

Text-to-Image Generation

Msc Laboratory Experiment

September 21, 2023

1 Aims:

The main aim of this experiment is to give the students the concepts of deep generative model (diffusion model) and foundation natural language processing model (transformer). Moreover, it will give them the idea of probability and statistics, stochastic process, convolutional neural network, machine learning, and computer simulation of deep learning. All the experiments will be performed using PyCharm or Jupyter Notebook.

2 Introduction:

Deep generative model (diffusion model or known as diffusion probabilistic model) is one of the key enabling technologies of deep learning, often typically refers to a specific type of probabilistic model used in machine learning and artificial intelligence. It is a powerful framework for generative modelling, particularly in image and data generation tasks. Technically, diffusion models are inspired by non-equilibrium thermodynamics, which focus on the process of generating data by iteratively refining a random noise source as opposed to the previously held belief that noise was a troublemaker. It was defined a Markov chain of diffusion steps slowly add random noise to data and then learn to reverse the diffusion process to construct desired data samples from noise. Hence, high-fidelity generation of images and other data could be obtained over the training, thereby decreasing the communication noise.

3 Model Architecture

Fig.1 shows the architecture of latent diffusion model whose diffusion and denoising processes happen on the latent vector \mathbf{z} . The denoising model is a time-conditioned U-Net, augmented with the cross-attention mechanism to handle flexible conditioning information for image generation. Each type of conditioning information is paired with a domain-specific encoder τ_θ to project the conditioning input \mathbf{y} to an intermediate representation that can be mapped into cross-attention component, $\tau_\theta(\mathbf{y}) \in \mathbb{R}^{M \times d_\tau}$:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right) \cdot \mathbf{V} \quad (1)$$

$$\text{where } \mathbf{Q} = \mathbf{W}_Q^{(i)} \cdot \varphi_i(\mathbf{z}_i), \mathbf{K} = \mathbf{W}_K^{(i)} \cdot \tau_\theta(y), \mathbf{V} = \mathbf{W}_V^{(i)} \cdot \tau_\theta(y) \quad (2)$$

$$\text{and } \mathbf{W}_Q^{(i)} \in \mathbb{R}^{d \times d_\epsilon^i}, \mathbf{W}_K^{(i)}, \mathbf{W}_V^{(i)} \in \mathbb{R}^{d \times d_\tau}, \varphi_i(\mathbf{z}_i) \in \mathbb{R}^{N \times d_\epsilon^i}, \tau_\theta(y) \in \mathbb{R}^{M \times d_\tau} \quad (3)$$

4 Forward (Reverse) Diffusion Process

Fig. 2 shows the Markov chain of diffusion steps to slowly add random noise to data and then learn to reverse the diffusion process to construct desired data samples from the noise. Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim \mathbf{q}(\mathbf{x})$, the forward diffusion process in which we add small

