Towards Better Expected Generation of Diffusion Models

ZHIYAO REN

Supervisor: Prof. Dacheng Tao

A thesis submitted in fulfilment of
the requirements for the degree of
Master of Philosophy

School of Computer Science
Faculty of Engineering
The University of Sydney
Australia

July 2024

To my parents and grandparents.

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Zhiyao Ren
School of Computer Science
Faculty of Engineering
The University of Sydney

Authorship Attribution Statement

This thesis was conducted at the University of Sydney, under the supervision of Prof. Dacheng Tao, between 2023 and 2024. This thesis contains several chapters that were previously published by Zhiyao Ren as the first author in the following publications:

(1) **Zhiyao Ren**, Yibing Zhan, Liang Ding, Gaoang Wang, Chaoyue Wang, Zhongyi Fan, and Dacheng Tao. Multi-Step Denoising Scheduled Sampling: Towards Alleviating Exposure Bias for Diffusion Models. In: *Proceedings of the AAAI Conference on Artificial Intelligence, 38(5), 4667-4675*. 2023, pp. 24345–24354. Presented in Chapter 3. I designed the research, implemented the systems, conducted the experiments, and wrote the draft of the paper.

(2) **Zhiyao Ren**, Yibing Zhan, Baosheng Yu, and Dacheng Tao. Reverse Prompt: Cracking the Recipe Inside Text-to-Image Generation. Presented in Chapter 4. I designed the research, implemented the systems, conducted the experiments, and wrote the draft of the paper.

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Zhiyao Ren Date

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Dacheng Tao	Date

Acknowledgements

Throughout my MPhil course, I have received a great deal of support and assistance from my supervisor, collaborators, colleagues, friends, and family.

I am profoundly grateful to my supervisor, Prof. Dacheng Tao, for providing me with the opportunity to undertake this project and for his invaluable guidance and support throughout my research journey. Your insights and encouragement have been pivotal to my development.

I would like to extend my sincere thanks to my co-supervisor, Dr. Yibing Zhan, for the exceptional kindness, unwavering support, and insightful discussions. Thank you for your patient guidance in teaching me how to become an outstanding researcher.

My gratitude also goes to my collaborators, Dr. Baosheng Yu, Dr. Liang Ding, Prof. Gaoang Wang, Dr. Chaoyue Wang and Mr. Zhongyi Fan. We accomplished excellent works together, and I could not have completed this thesis without your involvement.

I would like to also thank all my colleagues and friends, for your help and encouragement. Conducting research together at the University of Sydney has been an unforgettable experience.

In the end, my deepest gratitude goes to my family. Their unwavering love and support have been the cornerstone of my journey. Their constant encouragement and belief in my potential have provided me with the strength and motivation to persevere and succeed.

Abstract

Diffusion Models have achieved remarkable success in generative tasks; however, their generated results can still deviate from people's expectations, impacting the user experience. These issues may stem from two unresolved problems encountered during the training and application stages. During the training stage, there is a discrepancy between training and inference generation process, known as the exposure bias issue. This issue may affects the quality of the expected generation. During the application stage, the current use of Diffusion Models typically relies on textual prompt guidance; however, manually designing these prompts is complex and time-consuming. Users often struggle to accurately describe their ideas with prompts, leading to results that do not meet expectations. In this thesis, we primarily focus on the task of image generation, aiming to enable Diffusion Models to generate more expected results by 1) alleviating exposure bias problem and 2) decoding textual prompts from existing reference images to help people design better prompts. Our research significantly improves the quality of expected image generation in Diffusion Models and provides new insights for future research.

Contents

List of Abbreviations

DMs	Diffusion Models	1
IP	Input Perturbation	2
SS	Scheduled Sampling	2
VLMs	Vision-Language Models	3
MDSS	Multistep Denosing Scheduled Sampling	3
IRPE	image reverse prompt engineering	3
ARPO	automatic reverse prompt optimization	3
DDIM	Denosing Diffusion Implicit Models	7
DAD	Data As Demonstrator	9
LLMs	Large Language Models	10
T2I	text-to-image	10
DDPMs	Denoising Diffusion Probabilistic Models	12
SDSS	Single-step denoising scheduled sampling	19
FID	Frechet Inception Distance	21
AIGC	Artificial Intelligence Generated Content	29
RPE	reverse prompt engineering	30

List of Figures

List of Tables

CHAPTER 1

Introduction

Diffusion Models (DMs) [22] are novel generative models that initially degrade data by progressively adding noise and then learn how to reverse this process for generation. Compared with likelihood-based models (e.g. Normalizing Flow model [30, 47] and Variational Auto-Encoder model [27]) which require stringent constraints on the model to guarantee that the normalizing constant for likelihood computation is tractable and implicit generative models (e.g., Generative Adversarial Networks [20]) which need adversarial training that is unstable and difficult to train, Diffusion Models are unrestricted and stable in training which have swiftly gained substantial attention and become the state-of-the-art approach in generative tasks [42].

Diffusion Models have achieved great success in various domains, including image generation [17, 45, 52, 56], video generation [24, 61], audio generation [43], and 3D generation [33], etc. Over the past three years, research on DMs has primarily focused on developing more powerful network architectures [17, 8], designing faster and more efficient sampling process [63, 31, 41, 58], improving likelihood estimation accuracy [44, 2, 28], and enhancing applications in different modalities [45, 43] and scenarios [13, 55].

Despite extensive research and significant success, Diffusion Models can still generate images that do not align with user's expectations. Taking image generation task as example, this unexpected outcomes can manifest in two ways: 1) generating images that are noisy and distorted, and 2) generating images that do not align with the user's design goal. The generation of unexpected noisy images might be due to the challenge of the exposure bias problem in Diffusion Models. The exposure bias problem is a prevalent issue in the domain of recurrent processes, such as autoregressive text generation, leading to suboptimal performance

[53]. This problem arises from the disparity between the training and inference processes. In the training process, the input to the model is based on the deterministic calculation of the forward process. In the contrast, the input to the model is derived from the output of previous steps during the inference process, which may include errors from model predictions. This discrepancy between training and inference leads to noise continuously accumulating and amplifying during the inference process, which affects the quality of the image generation.

Alleviating the exposure bias issue can significantly reduce unexpected noise in generated images, but it does not guarantee that images are generated according to the user's design. Diffusion models also face the challenge of designing prompts in application. Text-driven generation is a common application paradigm for DMs, such as text-to-image generation [54, 49, 56, 5, 8], text-to-video generation [24, 6, 61], and text-to-3D generation [33, 11]. Users guide DMs to generate desired outcomes through textual prompts. However, manually designing prompts heavily depends on the designer's experience and expertise, which raises the usage threshold for DMs. Survey results have found that among users without special training, less than 30% of the prompts they designed could generate results that met their expectations. The difficulty in designing prompts results in users being unable to describe their design objectives accurately, leading to images that do not align with their ideas.

Only a few studies have focused on these challenges and attempted to address them, but these methods still have shortcomings:

(1) **Alleviating exposure bias problem.** Only a few works have recently attempted to address the exposure bias problem in the training of Diffusion Models. The Input Perturbation (IP) method [46] introduces perturbations to the model input to simulate the prediction errors during inference; however, the Gaussian noise perturbation is predefined and does not accurately reflect the actual noise in model predictions. The Scheduled Sampling (SS) method [16] introduces model noise through a step of inference. Nevertheless, the ground truth of SS method also contain the model noise. Moreover, both of these methods only calculate prediction errors in one step, whereas Diffusion Models is a multi-steps generative models.

(2) **Image Reverse Prompt Engineering.** To the best of our knowledge, we are the first to explore the problem of image reverse prompt engineering (IRPE) in literature. However, there are some previous methods can be used to generate prompts from images. CLIP-Interrogator [72] is the only method based on a prior dataset. It first collects a prompts dataset and then use CLIP similarity between image and text to select the prompt. However, the prompts it generates are inflexible and restricted to the dataset. Another method involves using the capabilities of Vision-Language Models (VLMs) for image captioning. Nevertheless, outputs from VLMs are often lengthy and complex, which is not well-suited to guiding image generation as prompts.

In this thesis, we first conduct research on the exposure bias issue to reduce noise in diffusion models. Then, we introduce new tasks and methods to lower the design threshold for prompts, aiming to generate images that meet user expectations. The main contributions of this thesis can be summarized as follow:

(1) **Proposing a novel training strategy to alleviate the exposure bias problem.** We detailed analyze the discrepancy between training and inference in DMs and propose an effective Multistep Denosing Scheduled Sampling (MDSS) training strategy to alleviate the exposure bias. The MDSS method 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 the accumulated prediction error by using multiple iterations of mathematical formulation initialized from the one-step prediction error of the model. The experiments results demonstrate that our MDSS performs best in alleviating exposure bias for DMs.

(2) **Proposing task and method for image reverse prompt engineering.** We explore the problem of IRPE, which involves crating a prompt that can be used to generate the given reference image, thereby enabling normal users to generate new images using prompts decoded from beautiful images as reference. We introduce an automatic reverse prompt optimization (ARPO) method to solve the IRPE problem. The ARPO method begins with an initial reverse prompt from BLIP2 and optimizes

it iteratively. This process involves generating a recreated image from the current reverse prompt, comparing this image with the reference image to draft candidate prompts for improvements, and finally selecting the best candidate prompt to update the reverse prompt. The qualitative and quantitative experiments shows the efficacy of ARPO for IRPE.

The remainder of this thesis is organized into four chapters. Chapter 2 presents an overview of the relevant literature pertaining to this thesis, Chapter 3 introduces the research on alleviating the exposure bias problem and Chapter 4 provides a details description the research on image reverse prompt engineering. Finally, Chapter 5 concludes the thesis by summarizing the contributions of the research, and outlining the future directions.

CHAPTER 2

Literature Review

This chapter briefly reviews several topics that are related to this thesis, including the background of Diffusion Models, exposure bias, reverse prompt engineering, and automatic prompt optimization.

2.1 Preliminary Knowledge of Diffusion Models

Diffusion Models are the topic of this thesis.The preliminary knowledge of Diffusion Models will be first introduced in this section. Diffusion Models 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 [22, 44].

Given data distribution $q(\mathbf{x}_0)$ and the noise schedule $\beta_1, \beta_2, \ldots, \beta_T$, the forward process corrupts the data as a Markov chain:

$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t I\right) \tag{2.1}$$

$$q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right) = \prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) \tag{2.2}$$

When T is large enough, we can achieve $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. As mentioned in [22], we can sample to any time step directly using input $\mathbf{x}_0 \sim q(\mathbf{x}_0)$:

$$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, \left(1-\bar{\alpha}_t\right) I\right) \tag{2.3}$$

Algorithm 1 DMs Standard Training Process

1: **repeat**
2: $x_0 \sim q(x_0)$;
3: $t \sim \mathbb{U}(\{1, \cdots T\})$;
4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
5: Take gradient descent step on
$\nabla_\theta \left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}, t \right) \right\|^2$;
6: **until** converged

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

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \epsilon \tag{2.4}$$

Using Bayes theorem, we can obtain the posterior reverse process distribution $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)$:

$$q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\mu}\left(x_t, x_0\right), \tilde{\beta}_t \mathbf{I}\right) \tag{2.5}$$

$$\tilde{\mu}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) := \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1-\bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t} \mathbf{x}_t \tag{2.6}$$

$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t \tag{2.7}$$

In the inference process, $\mathbf{x}_0$ is not available and $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ depends on the entire data distribution. Consequently, the reverse process is defined as a parameterized process:

$$p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \mu_\theta\left(\mathbf{x}_t, t\right), \Sigma_\theta\left(\mathbf{x}_t, t\right)\right) \tag{2.8}$$

$$p_\theta\left(\mathbf{x}_{0:T}\right) = p\left(\mathbf{x}_T\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) \tag{2.9}$$

Instead of learning the mean of reverse process, Ho et al. find that predicting the noise ϵ is a better option. Empirically, they propose simplifying the loss function as follows:

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0, t, \epsilon \sim \mathcal{N}(0, \mathbf{I})}\left[\left\|\epsilon - \epsilon_\theta\left(\mathbf{x}_t, t\right)\right\|^2\right] \tag{2.10}$$

The training and inference algorithms are described in Alg. 1 and Alg. 2, respectively.

Algorithm 2 DMs Standard Inference Process

$X_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
for $t = T, \cdots, 1$ **do**
  $z \sim \mathcal{N}(\mathbf{0}, I)$ if $t > 1$, else $z = 0$;
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta \left(\mathbf{x}_t, t \right) \right) + \sigma_t z$;
end for
return $\mathbf{x}_0$

Many methods demonstrate improvement in the outcomes of fewer steps inference, with Denosing Diffusion Implicit Models (DDIM) and DPM-Solver being the most prevalent. DDIMs substitute the Markov forward process utilized in DDPMs with a non-Markovian one:

$$q \left(\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \mathbf{x}_0 \right) = \prod_{t=1}^{T} q \left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0 \right) \tag{2.11}$$

$$q_\sigma \left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0 \right) = \mathcal{N} \left(\mathbf{x}_{t-1} \mid \tilde{\mu}_t \left(\mathbf{x}_t, \mathbf{x}_0 \right), \sigma_t^2 \mathbf{I} \right) \tag{2.12}$$

$$\tilde{\mu}_t \left(\mathbf{x}_t, \mathbf{x}_0 \right) := \sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}} t \mathbf{x}_0}{\sqrt{1 - \bar{\alpha}_t}} \tag{2.13}$$

Both DDPM and DDIM are special cases of the above equation, where DDPM corresponds to setting $\sigma_t^2 = \frac{\hat{\beta}_{t-1}}{\hat{\beta}_t} \beta_t$ and DDIM corresponds to $\sigma_t^2 = 0$. The generative process of DDIM changes such that the model first predicts the normal sample, and then, uses the normal sample to estimate the next step in the chain. The change leads to a faster sampling procedure with a small impact on the quality of the generated samples:

$$\begin{aligned} \mathbf{x}_{t-1} = & \sqrt{\bar{\alpha}_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta^{(t)} \left(\mathbf{x}_t \right)}{\sqrt{\bar{\alpha}_t}} \right) \\ & + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \cdot \epsilon_\theta^{(t)} \left(\mathbf{x}_t \right) + \sigma_t \epsilon_t \end{aligned} \tag{2.14}$$

The algorithm is described in Alg. 3.

DPM-Solver reformulates the diffusion ODEs process into a format that can be solved by an exponential integrator. To approximate the integral term, the ODE solver utilizes a Taylor

Algorithm 3 DDIM Inference Process

$X_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
for $t = T, \cdots, 1$ **do**
  $z \sim \mathcal{N}(\mathbf{0}, I)$ if $t > 1$, else $z = 0$;
  $\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta^{(t)}(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}} \cdot \epsilon_\theta\left(\mathbf{x}_t, t\right)$;
end for
return $\mathbf{x}_0$

Algorithm 4 DPM-Solver-1

$\tilde{\boldsymbol{x}}_{t_0} \leftarrow \boldsymbol{x}_T$
for $i \leftarrow 1$ to M **do**
  $\tilde{\boldsymbol{x}}_{t_i} = \frac{\sqrt{\bar{\alpha}_{t_i}}}{\sqrt{\bar{\alpha}_{t_{i-1}}}} \tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}\left(e^{h_i} - 1\right) \boldsymbol{\epsilon}_\theta\left(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}\right)$
end for
return $\tilde{\boldsymbol{x}}_{t_M}$

Algorithm 5 DPM-Solver-2

$\tilde{\boldsymbol{x}}_{t_0} \leftarrow \boldsymbol{x}_T$
for $i \leftarrow 1$ to M **do**
  $s_i \leftarrow t_\lambda\left(\frac{\lambda_{t_{i-1}} + \lambda_{t_i}}{2}\right)$
  $\boldsymbol{u}_i \leftarrow \frac{\sqrt{\bar{\alpha}_{s_i}}}{\sqrt{\bar{\alpha}_{t_{i-1}}}} \tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_i}\left(e^{\frac{h_i}{2}} - 1\right) \boldsymbol{\epsilon}_\theta\left(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}\right)$
  $\tilde{\boldsymbol{x}}_{t_i} \leftarrow \frac{\sqrt{\bar{\alpha}_{t_i}}}{\sqrt{\bar{\alpha}_{t_i - 1}}} \tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}\left(e^{h_i} - 1\right) \boldsymbol{\epsilon}_\theta\left(\boldsymbol{u}_i, s_i\right)$
end for
return $\tilde{\boldsymbol{x}}_{t_M}$

expansion:

$$\begin{aligned} \boldsymbol{x}_{t_{i-1} \to t_i} = \frac{\sqrt{\bar{\alpha}_{t_i}}}{\sqrt{\bar{\alpha}_{t_{i-1}}}} \tilde{\boldsymbol{x}}_{t_{i-1}} - \sqrt{\bar{\alpha}_{t_i}} \sum_{n=0}^{k-1} \hat{\boldsymbol{\epsilon}}_\theta^{(n)}\left(\hat{\boldsymbol{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}}\right) \\ \int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda} \frac{\left(\lambda - \lambda_{t_{i-1}}\right)^n}{n!} \mathrm{d}\lambda + \mathcal{O}\left(h_i^{k+1}\right) \end{aligned} \tag{2.15}$$

where $\lambda_t = log(\sqrt{\bar{\alpha}_t}/\sqrt{1-\bar{\alpha}})$ (one half of the log-SNR) and $h_i = \lambda_{t_i} - \lambda_{t_{i-1}}$. k is the order of Taylor expansion. The algorithms of DPM-Solver in 1,2,3 order are shown in Alg. 4, Alg. 5, and Alg. 6, respectively.

