Discourse on Generative Diffusion Models (29): Using DDPM for Discrete Encoding

Su Jianlin

February 14, 2025

A couple of days ago, I came across a new paper on arXiv, *Compressed Image Generation with Denoising Diffusion Codebook Models*, and I was truly impressed by the author's wild imagination—I can't help but share it with everyone.

As the title of this post suggests, the authors propose an idea called DDCM (**D**enoising **D**iffusion **C**odebook **M**odels), which restricts the noise sampling of DDPM to a finite set, and thereby achieves some remarkable effects, such as encoding samples into a sequence of discrete IDs and reconstructing them, just like VQ-VAE. Note that all of these operations are performed on a pre-trained DDPM, requiring no additional training.

1 Finite Set

Since DDCM only needs a pre-trained DDPM model to perform sampling, we will not repeat the model details of DDPM here. Readers who are not yet familiar with DDPM may review parts (1), (2), and (3) of our "Discourse on Generative Diffusion Models" series.

As we know, the generative sampling of DDPM starts from $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and iterates to $\boldsymbol{x}_0$ via

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}(\boldsymbol{x}_t) + \sigma_t \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{1}$$

For DDPM, every iteration step requires sampling a noise, and since $\boldsymbol{x}_T$ itself is also sampled noise, a total of $T+1$ noise vectors are fed in during the T-step iterative generation of $\boldsymbol{x}_0$. Assuming $\boldsymbol{x}_t \in \mathbb{R}^d$, the sampling process of DDPM is in fact a mapping $\mathbb{R}^{d\times(T+1)} \mapsto \mathbb{R}^d$.

The first ingenious idea of DDCM is to replace the noise sampling space with a finite set (a Codebook):

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}(\boldsymbol{x}_t) + \sigma_t \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{C}_t \tag{2}$$

together with $\boldsymbol{x}_T \sim \mathcal{C}_{T+1}$, where $\mathcal{C}_t$ is a set of K noises pre-sampled from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. In other words, before sampling we draw K vectors from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and keep them fixed; afterwards, every sampling operation draws uniformly from these K vectors. In this way, the generative process becomes a mapping $\{1, 2, \cdots, K\}^{T+1} \mapsto \mathbb{R}^d$.

What loss does this cause in generation quality? DDCM ran experiments and found the loss to be very small:

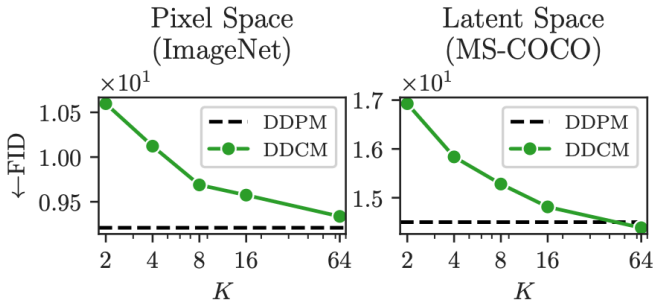

Figure 1: After changing the noise sampling space to a finite set, the FID of the generation results does not noticeably degrade.

We can see that $K = 64$ already essentially matches the FID, and on close inspection even $K = 2$ does not lose much, which indicates that making the sampling space finite can preserve the generative capability of DDPM. Note a detail here: the $\mathcal{C}_t$ at each step is independent, i.e., all the codebooks together contain $(T + 1)K$ noises, rather than sharing K noises. I did a simple reproduction and found that with sharing, $K \geq 8192$ is needed to maintain a similar effect.

2 Discrete Encoding

Now we consider an inverse problem: find the discrete encoding of a given $\boldsymbol{x}_0$, i.e., find the corresponding sequence $\boldsymbol{\varepsilon}_T \in \mathcal{C}_T, \cdots, \boldsymbol{\varepsilon}_1 \in \mathcal{C}_1$, such that the $\boldsymbol{x}_0$ generated by iterating $\boldsymbol{x}_{t-1} = \boldsymbol{\mu}(\boldsymbol{x}_t) + \sigma_t \boldsymbol{\varepsilon}_t$ is as close as possible to the given sample.

Since making the sampling space finite does not noticeably change the FID, we can assume that all samples from the same distribution can be generated via Eq. (2), so the solution to the above inverse problem exists in theory. But how do we find it? The intuitive idea is to work backwards from $\boldsymbol{x}_0$, but upon reflection we find this hard to operate: for instance, the first step $\boldsymbol{x}_0 = \boldsymbol{\mu}(\boldsymbol{x}_1) + \sigma_1 \boldsymbol{\varepsilon}_1$ involves both $\boldsymbol{x}_1$ and $\boldsymbol{\varepsilon}_1$ being unknown, making it difficult to pin them down simultaneously.

