CPE 490 590: Machine Learning for Engineering Applications

16 Diffusion Models

Rahul Bhadani
Electrical & Computer Engineering, The University of Alabama in Huntsville

Outline

1. Diffusion Models

2. Mathematical Framework for Diffusion Models

3. Latent Diffusion Models

Motivation

Diffusion Model as the State of the Art for Generative Models

Figures created using Midjourney v4 and GPT-4.

Diffusion Model are non-Adversarial

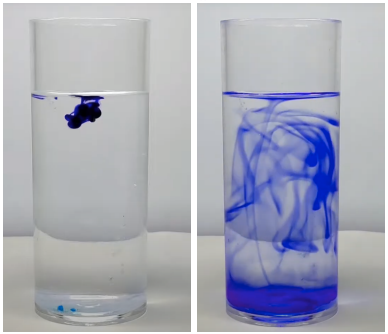
Adversarial Models such as GAN are difficult to train.

Discriminator can be too good!!

Diffusion Models

Inspiration behind Diffusion Model

Inspired by non-equilibrium behavior in Thermodynamics



The drop of ink eventually diffuses into the water until it reaches equilibrium.

Intuition behind Diffusion Model

In the physical world, the reversal of the diffusion process is not possible. But that's what the diffusion model is trying to do.

Intuition behind Diffusion Model

In the ink-water diffusion example, the drop of the ink at one spot includes some information.

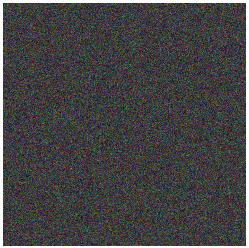
As the diffusion process progresses, we lose information, and eventually, we only have noise.

Intuition behind Diffusion Model

In the image domain:

Clear information

Some loss of information



Just noise

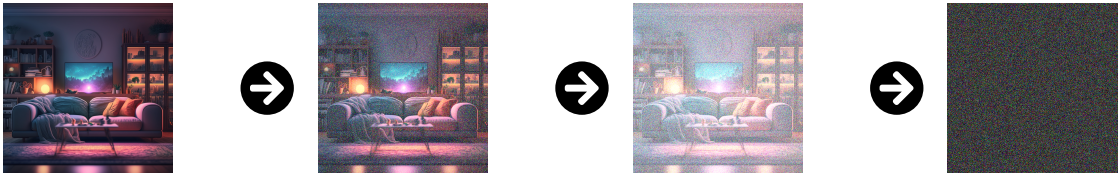
Working backward from just noise to a proper image is what a diffusion model tries to achieve.

The Working Principle of the Diffusion Model

Add noise to the original image and then later learn how to reverse this noise process.

The Working Principle of the Diffusion Model

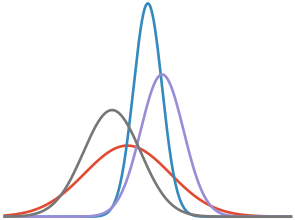
Apply noise following a Markov chain/Markov Process.



At each time step, we add a little bit of noise to the image until the image only consists of the noise. We have 100-1000 long Markov chain.

Noise in the Diffusion Model

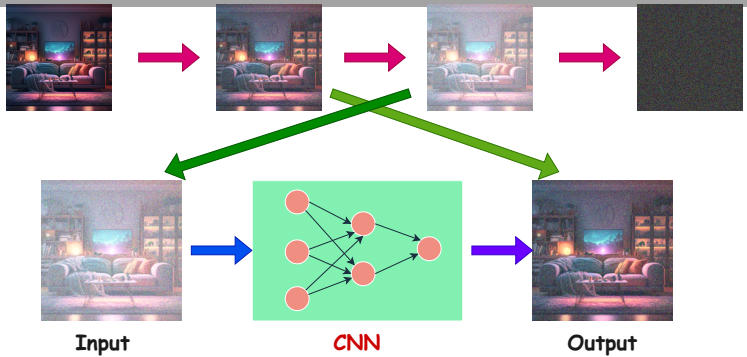
Gaussian Noise: $\mathcal{N}(\mu, \sigma^2)$



$$\begin{bmatrix} 25 \\ 67 \end{bmatrix} + \mathcal{N}(\mu, \sigma^2) = \begin{bmatrix} 26.63 \\ 66.55 \end{bmatrix}$$

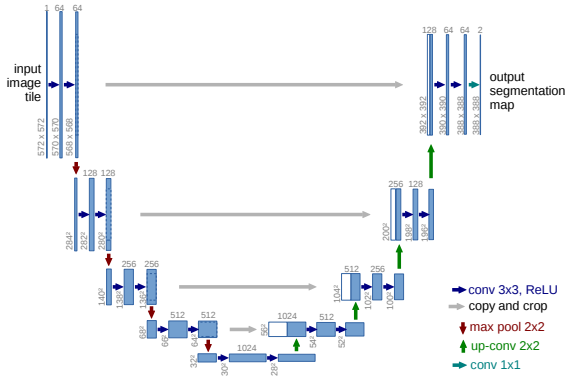
Repeat the process.

