Rethinking Diffusion Model in High Dimension

Anonymous Author(s)
Affiliation
Address
email

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 chain and SDE, while aligning with the degraded objective function. 3) Guided by this simple framework, more efficient inference methods can be discovered. *Code is available at Supplementary Material.*

1 Introduction

Diffusion models exhibit remarkable competitiveness in high-dimensional data generation scenarios, particularly in image generation [25]. Beyond delivering outstanding performance, diffusion models also possess various elegant mathematical formulations. [28] 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. [13] refined the objective function of diffusion models and introduced Denoised DPM. [30] generalized the noise addition process of diffusion models from discrete to continuous and formulated it as a stochastic differential equation (SDE). [17, 18] 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 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 conventional understandings. Traditionally, it is believed that in high-dimensional sparse scenarios, machine learning models cannot effectively learn complex hidden probability distributions. This discrepancy prompts a fundamental inquiry: **Are diffusion models truly learning these complex distributions as theorized, or is their operational mechanism different from what is commonly assumed?**

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

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

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 degraded objective function but also unifies most of inference methods, including DDPM Ancestral Sampling, DDIM [29], Euler, DPM-Solver [19], DPM-Solver++ [20], and DEIS [35].

Finally, we outlined the various benefits of the new inference framework, and, through experiments, demonstrated that more efficient inference methods exist on this inference framework.

2 Background

Given a batch of sampled data $X_0^0, X_0^1, \ldots, X_0^N$ from the random variable X_0, the diffusion model mixes the data with random noise in different proportions, forming a sequence of new variables $X_1, X_2, \cdots, X_T$. 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:

$$X_t = \sqrt{\bar{\alpha}_t} \cdot X_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \varepsilon \tag{1}$$

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:

$$X_t = (1 - \sigma_t) \cdot X_0 + \sigma_t \cdot \varepsilon \tag{2}$$

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

Markov Chain-based diffusion model 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[26], the mean can be expressed in another form:

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

As seen from equation (7) in [13], 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.

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

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

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

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

$$\min_\theta \int p(x_t) \left\| f_\theta(x_t) - \int p(x_0|x_t)\, x_0\, dx_0 \right\|^2 dx_t \tag{6}$$

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

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

$$\frac{\partial \log p(x_t)}{\partial x_t} = \int p(x_0|x_t) \frac{\partial \log p(x_t|x_0)}{\partial x_t} dx_0 \tag{7}$$

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

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

$$\frac{\partial \log p(x_t|x_0)}{\partial x_t} = -\frac{x_t - \sqrt{\bar{\alpha}_t}x_0}{1-\bar{\alpha}_t} = \underbrace{\frac{\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}}_{const=S_0} \cdot x_0 + \underbrace{\frac{-1}{1-\bar{\alpha}_t}}_{const=S_t} \cdot x_t \tag{8}$$

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

$$\frac{\partial \log p(x_t)}{\partial x_t} = \int p(x_0|x_t)(S_0 \cdot x_0 + S_t \cdot x_t)\, dx_0 = S_0 \int p(x_0|x_t)\, x_0\, dx_0 + S_t \cdot x_t \tag{9}$$

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

Flow Matching-based diffusion model The core of the flow matching-based diffusion model lies in learning the velocity field of the probability flow. According to Theorem 1 in [17], 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.

$$u(x_t) = \int p(x_0|x_t) u(x_t|x_0) dx_0 \tag{10}$$

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

$$u(x_t|x_0) \triangleq \frac{\mathrm{d}x_t}{\mathrm{d}t} = \varepsilon - x_0 \tag{11}$$

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

$$u(x_t) = \int p(x_0|x_t)(\varepsilon - x_0) dx_0 = \varepsilon - \int p(x_0|x_t) x_0 dx_0 \tag{12}$$

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

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

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

The specific proof can be found in Appendix A.1. The integrals above cannot be computed exactly and are typically approximated using Monte Carlo integration. In practice, the required samples are typically obtained via **Ancestral Sampling**. The detailed procedure 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 $p(x_0, x_t)$, and the individual X_t follows $p(x_t)$, and the individual X_0 follows $p(x_0|x_t = X_t)$.

3 Impact of sparsity on the objective function

we first show the form of the posterior probability distribution $p(x_0|x_t)$.

3.1 Form of the posterior $p(x_0|x_t)$

For convenience, we use a unified form to represent the two mixing ways in Eq. (1) and Eq. (2) as follows: $x_t = c_0 \cdot x_0 + c_1 \cdot \varepsilon$. When $c_0^2 + c_1^2 = 1$, this represents the mixing way of Markov

Table 1: Statistics of ImageNet-256(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

Chain-based and Score-based diffusion model. When $c_0 + c_1 = 1$, it represents the Flow Matching mixing way. 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 $p(x_0|x_t)$ has the following form:

$$p(x_0|x_t) = \text{Normalize}\left(\exp \frac{-(x_0-\mu)^2}{2\sigma^2}\, p(x_0)\right) \qquad \text{where } \mu = \frac{x_t}{c_0} \quad \sigma = \frac{c_1}{c_0} \tag{13}$$

Here, $p(x_0)$ is the hidden data distribution, which is unknown and cannot be sampled directly. It can only be randomly selected from the existing samples $\{X_0^0, X_0^1, \ldots, X_0^N\}$ ($X_0^i \sim p(x_0)$). The selection process can be considered as sampling from the following mixed Dirac delta distribution:$p(x_0) = \frac{1}{N}\sum_{i=0}^{N} \delta\left(x_0 - X_0^i\right)$. Substituting this into Equation (13), we get:

$$p(x_0|x_t) = \frac{1}{Z_c} \exp \frac{-(x_0-\mu)^2}{2\sigma^2} \sum_{i=0}^{N} \delta\left(x_0 - X_0^i\right) \tag{14}$$

Here, $\mu = \frac{x_t}{c_0}$, $\sigma = \frac{c_1}{c_0}$, and Z_c is normalization factor. It can be seen that when $p(x_0)$ is discrete, $p(x_0|x_t)$ is also discrete, 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

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:

$$\int x_0\, p(x_0|x_t) dx_t = \frac{1}{Z_c} \sum_{i=0}^{N} X_0^i \, \exp \frac{-(X_0^i-\mu)^2}{2\sigma^2} \tag{15}$$

The mean of $p(x_0|x_t)$ is a weighted sum of all X_0^i samples, and the weight is inversely proportional to the distance between X_0^i and μ. If one sample is much closer than all others, the weighted sum degrades to that single sample. This is more likely with sparse data.

Figure 2 presents an example with sparse data (X_0, blue), and small noise std (green circle). In this case, most of X_t remain near its origin data sample. This make $p(x_0|x_t)$ highly peaked at the closest X_0, causing its mean to degrade from a weighted sum to that single sample. We call this phenomenon **weighted sum degradation** and argue it potentially hinders the model learning the true data distribution.

Next, we analyze *weighted sum degradation* for conditional ImageNet-256 and ImageNet-512[7]. Both datasets have high pixel dims (196608 and 786432) and retain high latent dims (4096 and 16480) after VAE[15, 25] compression. As compression is typical, we will only consider the compressed case below.

We calculate the proportion of degradation. First, we sample X_t as in training (first randomly select an X_0, then sample X_t from $p(x_t|x_0 = X_0)$). Then, we determine whether $p(x_0|x_t = X_t)$ is degraded. If there exists an X_0' such that $p(x_0 = X_0'|x_t = X_t) > 0.9$, then we consider *weighted sum degradation* to be present; if $X_0' = X_0$, it is called *weighted sum degradation to* X_0.

Since noise level also affects weighted sum degradation, we calculate degradation rates separately for different t. We calculate the proportions of both *weighted sum degradation* and *degradation to* X_0 under two noise mixing schemes: VP (Equation (1)) and Flow Matching (Equation (2)).

Tables 3.2 and 3.2 present statistics for both datasets, showing several clear patterns:

- As t decreases, the *weighted sum degradation* phenomenon becomes more pronounced.
- The degradation rate of Flow Matching is higher than that of VP.

Table 2: Statistics of ImageNet-512(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

- The higher the dimension, the greater the proportion of degradation.

Besides, we observe severe degradation in both datasets for both VP and Flow Matching, especially for $t < 600$. Furthermore, due to limited sampling during training, each $p(x_0|x_t = X_t)$ cannot be sufficiently sampled, so **the actual degradation ratio should be higher than the statistics show**.

In high dimensions, each $p(x_0|x_t = X_t)$ should be complex. When *weighted sum degradation* occurs, it is equivalent to using a single sample as an estimator of the mean, which typically have large error. If we cannot provide an accurate fitting target, we argue that the model is unlikely to learn the ideal target accurately. Therefore, **it is necessary to reconsider if diffusion models can truly learn the hidden probability distribution and how they work**.

3.3 A simple way to understand the objective function

As shown previously, weighted sum degradation is significant in high dimensions, which reduces the fitting target to the original data sample (X_0). Therefore, we can understand the objective in a simple way:**predict the original data sample (X_0) from the noise-mixed sample (X_t)** .

From the perspective of the frequency spectrum, we can further understand the principle [10].

As seen in Figure 3, natural image spectra concentrate energy in low frequencies (bright centrally, dark peripherally), while noise have a uniform spectrum. Thus, when mixed with noise, high frequencies always have lower SNR (signal-noise-ratio) than low frequencies. As noise grows, high frequencies are submerged first, then low frequencies (Figure 4).

When training a model to predict X_0 from noise-mixed samples, the model prioritizes frequencies based on their SNR. It easily predicts non-submerged frequencies (likely copying them).For submerged frequencies, it prioritizes predicting the lower-frequency components, as they have relatively higher SNR and larger amplitudes (giving them more weight in the Euclidean loss).

Thus, the objective can be further understood as **filtering higher-frequency components – completing the filtered frequency components** (Figure 5). At large t, even some low frequencies are submerged, so the model prioritizes predicting low frequencies. At small t, only high frequencies are submerged, and the model works on predicting these details. This frequency-dependent process is confirmed during inference: early steps (large t) generate contours, while later steps (small t) add details.

4 A unified inference framework-Natural Inference

We know that current diffusion model inference methods are designed with strict mathematical techniques. 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 train-test matching, 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 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.

We first introduce a class of key operations contained in the new framework.

4.1 Self Guidence

Following the concept of Classifier Free Guidance [14], we introduce a new operation called Self Guidance. The principle of Classifier Free Guidance can be summarized as follows:

$$I_{out} = I_{bad} + \lambda \cdot (I_{good} - I_{bad}) \tag{16}$$

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[12, 27]. 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 predicted x_0 is used as I_{bad} and a later predicted 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. See Fig. 6(c).
- 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}. See Fig. 6(d).
- 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}. See Fig. 6(e).

As shown in Appendix B, **the linear combination of any two model outputs can be viewed as a single Self Guidance, while the linear combination of multiple model outputs can be viewed as a composition of multiple Self Guidances**.

