This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

NoiseCtrl: A Sampling-Algorithm-Agnostic Conditional Generation Method for Diffusion Models

Longquan Dai, He Wang, Jinhui Tang *
Nanjing University of Science and Technology
{dailongquan, wanghe, jinhuitang}@njust.edu.cn

Abstract

In training-free conditional generative tasks, diffusion models utilize differentiable loss functions to steer the generative reverse process, necessitating modifications to sampling algorithms like DDPM and DDIM. However, such adjustments likely reduce flexibility and reliability. In this paper, we propose NoiseCtrl, a sampling-algorithm-agnostic technique for controlled image generation. Essentially, diffusion models generate denoised results $\mathbf{z}_{t-1}$ by adding a predicted mean $\boldsymbol{\mu}_t$ with random noise $\boldsymbol{\epsilon}_t$. NoiseCtrl specifically adjusts the random noise while leaving the underlying sampling algorithms unchanged. At each step t, NoiseCtrl converts the unconditional Gaussian noise into conditional noise $\boldsymbol{\epsilon}'_t$ by substituting the isotropic Gaussian distribution with the von Mises–Fisher distribution. This substitution introduces a directional focus while preserving the randomness required for conditional image generation. Thanks to this non-intrusive design, NoiseCtrl is straightforward to integrate and has been extensively validated through experiments, demonstrating its adaptability for different diffusion algorithms and superior performance across various conditional generation tasks.

1. Introduction

In recent years, conditional diffusion models (CDMs) have achieved notable success. Their impact is particularly visible across a range of low-level vision tasks [6]. This includes advancements in image generation [36], inpainting [17], virtual dressing [23] and person synthesis[21, 22]. The achievements, from foundational works [9, 24, 27, 28] to cutting-edge models [4, 19, 35], highlight their versatility and effectiveness. Traditional conditional diffusion models, while effective, face challenges due to their reliance on training. These models often require additional training data and computational resources, which can be costly and limit their broader application.

Dhariwal et al. [4] pioneered the use of pre-trained diffusion models for conditional generative task by introducing $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{y}) = \nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t) + \nabla_{\mathbf{z}_t} \log p_t(\mathbf{y}|\mathbf{z}_t)$, In this framework, given condition $\mathbf{y}$ and latent $\mathbf{z}_t$ at the time step t, the pre-trained diffusion prior $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t)$ serves as a foundation for the generative process and the gradient of the time-dependent classifier $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{y}|\mathbf{z}_t)$ helps steer the generation towards the desired condition, then the condition-matching score $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{y})$ aligns the generated content with the condition. Building on this foundation, recent works [1, 2, 32, 34, 39], often termed as training-free methods, eliminate the need for re-training or fine-tuning the diffusion prior and have skillfully utilized differentiable loss functions to approximate the conditional probability $p_t(\mathbf{y}|\mathbf{z}_t)$. However, the approximation strategy can disrupt the stability of the sampling method and requires extensive experimentation to determine optimal settings. This interdependence between approximation and sampling highlights their complex relationship and the need for a more robust solution.

The sampling algorithm of diffusion models consists of two primary elements: an unconditional latent $\boldsymbol{\mu}_{t+1}$, which satisfies $\nabla_{\boldsymbol{\mu}_{t+1}} p_t(\boldsymbol{\mu}_{t+1}) = 0$ to maximize the distribution $p_t(\mathbf{z}_t)$, and a Gaussian noise $\boldsymbol{\epsilon}_{t+1}$. The output at step t is represented as $\mathbf{z}_t = \boldsymbol{\mu}_{t+1} + \sigma_{t+1}\boldsymbol{\epsilon}_{t+1}$, where σ_{t+1} be a constant. To achieve conditional generation, existing methods replace the unconditional distribution $p_t(\mathbf{z}_t)$ with the conditional distribution $p_t(\mathbf{z}_t|\mathbf{y})$. We diverge from this approach by proposing a two-stage noise sampling process. First, we uniformly sample a condition. Then, we generate output samples based on a distribution specific to that condition. In this two-step context, we argue that $\mathbf{z}_t = \boldsymbol{\mu}_{t+1} + \sigma_{t+1}\boldsymbol{\epsilon}_{t+1}$ remains unconditional because the sampling of $\boldsymbol{\epsilon}_{t+1}$ is not biased towards any specific condition. With a deterministic condition established in the first stage, devoid of randomness, the noise in the second stage becomes conditional, introducing a directional bias yet preserving its stochastic properties. In this paper, we disclose that NoiseCtrl, combining $\boldsymbol{\mu}_{t+1}$ with this conditional noise, will produce a result result is attentive to the given condition.

*Corresponding author.

Our approach distinguishes itself from prior methods by eliminating the need to approximate $p_t(\mathbf{z}_t|\mathbf{y})$. Instead, we focus solely on adapting the random noise, with modifications that are neutral to the specific sampling algorithm employed. Our contributions are as follows:

- We introduce a method that is the first to achieve training-free conditional generation by pioneering the transformation of isotropic Gaussian noise into conditional noise.
- We introduce a pioneering sampling-algorithm-agnostic method, which can be effortlessly integrated with any sampling approach for conditional generative task.
- Our method demonstrates the adaptability and superior performance of our method across a spectrum of conditional generation tasks.

2. Related Work

In this section, we review the current literature on conditional diffusion models (CDMs) and backward processes.

Training-required CDMs mainly fall into two categories. The first includes classifier-guided diffusion models like those introduced by Dhariwal and Nichol [4], which use a pre-trained diffusion model to train a time-dependent classifier to approximate the posterior probability $\log p_t(\mathbf{y}|\mathbf{z}_t)$. The second category features classifier-free models such as ControlNet [35] and its variations [11, 12, 15, 16, 32, 38]. These models bypass classifier training and directly train a conditional denoiser with paired data. While they ensure high-fidelity sample generation, they require significant training time.

Training-free CDMs employ existing classifiers to approximate the gradient $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{y}|\mathbf{z}_t)$. Chung et al. [2] utilized Tweedie's formula to address linear inverse problems, a strategy later extended to broader conditional generation scenarios by Chung et al. [3], Yu et al. [34], Bansal et al. [1] and Wang et al. [30]. Song et al. [26] attempted to reduce bias through multiple samples from an imprecise Gaussian distribution but encountered the computational burden of Monte Carlo simulations. Zhu et al. [39] applied guidance to a data sample $\mathbf{z}_0$ and projected it onto the intermediate data manifold $\mathbf{z}_t$. He et al. [7] refined this approach by incorporating an autoencoder to enforce guidance within the tangent space of the clean data manifold. Yang et al. [32] adeptly managed manifold deviations without imposing stringent assumptions, thereby enhancing both the quality of the samples and the efficiency of the process.

Sampling algorithms in diffusion models create images from Gaussian noise. Ho et al. [9] introduced the widely adopted DDPM sampling algorithm. To speed up DDPM's sampling process, Song et al. [25] proposed the DDIM algorithm. Since then, various methods, including CM [29], LCM [14], DPM [13], Euler [10], and SA [31], have been developed to achieve final results in only a few steps.

