Diffusion Models and an application to DALL-E2

Deep Learning and Neural Networks: Advanced Topics

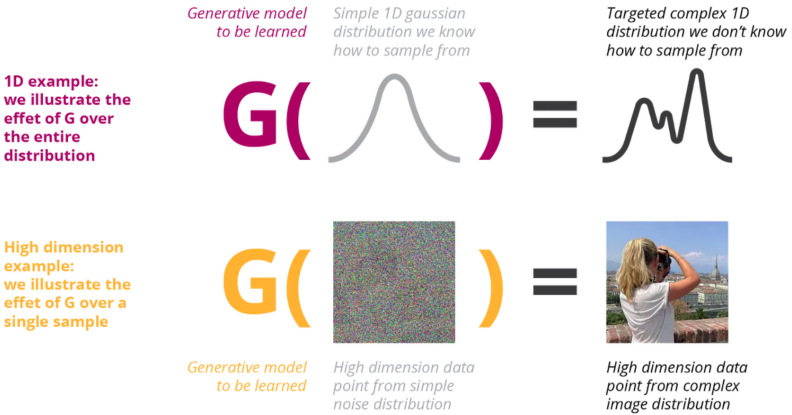
Fabio Brau

March 16, 2023

Scuola Superiore Sant'Anna, Pisa.

Introduction

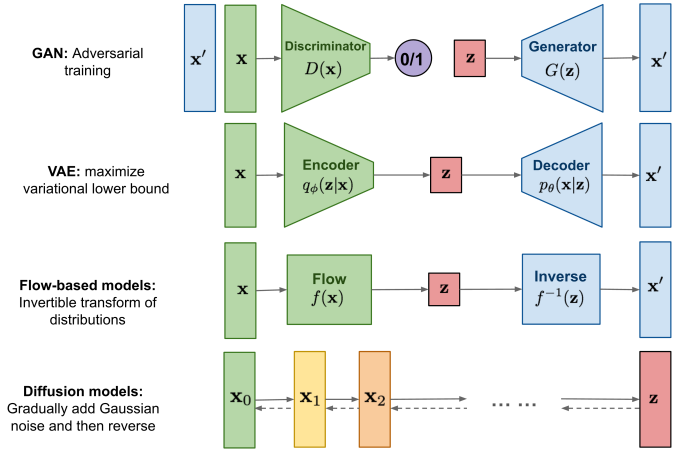
Generative Models



Schema of generative models[1]

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

Generative Models

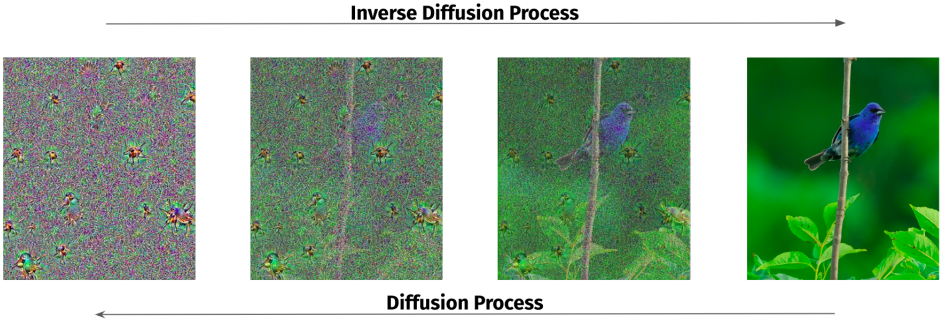


Overview of different generative models[2]

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

Diffusion Models

Diffusion models are generative models that aim at denoising data



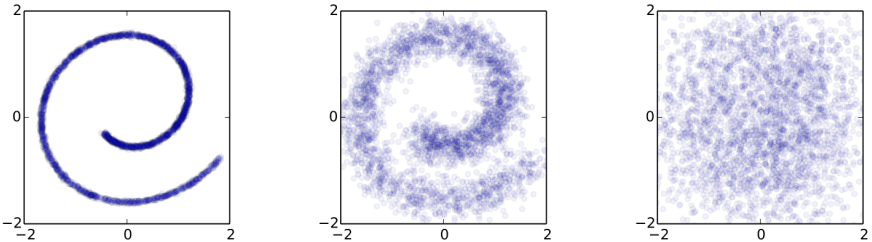
Timeline

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.

$t = 0$ $t = \frac{T}{2}$ $t = T$



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

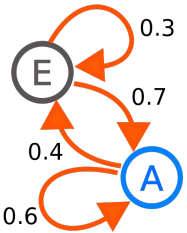
[3] Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics".