4.2 Natural Inference

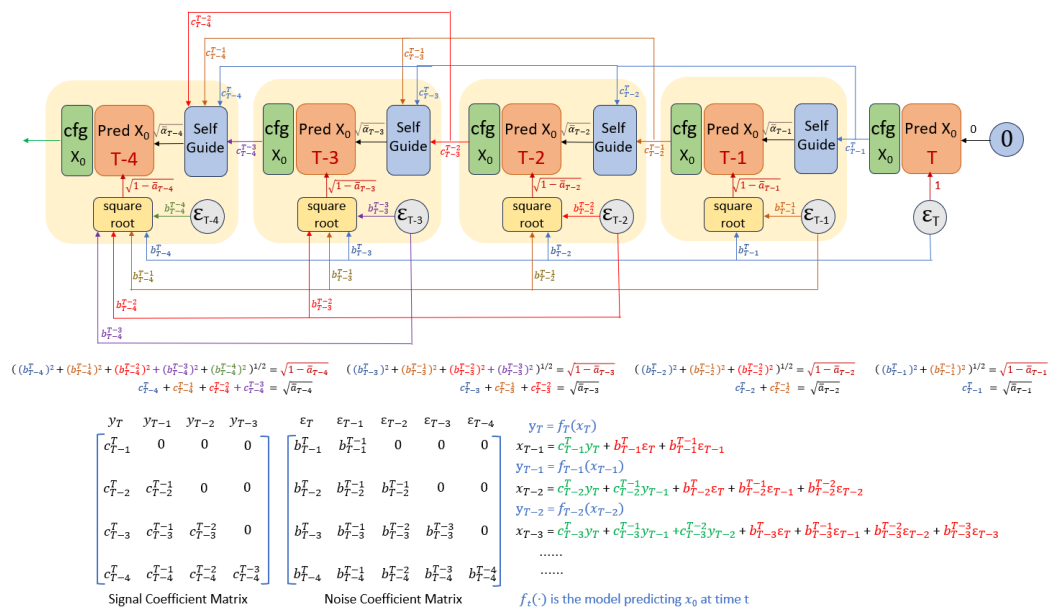


Figure 1: A new inference framework - Natural Inference

The new inference framework is illustrated in Figure 1, with the core ideas summarized as follows:

- It consists of T models that predict X_0;
- Each model takes two part inputs: signal (image) and noise;
- The image signal is a linear combination of outputs from previous models, while the noise is a linear combination of previous noise and newly added noise;

- At time t, the sum of the coefficients corresponding to the image signal ($\sum_{i=t+1}^{T} c_t^i$) equals $\sqrt{\bar{\alpha}_t}$, and the square root of the sum of the squared noise coefficients ($\sqrt{\sum_{i=t}^{T}(b_t^i)^2}$) equals $\sqrt{1-\bar{\alpha}_t}$. This means that **the magnitudes of the signal and noise remain consistent with those used during the training phase**.

As shown in Section 4.1, **the linear combination of image signals can be interpreted as a composition of multiple Self Guidance operations**. The linear combination of independent noise is still noise[31]. Since the input signal of each model depends only on the output signals of previous models, the inference framework exhibits an **autoregressive** structure.

In this paper, we refer to $\sqrt{\bar{\alpha}_t}$ as the **marginal signal coefficient**, and $\sqrt{1-\bar{\alpha}_t}$ as the **marginal noise coefficient**. The term $\sum_{i=t+1}^{T} c_t^i$ is referred to as the **equivalent marginal signal coefficient**, and $\sqrt{\sum_{i=t}^{T}(b_t^i)^2}$ is called the **equivalent marginal noise coefficient**. For clarity, all coefficients are organized into matrix form, as shown in the lower part of Figure 1. Due to the autoregressive property, the signal coefficient matrix has a lower triangular structure.

4.3 Represent sampling methods with Natural Inference Framework

This section briefly demonstrates how various sampling methods can be reformulated within the Natural Inference Framework. For more detailed explanations, please refer to Appendix C.

For first-order sampling methods (including DDPM, DDIM, ODE Euler, SDE Euler, and Flow Matching ODE Euler), their iterative procedures can all be expressed in the following form:

$$y_t = f_t(x_t) \tag{17}$$

$$x_{t-1} = d_{t-1} \cdot x_t + e_{t-1} \cdot y_t + g_{t-1} \cdot \varepsilon_{t-1} \tag{18}$$

Here, f_t is the model function predicting x_0 at step t. x_t, y_t, and ϵ_{t-1} are vectors, while d_{t-1}, e_{t-1}, and g_{t-1} are fixed scalars. For deterministic methods, g_{t-1} is zero.

Starting from x_T, we can iterate according to the above equation to further determine the expressions of x_{T-1}, x_{T-2}, $\cdots$, x_1, and x_0. Each x_t can be represented as two components: one is a linear combination of $\{y_i\}_{i=t+1}^{T}$, and the other is a linear combination of $\{\varepsilon_i\}_{i=t}^{T}$. Since d_{t-1}, e_{t-1}, and g_{t-1} are all known constants, the weights for each element in $\{y_i\}_{i=t+1}^{T}$ and $\{\varepsilon_i\}_{i=t}^{T}$ can be calculated.

The calculation results show that the sum of the coefficients corresponding to $\{y_i\}_{i=t+1}^{T}$ is approximately equal to $\sqrt{\bar{\alpha}_t}$, and the square root of the sum of squared coefficients for $\{\varepsilon_i\}_{i=t}^{T}$ is approximately $\sqrt{1-\bar{\alpha}_t}$. Moreover, the approximation error decreases as the number of sampling steps increases (see Figures 7-9 and Figures 13-14). Therefore, these sampling methods can be represented in the form of the Natural Inference framework.

The above computation can be quite complex, especially when the number of sampling steps is large. Therefore, it is necessary to seek more efficient computation methods. Symbolic computation software [32] offers a promising solution. With minor modifications to the original algorithm code, it can automatically compute the expression for each x_t. For more detailed information, please refer to the accompanying code.

For higher-order sampling methods, their iteration rules are relatively complex, but the expression for x_t can also be quickly calculated with the help of symbolic computation software. The calculation results indicate that DPMSOLVER, DPMSOLVER++, and DEIS yield results similar to those of first-order sampling methods (see Figures 10-12).

5 Advantages 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.
- 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 19 and 20 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. These configuration may be not optimal, and better configurations can be found through careful design and search.

5.1 Why can high order samplings speedup sampling ?

In this subsection, 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 Tables 8 and 10 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 section 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 Mid 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 value, 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[20]. So why do they work, while others do not? By comparing their coefficient matrices, 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 composition of Mid Self Guidances, while 3rd-order DEIS and DPMSolver contain negative values, indicating the presence of Fore Self Guidances. Therefore, it can be inferred that Mid Self Guidance (positive coefficients) is key to handling large CFGs, which can be further supported by the simple experiment in Appendix F.2.

Furthermore, we know that, compared to DDIM, 2nd-order DPMSolver++ not only adapts better to larger CFGs but also provides relatively better image quality. This can also be explained by the coefficient matrixs: compared to DDIM, the coefficient matrix (Table 11) 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 image quality.

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 within the coefficient matrix, we can control the sharpness of the generated image. This principle is demonstrated with a SD3 example. Table 13 shows the coefficient matrix for the original Euler method, and Table 14 shows the adjusted coefficient matrix. A key observation is that the Self Guidances in the Euler method primarily consist of composite Mid Self Guidance, with a significant amount of weight placed on earlier outputs. Conversely, the adjusted coefficient matrix eliminates weights for earlier outputs and assigns greater importance to more recent outputs, resulting in the generation of sharper images, as shown in Figure 17.

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 (larger t).

5.4 Better coefficient matrix

In this subsection, we will take the pre-trained model on the CIFAR10 dataset[16] as an example to demonstrate that there exist better coefficient matrices that can achieve better FID scores under the same number of sampling steps. The pre-trained model uses the ScoreSDE [30] 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. As shown in Table 3, the optimized coefficient matrices achieve better FID scores than DEIS, DPMSolver, and DPMSolver++ when using the same number of steps. All algorithms use the same time discretization schedule(quadratic), and for DEIS, DPMSolver, and DPMSolver++, all hyperparameters are tested to select the best FID as the result.

The optimized coefficient matrices are designed according to the following principles:

- The weight distribution tends to prioritize outputs closer to the current time step, reducing the weight of earlier outputs. In other words, *reducing the tail length*. Generally, when the number of steps is larger, the *tail* should be extended appropriately to avoid the **over-enhancement phenomenon**(see Appendix D); when the number of steps is smaller, the *tail* can be shortened.
- When the number of steps is small, Fore Self Guidance needs to be appropriately increased, which means adding more or larger negative coefficients before the diagonal elements.
- Conduct hyperparameter tuning to find more optimal coefficient matrices.

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 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 become large, which poses a challenge for manual adjustments. One automated solution is to use hyperparameter optimization methods to automatically search for the best configuration [34]; 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.

6 Conclusion

This paper investigates the operational principles of high-dimensional diffusion models. We begin by examining the objective function, analyzing the impact of high-dimensional data sparsity. Based on this analysis, we offer a novel perspective for understanding this objective. Concurrently, this paper introduces a new inference framework that not only unifies mainstream inference methods but also aligns with our proposed perspective on the objective function. Furthermore, this new framework effectively explains certain observed phenomena and can guide the design of more efficient inference algorithms. 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

[1] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[2] Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold diffusion: Inverting arbitrary image transforms without noise. *Advances in Neural Information Processing Systems*, 36:41259–41282, 2023.

[3] Fan Bao, Chongxuan Li, Jiacheng Sun, and Jun Zhu. Why are conditional generative models better than unconditional ones? *arXiv preprint arXiv:2212.00362*, 2022.

[4] Fan Bao, Chongxuan Li, Jiacheng Sun, Jun Zhu, and Bo Zhang. Estimating the optimal covariance with imperfect mean in diffusion probabilistic models. *arXiv preprint arXiv:2206.07309*, 2022.

[5] Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. *arXiv preprint arXiv:2201.06503*, 2022.

[6] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22669–22679, 2023.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[9] Sander Dieleman. Perspectives on diffusion, 2023.

[10] Sander Dieleman. Diffusion is spectral autoregression, 2024.

[11] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[12] Rafael Gonzalez and Richard Woods. *Digital image processing, 4th Edition*. Pearson education india, 2017.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[14] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[15] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[17] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[18] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[19] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

[20] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

[21] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[22] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[23] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[25] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[26] Sheldon Ross. A first course in probability. 2010.

[27] scikit-image Development Team. Unsharp masking, 2013.

[28] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[29] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[30] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[31] Marco Taboga. Linear combinations of normal random variables, 2021.

[32] SymPy Development Team. simpy documentation, 2013.

[33] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[34] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[35] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022.

[36] Zhenxin Zheng. The art of dpm, 2023.

[37] Zhenxin Zheng. Understanding diffusion probability model interactively, 2024.

