

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



Introduction

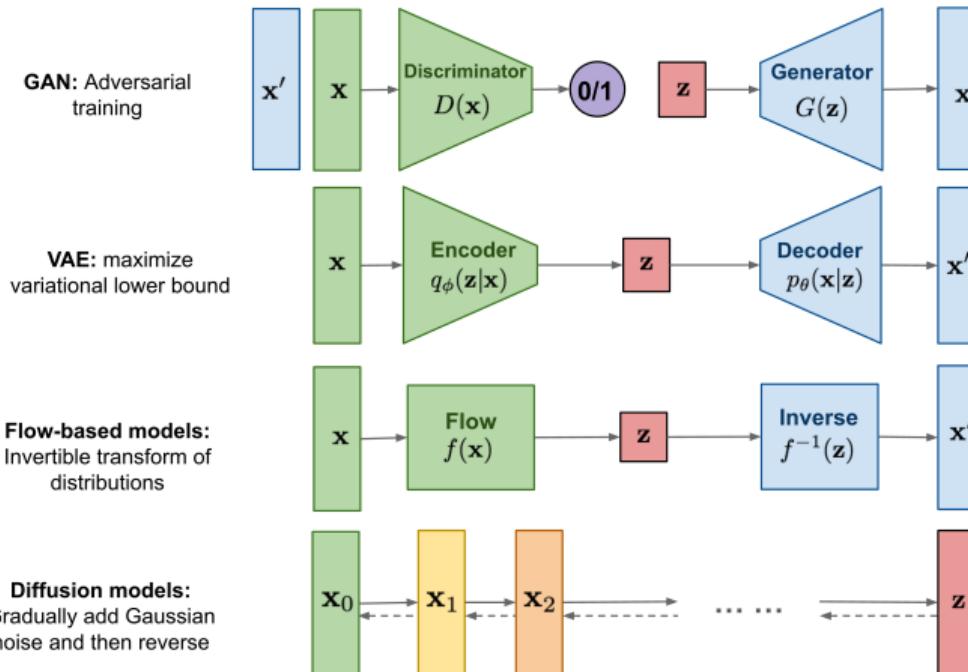
Diffusion Models

Broader Impacts

Introduction



Generative Models



Overview of different generative models¹

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

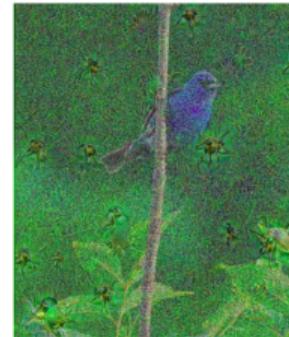
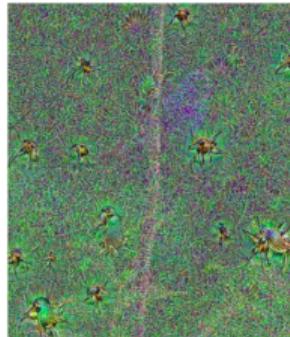
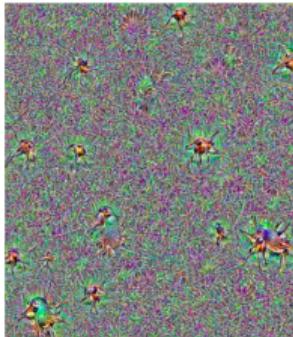
Diffusion Models



