A Survey on Generative Diffusion Model

Hanqun Cao, Cheng Tan, Zhangyang Gao, Guanyong Chen, Pheng-Ann Heng, Senior Member, IEEE, and Stan Z. Li, Fellow, IEEE

Abstract—Deep learning shows excellent potential in generation tasks thanks to deep latent representation. Generative models are classes of models that can generate observations randomly with respect to certain implied parameters. Recently, the diffusion Model has become a rising class of generative models by virtue of its power-generating ability. Nowadays, great achievements have been reached. More applications except for computer vision, speech generation, bioinformatics, and natural language processing are to be explored in this field. However, the diffusion model has its genuine drawback of a slow generation process, single data types, low likelihood, and the inability for dimension reduction. They are leading to many enhanced works. This survey makes a summary of the field of the diffusion model. We first state the main problem with two landmark works – DDPM and DSM, and a unified landmark work – Score SDE. Then, we present classified improved techniques for existing problems in the diffusion-based model field. For model speed-up improvement, we present a diverse range of advanced techniques to speed up the diffusion models – training schedule, training-free sampling, mixed-modeling, and score & diffusion unification. For data structure diversification, we present improved techniques for applying diffusion models in continuous space, discrete space, and constraint space. For likelihood optimization, we present theoretical methods for improving ELBO and minimizing the variational gap. For dimension reduction, we present several techniques to solve the high dimension problem. Regarding existing models, we also provide a benchmark of FID score, IS, and NLL according to specific NFE. Moreover, applications with diffusion models are introduced including computer vision, sequence modeling, audio, and AI for science. Finally, there is a summarization of this field together with limitations & further directions. Summation of existing well-classified methods is in our Github: [https://github.com/chq1155/A-Survey-on-Generative-Diffusion-Model]

Index Terms—Diffusion Model, advanced improvement on diffusion, diffusion application.

1 INTRODUCTION

How can we empower machines with human-like imagination? Deep generative models, e.g., VAE [1], [2], [3], [4], EBM [5], [6], [7], GAN [9], [10], [11], [12], [13], normalizing flow [14], [15], [16], [17], [18], [19] and diffusion models [20], [21], [22], [23], [24], have shown great potential in creating new patterns that humans cannot properly distinguish. We focus on diffusion-based generative models, which do not require aligning posterior distributions as VAE, dealing with intractable partition functions as EBM, training additional discriminators as GAN, or imposing network constraints as normalizing flow. Thanks to the aforementioned virtues, diffusion-based methods have drawn considerable attention from computer vision, and natural language processing to graph analysis. However, there is still a lack of systematic taxonomy and analysis of research progress on diffusion models.

Advances in the diffusion model have provided tractable probabilistic parameterization for describing the model, a stable training procedure with sufficient theoretical support, and a unified loss function design with high simplicity. The diffusion model aims to transform the prior data distribution into random noise before revising the transformations step by step to rebuild a brand new sample with the same distribution as the prior [25]. In recent years, the diffusion model has displayed its exquisite potential in the field of computer vision (CV) [20], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], sequence modeling [39], [40], [41], [42], audio processing [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], and AI for science [53], [54], [55], [56], [57]. Inspired by the so-far successes of the diffusion model in these popular domains, applying diffusion models to generation-related tasks of the other domains would be a favorable path for exploiting powerful generative capacity.

On the other hand, the diffusion model has the inherent drawback of plenty of sampling steps and a long sampling time compared to Generative Adversarial Networks (GANs) and Variational Auto-Encoders (VAEs). Since diffusion models leverage a Markov process to convert data distribution via tiny perturbations, a large number of diffusion steps are required in both the training and inference phases. Thus, it takes more time to sample from a random noise until it eventually alters to high-quality data similar to the prior. Furthermore, other problems such as likelihood optimization and the inability of dimension reduction also count. Therefore, lots of works aspired to accelerate the diffusion process along with improving sampling quality [61], [62], [63]. For example, DPM-solver takes advantage of ODE’s stability to generate samples of the State-of-the-art within 10 steps [64]. D3PM [55] proposes not only hybrid training loss but also text & categorical data. We summarize im-

H. Cao is with the Department of Math, The Chinese University of Hong Kong, Hong Kong, China, also with Zhejiang Lab, Hangzhou, China, and the AI Lab, School of Engineering, Westlake University, Hangzhou, China. Email: 1155141481@link.cuhk.edu.hk.

C. Tan and Z. Gao are with Zhejiang University, Hangzhou, China, and also with the AI Lab, School of Engineering, Westlake University, Hangzhou, China. Email: tancheng, gaozhangyang@westlake.edu.cn.

G. Chen is with Zhejiang Lab, Hangzhou, China. Email: gy-chen@zhejianglab.com.

P.-A. Heng is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China.

Stan Z. Li is with the AI Lab, School of Engineering, Westlake University, Hangzhou, China. Email: Stan.ZQ.Li@westlake.edu.cn.

H. Cao, C. Tan, and Z. Gao contributed equally to this work.

Manuscript received April 19, 2020; revised August 26, 2020.
Fig. 1. Generative Models Pipeline. (a) Generative Adversarial Network (GAN) [58] applies adversarial training strategy onto the generator to generate lifelike samples like input distributions. (b) Energy-Based Model (EBM) [59] designs a suitable energy function for pair-wise energy matching between conditions and samples, similar to a generative discriminator in GAN. (c) Variational Auto Encoder (VAE) [60] applies the encoder to project the prior into a latent space with reduced dimension from which the decoder can sample. (d) Normalizing Flow (NF) [18] employs a well-designed reversible flow function for turning input into latent variable before returning to samples with the inverse of flow function. (e) Diffusion model gradually injects noise into the original data until it turns to the known noise distribution before reversing each step in the sampling steps.


text

2 PROBLEM STATEMENT

2.1 Notions and Definitions

2.1.1 State

States are a set of data distributions that describe the whole process of diffusion models. The noise is gradually injected into the starting distribution, called starting state $x_0$. With enough steps of noise injection, the distribution finally comes into a known noise distribution (mostly Gaussian), which is called the prior state $x_T$ (Discrete) or $x_1$ (Continuous). Then, the other distributions between the starting state and the prior state are called intermediate states $x_t$.

2.1.2 Process & Transition Kernel

**Forward & Reverse Process & Kernel:** The process that transforms the starting state into the tractable noise is the forward/diffusion process $F$. The process following the opposite direction to the forward process is called reverse/denoised process $R$. The reverse process samples the noise gradients step by step into the samples as the starting state. In either process, the interchange between any two states is achieved by the transition kernel. To present a unified framework, the forward process consists of plenty of forward steps which are the forward transition kernels. The reverse process conducts similarly:

$$F(x, \sigma) = F_t(x_{t-1}, \sigma_{t-1}) \cdots \circ F_1(x_0, \sigma_1) \cdots \circ F_1(x_0, \sigma_1) \quad (1)$$

$$R(x, \sigma) = R_t(x_t, \sigma_t) \cdots \circ R_1(x_t, \sigma_1) \cdots \circ R_T(x_T, \sigma_T) \quad (2)$$

$$x_t = F_t(x_{t-1}, \sigma_t), \quad x_{t-1} = R_t(x_t, \sigma_t) \quad (3)$$

Different from the discrete case, for any time $0 \leq t < s \leq 1$, the forward process is defined:

$$F(x, \sigma) = F_{t_1}(x_s, \sigma_{t_1}) \circ F_{t_2}(x_s, \sigma_{t_2}) \cdots \circ F_{t_0}(x_0, \sigma_{t_0}) \quad (4)$$

$$R(x, \sigma) = R_{t_0}(x_s, \sigma_{t_0}) \circ R_{t_1}(x_s, \sigma_{t_1}) \cdots \circ R_{t_s}(x_T, \sigma_{t_s}) \cdots \circ R_{t_1}(x_1, \sigma_{t_1}) \cdots \circ R_{t_0}(x_0, \sigma_{t_0}) \quad (5)$$

$$x_s = F_{t_s}(x_s, \sigma_{t_s}), \quad x_t = R_{t_s}(x_s, \sigma_{t_s}) \quad (6)$$

where $F_t, R_t$ are the forward and reverse transition kernels at time $t$ with the variables intermediate state $x_{t-1} \& x_t$ and the noise scale $\sigma_t$. The most frequently used kernel is the Markov kernel since it ensures randomness and tractability in the forward process and the reverse process. The difference between this expression and normalizing flow is the variable noise scale, which controls the randomness of the whole process. When the noise is close to 0, the process will become the normalizing flow which is deterministic.

**Pipeline:** Denote the sampled data as $\tilde{x}_0$ the generalized process can be expressed as:

$$\tilde{x}_0 = [R_1(x_1, \sigma_1) \cdots \circ R_t(x_t, \sigma_t) \cdots \circ R_T(x_T, \sigma_T)] \circ [F_t(x_{t-1}, \sigma_{t-1}) \cdots \circ F_1(x_0, \sigma_1)] \quad (7)$$

$$\tilde{x}_0 = [R_0(x_t, \sigma_0) \cdots \circ R_{t_s}(x_s, \sigma_{t_s}) \cdots \circ R_{t_1}(x_T, \sigma_{t_1})] \circ [F_{t_1}(x_s, \sigma_{t_1}) F_{t_2}(x_s, \sigma_{t_2}) \circ F_{t_0}(x_0, \sigma_{t_0})] \quad (8)$$

- **Summarize essence mathematical formulation and derivation of fundamental algorithms in the field of diffusion model, including taking advantage of training strategy, and sampling algorithm.**
- **Present comprehensive and up-to-date classification of improved diffusion algorithms and divide them into four proposes, which are speed-up improvement, structure diversification, likelihood optimization, and dimension reduction.**
- **Provide extensive statements about the application of diffusion models on computer vision, natural language processing, bioinformatics, and speech processing which include domain-specialized problem formulation, related datasets, evaluation metrics, and downstream tasks, along with sets of benchmarks.**
- **Clarify current limitations of models and possible further-proof directions concerning the field of diffusion models.**
2.1.3 Discrete and continuous

Taking the perturbation kernel to sufficiently small, the whole discrete process will contain infinite steps. To tackle the mechanism behind this situation, the continuous process starting from time 0 and ending at time 1 is sued in many improved algorithms [66, 67] to obtain better performance. Compared to a discrete process, the continuous one enables the extraction of any information from any time state. Furthermore, assuming the change of perturbation kernel is slight enough, the continuous process enjoys better theoretical support.

2.1.4 Training Objective

The diffusion model as one type of the generative model follows the same training objective as variational autoregressive-encoder and normalizing flow, which is keeping starting distribution \( x_0 \) and sample distribution \( \tilde{x}_0 \) as close as possible. This is implemented by maximizing the log-likelihood [28]:

\[
\mathbb{E}_{F(x_0, \sigma)} \left[-\log R(x_T, \tilde{\sigma})\right]
\]

(9)

where the \( \tilde{\sigma} \) in the reverse process differs from the one in the forward process.

2.2 Problem Formulation

2.2.1 Denoised Diffusion Probabilistic Model

**DDPM Forward Process:** Based on the unified framework, DDPM chooses a sequence of noise coefficients \( \beta_1, \beta_2, \ldots, \beta_T \) for Markov transition kernels following specific patterns. The common choices are constant schedule, linear schedule, and cosine schedule. According to [68], different noise schedules have no clear effects in experiments. The DDPM forward steps are defined as:

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

(10)

By a sequence of diffusion steps from \( x_0 \) to \( x_T \), we have the Forward Diffusion Process:

\[
F(x_0, \beta) := q(\{x_t\}_{t=0}^T) := \prod_{t=1}^T q(x_t | x_{t-1})
\]

(11)

**DDPM Reverse Process:** Given the Forward Process above, we define the Reverse Process with learnable Gaussian transitions parameterized by \( \theta \) [68]:

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

(12)

By a sequence of reverse steps from \( x_T \) to \( x_0 \), we have the reverse process starting at \( p(x_T) = \mathcal{N}(x_T; 0, I) \):

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

(13)

Consequently, the distribution \( p_\theta(x_0) = \int p_\theta(x_{0:T}) dx_{1:T} \) should be the distribution of \( \tilde{x}_0 \).