Algorithm 6 DPM-Solver-3

$\tilde{\boldsymbol{x}}_{t_0} \leftarrow \boldsymbol{x}_T$, $r_1 \leftarrow \frac{1}{3}$, $r_1 \leftarrow \frac{2}{3}$
for $i \leftarrow 1$ to M **do**
  $s_{2i-1} \leftarrow t_\lambda\left(\lambda_{t_{i-1}} + r_1 h_i\right)$
  $s_{2i} \leftarrow t_\lambda\left(\lambda_{t_{i-1}} + r_2 h_i\right)$
  $\boldsymbol{u}_{2i-1} \leftarrow \frac{\sqrt{\bar{\alpha}_{s_{2i-1}}}}{\sqrt{\bar{\alpha}_{t_{i-1}}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_{2i-1}}\left(e^{r_1 h_i} - 1\right)\boldsymbol{\epsilon}_\theta\left(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}\right)$
  $\boldsymbol{D}_{2i-1} \leftarrow \boldsymbol{\epsilon}_\theta\left(\boldsymbol{u}_{2i-1}, s_{2i-1}\right) - \boldsymbol{\epsilon}_\theta\left(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}\right)$
  $\boldsymbol{u}_{2i} \leftarrow \frac{\sqrt{\bar{\alpha}_{s_{2i}}}}{\sqrt{\bar{\alpha}_{t_{i-1}}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_{2i}}\left(e^{r_2 h_i} - 1\right)\boldsymbol{\epsilon}_\theta\left(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}\right) - \frac{\sigma_{s_{2i}} r_2}{r_1}\left(\frac{e^{r_2 h_i} - 1}{r_2 h_i} - 1\right)\boldsymbol{D}_{2i-1}$
  $\boldsymbol{D}_{2i} \leftarrow \boldsymbol{\epsilon}_\theta\left(\boldsymbol{u}_{2i}, s_{2i}\right) - \boldsymbol{\epsilon}_\theta\left(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}\right)$
  $\tilde{\boldsymbol{x}}_{t_i} \leftarrow \frac{\sqrt{\bar{\alpha}_{t_i}}}{\sqrt{\bar{\alpha}_{t_{i-1}}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}\left(e^{h_i} - 1\right)\boldsymbol{\epsilon}_\theta\left(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}\right) - \frac{\sigma_{t_i}}{r_2}\left(\frac{e^{h_i} - 1}{h} - 1\right)\boldsymbol{D}_{2i}$
end for
return $\tilde{\boldsymbol{x}}_{t_M}$

2.2 Exposure Bias

To better understand the motivation about alleviating exposure bias for DMs, the background of exposure bias is introduced in this section. Exposure bias is a prevalent issue in recurrent processes, arising due to the teacher-forcing training method [4, 53, 70, 59]. Throughout the entire training process, the model is not exposed to its own predictions but given ground truth. However, during the sampling phase, the word predicted at a previous moment is used to predict the subsequent word. This discrepancy between training and sampling leads to inaccurate sampling. The Data As Demonstrator (DAD) [64] approach tackles this issue by feeding both ground truth words and predicted words during the training process. Scheduled sampling [4], on the other hand, replaces the teacher-forcing training method with a biased sampling approach that emulates the sampling process based on its own predictions.

A few works have recently attempted to address the exposure bias problem for DMs. The Input Perturbation [46] method introduces perturbation in the samples to simulate the inference

prediction errors. However, the perturbation is predefined and not equal to the true distribution of noise obtained from the model. [16] directly employ the scheduled sampling method [4] proposed for autoregressive text generation task, but cannot fully depict the discrepancy.

2.3 Reverse Prompt Engineering

Previous reverse prompt engineering methods can be divided into two categories: 1) selecting prompts from a prior dataset and 2) generating prompts through VLMs. Specifically, CLIP-Interrogator [72] is the only method based on a prior dataset. It first collects a dataset of over 100k image descriptions in tag format, then uses cosine similarity between image and text CLIP embeddings to filter this dataset and compose prompts. The primary limitation of CLIP-Interrogator is that the prompts it generates are restricted to the dataset. On the other hand, Vision-Language Models exhibit capabilities for image captioning. A notable instance is BLIP [35], which is a multimodal mixture of encoder-decoder pre-training on vision-language tasks. With the advancement of recent VLMs such as LLaVA [40, 38, 39], ShareGPT-4V [9], VILA [37], and GPT-4V [1] tokenize visual inputs and integrate them into Large Language Models (LLMs) alongside text tokens, demonstrating powerful capabilities in image and text reasoning. While VLMs can provide more comprehensive image descriptions, their outputs are often lengthy and complex, which is not well-suited to guiding image generation in text-to-image (T2I) models as prompts.

2.4 Automatic Prompt Optimization

As our ARPO method in Chapter 4 is inspired by automatic prompt optimization, the related works are introduced in this section. Designing high-quality prompts manually is time-consuming and labor-intensive, relying heavily on the designer's experience and knowledge. Consequently, there has been a growing need for automatically generating and optimizing prompts. Methods for automatic prompt optimization can be categorized into two types: real-gradient methods and imitated-gradient methods. Gradient-based methods generate optimized prompts by calculating numerical gradients for search or fine-tuning. For instance,

AutoPrompt [60] utilizes gradient-guided search to identify trigger tokens and integrate them with the original task input to create prompts. Imitated-gradient-based methods replace numerical gradients with other formats. APE [71] proposes selecting the optimal prompt by employing LLM in the roles of Inference, Scoring, and Resampling. Meanwhile APO [50] replaces numerical gradients by employing natural language gradients.

CHAPTER 3

Alleviating Exposure Bias for Diffusion Models

This chapter investigates the exposure bias issue for Diffusion Models (DMs), also known as Denoising Diffusion Probabilistic Models (DDPMs). The exposure bias issue is 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 chapter, 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.

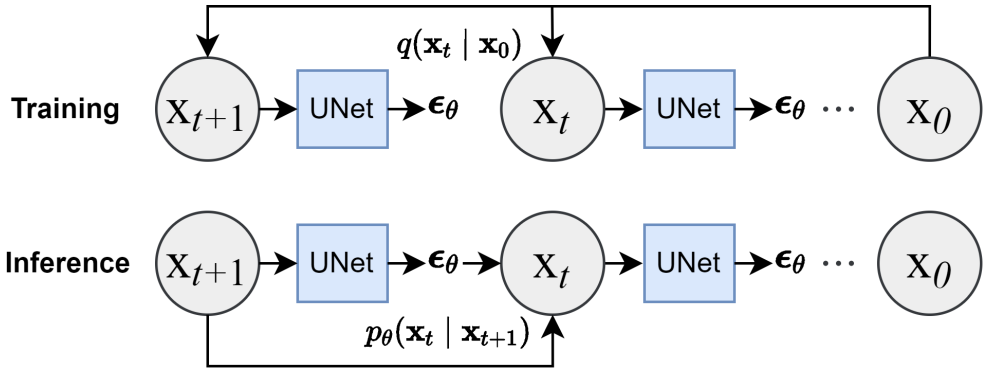


FIGURE 3.1. **The discrepancy between training and inference process.** In the training process, the input at time step t is derived from the forward diffusion process of $\mathbf{x}_0$. Contrarily, in the inference process, the input at time step t is obtained from the output of the previous step.

3.1 Introduction

Denoising Diffusion Probabilistic Models (DDPMs) [62, 22] are generative models, which first destruct data by progressively adding noise and then learn the reverse process for sample generation [68]. 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 [17], text-to-image generation [45, 52, 56], text-to-video generation [61], and audio generation [43]. Recent researches on DDPMs have been primarily focused on augmenting the classical method [22] in three key areas: efficient sampling [63, 31, 41, 58], improved likelihood estimation [44, 2, 28], and handling multi-modal tasks [45, 43]. 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 [53], arising from the disparity between the training and inference processes [4, 70, 59]. As shown in Fig. 3.1, in the training process of DDPMs, a real sample $\mathbf{x}_0$ 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 $\mathbf{x}_0$, noise schedule α_t, and a random standard Gaussian noise ϵ: $q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t) I)$. In contrast, in the inference process, the input to the model comes from the output of the previous steps $p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = \mathcal{N}(\mathbf{x}_t; \mu_\theta(\mathbf{x}_{t+1}, t+1), \Sigma_\theta(\mathbf{x}_{t+1}, t+1))$. 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 [16, 46, 36].

A few works [46, 16] 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 [46] introduces perturbation in the ground truth samples to simulate the inference prediction errors. Nevertheless, IP's perturbation, *e.g.,* Gaussian noise, is predefined and not equal to the true distribution of noise obtained from the model. Deng et al. directly employ the Scheduled Sampling method [4], originally proposed for autoregressive text generation tasks. However, the scheduled sampling method cannot fully depict the discrepancy since 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 [46, 36].

In light of the above issues, in this chapter, 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, requiring 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 [32], ImageNet 64×64 [15], and LSUN 64×64 [69] datasets. Compared to previous methods: IP and SS, our MDSS exhibits better generation quality improvements in DDPM, DDIM, and DPM-Solver.

Our contributions are summarised 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.

3.2 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 $\mathbf{x_0}$, noise schedule, time step t, and Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are deterministic, $\mathbf{x}_t$ is obtained by Eq. 2.4 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.

3.2.1 Scheduled Sampling with Denoising

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:

$$\hat{\mathbf{x}}_t = \mathbf{x}_t + \xi \tag{3.1}$$

where $\hat{\mathbf{x}}_t$ is the input of the current step with noise, $\mathbf{x}_t$ is the ground-truth input without noises, and ξ 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:

$$\begin{aligned} \hat{\mathbf{x}}_{t-1} &= \frac{1}{\sqrt{\alpha_t}}\left(\hat{\mathbf{x}}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\left(\hat{\mathbf{x}}_\mathbf{t}, t\right)\right) \\ &= \frac{1}{\sqrt{\alpha_t}}\left(\underbrace{\mathbf{x}_t + \xi}_{\text{Input Signal}} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\underbrace{\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_\mathbf{t} + \xi, t\right)}_{\text{Model Prediction}}\right) \end{aligned} \tag{3.2}$$

where we ignore the variance item for simplicity since it is given directly and devoid of model prediction noise in most of works [22].

It can be observed from Eq. 3.2 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 $\mathbf{x}_t$, which contains no noise, to obtain the training objective. $\mathbf{x}_t$ can be derived by sampling from the posterior distribution $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)$. By replacing $\mathbf{x}_0$ using ground truth ϵ_{t+1} in Eq. 2.6, we can obtain:

$$\begin{aligned}
\mathbf{x}_t &= \tilde{\mu}_{t+1}\left(\mathbf{x}_{t+1}, \mathbf{x}_0\right) \\
&= \frac{\sqrt{\alpha_{t+1}}\left(1-\bar{\alpha}_t\right)}{1-\bar{\alpha}_{t+1}} \mathbf{x}_{t+1} + \frac{\sqrt{\bar{\alpha}_t}\beta_{t+1}}{1-\bar{\alpha}_{t+1}} \mathbf{x}_0 \\
&= \frac{1}{\sqrt{\alpha_{t+1}}}\left(x_{t+1} - \frac{\beta_{t+1}}{\sqrt{\left(1-\bar{\alpha}_{t+1}\right)}} \boldsymbol{\epsilon}_{t+1}\right)
\end{aligned} \tag{3.3}$$

The meaning of posterior distribution is sampling to next step, when the model prediction $\boldsymbol{\epsilon}_\theta$ equals the ground truth $\boldsymbol{\epsilon}$. Hence, $\mathbf{x}_t$ derived from the posterior distribution represents the noise-free sampling result. Utilizing the variant of Eq. 2.4, we obtain the noise-free training object with $\mathbf{x}_t$ and $\mathbf{x}_0$:

$$\epsilon_t = \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}} \tag{3.4}$$

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:

$$\xi_t = \hat{\mathbf{x}}_t - \mathbf{x}_t \tag{3.5}$$

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:

$$\left\|\epsilon_t - \epsilon_\theta\left(\hat{\mathbf{x}}_t, t\right)\right\|^2 + \left\|\xi_t - \xi_\theta\left(\hat{\mathbf{x}}_t, t\right)\right\|^2 \tag{3.6}$$

More details of scheduled sampling with denoising can be found in Appendix A1.

3.2.2 Prediction Errors with Multi-step Accumulation

Previous methods [46, 16] 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:

$$\mathbf{x}_{t+k+1} = \sqrt{\bar{\alpha}_{t+k+1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t+k+1}}\epsilon \tag{3.7}$$

$$\mathbf{x}_{t+k} = \frac{1}{\sqrt{\alpha_{t+k+1}}}\left(\mathbf{x}_{t+k+1} - \frac{1-\alpha_{t+k+1}}{\sqrt{1-\bar{\alpha}_{t+k+1}}}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_{t+k+1}, t+k+1\right)\right) \tag{3.8}$$

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:

$$\mathbf{x}_t = \gamma_t \mathbf{x}_{t+k} + \omega_t \mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t}\beta_{t+1}}{1-\bar{\alpha}_{t+1}}\mathbf{x}_0 \tag{3.9}$$

$$\gamma_t = \prod_{i=t}^{t+k-1} \frac{\sqrt{\alpha_{i+1}}(1-\bar{\alpha}_i)}{1-\bar{\alpha}_{i+1}} \tag{3.10}$$

$$\omega_t = \sum_{j=t}^{t+k-2} \frac{\prod_{n=t}^{j}\left[\sqrt{\alpha_{n+1}}(1-\bar{\alpha}_n)\right]\sqrt{\bar{\alpha}_{j+1}}\beta_{j+2}}{\prod_{m=t+1}^{j+2}(1-\bar{\alpha}_m)} \tag{3.11}$$

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

Algorithm 7 Multi-step Denoising Scheduled Sampling

repeat
  $x_0 \sim q(x_0)$;
  $t \sim \mathbb{U}(\{1, \cdots T-k-1\})$;
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $\mathbf{x}_{t+k+1} = \sqrt{\bar{\alpha}_{t+k+1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t+k+1}}\boldsymbol{\epsilon}$;
  $\hat{\mathbf{x}}_{t+k} = \frac{1}{\sqrt{\alpha_{t+k+1}}}\left(\mathbf{x_{t+k+1}} - \frac{1-\alpha_{t+k+1}}{\sqrt{1-\bar{\alpha}_{t+k+1}}}\epsilon_\theta\left(\mathbf{x}_{t+k+1}, t+k+1\right)\right.$
  $\mathbf{x}_{t+k} = \frac{1}{\sqrt{\alpha_{t+k+1}}}\left(\mathbf{x_{t+k+1}} - \frac{1-\alpha_{t+k+1}}{\sqrt{1-\bar{\alpha}_{t+k+1}}}\boldsymbol{\epsilon}\right)$;
  if $k > 0$ **then**
    $\hat{\mathbf{x}}_t = \gamma_t\hat{\mathbf{x}}_{t+k} + \omega_t\mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t}\beta_{t+1}}{1-\bar{\alpha}_{t+1}}\mathbf{x}_0$
    $\mathbf{x}_t = \gamma_t\mathbf{x}_{t+k} + \omega_t\mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t}\beta_{t+1}}{1-\bar{\alpha}_{t+1}}\mathbf{x}_0$
  end if
  $\boldsymbol{\xi}_t = \hat{\mathbf{x}}_t - \mathbf{x}_t$;
  $\epsilon_t = \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}$;
  Take gradient descent step on
  $\nabla_\theta(\|\epsilon_t - \epsilon_\theta\left(\hat{\mathbf{x}}_t, t\right)\|^2 + \|\xi_t - \xi_\theta\left(\hat{\mathbf{x}}_t, t\right)\|^2)$
until converged

with noise and the ground truth expends, mimicking the noise accumulation observed in the sampling process.

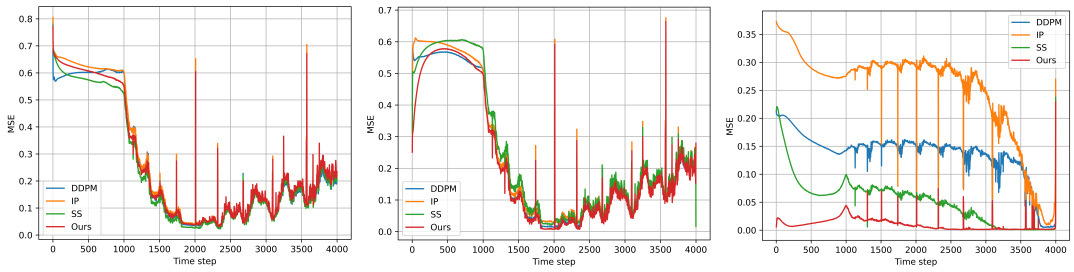
3.2.3 Algorithm

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. 7. 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. k represents the number of accumulation using mathmatical 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.

TABLE 3.1. **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.