Overview

Diffusion models are generative models that aim at denoising data

Inverse Diffusion Process

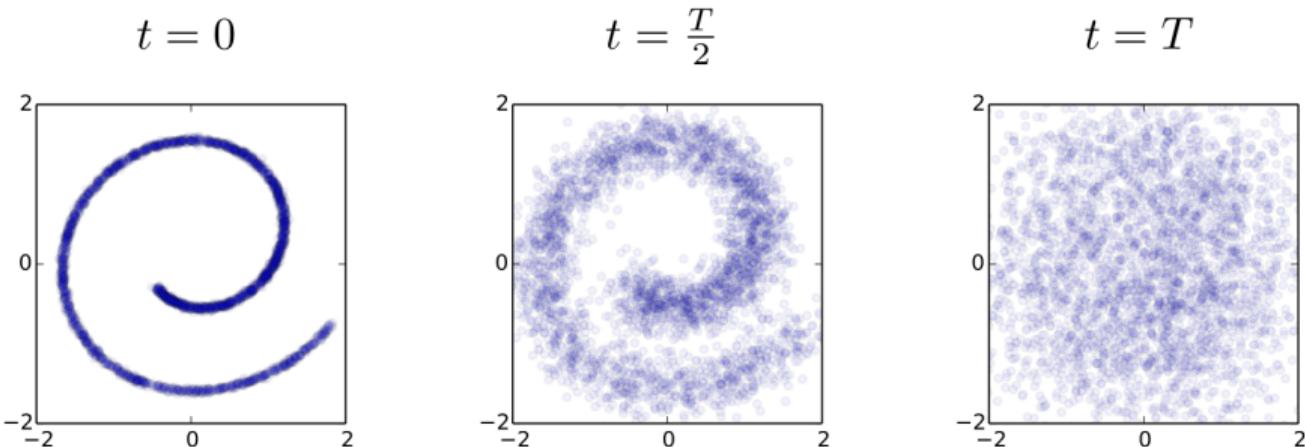


Diffusion Process

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.²

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

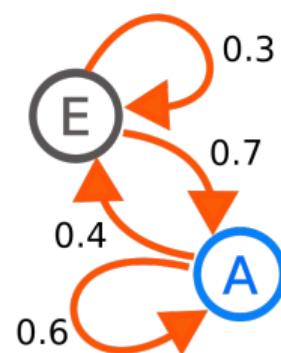
Reminder: Markov Chains with Discrete Time

Informal Definition

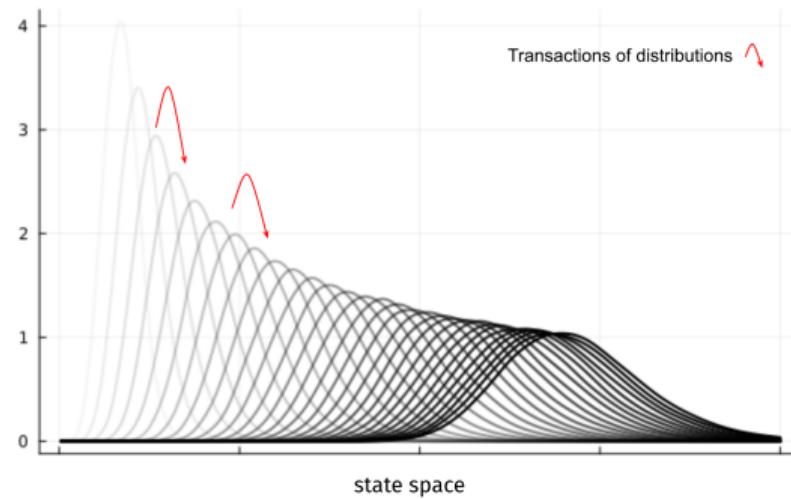
A sequence of random variables $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(t)}, \dots$, such that:

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

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

Reminder: MCDT with Discrete State Space

Definition

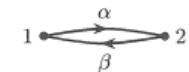
A sequence $\{\mathbf{x}^{(t)}\}_{t \in \mathbb{N}} \subseteq S$, a matrix $P = (p_{ij})$.

P is a stochastic matrix!

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

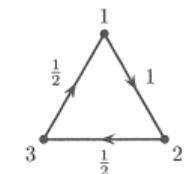
- Discrete state space: $S = \{s_0, \dots, s_n, \dots\}$

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



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

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



- Transaction Matrix: $\mathbb{P}(\mathbf{x}^{(t+1)} = s_j | \mathbf{x}^{(t)} = s_i) = p_{ij}$

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

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

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

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} \mid \mathbf{y}) p(\mathbf{y})$$