A Additional 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:

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

Proof:

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

$$\left\| f_\theta(x_t) - \int p(x_0|x_t) x_0 dx_0 \right\|^2 \tag{19}$$

$$= f_\theta^2(x_t) - 2 f_\theta(x_t) \int p(x_0|x_t) x_0 dx_0 + \left\| \int p(x_0|x_t) x_0 dx_0 \right\|^2 \tag{20}$$

$$= \int p(x_0|x_t) f_\theta^2(x_t) dx_0 - 2 f_\theta(x_t) \int p(x_0|x_t) x_0 dx_0 + C_1 \tag{21}$$

$$= \int p(x_0|x_t) \left(f_\theta^2(x_t) - 2 f_\theta(x_t) x_0 + x_0^2 \right) dx_0 - \int p(x_0|x_t) x_0^2 dx_0 + C_1 \tag{22}$$

$$= \int p(x_0|x_t) \left\| f_\theta^2(x_t) - x_0 \right\|^2 dx_0 - C_2 + C_1 \tag{23}$$

Where C_1 and C_2 are constants that do not depend on θ. In Equation (21), we apply $f_\theta^2(x_t) = f_\theta^2(x_t) \int p(x_0|x_t) dx_0 = \int p(x_0|x_t) f_\theta^2(x_t) dx_0$.

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

$$\int p(x_t) \left\| f_\theta(x_t) - \int p(x_0|x_t) x_0 dx_0 \right\|^2 dx_t \tag{24}$$

$$= \int p(x_t) \left(\int p(x_0|x_t) \left\| f_\theta^2(x_t) - x_0 \right\|^2 dx_0 - C_2 + C_1 \right) dx_t \tag{25}$$

$$= \iint p(x_0, x_t) \left\| f_\theta(x_t) - x_0 \right\|^2 dx_0 dx_t + \int p(x_t) \left(C_1 - C_2 \right) dx_t \tag{26}$$

$$= \iint p(x_0, x_t) \left\| f_\theta(x_t) - x_0 \right\|^2 dx_0 dx_t + C_3 \tag{27}$$

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:

$$\frac{\partial \log p(x_t)}{\partial x_t} = \int p(x_0|x_t) \frac{\partial \log p(x_t|x_0)}{\partial x_t} dx_0 \tag{28}$$

Proof:

$$\frac{\partial \log p(x_t)}{\partial x_t} = \frac{1}{p(x_t)} \frac{\partial p(x_t)}{\partial x_t} \tag{29}$$

$$= \frac{1}{p(x_t)} \frac{\partial \left(\int p(x_0) p(x_t|x_0) dx_0 \right)}{\partial x_t} \tag{30}$$

$$= \int \frac{p(x_0)}{p(x_t)} \frac{\partial p(x_t|x_0)}{\partial x_t} dx_0 \tag{31}$$

$$= \int \frac{p(x_0, x_t)/p(x_t)}{p(x_0, x_t)/p(x_0)} \frac{\partial p(x_t|x_0)}{\partial x_t} dx_0 \tag{32}$$

$$= \int \frac{p(x_0|x_t)}{p(x_t|x_0)} \frac{\partial p(x_t|x_0)}{\partial x_t} dx_0 \tag{33}$$

$$= \int p(x_0|x_t) \frac{\partial \log p(x_t|x_0)}{\partial x_t} dx_0 \tag{34}$$

A.3 Form of the posterior probability

The following derivation is based on [36] and [37].

Assume that x_t has the following form:

$$x_t = c_0 \cdot x_0 + c_1 \cdot \epsilon \qquad \text{where } c_0 \text{ and } c_1 \text{ is constant} \tag{35}$$

Then we have:

$$p(x_t|x_0) \sim \mathcal{N}(x_t; c_0 x_0, c_1^2) \tag{36}$$

According to Bayes' theorem, we have

$$p(x_0|x_t) = \frac{p(x_t|x_0)p(x_0)}{p(x_t)} \tag{37}$$

$$= \frac{p(x_t|x_0)p(x_0)}{\int p(x_t|x_0)p(x_0)dx_0} \tag{38}$$

$$= \text{Normalize}\big(p(x_t|x_0)p(x_0)\big) \tag{39}$$

where *Normalize* represents the normalization operator, and the normalization divisor is $\int p(x_t|x_0)p(x_0)dx_0$.

Substituting Equation (36) into this, we get:

$$p(x_0|x_t) = \text{Normalize}\left(\frac{1}{\sqrt{2\pi c_1^2}} \exp \frac{-(x_t - c_0 x_0)^2}{2c_1^2} \, p(x_0) \right) \tag{40}$$

$$= \text{Normalize}\left(\frac{1}{\sqrt{2\pi c_1^2}} \exp \frac{-(x_0 - \frac{x_t}{c_0})^2}{2\frac{c_1^2}{c_0^2}} \, p(x_0) \right) \tag{41}$$

$$= \text{Normalize}\left(\exp \frac{-(x_0 - \mu)^2}{2\sigma^2} \, p(x_0) \right) \tag{42}$$

$$\text{where } \mu = \frac{x_t}{c_0} \qquad \sigma = \frac{c_1}{c_0} \tag{43}$$

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

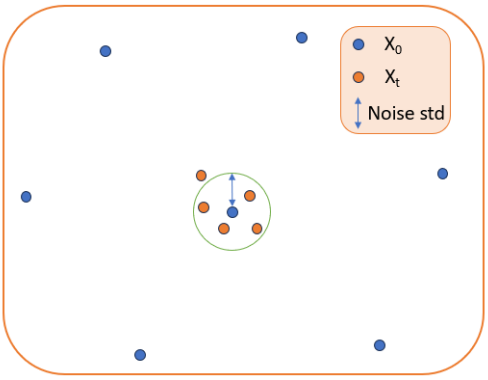


Figure 2: Impact of data sparsity on posterior probability distribution

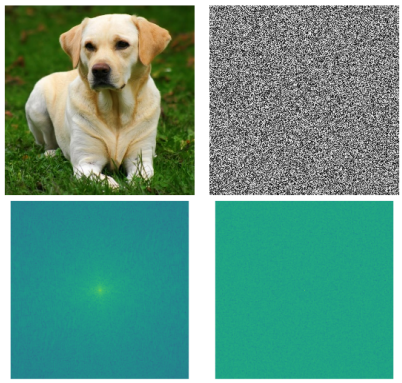


Figure 3: Left: Natural image and its spectrum. Right: noise and its spectrum

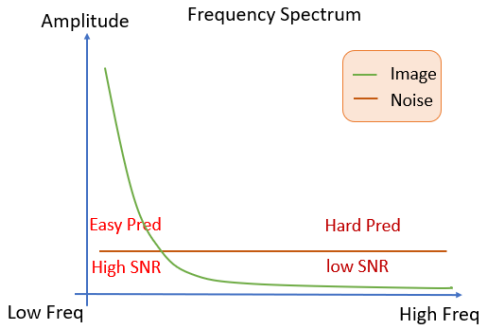


Figure 4: Image and noise frequency spectrum

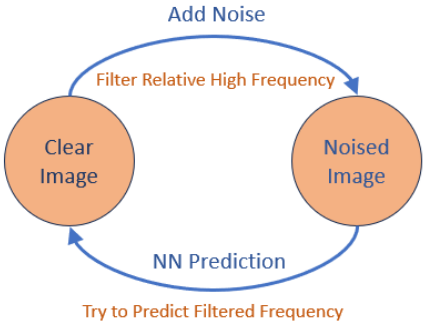


Figure 5: New perspective of training object function

B Self Guidance and its composition

In this section, we show that the linear combination of any two model outputs(prediting x_0) can be viewed as a Self Guidance operation.

As described in Section 3.2, the self guidance is defined as follows:

$$I_{out} = I_{bad} + \lambda \cdot (I_{good} - I_{bad}) \tag{44}$$

This equation can be further written as

$$I_{out} = \lambda \cdot I_{good} + (1 - \lambda) \cdot I_{bad} \tag{45}$$

$$= \eta_{good} \cdot I_{good} + \eta_{bad} \cdot I_{bad} \tag{46}$$

$$\text{where} \quad \eta_{good}, \eta_{bad} \in real \quad \eta_{good} + \eta_{bad} = 1 \tag{47}$$

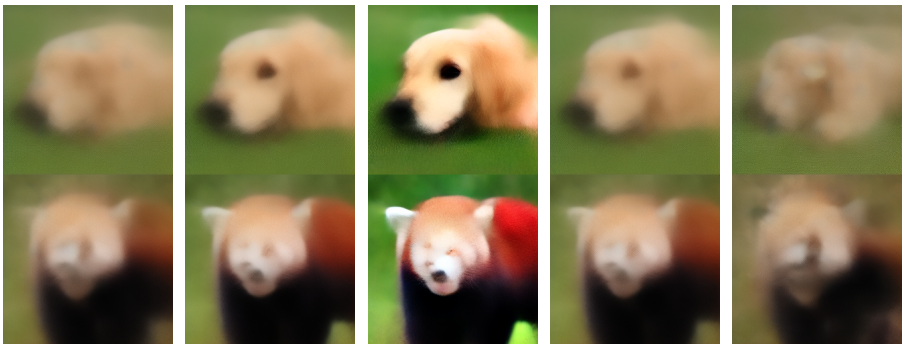
As shown above, the coefficients of I_{bad} and I_{good} can take any value, but the sum of I_{bad} and I_{good} must equal 1. For **Fore Self Guidence**, $\eta_{good} > 0$, $\eta_{bad} < 0$; for **Mid Self Guidence**, $\eta_{good} > 0$, $\eta_{bad} > 0$; for **Back Self Guidence**, $\eta_{good} < 0$,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

For the linear combination of any two model outputs, it can be written as:

$$I_{out} = a \cdot I_{good} + b \cdot I_{bad} = (a + b) \cdot (\frac{a}{a + b} \cdot I_{good} + \frac{b}{a + b} \cdot I_{bad}) \tag{48}$$

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

Thus, **the linear combination of any two I_{bad} and I_{good} can be represented as Self Guidence with a scaling factor**.



(a) (b) (c) (d) (e)

Figure 6: (a) Model output on t=540 (I_{bad}) (b) Model output on t=500 (I_{good}) (c) Output of Fore Self Guidence (d) Output of Mid Self Guidence (e) Output of Back Self Guidence

For the linear combination of multiple model outputs, it can be written as:

$$
\begin{aligned}
I_{out} &= a \cdot I_a + b \cdot I_b + c \cdot I_c && (49)\\
&= (a+b) \cdot (\frac{a}{a+b} \cdot I_a + \frac{b}{a+b} \cdot I_b) + c \cdot I_c && (50)\\
&= (a+b+c) \cdot \big(\frac{a+b}{a+b+c} \cdot (\frac{a}{a+b} I_a + \frac{b}{a+b} I_b) + \frac{c}{a+b+c} \cdot I_c\big) && (51)
\end{aligned}
$$

Thus, **the linear combination of multiple model outputs can be viewed as a composition of Self Guidences**.