At this point, the second ingenious idea of DDCM appears: it treats the discrete encoding of $\boldsymbol{x}_0$ as a conditional controlled generation problem! Specifically, starting from a fixed $\boldsymbol{x}_T$, we select $\boldsymbol{\varepsilon}_t$ in the following way:

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}(\boldsymbol{x}_t) + \sigma_t \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t = \operatorname*{argmax}_{\boldsymbol{\varepsilon} \in \mathcal{C}_t} \boldsymbol{\varepsilon} \cdot (\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)) \tag{3}$$

Here $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$ is the model that predicts $\boldsymbol{x}_0$ from $\boldsymbol{x}_t$, and its relation to $\boldsymbol{\mu}(\boldsymbol{x}_t)$ is:

$$\boldsymbol{\mu}(\boldsymbol{x}_t) = \frac{\alpha_t \bar{\beta}_{t-1}^2}{\bar{\beta}_t^2} \boldsymbol{x}_t + \frac{\bar{\alpha}_{t-1} \beta_t^2}{\bar{\beta}_t^2} \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t) \tag{4}$$

If readers have forgotten this part, they can review *Discourse on Generative Diffusion Models (3): DDPM = Bayes + Denoising*.

We will discuss the detailed derivation in the next section; for now, let us first contemplate Eq. (3). Its only difference from the random generation of Eq. (2) is that the optimal $\boldsymbol{\varepsilon}_t$ is selected via argmax, with the criterion being the inner-product similarity between $\boldsymbol{\varepsilon}$ and the residual $\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$; intuitively, this makes $\boldsymbol{\varepsilon}$ compensate as much as possible for the gap between the current $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$ and the target $\boldsymbol{x}_0$. Through iterating Eq. (3), the image is equivalently converted into $T-1$ integers (σ_1 is usually set to zero).

The rule looks quite simple, so how is the actual reconstruction quality? Below is a figure excerpted from the original paper; we can see it is rather stunning, and the original paper has more such figures. I also tried it on my own model and found that similar results can basically be reproduced, so the method is quite reliable.

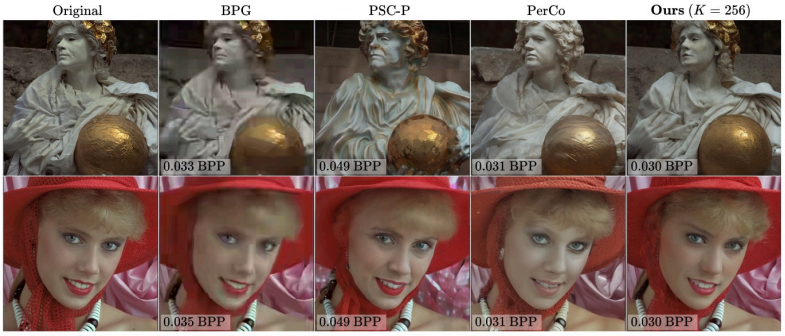

Figure 4. **Qualitative image compression results.** The presented images are taken from the Kodak24 (512×512) dataset. Our codec produces highly realistic outputs, while maintaining better fidelity to the original images compared to previous methods.

Figure 2: Discrete encoding reconstruction results of DDCM.

3 Conditional Generation

We just said that DDCM treats the encoding process as a conditional controlled generation process. How should we understand this statement? Starting from Eq. (1) of DDPM, it can equivalently be written as

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}(\boldsymbol{x}_t), \sigma_t^2 \boldsymbol{I}) \tag{5}$$

What we want to do now is to control the generation process given $\boldsymbol{x}_0$, so we add an extra condition $\boldsymbol{x}_0$ to the above distribution, i.e., change it to $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$. In fact, in DDPM, $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ has an analytical solution, which we already derived in *Discourse on Generative Diffusion Models (3): DDPM = Bayes + Denoising*:

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_{t-1}; \frac{\alpha_t \bar{\beta}_{t-1}^2}{\bar{\beta}_t^2}\boldsymbol{x}_t + \frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2}\boldsymbol{x}_0, \frac{\bar{\beta}_{t-1}^2\beta_t^2}{\bar{\beta}_t^2}\boldsymbol{I}\right) \tag{6}$$

