Multi-Step Denoising Scheduled Sampling: Towards Alleviating Exposure Bias for Diffusion Models

Zhiyao Ren¹, Yibing Zhan², Liang Ding², Gaoang Wang³, Chaoyue Wang¹, Zhongyi Fan², Dacheng Tao¹

¹The University of Sydney, Australia, ²JD Explore Academy, China, ³Zhejiang University, China

zren0130@uni.sydney.edu.au, zhanyibing@jd.com, liangding.liam@gmail.com, gaoangwang@intl.zju.edu.cn, chaoyue.wang@outlook.com, zyfanzy@foxmail.com, dacheng.tao@sydney.edu.au

Abstract

Denoising Diffusion Probabilistic Models (DDPMs) have achieved significant success in generation tasks. Nevertheless, the exposure bias issue, i.e., the natural discrepancy between the training (the output of each step is calculated individually by a given input) and inference (the output of each step is calculated based on the input iteratively obtained based on the model), harms the performance of DDPMs. To our knowledge, few works have tried to tackle this issue by modifying the training process for DDPMs, but they still perform unsatisfactorily due to 1) partially modeling the discrepancy and 2) ignoring the prediction error accumulation. To address the above issues, in this paper, we propose a multi-step denoising scheduled sampling (MDSS) strategy to alleviate the exposure bias for DDPMs. Analyzing the formulations of the training and inference of DDPMs, MDSS 1) comprehensively considers the discrepancy influence of prediction errors on the output of the model (the Gaussian noise) and the output of the step (the calculated input signal of the next step), and 2) efficiently models the prediction error accumulation by using multiple iterations of a mathematical formulation initialized from one-step prediction error obtained from the model. The experimental results, compared with previous works, demonstrate that our approach is more effective in mitigating exposure bias in DDPM, DDIM, and DPM-solver. In particular, MDSS achieves an FID score of 3.86 in 100 sample steps of DDIM on the CIFAR-10 dataset, whereas the second best obtains 4.78. The code will be available on GitHub.

Introduction

Denoising Diffusion Probabilistic Models (DDPMs) (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020) are generative models, which first destruct data by progressively adding noise and then learn the reverse process for sample generation (Yang et al. 2022). Due to the advantage of unrestricted model structure and stable training, DDPMs have swiftly gained substantial attention and become state-of-the-art approaches in generative tasks, including image generation (Dhariwal and Nichol 2021), text-to-image generation (Nichol et al. 2022; Ramesh et al. 2022; Saharia et al. 2022), text-to-video generation (Singer et al. 2022), and audio generation (Mittal et al. 2021). Recent researches on DDPMs have been primarily focused on augmenting the classical method (Ho, Jain, and Abbeel 2020) in three key areas: efficient sampling (Song, Meng, and Ermon 2021; Kong and Ping 2021; Lu et al. 2022; Salimans and Ho 2022), improved likelihood estimation (Nichol and Dhariwal 2021; Bao et al. 2022; Kingma et al. 2021), and handling multi-modal tasks (Nichol et al. 2022; Mittal et al. 2021). Nevertheless, the exposure bias issue of DDPMs has been generally overlooked.

The exposure bias problem is a prevalent issue, leading to suboptimal performance, in the domain of recurrent processes, such as autoregressive text generation (Ranzato et al. 2016), arising from the disparity between the training and inference processes (Bengio et al. 2015; Zhang et al. 2019; Schmidt 2019). As shown in Fig. 1, in the training process of DDPMs, a real sample $x_0$ is corrupted by introducing Gaussian noise as a Markov chain. The input to the model at step $t$ during training is obtained based on the real sample $x_0$, noise schedule $\alpha_t$, and a random standard Gaussian noise $\epsilon$: $q(x_0 | x_0) = \mathcal{N}(x_0; \sqrt{\alpha_t}x_0, (1 - \bar{\alpha}_t) I)$. In contrast, in the inference process, the input to the model comes from the output of the previous steps $p_0(x_t | x_{t-1}) = \mathcal{N}(x_t; \mu_0(x_{t-1} + 1), \Sigma_0(x_{t-1} + 1))$. The training process sources its input directly from the ground truth, while the inference process derives its input from model predictions with potential errors. The discrepancy between the training and inference, i.e., the exposure bias issue, harms the performance of DDPMs (Deng, Kojima, and}

---

**Figure 1:** The discrepancy between training and inference process. In the training process, the input at time step $t$ is derived from the forward diffusion process of $x_0$. Contrarily, in the inference process, the input at time step $t$ is obtained from the output of the previous step.
A few works (Ning et al. 2023; Deng, Kojima, and Rush 2022) have recently attempted to address the exposure bias in the training of DDPMs, but they still suffer from two problems. First, they partially model the discrepancy between training and inference of DDPMs. Specifically, the Input Perturbation (IP) method (Ning et al. 2023) introduces perturbation in the ground truth samples to simulate the inference of DDPMs. Secondly, they suffer from two problems: (1) the input of the next step of DDPMs is calculated based on the input and output of the model (Gaussian Noise). The input of the next step of SS still contains noise from previous prediction errors. Second, IP and SS only calculate prediction errors in one step for efficiency, and they ignore that the prediction errors would be accumulated through the interaction process and further side influence the performance (2023; 2023).

