

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

DALL-E2

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*

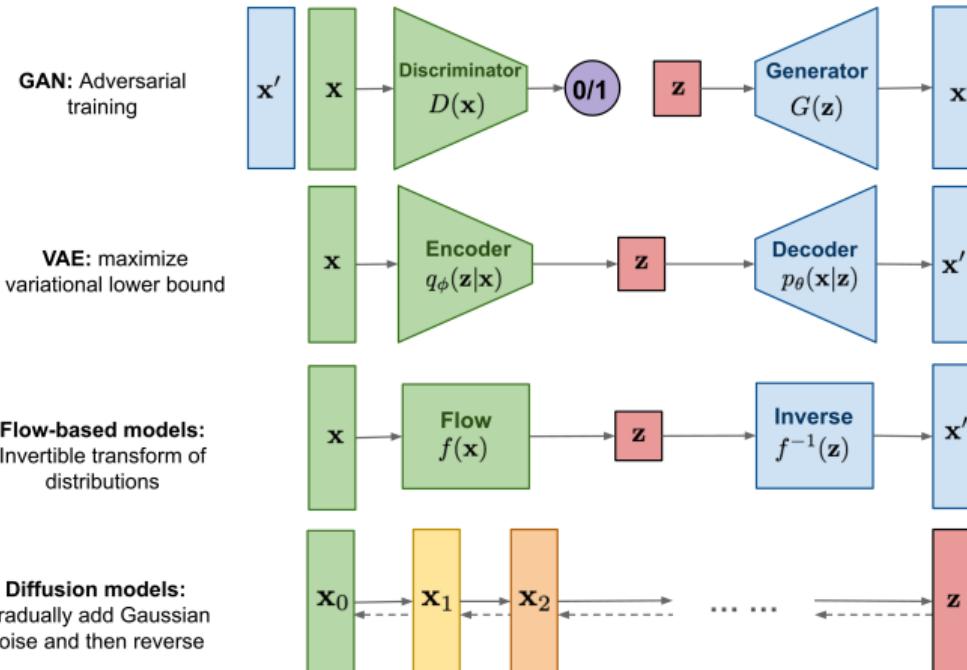
*High dimension data
point from simple
noise distribution*

*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



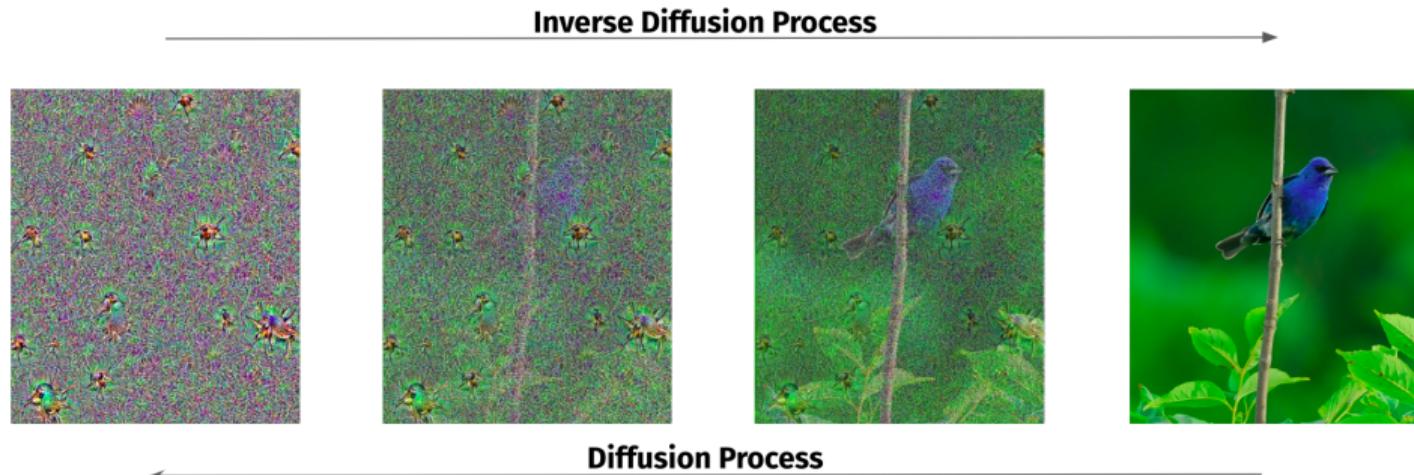
Overview of different generative models¹

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

Diffusion Models



Diffusion models are generative models that learn a inverse diffusion process

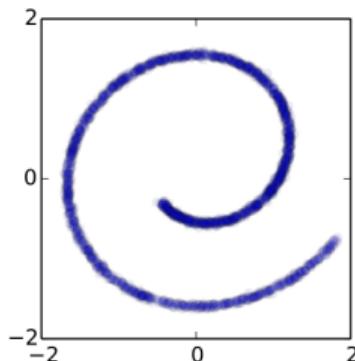


Timeline

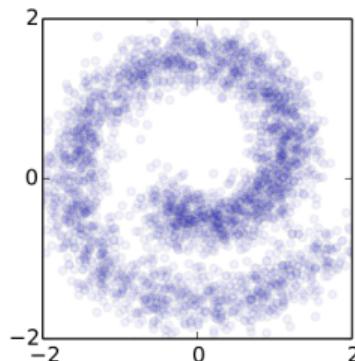
- 2015) *Deep Unsupervised Learning with Non-equilibrium Thermodynamics*. Sohl-Dickstein et al. ICML.
- 2020) *Denoising Diffusion Probabilistic Models*. Ho et al. NeurIPS.
- 2021) *Score-Based Generative Modeling Through SDE*. Song et al. ICLR.

Deep Unsupervised Learning using Non-Equilibrium Thermodynamics

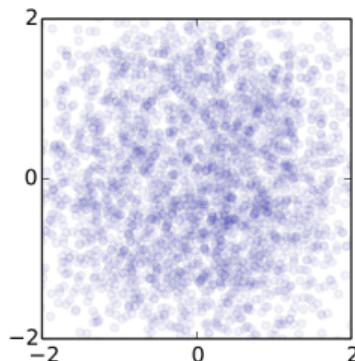
$t = 0$



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



$t = T$



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

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

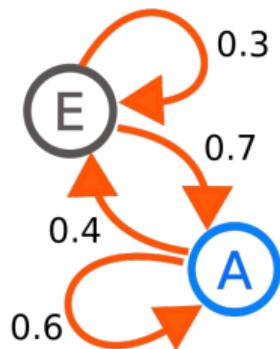
Reminder: Markov Chains with Discrete Time

Informal Definition

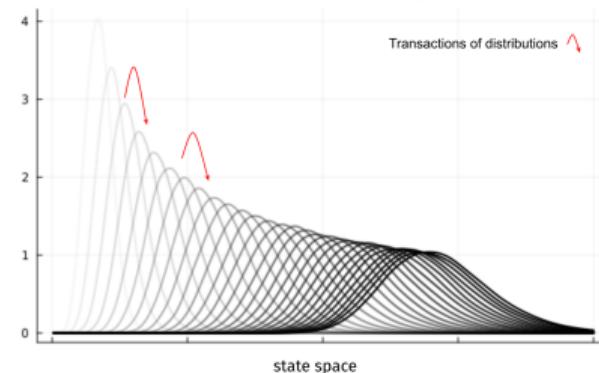
A sequence of random variables $x^{(0)}, x^{(1)}, \dots, x^{(t)}, \dots$, such that

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

(Markov Property)

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

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

(Homogeneous Property)

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

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

(Markov Property)

P is a stochastic matrix!

