Carnegie Mellon University

Intermediate Deep Learning

Spring 2025, Deep Learning for Engineers
March 20, 2025, 4th Session

Amir Barati Farimani
Associate Professor of Mechanical Engineering and Bio-Engineering
Carnegie Mellon University

Diffusion Models

https://arxiv.org/abs/2006.11239

Outline

1. Introduction
2. Method
3. Experiment
4. Conclusion

Denoising Diffusion Model

“Diffusion Models Beat GANs on Image Synthesis” Dhariwal & Nichol, OpenAI, 2021

“Cascaded Diffusion Models for High Fidelity Image Generation” Ho et al., Google, 2021

Carnegie Mellon University

Text To Image Generation

DALL. E2

"a teddy bear on a skateboard in times square"

"Hierarchical Text-Conditional Image Generation with CLIP Latents" Ramesh et al.,2022

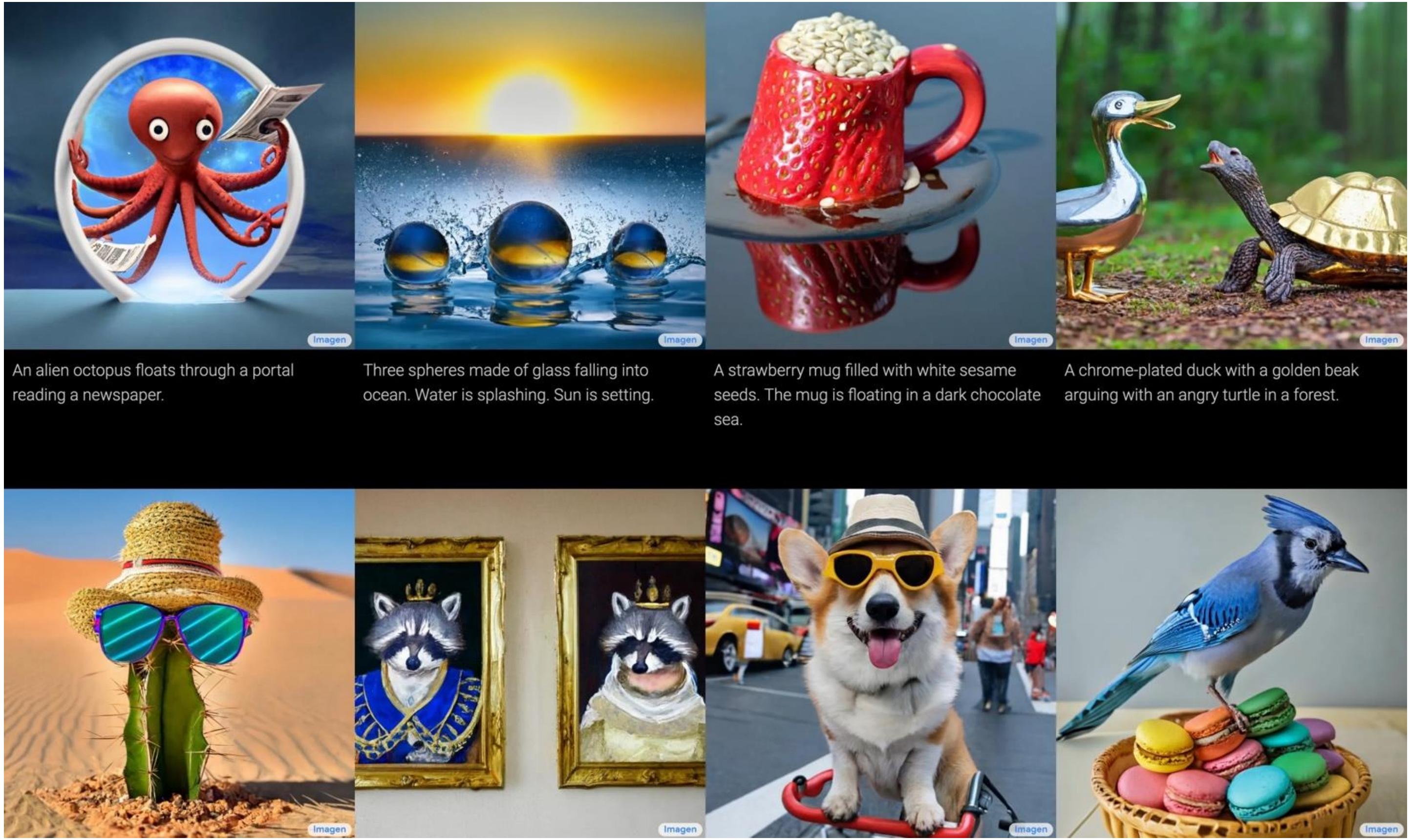
Imagen

A group of teddy bears in suit in a corporate office celebrating the birthday of their friend. There is a pizza cake on the desk.

"Photorealistic Tex-to-Image Diffusion Models with Deep Language Understanding", Saharia et al.,2022

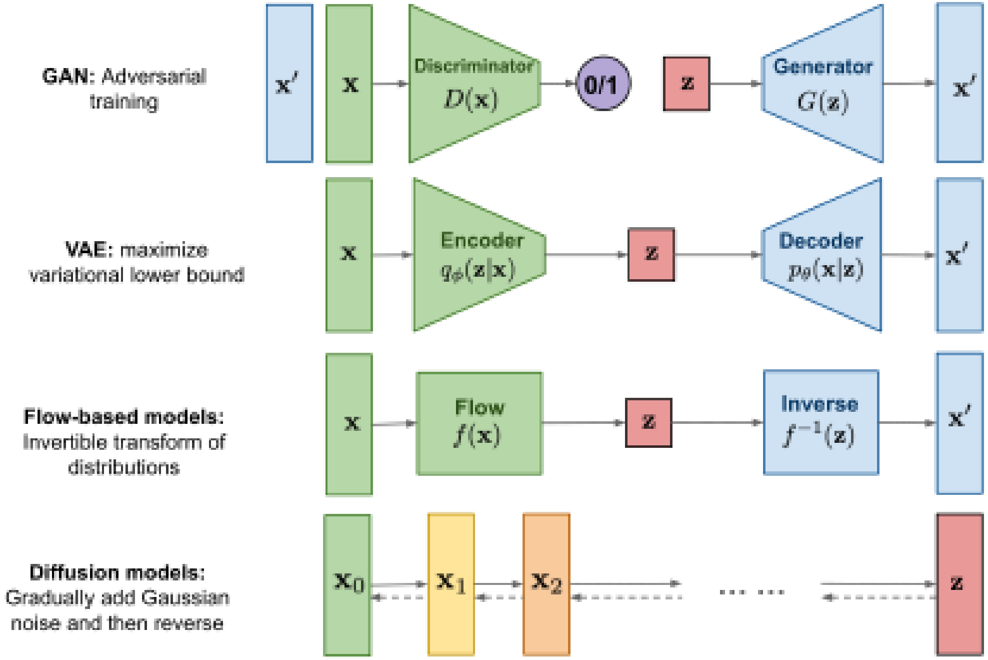
Carnegie Mellon University

Imagen



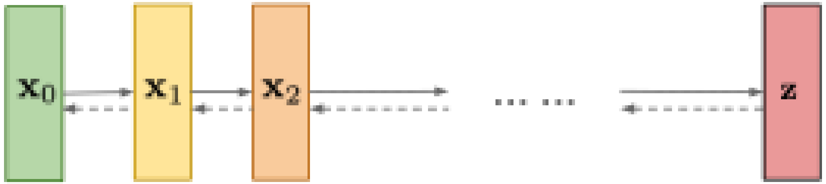
An alien octopus floats through a portal reading a newspaper.
Three spheres made of glass falling into ocean. Water is splashing. Sun is setting.
A strawberry mug filled with white sesame seeds. The mug is floating in a dark chocolate sea.
A chrome-plated duck with a golden beak arguing with an angry turtle in a forest.

