

# Diffusion Models and an application to DALL-E2

## 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

DALL-E2

Broader Impacts

# Introduction

---



# Generative Models

1D example:  
we illustrate the  
effet of G over  
the entire  
distribution

*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

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

High dimension  
example:  
we illustrate the  
effet of G over a  
single sample

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



*Generative model  
to be learned*

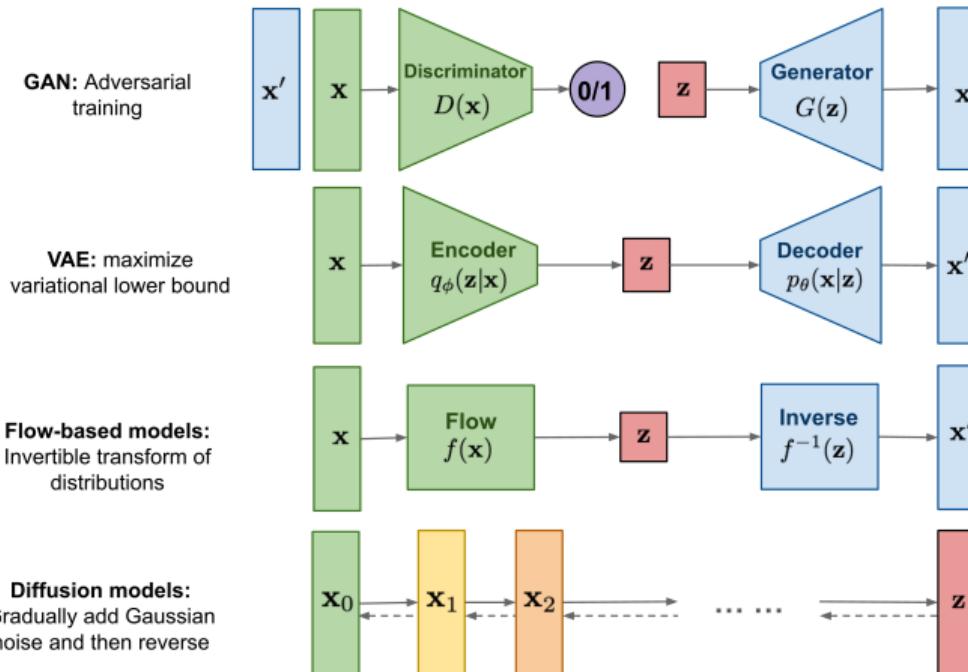
High dimension data  
point from simple  
noise distribution

High dimension data  
point from complex  
image distribution

Schema of generative models<sup>1</sup>

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

# Generative Models



Overview of different generative models<sup>2</sup>

<sup>2</sup>Credits <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

## Diffusion Models

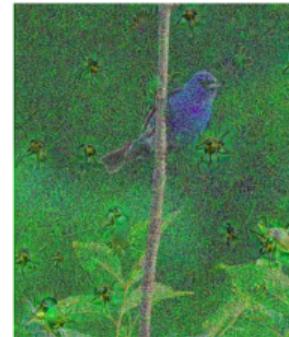
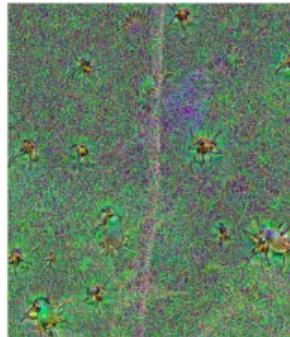
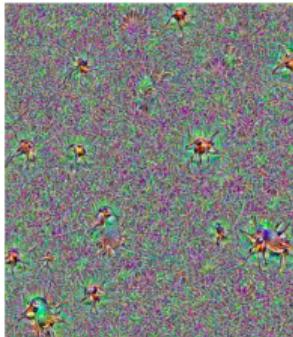
---



# Overview

*Diffusion models are generative models that aim at denoising data*

**Inverse Diffusion Process**

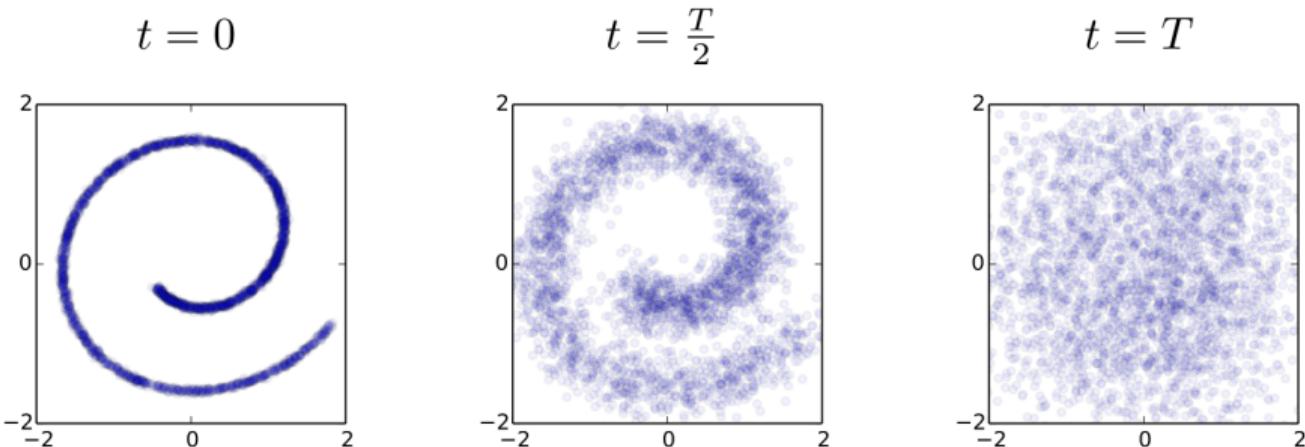


**Diffusion Process**

# 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.<sup>3</sup>

---

<sup>3</sup>Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics".

