Generative Diffusion Models (11): Unified Diffusion Model (Application)

Jianlin Su

September 21, 2022

In "Generative Diffusion Models (10): Unified Diffusion Model (Theory)", I claimed to have constructed a Unified Diffusion Model (UDM) framework that allows for more general diffusion methods and data types. Can the UDM framework actually achieve its intended purpose? This article demonstrates its generality through several specific examples.

1 Framework Review

First, UDM constructs the forward process by choosing a noise distribution $q(\boldsymbol{\varepsilon})$ and a transformation $\boldsymbol{\mathcal{F}}$:

$$\boldsymbol{x}_t = \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0, \boldsymbol{\varepsilon}), \quad \boldsymbol{\varepsilon} \sim q(\boldsymbol{\varepsilon}) \tag{1}$$

Then, sampling for the reverse process $\boldsymbol{x}_{t-1} \sim p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is achieved through the following decomposition:

$$\hat{\boldsymbol{x}}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t) \quad \& \quad \boldsymbol{x}_{t-1} \sim p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0 = \hat{\boldsymbol{x}}_0) \tag{2}$$

Where $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is the probability of estimating $\boldsymbol{x}_0$ given $\boldsymbol{x}_t$, generally modeled using a simple distribution $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ as an approximation. The training objective is essentially $-\log q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ or a simple variant thereof. When $\boldsymbol{x}_0$ is continuous data, $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is typically chosen as a conditional normal distribution; when $\boldsymbol{x}_0$ is discrete data, $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ can be an autoregressive or non-autoregressive model.

As for the baseline choice for $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$, it is:

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0) \quad \Leftrightarrow \quad \boldsymbol{x}_{t-1} = \boldsymbol{\mathcal{F}}_{t-1}(\boldsymbol{x}_0, \boldsymbol{\varepsilon}) \tag{3}$$

Starting from this baseline, different optimization results can be obtained under different conditions. If $\boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0, \boldsymbol{\varepsilon})$ is invertible with respect to $\boldsymbol{\varepsilon}$, then we can solve for $\boldsymbol{\varepsilon} = \boldsymbol{\mathcal{F}}_t^{-1}(\boldsymbol{x}_0, \boldsymbol{x}_t)$, leading to a better deterministic sampling method:

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mathcal{F}}_{t-1}(\boldsymbol{x}_0, \boldsymbol{\mathcal{F}}_t^{-1}(\boldsymbol{x}_0, \boldsymbol{x}_t)) \tag{4}$$

Furthermore, if $q(\boldsymbol{\varepsilon})$ is a standard normal distribution, we can obtain:

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mathcal{F}}_{t-1}(\boldsymbol{x}_0, \sqrt{1-\tilde{\sigma}_t^2}\boldsymbol{\mathcal{F}}_t^{-1}(\boldsymbol{x}_0, \boldsymbol{x}_t) + \tilde{\sigma}_t\boldsymbol{\varepsilon}) \tag{5}$$

2 Hot Diffusion

In this section, we prove that "Hot Diffusion Models" are a special case of UDM. Here, Hot Diffusion refers to the mainstream diffusion models introduced previously, such as DDPM and DDIM. This term originates from the "Cold Diffusion" paper discussed below.

Mainstream diffusion models handle continuous data and construct the forward process using additive Gaussian noise:

$$\boldsymbol{x}_t = \bar{\alpha}_t \boldsymbol{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{6}$$

The choice for $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is the normal distribution $\mathcal{N}(\boldsymbol{x}_0; \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t), \bar{\sigma}_t^2 \boldsymbol{I})$. Generally, $\bar{\sigma}_t$ is not treated as a trainable parameter, so after omitting constant terms, we have:

$$-\log q(\boldsymbol{x}_0|\boldsymbol{x}_t) = \frac{1}{2\bar{\sigma}_t^2} \|\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)\|^2 \tag{7}$$