Generative Models

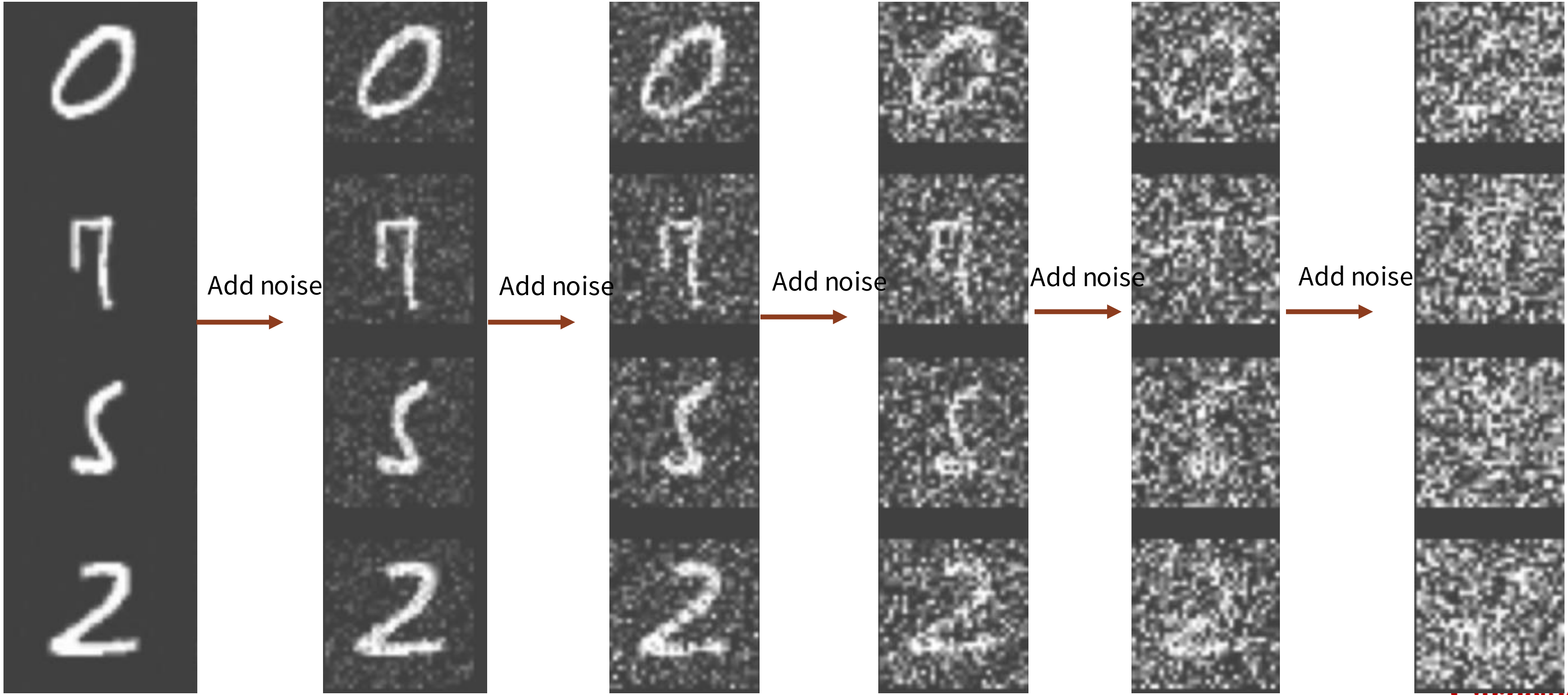


Diffusion Models

Diffusion models:
Gradually add Gaussian noise and then reverse



Diffusion Models



Carnegie Mellon University

Diffusion Models

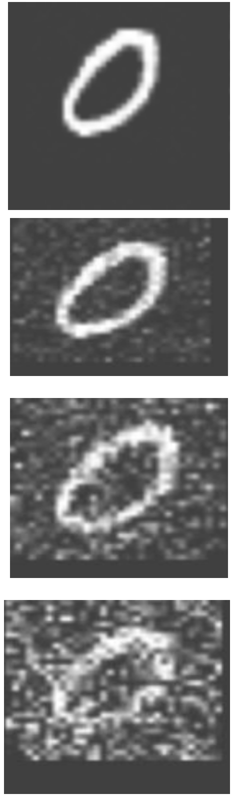
X1 → F → P(X1)=0.98

X2 → F → P(X2)=0.78

X3 → F → P(X3)=0.48

X4 → F → P(X4)=0.2

Diffusion Models



X1 — F → P(X1)=0.98

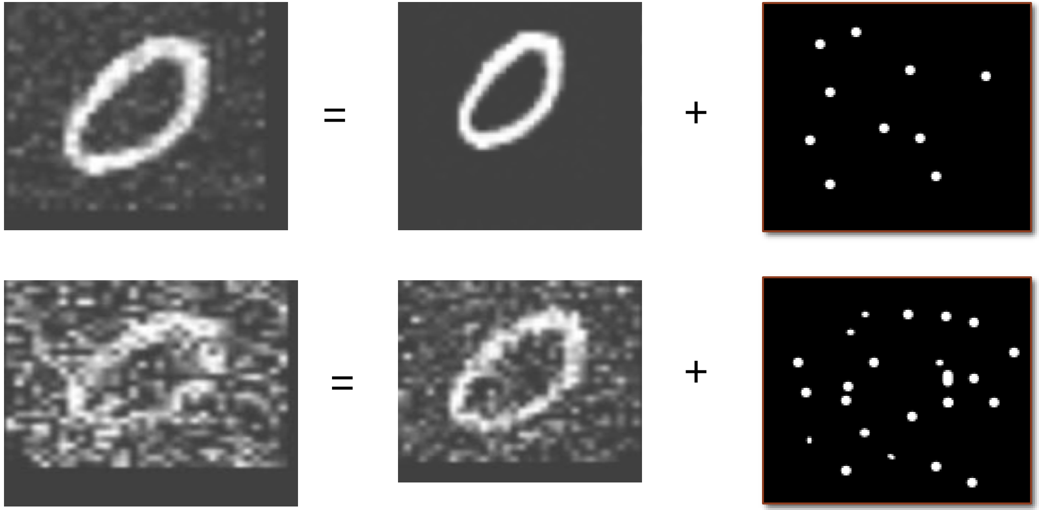
X2 — F → P(X2)=0.78

X3 — F → P(X3)=0.48