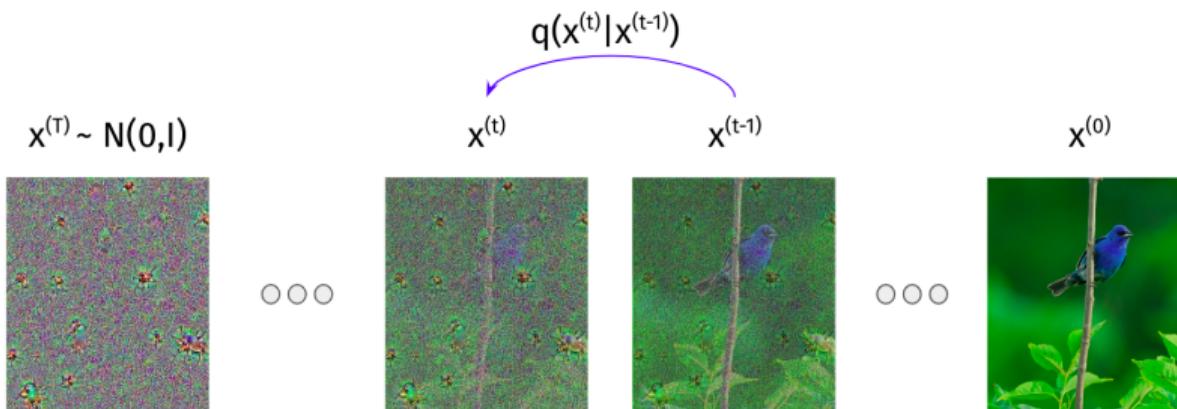
Marginal Distribution $p(\mathbf{x})$

$$p(\mathbf{x}) = \int_S p(\mathbf{x}, \mathbf{y}) \, d\mathbf{y} = \int_S p(\mathbf{x} \mid \mathbf{y}) p(\mathbf{y}) \, d\mathbf{y}$$

Forward Diffusion Process

“Adding noise to data...”

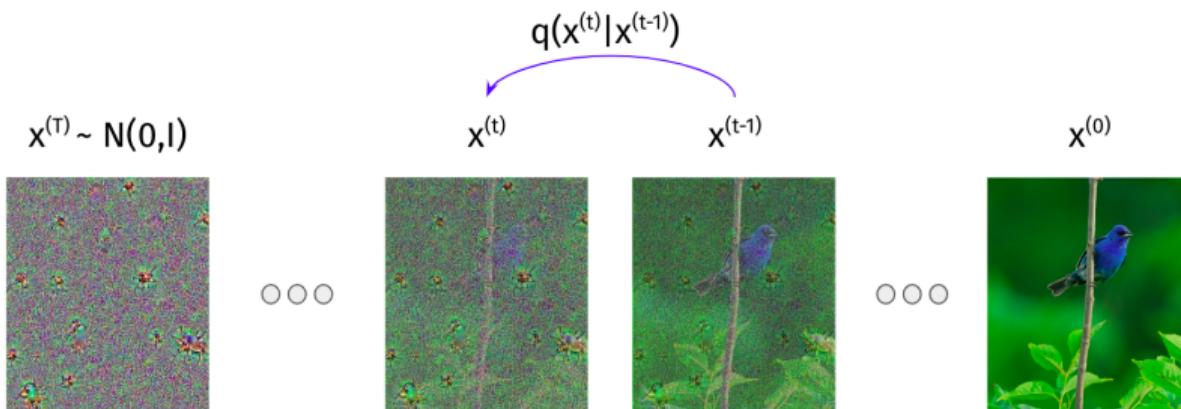
- Data Distribution: $\mathbf{x}^{(0)} \sim q$
- Transaction Kernel: $q(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)}) = \mathcal{N}(\mathbf{x}^{(t)}; \sqrt{1 - \beta_t} \mathbf{x}^{(t-1)}; \beta_t I)$
- Variance Scheduler: $\beta_1, \dots, \beta_T \in (0, 1]$



Forward Diffusion Process

“Adding noise to data...”

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



Forward Diffusion Process: Explicit Representation

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

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...

