

Rethinking Diffusion Model in High Dimension

Zhenxin Zheng

blair.star@163.com

Zhenjie Zheng

zjzheng@hku.hk

Abstract

Curse of Dimensionality is an unavoidable challenge in statistical probability models, yet diffusion models seem to overcome this limitation, achieving impressive results in high-dimensional data generation. Diffusion models assume that they can learn the statistical properties of the underlying probability distribution, enabling sampling from this distribution to generate realistic samples. But is this really how they work? To address this question, this paper conducts a detailed analysis of the objective function and inference methods of diffusion models, leading to several important conclusions that help answer the above question: 1) In high-dimensional sparse scenarios, the target of the objective function fitting degrades from a **weighted sum of multiple samples** to a **single sample**. 2) The mainstream inference methods can all be represented within a **simple unified framework**, without requiring statistical concepts such as Markov chains and SDEs. 3) Guided by this simple framework, more efficient inference methods can be discovered.

Code is available at <https://github.com/blairstar/NaturalDiffusion>

1 Introduction

Diffusion models exhibit remarkable competitiveness in high-dimensional data generation scenarios, particularly in image generation [Rom+22]. Beyond delivering outstanding performance, diffusion models also possess various elegant mathematical formulations. [SD+15] first introduced the diffusion model approach, which utilizes a Markov chain to transform complex data distributions into simple normal distributions and then learns the posterior probability distribution corresponding to the transformation process. [HJA20] refined the objective function of diffusion models and introduced Denoised DPM. [Son+20] generalized the noise addition process of diffusion models from discrete to continuous and formulated it as a stochastic differential equation (SDE). [Lip+22; LGL22] proposed a new optimization perspective based on flow matching, enabling the model to directly learn the velocity field of probability flows.

All three of the aforementioned models assume that the diffusion model can learn the statistical properties of the data distribution. In the Markov Chain formulation, it is assumed that the model can learn the posterior probability distribution. In the stochastic differential equation (SDE) formulation, it is assumed that the model can learn the score of the marginal distribution. In the flow matching approach, it is assumed that the model can learn the velocity field. However, this assumption contradicts previous understandings. Traditionally, it is believed that in high-dimensional sparse scenarios, machine learning models cannot effectively learn complex hidden probability distributions. This raises the need to rethink whether diffusion models truly learn complex probability distributions and whether they function as originally assumed.

To address these questions, this paper presents the following analyses, contributing to a deeper understanding of high-dimensional diffusion models.

First, we analyze the impact of sparsity on the objective function of diffusion models. When the dataset is sufficiently sparse, the objective function degrades into a different form, transitioning from a "weighted sum" to a "single sample". Based on this new form, we propose an alternative interpretation of the objective function.

Next, we introduce a new inference framework. This framework not only aligns with the new interpretation of the objective function but also unifies most mainstream inference methods, including DDPM ancestral sampling, DDIM [SME20], Euler, DPM-Solver [Lu+22a], DPM-Solver++ [Lu+22b], and DEIS [ZC22].

Finally, leveraging this new framework, we design a novel inference method and demonstrate through experiments that it is more efficient than existing mainstream inference approaches.

2 Background

Given a batch of sampled data $X_0^0, X_0^1, \dots, X_0^N$ from the random variable X_0 , the diffusion model mixes the data with random noise in different proportions, forming a sequence of new variables X_1, X_2, \dots, X_T . The signal-to-noise ratio (SNR), which represents the ratio of data to noise, gradually decreases, and by the final variable X_T , it almost consists entirely of random noise.

For the original diffusion model and VP SDE, they mix in the following way:

$$X_t = \sqrt{\bar{\alpha}_t} \cdot X_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon \quad (1)$$

Where $\bar{\alpha}_t$ gradually decreases from 1 to 0, and t takes discrete values from 1 to T .

For flow matching, it mixes in the following way:

$$X_t = (1 - \sigma_t) \cdot X_0 + \sigma_t \cdot \epsilon \quad (2)$$

Where σ_t also gradually decreases from 1 to 0, and in practice, $\sigma_t = t$ is often set. t takes continuous values, $t \in [0, 1]$.

2.1 Markov Chain-based Diffusion Models

For the Markov Chain-based diffusion model, its core lies in learning the conditional posterior probability $p(x_{t-1}|x_t)$. Since the posterior probability is approximately a Gaussian function, and its variance is relatively fixed, we can focus on learning the mean of the conditional posterior probability $E_{p(x_{t-1}|x_t)}(x_{t-1})$. According to the Total Law of Expectation[Ros10], the mean can be expressed in another form:

$$E_{p(x_{t-1}|x_t)}(x_{t-1}) = \int p(x_0|x_t) E_{p(x_{t-1}|x_0, x_t)}(x_{t-1}) dx_0 \quad (3)$$

As seen from equation (7) in [HJA20], the mean of $p(x_{t-1}|x_0, x_t)$ can be expressed as a linear combination of x_0 and x_t , i.e.

$$E_{p(x_{t-1}|x_0, x_t)}(x_{t-1}) = \underbrace{\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \cdot x_0}_{const=C_0} + \underbrace{\frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \cdot x_t}_{const=C_t} \quad (4)$$

Thus, the mean of $p(x_{t-1}|x_t)$ can be further expressed as

$$E_{p(x_{t-1}|x_t)}(x_{t-1}) = \int p(x_0|x_t) (C_0 \cdot x_0 + C_t \cdot x_t) dx_0 \quad (5)$$

$$= C_0 \int p(x_0|X_t) x_0 dx_0 + C_t \cdot x_t \int p(x_0|x_t) dx_0 \quad (6)$$

$$= C_0 \int p(x_0|x_t) x_0 dx_0 + C_t \cdot x_t \quad (7)$$

Therefore, the learning objective of the Markov Chain-based diffusion model can be considered as learning the mean of the posterior probability $p(x_0|x_t)$ ($\int p(x_0|x_t) x_0 dx_0$), i.e.

$$\min_{\theta} \int p(x_t) \left\| f_{\theta}(x_t) - \int p(x_0|x_t) x_0 dx_0 \right\|^2 dx_t \quad (8)$$

where $f_{\theta}(x_t)$ is a learnable neural network function, with input x_t .

2.2 Score-based Diffusion Model

For the score-based diffusion model, its core lies in learning the score of the marginal distribution $p(x_t)$ ($\frac{\partial \log p(x_t)}{\partial x_t}$). Similar to the Markov chain-based diffusion model mentioned above, by introducing another variable X_0 , the score can be expressed in another form:

$$\frac{\partial \log p(x_t)}{\partial x_t} = \int p(x_0|x_t) \frac{\partial \log p(x_t|x_0)}{\partial x_t} dx_0 \quad (9)$$

The proof of this relationship can be found in Appendix A.2.

Since $p(x_t|x_0) \sim \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, \sqrt{1 - \bar{\alpha}_t})$, the score of $p(x_t|x_0)$ can be expressed as

$$\frac{\partial \log p(x_t|x_0)}{\partial x_t} = -\frac{x_t - \sqrt{\bar{\alpha}_t}x_0}{1 - \bar{\alpha}_t} = \underbrace{\frac{\sqrt{\bar{\alpha}_t}}{1 - \bar{\alpha}_t} \cdot x_0}_{const=S_0} + \underbrace{\frac{-1}{1 - \bar{\alpha}_t} \cdot x_t}_{const=S_t} \quad (10)$$

Thus, the score of $p(x_t)$ can be expressed as

$$\frac{\partial \log p(x_t)}{\partial x_t} = \int p(x_0|x_t) (S_0 \cdot x_0 + S_t \cdot X_1) dx_0 \quad (11)$$

$$= S_0 \int p(x_0|x_t) x_0 dx_0 + S_t \cdot X_1 \quad (12)$$

Therefore, the learning objective of the score-based diffusion model can also be considered as learning the mean of the posterior probability $p(x_0|x_t)$ ($\int p(x_0|x_t) x_0 dx_0$).

2.3 Flow Matching-based Diffusion

The core of the flow matching-based diffusion model lies in learning the velocity field of the probability flow. According to Theorem 1 in [Lip+22], the velocity field $u(x_t)$ can be expressed as a weighted sum of the conditional velocity field $u(x_t|x_0)$, i.e.

$$u(x_t) = \int p(x_0|x_t) u(x_t|x_0) dx_0 \quad (13)$$

From equation 2, we know that the conditional velocity field $u(x_t|x_0)$ is

$$u(x_t|x_0) \triangleq \frac{dx_t}{dt} = \epsilon - x_0 \quad (14)$$

Thus, the velocity field $u(x_t)$ can be expressed as

$$u(x_t) = \int p(x_0|x_t) (\epsilon - x_0) dx_0 = \epsilon - \int p(x_0|x_t) x_0 dx_0 \quad (15)$$

Therefore, the learning objective of the flow matching-based diffusion model can also be considered as learning the mean of the posterior probability $p(x_0|x_t)$ ($\int p(x_0|x_t) x_0 dx_0$).

2.4 Predicting Posterior Mean is Equivalent to Predicting X_0

Fitting the mean of the conditional posterior probability $p(x_0|x_t)$ is equivalent to **predicting** X_0 , i.e.

$$\min_{\theta} \int p(x_t) \left\| f_{\theta}(x_t) - \int p(x_0|x_t) x_0 dx_0 \right\|^2 dx_t \iff \min_{\theta} \iint p(x_0, x_t) \|f_{\theta}(x_t) - x_0\|^2 dx_0 dx_t \quad (16)$$

The specific proof can be found in Appendix A.1.

3 Impact of Sparsity on the Objective Function

As mentioned in Section 2, the core of the diffusion model lies in learning to fit the mean of the posterior probability distribution $p(x_0|x_t)$ for each x_t . However, since x_t can take an infinite number of values, it is not feasible to exhaustively traverse all of them. The practical approach is to sample x_t based on the probability distribution $p(x_t)$, with higher probability values being sampled first. Since $p(x_t)$ is unknown, direct sampling is not possible; instead, ancestral sampling is used as an indirect sampling method. The specific sampling method is as follows: sample X_0 from $p(x_0)$, and then sample X_t from $p(x_t|x_0)$. The pair (X_0, X_t) follows the joint distribution $p(x_0, x_t)$, while the individual X_t follows $p(x_t)$.

Next, we analyze the characteristics of the mean of $p(x_t|x_0)$ encountered during the optimization process. To analyze the specific characteristics of the mean, we first introduce the form of the distribution $p(x_0|x_t)$.

3.1 Form of Posterior $p(x_0|x_t)$

For convenience, we use a unified form to represent the two mixing ways in Equation (1) and Equation (2) as follows:

$$x_t = c_0 \cdot x_0 + c_1 \cdot \epsilon \quad (17)$$

When $c_0^2 + c_1^2 = 1$, this represents the mixing way of the Markov Chain-based diffusion model and the Score-based diffusion model. When $c_0 + c_1 = 1$, it represents the Flow Matching mixing method. Under this representation, $p(x_t|x_0) \sim \mathcal{N}(x_t; c_0 x_0, c_1^2)$.

From the analysis in Appendix A.3, the posterior probability distribution $p(x_0|x_t)$ has the following form:

$$p(x_0|x_t) = \text{Normalize} \left(\exp \frac{-(x_0 - \mu)^2}{2\sigma^2} p(x_0) \right) \quad \text{where } \mu = \frac{x_t}{c_0} \quad \sigma = \frac{c_1}{c_0} \quad (18)$$

In the actual optimization, $p(x_0)$ is the hidden data distribution, which is unknown and cannot be sampled directly. It can only be randomly selected from the existing samples $\{X_0^0, X_0^1, \dots, X_0^N\}$ ($X_0^i \sim p(x_0)$). The selection process can be considered as sampling from the following mixed Dirac delta distribution:

$$p(x_0|x_t) = \frac{1}{N} \sum_{i=0}^N \delta(x_0 - X_0^i) \quad (19)$$

Substituting this into Equation (18), we get:

$$p(x_0|x_t) = \text{Normalize} \left(\exp \frac{-(x_0 - \mu)^2}{2\sigma^2} \sum_{i=0}^N \delta(x_0 - X_0^i) \right) \quad (20)$$

$$= \frac{1}{Z_c} \exp \frac{-(x_0 - \mu)^2}{2\sigma^2} \sum_{i=0}^N \delta(x_0 - X_0^i) \quad (21)$$

$$\text{where } \mu = \frac{x_t}{c_0} \quad \sigma = \frac{c_1}{c_0} \quad Z_c = \sum_{i=0}^N \exp \frac{-(X_0^i - \mu)^2}{2\sigma^2} \quad (22)$$

It can be seen that when $p(x_0)$ is a discrete distribution, $p(x_0|x_t)$ is also a discrete distribution, and the probability of each discrete value X_0^i is **inversely proportional to the distance between X_0^i and μ** .

