On the Design Fundamentals of Diffusion Models: A Survey

Ziyi Chang, George A. Koulieris, and Hubert P. H. Shum, *Senior Member, IEEE*

Abstract—Diffusion models are generative models, which gradually add and remove noise to learn the underlying distribution of training data for data generation. The components of diffusion models have gained significant attention with many design choices proposed. Existing reviews have primarily focused on higher-level solutions, thereby covering less on the design fundamentals of components. This study seeks to address this gap by providing a comprehensive and coherent review on component-wise design choices in diffusion models. Specifically, we organize this review according to their three key components, namely the forward process, the reverse process, and the sampling procedure. This allows us to provide a fine-grained perspective of diffusion models, benefiting future studies in the analysis of individual components, the applicability of design choices, and the implementation of diffusion models.

Index Terms—Diffusion Model, Forward Process, Reverse Process, Sampling Procedure, Deep Learning

1 Introduction

Diffusion models are an emerging class of generative models to obtain novel data [1]. During training, a diffusion model is trained by adding and removing noise when given a set of training samples. During inference, it generates a new sample using random noise as input.

Diffusion models have been applied in a wide range of fields. Diffusion models have shown impressive results in inpainting [2], text-to-image [3], super-resolution [4], colorization [5], and instance segmentation [6]. Diffusion models have also demonstrated superiority in speech generation [7], music synthesis [8], and audio enhancement [9]. Besides, they become increasingly popular in 3D shape generation [10], human motion synthesis [11], video synthesis [12], molecule synthesis [13], molecule trajectory prediction [14], and astronomical data synthesis [15].

The generic pipeline of diffusion models involves a forward process and a reverse process to learn a data distribution, as well as a sampling procedure to generate novel data. The pipeline is featured by a time-indexed multi-step chain where the three components move either forwards, i.e., timestep increases, or backwards, i.e., timestep decreases. The forward process moves forwards on the chain and perturbs the training data by adding noise at each timestep. The reverse process moves on the chain in the opposite direction. It optimizes a network to remove the aforementioned perturbation. The two processes jointly enable the network to be trained for distribution modelling. Finally, the sampling procedure obtains a random noise and uses the optimized network for data generation. It also moves backwards on the chain as the reverse process does. The main difference is that the network during the sampling procedure is already optimized and used for inference only. Typically, two distinctive formulations are available for realizing the three components in the generic pipeline. First, diffusion models may be formulated with discrete timesteps [16], [17]. Second, diffusion models may also be formulated with continuous timesteps. These components are defined by stochastic differential equations (SDEs) [5], [18]. This formulation unifies the one with discrete timesteps, and facilitates the theoretical analysis of diffusion models [19].

Although these three components generally define the generic pipeline, they are currently lacking a comprehensive survey. Some existing survey papers focus on higher-level solutions to facilitate applications, and explore less on the design details of each component [20], [21]. Some concentrate on specific domains or aspects of diffusion models, lacking insights on the holistic design fundamentals of the generic pipeline [22], [23], [24], [25], [26], [27]. Others focus on the application side, and thereby provide fewer observations on theoretical designs [28], [29], [30]. Overall, a survey that concentrates specifically on designing the aforementioned components of diffusion models is lacking.

This survey bridges the gap in the literature by offering a thorough and cohesive review of component-wise design fundamentals in diffusion models. In particular, we have organized design fundamentals of diffusion models into the forward process, the reverse process, and the sampling procedure, as shown in Figure 1. This breakdown is aligned with the generic pipeline. Drawing upon the latest research and implementations, we have summarized the main streams of design fundamentals for each component. Overall, this survey offers a fine-grained perspective of diffusion models, and facilitates future analysis of individual components, the applicability of design fundamentals, and the implementation of diffusion models.

The rest of this survey is organised as follows. Section 2 introduces the preliminaries of diffusion models, including the generic pipeline and its two formulations. Sections 3, 4, and 5 respectively review the design fundamentals of the forward process, the reverse process, and the sampling procedure, as visualised in Figure 1. Section 6 provides insights on the future trends and Section 7 gives a brief conclusion on diffusion models.

- *The authors are with the Department of Computer Science, Durham University, Durham, DH1 3LE, United Kingdom.*
- *Email: {ziyi.chang, georgios.a.koulieris, hubert.shum}@durham.ac.uk*
- *Corresponding author: Hubert P. H. Shum*

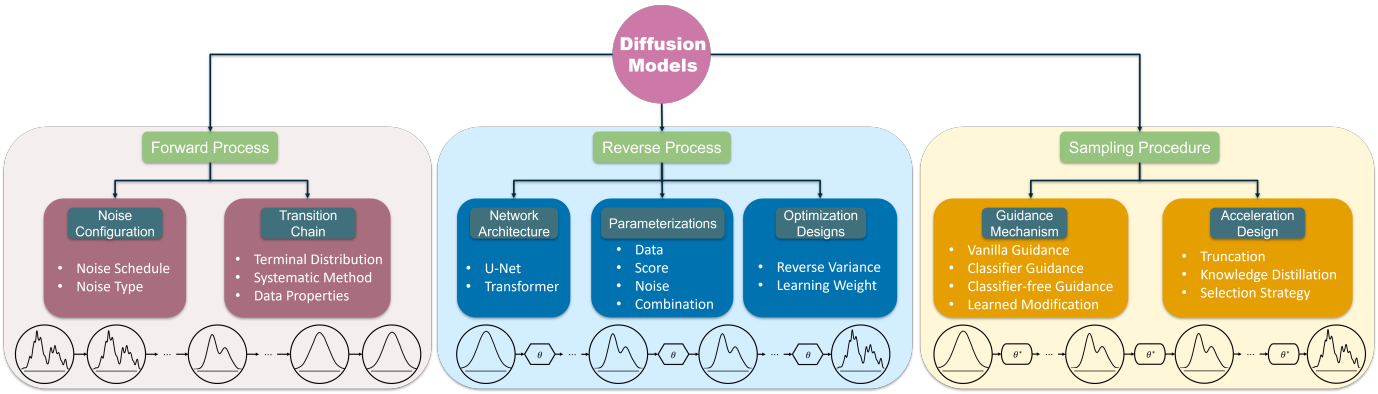


Fig. 1. The overview of diffusion models. The forward process, the reverse process, and the sampling procedure are the three core components of diffusion models, which are responsible for adding noise, training networks, and generating samples, respectively.

2 Preliminaries

This survey uses commonly-used notations and terminologies in existing papers, as shown in Table 1, and represents concepts with figures whose legends are defined in Table 2.

TABLE 1
Notations and terminologies.

Notation	Terminology
$\mathbb{E}$	Expectation
$\mathcal{N}(x;\mu,\Sigma)$	Variable x follows the Gaussian distribution with mean μ and variance Σ
x_0	Training sample
t	Timestep
T	Total number of timesteps
c	Condition
x_t	Perturbed sample at timestep t
ϵ_t	Noise at timestep t
$p(\cdot)$	Distribution
$p(x_0)$	Original distribution at timestep 0
$p(x_t)$	Perturbed distribution at timestep t
$p(x_T)$	Terminal distribution, i.e., data distribution at timestep T
θ	Parameters in denoising network
θ^*	Optimal parameters in denoising network
$p_\theta(\cdot)$	Predicted distribution by denoising network
$p(x_t\|x_{t-1})$	Forward transition
$p_\theta(x_{t-1}\|x_t)$	Reverse transition
$p_{\theta^*}(x_{t-1}\|x_t)$	Sampling transition
$\nabla_x \log p(\cdot)$	Score, i.e., gradient of a log distribution
I	Identity matrix
x_0^*	Newly generated data

2.1 The Generic Pipeline

The generic pipeline in diffusion models is generally specified by the forward process, the reverse process, and the sampling procedure. The generic pipeline is characterized by a timestep-indexed multi-step chain where the three components move forwards or backwards. The forward and reverse processes are implemented jointly to optimize a denoising network by gradually adding and removing noise [17]. After the optimization, the sampling procedure leverages the trained network to generate a novel sample

TABLE 2
Figure legends.

Notations	Meaning
θ	Trainable network with parameters θ
ξ^*	Fixed network with parameters ξ^*
θ^*	Component not in use
	Data distributions
	The distribution at a timestep
	Condition c
	Timestep t
	Combination, e.g., Addition.
p	With probability p for dropping out

$x_0^* \sim p_{\theta^*}(x_0) \approx p(x_0)$ whose distribution $p_{\theta^*}(x_0)$ conforms to the same distribution as the original one $p(x_0)$ [16].

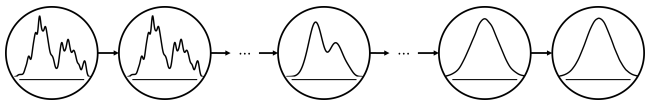


Fig. 2. The forward process perturbs the original unknown distribution by gradually adding noise to a given set of data samples through a chain of distribution transitions with multiple time steps. Each time step of the chain is denoted by a circle.

The forward process perturbs a training sample x_0 to $\{x_t\}_{t=1}^T$ as the timestep t increases, as shown in Figure 2. A forward transition $p(x_t|x_{t-1})$ describes such a perturbation where a small amount of noise ϵ_t is added between two timesteps. In other words, as the forward process moves on the chain, more and more noise is added through $p(x_t|x_{t-1})$, and thereby the perturbed sample x_t becomes noisier and noisier. Through multiple timesteps, the original distribution $p(x_0)$ is eventually perturbed to a tractable terminal distribution $p(x_T)$, which is usually full of noise [31]. Since only noise is added through the chain, the forward process does not require any parameters to be trained. In particular, the forward process is represented as a chain of forward

transitions:

$$p(x_T|x_0) := p(x_1|x_0)\cdots p(x_t|x_{t-1})\cdots p(x_T|x_{T-1}),$$
$$= \prod_{t=1}^{T} p(x_t|x_{t-1}), \tag{1}$$

where t is the timestep, T is the total number of timesteps, x_0 is a training sample at $t = 0$ and is then perturbed to be x_T after T timesteps, and $p(x_t|x_{t-1})$ is a forward distribution transition between two consecutive time steps.

The forward process has both similarities and differences with variational autoencoders (VAEs) [32]. Similar to VAEs, it usually perturbs $p(x_0)$ to the commonly-used isotropic Gaussian distribution $p(x_T) = \mathcal{N}(0, I)$ as the terminal distribution [33]. In contrast to learning an encoder to obtain $p(x_T)$ in VAEs, the forward process has no trainable parameters and only adds noise to x_0 for perturbation [17].

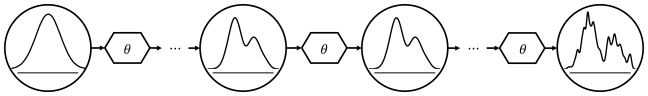


Fig. 3. The reverse process trains a neural network θ to recursively remove the noise that has been previously added by the forward process.

The reverse process trains a denoising network to recursively remove the noise, as shown in Figure 3. Instead of removing all noise in a single timestep like GANs [34], a denoising network is trained to iteratively remove the noise between two consecutive timesteps. The reverse process moves backwards on the multi-step chain as t decreases from T to 0. Such iterative noise removal is termed as the reverse transition $p_\theta(x_{t-1}|x_t)$, which is approximated by optimizing the trainable parameters θ in the denoising network [35]. In particular, the reverse process is formulated as a chain of reverse transitions:

$$p_\theta(x_0) := p(x_T)p_\theta(x_{T-1}|x_T)\cdots p_\theta(x_{t-1}|x_t)\cdots p_\theta(x_0|x_1),$$
$$= p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \tag{2}$$

where θ is the parameters of the denoising network and $p_\theta(x_{t-1}|x_t)$ is the reverse distribution transition. In particular, the reverse process is usually parameterized as:

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \tag{3}$$

where $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ are, respectively, the Gaussian mean and variance to be estimated by the network θ.

The denoising network is trained by the standard variational bound on negative log likelihood:

$$L = \mathbb{E}[D_{KL}\left(p(x_T|x_0)||p(x_T)\right) + \sum_{t\geq 1} D_{KL}\left(p(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)\right) - \log p_\theta(x_0|x_1)], \tag{4}$$

where $D_{KL}(\cdot\|\cdot)$ is the Kullback–Leibler (KL) divergence and computes the difference between two distributions. Overall, minimization of the objective L is to reduce the discrepancy between $p_\theta(x_0)$ and $p(x_0)$.

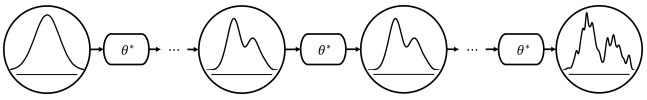


Fig. 4. The sampling procedure uses the trained denoising network θ^* and usually follows the same transitions as the reverse process.

The sampling procedure leverages the optimized denoising network θ^* to generate novel data x_0^*, as illustrated in Figure 4. It moves backwards on the chain to recursively apply the optimized network θ^* [36]. Concretely, it firstly obtains a sample x_T from the terminal distribution $p(x_T)$ and then uses the trained network to iteratively remove noise by the sampling transition $p_{\theta^*}(x_{t-1}|x_t)$. Through a chain of such transitions, it finally generates new data $x_0^* \sim p_{\theta^*}(x_0) \approx p(x_0)$. In particular, the sampling procedure is defined as a chain of sampling transitions:

$$p_{\theta^*}(x_0) := p(x_T)p_{\theta^*}(x_{T-1}|x_T)\cdots p_{\theta^*}(x_0|x_1),$$
$$= p(x_T)\prod_{t=1}^{T} p_{\theta^*}(x_{t-1}|x_t), \tag{5}$$

where θ^* represents the optimized parameters of the denoising network, $p(x_T)$ is the terminal distribution, and $p_{\theta^*}(x_{t-1}|x_t)$ is the sampling transition.

2.2 Discrete and Continuous Formulations

Diffusion models can be formulated in two distinct ways, i.e., discrete and continuous formulations, which differ in the definition of the timesteps [37]. The discrete formulation defines timesteps to be integer values, ranging from 0 to T [17], and results in a finite number of timesteps that is specified by the value of T. In contrast, the continuous formulation defines the timesteps as continuous values, which are constrained within the interval of $[0, 1]$, and allows for an infinite number of timesteps in theory [5].

2.2.1 The Discrete Formulation

Regarding the discrete formulation, denoising diffusion probabilistic model (DDPM) [17] is a popular configuration of such formulated diffusion models. It is straightforward to define, efficient to train, and capable to achieve high quality and high diversity in the generated samples [38].