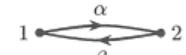
$$\forall i, \quad \sum_{j \in \mathbb{N}} p_{ij} = 1$$

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

=

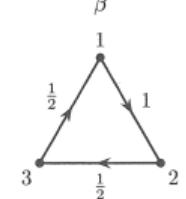
$$\mathbb{P}(\mathbf{x}^{(t+1)} = s_j | \mathbf{x}^{(t)} = s_{i_t})$$

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



(Homogeneous Property)

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



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

Definition (DTMC Continuous 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 $\kappa_t : S \times S \rightarrow [0, 1]$ 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. absence of memory.

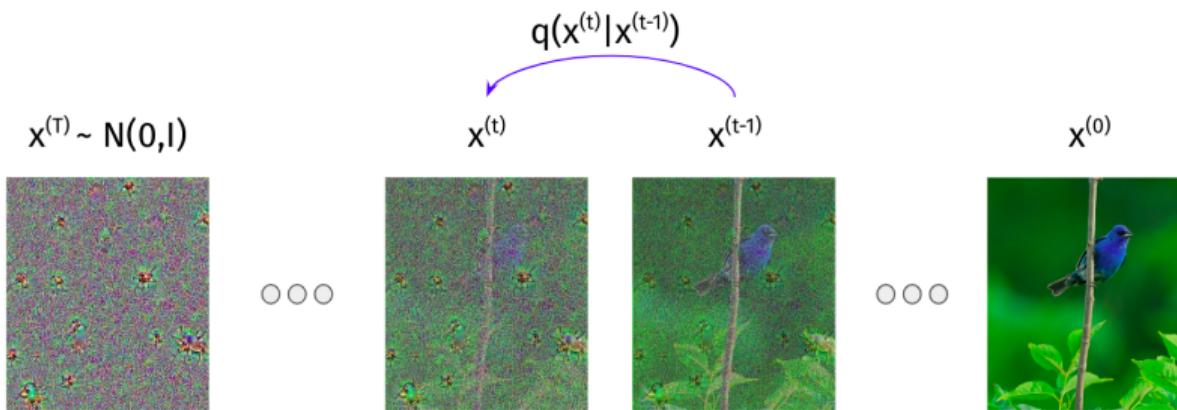
Notation

- 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: $q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1 - \beta_t} \mathbf{x}^{(t-1)}; \beta_t I\right)$
- Diffusion rates: $0 < \beta_1 < \dots < \beta_T = 1$

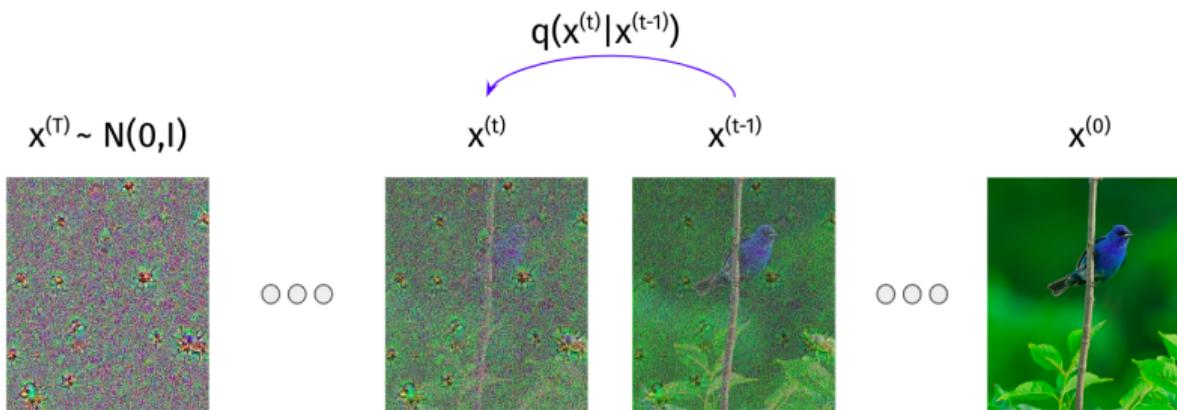


Forward Diffusion Process

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: $q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1 - \beta_t} \mathbf{x}^{(t-1)}; \beta_t I\right)$
- Diffusion rates: $0 < \beta_1 < \dots < \beta_T = 1$

Not Analytic!!



Forward Diffusion Process: Explicit Representation

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

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

Forward Diffusion Process: Distribution Representation

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

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

Forward Diffusion Process: Distribution Representation

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

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

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

Forward Diffusion Process: Distribution Representation

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

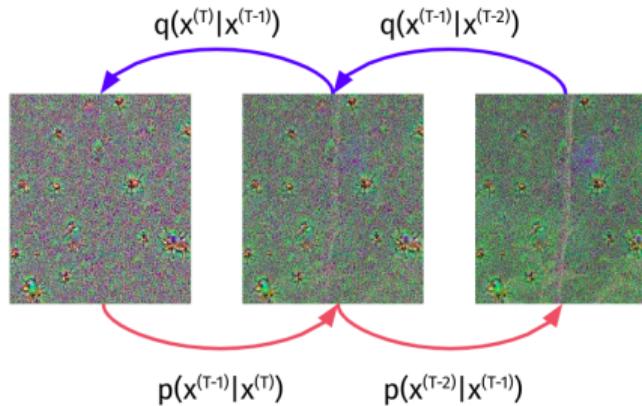
$$\begin{aligned} q(\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T)}) &= q\left(\mathbf{x}^{(T)} \mid \mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T-1)}\right) q\left(\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T-1)}\right) \\ &= q\left(\mathbf{x}^{(T)} \mid \mathbf{x}^{(T-1)}\right) q\left(\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T-1)}\right)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