3.2 Weighted Sum Degradation Phenomenon

In the previous subsection, we introduced the specific form of $p(x_0|x_t)$. Now, we further analyze the characteristics of the mean of $p(x_0|x_t)$. According to the definition of expectation, the mean of $p(x_0|x_t)$ can be expressed as:

$$\int x_0 p(x_0|x_t) dx_t = \frac{1}{Z_c} \sum_{i=0}^N X_0^i \exp \frac{-(X_0^i - \mu)^2}{2\sigma^2} \quad (23)$$

$$\text{where } \mu = \frac{x_t}{c_0} \quad \sigma = \frac{c_1}{c_0} \quad Z_c = \sum_{i=0}^N \exp \frac{-(X_0^i - \mu)^2}{2\sigma^2} \quad (24)$$

It can be observed that the mean of $p(x_0|x_t)$ is a weighted sum of all X_0^i samples, where the weight is inversely proportional to the distance between X_0^i and μ .

This weighted sum has a notable property: when only one sample is particularly close to μ while all other samples are relatively far, the weighted sum degrades into a single sample. This phenomenon becomes particularly apparent when the data is sparse.

Figure 1 presents a simple example. The blue points represent data samples, and the green circles indicate the standard deviation of the added Gaussian noise. It can be observed that when the samples are sparse and the radius of the Gaussian standard deviation is relatively small, a large number of X_t samples will be distributed only in the local region near the blue points. Consequently, the corresponding posterior probability $p(x_0|x_t)$ will collapse into a Dirac Delta distribution, meaning that $p(x_0|x_t)$ will have only a single nonzero value, located at the nearest blue point. As a result, the mean of $p(x_0|x_t)$ transitions from a weighted sum to a single sample. In this paper, we refer to this phenomenon as "weighted sum degradation".

"Weighted sum degradation" directly impacts whether the model can effectively learn the hidden data distribution.

Next, we analyze the degree of "weighted sum degradation" in two commonly used datasets for generative models: ImageNet-256 and ImageNet-512. These two datasets have relatively high dimensions, with pixel dimensions of 196,608 and 786,432, respectively. After compression using a Variational Autoencoder [KW+13; Rom+22], the latent space dimensions remain high, reaching 4,096 and 16,480, respectively. Since compression is a more commonly used approach, we will focus only on the compressed case in the following analysis.

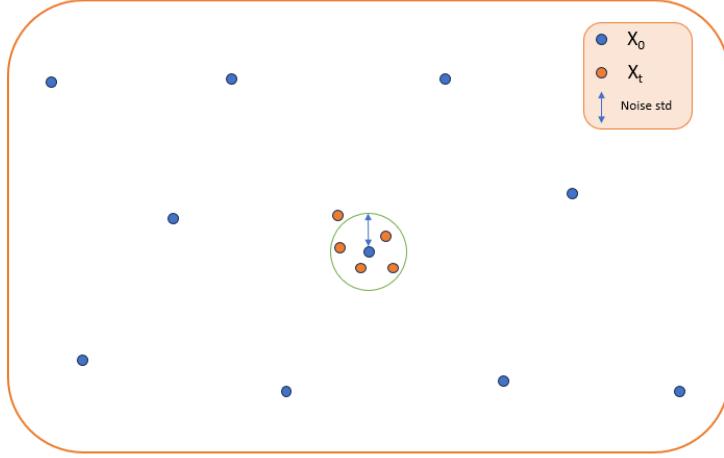


Figure 1: Impact of data sparsity on posterior probability distribution

Table 1: Statistics of ImageNet256(weighted sum degradation/weighted sum degradation to X_0)

merging\time	200	300	400	500	600	700	800	900
vp	1.00/1.00	1.00/1.00	1.00/0.98	0.91/0.57	0.41/0.01	0.02/0.00	0.00/0.00	0.00/0.00
flow	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/0.95	0.97/0.69	0.76/0.15	0.09/0.00

In order to quantitatively describe the degree of "weighted sum degradation", we define the following: A sample X_0 is drawn from $p(x_0|x_t)$, and a corresponding magnitude of random noise is mixed according to the value of t , resulting in X_t . This process ensures that $X_t \sim p(x_t)$. We then analyze the characteristics of the probability distribution $p(x_0|x_t = X_t)$. If there exists an $x_0 = X'_0$ such that the corresponding probability $p(x_0 = X'_0|x_t = X_t)$ is greater than 0.9, we consider this probability distribution to be approximately a Dirac Delta distribution, indicating the occurrence of "weighted sum degradation". If $X'_0 = X_0$, meaning that it degrades to the first-step sampled result X_0 , we refer to this as "weighted sum degradation to X_0 ".

It is evident that the sampling method used here is identical to the one used in actual training. This allows us to better determine whether the optimization process approximates $E_{p(x_0|x_t=X_t)}(x_0)$ and whether it degrades to X_0 .

As noted above, in addition to the sparsity of the data, the size of the noise standard deviation also affects "weighted sum degradation". Therefore, separate statistics are performed for different t values. Imagenet has 1000 classes, with each class sampled approximately 1000 times, so each t value is sampled approximately 10 million times. The proportions of "weighted sum degradation" and "weighted sum degradation to X_0 " are separately calculated. We analyze two different noise mixing modes: the Markov chain-based DDPM noise mixing mode (Equation (1)) and the Flow Matching mixing mode (Equation (2)).

Table 3.2 presents the specific statistical results for imagenet-256, and Table 3.2 presents the specific statistical results for imagenet-512. Several obvious patterns can be observed:

- As t decreases, the "weighted sum degradation" phenomenon becomes more pronounced.
- Flow Matching exhibits degeneration more frequently than VP.
- The higher the dimension, the greater the proportion of degradation.

Additionally, we can observe that regardless of whether it's VP or Flow Matching, **the degradation phenomenon is quite severe in both datasets**. When $t < 600$, the majority of means degenerate into a single sample. In fact, during the optimization process, due to the limitation on the number of samples, each $p(x_0|x_t = X_t)$ cannot be sufficiently sampled, so **the actual degradation ratio should be higher than the statistical results**.

Since high-dimensional data distributions are generally more complex, the corresponding $p(x_0|x_t = X_t)$ should also be more complex. When "weighted sum degradation" occurs, it is equivalent to using a single

Table 2: Statistics of ImageNet512(weighted sum degradation/weighted sum degradation to X_0)

merging\time	200	300	400	500	600	700	800	900
vp	1.00/1.00	1.00/1.00	1.00/0.98	0.98/0.57	0.87/0.08	0.50/0.00	0.03/0.00	0.00/0.00
flow	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/0.94	0.99/0.67	0.95/0.20	0.71/0.01

sample as the mean estimator, and this estimator typically has a large error. If we cannot provide an accurate fitting target, it can be inferred that the model is unlikely to learn the ideal target accurately. Therefore, it is necessary to reconsider whether diffusion models can truly learn the hidden probability distribution and whether they are operating according to the flow of probabilities.

3.3 A Simple Way to Understand the Objective Function

As seen in the previous section, the fitting objective of diffusion models in high-dimensional spaces exhibits a significant amount of "weighted sum degradation", with a large portion degrading to the result of the first step of ancestral sampling (i.e., X_0). Additionally, we know that the second step of ancestral sampling is a mixed noise process. Therefore, we can understand the objective function of high-dimensional diffusion models in a simple way: **predict the original signal from the noise-mixed signal**.

From the perspective of the frequency spectrum, we can further understand the principle [Die24].

In the frequency domain, the spectrum of a regular image signal is non-uniformly distributed, with low-frequency components occupying the majority and high-frequency components occupying a very small portion. Specifically, refer to Figure 2(a) and (b), where the lower half shows the spectrum. The center represents the low-frequency part, and the outer regions represent the high-frequency parts. It can be observed that the center is brighter, while the outer parts are darker. Gaussian noise signals, unlike regular signals, have a uniformly distributed spectrum, meaning all frequency components are the same. This can be seen in Figure 2(c) and (d), where the spectrum shows a uniformly bright plane.

Thus, if an image signal is mixed with random noise, **the signal-to-noise ratio of high-frequency components is always smaller than that of low-frequency components**. As the noise amplitude increases, the high-frequency components will be the first to be drowned out, and gradually, the low-frequency components will also be submerged. This process can be better understood with Figure 3.

When a neural network is trained to predict the original image signal from the noisy signal, due to the different signal-to-noise ratios of the frequency components, the fitting priority will differ for each frequency component. Let's divide the frequency components into two categories: one that is submerged and one that is not. For the **non-submerged** parts, the neural network will easily predict them and directly copy them to the output. For the **submerged** parts, the neural network will prioritize predicting the lower-frequency components for two reasons: lower-frequency components have a higher signal-to-noise ratio and are easier to predict; low-frequency components also have larger amplitudes, which effectively gives them more weight in the Euclidean distance loss function.

Thus, the objective function can be understood as "filtering higher-frequency components – completing the filtered frequency components", as shown in Figure 4. When t is large, noise overwhelms more frequency components, including not only high-frequency components but also some low-frequency components. At this point, the model prioritizes predicting low-frequency components. When t is small, noise only overwhelms high-frequency components, and the model directly predicts high-frequency components.

In the inference process, it is often observed that the early predictions of X_0 contain mostly contour information (when t is large), while later predictions of X_0 contain more detailed information (when t is small). This phenomenon confirms the analysis above.

4 A Unified Inference Framework-Natural Inference

We know that current diffusion model inference methods are designed with strict mathematical techniques, including ancestor sampling based on Markov Chains, inverse process solving based on stochastic differential equations, and so on. However, in the previous section, we understood that the objective function of the diffusion model can be interpreted in a simpler way — predicting the original image from a noisy image. Based on the principle of matching training and testing, we naturally have a question: can current inference methods also be understood in a similar, simpler way?

The answer is yes. Below, we will reveal that most of the current inference methods can be unified into a simple framework based on predicting x_0 , including ancestor sampling, DDIM, Euler, DPMsolver, DPMSolver++, DEIS, and Flow Matching solvers, among others.

Before introducing the new framework, let's first introduce a class of key operations contained in the new framework.