**Diffusion Training Objective:** By minimizing the negative log-likelihood (NLL), the minimization problem can be formulated as:

\[
\begin{array}{c|c}
\text{Notations} & \text{Descriptions} \\
\hline
T & \text{Discrete total time steps} \\
\tau & \text{Random time } t \\
\epsilon & \text{Random noise with normal distribution} \\
\mathcal{N} & \text{Normal distribution} \\
\beta & \text{Generalized process noise scale} \\
\sigma & \text{Noise scale of perturbation} \\
\sigma_I(t) & \text{Continuous-time } \sigma_I(t) \\
\sigma_I & \text{Continuous-time } \sigma_I \\
\alpha(t) & \text{Mean coefficient defined as } 1 - \beta_t \\
\alpha_I(t) & \text{Continuous-time } \alpha_I(t) \\
\gamma(t) & \text{Cumulative product of } \alpha(t) \\
y(t) & \text{Signal-to-Noise ratio} \\
\delta_t & \text{Step size of annealed Langevin dynamics} \\
x & \text{Unperturbed data distribution} \\
\tilde{x} & \text{Perturbed data distribution} \\
\tilde{x}_0 & \text{Starting distribution of data} \\
x_1 & \text{Diffused data at time } t \\
\tilde{x}_t & \text{Partly diffused data at time } t \\
\tilde{x}_T & \text{Random noise after diffusion} \\
F(x, \sigma) & \text{Forward/Diffusion process} \\
\aleph(x, \sigma) & \text{Reverse/Denoised process} \\
\rho(x_{t-1}|x_t) & \text{Forward/Diffusion step at time } t \\
R_t(x_t, \sigma_t) & \text{Reverse/Denoised step at time } t \\
\mathcal{R}(x_t, \sigma_t) & \text{Forward/Diffusion step at time } t \\
\mathcal{R}_(x_t, \sigma_{t'}) & \text{Reverse/Denoised step at time } t' \\
q(x_t|x_{t-1}) & \text{DDPM forward step at time } t \\
p(x_{t-1}|x_t) & \text{DDPM reverse step at time } t \\
\gamma(x, t) & \text{Drift coefficient of SDE} \\
\beta(t) & \text{Simplified diffusion coefficient of SDE} \\
\mathcal{R}(x, t) & \text{Degrader at time } t \text{ in Cold Diffusion} \\
\mathcal{R}(x, t) & \text{Reconstructor at time } t \text{ in Cold Diffusion} \\
\mu, \tilde{\mu} & \text{Standard Wiener process} \\
\nu \log p_\theta(x) & \text{Score function w.r.t } x \\
\mathcal{L}(\mu(x, t)) & \text{Mean coefficient of reversed step} \\
\Sigma_\theta(x_t, t) & \text{Variance coefficient of reversed step} \\
\psi(x_t, t) & \text{Noise prediction model} \\
s_\theta(x_t, t) & \text{Score network model} \\
\mathcal{R}(x_{t+1}, L_T) & \text{Forward loss, reversed loss, decoder loss} \\
\mathcal{L}_{\text{E}} & \text{Evidence Lower Bound} \\
\mathcal{L}_{\text{C}} & \text{Continuous evidence lower bound} \\
\mathcal{L}_{\text{C}} & \text{Simplified denoised diffusion loss} \\
\mathcal{L}_{\text{C}} & \text{Continuous } \mathcal{L}_{\text{C}} \\
\mathcal{L}_{\text{Gap}} & \text{Variational gap} \\
\mathcal{L}_{\text{KID}} & \text{Kernel inception distance} \\
\mathcal{L}_{\text{Recovery}} & \text{Recovery likelihood loss} \\
\mathcal{L}_{\text{Hybrid}} & \text{Hybrid diffusion loss} \\
\mathcal{L}_{\text{DDPM&GAN}} & \text{DDPM ELBO and GAN hybrid loss} \\
\mathcal{L}_{\text{DDPM&VAE}} & \text{DDPM ELBO and VAE hybrid loss} \\
\mathcal{L}_{\text{DDPM&Flow}} & \text{DDPM ELBO and normalizing flow hybrid loss} \\
\mathcal{L}_{\text{DSM}} & \text{Loss of denoised score matching} \\
\mathcal{L}_{\text{ISL}} & \text{Loss of implicit score matching} \\
\mathcal{L}_{\text{ISM}} & \text{Loss of sliced score matching} \\
\mathcal{L}_{\text{Distill}} & \text{Diffusion distillation loss} \\
\mathcal{L}_{\text{DDPM&Noise}} & \text{DDPM ELBO and reverse noise hybrid loss} \\
\mathcal{L}_{\text{Square}} & \text{Noise square loss} \\
\mathcal{L}_{\text{trajectory}} & \text{Process optimization loss} \\
\mathcal{L}_{\text{DDPM&Classifier}} & \text{DDPM ELBO and classification hybrid loss} \\
\theta & \text{learnable parameters} \\
\phi & \text{learnable parameters} \\
\end{array}
\]
show the training pattern of the noise prediction model $L_0$, the green part represents forward loss $L_T$, and the orange part constitutes the reverse loss $L_1$. Dashed lines with different colors show the training pattern of the noise prediction model $\epsilon_\theta$. Besides, in any step $1 \leq t \leq T$, the yellow lines denote the ancestral sampling process.

\[
\mathbb{E} \left[ -\log p_\theta (x_0) \right] \leq \mathbb{E}_q \left[ -\log p_\theta (x_{0:T}) \right] = \mathbb{E}_q \left[ -\log p (x_T) - \sum_{t \geq 1} \log p_\theta (x_{t-1} | x_t) \right] = \mathbb{E}_q \left[ D_{KL} (q (x_T | x_0) || p (x_T)) \right]
\]

\[
= \sum_{t \geq 1} \mathbb{E}_q \left[ D_{KL} (q (x_{t-1} | x_t, x_0) \| p_\theta (x_{t-1} | x_t)) \right] - \log p_\theta (x_0 | x_1)
\]

\[
L := L_0 + L_T
\]

Here we use the symbol of Ho et al. [68]. Denote $L_T$ as the prior loss. Denote $L_0$ as the reconstruction loss; Besides, denote $L_{1:T-1}$ as the consistent loss, which is the sum of the divergence between the posterior of the forwarding step and the corresponding reversing step.

2.2.2 Score Matching Formulation

The score matching model aims at solving the original data distribution estimation problem by approximating the gradient of data $\nabla_x \log p(x)$, which is called score. The main approach of score matching is to train a score network $x_\theta$ to predict the score $[69], [70]$, which is obtained using perturbing data with different noise schedules. The score-matching process is defined as:

Score Perturbation Process & Kernel: The perturbation process consists of a sequence of perturbation steps with increasing noise scales $\sigma_1, \ldots, \sigma_N$. The Gaussian perturbation kernel is defined as $q_\sigma (\tilde{x} | x) := \mathcal{N} (\tilde{x} | x, \sigma^2 I)$. For each noise scale $\sigma_t$, the score is equivalent to the gradient of the perturbation kernel. If we treat this increasing noise perturbation as a discrete process, the transition kernel between two neighbor states is

\[
x_i = x_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} \epsilon, \quad i = 1, \ldots, N
\]

where $N$ is the length of the noise scale sequence, and $\epsilon$ is random noise.

Score Matching Process: As noticed above, the goal of the score matching process is to obtain a score estimation network $s_\theta (x, \sigma)$ to be as close as possible to the gradient of perturbation kernel, which is

\[
L := \frac{1}{2} \mathbb{E} \left[ \| s_\theta (x, \sigma) - \nabla \log p (x) \|^2 \right]
\]

where $\theta$ is the learnable parameters in the score network.

DDPM & DSM Connection: To some extent, score matching, and denoising diffusion are the same processes. (1) Denoising mechanism: both DSM and DDPM follow the pattern of fetching information during the noising process and replaying the gradient of the denoising process. Moreover, noising schedule of DSM can be seen as an accumulation of constant-variance diffusion steps. (2) Training object: Both DSM and DDPM belong to a noise regression problem based on MLE. (3) Sampling method: both DSM and DDPM apply the idea of ancestral sampling, employing gradient to reconstruct the samples.

2.2.3 Score SDE Formulation

Score SDE [66] proposed a unified continuous framework based on the stochastic differential equation to describe diffusion and denoised score matching models. It not only presents the corresponding continuous set-up of DDPM of DSM based on score SDE but also proposes a density estimation ODE framework named probability flow ODE.
Fig. 3. Pipeline of Denoised Score Matching (DSM). The α’s in different time states on the top represent alternative noise scales. The transition states \( p_{\alpha_t}(\tilde{x}_t|x_t) \) are the output gradients of the perturbation. Dashed lines with different colors reveal that the scoring network \( s_\theta \) is trained by minimizing the sum of L2-loss between the output gradient and the score in each noise scale. Besides, in any noise state \( 1 \leq t \leq T \), the red lines denote the Langevin Dynamics sampling process.

**Forward Score SDE Process:** In Song et al. [66], Diffusion process can be viewed as a continuous case described by Stochastic Differential Equation. And it is equal to the solution to Itô SDE [71], which is composed of a drift part for mean transformation and a diffusion coefficient for noise description:

\[
dx = f(x, t) dt + g(t) dw, t \in [0, T]
\]

where \( w \) is the standard Wiener process/Brownian Motion, \( f(., t) \) is the drift coefficient of \( x(t) \), and \( g(.) \) is the simplified version of diffusion coefficient of \( x(t) \), which is assumed not dependent on \( x \). Besides, \( p_0, p_t(x) \) denote the data distribution and probability density of \( x(t) \). \( p_T \) denotes the original prior distribution which gains no information from \( p_0 \). When the coefficients are piece-wise continuous, the forward SDE equation admits a unique solution [72].

Similar to the discrete case, the forward transition in the SDE framework is derived as:

\[
F_{ts}(x(s), g_{ts}) := q(x \mid x_s) := \mathcal{N}(x_s \mid f_{ts}x_s, g_{ts}^2 I)
\]

\[
R_{ts}(x(t), g_{ts}) := q(x \mid x_t, x_0) = \mathcal{N}(x_t \mid f_{ts}x_0, g_{ts}^2 I) \cdot \frac{g_{ts}^2}{g_{00}^2} I
\]

\[
\text{where } f_{ts} = \frac{f(\cdot, t)}{f(\cdot, s)} \text{ and } g_{ts} = \sqrt{g(t)^2 - f_{ts}^2 g(s)^2}.
\]

**Reversed Score SDE Process:** In contrast to the Forward SDE Process, the Reversed SDE Process is defined with respect to the reverse-time Stochastic Differential Equation by running backward in time [66]:

\[
dx = [f(x, t) - g(t)^2 \nabla_x \log p_t(x)] dt + g(t) dw, t \in [0, T]
\]

Furthermore, \( \nabla_x \log p_t(x) \) is the score to be matched [73].

**Score SDE Training Objective:** The training objective of score SDE employs weighting scheme in the score loss compared to denoised score matching, which is

\[
L := \mathbb{E}_t \{ \lambda(t) \mathbb{E}_{x(0)} [\nabla_x \log p(x(t), x(0))]^2 \}
\]

where \( x(t), x(0) \) are corresponding continuous time variables of \( x_t, x_0 \).

**SDE-based DDPM & DSM:** Based on the SDE frameworks, the transition kernel of DDPM and DSM can be expressed as:

\[
dx = \frac{1}{2} \beta(t)x dt + \sqrt{\sigma(t)} dw
\]

where \( \beta(t) \) and \( \sigma(t) \) are the continuous-time variable of the discrete noise scales \( \beta_t \) and \( \sigma_t \). Moreover, the two kinds of SDE are called Variation Preserving (VP) and Variation Explosion (VE) SDE respectively.

**Probability Flow ODE:** Probability Flow ODE (Diffusion ODE) [66] is the continuous-time ODE that supports the deterministic process which shares the same marginal probability density with SDE. Inspired by Maoutsa et al. [74] and Chen et al. [75], any type of diffusion process can be derived into a special form of ODE. In the case that functions \( G \) is independent of \( x \), the probability flow ODE is
In contrast to SDE, probability flow ODE can be solved with larger step sizes as they have no randomness. Due to the advantages of ODE, several works such as PNDMs [76] and DPMSolver [64] obtain amazing results by modeling the diffusion problem as an ODE.

### 2.3 Training Strategy

#### 2.3.1 Denoising Diffusion Training Strategy

In order to minimize the negative log-likelihood, the only item we can be used to train is \( L_{t=T} \). By parameterizing the posterior \( q(x_t | x_{t-1}, x_0) \) using Baye’s rule, we have:

\[
\begin{align*}
q(x_t | x_{t-1}, x_0) &= \mathcal{N}(x_t | \mu_t(x_{t-1}, x_0), \beta_t I) \\
\end{align*}
\]

where \( \alpha_t \) is defined as \( 1 - \beta_t, \bar{\alpha}_t \) is defined as \( \prod_{k=1}^{t} \alpha_k \). Mean and variance schedules can be expressed as:

\[
\begin{align*}
\mu_t(x_t, x_0) &= \frac{\sqrt{\alpha_{t-1}} \beta_t x_0 + (1 - \bar{\alpha}_{t-1}) x_t}{1 - \bar{\alpha}_t} \\
\beta_t &= \frac{1}{1 - \bar{\alpha}_t} \bar{\alpha}_t \\
\end{align*}
\]

Keeping above parameterization as well as reparameterizing \( x_t \) as \( x_t(x_0, \sigma) \), \( L_{t=1} \) can be regarded as an expectation of L2-loss between two mean coefficients:

\[
L_{t=1} = \mathbb{E}_q \left[ \frac{1}{2 \sigma_t^2} \right] = \mathbb{E}_q \left[ \frac{1}{2 \sigma_t^2} \right] + C \tag{26}
\]

Simplifying \( L_{t=1} \) by reparameterizing \( \mu_t \) w.r.t \( \epsilon_\theta \), we obtain the simplified training objective named \( L_{\text{simple}} \):

\[
L_{\text{simple}} := \mathbb{E}_q \epsilon \left[ \frac{1}{2 \sigma_t^2} \right] = \mathbb{E}_q \epsilon \left[ \frac{1}{2 \sigma_t^2} \right] + C \tag{27}
\]

Most diffusion models until now use the training strategy of DDPMs. But there exist some exceptions. DDIM’s training objective [77] can be transformed by adding a constant from DDPM’s although it is independent of Markovian step assumption; Training pattern of Improved DDPM [61] named as \( L_{\text{hybrid}} \) is to combine training object of DDPM \( L_{\text{simple}} \) and a term with variational lower bound \( L_{\text{vb}} \). However, \( L_{\text{simple}} \) still takes the main effect of these training methods.

#### 2.3.2 Score Matching Training Strategy

Traditional score-matching techniques require massive computation cost for Hessian of log density function. To fix this problem, advanced methods find approaches to avoid Hessian computing. Implicit score matching (ISM) [73] treat the real score density as a non-normalized density function that can be optimized by neural network. Sliced score matching (SSM) [78] provide a unperturbed score estimation method through reverse-mode auto-differentiation by projecting score onto random vectors.

\[
L_{\text{ISM}} := \mathbb{E}_q \left[ \frac{1}{2} \| \nabla \phi(x) \|^2 + \nabla \phi(x) \right] \tag{28}
\]

\[
L_{\text{SSM}} := \mathbb{E}_{q, E} \mathbb{E}_{p_{\text{data}}} \left[ v^\top \nabla_x \phi(x)v + \frac{1}{2} \| \nabla \phi(x) \|^2 \right] \tag{29}
\]

However, because of the low-manifold problem in real data as well as the sampling problem in the low-density region, denoised score matching could be the better solution for improving score matching. Denoised score matching (DSM) [69] transforms the original score matching into a perturbation kernel learning by perturbing a sequence of increasing noise.

\[
L_{\text{DSM}} := \mathbb{E}_{q, E} \mathbb{E}_{p_{\text{data}}} \left[ \| \nabla \phi(x) \|^2 \right] \tag{30}
\]

According to Song et al., the noise distribution is defined to be \( q(x) = N(x | \sigma^2 I) \). Thus, for each given \( \sigma \), the specific expression denoising score matching objective is

\[
L(\theta; \sigma) := \mathbb{E}_{p_{\text{data}}} \mathbb{E}_{q, E} \left[ \| \nabla \phi(x) + \frac{x - \bar{x}}{\sigma^2} \|^2 \right] \tag{31}
\]

### 2.4 Sampling Algorithm

In the reverse process, the samples are rebuilt from random noise by extracting the gradient in each time step, which is called unconditional sampling. Besides, there is another class of sampling utilizing specific conditions. We call it conditional sampling. In this subsection, we present basic unconditional sampling algorithms for the three landmark works and effective conditional sampling algorithms in Appendix.