C Represent sampling methods with Natural Inference framework

C.1 Represent DDPM Ancestral Sampling with Natural Inference framework

This subsection will demonstrate that the DDPM Ancestor Sampling can be reformulated within the Natural Inference framework. The iterative process of the Ancestor Sampling is as follows:

$$
\begin{aligned}
y_t &= f_t(x_t)\\
x_{t-1} &= d_{t-1} \cdot x_t + e_{t-1} \cdot y_t + g_{t-1} \cdot \varepsilon_{t-1}\\
\text{where } d_{t-1} = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \quad & e_{t-1} = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \quad g_{t-1} = \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t}
\end{aligned} \tag{52}
$$

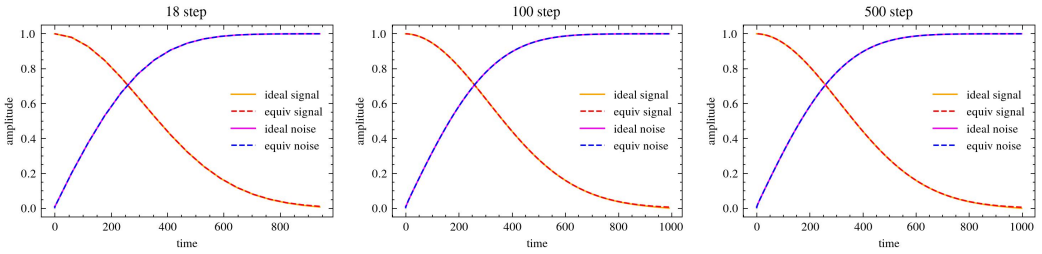
Here, f_t is the model function at the t-th step. In this case, we assume the model predicts x_0, but other forms of prediction models (such as predict ε or predict v) can be transformed into the form of predicting x_0. y_t is the output of f_t, which is the predicted x_0 at the t-th step.

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

$$
\begin{aligned}
x_T &= g_T \cdot \varepsilon_T \quad \text{where } g_T = 1\\
x_{T-1} &= d_{T-1} \cdot x_T + e_{T-1} \cdot y_T + g_{T-1} \cdot \varepsilon_{T-1}\\
&= e_{T-1} y_T + (d_{T-1} g_T \varepsilon_T + g_{T-1} \varepsilon_{T-1})
\end{aligned} \tag{53}
$$

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

$$
\begin{aligned}
x_{T-2} &= d_{T-2} \cdot x_{T-1} + e_{T-2} \cdot y_{T-1} + g_{T-2} \cdot \varepsilon_{T-2}\\
&= (d_{T-2} e_{T-1} y_T + e_{T-2} y_{T-1}) + (d_{T-2} d_{T-1} g_T \varepsilon_T + d_{T-2} g_{T-1} \varepsilon_{T-1} + g_{T-2} \varepsilon_{T-2})
\end{aligned} \tag{54}
$$



(a) (b) (c)

Figure 7: DDPM equivalent marginal coefficients and ideal margingal coefficients (a) 18 step (b) 100 step (c) 500 step

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

$$\begin{aligned} x_{T-3} &= d_{T-3} \cdot x_{T-2} + e_{T-3} \cdot y_{T-2} + g_{T-3} \cdot \varepsilon_{T-3} \\ &= (d_{T-3}d_{T-2}e_{T-1}y_T + d_{T-3}e_{T-2}y_{T-1} + e_{T-3}y_{T-2}) \\ &+ (d_{T-3}d_{T-2}d_{T-1}g_T\varepsilon_T + d_{T-3}d_{T-2}g_{T-1}\varepsilon_{T-1} + d_{T-3}g_{T-2}\varepsilon_{T-2} + g_{T-3}\varepsilon_{T-3}) \end{aligned} \tag{55}$$

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 7 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.

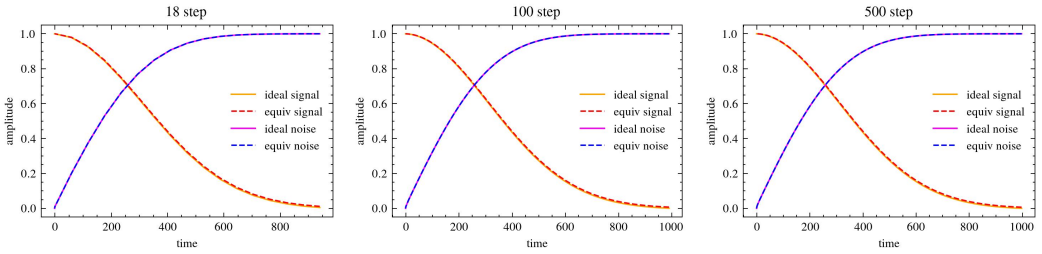
At this point, we have successfully demonstrated that the DDPM Ancestral Sampling process can be represented using the Natural Inference framework.

C.2 Represent DDIM with Natural Inference framework

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

$$\begin{aligned} y_t &= f_t(x_t) \\ x_{t-1} &= \sqrt{\bar{\alpha}_{t-1}} \cdot x_t + \sqrt{1-\bar{\alpha}_{t-1}} \cdot \frac{x_t - \sqrt{\bar{\alpha}_t} y_t}{\sqrt{1-\bar{\alpha}_t}} \cdot y_t \\ &= d_{t-1} \cdot x_t + e_{t-1} \cdot y_t \\ \text{where } d_{t-1} &= \frac{\sqrt{1-\bar{\alpha}_{t-1}}}{\sqrt{1-\bar{\alpha}_t}} \quad e_{t-1} = (\sqrt{\bar{\alpha}_{t-1}} - \frac{\sqrt{1-\bar{\alpha}_{t-1}}}{\sqrt{1-\bar{\alpha}_t}\sqrt{\bar{\alpha}_t}}) \end{aligned} \tag{56}$$

It can be seen that the iterative rule of DDIM is similar to those 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 way of DDPM Ancestral Sampling, each x_t corresponding to t can also be written in a similar form, as follows:



(a) (b) (c)

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

$$
\begin{aligned}
x_T &= g_T \cdot \varepsilon_T \qquad \text{where } g_T = 1 \\
x_{T-1} &= e_{T-1} y_T + d_{T-1} g_T \varepsilon_T \\
x_{T-2} &= (d_{T-2} e_{T-1} y_T + e_{T-2} y_{T-1}) + d_{T-2} d_{T-1} g_T \varepsilon_T \\
x_{T-3} &= (d_{T-3} d_{T-2} e_{T-1} y_T + d_{T-3} e_{T-2} y_{T-1} + e_{T-3} y_{T-2}) \\
&\quad + d_{T-3} d_{T-2} d_{T-1} g_T \varepsilon_T
\end{aligned} \tag{57}
$$

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 8 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.

C.3 Represent Flow Matching Euler Sampling with Natural Inference Framework

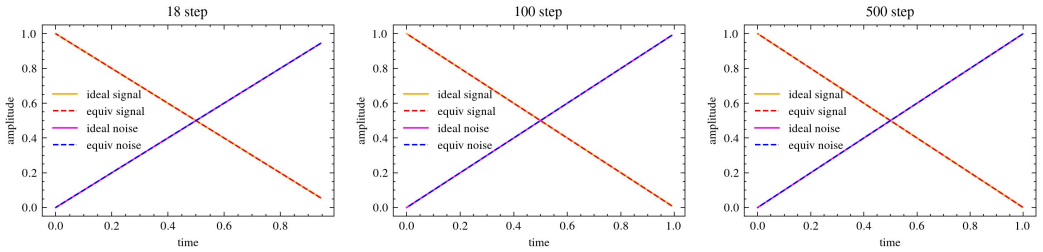
The noise mixing method of Flow Matching is shown in Equation (2). When using Euler discretized integral sampling, its iterative rule can be expressed as follows:

$$
\begin{aligned}
y_i &= f_i(x_i) \\
x_{i-1} &= x_i + (t_{i-1} - t_i)(-y_i + \epsilon) \\
&= x_i + (t_{i-1} - t_i)\frac{x_i - y_i}{t_i} \\
&= d_{i-1} \cdot x_i + e_{i-1} \cdot y_i \\
\text{where } d_{i-1} &= \frac{t_{i-1}}{t_{t_i}} \quad e_{i-1} = (1 - \frac{t_{i-1}}{t_{t_i}})
\end{aligned} \tag{58}
$$

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

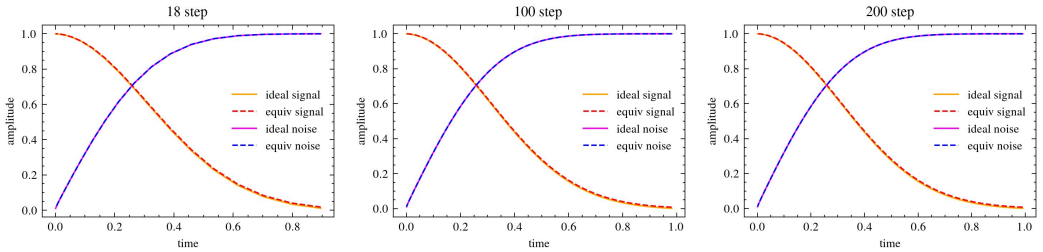
It can be observed that the iterative rule of the Euler algorithm in Flow Matching is similar to that of DDIM, so each x_i can also be expressed in a similar form.

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 9, which shows the results for 18



(a) (b) (c)

Figure 9: Flow matching euler sampler equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 100 step (c) 500 step



(a) (b) (c)

Figure 10: DEIS equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 100 step (c) 500 step

steps, 200 steps, and 500 steps, respectively. Table 7 presents the signal coefficient matrix for 18 steps.

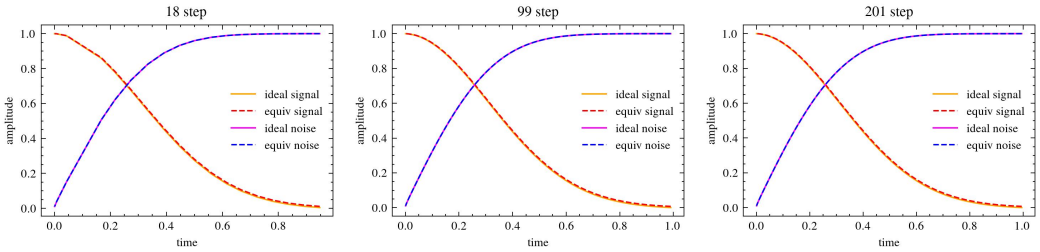
C.4 Represent high order samplers with Natural Inference framework

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, it is quite difficult to directly compute the expression of each x_i. 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[32], and For specific details, please refer to the code attached in this paper.

The compuation results show 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 $\sqrt{1-\bar{\alpha}_i}$.

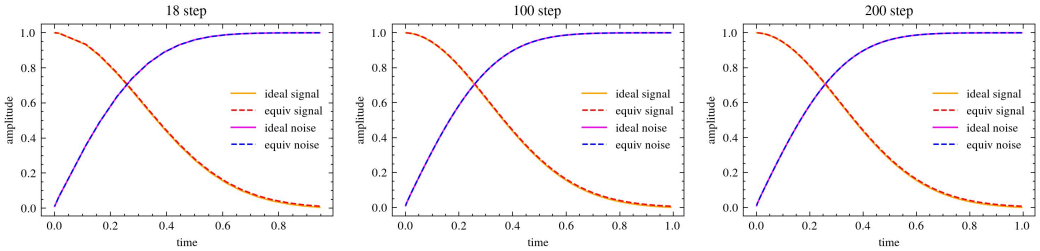
Figure 10 shows the results for the DEIS(tab3) algorithm, Figure 11 presents the results for the third-order DPMSolver, and Figure 12 illustrates the results for the second-order DPMSolver++. It can be observed that these higher-order sampling algorithms exhibit the same properties and can also be represented 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).