In light of the above issues, in this paper, we propose a multi-step denoising scheduled sampling (MDSS) strategy to alleviate exposure bias for DDPMs. To comprehensively alleviate the influence of prediction errors, MDSS considers the exposure bias from two aspects, requires the output of the model in the current step, e.g., the Gaussian noise, to be accurately predicted, and the noise influence on the input of the next step to be reduced. To mitigate the prediction error accumulation influence, we model the accumulated prediction errors time-efficiently by using multiple iterations of a mathematical formulation initialized from the one-step prediction error obtained from the model. The process starts with a one-step model prediction to introduce the model noise and uses multiple iterations of the mathematical formulation to model the prediction error accumulation as similarly as possible. In addition, to further reduce the implementation complexity, we validate that our MDSS could improve the performance by finetuning a well-trained DDPM with small retraining steps. We conduct extensive experiments on CIFAR-10 (Krizhevsky, Hinton et al. 2009), ImageNet 64 × 64 (Deng et al. 2009), and LSUN 64 × 64 (Yu et al. 2015) datasets. Compared to previous methods: IP and SS, our MDSS exhibits better generation quality improvements in DDPM, DDIM, and DPM-Solver.

Our contributions are summarized as follows:

- We detailed analyze the discrepancy between training and inference of DDPMs and propose an effective multi-step denoising scheduled sampling (MDSS) strategy to alleviate the exposure bias for DDPMs.
- MDSS comprehensively considers the discrepancy influence of prediction errors on both the output of the model and the output of the calculated input signal per step and efficiently models accumulated prediction error by using multiple iterations of mathematical formulation initialized from the one-step prediction error of the model.
- Extensive experiments were conducted to compare the performance of current works for solving exposure bias in DDPMs. The experimental results demonstrate that our MDSS performs the best.

## Preliminary Knowledge

### Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models (DDPMs) consist of two processes: the forward process corrupts the data through the addition of Gaussian noise, and the reverse process reverts the forward process and generates data from standard Gaussian noise (Ho, Jain, and Abbeel 2020; Nichol and Dhariwal 2021).

Given data distribution \( q(x_0) \) and the noise schedule \( \beta_1, \beta_2, \ldots, \beta_T \), the forward process corrupts the data as a Markov chain:

\[
q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \quad (1) 
\]

\[
q(x_{1:T} | x_0) = \prod_{t=1}^{T} q(x_t | x_{t-1}) \quad (2)
\]

When \( T \) is large enough, we can achieve \( x_T \sim \mathcal{N}(0, I) \).

As mentioned in (Ho, Jain, and Abbeel 2020), we can sample to any time step directly using input \( x_0 \sim q(x_0) \):\[
q(x_t | x_0) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_0, (1-\alpha_t)I) \quad (3)
\]

where \( \alpha_t = 1-\beta_t \) and \( \tilde{\alpha}_t = \prod_{i=1}^{t} \alpha_i \). Utilizing the reparameter skill, we are able to sample any step \( x_t \) with \( \epsilon \sim \mathcal{N}(0, I) \):

\[
x_t = \sqrt{\alpha_t}x_0 + \sqrt{1-\alpha_t} \epsilon \quad (4)
\]

Using Bayes theorem, we can obtain the posterior reverse process distribution \( q(x_{t-1} | x_t, x_0) \):

\[
q(x_{t-1} | x_t, x_0) = \mathcal{N}(x_{t-1}; \mu_t(x_t, x_0), \tilde{\beta}_t I) \quad (5)
\]

\[
\mu_t(x_t, x_0) := \frac{\sqrt{\alpha_{t-1}}\beta_t x_0 + \sqrt{\beta_t} (1-\alpha_{t-1}) x_t}{1-\alpha_t} \quad (6)
\]

\[
\tilde{\beta}_t := \frac{1-\alpha_t}{1-\alpha_{t-1}} \beta_t \quad (7)
\]