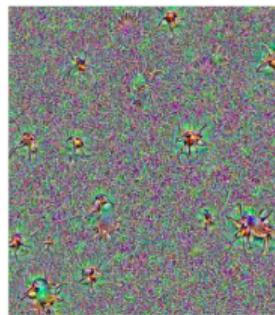
$$\begin{aligned} q(\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T)}) &= q\left(\mathbf{x}^{(T)} \mid \mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T-1)}\right) q\left(\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T-1)}\right) \\ &= q\left(\mathbf{x}^{(T)} \mid \mathbf{x}^{(T-1)}\right) q\left(\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T-1)}\right)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

Distributional Representation

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

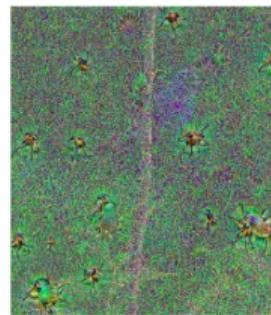
Reverse Diffusion Process

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



○ ○ ○

$$x^{(t)}$$



$$x^{(t-1)}$$

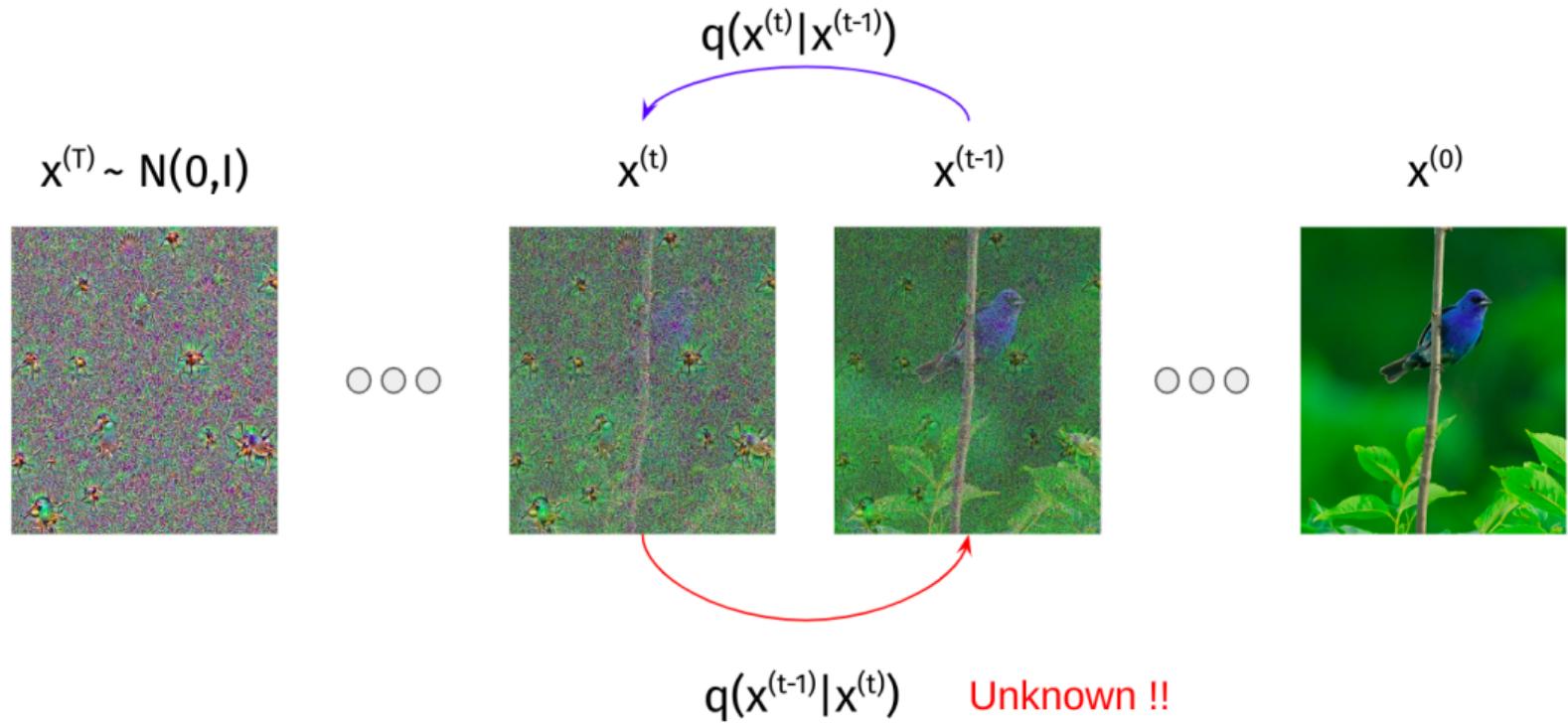


○ ○ ○

$$x^{(0)}$$



Reverse Diffusion 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

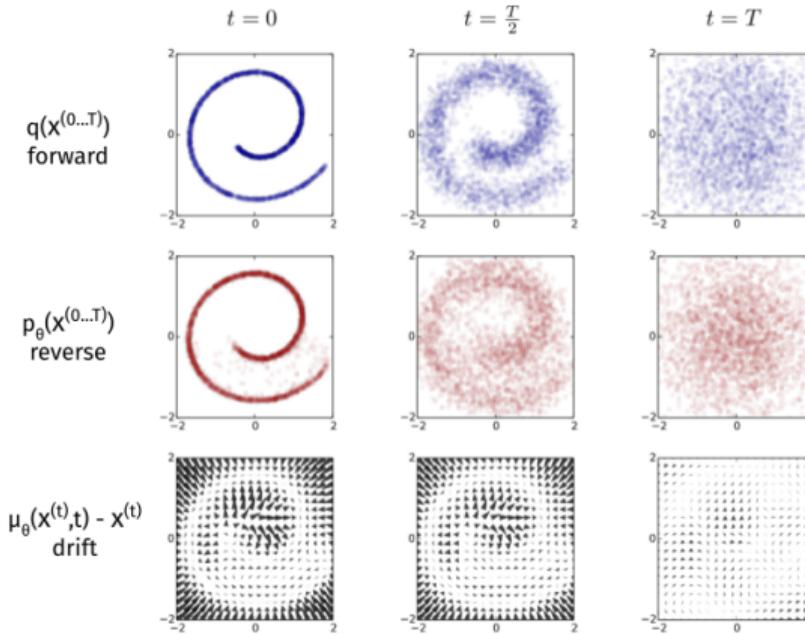
If $|\beta_i - \beta_{i+1}| \approx 0$, i.e. diffusion slow enough, then

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

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