3. The Intuition Behind Noisy Control

In each sampling step of the diffusion model, an additional random noise $\boldsymbol{\epsilon}_t$, sampled from the Gaussian distribution, is introduced. This process can be confirmed by examining Equation (1), which illustrates the DDPM sampling algorithm [9]. Other sampling algorithms are similar.

$$\begin{aligned} \boldsymbol{\mu}_t &= \frac{1}{\sqrt{\alpha_t}}\mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) \\ \mathbf{z}_{t-1} &= \boldsymbol{\mu}_t + \sigma_t \underbrace{\boldsymbol{\epsilon}_t}_{\text{Gaussian Noise}} \end{aligned} \tag{1}$$

Here $\sigma_t^2 = \frac{(1-\alpha_t)(1-\sqrt{\bar{\alpha}_{t-1}})}{1-\bar{\alpha}_t}$, $\bar{\alpha}_t = \prod_{i=0}^{t} \alpha_i$, α_0, …, α_T represent the noise schedule and $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$ is the denoiser trained to estimate the added isotropic Gaussian noise $\boldsymbol{\epsilon}_t$ at each step. In this section, we introduce the intuition behind controlling the direction of isotropic Gaussian to enable conditional generation in diffusion models.

3.1. Gaussian Distribution in Polar Coordinates

Let $\mathbf{X} = (\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_n)$ be a n-dimensional Gaussian random vector, $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. The transformation from Cartesian coordinates to polar coordinates involves:
Radius r: The distance from the origin is calculated as

$$\mathrm{r} = \sqrt{\mathrm{X}_1^2 + \mathrm{X}_2^2 + \cdots + \mathrm{X}_n^2}. \tag{2}$$

Angles $\boldsymbol{\psi}$: The angles $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_{n-2}, \phi)$, describing the direction of the vector, are defined as:

$$\begin{aligned} \psi_k &= \arccos\left(\frac{\mathrm{X}_i}{\mathrm{r}\prod_{j=1}^{k-1}\sin(\psi_j)}\right) \\ \phi &= \arctan(\mathrm{X}_n, \mathrm{X}_{n-1}) \end{aligned}$$

where $0 \le \psi_k < \pi$, $0 \le \phi < 2\pi$ and $1 \le k < n-1$. Then we have the following properties:
1. Radius Distribution: The radius r follows the chi distribution (3), where $\Gamma(\cdot)$ denote the gamma function.

$$p_{\mathrm{r}}(\mathrm{r}) = \frac{\mathrm{r}^{n-1}\exp(-\frac{\mathrm{r}^2}{2})}{2^{\frac{n}{2}-1}\Gamma(\frac{n}{2})}, \quad \mathrm{r} \ge 0 \tag{3}$$

2. Angular Distribution: The angular coordinates ψ_1, $\ldots, \psi_{n-2}$, ϕ are uniformly distributed on the surface of a unit hypersphere in n-dimensional space. Let A_{n-1} be the surface area of the unit hypersphere in $n-1$ dimensions. The probability density function of the angles is given by:

$$p_{\boldsymbol{\psi}}(\boldsymbol{\psi}) = \frac{1}{A_{n-1}}, \tag{4}$$

Joint Distribution: The joint distribution of the radius and angles in polar coordinates is the product of the distributions of r and $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_{n-2}, \phi)$:

$$p(\mathrm{r}, \boldsymbol{\psi}) = f_r(r) \cdot p_{\boldsymbol{\psi}}(\boldsymbol{\psi}) \tag{5}$$

3.2. Piecewise Gaussian Distribution

The Gaussian distribution covers the entire $\mathbb{R}^n$ space. By partitioning $\mathbb{R}^n$ into separate regions, we can form a piecewise Gaussian distribution, which allows for different statistical properties in each segment. A simple method for this partitioning is by dividing the space using angles in polar coordinates. For each angular coordinate, we can set a series of midpoints that define the central angle of each segment. For instance, if ψ_1 ranges from 0 to π, we can define p_1 midpoints $\psi_{1,m} = \left(\frac{2m-1}{2p_1}\right)\pi$ with $m = 1, 2, \ldots, p_1$.

These midpoints divide ψ_1 into p_1 segments. Each midpoint $\psi_{1,m}$ serves as the center of a segment $\Delta_{1,m} = \left[\psi_{1,m} - \frac{\pi}{2p_1-1}, \psi_{1,m} + \frac{\pi}{2p_1-1}\right)$. Similarly, the remaining angles $\psi_2, \ldots, \psi_{n-2}, \phi$ are divided into intervals centered around their respective midpoints.

By combining the intervals of each angular coordinate, we create an angular set $\mathbf{\Delta_d} = \Delta_{1,i} \times \Delta_{1,j} \times \cdots \times \Delta_{n-1,k}$, which defines a segment in polar coordinates. The midpoints of these intervals determine a unique orientation $\mathbf{d}$ for each segment $\mathbf{\Delta_d}$. With a total of N regions, we define N direction vectors, each representing the orientation of a segment. For a segment associated with the direction vector $\mathbf{d}$, the piecewise Gaussian distribution is defined as

$$p_{\mathrm{r},\boldsymbol{\psi};\mathbf{d}}(\mathrm{r}, \boldsymbol{\psi}|\mathbf{d}) = \frac{A_{n-1}}{N} f_r(r) \cdot p_{\boldsymbol{\psi}}(\boldsymbol{\psi}) \quad \boldsymbol{\psi} \in \mathbf{\Delta_d} \tag{6}$$

3.3. Conditional Noise

When the direction vector $\mathbf{d}$ serves as a conditioning factor, the piecewise Gaussian distribution $p(\mathrm{r}, \boldsymbol{\psi}|\mathbf{d})$ effectively acts as a conditional distribution. By assuming that the distribution $p_{\mathbf{d}}(\mathbf{d})$ of $\mathbf{d}$ is uniform, we can derive Equation (7) which shows how the Gaussian distribution $p(\mathrm{r}, \boldsymbol{\psi})$ is represented as a weighted sum of the piecewise Gaussian distributions $p(\mathrm{r}, \boldsymbol{\psi}|\mathbf{d})$ across each segment's orientation $\mathbf{d}$.

$$p(\mathrm{r}, \boldsymbol{\psi}) = \int_{\mathbf{\Delta}} p_{\mathrm{r},\boldsymbol{\psi};\mathbf{d}}(\mathrm{r}, \boldsymbol{\psi}|\mathbf{d}) p_{\mathbf{d}}(\mathbf{d}) d\mathbf{d} \tag{7}$$

This equation elucidates that when the noise term $\boldsymbol{\epsilon}_t$ is Gaussian-distributed, the outcome is unconditional, meaning it does not favor any particular direction. However, by introducing noise with a directional bias $\mathbf{d}$ aligned with the piecewise Gaussian distribution $p(\mathrm{r}, \boldsymbol{\psi}|\mathbf{d})$, the final result will align with this directional preference, enabling the outcome to be influenced by the specified orientation.