#### 2.4.1 Unconditional Sampling

**Ancestral Sampling** The initial idea of ancestral sampling [79] is reconstructed with the gradient of inverse Markovian step by step.

**Langevin Dynamics Sampling** With a fixed step size \( \epsilon > 0 \), Langevin dynamics can produce samples from a probability density \( p(x) \) through only the score function (Song et al.) \( \nabla_x \phi(x) \).

**Predictor-corrector (PC) Sampling** PC sampling [80] is inspired by a type of ODE black-box ODE solver [81], [82], [83] to produce high-quality samples and trade-off accuracy for efficiency for all reversed SDE. The sampling procedure comprises a predictor sampler and a corrector sampler.

#### 2.4.2 Conditional Sampling

**Labeled Condition** Sampling with labeled conditions provides gradient guidance in each sampling step. Usually, an additional classifier with UNet Encoder architecture for generating condition gradients for specific labels is needed. The labels can be text & categorical label [84], [85], [86], [87], [88], binary label [89], [90], or extracted features [26], [91], [92]. It is firstly presented by [84], and current conditional sampling methods are similar in theory.

**Unlabeled Condition** Except for label-guidance sampling, unlabeled condition sampling only takes self-information as guidance. Conducting in a self-supervised manner [93], [94], it is often applied in denoising [95], resolution [96], and inpainting [41] tasks.
3 ALGORITHM IMPROVEMENT

Nowadays, the main constraint of the diffusion model is its low speed and high computation cost. Although conditional diffusion with strong guidance can achieve high-fidelity samples within 10 steps [62], [97], unconditional sampling speed remains incomparable to GAN and VAE. Besides, handling diverse data distribution, optimizing log-likelihood, and dimension reduction techniques still count. In this section, we classify the improved algorithm w.r.t. the mainstream problems. For each problem, we present the significance and detailed taxonomy of improved techniques.

3.1 Speed-up Improvement

Although diffusion models enjoy high-fidelity generation, low sampling speed limits models’ practicality. To improve this situation, advanced techniques can be divided into four categories, including training scheme enhancement, training-free accelerated sampling, mix-modeling design, and score-diffusion unification design.

3.1.1 Training Schedule

Improving the training schedule means modifying traditional training settings, such as diffusion schemes and noise schemes, which are independent of sampling. Recent studies have shown the key factors in training schemes influencing learning patterns and models’ performance. In this sub-section, we divide the training enhancement into three categories: knowledge distillation, diffusion scheme learning, and noise scale design.

Knowledge Distillation

Knowledge distillation is an emerging method for obtaining efficient small-scale networks by transferring “knowledge” from complex teacher models with high learning capacity to simple student models [154], [155]. Thus, student models equip the advantages in model compression and model acceleration [156], [157].

\[
\theta_{stu} := \theta_{stu} - \gamma \cdot \nabla_x \left[ \left( R_{teca}(R_{teca}(x_t, t - 1)) - R_{stu}(x_t, t) \right) \right]
\]  

(32)

Salimans et al. [62] first applies the core idea into diffusion model improvement by progressively distilling knowledge from one sampling model to another. In each distillation stage, student models learn to conduct two-step updates from teacher models in a one-step manner to halve their sampling steps. Unlike progressive distillation, denoising student [98] distills knowledge from scratch by minimizing KL Divergence between two categorical distributions.

Diffusion Scheme Learning

Similar to VAE, the forward diffusion process can be viewed as an encoder that projects data into many latent spaces. Thus, an effective yet expressive diffusion pattern is necessary for effective reverse decoding. Compared to VAE, the diffusion model encodes data onto latent spaces with the same dimension to achieve high expressiveness in a more complex way. Thus, we divide current methods as projecting approaches exploration and encoding degree optimization.

For encoding degree optimization methods, CCDF [95] and Franzese et al., [103] establish an optimization problem where the number diffusion step is treated as a variable for minimizing ELBO from a theoretical perspective [158], [159]. Another approach is based on truncation, which conducts a trade-off between generating speed and sample fidelity. Truncating patterns samples from less diffused data generated by GAN and VAE in a one-step manner. TDPM [79] truncates both diffusion and sampling processes by sampling from implicit generative distribution learned by GAN and conditional transport (CT) [160]. Similarly, Early Stop (ES) DDPM [101] learns from latent space to generate implicit distributions.

\[
\hat{x} := R_{v0}(F_{v0}(x_0, \sigma_{v0}^-), \sigma_{v0}^-), \quad t^* \in [0, T]
\]  

(33)

For projecting approaches exploration, some works focus on the diversity of diffusion kernels. Soft diffusion [102], and blurring diffusion model [100] succeed in proving linear corruptions, including blurring and mask, can also serve as transition kernels.

Noise Scale Designing

In the traditional diffusion process, each transition step is determined by the injected noise, which is equivariant to a random walk on the forward and reversed trajectories. Thus, noise scale designing has the potential for reasonable generation and fast convergence. Unlike traditional DDPM, existing methods treat noise scale as a learnable parameter throughout the whole process.

\[
\text{SNR}(t) := \alpha^2_t / \beta^2_t = \exp(g_q(t)), \quad \alpha^2_t = \text{sigmoid}(g_q(t))
\]  

(34)

\[
L_T(x) = \frac{T}{2} \mathbb{E}_{z \sim N(0, 1)} \left[ \left( \text{SNR}(t) - \text{SNR}(t) \right) \| x - \tilde{x}_t(z, t) \|_2^2 \right]
\]

\[
L_{\infty}(x) = \frac{1}{2} \mathbb{E}_{v \sim N(0, 1)} \left[ \text{SNR}_{\max} \|| x - \tilde{x}(z_v, v) \|_2^2 \right]
\]  

(35)

Among the forward noise design methods, VDM [67] parameterizes the noise scalar as signal-to-noise ratio for connecting noise scale and training loss and model types. FastDDPM [105] obtains forward noise from the discrete-time variables or variance scalar, connecting noise design to ELBO optimization. In the reverse noise design methods, improved DDPM [61] learns the reverse noise scale implicitly by training a hybrid loss containing \(L_{simple}\) and \(L_{vbl}\). Besides, San Roman et al. employs a noise prediction network to update the reverse noise scale directly before conducting ancestral sampling in each step.

3.1.2 Training-Free Sampling

Training enhancement methods focus on changing the training pattern and noise schemes for sampling speed-up, it can also be achieved by designing advanced sampling algorithms. Based on the fact that the gradient of data is stored in the pre-trained diffusion models, training-free methods apply pre-trained information directly to the advanced sampling algorithms with fewer steps and higher fidelity, which are free from model re-training. In this subsection, we divide them into four categories: analytical methods,
TABLE 2
Classification of Improved Diffusion Techniques

| Training Scheme                      | Diffusion Scheme | Improvement          |
|--------------------------------------|------------------|----------------------|
| Knowledge Distillation               | Progressive Distillation | [62]            |
| Denoising Student                    | TDDPM [99]        | / Blurring Diffusion [100] |
| Diffusion Scheme Learning            | ES-DDPM [101]     | / Soft Diffusion [102] |
| Noise Scale Designing                | CCDF [95]         | / [103] / [104]      |
| FastDPM [105]                        |                   | / Improved DDPM [61] |
|                                           | Analytic-DPM [107] / SN&PDR-DDPM [108] |
| Speed-Prior                          |                   |                      |
| Dynamic Programming                  |                   |                      |
| GAN-based: TDDPM [99] / Denoising GAN [63] |
| Acceleration Mixture                 |                   |                      |
| VAE-based: DiffuseVAE [116] / ES-DDPM [101] |
| Flow-based: DiffFlow [117]           |                   |                      |
| Mixed-Modeling                       |                   |                      |
| Expressiveness Mixture               |                   |                      |
| LSGM [118] / Score-flow               |                   |                      |
| PDM [120], INDM [121]               |                   |                      |
| FastDPM [105] / Efficient Sampling   |                   |                      |
| VDM [67], PNDM [76], edm [112]      |                   |                      |
| Reformulation: Score SDE [66] / gDDIM [110] |
| Connection: Cold Diffusion           |                   |                      |
| GCDM [119], DMM [120], [126], [128] |                   |                      |
| Score-Diffusion Unification          |                   |                      |
| Unification                          |                   |                      |
| Non-linear                           |                   |                      |
| Continuous Space                     |                   |                      |
| LSGM [118] / Score-flow               |                   |                      |
| PDM [120], INDM [121]               |                   |                      |
| Others                               |                   |                      |
| Point Cloud                          |                   |                      |
| LSGM [118] / Score-flow               |                   |                      |
| PDM [120], INDM [121]               |                   |                      |
| Discrete Space                       |                   |                      |
| Categorical                          |                   |                      |
| D3PM [65] / Argmax [134] / ARDM [135] / [136] |
| Vector Quantized                     |                   |                      |
| VQ-diffusion [137] / Improved VQ-Diffusion [138] |
| Constrained Space                    |                   |                      |
| Manifold                             |                   |                      |
| RCGSM [144], PNDM [76], RDM [145]   |                   |                      |
| Boomerang [146] / [147]             |                   |                      |
| Graphs                               |                   |                      |
| EDP-GNN [148], Graph GDP [149]      |                   |                      |
| Improved ELBO                        |                   |                      |
| Likelihood Optimization              |                   |                      |
| Score Connection                     |                   |                      |
| Improved DDPM [61] / FastDPM [105]  |                   |                      |
| VDM [67], D3PM [65]                 |                   |                      |
| Re-Design                            |                   |                      |
| Improved DDPM [61] / FastDPM [105]  |                   |                      |
| VDM [67], D3PM [65]                 |                   |                      |
| Variational Gap Optimization         |                   |                      |
| Theoretical                          |                   |                      |
| INDIM [120], PDM [121], [129]       |                   |                      |
| Dimensional Reduction                |                   |                      |
| Mixed-Modeling                       |                   |                      |
| Dimensional Projection               |                   |                      |
| LSGM [118], PDM [120], INDM [121]   |                   |                      |
| Dimensional Projection               |                   |                      |
| DVDP [153]                           |                   |                      |
implicit sampler, differential equation solver sampler, and dynamic programming adjustment.

**Analytical Method**

Existing training-free sampling methods take reverse covariance scales as a hand-crafted sequence of noises without considering them dynamically. Starting from KL-divergence optimization, analytical methods set the reverse mean and covariance as optimal solutions. Analytic-DPM \[107\] and extended Analytic-DPM \[108\] jointly propose optimal reverse solutions under correction for each state. Analytical methods enjoy a theoretical guarantee for the approximation error, but they are limited in particular distributions due to the pre-assumptions.

**Implicit Sampler**

Instead of extracting noise information step by step, the implicit sampler follows the jump-step pattern using knowledge from the pre-trained diffusion model. It follows the assumption that information from multi-step intervals can be reversed along a determinant trajectory without randomness during sampling. Song et al. \[77\] proposes DDIM, which follows the discrete pattern of probability flow ODE with Neural ODE formulation \[75\]:

\[
\frac{dx(t)}{dt} = \frac{t}{\sqrt{\sigma^2(t) + 1}} \frac{\partial^2 f}{\partial x^2} \tag{36}
\]

where \(\sigma_t\) is parameterized by \(\sqrt{1 - \alpha_t / \sqrt{\alpha_t}}\), and \(x\) is parameterized as \(\sqrt{x / \sqrt{\alpha_t}}\). Besides, the probability can be treated as one kind of Score SDE, which is derived from the discrete formulation:

\[
\frac{\alpha_t - \Delta_t}{\sqrt{\alpha_t - \Delta_t}} = \frac{\alpha_t}{\sqrt{\alpha_t}} + \left( \frac{1 - \alpha_t - \Delta_t}{\alpha_t - \Delta_t} - \frac{1 - \alpha_t}{\alpha_t} \right) \frac{\partial^2 f}{\partial x^2} \tag{37}
\]

Besides, the implicit sampler is actually one type of neural ODE solver. On the one hand, part of the methods employ advanced ODE solvers, such as PNDM \[76\], edm \[112\], DEIS \[109\], gDDIM \[110\], and DPM-Solver \[64\]. On the other hand, Watson et al. proposed dynamic programming based jump-step method for sampling the optimal implicit route along the reversed trajectory. Further improved works with strong theoretical support, like manifold hypothesis and sparsity, are expected.

**Differential Equation Solver Sampler**

Differential Equation (DE) Solver Sampler minimizes approximation error during reverse sampling with ODE/SDE-based numerical solvers \[64\]. With a strong assumption on the infinite-time continuous process, differential equation solvers achieve the leading performances. Generally, there are two basic DE formulations: the SDE formulation enjoys randomness when walking in the joint distribution field, ODE formulation with deterministic mapping enjoys faster speed. \[23\] As for the DE solvers \[163\], \[165\], higher-order DE solvers have smaller approximation errors and higher order of convergence, it requires more evaluations \[164\] and suffers from instability issues. In this subsection, we introduce the current algorithms based on the trade-off of different frameworks and DE solvers. We divide them into speed-prior and accuracy-prior.