Concretely, the forward transition in DDPM is defined to add isotropic Gaussian noise $\epsilon_t \sim \mathcal{N}(0, I)$:

$$p(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I), \tag{6}$$

where β_t is the noise schedule, which is a hyper-parameter to control the amount of noise to be added in each timestep. As all forward transitions (Eq. 6) are Gaussian, the forward process (Eq. 1) in DDPM is simplified as:

$$p(x_t|x_0) := \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I), \tag{7}$$

where $\bar{\alpha}_t$ is defined as $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ and $\alpha_t = 1 - \beta_t$. In theory, $\bar{\alpha}_t$ has a similar effect with β_t in Eq. 6. Please refer to appendix A for the detailed derivation of Eq. 7 from Eq. 6.

The reverse process has the same functional form as the forward process [1]. In DDPM configuration, the transition Eq. 3 in the reverse process is formulated as:

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)), \beta_t I), \tag{8}$$

where the variance $\Sigma_\theta(x_t, t)$ in Eq. 3 is empirically fixed as the noise schedule β_t, and $\mu_\theta(x_t, t)$ is reparametrized by the noise prediction $\epsilon_\theta(x_t, t)$. Accordingly, the training objective defined in Eq. 4 is also simplified as:

$$L = \mathbb{E}_{x_t, t}\left[\|\epsilon_t - \epsilon_\theta(x_t, t)\|_2^2\right]. \tag{9}$$

Finally, the sampling process obtains $x_T \sim p(x_T)$, and applies $p_{\theta^*}(x_{t-1}|x_t)$ to generate x_0^*.

2.2.2 The Continuous Formulation

The continuous formulation manipulates data distributions in continuous time. Noise is added in an infinitesimal interval between timesteps. Therefore, a stochastic differential equation (SDE) is adopted in such formulated diffusion models to describe changes in continuous timesteps [5].

Concretely, the forward transition to add noise is formulated as a forward SDE:

$$dx = f(x, t)dt + g(t)dw, \tag{10}$$

where w is the standard Wiener process and accounts for noise in the forward transition, and $f(x, t)$ and $g(t)$ are the drift and diffusion coefficients to account for the mean and variance in the forward transitions, respectively.

At the same time. a reverse SDE for the reverse transition (Eq. 3) is also determined by these coefficients [39]:

$$dx = \left[f(x, t) - g^2(t)s_\theta(x_t, t)\right]dt + g(t)dw, \tag{11}$$

where the output of the denoising network $s_\theta(x_t, t) = \nabla_x \log p(x_t)$ is the score [40]. Likewise, $\left[f(x, t) - g^2(t)s_\theta(x_t, t)\right]$ and $g(t)$ account for the mean and the variance in Eq. 3. The training objective is defined as:

$$L = \mathbb{E}_t\left[\lambda(t)\mathbb{E}_{x_0}\mathbb{E}_{x_t|x_0}\left[\|s_\theta(x_t, t) - \nabla_x \log p(x_t|x_0)\|_2^2\right]\right], \tag{12}$$

where $\lambda(t)$ is the weighting function. Appendix C presents the equivalence of using $\nabla_x \log p(x_t)$ and $\nabla_x \log p(x_t|x_0)$ in the optimization objective.

Finally, the sampling procedure obtains x_T, and applies the trained network θ^* to generate novel data.

3 The Forward Process

The forward process defines the way data to be perturbed by configuring the noise and specifying a transition chain. The way the perturbation happens affects not only the forward process but also the reverse process and the sampling procedure [41]. Noise configuration involves the schedule and the type of noise to be added [42]. The transition chain specifies how the data distribution are transformed [43].

3.1 The Noise Configuration

The schedule [41] and the type of noise [44] are configured for effective and expressive diffusion models. Specifically, a suitable schedule provides such a noise intensity that perturbs the data samples at an appropriate speed [18], [45]. Moreover, different types of noise have an influence on the modeling capabilities of diffusion models [45].

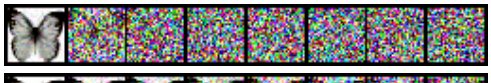
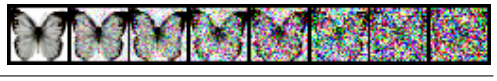
3.1.1 The Noise Schedule

The noise schedule controls the amount of noise ϵ_t to be added at timestep t. It schedules a value β_t as the noise level for each timestep t [1]. The noise ϵ_t is scaled by the noise level and then is used to perturb x_0 [16].

A suitable schedule encourages a balance between exploration and exploitation. Exploration describes the ability of generalization on data not seen during training [46] while exploitation refers to the convergence situation where a model fits the training data well [47]. According to the definition of the forward process in Eq. 1, the larger the amount of noise that is used, the faster the data structures are destroyed, and vice versa. On the one hand, a sufficient amount of noise is necessary to encourage exploration to generalize well on unseen data, while excessive noise may result in sub-optimal convergence or inconvergence of a model, which cannot adequately recover the details of data [48]. On the other hand, too little noise boosts exploitation to fit distributions well but undermines generalization [49], [50]. Empirically, smaller noise levels are scheduled at early timesteps for exploitation, and higher levels are assigned at late timesteps for exploration for a better balance [51].

Data are often considered when a noise schedule is designed. From the viewpoint of data representation, the number of data dimensions and the maximum Euclidean distance of training samples need to be considered to design the noise schedule [52]. Furthermore, the noise schedule should align with the complexity and redundancy of the data [41]. For example, larger images may require more noise than smaller ones [53]. In the context of data reconstruction, adding sufficient noise is beneficial for easy modeling of the distribution while adding little noise helps accurate reconstruction [54].

TABLE 3
Comparison of several streams of noise schedules.

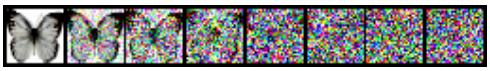
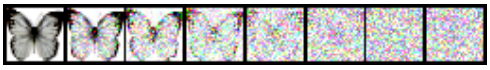
Noise Schedule	Visualization
Linear [17]	
Simple Linear [41]	
Cosine [53]	
Exponential [16]	
Sigmoid [55]	

The noise schedule can be learned by a network or empirically designed using mathematical formulations. To learn a noise schedule, existing methods recognize it as a parameter to be learned jointly with other parameters [54]. These parameters are usually optimized by maximizing a variational lower bound on the log-likelihood [56]. Since the noise schedule is learned by the network, it can be different for training and sampling to achieve optimal results [50]. In contrast, manually-designed noise schedules are formulated with a wide variety of mathematical heuristics. Some design the noise schedule to be affine [17], [41], or to have an exponential relationship [16] with the timestep. Both are thought to perturb data quicker than necessary. Consequently, other functions, such as cosine [53], sigmoid [55], and mathematical integral [18], are used to achieve a more appropriate perturbation speed in the forward process. Table 3 shows several examples of designed noise schedules.

3.1.2 The Noise Type

The selection of the noise type leads to improved distribution approximation and greater degrees of freedom, accentuating its significance on expressiveness of diffusion models. Specifically, selecting an appropriate noise type enhances the model capacity as it fits the perturbed distributions at different timesteps more accurately [33], [45]. Additionally, different types offer varying degrees of freedom [42]. This brings more flexibility in modeling distributions.

TABLE 4
Comparison of several streams of noise types.

Noise Type	Visualization
Gaussian [16], [17]	
Gamma [42], [45]	
Soft [44], [57]	

Different noise types have been developed based on empirical experiments. Inheriting from the denoising score matching [58], isotropic Gaussian noise is commonly used for its simplicity and compatibility [1], [16], [17]. Several variants of isotropic Gaussian noise, such as mixture of Gaussian noise [42] and non-isotropic Gaussian noise [59], have also been applied for increased expressiveness. When correlation exists in a data sample, e.g. frames of a video, the noise are designed to be correlated as well [60]. Additionally, Gamma distribution is another feasible and promising alternative. It has one more degree of freedom and fits distributions better [45]. Soft corruptions can be recognized as a generalized noise for perturbation. This type of noise is more than traditional statistical distribution and supports various operators like masking to perturb data [44]. Such operators also destroy data structures as the aforementioned noise does. This greatly extends the expressive power as a wide variety of operators become available [57]. Overall, these attempts on alternative types pave the way towards more generalized diffusion models. On the other hand, their re-formulations imposed by new noise type may be costly [42]. Table 4 visualizes different types of noise.

3.2 The Transition Chain

The chain of transitions controls the way of perturbing the given data distribution. Varying the terminal distribution in the forward process helps train the denoising network in the reverse process effectively and efficiently [61]. The systematic method is an emerging transition design to increase expressiveness [62]. The chain is also adapted for different data properties [63]. We discuss all those, next.

3.2.1 The Terminal Distribution

A large discrepancy between the original distribution $p(x_0)$ and the terminal distribution $p(x_T)$ may lead to a suboptimal learning outcome for diffusion models. Through the chain of transitions, $p(x_T)$ is determined by adding noise to the original distribution $p(x_0)$ after T timesteps. Their discrepancy is usually significant because x_T is full of noise with almost no original structures remaining [64]. A denoising network is trained to overcome the discrepancy by transforming $x_T \sim p(x_T)$ back to $x_0 \sim p(x_0)$. Consequently, such a large discrepancy may lead to inefficiencies and slow convergence of optimization in the reverse process. In other words, it may require more timesteps to correct this large discrepancy through the denoising network [65].

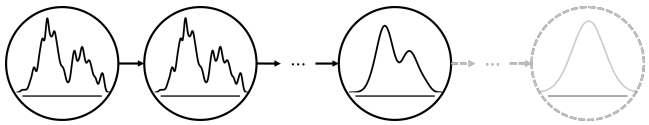


Fig. 5. The transition chain no longer seek an isotropic Gaussian distribution as the terminal distribution. The grey, dashed parts represent that the transition no longer approaches the isotropic Gaussian distribution.

The chain, instead, seeks to maintain as many structures of x_0 as possible in the terminal distribution to reduce the discrepancy, as shown in Figure 5. An isotropic Gaussian distribution has no information about the given data samples despite its simplicity to use [1], [17]. Instead, other approaches typically consider the statistics of the training dataset like the mean and variance to indicate data structures [66]. This mitigates the potential discrepancy and improves the convergence [62]. Such directly involving statistics still requires prior knowledge on the exact formulation of the terminal distribution [64], [67]. An extra network is thereby developed to learn the distribution $p(x_t)$ at an earlier timestep $t < T$ and $p(x_t)$ is then recognized as the new terminal distribution. As $t < T$, more data structures remain in $p(x_t)$ where less noise has been added [68]. The network is often pre-trained to avoid optimizing jointly with the denoising network for an easier optimization [69].

3.2.2 The Systematic Method

The systematic method involves more than one chain of transitions in the generic pipeline. It becomes increasingly popular for its impressive performance and flexibility [70]. Multiple chains are jointly applied on flexible inputs. For example, inputs may be a pair of data and its label [5], or

the decomposed data in subspace [71]. Figure 6 shows an example of using two transition chains on decomposed data.

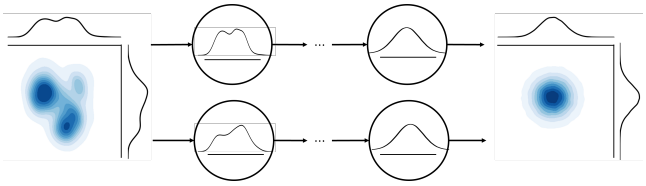


Fig. 6. A systematic method allows the forward process to separately transform the original data in orthogonal subspace.

The systematic method has various benefits. It increases the expressive power by supporting multiple data components and types in the forward process. With more than one chains, the given samples can be decomposed into several components that are compatible with the the generic pipeline [72]. This increases the expressiveness, and avoids inaccurate high-dimensional extrapolation and high computational cost [73]. Moreover, transitions are feasible to apply different speeds of perturbations in the forward process based on their properties [74], [75]. Different data types are also supported by the systematic method. For instance, discrete and continuous data are both transformed using two different transitions simultaneously [13], [76]. Additionally, the systematic method enables incorporating conditions by perturbing the given condition with an extra chain. For example, label vectors are perturbed and learned jointly with data [5], [77]. Sometimes domain knowledge becomes applicable if the systematic method is used. For example, statistical mechanics in physics is suitable for multiple transition chains leading to better performance [43].

Despite those advantages, the systematic method may require to change the equations of the forward process, and thus, the reverse process and the sampling procedure. The generic pipeline may need to be re-derived [13], which is usually more complicated [73] for computation.

3.2.3 Data Properties

Generalizing diffusion models to a wide range of applications requires adaptations according to data properties. Data in tasks such as text generation and speech generation are represented with different types [78]. Furthermore, the data manifold itself is different in some areas like scientific computing [79]. Latent features are also important in large scale tasks for more compacted representations [80].

Different data types require adaptations of the transition chain. Essentially, different data types have varying characteristics. For instance, categorical data are represented as discrete distributions, while continuous data are modeled as arbitrary values [81], [82]. Although both discrete and continuous data are compatible with diffusion models, their generic pipeline has an inductive bias of continuous data [13]. The inherent data difference requires effort to adapt the transition chain to improve the modeling accuracy for different data types [66], [83]. Some efforts focus on converting discrete data to continuous representations before they are calculated [84], [85]. This is straightforward but accuracy may be lost. Others concentrate on re-designing the transitions with discrete distributions [63], [86], [87] such as the Bernoulli distribution. Despite their superior performance, their derivations sometimes are task-specific and expensive to be adapted to other tasks.

Data manifolds also demand modifications of the transition chain in the generic pipeline. Apart from the commonly used Euclidean manifold, the Riemannian manifold is also explored for handling some scientific data [88]. Riemannian manifold-based diffusion models are motivated by their ability to preserve geometric structures of the underlying data, which are important in many scientific fields [89]. Riemannian manifold-based diffusion models have shown promising results in various applications such as medical imaging and neuroscience [90].

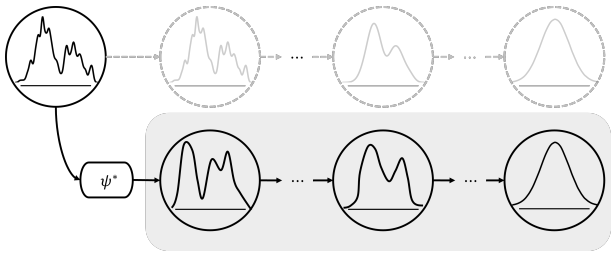


Fig. 7. The transition chain in a latent space. ψ^* is a pre-trained encoder. Data are no longer manipulated in the original space (dashed, grey). They are now transformed within the latent space (rounded rectangle).