X4 — F → P(X4)=0.2

$$\frac{\text{Grad}(P(X1))}{\text{Grad}(X1)}$$

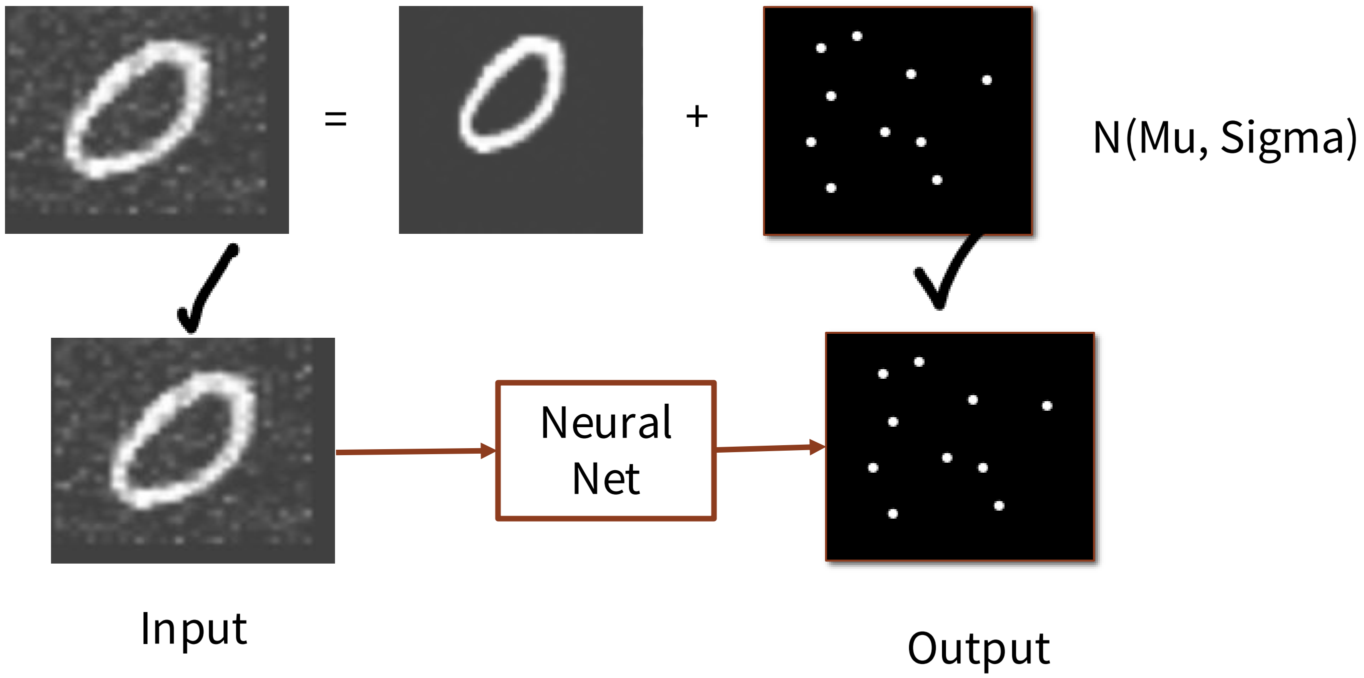
Diffusion Models



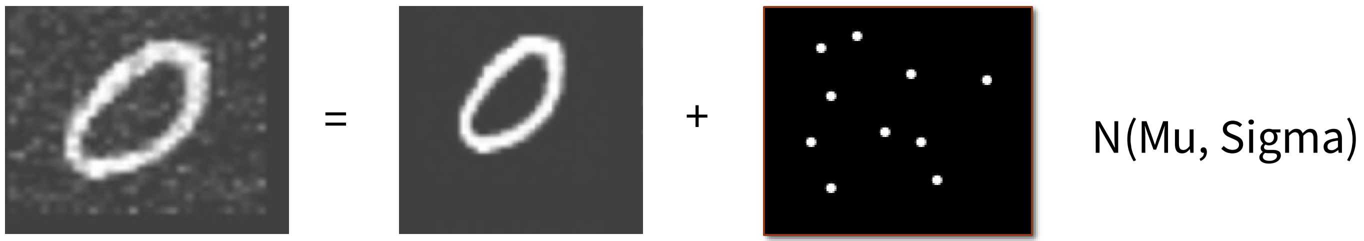
N(Mu, Sigma)

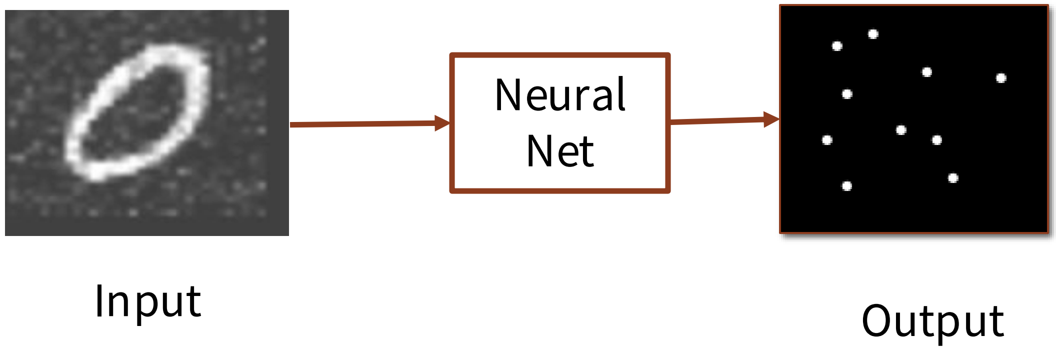
N(Mu, Sigma)

Diffusion Models

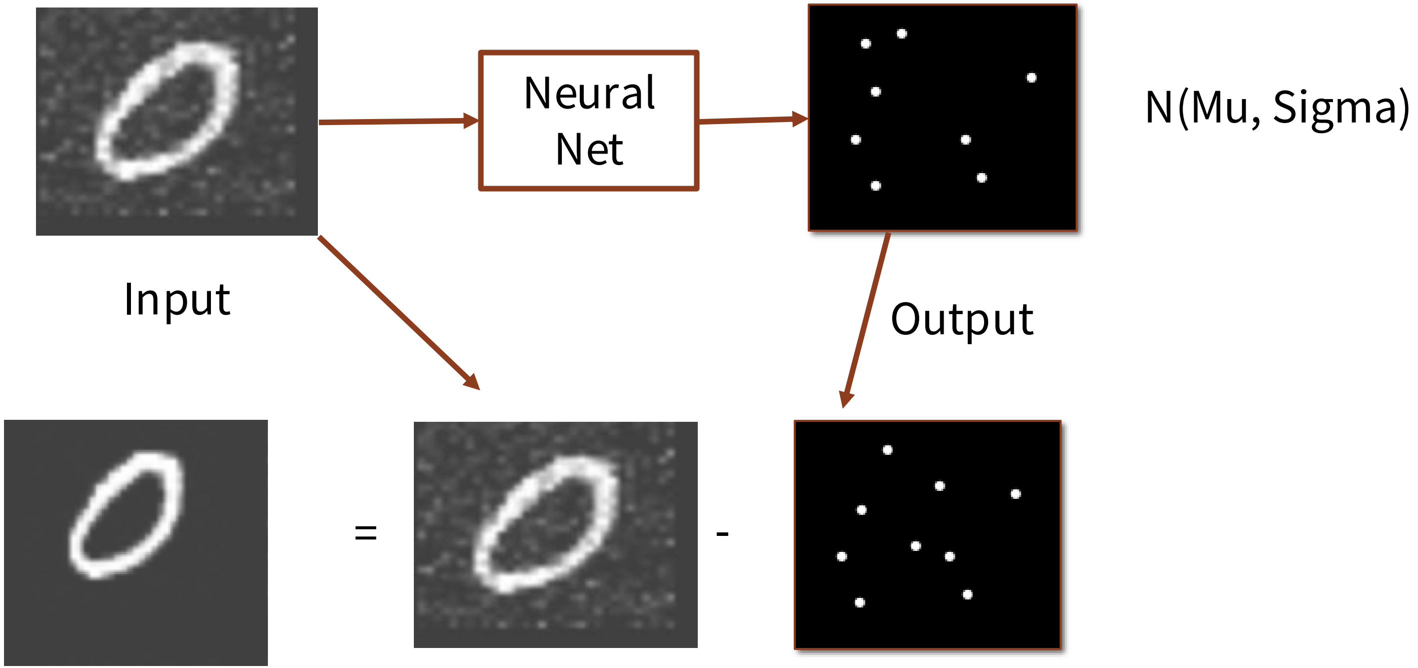


Diffusion Models





Diffusion Models



Introduction

Goal: Learn $p_\theta(x_{t-1}|x_t)$ to approximate $q(x_{t-1}|x_t)$ to produce samples matching the data after finite time