(a) (b) (c)

Figure 11: dpmsolver3s equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 99 step (c) 201 step

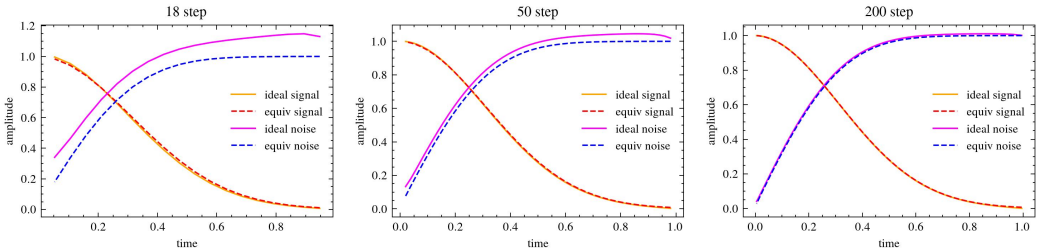


(a) (b) (c)

Figure 12: dpmsolver++2s equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 99 step (c) 201 step

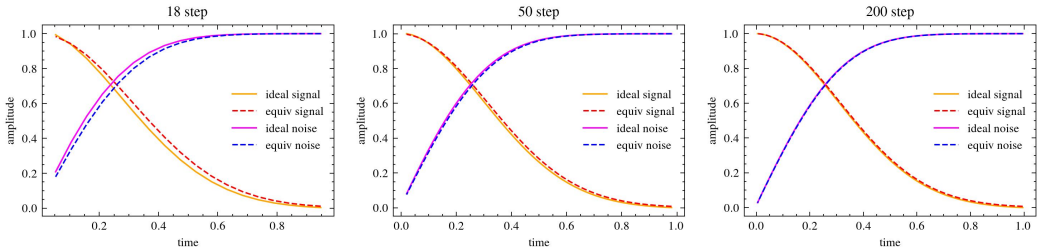
C.5 Represent SDE Euler and ODE Euler with Natural Inference framework

For SDE Euler and ODE Euler, the expressions for each x_t can also be automatically computed using SymPy. The compuation results indicate that these two algorithms yield results similar to previous algorithms, but they suffer from relatively larger errors, especially when the number of steps is small. For details, see Figure 13 and Figure 14.



(a) (b) (c)

Figure 13: SDE Euler equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 50 step (c) 200 step



(a) (b) (c)

Figure 14: ODE Euler equivalent marginal coefficients and ideal marginal coefficients (a) 18 step (b) 50 step (c) 200 step

D Over Enhancement phenomenon

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

Over-enhancement phenomenon As established in Section 3.3, fitting x_0 essentially compensates for higher-frequency components that are drowned out by noise, which in turn enriching the details in the input image. Consequently, the model can be regarded as an image quality enhancement operator. Furthermore, the preceding analysis has also demonstrated that Self Guidance operations are also capable of improving image quality. It follows that when **a multitude of enhancement operations** are employed, the propensity for the **over-enhancement phenomenon** is heightened. This can occur, for instance, with frequent and dense model enhancements—where sampling time points are proximate and time intervals are minimal—or through the application of potent Self Guidance operations.

This phenomenon can be intuitively illustrated through the following procedure: Given a standard input image, by fixing the model time point t and the noise level, and then iteratively applying model enhancements, the *over-enhancement* phenomenon progressively becomes apparent. With a fixed t and a zero time interval between applications, this setup fulfills the condition of dense enhancement, as previously discussed. In this process, the output image from one iteration serves directly as the input for the next. This can be interpreted as a form of Self Guidance operating without a *tail* component and with $\lambda = 1$ at a high intensity level.

The specific visual outcomes are depicted in Figure 15. The first row displays the original image, while the second row presents the result after a single application of the model. A decrease in image quality is observable at this stage, primarily 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. From the fourth row onwards, the *over-enhancement* phenomenon progressively emerges and intensifies.

Furthermore, it is evident that the characteristics of the *over-enhancement* phenomenon vary with different time points t. For smaller values of t, the image exhibits a pronounced grainy texture and retains more high-frequency information. Conversely, at larger values, the image tends to become overly simplified and deficient in detail.

Train-test mismatch in input domain This phenomenon may be related to the **train-test mismatch** in the model input domain. During training, the input images are natural, 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.

To validate this analysis, a straightforward experiment can be conducted. For a model demonstrably exhibiting the *over-enhancement* phenomenon, its training data is augmented by incorporating samples akin to those depicted in the second and third rows of Figure 15. Subsequent to fine-tuning this modified model, over-enhanced images are generated using the identical procedure. The results

(a) (b) (c)

Figure 15: over enhancing phenomenon on ddpm latent model trained on imageget-256 (a) t=100 (b) t=300 (c) t=500

of this experiment are illustrated in Figure 16. It is evident that the image quality has significantly improved: for smaller t, graininess is notably diminished, while for larger t, a greater degree of detail is preserved.

(a) (b) (c)

Figure 16: improved over enhancing phenomenon after finetuning with model output images (a) t=100 (b) t=300 (c) t=500

E Coefficient matrixes

E.1 DDPM coefficient matrix

Table 4: DDPM's signal coefficient matrix on Natural Inference framework

time	999	940	881	823	764	705	646	588	529	470	411	353	294	235	176	118	059	000	sum
940	0.008	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.008
881	0.005	0.013	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.017
823	0.003	0.008	0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.031
764	0.002	0.005	0.013	0.032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.051
705	0.001	0.003	0.008	0.02	0.047	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.079
646	0.001	0.002	0.005	0.013	0.031	0.067	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.119
588	0.0	0.001	0.004	0.009	0.021	0.046	0.09	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.172
529	0.0	0.001	0.003	0.006	0.015	0.032	0.062	0.12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.24
470	0.0	0.001	0.002	0.005	0.01	0.022	0.044	0.085	0.154	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.323
411	0.0	0.0	0.001	0.003	0.007	0.016	0.031	0.06	0.109	0.192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.42
353	0.0	0.0	0.001	0.002	0.005	0.011	0.022	0.042	0.076	0.135	0.232	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.526
294	0.0	0.0	0.001	0.002	0.003	0.007	0.015	0.028	0.051	0.091	0.156	0.284	0.0	0.0	0.0	0.0	0.0	0.0	0.639
235	0.0	0.0	0.0	0.001	0.002	0.005	0.009	0.018	0.033	0.057	0.099	0.18	0.345	0.0	0.0	0.0	0.0	0.0	0.749
176	0.0	0.0	0.0	0.001	0.001	0.003	0.005	0.01	0.018	0.032	0.056	0.101	0.195	0.426	0.0	0.0	0.0	0.0	0.849
118	0.0	0.0	0.0	0.0	0.001	0.001	0.002	0.005	0.008	0.015	0.026	0.047	0.09	0.196	0.536	0.0	0.0	0.0	0.927
059	0.0	0.0	0.0	0.0	0.0	0.0	0.001	0.001	0.002	0.004	0.007	0.013	0.024	0.053	0.145	0.728	0.0	0.0	0.98
000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.998	0.0	1.0
-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0

Table 5: DDPM's noise coefficient matrix on Natural Inference framework

time	999	940	881	823	764	705	646	588	529	470	411	353	294	235	176	118	059	000	-01	norm
940	0.561	0.828	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
881	0.326	0.481	0.814	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
823	0.197	0.292	0.494	0.795	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.999
764	0.123	0.181	0.307	0.494	0.782	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.999
705	0.079	0.117	0.197	0.318	0.502	0.763	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.997
646	0.052	0.077	0.131	0.211	0.333	0.506	0.741	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.993
588	0.036	0.053	0.09	0.144	0.228	0.347	0.508	0.712	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.985
529	0.025	0.037	0.062	0.1	0.159	0.241	0.353	0.496	0.687	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.971
470	0.018	0.026	0.044	0.071	0.112	0.17	0.249	0.349	0.485	0.653	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.946
411	0.012	0.018	0.031	0.05	0.079	0.12	0.176	0.247	0.342	0.462	0.613	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.907
353	0.009	0.013	0.022	0.035	0.056	0.084	0.123	0.173	0.24	0.324	0.43	0.564	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
294	0.006	0.009	0.015	0.024	0.037	0.057	0.083	0.117	0.162	0.218	0.29	0.38	0.513	0.0	0.0	0.0	0.0	0.0	0.0	0.769
235	0.004	0.005	0.009	0.015	0.024	0.036	0.053	0.074	0.102	0.138	0.183	0.24	0.324	0.449	0.0	0.0	0.0	0.0	0.0	0.662
176	0.002	0.003	0.005	0.008	0.013	0.02	0.03	0.042	0.058	0.078	0.103	0.135	0.183	0.253	0.375	0.0	0.0	0.0	0.0	0.529
118	0.001	0.001	0.002	0.004	0.006	0.009	0.014	0.019	0.027	0.036	0.048	0.062	0.084	0.117	0.173	0.285	0.0	0.0	0.0	0.375
059	0.0	0.0	0.001	0.001	0.002	0.003	0.004	0.005	0.007	0.01	0.013	0.017	0.023	0.032	0.047	0.077	0.173	0.0	0.0	0.201
000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.01
-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00

E.2 DDIM coefficient matrix

Table 6: DDIM's signal coefficient matrix on the Natural Inference framework

time	999	940	881	823	764	705	646	588	529	470	411	353	294	235	176	118	059	000	sum
940	0.005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.005
881	0.005	0.008	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.013
823	0.005	0.008	0.013	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.026
764	0.005	0.008	0.013	0.019	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.045
705	0.005	0.008	0.013	0.019	0.028	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.074
646	0.005	0.008	0.013	0.019	0.028	0.04	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.113
588	0.005	0.008	0.012	0.019	0.028	0.04	0.053	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.166
529	0.005	0.008	0.012	0.019	0.028	0.039	0.052	0.07	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.234
470	0.005	0.008	0.012	0.018	0.027	0.038	0.051	0.069	0.089	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.317
411	0.005	0.007	0.011	0.018	0.026	0.037	0.049	0.066	0.086	0.111	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.415
353	0.004	0.007	0.011	0.017	0.024	0.034	0.046	0.062	0.08	0.104	0.132	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.521
294	0.004	0.006	0.01	0.015	0.022	0.031	0.041	0.056	0.073	0.094	0.12	0.163	0.0	0.0	0.0	0.0	0.0	0.0	0.634
235	0.003	0.005	0.008	0.013	0.019	0.027	0.036	0.048	0.063	0.081	0.103	0.14	0.199	0.0	0.0	0.0	0.0	0.0	0.745
176	0.003	0.004	0.007	0.01	0.015	0.021	0.029	0.038	0.05	0.065	0.082	0.112	0.159	0.25	0.0	0.0	0.0	0.0	0.845
118	0.002	0.003	0.005	0.007	0.011	0.015	0.02	0.027	0.035	0.046	0.058	0.08	0.113	0.177	0.325	0.0	0.0	0.0	0.924
059	0.001	0.002	0.003	0.004	0.006	0.008	0.011	0.015	0.019	0.025	0.031	0.043	0.06	0.095	0.174	0.483	0.0	0.0	0.978
000	0.0	0.0	0.0	0.0	0.0	0.0	0.001	0.001	0.001	0.001	0.002	0.002	0.003	0.005	0.009	0.024	0.951	0.0	1.0
-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0