	Input	Denoise model prediction	Denoise input signal
IP	$\hat{\mathbf{x}}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\left(\boldsymbol{\epsilon} + \gamma_t \boldsymbol{\varepsilon}\right)$ (✕)	Given ϵ (✓)	✕
SS	$\hat{\mathbf{x}}_t = \frac{1}{\sqrt{\alpha_{t+1}}}\left(\mathbf{x}_{t+1} - \frac{1-\alpha_{t+1}}{\sqrt{1-\bar{\alpha}_{t+1}}}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_{t+1}, t\right)\right)$ (✓)	$\epsilon = \frac{\hat{\mathbf{x}}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}$ (✕)	✕
MDSS	$\hat{\mathbf{x}}_t = \gamma\hat{\mathbf{x}}_{t+k} + \omega\mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t}\beta_{t+1}}{1-\bar{\alpha}_{t+1}}\mathbf{x}_0$ (✓)	$\epsilon = \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}$ (✓)	$\xi = \hat{\mathbf{x}}_t - \mathbf{x}_t$



(a) Input without errors (b) Input with one-step errors (c) Input with previous accumulated errors

FIGURE 3.2. **The Mean Squared Errors** between the model prediction and the ground truth for different methods.

3.2.4 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. 3.1 presents the comparison. For the training input, IP uses Gaussian noise to emulate model prediction noise, which may be different from the actual noise in the inference process. For the model prediction, SS ignores the influence of the input signal, as shown in Eq. 3.2, and only such a model prediction cannot fully address the exposure bias and may introduce new noises. Only MDSS considers the influence of input signal and prediction error accumulation.

We conduct three experiments with well-trained models to demonstrate the efficacy of our method further. In our experiments, we employ various inputs and measure the Mean Squared Error (MSE) between the model prediction and ground truth at every step. Firstly, the model's input is calculated by ground truth and does not include errors. Fig. 3.2(a) shows that all

methods perform nearly the same when the model inputs are free of noise. Secondly, the input of the model comes from one step of sampling of the previous step through the model. Fig. 3.2(b) reveals that when the input incorporates one step of model prediction noise, MDSS yields the most accurate results. Last, the input of the model comes from the sample of previous steps, which contains the accumulation errors. Fig. 3.2(c) shows that the error is prominent in the initial stages, likely due to the data distribution being close to standard. As the sampling process progresses, the model error diminishes rapidly. After hundreds of steps, the errors of prediction increase due to the exposure bias problem. However, using MDSS, the model prediction error did not show a trend of increasing during the sampling process, which reflects that we have well-alleviated exposure bias.

3.3 Experiment

3.3.1 Experimental Setup

We evaluate our method across unconditional image generation tasks on three datasets: CIFAR-10 [32], ImageNet 64×64 [15], and LSUN tower 64×64 [69]. For the CIFAR-10 and ImageNet 64×64 datasets, we fine-tune on the well-trained iDDPM [44] models, and for LSUN tower 64×64, we keep training on the ADM [23] model. We employ the Frechet Inception Distance (FID) [21] to evaluate the quality of the generated images. We also conducted experiments on Stable Diffusion V1.5 [54] to validate the performance of our methods on text-to-image models. 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, and qualitative comparison can be found in Appendix A2.

3.3.2 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. 3.2. 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. 3.3. 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 a predefined noise, such as the Gaussian Noise of IP, may contain a gap when compared with prediction errors of the inference process. MDSS and SDSS perform better than SS, partly because MDSS and SDSS model more accurately the prediction errors. 2) MDSS achieves better sampling results than SDSS partly because the non-markov process of DDIM might make it more susceptible to error accumulation.

We conducted experiments using DPM-Solver-1, 2, and 3 for the DPM-Solver sampling method. The CIFAR-10 results are shown in Table. 3.4, while results for ImageNet and LSUN are available in Appendix A4. We can draw the following observations: 1) DPM-Solver-1 is fundamentally similar to DDIM; therefore, our method can achieve superior sampling results compared to other approaches. 2) In DPM-Solver-2 and 3, while our approach outperforms IP and SS, there is minimal or no improvement compared to the DDPM baseline. One reason could be that DDPM and DDIM require a single model prediction during sampling, whereas DPM Solver undergoes two or three model predictions. The modeling of prediction errors should be adjusted based on DPM Solver inference, which is one of our future works.

To validate that our MDSS method can be applied to text-to-image models, we conducted experiments on Stable Diffusion V1.5. We employed MDSS to fine-tune Stable Diffusion V1.5 on a subset of 50k images from ImageNet 256×256 with 7k iterations. The FID score

TABLE 3.2. **DDPM results** on CIFAR-10, ImageNet, and LSUN tower with varying inference steps.

Dataset	Steps	DDPM	IP	SS	SDSS	MDSS
CIFAR-10	30	7.81	6.40	7.95	7.92	**5.54**
	100	3.72	**3.25**	3.62	3.47	3.49
	250	3.23	**3.02**	3.17	3.08	3.62
ImageNet	30	34.35	34.52	32.60	32.32	**31.76**
	100	25.32	24.88	23.86	**23.22**	24.28
	250	24.88	24.24	22.82	**22.48**	23.69
LSUN	30	5.98	**5.52**	5.76	5.72	5.57
	100	2.32	2.27	2.25	**2.24**	2.26
	250	2.22	1.95	1.93	**1.90**	1.92

TABLE 3.3. **DDIM results** on CIFAR-10, ImageNet, and LSUN tower with varying inference steps.

Dataset	Steps	DDPM	IP	SS	SDSS	MDSS
CIFAR-10	30	7.47	5.35	7.02	6.92	**5.25**
	100	4.99	6.95	4.78	4.53	**3.86**
	250	4.48	8.60	4.11	3.94	**3.92**
ImageNet	30	28.11	43.74	27.82	27.47	**27.18**
	100	25.33	49.73	24.81	**24.75**	24.96
	250	24.29	55.22	23.69	**23.44**	23.86
LSUN	30	4.26	37.38	4.01	3.94	**3.84**
	100	2.96	27.20	2.79	**2.61**	2.67
	250	4.48	32.92	4.38	4.56	**4.13**

improved, decreasing from 10.29 to 9.58. The experiment result illustrates that our MDSS method imposes no constraints on the model details, and it can be used to various diffusion models.

3.3.3 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 denosing. We first assess the impact of denoise 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. 3.5. We can conclude that using an additional

TABLE 3.4. **DPM-Solver-1,2,3 results** on CIFAR-10 with varying inference processes.

Methods	Steps	DDPM	IP	SS	SDSS	MDSS
DPM-Solver-1	10	26.08	41.96	25.39	24.96	**12.11**
	20	11.46	20.79	11.17	11.04	**5.41**
	50	6.03	14.43	5.99	5.89	**4.84**
DPM-Solver-2	10	**10.43**	42.03	13.04	11.98	12.149
	20	**3.55**	7.43	4.65	4.48	5.319
	50	**3.28**	11.39	4.04	4.02	4.845
DPM-Solver-3	10	**5.99**	33.22	6.43	6.34	7.80
	20	4.07	13.32	4.23	**4.01**	5.51
	50	4.01	12.36	4.00	**3.94**	5.33

TABLE 3.5. **Comparison of denoising input signal** on DDPM and DDIM.

	DDPM SDSS w/o	DDPM SDSS	DDIM SDSS w/o	DDIM SDSS
30 steps	7.93	**7.92**	6.93	**6.92**
100 steps	3.55	**3.47**	4.60	**4.53**
250 steps	3.09	**3.08**	4.05	**3.94**

model channel to remove noise from the input signal can further mitigate exposure bias and enhance generated results. Nevertheless, the improvement is slight compared with the model prediction denoising effect. This might be because exposure bias predominantly stems from the noise introduced by model prediction. The input containing noise can lead to large deviations in model predictions. However, the noise itself has little effect on the sampled results directly.

The effect of multi-step training. We discuss the effection 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. 3.7 and Table. 3.6. 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

TABLE 3.6. **Comparison of multi-step training** using different steps on DDPM.

	single-step	4 steps	10 steps	20 steps	50 steps
30 steps	7.92	**5.54**	5.92	6.70	8.31
100 steps	3.47	**3.49**	4.20	3.64	4.12
250 steps	**3.08**	3.62	3.75	3.58	3.32

TABLE 3.7. **Comparison of multi-step training** using different steps on DDIM.

	single-step	4 steps	10 steps	20 steps	50 steps
30 steps	6.92	5.25	**4.45**	4.52	6.98
100 steps	4.53	**3.86**	4.04	4.05	4.61
250 steps	3.94	3.92	3.96	**3.79**	4.09

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. 3.3. 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. This demonstrates that our method can achieve convergence results with fewer training iterations. Compared to training from scratch, which requires around 200,000 iterations, our method reduces the training time by approximately a factor of 10.

3.3.4 Discussion of Modifying MSDD Based on DDIM

Many methods modify the inference process to achieve improved results when utilizing fewer sampling steps. For example, while DDIM and DDPM undergo identical training process, their inference methods are a little distinct. Analytically, by utilizing DDIM in Eq. 2.14 instead of DDPM in scheduled sampling, we can achieve a noise distribution that more closely

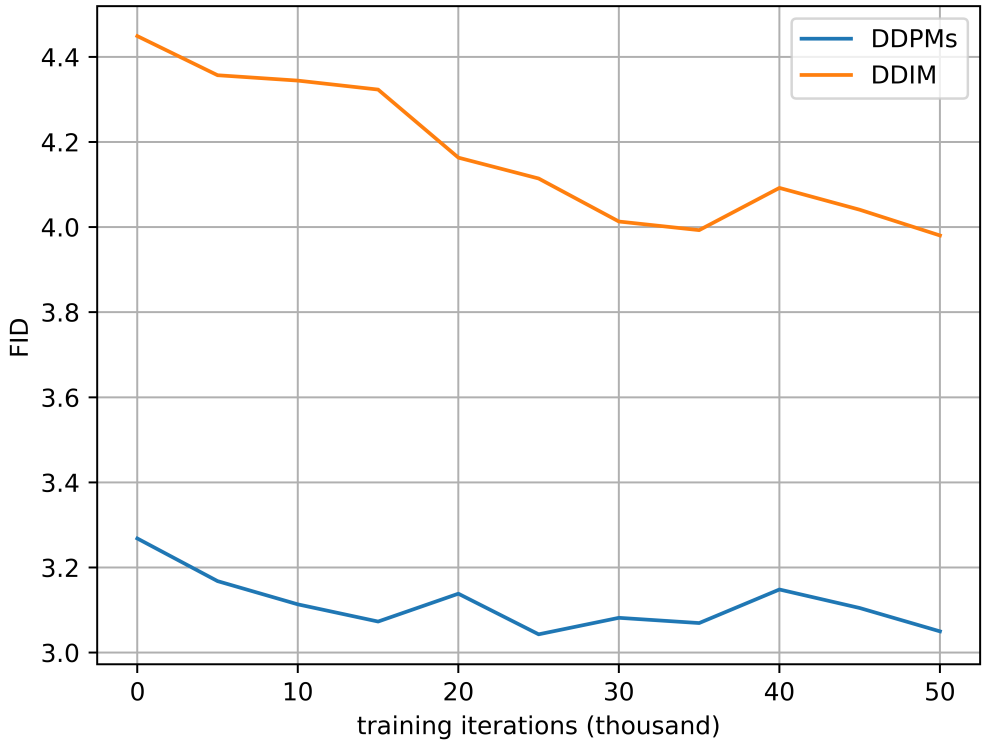


FIGURE 3.3. FID scores with respect to **the number of training iterations**. Each FID result is computed with 250 inference steps.

TABLE 3.8. Comparison of SDSS/MDSS **with/without modification** based on DDIM sampling.

	SDSS	DDIM SDSS	MDSS	DDIM MDSS
30 steps	6.92	**5.42**	5.25	**4.82**
100 steps	4.53	**4.21**	**3.86**	3.94
250 steps	**3.94**	4.51	**3.92**	4.63

mirrors the actual DDIM inference process. We compare the effect of SDSS and MDSS with the DDIM revised version, and the results are shown in Table. 3.8. 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 sampling with minimal steps, we can employ the DDIM scheduled 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.

3.3.5 Qualitative Comparison

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. Fig. 3.4 displays the generated samples using 30 steps DDIM on CIFAR10 dataset. We provide additional qualitative comparisons in Appendix. The images produced by MDSS showcase more comprehensive image details, further affirming the effectiveness of MDSS in reducing exposure bias and creating images of higher quality.

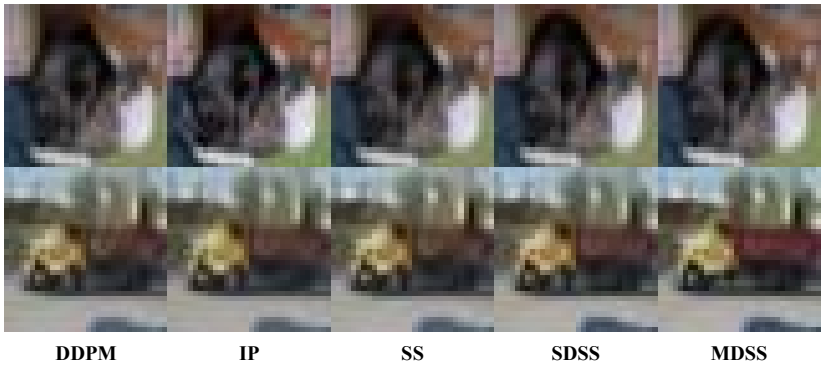


FIGURE 3.4. Qualitative comparison of DDPM, IP, SS, SDSS and MDSS on CIFAR10.

3.4 Summary

In this chapter, we propose a novel training method called multi-step denoising scheduled sampling (MDSS) to mitigate the exposure bias issue. Specifically, MDSS 1) comprehensively denoises the errors in model prediction and input signal and 2) efficiently models the prediction error accumulation 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.

CHAPTER 4

Reversing Prompts from Images

Generating images from textual prompts has become very popular, though it often requires significant effort in prompt engineering to create the desired images. In this chapter, we explore the problem of decoding textual prompts from existing reference images, which we call **image reverse prompt engineering**. This technique allows us to gain insights from reference images, uncover the creative processes of great artists, and facilitate novel image generation. We propose an **automatic reverse prompt engineering** method (ARPO) to solve this problem. Specifically, our method begins with an initial reverse prompt and optimizes it iteratively by: 1) generating a recreated image from the current reverse prompt; 2) comparing the recreated image with the reference image to draft candidate prompts for improvements; and 3) selecting the best candidate prompts to update the reverse prompt. This iterative process converges quickly, demonstrating effectiveness in real-world applications. We evaluate our method by comparing it with various baseline methods, including hand-crafted methods, image caption methods, data-driven methods, and commercial services. Both quantitative and qualitative results highlight the effectiveness of our method in generating high-quality reverse prompts.

4.1 Introduction

In recent years, the concept of Artificial Intelligence Generated Content (AIGC) has gained significant popularity, especially due to the great success of text-driven generation applications such as text-to-image [54, 49, 56, 5, 8], text-to-video [24, 6, 61], and text-to-3D [33, 11]. Meanwhile, there has been a significant increase in demand for production-ready text prompts,

known as prompt engineering, which require meticulous crafting and iteration to ensure precision, concreteness, and optimization for the intended use case. For example, many popular AIGC communities such as Lexica, CivitAI, and Stable Diffusion Online[1] have offered meticulously crafted text prompts to assist users in generating their desired images. The effectiveness of those production-ready text prompts has motivated us to further explore cracking the recipe inside text-driven generation, specifically in terms of text prompts. That is, for image generation, this will enable us normal people to generate new images using text prompts decoded from those beautiful reference images, whether they are natural photos taken by a camera, artworks crafted by humans, or images produced by unknown AIGC models.

(a) Reference Image (b) Hand-Crafted (c) GPT-4V (d) ARPO (ours)

FIGURE 4.1. **Illustration of images generated by Stable-Diffusion-V1.5 with different reverse prompts.** (a) Reference image for reverse prompt engineering. (b) Image generated with a hand-crafted reverse prompt. (c) Image generated with the reverse prompt of GPT-4V model. (d) Image generated with the reverse prompt by our ARPO model.

Crafting a prompt likely responsible for generating the desired output is typically referred to as reverse prompt engineering (RPE). A main difference between prompt engineering and reverse prompt engineering is whether the desired output is available before constructing prompts. Basically, many recent prompt generators use large language models (LLMs) to generate prompts based on task descriptions or examples [71, 26], which somewhat can also be seen as forms of reverse prompt engineering, though they rely on alternatives to the desired output rather than the specific output. In this chapter, we consider reverse prompt engineering for text-to-image generation, i.e., crafting a prompt that can be used to generate

[1] https://lexica.art; https://civitai.com; https://stablediffusionweb.com

Method	Prompt
Human Design	fantastical landscape, digital painting style, huge landscape, people in the middle of the road, green trees
GPT4-V	This image depicts a fantastical landscape painting with a strong sense of otherworldly atmosphere. In the foreground, a solitary figure is seen from behind, ... landscape dotted with classical ruins and statues. The figure appears contemplative or on a journey ... suggesting a cosmic event or a sky from another world. The colors are rich and warm, with a palette dominated by golds, oranges ... science fiction feel of the scene. This structure, along with the ruins ... contemplate the narrative and the world in which this scene takes place. The painting blends elements of classical art with speculative fiction ... bridge the past and the imagination.
IPRE (ours)	Content: imaginative landscape, exploration, figures placed centrally Style: surreal quality, exaggerated proportions, blend of realism and fantasy, dramatic sky, digital painting style with high level of detail and sense of depth

TABLE 4.1. **The results of various methods for reverse-engineering image prompts.** The images in Figure 4.1 are generated by the corresponding prompts.

the given reference image. We refer to this problem as image reverse prompt engineering (IRPE). This process not only allows us to unravel the complex relationships between prompts and the generated image, but also enables us to generate high-quality and new images similar to reference images. Similar to those aforementioned LLM-based prompt generators, a simple and straightforward IRPE model can be achieved by exploring large multimodal models, especially those popular vision-language models (VLMs) like BLIP series [35, 34, 14], VILA [37], and GPT-4V [1]. Figure 4.1 and Table 4.1 illustrate images generated using prompts directly constructed by the above mentioned VLMs, demonstrating a clear gap between reference and generated images.