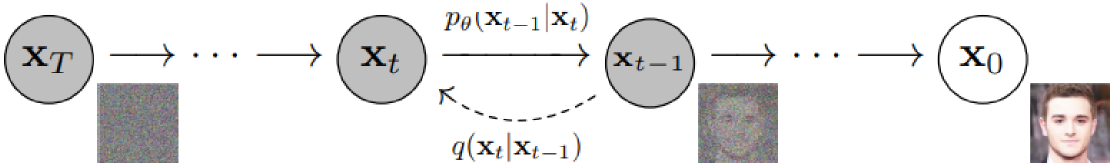
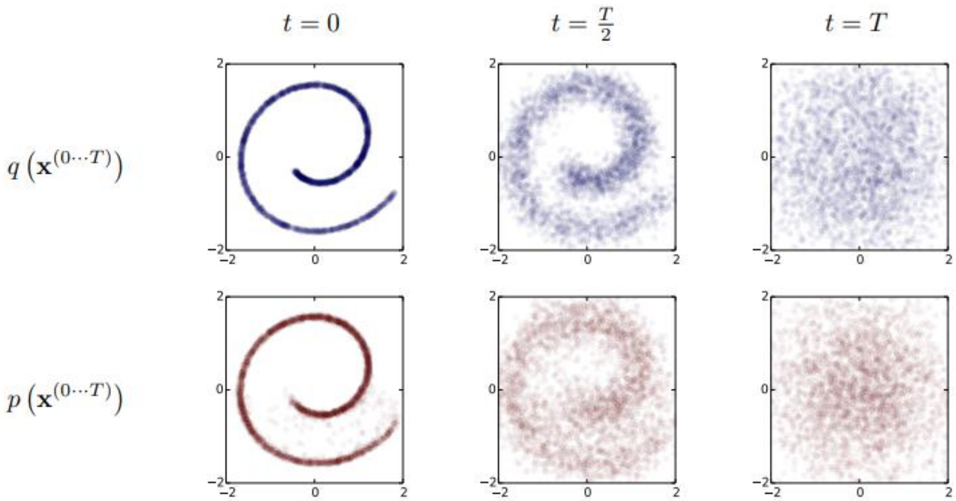


Figure 2: The directed graphical model considered in this work.



Carnegie Mellon University

Denoising Diffusion Models
Learning to generate by denoising

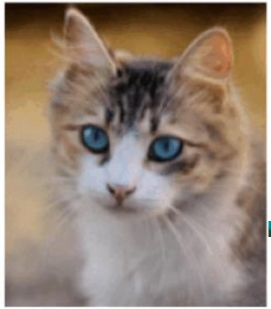
Denoising diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising

Forward diffusion process (fixed)

Data  Noise

Reverse denoising process (generative)



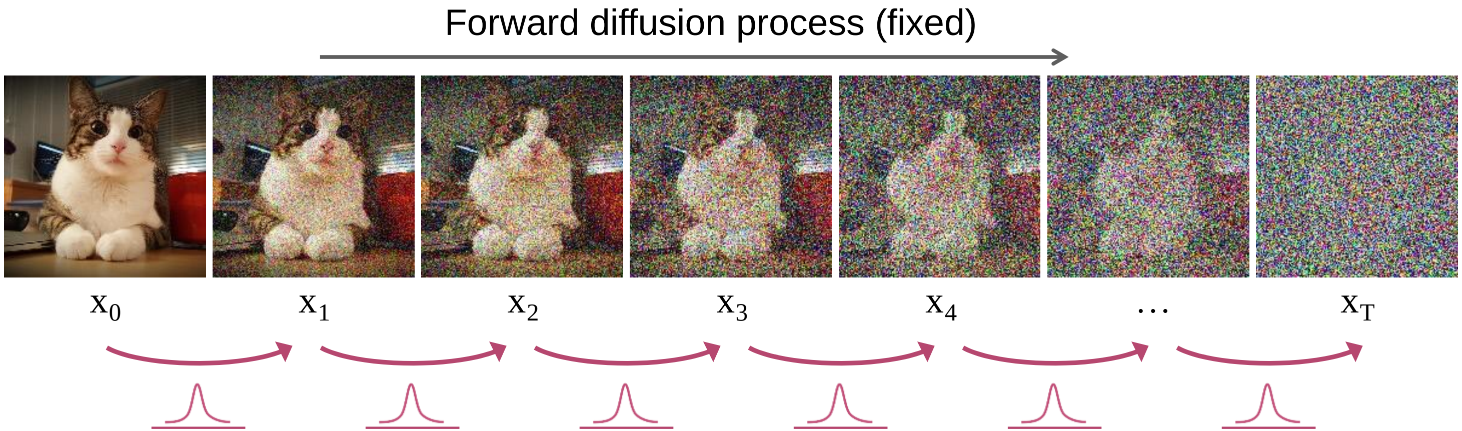
Carnegie Mellon University

Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015 Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

Forward Diffusion Process

The formal definition of the forward process in T steps:

Data



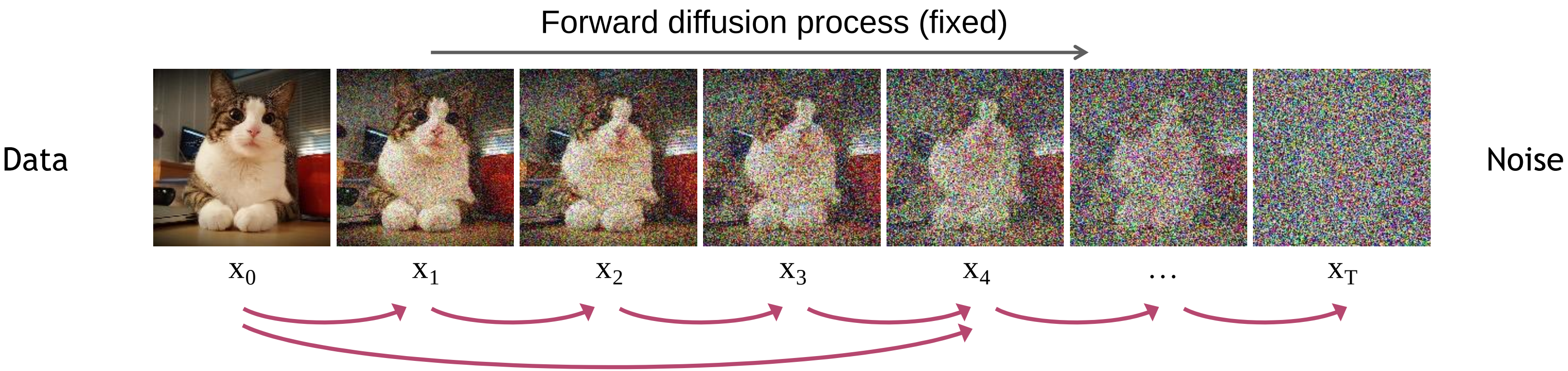
Noise

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

(joint)

