Meeting Note

Author: Henry Lin
Date: 2024/08/26 – 2024/08/30

Weekly Summary
Denoising Diffusion Probabilistic Models (DDPM) [1].

Plan for Next Week
Chain-of-Thought Prompting Elicits Reasoning in Large Language Models (arXiv: 2201.11903)

Details
Before the diffusion model, there are many other types of generative model such as generative adversarial networks (GANs) [4], variational autoencoders (VAEs) [5], and flows [6, 7], but models like GAN are not stable since there are two networks have to be trained. The authors proposed a non-parametric diffusion model to make the (pure) noised image nearly irrelated to the original image and a parametric model to do the denoising process [Re 1].

The first part is the diffusion process (forward process) q which simply and gradually adds Gaussian noise to the data according to the variance schedule β_1, … β_T (see code forward_process.py):

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

where $\mathcal{N}(\cdot)$ denotes Gaussian distribution (normal distribution), $\sqrt{1-\beta_t}\mathbf{x}_{t-1}$ denotes the mean of Gaussian distribution, and $\beta_t\mathbf{I}$ denotes covariance matrix of Gaussian distribution where $\mathbf{I}$ is the identity matrix (in addition, the authors fix β_1, … β_T from 0.0004 to 0.02 instead of using a learnable β_t.) And we can simplify the equation as:

$$\mathbf{x_t} = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon; \text{ where } \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

Furthermore, if we want to get $\mathbf{x}_t$ from $\mathbf{x}_0$ directly, the iteration is too costly, so we assume $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, then we can get the equation to obtain $\mathbf{x}_t$ from $\mathbf{x}_0$ for any t as follow:

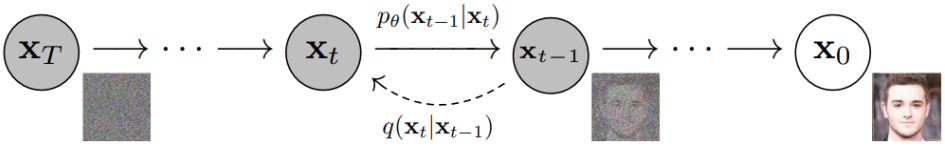

Figure 1. The diffusion process q and denoising process p_θ.

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

And if we process the original input $\mathbf{x}_0$ forward for T steps as follow:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

we will obtain $\mathbf{x}_T$ which is approximately equal to normal distribution $\mathcal{N}(0, \mathbf{I})$ (i.e. the pure noise) [Re 2].

To obtain $\mathbf{x}_{T-1}$ from $\mathbf{x}_T$ (we will use it later), we can use the equation which is called forward process posterior:

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

where $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$.

The second part is the reverse process (denoising process) p_θ which attempts to reverse the noised data from $\mathbf{x}_t$ to $\mathbf{x}_{t-1}$ by removing the noise. It means that the goal of the reverse process in diffusion model is to predict the noise component in $\mathbf{x}_t$, and subtract the predicted noise from $\mathbf{x}_t$ to obtain $\mathbf{x}_{t-1}$. The p_θ can be written as follow (θ means that the parameters of the distribution of the reverse process is learned using neural network):

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

where $\mu_\theta(\mathbf{x}_t, t)$ denotes the mean, and $\sum_\theta(\mathbf{x}_t, t)$ denotes the covariance matrix which is set to $\sigma_t^2\mathbf{I}$ as untrained time dependent constant. For the variable σ_t^2, the authors also tried both $\sigma_t^2 = \beta_t$ and $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$, but got the similar results. And we can simplify the equation as:

$$\bar{\mathbf{x}}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \sqrt{\sum_\theta(\mathbf{x}_t, t)}\epsilon$$

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** $\mathbf{x}_0$

And if we process the noised image $\mathbf{x}_T$ reservedly for T steps, that is, a Markov chain with learned Gaussian transition starts with $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; 0, \mathbf{I})$:

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

we will obtain $\bar{\mathbf{x}}_0$ which is approximately equal to the original image $\mathbf{x}_0$.