For the accuracy-prior methods, Itô-Taylor Sampling Scheme \[113\] has been proposed using a high-order SDE solver. Besides, ideal derivative substitution is applied to parameterize the score function in a tricky way that avoids higher-order derivative computing. For the speed-prior methods combining linear solvers and higher-order solvers, Gotta Go Fast \[113\] achieves an algorithm based on directional guidance on step size adjustment. edm \[112\] employs second-order Heun’s solver and adjusted time steps on deterministic diffusion ODE. PNDM \[76\] has explored that different numerical solvers can share the same gradient, leading to exploring the linear multi-step method after using three steps of a higher-order solver (Runge-Kutta method) in Diffusion ODE. Besides, DPM-solver \[64\] proves that a unified solver cross alternative orders may perform better.

\[
dx_t = -\sigma(t) \sigma(t) \nabla_x \log p(x; \sigma(t)) dt + \nabla_x(p(x; \sigma(t))) \nabla_x \log q_t(x_t) dt + \sqrt{2\beta(t) \sigma(t)} \omega_t, \tag{38}
\]

Furthermore, from the perspective of differential equation formulation, DPM-solver, and DEIS \[109\] created a new viewpoint except for SDE and Diffusion ODE. The trade-off extrusion ODE can be seen as a semi-linear form by which the discretization errors are reduced. DEIS improved numerical DDIM with a multi-step PC-sampling method \[161\] with exponential integrator \[165\]. DPM-solver provided a theoretical guarantee on approximation error. Currently, semi-linear-based ODE performs the best but still requires other techniques, such as threshold limit \[109\] and analytical form \[64\].

\[
x_t = f(t) x_t - \frac{1}{2} \sigma^2(t) \nabla_x \log q_t(x_t) \tag{39}
\]

Moreover, Denoising MCMC \[114\] applies truncation idea into differential equation sampling. To accelerate reverse sampling, it conducts differential sampling from data and variance generated by MCMC with a higher speed.

**Dynamic Programming Adjustment**

Dynamic programming (DP) achieves the traversal of all choices to find the optimized solution in a reduced time by memorization technique \[166\], \[167\]. Assuming that each path from one state to another state shares the same KL divergence with others, dynamic programming algorithms explore the optimal traversal along the trajectory. Current DP-based methods \[111\], \[168\] take \(O(T^2)\) of computational cost via optimizing the sum of ELBO losses.

### 3.1.3 Mixed-Modeling

Mixed-modeling applies fast-sampling, and high-expressiveness generative models in diffusion pipeline \[169\], \[170\], \[171\], \[172\]. For diffusion mixed-modeling, diffusion models take the virtue of high-speed sampling of others (such as adversarial training network and autoregressive encoder) and high expressiveness (such as
normalizing flow). Thus, designing mixed models not only performs a promising enhancement but also helps perceive connections between diffusion models and others. Mixed modeling improvement can be classified into two classes from the perspective of mixing purposes: acceleration mixture and expressiveness mixture.

**Acceleration Mixture**

Acceleration mixture aims at applying high-speed generation of VAEs and GANs to save plenty of steps on sampling less perturbed data from random noise. One type of models generate predicted \( x_0 \) with VAE [116] and GAN [63]. Another type of model like ES-DDPM [101] reconstructs intermediate samples as the starting points of the denoised process, which can be viewed as the early stop technique.

\[
x_{t-1} := R_t(x_t, \sigma_t | \tilde{x}) , \quad \tilde{x} := \text{Dec} (\text{Enc}(x_0))
\]

**Expressiveness Mixture**

Expressiveness mixture support diffusion models on expressing data or noise in a different pattern. High-expressiveness data combined with fast-sampling generative models achieve speed-up by obtaining mean and variance more accurately. The high expressiveness methods can be divided into noise modulation, space projection, and kernel expressiveness. As for noise modulation, DiffFlow [117] employs a flow function in each SDE-based diffusion step for noise modulation through a minimizing process w.r.t. KL-Divergence. Benefiting from specific spaces’ properties, space projection methods leverage NFs into data transformation. Both LSGM [118] and PDM [120] obtained latent variables using VAE and flow function, respectively, to take advantage of fast computing. Besides, DiffuseVAE employs VAE to generate condition \( x_0 \) in each sampling step.

\[
\tilde{x} := f_{\phi}^{-1} (R_{T_0}(f_{\phi}(x_0), \sigma_{T_0}), \sigma_{T_0})
\]

**3.1.4 Score & Diffusion Unification**

ScoreSDE [66] first contributed to score & diffusion unification. It builds a unified continuous framework, linking diffusion and perturbation processes to provide a universal tool for generation tasks. Score-diffusion unification models work because the insight from landmark unification helps in exploring efficient sampling mechanisms. Furthermore, generalized works provide multi-view for benefiting diffusion models. There are two categories of works: diffusion reformulation works and diffusion & score connection works.

Reformulation problems unify diffusion pipeline based on one or two variables (such as time \( t \) and signal-to-noise ratio). FastDDPM [105], VDM [67], and f-DM [124] unify DDPM w.r.t. noise schedules by the noise-time bijective map, signal-to-noise ratio, and signal transformation. Generalized DDIM (gDDIM) [110] unifies the DDIM family according to the transition kernel during each step, benefiting implicit acceleration from the bottom. With lighter reparameterization, the methods usually enjoy simplified training and controllable sampling through prejudice optimization. Furthermore, generalized frameworks support exploration of new diffusion pattern.

Connection problems link the score and diffusion frameworks to extend them into a higher view. Gong et al. [127] reveals the hidden connection between score matching with the normalizing flow, expressing score matching by flow ODEs [75, 175]. Bortoli et al. [128] presents a variational score matching approach for simulating diffusion bridges using Doob-h transformation [176], GGDM [111] and DMM [128] generalizes diffusion models with non-Markovian samplers and a vast range of marginal variance to explore formulations of a wider diffusion family. Cold Diffusion...
proposes a unified training and inference framework available for any transition kernels and data distributions. Huang et al. [129] presents a variational form for likelihood estimation, enhancing theoretical support for variational gap minimization. Viewing diffusion model with less presumption, more fundamental setting can be solved and explained.

3.2 Data Structure Diversification

Diffusion methods are mostly conducted on image generation tasks, which limits the high-fidelity generation potential on applications in the other fields. until now, diffusion mechanism is proved to work in inter-disciplinary tasks with diverse data types [48], [55]. More importantly, the traditional patterns of diffusion based on Gaussian perturbed kernels and Gaussian noise prior are expected to be extended for universal practices. To improve diffusion model’s generalization power, we divided the distribution diversification into three aspects: discrete space, continuous space, and Constrained space with structural constraints.

3.2.1 Continuous Space

Non-linear Space

While existing denoising and super-resolution methods handle linear perturbation, non-linear space has great effects on low-level vision tasks such as phase retrieval and non-uniform deblurring. Kawar et al., [131] and DPS [130] apply pseudo-inverse operator and posterior sampling approximation into non-linear noise prediction to solve JPEG artifact correction, image deblurring, and phase retrieval.

Image & Point Cloud

Point cloud generation is first proposed by Luo et al. [33], which generates latent samples for point cloud data, and conducts transformation to obtain high-quality 3d shapes. Other techniques such as [34], [132], [133] accomplish the shape generation and completion tasks similarly. Some slight improvements used in latent space transformation such as canonical map [133], condition feature extraction sub-nets [34], and point-voxel representation [132].

Latent Space

Similar to Expressiveness mixture modeling, latent space data distributions are often processed for diffusion application since different types of complex data structures require a unified approach to generalize and analyze. Most current methods project data into continuous space, obtaining promising performance with the aid of high-quality generation power of diffusion models such as EDM [21] and antigen-diffusion [22]. Thus, latent space processing should be a beneficial pattern utilized in new application fields.

Function

Traditional diffusion processes conducted by Gaussian distribution are limited for some real-world tasks, which leads to continuous function probabilistic modeling. Dutordoir et al., [177] proposes the first diffusion model sampling on the function space. It captures the multi-dimensional distributions by sampling from joint posteriors.

3.2.2 Discrete Space

Deep generative models have many significant achievements in natural language processing [178], [179], multimodal learning [85], [180], and AI for science [181], [182] with relevant architectures and advanced techniques. Among these successes, processing discrete data such as sentences, residue, atom, and vector-quantized data is necessary for eliminating inductive bias. So, based on the previous fortune, it seems that conducting relevant tasks with diffusion models is promising foreground. We divide the main problem into processing text & categorical data, and vector-quantized data.

Text & Categorical

To process categorical features, D3PM [65] first promoted diffusion algorithm onto discrete space to deal with discrete data like sentences and images employing defining w.r.t. categorical distribution Cat():

$$q(x_t | x_{t-1}) = \text{Cat}(x_t; p = x_{t-1}O_t)$$

(42)

Similar to D3PM, multi-nominal diffusion [134] and ARDM [135] extended the categorical diffusion into multi-nominal data for generating language text & segmentation map and Lossless Compression.

Vector-Quantized

To handle the multi-model problem such as text-to-image generation, text-to-3d generation, and text-to-image editing, vector-quantized (VQ) data is proposed to combine data from different fields into the codebook. VQ data processing achieved great performance in autoregressive encoders [183]. Gu et al. [137] first applied diffusion techniques into VQ data, solving unidirectional bias as well as accumulation prediction error problems existing in VQ-VAE. Further text-to-image works such as Cohen et al. [139] & Improved VQ-Diffusion [138], text-to-pose works such as Xie et al. [140] & Guo et al. [141], and text-to-multimodal works such as Weinbach et al. [142] & Xu et al. [143] are built on this core idea. The transition process driven by the probability transition matrix $Q$ and categorical representation vector $v$ is defined by

$$q(x_t | x_{t-1}) = v^T(x_t)Q_tv(x_{t-1})$$

(43)

$$Q_t = \begin{bmatrix} 
\alpha_t + \beta_t & \beta_t & \beta_t & \cdots & 0 \\
\beta_t & \alpha_t + \beta_t & \beta_t & \cdots & 0 \\
\beta_t & \beta_t & \alpha_t + \beta_t & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\gamma_t & \gamma_t & \gamma_t & \cdots & 1 
\end{bmatrix}$$

(44)
3.2.3 Constrained Space

Graph-based neural networks step over traditional data constraints and re-express latent connections among existing data such as social networks [184], molecular [186], [187], and weather conditions [188], [189]. Moreover, manifold learning methods hold the advantages of non-redundant expression and comprehensive portrayals, such as protein and RNA. Thus, constrained space extension methods are based on the Riemann manifold and graph.

Manifold Space

Most current data structures, such as images and video, are defined in a flat-geometry manifold (Euclidean space). However, there exists a series of data in the field of robotics [190], geoscience [191], and protein modeling [192] defined in Riemannian manifold [193], where current methods for Euclidean space cannot capture the high-dimensional Riemann feature. Thus, recent methods RDM [145], RGSM [144], and Boomerang [146] applied diffusion sampling into the Riemannian manifold based on score SDE framework [66]. Besides, there are relevant theoretical works [76], [147] providing comprehensive support for manifold sampling.

Graph

According to [194], graph-based neural networks are becoming an increasingly popular trend due to the high expressiveness in the human pose [141], molecules [195], and proteins [57]. Many current methods apply diffusion theories to graph space. In EDP-GNN [148], Pan et al. [151], and GraphGDP [149], graph data is processed through adjacency matrices for capturing the graph’s permutation invariance. NVDiff [150] reconstructs node positions by reverse SDE.

3.3 Likelihood Optimization

Most variational methods [183], [196] and diffusion methods [68] train models by the principle of variational evidence lower bound (ELBO) since the log-likelihood is not tractable. However, sometimes the log-likelihood still needs to be competitive because the variational gap between ELBO and log-likelihood is not minimized simultaneously. Thus, several methods [61], [118] directly focus on the likelihood optimization problem to solve this problem. And, the solutions can be classified into two classes – improved ELBO and variational gap optimization.

3.3.1 Improved ELBO

Score Connection

Inspired by [197], [198], score connection methods provide a new connection between ELBO optimization and score matching, solving the likelihood optimization problems via improved score training. Score-flow [119] treats the forward KL divergence in ELBO as optimizing a score-matching loss with a weighted scheme. Huang et al. [129] treated Brownian motion as a latent variable to track the log-likelihood estimation explicitly, and it builds the bridge between the estimation and weighted score matching in the variational framework. Analytic-DPM [107] enhances ELBO by analyzing the KL Divergence and reverse covariance & mean. Similarly, NCSN++ [152] bridges the theoretical gap by introducing a truncation factor to ELBO.

\[
E^\infty := \mathbb{E}[-\frac{1}{2} \int_0^T \|a(\omega, s)\|^2_2 \, ds + \log p_0 (x_T) - \int_0^T \nabla \cdot \mu_\theta \, ds \mid x_0 = x] 
\]  
(45)

Re-Design

Compared to loss transformation techniques, re-Design methods directly tighten the ELBO by re-designing noise scale and training objectives. VDM [67] and DDPM++ [152] connect the advanced training objectives concerning signal-to-noise ratio and truncate factors, respectively, optimizing ELBO via finding optimal factors. Improved DDPM [61] and D3PM [65] propose hybrid loss functions based on ELBO with a weighted scheme for improving ELBO.

\[
L_{\text{hybrid}} = L_{\text{simple}} + \lambda L_{\text{vb}} 
\]  
(46)

3.3.2 Variational Gap Optimization

Apart from designing advanced ELBO, minimizing the variational gap is still one approach to maximize the log-likelihood. Based on the success of variational gap optimization [199] in the VAE field, INDM [120] applies the flow model to express the variational gap, minimizing the gap by jointly training the bidirectional flow model and linear diffusion model on latent space. Additionally, PDM accomplishes the variational gap expression by introducing encoder loss of VAE. With collective training, a unique optimal solution exists to eliminate the gap.

\[
L^*_\text{vb} := L_{h_\theta} + L_{\text{vb}} \left( \left[ x^0_T \right]_{t=0}^T ; \theta \right) + L_0 \left( \left[ z^0_T \right]_{t=0}^T ; \theta \right) 
\]  
(47)

3.4 Dimension Reduction

Unlike the variational auto-encoder that projects data into the latent lower dimension, inferencing on the high-dimensional dataset is extremely consuming. However, diffusion models enjoy high expressiveness from equal-dimension transitions considering that dimension reduction may cause the information to be missing. Actually, diffusing on the low-dimensional manifold has wide applications in graph-based representations. Thankfully, reduced-dimension diffusion can be achieved with the aid of latent projection and dimension projection techniques.