In the inference process, \( x_0 \) is not available and \( q(x_{t-1} | x_t) \) depends on the entire data distribution. Consequently, the reverse process is defined as a parameterized process:

\[
p_\theta(x_{t-1} | x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (8)
\]

\[
p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} | x_t) \quad (9)
\]

Instead of learning the mean of reverse process, Ho, Jain, and Abbeel find that predicting the noise \( \epsilon \) is a better option. Empirically, they propose simplifying the loss function as follows:

\[
L_{\text{simple}} = \mathbb{E}_{x_0, \epsilon \sim \mathcal{N}(0, I)} [ \| \epsilon - \epsilon_\theta(x_t, t) \|^2 ] \quad (10)
\]

The training and inference algorithms are described in Alg. 1 and Alg. 2, respectively.
Algorithm 1: DDPMs Standard Training Process
1: repeat
2: \(x_{0} \sim q(x_{0});\)
3: \(t \sim \mathcal{U}\{1, \cdots, T\};\)
4: \(\epsilon \sim \mathcal{N}(0, I);\)
5: Take gradient descent step on \(\nabla_{\theta} \mathbb{E}[\epsilon - \epsilon_{t}(\sqrt{\alpha_{t}}x_{0} + \sqrt{1 - \alpha_{t}}\epsilon, t)]^{2};\)
6: until converged

Algorithm 2: DDPM Standard Inference Process
1: \(x_{T} \sim \mathcal{N}(0, I);\)
2: for \(t = T, \cdots, 1\) do
3: \(z \sim \mathcal{N}(0, I)\) if \(t > 1\), else \(z = 0;\)
4: \(x_{t-1} = \frac{1}{\sqrt{\alpha_{t}}} \left( x_{t} - \frac{\alpha_{t}}{1 - \alpha_{t}} \epsilon_{t}(x_{t}, t) \right) + \sigma_{t}z;\)
5: end for
6: return \(x_{0}\)

Fast Inference Methods: DDIM and DPM-Solver

Many methods demonstrate improvement in the outcomes of fewer steps inference, with DDIM and DPM-Solver being the most prevalent. DDIM (2021) proposes a more generalized non-Markov process with the same marginal distribution of \(x_{t}\). The inference process changes such that the model first predicts the normal sample \(x_{0}\), and then, the normal sample \(x_{0}\) is used to estimate the next step in the chain. The reverse process can be sampled as follows:

\[
x_{t-1} = \frac{\sqrt{\alpha_{t-1}}}{\sqrt{\alpha_{t}}} \left( x_{t} - \frac{\alpha_{t}}{1 - \alpha_{t}} \epsilon_{t}(t) (x_{t}) \right) + \frac{\sigma_{t}}{\sqrt{1 - \alpha_{t}}} \epsilon_{t} + \sigma_{t}z; \tag{11}
\]

where \(\sigma_{t} = \sqrt{(1 - \alpha_{t-1}) / (1 - \alpha_{t}) \sqrt{1 - \alpha_{t}} / \alpha_{t-1}}.\)

DPM-Solver (Lu et al. 2022) proposes an exact formulation of the solution of diffusion ODEs by Taylor expansion (first order to third order):

\[
x_{t_{i}-1} = \frac{\sqrt{\alpha_{t_{i}}}}{\sqrt{\alpha_{t_{i}-1}}} \bar{x}_{t_{i}-1} - \sqrt{\alpha_{t_{i}}} \sum_{n=0}^{k-1} \epsilon_{t_{i}}(n) \left( \bar{x}_{t_{i-1}}, \lambda_{t_{i-1}} \right) \int_{\lambda_{t_{i}}}^{\lambda_{t_{i}}} e^{-\lambda} \left( \lambda - \lambda_{t_{i-1}} \right)^{n} n! d\lambda + o(k^{1+1}) \tag{12}
\]

where \(\lambda_{t_{i}} = \log(\sqrt{\alpha_{t_{i}}}/\sqrt{1 - \alpha_{t}})\) (one half of the log-SNR) and \(h_{t_{i}} = \lambda_{t_{i}} - \lambda_{t_{i}-1}.\)

More introductions of DDIM and DPM-Solver can be found in Appendix.

Multi-step Denosing Scheduled Sampling

This section presents our multi-step denosing scheduled sampling (MDSS). As shown in line 5 in Alg. 1 and line 4 in Alg. 2, the inputs of the training and inference are different. Specifically, the input of the training process originates from the forward process. When \(x_{0}\), noise schedule, time step \(t\), and Gaussian noise \(\epsilon \sim \mathcal{N}(0, I)\) are deterministic, \(x_{t}\) is obtained by Eq. 4 and is thus also deterministic. However, the input of the inference process comes from the sampling results of previous steps, containing non-negligible errors, and is not exposed to the model during training. Besides, the subsequent multiple-step inference process will continuously amplify errors, further impacting the final sampling outcomes. Therefore, we design MDSS to alleviate the exposure bias issue in training by exposing and denoising the accumulated prediction errors of the inference process.