Informal Definition

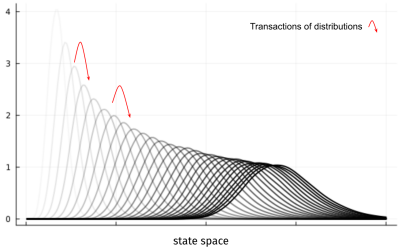
A sequence of random variables $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(t)}, \cdots$, such that

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

Discrete State Space S



Continuous State Space S



Reminder: Disrete Time Markov Chain with Discrete State Space

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

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

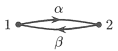
Definition
A sequence $\left\{\mathbf{x}^{(t)}\right\}_{t\in\mathbb{N}} \subseteq S$, a matrix $P = (p_{ij})$.

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

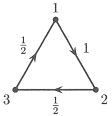
P is a stochastic matrix!

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

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



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



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

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

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

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

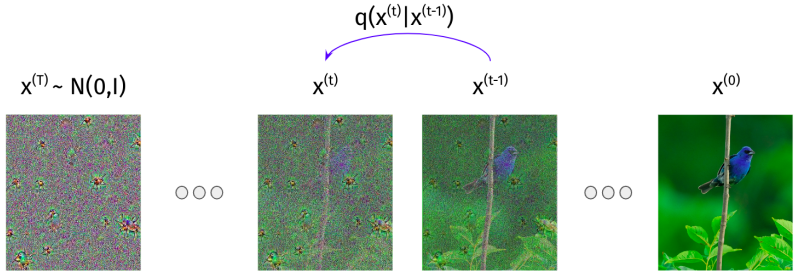
$$p\left(\mathbf{x}, \mathbf{y}\right) = p(\mathbf{x} \,|\, \mathbf{y}) \, p\left(\mathbf{y}\right)$$

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

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

"Adding noise to data..."

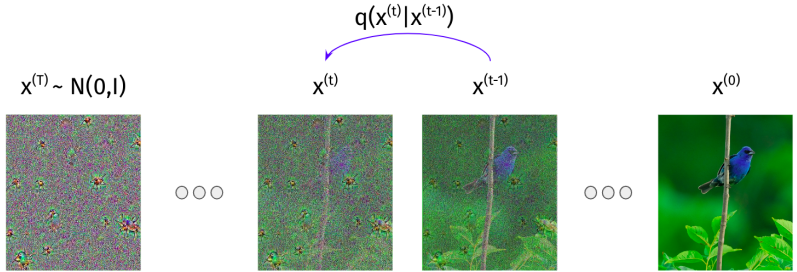
- Data Distribution: $\mathbf{x}^{(0)} \sim q$
- Transaction Kernel: $q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1-\beta_t}\mathbf{x}^{(t-1)}; \beta_t I\right)$
- Variance Scheduler: $0 < \beta_1 < \cdots < \beta_T < 1$



"Adding noise to data..."

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

Not Analytic!!



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 ≈ 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

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

Markov property allows breaking up distributional Representation...

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

Markov property allows breaking up distributional Representation...

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

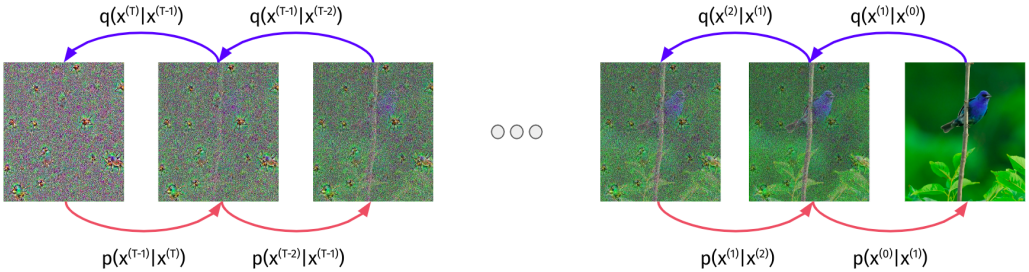
Markov property allows breaking up distributional Representation...