Latent Projection

Several mix-modeling methods project training data onto the latent space with a lower dimension by flow function and VAE-encoder, conducting diffusion and denoising processes. LSGM [118], INDM [121], and PDM [120] follow the pattern to learn smoother models in a smaller space, triggering fewer network evaluations and faster sampling [118]. Furthermore, weighting training techniques that use joint training of both diffusion models and projecting models based on ELBO maximization and log-likelihood maximization are employed.

\[
L := L_{\text{Enc}} (z_0 | x) + L_{\text{Dec}} (x | z_0) + L_{\text{simple}} \left( \left[ x^0_T \right]_{t=0}^T ; \theta \right) 
\]  
(48)
Dimension Projection
Dimension projection aims to clear spatial redundancy on image manifolds by decomposing the invertible signal by multiple orthogonal ones. DVDP [153] conducts subspace inference during perturbation and reconstruction, which can be seen as a mixture of DDPM and VAE. Besides, the theoretical analysis behind the reduction scale of dimensionality and down-sampling & up-sampling steps are worth to be explored.

4 APPLICATION
Benefiting from the powerful ability to generate realistic samples, diffusion models have been widely used in various fields such as computer vision [200], natural language processing, and bioinformatics.

4.1 Computer vision
4.1.1 Low-level vision
CMDE [27] empirically compared score-based diffusion methods in modeling conditional distributions of visual image data and introduced a multi-speed diffusion framework. By leveraging the controllable diffusion speed of the condition, CMDE outperformed the vanilla conditional denoising estimator [69] in terms of FID scores in in-painting and super-resolution tasks. DDRM [201] proposed an efficient, unsupervised posterior sampling method served for image restoration. Motivated by variational inference, DDRM demonstrated successful applications in super-resolution, deblurring, inpainting, and colorization of diffusion models. Palette [97] further developed a unified diffusion-based framework for low-level vision tasks such as colorization, inpainting, cropping, and restoration. With its simple and general idea, this work demonstrated the superior performance of diffusion models compared to GAN models.

DiffC [202] proposed an unconditional generative approach that encoded and denoised corrupted pixels with a single diffusion model, which showed the potential of diffusion models in lossy image compression. SRDiFf [29] exploited the diffusion-based single-image super-resolution model and showed competitive results. RePaint [203] was a free-form inpainting method that directly employed a pre-trained diffusion model as the generative prior and only replaced the reverse diffusion by sampling the unmasked regions using the given image information. Though there was no modification to the vanilla pre-trained diffusion model, this method was able to outperform autoregressive and GAN methods under extreme tasks.

4.1.2 High-level vision
FSDM [30] was a few-shot generation framework based on conditional diffusion models. Leveraging advances in vision transformers and diffusion models, FSDM can adapt quickly to various generative processes at test-time and performs well under few-shot generation with strong transfer capability. CARD [31] proposed classification and regression diffusion models, combining a denoising diffusion-based conditional generative model and a pre-trained conditional mean estimator to predict data distribution under given conditions. Though approaching supervised learning from a conditional generation perspective and training with objectives indirectly related to the evaluation metrics, CARD presented a strong ability in uncertainty estimation with the help of diffusion models. Motivated by CLIP [204], GLIDE [85] explored realistic image synthesis conditioned on the text and found that diffusion models with classifier-free guidance yielded high-quality images containing a wide range of learned knowledge. DreamFusion [205] extends GLIDE’s achievement into 3D space. To obtain expressive generative models within a smooth and limited space, LSGM [118] built a diffusion model trained in the latent space with the help of a variational autoencoder framework. SegDiff [206] extended diffusion models for performing image-level segmentation by summing up feature maps from a diffusion-based probabilistic encoder and an image feature encoder. Video diffusion [26], on the other hand, extended diffusion models in the time axis and performed video-level generation by utilizing a typically designed reconstruction-guided conditional sampling method. VQ-Diffusion [32] improved vanilla vector quantized diffusion by exploring classifier-free guidance sampling for discrete diffusion models and presenting a high-quality inference strategy. This method showed superior performance on large datasets such as ImageNet [207] and MSCOCO [208]. Diff-SCM [209] built a deep structural model based on the generative diffusion model. It achieved counterfactual estimation by inferring latent variables with deterministic forward diffusion and intervening in the backward process.

4.1.3 3D vision
[33] was an early work on diffusion-based 3D vision tasks. Motivated by the non-equilibrium thermodynamics, this work analogized points in point clouds as particles in a thermodynamic system and employed the diffusion process in point cloud generation, which achieved competitive performance. PVD [210] was a concurrent work on diffusion-based point cloud generation but performed unconditional generation without additional shape encoders, while a hybrid and point-voxel representation was employed for processing shapes. PDR [34] proposed a paradigm for diffusion-based point cloud completion that applied a diffusion model to generate a coarse completion based on the partial observation and refined the generated output by another network. To deal with point cloud denoising, [35] introduced a neural network to estimate the score of the distribution and denoised point clouds by gradient ascent.

4.1.4 Video modeling
Video diffusion [26] introduced the advances in diffusion-based generative models into the video domain. RVD [211] employed diffusion models to generate a residual to a deterministic next-frame prediction conditioned on the context vector. FDM [212] applied diffusion models to assist long video prediction and performed photo-realistic videos.

MCVD [36] proposed a conditional video diffusion framework for video prediction and interpolation based on masking frames in a blockwise manner. RaMVID [37] extended image diffusion models to videos with 3D convolutional neural networks and designed a conditioning technique for video prediction, infilling, and upsampling.
4.1 Medical application

It is a natural choice to apply diffusion models to medical images. Score-MRI [38] proposed a diffusion-based framework to solve magnetic resonance imaging (MRI) reconstruction. [213] was a concurrent work but provided a more flexible framework that did not require a paired dataset for training. With a diffusion model trained on medical images, this work leveraged the physical measurement process and focused on sampling algorithms to create image samples that are consistent with the observed measurements and the estimated data prior. R2D2+ [214] combined diffusion-based MRI reconstruction and super-resolution into the same network for end-to-end high-quality medical image generation. [215] explored the application of the generative diffusion model to medical image segmentation and performed counterfactual diffusion.

4.2 Sequential modeling

4.2.1 Natural language processing

Benefited by the non-autoregressive mechanism of diffusion models, Diffusion-LM [39] took advantage of continuous diffusions to iteratively denoise noisy vectors into word vectors and performed controllable text generation tasks. Bit Diffusion [40] proposed a diffusion model for generating discrete data and was applied to image caption tasks.

4.2.2 Time series

To deal with time series imputation, CSDI [41] utilized score-based diffusion models conditioned on observed data. Inspired by masked language modeling, a self-supervised training procedure was developed that separates observed values into conditional information and imputation targets. SSSD [42] further introduced structured state space models to capture long-term dependencies in time series data. CSDE [254] proposed a probabilistic framework to model stochastic dynamics and introduced Markov dynamic programming and multi-conditional forward-backward losses to generate complex time series.

4.3 Audio

WaveGrad [43] and DiffWave [14] were seminal works that applied diffusion models to raw waveform generation and obtained superior performance. GradTTS [45] and DiffTTS [265] also implemented diffusion models but generated mel feature instead of raw waves. DiffVC [264] further challenged the one-shot many-to-many voice conversion problem and developed a stochastic differential equation solver. DiffSinger [46] extended the common sound generation to singing voice synthesis based on a shallow diffusion mechanism. DiffSound [259] proposed a sound generation framework conditioned on the text that employed a discrete diffusion model to replace the autoregressive decoder to overcome the unidirectional bias and accumulated errors. EditTTS [47] was also a diffusion-based audio model for the text-to-speech task. Through coarse perturbations in the prior space, desired edits were induced during denoising reversal. Guided-TTS [48] and Guided-TTS2 [49] were also an early series of text-to-speech models that successfully applied diffusion models in sound generation.

4.4 AI for science

4.4.1 Molecular conformation generation

ConfGF [268] was an early work on diffusion-based molecular conformation generation models. While preserving rotation and translation equivariance, ConfGF generated samples by Langevin dynamics with physically inspired gradient fields. However, ConfGF only modeled local distances between the first-order, the second-order, and the third-order neighbors and thus failed to capture long-range interactions between non-bounded atoms. To tackle this challenge, DGM [53] proposed to dynamically construct molecular graph structures between atoms based on their spatial proximity. GeoDiff [54] found that the model was fed with perturbed distance matrices during diffusion learning, which might violate mathematical constraints. Thus, GeoDiff introduced a roto-translational invariant Markov process to impose constraints on the density. EDM [21] further extended the above methods by incorporating discrete atom features and deriving the equations required for likelihood computation. Torsional diffusion [55] operated on the space of torsional angles and produced molecular conformations according to a diffusion process limited to the most flexible degrees of freedom. Based on previous geometric deep learning methods, DiffDock [271] conducts denoised score matching on transition, rotation, and torsion angle to generate drug conformation in protein-ligand complexes.

4.4.2 Material design

CDVAE [269] explored the periodic structure of stable material generation. To address the challenge that stable materials exist only in a low-dimensional subspace with all possible periodic arrangements of atoms, CDVAE designed a diffusion-based network as a decoder with output gradients leading to local minima of energy and updated atom types to capture specific local bonding preferences depending on the neighbors. Inspired by the recent success of antibody modeling [277], [278], [279], the recent work [56] developed...
| **Table 3**
| Classification of Diffusion-based model Applications |

| **Diffusion Application** | **Low-level Vision** | **High-level Vision** |
|---------------------------|----------------------|----------------------|
| **Generation**            | DDPM [68]/Score SDE [66]/CMDE [27] | Luo et al. [33]/PVD [210]/PDR [34]/LION Latent-NeRF [235]/3D-LDM [236]/ |
| **Inpainting & Reconstruction** | RePaint [203]/DDRM [201]/SRDiff [29]/DriftRec [217]/JDDRM [351]/DiffGar [218] | SegDiff [206]/DiffusionDet [91]/Label-Seg [92]/Multi-class Seg [228]/DiffusionInst [229]/Impact Seg [230] |
| **Denoising & Deblurring** | Whang et al. [219]/Batzolıs [220]/Ren et al. [221]/Wang et al. [222]/Voltri et al. [223]/GenIE [224] | PsDM [30]/CARD [31]/GLIDE [85]/Photorealistic [231]/VQ-Diffusion [32]/Improved VQ [138]/DiffusionCLIP [232]/3DDesigner [233] |
| **Translation**           | Palette [97]/Sinfusion [225]/Kwon et al. [226]/IVLR [96]/Batzolis et al. [27]/Dual Translation [227] | DiffAnoDiff [251]/DiffAno [89]/MedSegDiff [252]/DiffSeg [230] |
| **Detection & Segmentation** | Palette [97]/Sinfusion [225]/Kwon et al. [226]/IVLR [96]/Batzolis et al. [27]/Dual Translation [227] | DiffAnoDiff [251]/DiffAno [89]/MedSegDiff [252]/DiffSeg [230] |
| **Multi-modal Generation & Editing** | Palette [97]/Sinfusion [225]/Kwon et al. [226]/IVLR [96]/Batzolis et al. [27]/Dual Translation [227] | DiffAnoDiff [251]/DiffAno [89]/MedSegDiff [252]/DiffSeg [230] |
| **3D Vision**             | PVD [210]/PDR [34]/Luo et al. [35]/Cheng et al. [133]/ReFu [238]/ReFu [239] | PVD [210]/PDR [34]/Luo et al. [35]/Cheng et al. [133]/ReFu [238]/ReFu [239] |
| **Video Modeling**        | PhysDiff [240]/DiffPose [241]/Diffpose [242]/DreamFusion [205]/SDFusion [244] | VDM [26]/MCVD [36]/RVD [211]/FDM [212]/RaMViD [37]/VIDM [245]/Make-A-Video [246] |
| **Medical Imaging**       | Self-Score [248]/CCDF [95]/Inverse-Score [213]/CCDF [95]/ | Score-MRI [38]/Inverse-Score [213]/R2D2+ [214]/ |
| **Sequential Modeling**   | Argmax [134] | Argmax [134] |
| **Time Series**           | CSDE [254]/CARD [31]/Time Grad [255] | CSDE [254]/CARD [31]/Time Grad [255] |
| **Generation & Enhancement** | Diffusion-LM [39]/Bit Diffusion [40]/D3PM [65] | Diffusion-LM [39]/Bit Diffusion [40]/D3PM [65] |
| **Audio & Speech**        | NU-Wave [256]/SRT-Net [257]/DDSP [258] | NU-Wave [256]/SRT-Net [257]/DDSP [258] |
| **Text to Speech**        | EdiTTS [47]/Diff-TTS [266]/Guided-TTS [48]/Guided-TTS2 [49] | EdiTTS [47]/Diff-TTS [266]/Guided-TTS [48]/Guided-TTS2 [49] |
| **Molecule & Protein Design** | ConfGF [268]/DGSM [53]/GeoDiff [54]/EDM [21]/CDVAE [269] | ConfGF [268]/DGSM [53]/GeoDiff [54]/EDM [21]/CDVAE [269] |
| **AI for Science**        | DiffSBDD [272]/Luo et al. [56]/Anand et al. [57]/Levkovitch et al. [266]/SpecGrad [50]/ItôTTS [24]/InferGrad [267] | DiffSBDD [272]/Luo et al. [56]/Anand et al. [57]/Levkovitch et al. [266]/SpecGrad [50]/ItôTTS [24]/InferGrad [267] |
| **Folding**               | DiffFolding [275]/Trippe et al. [276] | DiffFolding [275]/Trippe et al. [276] |
a diffusion-based generative model that explicitly targeted specific antigen structures and generated antibodies. The proposed method jointly sampled antibody sequences and structures and iteratively generated candidates in the sequence-structure space.

Anand et al. [57] introduced a diffusion-based generative model for both protein structure and sequence and learned the structural information that is equivariant to rotations and translations. ProteinSGM [270] formulated protein design as an image inpainting problem and applied conditional diffusion-based generation to precisely model the protein structure. DiffFolding [275] generates protein backbone concentrating on internal angles by traditional DDPM idea.