In the remaining part, we first elaborate on the scheduled sampling with denoising for DDPMs and then present the modeling of prediction errors with accumulation. Next, we introduce our algorithm, and last, we compare our MDSS with current methods towards exposure bias for DDPMs, i.e., IP and SS.

Scheduled Sampling with Denosing

We first formulate and analyze the influence of prediction errors in one step. For simplicity, we suppose the input with prediction errors is represented as:

\[
\hat{x}_{t} = x_{t} + \xi \tag{13}
\]

where \(\hat{x}_{t}\) is the input of the current step with noise, \(x_{t}\) is the ground-truth input without noises, and \(\xi\) is the prediction errors modeling from the inference process. Here, we only consider additive noise following IP and SS. The discussion of other types of noises remains a challenge for future work.

According to the equation of inference process in line 4 of Alg. 2, we can obtain subsequent inference step with noise:

\[
\hat{x}_{t-1} = \frac{1}{\sqrt{\alpha_{t}}} \left( \hat{x}_{t} - \frac{1 - \alpha_{t}}{\sqrt{1 - \alpha_{t}}} \epsilon_{t}(\hat{x}_{t}, t) \right) \tag{14}
\]

where we ignore the variance item for simplicity since it is given directly and devoid of model prediction noise in most of works (Ho, Jain, and Abbeel 2020).

It can be observed from Eq. 14 that there are two types of influence for the output of current step: the model prediction and the input signal. Previous methods only consider model prediction. In contrast, we comprehensively mitigate both influences in the sampling process.

For influence within the model prediction, we mitigate it by training with noise-free training objective. We utilize \(x_{t}\), which contains no noise, to obtain the training objective. \(x_{t}\) can be derived by sampling from the posterior distribution \(q(x_{t-1} | x_{t}, x_{0})\). By replacing \(x_{0}\) using ground truth \(\epsilon_{t+1}\) in Eq. 6, we can obtain:

\[
x_{t} = \hat{\mu}_{t+1}(x_{t+1}, x_{0}) = \sqrt{\alpha_{t+1}}(1 - \alpha_{t}) x_{t+1} + \sqrt{\alpha_{t} \beta_{t+1}} x_{0} \tag{15}
\]

\[
= \frac{1}{\sqrt{\alpha_{t+1}}} \left( x_{t+1} - \frac{\beta_{t+1}}{\sqrt{(1 - \alpha_{t+1})}} \epsilon_{t+1} \right)
\]
The meaning of posterior distribution is sampling to next step, when the model prediction $\epsilon_0$ equals the ground truth $\epsilon$. Hence, $x_t$ derived from the posterior distribution represents the noise-free training result. Utilizing the variant of Eq. 4, we obtain the noise-free training object with $x_t$ and $x_0$:

$$
\epsilon_t = \frac{x_t - \sqrt{\alpha_t}x_0}{\sqrt{1 - \alpha_t}}
$$

(16)

For influence within the input signal, we incorporate an extra denoising channel into the model’s output to achieve the denoising task. During the training process, the additional channel is trained to predict the noise from the input containing noise. The training objective is the different between input from model prediction and posterior distribution calculation:

$$
\xi_t = \hat{x}_t - x_t
$$

(17)

During the inference process, the input signal is denoised by subtracting the output of the denoising channel.

In summary, the loss function for the entire training process is given as:

$$
\|\epsilon_t - \epsilon_0(\hat{x}_t, t)\|^2 + \|\xi_t - \xi_0(\hat{x}_t, t)\|^2
$$

(18)

More details of scheduled sampling with denoising can be found in Appendix.

**Prediction Errors with Multi-step Accumulation**

Previous methods (Ning et al. 2023; Deng, Kojima, and Rush 2022) generally ignore the accumulation of prediction errors: IP uses Gaussian noise, and SS only considers one-step prediction error for time efficiency. Our proposed MDSS tries to model the prediction errors with multi-step accumulations. One possible solution is to obtain the signal of time $t$ based on the iteration process of Alg. 2. However, such an intuitive manner requires much calculation and is ineffectively applied to the training process. Therefore, MDSS models the accumulated prediction errors as similarly as possible by using multiple iterations of mathematical formulation initialized from one-step prediction error obtained from the model. Here, the one-step prediction error from the model is used to introduce model noises, and the multiple iterations of the mathematical formulation are used to model the accumulation quickly.