Once a conditional direction $\mathbf{d}$ is specified, the conditional noise can be generated in two stages: First, sample a random condition direction $\mathbf{u}$ uniformly from the angular set $\mathbf{\Delta_d}$; second, sample a random scale s according to the chi distribution. The final result is then $\mathrm{s}\mathbf{u}$. While the first step is conceptually straightforward, it can be computationally intensive, and there is no existing library that implements it efficiently. To address this, we propose using the von Mises-Fisher distribution (8) as an effective approximation for the original distribution, where κ is the concentration parameter with $\kappa \geq 0$, $\mathbf{d}$ is the mean direction (a unit vector), $\mathbf{u}$ is any point on the unit sphere in $\mathbb{R}^n$ and $I_{\frac{n}{2}-1}(\kappa)$ is the modified Bessel function of the first kind of order $\frac{n}{2} - 1$.

$$p_{\mathbf{u};\mathbf{d},\kappa}(\mathbf{u}; \mathbf{d}, \kappa) = \frac{\kappa^{\frac{n}{2}-1}}{(2\pi)^{\frac{n}{2}} I_{\frac{n}{2}-1}(\kappa)} \exp(\kappa \mathbf{d}^T \boldsymbol{u}) \tag{8}$$

This substitution is justified because the von Mises-Fisher distribution effectively models angular data within a specified interval. It captures the clustering of data points around a mean direction on the surface of a sphere, characterized by a concentration parameter that reflects the tightness of this clustering. Importantly, many Python libraries offer efficient sampling algorithms for this distribution, making it a practical choice for our needs.

4. NoiseCtrl

Gaussian noise, being all-encompassing, means that reverse processes like DDPM (1) are directionally unbiased. However, by replacing the isotropic Gaussian noise with conditional noise drawn from the piecewise Gaussian distribution $f(\mathrm{r}, \boldsymbol{\psi}|\mathbf{d})$, the generated output becomes influenced by the specified condition. One critical challenge in this conditional generation lies in estimating the direction vector $\mathbf{d}$ accurately and efficiently.

4.1. Direction Vector Estimation (Slow)

In n-dimensional space, the transformation from polar to Cartesian coordinates, is given by formulas that express each Cartesian coordinate as a function of r and $\boldsymbol{\psi}$. This transformation can be recursively formulated as:

$$\begin{aligned} X_k &= r\cos(\psi_k)\prod_{i=1}^{k-1}\sin(\psi_i), \quad k = 1, 2, \ldots, n-1 \\ X_n &= r\sin(\phi)\prod_{i=1}^{n-2}\sin(\psi_i) \end{aligned} \tag{9}$$

We define this transformation as $\mathbf{X} = \boldsymbol{T}(\mathrm{r}, \boldsymbol{\psi})$. Accoridng to section 3.1, we reformulate Equation (1) as follows:

$$\mathbf{z}_t = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_{t+1}, t+1) + \boldsymbol{\sigma}_{t+1}\boldsymbol{T}(\mathrm{r}_{t+1}, \boldsymbol{\psi}_{t+1}) \tag{10}$$

where r_{t+1} follows a chi distribution and $\boldsymbol{\psi}_{t+1}$ are uniformly distributed across the surface of a unit hypersphere.

To achieve controlled image generation, we anticipate that $\mathbf{z}_t$ in Equation (10) maximizes $\log p(\mathbf{y}|\mathbf{z}_t)$. However, modeling this probability remains open. One approach is to employ the energy function [37] defined as follows:

$$p(\mathbf{y}|\mathbf{z}_t) = \frac{\exp\left(-\lambda \mathcal{E}(\hat{\boldsymbol{z}}_{0|t}(\mathbf{z}_t), \mathbf{y})\right)}{Z} \tag{11}$$

where λ denotes the temperature coefficient. Z is a normalizing constant, $\hat{\boldsymbol{z}}_{0|t}(\mathbf{z}_t) = \frac{(\mathbf{z}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t, t))}{\sqrt{\alpha_t}}$ estimates $\mathbf{z}_0$ according to $\mathbf{z}_t$ and $\mathcal{E}(\cdot,\cdot)$ is the energy function used to measure the closeness of two vectors. Due to Equation (11), maximizing $p(\mathbf{y}|\mathbf{z}_t)$ is equivalent to minimizing the the energy function $\mathcal{E}(\hat{\boldsymbol{z}}_{0|t}(\mathbf{z}_t), \mathbf{y})$. Given $\mathbf{y}$, we establish a constraint based on $\mathbf{y}$ and $\mathcal{E}(\cdot,\cdot)$. This constraint is formalized as $\mathcal{E}(\hat{\boldsymbol{z}}_{0|t}(\mathbf{z}_t), \mathbf{y}) \approx 0$. By substituting Equation (10) into $\mathcal{E}(\cdot,\cdot)$ and letting $\boldsymbol{z}_{0|t}(\boldsymbol{\psi}) = \hat{\boldsymbol{z}}_{0|t}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_{t+1}, t+1) + \boldsymbol{\sigma}_{t+1}\boldsymbol{T}(\mathrm{r}_{t+1}, \boldsymbol{\psi}))$, we derive a loss function $\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi})$.

$$\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi}) = \mathcal{E}(\boldsymbol{z}_{0|t}(\boldsymbol{\psi}), \boldsymbol{y}) \tag{12}$$

The optimal angles $\boldsymbol{\psi}^*$ can be obtained by solving

$$\begin{aligned} &\arg\min_{\boldsymbol{\psi}} \mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi}) \\ &\text{s.t. } \boldsymbol{\psi} \in \mathbb{C} \end{aligned} \tag{13}$$

where $\mathbb{C} = \{\psi_i, \phi | 0 \le \psi_i < \pi,\ 0 \le \phi < 2\pi,\ 1 \le i < n-1\}$. The solution provides the necessary angles and further defines the direction vector $\mathbf{d}$ according to the optimal angles $\boldsymbol{\psi}^*$ that steer the image generation process.

4.2. Fast Estimation in Polar Coordinates

The nonlinear loss function $\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi})$ demands numerous iterations to find its minima, rendering the backward process time-consuming. To surmount this obstacle, we propose a swift sampling technique that employs a Taylor expansion around a randomly chosen point $\boldsymbol{\psi}' \in \mathbb{C}$ to approximate the loss function. By using $\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi}) \approx \mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi}') + \left[\nabla_{\boldsymbol{\psi}'}\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi}')\right]^T(\boldsymbol{\psi} - \boldsymbol{\psi}')$, optimization (13) simplifies to:

$$\begin{aligned} &\arg\min_{\boldsymbol{\psi}} \left[\nabla_{\boldsymbol{\psi}'}\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi}')\right]^T(\boldsymbol{\psi} - \boldsymbol{\psi}') \\ &\text{s.t. } \boldsymbol{\psi} \in \mathbb{C} \end{aligned} \tag{14}$$

This linear optimization achieves its minima at the boundary of $\mathbb{C}$. The analytical expression thus is:

$$\psi_k^* = \begin{cases} 0, & \text{if } \nabla_{\psi_k'}\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi}') \ge 0 \\ \pi - \epsilon, & \text{if } \nabla_{\psi_k'}\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi}') < 0 \end{cases} \tag{15}$$

$$\phi^* = \begin{cases} 0, & \text{if } \nabla_{\phi'}\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi}') \ge 0 \\ 2\pi - \epsilon, & \text{if } \nabla_{\phi'}\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{\psi}') < 0 \end{cases} \tag{16}$$

where ϵ is a very small positive number used to ensure that the variables approach their upper bounds without exceeding them. By leveraging the closed-form solution, we can speed up the determination of the optimal angles $\boldsymbol{\psi}^* = \{\psi_1^*, \cdots, \psi_{n-2}^*, \phi^*\}$. This approach balances the precision needed for accurate image generation with computational efficiency. It ensures that we can generate high-quality images that closely align with the given prompt within a reasonable timeframe, offering an innovative method for efficient sampling in polar coordinates.