4.1 Self Guidance

First, following the concept of Classifier Free Guidance [HS22], we introduce a new operation called Self Guidance. The principle of Classifier Free Guidance can be summarized as follows:

$$I_{out} = I_{bad} + \lambda \cdot (I_{good} - I_{bad}) \quad (25)$$

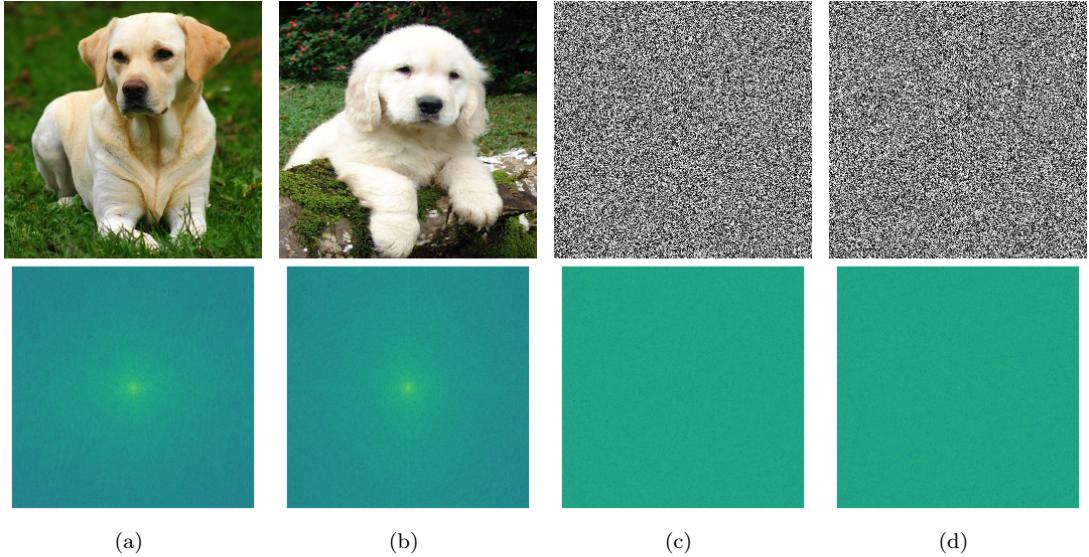


Figure 2: (a)(b) normal image and its frequency spectral (c)(d) gaussian noise and its frequency spectral

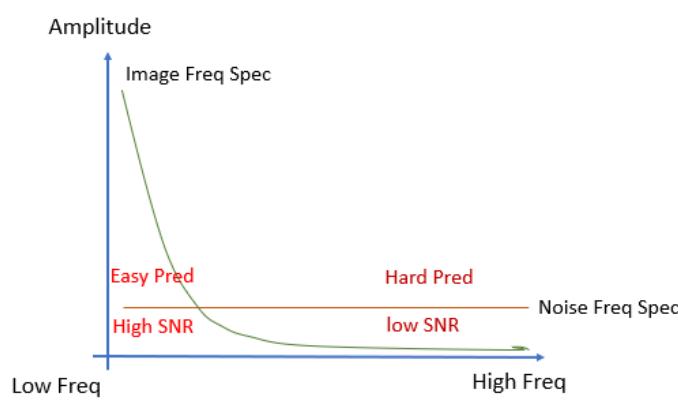


Figure 3: Image and noise frequency spectrum

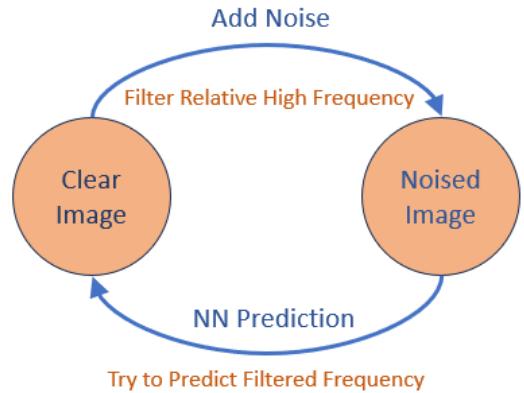


Figure 4: New perspective of training object function

Where I_{bad} is the output of a less capable model, I_{good} is the output of a more capable model, and both models share the same input. λ controls the degree of guidance.

In fact, Classifier Free Guidance is somewhat similar to Unsharp Masking algorithm in traditional image enhancing processing[GW17]. In Unsharp Masking algorithm, I_{good} is the original image, and I_{bad} is the image after Gaussian blur. The term $(I_{good} - I_{bad})$ provides the edge information, which, when added to the original image I_{good} , results in an image with sharper edges. Therefore, Classifier Free Guidance can also be considered as an **image enhancement** operation.

In the diffusion model inference process, a series of predicted x_0 are generated, where the quality of x_0 starts poor and improves over time. If an earlier x_0 is used as I_{bad} and a later x_0 is used as I_{good} , then in this paper, we refer to this operation as Self Guidance, because both I_{bad} and I_{good} are outputs of the same model, and no additional model is needed.

Based on the value of λ , we further classify Self Guidance as follows:

- When $\lambda > 1$, it is called **Fore Self Guidance**, where the output improves the quality.
- When $0 < \lambda < 1$, it is called **Mid Self Guidance**, where the output is a linear interpolation between I_{bad} and I_{good} , with a quality worse than I_{good} but better than I_{bad} .
- When $\lambda < 0$, it is called **Back Self Guidance**, where the output is not only worse than I_{good} , but also worse than I_{bad} .

Figure 5 provides two examples: (a) is the earlier model output ($t=540$), with poor quality (I_{bad}), (b) is the later model output ($t=500$), with better quality (I_{good}), (c) is the result of Fore Self Guidance, (d) is the result of Mid Self Guidance, and (e) is the result of Back Self Guidance.

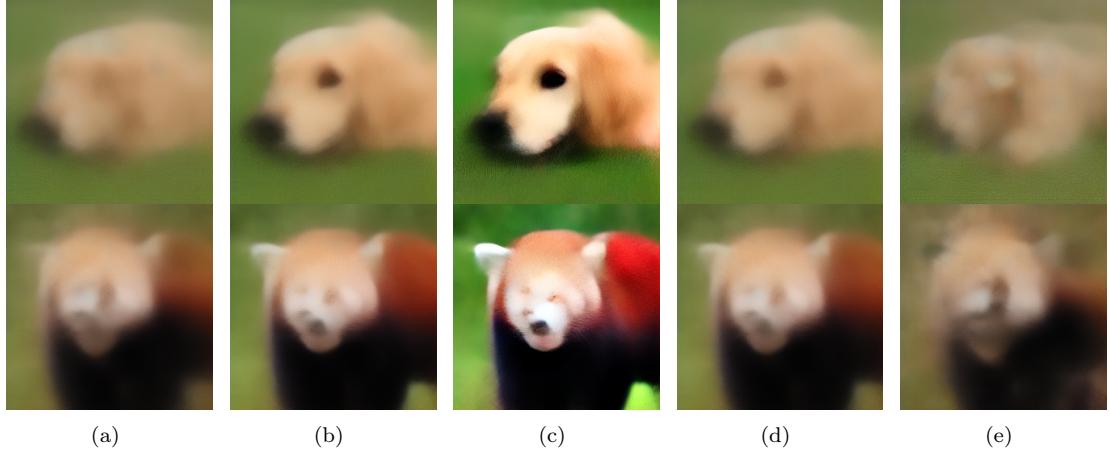


Figure 5: (a) Model output on $t=540$ (b) Model output on $t=500$ (c) Fore Self Guidance (d) Mid Self Guidance (e) Back Self Guidance

Equation (25) can be further written as

$$I_{out} = \lambda \cdot I_{good} + (1 - \lambda) \cdot I_{bad} \quad (26)$$

$$= C_{good} \cdot I_{good} + C_{bad} \cdot I_{bad} \quad (27)$$

$$\text{where } C_{good}, C_{bad} \in \text{real} \quad C_{good} + C_{bad} = 1 \quad (28)$$

As shown above, the coefficients of I_{bad} and I_{good} can take any value, but the sum of I_{bad} and I_{good} must equal 1. For **Fore Self Guidance**, $C_{good} > 0$, $C_{bad} < 0$; for **Mid Self Guidance**, $C_{good} > 0$, $C_{bad} > 0$; for **Back Self Guidance**, $C_{good} < 0$, $C_{bad} > 0$.

Thus, the following conclusion holds:

The linear combination of any two I_{bad} and I_{good} can be represented as Self Guidance with a scaling factor, i.e.

$$I_{out} = a \cdot I_{good} + b \cdot I_{bad} = (a + b) \cdot \left(\frac{a}{a + b} \cdot I_{good} + \frac{b}{a + b} \cdot I_{bad} \right) \quad (29)$$

Since the sum of the two coefficients equals 1, the operation inside the parentheses is a Self Guidance operation,

4.2 Natural Inference

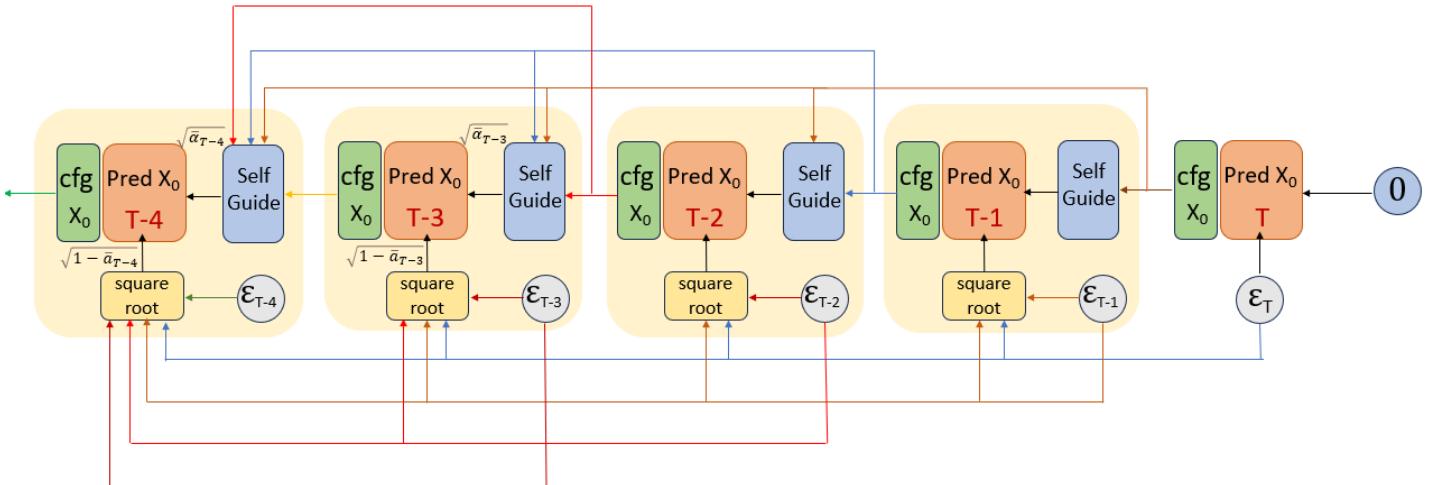


Figure 6: A new inference framework - Natural Inference

The idea of the new inference framework is shown in equation (30) and Figure 6, which includes T predictions of X_0 . The prediction starts with the model corresponding to the maximum t and then transitions sequentially

to smaller t values. Each prediction model includes two parts of input: the image signal input and the noise signal input, where the ratio of the two is kept the same as in the training phase's mixing ratio. The image signal is a linear weighted sum of outputs from previous models; the noise signal is a linear weighted sum of previous noise and the newly added noise. As mentioned in Section 4.1, the **linear weighted sum can be understood as a composition of multiple Self Guidance operations**. Since the noises are independent of each other, the amplitude of the total noise is calculated using the root of the sum of squares.