Latent representation is another feasible choice for transitions, as illustrated in Figure 7. Despite positive generation results achieved in the original space, the high dimensionality of data often leads to considerable computational cost and redundancy [91]. Empirical evidence has shown that some transitions in diffusion models are responsible for learning latent representations, which are usually in low-dimensional space [92] and semantically meaningful [93], [94]. A pre-trained encoder is developed to map x_0 into latent code z_0 ahead of diffusion models and its corresponding decoder maps the generated latent code z_0 back to x_0 [95]. Usually latent features are more densely represented by filtering less relevant information. Therefore, all the capacity of diffusion models are encouraged to learn more abstract, semantical relationships effectively [80].

4 The Reverse Process

The reverse process focuses on training a denoising network to remove noise [96]. The denoising network is configured by its network architecture [97] and its output parameterizations [98]. To train the configured network, optimization designs are also developed [52].

4.1 Network Architectures

U-Net [99] and Transformer [100] are two commonly adapted architectures for denoising networks [101]. The denoising network is trained to realize the reverse transitions (Eq. 3). It is theoretically flexible to incorporate a wide variety of architectures that keep the dimensionality unchanged during all transitions [102]. Both U-Net and Transformer become increasingly popular for their high capacity on modelling complex relationships in a wide range of applications. While other architectures may also

theoretically be compatible without changing dimensions like GAN [103], they are often adopted for task-specific purposes, e.g., adopting GAN for fast generation [48], and may not be generally applicable to other purposes.

4.1.1 U-Net

From a theoretical standpoint, U-Net is a U-shaped encoder-decoder architecture for general purpose. It was originally proposed for image segmentation [99] and later was adapted to a large variety of tasks. Its encoder extracts high-level features from data and usually contains downsampling layers to compress data. Its decoder leverages such features for different purposes and usually upsamples back to the original dimensionality of the data. This architecture forms an information bottleneck [104] and encourages the network to learn features effectively. Therefore, U-Net has been adopted for a wide variety of tasks.

Various architecture modifications are applied to adapt U-Net as the denoising network. One implementation, known as PixelCNN++ [105] based on a Wide ResNet [106], is adapted as the denoising network by replacing weight normalization [107] with group normalization [108] for learning efficiency [17]. Cross-attention [109] is introduced for higher capacity [110]. Normalization layers are also explored as the conditions in diffusion models [38], [53]. Although various architecture modifications are implemented, the overall architecture of U-Net remains intact [111].

4.1.2 Transformer

Transformer is also an encoder-decoder architecture. Both its encoder and decoder are featured by self-attention functions, which differentially measure the significance between all inputs regardless of their spatial locations [100]. This enables to better capture global dependencies in many tasks.

Transformer is increasingly adapted as an alternative architecture for the denoising network. In principle, a transformer can directly substitute U-Nets because it can also maintain the data dimensions [112]. Nevertheless, direct substitution empirically does not yield better quality because transformers are known to model global relations and may suffer from losing short-range dependency [113]. Therefore, some U-Net structures are usually added to transformers to retain the benefits of U-Net as much as possible [114]. For example, extra long skipping connections in U-Net also play a central role in transformer-based denoising networks [115]. Despite these improvements, not all U-Net structures are essential. For example, the down-sampling and up-sampling operators are not necessary for transformer-based denoising networks [115].

Transformer-based denoising networks also exhibit extra desirable properties when compared with U-Net architectures. For example, the transformer-based denoising network achieves comparable performance in conditional image generation [116], [117], and better quality with less network complexity in unconditional image generation task [118]. Graph transformers are also explored to capture graph relationships [119]. Transformers also encourage scalability [120], [121], and multi-modality [122] in diffusion models. The transformer encoder is also separately adopted with other task-specific decoders in the denoising network. This is enabled by the disentanglement in an encoder-decoder structure [97]. For example, a strong transformer encoder is used to achieve better performance in image generation [97] and motion synthesis [11].

4.2 Parameterizations of the Reverse Mean

The output of the denoising network is applied to parameterize the reverse mean $\mu_\theta(x_t, t)$ in the reverse transition (Eq. 3). Different parameterization ways all center on the estimation of the original data x_0. Specifically, the true value of the reverse mean, denoted as $\mu(x_t, t)$, is formulated as:

$$\mu(x_t, t) := \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t}, \quad (13)$$

where x_0 is the original data but is unavailable during the reverse process. Therefore, x_0 needs to be estimated from the observed purturbed data x_t and timestep t by the denoising network. One parameterization way is to directly output the estimation $\hat{x}_0$ by the denoising network and replace x_0 with $\hat{x}_0$ in Eq. 13. An indirect parameterization way designs the denoising network to predict the noise $\hat{\epsilon}_t$, which is the residual between the unknown x_0 and the observed x_t [17]. Another indirect parameterization way is based on the probabilistic viewpoint and predicts the score $\hat{s}_t$ via the denoising network. $\hat{s}_t$ is the gradient that points towards the unknown x_0 from the current position x_t in data space. Combinations among aforementioned ones are also proposed for special tasks. Figure 5 shows a comparison of different outputs and their corresponding parameterization ways. Different outputs by denoising networks are equivalent to each other [18], [54]. Please refer to Appendix B for a detailed explanation on the equivalence.

While essentially equivalent, different outputs as well as corresponding parameterizations show unique characteristics in particular aspects. Using $\hat{x}_0$ mainly supports better accuracy in the initial stage of the reverse process while $\hat{\epsilon}_t$ is preferable in the late stage. Employing $\hat{s}_t$ avoids computing the normalizing constant, which is a common problem in the context of distribution modelling. Combining the aforementioned ones provides the flexibility to retain their benefits.

TABLE 5
Visualization of parameterization ways. Outputs are the values predicted by a denoising network. The parameterization is the formulation to use the corresponding output in the reverse process. Each visualization is about the corresponding output.

Output	Parameterization	Visualization
Data $\hat{x}_0$	$\mu_\theta(x_t,t) := \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{x}_0}{1-\bar{\alpha}_t}$	
Score $\hat{s}_t$	$dx := \left[f(x,t) - g^2(t)\hat{s}_t\right]dt + g(t)dw$	
Noise $\hat{\epsilon}_t$	$\mu_\theta(x_t,t) := \frac{1}{\sqrt{\alpha_t}}x_t - \frac{1-\alpha_t}{\sqrt{\alpha_t(1-\bar{\alpha}_t)}}\hat{\epsilon}_t$	
Combination $\hat{c}_t$	$\mu_\theta(x_t,t) := \mathcal{C}(\hat{x}_0, \hat{s}_t, \hat{\epsilon}_t)$	N/A

4.2.1 Data

Predicting the original data x_0 provides a straightforward denoising direction. $\hat{x}_0$ indicates a denoising goal towards which x_t should be changed. In particular, given the observation x_t at timestep t, the parameterization is defined as:

$$\mu_\theta(x_t, t) := \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{x}_0}{1-\bar{\alpha}_t}, \quad (14)$$

where α_t indicates the noise level at timestep t as previously defined in Section 2.2.1.

Parameterizing with $\hat{x}_0$ is advantageous at the beginning of the sampling procedure, while it leads to inaccuracy when approaching the end of the sampling procedure. Empirical results show that the estimated mean $\mu_\theta(x_t, t)$, which is parameterized by $\hat{x}_0$, is closer to the ground truth $\mu(x_t, t)$ at the beginning of the sampling procedure [3], [123]. This is because $\hat{x}_0$ helps the denoising network with an overall understanding of the global structure [22]. On the contrary, when approaching the end of the sampling procedure where substantial structures have already been formed and only small noise artifacts need to be removed, finer details are difficult to be recovered [124]. In other words, the information brought by $\hat{x}_0$ becomes less effectiveness in this case.

4.2.2 Score

Score is the gradient of the logarithm of a distribution [40]. The gradient indicates the most possible changes between two timesteps [125]. Therefore, as shown in Figure 8, denoising samples by the score forms a trajectory in data space [126]. In particular, given the observed x_t and timestep t, the predicted score is defined as $\hat{s}_t := \nabla_x \log p(x_t)$ and the corresponding parameterization is the reverse SDE:

$$dx := \left[f(x,t) - g^2(t)\hat{s}_t\right] dt + g(t)dw, \quad (15)$$

where $f(x,t)$ and $g(t)$ are the coefficients as previously introduced in Section 2.2.2.

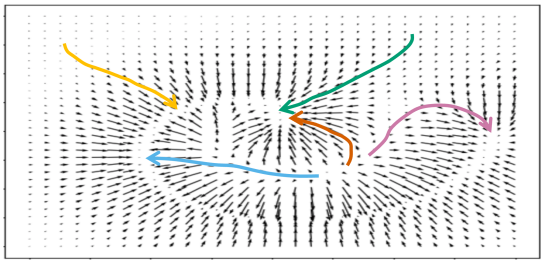


Fig. 8. Visualization of the trajectory by predicting score. A score is a direction for next timesteps. Samples are denoised in the direction at each position. Colors represent trajectories of different samples.

Predicting score in the reverse process avoids the estimation of the constant for normalizing the probabilistic distributions. Instead of representing a probability distribution by probability [127], score computes the gradient of the logarithm of a distribution. This avoids estimating the normalizing constant, which is computationally expensive or sometimes infeasible [128]. In particular, the predicted distribution is usually defined as:

$$p_\theta(x) = \frac{\exp^{-f_\theta(x)}}{Z_\theta}, \quad (16)$$

where $Z_\theta > 0$ is a normalizing constant to be estimated so that $\int p_\theta(x)dx = 1$. Predicting score avoids this problem:

$$\nabla_x \log p_t(x), \quad (17)$$
$$= -\nabla_x f_\theta(x) - \nabla_x \log Z_\theta, \quad (18)$$
$$= -\nabla_x f_\theta(x), \quad (19)$$

where $\nabla_x \log Z_\theta = 0$ as Z_θ is a constant with respect to x.

4.2.3 Noise

Noise estimation predicts the noise added in the forward process. Generally, the predicted noise is scaled according to the noise schedule and then subtracted from the observation [17], [129], as shown in Figure 9. In particular, given the observation at a current timestep, the prediction of noise is denoted as $\hat{\epsilon}_t$ and the parameterization is defined as:

$$\mu_\theta(x_t, t) := \frac{1}{\sqrt{\alpha_t}}x_t - \frac{1-\alpha_t}{\sqrt{\alpha_t(1-\bar{\alpha}_t)}}\hat{\epsilon}_t, \quad (20)$$

where α_t indicates the noise level at timestep t as previously defined in Section 2.2.1.

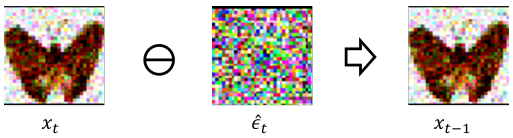


Fig. 9. Visualization of the noise-based parameterization. ⊖ means $\hat{\epsilon}_t$ has a subtractive relationship with x_t, and ⇨ means this results in x_{t-1}.

The consistent magnitude and residual effect of $\hat{\epsilon}_t$ are advantageous. The fixed statistics of noise, e.g. $\hat{\epsilon}_t \sim \mathcal{N}(0, I)$, lead to a consistent magnitude to be predicted. This encourages the learning of the denoising network [54]. Besides, the residual effect to preserve the input x_t in x_{t-1} is available by predicting zero noise [130]. This becomes increasingly beneficial towards the end of the reverse process where only minor modifications are needed [124].

A large deviation between the ground truth noise ϵ_t and the predicted noise $\hat{\epsilon}_t$ may occur at the beginning of the sampling procedure, and is hard to be corrected in the following timesteps. The sampling procedure starts from samples with large noise, with almost no clue for the denoising network to predict noise accurately [17]. This potentially leads to a deviation [124]. The deviation is scaled up by the noise schedule in Eq. 20. The scheduled level of noise is usually large at the beginning of the sampling procedure. Even for a small noise estimation error, the deviation will be sharply enlarged. Moreover, the denoising network is limited to predicting noise, which has a residual effect in the noise-based parameterization. The magnitude of potential correction at each timestep is relatively small, and thereby more timesteps are required to correct such a deviation [22].

4.2.4 Combinations

Combining two or more predictions is also possible for task-specific benefits. Abstractly, the combination is denoted as $\hat{c}_t := \mathcal{C}(\hat{x}_0, \hat{s}_t, \hat{\epsilon}_t)$ where $\mathcal{C}$ stands for a combination operator. Therefore, the parameterization is:

$$\mu_\theta(x_t, t) = \hat{c}_t. \quad (21)$$

This has a wide variety of feasible implementations. The output to be combined and the combination operators can be very diverse [20]. Velocity prediction is one example that linearly combines $\hat{x}_0$ and $\hat{\epsilon}_t$ [131], which is designed as:

$$\mu_\theta(x_t, t) := \alpha_t \hat{\epsilon}_t - \sigma_t \hat{x}_0, \tag{22}$$

where α_t and σ_t are the scaling factor and noise schedule respectively. Such a combination has better stability in the task of distilling diffusion models [98]. Combination is also used to avoid noise existing in $\hat{x}_0$ [132] or to achieve higher likelihood [133]. Dynamically alternating between $\hat{x}_0$ and $\hat{\epsilon}_t$ is found to accelerate the generation [124].

4.3 Optimization Designs

Many optimization designs on the learning objectives are developed to train the denoising network [134], with the reverse variance and the learning weight being the two main factors. Optimizing over the reverse variance assists the fitting of the denoising network. The learning weight controls the attention of learning priorities.

4.3.1 The Reverse Variance

Modelling the reverse variance improves the training efficiency of diffusion models. An appropriate variance minimizes the discrepancy between the predicted reverse transition $p_\theta(x_{t-1}|x_t)$ and the forward transition $p(x_t|x_{t-1})$, fitting the forward process better [135]. This facilitates less timesteps to be used, and improves overall efficiency.

Many efforts to model the reverse variance are attempted. Some empirically adopt a handcrafted value for each timestep. The noise schedule is a popular option for its simplicity and empirical performance [136]. Scaling the schedule by a factor is also researched but does not lead to a large difference [17]. Both choices are considered as upper and lower bounds on reverse process entropy [1], and the interpolation between them is learned for flexibility [53]. Others find the optimal variance can be solved analytically. Its formulation is explicitly derived from the predicted score [135], and improves the efficiency of generation [137].