Diffusion Kernel

The formal definition of the forward process in T steps:



Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\ \epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Forward Process

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

$$q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, 1-\bar{\alpha}_t I)$$

- It admits sampling x_t at an arbitrary timestep t

x_0 x_{50} x_{100} x_{300} x_{500} x_{700} x_{900} x_{1000}

Variance schedule

β_t linear from 0.0001 ~ 0.02

Timestep

T = 1000

Notation

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s \text{ , } \bar{\alpha_T} \to 0$$

$$\epsilon \sim N(0, 1)$$

Reparameterization

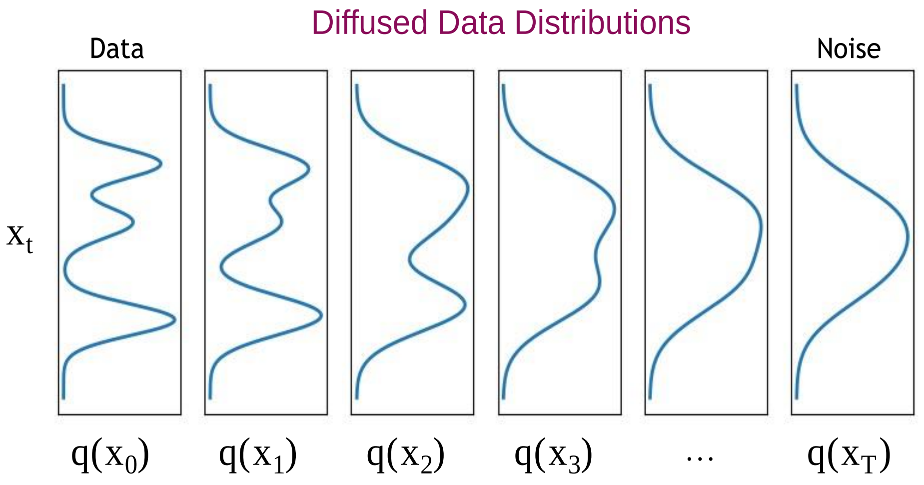
$$N(\mu, \sigma^2) = \mu + \sigma\epsilon$$

Carnegie Mellon University

Distribution in the forward diffusion

$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}}\, d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}}\, \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}}\, d\mathbf{x}_0$$

The diffusion kernel is Gaussian convolution.

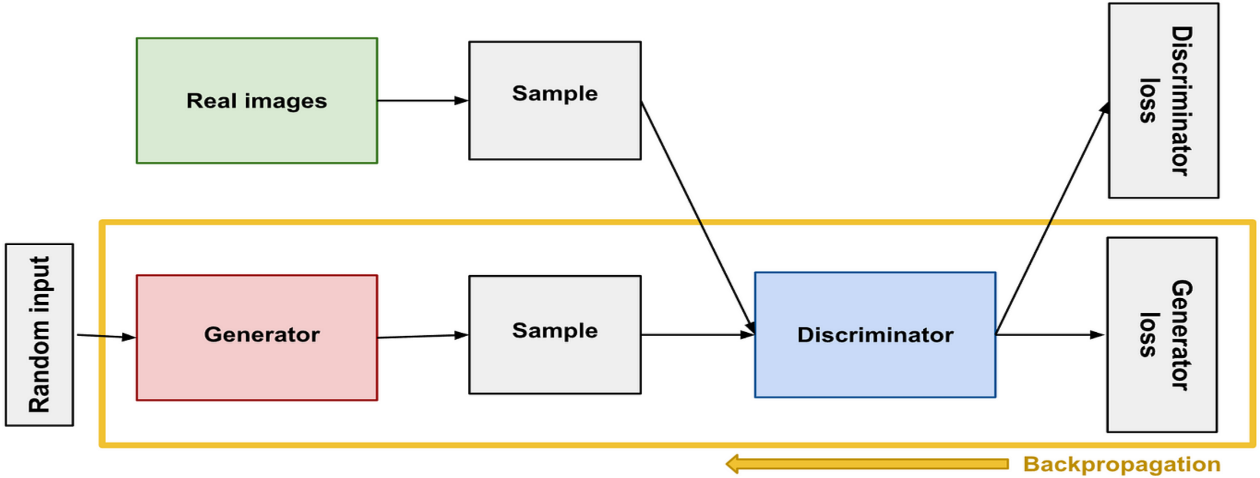


We can sample $\mathbf{x}_t \sim q(\mathbf{x}_t)$ by first sampling $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and then sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ (i.e., ancestral sampling).

Reverse Process

Reverse diffusion process

As you probably figured out, the goal of the reverse diffusion process is to convert pure noise into an image. To do that, we're going to use some neural network (ignore architecture for now, we'll get into it soon). If you're familiar with GANs (Generative Adversarial Networks) (Fig. 6), we're trying to train something similar to the *generator network*. The only difference is that our network will have an easier job because it doesn't have to do all the work in one step.

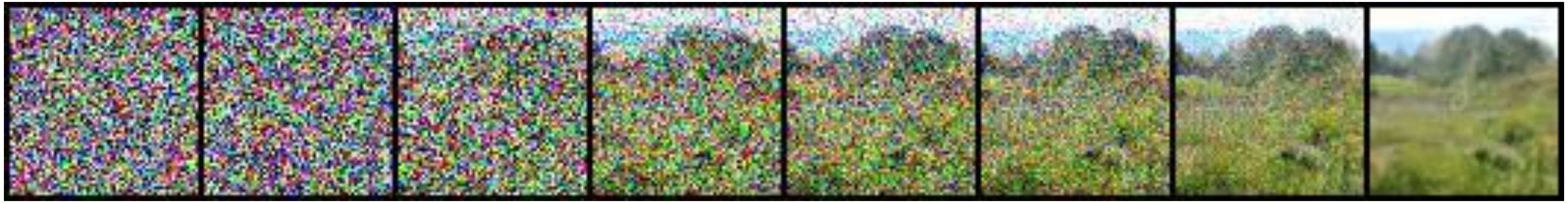


Carnegie Mellon University

Reverse Process

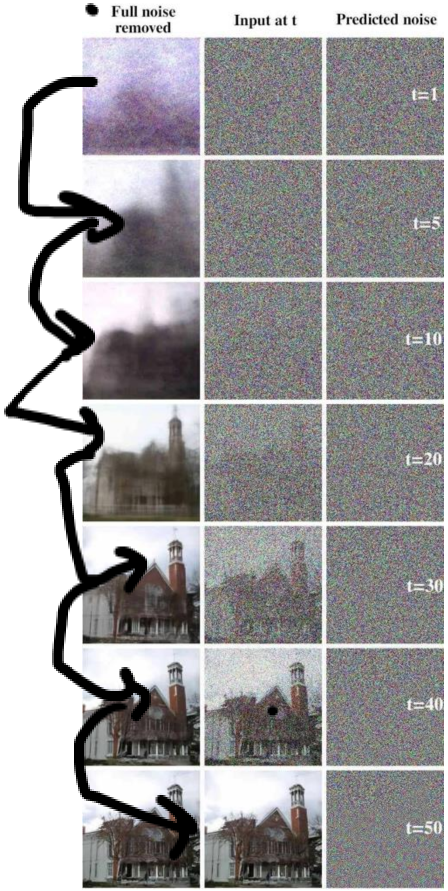
$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$
$$= N(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)), \beta_t)$$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) + \sqrt{\beta_t}\epsilon$$



x_{1000} x_{800} x_{400} x_{200} x_{150} x_{100} x_{50} x_0

Reverse Process



$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$$= N(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)), \beta_t)$$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) + \sqrt{\beta_t}\epsilon$$