Further introducing the parameterization $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t) = \frac{1}{\bar{\alpha}_t}\Big(\boldsymbol{x}_t - \bar{\beta}_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\Big)$ and combining it with $\boldsymbol{x}_t = \bar{\alpha}_t \boldsymbol{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}$, we get:

$$-\log q(\boldsymbol{x}_0|\boldsymbol{x}_t) = \frac{\bar{\beta}_t^2}{2\bar{\sigma}_t^2 \bar{\alpha}_t^2} \Big\|\boldsymbol{\varepsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\bar{\alpha}_t \boldsymbol{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t)\Big\|^2 \tag{8}$$

Experiments show that omitting the preceding coefficients yields better results, so the final training objective is generally $\|\boldsymbol{\varepsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\bar{\alpha}_t \boldsymbol{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t)\|^2$. Regarding the choice of $\bar{\sigma}_t$ during the sampling process, one can refer to "Generative Diffusion Models (7): Optimal Diffusion Variance Estimation (Part 1)" and "Generative Diffusion Models (8): Optimal Diffusion Variance Estimation (Part 2)".

Finally, regarding $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$, we have:

$$\begin{aligned}\boldsymbol{x}_{t-1} =& \bar{\alpha}_{t-1} \boldsymbol{x}_0 + \bar{\beta}_{t-1} \boldsymbol{\varepsilon} \\ \sim& \bar{\alpha}_{t-1} \boldsymbol{x}_0 + \sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}\, \boldsymbol{\varepsilon}_1 + \sigma_t \boldsymbol{\varepsilon}_2\end{aligned}, \quad \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{9}$$

Solving for $\boldsymbol{\varepsilon}$ from $\boldsymbol{x}_t = \bar{\alpha}_t \boldsymbol{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}$ gives $\boldsymbol{\varepsilon} = (\boldsymbol{x}_t - \bar{\alpha}_t \boldsymbol{x}_0)/\bar{\beta}_t$. Replacing $\boldsymbol{\varepsilon}_1$, we eventually obtain the general reverse process for $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ as:

$$\boldsymbol{x}_{t-1} = \bar{\alpha}_{t-1} \boldsymbol{x}_0 + \sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}\, \frac{\boldsymbol{x}_t - \bar{\alpha}_t \boldsymbol{x}_0}{\bar{\beta}_t} + \sigma_t \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{10}$$

And $\hat{\boldsymbol{x}}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ implies:

$$\hat{\boldsymbol{x}}_0 = \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t) + \bar{\sigma}_t \boldsymbol{\varepsilon} = \frac{1}{\bar{\alpha}_t}\Big(\boldsymbol{x}_t - \bar{\beta}_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\Big) + \bar{\sigma}_t \boldsymbol{\varepsilon} \tag{11}$$

Combining the two equations above yields the reverse process of the most general mainstream diffusion model framework. DDPM takes $\bar{\sigma}_t = 0, \sigma_t = \frac{\bar{\beta}_{t-1}\beta_t}{\bar{\beta}_t}$, DDIM takes $\bar{\sigma}_t = 0, \sigma_t = 0$, and Analytical-DPM re-estimates the optimal non-zero $\bar{\sigma}_t$.

3 Cold Diffusion

Next, we prove that "Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise" is also a special case of UDM. Cold Diffusion also handles continuous data. As seen from the title, it focuses on using arbitrary (noise-free) transformations to construct the forward process. To the best of my knowledge, this is the first paper to attempt a general forward process. UDM was heavily inspired by it during its construction, and I would like to express my gratitude to the original authors.

Cold Diffusion constructs the forward process through a deterministic transformation $\boldsymbol{x}_t = \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0)$. To facilitate subsequent analysis, we introduce a more general forward process:

$$\boldsymbol{x}_t = \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0) + \sigma \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim q(\boldsymbol{\varepsilon}) \tag{12}$$

The transformation $\boldsymbol{\mathcal{F}}$ can be any form of degradation of the original data, such as blurring, masking, or pooling for images. If a deterministic transformation is required, we can simply let $\sigma \to 0$ after the fact.