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

Distributional Representation

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

Reverse Diffusion Process



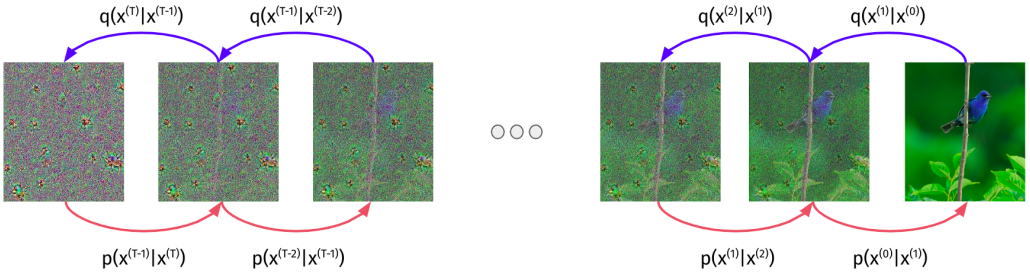
Fixed Forward Process

Initial Distribution

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

Gaussian Transaction Kernel

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



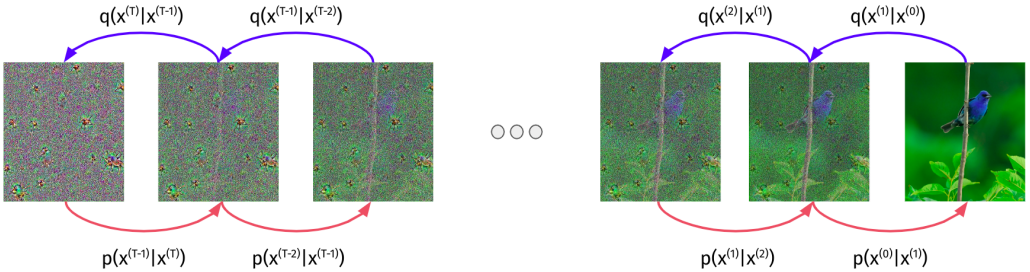
Fixed Forward Process

Initial Distribution

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

Gaussian Transaction Kernel

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



Initial Distribution

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

Approximation of

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

Gaussian Kernel with parameters

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

Learned Reverse Process

Forward Diffusion Process

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

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

Reverse Diffusion Process

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

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

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

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 ≈ Gaussian DP[4]

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

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

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

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 ≈ Gaussian DP[4]

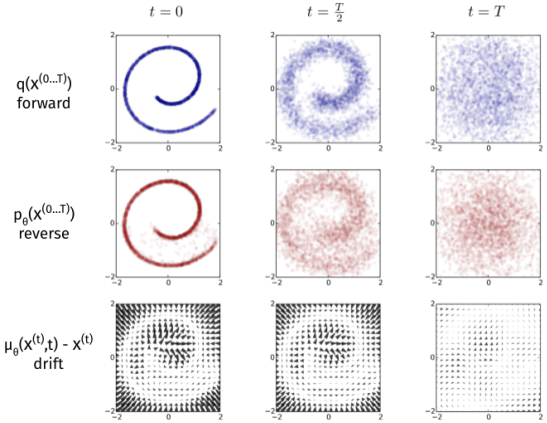
If $|\beta_i - \beta_{i+1}| \approx 0$, i.e. diffusion **slow enough**, then

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

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

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

Visualization of Diffusion Process: 2D dimensional case



5

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

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

Goal

Search for the best parameters θ

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

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

Estimated Reverse Process

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

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

Goal

Search for the best parameters θ

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

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

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

Goal

Search for the best parameters θ

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

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

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

Goal

Search for the best parameters θ

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

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

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!!

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

Goal: Deduce a tractable loss function

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

Goal: Deduce a tractable loss function

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

Simplification I: Minimize the Cross Entropy

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

Goal: Deduce a tractable loss function

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

Simplification I: Minimize the Cross Entropy

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

Minimize the Cross Entropy Loss

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

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

Minimize the Cross Entropy Loss

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

Observation: Marginal Distribution

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

Minimize the Cross Entropy Loss

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

Observation: Marginal Distribution

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

Simplification II: Jensen Inequality

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

...after some algebraic steps :)

Reformulated Loss Function

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

where,

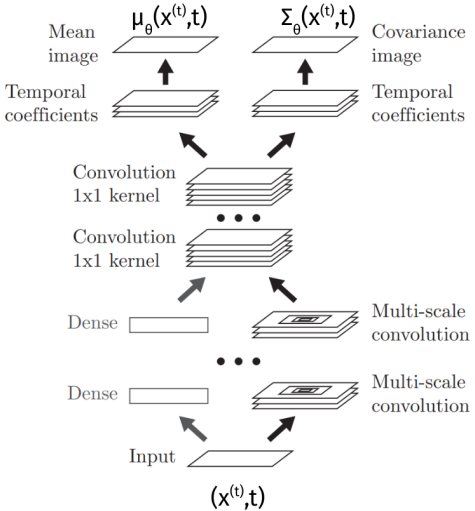
$$\mathcal{L}_T = \mathbb{E}_{q(\mathbf{x}^{(0\ldots T)})} \left[D_{KL} \left(q(\mathbf{x}^{(T)} \mid \mathbf{x}^{(0)}) \quad || \quad p_\theta(\mathbf{x}^{(T)}) \right) \right]$$

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

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

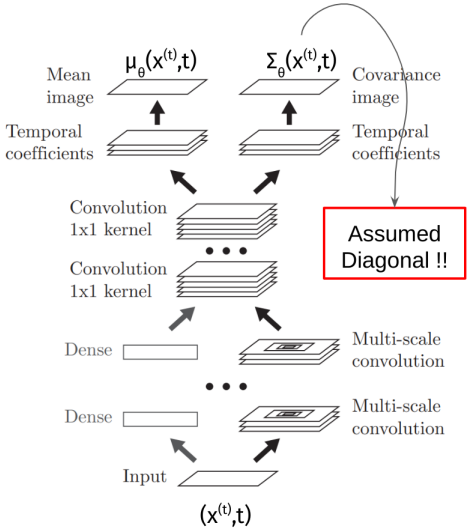
Note.
$\mathcal{L}_T$ is constant.
$\mathcal{L}_0$, $\mathcal{L}_t$ explicit.