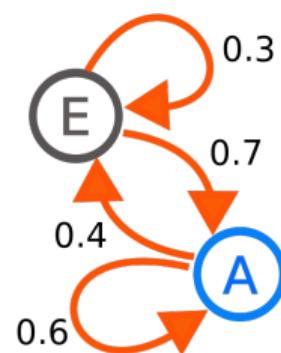
# Reminder: Markov Chains with Discrete Time

## Informal Definition

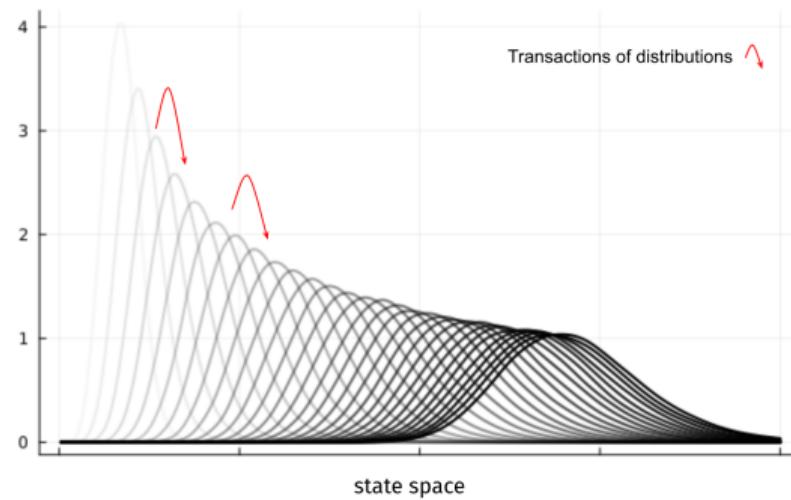
A sequence of random variables  $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(t)}, \dots$ , such that:

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

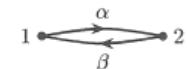
A sequence  $\{\mathbf{x}^{(t)}\}_{t \in \mathbb{N}} \subseteq S$ , a matrix  $P = (p_{ij})$ .

$P$  is a stochastic matrix!

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

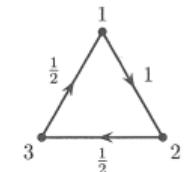
- Discrete state space:  $S = \{s_0, \dots, s_n, \dots\}$

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



- 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  $p(\mathbf{x} \mid \mathbf{y})$

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

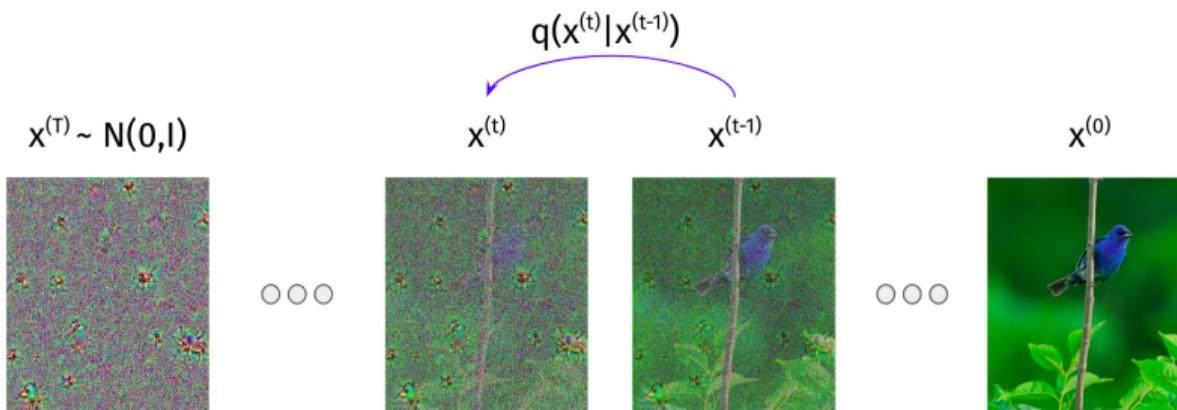
Marginal Distribution  $p(\mathbf{x})$

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

# Forward Diffusion Process

“Adding noise to data...”

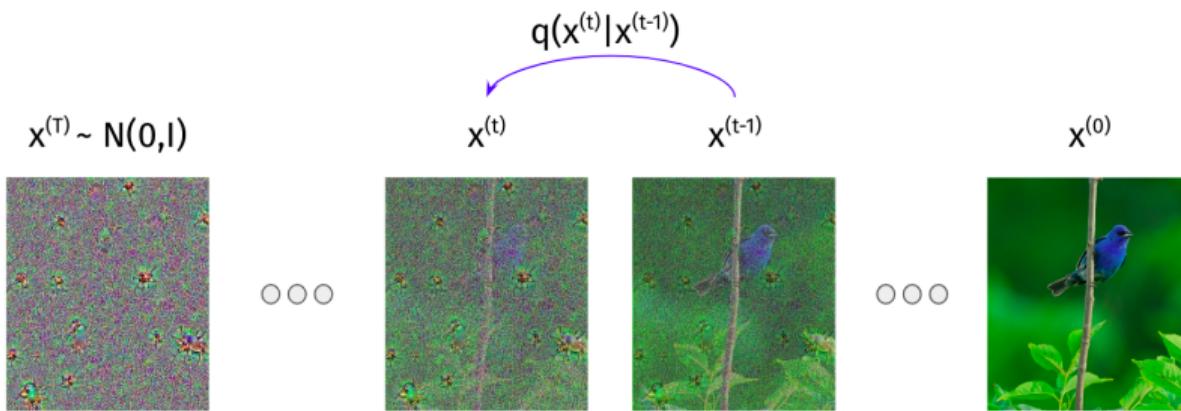
- Data Distribution:  $\mathbf{x}^{(0)} \sim q$
- Transaction Kernel:  $q(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)}) = \mathcal{N}(\mathbf{x}^{(t)}; \sqrt{1 - \beta_t} \mathbf{x}^{(t-1)}; \beta_t I)$
- Variance Scheduler:  $\beta_1, \dots, \beta_T \in (0, 1]$



# 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

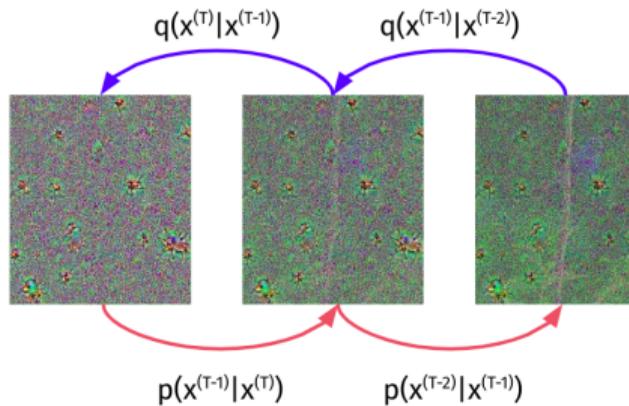
Markov property allows breaking up distributional Representation...