Training Objective

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] = \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] =: L$$

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}), \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}) - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon})\right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}),t)\right\|^2\right]$$

$$= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}),t)\right\|^2\right]$$

$$\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

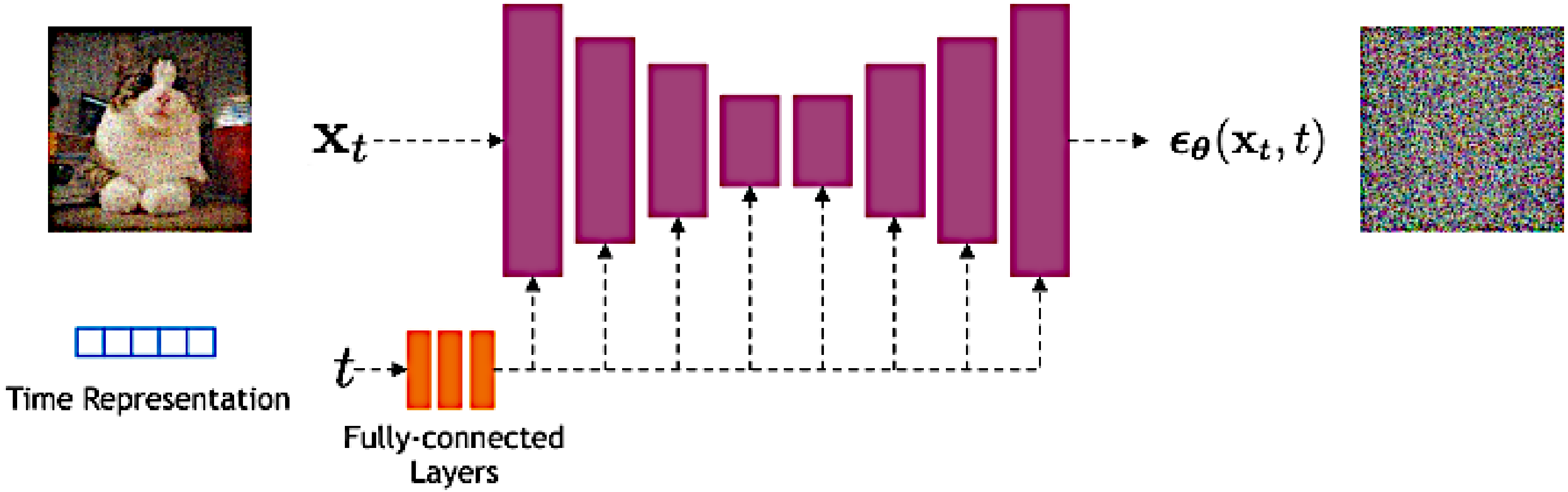
$$L_{\mathrm{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

We can use our model to predict x_0 or μ_θ or ϵ_θ , and the author choose to predict ϵ_θ

Note: $$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$$

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t,t) = \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t))\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right)$$

Network Architecture



- Unet with residual block and self-attention and add time embeding to conditioned with time step

Training and Sampling

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
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** $t = T, \ldots, 1$ **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** $\mathbf{x}_0$

- Forward Process
- Loss Function

- Reverse Process

Experiment

When conditioned on the same latent, CelebA-HQ 256×256 samples share high-level attributes. Bottom-right quadrants are $\mathbf{x}_t$, and other quadrants are samples from $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$.

Carnegie Mellon University

Experiment

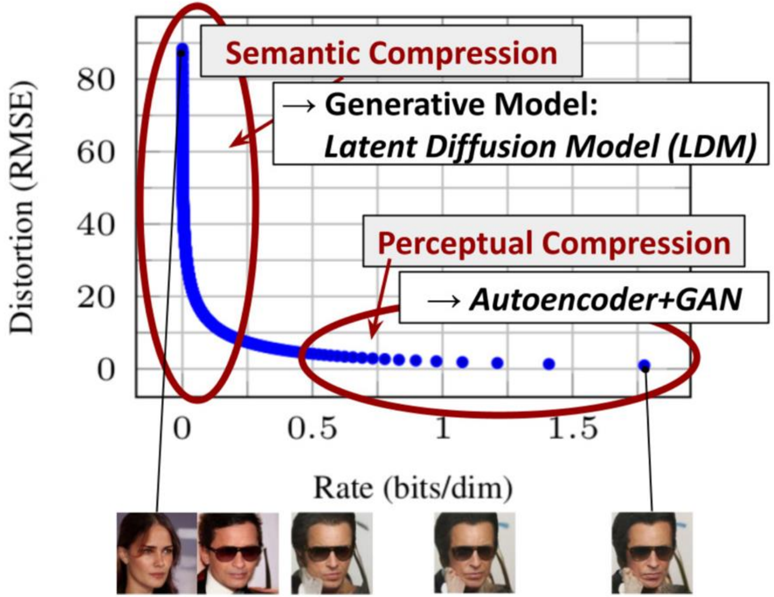
Unconditional CIFAR10 reverse process parameterization and training objective ablation. Blank entries were unstable to train and generated poor samples with out-of-range scores.

Objective	IS	FID
$\tilde{\mu}$ prediction (baseline)		
L, learned diagonal $\mathbf{\Sigma}$	7.28±0.10	23.69
L, fixed isotropic $\mathbf{\Sigma}$	8.06±0.09	13.22
$\lVert\tilde{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}_\theta\rVert^2$	–	–
ϵ prediction (ours)		
L, learned diagonal $\mathbf{\Sigma}$	–	–
L, fixed isotropic $\mathbf{\Sigma}$	7.67±0.13	13.51
$\lVert\tilde{\boldsymbol{\epsilon}} - \boldsymbol{\epsilon}_\theta\rVert^2$ (L_{simple})	**9.46±0.11**	**3.17**

Carnegie Mellon University

Latent Diffusion Model

Latent diffusion model (**LDM**; Rombach & Blattmann, et al. 2022) runs the diffusion process in the latent space instead of pixel space, making training cost lower and inference speed faster. It is motivated by the observation that most bits of an image contribute to perceptual details and the semantic and conceptual composition still remains after aggressive compression. LDM loosely decomposes the perceptual compression and semantic compression with generative modeling learning by first trimming off pixel-level redundancy with autoencoder and then manipulate/generate semantic concepts with diffusion process on learned latent