Neural Network that estimate μ_θ and Σ_θ



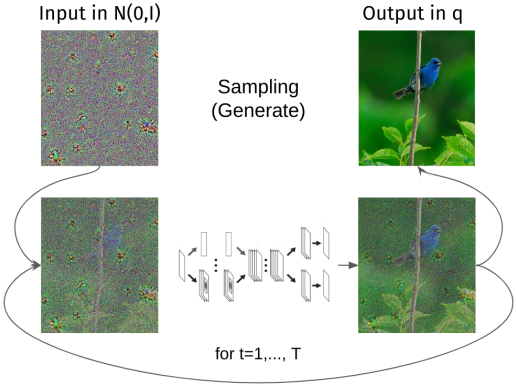
Proposed Neural Network for CIFAR10 image generation. T=1000

Neural Network that estimate μ_θ and Σ_θ



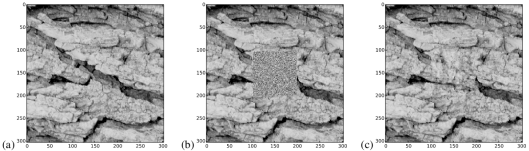
Proposed Neural Network for CIFAR10 image generation. T=1000

Sampling or Generative Stage

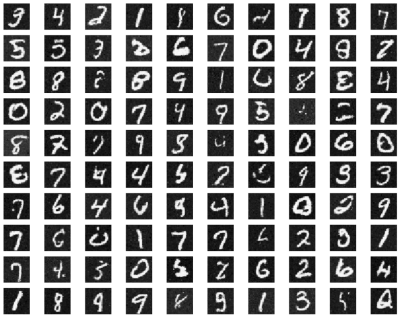


Experiments[6]

Bark Dataset



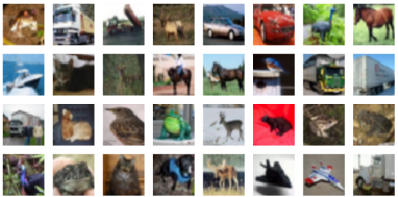
MNIST



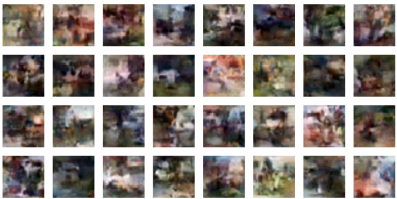
[6] Sohl-Dickstein et al., “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”.

Experiments[7]

CIFAR10 (original)



CIFAR10 (generated)



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

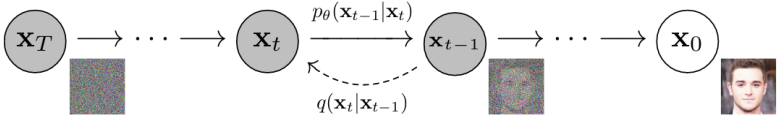
Timeline

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

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

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

Small technical improvements highly impact the performance...[8]



Simplification I: Diagonal Uniform Covariance Matrix

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

where

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

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

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

Simplification II: Estimating the commited error

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

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

Simplification II: Estimating the commited error

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

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

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

Simplification II: Estimating the commited error

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

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

Simplification III: Training on random istants t

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

where

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

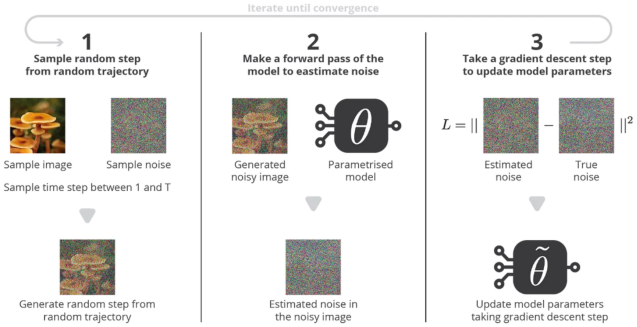
Training and Sampling Procedure

Algorithm 1 Training

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

Algorithm 2 Sampling

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



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

Experiments: Sample Quality

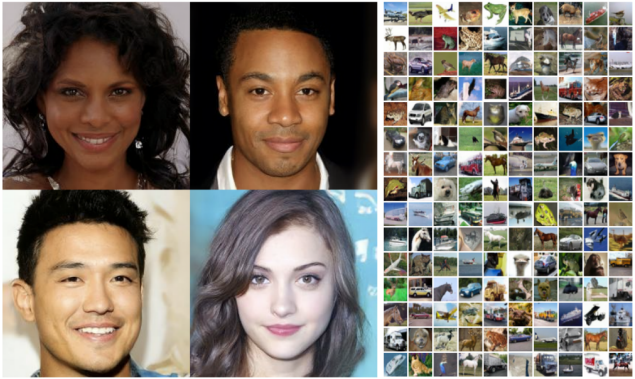


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

Objective	IS	FID
$\tilde{\mu}$ prediction (baseline)		
L, learned diagonal 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})	$\mathbf{9.46 \pm 0.11}$	**3.17**

Metrics for CIFAR10

Note.