Specifically, we first obtain the one-step prediction error from the model by using:

$$
x_{t+k+1} = \sqrt{\alpha_{t+k+1}}x_0 + \sqrt{1 - \alpha_{t+k+1}}\epsilon
$$

(19)

$$
x_{t+k} = \frac{1}{\sqrt{\alpha_{t+k+1}}} (x_{t+k+1} - \frac{1 - \alpha_{t+k+1}}{\sqrt{1 - \alpha_{t+k+1}}} \epsilon_0 (x_{t+k+1}, t + k + 1)
$$

(20)

Then, we simulate the error accumulation by using the posterior distribution, which conducts the reverse process without model prediction. For further simulating of one step, we can sample by posterior directly. In order to calculate multi-step sample of posterior, we can sample as:

$$
x_t = \gamma_t x_{t+k} + \omega_t x_0 + \frac{\sqrt{\alpha_t}}{1 - \alpha_{t+1}} x_0
$$

(21)

**Algorithm 3: Multi-step Denoising Scheduled Sampling**

1: repeat
2: $x_0 \sim q(x_t)$;
3: $t \sim U\{1, \cdots, T - k - 1\}$;
4: $\epsilon \sim \mathcal{N}(0, 1)$;
5: $x_{t+k+1} = \sqrt{\alpha_{t+k+1}}x_0 + \sqrt{1 - \alpha_{t+k+1}}\epsilon$;
6: $\hat{x}_{t+k} = \frac{1}{\sqrt{\alpha_{t+k+1}}} (x_{t+k+1} - \frac{1 - \alpha_{t+k+1}}{\sqrt{1 - \alpha_{t+k+1}}} \epsilon_0 (x_{t+k+1}, t + k + 1)$;
7: $x_{t+k} = \frac{1}{\sqrt{\alpha_{t+k+1}}} (x_{t+k+1} - \frac{1 - \alpha_{t+k+1}}{\sqrt{1 - \alpha_{t+k+1}}} \epsilon_0 (x_{t+k+1}, t + k + 1)$;
8: if $k > 0$ then
9: $x_t = \gamma_t x_{t+k} + \omega_t x_0 + \frac{\sqrt{\alpha_t}}{1 - \alpha_t} x_0$;
10: $x_t = \gamma_t x_{t+k} + \omega_t x_0 + \frac{\sqrt{\alpha_t}}{1 - \alpha_t} x_0$;
11: end if
12: $\xi_t = \hat{x}_t - x_t$;
13: $\epsilon_t = \frac{x_t - \sqrt{\alpha_t} x_0}{\sqrt{1 - \alpha_t}}$;
14: Take gradient descent step on
15: until converged

$$
\gamma_t = \prod_{i=t}^{t+k-1} \frac{\sqrt{\alpha_i + 1} (1 - \alpha_i)}{1 - \alpha_{i+1}}
$$

(22)

$$
\omega_t = \sum_{j=t}^{t+k-2} \prod_{m=t+1}^{j} \left[ \frac{\sqrt{\alpha_m + 1} (1 - \alpha_m)}{1 - \alpha_{m+1}} \right] \prod_{m=t}^{j+2} \left[ \frac{\sqrt{\alpha_{m+1} + 1} (1 - \alpha_{m+1})}{1 - \alpha_{m+2}} \right]
$$

(23)

Through above process, step $t + k$ with model errors is quickly simulated for $k$ steps errors accumulation to obtain, step $t$, the inputs for training. The disparity between the input with noise and the ground truth expends, mimicking the noise accumulation observed in the sampling process.

**Algorithm**

The training process by introducing the prediction errors with multi-step accumulation in the scheduled sampling with denoising, the algorithm of MDSS is described in Alg. 3. The training process still follows scheduled sampling, which introduces prediction errors by the scheduled ratio. For brevity, the algorithm only outlines the steps of introducing errors. More details can be found in Appendix. $k$ represents the number of accumulation using mathematical formulation. When $k = 0$, it defaults to a Single-step denoising scheduled sampling (SDSS).

Furthermore, we validate our MDSS could be applied to a well-trained model. In such a manner, only requiring a small number of retraining steps, MDSS can improve the performance of a given DDPM, saving a lot of time and computational resources when compared with training from scratch.

**Discussion**

In this subsection, we compare our proposed approach with IP and SS, two state-of-the-art methods for addressing exposure bias for DDPMs, highlighting the reasons behind the superior sampling outcomes achieved by our method.
Table 1: The comparison of MDSS with IP and SS. The symbol ✓ denotes that the process is analytically accurate and involved, while ✗ indicates that the process is either analytically incorrect or not covered.