Next, the choice for $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is a normal distribution measured by the l_1 norm:

$$q(\boldsymbol{x}_0|\boldsymbol{x}_t) = \frac{1}{Z(\tau)} \int e^{-\|\boldsymbol{x}_0 - \boldsymbol{\mathcal{G}}_t(\boldsymbol{x}_t)\|_1/\tau} d\boldsymbol{x}_0 \tag{13}$$

where $Z(\tau)$ is the normalization factor. Taking τ as a fixed value, after removing constant terms, we have $-\log q(\boldsymbol{x}_0|\boldsymbol{x}_t) \propto \|\boldsymbol{x}_0 - \boldsymbol{\mathcal{G}}_t(\boldsymbol{x}_t)\|_1$. Combined with $\boldsymbol{x}_t = \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0)$, the training objective is to minimize:

$$\|\boldsymbol{x}_0 - \boldsymbol{\mathcal{G}}_t(\boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0))\|_1 \tag{14}$$

In the reverse process, Cold Diffusion directly ignores the variance of $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ (i.e., letting $\tau \to 0$), resulting in $\hat{\boldsymbol{x}}_0 = \boldsymbol{\mathcal{G}}_t(\boldsymbol{x}_t)$. If $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ takes the baseline choice $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$, i.e., $\boldsymbol{x}_{t-1} = \boldsymbol{\mathcal{F}}_{t-1}(\boldsymbol{x}_0) + \sigma\boldsymbol{\varepsilon}$, then substituting $\hat{\boldsymbol{x}}_0$ and taking the limit $\sigma \to 0$ yields:

$$\hat{\boldsymbol{x}}_0 = \boldsymbol{\mathcal{G}}_t(\boldsymbol{x}_t), \quad \boldsymbol{x}_{t-1} = \boldsymbol{\mathcal{F}}_{t-1}(\hat{\boldsymbol{x}}_0) \tag{15}$$

This is the "Naive Sampling" from the original paper. If we solve for $\boldsymbol{\varepsilon} = (\boldsymbol{x}_t - \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0))/\sigma$ from $\boldsymbol{x}_t = \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0) + \sigma\boldsymbol{\varepsilon}$ and substitute it into $\boldsymbol{x}_{t-1} = \boldsymbol{\mathcal{F}}_{t-1}(\boldsymbol{x}_0) + \sigma\boldsymbol{\varepsilon}$, we get:

$$\hat{\boldsymbol{x}}_0 = \boldsymbol{\mathcal{G}}_t(\boldsymbol{x}_t), \quad \boldsymbol{x}_{t-1} = \boldsymbol{x}_t + \boldsymbol{\mathcal{F}}_{t-1}(\hat{\boldsymbol{x}}_0) - \boldsymbol{\mathcal{F}}_t(\hat{\boldsymbol{x}}_0) \tag{16}$$

This is the "Improved Sampling" from the original paper.

Overall, Cold Diffusion was the first to successfully implement a forward process with general transformations. However, because it emphasizes "Without Noise" too much, it has theoretical flaws that are difficult to overcome. For example, for image data of size $w \times w \times 3$, if Cold Diffusion uses a blurring operation for the forward process, the final result might be equivalent to a 3-dimensional vector. Since Cold Diffusion's reverse process is also deterministic, it means Cold Diffusion transforms a $3w^2$-dimensional image into 3 dimensions through a deterministic transformation and then reconstructs it back to $3w^2$ dimensions. This intermediate process inevitably involves severe information loss, which limits the clarity of reconstruction and, consequently, the clarity of generation.

To solve this problem, one cannot reject the existence of noise in the forward or reverse processes. Noise implies uncertainty, uncertainty implies "one-to-many," and "one-to-many" allows for a "many-to-one" forward process, which permits information loss. In fact, Cold Diffusion itself realized that a 3-dimensional vector is insufficient to generate complete $3w^2$-dimensional data; in the generation process, it actually adds slight $3w^2$-dimensional random noise to this 3-dimensional vector. Experiments showed that this operation improved generation results. This operation is roughly equivalent to a forward process where $\sigma > 0$.