Algorithm 1 NoiseCtrl for conditional DDPM sampling

Input: pure noise $\mathbf{z}_T$, iteration number T, concentration parameter κ.
for $t = T$ **to** 1 **do**
  $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) = \frac{1}{\sqrt{\alpha_t}}\mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$
  if Fast Estimation in Cartesian Coordinates **then**
    Determine $\mathbf{d}$ by solving optimization (17)
  else if Fast Estimation in Polar Coordinates **then**
    Determine $\mathbf{d}$ by solving optimization (14)
  else
    Determine $\mathbf{d}$ by solving optimization (13)
  end if
  $\boldsymbol{\epsilon}'_t = \mathrm{r}\mathbf{u}$, where $\mathrm{r} \sim f_{\mathrm{r}}(\mathrm{r}),\ \mathbf{u} \sim f_{\mathbf{u};\mathbf{d},\kappa}(\mathbf{u};\mathbf{d},\kappa)$
  $\mathbf{z}_{t-1} = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) + \sigma_t\boldsymbol{\epsilon}'_t$
end for
Return $\mathbf{z}_0$

4.3. Fast Estimation in Cartesian Coordinates

Substituting the angle vector $\boldsymbol{\psi}$ in polar coordinates with a direction vector $\mathbf{d}$ in cartesian coordinates offers an alternative method for determining the direction. To be precise, let $\mathbb{S}$ represent the sphere surface with radius 1, the linear optimization problem (14) can be transformed to

$$\begin{aligned} &\arg\min_{\mathbf{d}} \left[\nabla_{\mathbf{z}'}\mathcal{L}'_{\boldsymbol{y}}(\mathbf{z}')\right]^T(\|\mathbf{z}'\|_2\mathbf{d} - \mathbf{z}') \\ &\text{s.t. } \mathbf{d} \in \mathbb{S} \end{aligned} \tag{17}$$

where $\mathcal{L}'_{\boldsymbol{y}}(\mathbf{z}') = \mathcal{E}(\hat{\boldsymbol{z}}_{0|t}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_{t+1}, t+1) + \boldsymbol{\sigma}_{t+1}\mathbf{z}'), \boldsymbol{y})$ and $\mathbf{z}'$ is a random Gaussian point. According to [32], the closed-form solution of this manifold optimization problem can be formulated as

$$\mathbf{d} = -\frac{\nabla_{\mathbf{z}'}\mathcal{L}'_{\boldsymbol{y}}(\mathbf{z}')}{||\nabla_{\mathbf{z}'}\mathcal{L}'_{\boldsymbol{y}}(\mathbf{z}')||_2}. \tag{18}$$

4.4. NoiseCtrl Sampling

By substituting the Gaussian noise $\boldsymbol{\epsilon}_t$ with the conditional noise $\boldsymbol{\epsilon}'$, we can transform arbitrary backward processes in the diffusion model from an unconditional to corresponding conditional form. This adjustment allows the model to incorporate additional information or constraints while maintaining original form. For example, by substituting $\boldsymbol{\epsilon}'$ into DDPM (1), we obtain corresponding conditional version:

$$\mathbf{z}_{t-1} = \boldsymbol{\mu}_t + \sigma_t \underbrace{\boldsymbol{\epsilon}'_t}_{\text{Conditional Noise}} \tag{19}$$

where the conditional noise $\boldsymbol{\epsilon}'_t = \mathrm{r}\mathbf{u}$, r and $\mathbf{u}$ are sampled from the joint distribution $f_{\mathrm{r},\mathbf{u};\mathbf{d},\kappa}(\mathrm{r}, \mathbf{u}; \mathbf{d}, \kappa)$

$$f_{\mathrm{r},\mathbf{u};\mathbf{d},\kappa}(\mathrm{r}, \mathbf{u}; \mathbf{d}, \kappa) = f_r(r) \cdot f_{\mathbf{u};\mathbf{d},\kappa}(\mathbf{u}; \mathbf{d}, \kappa) \tag{20}$$