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

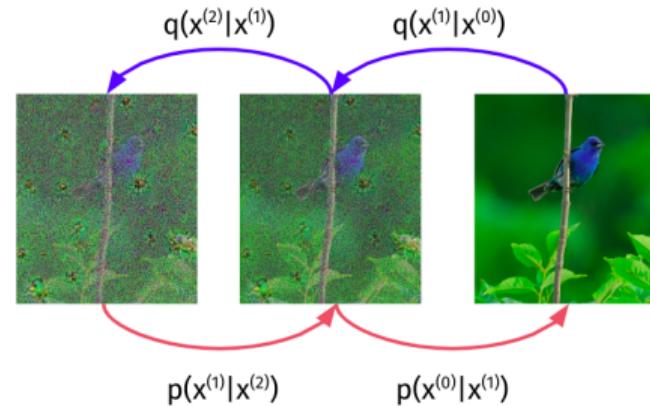
Distributional Representation

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

Reverse Diffusion Process



○ ○ ○

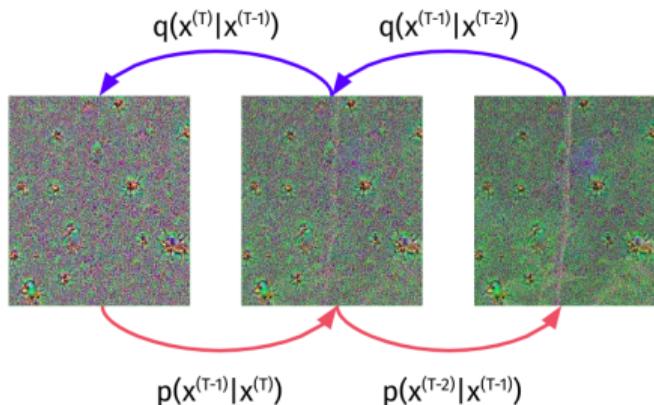


Reverse Diffusion Process

Fixed Forward Process

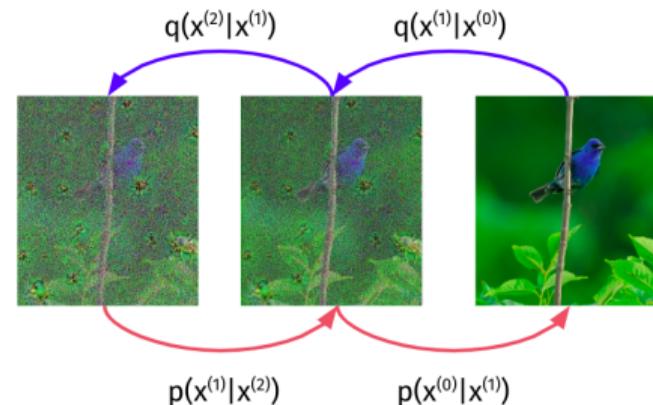
Initial Distribution

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



Gaussian Transaction Kernel

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

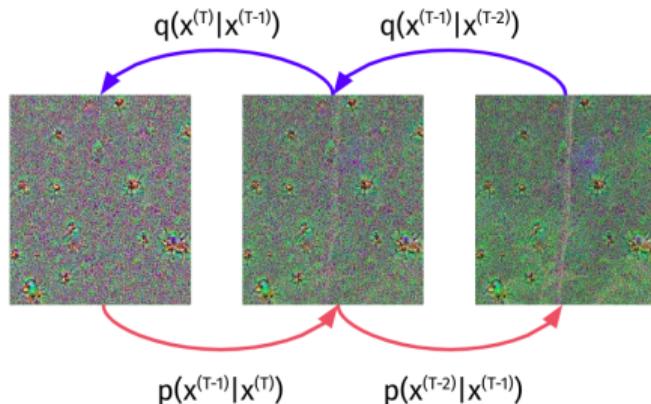


Reverse Diffusion Process

Fixed Forward Process

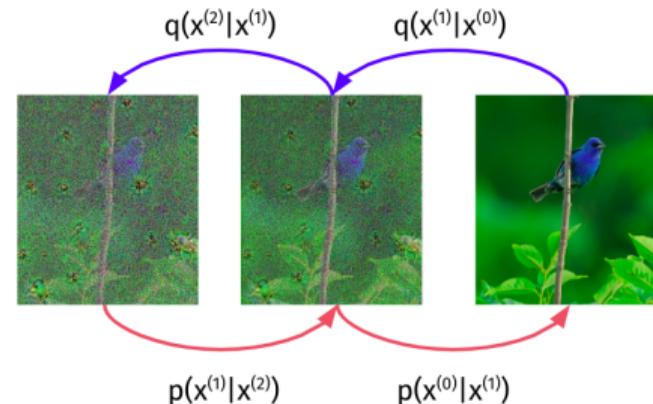
Initial Distribution

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



Gaussian Transaction Kernel

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



Learned Reverse Process

Initial Distribution

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

Approximation of

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

Gaussian Kernel with parameters

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

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

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

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 the reverse diffusion process can be approximated by an other diffusion process with a gaussian transition kernel

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

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 the reverse diffusion process can be approximated by an other diffusion process with a gaussian transition kernel

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

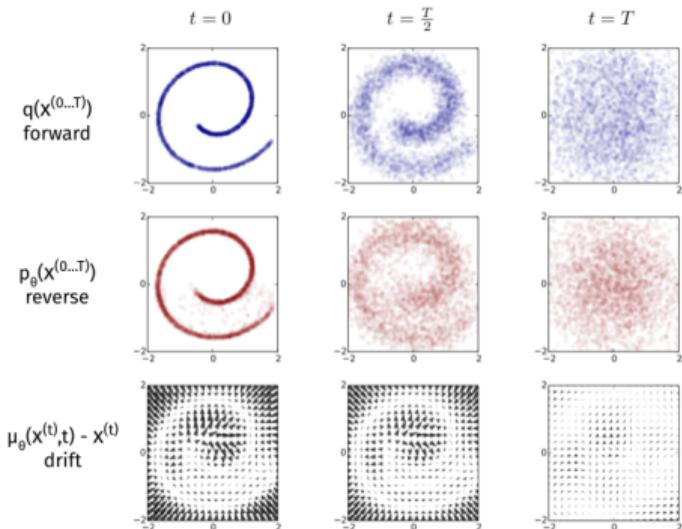
Mean $\boldsymbol{\mu}_\theta$ and covariance $\boldsymbol{\Sigma}_\theta$ are the output of a neural model !!