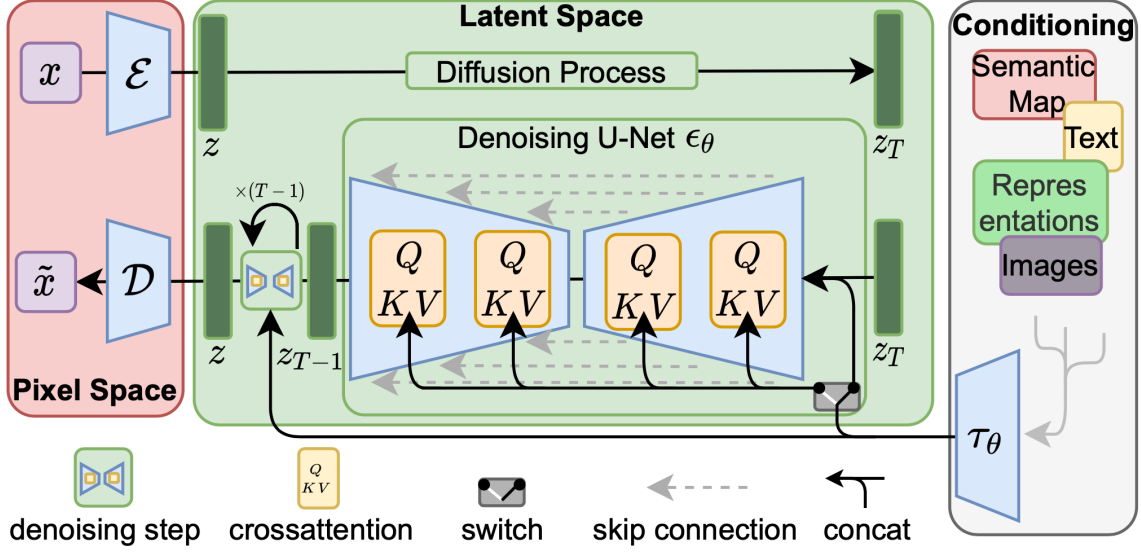


Figure 1: One kind of diffusion model architecture.

amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $\mathbf{x}_1, \dots, \mathbf{x}_T$. The step size are controlled by a variance schedule $\{\beta_t \in (0, 1)\}_{t=1}^T$.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad (4)$$

The data sample \mathbf{x}_0 gradually loses its distinguishable features as the step t becomes larger. Eventually when $T \rightarrow \infty$, \mathbf{x}_t is equivalent to an isotropic Gaussian distribution.

A nice property of the above process is that we can sample \mathbf{x}_t at any arbitrary time step t in a closed form using reparameterization trick. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$:

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_{t-1} && \text{;where } \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \dots \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1 - \alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} && \text{;where } \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians (*).} \\ &= \dots \\ &= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon} \\ q(\mathbf{x}_t|\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I}) \end{aligned} \quad (5)$$

Usually, we can afford a larger update step when the sample gets noisier, so $\beta_1 < \beta_2 < \dots < \beta_T$ and therefore $\bar{\alpha}_1 > \dots > \bar{\alpha}_T$.

If we can reverse the above process and sample from $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we will be able to recreate the true sample from a Gaussian noise input, $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that if β_t is small enough, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ will also be Gaussian. Unfortunately, we cannot easily estimate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ because it needs to use the entire dataset and therefore we need to learn a model p_θ to approximate these conditional probabilities in order to run the reverse diffusion process.

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \quad (6)$$

It is noteworthy that the reverse conditional probability is tractable when conditioned on \mathbf{x}_0 :

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\boldsymbol{\beta}}_t\mathbf{I}) \quad (7)$$

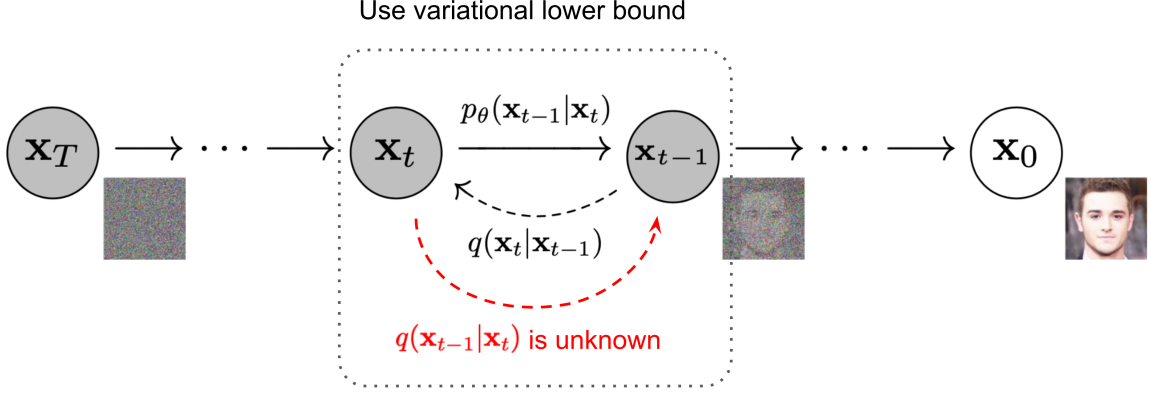


Figure 2: The Markov chain of forward (reverse) diffusion process of generating a sample by slowly adding (removing) noise..