Since the input image signal of the current model depends on the output image signal of the previous model, this inference framework has a **autoregressive** form, except that the processing of the past image signals is quite simple, just a simple weighted sum.

$$\begin{array}{cc}
 \begin{array}{c} y_T & y_{T-1} & y_{T-2} & y_{T-3} \\ \hline x_{T-1} & c_{T-1}^T & 0 & 0 & 0 \\ x_{T-2} & c_{T-2}^T & c_{T-2}^{T-1} & 0 & 0 \\ x_{T-3} & c_{T-3}^T & c_{T-3}^{T-1} & c_{T-3}^{T-2} & 0 \\ x_{T-4} & c_{T-4}^T & c_{T-4}^{T-1} & c_{T-4}^{T-2} & c_{T-4}^{T-3} \end{array} & \text{Signal Coefficient Matrix} \\
 \begin{array}{c} \varepsilon_T & \varepsilon_{T-1} & \varepsilon_{T-2} & \varepsilon_{T-3} & \varepsilon_{T-4} \\ \hline b_{T-1}^T & b_{T-1}^{T-1} & 0 & 0 & 0 \\ b_{T-2}^T & b_{T-2}^{T-1} & b_{T-2}^{T-2} & 0 & 0 \\ b_{T-3}^T & b_{T-3}^{T-1} & b_{T-3}^{T-2} & b_{T-3}^{T-3} & 0 \\ b_{T-4}^T & b_{T-4}^{T-1} & b_{T-4}^{T-2} & b_{T-4}^{T-3} & b_{T-4}^{T-4} \end{array} & \text{Noise Coefficient Matrix}
 \end{array}$$

Figure 7: Coefficient matrix on the natural inference framework

$$\begin{aligned}
 y_T &= f_T(\varepsilon_T) \\
 y_{T-1} &= f_{T-1}(N_{T-1} + S_{T-1}) \\
 \text{where } N_{T-1} &= b_{T-1}^{T-1}\varepsilon_{T-1} + b_{T-1}^T\varepsilon_T & S_{T-1} &= c_{T-1}^T y_T \\
 \sqrt{(b_{T-1}^{T-1})^2 + (b_{T-1}^T)^2} &= \sqrt{1 - \bar{\alpha}_{T-1}} & c_{T-1}^T &= \sqrt{\bar{\alpha}_{T-1}} \\
 &\dots && \\
 y_t &= f_t(N_t + S_t) & (30) \\
 \text{where } N_t &= \sum_{i=t}^T b_t^i \varepsilon_t & S_t &= \sum_{i=t+1}^T c_t^i y_i \\
 \sqrt{\sum_{i=t}^T (b_t^i)^2} &= \sqrt{1 - \bar{\alpha}_t} & \sum_{i=t+1}^T c_t^i &= \sqrt{\bar{\alpha}_t} \\
 &\dots
 \end{aligned}$$

Here, f_t is the model function predicting x_0 at step t , y_t is the output of the f_t model function, i.e., the x_0 predicted at step t , ε_t is the independent standard Gaussian noise, and b_t^i and c_t^i are constant coefficients. The sum of independent Gaussian noises is also Gaussian noise, and its standard deviation equals the square root of the sum of the squares of the original standard deviations.

In this paper, $\sqrt{\bar{\alpha}_t}$ is called the **signal marginal coefficient**, $\sqrt{1 - \bar{\alpha}_t}$ is called the **marginal noise coefficient**, $\sum_{i=t+1}^T c_t^i$ is called the **equivalent marginal signal coefficient**, and $\sqrt{\sum_{i=t}^T (b_t^i)^2}$ is called the **equivalent marginal noise coefficient**. To facilitate the presentation of all c_t^i and b_t^i , we organize all the coefficients into a matrix form, as shown in Figure 7, and this matrix is called the coefficient matrix. As can be seen, due to the autoregressive nature, the signal coefficient matrix takes the form of a lower triangular matrix.

4.3 Represent DDPM Ancestral Sampling with Natural Inference Framework

Next, we will reveal that the ancestor sampling algorithm of DDPM can be equivalently transformed into the form of the Natural Inference framework. The iterative process of the ancestor sampling algorithm is as follows:

$$\begin{aligned}
 y_t &= f_t(x_t) \\
 x_{t-1} &= d_{t-1} \cdot x_t + e_{t-1} \cdot y_t + g_{t-1} \cdot \varepsilon_{t-1} \\
 \text{where } d_{t-1} &= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} & e_{t-1} &= \frac{\sqrt{\alpha_{t-1}}\beta_t}{1 - \bar{\alpha}_t} & g_{t-1} &= \sqrt{\frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}}\beta_t & (31)
 \end{aligned}$$

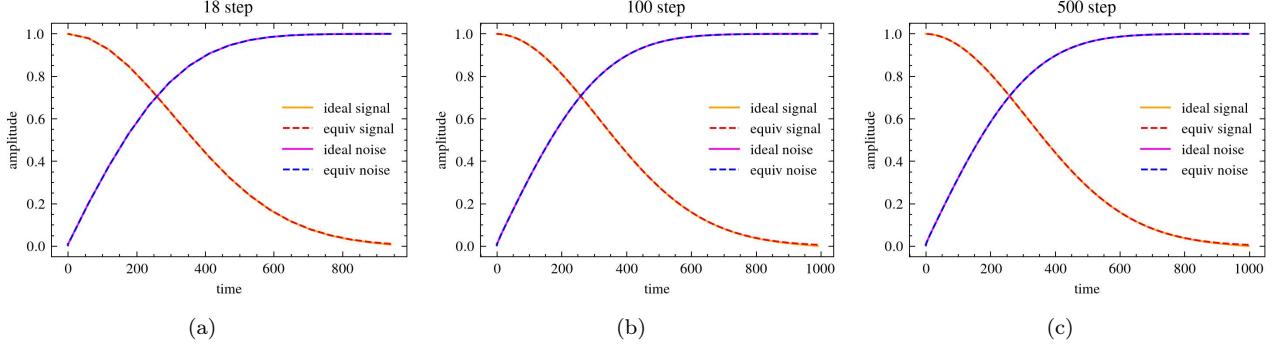


Figure 8: DDPM equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 100 step (c) 500 step

Here, f_t is the model function at the t -th step. In this case, we assume the model predicts x_0 , but other forms of prediction models can be transformed into the form of predicting x_0 . y_t is the output of f_t , which is the predicted x_0 at the t -th step.

According to the above iterative algorithm, x_{T-1} can be expressed as

$$\begin{aligned} x_T &= g_T \cdot \varepsilon_T \quad \text{where } g_T = 1 \\ x_{T-1} &= d_{T-1} \cdot x_T + e_{T-1} \cdot y_T + g_{T-1} \cdot \varepsilon_{T-1} \\ &= e_{T-1} y_T + (d_{T-1} g_T \varepsilon_T + g_{T-1} \varepsilon_{T-1}) \end{aligned} \quad (32)$$

Based on the result of x_{T-1} , the result of x_{T-2} can be written as

$$\begin{aligned} x_{T-2} &= d_{T-2} \cdot x_{T-1} + e_{T-2} \cdot y_{T-1} + g_{T-2} \cdot \varepsilon_{T-2} \\ &= (d_{T-2} e_{T-1} y_T + e_{T-2} y_{T-1}) + (d_{T-2} d_{T-1} g_T \varepsilon_T + d_{T-2} g_{T-1} \varepsilon_{T-1} + g_{T-2} \varepsilon_{T-2}) \end{aligned} \quad (33)$$

Based on the result of x_{T-2} , the result of x_{T-3} can be further written as

$$\begin{aligned} x_{T-3} &= d_{T-3} \cdot x_{T-2} + e_{T-3} \cdot y_{T-2} + g_{T-3} \cdot \varepsilon_{T-3} \\ &= (d_{T-3} d_{T-2} e_{T-1} y_T + d_{T-3} e_{T-2} y_{T-1} + e_{T-3} y_{T-2}) \\ &\quad + (d_{T-3} d_{T-2} d_{T-1} g_T \varepsilon_T + d_{T-3} d_{T-2} g_{T-1} \varepsilon_{T-1} + d_{T-3} g_{T-2} \varepsilon_{T-2} + g_{T-3} \varepsilon_{T-3}) \end{aligned} \quad (34)$$

Similarly, each x_t can be recursively written in a similar form. It can be observed that each x_t can be decomposed into two parts: one part is a weighted sum of past predictions of x_0 (i.e., y_t), and the other part is a weighted sum of past noise and newly added noise. Since d_t , e_t , and g_t are all known constants, the equivalent signal coefficient and equivalent noise coefficient for each x_t can be accurately computed.

The computation results show that the equivalent signal coefficient of each x_t is almost equal to $\sqrt{\bar{\alpha}_t}$, and the equivalent noise coefficient is approximately $\sqrt{1 - \bar{\alpha}_t}$. Moreover, the slight error diminishes as the number of sampling steps T increases. Specifically, Figure 8 illustrates the results for 18 steps, 100 steps, and 500 steps.

Table 4 presents the complete coefficients of each x_t with respect to y_t in matrix form, where each row corresponds to an x_t . Table 5 provides the complete coefficients of each x_t with respect to ε_t in matrix form. It can be seen that the noise coefficient matrix differs slightly from the signal coefficient matrix, with an additional nonzero coefficient appearing to the right of the diagonal elements. This indicates that a small amount of new noise is introduced at each step, causing the overall noise "pattern" to change at a slow rate.

At this point, we have successfully demonstrated that the DDPM ancestral sampling process can be represented using the Natural Inference framework.

4.4 Represent DDIM with Natural Inference Framework

The iterative rule of the DDIM sampling algorithm can be expressed in the following form:

$$\begin{aligned} y_t &= f_t(x_t) \\ x_{t-1} &= \sqrt{\bar{\alpha}_{t-1}} \cdot y_t + \sqrt{1 - \bar{\alpha}_{t-1}} \cdot \frac{x_t - \sqrt{\bar{\alpha}_t} y_t}{\sqrt{1 - \bar{\alpha}_t}} \cdot y_t \\ &= d_{t-1} \cdot x_t + e_{t-1} \cdot y_t \\ \text{where } d_{t-1} &= \frac{\sqrt{1 - \bar{\alpha}_{t-1}}}{\sqrt{1 - \bar{\alpha}_t}} \quad e_{t-1} = (\sqrt{\bar{\alpha}_{t-1}} - \frac{\sqrt{1 - \bar{\alpha}_{t-1}}}{\sqrt{1 - \bar{\alpha}_t} \sqrt{\bar{\alpha}_t}}) \end{aligned} \quad (35)$$

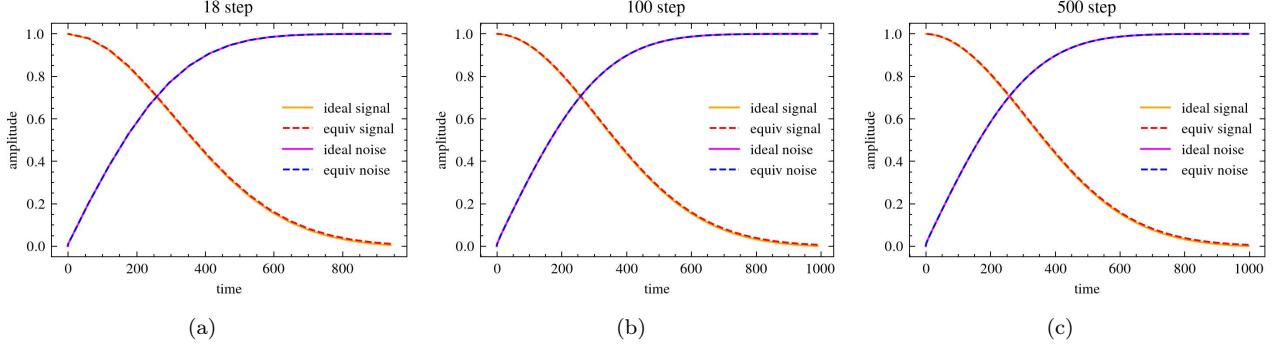


Figure 9: DDIM equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 100 step (c) 500 step

It can be seen that the update formula of DDIM is similar to the update formula of DDPM ancestral sampling, except that the term $g_{t-1} \cdot \varepsilon_{t-1}$ is missing, meaning that no new noise is added at each step. Following the recursive method of DDPM ancestral sampling, each x_t corresponding to t can also be written in a similar form, as follows:

$$\begin{aligned}
x_T &= g_T \cdot \varepsilon_T \quad \text{where } g_T = 1 \\
x_{T-1} &= e_{T-1} y_T + d_{T-1} g_T \varepsilon_T \\
x_{T-2} &= (d_{T-2} e_{T-1} y_T + e_{T-2} y_{T-1}) + d_{T-2} d_{T-1} g_T \varepsilon_T \\
x_{T-3} &= (d_{T-3} d_{T-2} e_{T-1} y_T + d_{T-3} e_{T-2} y_{T-1} + e_{T-3} y_{T-2}) \\
&\quad + d_{T-3} d_{T-2} d_{T-1} g_T \varepsilon_T
\end{aligned} \tag{36}$$

It can be seen that the form of DDIM is slightly different from DDPM. Since DDIM does not introduce new noise at each step, there is only one noise term.