Idea Behind Removing the Noise (Working Backward)



The type of CNN used in the original paper is called UNet.

UNet Architecture



Source: https://arxiv.org/pdf/1505.04597.pdf

UNet Architecture for Diffusion

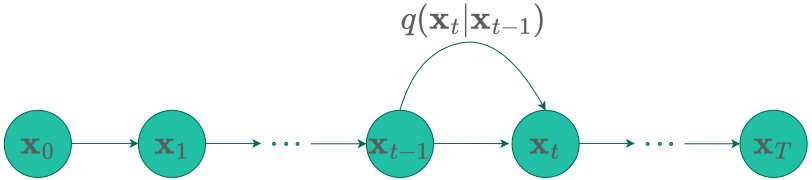
- U-net consists of a series of up/down sampling convolution blocks which are connected via residual connection, along with a middle block. This set up allow allows one to global location and context at the same time in the learning process. It works with very few training samples and provides better performance for segmentation tasks.
- In the Diffusion model, UNet architecture is combined with time-step embedding called sinusoidal embedding. This is similar to the positional embedding seen in the transformer. This block combines sinusoidal embedding and convolution layers to learn the time step of the noisy images in the diffusion process.

Mathematical Framework for Diffusion Models

Formulating Diffusion Model: Forward Process

- Consider an original image data sampled from the real distribution $\mathbf{x}_0 \sim q(\mathbf{x})$.
- We add small Gaussian noise to the sample data in T steps, producing a sequence of noisy samples $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$.
- The step sizes are controlled by the diffusion rate $\{\beta_t \in (0, 1)\}$, a hyperparameter.
- The data sample $\mathbf{x}_0$ slowly loses its distinguishable features as the step t becomes larger.

Forward Process: Adding Noise



- If the noise to be added is Gaussian, then we use the analytical distribution (also known as **transition kernel**) of the noisy data at time step t as $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$. **Note: this is a design choice.**
- The conditional distribution of the noisy data for the subsequent Markov chain (i.e. joint distribution of $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$) is written as

$$q(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T|\mathbf{x}_0) = \prod_{t=0}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

.

Forward Process: Adding Noise

We sample $\mathbf{x}_t$ at any arbitrary time step t in a closed form using reparameterization trick

$$\alpha_1 = 1 - \beta_t, \quad \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i.$$

This can help us write analytical form of $q(\mathbf{x}_t|\mathbf{x}_0)$ for all $t \in \{0, 1, \cdots, T\}$ as

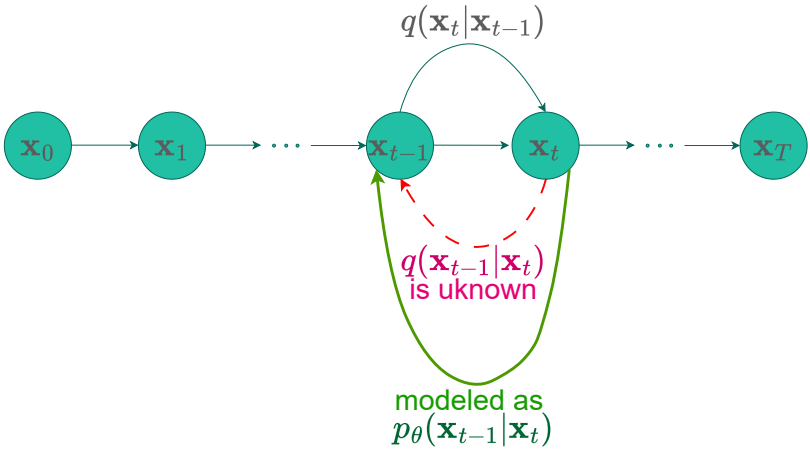
$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I}).$$

- Given $\mathbf{x}_0$, we can sample $\mathbf{x}_t$, by supplying small Gaussian perturbation $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$ and applying a transformation

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{(1 - \bar{\alpha}_t)}\varepsilon.$$

Intuitively speaking, this forward process slowly injects noise into data until all structures are lost.

Reverse Diffusion Process: Removing Noise



Reverse Diffusion Process: Removing Noise

- We need to learn a model p_θ to approximate these conditional probabilities $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to run the reverse diffusion process.
- We start with choosing the prior distribution $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; 0, \mathbf{I})$ (because in forward process, we end up with $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; 0, \mathbf{I})$.
- The learnable transition kernel $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ can be obtained using a neural network model.
- The learnable transition kernel can be mathematically written as a parametrized equation

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)).$$

  μ_θ and Σ_θ are parameters to be learned using a neural network (UNet).
- With this reverse Markov chain in hand, we can generate a data sample $\mathbf{x}_0$ by first sampling a noise vector $\mathbf{x}_T \sim p(\mathbf{x}_T)$, then iteratively sampling from the learnable transition kernel $x_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ until $t = 1$.