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

Visualization of Diffusion Process: 2D dimensional case⁴

- **(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

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



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 manipulations :)

Reformulated Loss Function

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

where,

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

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

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

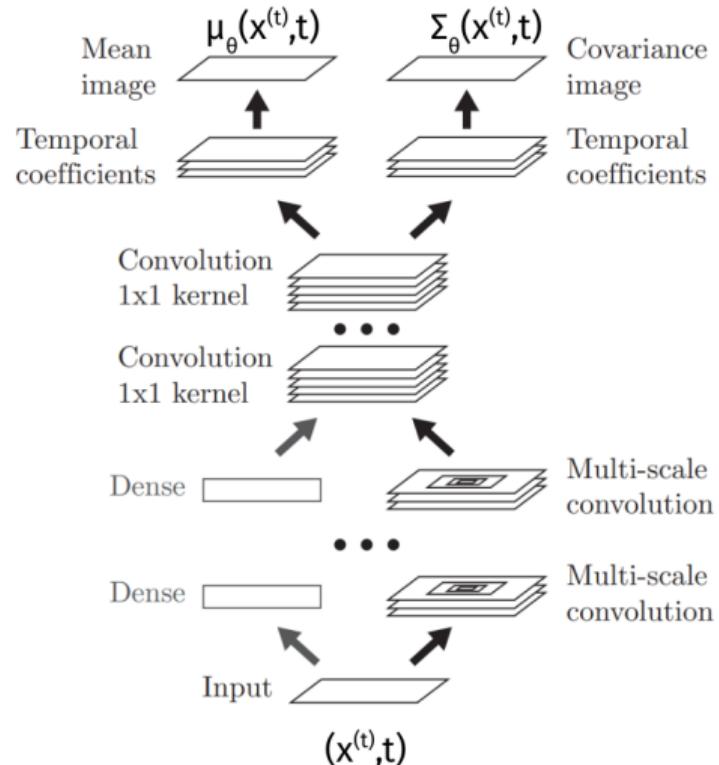
Observation

Note that in \mathcal{L}_T does not depend on θ . \mathcal{L}_t allows an explicit formulation being the forward transition gaussian and the reverse transition gaussian with parameterized mean and covariance. 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 $\mu_\theta, \Sigma_\theta$.
- The diffusions rate β_t are trained as the other parameters and fixed.

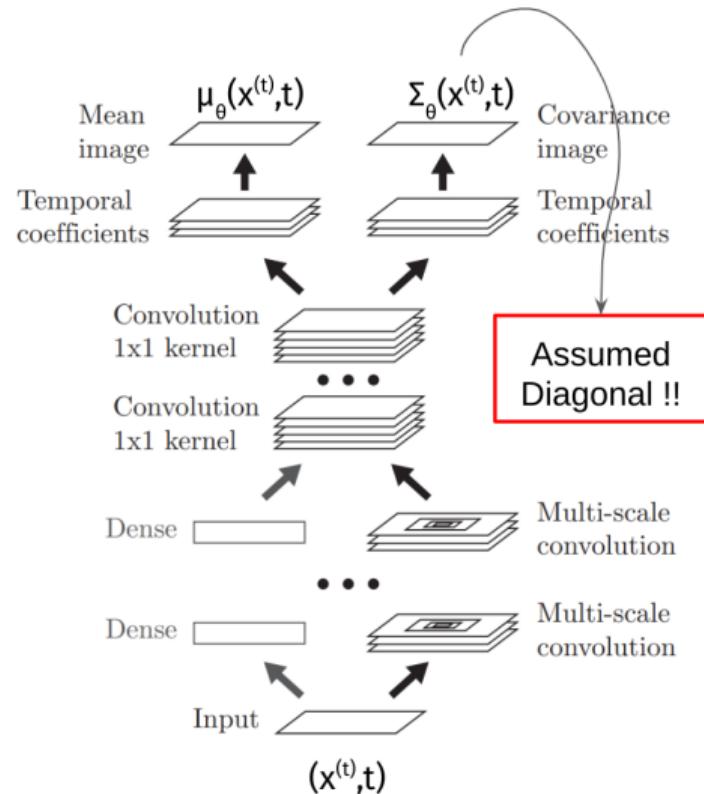


Proposed Neural Network for CIFAR10 image generation.
T=1000

Neural Network that estimates μ_θ and Σ_θ

Description of the model

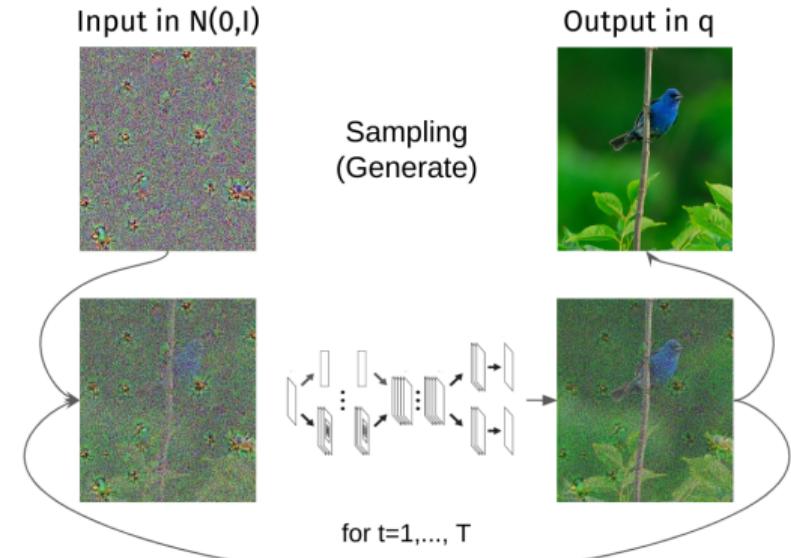
- Simple model, only convolutions, no residual layers.
- For each input $x \in \mathbb{R}^n$ and t , the model returns a vector $y = f_\theta(x) \in \mathbb{R}^{2n}$ that stores the mean and a diagonal covariance $\mu_\theta, \Sigma_\theta$.
- The diffusions rate β_t are trained as the other parameters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