or written as

$$\begin{aligned}\boldsymbol{x}_{t-1} &= \frac{\alpha_t \bar{\beta}_{t-1}^2}{\bar{\beta}_t^2}\boldsymbol{x}_t + \frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2}\boldsymbol{x}_0 + \frac{\bar{\beta}_{t-1}\beta_t}{\bar{\beta}_t}\boldsymbol{\varepsilon}_t \\ &= \underbrace{\frac{\alpha_t \bar{\beta}_{t-1}^2}{\bar{\beta}_t^2}\boldsymbol{x}_t + \frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2}\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)}_{\boldsymbol{\mu}(\boldsymbol{x}_t)} + \frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2}(\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)) + \underbrace{\frac{\bar{\beta}_{t-1}\beta_t}{\bar{\beta}_t}}_{\sigma_t}\boldsymbol{\varepsilon}_t\end{aligned} \tag{7}$$

where $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Compared with Eq. (1), the above has an extra term $\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$, which guides the generation result toward $\boldsymbol{x}_0$. But remember that our task is to *encode* $\boldsymbol{x}_0$ discretely, so the generation process cannot explicitly involve $\boldsymbol{x}_0$, otherwise the causality would be inverted. To this end, we hope that the term $\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$ can be compensated from $\boldsymbol{\varepsilon}_t$, so we adjust the selection rule of $\boldsymbol{\varepsilon}_t$ to

$$\boldsymbol{\varepsilon}_t = \operatorname*{argmin}_{\boldsymbol{\varepsilon}\in\mathcal{C}_t} \left\| \frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2}(\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)) - \frac{\bar{\beta}_{t-1}\beta_t}{\bar{\beta}_t}\boldsymbol{\varepsilon} \right\| \tag{8}$$

Since the vectors in $\mathcal{C}_t$ are all pre-sampled from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, similar to the "unit-norm lemma" in *The Amazing Johnson–Lindenstrauss Lemma: Theory*, we can assume the vectors in $\mathcal{C}_t$ have roughly the same norm. Under this assumption, the above is also equivalent to

$$\boldsymbol{\varepsilon}_t = \operatorname*{argmax}_{\boldsymbol{\varepsilon}\in\mathcal{C}_t} \boldsymbol{\varepsilon} \cdot (\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)) \tag{9}$$

which yields Eq. (3) of DDCM.

4 Importance Sampling

In the above derivation, we used the statement that "we hope the term $\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$ can be compensated from $\boldsymbol{\varepsilon}_t$, so we change the selection rule of $\boldsymbol{\varepsilon}_t$ to Eq. (8)". Although this seems intuitive, it is mathematically not rigorous—strictly speaking, it is even wrong. In this section we make this part rigorous.

Reviewing Eq. (7) again: the derivation up to Eq. (7) is rigorous; the non-rigorous part lies in how Eq. (7) is connected to $\mathcal{C}_t$. Think about it: what happens when we select $\boldsymbol{\varepsilon}_t$

according to (8) as $K \to \infty$? At that point we can consider $\mathcal{C}_t$ to already cover the whole $\mathbb{R}^d$, so $\frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2}(\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)) = \frac{\bar{\beta}_{t-1}\beta_t}{\bar{\beta}_t}\boldsymbol{\varepsilon}_t$; that is, Eq. (7) becomes a deterministic transformation:

$$\boldsymbol{x}_{t-1} = \frac{\alpha_t\bar{\beta}_{t-1}^2}{\bar{\beta}_t^2}\boldsymbol{x}_t + \frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2}\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t) + \frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2}(\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)) = \frac{\alpha_t\bar{\beta}_{t-1}^2}{\bar{\beta}_t^2}\boldsymbol{x}_t + \frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2}\boldsymbol{x}_0 \quad (10)$$

In other words, when $K \to \infty$ the original stochastic sampling trajectory cannot be recovered, which is really not a scientific state of affairs. We believe that making the sampling space finite should be an approximation of the continuous sampling space; when $K \to \infty$, the continuous sampling trajectory should be recovered, so that each step of the discretization has more theoretical guarantee. Or, in a more mathematical expression, we believe the necessary condition is

$$\lim_{K\to\infty} \text{DDCM} = \text{DDPM} \quad (11)$$

Going from Eq. (1) to Eq. (7), we can also view it as the noise distribution changing from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ to $\mathcal{N}(\frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2\sigma_t}(\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)), \boldsymbol{I})$; but for the sake of encoding, we cannot sample directly from the latter and can only sample from the finite set $\mathcal{C}_t$. To satisfy the necessary condition, i.e., to make the sampling result closer to sampling from $\mathcal{N}(\frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2\sigma_t}(\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)), \boldsymbol{I})$, we can weight $\boldsymbol{\varepsilon} \in \mathcal{C}_t$ by its probability density function:

$$p(\boldsymbol{\varepsilon}) \propto \exp\left(-\frac{1}{2}\left\|\boldsymbol{\varepsilon} - \frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2\sigma_t}(\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t))\right\|^2\right), \quad \boldsymbol{\varepsilon} \in \mathcal{C}_t \quad (12)$$

That is to say, the most reasonable way is to apply a Softmax to $-\frac{1}{2}\left\|\boldsymbol{\varepsilon} - \frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2\sigma_t}(\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t))\right\|^2$ and then do importance sampling according to the probabilities, rather than directly taking the argmax. It is just that when K is relatively small, the distribution after Softmax will be close to a one-hot distribution, so sampling by probability is approximately the argmax.