4 Editing Models

The two examples above both deal with continuous data. As we mentioned, UDM in principle does not restrict the data type. In this section, we introduce a discrete example, showing that text generation models based on editing operations can essentially be seen as special cases of UDM.

For simplicity, let's consider the generation of fixed-length sentences of length l, such as five-character or seven-character quatrains. Variable-length sentences are possible but slightly more complex in detail. We define the forward process $\boldsymbol{x}_t = \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0, \boldsymbol{\varepsilon})$ as "random replacement":

Randomly select t tokens in the sentence and replace them with other random tokens.

where $t \leq l$. When $t = l$, $\boldsymbol{x}_t$ is a sequence of l completely random tokens.

In this case, $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is a model that predicts the original sequence from the randomly replaced sequence, using either an autoregressive or non-autoregressive model, with cross-entropy as the loss function. Note that $\boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0, \boldsymbol{\varepsilon})$ is necessarily non-invertible with respect to the noise (i.e., given $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$, there is more than one way to transform $\boldsymbol{x}_0$ into $\boldsymbol{x}_t$). Therefore, we can only use the baseline choice $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$, which means the generation process is:

1. Randomly select l tokens as the initial $\boldsymbol{x}_l$;
2. Predict $\hat{\boldsymbol{x}}_0$ from $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$;
3. Randomly select $t-1$ tokens from $\hat{\boldsymbol{x}}_0$ and replace them with other random tokens to obtain $\boldsymbol{x}_{t-1}$;
4. Repeat steps 2 and 3 until the final $\boldsymbol{x}_0$ is obtained.

However, the performance of such an algorithm will not be very good because the prediction from step 2 is often "ruined" by the random replacement in step 3, feeling like "one step forward, two steps back." To improve performance, a better sampling scheme is needed, which requires $\boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0, \boldsymbol{\varepsilon})$ to be invertible with respect to the noise—meaning that given $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$, the transformation can be uniquely identified. To this end, we define the forward process as:

> Randomly select t tokens in the sentence and replace them with **different** tokens.

The difference here is that during random replacement, the original token must be replaced by a token that is different from the original. Without this restriction, it might be replaced by the same token. With this constraint, we can directly compare the differences between $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$ to see what was modified, thereby changing the random replacement in step 3 to a transformation from $\hat{\boldsymbol{x}}_0$ to $\boldsymbol{x}_t$:

1. Randomly select l tokens as the initial $\boldsymbol{x}_l$;
2. Predict $\hat{\boldsymbol{x}}_0$ from $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$, requiring that $\hat{\boldsymbol{x}}_0$ and $\boldsymbol{x}_t$ have t different tokens (this is easier to implement with a non-autoregressive model);
3. Randomly select one of the tokens in $\boldsymbol{x}_t$ that is different from $\hat{\boldsymbol{x}}_0$, and replace it with the token at the corresponding position in $\hat{\boldsymbol{x}}_0$ to obtain $\boldsymbol{x}_{t-1}$;
4. Repeat steps 2 and 3 until the final $\boldsymbol{x}_0$ is obtained.

In this way, the valid parts of each prediction $\hat{\boldsymbol{x}}_0$ (the parts where $\hat{\boldsymbol{x}}_0$ matches $\boldsymbol{x}_t$) are preserved, and $\boldsymbol{x}_{t-1}$ only modifies one token compared to $\boldsymbol{x}_t$, making the generation process a stable, progressive generation. The difference from a standard autoregressive model is the removal of the left-to-right generation constraint.

5 Masking Models

If the above model is still vague, here is another simple example to help understanding. Again, consider the generation of fixed-length sentences of length l. We define the forward process $\boldsymbol{x}_t = \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0, \boldsymbol{\varepsilon})$ as "random masking":

> Randomly select t tokens in the sentence and replace them with [MASK].

where $t \leq l$. When $t = l$, $\boldsymbol{x}_t$ consists of l [MASK] tokens.