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

"Diffusion models get comparable result to Generative Adversarial Networks and Variational Autoencoders"

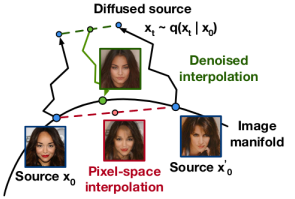
Table 1: CIFAR10 results. NLL measured in bits/dim.

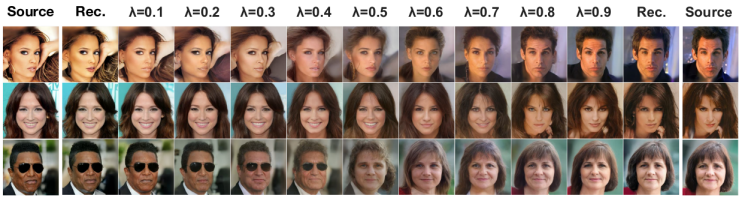
Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [55]	8.87 ± 0.12	25.32	
SNGAN [39]	8.22 ± 0.05	21.7	
SNGAN-DDLS [4]	9.09 ± 0.10	15.42	
StyleGAN2 + ADA (v1) [29]	$\mathbf{9.74} \pm 0.05$	3.26	
Ours (L, fixed isotropic $\mathbf{\Sigma}$)	7.67 ± 0.13	13.51	≤ 3.70 (3.69)
Ours (L_{simple})	9.46 ± 0.11	**3.17**	≤ 3.75 (3.72)

CIFAR10 results. NLL measured in bits/dim[10]

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

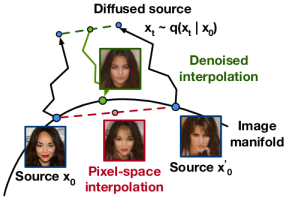
Experiments: Images Interpolation

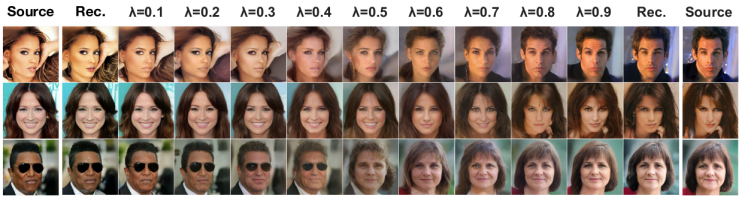




$$\mathbf{x}_\lambda^{(T)} := \lambda \mathbf{x}_{\texttt{source}_\texttt{r}}^{(T)} + (1-\lambda)\mathbf{x}_{\texttt{source}_\texttt{l}}^{(T)}, \quad \lambda \in [0,1]$$

Experiments: Images Interpolation





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

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

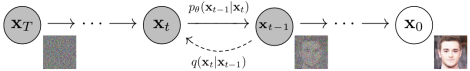
Timeline

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

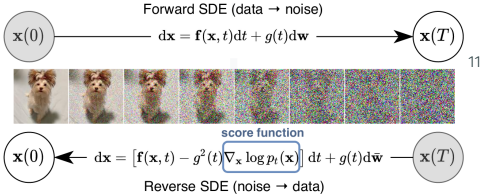
2020) *Denoising Diffusion Probabilistic Models*. Ho et al. NeurIPS.✓

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

Discrete Time



Continuous Time



11

Stochastic Process

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

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

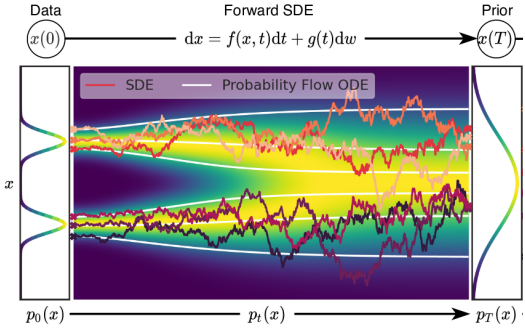
Continous Diffusion Process

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

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 $\nabla_{\mathbf{x}} p_t(\mathbf{x})$