Currently, crafting successful production-level prompts is usually a time-consuming iterative process involving both humans and AIGC models: humans refine prompts by comparing reference and generated images, while AIGC models generate new images with refined prompts. Additionally, some prompt generators mainly rely on collecting large-scale dataset for specific characteristics [72], and commercial reverse prompt engineering services/models also lack exposure of technical details[2]. To address these issues, we propose an automatic prompt optimization framework for image reverse prompt engineering, referred to as ARPO. Specifically, when given a reference image, we initially set the reverse prompt using VLMs like BLIP2 or a hand-crafted one. Subsequently, we iteratively optimize this prompt with three main steps: 1) **Image Generation**: a text-to-image model creates a new image from the current reverse prompt; 2) **Prompt Generation**: a LLM/VLM-empowered prompt generator

[2]https:www.phot.ai; https://animegenius.live3d.io/; https://imagetopromptai.com

generates candidate prompts by comparing generated and reference images; and 3) **Prompt Selection**: an optimal prompt updates the reverse prompt. Inspired by automatic prompts in language models [60, 71, 67, 50], particularly the imitated-gradient methods that use text instead of numerical gradients for guiding optimization in LLM [50], our ARPO employs similar approach using imitated-gradient generated by VLMs and LLMs. This iterative process utilized imitated-gradient guided search, continuing until the generated image aligns closely with the reference or the maximum steps are reached. Last but not least, the goal of image reverse prompt engineering is not just to recreate the reference image, but also generate novel images with desired elements like content and style. This novel image generation can be easily achieved by editing the obtained reverse prompt, as detailed in Section 4.2.3.

To evaluate our approach, we extensively compare images generated using our method's reverse prompts with those from several baseline methods and black-box commercial services. Additionally, we introduce a new dataset of 100 images, including 50 AI-generated [65] and 50 human-created [57, 25] images, for further quantitative analysis. Our evaluation employs four metrics—`CLIP-T`, `CLIP-I`, `ViT`, and `DINO`—to assess prompt and image fidelity in detail [51, 7, 19]. We also conduct a user study to evaluate image fidelity. The comprehensive experimental results demonstrate the effectiveness of our ARPO method. Furthermore, we perform ablation studies to explore the impact of individual design choices within our method. We hope this work can inspire further exploration of reverse prompt engineering in image as well as other domains such as 3D and video.

Our main contributions can be summarised as follows:

- We explore the problem of reverse prompt engineering for text-to-image generation, to the best of our knowledge, which has not been investigated in previous literature.
- We introduce an automatic reverse prompt optimization method, ARPO, designed to generate high-quality reverse prompts. These prompts can facilitate the creation of new images with content and/or style similar to the reference image.
- We conduct comprehensive experiments, illustrating the efficacy of our proposed method for image reverse prompt engineering.

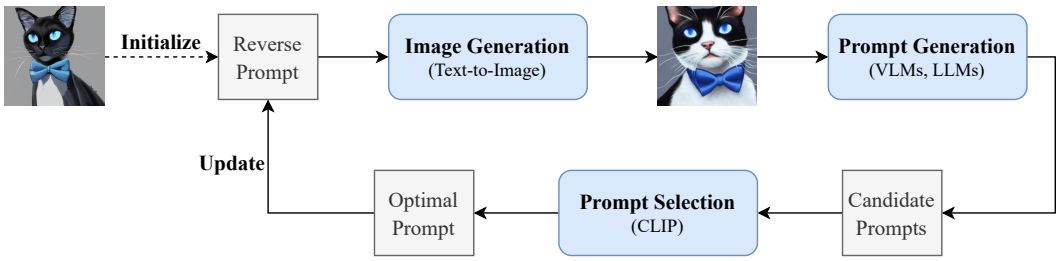


FIGURE 4.2. **The main ARPO framework** consists of three main components: image generation, prompt generation, and prompt selection.

4.2 Method

In this section, we first describe the problem of image reverse prompt engineering (IRPE). We then introduce the automatic reverse prompt optimization framework, and refer to it as ARPO. Lastly, we discuss the application of IRPE in novel image generation by editing the reverse prompt.

4.2.1 Image Reverse Prompt Engineering

For text-to-image generation, we start by creating a textual prompt P to describe the desired image. We then utilize a standard text-to-image generation model, denoted as f, to produce the resulting image $I_g = f(P)$. An important question arises: does the inverse function f^{-1} exist, allowing us to understand the image generation process, i.e., $P = f^{-1}(I_g)$? However, directly learning such invertible generative models is typically complex, as evidenced by prior research [18, 29, 10, 30, 3]. Moreover, exploring flow-based generative models or invertible networks falls beyond the scope of this chapter. Given that recent state-of-the-art text-driven generation models lack invertibility, we do not make assumptions about the underlying generative models. Instead, our aim is to offer a practical and accessible solution through prompt engineering. Specifically, we decode a reference image I_r to derive a reverse prompt P that best preserves crucial image information such as content and style. This process can be formulated as $P^* \in \texttt{argmin}_P \mathcal{L}(I_r, f(P))$, where $\mathcal{L}$ represents the criterion for assessing image discrepancy.

4.2.2 Automatic Reverse Prompt Optimization

The main ARPO framework, illustrated in Figure 4.2, iteratively optimizes an initialized reverse prompt through three key steps: image generation, prompt generation, and prompt selection. Below, we elaborate on these crucial steps employed by the ARPO framework.

Prompt Initialization Recently, creating a simple reverse prompt has become very easy for us using either image caption models, visual-language models, or hand-crafted approaches. We thus begin automatic prompt optimization with an initial reverse prompt to significantly reduce the required iterations. In this chapter, we utilize the image caption interface of the popular BLIP2 [34] model for prompt initialization. For instance, the initialized simple reverse prompt for the reference image in Figure 4.2 is "*a black and white cat with blue eyes wearing a bow tie*". That is, the initialized reverse prompt P^0 for a given reference image I_r can be obtained as follows:

$$P^0 = \texttt{BLIP2}(I_r). \tag{4.1}$$

Image Generation Since various text-to-image models often exhibit distinct instruction tuning preferences, we do not aim for a generalized reverse prompt process across all models. Instead, we optimize the reverse prompt based on feedback from each specific text-to-image model. Specifically, at the t-th iteration, we employ current reverse prompt P^t to generate a new image I_g^t as follows:

$$I_g^t = \texttt{TextToImage}(P^t), \tag{4.2}$$

where `TextToImage` indicates a given text-to-image model for image generation. Unless stated otherwise, we utilize the popular Stable-Diffusion-V1.5 [54].

Prompt Generation The motivation for image generation mentioned above is to identify the difference between the generated image I_g^t and the reference image I_r. This difference provides crucial information to propose candidate reverse prompts aimed at minimizing the difference between the next iteration's generated image I_g^{t+1} and the reference image. The main prompt generation frameworks are illustrated in Figure 4.3. Specifically, the vanilla prompt generation framework employs VLMs to compare the difference between reference and generated images. Subsequently, this difference prompt is utilized to generate

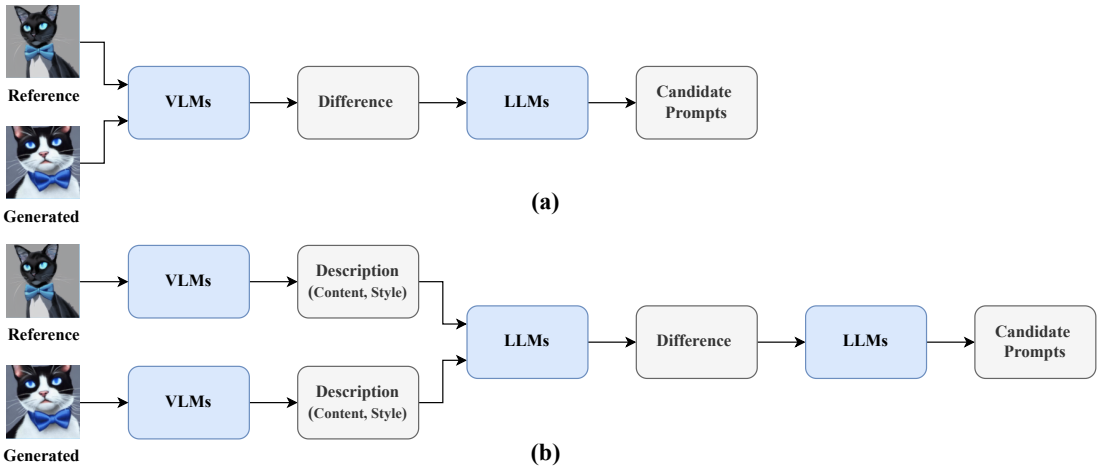


FIGURE 4.3. **The main prompt generation frameworks used in ARPO.** (a) The vanilla prompt generation framework. (b) The enhanced prompt generation framework.

candidate reverse prompts via LLMs. However, we observed a significant performance gap between GPT-4V and open-source VLMs like VILA. To address this, we enhance the prompt generation framework by first generating descriptions of image content and style, building on the chain-of-thought (CoT) prompting strategies [66]. Since GPT-4V performs well in directly comparing image differences, we use the vanilla prompt generation framework for GPT-4V and the enhanced framework for other open-source VLMs. This decision is mainly based on cost-effectiveness: using GPT-4V as the VLM, the vanilla framework costs less than one dollar per case, while the enhanced framework costs around four dollars per case, with minimal observed improvements.

We detail the vanilla framework in Figure 4.3(a) as follows. At the t-th iteration, we utilize the VLM model to compare the difference between the generated image I_g^t and the reference image I_r,

$$P_{\text{diff}}^t = \{P_{\text{diff}}^{t,0}, P_{\text{diff}}^{t,1}, \dots\} = \texttt{VLM}(I_g^t, I_r, \texttt{template}), \tag{4.3}$$

where `template` indicates the template or instruction used by the VLM model to compare image difference (see details and examples in Appendix B2.1). The descriptions comprehensively analyze differences between the two images from multiple perspectives, including content, style, color, and composition. An intuitive example of the difference description $P_{\text{diff}}^{t,1}$ is " *Image 1 depicts a cat with a more stylized and artistic rendering, showcasing a softer*

texture and a more whimsical feel with its exaggerated large blue eyes and a light blue bow tie that has a silky texture". Next, we employ a LLM model to generate a candidate prompt for minimizing each difference, i.e.,

$$P^t_{\text{cand}} = \{P^{t,0}_{\text{cand}}, P^{t,1}_{\text{cand}}, \dots\} = \texttt{LLM}(P^t_{\text{diff}}, P^t, \texttt{template}), \tag{4.4}$$

where `template` indicates the template or instruction used by the LLM model to generate candidate prompts (see details and examples in Appendix B2.1). These generate candidate prompts are actually suggestions for minimizing each of the above-mentioned differences. An example of generated candidate prompts P^t_{cand} can be "{ *stylized artistic rendering of a cat; exaggerated large blue eyes; light blue silky bow tie; smooth fur texture; cool tone background; serene mood; whimsical feel; illustrative and fantastical style; soft texture visual; monochromatic color scheme*}". For the enhanced framework, it primarily enhances image difference descriptions, while the overall framework remains similar to the vanilla version. See additional details about the enhanced framework in Appendix B2.2.

Prompt Selection Directly using all candidate prompts leads to saturation in the optimization process, similar to the motivation of controlling training dynamics with learning rate and batch size in deep learning. Motivated by [72], we use a similar greedy prompt selection algorithm, as shown in Algorithm 8, to identify useful candidate prompts for obtaining the optimal reverse prompt and updating the current one. The similarity between a reverse prompt and the reference image, i.e., `ClipSimilarity`, is calculated by the cosine similarity between text and image embeddings from the CLIP model [51]. At the t-th iteration, through the prompt selection process, we obtain a subset of candidate prompts $\Delta P \subseteq P^t_{\text{cand}}$ that improves current reverse prompt P_t. We thus update the reverse prompt for the next iteration as

$$P^{t+1} = P^t + \Delta P. \tag{4.5}$$

This iterative optimization process stops when $\Delta P = \varnothing$ or the maximum steps are reached.

Algorithm 8 The Greedy Prompt Selection Algorithm.

Current prompt P^t, Candidate prompts P^t_{cand}, Reference image I_r, Similarity metric `ClipSimilarity`.
$\Delta P = \varnothing$; $s_{\text{max}} = \texttt{ClipSimilarity}(I_r, P^t)$
while $P^t_{\text{cand}} \neq \varnothing$ **do**
  $p_{\text{opt}}, s = \max_{p \in P^t_{\text{cand}}} \texttt{ClipSimilarity}(I_r, P^t + \Delta P + p)$
  if $s \geq s_{max}$ **then**
    $\Delta P = \Delta P + p_{\text{opt}}$
  end if
  $P^t_{\text{cand}} = P^t_{\text{cand}} - p_{\text{opt}}$
  $s_{\text{max}} = \max(s_{\text{max}}, s)$
end while
return ΔP

4.2.3 Novel Image Generation

Image reverse prompt engineering helps us understand the intricate connections between prompts and generated images, facilitating the research on text-to-image generation. From the view of practical applications, this technique also goes beyond recreating reference images with text-to-image generation models. For example, as depicted in Figure 4.4, it enables straightforward novel image generation through prompt editing. It is worth noting that the reference image does not necessarily have to be AI-generated; it can be any available image. Therefore, this technology holds promising potential for collaboration between AI and human content generators.

(a) Reference Image (b) Recreated Image (c) Novel Image (Content) (d) Novel Image (Style)

FIGURE 4.4. **Illustration of novel image generation.** (a) Reference image. (b) Image generated with the reverse prompt. (c) Image generated by editing the content-related description in the reverse prompt. (d) Image generated by editing the style-related description in the reverse prompt.

4.3 Experiments

In this section, we evaluate the proposed method with both quantitative and qualitative experiments. We also conduct ablation studies to understand the impact of individual design choices.

4.3.1 Dataset and Evaluation Metrics

To evaluate our ARPO method, we constructed a benchmark dataset of 100 images: 50 human-created (artistic paintings from WikiArt [57] and photographs from [25]) and 50 AI-generated from DiffusionDB [65]. This dataset ensures a diverse set of test images with different styles, including artistic paintings and photographs. We employ two main evaluation metrics: 1) **prompt fidelity** measures the cosine similarity between the CLIP [51] embeddings of the reverse prompt and the reference image. This metric is referred to as `CLIP-T`; 2) **image fidelity** calculates the cosine similarity between the embeddings of the reference image and the recreated image using the reverse prompt. Specifically, we consider three embedding extractors, CLIP (image encoder) [51], DINO [7], and ViT [19], and the metrics are referred to as `CLIP-I`, `DINO`, and `ViT`, respectively. `CLIP-I` measures the cosine similarity between the CLIP embeddings of the generated and reference images, providing a comparison of the semantic differences. `DINO` calculates the cosine similarity between the ViT-S/16 DINO embeddings of the generated and reference images, focusing primarily on comparing the style and detail differences. `ViT` measures the cosine similarity between the ViT-L/16 embeddings of the generated and reference images, primarily responsible for extracting and comparing more complex features of the images.

4.3.2 Quantitative Comparisons

We compare the following methods for image reverse prompt engineering: 1) **VLM-based methods** (BLIP2 [34], ShareCaptioner [9], VILA [37], GPT-4V [1]); 2) **data-driven methods** (CLIP-Interrogator [72]); and 3) **our ARPO variants**. As demonstrated in Table 4.2, advanced VLMs typically excel in generating better reverse prompts. However, attempting to directly

TABLE 4.2. **Results of the image and prompt fidelity comparisons**. The image fidelity metrics are assessed by running each experiment three times with different random seeds to calculate mean and variance. For our ARPO variants, ARPO (VILA, LLaMA2) indicates that we use VILA as VLM and LLaMA2 as LLM in prompt generation.

Method	CLIP-T	CLIP-I	DINO	ViT
BLIP2 [34]	26.42	75.81±0.22	46.06±0.24	44.74±0.16
ShareCaptioner [9]	26.72	76.28±0.05	48.11±0.27	44.97±0.71
VILA [37]	27.76	76.27±0.26	48.53±0.22	45.83±0.05
GPT-4V [1]	28.39	78.14±0.07	50.40±0.18	45.86±0.14
CLIP-Interrogator [72]	30.56	80.62±0.13	50.43±0.14	46.30±0.34
ARPO (VILA, Mistral)	35.03	81.55±0.04	52.37±0.51	49.22±0.43
ARPO (VILA, LLaMA2)	35.14	81.38±0.01	53.37±0.24	49.35±0.14
ARPO (LLaVA-Next, Mistral)	35.27	81.50±0.12	53.69±0.23	50.26±0.14
ARPO (LLaVA-Next, LLaMA2)	**35.58**	**83.01±0.02**	**54.00±0.45**	**51.40±0.27**

TABLE 4.3. Results of using **SDXL**.