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

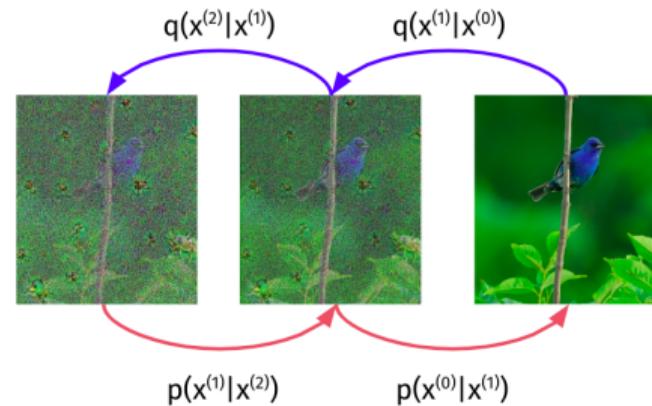
## Distributional Representation

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

# Reverse Diffusion Process



○ ○ ○



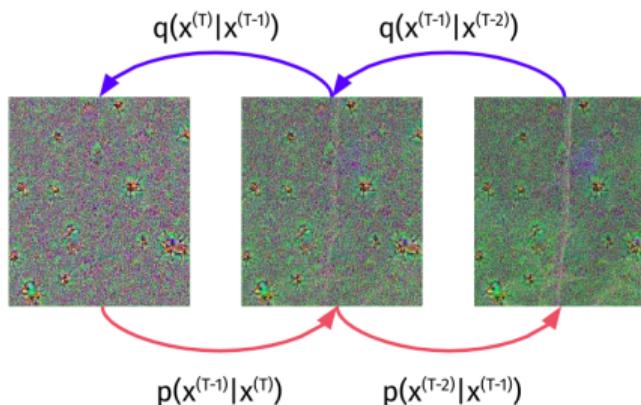
Learned Reverse Process

# Reverse Diffusion Process

## Fixed Forward Process

Initial Distribution

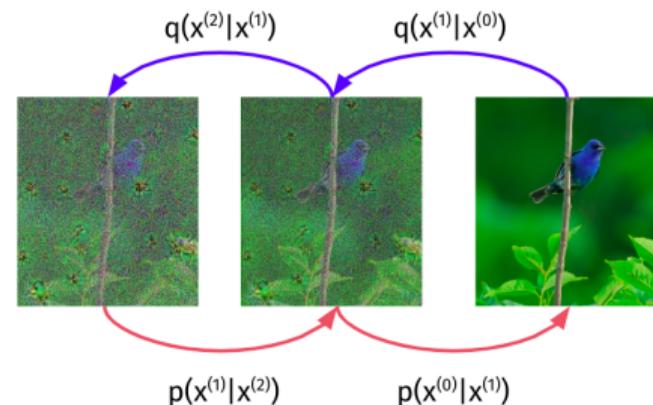
$$q(\mathbf{x}^{(0)})$$



Gaussian Transaction Kernel

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

○ ○ ○



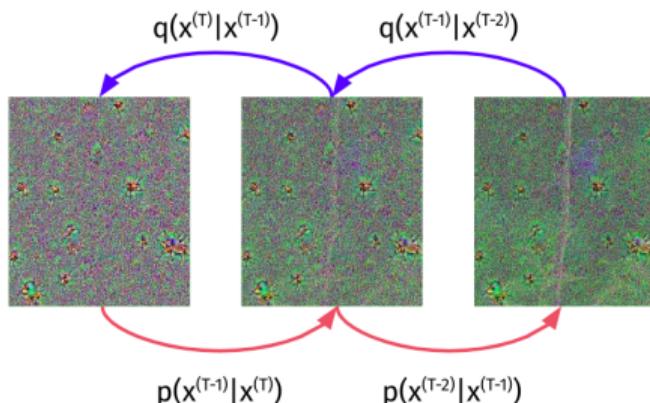
Learned Reverse Process

# Reverse Diffusion Process

## Fixed Forward Process

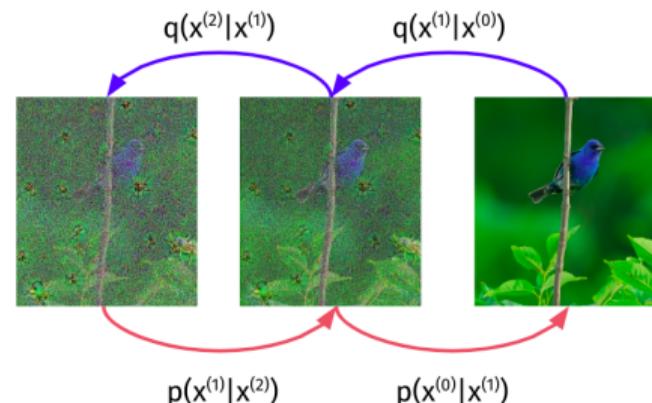
Initial Distribution

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



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

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

Learned Reverse Process

Approximation of

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

Gaussian Kernel with parameters

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

# 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

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

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

---

<sup>4</sup>Sohl-Dickstein et al., “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”.

# Reverse Diffusion Process

## Forward Diffusion Process

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

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

## Reverse Diffusion Process

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

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

Theorem. Reverse of Gaussian DP is  $\approx$  Gaussian DP<sup>4</sup>

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

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

<sup>4</sup>Sohl-Dickstein et al., “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”.