E.3 Flow Matching Coefficient Matrix

Table 7: Flow Matching Euler sampler's signal coefficient matrix on Natural Inference framework

time	1.000	0.944	0.889	0.833	0.778	0.722	0.667	0.611	0.556	0.500	0.444	0.389	0.333	0.278	0.222	0.167	0.111	0.056	sum
0.944	0.056	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.056
0.889	0.052	0.059	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.111
0.833	0.049	0.055	0.062	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.167
0.778	0.046	0.051	0.058	0.067	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.222
0.722	0.042	0.048	0.054	0.062	0.071	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.278
0.667	0.039	0.044	0.05	0.057	0.066	0.077	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.333
0.611	0.036	0.04	0.046	0.052	0.06	0.071	0.083	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.389
0.556	0.033	0.037	0.042	0.048	0.055	0.064	0.076	0.091	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.444
0.500	0.029	0.033	0.038	0.043	0.049	0.058	0.068	0.082	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
0.444	0.026	0.029	0.033	0.038	0.044	0.051	0.061	0.073	0.089	0.111	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.556
0.389	0.023	0.026	0.029	0.033	0.038	0.045	0.053	0.064	0.078	0.097	0.125	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.611
0.333	0.02	0.022	0.025	0.029	0.033	0.038	0.045	0.055	0.067	0.083	0.107	0.143	0.0	0.0	0.0	0.0	0.0	0.0	0.667
0.278	0.016	0.018	0.021	0.024	0.027	0.032	0.038	0.045	0.056	0.069	0.089	0.119	0.167	0.0	0.0	0.0	0.0	0.0	0.722
0.222	0.013	0.015	0.017	0.019	0.022	0.026	0.03	0.036	0.044	0.056	0.071	0.095	0.133	0.2	0.0	0.0	0.0	0.0	0.778
0.167	0.01	0.011	0.012	0.014	0.016	0.019	0.023	0.027	0.033	0.042	0.054	0.071	0.1	0.15	0.25	0.0	0.0	0.0	0.833
0.111	0.007	0.007	0.008	0.01	0.011	0.013	0.015	0.018	0.022	0.028	0.036	0.048	0.067	0.1	0.167	0.333	0.0	0.0	0.889
0.056	0.003	0.004	0.004	0.005	0.005	0.006	0.008	0.009	0.011	0.014	0.018	0.024	0.033	0.05	0.083	0.167	0.5	0.0	0.944
0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0

E.4 DEIS coefficient matrix

Table 8: DEIS sampler's signal coefficient matrix on Natural Inference framework

time	1.000	0.895	0.796	0.703	0.616	0.534	0.459	0.389	0.324	0.266	0.213	0.167	0.126	0.090	0.061	0.037	0.019	0.007	sum
0.895	0.011	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.011
0.796	0.002	0.033	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.034
0.703	0.014	-0.01	0.072	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.076
0.616	-0.005	0.058	-0.043	0.13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.14
0.534	0.014	-0.013	0.09	-0.046	0.183	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.229
0.459	-0.004	0.054	-0.037	0.135	-0.046	0.235	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.337
0.389	0.011	-0.005	0.069	-0.02	0.165	-0.046	0.283	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.457
0.324	-0.001	0.038	-0.015	0.093	-0.004	0.19	-0.047	0.324	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.577
0.266	0.007	0.004	0.041	0.004	0.105	0.009	0.209	-0.053	0.363	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.689
0.213	0.001	0.023	-0.001	0.055	0.017	0.113	0.016	0.223	-0.063	0.401	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.785
0.167	0.004	0.006	0.022	0.012	0.06	0.025	0.116	0.015	0.234	-0.076	0.441	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.86
0.126	0.001	0.013	0.003	0.03	0.018	0.062	0.026	0.117	0.009	0.245	-0.094	0.487	0.0	0.0	0.0	0.0	0.0	0.0	0.916
0.090	0.002	0.005	0.011	0.01	0.032	0.02	0.06	0.021	0.115	-0.003	0.257	-0.115	0.541	0.0	0.0	0.0	0.0	0.0	0.954
0.061	0.001	0.006	0.002	0.015	0.011	0.03	0.016	0.056	0.012	0.114	-0.02	0.271	-0.141	0.606	0.0	0.0	0.0	0.0	0.977
0.037	0.001	0.002	0.005	0.004	0.014	0.009	0.027	0.01	0.051	-0.0	0.112	-0.042	0.284	-0.173	0.687	0.0	0.0	0.0	0.99
0.019	0.0	0.002	0.001	0.006	0.004	0.012	0.005	0.022	0.002	0.045	-0.014	0.11	-0.066	0.292	-0.208	0.785	0.0	0.0	0.997
0.007	0.0	0.0	0.002	0.001	0.004	0.002	0.008	0.001	0.017	-0.005	0.039	-0.027	0.103	-0.088	0.285	-0.244	0.902	0.0	0.999
0.001	-0.0	0.0	-0.0	0.001	-0.0	0.002	-0.001	0.005	-0.003	0.012	-0.012	0.033	-0.039	0.09	-0.111	0.262	-0.319	1.078	1.0

E.5 DPMSolver coefficient matrix

Table 9: DPMSolver2S's signal coefficient matrix on Natural Inference framework

time	1.000	0.946	0.889	0.835	0.778	0.724	0.667	0.614	0.556	0.502	0.445	0.390	0.334	0.277	0.223	0.161	0.112	0.016	sum
0.946	0.005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.005
0.889	-0.008	0.021	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.012
0.835	-0.008	0.021	0.011	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.023
0.778	-0.008	0.021	-0.017	0.045	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.041
0.724	-0.008	0.021	-0.017	0.045	0.024	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.064
0.667	-0.008	0.02	-0.017	0.045	-0.029	0.088	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.099
0.614	-0.008	0.02	-0.017	0.045	-0.029	0.087	0.044	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.143
0.556	-0.008	0.02	-0.017	0.045	-0.029	0.086	-0.044	0.149	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.203
0.502	-0.008	0.02	-0.016	0.044	-0.028	0.085	-0.043	0.146	0.073	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.272
0.445	-0.008	0.019	-0.016	0.042	-0.027	0.082	-0.042	0.142	-0.059	0.225	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.359
0.390	-0.007	0.018	-0.015	0.04	-0.026	0.078	-0.04	0.135	-0.056	0.215	0.112	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.454
0.334	-0.007	0.017	-0.014	0.038	-0.024	0.073	-0.037	0.126	-0.052	0.2	-0.077	0.318	0.0	0.0	0.0	0.0	0.0	0.0	0.559
0.277	-0.006	0.015	-0.012	0.034	-0.022	0.065	-0.033	0.112	-0.047	0.179	-0.069	0.285	0.168	0.0	0.0	0.0	0.0	0.0	0.669
0.223	-0.005	0.013	-0.011	0.029	-0.019	0.056	-0.028	0.097	-0.04	0.154	-0.059	0.245	-0.112	0.45	0.0	0.0	0.0	0.0	0.768
0.161	-0.004	0.01	-0.008	0.022	-0.014	0.043	-0.022	0.074	-0.031	0.118	-0.046	0.188	-0.086	0.346	0.278	0.0	0.0	0.0	0.869
0.112	-0.003	0.007	-0.006	0.016	-0.01	0.031	-0.016	0.054	-0.023	0.086	-0.033	0.137	-0.063	0.252	-0.235	0.735	0.0	0.0	0.932
0.016	-0.001	0.001	-0.001	0.003	-0.002	0.006	-0.003	0.01	-0.004	0.015	-0.006	0.024	-0.011	0.045	-0.042	0.13	0.833	0.0	0.998
0.001	-0.0	0.0	-0.0	0.0	-0.0	0.001	-0.0	0.002	-0.001	0.003	-0.001	0.004	-0.002	0.007	-0.007	0.022	-4.895	5.867	1.0

Table 10: DPMSolver3S's signal coefficient matrix on Natural Inference framework

time	1.000	0.948	0.892	0.834	0.782	0.727	0.667	0.615	0.560	0.500	0.447	0.391	0.334	0.273	0.217	0.167	0.044	0.009	sum
0.948	0.004	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.004
0.892	-0.004	0.016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.012
0.834	0.019	-0.033	0.037	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.024
0.782	0.019	-0.033	0.037	0.016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.039
0.727	0.019	-0.033	0.037	-0.012	0.052	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.063
0.667	0.019	-0.033	0.037	0.049	-0.078	0.104	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.099
0.615	0.019	-0.033	0.037	0.049	-0.077	0.104	0.042	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.141
0.560	0.019	-0.032	0.036	0.048	-0.076	0.103	-0.024	0.125	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.198
0.500	0.019	-0.032	0.036	0.047	-0.075	0.101	0.093	-0.134	0.219	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.274
0.447	0.018	-0.031	0.035	0.046	-0.073	0.098	0.09	-0.13	0.213	0.089	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.356
0.391	0.017	-0.029	0.033	0.044	-0.069	0.093	0.086	-0.124	0.203	-0.04	0.238	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.452
0.334	0.016	-0.027	0.031	0.041	-0.064	0.087	0.08	-0.115	0.188	0.147	-0.191	0.368	0.0	0.0	0.0	0.0	0.0	0.0	0.559
0.273	0.014	-0.024	0.027	0.036	-0.057	0.077	0.071	-0.102	0.167	0.131	-0.17	0.327	0.178	0.0	0.0	0.0	0.0	0.0	0.675
0.217	0.012	-0.02	0.023	0.031	-0.049	0.065	0.06	-0.087	0.142	0.111	-0.144	0.277	-0.078	0.435	0.0	0.0	0.0	0.0	0.778
0.167	0.01	-0.017	0.019	0.025	-0.039	0.053	0.049	-0.07	0.115	0.09	-0.117	0.225	0.248	-0.336	0.605	0.0	0.0	0.0	0.859
0.044	0.003	-0.005	0.006	0.007	-0.012	0.016	0.015	-0.021	0.035	0.027	-0.035	0.068	0.074	-0.101	0.181	0.73	0.0	0.0	0.987
0.009	0.001	-0.001	0.001	0.002	-0.003	0.004	0.004	-0.006	0.009	0.007	-0.009	0.018	0.02	-0.027	0.048	-1.201	2.132	0.0	0.999
0.001	0.0	-0.0	0.0	0.001	-0.001	0.001	0.001	-0.001	0.002	0.002	-0.002	0.005	0.005	-0.007	0.013	6.607	-10.588	4.963	1.0