The training objective is to minimize the negative log likelihood [Re 3] (i.e. maximize the log likelihood) of $\bar{\mathbf{x}}_0$ belonging to $\mathbf{x}_0$. We can define the loss function L as:

$$L = -\log p_\theta(\mathbf{x}_0); \text{ where } p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}$$

But it is non-sense if we integrate over a very high dimensional space directly, the authors reformulate the loss function using variational bound (variational lower bound, VLB) as follow:

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

which is equal to [Re 4]:

$$\mathbb{E}_q \Bigg[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Bigg]$$

After further simplify [1, 8, 9], we then obtain the final loss function:

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

which is basically:

$$L_{\text{simple}} = \mathbb{E}\left[||\epsilon - \epsilon_\theta(\mathbf{x}_t, t)||^2 \right]$$

where $\epsilon_\theta(\mathbf{x}_t, t)$ denotes the predicted noise components in $\mathbf{x}_t$.Consequentially,

we can say that the objective is to make the predicted noise as similar as possible to the generated noise.

Figure 2. The results of unconditional CIFAR10 generation.

Figure 3. CelebA-HQ generated images.

Figure 3. Interpolations of CelebA-HQ 256x256 images with 500 timesteps of diffusion. The interpolated latent can be easily find using $\bar{\mathbf{x}}_t = (1-\lambda)\mathbf{x}_0 + \lambda\mathbf{x}'_0$ (see code interpolation.py).

Remarks

[Re 1]

The contribution of this study is not the architecture of denoising diffusion model, in fact, they use PixelCNN++ [2] as the backbone which is a U-Net based Wide ResNet [3] model.

[Re 2]

The webpage [8] is helpful for understanding the equations in this paper although there are mistakes in some of the equation derivations (Correct derivations can be found in [9]).

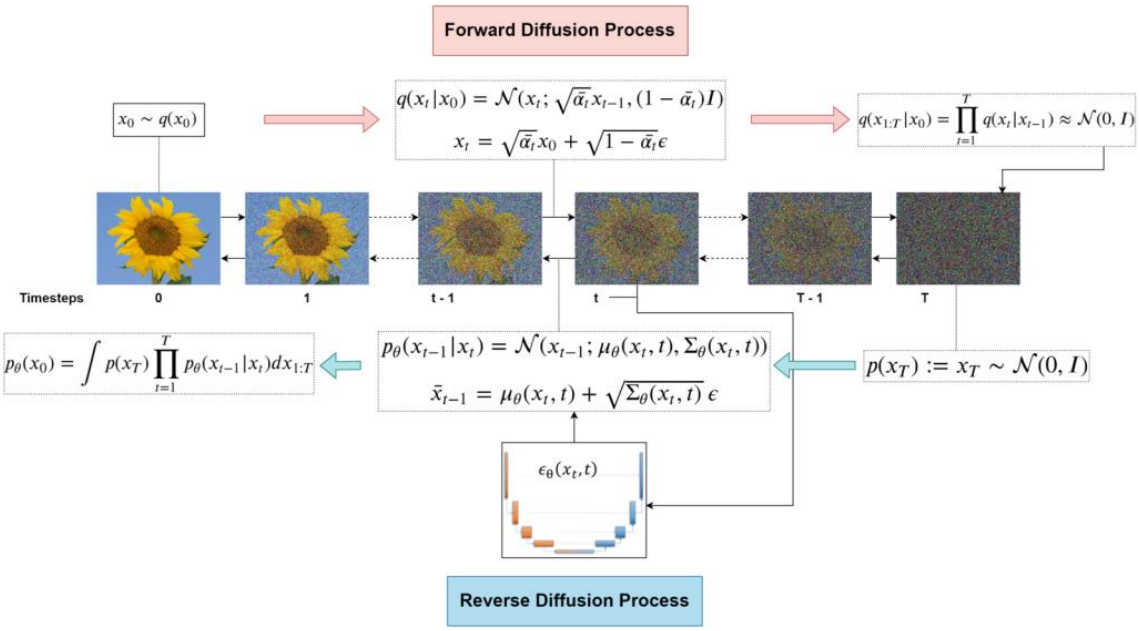

Figure 2. The overall process of the forward and reverse steps, along with the corresponding equations.