# Reverse Diffusion Process

## Forward Diffusion Process

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

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

## 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 DP<sup>4</sup>

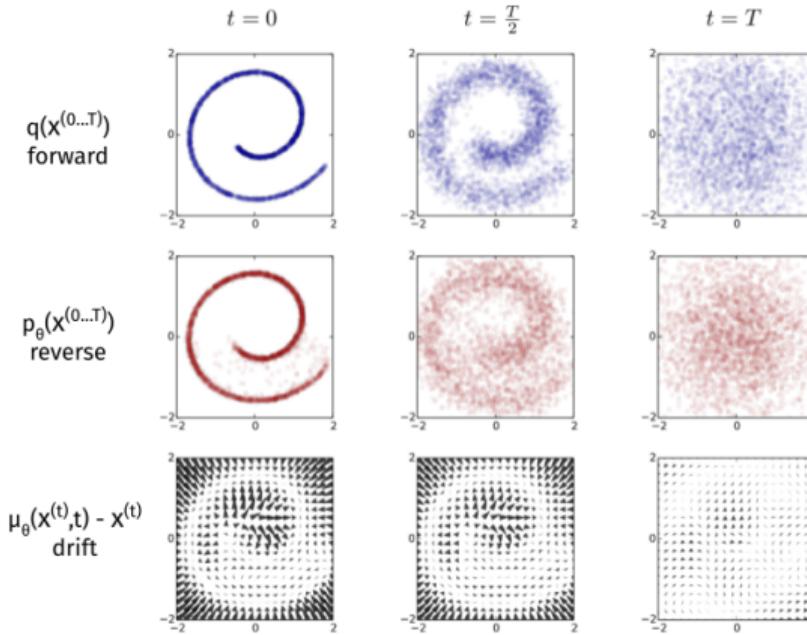
If  $|\beta_i - \beta_{i+1}| \approx 0$ , i.e. diffusion slow enough, then

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

Mean  $\mu_\theta$  and covariance  $\Sigma_\theta$  have to be learned!!

<sup>4</sup>Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics".

# Visualization of Diffusion Process: 2D dimensional case



5

<sup>5</sup>Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics".

# Training of $\mu_\theta$ and $\Sigma_\theta$

## Aim

Search for the best parameters  $\theta$

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

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 $\mu_\theta$ and $\Sigma_\theta$

## Aim

Search for the best parameters  $\theta$

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

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 $\mu_\theta$ and $\Sigma_\theta$

## Aim

Search for the best parameters  $\theta$

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

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 $\mu_\theta$ and $\Sigma_\theta$

## Aim

Search for the best parameters  $\theta$

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

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

## Estimated Reverse Process

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

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

## 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??

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



## Training of $\mu_\theta$ and $\Sigma_\theta$

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 $\mu_\theta$ and $\Sigma_\theta$

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 $\mu_\theta$ and $\Sigma_\theta$

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 $\mu_\theta$ and $\Sigma_\theta$

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 $\mu_\theta$ and $\Sigma_\theta$

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 $\mu_\theta$ and $\Sigma_\theta$

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 $\mu_\theta$ and $\Sigma_\theta$

...after some algebraic steps :)

### Reformulated Loss Function

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

where,

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

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

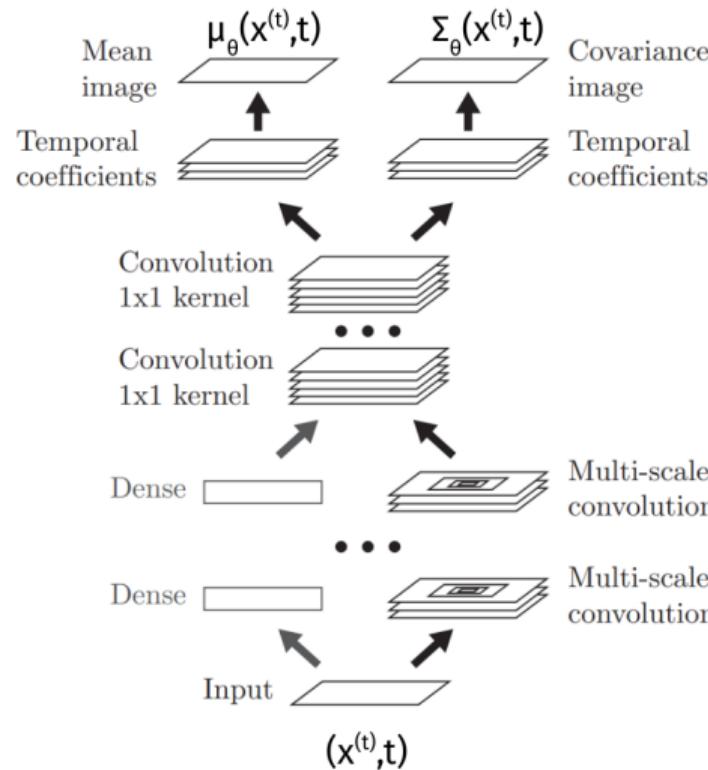
$$\mathcal{L}_0 = \mathbb{E}_{q(\mathbf{x}^{(0\dots T)})} \left[ p_\theta(\mathbf{x}^{(0)} | \mathbf{x}^{(1)}) \right]$$

Note.

$\mathcal{L}_T$  is constant.

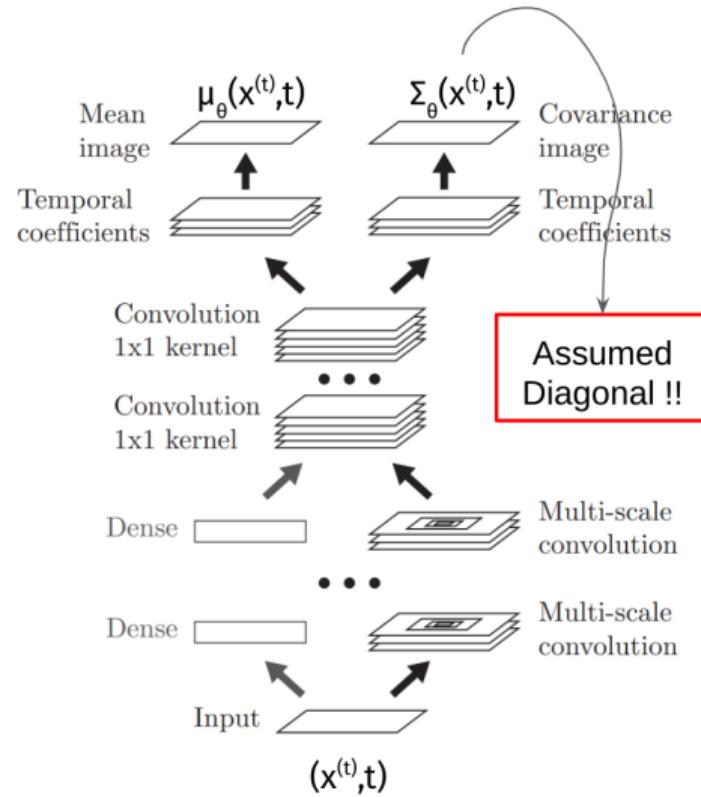
$\mathcal{L}_0, \mathcal{L}_t$  explicit.