In this case, $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is a model that predicts the original sequence from a sequence with [MASK] tokens, generally implemented using a BERT-like MLM model (non-autoregressive), with cross-entropy as the loss function. The baseline generation process is:

1. Use l [MASK] tokens as the initial $\boldsymbol{x}_l$;
2. Sample $\hat{\boldsymbol{x}}_0$ from $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$;
3. Randomly select $t-1$ tokens from $\hat{\boldsymbol{x}}_0$ and replace them with [MASK] to obtain $\boldsymbol{x}_{t-1}$;
4. Repeat steps 2 and 3 until the final $\boldsymbol{x}_0$ is obtained.

Note that in this case, $\boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_0, \boldsymbol{\varepsilon})$ is invertible with respect to the noise; we can clearly see from $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$ which tokens were replaced by [MASK]. Therefore, we can construct an improved generation process:

1. Use l [MASK] tokens as the initial $\boldsymbol{x}_l$;
2. Sample $\hat{\boldsymbol{x}}_0$ from $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$, noting that we only need to sample tokens that were originally [MASK]; non-[MASK] tokens remain unchanged;
3. Randomly select $t-1$ positions from the t [MASK] positions in the original $\boldsymbol{x}_t$, and replace the tokens at these positions in $\hat{\boldsymbol{x}}_0$ with [MASK] to obtain $\boldsymbol{x}_{t-1}$;
4. Repeat steps 2 and 3 until the final $\boldsymbol{x}_0$ is obtained.

Of course, steps 2 and 3 can be merged into a more direct step:

> 2 & 3. Randomly select 1 position from the t [MASK] positions in $\boldsymbol{x}_t$, and sample a token to replace it according to the probability at that position from $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$, resulting in $\boldsymbol{x}_{t-1}$.

This is almost identical to Gibbs sampling based on an MLM model (refer to "Searching for Text (3): Text Sampling based on BERT"). From the "Editing Model" and "Masking Model" examples, we can see that many "progressive generation" models can be reformulated using the UDM framework. Conversely, any progressive generation method we can think of can be attempted to be formulated using the UDM framework.

6 Encoding Models

The forward processes we discussed earlier were all without trainable parameters, meaning they were pre-designed procedures. However, this is not strictly necessary. We can generalize the diffusion process of DDPM as:

$$\boldsymbol{x}_t = \bar{\alpha}_t \boldsymbol{\mathcal{F}}(\boldsymbol{x}_0) + \bar{\beta}_t \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{17}$$

where $\boldsymbol{\mathcal{F}}(\boldsymbol{x}_0)$ is an encoding model for $\boldsymbol{x}_0$, which can have trainable parameters. The training objective then becomes:

$$-\log q(\boldsymbol{x}_0|\boldsymbol{x}_t) = -\log q(\boldsymbol{x}_0|\bar{\alpha}_t \boldsymbol{\mathcal{F}}(\boldsymbol{x}_0) + \bar{\beta}_t \boldsymbol{\varepsilon}) \tag{18}$$

except that $\boldsymbol{\mathcal{F}}$ also has trainable parameters. The reverse process is similar, except that after sampling $\hat{\boldsymbol{x}}_0 \sim q(\boldsymbol{x}_0|\boldsymbol{x}_1)$, we directly return $\hat{\boldsymbol{x}}_0$. Specifically, because of the additional encoding model $\boldsymbol{\mathcal{F}}$, the input $\boldsymbol{x}_0$ can be either discrete or continuous data. This provides a method similar to VAE for encoding data distributions into a normal distribution of latent variables.

7 Summary

This article primarily applies the Unified Diffusion Model (UDM) framework constructed in the previous post to derive several specific examples, including mainstream diffusion models, Cold Diffusion, text editing generation, and encoding models.