Proposed Neural Network for CIFAR10 image generation.
T=1000

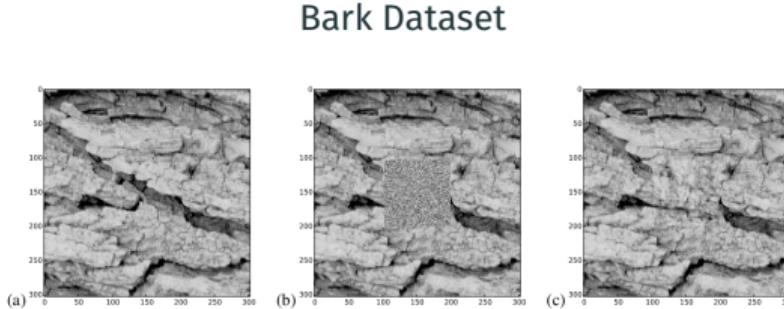
Sampling or Generative Stage

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.
 $T=1000$

Experiments⁵



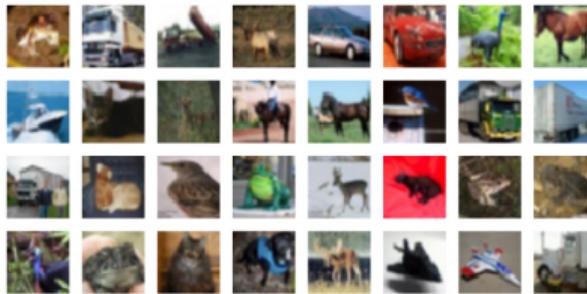
MNIST

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

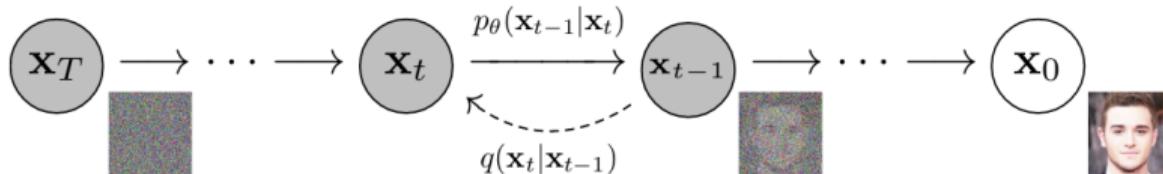
2015) ...*Non-equilibrium Thermodynamics*. Sohl-Dickstein et al. ICML. ✓

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

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

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

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

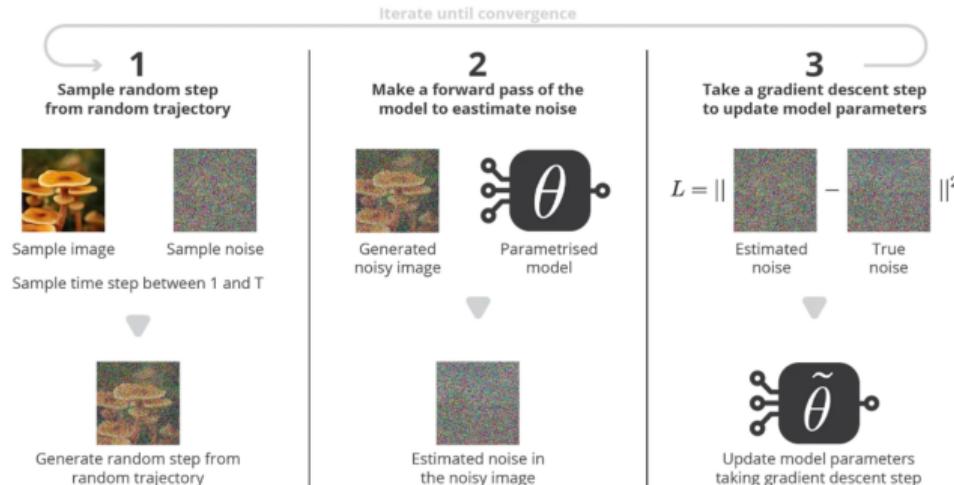
Training and Sampling Procedure

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”

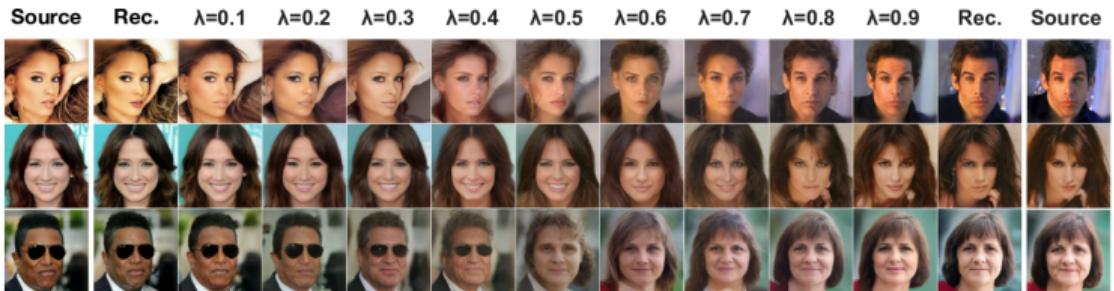
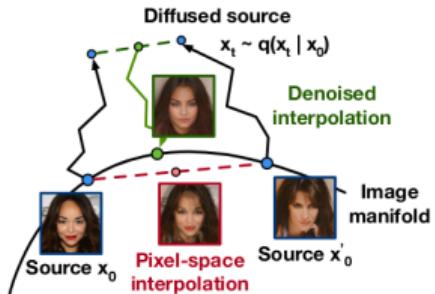
Table 1: CIFAR10 results. NLL measured in bits/dim.

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
PixelCNN [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

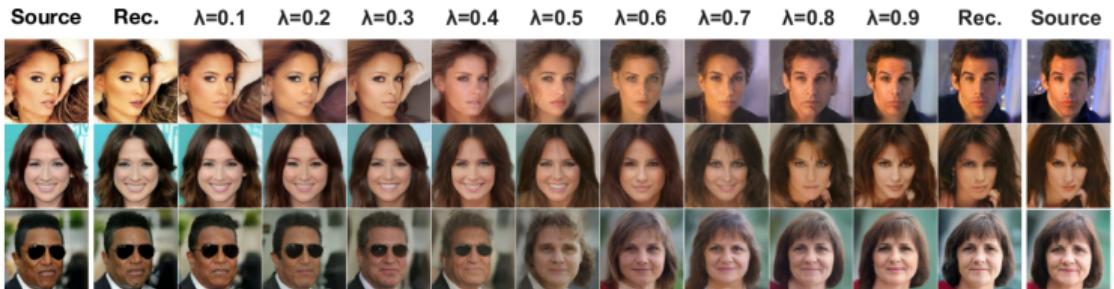
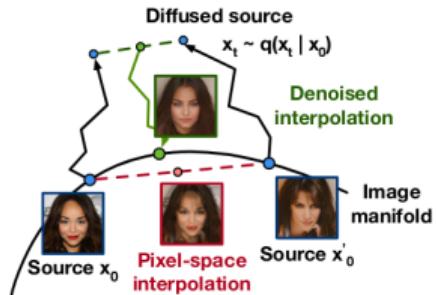
⁹ Ho, Jain, and Abbeel, “Denoising Diffusion Probabilistic Models”.