The computation results show that the equivalent signal coefficients of each x_t are approximately equal to $\sqrt{\bar{\alpha}_t}$, and the equivalent noise coefficient contains only the term related to ε_T , whose coefficient is almost equal to $\sqrt{1 - \bar{\alpha}_t}$. Figure 9 illustrates the results for 18 steps, 100 steps, and 500 steps, respectively. It can be observed that the errors in the equivalent coefficients are minimal and almost indistinguishable. Table 6 presents the complete signal coefficient matrix for 18 steps.

Therefore, the sampling process of DDIM can also be represented using the Natural Inference framework.

4.5 Represent Flow Matching Euler Sampling with Natural Inference Framework

The noise mixing method of Flow Matching is shown in Equation (2). When using Euler discretized integral sampling, its update rule can be expressed as follows:

$$\begin{aligned}
y_i &= f_i(x_i) \\
x_{i-1} &= x_i + (t_{i-1} - t_i)(-y_i + \epsilon) \\
&= x_i + (t_{i-1} - t_i) \frac{x_i - y_i}{t_i} \\
&= d_{i-1} \cdot x_i + e_{i-1} \cdot y_i \\
\text{where } d_{i-1} &= \frac{t_{i-1}}{t_{t_i}} \quad e = (1 - \frac{t_{i-1}}{t_{t_i}})
\end{aligned} \tag{37}$$

where f is the model predicting x_0 , and y_i is the output value of the model f_i corresponding to the discrete point t_i .

It can be observed that the recursive rule of the Euler algorithm in Flow Matching is similar to that of DDIM, so each x_i can also be expressed in a similar form.

The computation results show that for each discrete point t_i , the equivalent signal coefficient of x_i is **exactly equal to** $1 - t_i$, and the equivalent noise coefficient has only the ε_N term, whose coefficient is **exactly equal to** t_i . The specific results can be seen in Figure 10, which shows the results for 18 steps, 200 steps, and 500 steps, respectively. Table 7 presents the signal coefficient matrix for 18 steps.

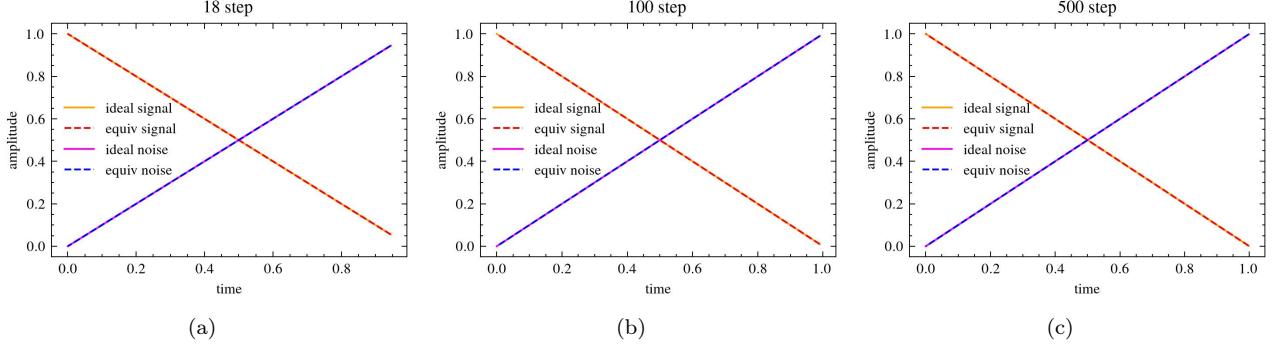


Figure 10: Flow matching euler sampler equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 100 step (c) 500 step

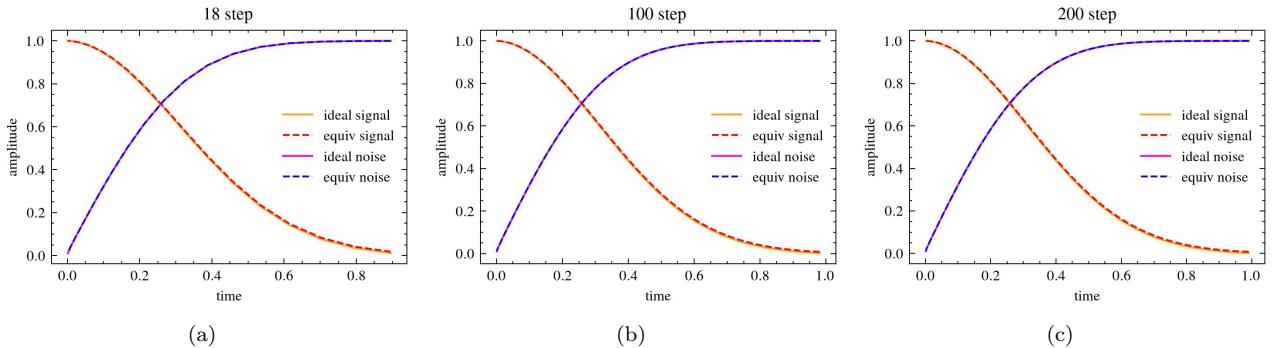


Figure 11: DEIS equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 100 step (c) 500 step

4.6 Represent High Order Samplers with Natural Inference Framework

In the previous sections, most first-order sampling algorithms have already been represented using the Natural Inference framework. For second-order and higher-order sampling algorithms, since the update rules of x_i are more complex, directly expressing the form of each x_i using the aforementioned derivation method becomes difficult. Therefore, alternative solutions must be sought. Symbolic computation tools provide a suitable solution to this challenge, as they can automatically analyze complex mathematical expressions. With slight modifications to the original algorithm code, they can automatically compute the coefficients of each y_i term and ε_i term. The toolkit used in this paper is SymPy, and For specific details, please refer to the code attached in this paper.

Experimental calculations reveal that DEIS, DPMsolver, and DPMsolver++ yield the same conclusion as DDIM: each x_i can be decomposed into two parts, with its equivalent signal coefficient approximately equal to $\sqrt{\bar{\alpha}_i}$ and its equivalent noise coefficient approximately equal to $\sqrt{1 - \bar{\alpha}_i}$.

Figure 11 shows the results for the DEIS(tab3) algorithm, Figure 12 presents the results for the third-order DPMsolver, and Figure 13 illustrates the results for the second-order DPMsolver++. It can be observed that these higher-order sampling algorithms exhibit the same properties and can also be represented using the Natural Inference framework.

Table 8 provides the coefficient matrix for the third-order DEIS algorithm (18 steps). Table 9 and Table 10 present the coefficient matrices for the second-order and third-order DPMsolver algorithms (18 steps). Table 11 and Table 12 provide the coefficient matrices for the second-order and third-order DPMsolver++ algorithms (18 steps).

5 Application of the Natural Inference Framework

Thus, we have used a completely new perspective to explain high-dimensional diffusion models, including the objective function during training and the inference algorithm during testing. This new perspective has several advantages:

- The new perspective maintains training-testing consistency, where the goal during training is to predict x_0 , and the goal during testing is also to predict x_0 . This consistency helps in analyzing the model's issues

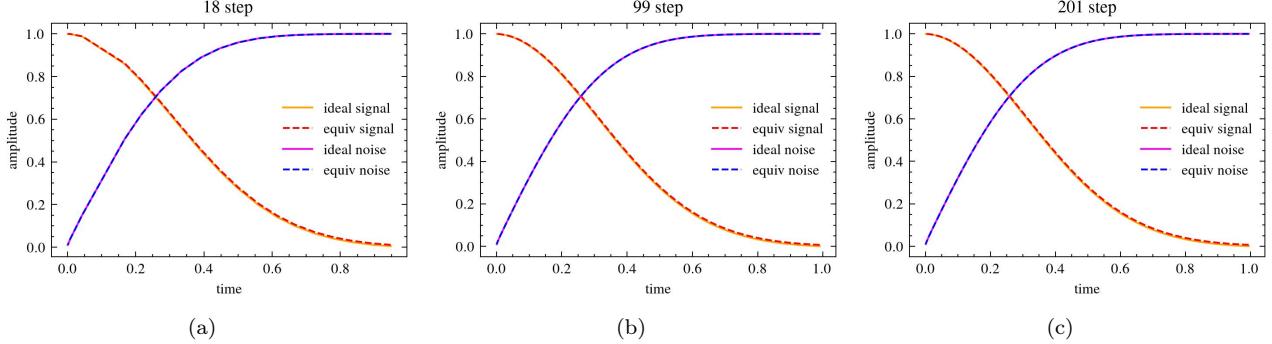


Figure 12: dpm solver 3s equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 99 step (c) 201 step

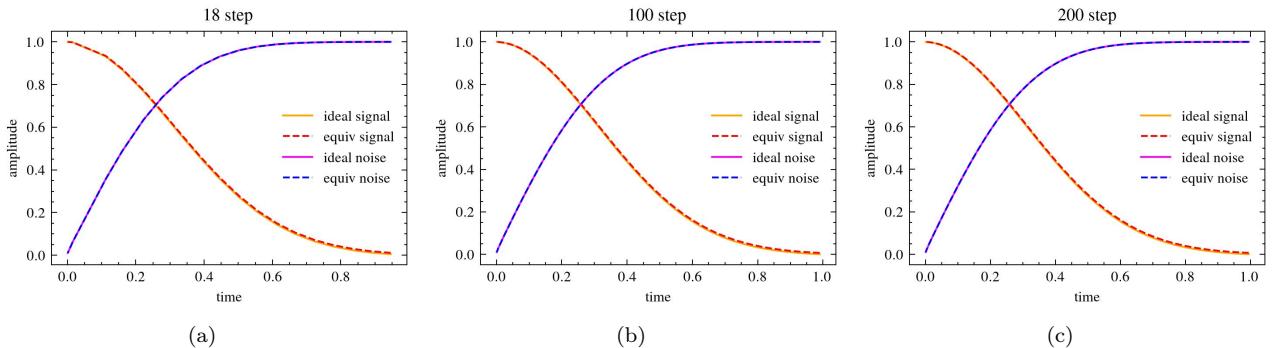


Figure 13: dpm solver ++ 2s equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 99 step (c) 201 step

and designing solutions.

- The new perspective divides the inference process into a series of operations for predicting x_0 , each of which has clear input image signals and output image signals. This makes the inference process more visual and interpretable, providing significant help for debugging and problem analysis. Figures 18 and 19 provide a visualization of the complete inference process
- Under this new perspective, the current sampling algorithms are just specific parameter configurations of the Natural Inference framework, and they are not optimal. In the following chapters, we will see that with careful design, better parameter configurations can be found. Additionally, the new framework also provides an extra degree of freedom. By controlling the degree of Self Guidance, the sharpness of the generated image can be flexibly controlled.

5.1 Why can High Order Samplings Speedup Sampling ?

In this section, we will explain, based on the Natural Inference framework, why algorithms like DEIS and DPMsolver perform better than DDPM and DDIM under low sampling steps.

Tables 4 and 6 show the coefficient matrices for DDPM and DDIM under the Natural Inference framework, while the tables for DEIS and DPMSolver show their coefficient matrices. A careful comparison reveals a significant difference: The coefficients in DEIS and DPMSolver contain negative values, which, according to the analysis in 4.1, can be considered as Fore Self Guidance operations, which can enhance image quality. In contrast, the coefficients in DDPM and DDIM are all positive, corresponding to Middle Self Guidance, which can not enhance image quality. Therefore, the key to the better performance of higher-order sampling algorithms lies in the design of Self Guidance.