# Neural Network that estimate $\mu_\theta$ and $\Sigma_\theta$



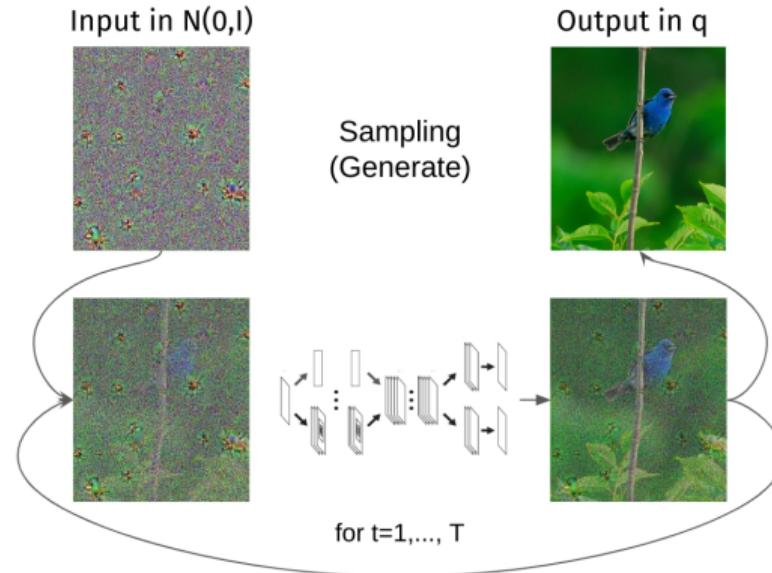
Prposed Neural Network for CIFAR10 image generation. T=1000

# Neural Network that estimate $\mu_\theta$ and $\Sigma_\theta$

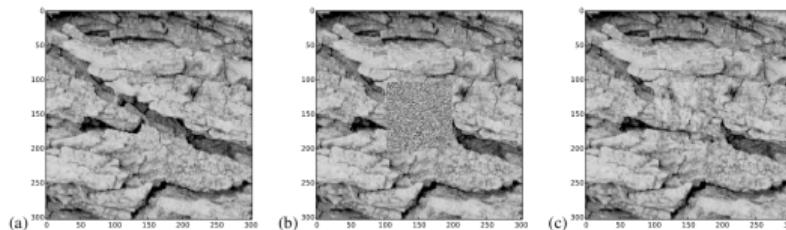


Proposed Neural Network for CIFAR10 image generation. T=1000

# Sampling or Generative Stage



# Experiments



MNIST



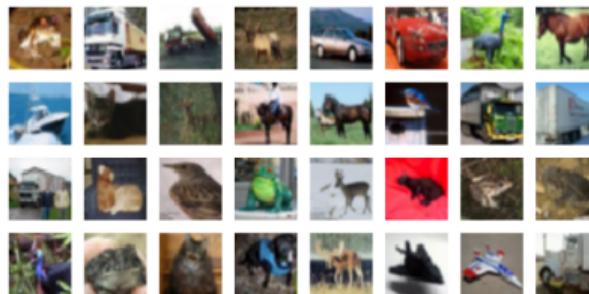
6

---

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

# Experiments

CIFAR10 (original)



CIFAR10 (generated)



7

---

<sup>7</sup>Sohl-Dickstein et al., “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”.

## Timeline

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

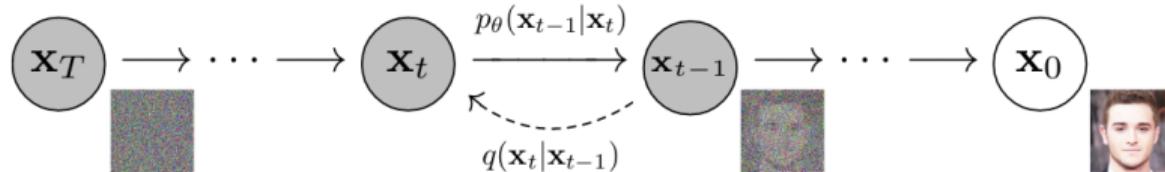


2020) *Denoising Diffusion Probabilistic Models*. Ho et al. NeurIPS.