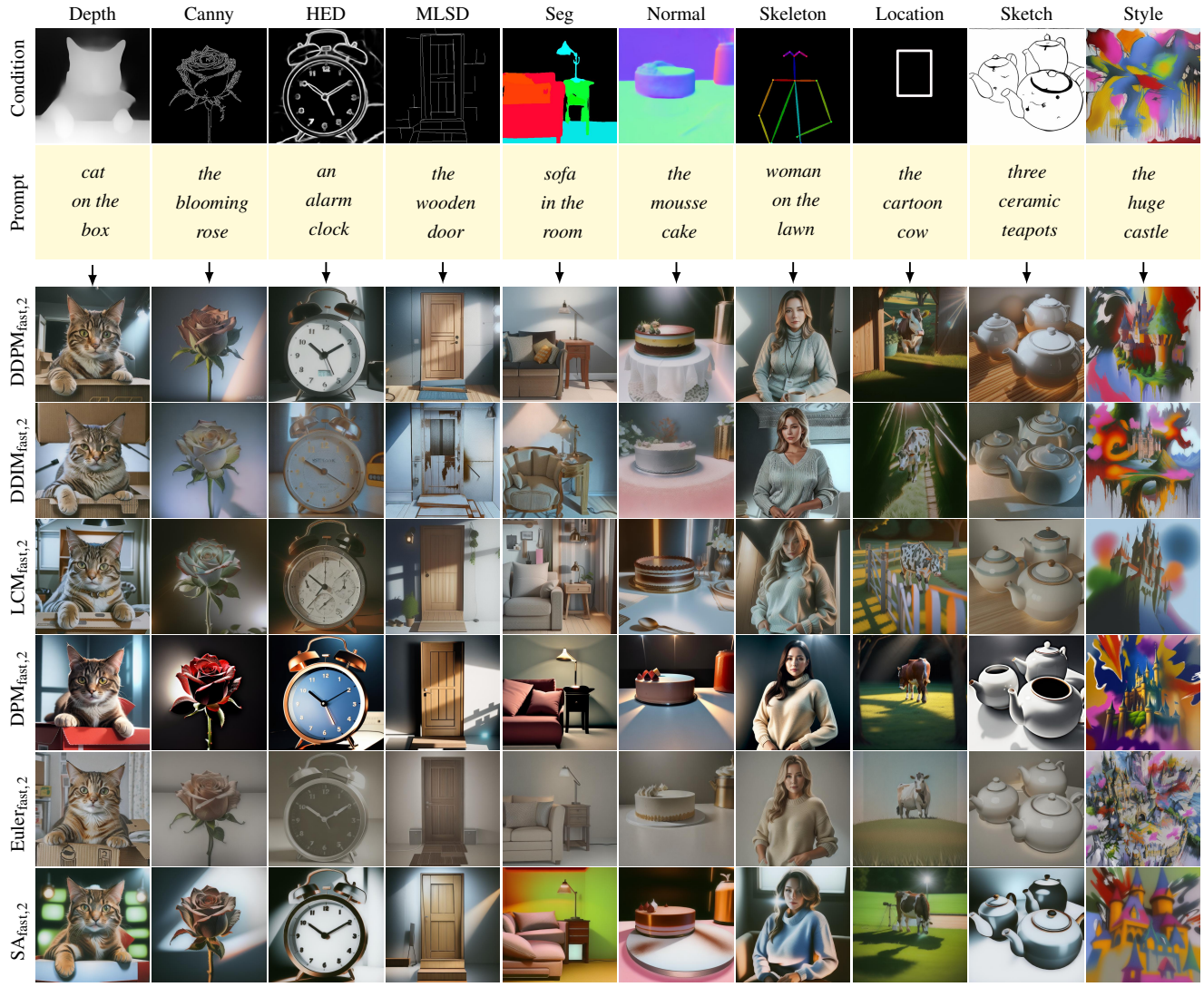

Figure 1. NoiseCtrl Illustration for Conditional Generation Using the Stable Diffusion Model with DDPM [9], DDIM [25], LCM [14], DPM [13], Euler [10], and SA [31] sampling algorithms. NoiseCtrl is designed to be sampling-algorithm-agnostic; It maintains the original sampling algorithm intact but introduces a directional distribution to substitute the isotropic Gaussian distribution for noise sampling. The figure displays the outcomes of these five sampling algorithms under 10 distinct conditions, validating NoiseCtrl's compatibility and effectiveness across different conditional generation tasks with different sampling algorithms.

Without sacrificing generality, we introduce NoiseCtrl for conditional DDPM sampling in Algorithm 1, where the conditional noise computation is indicated within the red box. Other sampling algorithms, such as DDPM [9], DDIM [25], LCM [14], DPM [13], Euler [10], and SA [31], can be adapted by simply replacing the noise component with the noise generated by our method. This noise substitution strategy highlights the adaptability of our approach, which is its most significant contribution. The efficacy and extensive compatibility of NoiseCtrl have been substantiated through a series of experiments detailed in the subsequent sections.

Although both NoiseCtrl and DSG [32] utilize the same closed-form solution (18), there are significant differences between the two methods. First, NoiseCtrl is a sampling-algorithm-agnostic method due to its condition noise generation method, whereas DSG is not. Second, NoiseCtrl incorporates a piecewise Gaussian distribution (6) or its approximation (20) to introduce randomness into the denoising equation, while DSG uses Gaussian noise. Third, while DSG computes the gradient of $\mathcal{L}'_{\boldsymbol{y}}(\mathbf{z})$ at the point $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$, NoiseCtrl calculates this gradient at a random point $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) + \boldsymbol{\sigma}_t \mathbf{z}'$, where $\mathbf{z}'$ is a random point sampled from Gaussian distribution.

	Method	UGD [1]	FreeDom [34]	DSG [32]	$\text{DDPM}_{\text{fast},2}$	$\text{DDIM}_{\text{fast},2}$	$\text{LCM}_{\text{fast},2}$	$\text{DPM}_{\text{fast},2}$	$\text{Euler}_{\text{fast},2}$	$\text{SA}_{\text{fast},2}$
FID↓	Depth	26.1142	27.3437	27.2473	**22.7612**	23.7019	24.8210	23.4144	23.3453	23.7381
	Canny	26.3717	27.1109	26.8372	22.6373	**21.2472**	22.9672	22.1386	21.8631	22.1864
	Style	28.8945	**26.0756**	26.3654	26.3752	26.5217	26.4217	27.1384	27.2373	27.4534
CLIP-scores↑	Depth	0.2654	0.2582	0.2470	0.2943	0.2959	0.2831	0.2894	**0.2961**	0.2942
	Canny	0.2561	0.2485	0.2508	0.3041	0.2997	0.2982	0.2931	0.3001	**0.3083**
	Style	0.3046	0.2961	**0.3449**	0.3081	0.3271	0.3115	0.3181	0.3267	0.3071
CLIP-acs ↑	Depth	5.2341	5.2237	5.2267	5.4224	**5.4254**	5.4115	5.4218	5.4196	5.4203
	Canny	5.2238	5.1216	5.1486	**5.4220**	5.4189	5.4138	5.4209	5.4134	5.4210
	Style	5.4001	4.8576	5.6667	**6.3423**	5.6600	5.6797	5.1745	5.8597	4.5815
Controllability	Depth	95.1675	97.0178	96.6471	86.9427	87.8367	87.9368	**86.3571**	87.2057	86.9538
	Canny	0.3754	0.3573	0.3485	0.5001	0.5024	0.4937	0.4983	0.4823	**0.5073**
	Style	**6.75e-5**	6.93e-5	7.28e-5	6.91e-5	6.85e-5	7.18e-5	7.27e-5	7.32e-5	6.85e-5

Table 1. Quantitative Comparison for Training-free Generation with Depth, Canny and Style Conditions. A total of 9 methods are compared, comprising 3 training-free methods and 6 NoiseCtrl variants. We evaluate the final results using FID [8], CLIP-scores [18], CLIP-acs [20], and MSE and Style loss [5] to measure controllability. The best scores are highlighted in bold.

5. Experiment

In this section, we evaluate our methods through a variety of experiments. We start by detailing the experimental setup. We then demonstrate its seamless integration with various sampling methods, including DDPM [9], DDIM [25], LCM [14], DPM [13], Euler [10], and SA [31], for diverse conditional generation tasks. We also show that NoiseCtrl is compatible with both training-based CMDs like ControlNet [35], T2I-Adapter [15], and IP-Adapter [33], and training-free CMDs such as UGD [1], FreeDom [34], and DSG [32]. We compare our method with previous approaches using both quantitative and qualitative metrics. Lastly, we discuss the impact of the newly introduced parameter κ on the final results.

5.1. Experimental Setting

To configure NoiseCtrl sampling, we need to specify the algorithm for sampling, the method for calculating the direction vector, and the distribution from which the conditional noise is sampled. We thus represent a NoiseCtrl algorithm with the notation $\mathbf{S}_{\mathbf{D},\mathbf{N}}$, where $\mathbf{S}$ signifies the sampling algorithm, such as DDPM [9], DDIM [25], LCM [14], DPM [13], Euler [10], or SA [31]. The subscript $\mathbf{D}$ indicates the method for determining the direction vector, for which we provide both the slow version as described in Section 4.1 and the fast version as detailed in Sections 4.2 and 4.3. The two fast versions are mathematically equivalent, so we do not distinguish between them. The subscript $\mathbf{N}$ specifies the distribution from which the conditional noise is sampled. In this paper, we suggest sampling conditional noise from a specific distribution, as defined in equation (6). Due to the low efficiency of our implementation for the angular distribution $f_{\boldsymbol{\psi}}(\boldsymbol{\psi})$ with $\boldsymbol{\psi} \in \boldsymbol{\Delta}_{\mathbf{d}}$, we propose using the von Mises-Fisher distribution, as given in equation (8), as a substitute. Consequently, the conditional noise is sampled from the distribution (20). In the subsequent experiments, we use subscripts 1 and 2 to denote the conditional noise sampled from these two distributions, respectively. Thus, the NoiseCtrl sampling integrated with the DDPM algorithm, fast method for direction vector, and the second conditional distribution could be denoted as $\mathbf{DDPM}_{\mathbf{fast},\mathbf{1}}$. The von Mises-Fisher distribution introduces a hyperparameter κ. In the following experiments, we set it to 10^4. Regarding the configuration of the sampling algorithm, we utilize the default settings recommended by the authors.