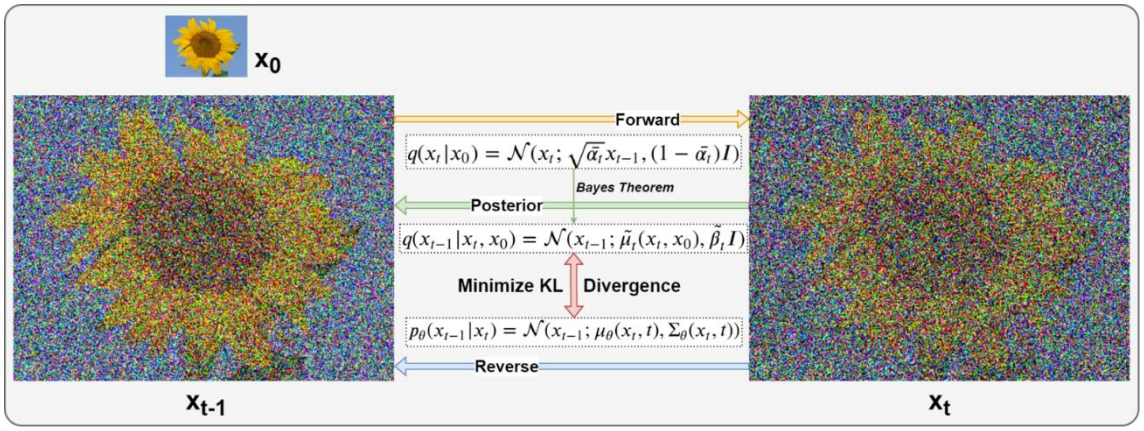

Figure 3. The related equations of training process.

[Re 3]
Likelihood is a measure of how likely a particular set of parameters is, given some observed data. It's essentially the probability of observing the data, given a specific model and its parameters. The key point is, we are interested in the parameters, not the data. We want to find the parameters that maximize the likelihood of observing the given data. The log likelihood is simply the log of likelihood. Both likelihood and log likelihood are used to find the best parameters for a model, with log likelihood being the preferred method in practice due to its computational advantages.

[Re 4]
The derivation of the equation.

$$
\begin{aligned}
L &= \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \\
&= \mathbb{E}_q\left[D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T)) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]
\end{aligned}
$$

where the $\mathbb{E}$ denotes expectation operator, representing the expected value of a random variable.

References

[1] HO, Jonathan; JAIN, Ajay; ABBEEL, Pieter. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 2020, 33: 6840-6851.

[2] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. In International Conference on Learning Representations, 2017.

[3] ZAGORUYKO, Sergey; KOMODAKIS, Nikos. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[4] GOODFELLOW, Ian, et al. Generative adversarial nets. *Advances in neural information processing systems*, 2014, 27.

[5] KINGMA, Diederik P.; WELLING, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[6] DINH, Laurent; KRUEGER, David; BENGIO, Yoshua. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[7] REZENDE, Danilo; MOHAMED, Shakir. Variational inference with normalizing flows. In: *International conference on machine learning*. PMLR, 2015. p. 1530-1538.

[8] Vaibhav Singh, "An In-Depth Guide to Denoising Diffusion Probabilistic Models DDPM – Theory to Implementation," learnopencv.com. Access: Aug, 12, 2024. [Online.] Available: https://learnopencv.com/denoising-diffusion-probabilistic-model

[9] DZ, "Intro to Diffusion Model — Part 3," medium.com. Access: Aug, 12, 2024. [Online.] Available: https://dzdata.medium.com/intro-to-diffusion-model-part-3-5d699e5f0714

Comment

This week, we improved out meeting note by:

(1) Adding the remark reference symbol [Re No.], which can help finding the remarks faster.