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

# Denoising Diffusion Probabilistic Model

Small technical improvements highly impact the performance...<sup>8</sup>



## Simplification I: Diagonal Uniform Covariance Matrix

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

where

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

<sup>8</sup>Ho, 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

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

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

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

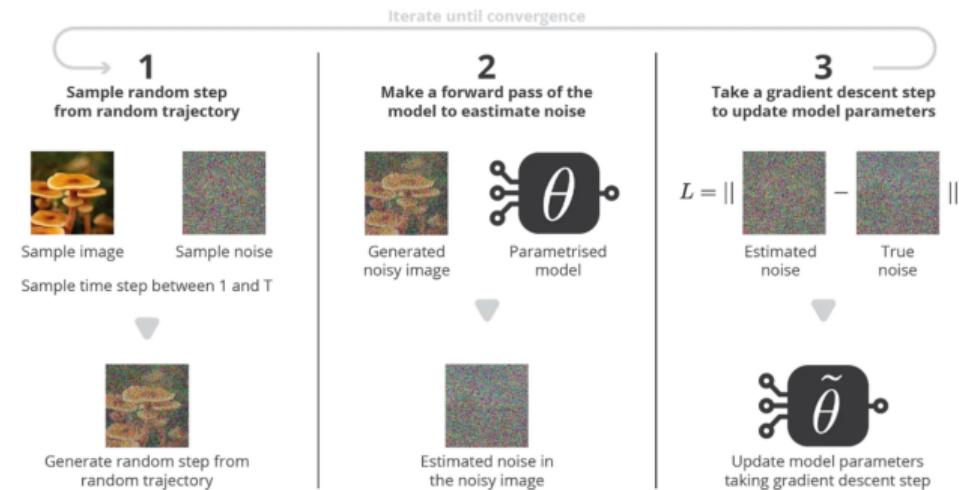
# Training and Sampling Procedure

## 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:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```



# Experiments: Sample Quality

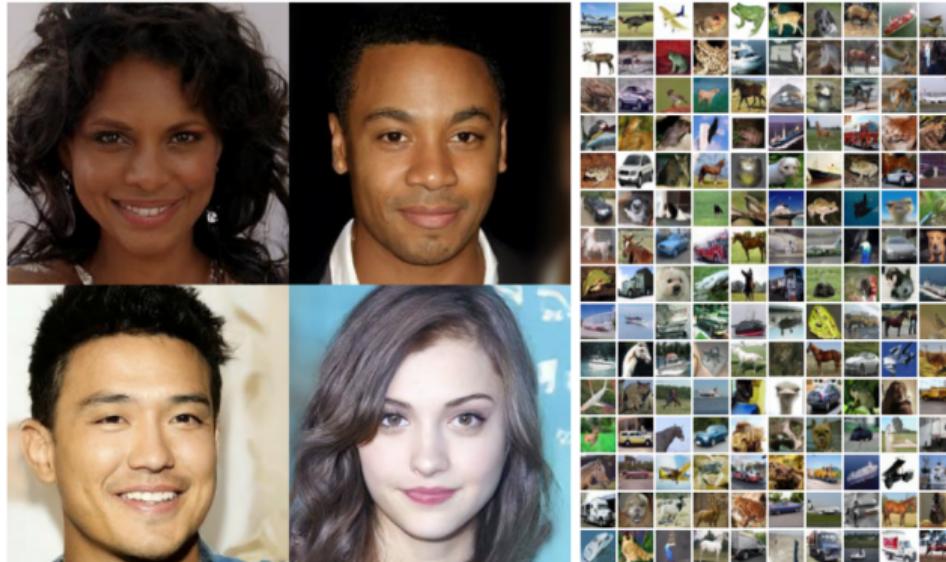


Figure 1: Generated samples on CelebA-HQ  $256 \times 256$  (left) and unconditional CIFAR10 (right)

| Objective  | IS                                | FID         |
|--|-----------------------------------|-------------|
| <b><math>\tilde{\mu}</math> prediction (baseline)</b>          |                                   |             |
| $L$ , learned diagonal $\Sigma$                                | $7.28 \pm 0.10$                   | 23.69       |
| $L$ , fixed isotropic $\Sigma$                                 | $8.06 \pm 0.09$                   | 13.22       |
| $\ \tilde{\mu} - \tilde{\mu}_\theta\ ^2$                       | -                                 | -           |
| <b><math>\epsilon</math> prediction (ours)</b>                 |                                   |             |
| $L$ , learned diagonal $\Sigma$                                | -                                 | -           |
| $L$ , fixed isotropic $\Sigma$                                 | $7.67 \pm 0.13$                   | 13.51       |
| $\ \tilde{\epsilon} - \epsilon_\theta\ ^2 (L_{\text{simple}})$ | <b><math>9.46 \pm 0.11</math></b> | <b>3.17</b> |

Metrics for CIFAR10

Note.

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

# Experiments: Diffusion vs GAN/VAE

*“Diffusion models get comparable result to Generative Adversarial Networks and Variational Autoencoders”*

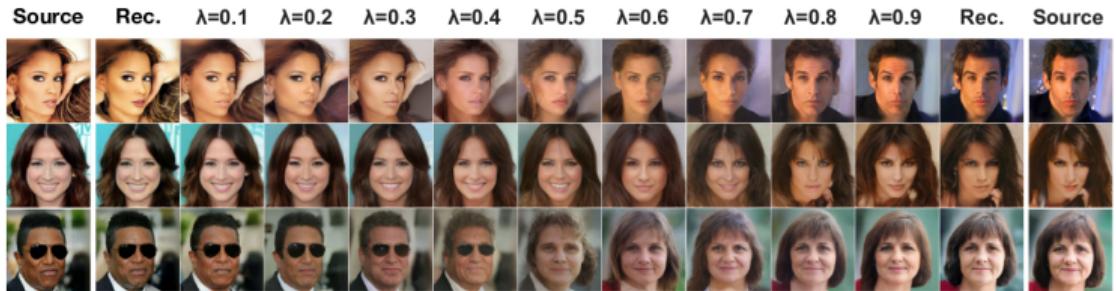
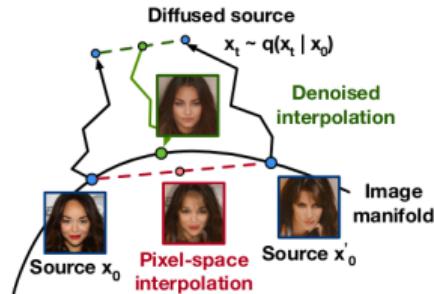
Table 1: CIFAR10 results. NLL measured in bits/dim.

| Model  | IS                                | FID         | NLL Test (Train)   |
|--|-----------------------------------|-------------|--------------------|
| <b>Conditional</b>                           |                                   |             |                    |
| EBM [4]                                      | 8.30                              | 37.9        |                    |
| JEM [4]                                      | 8.76                              | 38.4        |                    |
| BigGAN [3]                                   | 9.22                              | 14.73       |                    |
| StyleGAN2 + ADA (v1) [20]                    | <b>10.06</b>                      | <b>2.67</b> |                    |
| <b>Unconditional</b>                         |                                   |             |                    |
| Diffusion (original) [3]                     |                                   |             | $\leq 5.40$        |
| Gated PixelCNN [59]                          | 4.60                              | 65.93       | 3.03 (2.90)        |
| Sparse Transformer [2]                       |                                   |             | <b>2.80</b>        |
| PixelIQN [44]                                | 5.29                              | 49.46       |                    |
| EBM [4]                                      | 6.78                              | 38.2        |                    |
| NCSNv2 [54]                                  |                                   |             | 31.75              |
| NCSN [53]                                    | $8.87 \pm 0.12$                   | 25.32       |                    |
| SNGAN [29]                                   | $8.22 \pm 0.05$                   | 21.7        |                    |
| SNGAN-DDLS [4]                               | $9.09 \pm 0.10$                   | 15.42       |                    |
| StyleGAN2 + ADA (v1) [20]                    | <b>9.74 <math>\pm 0.05</math></b> | 3.26        |                    |
| Ours ( $L$ , fixed isotropic $\Sigma$ )      | $7.67 \pm 0.13$                   | 13.51       | $\leq 3.70$ (3.69) |
| <b>Ours (<math>L_{\text{simple}}</math>)</b> | $9.46 \pm 0.11$                   | <b>3.17</b> | $\leq 3.75$ (3.72) |