5.2 Why can Low order Sampling Adapt to Big CFG ?

We know that, with a larger cfg, most higher-order sampling algorithms, including 3rd-order DEIS and DPMSolver, exhibit a similar over-exposure problem, while DDIM and 2nd-order DPMSolver++ can alleviate this issue. So why do they work, while others do not? By comparing their coefficient matrices under the Natural

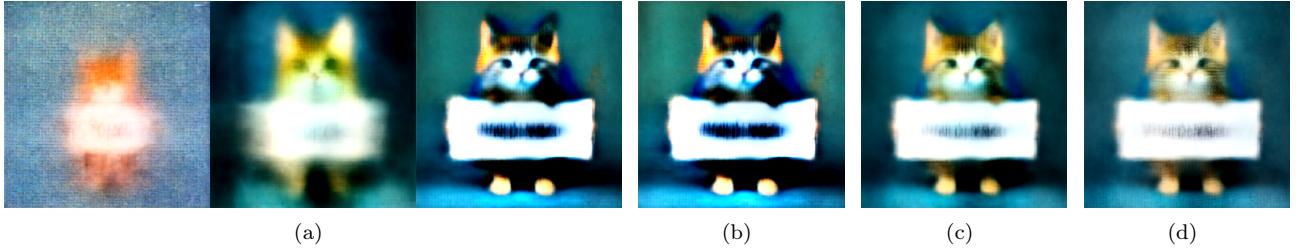


Figure 14: (a) the first three output(x_0^0, x_0^1, x_0^2) (b) $0.00 \cdot x_0^0 - 0.17 \cdot x_0^1 + 1.17 \cdot x_0^2$ (c) $0.00 \cdot x_0^0 + 0.49 \cdot x_0^1 + 0.51 \cdot x_0^2$ (d) $0.32 \cdot x_0^0 + 0.33 \cdot x_0^1 + 0.35 \cdot x_0^2$

Inference framework, we can identify the core of the problem. Tables 6 and 11 show the coefficient matrices for DDIM and 2nd-order DPMSolver++, while tables 8 and 10 show the coefficient matrices for 3rd-order DEIS and DPMSolver. A comparison reveals that the coefficients of 2nd-order DPMSolver++ and DDIM are all positive, representing a composite of Middle Self Guidance operations, while 3rd-order DEIS and DPMSolver contain negative values, indicating the presence of Fore Self Guidance operations. Therefore, Middle Self Guidance (positive coefficients) is key to handling larger cfgs.

Figure 14 presents a small experiment demonstrating that, with a larger cfg, the output from Mid Self Guidance significantly outperforms Fore Self Guidance. Figure 14(a) shows the outputs for the first three steps of SD3 (cfg=7), and Figures 14(b)~(d) show results from different linear combinations. Figure 14(b) shows the result with negative coefficients, Figure 14(d) shows the result with two positive coefficients, and Figure 14(d) shows the result with three positive coefficients. It is clear that when there are more positive coefficients, the image quality is better.

Furthermore, we know that, compared to DDIM, 2nd-order DPMSolver++ not only adapts better to larger cfgs but also provides relatively better acceleration. This can also be explained by the coefficient matrix: compared to DDIM, the coefficient matrix of 2nd-order DPMSolver++ has more zero elements, with more weight placed on the diagonal elements, i.e., the most recent outputs of the current step, leading to better acceleration.

5.3 A Way to Control Image Sharpness

As mentioned in section 4.1, different Self Guidance operations affect the sharpness of the generated image. Therefore, by reasonably designing the weight distribution of the coefficient matrix, we can control the sharpness of the generated image. Below is an example with SD3. Table 13 shows the coefficient matrix for the original Euler method, and Table 14 shows the adjusted coefficient matrix. It can be observed that the Self Guidance operations in the Euler method are mostly composite operations of Mid Self Guidance, with a significant amount of weight placed on earlier outputs. In the adjusted coefficient matrix, the weights for earlier outputs are set to zero, and more weight is placed on the more recent outputs. As a result, the adjusted coefficient matrix produces sharper images, as shown in Figure 15.

Note that although the adjusted coefficient matrix minimizes the weight of earlier outputs, each row still retains at least three non-zero elements. This is primarily to accommodate the influence of larger cfgs, especially for the earlier steps.

5.4 Better Coefficient Matrix

As analyzed in section 4, the current mainstream sampling algorithms correspond only to a coefficient matrix of the Natural Inference framework. So, are there more efficient coefficient matrices? The answer is yes. Below, we will take the pre-trained model on the CIFAR10 dataset as an example to demonstrate that there exist more efficient coefficient matrices that can achieve higher FID scores under the same number of steps. The pre-trained model uses the ScoreSDE [Son+20] released cifar10_ddpmpp_continuous(vp).

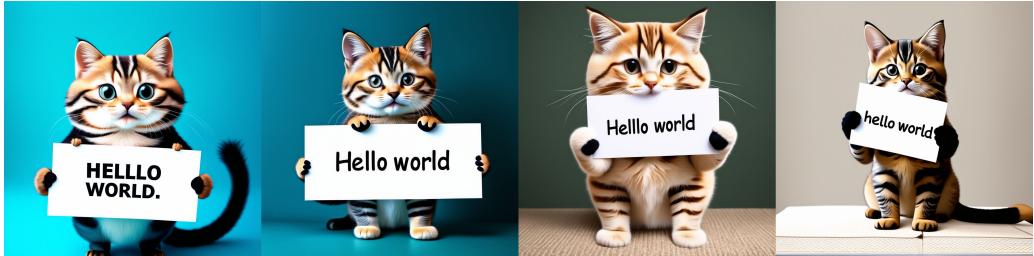
Table 15 shows an optimized coefficient matrix for 5 steps, Table 16 shows an optimized coefficient matrix for 10 steps, and Table 17 shows an optimized coefficient matrix for 15 steps. Table 3 compares the optimized coefficient matrices with DEIS, DPMSolver, and DPMSolver++. It can be observed that, under the same number of steps, the optimized coefficient matrices achieve better FID scores than DEIS, DPMSolver, and DPMSolver++. All algorithms use the same time discretization schedule, and for DEIS, DPMSolver, and DPMSolver++, all hyperparameters are tested to select the lowest FID as the result.

The core ideas behind the coefficient matrix are as follows:

- The weight distribution tends to prioritize outputs closer to the current time step, reducing the weight of



(a)



(b)

Figure 15: (a) Euler sampling result (d) Result with adjusted coefficient matrix

earlier outputs. In other words, *reducing the tail length*. Generally, when the number of sampling steps is larger, the *tail* should be extended appropriately to avoid the "over-enhancement" phenomenon(see section 5.5); when the number of sampling steps is smaller, the *tail* can be shortened.

- When the number of steps is small, an appropriate increase in Fore Self Guidance is necessary, meaning adding larger negative coefficients before the diagonal elements.

Table 3: Comparison of different algorithm (measured with FID)

step\method	DEIS	DPMsolver	DPMsolver++	optimized matrix
5 step	15.60	13.16	11.50	8.66
10 step	3.94	5.06	4.64	3.58
15 step	3.55	4.15	3.92	2.93

5.5 Over Enhancement Phenomenon

This subsection introduces a phenomenon called "over-enhancement," which influences the design of the coefficient matrix.

In section 3.3, we know that fitting x_0 essentially compensates for the high-frequency components that are drowned out by noise, thereby increasing the details in the input image. Therefore, the model can be seen as an image quality enhancement operator. Additionally, from the previous analysis, we also know that Self Guidance operations can enhance the image quality. When too many *enhancement* operations are applied, the *over-enhancement* phenomenon becomes more likely. For example, frequent and dense model enhancements (where the sampling time points are close together, and the time intervals are small), or strong Self Guidance operations.

This phenomenon can be intuitively demonstrated through the following procedure: Given a normal input image, fix the model time points t and the noise, and continuously apply model enhancements multiple times. The *over-enhancement* phenomenon gradually becomes apparent. With t fixed and the interval being zero, this satisfies the requirement for density. The model output image is directly used as the input image for the next iteration, which can be seen as Self Guidance without *tail* and $\lambda = 1$ with a large intensity. The specific effects can be seen in the process in Figure 16. The first row shows the original image, and the second row shows the result after the first model application. We can observe that the image quality has decreased, mainly because some high-frequency components are drowned out by the noise, leading to a low signal-to-noise ratio, which makes it difficult for the model to predict. Starting from the fourth row, the *over-enhancement* phenomenon gradually appears and becomes more severe. Additionally, it can be seen that at different time points t , the *over-enhancement* phenomenon has different characteristics. When t is small, the image has a prominent "grainy"

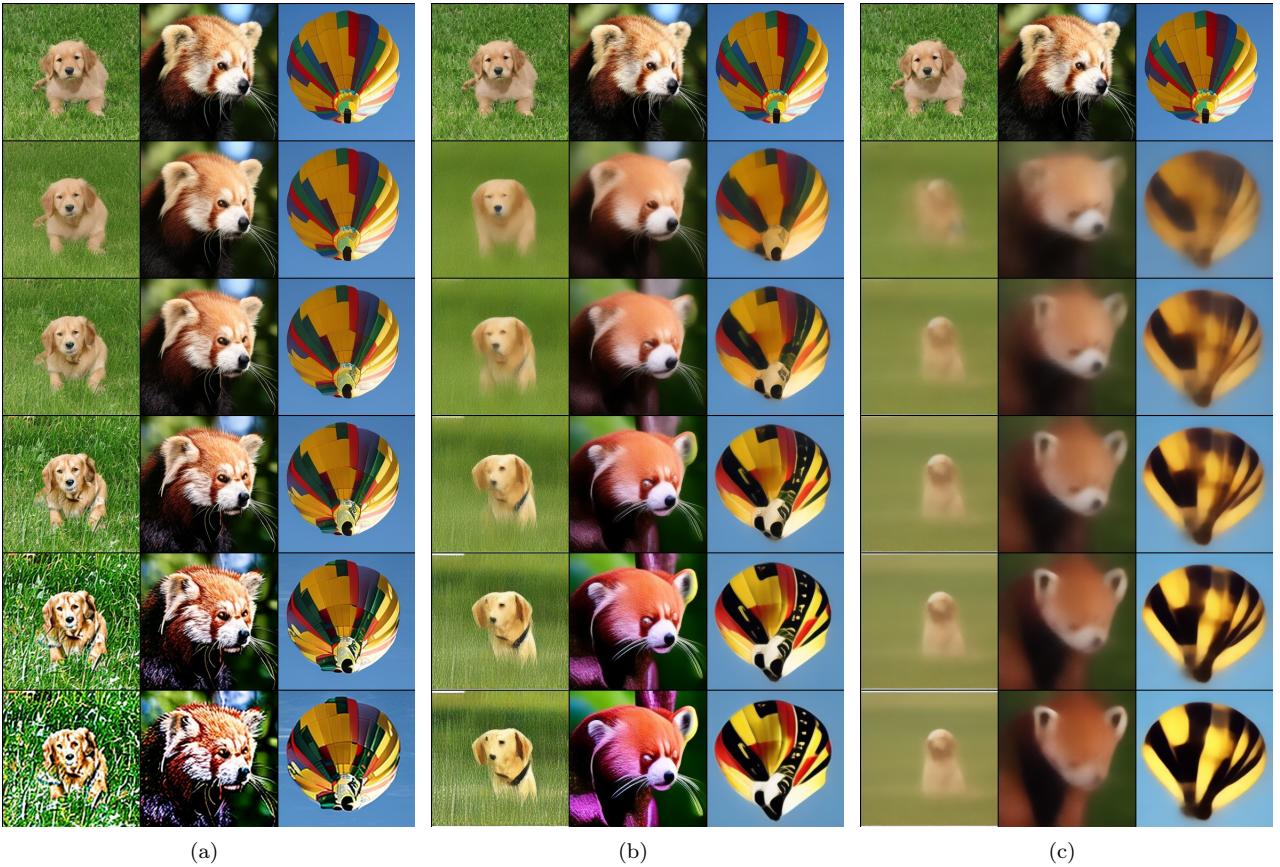


Figure 16: over enhancing phenomenon on ddpm latent model trained on imageget-256 (a) on $t=100$ (b) $t=300$ (c) $t=500$

appearance, with more high-frequency information, while at larger t , the image becomes simplified and lacks details.

This phenomenon may be related to the **train-test mismatch** of the model input domain. During training, the input images are normal, containing complete frequency components, while during inference, the input images are not complete frequency components. Early in the inference process, the model usually only has low-frequency components, and as t increases, higher-frequency components gradually appear. Therefore, there is a certain difference between these two data domains.