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



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

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

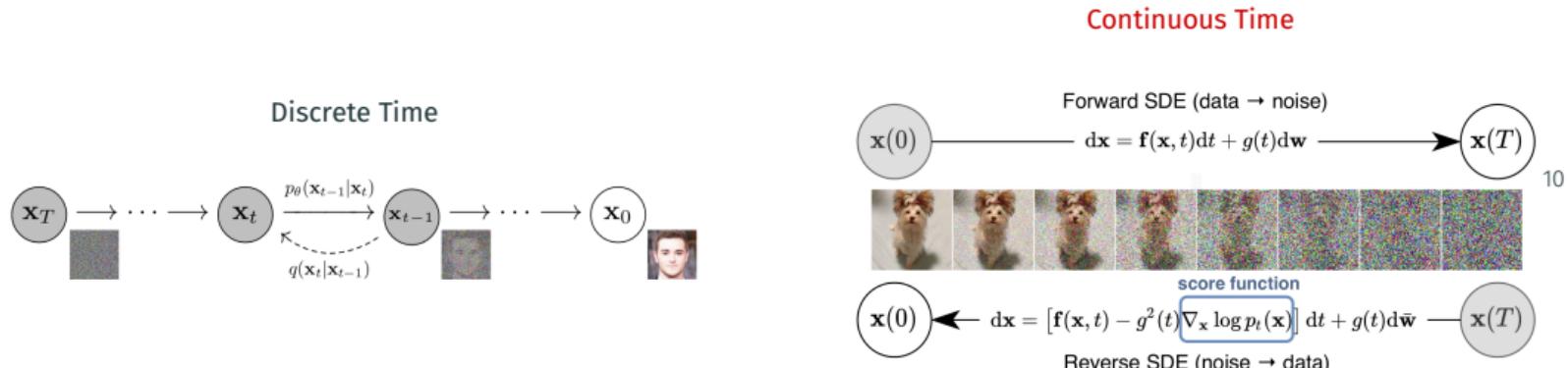
Timeline

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)



Stochastic Process

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

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

Continuous Diffusion Process described by SDE

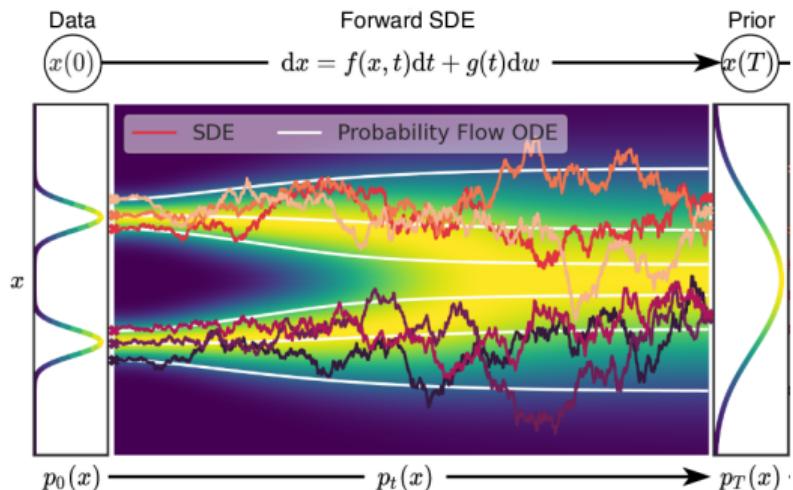
Continuous Diffusion Process

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

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 $\nabla_x p_t(x)$

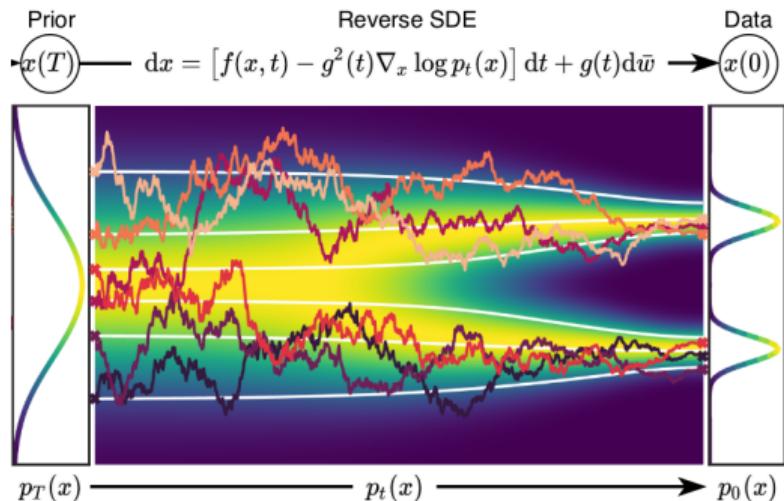
Theorem

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



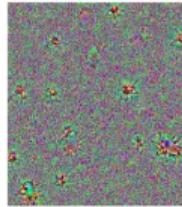
DALL-E2



Introduction

A Ramesh et al. (Open AI). “Hierarchical Text-Conditional Image Generation with CLIP Latents.”. February 2022.

From an analytic distribution to image distribution



Unconditional
Generative Model



<sos>A Blue-and-black tanager hanging on a branch looking forward to the left with a green background .<eos>

Text-Conditioned
Generative Model



From a distribution in the a latent space
to image distribution



vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck



an espresso machine that makes coffee from human souls, artstation



panda mad scientist mixing sparkling chemicals, artstation

Introduction

DALL-E2: a new AI system that can create realistic images and art from a description in natural language

An astronaut Teddy bears A bowl of soup

mixing sparkling chemicals as mad scientists shopping for groceries working on new AI research

as a 1990s Saturday morning cartoon as digital art in a steampunk style



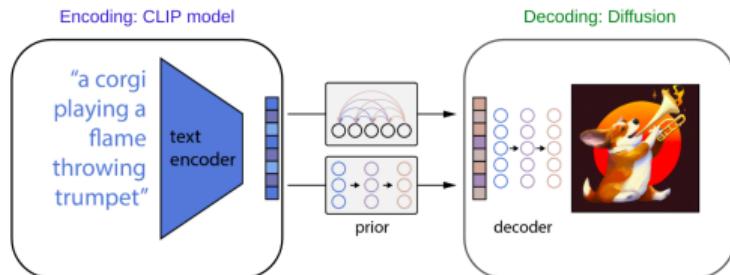
11

¹¹Credits: DALL-E2 website

Overview: CLIP + Diffusion

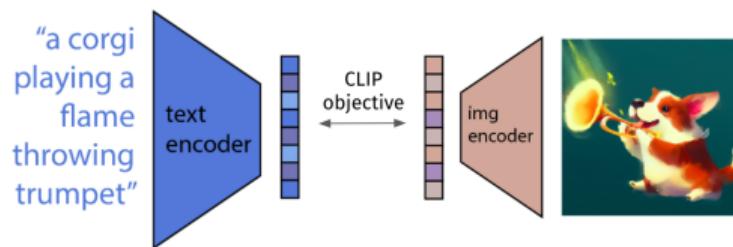
12

Text-to-Image Generation with CLIP



Diffusion models allows generating images with high sample fidelity and photorealism