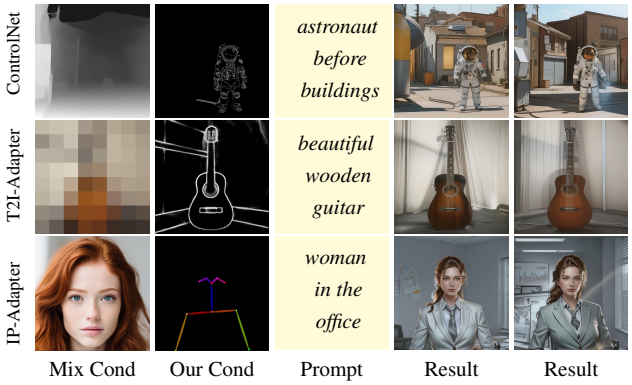

Figure 2. NoiseCtrl Illustration for Conditional Generation with Multiple Hints. In addition to the Stable Diffusion Model, our method is also compatible with other diffusion models such as ControlNet [35], T2I-Adapter [15], and IP-Adapter [33]. While these models can handle a single condition, integrating NoiseCtrl with them introduces an additional condition.

5.2. NoiseCtrl Compatibility

This section evaluates NoiseCtrl's compatibility. In the first experiment, we employ the Stable Diffusion v1.5 with NoiseCtrl and six sampling algorithms to conduct ten condi-

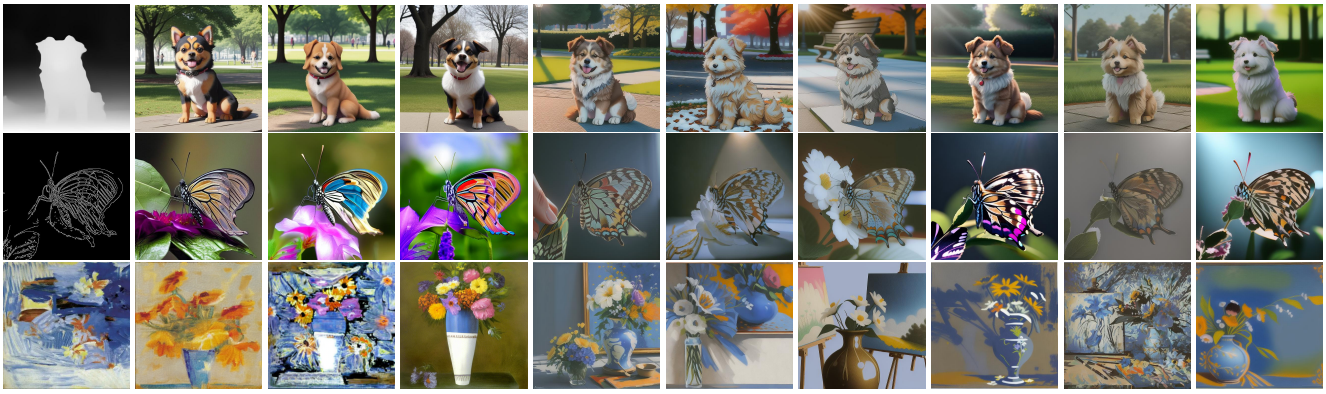

Figure 3. Visual Comparison for Training-free Generation with Depth, Canny and Style Conditions. Consistent with the quantitative comparison in Table 1, we compare a total of 9 methods, which include 3 training-free methods and 6 NoiseCtrl variants.

Method	Time(Sec)	Method	Time(Sec)	Method	Time(Sec)	Method	Time(Sec)
UGD [1]	4507	DDPM [9]	6	DDPM$_{\text{slow},1}$ / DDPM$_{\text{slow},2}$	76 / 54	DDPM$_{\text{fast},1}$ / DDPM$_{\text{fast},2}$	23 / 16
FreeDom [34]	116	DDIM [25]	6	DDIM$_{\text{slow},1}$ / DDIM$_{\text{slow},2}$	74 / 53	DDIM$_{\text{fast},1}$ / DDIM$_{\text{fast},2}$	21 / 15
DSG [32]	121	LCM [14]	4	LCM$_{\text{slow},1}$ / LCM$_{\text{slow},2}$	68 / 47	LCM$_{\text{fast},1}$ / LCM$_{\text{fast},2}$	16 / 10
ControlNet [35]	6	DPM [13]	4	DPM$_{\text{slow},1}$ / DPM$_{\text{slow},2}$	64 / 45	DPM$_{\text{fast},1}$ / DPM$_{\text{fast},2}$	19 / 12
T2I-Adapter [15]	4	Euler [10]	6	Euler$_{\text{slow},1}$ / Euler$_{\text{slow},2}$	78 / 55	Euler$_{\text{fast},1}$ / Euler$_{\text{fast},2}$	22 / 15
IP-Adapter [33]	6	SA [31]	5	SA$_{\text{slow},1}$ / SA$_{\text{slow},2}$	69 / 48	SA$_{\text{fast},1}$ / SA$_{\text{fast},2}$	18 / 11

Table 2. Efficiency Comparison. In the first column, we present the inference times of 6 CDMs, encompassing both training-based and training-free methods. NoiseCtrl's compatibility with arbitrary sampling algorithms allows us to list 6 such algorithms as the baseline for NoiseCtrl in the third column. In the fifth and seventh columns, we compare 24 NoiseCtrl variants against their respective baselines.

tional generation tasks. In the second experiment, we integrate NoiseCtrl with three training-required CDMs to manage multiple conditional generation tasks. Lastly, we note that we only present the results for NoiseCtrl with the fast direction vector estimation algorithm, as the outcomes for the slow algorithm are nearly identical to the fast version.

Figure 1 displays the conditional generation results using the Stable Diffusion v1.5 with six sampling algorithms: DDPM [9], DDIM [25], LCM [14], DPM [13], Euler [10], and SA [31]. The variety of conditions includes depth, canny edge, HED, MLSD, segmentation, normal, skeleton, location, and sketch images. The visual results confirm that NoiseCtrl is compatible with a range of sampling methods.

While the experiments in Figure 1 use the Stable Diffusion, NoiseCtrl is model-agnostic. In Figure 2, we illustrate NoiseCtrl's compatibility with three training-based CDMs: ControlNet [35], T2I-Adapter [15], and IP-Adapter [33], demonstrating its flexibility and effectiveness with different diffusion models.