A simple experiment can be conducted to verify whether this analysis is reasonable. For a model that has clearly shown the *over-enhancement* phenomenon, mix some data similar to the second and third rows of Figure 16 into the training data, fine-tune the model, and then generate over-enhanced images in the same way. The results are shown in Figure 17. It can be seen that the image quality has significantly improved: when t is small, the graininess has notably reduced, and when t is large, more details are preserved.

5.6 Limitations and Directions for Improvement

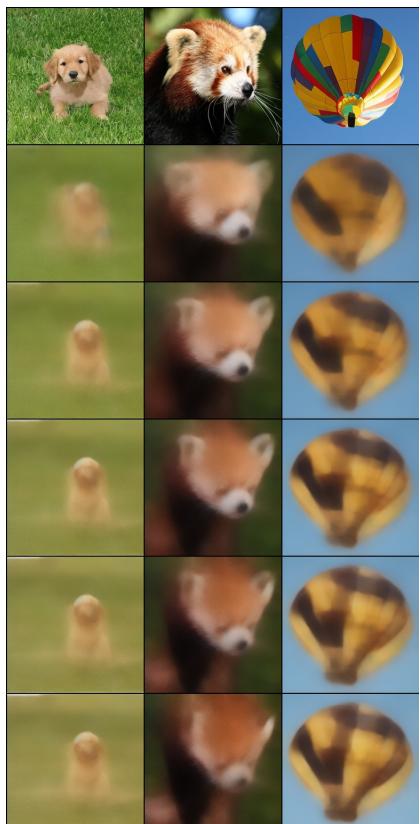
Although there are more optimal coefficient matrices than the existing sampling methods, the optimal coefficient matrix does not have a fixed form. When the model, number of steps, and time discretization strategy change, the coefficient matrix may also need to be adjusted accordingly. When the number of steps is large, the number of adjustable parameters in the coefficient matrix increases, which can pose a challenge for manual adjustments. One automated solution is to use hyperparameter optimization methods to automatically search for the best configuration [YS20]; another solution is to replace the Self Guidance operation (linear weighted sum) with a neural network, which can be optimized separately or jointly with the original model parameters. For the joint optimization approach, the model will take on an **autoregressive** form, whether during the training phase or the inference phase.



(a)



(b)



(c)

Figure 17: improved over enhancing phenomenon after finetuning with model output images (a) on $t=100$ (b) $t=300$ (c) $t=500$

6 Conclusion

This paper mainly attempts to analyze the working principles of high-dimensional diffusion models. Starting from the optimization objective function, we analyze the impact of high-dimensional data sparsity on the objective function. Based on this, we provide a new perspective to understand the objective function. At the same time, this paper presents a new inference framework. The new framework not only unifies existing mainstream inference methods but also aligns with the new perspective of the objective function. Additionally, the new inference framework can effectively explain certain phenomena and guide the design of more efficient inference methods. It is hoped that this paper will inspire the community to rethink the working principles of high-dimensional diffusion models and further improve training and inference methods.

References

- [And82] Brian DO Anderson. “Reverse-time diffusion equation models”. In: *Stochastic Processes and their Applications* 12.3 (1982), pp. 313–326.
- [Ban+23] Arpit Bansal et al. “Cold diffusion: Inverting arbitrary image transforms without noise”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 41259–41282.
- [Bao+22a] Fan Bao et al. “Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models”. In: *arXiv preprint arXiv:2201.06503* (2022).
- [Bao+22b] Fan Bao et al. “Estimating the optimal covariance with imperfect mean in diffusion probabilistic models”. In: *arXiv preprint arXiv:2206.07309* (2022).
- [Bao+22c] Fan Bao et al. “Why are conditional generative models better than unconditional ones?” In: *arXiv preprint arXiv:2212.00362* (2022).
- [Bao+23] Fan Bao et al. “All are worth words: A vit backbone for diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 22669–22679.
- [Die24] Sander Dieleman. *Diffusion is spectral autoregression*. 2024. URL: <https://sander.ai/2024/09/02/spectral-autoregression.html>.
- [DN21] Prafulla Dhariwal and Alexander Nichol. “Diffusion models beat gans on image synthesis”. In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.
- [Ess+24] Patrick Esser et al. “Scaling rectified flow transformers for high-resolution image synthesis”. In: *Forty-first international conference on machine learning*. 2024.
- [GW17] Rafael Gonzalez and Richard Woods. *Digital image processing, 4th Edition*. Pearson education india, 2017. Chap. 3.6, pp. 182–183.
- [HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.
- [HS22] Jonathan Ho and Tim Salimans. “Classifier-free diffusion guidance”. In: *arXiv preprint arXiv:2207.12598* (2022).
- [KW+13] Diederik P Kingma, Max Welling, et al. *Auto-encoding variational bayes*. 2013.
- [LGL22] Xingchao Liu, Chengyue Gong, and Qiang Liu. “Flow straight and fast: Learning to generate and transfer data with rectified flow”. In: *arXiv preprint arXiv:2209.03003* (2022).
- [Lip+22] Yaron Lipman et al. “Flow matching for generative modeling”. In: *arXiv preprint arXiv:2210.02747* (2022).
- [Lu+22a] Cheng Lu et al. “Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 5775–5787.
- [Lu+22b] Cheng Lu et al. “Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models”. In: *arXiv preprint arXiv:2211.01095* (2022).
- [ND21] Alexander Quinn Nichol and Prafulla Dhariwal. “Improved denoising diffusion probabilistic models”. In: *International conference on machine learning*. PMLR. 2021, pp. 8162–8171.
- [Pod+23] Dustin Podell et al. “Sdxl: Improving latent diffusion models for high-resolution image synthesis”. In: *arXiv preprint arXiv:2307.01952* (2023).
- [PX23] William Peebles and Saining Xie. “Scalable diffusion models with transformers”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 4195–4205.
- [Rad+21] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

- [Rom+22] Robin Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.
- [Ros10] Sheldon Ross. “A first course in probability”. In: (2010).
- [SD+15] Jascha Sohl-Dickstein et al. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International conference on machine learning*. pmlr. 2015, pp. 2256–2265.
- [SME20] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising diffusion implicit models”. In: *arXiv preprint arXiv:2010.02502* (2020).
- [Son+20] Yang Song et al. “Score-based generative modeling through stochastic differential equations”. In: *arXiv preprint arXiv:2011.13456* (2020).
- [Vin11] Pascal Vincent. “A connection between score matching and denoising autoencoders”. In: *Neural computation* 23.7 (2011), pp. 1661–1674.
- [YS20] Li Yang and Abdallah Shami. “On hyperparameter optimization of machine learning algorithms: Theory and practice”. In: *Neurocomputing* 415 (2020), pp. 295–316.
- [ZC22] Qinsheng Zhang and Yongxin Chen. “Fast sampling of diffusion models with exponential integrator”. In: *arXiv preprint arXiv:2204.13902* (2022).
- [Zhe23] zhenxin Zheng. *The Art of DPM*. 2023. URL: https://github.com/blairstar/The_Art_of_DPM.
- [Zhe24] zhenxin Zheng. *Understanding Diffusion Probability Model Interactively*. 2024. URL: <https://huggingface.co/spaces/blairzheng/DPMInteractive>.

A Some Proofs

A.1 Predicting Posterior Mean is Equivalent to Predicting X_0

In the following, we will prove that the following two objective functions are equivalent:

$$\min_{\theta} \int p(x_t) \left\| f_{\theta}(x_t) - \int p(x_0|x_t) x_0 dx_0 \right\|^2 dx_t \iff \min_{\theta} \iint p(x_0, x_t) \|f_{\theta}(x_t) - x_0\|^2 dx_0 dx_t \quad (38)$$

Proof:

For $\|f_{\theta}(x_t) - \int p(x_0|x_t)x_0 dx_0\|^2$, the following relation holds:

$$\left\| f_{\theta}(x_t) - \int p(x_0|x_t)x_0 dx_0 \right\|^2 = f_{\theta}^2(x_t) - 2f_{\theta}(x_t) \int p(x_0|x_t)x_0 dx_0 + \left\| \int p(x_0|x_t)x_0 dx_0 \right\|^2 \quad (39)$$

$$= \int p(x_0|x_t) f_{\theta}^2(x_t) dx_0 - 2f_{\theta}(x_t) \int p(x_0|x_t)x_0 dx_0 + C_1 \quad (40)$$

$$= \int p(x_0|x_t) (f_{\theta}^2(x_t) - 2f_{\theta}(x_t)x_0 + x_0^2) dx_0 - \int p(x_0|x_t)x_0^2 dx_0 + C_1 \quad (41)$$

$$= \int p(x_0|x_t) \|f_{\theta}^2(x_t) - x_0\|^2 dx_0 - C_2 + C_1 \quad (42)$$

Where C_1 and C_2 are constants that do not depend on θ .

Substituting the above relation into the objective function for predicting the mean, we get:

$$\int p(x_t) \left\| f_{\theta}(x_t) - \int p(x_0|x_t)x_0 dx_0 \right\|^2 dx_t = \int p(x_t) \left(\int p(x_0|x_t) \|f_{\theta}^2(x_t) - x_0\|^2 dx_0 - C_2 + C_1 \right) dx_t \quad (43)$$

$$= \iint p(x_0, x_t) \|f_{\theta}(x_t) - x_0\|^2 dx_0 dx_t + \int p(x_t) (C_1 - C_2) dx_t \quad (44)$$

$$= \iint p(x_0, x_t) \|f_{\theta}(x_t) - x_0\|^2 dx_0 dx_t + C_3 \quad (45)$$

That is, the two objective functions differ only by a constant that does not depend on the optimization parameters. Therefore, the two objective functions are equivalent.

A.2 Conditional Score

Below is the proof of the following relation:

$$\frac{\partial \log p(x_t)}{\partial x_t} = \int p(x_0|x_t) \frac{\partial \log p(x_t|x_0)}{\partial x_t} dx_0 \quad (46)$$

Proof:

$$\frac{\partial \log p(x_t)}{\partial x_t} = \frac{1}{p(x_t)} \frac{\partial p(x_t)}{\partial x_t} \quad (47)$$

$$= \frac{1}{p(x_t)} \frac{\partial (\int p(x_0)p(x_t|x_0)dx_0)}{\partial x_t} \quad (48)$$

$$= \int \frac{p(x_0)}{p(x_t)} \frac{\partial p(x_t|x_0)}{\partial x_t} dx_0 \quad (49)$$

$$= \int \frac{p(x_0, x_t)/p(x_t)}{p(x_0, x_t)/p(x_0)} \frac{\partial p(x_t|x_0)}{\partial x_t} dx_0 \quad (50)$$

$$= \int \frac{p(x_0|x_t)}{p(x_t|x_0)} \frac{\partial p(x_t|x_0)}{\partial x_t} dx_0 \quad (51)$$

$$= \int p(x_0|x_t) \frac{\partial \log p(x_t|x_0)}{\partial x_t} dx_0 \quad (52)$$

A.3 Form of the Posterior Probability

The following derivation refers to the contents of [Zhe23] and [Zhe24].

Assume that x_t has the following form:

$$x_t = c_0 \cdot x_0 + c_1 \cdot \epsilon \quad \text{where } c_0 \text{ and } c_1 \text{ is constant} \quad (53)$$

Then we have:

$$p(x_t|x_0) \sim \mathcal{N}(x_t; c_0 x_0, c_1^2) \quad (54)$$

According to Bayes' theorem, we have

$$p(x_0|x_t) = \frac{p(x_t|x_0)p(x_0)}{p(x_t)} \quad (55)$$

$$= \frac{p(x_t|x_0)p(x_0)}{\int p(x_t|x_0)p(x_0)dx_0} \quad (56)$$

$$= \text{Normalize}(p(x_t|x_0)p(x_0)) \quad (57)$$

where *Normalize* represents the normalization operator, and the normalization divisor is $\int p(x_t|x_0)p(x_0)dx_0$. Substituting equation (54) into this, we get:

$$p(x_0|x_t) = \text{Normalize} \left(\frac{1}{\sqrt{2\pi c_1^2}} \exp \frac{-(x_t - c_0 x_0)^2}{2c_1^2} p(x_0) \right) \quad (58)$$

$$= \text{Normalize} \left(\frac{1}{\sqrt{2\pi c_1^2}} \exp \frac{-(x_0 - \frac{x_t}{c_0})^2}{2\frac{c_1^2}{c_0^2}} p(x_0) \right) \quad (59)$$

$$= \text{Normalize} \left(\exp \frac{-(x_0 - \mu)^2}{2\sigma^2} p(x_0) \right) \quad (60)$$

$$\text{where } \mu = \frac{x_t}{c_0} \quad \sigma = \frac{c_1}{c_0} \quad (61)$$

In the above derivation, due to the presence of the normalization operator, we can ignore the factor $\frac{1}{\sqrt{2\pi c_1^2}}$.