Method	CLIP-T	CLIP-I	DINO	ViT
BLIP2 [34]	26.42	75.57	48.95	44.80
ShareCaptioner [9]	30.56	79.99	51.21	47.31
VILA [37]	26.72	76.04	52.88	48.33
GPT-4V [1]	27.76	77.53	53.27	49.00
CLIP-Interrogator [72]	28.39	77.92	54.83	50.35
ARPO (ours)	**35.40**	**81.96**	**55.78**	**52.37**

TABLE 4.4. Results of using **PixArt-α**.

Method	CLIP-T	CLIP-I	DINO	ViT
BLIP2 [34]	26.42	74.49	46.61	43.77
ShareCaptioner [9]	26.72	76.98	50.65	47.68
VILA [37]	27.76	76.90	51.01	47.84
GPT-4V [1]	28.39	78.13	**53.79**	49.09
CLIP-Interrogator [72]	30.56	77.94	49.28	45.02
ARPO (ours)	**35.46**	**79.38**	52.99	**49.48**

utilize general VLMs for this purpose does not yield competitive results, even with the considerable capabilities of GPT-4V. This limitation may arise from their preference for lengthy and complex descriptions, which pose challenges for adaptation by text encoders in Text-to-Image (T2I) models. It is noteworthy that the CLIP-Interrogator outperforms GPT-4V, primarily due to the utilization of a high-quality dataset tailored for specific purposes. Nevertheless, the

limited scope of such a specific dataset hinders achieving significantly better results, as reverse prompts generated by CLIP-Interrogator are constrained to the existing entries in the dataset. Compared to baseline methods, the proposed ARPO consistently achieves significantly better results across various choices of VLM/LLM components. This highlights the effectiveness and robustness of our method for image reverse prompt engineering.

To showcase the adaptability of our ARPO method to different text-to-image models, we explore two additional popular models, SDXL [49] and PixArt-α [8], for comparison with baseline methods. Here, we utilize the combination of LLaVA-Next and LLaMA2 for prompt generation in our ARPO framework. As illustrated in Table 4.3, when using SDXL for text-to-image generation, our ARPO consistently outperforms baseline methods by a significant margin, similar to the observations made with Stable-Diffusion-V1.5 cases in Table 4.2. Furthermore, Table 4.4 presents the results obtained using PixArt-α as the text-to-image model. Specifically, we observe comparable results between GPT-4V and our method. This could be because PixArt-α uses Google-T5 [12] as the text encoder, which is better suited to handle the detailed and complex descriptions generated by GPT-4V. In summary, these experimental results highlight the strong performance and adaptability of our method across various text-to-image models.

4.3.3 User Study

We also conduct a user study to compare images generated by six methods: hand-crafted, GPT-4V, CLIP-Interrogator, a commercial service, and two variants of our ARPO method. The study involved 40 cases, with 50 users ranking the methods based on similarity to the reference image, considering content, style, and overall preference. The first rank received a score of 6, and the sixth rank received a score of 1. As shown in Figure 4.5, both ARPO variants achieved comparable performances, clearly surpassing the hand-crafted method, GPT-4V, CLIP-Interrogator, and the commercial service.

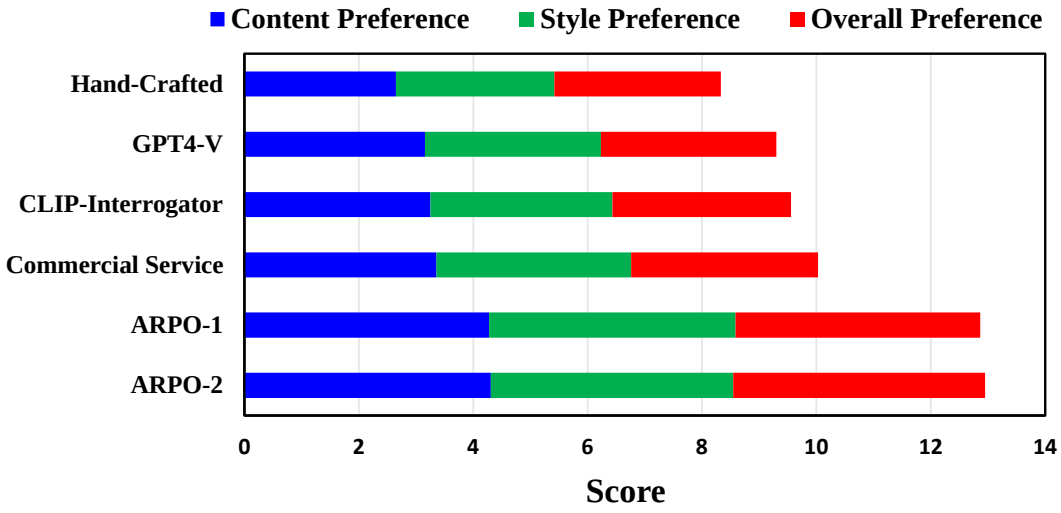


FIGURE 4.5. **The results of our user study.** ARPO-1 uses the combination of LLaVA-Next and LLaMA2, while ARPO-2 uses the combination of GPT-4V and GPT-4 for candidate prompt generation.

FIGURE 4.6. **Illustration of recreated images** using the reverse prompts from different methods.

4.3.4 Qualitative Comparisons

We compare recreated images using reverse prompts from our ARPO method and several baseline methods, including the hand-crafted method, BLIP2, Share-Captioner, VILA, GPT-4V, CLIP-Interrogator, and commercial services like `Phot.AI`[3]. In Figure 4.6, the reference

[3]https://www.phot.ai

TABLE 4.5. **Ablation studies** on Prompt Generator and CLIP Filter.

Method	CLIP-T	CLIP-I	DINO	ViT
ARPO	35.58	83.01±0.02	54.00±0.45	51.40±0.27
w/o Enhanced Prompt Generation	33.04	78.54±0.18	50.96±0.25	47.98±0.11
w/o Prompt Selection	27.09	78.65±0.26	49.76±0.05	47.82±0.16

images in different rows are a natural image, an art painting, an AI-generated image, and an AI-generated painting, respectively. Notably, the hand-crafted reverse prompts were designed by an experienced AIGC researcher, yet the recreated images still show significant discrepancies from the reference images. Furthermore, images recreated with reverse prompts from general VLMs such as BLIP2, Share-Captioner, VILA, and GPT-4V often struggle with art paintings or AI-generated paintings in terms of both content and style. Methods specifically designed for reverse prompt engineering, such as CLIP-Interrogator and commercial services, usually perform better in capturing details (e.g., the fourth row) but may also show significant discrepancies from the reference image in some cases (e.g., the second row). Compared to these baseline methods, our ARPO method better preserves the various elements of the reference image. For example, for artistic paintings that require more detailed prompts to guide image generation, our ARPO method excels in imitating style, layout, and color.

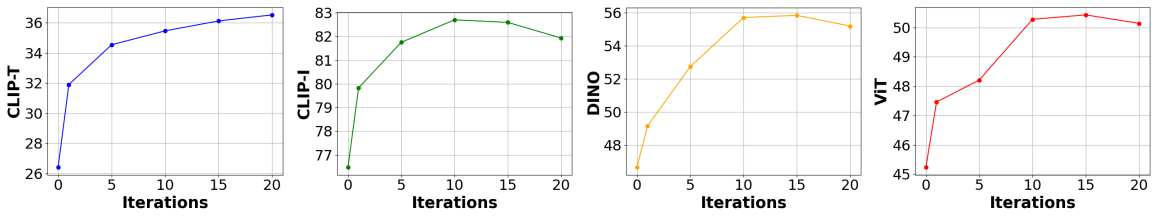
4.3.5 Ablation Studies

Prompt Generation To generate candidate prompts, we have introduced an enhanced framework to fully utilize open-source VLMs/LLMs, as described in Section 4.2.2. As shown in Table 4.5, the describe-then-compare strategy in the enhanced framework significantly mitigates the performance degradation caused by some weaker VLMs/LLMs. This also suggests the importance of the chain-of-thought method, which enables complex reasoning through intermediate steps.

Prompt Selection As shown in Table 4.5, using all generated candidate prompts to update the reverse prompt leads to significant performance degradation. This highlights the importance of the prompt selection process, likely because: 1) prompt generation models often provide

incorrect or unhelpful suggestions for reducing differences between reference and generated images, and 2) handling these moderate suggestions slows the optimization process.

Optimization Steps In Figure 4.7, we illustrate performances across different numbers of optimization steps. Specifically, a maximum of 10 steps is typically sufficient in most cases. When running more iterations, there may be some performance fluctuations, but differences in the generated images are typically imperceptible to the human eye until many more optimization steps are taken. Therefore, employing an early stop strategy would be useful, otherwise it is also easy for us to empirically determine the proper number of steps in practice. Notably, even with two optimization steps, it already consistently outperforms all other methods.



(a) CLIP-T (b) CLIP-I (c) DINO (d) ViT

FIGURE 4.7. The **influence** of **optimization steps**.

4.4 Summary

In this chapter, we addressed the challenge of image reverse prompt engineering and introduced an automatic reverse prompt optimization method. Using the obtained reverse prompts, we can easily generate novel images with content or style similar to the reference image. Our method consistently outperforms many baseline approaches, including hand-crafted methods, image caption methods, data-driven methods, and commercial services, as demonstrated by both quantitative and qualitative results. This research underscores the promising potential of reverse prompt engineering for text-to-image generation. We hope that this research can not only advance the study of image reverse prompt engineering but also extend to text-to-3D and text-to-video generation.

4.5 Limitation

A major limitation of the ARPO method is that it requires a multi-step optimization process to generate reverse prompts, resulting in a higher time cost compared to other methods. It would be beneficial to further improve the efficiency of the proposed ARPO method or explore a novel efficient image reverse prompt engineering solution.

CHAPTER 5

Conclusions

This chapter delivers an extensive summary of the contributions discussed in the preceding chapters and suggests possible avenues for future research.

5.1 Summary of Contributions

In this thesis, we explore how to further enhance Diffusion Models for generate expected results from both theoretical and application perspectives. On the theoretical level, we investigate the exposure bias problem inherent in Diffusion Models, while on the application level, we address the challenges associated with designing textual prompts for Diffusion Models. The detailed contributions can be summarized as follow:

(1) Exposure bias can cause noise generated by model predictions in the inference process of Diffusion Models being continuously accumulated and amplified, ultimately resulting in generated outcomes that do not meet expectations. To alleviate exposure bias problem, we first analyze the causes. Upon analyzing the formula for the inference process, we discovered that noise from the model prediction influences both the model's subsequent prediction and the input signal in the following step. To reduce this noise and bridge the gap between the training and inference processes, we first propose a single-step denoising scheduled sampling (SDSS). Within this framework, We propose a new training object to reduce the noise in model prediction and an additional denoiser to reduce the noise in the input signal. To account for error accumulation across multiple steps, we introduce the multi-step denoising scheduled sampling (MDSS), designed to mathematically emulate this accumulation.

The MDSS method can be plugged into any existing DMs, requiring merely a few additional training iterations on the well-trained model. Our method can effectively alleviate the exposure bias issue, reducing noise in the generate results of DMs, improving the generation quality, and ensuring that the results meet expectations.

(2) Designing prompts for Diffusion Models requires experience and skill. An inappropriate prompt will not effectively reflect the creator's intent, resulting in outcomes that do not meet expectations. To address the challenge of designing difficult textual prompts, we introduce the image reverse prompt engineering (IRPE) task, which generates reverse prompts from existing images to serve as design references. We propose an automatic prompt optimization framework for IRPE, called ARPO. The ARPO method first generate an initially reverse prompt through BLIP2 and then optimize this prompt with three main steps: 1) Image Generation: generating a new image from the current reverse prompt; 2) Prompt Generation: Comparing candidate prompts by comparing generated and reference images; 3) Prompt Selection: updating the reverse prompt. This research underscores the promising potential of reverse prompt engineering for text-to-image generation and provides a novel and effective auxiliary method for designing prompts. Our method provides users with prompt references from images, enabling them to design prompts that are more suitable for DMs, effectively aligning the generated images with user expectations.

5.2 Future Research

The research in this thesis is in its early stages and has yet to reach saturation. In the end of this thesis, we list some potential directions for future research:

(1) **Other methods to address these two challenges.** For the challenges of exposure bias and prompt designing, there are still other methods worth exploring. Our MDSS method mitigates exposure bias by modifying the model's training strategy. Exploring how to alleviate exposure bias by modifying the inference strategy presents

a valuable research direction. Additionally, our proposed ARPO method is a multi-round optimization and training-free approach. Investigating how to generate reverse prompts through a single round of optimization by training VLMs and LLMs could also be worthwhile.

(2) **Other methods to better generate expected results.** Besides the two challenges highlighted in this thesis, we believe there are other challenges within Diffusion Models that await resolution, which could lead to outcomes more in line with expectations. These additional challenges might involve building more robust models, automatically refining user-designed prompts, or improving the interaction between users and Diffusion Models.

(3) **Exploring different tasks.** This paper primarily focuses on the image generation tasks. However, the challenges of exposure bias and designing prompts also exist in other generative tasks, such as video and audio generation. Investigating how to apply our proposed MDSS and ARPO methods to these different generation tasks is a direction worth exploring.

Bibliography

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat et al. 'Gpt-4 technical report'. In: *arXiv preprint* (2023).

[2] Fan Bao, Chongxuan Li, Jun Zhu and Bo Zhang. 'Analytic-DPM: an Analytic Estimate of the Optimal Reverse Variance in Diffusion Probabilistic Models'. In: *ICLR*. 2022.

[3] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud and Jörn-Henrik Jacobsen. 'Invertible residual networks'. In: *ICML*. 2019.

[4] Samy Bengio, Oriol Vinyals, Navdeep Jaitly and Noam Shazeer. 'Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks'. In: *NeurIPS*. 2015.

[5] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo et al. 'Improving image generation with better captions'. In: *Computer Science* (2023).

[6] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler and Karsten Kreis. 'Align your latents: High-resolution video synthesis with latent diffusion models'. In: *CVPR*. 2023.

[7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski and Armand Joulin. 'Emerging properties in self-supervised vision transformers'. In: *ICCV*. 2021.

[8] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu et al. 'PixArt-α: Fast Training of Diffusion Transformer for Photorealistic Text-to-Image Synthesis'. In: *ICLR*. 2024.

[9] Lin Chen, Jisong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao and Dahua Lin. 'Sharegpt4v: Improving large multi-modal models with better captions'. In: *arXiv preprint* (2023).

[10] Ricky TQ Chen, Jens Behrmann, David K Duvenaud and Jörn-Henrik Jacobsen. 'Residual flows for invertible generative modeling'. In: *NeurIPS*. 2019.

[11] Yang Chen, Yingwei Pan, Haibo Yang, Ting Yao and Tao Mei. 'VP3D: Unleashing 2D Visual Prompt for Text-to-3D Generation'. In: *CVPR*. 2024.

[12] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma et al. 'Scaling instruction-finetuned language models'. In: *Journal of Machine Learning Research* (2024).

[13] Guillaume Couairon, Jakob Verbeek, Holger Schwenk and Matthieu Cord. 'Diffedit: Diffusion-based semantic image editing with mask guidance'. In: *arXiv preprint* (2022).

[14] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung and Steven Hoi. 'Instructblip: Towards general-purpose vision-language models with instruction tuning'. In: *NeurIPS*. 2023.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei. 'Imagenet: A large-scale hierarchical image database'. In: *CVPR*. 2009.

[16] Yuntian Deng, Noriyuki Kojima and Alexander M Rush. 'Markup-to-Image Diffusion Models with Scheduled Sampling'. In: *ICLR*. 2022.

[17] Prafulla Dhariwal and Alexander Quinn Nichol. 'Diffusion Models Beat GANs on Image Synthesis'. In: *NeurIPS*. 2021.

[18] Laurent Dinh, David Krueger and Yoshua Bengio. 'Nice: Non-linear independent components estimation'. In: *arXiv preprint* (2014).

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly et al. 'An image is worth 16x16 words: Transformers for image recognition at scale'. In: *ICLR*. 2021.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. 'Generative adversarial nets'. In: *NeurIPS*. 2014.

[21] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler and Sepp Hochreiter. 'Gans trained by a two time-scale update rule converge to a local nash equilibrium'. In: *NeurIPS*. 2017.

[22] Jonathan Ho, Ajay Jain and Pieter Abbeel. 'Denoising Diffusion Probabilistic Models'. In: *NeurIPS 2020*. 2020.

[23] Jonathan Ho and Tim Salimans. 'Classifier-free diffusion guidance'. In: *ArXiv preprint* (2022).

[24] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi and David J Fleet. 'Video diffusion models'. In: *NeurIPS*. 2022.

[25] Yongcheng Jing, Xiao Liu, Yukang Ding, Xinchao Wang, Errui Ding, Mingli Song and Shilei Wen. 'Dynamic instance normalization for arbitrary style transfer'. In: *AAAI*. 2020.

[26] Gurusha Juneja and Amit Sharma. 'A universal prompt generator for large language models'. In: *NeurIPS workshop*. 2023.

[27] Diederik P Kingma and Max Welling. 'Auto-encoding variational bayes'. In: *arXiv preprint* (2013).

[28] Diederik Kingma, Tim Salimans, Ben Poole and Jonathan Ho. 'Variational diffusion models'. In: *NeurIPS*. 2021.