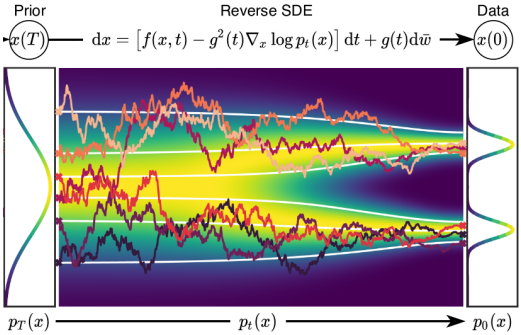
Theorem

Reverse Process is still a diffusion process

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

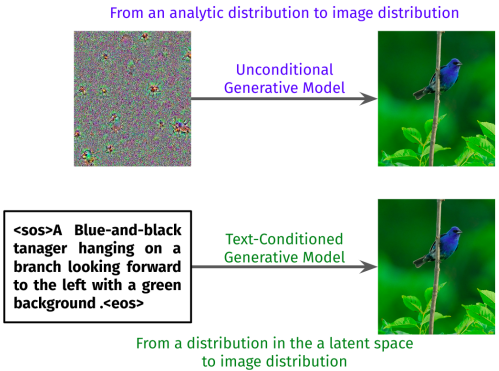
where

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



DALL-E2

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



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

→

12

[12] Credits: DALL-E2 website

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

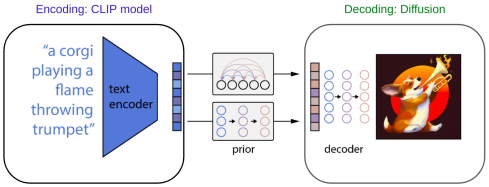
in a photorealistic style in the style of Andy Warhol as a pencil drawing

→

12

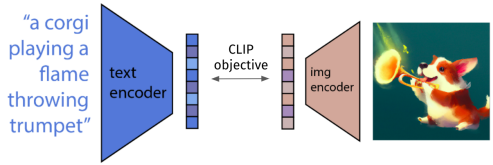
[12] Credits: DALL-E2 website

Text-to-Image Generation with CLIP



Diffusion models allows generating images with high sample fedelity and photorealism

CLIP model



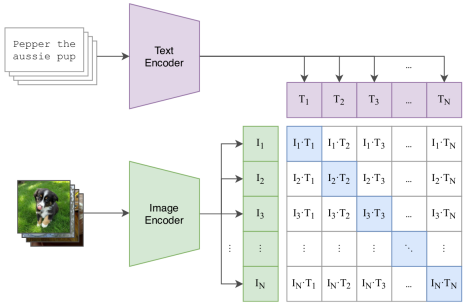
Embedding of images and text in the same latent space

[13] Ramesh et al., “Hierarchical Text-Conditional Image Generation with CLIP Latents”.

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.

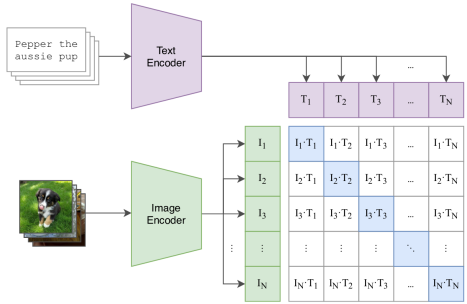
14

[14] Radford et al., “Learning Transferable Visual Models From Natural Language Supervision”.

4. **Aim**: Zero-shot classification (of unseen datasets, see later.)

5. The **Text Encoder** is a Transformer for NLP ($63M$ of parameters, next lectures).

6. The **Image Encoder** include ResNet-50 and Vision-Transformers

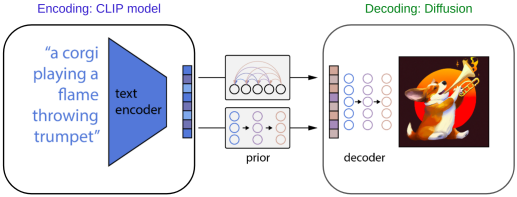


CLIP allows embedding of images and text in the same latent space.

14

[14] Radford et al., “Learning Transferable Visual Models From Natural Language Supervision”.

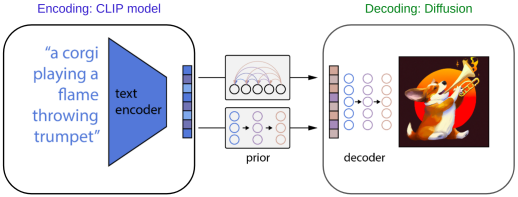
DALL-E2 with higher detail



Let (x, y) be an (image, caption) pair.

1. **CLIP** such that $(x, y) \mapsto (z_i, z_t)$.
2. **Prior** which estimates $q(\mathbf{z}_i \mid \mathbf{y})$
3. **Decoder** which estimate $q(\mathbf{x} \mid \mathbf{z}_i, \mathbf{y})$

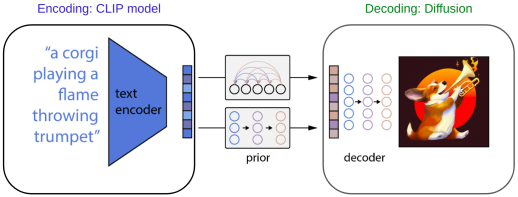
DALL-E2 with higher detail



Let (x, y) be an (image, caption) pair.