CLIP model

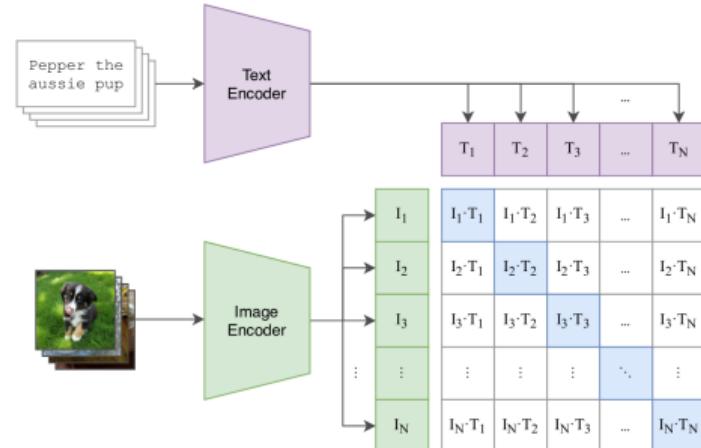


Embedding of images and text in the same latent space

¹²Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents".

CLIP: Latent Embedding model

1. The **Text Encoder** takes as input a text y in and embeds it in $z \in \mathbb{R}^n$.
2. The **Image Encoder** takes as input an image x and embeds it in the latent space \mathbb{R}^n .
3. **Property:** The two embedded objects can be directly compared through cosine similarity.

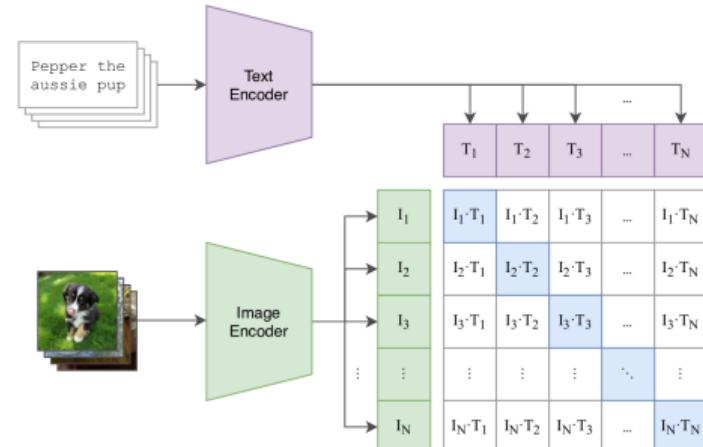


CLIP allows embedding of images and text in the same latent space.

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

CLIP: Latent Embedding model

4. **Aim:** Initially the model was designed for performing a Zero-shot classification (classification of unseen data).
5. The **Text Encoder** is a Transformer for NLP (63M of parameters, 12 layers 512-wide model with 8 attention head. Token lenght 76 (lol)).
6. The **Image Encoder** include ResNet-50 and Vision-Transformers



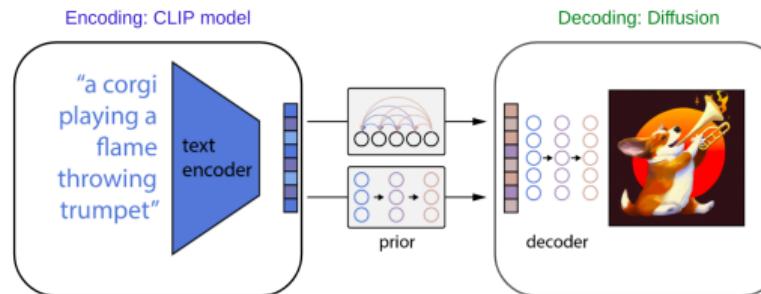
CLIP allows embedding of images and text in the same latent space.

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

DALL-E2 with higher detail

Dall-E2 is composed by the following modules. Let (x, y) be an (image, caption) pair,

1. CLIP such that $(x, y) \mapsto (z_i, z_t)$.
2. Prior which estimates $q(z_i | y)$
3. Decoder which estimate $q(x | z_i, y)$

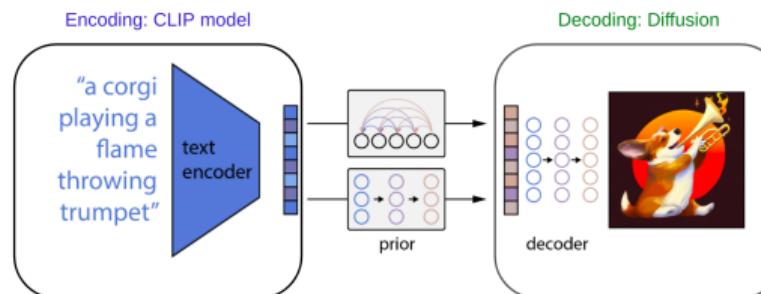


DALL-E2 with higher detail

Dall-E2 is composed by the following modules. Let (x, y) be an (image, caption) pair,

1. CLIP such that $(x, y) \mapsto (z_i, z_t)$.
2. Prior which estimates $q(z_i | y)$
3. Decoder which estimate $q(x | z_i, y)$

Provide an image embedding conditioned to a text.
Provide an Image conditioned to a text and Image Emb.

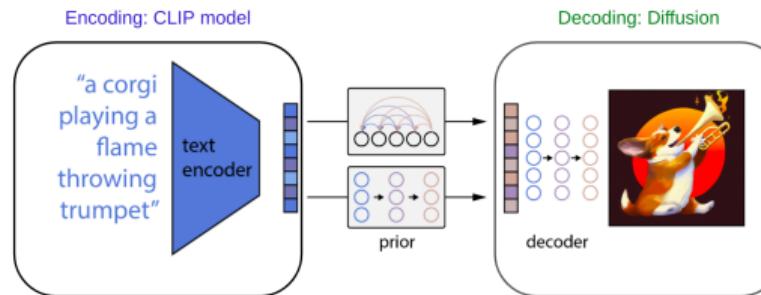


DALL-E2 with higher detail

Dall-E2 is composed by the following modules. Let (x, y) be an (image, caption) pair,

1. CLIP such that $(x, y) \mapsto (z_i, z_t)$.
2. Prior which estimates $q(z_i | y)$
3. Decoder which estimate $q(x | z_i, y)$

Provide an image embedding conditioned to a text.
Provide an Image conditioned to a text and Image Emb.