[29] Durk P Kingma and Prafulla Dhariwal. 'Glow: Generative flow with invertible 1x1 convolutions'. In: *NeurIPS*. 2018.

[30] Ivan Kobyzev, Simon JD Prince and Marcus A Brubaker. 'Normalizing flows: An introduction and review of current methods'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2020).

[31] Zhifeng Kong and Wei Ping. 'On fast sampling of diffusion probabilistic models'. In: *ArXiv preprint* (2021).

[32] Alex Krizhevsky, Geoffrey Hinton et al. 'Learning multiple layers of features from tiny images'. In: (2009).

[33] Kyungmin Lee, Kihyuk Sohn and Jinwoo Shin. 'DreamFlow: High-Quality Text-to-3D Generation by Approximating Probability Flow'. In: *ICLR*. 2024.

[34] Junnan Li, Dongxu Li, Silvio Savarese and Steven Hoi. 'Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models'. In: *ICML*. 2023.

[35] Junnan Li, Dongxu Li, Caiming Xiong and Steven Hoi. 'Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation'. In: *ICML*. 2022.

[36] Mingxiao Li, Tingyu Qu, Wei Sun and Marie-Francine Moens. 'Alleviating Exposure Bias in Diffusion Models through Sampling with Shifted Time Steps'. In: *ArXiv preprint* (2023).

[37] Ji Lin, Hongxu Yin, Wei Ping, Yao Lu, Pavlo Molchanov, Andrew Tao, Huizi Mao, Jan Kautz, Mohammad Shoeybi and Song Han. 'Vila: On pre-training for visual language models'. In: *arXiv preprint* (2023).

[38] Haotian Liu, Chunyuan Li, Yuheng Li and Yong Jae Lee. 'Improved baselines with visual instruction tuning'. In: *arXiv preprint* (2023).

[39] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen and Yong Jae Lee. *LLaVA-NeXT: Improved reasoning, OCR, and world knowledge*. 2024. URL: https://llava-vl.github.io/blog/2024-01-30-llava-next/.

[40] Haotian Liu, Chunyuan Li, Qingyang Wu and Yong Jae Lee. 'Visual instruction tuning'. In: *NeurIPS*. 2023.

[41] 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'. In: *NeurIPS*. 2022.

[42] Calvin Luo. 'Understanding diffusion models: A unified perspective'. In: *ArXiv preprint* (2022).

[43] Gautam Mittal, Jesse Engel, Curtis Hawthorne and Ian Simon. 'Symbolic music generation with diffusion models'. In: *ArXiv preprint* (2021).

[44] Alexander Quinn Nichol and Prafulla Dhariwal. 'Improved Denoising Diffusion Probabilistic Models'. In: *ICML*. 2021.

[45] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever and Mark Chen. 'GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models'. In: *ICML*. 2022.

[46] Mang Ning, Enver Sangineto, Angelo Porrello, Simone Calderara and Rita Cucchiara. 'Input Perturbation Reduces Exposure Bias in Diffusion Models'. In: *ArXiv preprint* (2023).

[47] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed and Balaji Lakshminarayanan. 'Normalizing flows for probabilistic modeling and inference'. In: *Journal of Machine Learning Research (JMLR)* (2021).

[48] Gaurav Parmar, Richard Zhang and Jun-Yan Zhu. 'On aliased resizing and surprising subtleties in gan evaluation'. In: *CVPR*. 2022.

[49] 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'. In: *ICLR*. 2024.

[50] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu and Michael Zeng. 'Automatic prompt optimization with" gradient descent" and beam search'. In: *EMNLP*. 2023.

[51] 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: *ICML*. 2021.

[52] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu and Mark Chen. 'Hierarchical text-conditional image generation with clip latents'. In: *arXiv preprint* (2022).

[53] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli and Wojciech Zaremba. 'Sequence Level Training with Recurrent Neural Networks'. In: *ICLR*. 2016.

[54] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser and Björn Ommer. 'High-Resolution Image Synthesis With Latent Diffusion Models'. In: *CVPR*. 2022.

[55] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein and Kfir Aberman. 'Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation'. In: *CVPR*. 2023.

[56] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans et al. 'Photorealistic text-to-image diffusion models with deep language understanding'. In: *NeurIPS*. 2022.

[57] Babak Saleh and Ahmed Elgammal. 'Large-scale classification of fine-art paintings: Learning the right metric on the right feature'. In: *arXiv preprint* (2015).

[58] Tim Salimans and Jonathan Ho. 'Progressive Distillation for Fast Sampling of Diffusion Models'. In: *ICLR*. 2022.

[59] Florian Schmidt. 'Generalization in Generation: A closer look at Exposure Bias'. In: *Proceedings of the 3rd Workshop on Neural Generation and Translation*. 2019.

[60] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace and Sameer Singh. 'Autoprompt: Eliciting knowledge from language models with automatically generated prompts'. In: *EMNLP*. 2020.

[61] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni et al. 'Make-a-video: Text-to-video generation without text-video data'. In: *ICLR*. 2023.

[62] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan and Surya Ganguli. 'Deep Unsupervised Learning using Nonequilibrium Thermodynamics'. In: *ICML*. 2015.

[63] Jiaming Song, Chenlin Meng and Stefano Ermon. 'Denoising Diffusion Implicit Models'. In: *ICLR*. 2021.

[64] Arun Venkatraman, Martial Hebert and J. Andrew Bagnell. 'Improving Multi-Step Prediction of Learned Time Series Models'. In: *AAAI*. 2015.

[65] Zijie J Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover and Duen Horng Chau. 'Diffusiondb: A large-scale prompt gallery dataset for text-to-image generative models'. In: *ACL*. 2023.

[66] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou et al. 'Chain-of-thought prompting elicits reasoning in large language models'. In: *NeurIPS*. 2022.

[67] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou and Xinyun Chen. 'Large language models as optimizers'. In: *ICLR*. 2024.

[68] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui and Ming-Hsuan Yang. 'Diffusion models: A comprehensive survey of methods and applications'. In: *ArXiv preprint* (2022).

[69] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser and Jianxiong Xiao. 'Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop'. In: *ArXiv preprint* (2015).

[70] Wen Zhang, Yang Feng, Fandong Meng, Di You and Qun Liu. 'Bridging the Gap between Training and Inference for Neural Machine Translation'. In: *ACL*. Florence, Italy, 2019.

[71] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan and Jimmy Ba. 'Large language models are human-level prompt engineers'. In: *arXiv preprint* (2022).

[72] *clip-interrogator*. https://github.com/pharmapsychotic/clip-interrogator. 2023.

APPENDIX A

Additional Details in Chapter 3

A1 More Details of MDSS

A1.1 Derivation of Equation 3.9

In Equation 3.9, we compute a multi-step posterior inverse process. Given that the posterior inverse process is a Markov chain, it can be computed using a *For loop* as demonstrated in Alg. 9.

Algorithm 9 Multi-step posterior inverse process

1: **for** $i \leftarrow k$ to 1 **do**
2: $\quad y = \frac{\sqrt{\alpha_{t+i}}(1-\bar{\alpha}_{t+i-1})}{1-\bar{\alpha}_{t+i}}\mathbf{x} + \frac{\sqrt{\bar{\alpha}_{t+i-1}}\beta_{t+i}}{1-\bar{\alpha}_{t+i}}\mathbf{x}_0$
3: $\quad x = y$
4: **end for**
5: **return** x

The algorithm demonstrates that the intermediate time step can be omitted. For instance, when $k = 2$, the result of the multi-step posterior process can be expressed as follow:

$$\begin{aligned}\mathbf{x}_t =& \frac{\sqrt{\alpha_{t+1}}(1-\bar{\alpha}_t)\sqrt{\alpha_{t+2}}(1-\bar{\alpha}_{t+1})}{(1-\bar{\alpha}_{t+1})(1-\bar{\alpha}_{t+2})}\mathbf{x}_{t+2} + \\ & \frac{\sqrt{\alpha_{t+1}}(1-\bar{\alpha}_t)\sqrt{\bar{\alpha}_{t+1}}\beta_{t+2}}{(1-\bar{\alpha}_{t+1})(1-\bar{\alpha}_{t+2})}\mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t}\beta_{t+1}}{1-\bar{\alpha}_{t+1}}\mathbf{x}_0\end{aligned} \tag{A.1}$$

It can be observed that all variants are derived from the noise schedule except $\mathbf{x}_{t+k}$ and x_0. We summarize the variants related to the step k as γ and ω. These variants are contingent upon the time step t. In light of your valuable correction, we will amend them to γ_t and ω_t in Algorithm 3. By specifying the step k before training, it becomes feasible to calculate variants for each time step in advance. Therefore, we can remove the *For loop* in training algorithm and speed up the training process.

A1.2 Scheduled Sampling

In the scheduled sampling method, a random decision determines whether the training adheres to the original DDPM method or transitions to our MDSS method. We use γ to represent the probability of using MDSS in a mini-batch of training. When $\gamma = 0$, the model training process is just as DDPMs, while when $\gamma = 1$ the training process only uses MDSS. We incorporate MDSS at the beginning of training, while retaining a slight portion of the original DDPM during the final training phase. Therefore, the minimum γ is 0.1 and the maximum γ is 0.9. We utilize the linear increase strategy:

$$\gamma = \min(0.9, 0.1 + \frac{t}{T}) \tag{A.2}$$

where t is the current training step and T is the maximum training iterations.

A1.3 Model Structure of Denoising Input Signal

In MDSS, in order to mitigate the errors present in the input signal, we propose a new denoiser. This denoiser shares the same UNet with the original output and adds an additional output channel in the last layer to predict the noise in the input signal. The additional channel has the same shape as the original one. The model structure is shown in Fig. A.1.

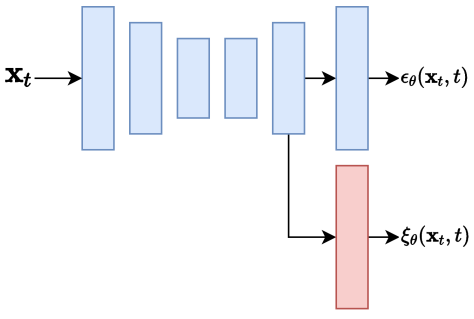


FIGURE A.1. **The schematic diagram of the model structure.** The blue layer keeps the structure of the original UNet unchanged and adds a new channel in red in the last layer for predicting noise in the input signal.

A2 More Details of Experimental Setup

For the CIFAR-10 and ImageNet 64×64 datasets, we fine-tune on the well-trained iDDPM models, and for LSUN tower 64×64, we keep training on the ADM model. We use Pytorch 2.0 and all experiments on NVIDIA V100s (32G memory). We use 2 GPUs to train CIFAR-10, 8 GPUs to train ImageNet 64×64, and 4 GPUs to train LSUN tower 64×64. We train all models with the AdamW optimizer. We utilize Frechet Inception Distance (FID) to evaluate the quality of the generated images. To evaluate the FID metric, we generate 50,000 images as a sample batch, except for ImageNet 64×64 where we produce 10K images. The reference batch for CIFAR-10 consists of the entire training set, while the reference batch for ImageNet 64×64 and LSUN tower 64×64 are randomly chosen images from the training set. The hyperparameters are reported in Table. A.1.

TABLE A.1. **Hyperparameters** in CIFAR-10, ImageNet and LSUN experiments.

	CIFAR-10	ImageNet	LSUN
Image size	32	64	64
Diffusion steps	4000	4000	1000
Noise schedule	cosine	cosine	cosine
Channels	128	128	192
Residual blocks	3	3	3
Channels multipe	1,2,2,2	1,2,3,4	1,2,3,4
Attention resolution	16,8	16,8	32,16,8
Iterations	50k	10k	50k
Batch size	128	128	256
Dropout	0.3	0	0
EMA rate	0.9999	0.9999	0.9999
Learning rate	1e-4	1e-5	1e-4
Sample batch	50k	10k	50k

A3 More Discussion of Training based on well-trained Models

In this section, we compare the results of DDPM and MDSS fine-tuning on the well-trained model. We conduct our experiments on CIFAR-10 and calculate the FID of 250 sampling

TABLE A.2. **DPM-Solver-1,2,3 results** on ImageNet 64×64.

Methods	Steps	DDPM	IP	SS	SDSS	MDSS
DPM-Solver-1	10 steps	64.31	127.18	63.48	63.45	**63.29**
	20 steps	39.30	82.59	40.18	39.87	**38.67**
	50 steps	29.97	66.71	29.91	29.88	**29.45**
DPM-Solver-2	10 steps	42.19	66.58	40.33	**39.90**	41.49
	20 steps	**27.92**	41.17	28.69	28.34	28.81
	50 steps	**25.72**	56.67	26.10	26.33	26.53
DPM-Solver-3	10 steps	**103.12**	189.25	180.20	169.97	112.58
	20 steps	**26.83**	85.74	28.91	28.11	28.46
	50 steps	25.42	59.29	25.71	**25.35**	25.94

steps on every 10,000 iterations. The results are shown in Fig. A.2. We can find out that, using MDSS, the FID experiences a sharp decline in the initial phases of training and begins to converge after approximately 20,000 iterations in MDSS training. In contrast, using the original optimization of DDPMs, the FID remains stable without showing any noticeable decrease.

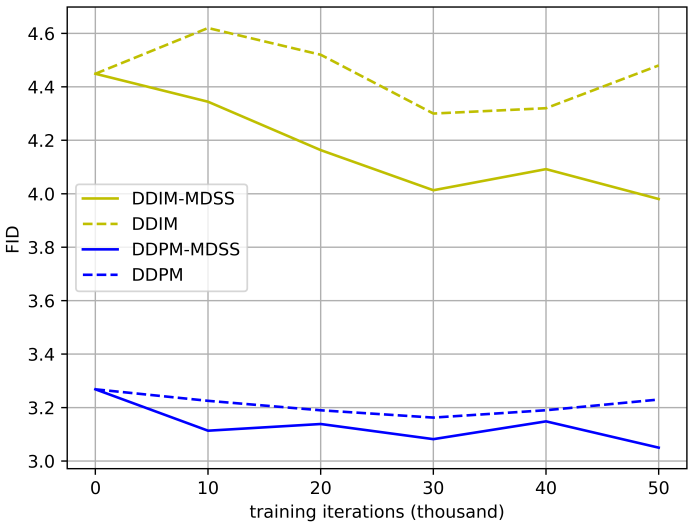


FIGURE A.2. FID scores with respect to **the number of training iterations**. Each FID result is computed with 250 inference steps.

A4 DPM-Solver Results on ImageNet and LSUN

The results of DPM-Solver on ImageNet 64×64 and LSUN are shown in Table. A.2 and Table. A.3. We can draw a conclusion that in DPM-Solver-1, our method can achieve superior sampling results compared to other approaches, and in DPM-Solver-2 and 3, while our

TABLE A.3. **DPM-Solver-1,2,3 results** on LUSN tower 64×64.

Methods	Steps	DDPM	IP	SS	SDSS	MDSS
DPM-Solver-1	10 steps	21.32	97.53	21.37	21.85	**20.83**
	20 steps	9.71	57.46	9.84	10.95	**9.47**
	50 steps	6.70	44.93	6.85	6.77	**6.15**
DPM-Solver-2	10 steps	8.93	56.69	8.77	9.82	**8.64**
	20 steps	**4.95**	18.10	5.17	5.74	5.15
	50 steps	**4.77**	36.30	5.43	5.58	5.22
DPM-Solver-3	10 steps	71.46	419.92	69.45	82.50	**59.05**
	20 steps	**5.34**	69.54	6.38	6.32	6.17
	50 steps	**4.90**	40.68	5.88	5.72	5.16

approach outperforms IP and SS, there is minimal or no improvement compared to the DDPM baseline.

A5 CLIP-FID Results

We can use CLIP-FID [48] to evaluate the quality of the generated images. In this section, we use CLIP-FID to evaluate images generated using DDPM and DDIM sampling methods on CIFAR-10. We sample the images for 30, 100, and 250 steps. The experimental results are shown in Fig. A.4. As shown in the experiments, our method still achieves the state-of-the-art results in most cases.

Steps	DDPM	IP	SS	SDSS	MDSS
DDPM 30	4.43	3.36	4.23	3.97	**3.26**
DDPM 100	1.63	**1.30**	1.53	1.46	2.17
DDPM 250	1.05	**0.99**	1.02	1.01	2.02
DDIM 30	3.28	2.11	2.93	2.79	**2.02**
DDIM 100	1.69	2.72	1.42	1.34	**1.06**
DDIM 250	1.26	3.12	1.15	**1.02**	1.05

TABLE A.4. CLIP-based FID results on DDPM and DDIM inference method on CIFAR-10.