4.3.2 The Learning Weight

Learning priorities are balanced by weights in the learning objective to enhance the learning quality. The change of learning priorities is common in deep learning [138] and has also been observed in the reverse process [139], [140]. Generally, a diffusion model learns features with semantic correspondence [141]. In other words, it pays more attention to global structures at the beginning of the reverse process [142] and then changes to local details when approaching its end [143]. Beside, it is shown to understand high-level phenomena like compositionality [144]. A balance is achievable through adjusting weights and beneficial for training [145].

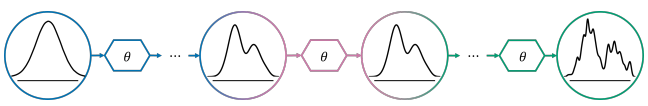


Fig. 10. Learning priority changes in the reverse process, which is denoted by different colours.

Weights are usually related with the noise schedule. Directly using the schedule as the weight emphasizes a better learning of global structure by a larger learning weight at the beginning of the reverse process [16], [146]. While it is simple to use, the pre-defined schedule is not flexible and may deviate away from the actual demands. A function of the noise schedule such as the signal-to-noise (SNR) ratio is designed to compute the weights. The actual remaining noise is measured rather than the scheduled one [54]. It takes the data into account and better balances the learning of local details and global structures [143].

5 Sampling Procedure

The sampling procedure in a diffusion model usually follows the transitions in the reverse process but uses a trained denoising network. It moves backwards on the chain and is responsible to transform a sample x_T from a terminal distribution $p(x_T)$ to generate new data $\hat{x}_0 \sim p_{\theta^*}(x_0) \approx p(x_0)$.

Conditional and fast generation are two focused areas of the sampling procedure in diffusion models. Without modelling conditions, diffusion models usually do not generate data of high quality when data are considered to follow a conditional distribution [38], [147], e.g., images from LSUN [148] with ten scene categories are considered to follow a conditional distribution. Effective mechanisms of guidance are designed to modify transitions in the sampling procedure to be compatible with conditions. Furthermore, the sampling procedure is several times slower when compared with other generative models [149]. The long generation time is mainly attributed to the large number of timesteps [61]. Thus, designs for acceleration are explored to reduce timesteps without heavily impairing quality.

5.1 Guidance Mechanisms

A guidance mechanism modifies the denoising direction. It corrects the unconditional direction based on the given conditions [150], and thereby reduces the discrepancy between the modified and true conditional distributions [151], [152]. The condition $c \in \mathcal{C}$ can be diverse, e.g., images [153], texts [154], or 2D poses [155]. Without loss of generality, guidance is discussed using score as the output.

A wide variety of mechanisms are proposed to incorporate condition c with diffusion models. As temporal conditions, i.e. timesteps t, are inherently compatible with diffusion models [17], vanilla guidance extends them to incorporate c through operations such as addition [38]. Despite its convenience, such a guidance mechanism is weak and sometimes is not working well to modify the denoising direction [22]. To achieve effective and adjustable strength of conditions, classifier guidance leverages an extra pre-trained classifier to change the denoising direction [38]. The strength is adjusted by scaling the change with a weight. Nonetheless, training a classifier on data with noise leads to extra cost and training instability [156]. To avoid such problems, classifier-free guidance combines the vanilla guidance and unconditional model for guidance. Moreover, learning the modification as guidance instead of manually deriving the guidance is emerging for its flexibility [155].

5.1.1 Vanilla Guidance

Vanilla guidance incorporates the given conditions c jointly with timesteps t as the guidance. A timestep t itself is inherently taken as a condition by a denoising network [157], [158]. A variety of operations such as addition [159], [160], [161], [162], and attention layer [163], [164], [165], [166] are available for this guidance mechanism. Figure 11 shows the condition is added to a timestep in this mechanism.

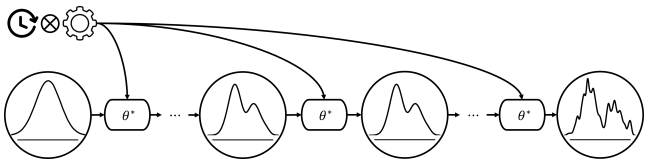


Fig. 11. Vanilla guidance merges the condition c with the denoising network θ^* by adding to each timestep t.

While vanilla guidance is simple, its effectiveness is undermined by the lack of adjustable conditional strength [156]. Empirical evidence shows that a conditional diffusion model trained with vanilla guidance may not conform to the conditions or under-perform in conditional generation [22].

5.1.2 Classifier Guidance

For effective and adjustable strength of conditions, an extra classifier with a weight is trained for classifier guidance. The gradient of the classifier is scaled by the weight and then is used to modify the unconditional denoising direction, as shown in Figure 12. In other words, the weight controls how much to encourage the gradient-based modification [167]. To obtain the gradient as accurate as possible, the classifier is trained on data with noise at each timestep. In particular, classifier guidance is formulated as:

$$\nabla_x \log p(x|c) = \nabla_x \log p(x) + w\nabla_x \log p(c|x), \tag{23}$$

where $\nabla_x \log p(x|c)$ and $\nabla_x \log p(x)$ are conditional and unconditional scores, respectively, $\nabla_x \log p(c|x)$ is the gradient of a classifier, and w is the weight. When $w = 0$, this mechanism becomes unconditional. As the weight increases, the denoising network is more and more constrained to produce samples that satisfy conditions.

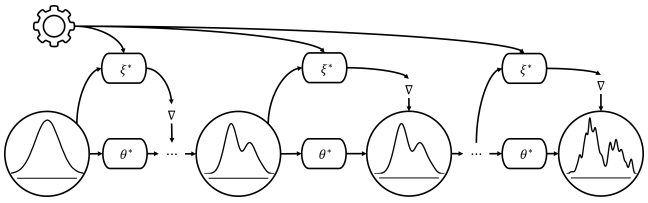


Fig. 12. Classifier guidance leverages an extra classifier network ξ^* to compute a gradient ∇ as the modification on the denoising network θ^*. The timestep condition t is omitted here for visualization.

Classifier guidance provides control in a similar way of gradient-based adversarial attack. A sample is updated to satisfy the classifier by using the classifier gradient [156], which is a similar way in gradient-based adversarial attack [168], [169]. Such updates increases the probability of the sample for which the classifier assigns high likelihood to the correct label [170]. Furthermore, different pre-trained classifiers can be applied in a plug-and-play manner [171].

Additionally learning a classifier may lead to extra cost and training instability. The extra expense is further scaled up because the classifier is trained on data with every scheduled noise level [156]. Moreover, training the classifier on data with noise tends to be unstable [172], [173]. The data structure is almost destroyed because more and larger noise is added according to the noise schedule. Therefore, the quality of classifier gradient may not be consistent [174]. Sometimes its direction is arbitrary or even opposite [175], [176] and leads to less effective or wrong guidance [177].

5.1.3 Classifier-Free Guidance

To avoid the extra classifier, classifier-free guidance replaces the classifier by a mixture of unconditional model and vanilla guidance. It encourages the model in the direction of guidance and simultaneously discourages away from unconditional direction [178]. As shown in Figure 13, instead of training two models, a conditional model and an unconditional one are formulated uniformly by dropping out conditions c with a probability p [156]. The two models are learned jointly as if they were a singular conditional model [22]. In particular, classifier-free guidance is formulated as:

$$\nabla_x \log p(x|c) = w\nabla_x \log p(x|c) + (1-w)\nabla_x \log p(x), \tag{24}$$

where w is the weight of conditions.

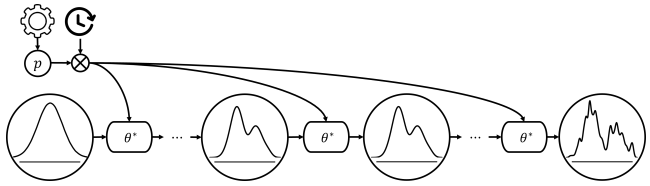


Fig. 13. Classifier-free guidance is based on a mixture of vanilla guidance and unconditional model θ^*. A probability p controls whether to drop out the conditions during training.

The weight is slightly different from its counterpart in classifier guidance. When $w = 0$, the classifier-free guidance becomes unconditional models without vanilla guidance. The vanilla guidance is a special case when $w = 1$. In this case, the unconditional model is suppressed and conditions are incorporated through vanilla guidance [35]. If $w > 1$, the classifier-free guidance restrains the unconditional model and prioritizes conditions further by larger weights. The score from classifier-free guidance deviates quickly away from the unconditional score, and thus, samples that better satisfy the conditions will be generated [179].

Instead of removing classifiers, self-guidance [150] reduces or removes the requirement of annotation by using internal values like activations and attention maps [180] to compute guidance. Such design helps finer-grained control and is compatible with aforementioned guidance.

5.1.4 Learned Modifications

Modifications for guidance can be learned for more flexibility. A network is deployed to directly learning the modification on the output of unconditional networks [181]. Thus, it is more flexible than the aforementioned ones because of fewer manual designs. For example, a pre-trained unconditional denoising network is commonly augmented

by its identically copied network including the parameters. The outputs from the two networks are usually added and other complex operations are also possible. During training, only the copied network is updated to automatically learn suitable modifications to correct the output from the pre-trained network. Although fixing the original network and copying its parameters to the identical one avoid completely retraining, learning the copied network may still be difficult due to the larger number of parameters [182]. Nevertheless, this flexibility greatly encourages the pursuit of unified and multi-modality guidance [183].

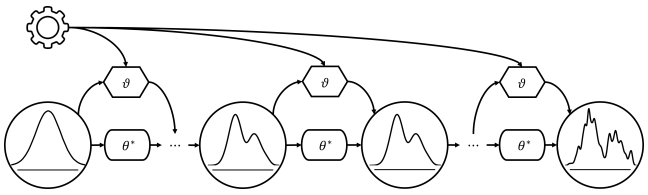


Fig. 14. Applying an extra network ϑ to directly learn the required modification for guidance. Timestep condition is omitted here.

5.2 Acceleration Designs

Reducing the number of timesteps for generation is the main goal of acceleration. Generally, the denoising network needs to wait the results from the timestep $t+1$ to accomplish the transition at the current timestep t [61]. The inference speed is significantly slowed down especially when a large number of timesteps are required in the sampling procedure [48]. Efforts have been contributed to reduce the timesteps. Truncation directly cuts the sampling procedure at a certain timestep [69]. Knowledge distillation is adopted to learn a student model that has fewer timesteps in its sampling procedure [184]. Selection strategies are also developed to select a subset of timesteps for fast generation [61].

5.2.1 Truncation

Truncation involves a partial sampling procedure with an extra network. It usually selects an intermediate timestep t', and obtains a sample from the corresponding distribution $p(x_{t'})$ for the generation [185], as shown in Figure 15. In other words, the process truncates the whole chain at t', and thereby fewer timesteps remain in the partial chain [186]. An extra network needs to be additionally trained to model $p(t')$ that may not be tractable [17]. Overall, truncation is theoretically effective in acceleration [65], which is proved by the stochastic contraction theory [187].

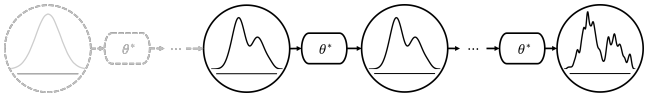


Fig. 15. The sampling procedure is truncated and starts from a selected timestep. The grey, dashed parts represent discarding for generation.

Truncation effects can be two-sided. One the one hand, truncation comes with several benefits. Not only inference, but also training can be accelerated [68] as both the forward and reverse processes do not require computation at these truncated timesteps. Truncation also strikes a balance between acceleration and quality. It has an adaptable intercepting point to balance the generation quality and efficiency. The selection of such a point depends on the data complexity [68] and the degree of corruptions [65]. Besides, truncation takes advantage of the properties of the involved extra network, which is often another generative model [69]. On the other hand, truncation may lead to an increased training expense because the extra network needs to learn $p(t')$ as accurately as possible [69].

5.2.2 Knowledge Distillation

Knowledge distillation is a network compression technique. It involves compressing an expensive but high-performing teacher model into a smaller student model [188]. After the training, the student network can perform as well as the teacher with a smaller model size.

The same idea is applied to learn a new sampling procedure with fewer timesteps. In terms of diffusion models, it involves the original sampling procedure as the teacher model and a new one with fewer timesteps as the student model [189], [190]. Figure 16 shows an example method of progressive distillation on the sampling procedure.

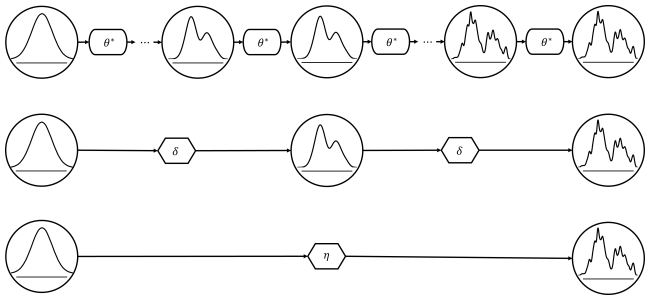


Fig. 16. Knowledge distillation learns student denoising networks δ and η with fewer timesteps, based on the teacher denoising network θ^*.

Knowledge distillation is applied to merge several time steps into one single time step for the new sampling procedure. Directly distilling all timesteps in the teacher sampling procedure into a single timestep in the student one, theoretically reduces significantly generation time [191], [192]. However, this relies on collecting a large dataset of samples from the teacher model for knowledge distillation, which itself is computationally expensive [193], [194]. Instead, the number of timesteps can be progressively reduced [98]. For example, after the student sampling procedure is trained to merge two timesteps in the teacher sampling procedure, it becomes the teacher, and a new student procedure is trained to further reduce the number of sampling steps [195].

5.2.3 Selection Strategies

Many strategies are developed to select a subset of timesteps without undermining quality, and thereby form a shorter sampling procedure [196], [197]. Timesteps around the end of the sampling procedure influence quality less, and they are often dropped out [198]. Figure 17 shows a shorter sampling procedure with selected timesteps. Some may not directly reduce the number of timesteps but select a subset of timesteps for parallel computation [199]. In this way, each single processor deals with fewer timesteps.

Applying differential equation solvers is a popular strategy for those with continuous formulations. They are usually based on well-established mathematical solvers to handle adaptive step sizes and noise schedules. As a continuous formulation itself is based on SDEs, such solvers are straightforward to be adopted. Many existing SDE solvers are available for the sampling procedure like stochastic Runge-Kutta solver [5], Diffusion Exponential Integrator Sampler (DEIS) [200], and Itô-Taylor Sampling by higher-order numerical schemes [201]. Solvers do not necessarily select timesteps uniformly [149]. Dynamic step size is used in the Euler-Maruyama (EM) solver [5]. The selection sometimes also brings improvement on quality [202].