1. **CLIP** such that $(x, y) \mapsto (z_i, z_t)$.
2. **Prior** which estimates $q(\mathbf{z}_i \mid \mathbf{y})$ — *Provide an image embedding conditioned to a text.*
3. **Decoder** which estimate $q(\mathbf{x} \mid \mathbf{z}_i, \mathbf{y})$ — *Provide an Image conditioned to a text and Image Emb.*

DALL-E2 with higher detail



Let (x, y) be an (image, caption) pair.

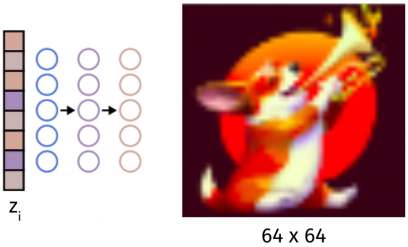
1. **CLIP** such that $(x, y) \mapsto (z_i, z_t)$.
2. **Prior** which estimates $q(\mathbf{z}_i \mid \mathbf{y})$ — *Provide an image embedding conditioned to a text.*
3. **Decoder** which estimate $q(\mathbf{x} \mid \mathbf{z}_i, \mathbf{y})$ — *Provide an Image conditioned to a text and Image Emb.*

Observation

$$p(\mathbf{x} \mid \mathbf{y}) \underbrace{=}_{\text{Deterministic}} p(\mathbf{x}, \mathbf{z}_i \mid \mathbf{y}) \underbrace{=}_{\text{Bayes Rule}} p(\mathbf{x} \mid \mathbf{z}_i, \mathbf{y}) p(\mathbf{z}_i \mid \mathbf{y})$$

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.

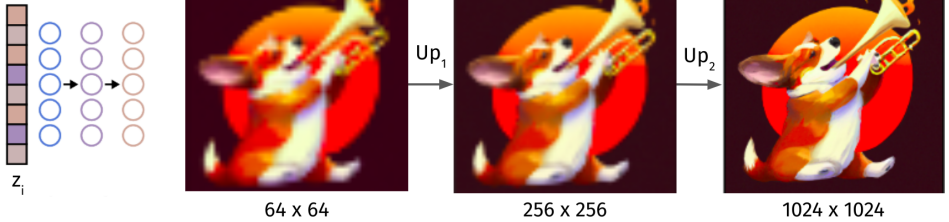


Step 1: Generate a low-resolution image from z_i

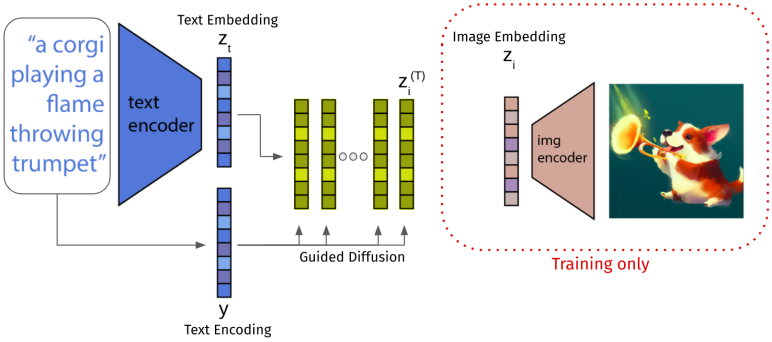
A guided diffusion model is used to get a low resolution image from the image embedding.

Step 2. Upsampling Procedure

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

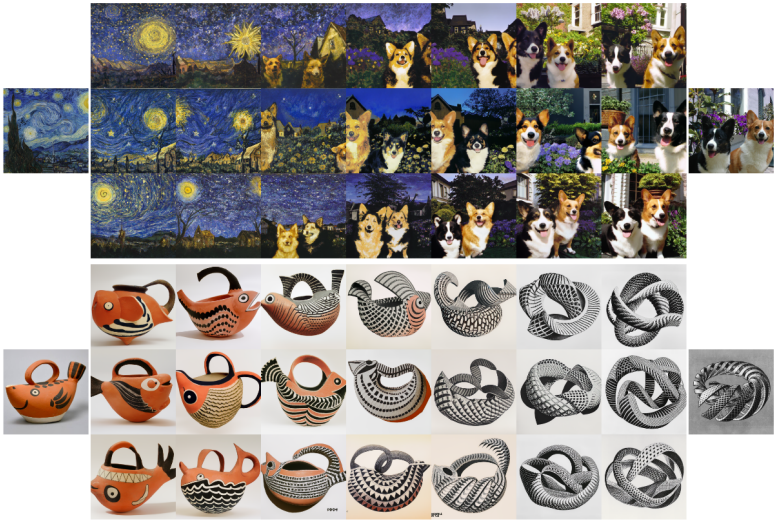


Diffusion Prior: "The image embedding $\mathbf{z}_i$ is directly modelled using a Gaussian diffusion model conditioned on the caption $\mathbf{y}$".