Aim

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

Aim

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

Aim

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

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

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

Easy??

Training of μ_θ and Σ_θ

Aim

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

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

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

Easy??

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



Training of μ_θ and Σ_θ

Aim: 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

Aim: 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

Aim: 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,

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

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

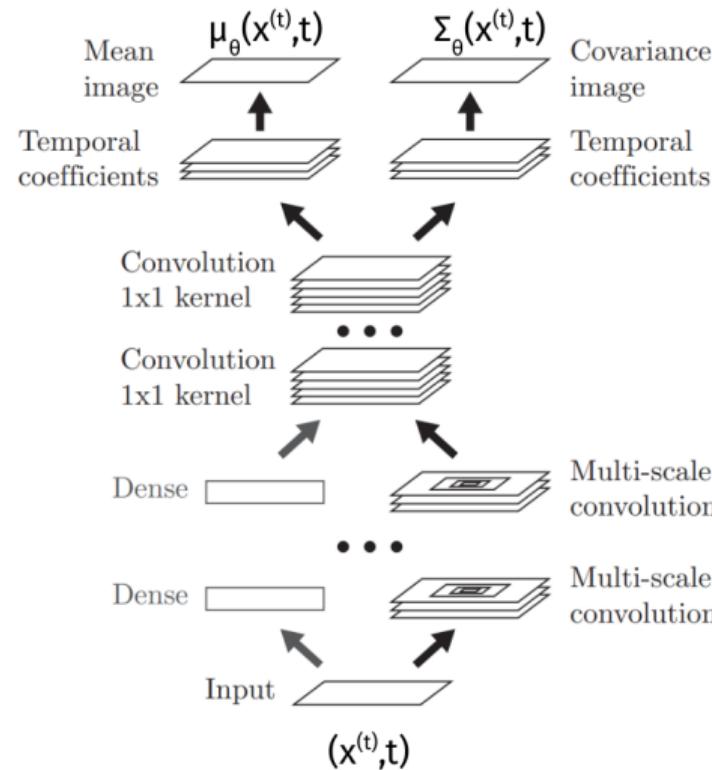
$$\mathcal{L}_0 = \mathbb{E}_{q(\mathbf{x}^{(0\dots T)})} \left[p_\theta(\mathbf{x}^{(0)} | \mathbf{x}^{(1)}) \right]$$

Note.

\mathcal{L}_T is constant.