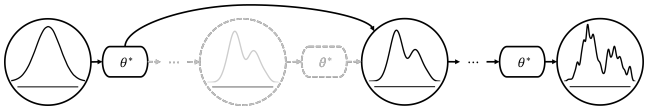


Fig. 17. Selection strategies for the sampling procedure skip the selected time steps for generation.

More general strategies are also developed for both formulations. They sometimes also modify the forward and reverse processes for fast inference, leading to retraining diffusion models. Some strategies leverage particular mathematical tools for acceleration. A non-Markovian reverse process is proposed for few-step sampling [136], which theoretically is equivalent with the probability flow sampler [5]. An exact analytical solution of diffusion models are derived but stochasticity is sacrificed [203]. A Taylor expansion process is applied to disregard non-contributory diffusion steps [204]. Others rely on extra networks or algorithms for fast sampling. Some strategies conduct modifications based on noise schedules. If continuous noise levels are used, an extra network is used to adjust the noise levels of a few-step discrete time reverse diffusion process [50], [205]. The selection of timesteps is directly learned by extra networks [206], [207]. A Dynamic Programming algorithm [208] was also designed to reduce the number of timesteps.

6 Future Trends

We provide our insights on emerging or important trends in this field to encourage more studies in the future.

6.1 Theory

The theory of diffusion models can inspire significant future research. Diffusion models are heavily based on mathematical formulations [19], which facilitate future improvements from mathematical perspectives. For example, deriving new formulations for destroying data dimensions [209] leads to a generalized diffusion model. Building connections with well-established fields is also beneficial to achieve better understanding on the theory of diffusion models [210], [211]. Explainable techniques can also assist with understanding on theories of diffusion models [212]. Besides, the success of diffusion models highlights the virtue of auto-regressive generation where a self-correction mechanism is enabled to achieve better quality [213], [214]. This will contributes to the theoretical designs of generative models in the future.

6.2 Architecture

The improvement of the architecture in diffusion models has a lot of future potential. Current backbones of denoising networks are mainly U-Net [17] and Transformer [116]. They have achieved impressive results but still have inherent drawbacks in some applications. At the same time, a wide variety of well-established network architectures are available in machine learning with appealing advantages. Applying and adapting these network architectures as the denoising network will introduce additional benefits and unleash the potential for diffusion models [215]. Compression of the architecture with fewer parameters is actively being researched [216]. The conditional mechanism is also developing fast for multi-grained guidance. Besides the backbone choice, the optimization is also a promising avenue. Deeper understanding on the model behaviours has been proposed as more experiments are conducted and more theories are developed [22]. For example, the strategy of reinforcement learning has been explored to train diffusion models [217] These understandings facilitate improving the training speed and efficiency in the future.

6.3 Data

Developing data-efficient diffusion models is an important future trend. Conventionally, diffusion models implicitly assume abundant data to train on, especially labelled data for conditional models [218]. This assumption may be violated when acquiring data is expensive or sometimes impossible. Some research explores to improve the training on low quality data [219], [220], [221]. Other research is seeking to combine few-shot learning to efficiently leverage the limited data for training [222], [223]. Several pioneering explorations also consider one-shot [224], [225] and zero-shot [226], [227] learning with diffusion models, further reducing the dependency on abundant training data [228]. These attempts highlight the possibility of applying diffusion models in the absence of data. Besides, changing the way of supervision will be helpful to train conditional models with limited labelled data. Currently, the unsupervised [229], self-supervised [230] or semi-supervised [231] manners have been explored to train conditional diffusion models. They have shown encouraging results in different areas and will facilitate future exploration in the abscence of labelled data. Their success also indicates future opportunities of applying other available supervision schemes like weakly supervised [232] learning to conditional diffusion models.

Supporting multiple data modalities in a diffusion model also has broad prospects. Incorporating information from several modalities will greatly promote the generative power and applicable flexibility of diffusion models [122]. Currently, many efforts focus on two modalities, usually leveraging text as an extra modality in addition to the data of interest [233]. Initial attempts have also supported other modalities from a wide variety like human pose for the guidance in conditional diffusion models [155]. In the future, exploring other modalities is still worth more efforts for better flexibility. More importantly, finding an incorporation for three or more data modalities to be mixed in a single diffusion model is another potential boost for their generative power [234], [235].

6.4 Applications

A wider range of applications beyond generation are promising. Since diffusion models were proposed, they have been applied in a large variety of tasks [236] because of their powerful generative ability. For example, diffusion models are recently applied in policy representation in reinforcement learning [237], neural architecture searching [238], and named entity recognition [239]. Data generated by diffusion models are also employed as the proxy for training new models when the original data are limited [240], [241]. Adapting pre-trained diffusion models to another domain is also popular, e.g., applying an image model for video generation [242]. Recent advances on diffusion models have also witnessed significant increase of successful interdisciplinary applications, especially AI for Science (AI4Science) [243]. Diffusion models have increasingly been combined with Physics [244], [245], Chemistry [246], [247], etc. They are not only applying and adapting diffusion models to solve problems in these domains, but also leveraging knowledge in a disciplinary to theoretically improve diffusion models [248], [249], [250]. Expanding the application scenarios of diffusion models will bring more opportunities to both academic and industrial fields.

Diffusion models are also actively investigated for or against adversarial attack. Diffusion models themselves are analysed against adversarial attack, which is usually measured by robustness. Diffusion models with higher robustness can work vigorously and consistently [251]. Developing diffusion models with higher robustness have attracted more and more public attention to defend adversarial attacks [252], [253]. Diffusion models are also deployed to help other systems with defending adversarial attacks [254], [255], [256], [257]. They are also employed to attack other systems by generating adversarial samples [258], [259].

6.5 Societal Impacts

Despite many successful applications, the potential misuse of diffusion models needs to be regulated. Benefiting from academic and industrial efforts, diffusion models become more and more powerful to produce synthetic data, which may be increasingly difficult to be distinguished by real-world data. While synthetic data reduces the cost of obtaining real data in some fields, such an ability may be abused with harmful societal impacts [260]. For example, misinformation and spam can be spread via manipulated data [261], [262]. Distinguishing these manipulated data becomes more expensive and even more difficult [263]. This requires continuous investment of efforts on techniques and policies as diffusion models are also swiftly developing.

Pre-trained diffusion models may have a risk of data leakage. A concern of data privacy is quickly getting public attention by the possibility of recovering the training data from pre-trained diffusion models [264], [265]. Such data may contain confidential/personal information [266]. Training on privacy-sensitive data requires additional safety considerations and efforts in the future [267].

Reproducing or exacerbating biases by diffusion models is another concern. Diffusion models can be biased in data generation for various reasons such as biased datasets [268], [269]. They are found prone to replicating their training data because such data usually have higher likelihoods [270], [271]. This preference may lead to a systemically biased model when the training data themselves have bias [272]. Biased models will bring users with unfair stereotypes on some concepts and lead to harmful societal impacts [273]. For example, a diffusion model trained for face generation with FFHQ dataset [274] (whose data are collected mainly from people between the ages of 21 and 40) will also be more inclined to generate results from that age group and less supportive of other age groups. More efforts are required in the future to encourage fair and unbiased diffusion models.

7 Conclusion

Diffusion models are an emerging type of deep generative models involving three main components: a forward process and a reverse process for optimization, and a sampling procedure for generation. Diffusion models become popular for their high-quality results in various tasks. The success of diffusion models is closely related with various design fundamentals of the three components. The forward process focuses on perturbing $p(x_0)$ to $p(x_T)$ by gradually adding noise to x_0 by $p(x_t|x_{t-1})$. It is designed based on noise injection and the multi-step chain of transitions. The reverse process focuses on training a denoising network to remove noise. The reverse process includes network-related and optimization-related choices. The sampling procedure uses the trained denoising network for generation and mainly focuses on guidance and acceleration. To achieve conditional generation and fast sampling, guidance and acceleration techniques are designed for the sampling procedure. These designs have all contributed to the current powerful diffusion models. Several future trends that are emerging and important have been introduced to boost this field.

Appendix A
Derivation of the Forward Process

According to the definition of diffusion process, we have:

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{\beta_t}\epsilon_t \tag{25}$$

where each ϵ_t is an i.i.d. standard Gaussian. Then, by recursion, we have:

$$\begin{aligned} x_t =& \sqrt{\alpha_t\alpha_{t-1}} x_{t-2} + \sqrt{\alpha_t\beta_{t-1}}\epsilon_{t-1} + \sqrt{\beta_t}\epsilon_t \\ =& \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}} x_{t-3} + \sqrt{\alpha_t\alpha_{t-1}\beta_{t-2}}\epsilon_{t-2} \\ &+ \sqrt{\alpha_t\beta_{t-1}}\epsilon_{t-1} + \sqrt{\beta_t}\epsilon_t \\ &\cdots \\ =& \sqrt{\bar{\alpha}_t} x_0 + \sqrt{\alpha_t\alpha_{t-1}\cdots\alpha_2\beta_1}\epsilon_1 \\ &+ \cdots + \sqrt{\alpha_t\beta_{t-1}}\epsilon_{t-1} + \sqrt{\beta_t}\epsilon_t. \end{aligned} \tag{26}$$

As a result, $q(x_t|x_0)$ is still Gaussian. Its mean vector is $\sqrt{\bar{\alpha}_t}x_0$, and its covariance matrix is $(\alpha_t\alpha_{t-1}\cdots\alpha_2\beta_1 + \cdots + \alpha_t\beta_{t-1} + \beta_t)I = (1-\bar{\alpha}_t)I$. Formally, we have:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, \tag{27}$$

and in a probabilistic perspective:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I), \tag{28}$$

which is the same as Eq. 7.

APPENDIX B
PROOF OF EQUIVALENT OUTPUTS

For completeness, we provide the proof of equivalent output representations. The original one can be found in [22].

The distribution formulation of $q(x_t|x_0)$ (Eq. 7) can be explicitly written as:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, \tag{29}$$

and can be reorganized as:

$$x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_0}{\sqrt{\bar{\alpha}_t}}, \tag{30}$$

where x_0 is dependent on ϵ_t when x_t is given. This allows to predict ϵ_t for an indirect estimation of x_0. Therefore, we have:

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}_0}{\sqrt{\bar{\alpha}_t}}, \tag{31}$$

which shows the equivalence of predicting x_0 and ϵ_t.

Mathematically, the Tweedie's Formula for a Gaussian variable states that:

$$\mathbb{E}[\mu_z|z] = z + \Sigma_z \nabla_z \log p(z), \tag{32}$$

where μ_z and Σ_z is the mean and variance of z, respectively. From Eq. 7, we know its mean is:

$$\mu_{x_t} = \sqrt{\bar{\alpha}_t}x_0, \tag{33}$$

and its variance is:

$$\Sigma_{x_t} = 1 - \bar{\alpha}_t. \tag{34}$$

Then, by Tweedie's Formula, we have

$$\sqrt{\bar{\alpha}_t}x_0 = x_t + (1-\bar{\alpha}_t)\nabla_{x_t}\log p(x_t), \tag{35}$$

where x_0 is dependent on the score $\nabla_{x_t}\log p(x_t)$. Therefore, if the estimation of the score derives an estimation of x_0:

$$\hat{x}_0 = \frac{x_t + (1-\bar{\alpha}_t)\nabla_x \log p(x_t)}{\sqrt{\bar{\alpha}_t}}. \tag{36}$$

APPENDIX C
EQUIVALENT OPTIMIZATION OBJECTIVES

For completeness, we provide the proof of equivalent optimization of two objectives. The original proof can be found in the Appendix of [58].

Diffusion models should have been optimised by the unconditional form:

$$\mathcal{J}_{uncon}(\theta) = \mathbb{E}_{q_\sigma(\tilde{x})}[||s_\theta(\tilde{x}) - \frac{\partial \log q_\sigma(\tilde{x})}{\partial \tilde{x}}||^2], \tag{37}$$

where s_θ is the prediction of our network with learnable parameters θ. However, they are usually optimized by the conditional form:

$$\mathcal{J}_{con}(\theta) = \mathbb{E}_{q_\sigma(\tilde{x},x)}[||s_\theta(\tilde{x}) - \frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}}||^2]. \tag{38}$$

We can prove that $\mathcal{J}_{uncon}(\theta) = \mathcal{J}_{con}(\theta) - C_2 + C_1$ where C_1 and C_2 are both constants that do not depend on θ.

Given the unconditional form $\mathcal{J}_{uncon}(\theta)$:

$$\begin{aligned}\mathcal{J}_{uncon}(\theta) &= \mathbb{E}_{q_\sigma(\tilde{x})}[||s_\theta(\tilde{x}) - \frac{\partial \log q_\sigma(\tilde{x})}{\partial \tilde{x}}||^2], \\ &= \mathbb{E}_{q_\sigma(\tilde{x})}[||s_\theta(\tilde{x})||^2] - 2\varphi(\theta) + C_1,\end{aligned} \tag{39}$$

where $C_1 = \mathbb{E}_{q_\sigma(\tilde{x})}[||\frac{\partial \log q_\sigma(\tilde{x})}{\partial \tilde{x}}||^2]$ is a constant that does not depend on θ and

$$\begin{aligned}\varphi(\theta) &= \mathbb{E}_{q_\sigma(\tilde{x})}[< s_\theta(\tilde{x}), \frac{\partial \log q_\sigma(\tilde{x})}{\partial \tilde{x}} >], \\ &= \int_{\tilde{x}} q_\sigma(\tilde{x}) < s_\theta(\tilde{x}), \frac{\partial \log q_\sigma(\tilde{x})}{\partial \tilde{x}} > d\tilde{x}, \\ &= \int_{\tilde{x}} q_\sigma(\tilde{x}) < s_\theta(\tilde{x}), \frac{\frac{\partial q_\sigma(\tilde{x})}{\partial \tilde{x}}}{q_\sigma(\tilde{x})} > d\tilde{x}, \\ &= \int_{\tilde{x}} < s_\theta(\tilde{x}), \frac{\partial q_\sigma(\tilde{x})}{\partial \tilde{x}} > d\tilde{x}, \\ &= \int_{\tilde{x}} < s_\theta(\tilde{x}), \frac{\partial \int_x q_0(x) q_\sigma(\tilde{x}|x) dx}{\partial \tilde{x}} > d\tilde{x}, \\ &= \int_{\tilde{x}} < s_\theta(\tilde{x}), \int_x q_0(x) \frac{\partial q_\sigma(\tilde{x}|x) dx}{\partial \tilde{x}} > d\tilde{x}, \\ &= \int_{\tilde{x}} < s_\theta(\tilde{x}), \int_x q_0(x) q_\sigma(\tilde{x}|x) \frac{\partial \log q_\sigma(\tilde{x}|x) dx}{\partial \tilde{x}} > d\tilde{x}, \\ &= \int_{\tilde{x}} \int_x q_0(x) q_\sigma(\tilde{x}|x) < s_\theta(\tilde{x}), \frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} > dx d\tilde{x}, \\ &= \int_{\tilde{x}} \int_x q_\sigma(\tilde{x}, x) < s_\theta(\tilde{x}), \frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} > dx d\tilde{x}, \\ &= \mathbb{E}_{q_\sigma(\tilde{x},x)}[< s_\theta(\tilde{x}), \frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} >].\end{aligned} \tag{40}$$

Therefore, the unconditional form can be written as follows:

$$\begin{aligned}\mathcal{J}_{uncon}(\theta) =& \mathbb{E}_{q_\sigma(\tilde{x})}[||s_\theta(\tilde{x})||^2] \\ &- 2\mathbb{E}_{q_\sigma(\tilde{x},x)}[< s_\theta(\tilde{x}), \frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} >] \\ &+ C_1.\end{aligned} \tag{41}$$

At the same time, the conditional form can be written as:

$$\begin{aligned}\mathcal{J}_{con}(\theta) =& \mathbb{E}_{q_\sigma(\tilde{x},x)}[||s_\theta(\tilde{x}) - \frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}}||^2] \\ =& \mathbb{E}_{q_\sigma(\tilde{x})}[||s_\theta(\tilde{x})||^2] \\ &- 2\mathbb{E}_{q_\sigma(\tilde{x},x)}[< s_\theta(\tilde{x}), \frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} >] \\ &+ C_2,\end{aligned} \tag{42}$$

where $C_2 = \mathbb{E}_{q_\sigma(\tilde{x},x)}[||\frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}}||^2]$.