5 Conclusions & Discussions
The diffusion model becomes increasingly crucial to a wide range of applied fields. To utilize the power of the diffusion model, this paper provides a comprehensive and up-to-date review of several aspects of diffusion models using detailed insights on various attitudes, including theory, improved algorithms, and applications. We hope this survey serves as a guide for readers on diffusion model enhancement and its application.

6 Limitations & Further Directions
Attention on diffusion model class: Most existing improvements and application algorithms are based on the original setting as DDPM. However, many aspects are ignored by researchers concerning the generalized setting of diffusion-based models. Further meaningful works that explore prior distribution, transition kernel, sampling algorithm, and diffusion schemes are expected. Diffusion models should be viewed as a class, but not a branch of DDPM-based models.

Training objective & evaluation metric: Most diffusion-based models set training objectives as evidence of lower bound (ELBO) of negative log-likelihood. However, we have no clear theory that ELBO and NLL are optimized simultaneously. Therefore, the inconsistency may lead to a hidden mismatch between the real goal and the practical refinement of designing. Consequently, further analytical approaches linking log-likelihood optimization to existing variables or creating novel training objectives consistent with the likelihood may guide a significant enhancement of the model’s performance. Furthermore, current evaluation metrics like FID and IS scores cannot perfectly match the primary goals since data distributions are not equivariant to likelihood matching. The ideal evaluation metric should test the sample diversity as well as the recovery effect of diffusion models. A diversity score considering enough classes like CLIP [204] may be an available solution. A recovery score considering real-world data on the manifold for distribution distance will describe the model’s generative ability more accurately and comprehensively. To sum up, the training objective and evaluation metric need to follow the initial goal.

Application and inductive bias: Various fields such as AI for science and natural language processing achieve significant progress with the aid of generative models but require complex modeling to produce the inductive bias. There is a range of tasks that still require refinement with diffusion models to obtain better performance than existing generative networks. For current tasks based on diffusion models, the corresponding frameworks are dominated by score-based networks and DDPM. Accordingly, improvement algorithms with reduced steps should draw much attention, which is one of our motivations for this survey.

Acknowledgments
This work is supported by the Science and Technology Innovation 2030 - Major Project (No. 2021ZD0150100) and the National Natural Science Foundation of China (No. U21A20427). We appreciate Dr. Jingdong Wang provided the idea for our classification figures.

References
[1] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” in ICML, 2014. [document]
[2] C. Doersch, “Tutorial on variational autoencoders,” arXiv preprint arXiv:1606.05908, 2016. [document]
[3] D. P. Kingma, M. Welling et al., “An introduction to variational autoencoders,” Foundations and Trends® in Machine Learning, 2019. [document]
[4] A. Oussidir and A. Elhassouny, “Deep generative models: Survey,” in ISCV. IEEE, 2016. [document]
[5] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, “A tutorial on energy-based learning,” Predicting structured data, 2006. [document]
[6] J. Ngiam, Z. Chen, P. W. Koh, and A. Y. Ng, “Learning deep energy models,” in ICML, 2011. [document]
[7] A. G. ALIAS PARTH GOYAL, N. K. Ke, S. Ganguli, and Y. Bengio, “Variational walkback: Learning a transition operator as a stochastic recurrent net,” NIPS, vol. 30, 2017. [document]
[8] T. Kim and Y. Bengio, “Deep directed generative models with energy-based probability estimation,” arXiv preprint arXiv:1606.03439, 2016. [document]
[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” Communications of the ACM, 2020. [document]
[10] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Ser goupta, and A. A. Bharath, “Generative adversarial networks: An overview,” IEEE Signal Process, 2018. [document]
[11] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye, “A review on generative adversarial networks: Algorithms, theory, and applications,” IEEE TKDE, 2021. [document]
[12] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” arXiv preprint arXiv:1411.1784, 2014. [document]
[13] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” in ICML, 2017. [document]
[14] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real nvp,” arXiv preprint arXiv:1605.08803, 2016. [document]
[15] D. Rezende and S. Mohamed, “Variational inference with normalizing flows,” in ICML, 2015. [document]
[16] I. Kobyzev, S. J. Prince, and M. A. Brubaker, “Normalizing flows: An introduction and review of current methods,” IEEE TPAMI, 2020. [document]
[17] S. Bond-Taylor, A. Leach, Y. Long, and C. Willcocks, “Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models.” IEEE TPAMI, 2021. [document]
[18] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, “Normalizing flows for probabilistic modeling and inference.” J. Mach. Learn. Res., vol. 22, no. 57, pp. 1–64, 2021. [document]
[19] C. Winkler, D. Worrall, E. Hoogeboom, and M. Welling, “Learning likelihoods with conditional normalizing flows,” arXiv preprint arXiv:1912.00042, 2019. [document]
[48x477]

[89x280]

[92x440]

[96x760]

[99x590]

[101] Z. Lyu, X. Xu, C. Yang, D. Lin, and B. Dai, “Accelerating diffusion models via early stop of the diffusion process,” arXiv preprint arXiv:2205.12534, 2022. 3.1.1, 3.1.2, 3.2.1, 4.1.2, 3, 7, 8

[102] G. Daras, M. Delbracio, H. Talebi, A. G. Dimakis, and P. Milanfar, “Soft diffusion: Score matching for general corruptions,” arXiv preprint arXiv:2209.05442, 2022. 3.1.1, 3.1.2

[103] G. Franzese, S. Rossi, L. Yang, A. Finamore, D. Rossi, M. Filippone, and P. Michiardi, “How much is enough? a study on diffusion times in score-based generative models,” 2022. Available: https://arxiv.org/abs/2206.05173

[104] V. Karras, M. Aittala, T. Aila, and S. Laine, “Elucidating the dependence of diffusion models on manifold,” arXiv preprint arXiv:2202.09778, 2022. 2.2.3, 2.3.1, 2, 3.1.2, 3.2.3, 4, 8

[105] Z. Kong and W. Ping, “On fast sampling of diffusion probabilistic models,” arXiv preprint arXiv:2106.00132, 2021. 3.1.1, 3.1.2, 3.1.4, 7, 8

[106] C.-H. Chao, W.-F. Sun, B.-W. Cheng, and C.-Y. Lee, “Quasi-random sampling of diffusion probabilistic models,” arXiv preprint arXiv:2209.05586, 2022. 2.4.2, 3.1.1, 3.1.2

[107] F. Bao, C. Li, J. Zhu, and B. Zhang, “Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models,” arXiv preprint arXiv:2201.06503, 2022. 3.1.1, 3.1.2

[108] F. Bao, C. Li, J. Sun, J. Zhu, and B. Zhang, “Estimating the optimal covariance with imperfect mean in diffusion probabilistic models,” arXiv preprint arXiv:2206.07309, 2022. 3.1.1, 3.1.3, 5, 6

[109] J. Wolke, F. Bieder, R. Sandkühler, and P. C. Cattin, “Dif- fuser models for medical anomaly detection,” arXiv preprint arXiv:2203.04306, 2022. 2.4.2

[110] J. K. Choi, S. Kim, Y. Jeong, Y. Gwon, and S. Yoon, “Itvr: Conditioning method for denoising diffusion probabilistic models,” in CVPR, 2021, pp. 14367–14376.

[111] V. T. Hu, D. W. Zhang, Y. M. Asano, G. J. Burghouts, and C. G. Snoek, “Self-guided diffusion models,” arXiv preprint arXiv:2210.06462, 2022. 3.1.1, 3.1.2

[112] S. Chen, P. Sun, Y. Song, and Y. Luo, “Diffusiondet: Diffus- sion model for object detection,” arXiv preprint arXiv:2211.09788, 2022. 2.4.2

[113] K. Packhäuser, L. Follo, F. Thamm, and A. Maier, “Generation of anonymous chest radiographs using latent diffusion models for training thoracic abnormality classification systems,” arXiv preprint arXiv:2211.01323, 2022. 2.4.2

[114] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McMillan, A. G. Dimakis, and P. Milanfar, “Dpm-solver++: Image generation and editing with text-guided diffusion models,” arXiv preprint arXiv:2201.02388, 2021. 3.1.1, 3.1.2, 3.1.3

[115] H. Zheng, P. He, W. Chen, and M. Zhou, “Truncated diffusion probabilistic models,” arXiv preprint arXiv:2202.09671, 2022. 3.1.1, 3.1.2

[116] G. Daras, M. Delbracio, H. Talebi, A. G. Dimakis, and P. Milanfar, “Soft diffusion: Score matching for general corruptions,” arXiv preprint arXiv:2209.05442, 2022. 3.1.1, 3.1.2

[117] V. Khrulkov and I. Oseledets, “Understanding ddpm latent codes through optimal transport,” arXiv preprint arXiv:2202.07477, 2022.

[118] J. Wolke, F. Bieder, R. Sandkühler, and P. C. Cattin, “Dif- fuser models for medical anomaly detection,” arXiv preprint arXiv:2203.04306, 2022. 2.4.2

[119] J. Song, C. Meng, and S. Ermon, “Denoising diffusion implicit models,” in JMLR, vol. 20, no. 4, 2019.

[120] D. Maoutsa, S. Reich, and M. Opper, “Interacting particle solu- tions of fokker–planck equations through gradient–log–density estimation,” Entropy, vol. 22, no. 8, p. 802, 2020.

[121] R. M. Neal, “Annealed importance sampling,” in JACM, vol. 6, no. 1, pp. 37–49, 1999.

[122] P. Dhariwal and A. Nichol, “Diffusion models beat gans on image synthesis,” NIPS, vol. 34, pp. 8780–8794, 2021.

[123] A. Hyvärinen and P. Dayan, “Estimation of non-normalized density models by score matching,” in Journal of Machine Learning Research, vol. 6, no. 4, 2005.

[124] Y. Song and S. Ermon, “Generative modeling by estimating gradients of the data distribution,” NIPS, vol. 32, 2019.

[125] S. Lyu, “Interpretation and generalization of score matching,” arXiv preprint arXiv:1205.2629, 2012.

[126] L. Arnold, “Stochastic differential equations,” New York, 1974.

[127] B. Oksendal, Stochastic differential equations: an introduction with applications. Springer Science & Business Media, 2007.

[128] A. Babenko, “Label-efficient semantic segmentation with diffusion models,” arXiv preprint arXiv:2205.00132, 2021.

[129] A. Hyvärinen and P. Dayan, “Estimation of non-normalized density models by score matching,” in Journal of Machine Learning Research, vol. 6, no. 4, 2005.

[130] A. Hyvärinen and P. Dayan, “Estimation of non-normalized density models by score matching,” in Journal of Machine Learning Research, vol. 6, no. 4, 2005.

[131] B. Oksendal, Stochastic differential equations: an introduction with applications. Springer Science & Business Media, 2007.

[132] A. Babenko, “Label-efficient semantic segmentation with diffusion models,” arXiv preprint arXiv:2205.00132, 2021.

[133] A. Babenko, “Label-efficient semantic segmentation with diffusion models,” arXiv preprint arXiv:2205.00132, 2021.

[134] A. Babenko, “Label-efficient semantic segmentation with diffusion models,” arXiv preprint arXiv:2205.00132, 2021.

[135] A. Babenko, “Label-efficient semantic segmentation with diffusion models,” arXiv preprint arXiv:2205.00132, 2021.

[136] A. Babenko, “Label-efficient semantic segmentation with diffusion models,” arXiv preprint arXiv:2205.00132, 2021.
[268] C. Shi, S. Luo, M. Xu, and J. Tang, “Learning gradient fields for molecular conformation generation,” in *ICML*. PMLR, 2021, pp. 9558–9568. 

[269] T. Xie, X. Fu, O.-E. Ganea, R. Barzilay, and T. S. Jaakkola, “Crystal diffusion variational autoencoder for periodic material generation,” in *ICLRR*, 2021. 

[270] J. S. Lee and P. M. Kim, “Proteinsgm: Score-based generative modeling for de novo protein design,” *bioRxiv*, 2022. 

[271] G. Corso, H. Stärk, B. Jing, R. Barzilay, and T. S. Jaakkola, “Diffdock: Diffusion steps, twists, and turns for molecular docking,” *arXiv* preprint arXiv:2210.01776, 2022. 

[272] A. Schneuing, Y. Du, C. Harris, A. Jamasb, I. Igashov, W. Du, T. Blundell, P. Lio, C. Gomes, M. Welling et al., “Structure-based drug design with equivariant diffusion models,” *arXiv* preprint arXiv:2210.13695, 2022. 

[273] C. Shi, C. Wang, J. Lu, B. Zhong, and J. Tang, “Protein sequence and structure co-design with equivariant translation,” *arXiv* preprint arXiv:2210.08761, 2022. 

[274] I. Igashov, H. Stärk, C. Vignac, V. G. Satorras, P. Frossard, M. Welling, M. Bronstein, and B. Correia, “Equivariant 3d-conditional diffusion models for molecular linker design,” *arXiv* preprint arXiv:2210.05274, 2022. 

[275] K. E. Wu, K. K. Yang, R. v. d. Berg, J. Y. Zou, A. X. Lu, and A. P. Amini, “Protein structure generation via folding diffusion,” *arXiv* preprint arXiv:2207.12598, 2022. 

[276] T. Fu and J. Sun, “Antibody complementarity determining regions (cdrs) design using constrained energy model,” in *SIGKDD*, 2022, pp. 389–399. 

[277] W. Jin, R. Barzilay, and T. S. Jaakkola, “Antibody-antigen docking and design via hierarchical structure refinement,” in *ICML*. PMLR, 2022, pp. 10 217–10 227. 

[278] J. Ho and T. Salimans, “Classifier-free diffusion guidance,” *arXiv* preprint arXiv:2207.12598, 2022. 

[279] A. Borji, “Pros and cons of gan evaluation measures: New developments,” *Computer Vision and Image Understanding*, vol. 215, p. 103329, 2022. 

[280] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *CVPR*, June 2019. 