What about the Loss Function?

- The primary objective in training the UNet is the reverse Markov chain to match the actual time reversal of the forward Markov chain.
- We need to adjust the parameter θ in such a way so that joint probability distribution of the reverse Markov chain $p_\theta(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T) = p(\mathbf{x}_T) \prod_{i=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ should closely approximate of the forward process $q_\theta(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T) = q(\mathbf{x}_0) \prod_{i=1}^{T} q_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$.
- This is done by minimizing the distance between these two joint probability distributions using a metric called **Kullback-Leibler (KL) divergence**.
- KL-divergence.

KL Divergence as Loss Function

- KL divergence for training the diffusion model is written as

$$\begin{aligned}
\mathrm{KL}(q,p) &= -\mathbb{E}_q(\mathbf{x}_0,\mathbf{x}_1,\cdots,\mathbf{x}_T)[\log p_\theta(\mathbf{x}_0,\mathbf{x}_1,\cdots,\mathbf{x}_T)] + \text{constant} \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t=1}^{T}\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] + \text{constant} \qquad (1) \\
&\geq \mathbb{E}[-\log p_\theta(\mathbf{x}_0)] + \text{constant}
\end{aligned}$$

The last inequality comes from Jensens's Inequality, i.e.

$$\mathbb{E}[g(X)] \geq g(\mathbb{E}[X])$$

Latent Diffusion Models

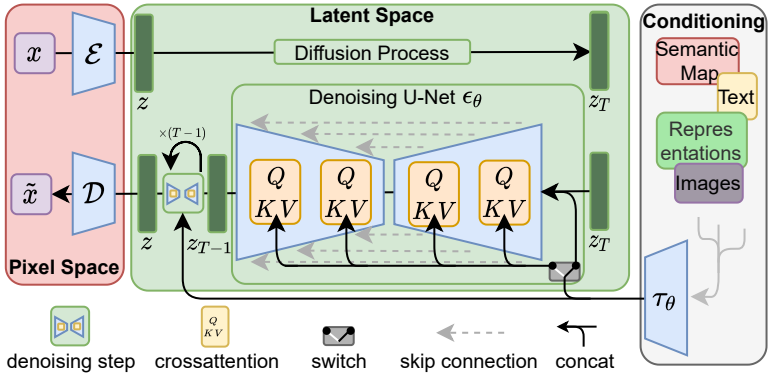
Latent Diffusion Model

The diffusion model runs in a latent space instead of a pixel space, making training costs lower and inference faster.

Latent: hidden or compression space (like PCA)!

One example of the Latent Diffusion Model is StableDiffusion (the one that lets you generate images from the text).

Latent Diffusion Model Architecture



Code Demo

UNet Diffusion Model on Fashion MNIST Dataset:
https://github.com/rahulbhadani/CPE490_590_Sp2025/blob/master/Code/Chapter_16_DiffusionModel.ipynb

UNet Diffusion Model on Digit MNIST Dataset:
https://colab.research.google.com/drive/1RqgzkjEL-W3lUOhUkzAoEcVlnM0Cu1Fp?usp=sharing

References

- Original Paper on Diffusion Model, Deep Unsupervised Learning using Nonequilibrium Thermodynamics: https://proceedings.mlr.press/v37/sohl-dickstein15.pdf
- Deep Learning Notes by Prof. LI Hongsheng: https://dl.ee.cuhk.edu.hk/slides/diffusion.pdf
- Diffusion Models: A Comprehensive Survey of Methods and Applications: https://dl.acm.org/doi/full/10.1145/3626235?casa_token=tLriJS0Ll7EAAAAA:G_ue_8JwODm_PGRul_bBdK70QpTLetomFM1ioVMzebrqCaAtiozqPpcs2hV62SMAUIz01-TvweBV
- Understanding Diffusion Models: A Unified Perspective: https://arxiv.org/pdf/2208.11970.pdf
- U-Net: Convolutional Networks for Biomedical Image Segmentation: https://arxiv.org/pdf/1505.04597.pdf
- ConvNext: A ConvNet for the 2020s: https://arxiv.org/pdf/2201.03545.pdf
- ACC-UNet: A Completely Convolutional UNet model for the 2020s: https://arxiv.org/pdf/2308.13680.pdf
- Latent Diffusion Paper: https://arxiv.org/pdf/2112.10752.pdf

References

- Stable Diffusion Github Repo: https://github.com/CompVis/stable-diffusion?tab=readme-ov-file
- Diffusion Models Beat GANs on Image Synthesis (Guided Diffusion): https://arxiv.org/pdf/2105.05233.pdf
- Guided Diffusion Github Repo: https://github.com/openai/guided-diffusion

The End