In conclusion, the two objectives are equivalent in optimising θ because

$$\mathcal{J}_{uncon}(\theta) = \mathcal{J}_{con}(\theta) - C_2 + C_1. \tag{43}$$

REFERENCES

[1] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning*. PMLR, 2015, pp. 2256–2265.

[2] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi, "Palette: Image-to-image diffusion models," in *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*, 2022, pp. 1–10.

[3] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

[4] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Cascaded diffusion models for high fidelity image generation." *J. Mach. Learn. Res.*, vol. 23, pp. 47–1, 2022.

[5] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[6] B. Kim, Y. Oh, and J. C. Ye, "Diffusion Adversarial Representation Learning for Self-supervised Vessel Segmentation," Sep. 2022, arXiv:2209.14566 [cs, eess].

[7] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley, "AudioLDM: Text-to-Audio Generation with Latent Diffusion Models," Jan. 2023, arXiv:2301.12503 [cs, eess].

[8] G. Mittal, J. Engel, C. Hawthorne, and I. Simon, "Symbolic music generation with diffusion models," *arXiv preprint arXiv:2103.16091*, 2021.

[9] Y.-J. Lu, Y. Tsao, and S. Watanabe, "A study on speech enhancement based on diffusion probabilistic model," in *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2021, pp. 659–666.

[10] G. Nam, M. Khlifi, A. Rodriguez, A. Tono, L. Zhou, and P. Guerrero, "3D-LDM: Neural Implicit 3D Shape Generation with Latent Diffusion Models," Dec. 2022, arXiv:2212.00842 [cs].

[11] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-Or, and A. H. Bermano, "Human motion diffusion model," *arXiv preprint arXiv:2209.14916*, 2022.

[12] D. Zhou, W. Wang, H. Yan, W. Lv, Y. Zhu, and J. Feng, "MagicVideo: Efficient Video Generation With Latent Diffusion Models," Nov. 2022, arXiv:2211.11018 [cs].

[13] H. Huang, L. Sun, B. Du, and W. Lv, "Conditional diffusion based on discrete graph structures for molecular graph generation," *arXiv preprint arXiv:2301.00427*, 2023.

[14] S. Luo, C. Shi, M. Xu, and J. Tang, "Predicting molecular conformation via dynamic graph score matching," *Advances in Neural Information Processing Systems*, vol. 34, pp. 19 784–19 795, 2021.

[15] B. Rémy, F. Lanusse, Z. Ramzi, J. Liu, N. Jeffrey, and J.-L. Starck, "Probabilistic mapping of dark matter by neural score matching," *arXiv preprint arXiv:2011.08271*, 2020.

[16] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[18] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," *arXiv preprint arXiv:2206.00364*, 2022.

[19] D. McAllester, "On the mathematics of diffusion models," *arXiv preprint arXiv:2301.11108*, 2023.

[20] H. Cao, C. Tan, Z. Gao, G. Chen, P.-A. Heng, and S. Z. Li, "A survey on generative diffusion model," *arXiv preprint arXiv:2209.02646*, 2022.

[21] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, Y. Shao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *arXiv preprint arXiv:2209.00796*, 2022.

[22] C. Luo, "Understanding diffusion models: A unified perspective," 2022.

[23] W. Fan, C. Liu, Y. Liu, J. Li, H. Li, H. Liu, J. Tang, and Q. Li, "Generative diffusion models on graphs: Methods and applications," *arXiv preprint arXiv:2302.02591*, 2023.

[24] A. Ulhaq, N. Akhtar, and G. Pogrebna, "Efficient diffusion models for vision: A survey," *arXiv preprint arXiv:2210.09292*, 2022.

[25] Z. Guo, J. Liu, Y. Wang, M. Chen, D. Wang, D. Xu, and J. Cheng, "Diffusion models in bioinformatics: A new wave of deep learning revolution in action," *arXiv preprint arXiv:2302.10907*, 2023.

[26] A. Kazerouni, E. K. Aghdam, M. Heidari, R. Azad, M. Fayyaz, I. Hacihaliloglu, and D. Merhof, "Diffusion models for medical image analysis: A comprehensive survey," *arXiv preprint arXiv:2211.07804*, 2022.

[27] C. Zhang, C. Zhang, S. Zheng, M. Zhang, M. Qamar, S.-H. Bae, and I. S. Kweon, "A survey on audio diffusion models: Text to speech synthesis and enhancement in generative ai," *arXiv preprint arXiv:2303.13336*, vol. 2, 2023.

[28] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *arXiv preprint arXiv:2209.04747*, 2022.

[29] C. Zhang, C. Zhang, M. Zhang, and I. S. Kweon, "Text-to-image diffusion model in generative ai: A survey," *arXiv preprint arXiv:2303.07909*, 2023.

[30] Y. Zhu and Y. Zhao, "Diffusion models in nlp: A survey," *arXiv preprint arXiv:2303.07576*, 2023.

[31] 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," *arXiv preprint arXiv:2206.05173*, 2022.

[32] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[33] S. J. Prince, *Understanding Deep Learning*. MIT Press, 2023.

[34] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[35] T. Pang, C. Lu, C. Du, M. Lin, S. Yan, and Z. Deng, "On calibrating diffusion probabilistic models," *arXiv preprint arXiv:2302.10688*, 2023.

[36] K. Pandey, A. Mukherjee, P. Rai, and A. Kumar, "Vaes meet diffusion models: Efficient and high-fidelity generation," in *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[37] Z. G. Zhang, *Fundamentals of Stochastic Models*. CRC Press, 2023.

[38] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8780–8794, 2021.

[39] B. D. Anderson, "Reverse-time diffusion equation models," *Stochastic Processes and their Applications*, vol. 12, no. 3, pp. 313–326, 1982.

[40] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching." *Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

[41] T. Chen, "On the importance of noise scheduling for diffusion models," *arXiv preprint arXiv:2301.10972*, 2023.

[42] E. Nachmani, R. S. Roman, and L. Wolf, "Non gaussian denoising diffusion models," *arXiv preprint arXiv:2106.07582*, 2021.

[43] T. Dockhorn, A. Vahdat, and K. Kreis, "Score-based generative modeling with critically-damped langevin diffusion," *arXiv preprint arXiv:2112.07068*, 2021.

[44] 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.

[45] E. Nachmani, R. S. Roman, and L. Wolf, "Denoising diffusion gamma models," *arXiv preprint arXiv:2110.05948*, 2021.

[46] M. Yi, J. Sun, and Z. Li, "On the generalization of diffusion model," 2023.

[47] V. De Bortoli, "Convergence of denoising diffusion models under the manifold hypothesis," *arXiv preprint arXiv:2208.05314*, 2022.

[48] Z. Xiao, K. Kreis, and A. Vahdat, "Tackling the generative learning trilemma with denoising diffusion gans," *arXiv preprint arXiv:2112.07804*, 2021.

[49] Z. Li, Y. Chen, and F. T. Sommer, "Learning energy-based models in high-dimensional spaces with multi-scale denoising score matching," *arXiv preprint arXiv:1910.07762*, 2019.

[50] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "Wavegrad: Estimating gradients for waveform generation," *arXiv preprint arXiv:2009.00713*, 2020.

[51] C. Meng, Y. Song, W. Li, and S. Ermon, "Estimating high order gradients of the data distribution by denoising," *Advances in Neural Information Processing Systems*, vol. 34, pp. 25 359–25 369, 2021.

[52] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," *Advances in neural information processing systems*, vol. 33, pp. 12 438–12 448, 2020.

[53] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8162–8171.

[54] D. Kingma, T. Salimans, B. Poole, and J. Ho, "Variational diffusion models," *Advances in neural information processing systems*, vol. 34, pp. 21 696–21 707, 2021.

[55] A. Jabri, D. Fleet, and T. Chen, "Scalable adaptive computation for iterative generation," *arXiv preprint arXiv:2212.11972*, 2022.

[56] Q. Huang, D. S. Park, T. Wang, T. I. Denk, A. Ly, N. Chen, Z. Zhang, Z. Zhang, J. Yu, C. Frank *et al.*, "Noise2music: Text-conditioned music generation with diffusion models," *arXiv preprint arXiv:2302.03917*, 2023.

[57] A. Bansal, E. Borgnia, H.-M. Chu, J. S. Li, H. Kazemi, F. Huang, M. Goldblum, J. Geiping, and T. Goldstein, "Cold diffusion: Inverting arbitrary image transforms without noise," *arXiv preprint arXiv:2208.09392*, 2022.

[58] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural computation*, vol. 23, no. 7, pp. 1661–1674, 2011.

[59] V. Voleti, C. Pal, and A. Oberman, "Score-based denoising diffusion with non-isotropic gaussian noise models," *arXiv preprint arXiv:2210.12254*, 2022.

[60] S. Ge, S. Nah, G. Liu, T. Poon, A. Tao, B. Catanzaro, D. Jacobs, J.-B. Huang, M.-Y. Liu, and Y. Balaji, "Preserve your own correlation: A noise prior for video diffusion models," *arXiv preprint arXiv:2305.10474*, 2023.

[61] Z. Kong and W. Ping, "On fast sampling of diffusion probabilistic models," *arXiv preprint arXiv:2106.00132*, 2021.

[62] S.-g. Lee, H. Kim, C. Shin, X. Tan, C. Liu, Q. Meng, T. Qin, W. Chen, S. Yoon, and T.-Y. Liu, "Priorgrad: Improving conditional denoising diffusion models with data-driven adaptive prior," *arXiv preprint arXiv:2106.06406*, 2021.

[63] L. Zheng, J. Yuan, L. Yu, and L. Kong, "A reparameterized discrete diffusion model for text generation," *arXiv preprint arXiv:2302.05737*, 2023.

[64] H. Zheng, P. He, W. Chen, and M. Zhou, "Truncated diffusion probabilistic models and diffusion-based adversarial autoencoders," in *The Eleventh International Conference on Learning Representations*, 2023.

[65] H. Chung, B. Sim, and J. C. Ye, "Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 413–12 422.

[66] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov, "Grad-tts: A diffusion probabilistic model for text-to-speech," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8599–8608.

[67] J. Liu, C. Li, Y. Ren, F. Chen, and Z. Zhao, "Diffsinger: Singing voice synthesis via shallow diffusion mechanism," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, 2022, pp. 11 020–11 028.

[68] H. Zheng, P. He, W. Chen, and M. Zhou, "Truncated diffusion probabilistic models and diffusion-based adversarial autoencoders," *arXiv preprint arXiv:2202.09671*, 2022.

[69] Z. Lyu, X. Xu, C. Yang, D. Lin, and B. Dai, "Accelerating diffusion models via early stop of the diffusion process," *arXiv preprint arXiv:2205.12524*, 2022.

[70] P. Avdeyev, C. Shi, Y. Tan, K. Dudnyk, and J. Zhou, "Dirichlet diffusion score model for biological sequence generation," *arXiv preprint arXiv:2305.10699*, 2023.

[71] Y. Duan, J. Zhou, Z. Wang, Y.-C. Chang, Y.-K. Wang, and C.-T. Lin, "Domain-specific denoising diffusion probabilistic models for brain dynamics," *arXiv preprint arXiv:2305.04200*, 2023.

[72] H.-Y. Choi, S.-H. Lee, and S.-W. Lee, "Dddm-vc: Decoupled denoising diffusion models with disentangled representation and prior mixup for verified robust voice conversion," 2023.

[73] B. Jing, G. Corso, R. Berlinghieri, and T. Jaakkola, "Subspace diffusion generative models," *arXiv preprint arXiv:2205.01490*, 2022.

[74] G. Batzolis, J. Stanczuk, C.-B. Schönlieb, and C. Etmann, "Non-uniform diffusion models," *arXiv preprint arXiv:2207.09786*, 2022.

[75] N. Liu, S. Li, Y. Du, A. Torralba, and J. B. Tenenbaum, "Compositional visual generation with composable diffusion models," *arXiv preprint arXiv:2206.01714*, 2022.

[76] J. Jo, S. Lee, and S. J. Hwang, "Score-based generative modeling of graphs via the system of stochastic differential equations," *arXiv preprint arXiv:2202.02514*, 2022.

[77] G. Batzolis, J. Stanczuk, C.-B. Schönlieb, and C. Etmann, "Conditional image generation with score-based diffusion models," *arXiv preprint arXiv:2111.13606*, 2021.

[78] S. Xu, "Clip-diffusion-lm: Apply diffusion model on image captioning," *arXiv preprint arXiv:2210.04559*, 2022.