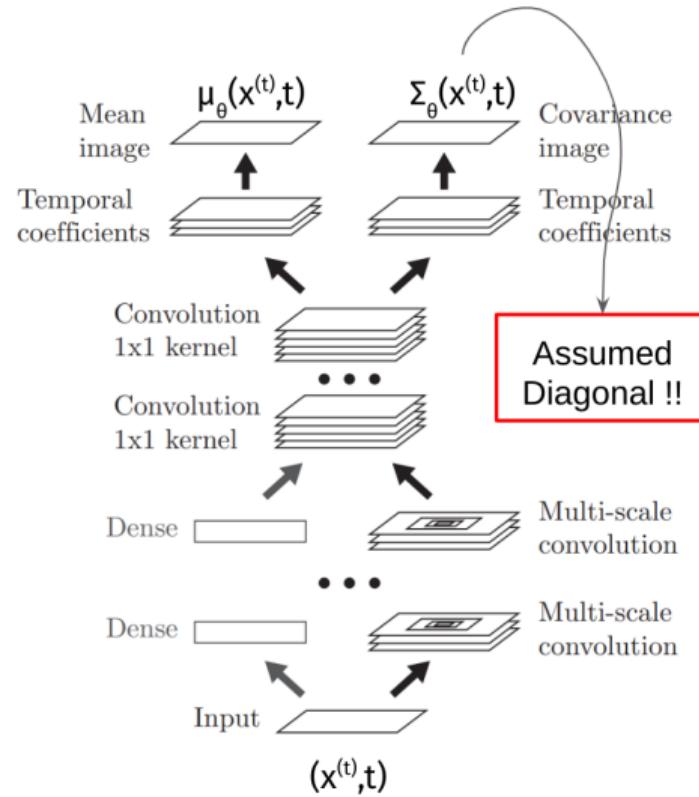
$\mathcal{L}_0, \mathcal{L}_t$ explicit.

Neural Network that estimate μ_θ and Σ_θ



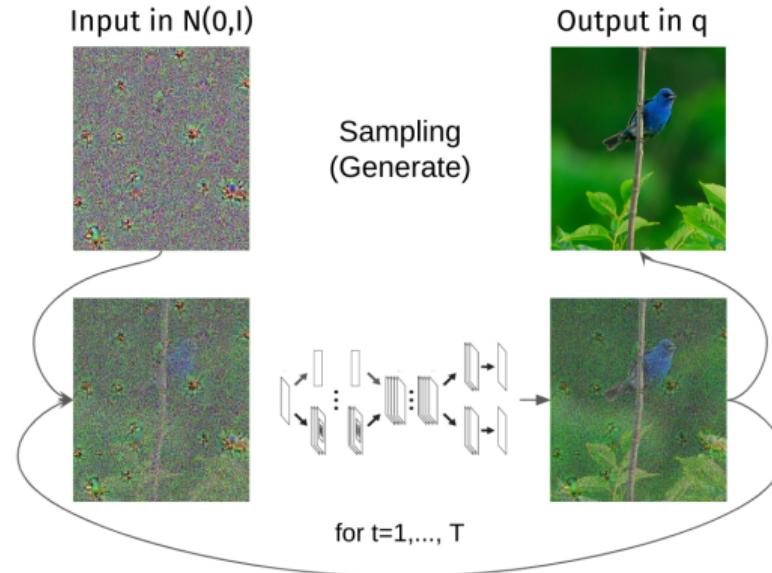
Proposed Neural Network for CIFAR10 image generation. T=1000