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 $\mathbf{z}_i^{(0)} = \mathbf{z}_t$

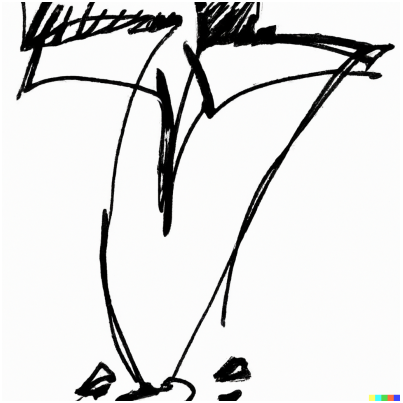
Experiments:Image Interpolation



For each row, the decoder is applied to a combination of $\mathbf{z}_i(\textit{left})$ and $\mathbf{z}_i(\textit{right})$.

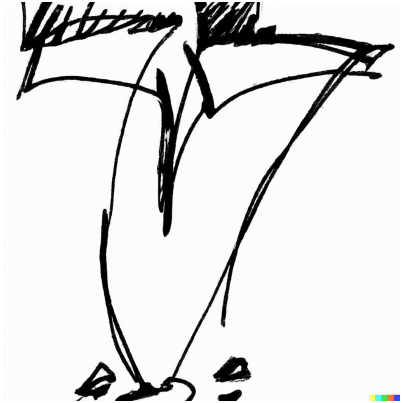
Representation of the Dignity

Representation of the Dignity



DALL-E2

Representation of the Dignity



DALL-E2

Matt Groening (S8 E6)

Experiments: Ricreate Anchient Works

Benjamin Hilton
@benjamin_hilton · **Follow**

Got access to DALL-E - here, have a medieval painting of the wifi not working

10:45 PM · Apr 27, 2022

Lapine
@LapineDeLaTerre · **Follow**

An ancient Egyptian painting depicting an argument over whose turn it is to take out the trash #dalle

10:23 AM · Apr 24, 2022

Conclusion

Broader Impacts

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

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

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

Summary

1. Diffusion Generative Models (from 2015 to 2022)

2. Overview on CLIP model for representation learning.

3. The Diffusion models inside DALL-E2

Thanks for the attention

Fabio Brau

Scuola Superiore Sant'Anna, Pisa

fabio.brau@santannapisa.it

retis.santannapisa.it/~f.brau

linkedin.com/in/fabio-brau

TELECOMMUNICATIONS, COMPUTER ENGINEERING, AND PHOTONICS INSTITUTE

Sant'Anna School of Advanced Studies – Pisa

ECCELLENZA MIUR 2018-2022

ROBOTICS & AI

Sant'Anna Scuola Universitaria Superiore Pisa

retis Real-Time Systems Laboratory

Proof Details

Proof of Explicit Representation of Forward Diffusion Process

Let us proceeding by induction by assuming $\mathbf{x}^{(t)} = \sqrt{1-\alpha_t}\,\mathbf{x}^{(0)} + \sqrt{\alpha_t}\,\boldsymbol{\varepsilon}$ where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, I)$ and where $\alpha_t = 1 - \prod_{i=0}^{t}(1-\beta_i)$.

$$
\begin{aligned}
\mathbf{x}^{(t+1)} &= \sqrt{1-\beta_{t+1}}\,\mathbf{x}^{(t)} + \sqrt{\beta_{t+1}}\,\boldsymbol{\varepsilon}_{t+1} \\
&= \sqrt{1-\beta_{t+1}}\,\left(\sqrt{1-\alpha_t}\,\mathbf{x}^{(0)} + \sqrt{\alpha_t}\,\boldsymbol{\varepsilon}\right) + \sqrt{\beta_{t+1}}\,\boldsymbol{\varepsilon}_{t+1} \\
&= \sqrt{\left(\prod_{i=0}^{t+1}(1-\beta_i)\right)}\mathbf{x}^{(0)} + \sqrt{(1-\beta_{t+1})\alpha_t + \beta_{t+1}}\,\tilde{\boldsymbol{\varepsilon}}
\end{aligned}
\tag{2}
$$

where the last term of the summation is obtained by observing that, since $\sqrt{(1-\beta_{t+1})\alpha_t}\,\boldsymbol{\varepsilon}$ and $\sqrt{\beta_{t+1}}\,\boldsymbol{\varepsilon}_{t+1}$ are independent, then the variance of their sum (that still has a gaussian distribution) is given by $(1-\beta_{t+1})\alpha_t + \beta_{t+1}$.