[79] V. De Bortoli, E. Mathieu, M. Hutchinson, J. Thornton, Y. W. Teh, and A. Doucet, "Riemannian score-based generative modeling," *arXiv preprint arXiv:2202.02763*, 2022.

[80] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[81] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling, "Argmax flows and multinomial diffusion: Learning categorical distributions," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 454–12 465, 2021.

[82] J. Lee, W. Im, S. Lee, and S.-E. Yoon, "Diffusion probabilistic models for scene-scale 3d categorical data," *arXiv preprint arXiv:2301.00527*, 2023.

[83] J. E. Santos, Z. R. Fox, N. Lubbers, and Y. T. Lin, "Blackout diffusion: Generative diffusion models in discrete-state spaces," *arXiv preprint arXiv:2305.11089*, 2023.

[84] T. Chen, R. Zhang, and G. Hinton, "Analog bits: Generating discrete data using diffusion models with self-conditioning," *arXiv preprint arXiv:2208.04202*, 2022.

[85] S. Gong, M. Li, J. Feng, Z. Wu, and L. Kong, "Diffuseq: Sequence to sequence text generation with diffusion models," *arXiv preprint arXiv:2210.08933*, 2022.

[86] S. Dieleman, L. Sartran, A. Roshannai, N. Savinov, Y. Ganin, P. H. Richemond, A. Doucet, R. Strudel, C. Dyer, C. Durkan *et al.*, "Continuous diffusion for categorical data," *arXiv preprint arXiv:2211.15089*, 2022.

[87] M. Reid, V. J. Hellendoorn, and G. Neubig, "Diffuser: Diffusion via edit-based reconstruction," in *The Eleventh International Conference on Learning Representations*, 2022.

[88] Y.-H. Park, M. Kwon, J. Jo, and Y. Uh, "Unsupervised discovery of semantic latent directions in diffusion models," *arXiv preprint arXiv:2302.12469*, 2023.

[89] J. Thornton, M. Hutchinson, E. Mathieu, V. De Bortoli, Y. W. Teh, and A. Doucet, "Riemannian diffusion schr\" odinger bridge," *arXiv preprint arXiv:2207.03024*, 2022.

[90] C.-W. Huang, M. Aghajohari, A. J. Bose, P. Panangaden, and A. Courville, "Riemannian diffusion models," *arXiv preprint arXiv:2208.07949*, 2022.

[91] K. Abstreiter, S. Mittal, S. Bauer, B. Schölkopf, and A. Mehrjou, "Diffusion-Based Representation Learning," Aug. 2022, arXiv:2105.14257 [cs].

[92] K. Pandey, A. Mukherjee, P. Rai, and A. Kumar, "Diffusevae: Efficient, controllable and high-fidelity generation from low-dimensional latents," *arXiv preprint arXiv:2201.00308*, 2022.

[93] K. Preechakul, N. Chatthee, S. Wizadwongsa, and S. Suwajanakorn, "Diffusion autoencoders: Toward a meaningful and decodable representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 619–10 629.

[94] M. Kwon, J. Jeong, and Y. Uh, "Diffusion Models already have a Semantic Latent Space," Oct. 2022, arXiv:2210.10960 [cs].

[95] C. H. Wu and F. De la Torre, "Unifying diffusion models' latent space, with applications to cyclediffusion and guidance," *arXiv preprint arXiv:2210.05559*, 2022.

[96] D. Kim, S. Shin, K. Song, W. Kang, and I.-C. Moon, "Soft truncation: A universal training technique of score-based diffusion model for high precision score estimation," in *International Conference on Machine Learning*. PMLR, 2022, pp. 11 201–11 228.

[97] H. Cao, J. Wang, T. Ren, X. Qi, Y. Chen, Y. Yao, and L. Zhang, "Exploring vision transformers as diffusion learners," *arXiv preprint arXiv:2212.13771*, 2022.

[98] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," *arXiv preprint arXiv:2202.00512*, 2022.

[99] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[100] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[101] H. Phung, Q. Dao, and A. Tran, "Wavelet diffusion models are fast and scalable image generators," 2023.

[102] R. O'Connor, "Introduction to diffusion models for machine learning," Sep 2022.

[103] 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*, vol. 63, no. 11, pp. 139–144, 2020.

[104] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 ieee information theory workshop (itw)*. IEEE, 2015, pp. 1–5.

[105] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," *arXiv preprint arXiv:1701.05517*, 2017.

[106] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[107] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[108] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[109] C.-F. R. Chen, Q. Fan, and R. Panda, "Crossvit: Cross-attention multi-scale vision transformer for image classification," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 357–366.

[110] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *arXiv preprint arXiv:2205.11487*, 2022.

[111] P. Zhuang, S. Abnar, J. Gu, A. Schwing, J. M. Susskind, and M. Á. Bautista, "Diffusion probabilistic fields," in *The Eleventh International Conference on Learning Representations*, 2023.

[112] H. Lu, G. Yang, N. Fei, Y. Huo, Z. Lu, P. Luo, and M. Ding, "Vdt: An empirical study on video diffusion with transformers," 2023.

[113] X. Yang, S.-M. Shih, Y. Fu, X. Zhao, and S. Ji, "Your vit is secretly a hybrid discriminative-generative diffusion model," *arXiv preprint arXiv:2208.07791*, 2022.

[114] X. Jing, Y. Chang, Z. Yang, J. Xie, A. Triantafyllopoulos, and B. W. Schuller, "U-dit tts: U-diffusion vision transformer for text-to-speech," 2023.

[115] F. Bao, C. Li, Y. Cao, and J. Zhu, "All are worth words: a vit backbone for score-based diffusion models," *arXiv preprint arXiv:2209.12152*, 2022.

[116] P. Chahal, "Exploring transformer backbones for image diffusion models," *arXiv preprint arXiv:2212.14678*, 2022.

[117] S. Bond-Taylor, P. Hessey, H. Sasaki, T. P. Breckon, and C. G. Willcocks, "Unleashing transformers: parallel token prediction with discrete absorbing diffusion for fast high-resolution image generation from vector-quantized codes," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII*. Springer, 2022, pp. 170–188.

[118] W. Peebles and S. Xie, "Scalable diffusion models with transformers," *arXiv preprint arXiv:2212.09748*, 2022.

[119] F. Giuliari, G. Scarpellini, S. James, Y. Wang, and A. Del Bue, "Positional diffusion: Ordering unordered sets with diffusion probabilistic models," *arXiv preprint arXiv:2303.11120*, 2023.

[120] H. Liu, R. Huang, X. Lin, W. Xu, M. Zheng, H. Chen, J. He, and Z. Zhao, "Vit-tts: Visual text-to-speech with scalable diffusion transformer," 2023.

[121] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[122] F. Bao, S. Nie, K. Xue, C. Li, S. Pu, Y. Wang, G. Yue, Y. Cao, H. Su, and J. Zhu, "One transformer fits all distributions in multi-modal diffusion at scale," *arXiv preprint arXiv:2303.06555*, 2023.

[123] J. Guan, W. W. Qian, X. Peng, Y. Su, J. Peng, and J. Ma, "3d equivariant diffusion for target-aware molecule generation and affinity prediction," *arXiv preprint arXiv:2303.03543*, 2023.

[124] Y. Benny and L. Wolf, "Dynamic dual-output diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 482–11 491.

[125] Y. Song, S. Garg, J. Shi, and S. Ermon, "Sliced score matching: A scalable approach to density and score estimation," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 574–584.

[126] D. Chen, Z. Zhou, J.-P. Mei, C. Shen, C. Chen, and C. Wang, "A geometric perspective on diffusion models," 2023.

[127] Y. Song and D. P. Kingma, "How to train your energy-based models," *arXiv preprint arXiv:2101.03288*, 2021.

[128] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[129] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang, "Geodiff: A geometric diffusion model for molecular conformation generation," *arXiv preprint arXiv:2203.02923*, 2022.

[130] C.-W. Huang, J. H. Lim, and A. C. Courville, "A variational perspective on diffusion-based generative models and score matching," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 863–22 876, 2021.

[131] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet *et al.*, "Imagen video: High definition video generation with diffusion models," *arXiv preprint arXiv:2210.02303*, 2022.

[132] S. Lin, B. Liu, J. Li, and X. Yang, "Common diffusion noise schedules and sample steps are flawed," *arXiv preprint arXiv:2305.08891*, 2023.

[133] K. Zheng, C. Lu, J. Chen, and J. Zhu, "Improved techniques for maximum likelihood estimation for diffusion odes," *arXiv preprint arXiv:2305.03935*, 2023.

[134] K. Oko, S. Akiyama, and T. Suzuki, "Diffusion models are minimax optimal distribution estimators," *arXiv preprint arXiv:2303.01861*, 2023.

[135] 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.

[136] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[137] 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.

[138] J. Singh, S. Gould, and L. Zheng, "High-Fidelity Guided Image Synthesis with Latent Diffusion Models," Nov. 2022, arXiv:2211.17084 [cs, stat].

[139] E. Hedlin, G. Sharma, S. Mahajan, H. Isack, A. Kar, A. Tagliasacchi, and K. M. Yi, "Unsupervised semantic correspondence using stable diffusion," 2023.

[140] K. Deja, A. Kuzina, T. Trzciński, and J. M. Tomczak, "On analyzing generative and denoising capabilities of diffusion-based deep generative models," *arXiv preprint arXiv:2206.00070*, 2022.

[141] J. Zhang, C. Herrmann, J. Hur, L. P. Cabrera, V. Jampani, D. Sun, and M.-H. Yang, "A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence," 2023.

[142] G. Luo, L. Dunlap, D. H. Park, A. Holynski, and T. Darrell, "Diffusion hyperfeatures: Searching through time and space for semantic correspondence," 2023.

[143] J. Choi, J. Lee, C. Shin, S. Kim, H. Kim, and S. Yoon, "Perception prioritized training of diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 472–11 481.

[144] B. Krojer, E. Poole-Dayan, V. Voleti, C. Pal, and S. Reddy, "Are diffusion models vision-and-language reasoners?" 2023.

[145] A. Vahdat, K. Kreis, and J. Kautz, "Score-based generative modeling in latent space," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11 287–11 302, 2021.

[146] Y. Song, C. Durkan, I. Murray, and S. Ermon, "Maximum likelihood training of score-based diffusion models," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1415–1428, 2021.

[147] S. Chen, S. Chewi, J. Li, Y. Li, A. Salim, and A. R. Zhang, "Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions," *arXiv preprint arXiv:2209.11215*, 2022.

[148] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[149] A. Jolicoeur-Martineau, K. Li, R. Piché-Taillefer, T. Kachman, and I. Mitliagkas, "Gotta go fast when generating data with score-based models," *arXiv preprint arXiv:2105.14080*, 2021.

[150] 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.

[151] W. Cho, H. Ravi, M. Harikumar, V. Khuc, K. K. Singh, J. Lu, D. I. Inouye, and A. Kale, "Towards enhanced controllability of diffusion models," *arXiv preprint arXiv:2302.14368*, 2023.

[152] K. Deja, T. Trzcinski, and J. M. Tomczak, "Learning Data Representations with Joint Diffusion Models," Jan. 2023, arXiv:2301.13622 [cs, stat].

[153] Y. He, Z. Cai, X. Gan, and B. Chang, "Diffcap: Exploring continuous diffusion on image captioning," *arXiv preprint arXiv:2305.12144*, 2023.

[154] Z. Xue, G. Song, Q. Guo, B. Liu, Z. Zong, Y. Liu, and P. Luo, "Raphael: Text-to-image generation via large mixture of diffusion paths," 2023.

[155] L. Zhang and M. Agrawala, "Adding conditional control to text-to-image diffusion models," *arXiv preprint arXiv:2302.05543*, 2023.

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

[157] A. Bansal, H.-M. Chu, A. Schwarzschild, S. Sengupta, M. Goldblum, J. Geiping, and T. Goldstein, "Universal guidance for diffusion models," *arXiv preprint arXiv:2302.07121*, 2023.

[158] P.-C. Chen, H. Tsai, S. Bhojanapalli, H. W. Chung, Y.-W. Chang, and C.-S. Ferng, "A simple and effective positional encoding for transformers," *arXiv preprint arXiv:2104.08698*, 2021.

[159] O. Avrahami, O. Fried, and D. Lischinski, "Blended latent diffusion," Jun. 2022, arXiv:2206.02779 [cs].

[160] L. Zhou, Y. Du, and J. Wu, "3d shape generation and completion through point-voxel diffusion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5826–5835.

[161] Z. Lyu, Z. Kong, X. Xu, L. Pan, and D. Lin, "A conditional point diffusion-refinement paradigm for 3d point cloud completion," *arXiv preprint arXiv:2112.03530*, 2021.

[162] Z. Chang, E. J. Findlay, H. Zhang, and H. P. Shum, "Unifying human motion synthesis and style transfer with denoising diffusion probabilistic models," *arXiv preprint arXiv:2212.08526*, 2022.

[163] H. Chefer, Y. Alaluf, Y. Vinker, L. Wolf, and D. Cohen-Or, "Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models," *arXiv preprint arXiv:2301.13826*, 2023.

[164] W.-D. K. Ma, J. Lewis, W. B. Kleijn, and T. Leung, "Directed diffusion: Direct control of object placement through attention guidance," *arXiv preprint arXiv:2302.13153*, 2023.

[165] S. Hong, G. Lee, W. Jang, and S. Kim, "Improving sample quality of diffusion models using self-attention guidance," *arXiv preprint arXiv:2210.00939*, 2022.

[166] C. Peng, P. Guo, S. K. Zhou, V. Patel, and R. Chellappa, "Towards performant and reliable undersampled mr reconstruction via diffusion model sampling," *arXiv preprint arXiv:2203.04292*, 2022.

[167] M. W. Shen, E. Hajiramezanali, G. Scalia, A. Tseng, N. Diamant, T. Biancalani, and A. Loukas, "Conditional diffusion with less explicit guidance via model predictive control," *arXiv preprint arXiv:2210.12192*, 2022.

[168] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2484–2493.

[169] C. Guo, A. Sablayrolles, H. Jégou, and D. Kiela, "Gradient-based adversarial attacks against text transformers," *arXiv preprint arXiv:2104.13733*, 2021.

[170] B. Kawar, R. Ganz, and M. Elad, "Enhancing diffusion-based image synthesis with robust classifier guidance," *arXiv preprint arXiv:2208.08664*, 2022.

[171] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug & play generative networks: Conditional iterative generation of images in latent space," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4467–4477.

[172] S. Dodge and L. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," in *2017 26th international conference on computer communication and networks (ICCCN)*. IEEE, 2017, pp. 1–7.