Neural Network that estimate μ_θ and Σ_θ

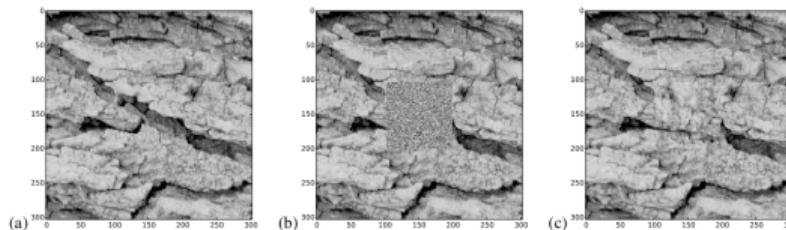


Proposed Neural Network for CIFAR10 image generation. T=1000

Sampling or Generative Stage



Experiments



MNIST

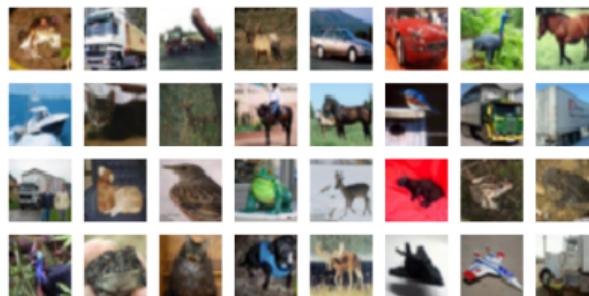


5

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

Experiments

CIFAR10 (original)



CIFAR10 (generated)



6

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

Timeline

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

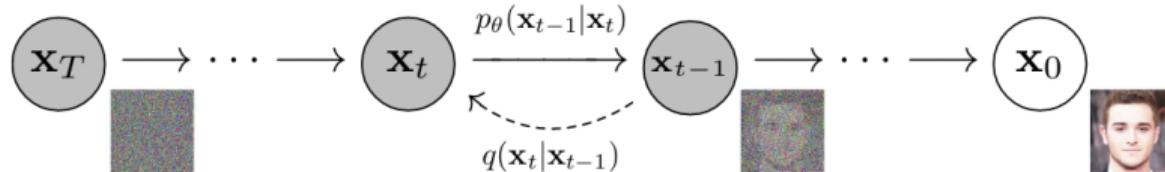


2020) *Denoising Diffusion Probabilistic Models*. Ho et al. NeurIPS.

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".

Denoising Diffusion Probabilistic Model

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

Simplification II: Estimating the commited error

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

Denoising Diffusion Probabilistic Model

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

Simplification II: Estimating the commited 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 aim at estimating $\mu_\theta(\mathbf{x}^{(t)}, t)$ and $\Sigma_\theta(\mathbf{x}^{(t)}, t)$.

Simplification II: Estimating the commited 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:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left( \mathbf{x}_t - \frac{1 - \bar{\alpha}_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```

Experiments: Sample Quality

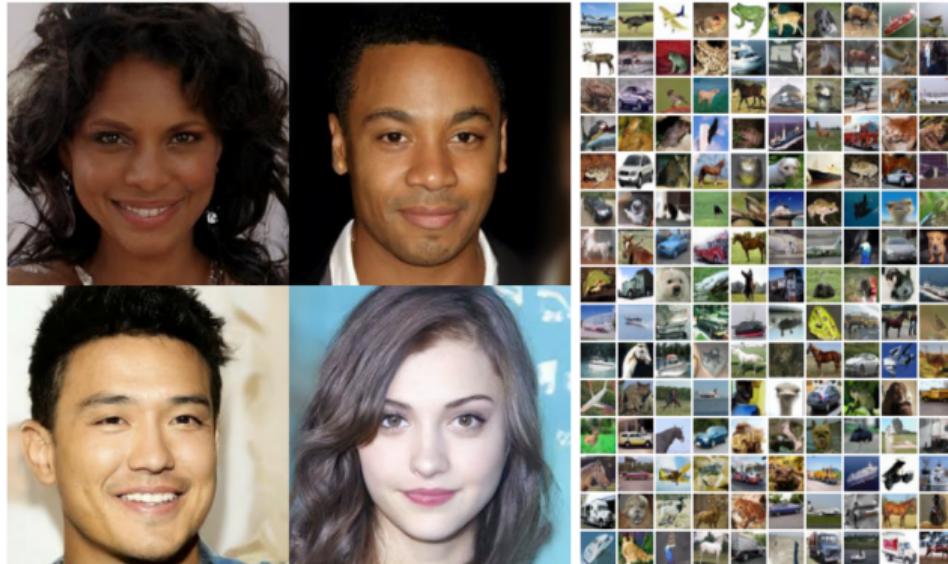


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

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. High FID \Rightarrow high quality
2. Training improved

Broader Impacts



"We also found discrepancies across gender and race for people categorized into the 'crime' and 'non-human' categories..."⁸

⁸Radford 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 proceed 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 $(1 - \beta_{t+1})\alpha_t + \beta_{t+1}$.