5.3. NoiseCtrl Comparison

In this section, we carry out an extensive evaluation of NoiseCtrl, encompassing both qualitative and quantitative analyses to assess the generation quality. Furthermore, we also include a discussion on sampling efficiency.

Quantitative Evaluation: Our quantitative analysis is performed using the COCO2017 validation set at a resolution of 512×512. For quantitative evaluation, we randomly associate one caption with each image, resulting in 5,000 generated images for assessment. For style, we use the same captions and random select 5000 images from WikiArt dataset. We evaluate generation quality using FID [8], CLIP text-image score [18], CLIP aesthetic score [20]. We also evaluate controllability using MSE (Mean Squared Error) for Depth and Canny edge condition and Style loss [5] for Style condition. We take these metrics to compare the conditions extracted with the natural images (ground truth) with the generated images and report statistical data in Table 1. This table presents a comparison of 9 methods, which include three training-free methods: UGD [1], FreeDom [34], and DSG [32], as well as six NoiseCtrl variants: DDPM$_{\text{fast},2}$, DDIM$_{\text{fast},2}$, LCM$_{\text{fast},2}$, DPM$_{\text{fast},2}$, Euler$_{\text{fast},2}$, and SA$_{\text{fast},2}$. Note that we exclude training-based CDMs from this comparison, as it would be unfair to compare them with training-free CDMs that do not require additional training. The quantitative analysis reveals our model's superior performance in terms of FID, CLIP-score, and CLIP-acs across the majority of conditions, as well as its enhanced controllability, as indicated in the Controllability rows.

Qualitative Evaluation: Figure 3 displays the outcomes of NoiseCtrl for image generation across a variety of prompts and image conditions. The prompts, arranged from left to right, are as follows: "dog in the park," "beautiful butterfly," and "vase with flowers." The corresponding Depth maps, Canny edges, and Style conditions are detailed in the left column of the figure. Our model showcases its compatibility with the diverse conditional types. The generated images closely match the textual prompts and guidance images, thereby validating the model's effectiveness and adaptability in the realm of high quality conditional image synthesis.

Efficiency Evaluation: We assess the efficiency of our method, Table 2 records the inference times of 24 NoiseCtrl variants. We observe that the NoiseCtrl variant with the fast direction vector estimation algorithm and conditional noise sampled from the second distribution (20) is nearly identical to the original sampling algorithm, as shown in the third column. Compared to the second noise distribution (20), the variant with the first noise distribution (6) slightly increases inference time due to implementation inefficiency. We note that the variant using the slow version for direction vector estimation consumes the most time, as minimizing (13) is computationally expensive.

5.4. κ Investigation

The value κ, being closely linked to the controllability of NoiseCtrl, servers the concentration parameter of the von Mises-Fisher distribution (8). Another key parameter is the mean direction $\mathbf{d}$. For a vector $\mathbf{u}$ sampled from this distribution, the dot product $\mathbf{d}^T\mathbf{u}$ quantifies the angle between $\mathbf{u}$ and $\mathbf{d}$. Thus, the expectation of the dot product $\mathbf{d}^T\mathbf{u}$ reflects the degree to which the sampled vector $\mathbf{u}$ is oriented towards the mean direction $\mathbf{d}$. A smaller $\mathbb{E}[\mathbf{d}^T\mathbf{u}]$ indicates higher controllability. In n-dimensional space, the expected value $\mathbb{E}[\mathbf{d}^T\mathbf{u}]$ of the dot product between a vector $\mathbf{u}$ and the mean direction vector $\mathbf{d}$ is given by:

$$\mathbb{E}[\mathbf{d}^T\mathbf{u}] = \frac{\kappa}{\kappa + (n-1)}.$$

This formula shows that as κ increases, the sampled vector $\mathbf{u}$ aligns more closely with the given direction $\mathbf{d}$, while a smaller κ (indicating a more diffuse distribution) results in a smaller expected dot product.

Figure 4 illustrates the effect of κ on image generation. The bottom chart features two y-axes, left y-axis represents the expectation $\mathbb{E}[\mathbf{d}^T\mathbf{u}]$ and right y-axis indicates controllability measured by MSE. The x-axis denotes κ. Observing the blue line, we see that as κ increases, the expectation value decreases. In contrast, the controllability increases, as demonstrated by the red line. The bottom part presents visual examples for different κ values. These observations confirm that smaller κ values lead to more random images, while larger κ values enable controlled generation.

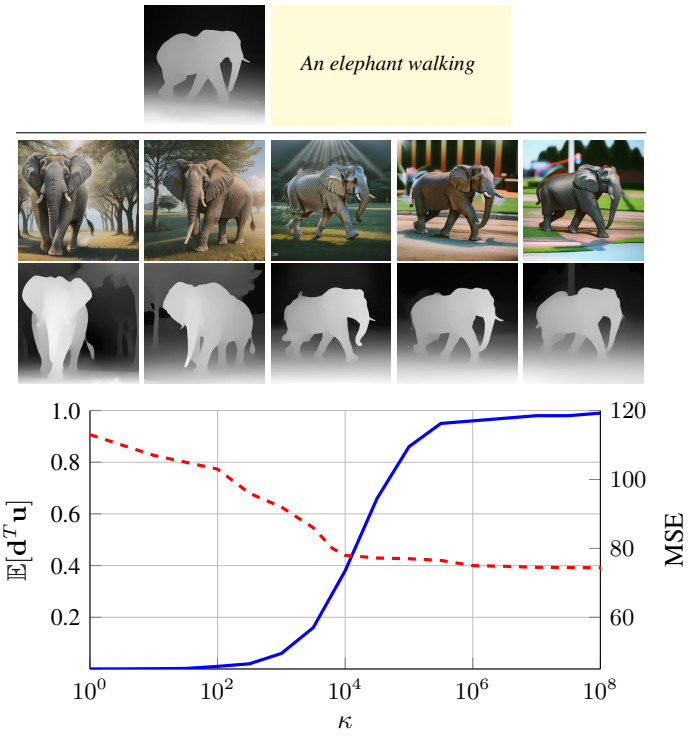

Figure 4. Impact of κ Illustration. The parameter κ determines the angular expectation between the sampled vector $\mathbf{u}$ and the mean vector $\mathbf{d}$. A smaller angular expectation indicates stronger control. The red line in the bottom chart represents the dot product $\mathbf{d}^T\mathbf{u}$ relative to κ. The blue lines track the MSE value, which measures the controllability of the method; a higher score signifies a lack of control and randomness. It is observed that as κ increases, the controllability improves. The top part illustrates the generation results across various κ values, with the associated condition depicted in the first row. From left to right, each column in the middle figure corresponds to κ values of 10^0, 10^2, 10^4, 10^6, and 10^8. The visual outcomes support the conclusions indicated by the blue line.