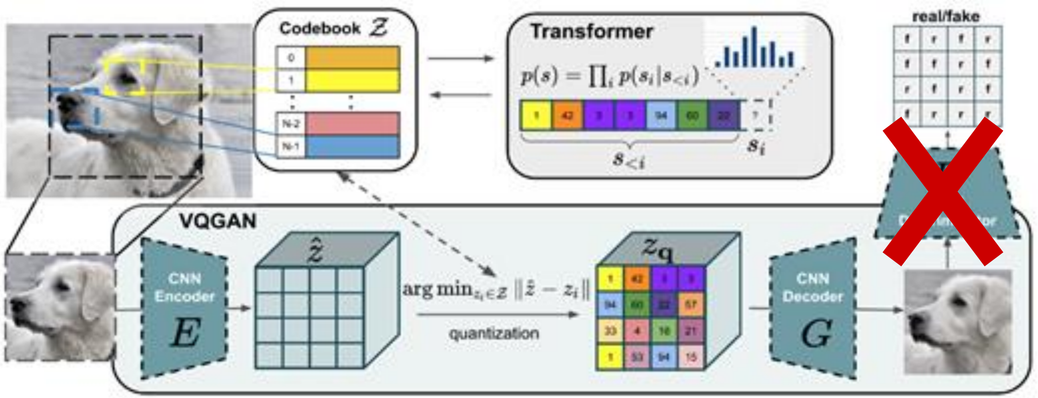


Carnegie Mellon University

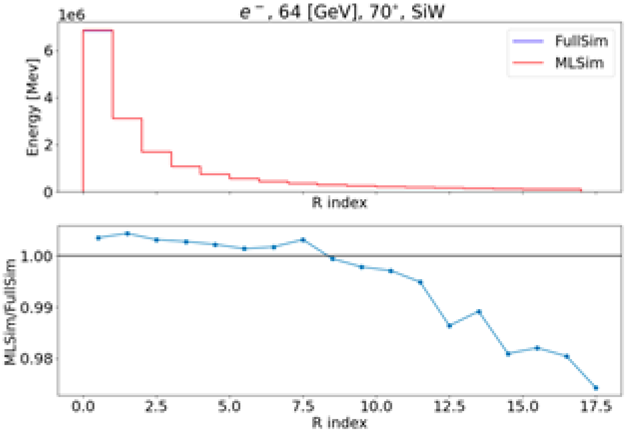
Autoregressive model

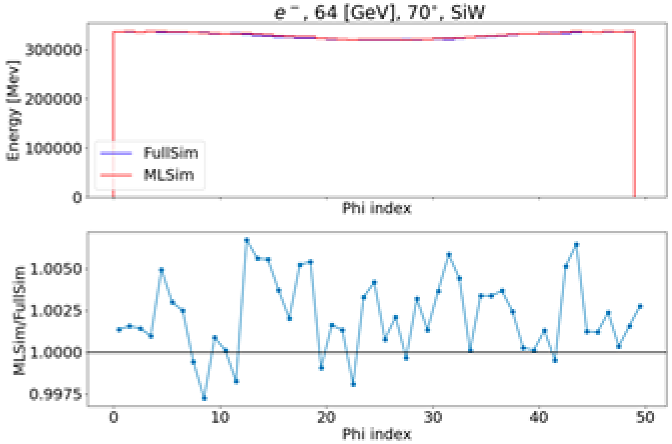
VQ-VAE + Transformers:

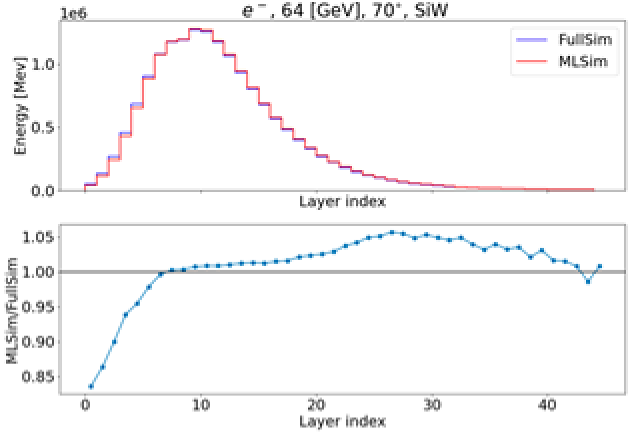
- VQ-VAE to build a codebook (dictionary) of shower features.
- Transformer to predict those codebook vectors (shower features) autoregressively, starting from Layer 0.
  - ❿ VQVAE sees whole shower. Decodes it into 64* tokens.
  - ❿ Transformer sees previous tokens, outputs probabilities over the next one.
- Advantages:
  - ❿ 64 forward passes needed.
  - ❿ Shorter sequence.
- ~10-20 mins per epoch.



Autoregressive model







Carnegie Mellon University

Conditional Image Generation: Guided Diffusion

- There have even been methods that incorporate image embeddings into the diffusion in order to "guide" the generation. Mathematically, guidance refers to conditioning a prior data distribution p(x)$p(\mathbf{x})$ with a condition yy, i.e. the class label or an image/text embedding, resulting in p(x|y)$p(\mathbf{x}|y)$. To turn a diffusion model pθp_θ into a conditional diffusion model, we can add conditioning information yy at each diffusion step. pθ(x0:T|y)=pθ(xT)∏t=1Tpθ(xt−1|xt,y)$p_\theta(\mathbf{x}_{0:T}|y)=p_\theta(\mathbf{x}_T)\prod_{t=1}^{T}p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t,y)$The fact that the conditioning is being seen at each timestep may be a good justification for the excellent samples from a text prompt.
- In general, guided diffusion models aim to learn ∇logpθ(xt|y)$\nabla \log p_\theta(\mathbf{x}_t|y)$. So using the Bayes rule, we can write:∇xtlogpθ(xt|y)=∇xtlog(pθ(y|xt)pθ(xt)pθ(y))=∇xtlogpθ(xt)+∇xtlog(pθ(y|xt))$\nabla_{\mathbf{x}_t}\log p_\theta(\mathbf{x}_t|y)=\nabla_{\mathbf{x}_t}\log\left(\frac{p_\theta(y|\mathbf{x}_t)p_\theta(\mathbf{x}_t)}{p_\theta(y)}\right)=\nabla_{\mathbf{x}_t}\log p_\theta(\mathbf{x}_t)+\nabla_{\mathbf{x}_t}\log(p_\theta(y|\mathbf{x}_t))$
- $p_\theta(y)$ is removed since the gradient operator ∇xt$\nabla_{\mathbf{x}_t}$ refers only to xt$\mathbf{x}_t$, so no gradient for yy. Moreover remember that log(ab)=log(a)+log(b)$\log(ab)=\log(a)+\log(b)$.And by adding a guidance scalar term ss, we have:∇logpθ(xt|y)=∇logpθ(xt)+s·∇log(pθ(y|xt))$\nabla\log p_\theta(\mathbf{x}_t|y)=\nabla\log p_\theta(\mathbf{x}_t)+s\cdot\nabla\log(p_\theta(y|\mathbf{x}_t))$Using this formulation, let's make a distinction between classifier and classifier-free guidance. Next, we will present two family of methods aiming at injecting label information.

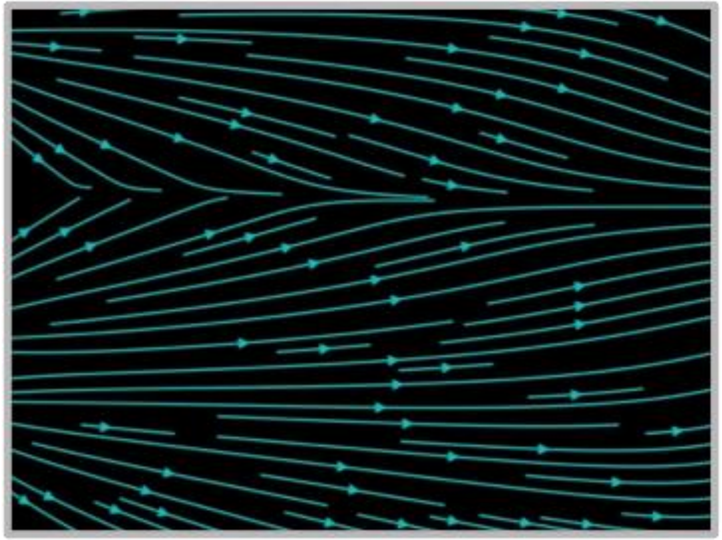
Carnegie Mellon University

Diffusion Process as a differential Equation

Crash Course in Differential Equations

Ordinary Differential Equation (ODE): dx /dt

x = f(x,t) or dx f(x,t)dt

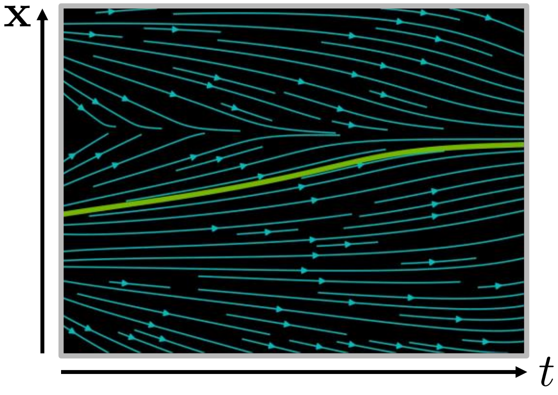


Carnegie Mellon University

Diffusion Process as a differential Equation

Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



Analytical Solution:

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$$

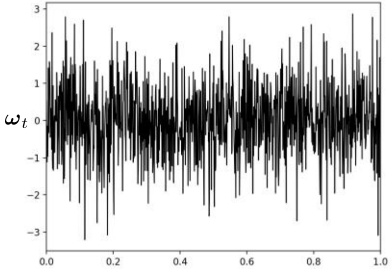
Iterative Numerical Solution:

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$$

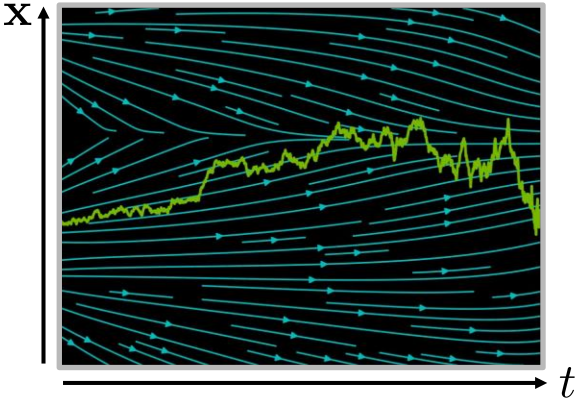
Stochastic Differential Equation (SDE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{drift coefficient}} + \underbrace{\sigma(\mathbf{x}, t)}_{\text{diffusion coefficient}}\boldsymbol{\omega}_t$$

Wiener Process (Gaussian White Noise)

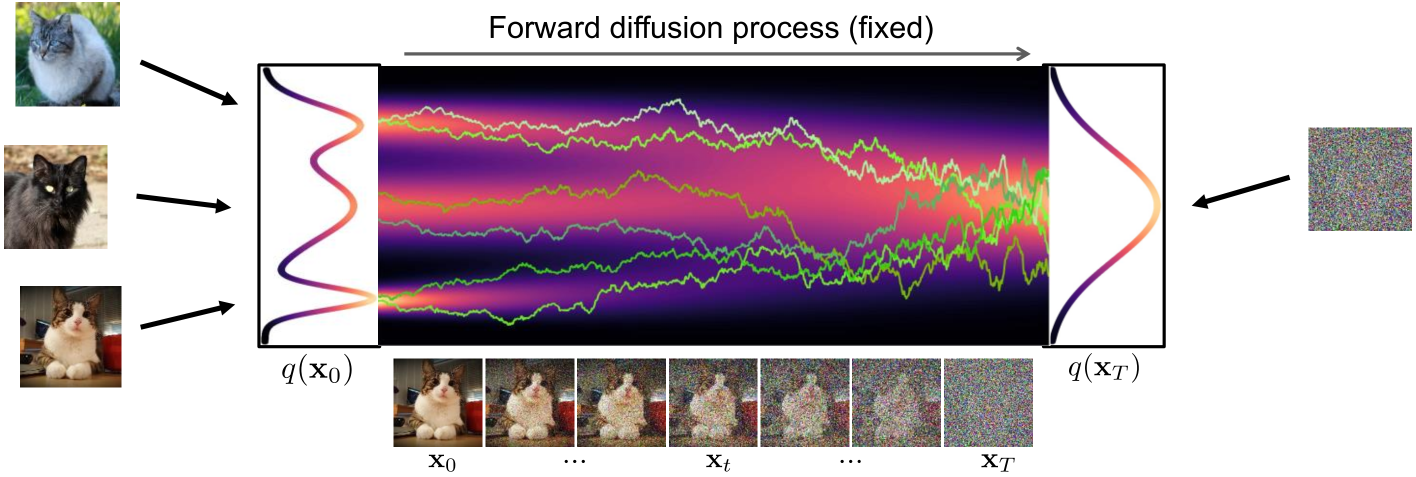


$$\left(\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \sigma(\mathbf{x}, t)\mathrm{d}\boldsymbol{\omega}_t \right)$$



$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

Forward Diffusion Process as Stochastic Differential Equation



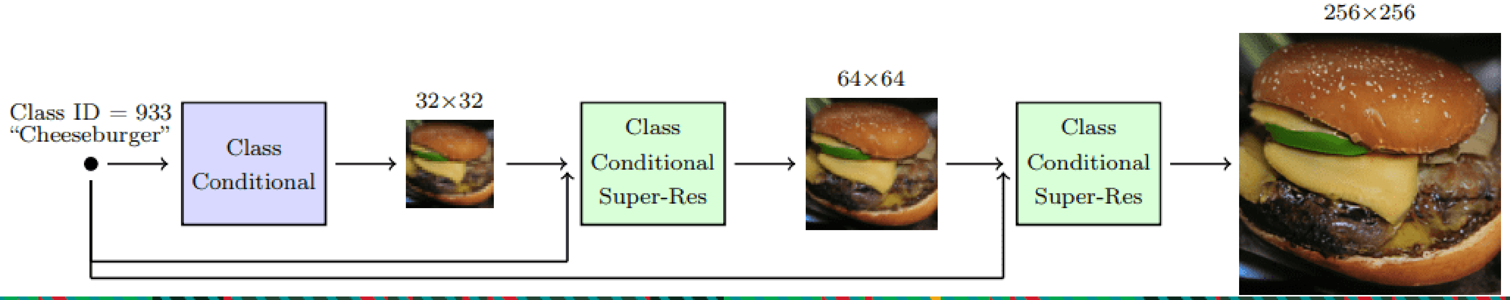
Forward Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\substack{\text{drift term}\\ \text{(pulls towards mode)}}} + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t}_{\substack{\text{diffusion term}\\ \text{(injects noise)}}}$$

Song et al,. ICLR, 2021

Scaling up diffusion

- You might be asking what is the problem with these models. Well, it's computationally very expensive to scale these U-nets into high-resolution images. This brings us to two methods for scaling up diffusion models to higher resolutions: cascade diffusion models and latent diffusion models.
- **Cascade diffusion models**
- Ho et al. 2021 introduced cascade diffusion models in an effort to produce high-fidelity images. A cascade diffusion model consists of a pipeline of many sequential diffusion models that generate images of increasing resolution. Each model generates a sample with superior quality than the previous one by successively upsampling the image and adding higher resolution details. To generate an image, we sample sequentially from each diffusion model.



Carnegie Mellon University

Conclusion

- Achieved high quality image generation without adversarial training
- Aquire simple training objective , Reverse process predict ε instead of μ
- Many follow-up research Improve diffusion base model based on this architecture