B Coefficient Matrixs

B.1 DDPM Coefficient Matrix

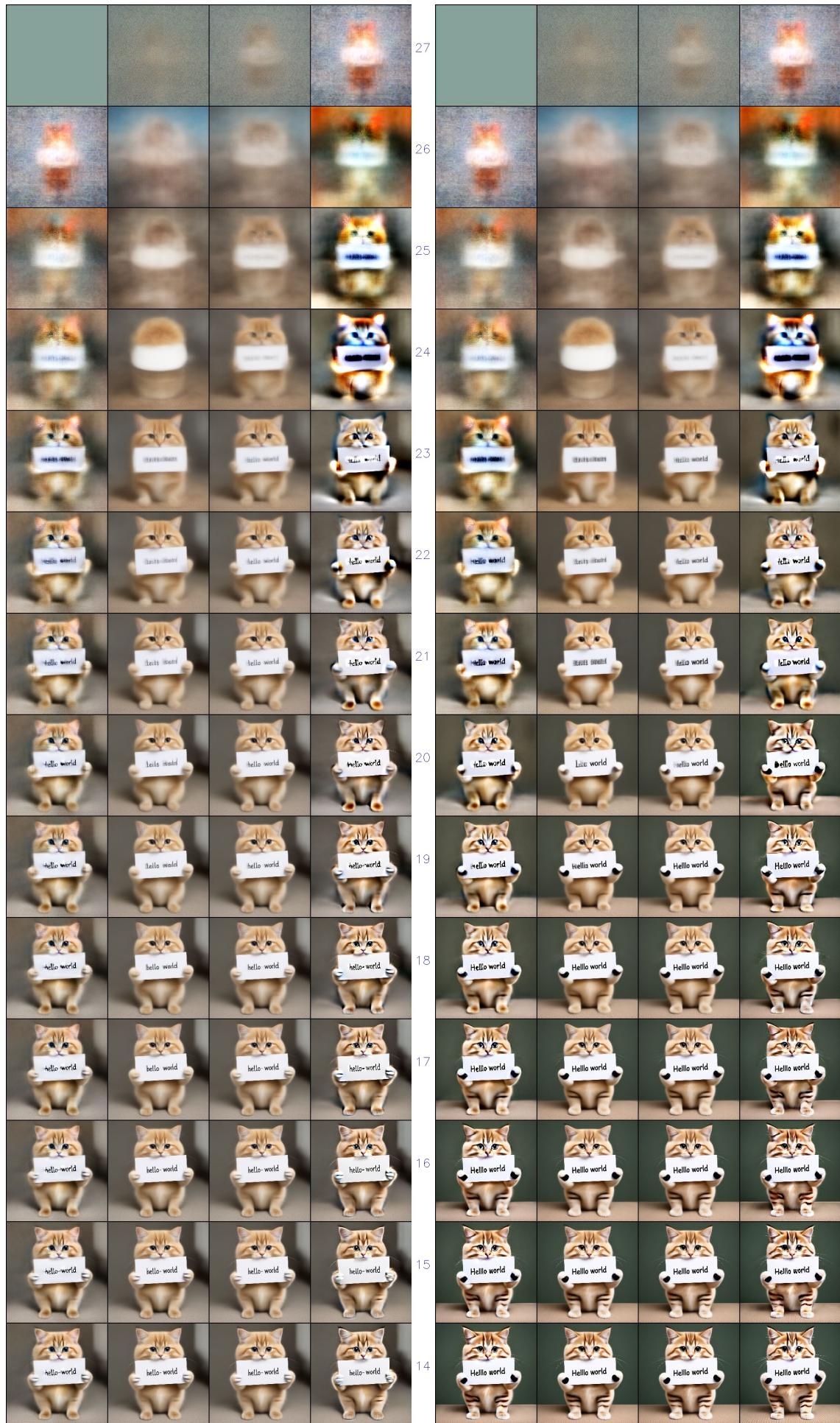


Figure 18: Inference process visualization: first half.

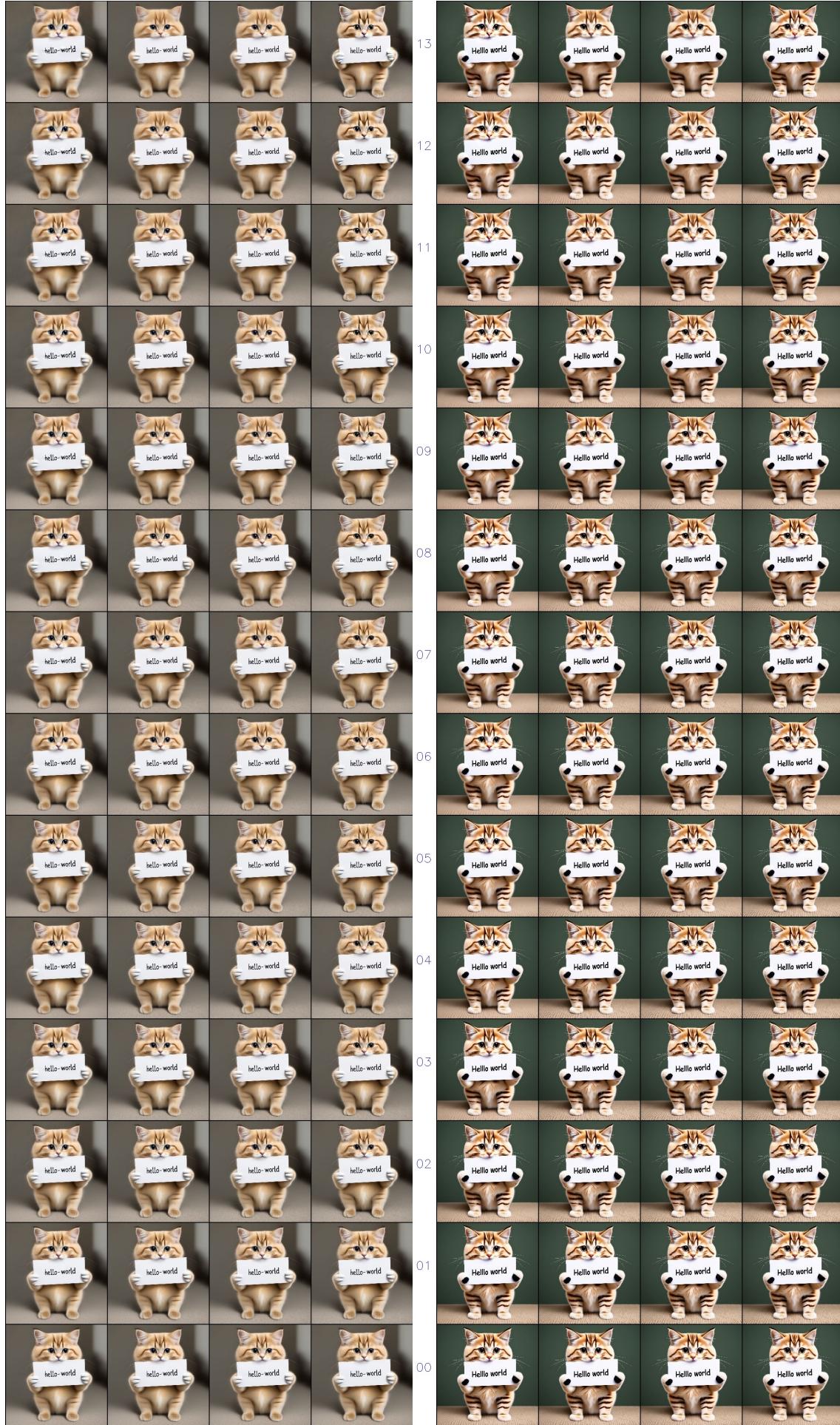


Figure 19: Inference process visualization: second half

D Optimized Coefficient Matrix for Pretrained CIFAR10 Model

To clearly understand the relative proportions of the coefficients in each row, the coefficients in Table 15 have been scaled to ensure that the diagonal elements equal 1. When using these coefficients, each row needs to be normalized to its corresponding Marginal Coefficient for each step. For example, the first row should be normalized to 0.118, and the second row should be normalized to 0.487. The same normalization process applies to Tables 16 and 17.

Table 15: optimized coefficient matrix for 5 step

time	0.650	0.375	0.176	0.051	0.001	marginal coeff
0.650	1	0	0	0	0	0.118
0.375	-0.291	1	0	0	0	0.487
0.176	0	0.133	1	0	0	0.85
0.051	0	0	-0.337	1	0	0.985
0.001	0	0	0	-0.583	1	1

Table 16: optimized coefficient matrix for 10 step

time	0.816	0.650	0.503	0.375	0.266	0.176	0.104	0.051	0.017	0.001	marginal coeff
0.816	1	0	0	0	0	0	0	0	0	0	0.035
0.650	0	1	0	0	0	0	0	0	0	0	0.118
0.503	0	-0.3	1	0	0	0	0	0	0	0	0.276
0.375	0	0.52	-0.3	1	0	0	0	0	0	0	0.487
0.266	0	0.2	0.4	-0.3	1	0	0	0	0	0	0.694
0.176	0	0	0.2	0.4	-0.2	1	0	0	0	0	0.85
0.104	0	0	0	0	0.35	-0.15	1	0	0	0	0.943
0.051	0	0	0	0	0	0.37	-0.2	1	0	0	0.985
0.017	0	0	0	0	0	0	0.1	-0.33	1	0	0.998
0.001	0	0	0	0	0	0	0	0.05	-0.34	1	1

Table 17: optimized coefficient matrix for 15 step

time	0.875	0.758	0.650	0.550	0.459	0.375	0.300	0.234	0.176	0.126	0.084	0.051	0.026	0.009	0.001	marginal coeff
0.875	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.021
0.758	0.24	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.055
0.650	0.1	0.48	1	0	0	0	0	0	0	0	0	0	0	0	0	0.118
0.550	0	0.2	0.41	1	0	0	0	0	0	0	0	0	0	0	0	0.216
0.459	0	0	0.2	-0.2	1	0	0	0	0	0	0	0	0	0	0	0.343
0.375	0	0.3	-0.14	0.56	-0.77	1	0	0	0	0	0	0	0	0	0	0.487
0.300	0	0	0.23	-0.06	0.3	-0.85	1	0	0	0	0	0	0	0	0	0.629
0.234	0	0	0	0.26	-0.01	0.85	-0.86	1	0	0	0	0	0	0	0	0.753
0.176	0	0	0	0.25	0	0.82	-0.78	1	0	0	0	0	0	0	0	0.85
0.126	0	0	0	0	0.23	-0.02	0.9	-0.2	1	0	0	0	0	0	0	0.919
0.084	0	0	0	0	0	0.2	-0.04	0.7	-0.4	1	0	0	0	0	0	0.961
0.051	0	0	0	0	0	0	0.17	-0.07	0.62	-0.66	1	0	0	0	0	0.985
0.026	0	0	0	0	0	0	0	0.14	-0.09	0.57	-0.88	1	0	0	0	0.995
0.009	0	0	0	0	0	0	0	0	0.11	-0.11	0.31	-0.49	1	0	0	0.999
0.001	0	0	0	0	0	0	0	0	0	0.08	-0.11	0.24	-0.31	1	1	1