Table 2: The Mean Squared Errors between the model prediction and the ground truth for different methods.

Figure 2: The Mean Squared Errors between the model prediction and the ground truth for different methods.

Experiment

Experimental Setup

We evaluate our method across unconditional image generation tasks on three datasets: CIFAR-10 (Krizhevsky, He, and Hinton et al. 2009), ImageNet 64×64 (Deng et al. 2009), and LSUN tower 64×64 (Yu et al. 2015). For the CIFAR-10 and ImageNet 64×64 datasets, we fine-tune on the well-trained iDDPM (Nichol and Dhariwal 2021) models, and for LSUN tower 64×64, we keep training on the ADM (Ho and Salimans 2022) model. We employ the Frechet Inception Distance (FID) (Heusel et al. 2017) to evaluate the quality of the generated images. In order to visually show the effect of MDSS on image synthesis, we set the same random seed in the sampling phase to ensure a similar trajectory for all methods. More details regarding training hyperparameters, network architecture, FID evaluating settings, CLIP-FID (Kynkaanniemi et al. 2022; Rangwani et al. 2023) results and qualitative comparison can be found in Appendix.

Main Comparison

In this section, we compare MDSS with IP and SS on DDPM, DDIM, and DPM-Solver. For DDPM and DDIM, we sample 30, 100, and 250 steps. For DPM-Solver, we sample 10, 20, and 50 steps. In the following results, we highlight the best result and underline the second-best result.

The results of DDPM are shown in Table 2. We can draw the following conclusions: 1) Exposure bias drops the generation performance of DDPMs, and MDSS outperforms SS and yields competitive results with IP. 2) We find that SDSS outperforms MDSS when sampling steps are larger, while MDSS performs exceptionally well with fewer sampling steps. This may be because the noise accumulation in multi-step is more effective at mitigating the fewer-step sampling process, which is prone to more significant errors.

The results of DDIM are shown in Table 3. We can draw conclusions that: 1) Our method significantly enhances the sampling results and achieves a greater FID improvement than DDPM sampling. It is important to note that the IP method yielded subpar results in DDIM. This validates that...
The results of DDPM and DDIM are shown in Table 3. The effect of DDPM and DDIM on CIFAR-10, ImageNet, and LSUN with varying inference steps is shown in Table 2 and Table 3. The Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)

Ablation Study

In this subsection, we conduct extensive ablation studies on the CIFAR-10 dataset to elucidate the impact of methodological components within our method.

The effect of denoising. We first assess the impact of denoising the input signal. We compare the performance of SDSS and SDSS without denoising the input signal (SDSS w/o). The results of DDPM and DDIM are shown in Table 4.

Table 2: DDPM results on CIFAR-10, ImageNet, and LSUN with varying inference steps.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Steps</th>
<th>DDPM</th>
<th>IP</th>
<th>SS</th>
<th>SDSS</th>
<th>MDSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>100</td>
<td>3.72</td>
<td>3.25</td>
<td>3.62</td>
<td>3.47</td>
<td>3.49</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>3.23</td>
<td>3.02</td>
<td>3.17</td>
<td>3.08</td>
<td>3.62</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>34.35</td>
<td>34.52</td>
<td>32.60</td>
<td>32.32</td>
<td>31.76</td>
</tr>
<tr>
<td>ImageNet</td>
<td>100</td>
<td>25.32</td>
<td>24.88</td>
<td>23.86</td>
<td>23.22</td>
<td>24.28</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>24.88</td>
<td>24.24</td>
<td>22.82</td>
<td>22.48</td>
<td>23.69</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>5.98</td>
<td>5.52</td>
<td>5.76</td>
<td>5.72</td>
<td>5.57</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>2.32</td>
<td>2.27</td>
<td>2.25</td>
<td>2.24</td>
<td>2.26</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>2.22</td>
<td>1.95</td>
<td>1.93</td>
<td>1.90</td>
<td>1.92</td>
</tr>
</tbody>
</table>

