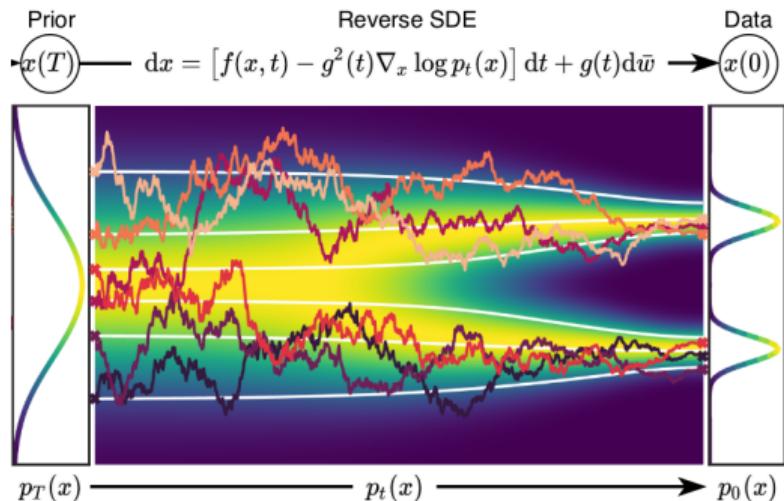
Theorem

Reverse Process is still a diffusion process

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

where

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



Conclusion



1. Diffusion models, as generative models, can be used for malicious porpose. 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.

¹¹Ho, Jain, and Abbeel, “Denoising Diffusion Probabilistic Models”.

1. Diffusion models, as generative models, can be used for malicious porpose. 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”

¹¹Ho, 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 $(1 - \beta_{t+1})\alpha_t + \beta_{t+1}$.