A6 Qualitative Results

In this section, we present the example images generated using DDPM and DDIM of MDSS. We sample the images for 30, 100, and 250 steps, respectively. The generated images of

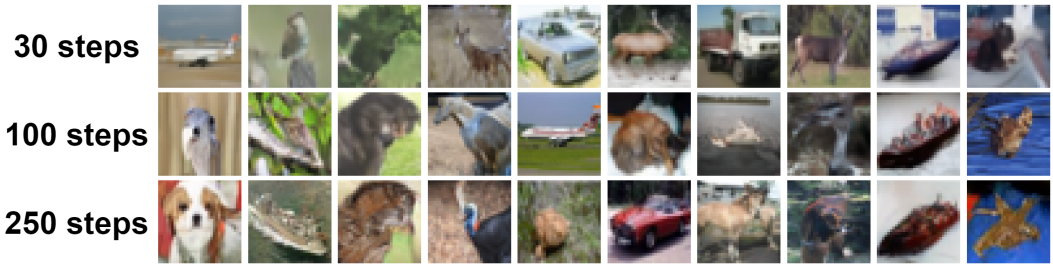


FIGURE A.3. **CIFAR-10 DDPM sample results** for **varying time steps** of MDSS

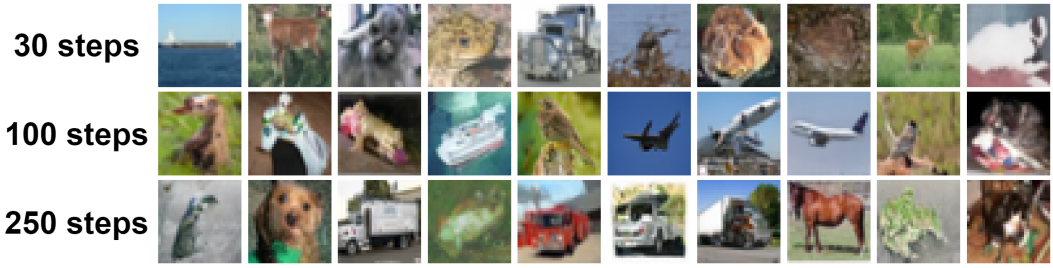


FIGURE A.4. **CIFAR-10 DDIM sample results** for **varying time steps** of MDSS

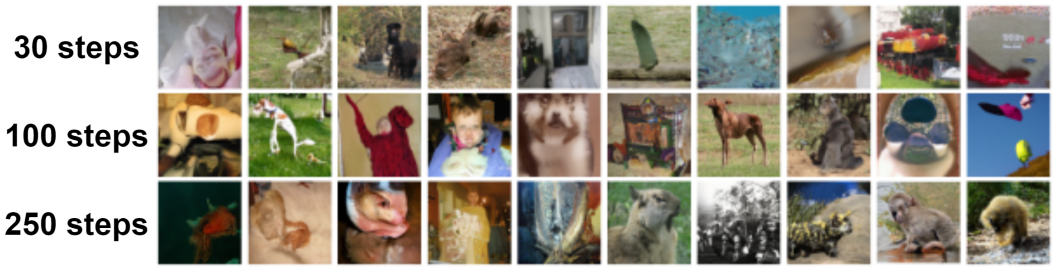


FIGURE A.5. **ImageNet** 64×64 **DDPM sample results** for **varying time steps** of MDSS

CIFAR-10, ImageNet 64×64 and LSUN tower 64×64 on DDPM can be found in Fig. A.3, Fig. A.5 and Fig. A.7, while the DDIM generated images is shown in Fig. A.4, Fig. A.6 and Fig. A.8. We compared the generated images of MDSS and DDPM baseline with DDPM and DDIM generated method in Fig. A.9 and Fig. A.10. It can be seen that we can generated images with competitive quality for both CIFAR-10, ImageNet 64×64, and LSUN tower 64×64. By comparing with the generated results of DDPM baseline, MDSS can achieve better results with fewer steps, especially in DDIM 30-step and 100-step sampling.

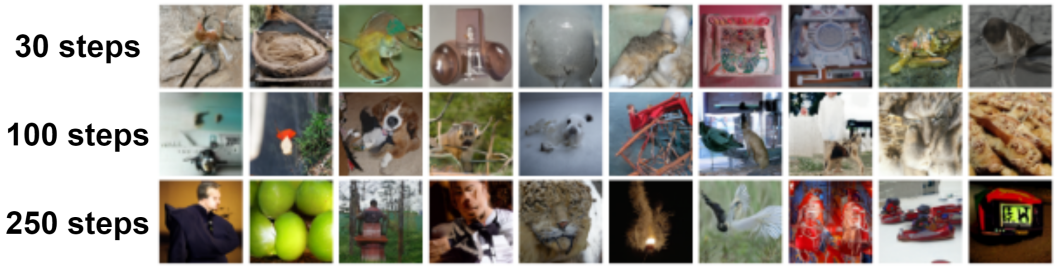


FIGURE A.6. **ImageNet** 64×64 **DDIM sample results** for **varying time steps** of MDSS

FIGURE A.7. **LSUN tower** 64×64 **DDPM sample results** for **varying time steps** of MDSS

FIGURE A.8. **LSUN tower** 64×64 **DDIM sample results** for **varying time steps** of MDSS

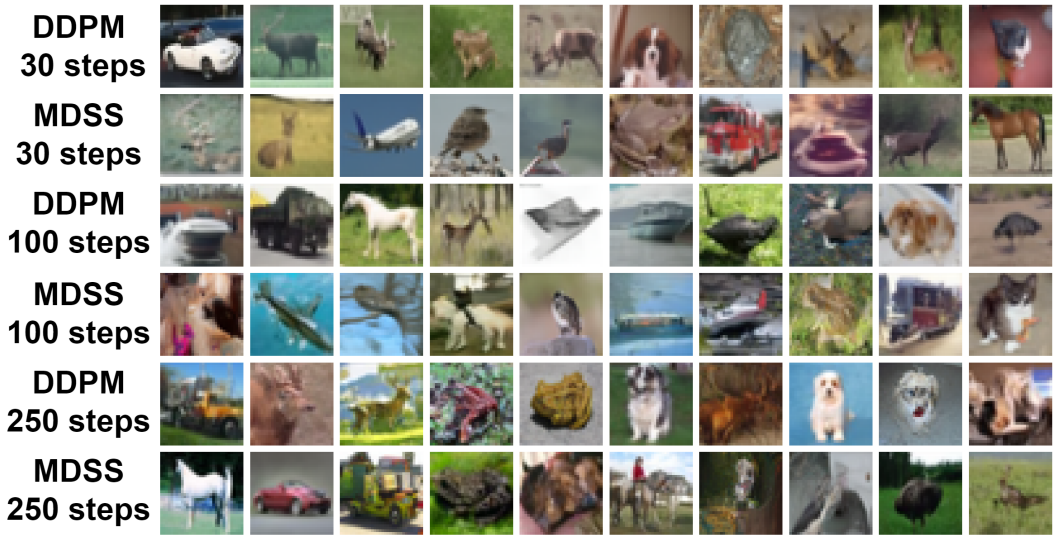


FIGURE A.9. **Compared the DDPM sample results** between DDPM baseline and MDSS.

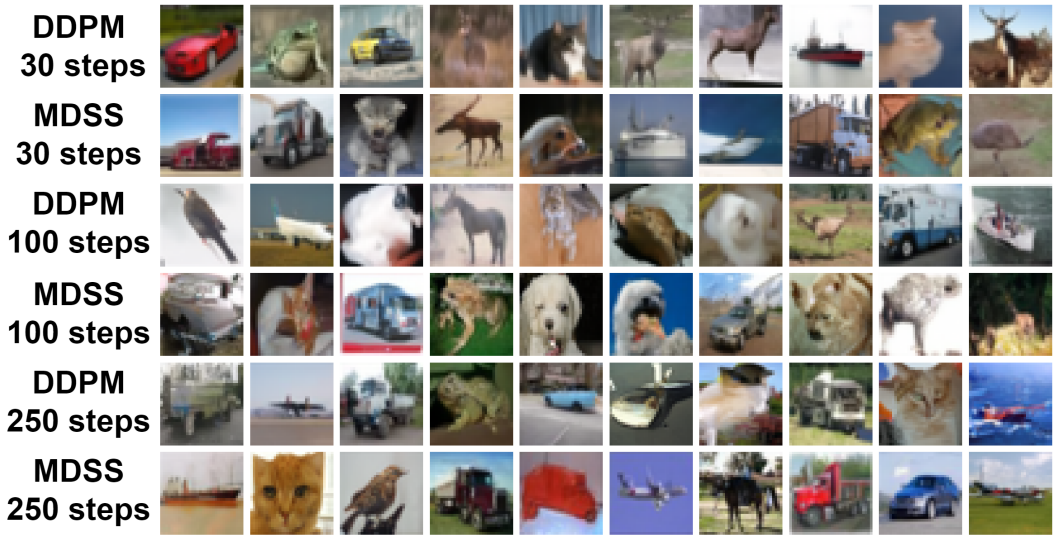


FIGURE A.10. **Compared the DDIM sample results** between DDPM baseline and MDSS.

APPENDIX B

Additional Details in Chapter 4

B1 Computational Resource

For the ARPO method with closed-source models and vanilla prompt generation framework, we utilize GPT4-V (gpt-4-vision-preview) and GPT4 (gpt-4-1106-preview) as the VLM and LLM. For the ARPO with open-source models and enhanced prompt generation process, we try to utilize VILA (VILA-13b, CC-BY-NC-4.0) or LLaVA-Next (llava-v1.6-vicuna-13b, LLAMA 2 Community License) as our VLM and LLaMA2 (Llama-2-13b-chat-hf, Llama 2 Community License) or Mistral (Mistral-7B-Instruct-v0.2, APACHE-2.0) as our LLM. We employ Stable-Diffusion-V1.5 (stable-diffusion-v1-5, creativeml-openrail-m), SDXL (stable-diffusion-xl-base-1.0, openrail++), and PixArt-α (PixArt-XL-2-512$\times$512,openrail++) as our Text-to-Image model to test our method can handle different image generators. In the prompt initialization step, we use BLIP2 (blip2-opt-2.7b, MIT License) to get a concise image caption. In the step of prompt selection, we use CLIP-ViT-L-14 (MIT License) to calculate the cosine similarity between text and image CLIP embeddings.

We deploy our closed-source models version on a single NVIDIA 4090 24G GPU and our open-source models version on two NVIDIA A100 40G GPUs. The closed-source version takes about 20 seconds and costs approximately 0.07 USD per iteration. In contrast, due to the additional image description process in the enhanced prompt generation framework, the open-source version takes around 90 seconds per iteration but is completely free.

B2 More details about Prompt Generation

B2.1 Vanilla Framework

In the framework of the vanilla prompt generation process, we first utilize the VLM model to compare the difference between the generated image and the reference image. The template used by VLM model to compare image differences is shown in Figure B.1. An example of difference description generated by GPT4-V is shown in Figure B.2. Then we employ LLM to generate candidate prompts for minimizing each of the identified differences. The template used by LLM is shown in Figure B.3 and a candidate prompts example generated by GPT-4 is shown in Figure B.4.

Template used by the VLLM to compare image difference.

The first image is Image 1 and the second image is Image 2. You need to describe the difference between Image 1 and Image 2. Let's think step by step.

FIGURE B.1. Template used by the VLLM to **compare image difference** between the reference image and the generated image.

An example of difference description

Image 1 and Image 2 both feature a cat wearing a bow tie, but there are notable differences. Image 1 depicts a cat with a more stylized and artistic rendering, showcasing a softer texture and a more whimsical feel with its exaggerated large blue eyes and a light blue bow tie that has a silky texture. The cat's fur appears smooth and the overall image has a cool tone, contributing to a serene mood. In contrast, Image 2 presents a cat with a realistic photograph quality, with sharper textures seen in the fur, a solid black and white color pattern on the fur, and more natural but still vivid blue eyes. The bow tie here is darker and less reflective, suggesting a different material. The background of Image 2 is a neutral brown, which contrasts with the cooler, more monochromatic background of Image 1. The mood in Image 2 feels more straightforward and less fanciful than in Image 1. The layout and scale of the subjects are similar, with both cats positioned centrally and occupying a comparable amount of space within the frame, but the style of Image 1 is more illustrative and fantastical, while Image 2 is more realistic and lifelike.

FIGURE B.2. An example of **difference description** generated by GPT4-V

Template used by the LLM to generate prompt candidates.

Generate image promts that incorporate the following difference between Image 1 and Image 2: {difference}.
For the specific contrasts identified in the differences between Image 1 and Image 2, the image prompts should guide the creation of images that align more closely with Image 1.
The prompts should be structured as a series of keywords or short phrases, separated by commas. Please list all possible prompts in a python list format. Your answer should only contain a python list. Let's think step by step.

FIGURE B.3. Template used by the LLM to **generate prompt candidates** based on the difference between two image.

An example of candidate prompts

[stylized artistic rendering of a cat, exaggerated large blue eyes, light blue silky bow tie, smooth fur texture, cool tone background, serene mood, whimsical feel, illustrative and fantastical style, soft texture visual, monochromatic color scheme]

FIGURE B.4. An example of **candidate prompts** generated by GPT4

B2.2 Enhanced Framework

Using open-source models like VILA and LLaVA-Next with relatively lower performing compared to GPT4-V to generate image differences has two significant challenges: 1) Some VLMs are trained solely on single image-text pairs and lack the capability for multi-image inference. 2) Their comparative results are often not comprehensive, capturing only partial differences. To address this shortcoming, we propose an enhanced prompt generation framework for open-source models. The enhanced framework is primarily used to improve the descriptions of image differences, while the overall framework remains similar to the vanilla version.

In this framework, we propose only using VLM to generate description for one image at a time. In order to provide a comprehensive description of the image, we divide the description of the image into two parts: content and style. Content focuses on capturing information such as the objects, characters, events, backgrounds, and actions, etc. depicted the image,

while style concentrates on aspects such as artistic style, colour, media, and lighting, etc. The description process is

$$D = \texttt{LLM}(I, \texttt{template}) \tag{B.1}$$

where `template` indicates the template or instruction used by the VLM to descibe image. VLM uses different templates to generate descriptions for content and style. The templates for content descriptions and style descriptions are shown in Figure B.5 and Figure B.6. An example of content descriptions and style descriptions generated by VILA is shown in Figure B.7.

The VLM will generate separate descriptions for content and style for both the reference image and the generated image respectively. Following this, a LLM uses these descriptions to generate difference descriptions. It compares the content description of the reference image D_r with that of the generated image D_g^t, and similarly for the style descriptions,

$$P_{\text{diff}}^t = \texttt{LLM}(D_g^t, D_r, \texttt{template}) \tag{B.2}$$

where `template` indicates the template or instruction used by LLM to generate image differences. The details of this template is shown in Figure B.8. An example of content differences generated by LLaMA2 is shown in Figure B.9. Subsequent steps are the same as for the vanilla framework, utilizing LLM to generate candidate prompts.

Template used by the VLM to generate content description.

You are an expert in describing image, please describe the content of the image. This includes indentifying objects, environments, events, background, actions, etc. in the image.

FIGURE B.5. Template used by the VLM to **generate content description**.

B2.3 Trade-offs between Vanilla and Enhanced Framework

Compared to the vanilla framework, the enhanced framework can more effectively harness the potential of VLMs, yielding better candidate prompts. However, we apply the enhanced framework exclusively to open-source models, while continuing to use the vanilla framework

Template used by the VLM to generate style description.

You are an expert in image analysis, please describe the style of the image. This includes identifying the medium of the image, the art style, the artist's style, the creative technique, the lighting, the colours and the resolution, etc.

FIGURE B.6. Template used by the VLM to **generate style description**.

An example of content and style descriptions.

Content Descriptions
The image portrays a black cat, adorned with a blue bow tie, standing against a gray background. The cat's eyes are a striking shade of blue, adding a touch of whimsy to the scene. The bow tie, a symbol of elegance and formality, contrasts beautifully with the cat's black fur. The gray background serves to highlight the cat and its accessory, making them the focal point of the image. The cat's position and the direction of its gaze suggest it is looking directly at the viewer, creating a sense of connection between the viewer and the subject of the image. The overall composition of the image is balanced and harmonious, with each element - the cat, the bow tie, and the background - contributing to a visually pleasing scene.

Style Descriptions
The image portrays a black cat with striking blue eyes, adorned with a blue bow tie. The cat is positioned in the center of the image, facing the viewer directly. The background is a uniform gray, which serves to highlight the cat and its vibrant attire. The medium of the image is digital art, as evidenced by the crisp lines and smooth gradients. The art style is realistic, with attention to detail in the cat's features and the intricate design of the bow tie. The creative technique used is digital painting, which allows for precise control over the color palette and the overall composition of the image. The lighting is even, with no shadows or highlights, further emphasizing the cat as the focal point of the image. The colors are vivid and saturated, particularly the blue of the cat's eyes and the bow tie, which stand out against the gray background. The resolution of the image is high, allowing for a clear and detailed view of the cat and its surroundings. Overall, the image is a beautiful representation of a black cat, captured in a striking and unique pose

FIGURE B.7. An example of **content and style descriptions** generated by VILA.

for GPT4-V and GPT-4. This decision is based on some disadvantages associated with the enhanced framework. Specifically, the enhanced framework not only adds an image

Template used by the LLM to generate image differences.

I have descriptions of Image 1 and Image 2.
The descriptions of Image 1: {image1}
The descriptions of Image 2: {image2}
Please identify the differences between Image 1 and Image 2 based on their descriptions.
Let's think step by step.

FIGURE B.8. Template used by the LLM to **generate image differences** based on the image description.

An example of content difference.