Using Bayes' rule, we have:

$$\begin{aligned}
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{1 - \bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1 - \bar{\alpha}_t}\right)\right) \\
&= \exp\left(-\frac{1}{2}\left(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t}\mathbf{x}_t\mathbf{x}_{t-1} + \alpha_t\mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\mathbf{x}_{t-1} + \bar{\alpha}_{t-1}\mathbf{x}_0^2}{1 - \bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1 - \bar{\alpha}_t}\right)\right) \\
&= \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}}\mathbf{x}_0\right)\mathbf{x}_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0)\right)\right)
\end{aligned} \tag{8}$$

where $C(\mathbf{x}_t, \mathbf{x}_0)$ is some function not involving \mathbf{x}_{t-1} , and details are omitted. Following the standard Gaussian density function, the mean and variance can be parameterized as follows (recall that $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$):

$$\begin{aligned}
\tilde{\beta}_t &= 1/\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}}\right) = 1/\left(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t(1 - \bar{\alpha}_{t-1})}\right) = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t \\
\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) &= \left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}}\mathbf{x}_0\right) / \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}}\right) \\
&= \left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}}\mathbf{x}_0\right) \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t \\
&= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}\mathbf{x}_0
\end{aligned} \tag{9}$$

We can represent $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}_t)$ and plug it into the above equation and obtain:

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_t &= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}_t) \\
&= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)
\end{aligned} \tag{10}$$

5 Loss Function:

Recall that we need to learn a neural network to approximate the conditioned probability distributions in the reverse diffusion process, $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$. We would like to train $\boldsymbol{\mu}_\theta$ to predict $\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)$. Because \mathbf{x}_t is available as input at training time, we can

reparameterize the Gaussian noise term instead to make it predict ϵ_t from the input \mathbf{x}_t at time step t :

$$\begin{aligned}\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) &= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) \\ \text{Thus } \mathbf{x}_{t-1} &= \mathcal{N}(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))\end{aligned}\tag{11}$$

The loss term L_t is parameterized to minimize the difference from $\tilde{\boldsymbol{\mu}}$:

$$\begin{aligned}L_t &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2 \|\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\|_2^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|_2^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2 \|\boldsymbol{\Sigma}_\theta\|_2^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_t \right) - \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) \right\|_2^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{(1 - \alpha_t)^2}{2 \alpha_t (1 - \bar{\alpha}_t) \|\boldsymbol{\Sigma}_\theta\|_2^2} \|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|_2^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{(1 - \alpha_t)^2}{2 \alpha_t (1 - \bar{\alpha}_t) \|\boldsymbol{\Sigma}_\theta\|_2^2} \|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_t, t)\|_2^2 \right]\end{aligned}\tag{12}$$

Empirically, Ho et al. (2020) found that training the diffusion model works better with a simplified objective that ignores the weighting term:

$$\begin{aligned}L_t^{\text{simple}} &= \mathbb{E}_{t \sim [1, T], \mathbf{x}_0, \epsilon_t} \left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|_2^2 \right] \\ &= \mathbb{E}_{t \sim [1, T], \mathbf{x}_0, \epsilon_t} \left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_t, t)\|_2^2 \right]\end{aligned}\tag{13}$$

6 Exercise

Exercise 1: Due to time constraints, it is challenging to fully develop our own text-to-image model code. Therefore, the code for the Latent Diffusion model based on the training set has already been provided, and it is needed to write your own testing set code for this model. Additionally, it is aimed to bring the training model’s loss within an acceptable range.

Note: In deep learning, different models have different metrics for evaluating loss values. Therefore, here, we consider the loss converging as an acceptable range.

Exercise 2: With the aim of assessing the performance of the final model, we need to generate images for the following six specific texts:

- a drawing of a star with a jewel in the center,
- a drawing of a woman in a red cape.
- a drawing of a dragon sitting on its hind legs
- a drawing of a blue sea turtle holding a rock
- a blue and white bird with its wings spread
- a blue and white stuffed animal sitting on top of a white surface

Exercise 3: Let’s take a moment to reflect on what areas can still be further improved in the current latent diffusion model or text-to-image model, such as aspects related to the quality and speed of generating images.

References

- [1] Jonathan Ho et al, “Denoising diffusion probabilistic models.,” *arxiv Preprint arxiv:2105.05233*, 2021.
- [2] Rombach & Blattmann, et al., “High-Resolution Image Synthesis with Latent Diffusion Models.,” *CVPR*, 2022.



Figure 3: The examples of the generating images for the latent diffusion model in Pokemon style.