Table 3: DDIM results on CIFAR-10, ImageNet, and LSUN with varying inference steps.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Steps</th>
<th>DDPM</th>
<th>IP</th>
<th>SS</th>
<th>SDSS</th>
<th>MDSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>100</td>
<td>4.99</td>
<td>6.95</td>
<td>4.78</td>
<td>4.53</td>
<td>3.86</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>4.48</td>
<td>8.60</td>
<td>4.11</td>
<td>3.94</td>
<td>3.92</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>28.11</td>
<td>43.74</td>
<td>27.82</td>
<td>27.47</td>
<td>27.18</td>
</tr>
<tr>
<td>ImageNet</td>
<td>100</td>
<td>25.33</td>
<td>49.73</td>
<td>24.81</td>
<td>24.75</td>
<td>24.96</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>24.29</td>
<td>55.22</td>
<td>23.69</td>
<td>23.44</td>
<td>23.86</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>4.26</td>
<td>37.38</td>
<td>4.01</td>
<td>3.94</td>
<td>3.84</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>2.96</td>
<td>27.20</td>
<td>2.79</td>
<td>2.61</td>
<td>2.67</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>4.48</td>
<td>32.92</td>
<td>4.38</td>
<td>4.56</td>
<td>4.13</td>
</tr>
</tbody>
</table>

Table 4: DPM-Solver-1, 2, 3 on CIFAR-10 with varying inference steps.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Steps</th>
<th>DDPM</th>
<th>IP</th>
<th>SS</th>
<th>SDSS</th>
<th>MDSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDPM</td>
<td>10</td>
<td>26.08</td>
<td>41.96</td>
<td>25.39</td>
<td>24.96</td>
<td>12.11</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>6.03</td>
<td>14.43</td>
<td>5.99</td>
<td>5.89</td>
<td>4.84</td>
</tr>
<tr>
<td>DPM-Solver-1</td>
<td>10</td>
<td>10.43</td>
<td>42.03</td>
<td>13.04</td>
<td>11.98</td>
<td>12.149</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>3.28</td>
<td>11.39</td>
<td>4.04</td>
<td>4.02</td>
<td>4.845</td>
</tr>
<tr>
<td>DPM-Solver-2</td>
<td>10</td>
<td>5.99</td>
<td>33.22</td>
<td>6.43</td>
<td>6.34</td>
<td>7.80</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>4.07</td>
<td>13.42</td>
<td>4.23</td>
<td>4.01</td>
<td>5.51</td>
</tr>
<tr>
<td>DPM-Solver-3</td>
<td>10</td>
<td>5.99</td>
<td>33.22</td>
<td>6.43</td>
<td>6.34</td>
<td>7.80</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>4.07</td>
<td>13.42</td>
<td>4.23</td>
<td>4.01</td>
<td>5.51</td>
</tr>
</tbody>
</table>

Table 5: Comparison of denoising input signal on DDPM and DDIM.

Table 6: Comparison of denoising input signal on DDPM and DDIM.

The effect of multi-step training. We discuss the effect of prediction error accumulation by conducting experiments with varying steps, k: 4, 10, 20, and 50. The DDPM and DDIM inference results are presented in Table 6 and Table 7. In DDPM, the multi-step training approach only offers improvements at fewer inference steps. This might be due to the accumulation of errors in fewer inferences is more significant and can be mitigated by MDSS. Incorporating more steps in MDSS also leads to worse DDPM results. For DDIM, performance is enhanced using multi-step training. Significant results can also be obtained at certain sampling steps using longer multi-steps. This indicates that there is more significant noise in DDIM, and therefore MDSS can be used for more sampling steps and longer error accumulation. Choosing the number of multi-step training steps requires careful consideration to prevent exacerbating exposure bias. We advise beginning with a conservative number of steps, such as four steps, and incrementally increasing it.

Number of iterations in continue training. In our experiments, we fine-tune a well-trained DDPM using MDSS. In this section, we conduct experiments on the CIFAR-10 dataset, assessing the result of different training iterations. We calculate the FID of 250 sampling steps on every 5,000 iterations, and the result is shown in Fig. 3. The FID experiences a sharp decline in the initial phases of training and begins to converge after approximately 20,000 iterations, but we do not notice a decline if MDSS is not used.
training process, the model is not exposed to its own pre-

dictions but given ground truth. However, during the sam-
pling phase, the word predicted at a previous moment is used to predict the subsequent word. This discrepancy be-
tween training and sampling leads to inaccurate sampling. The Data As Demonstrator (DAD) (2015) approach tackles this issue by feeding both ground truth words and pre-
dicted words during the training process. Scheduled sam-
pling (2015), on the other hand, replaces the teacher-forcing training method with a biased sampling approach that emu-
lates the sampling process based on its own predictions. This paper discusses the Exposure bias in DDPMs.

To avoid the exposure bias problem caused by autoregres-
sive factorization, another potential way is switching to non-
autoregressive generation which has been validated in the
field of natural language processing (2018; 2021b; 2021a; 2022) and will be explored in the future.