CIFAR10 results. NLL measured in bits/dim<sup>10</sup>

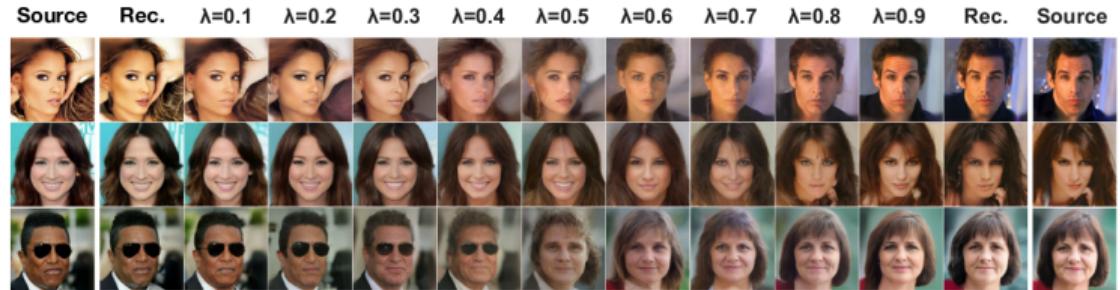
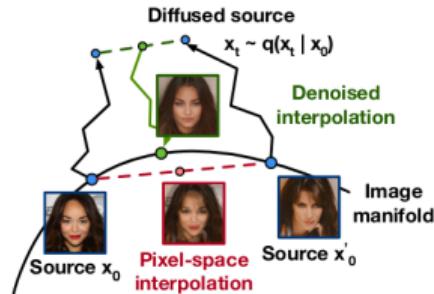
<sup>10</sup>Ho, Jain, and Abbeel, “Denoising Diffusion Probabilistic Models”.

# 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



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

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

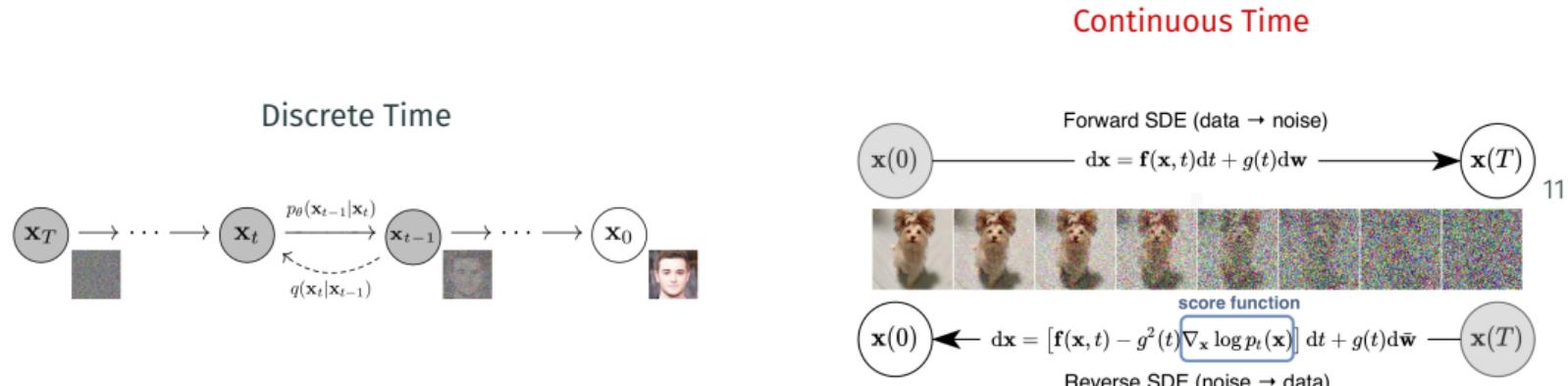
## Timeline

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  $p_t(\mathbf{x})$ ;

<sup>11</sup>Song et al., “Score-based generative modeling through stochastic differential equations”.

# Continuous Diffusion Process described by SDE

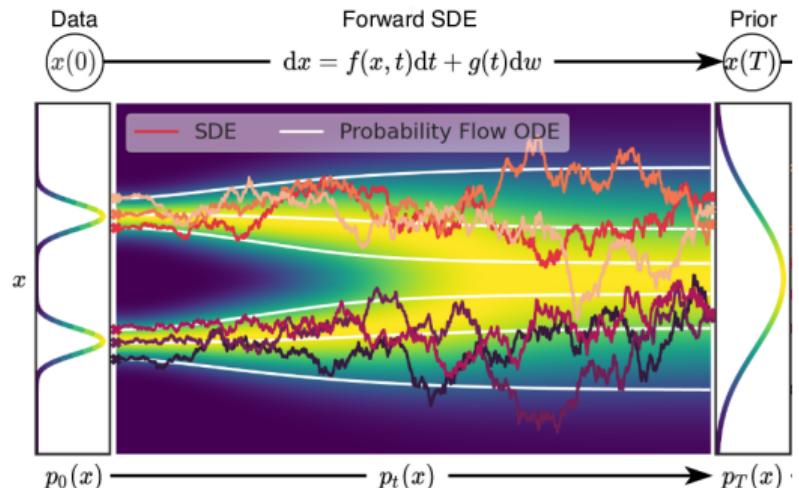
## Continuous Diffusion Process

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

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

## Weiner Process

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



# Continuous Diffusion Process described by SDE

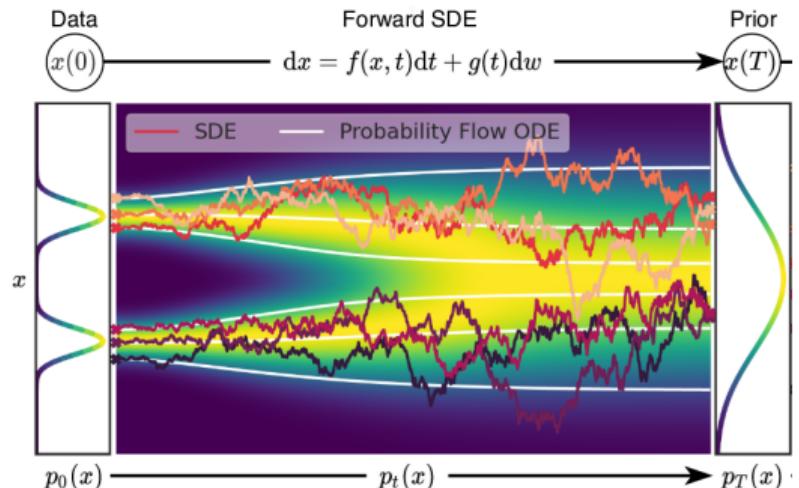
## Continuous Diffusion Process

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

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 explicitly given by $\nabla_x p_t(\mathbf{x})$