6. Conclusion

We propose the novel NoiseCtrl, a Sampling-Algorithm-Agnostic conditional sampling method designed to tackle a diverse array of conditional generation tasks. Our method is distinguished by two key innovations: 1. We introduce the use of conditional noise in place of Gaussian noise for coherent generation without altering existing sampling algorithms. 2. We present both fast and slow direction vector estimation methods to determine the condition of the conditional noise, and we advocate for the use of the von Mises-Fisher distribution instead of the uniform distribution for expedited sampling. NoiseCtrl's versatility positions it as an excellent plug-and-play enhancement for conditional diffusion models, offering substantial performance improvements without necessitating code modifications or significant computational overhead.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (62372237, 62332010) and the Major Science and Technology Projects in Jiangsu Province under Grant BG2024042

References

[1] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal Guidance for Diffusion Models. In *International Conference on Learning Representations*, 2024. 1, 2, 6, 7

[2] Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving Diffusion Models for Inverse Problems using Manifold Constraints. In *Advances in Neural Information Processing Systems*, 2022. 1, 2

[3] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion Posterior Sampling for General Noisy Inverse Problems. In *International Conference on Learning Representations*, 2023. 2

[4] Prafulla Dhariwal and Alexander Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*, 2021. 1, 2

[5] Chengming Feng, Jing Hu, Xin Wang, Shu Hu, Bin Zhu, Xi Wu, Hongtu Zhu, and Siwei Lyu. Controlling Neural Style Transfer with Deep Reinforcement Learning. In *International Joint Conference on Artificial Intelligence*, 2023. 6, 7

[6] Chunming He, Yuqi Shen, Chengyu Fang, Fengyang Xiao, Longxiang Tang, Yulun Zhang, Wangmeng Zuo, Zhenhua Guo, and Xiu Li. Diffusion Models in Low-Level Vision: A Survey, 2024. 1

[7] Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J. Zico Kolter, Ruslan Salakhutdinov, and Stefano Ermon. Manifold Preserving Guided Diffusion. In *International Conference on Learning Representations*, 2024. 2

[8] 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 *Advances in Neural Information Processing Systems*, 2017. 6, 7

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, 2020. 1, 2, 5, 6, 7

[10] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing Systems*, 2022. 2, 5, 6, 7

[11] Ming Li, Taojiannan Yang, Huafeng Kuang, Jie Wu, Zhaoning Wang, Xuefeng Xiao, and Chen Chen. ControlNet++: Improving Conditional Controls with Efficient Consistency Feedback. In *European Conference on Computer Vision*, 2024. 2

[12] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. GLIGEN: Open-Set Grounded Text-to-Image Generation. In *Conference on Computer Vision and Pattern Recognition*, 2023. 2

[13] 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 *Advances in Neural Information Processing Systems*, 2022. 2, 5, 6, 7

[14] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent Consistency Models: Synthesizing High-Resolution Images with Few-Step Inference. *arXiv*, 2023. 2, 5, 6, 7

[15] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2I-Adapter: Learning Adapters to Dig Out More Controllable Ability for Text-to-Image Diffusion Models. In *AAAI Conference on Artificial Intelligence*, 2024. 2, 6, 7

[16] Can Qin, Shu Zhang, Ning Yu, Yihao Feng, Xinyi Yang, Yingbo Zhou, Huan Wang, Juan Carlos Niebles, Caiming Xiong, Silvio Savarese, Stefano Ermon, Yun Fu, and Ran Xu. UniControl: A Unified Diffusion Model for Controllable Visual Generation In the Wild. In *Advances in Neural Information Processing Systems*, 2023. 2

[17] Weize Quan, Jiaxi Chen, Yanli Liu, Dong-Ming Yan, and Peter Wonka. Deep Learning-Based Image and Video Inpainting: A Survey. *International Journal of Computer Vision*, 2024. 1

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning*, 2021. 6, 7

[19] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis With Latent Diffusion Models. In *Conference on Computer Vision and Pattern Recognition*, 2022. 1

[20] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5B: An Open Large-Scale Dataset for Training Next Generation Image-Text Models. In *Advances in Neural Information Processing Systems*, 2022. 6, 7

[21] Fei Shen and Jinhui Tang. Imagpose: A unified conditional framework for pose-guided person generation. In *Conference on Neural Information Processing Systems*, 2024. 1

[22] Fei Shen, Hu Ye, Jun Zhang, Cong Wang, Xiao Han, and Wei Yang. Advancing pose-guided image synthesis with progressive conditional diffusion models. In *International Conference on Learning Representations*, 2023. 1

[23] Fei Shen, Xin Jiang, Xin He, Hu Ye, Cong Wang, Xiaoyu Du, Zechao Li, and Jinghui Tang. Imagdressing-v1: Customizable virtual dressing. In *AAAI Conference on Artificial Intelligence*, 2025. 1

[24] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, 2015. 1

[25] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*, 2020. 2, 5, 6, 7

[26] Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-Guided Diffusion Models for Plug-and-Play Controllable Generation. In *International Conference on Machine Learning*, 2023. 2

[27] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, 2019. 1

[28] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021. 1

[29] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency Models. In *International Conference on Machine Learning*, 2023. 2

[30] He Wang, Longquan Dai, and Jinhui Tang. EMControl: Adding Conditional Control to Text-to-Image Diffusion Models via Expectation-Maximization. In *AAAI Conference on Artificial Intelligence*, 2025. 2

[31] Shuchen Xue, Mingyang Yi, Weijian Luo, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhi-Ming Ma. SA-Solver: Stochastic Adams Solver for Fast Sampling of Diffusion Models. In *Conference on Neural Information Processing Systems*, 2023. 2, 5, 6, 7

[32] Lingxiao Yang, Shutong Ding, Yifan Cai, Jingyi Yu, Jingya Wang, and Ye Shi. Guidance with Spherical Gaussian Constraint for Conditional Diffusion. In *International Conference on Machine Learning*, 2024. 1, 2, 4, 5, 6, 7

[33] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. IP-Adapter: Text Compatible Image Prompt Adapter for Text-to-Image Diffusion Models. *arXiv*, 2023. 6, 7

[34] Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. FreeDoM: Training-Free Energy-Guided Conditional Diffusion Model. In *International Conference on Computer Vision*, 2023. 1, 2, 6, 7

[35] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models. In *International Conference on Computer Vision*, 2023. 1, 2, 6, 7

[36] Tianyi Zhang, Zheng Wang, Jing Huang, Mohiuddin Muhammad Tasnim, and Wei Shi. A Survey of Diffusion Based Image Generation Models: Issues and Their Solutions. *arXiv*, 2023. 1

[37] Min Zhao, Fan Bao, Chongxuan Li, and Jun Zhu. EGSDE: Unpaired Image-to-Image Translation via Energy-Guided Stochastic Differential Equations. In *Advances in Neural Information Processing Systems*, 2022. 3

[38] Shihao Zhao, Dongdong Chen, Yen-Chun Chen, Jianmin Bao, Shaozhe Hao, Lu Yuan, and Kwan-Yee K Wong. Uni-ControlNet: All-in-One Control to Text-to-Image Diffusion Models. In *Advances in Neural Information Processing Systems*, 2024. 2

[39] Yuanzhi Zhu, Kai Zhang, Jingyun Liang, Jiezhang Cao, Bihan Wen, Radu Timofte, and Luc Van Gool. Denoising Diffusion Models for Plug-and-Play Image Restoration. In *Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 1, 2