### Conclusion

In this paper, we propose a novel training method called multi-step denoising scheduled sampling (MDSS) to miti-
gate the exposure bias issue. Specifically, MDSS 1) comprehensively denoises the errors in model prediction and input signal and 2) efficiently models the prediction error accu-
mulation by mathematical formulation. Our method can be
plugged into any existing DDPMs, requiring merely a few additional training iterations on the well-trained model. The experiments showcase that MDSS achieves better results in alleviating exposure bias problems compared with state-of-the-art works: IP and SS.

Even though our method achieves excellent results in both
DDPM and DDIM inference, it is not well-compatible with the DPM-Solver method. Besides, we assume that the noise of model prediction is additive. We leave the discussion of
DPM-Solver and other types of noises as our future work.

---

### Table 6: Comparison of multi-step training using different steps on DDPM.

<table>
<thead>
<tr>
<th>steps</th>
<th>single-step</th>
<th>4 steps</th>
<th>10 steps</th>
<th>20 steps</th>
<th>50 steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 steps</td>
<td>6.92</td>
<td>5.25</td>
<td><strong>4.45</strong></td>
<td>4.52</td>
<td>6.98</td>
</tr>
<tr>
<td>100 steps</td>
<td>4.53</td>
<td>3.86</td>
<td>4.04</td>
<td>4.05</td>
<td>4.61</td>
</tr>
<tr>
<td>250 steps</td>
<td>3.94</td>
<td>3.92</td>
<td>3.96</td>
<td><strong>3.79</strong></td>
<td>4.09</td>
</tr>
</tbody>
</table>

Table 7: Comparison of multi-step training using different steps on DDIM.

<table>
<thead>
<tr>
<th>steps</th>
<th>single-step</th>
<th>4 steps</th>
<th>10 steps</th>
<th>20 steps</th>
<th>50 steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 steps</td>
<td>7.92</td>
<td><strong>5.54</strong></td>
<td>5.92</td>
<td>6.70</td>
<td>8.31</td>
</tr>
<tr>
<td>100 steps</td>
<td>3.47</td>
<td><strong>3.49</strong></td>
<td>4.20</td>
<td>3.64</td>
<td>4.12</td>
</tr>
<tr>
<td>250 steps</td>
<td><strong>3.08</strong></td>
<td>3.62</td>
<td>3.75</td>
<td>3.58</td>
<td>3.32</td>
</tr>
</tbody>
</table>

Table 8: Comparison of SDSS/MDSS with/without modification based on DDIM sampling.

<table>
<thead>
<tr>
<th>steps</th>
<th>SDSS</th>
<th>DDIM</th>
<th>SDSS</th>
<th>MDSS</th>
<th>DDIM</th>
<th>MDSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 steps</td>
<td>6.92</td>
<td><strong>5.42</strong></td>
<td>5.25</td>
<td><strong>4.82</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100 steps</td>
<td>4.53</td>
<td><strong>4.21</strong></td>
<td><strong>3.86</strong></td>
<td>3.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>250 steps</td>
<td><strong>3.94</strong></td>
<td>4.51</td>
<td><strong>3.92</strong></td>
<td>4.63</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### Discussion of Modifying MSDD Based on DDIM

Many methods modify the inference process to achieve im-
proved results when utilizing fewer sampling steps. For ex-
ample, while DDIM and DDPM undergo identical training
process, their inference methods are a little distinct. Ana-
lytically, by utilizing DDIM in Eq. 11 instead of DDPM in
scheduled sampling, we can achieve a noise distribution that
more closely mirrors the actual DDIM inference process. We
compare the effect of SDSS and MDSS with the DDIM re-
vised version, and the results are shown in Table. 8. It can
be observed that with a modification, the performance of
SDSS/MDSS is further improved when sampling steps are
few. We suggest that When leveraging DDIM for quick sam-
pling with minimal steps, we can employ the DDIM sched-
uled sampling to boost the quality of the results. We can
adopt a similar approach with the DPM-Solver and we leave
this as our feature work.

### Related Work

**Denoising Diffusion Probabilistic Models** Several en-
hancements have been proposed based on the Denoising Diff-
fusion Probabilistic Model (DDPM) (2020). For instance,
Nichol and Dhariwal introduced the cosine noise sched-
ule and a method for learning variances \( \Sigma_t \). Dhariwal and Nichol proposed additional classifier guidance and an im-
proved U-net model, demonstrating that the diffusion model
achieve superior image quality. Ho and Salimans proposed a classifier-free guidance
that can achieve state-of-the-art results without necessitating
the training of an additional classifier.

**Exposure bias** Exposure bias is a prevalent issue in re-
current processes, arising due to the teacher-forcing training
method (2015; 2016; 2019; 2019). Throughout the entire
training process, the model is not exposed to its own pre-

References