E.6 DPMSolver++ coefficient matrix

Table 11: DPMSolverpp2S's signal coefficient matrix on Natural Inference framework

time	1.000	0.946	0.889	0.835	0.778	0.724	0.667	0.614	0.556	0.502	0.445	0.390	0.334	0.277	0.223	0.161	0.112	0.016	sum
0.946	0.005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.005
0.889	0.0	0.012	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.012
0.835	0.0	0.012	0.011	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.023
0.778	0.0	0.012	0.0	0.029	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.041
0.724	0.0	0.012	0.0	0.029	0.024	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.064
0.667	0.0	0.012	0.0	0.028	0.0	0.059	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.099
0.614	0.0	0.012	0.0	0.028	0.0	0.058	0.044	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.143
0.556	0.0	0.012	0.0	0.028	0.0	0.058	0.0	0.105	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.203
0.502	0.0	0.012	0.0	0.028	0.0	0.057	0.0	0.103	0.073	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.272
0.445	0.0	0.011	0.0	0.027	0.0	0.055	0.0	0.1	0.0	0.166	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.359
0.390	0.0	0.011	0.0	0.025	0.0	0.052	0.0	0.095	0.0	0.159	0.112	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.454
0.334	0.0	0.01	0.0	0.024	0.0	0.049	0.0	0.089	0.0	0.147	0.0	0.241	0.0	0.0	0.0	0.0	0.0	0.0	0.559
0.277	0.0	0.009	0.0	0.021	0.0	0.044	0.0	0.079	0.0	0.132	0.0	0.216	0.168	0.0	0.0	0.0	0.0	0.0	0.669
0.223	0.0	0.008	0.0	0.018	0.0	0.037	0.0	0.068	0.0	0.113	0.0	0.185	0.0	0.338	0.0	0.0	0.0	0.0	0.768
0.161	0.0	0.006	0.0	0.014	0.0	0.029	0.0	0.052	0.0	0.087	0.0	0.143	0.0	0.26	0.278	0.0	0.0	0.0	0.869
0.112	0.0	0.004	0.0	0.01	0.0	0.021	0.0	0.038	0.0	0.064	0.0	0.104	0.0	0.189	0.0	0.501	0.0	0.0	0.932
0.016	0.0	0.001	0.0	0.002	0.0	0.004	0.0	0.007	0.0	0.011	0.0	0.018	0.0	0.034	0.0	0.089	0.833	0.0	0.998
0.001	0.0	0.0	0.0	0.0	0.0	0.001	0.0	0.001	0.0	0.002	0.0	0.003	0.0	0.006	0.0	0.015	0.0	0.972	1.0

Table 12: DPMSolverpp3S's signal coefficient matrix on Natural Inference framework

time	1.000	0.948	0.892	0.834	0.782	0.727	0.667	0.615	0.560	0.500	0.447	0.391	0.334	0.273	0.217	0.167	0.044	0.009	sum
0.948	0.004	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.004
0.892	0.025	-0.014	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.012
0.834	0.046	0.0	-0.022	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.024
0.782	0.046	0.0	-0.022	0.016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.039
0.727	0.046	0.0	-0.022	0.085	-0.045	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.063
0.667	0.046	0.0	-0.022	0.144	0.0	-0.068	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.099
0.615	0.045	0.0	-0.022	0.143	0.0	-0.068	0.042	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.141
0.560	0.045	0.0	-0.022	0.142	0.0	-0.067	0.211	-0.111	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.198
0.500	0.044	0.0	-0.021	0.139	0.0	-0.066	0.334	0.0	-0.156	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.274
0.447	0.043	0.0	-0.021	0.135	0.0	-0.064	0.325	0.0	-0.151	0.089	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.356
0.391	0.041	0.0	-0.02	0.129	0.0	-0.061	0.31	0.0	-0.144	0.415	-0.217	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.452
0.334	0.038	0.0	-0.018	0.119	0.0	-0.057	0.288	0.0	-0.134	0.6	0.0	-0.277	0.0	0.0	0.0	0.0	0.0	0.0	0.559
0.273	0.034	0.0	-0.016	0.106	0.0	-0.05	0.255	0.0	-0.119	0.533	0.0	-0.246	0.178	0.0	0.0	0.0	0.0	0.0	0.675
0.217	0.029	0.0	-0.014	0.09	0.0	-0.043	0.217	0.0	-0.101	0.452	0.0	-0.209	0.749	-0.393	0.0	0.0	0.0	0.0	0.778
0.167	0.023	0.0	-0.011	0.073	0.0	-0.035	0.176	0.0	-0.082	0.368	0.0	-0.17	0.962	0.0	-0.445	0.0	0.0	0.0	0.859
0.044	0.007	0.0	-0.003	0.022	0.0	-0.01	0.053	0.0	-0.025	0.11	0.0	-0.051	0.288	0.0	-0.133	0.73	0.0	0.0	0.987
0.009	0.002	0.0	-0.001	0.006	0.0	-0.003	0.014	0.0	-0.007	0.029	0.0	-0.013	0.076	0.0	-0.035	2.235	-1.304	0.0	0.999
0.001	0.0	0.0	-0.0	0.002	0.0	-0.001	0.004	0.0	-0.002	0.008	0.0	-0.004	0.02	0.0	-0.009	2.116	0.0	-1.134	1.0

F SD3's coefficient matrix and inference process visualization

F.1 Coefficient matrix and its corresponding outputs

Table 13: SD3's signal coefficient matrix for Flow Matching Euler sampling

time	1.00	0.99	0.97	0.96	0.95	0.93	0.91	0.90	0.88	0.86	0.84	0.81	0.79	0.76	0.74	0.71	0.68	0.64	0.60	0.56	0.52	0.46	0.41	0.35	0.28	0.20	0.11	0.01
0.99	1.26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.97	1.26	1.33	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.96	1.26	1.33	1.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.95	1.26	1.33	1.4	1.47	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.93	1.26	1.33	1.4	1.47	1.56	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.91	1.26	1.33	1.4	1.47	1.56	1.65	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.90	1.26	1.33	1.4	1.47	1.56	1.65	1.74	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.88	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.86	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.84	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.81	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.79	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.76	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.74	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.71	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.68	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.64	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.60	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.56	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.52	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.46	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.41	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	0.0	0.0	0.0	0.0	0.0	0.0
0.35	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	0.0	0.0	0.0	0.0	0.0
0.28	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	6.93	0.0	0.0	0.0	0.0
0.20	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	6.93	7.82	0.0	0.0	0.0
0.11	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	6.93	7.82	8.89	0.0	0.0
0.01	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	6.93	7.82	8.89	10.2	0.0
0.00	1.26	1.33	1.4	1.47	1.56	1.65	1.74	1.85	1.97	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	6.93	7.82	8.89	10.2	0.89

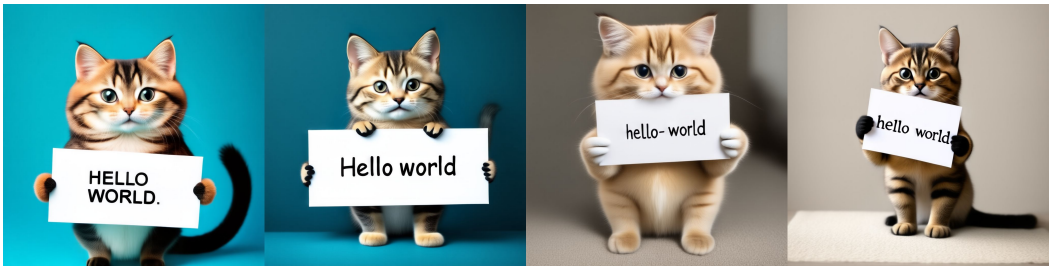
Note that, for readability, the coefficients in Table 13 are the original coefficients multiplied by 100. When using them, they should be normalized to the corresponding Marginal Coefficient for each step. For example, the first row should be normalized to 0.0126, and the second row should be normalized to 0.0259. The usage of Table 14 is the same.

F.2 Mid Self Guidance is better suited for large CFG

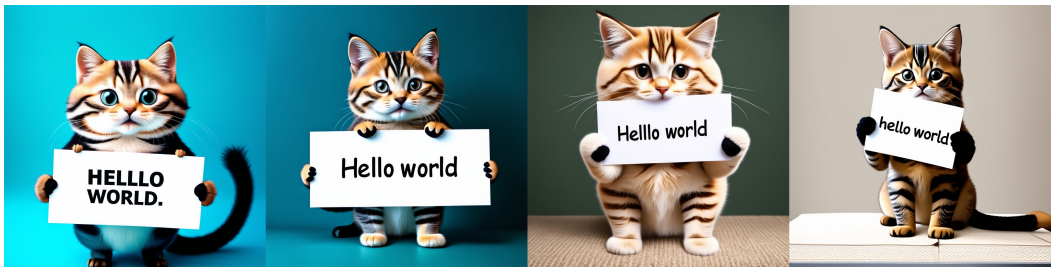
Figure 18 presents a small experiment demonstrating that, the output from Mid Self Guidance significantly outperforms Fore Self Guidance when using large CFG values. Figure 18(a) shows the

Table 14: SD3's signal coefficient matrix with more sharpness

time	1.00	0.99	0.97	0.96	0.95	0.93	0.91	0.90	0.88	0.86	0.84	0.81	0.79	0.76	0.74	0.71	0.68	0.64	0.60	0.56	0.52	0.46	0.41	0.35	0.28	0.20	0.11	0.01
0.99	1.26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.97	1.26	1.33	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.96	1.26	1.33	1.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.95	0.0	1.33	1.4	1.47	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.93	0.0	1.33	1.4	1.47	1.56	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.91	0.0	0.0	1.44	1.56	1.56	1.65	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.90	0.0	0.0	0.0	0.0	1.56	1.65	1.74	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.88	0.0	0.0	0.0	0.0	0.0	1.65	1.74	1.85	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.86	0.0	0.0	0.0	0.0	0.0	1.65	1.74	1.85	1.97	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.84	0.0	0.0	0.0	0.0	0.0	0.0	1.74	1.85	1.97	2.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.81	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.85	1.97	2.1	2.24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.79	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.97	2.1	2.24	2.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.76	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.1	2.24	2.4	2.57	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.74	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.1	2.24	2.4	2.57	2.76	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.71	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.1	2.24	2.4	2.57	2.76	2.98	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.68	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.1	2.24	2.4	2.57	2.76	2.98	3.22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.64	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.1	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.56	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.52	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.46	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	0.0	0.0	0.0	0.0	0.0	0.0	0
0.41	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.24	2.4	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	0.0	0.0	0.0	0.0	0.0	0
0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	0.0	0.0	0.0	0.0	0
0.28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	6.93	0.0	0.0	0.0	0
0.20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	6.93	7.82	0.0	0.0	0
0.11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.57	2.76	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	6.93	7.82	8.89	0.0	0
0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	6.93	7.82	8.89	10.2	0
0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.98	3.22	3.49	3.8	4.15	4.56	5.02	5.56	6.19	6.93	7.82	8.89	10.2	15



(a)



(b)

Figure 17: (a) Result for original Euler sampling (Table 13) (b) Result for adjusted coefficient matrix (Table 14)

outputs for the first three steps of SD3 (cfg=7), and Figures 18(b)−(d) show results from different linear combinations. (b) using negative coefficients, (c) using two positive coefficients, and (d) using three positive coefficients. It is clear that when there are more positive coefficients, the image quality is better.