[281] Y. LeCun and C. Cortes, “MNIST handwritten digit database.”
Algorithm 1 Ancestral Sampling

\[ x_T \sim N(0, I) \]
for \( i = T, \ldots, 1 \) do
\[ z \sim N(0, I) \]
\[ x_{t-1} = \frac{1}{\sqrt{\sigma_t}} x_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon \theta (x_t, t) + \sigma_t z \]
end for
return \( x_0 \)

Algorithm 2 Annealed Langevin Dynamics Sampling

Initialize \( x_0 \)
for \( i = 1, \ldots, L \) do
\[ \alpha_t \leftarrow e^{-\sigma_t^2/\sigma_i^2} \]
for \( t = 1, \ldots, L \) do
\[ z_t \sim N(0, I) \]
\[ \bar{x}_t = \bar{x}_{t-1} + \alpha_t \bar{s}_t (\bar{x}_{t-1}, \sigma_t) + \sqrt{\alpha_t} z_t \]
end for
\[ x_0 \leftarrow \bar{x}_T \]
end for
return \( \bar{x}_T \)

Algorithm 3 Predictor-Corrector Sampling

\[ x_T \sim N\left(0, \sigma_{\max}^2 I\right) \]
for \( i = N - 1 \) to 0 do
\[ z \sim N(0, I) \]
if Variance Exploding SDE then
\[ x'_i \leftarrow x_{i+1} + \left(\alpha_t^2 - \sigma_t^2\right) s_t (x_{i+1}, \sigma_{i+1}) \]
\[ x_i \leftarrow x'_i + \sqrt{\alpha_t^2 - \sigma_t^2} z \]
else if Variance Preserving SDE then
\[ x'_i \leftarrow \left(2 - \sqrt{1 - \beta_t^2} \right) x_{i+1} + \beta_t s_t (x_{i+1}, i + 1) \]
\[ x_i \leftarrow x'_i + \sqrt{\beta_t^2} z \]
end if
for \( j = 1 \) to \( M \) do
\[ z \sim N(0, I) \]
\[ x_i \leftarrow x_i + \epsilon_j s_t (x_i, \sigma_i) + \sqrt{2\epsilon_i} z \]
end for
end for
return \( x_0 \)

A.2 Conditional Sampling

A.2.1 Labeled Condition

Algorithm 4 Classifier-guided Diffusion Sampling

Input: class label \( y \), gradient scale \( s \)
\[ x_T \sim N(0, I) \]
for \( i = T, \ldots, 1 \) do
if DDPM Sampling then
\[ \mu, \Sigma \leftarrow \mu_{\theta}(x_i), \Sigma_{\phi}(x_i) \]
\[ x_{t-\eta} \leftarrow \text{sample from } N \left( \mu + s \Sigma \nabla_x \log p_{\phi} (y | x_t), \Sigma \right) \]
end if
if DDIM Sampling then
\[ \hat{e} \leftarrow (1 + w) \epsilon_{\theta} (x_t) - \sigma_t \nabla_x \log p_{\phi} (y | x_t) \]
\[ x_{t-1} \leftarrow \sqrt{\alpha_t} \left( x_{t-1} - \sigma_t \hat{e} \right) + \sqrt{1 - \alpha_t} \hat{e} \]
end if
end for
return \( x_0 \)

A.3 Unlabeled Condition

Algorithm 6 Self-guided Conditional Sampling

Input: guidance \( w \), annotation map \( f_{\theta}, g_{\phi} \), dataset \( D \), label \( k \), segmentation label \( k_s \), image guidance \( k \)
\[ x_T \sim N(0, I) \]
for \( i = T, \ldots, 1 \) do
\[ z \sim N(0, I) \]
if Self Guidance then
\[ \hat{e} \leftarrow (1 - w) \epsilon_{\theta} (x_i, t) + w \epsilon_{\theta} (x_i, t; f_{\phi} \left(g_{\phi}(x, D); D\right)) \]
else if Self-Labeled Guidance then
\[ \hat{e} \leftarrow \epsilon_{\theta} (x_i, \text{concat}[t, k]) \]
else if Self-Boxed Guidance then
\[ \hat{e} \leftarrow \epsilon_{\theta} (\text{concat}[x_i, k_s], \text{concat}[t, k]) \]
else if Self-Segmented Guidance then
\[ \hat{e} \leftarrow \epsilon_{\theta} (\text{concat}[k_s, k], \text{concat}[t, k]) \]
end if
\[ x_{t-1} = \frac{1}{\sqrt{\sigma_t}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \hat{e} \right) + \sigma_t z \]
end for
return \( x_0 \)
APPENDIX B

EVALUATION METRIC

B.1 Inception Score (IS)

The inception score is built on valuing the diversity and resolution of generated images based on the ImageNet dataset [281, 282]. It can be divided into two parts: diversity measurement and quality measurement. Diversity measurement denoted by $p_{IS}$ is calculated w.r.t. the class entropy of generated samples: the larger the entropy is, the more diverse the samples will be. Quality measurement denoted by $q_{IS}$ is computed through the similarity between a sample and the related class images using entropy. It is because the samples will enjoy high resolution if they are closer to the specific class of images in the ImageNet dataset. Thus, to lower $q_{IS}$ and higher $p_{IS}$, the KL divergence [283] is applied to inception score calculation:

$$IS = D_{KL}(p_{IS} \parallel q_{IS})$$

$$= \mathbb{E}_{x \sim p_{IS}} \left[ \log \frac{p_{IS}}{q_{IS}} \right]$$

$$= \mathbb{E}_{x \sim p_{IS}} \left[ \log(p_{IS}) - \log(q_{IS}) \right]$$

(B.1)

B.2 Frechet Inception Distance (FID)

Although there are reasonable evaluation techniques in the Inception Score, the establishment is based on a specific dataset with 1000 classes and a trained network that consists of randomness such as initial weights, and code framework. Thus, the bias between ImageNet and real-world images may cause an inaccurate outcome. Furthermore, the number of sample batches is much less than 1000 classes, leading to a value.

FID is proposed to solve the bias from the specific reference datasets. The score shows the distance between real-world data distribution and the generated samples using the mean and the covariance [284].

$$FID = \left\| \mu_r - \mu_g \right\|^2 + \text{Tr} \left( \Sigma_r + \Sigma_g - 2 \left( \Sigma_r \Sigma_g \right)^{1/2} \right)$$

(B.2)

where $\mu_r, \Sigma_r$ are the mean and covariance of generated samples, and $\mu_r, \Sigma_r$ are the mean and covariance of real-world data.

B.3 Negative Log Likelihood (NLL)

According to Razavi et al., [285] negative log-likelihood is seen as a common evaluation metric that describes all modes of data distribution. Lots of works on normalizing flow field [286], [287] and VAE field [288], [289] uses NLL as one of the choices for evaluation. Some diffusion models like improved DDPM [61] regard the NLL as the training objective.

$$\text{NLL} = \mathbb{E} \left[ - \log p_{\theta}(x) \right]$$

APPENDIX C

BENCHMARKS

The benchmarks of landmark models along with improved techniques corresponding to FID score, Inception Score, and NLL are provided on diverse datasets which includes CIFAR-10 [290], ImageNet [207], and CelebA-64 [291]. In addition, some dataset-based performances such as LSUN [292], FFHQ [293], and MNIST [294] are not presented since there is much less experiment data. The selected performance are listed according to NFE in descending order to compare for easier access.

C.1 Benchmarks on CelebA-64

| Method                | NFE  | FID  | NLL  |
|-----------------------|------|------|------|
| NPR-DDPM [108]        | 1000 | 3.15 | -    |
| SN-DDPM [108]         | 100  | 2.90 | -    |
| NCSN [69]             | 1000 | 10.23| -    |
| NCSN ++ [152]         | 1000 | 1.92 | 1.97 |
| DDPM ++ [152]         | 1000 | 1.90 | 2.10 |
| DiffuseVAE [116]      | 1000 | 4.76 | -    |
| Analytic DPM [107]    | 1000 | 2.66 | -    |
| ES-DDPM [101]         | 200  | 2.35 | -    |
| PNDM [76]             | 200  | 2.71 | -    |
| ES-DDPM [101]         | 100  | 3.01 | -    |
| PNDM [76]             | 100  | 2.81 | -    |
| Analytic DPM [107]    | 100  | 4.27 | -    |
| NPR-DDPM [108]        | 100  | 3.04 | -    |
| ES-DDPM [101]         | 50   | 3.97 | -    |
| PNDM [76]             | 50   | 3.34 | -    |
| NPR-DDPM [108]        | 50   | 6.04 | -    |
| SN-DDDM [108]         | 50   | 3.83 | -    |
| DPM-Solver Discrete [64] | 36  | 2.71 | -    |
| ES-DDPM [101]         | 20   | 4.90 | -    |
| PNDM [76]             | 20   | 5.51 | -    |
| DPM-Solver Discrete [64] | 20  | 2.82 | -    |
| ES-DDDP [101]         | 10   | 4.64 | -    |
| PNDM [76]             | 10   | 7.71 | -    |
| Analytic DPM [107]    | 10   | -    | 2.97 |
| NPR-DDPM [108]        | 10   | 28.37| -    |
| SN-DDDM [108]         | 10   | 20.60| -    |
| NPR-DDPM [108]        | 10   | 14.98| -    |
| SN-DDDM [108]         | 10   | 10.20| -    |
| DPM-Solver Discrete [64] | 10  | 6.92 | -    |
| ES-DDDP [101]         | 5    | 9.15 | -    |
| PNDM [76]             | 5    | 11.30| -    |

C.2 Benchmarks on ImageNet-64

| Method                | NFE  | FID  | IS   | NLL  |
|-----------------------|------|------|------|------|
| MCG [295]             | 1000 | 25.4 | -    | -    |
| Analytic DPM [107]    | 1000 | -    | 3.61 | -    |
| ES-DDDP [101]         | 900  | 2.07 | 55.29| -    |
| Efficient Sampling [111] | 256 | 3.87 | -    | -    |
| Analytic DPM [107]    | 200  | -    | 3.64 | -    |
| NPR-DDPM [108]        | 200  | 16.96| -    | -    |
| SN-DDPM [108]         | 200  | 16.61| -    | -    |
| ES-DDDP [101]         | 100  | 3.75 | 48.63| -    |
| DPM-Solver Discrete [64] | 25  | 3.75 | 48.63| -    |
| ES-DDDP [101]         | 25   | 18.4 | 18.12| -    |
| Analytic DPM [107]    | 25   | -    | 3.83 | -    |
| NPR-DDPM [108]        | 25   | 28.27| -    | -    |
| SN-DDDM [108]         | 25   | 27.58| -    | -    |
| DPM-Solver Discrete [64] | 20  | 18.53| -    | -    |
| ES-DDDP [101]         | 10   | 8.93 | 48.81| -    |
| GGD [115]             | 10   | 37.32| 14.76| -    |
| Analytic DPM [107]    | 10   | 24.4 | -    | -    |
| SN-DDDM [108]         | 10   | 4.25 | 48.04| -    |
| ES-DDDP [101]         | 5    | 55.14| 12.9 | -    |
### C.3 Benchmarks on CIFAR-10 Dataset

#### TABLE 6

| Method                  | NFE | FID   | IS     | NLL     |
|-------------------------|-----|-------|--------|---------|
| Improved DDPM [61]      | 4000 | 2.90  | -      | -       |
| VE SDE [66]            | 2000 | 2.20  | 9.89   | -       |
| VP SDE [66]            | 2000 | 2.41  | 9.68   | 3.13    |
| sub-VP SDE [66]        | 2000 | 2.41  | 9.57   | 2.92    |
| DDPM [65]              | 1000 | 3.17  | 9.46   | 3.72    |
| NCSN [69]              | 1000 | 25.32 | -      | -       |
| NCSNv2 [296]           | 1000 | 10.87 | 8.40   | -       |
| Efficient Sampling [113] | 1000 | 2.94  | -      | -       |
| NCSN++ [152]           | 1000 | 2.33  | 10.11  | 3.04    |
| DDPM++ [152]           | 1000 | 2.47  | 9.76   | 2.91    |
| NPR-DDPM [108]         | 2000 | -     | -      | 3.59    |
| SN-DDPM [108]          | 2000 | -     | -      | -       |
| Gotta Go Fast VP [109] | 1000 | 2.49  | -      | -       |
| Gotta Go Fast VE [109] | 1000 | 3.14  | -      | -       |
| INDM [121]             | 1000 | 2.28  | -      | 3.09    |