1. Color scheme: Image 1 features a predominantly gray and blue color scheme, while Image 2 has a more muted color scheme with a focus on black, white, and green.
2. Subject: Image 1 features a black cat with a blue bow tie, while Image 2 features a black and white cat with striking green eyes.
3. Accessories: Image 1 features a blue bow tie, while Image 2 does not have any discernible accessories.
4. Background: Image 1 has a gray background, while Image 2 has a plain gray background.
5. Composition: Image 1 has a more balanced composition, with the cat, bow tie, and background all contributing to a visually pleasing scene. Image 2 has a more centralized composition, with the cat positioned in the center of the frame and the background fading into the distance.
6. Focal point: Image 1 features the blue bow tie as a focal point, while Image 2 features the cat's green eyes as the focal point.
7. Mood: Image 1 has a more playful and whimsical mood, while Image 2 has a more curious and inquisitive mood.
8. Context: Image 1 provides more context, with the background and accessories giving a sense of setting and purpose. Image 2 does not provide any additional context or information about the cat's surroundings or actions.

FIGURE B.9. An example of **content difference** generated by LLaMA2.

description process but also divides the tasks into content and style categories. This design significantly increases the number of tokens input and output to VLMs and LLMs. For the OPENAI API, which charges based on the number of input and output tokens, using the enhanced framework increases the cost to 4 USD, approximately four times the expense

of using the vanilla framework. Additionally, the inference time is substantially extended, requiring more than four times the duration compared to the vanilla framework.

Considering the excellent performance of GPT4-V, the benefits of using the enhanced framework are not as significant as with open-source models. Thus, we balanced performance against costs. We introduce two ARPO version: a closed-source version based on closed-source models using the vanilla framework, which offers faster processing speed but incurs API costs, and a open-source version using the enhanced framework with open-source models, which requires more processing time but is completely free. The performance of these two versions is similar, allowing users to trade-off time and cost to choose the suitable method for their needs.

B3 Experiments

B3.1 Experiments of Hand-crafted, Commercial Services and Closed-source ARPO

We conduct quantitative experiments of human hand-crafted prompts and commercial services (ImageToPromptAI and Phot.AI) and closed-source (GPT4-V and GPT4) ARPO method. Considering the extensive manpower required of human hand-crafted prompts and the expensive cost of the commercial services and OPENAI API, we randomly select a subset of 40 images for experiments. For a fair comparison, we recalculate the results for the two best performing baseline methods, GPT4-V and CLIP-Interrogator, and the best performing open-source IPRE with LLaVA-Next and LLaMA2, on this subset. The experimental results are presented in Table B.1.

Experimental results show that hand-created reverse prompts struggle to accurately reproduce the reference image. This highlights the challenges of manually designing reverse prompts and further emphasizes the importance of our method. The experimental results of commercial services shows that our ARPO method can achieve better performance than these commercial services, even though these commercial services are expensive. For the experimental results indicate that using closed-source models with vanilla prompt generation can achieve results similar to those using open-source models with enhanced prompt generation. Closed-source ARPO offers faster generation speeds, while open-source ARPO is completely free. Users are able to trade-off time and cost to choose the suitable method for their needs.

B3.2 The Results of Each Iteration

We present an example showcasing the reverse prompts and corresponding generated image during the ARPO iterative process. Figure B.10 illustrates the reverse prompt generated at each step of all iterations, alongside the images produced using these prompts. We highlight the newly added prompts in each iteration by bolding them. We observe that the new reverse prompts added during each iteration accurately address the discrepancies observed in the

TABLE B.1. **Results of the image and prompt fidelity comparison on a subset.** Considering manpower resources and cost expenditures, we validate the performance of hand-crafted reverse prompt, commercial services, and closed-source ARPO methods on a 40 images subset.

Method	CLIP-T	CLIP-I	DINO	ViT
Hand-crafted	25.07	72.88±0.04	40.69±0.84	37.10±0.06
GPT4-V	28.83	78.51±0.19	50.33±0.07	47.20±0.26
CLIP-Interrogator	30.49	79.97±0.03	49.34±0.96	46.47±0.79
ImageToPromptAI	29.20	78.34±0.16	48.92±0.15	45.54±0.17
Phot.AI	30.68	80.04±0.23	50.71±0.36	46.89±0.27
Open-source ARPO	35.72	**83.45±0.04**	**56.36±0.65**	52.64±0.82
Closed-source ARPO	**36.01**	83.04±0.08	55.79±0.41	**52.70±0.04**

previously generated image. For example, in Iteration 3, the generated image features a smile dog, which does not match the atmosphere of the reference image. To rectify this, "serious expression" is added to the reverse prompt in Iteration 4, which helps align the image more closely with the intended mood of the reference image. Furthermore, the generated image in Iteration 4 appears too static, "sense of speed" is added in the subsequent iteration to infuse dynamism into the image, thereby moving it closer to the aesthetics of the reference image.

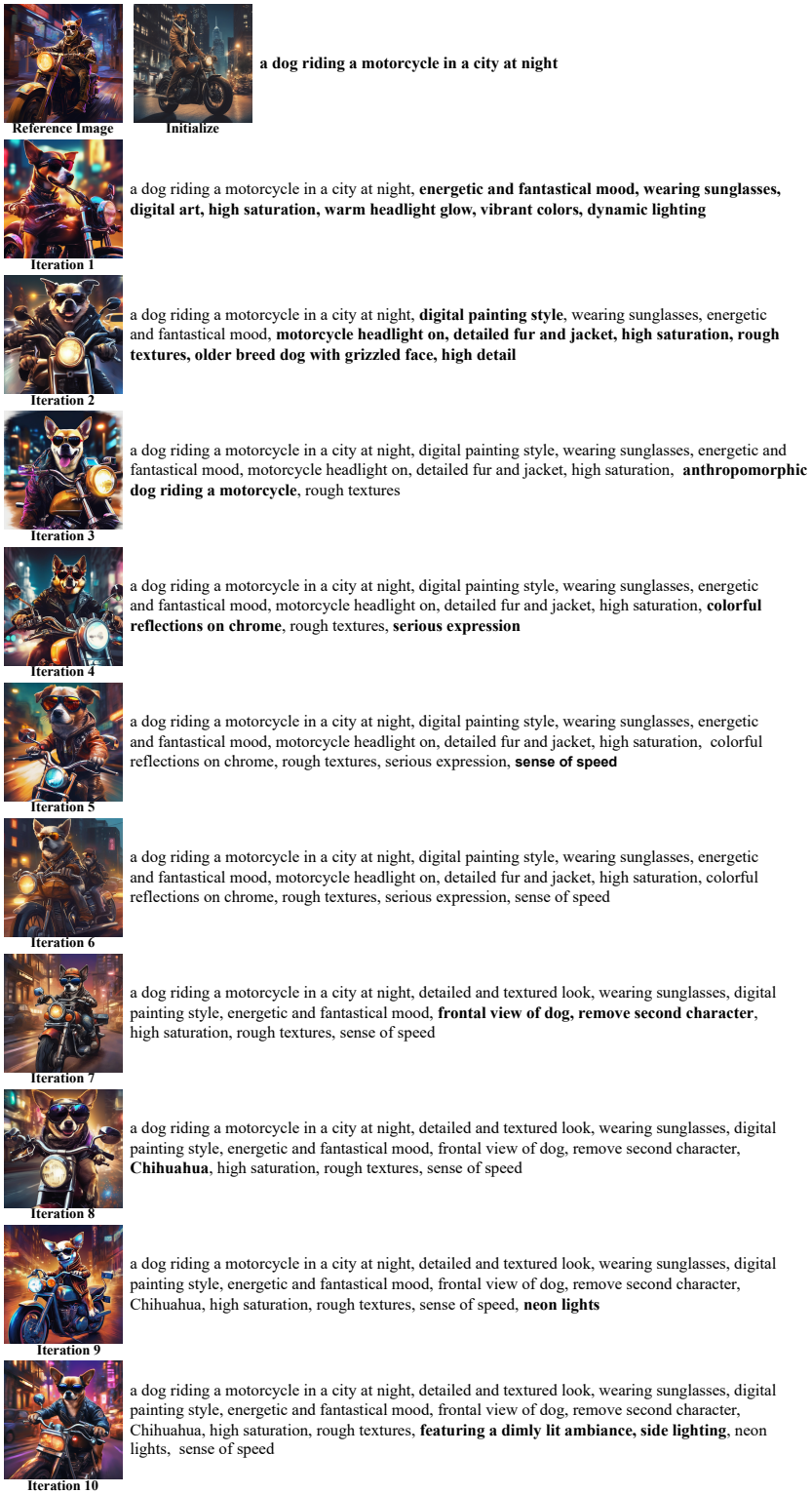


FIGURE B.10. **The reverse prompt and generated image** during the ARPO iterative process. We highlight the newly added prompts in each iteration by **bolding** them.

B3.3 More Results of ARPO

We present additional examples demonstrating the capabilities of our ARPO method in reverse-engineering image prompts. Figure B.11 displays the reference image used for prompt reverse engineering, the reverse prompt generated by out ARPO method, and the image generated by the reverse prompt.

FIGURE B.11. **More results of ARPO**.

B3.4 More Results of Qualitative Comparisons

We provide more qualitative comparison results in Figure B.12. The reference images in the first four rows are nature images and art paintings and the reference images in the last four rows come from AI-generated images and paintings.

FIGURE B.12. **Qualitative comparisons** of recreated images using the reverse prompts from different methods.

B3.5 More Results of Novel Image Generation

We provide additional examples of new images generated by editing reverse prompts produced through the ARPO method. To facilitate user editing, our method additionally provides categorization of reverse prompts into content and style. We edit the *italicized* reverse prompt with the **bolded** words under the images to generated novel images.

FIGURE B.13. **More results of Novel Image Generation**. We replace the *italicized* reverse prompt with the **bolded** words under the images to generated novel images.

B3.6 Comparison of Utilizing Reverse Prompts for Novel Creations

We compare the novel images generated using edited reverse prompts from our ARPO method and several baseline methods, including the hand-crafted method, GPT4-V, CLIP-Interrogator, and commercial service (Phot.AI). The same edits are applied to reverse prompts generated by these different methods, and the edited reverse prompts are then used to generate novel images. The comparison results are shown in Figure B.14. The results show that due to our ARPO mehtod generates superior reverse prompt compared to other methods, the novel images generated by editing reverse prompt can more accurately retain significant elements of content and style from the reference image.

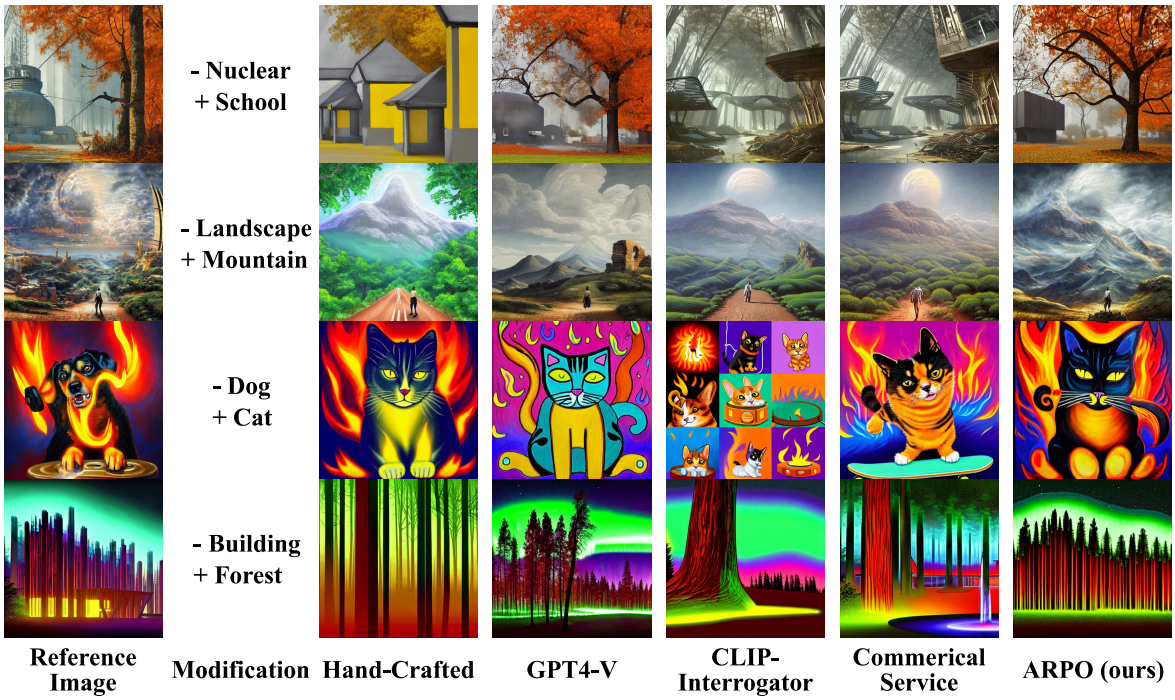


FIGURE B.14. **Comparison of editing reverse prompts for novel creations**. For modification, "- Dog + Cat" indicates that we replace all instances of the word "dog" in reverse prompt with the word "cat".

B3.7 The Setting of User Study

We conduct a user study to compare the image fidelity of our open-source and closed-source ARPO methods with hand-crafted reverse prompts, GPT4-V, CLIP-Interrogator, and

commercial service (Phot.AI). We choose 40 cases for comparison. We ask users to rank the similarity between the generated image and the reference image based on content preservation, style approximation, and overall preference. Content Preference focuses on comparing the similarity of the image in aspects such as objects, characters, events, background, and actions. Style Preference focuses on comparing the similarity of the image in aspects such as artistic style, color, medium, and lighting. Overall Preference is a comprehensive comparison.

We collect feedback from 50 users. Considering portability and labor costs, we conduct this user study in China. We offer 4 CNY (about 0.55 USD) to each participant. Screenshot of the questionnaire and its English translation are shown in Figure B.15 and Figure B.16.

FIGURE B.15. **The screenshot** of the questionnaire for user study

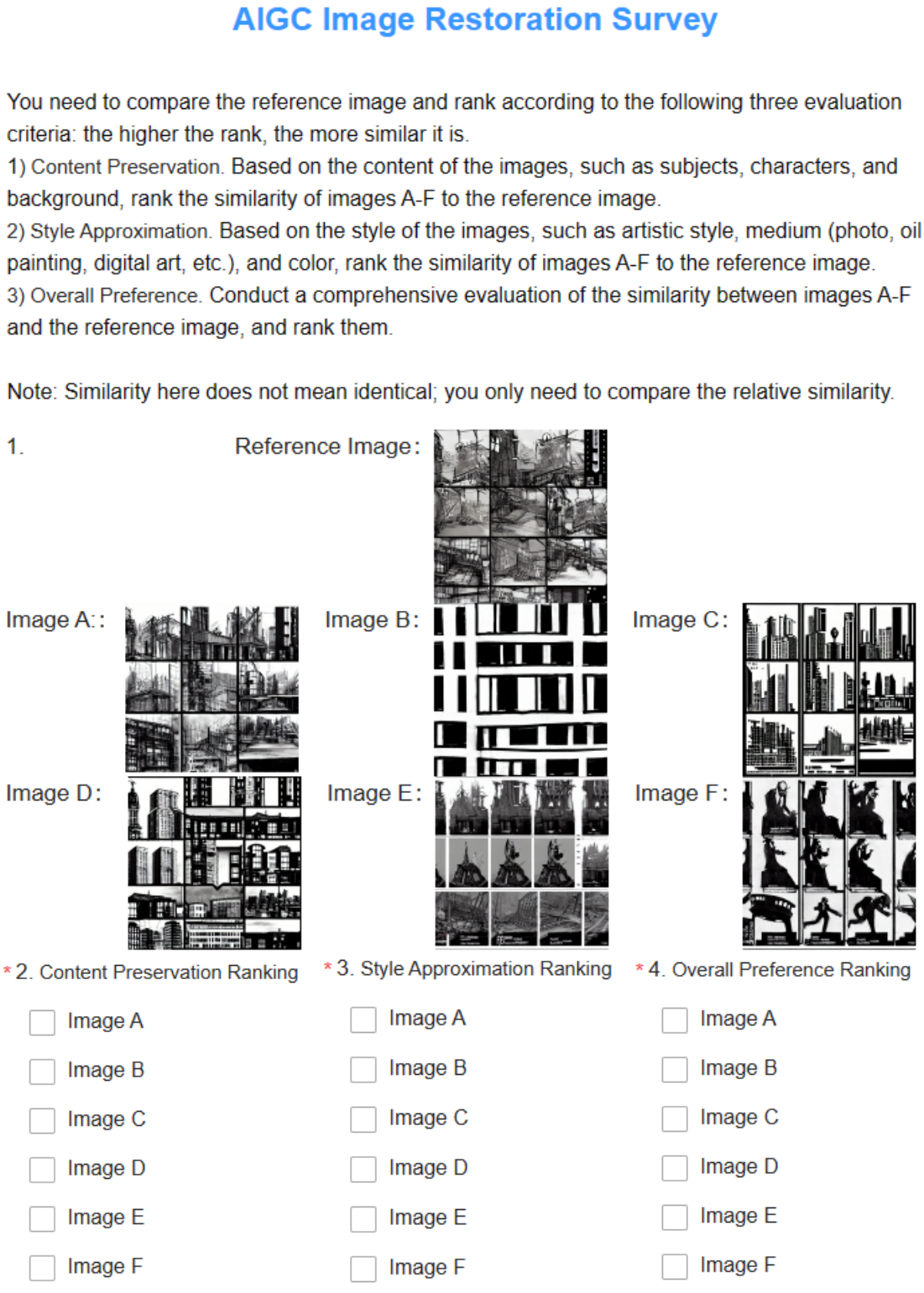


FIGURE B.16. **Translation of the screenshot** of the questionnaire for user study