(a)

(b)

(c)

(d)

Figure 18: (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) $0.32 \cdot x_0^0 + 0.33 \cdot x_0^1 + 0.35 \cdot x_0^2$

F.3 Inference process visualization

Figures 19 and 20 provide a visualization of the complete inference process. The left half shows the inference process using the coefficient matrix from Table 13, and the right half shows the inference process using the coefficient matrix from Table 14. The first column shows the result of Self Guidance, which is also the input image signal to the model. The second column shows the model output without conditioning, the third column shows the conditioned model output, and the fourth column shows the result of Classifier Free Guidance. For each model operation, there is a clear image signal input and image signal output, which greatly enhances intuitive understanding of the operation's purpose and facilitates efficient debugging and problem analysis.

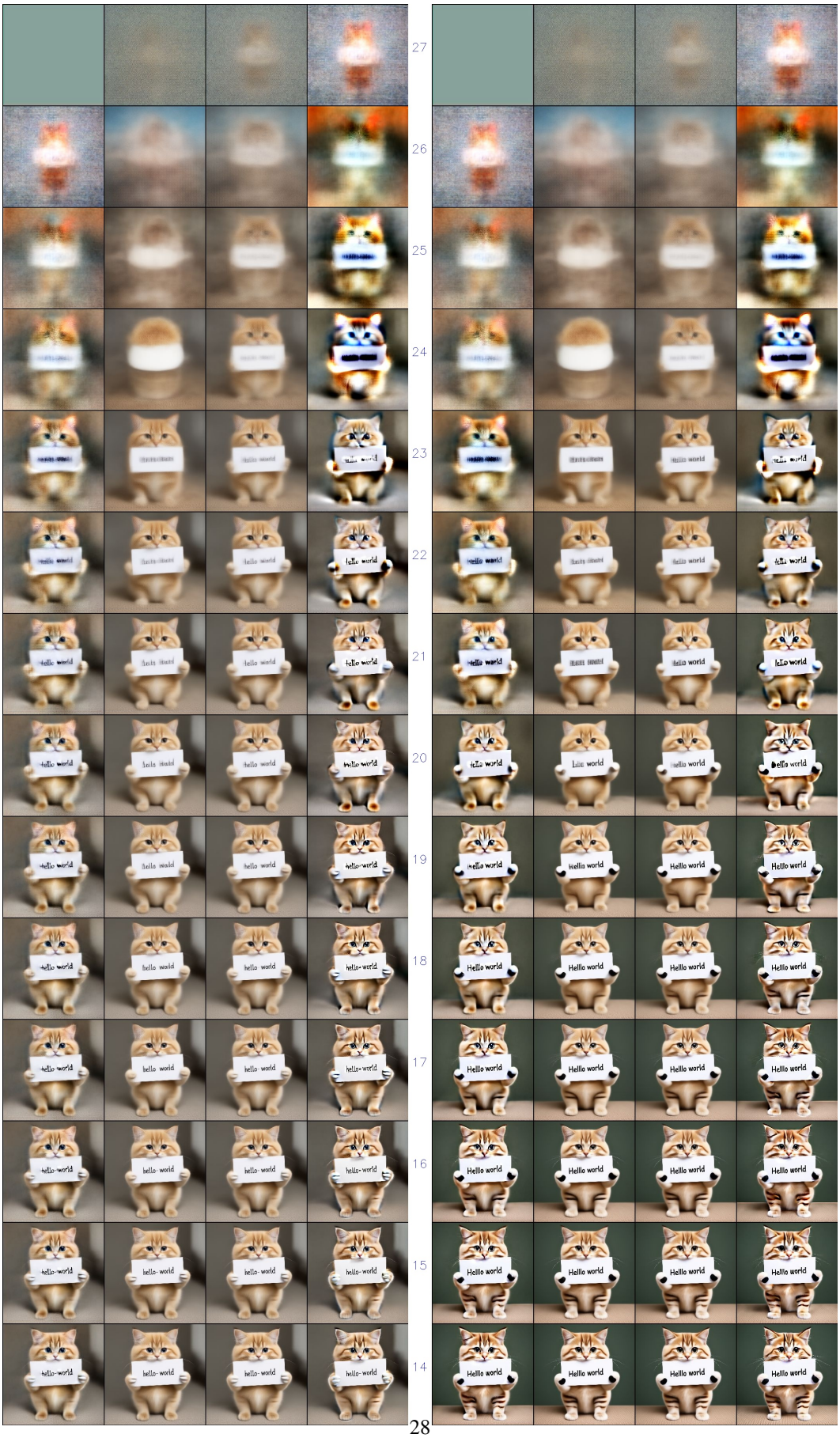


Figure 19: Inference process visualization: first half.

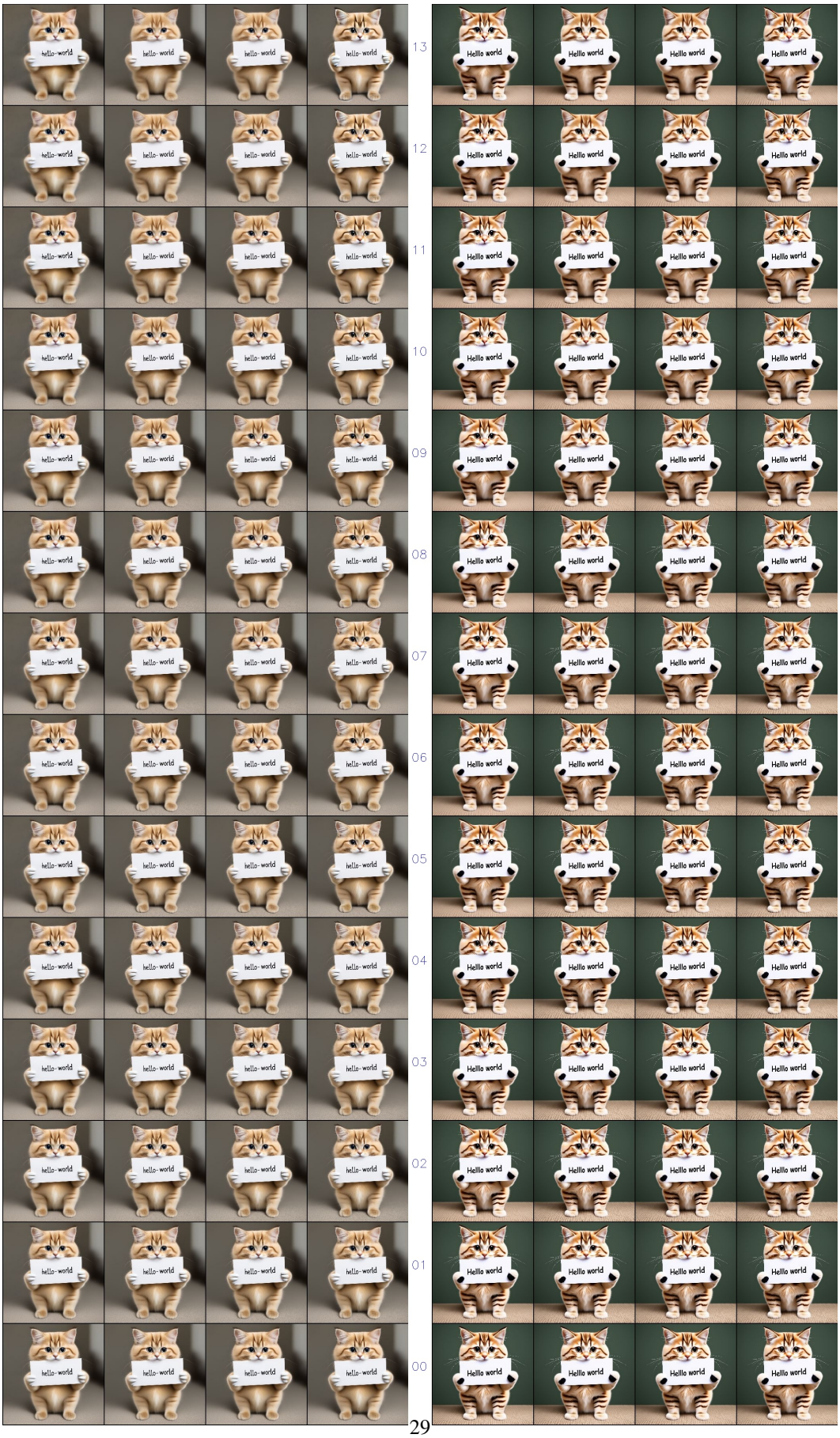


Figure 20: Inference process visualization: second half

G 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	1.000	0.650	0.375	0.176	0.051	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	1.000	0.816	0.650	0.503	0.375	0.266	0.176	0.104	0.051	0.017	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	1.000	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	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	0.25	0	0.82	-0.78	1	0	0	0	0	0	0	0.85
0.126	0	0	0	0	0	0.23	-0.02	0.9	-0.2	1	0	0	0	0	0	0.919
0.084	0	0	0	0	0	0	0.2	-0.04	0.7	-0.4	1	0	0	0	0	0.961
0.051	0	0	0	0	0	0	0	0.17	-0.07	0.62	-0.66	1	0	0	0	0.985
0.026	0	0	0	0	0	0	0	0	0.14	-0.09	0.57	-0.88	1	0	0	0.995
0.009	0	0	0	0	0	0	0	0	0	0.11	-0.11	0.31	-0.49	1	0	0.999
0.001	0	0	0	0	0	0	0	0	0	0	0.08	-0.11	0.24	-0.31	1	1

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction have been carefully written to accurately summarize the paper's core contributions.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5.5 discusses the potential constraints of our proposed inference framework.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All theoretical results presented are accompanied by a clear statement of their underlying assumptions and their complete proofs are provided in Appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides detailed descriptions of experiment information to reproduce the main result.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code is provided in the supplement, along with comprehensive instructions detailing dependencies, setup, and execution steps necessary to faithfully reproduce the key experimental findings.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The paper provides a thorough description of the experimental setup to understand the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The experiments reported are deterministic, yielding identical results across runs with fixed random seed.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The experiments in this paper do not include model training, only model inference, and thus require relatively few computational resources.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We have carefully reviewed the NeurIPS Code of Ethics and confirm that our research fully conforms to its principles in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper is primarily theoretical and is not expected to have significant direct societal impacts.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work does not involve the public release of any new datasets or models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All external assets (datasets, code libraries, pretrained models) used in this paper are properly credited via citation to their original creators/owners.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Our work does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.