#### TABLE 7

| Method                  | NFE | FID   | IS     | NLL     |
|-------------------------|-----|-------|--------|---------|
| Diffusion Step [103]    | 600 | 3.72  | -      | -       |
| ES-DDPM [101]           | 600 | 3.17  | -      | -       |
| Diffusion Step [103]    | 400 | 14.38 | -      | -       |
| Diffusion Step [103]    | 200 | 5.44  | -      | -       |
| NPR-DDPM [108]          | 200 | 4.10  | -      | -       |
| SN-DDPM [108]           | 200 | 3.72  | -      | -       |
| Gotta Go Fast VP [109]  | 180 | 2.44  | -      | -       |
| Gotta Go Fast VE [109]  | 180 | 3.40  | -      | -       |
| LSGM [118]              | 138 | 2.10  | -      | -       |
| Diffusion Step [103]    | 100 | 4.16  | -      | -       |
| FastDDPM [105]          | 100 | 2.86  | -      | -       |
| TDPM [99]               | 100 | 3.10  | 9.34   | -       |
| NPR-DDPM [108]          | 100 | 4.52  | -      | -       |
| SN-DDPM [108]           | 100 | 3.83  | -      | -       |
| Diffusion [117]         | 100 | 11.71 | 8.27   | -       |
| DiffFlow [117]          | 100 | 14.14 | 3.04   | -       |
| Analytic DPM [107]      | 100 | -     | -      | 3.59    |
| Efficient Sampling [113] | 64  | 3.08  | -      | -       |
| DPM-Solver [64]         | 51  | 2.59  | -      | -       |
| DDIM [77]               | 50  | 4.67  | -      | -       |
| FastDDPM [105]          | 50  | 3.2   | -      | -       |
| NPR-DDPM [108]          | 50  | 5.31  | -      | -       |
| SN-DDPM [108]           | 50  | 4.17  | -      | -       |
| Improved DDPM [61]      | 50  | 4.99  | -      | -       |
| TDPM [99]               | 50  | 3.3   | 9.22   | -       |
| DEIS [113]              | 50  | 2.57  | -      | -       |
| gDDIM [110]             | 50  | 2.28  | -      | -       |
| DPM-Solver Discrete [64] | 44  | 3.48  | -      | -       |
| edm [112]               | 35  | 1.79  | -      | -       |
| Efficient Sampling [113] | 32  | 3.17  | -      | -       |
| Improved DDPM [61]      | 25  | 7.53  | -      | -       |
| GGDGM [115]             | 25  | 4.25  | 9.19   | -       |
| NPR-DDPM [108]          | 25  | 7.99  | -      | -       |
| SN-DDPM [108]           | 25  | 6.05  | -      | -       |
| DDIM [77]               | 20  | 6.84  | -      | -       |
| FastDDPM [105]          | 20  | 5.05  | -      | -       |
| DEIS [113]              | 20  | 2.86  | -      | -       |
| DPM-Solver [64]         | 20  | 2.87  | -      | -       |
| DPM-Solver Discrete [64] | 20  | 3.72  | -      | -       |
| Efficient Sampling [113] | 16  | 3.41  | -      | -       |
| NPR-DDPM [108]          | 10  | 19.94 | -      | -       |
| SN-DDPM [108]           | 10  | 16.33 | -      | -       |
| DDIM [77]               | 10  | 13.36 | -      | -       |
| FastDDPM [105]          | 10  | 9.90  | -      | -       |
| GDGD [115]              | 10  | 8.23  | 8.90   | -       |
| Analytic DPM [107]      | 10  | -     | -      | 4.11    |
| DEIS [113]              | 10  | 4.17  | -      | -       |
| DPM-Solver [64]         | 10  | 6.96  | -      | -       |
| DPM-Solver Discrete [64] | 10  | 10.16 | -      | -       |
| Progressive Distillation [62] | 8  | 2.57  | -      | -       |
| Denoising Diffusion GAN [63] | 8  | 4.36  | 9.43   | -       |
| GGDGM [115]             | 5   | 13.77 | 8.33   | -       |
| DEIS [113]              | 5   | 15.37 | -      | -       |
| Progressive Distillation [62] | 4  | 3.00  | -      | -       |
| TDPM [99]               | 4   | 3.41  | 9.00   | -       |
| Denoising Diffusion GAN [63] | 4  | 3.75  | 9.63   | -       |
| Progressive Distillation [62] | 2  | 4.51  | -      | -       |
| TDPM [99]               | 2   | 4.47  | 8.97   | -       |
| Denoising Diffusion GAN [65] | 2  | 4.08  | 9.80   | -       |
| Denoising student [98]  | 2   | 9.36  | 8.36   | -       |
| Progressive Distillation [62] | 1  | 9.12  | -      | -       |
| TDPM [99]               | 1   | 8.91  | 8.65   | -       |
### Table 8
Details for Improved Diffusion Methods

| Method | Year | Data | Model | Framework | Training | Sampling | Code |
|--------|------|------|-------|-----------|----------|----------|------|
| **Landmark Works** | | | | | | | |
| DPM [27] | 2015 | RGB Image | Discrete | Diffusion | L_simple | Ancestral | [code] |
| DDPN [68] | 2020 | RGB Image | Discrete | Diffusion | L_simple | Ancestral | [code] |
| NCSN [67] | 2019 | RGB Image | Discrete | Score | LSDM | Langevin dynamics | [code] |
| NCSN+V | 2020 | RGB Image | Discrete | Score | LSDM | Langevin dynamics | [code] |
| Score SDE [66] | 2020 | RGB Image | Continuous | SDE | LSDM | PC-Sampling | [code] |
| **Improved Works** | | | | | | | |
| Progressive Distill [62] | 2022 | RGB Image | Discrete | Diffusion | L_simple | LSDM Sampling | [code] |
| Demosing Student [88] | 2021 | RGB Image | Discrete | Diffusion | L_simple | LSDM Sampling | [code] |
| TDPM [99] | 2022 | RGB Image | Discrete | Diffusion | L_simple | LSDM Sampling | [code] |
| ES-DDPM [101] | 2022 | RGB Image | Discrete | Diffusion | L_simple | LSDM Sampling | [code] |
| CCDF [95] | 2021 | RGB Image | Discrete | SDE | LSDM | Langevin dynamics | [code] |
| Franzese’s Model [103] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin dynamics | [code] |
| FastDPM [105] | 2021 | RGB Image | Discrete | Diffusion | L_simple | LSDM Sampling | [code] |
| Improved DDPM [61] | 2021 | RGB Image | Discrete | Diffusion | L_simple | LSDM Sampling | [code] |
| VDM [67] | 2022 | RGB Image | Both | Diffusion | L_simple | Ancestral | [code] |
| San-Roman’s Model [106] | 2021 | RGB Image | Discrete | Diffusion | L_simple | LSDM | PC-Sampling | [code] |
| Analytic-DDPM [107] | 2022 | RGB Image | Discrete | Score | Lsquare | Diffusion Sampling | [code] |
| NPR-DDPM [158] | 2022 | RGB Image | Discrete | Diffusion | L_simple | LSDM | PC-Sampling | [code] |
| SN-DDPM [108] | 2022 | RGB Image | Discrete | Score | L_simple | LSDM | PC-Sampling | [code] |
| DDIM [72] | 2021 | RGB Image | Continuous | SDE & ODE | LSDM | Langevin & Flow Sampling | [code] |
| gDDIM [110] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| INDM [121] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Ito-Taylor [23] | 2021 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Gotta Go Fast [109] | 2021 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| DPM-Solver [64] | 2022 | RGB Image | Continuous | ODE | LSDM | Langevin & Flow Sampling | [code] |
| edm [112] | 2022 | RGB Image | Continuous | ODE | LSDM | Langevin & Flow Sampling | [code] |
| PNDRM [76] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| DDSS [115] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| DDIM [111] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Diffusion GAN [63] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| DiffuseVAE [116] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| DiffFlow [117] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| LGSCM [118] | 2021 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Score-flow [119] | 2021 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| PDM [120] | 2021 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Score-Flow [122] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Song’s Model [123] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Huang’s Model [129] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| De Bortoli’s Model [128] | 2021 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| PVD [132] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Luo’s Model [133] | 2021 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Lyu’s Model [127] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| D3PM [65] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Argmax [134] | 2022 | Categorical Data | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| ARDM [135] | 2022 | Categorical Data | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Campbell’s Model [136] | 2022 | Categorical Data | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| VQ-diffusion [137] | 2022 | Vector-Quantized | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Improved VQ-diffusion [138] | 2022 | Vector-Quantized | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Cohen’s Model [139] | 2022 | Vector-Quantized | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| Xie’s Model [140] | 2022 | Vector-Quantized | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| RGSN [144] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| RDM [145] | 2022 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| EDP-CN [146] | 2020 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
| NCSN++ [152] | 2021 | RGB Image | Continuous | SDE | LSDM | Langevin & Flow Sampling | [code] |
## TABLE 9
Details for Diffusion Applications

| Method                        | Year | Data      | Framework | Downstream Task                                      | Code   |
|-------------------------------|------|-----------|-----------|-------------------------------------------------------|--------|
| **Computer Vision**           |      |           |           |                                                       |        |
| CMDE                          | 2021 | RGB-Image | SDE       | Inpainting, Super-Resolution, Edge to image translation | [code] |
| DDMM                          | 2022 | RGB-Image | Diffusion | Super-Resolution, Deblurring, Inpainting, Colorization | [code] |
| Palette                       | 2022 | RGB-Image | Diffusion | Colorization, Inpainting, Uncropping, JPEG Restoration| [code] |
| DiffC                         | 2022 | RGB-Image | SDE       | Compression                                           |        |
| SRDiff                        | 2021 | RGB-Image | Diffusion | Super-Resolution                                      |        |
| RePaint                       | 2021 | RGB-Image | Diffusion | Inpainting, Super-resolution, Edge to Image Translation | [code] |
| FSDM                          | 2022 | RGB-Image | Diffusion | Few-shot Generation                                   |        |
| CARD                          | 2022 | RGB-Image | Diffusion | Conditional Generation                               | [code] |
| GLIDE                         | 2022 | RGB-Image | Diffusion | Conditional Generation                               | [code] |
| LSGM                          | 2022 | RGB-Image | SDE       | UnConditional & Conditional Generation              | [code] |
| SegDiff                       | 2022 | RGB-Image | Diffusion | Segmentation                                          |        |
| VQ-Diffusion                  | 2022 | VQ Data   | Diffusion | Text-to-Image Synthesis                               | [code] |
| DreamFusion                   | 2023 | VQ Data   | Diffusion | Text-to-Image Synthesis                               | [code] |
| Text-to-Sign VQ               | 2022 | VQ Data   | Diffusion | Conditional Pose Generation                          |        |
| Improved VQ-Diff TES          | 2022 | VQ Data   | Diffusion | Text-to-Image Synthesis                               |        |
| Luo’s Model                   | 2021 | Point Cloud| Diffusion | Point Cloud Generation                               | [code] |
| PVD                           | 2021 | Point Cloud| Diffusion | Point Cloud Generation                               | [code] |
| Cheng’s Model                 | 2022 | Point Cloud| Diffusion | Point Cloud Generation                               | [code] |
| Luo’s Model                   | 2022 | Point Cloud| Score     | Point Cloud Denoising                                 |        |
| VDM                           | 2022 | Video     | Diffusion | Text-Conditioned Video Generation                      | [code] |
| RVD                           | 2022 | Video     | Diffusion | Video Forecasting, Video compression                  | [code] |
| FDM                           | 2022 | Video     | Diffusion | Video Forecasting, Long-range Video modeling          |        |
| MCVD                          | 2022 | Video     | Diffusion | Video Prediction, Video Generation, Video Interpolation| [code] |
| RaMvit                        | 2022 | Video     | SDE       | Conditional Generation                               |        |
| Score-MRI                     | 2022 | MRI       | SDE       | MRI Reconstruction                                    |        |
| Song’s Model                  | 2022 | MRI, CT   | SDE       | MRI Reconstruction, CT Reconstruction                |        |
| Pair2                         | 2022 | MRI       | SDE       | MRI Denoising                                         |        |
| **Sequence Modeling**         |      |           |           |                                                       |        |
| Diffusion-LM                  | 2022 | Text      | Diffusion | Conditional Text Generation                          | [code] |
| Bit Diffusion                 | 2022 | Text      | Diffusion | Image-Conditional Text Generation                    | [code] |
| D3PM                          | 2022 | Text      | Diffusion | Text Generation                                       |        |
| Argmax                        | 2021 | Text      | Diffusion | Test Segmentation, Text Generation                    | [code] |
| CSDE                          | 2022 | Time Series| Diffusion | Series Imputation                                    | [code] |
| SSSD                          | 2022 | Time Series| Diffusion | Series Imputation                                    | [code] |
| CSDE                          | 2022 | Time Series| SDE       | Series Imputation, Series Predicton                   | [code] |
| **Audio & Speech**            |      |           |           |                                                       |        |
| WaveGrad                      | 2020 | Audio     | Diffusion | Conditional Wave Generation                          | [code] |
| DiffWave                      | 2021 | Audio     | Diffusion | Conditional & Unconditional Wave Generation          | [code] |
| GradTTS                       | 2021 | Audio     | SDE       | Wave Generation                                       | [code] |
| Diff-TTS                      | 2021 | Audio     | Diffusion | non-AR mel-Spectrogram Generation, Speech Synthesis  | [code] |
| DiffVC                        | 2021 | Audio     | SDE       | Voice conversion                                      | [code] |
| DiffSVC                       | 2022 | Audio     | Diffusion | Voice Conversion                                      | [code] |
| DiffSinger                    | 2021 | Audio     | Diffusion | Singing Voice Synthesis                              | [code] |
| Diffsound                     | 2021 | Audio     | Diffusion | Text-to-sound Generation tasks                       | [code] |
| EdTTS                         | 2021 | Audio     | SDE       | fine-grained pitch, content editing                  | [code] |
| Guided-TTS                    | 2022 | Audio     | SDE       | Conditional Speech Generation                        | [code] |
| Guided-TTS2                   | 2022 | Audio     | SDE       | Spectrograms-Voice Generation                        | [code] |
| Levkovitch’s Model            | 2022 | Audio     | SDE       | Spectrograms-Voice Generation                        | [code] |
| SpecGrd                       | 2020 | Audio     | Diffusion | Spectrograms-Voice Generation                        | [code] |
| ItoTTS                        | 2022 | Audio     | SDE       | Spectrograms-Voice Generation                        | [code] |
| ProDiff                       | 2022 | Audio     | Diffusion | Text-to-Speech Synthesis                             | [code] |
| BinauralGrad                  | 2022 | Audio     | Diffusion | Binaural Audio Synthesis                             | [code] |
| **AI For Science**            |      |           |           |                                                       |        |
| ContGF                        | 2021 | Molecular | Score     | Conformation Generation                              | [code] |
| DGLSM                         | 2022 | Molecular | Score     | Conformation Generation, Sidechain Generation       | [code] |
| GeoDiff                       | 2022 | Molecular | Diffusion | Conformation Generation                              | [code] |
| EDM                           | 2022 | Molecular | SDE       | Conformation Generation                              | [code] |
| Torsional Diff                | 2022 | Molecular | Diffusion | Molecular Generation                                 | [code] |
| DiffWave                      | 2022 | Molecular | Diffusion | Conformation Generation, molecular docking          | [code] |
| CDVAE                         | 2022 | Protein   | Score     | Periodic Material Generation                        | [code] |
| Luo’s Model                   | 2022 | Protein   | Diffusion | CDR Generation                                       | [code] |
| Anand’s Model                 | 2022 | Protein   | Diffusion | Protein Sequence and Structure Generation            |        |
| ProteinSGM                    | 2022 | Protein   | SDE       | de novo protein design                               |        |
| DiffFolding                   | 2022 | Protein   | Diffusion | Protein Inversion Folding                            | [code] |