Pipeline of Generation

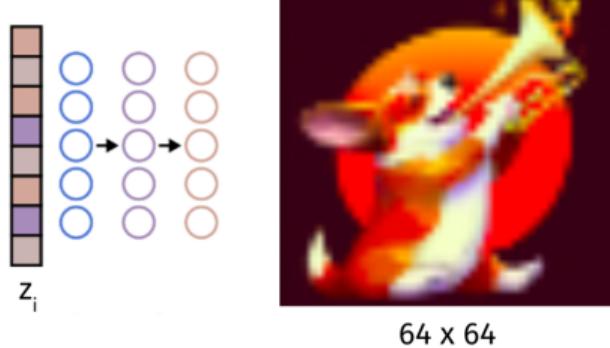
The aim of the DALL-E2 is to recover the distribution of the image $p(x|y)$ conditioned to the text y .

The **Text Encoder** returns the embedding z_t which is processed by the **prior** to find a good estimation of the embedding of the (desired) output image z_i . The **decoder** inverts the embedding z_i and return the image x .

Decoder: Guided Diffusion models

Step 1: Generate a low-resolution image from z_i

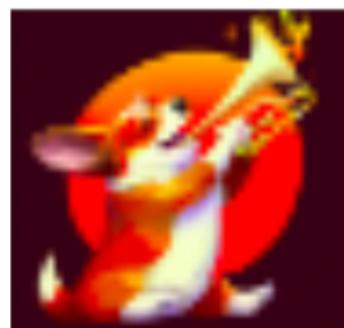
A guided diffusion model is used to get a low resolution image from the image embedding.



Decoder: Guided Diffusion models

Step 1: Generate a low-resolution image from z_i

A guided diffusion model is used to get a low resolution image from the image embedding.



64 x 64

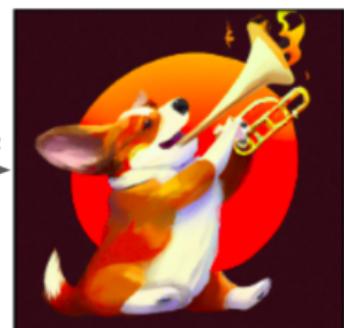
Up_1



256 x 256

Step 2. Upsampling Procedure

Two diffusion models are used to increase the resolution of the image from low to super-high



1024 x 1024

The guided diffusion model¹⁴

Guidance through classification

A neural classifier that estimate $p_\Phi(y|x)$ can be used to **guide** a diffusion model to produce an image of a certain class y .

$$\tilde{\mu}_\theta(x^{(t)}|y) = \mu_\theta(x^{(t)}|y) + \nabla_{x^{(t)}} \log(p_\Phi(y|x^{(t)}))$$

Guidance through CLIP similarity

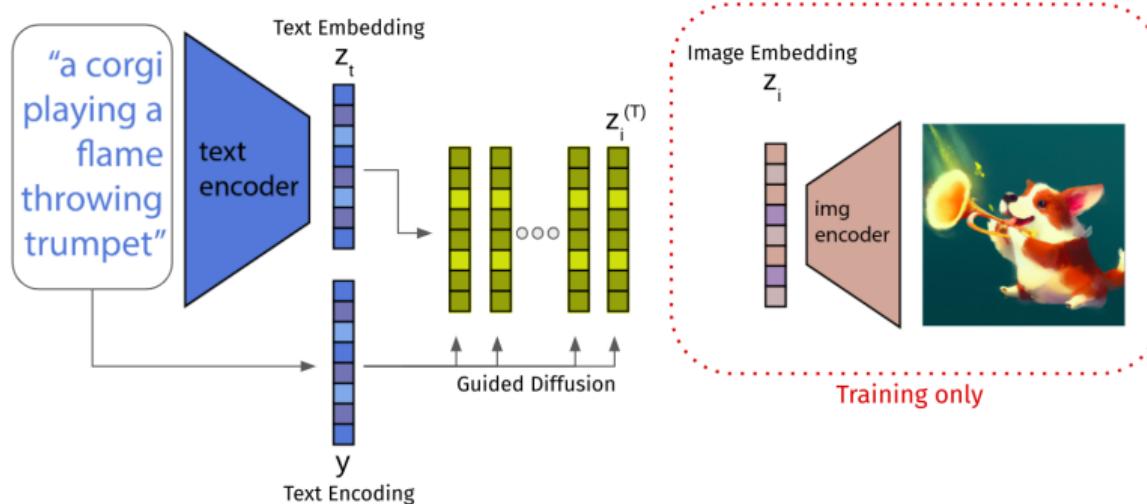
To apply the same idea to diffusion models, we can replace the classifier with a CLIP model in classifier guidance. Given a caption y , the mean of the diffusion process is perturbed by using the dot product of the embeddings

$$\tilde{\mu}_\theta(x^{(t)}|y) = \mu_\theta(x^{(t)}|y) + \nabla_{x^{(t)}} z_t(y) \cdot z_i(x^{(t)})$$

¹⁴Nichol et al., “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models”.

Prior

Diffusion Prior: “The image embedding \mathbf{z}_i is directly modelled using a Gaussian diffusion model conditioned on the caption \mathbf{y} ”.



Training Loss Function $L_{prior} = \mathbb{E}_{t \sim [1, T], \mathbf{z}_i^{(t)} \sim q_t} \left[\|f_\theta(\mathbf{z}_i^{(t)}, \mathbf{y}, t) - \mathbf{z}_i\|^2 \right]$ where $\mathbf{z}_i^{(0)} = \mathbf{z}_t$

Experiments: Image Generation



a espresso machine that makes coffee from human souls, artstation



panda mad scientist mixing sparkling chemicals, artstation



a corgi's head depicted as an explosion of a nebula



a dolphin in an astronaut suit on saturn, artstation



a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese



a teddy bear on a skateboard in times square

Generation examples in the original paper.

Experiments: Image Interpolation



For each row, the decoder is applied to a combination of $\mathbf{z}_i(\text{left})$ and $\mathbf{z}_i(\text{right})$. The intermediate variations naturally blend the content and style from both input images

Experiments: Hands Generation



Random samples from unCLIP for prompt “A close up of a handpalm with leaves growing from it.”

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.

¹⁵Ho, Jain, and Abbeel, “Denoising Diffusion Probabilistic Models”.

1. Diffusion models, as generative models, can be used for malicious purpose. 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.¹⁵”

¹⁵ Ho, Jain, and Abbeel, “Denoising Diffusion Probabilistic Models”.

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