5 General Form

We can also generalize the above results to Classifier-Guidance generation. According to the derivation in *Discourse on Generative Diffusion Models (9): Controlling Generation Results with Conditions*, adding Classifier-Guidance to Eq. (1) yields

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}(\boldsymbol{x}_t) + \sigma_t^2\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{y}|\boldsymbol{x}_t) + \sigma_t\boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \quad (13)$$

i.e., the new term $\sigma_t^2\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{y}|\boldsymbol{x}_t)$ is added, where $p(\boldsymbol{y}|\boldsymbol{x}_t)$ is a classifier for noisy samples. If we only have a noise-free sample classifier $p_o(\boldsymbol{y}|\boldsymbol{x})$, then we can also let $p(\boldsymbol{y}|\boldsymbol{x}_t) = p_o(\boldsymbol{y}|\boldsymbol{\mu}(\boldsymbol{x}_t))$.

Under the same derivation, we can obtain a selection rule similar to Eq. (8):

$$\boldsymbol{\varepsilon}_t = \mathop{\text{argmin}}_{\varepsilon\in\mathcal{C}_t} \left\|\sigma_t^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t) - \sigma_t \boldsymbol{\varepsilon}\right\| \tag{14}$$

or approximately

$$\boldsymbol{\varepsilon}_t = \mathop{\text{argmax}}_{\varepsilon\in\mathcal{C}_t} \boldsymbol{\varepsilon} \cdot \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t) \tag{15}$$

and of course we can also construct the sampling distribution according to Eq. (12):

$$p(\boldsymbol{\varepsilon}) \propto \exp\left(-\frac{1}{2}\left\|\boldsymbol{\varepsilon} - \sigma_t \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t)\right\|^2\right), \quad \boldsymbol{\varepsilon}\in\mathcal{C}_t \tag{16}$$

These are all simple generalizations of the aforementioned results.

6 Personal Commentary

At this point, our introduction to DDCM is basically complete; for more details, please refer to the original paper. The authors have not yet open-sourced the code; here I provide my own reference implementation:

DDCM: https://github.com/bojone/Keras-DDPM/blob/main/ddcm.py

If you already know something about diffusion models and have a ready-made diffusion model at hand, I strongly recommend trying it yourself. In fact, the principle of DDCM is easy to understand and the code is not hard to write, but only by trying it personally can you experience that breathtaking sense of amazement. The last time I had the same feeling was with Upsample Guidance, a training-free technique for generating large images introduced in *Discourse on Generative Diffusion Models (23): Signal-to-Noise Ratio and Large Image Generation (Part 2)*, which likewise reflects the author's ingenious conception.

But in terms of long-term impact, I personally think Upsample Guidance is still inferior to DDCM. Discrete encoding of images is one of the mainstream routes for multimodal LLMs; it plays the role of an "image tokenizer" and is a crucial link. DDCM can be said to open up a brand-new route beyond VQ and FSQ, and thus may have more far-reaching potential impact. In the original paper, DDCM only defines itself as a compression method, which is somewhat "underestimating itself".

As a discrete encoding model, DDCM has another outstanding advantage: its discrete codes are naturally 1D, unlike schemes such as VQ and FSQ whose encodings usually retain the 2D character of the image (except models like TiTok that use the Q-Former idea to convert to 1D). This means that when we use these codes for autoregressive generation, we no longer need to consider the "ordering" problem (see *Thoughts on Multimodality Behind Closed Doors (2): Autoregression*), which is a great relief.

Of course, at present there is still some room for improvement. For example, the encoding and generation currently happen simultaneously, which means the encoding speed of DDCM is as slow as the sampling speed of DDPM—still rather unacceptable at present. Yet we cannot casually apply accelerated sampling tricks, because acceleration means reducing T, and reducing T means shortening the code length, i.e., increasing the compression ratio, which would noticeably increase the reconstruction loss.

Overall, I personally think DDCM is a very interesting discrete encoding method whose potential awaits excavation and which simultaneously urgently awaits further optimization.

7 Article Summary

This article introduced a new idea for diffusion models: it restricts the noise in the DDPM generation process to a finite set and, combined with the idea of conditional generation, turns DDPM training-free into a discrete autoencoder similar to VQ-VAE.