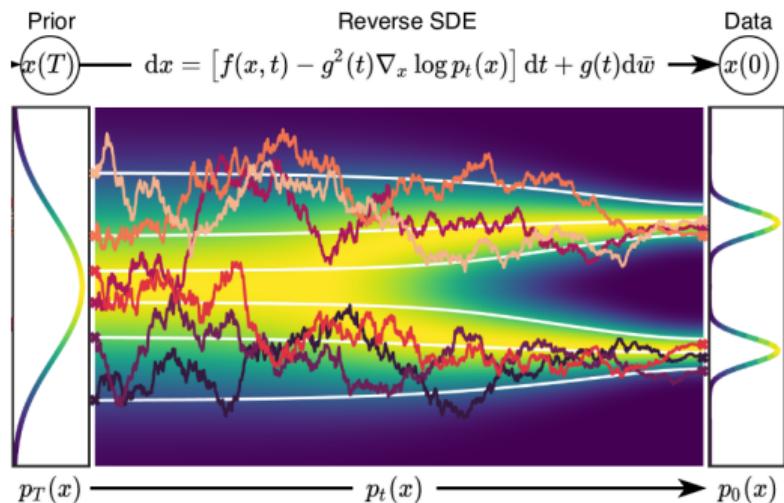
## Theorem

Reverse Process is still a diffusion process

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

where

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



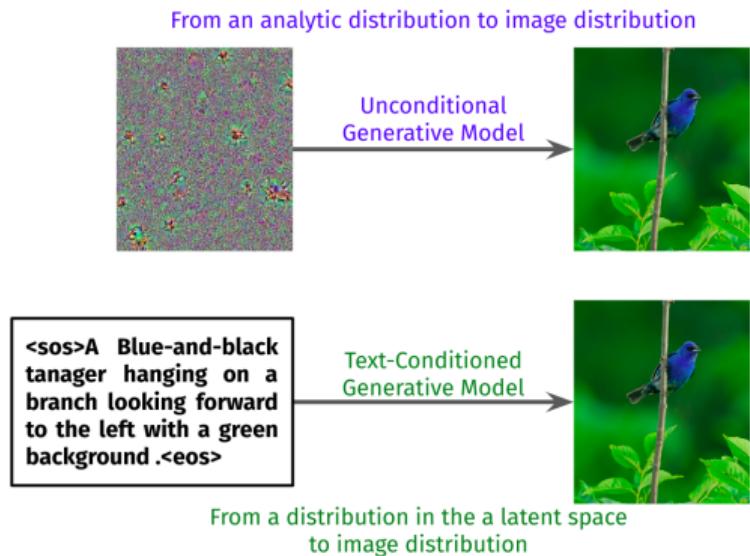
DALL-E2

---



# Introduction

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



# 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



# 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

riding a horse lounging in a tropical resort in space playing basketball with cats in space

in a photorealistic style in the style of Andy Warhol as a pencil drawing



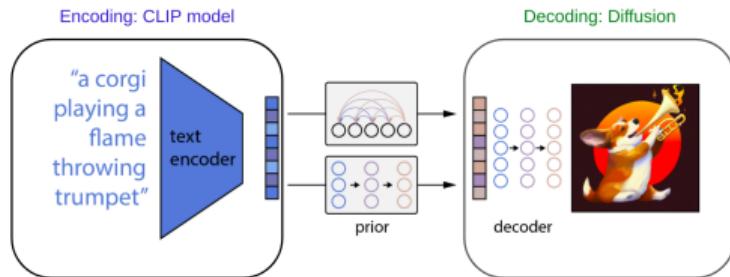
12

---

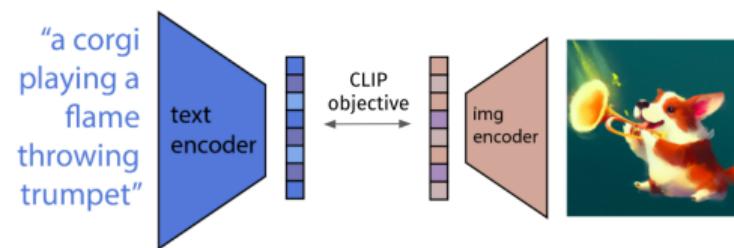
<sup>12</sup>Credits: DALL-E2 website

# Overview: CLIP + Diffusion

## Text-to-Image Generation with CLIP



## CLIP model



# CLIP: Latent Embedding model

<++>



## Broader Impacts

---

# Image Generation

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

---

<sup>13</sup>Ho, Jain, and Abbeel, “Denoising Diffusion Probabilistic Models”.

# Image Generation

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

*“If samples from generative models trained on these datasets proliferate throughout the internet, then these biases will only be reinforced further.<sup>13</sup>”*

---

<sup>13</sup>Ho, Jain, and Abbeel, “Denoising Diffusion Probabilistic Models”.

## CLIP Model

*“We also found discrepancies across gender and race for people categorized into the ‘crime’ and ‘non-human’ categories...”<sup>14</sup>*

---

<sup>14</sup>Radford et al., “Learning Transferable Visual Models From Natural Language Supervision”.

# Thanks for the attention

Fabio Brau

-  Scuola Superiore Sant'Anna, Pisa
-  fabio.brau@santannapisa.it
-  retis.santannapisa.it/~f.brau
-  linkedin.com/in/fabio-brau

TELECOMMUNICATIONS,  
COMPUTER  
ENGINEERING,  
AND PHOTONICS  
INSTITUTE



Sant'Anna  
School of Advanced Studies – Pisa

ECCELLENZA  
MIUR 2018-2022

ROBOTICS & AI



Sant'Anna  
Scuola Universitaria Superiore Pisa

  
*Retis*  
Real-Time Systems Laboratory

## Proof Details

---



## Proof of Explicit Representation of Forward Diffusion Process

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