[173] H. Hosseini, B. Xiao, and R. Poovendran, "Google's cloud vision api is not robust to noise," in *2017 16th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2017, pp. 101–105.

[174] B. Wallace, A. Gokul, S. Ermon, and N. Naik, "End-to-end diffusion latent optimization improves classifier guidance," *arXiv preprint arXiv:2303.13703*, 2023.

[175] P. Grünwald and J. Langford, "Suboptimal behavior of bayes and mdl in classification under misspecification," in *Learning Theory: 17th Annual Conference on Learning Theory, COLT 2004, Banff, Canada, July 1-4, 2004. Proceedings 17*. Springer, 2004, pp. 331–347.

[176] S. Srinivas and F. Fleuret, "Rethinking the role of gradient-based attribution methods for model interpretability," *arXiv preprint arXiv:2006.09128*, 2020.

[177] C.-H. Chao, W.-F. Sun, B.-W. Cheng, Y.-C. Lo, C.-C. Chang, Y.-L. Liu, Y.-L. Chang, C.-P. Chen, and C.-Y. Lee, "Denoising likelihood score matching for conditional score-based data generation," *arXiv preprint arXiv:2203.14206*, 2022.

[178] J. Zhao, H. Zheng, C. Wang, L. Lan, W. Huang, and W. Yang, "Null-text guidance in diffusion models is secretly a cartoon-style creator," *arXiv preprint arXiv:2305.06710*, 2023.

[179] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann *et al.*, "Imitating human behaviour with diffusion models," *arXiv preprint arXiv:2301.10677*, 2023.

[180] D. Epstein, A. Jabri, B. Poole, A. A. Efros, and A. Holynski, "Diffusion self-guidance for controllable image generation," 2023.

[181] C. Mou, X. Wang, L. Xie, J. Zhang, Z. Qi, Y. Shan, and X. Qie, "T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models," *arXiv preprint arXiv:2302.08453*, 2023.

[182] C. Xiang, F. Bao, C. Li, H. Su, and J. Zhu, "A closer look at parameter-efficient tuning in diffusion models," *arXiv preprint arXiv:2303.18181*, 2023.

[183] S. Zhao, D. Chen, Y.-C. Chen, J. Bao, S. Hao, L. Yuan, and K.-Y. K. Wong, "Uni-controlnet: All-in-one control to text-to-image diffusion models," 2023.

[184] D. Yang, S. Liu, J. Yu, H. Wang, C. Weng, and Y. Zou, "Norespeech: Knowledge distillation based conditional diffusion model for noise-robust expressive tts," *arXiv preprint arXiv:2211.02448*, 2022.

[185] B. Tzen and M. Raginsky, "Theoretical guarantees for sampling and inference in generative models with latent diffusions," in *Conference on Learning Theory*. PMLR, 2019, pp. 3084–3114.

[186] G. Liu, H. Sun, J. Li, F. Yin, and Y. Yang, "Accelerating diffusion models for inverse problems through shortcut sampling," 2023.

[187] S. W. Park, D. W. Shu, and J. Kwon, "Generative adversarial networks for markovian temporal dynamics: Stochastic continuous data generation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8413–8421.

[188] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[189] W. Sun, D. Chen, C. Wang, D. Ye, Y. Feng, and C. Chen, "Accelerating diffusion sampling with classifier-based feature distillation," *arXiv preprint arXiv:2211.12039*, 2022.

[190] T. Huang, Y. Zhang, M. Zheng, S. You, F. Wang, C. Qian, and C. Xu, "Knowledge diffusion for distillation," 2023.

[191] E. Luhman and T. Luhman, "Knowledge distillation in iterative generative models for improved sampling speed," *arXiv preprint arXiv:2101.02388*, 2021.

[192] H. Zheng, W. Nie, A. Vahdat, K. Azizzadenesheli, and A. Anandkumar, "Fast sampling of diffusion models via operator learning," *arXiv preprint arXiv:2211.13449*, 2022.

[193] C. Meng, R. Gao, D. P. Kingma, S. Ermon, J. Ho, and T. Salimans, "On distillation of guided diffusion models," *arXiv preprint arXiv:2210.03142*, 2022.

[194] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," *arXiv preprint arXiv:2303.01469*, 2023.

[195] W. Luo, "A comprehensive survey on knowledge distillation of diffusion models," *arXiv preprint arXiv:2304.04262*, 2023.

[196] F. Vargas, W. S. Grathwohl, and A. Doucet, "Denoising diffusion samplers," in *The Eleventh International Conference on Learning Representations*, 2023.

[197] D. Watson, W. Chan, J. Ho, and M. Norouzi, "Learning fast samplers for diffusion models by differentiating through sample quality," in *International Conference on Learning Representations*, 2021.

[198] P. A. Golnari, Z. Yao, and Y. He, "Selective guidance: Are all the denoising steps of guided diffusion important?" *arXiv preprint arXiv:2305.09847*, 2023.

[199] A. Shih, S. Belkhale, S. Ermon, D. Sadigh, and N. Anari, "Parallel sampling of diffusion models," 2023.

[200] Q. Zhang and Y. Chen, "Fast sampling of diffusion models with exponential integrator," *arXiv preprint arXiv:2204.13902*, 2022.

[201] H. Tachibana, M. Go, M. Inahara, Y. Katayama, and Y. Watanabe, "It\^{o}-taylor sampling scheme for denoising diffusion probabilistic models using ideal derivatives," *arXiv preprint arXiv:2112.13339*, 2021.

[202] M. Li, T. Qu, W. Sun, and M.-F. Moens, "Alleviating exposure bias in diffusion models through sampling with shifted time steps," 2023.

[203] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps," *arXiv preprint arXiv:2206.00927*, 2022.

[204] G. Fang, X. Ma, and X. Wang, "Structural pruning for diffusion models," *arXiv preprint arXiv:2305.10924*, 2023.

[205] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, "Image super-resolution via iterative refinement," *arXiv preprint arXiv:2104.07636*, 2021.

[206] R. San-Roman, E. Nachmani, and L. Wolf, "Noise estimation for generative diffusion models," *arXiv preprint arXiv:2104.02600*, 2021.

[207] Z. Duan, C. Wang, C. Chen, J. Huang, and W. Qian, "Optimal linear subspace search: Learning to construct fast and high-quality schedulers for diffusion models," 2023.

[208] D. Watson, J. Ho, M. Norouzi, and W. Chan, "Learning to efficiently sample from diffusion probabilistic models," *arXiv preprint arXiv:2106.03802*, 2021.

[209] A. Campbell, W. Harvey, C. Weilbach, V. D. Bortoli, T. Rainforth, and A. Doucet, "Trans-dimensional generative modeling via jump diffusion models," 2023.

[210] J.-Y. Franceschi, M. Gartrell, L. D. Santos, T. Issenhuth, E. de Bézenac, M. Chen, and A. Rakotomamonjy, "Unifying gans and score-based diffusion as generative particle models," 2023.

[211] F. Vargas, T. Reu, and A. Kerekes, "Expressiveness remarks for denoising diffusion models and samplers," *arXiv preprint arXiv:2305.09605*, 2023.

[212] S. Lee, B. Hoover, H. Strobelt, Z. J. Wang, S. Peng, A. Wright, K. Li, H. Park, H. Yang, and D. H. Chau, "Diffusion explainer: Visual explanation for text-to-image stable diffusion," *arXiv preprint arXiv:2305.03509*, 2023.

[213] B. Han, H. Peng, M. Dong, C. Xu, Y. Ren, Y. Shen, and Y. Li, "Amd autoregressive motion diffusion," *arXiv preprint arXiv:2305.09381*, 2023.

[214] T. Wu, Z. Fan, X. Liu, Y. Gong, Y. Shen, J. Jiao, H.-T. Zheng, J. Li, Z. Wei, J. Guo *et al.*, "Ar-diffusion: Auto-regressive diffusion model for text generation," *arXiv preprint arXiv:2305.09515*, 2023.

[215] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," *arXiv preprint arXiv:2009.09761*, 2020.

[216] B.-K. Kim, H.-K. Song, T. Castells, and S. Choi, "On architectural compression of text-to-image diffusion models," 2023.

[217] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine, "Training diffusion models with reinforcement learning," 2023.

[218] G. Giannone, D. Nielsen, and O. Winther, "Few-shot diffusion models," *arXiv preprint arXiv:2205.15463*, 2022.

[219] B. Kawar, N. Elata, T. Michaeli, and M. Elad, "Gsure-based diffusion model training with corrupted data," 2023.

[220] G. Daras, K. Shah, Y. Dagan, A. Gollakota, A. G. Dimakis, and A. Klivans, "Ambient diffusion: Learning clean distributions from corrupted data," 2023.

[221] J. Chen, R. Zhang, T. Yu, R. Sharma, Z. Xu, T. Sun, and C. Chen, "Label-retrieval-augmented diffusion models for learning from noisy labels," 2023.

[222] W. Hu, X. Jiang, J. Liu, Y. Yang, and H. Tian, "Meta-dm: Applications of diffusion models on few-shot learning," *arXiv preprint arXiv:2305.08092*, 2023.

[223] J. Zhu, H. Ma, J. Chen, and J. Yuan, "Few-shot image generation with diffusion models," *arXiv preprint arXiv:2211.03264*, 2022.

[224] D. Svitov, D. Gudkov, R. Bashirov, and V. Lemptisky, "Dinar: Diffusion inpainting of neural textures for one-shot human avatars," *arXiv preprint arXiv:2303.09375*, 2023.

[225] F. P. Papantoniou, A. Lattas, S. Moschoglou, and S. Zafeiriou, "Relightify: Relightable 3d faces from a single image via diffusion models," *arXiv preprint arXiv:2305.06077*, 2023.

[226] Z. Wang, Y. Yang, M. Sermesant, H. Delingette, and O. Wu, "Zero-shot-learning cross-modality data translation through mutual information guided stochastic diffusion," *arXiv preprint arXiv:2301.13743*, 2023.

[227] Y. Li, H.-C. Shao, X. Liang, L. Chen, R. Li, S. Jiang, J. Wang, and Y. Zhang, "Zero-shot medical image translation via frequency-guided diffusion models," *arXiv preprint arXiv:2304.02742*, 2023.

[228] M. Kang, W. Han, S. J. Hwang, and E. Yang, "Zet-speech: Zero-shot adaptive emotion-controllable text-to-speech synthesis with diffusion and style-based models," 2023.

[229] A. O. Tur, N. Dall'Asen, C. Beyan, and E. Ricci, "Exploring diffusion models for unsupervised video anomaly detection," *arXiv preprint arXiv:2304.05841*, 2023.

[230] Y. Miao, L. Zhang, L. Zhang, and D. Tao, "Dds2m: Self-supervised denoising diffusion spatio-spectral model for hyperspectral image restoration," *arXiv preprint arXiv:2303.06682*, 2023.

[231] Z. You, Y. Zhong, F. Bao, J. Sun, C. Li, and J. Zhu, "Diffusion models and semi-supervised learners benefit mutually with few labels," *arXiv preprint arXiv:2302.10586*, 2023.

[232] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National science review*, vol. 5, no. 1, pp. 44–53, 2018.

[233] L. Lian, B. Li, A. Yala, and T. Darrell, "Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models," 2023.

[234] Z. Tang, Z. Yang, C. Zhu, M. Zeng, and M. Bansal, "Any-to-any generation via composable diffusion," *arXiv preprint arXiv:2305.11846*, 2023.

[235] C. Qin, S. Zhang, N. Yu, Y. Feng, X. Yang, Y. Zhou, H. Wang, J. C. Niebles, C. Xiong, S. Savarese *et al.*, "Unicontrol: A unified diffusion model for controllable visual generation in the wild," *arXiv preprint arXiv:2305.11147*, 2023.

[236] X. Han, H. Zheng, and M. Zhou, "Card: Classification and regression diffusion models," *arXiv preprint arXiv:2206.07275*, 2022.

[237] L. Yang, Z. Huang, F. Lei, Y. Zhong, Y. Yang, C. Fang, S. Wen, B. Zhou, and Z. Lin, "Policy representation via diffusion probability model for reinforcement learning," 2023.

[238] S. An, H. Lee, J. Jo, S. Lee, and S. J. Hwang, "Diffusionnag: Task-guided neural architecture generation with diffusion models," 2023.

[239] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "Diffusionner: Boundary diffusion for named entity recognition," 2023.

[240] Z. Li, Y. Li, P. Zhao, R. Song, X. Li, and J. Yang, "Is synthetic data from diffusion models ready for knowledge distillation?" 2023.

[241] S. Pan, E. Abouei, J. Wynne, T. Wang, R. L. J. Qiu, Y. Li, C.-W. Chang, J. Peng, J. Roper, P. Patel, D. S. Yu, H. Mao, and X. Yang, "Synthetic ct generation from mri using 3d transformer-based denoising diffusion model," 2023.

[242] Z. Zhang, B. Li, X. Nie, C. Han, T. Guo, and L. Liu, "Towards consistent video editing with text-to-image diffusion models," 2023.

[243] M. Zhang, M. Qamar, T. Kang, Y. Jung, C. Zhang, S.-H. Bae, and C. Zhang, "A survey on graph diffusion models: Generative ai in science for molecule, protein and material," *arXiv preprint arXiv:2304.01565*, 2023.

[244] Y. Hatanaka, Y. Glaser, G. Galgon, G. Torri, and P. Sadowski, "Diffusion models for high-resolution solar forecasts," *arXiv preprint arXiv:2302.00170*, 2023.

[245] A. Shmakov, K. Greif, M. Fenton, A. Ghosh, P. Baldi, and D. Whiteson, "End-to-end latent variational diffusion models for inverse problems in high energy physics," *arXiv preprint arXiv:2305.10399*, 2023.

[246] A. Morehead and J. Cheng, "Geometry-complete diffusion for 3d molecule generation," *arXiv preprint arXiv:2302.04313*, 2023.

[247] C. Fu, K. Yan, L. Wang, W. Y. Au, M. McThrow, T. Komikado, K. Maruhashi, K. Uchino, X. Qian, and S. Ji, "A latent diffusion model for protein structure generation," *arXiv preprint arXiv:2305.04120*, 2023.

[248] G. Raya and L. Ambrogioni, "Spontaneous symmetry breaking in generative diffusion models," 2023.

[249] X. Peng, J. Guan, Q. Liu, and J. Ma, "Moldiff: Addressing the atom-bond inconsistency problem in 3d molecule diffusion generation," *arXiv preprint arXiv:2305.07508*, 2023.

[250] J. Yim, B. L. Trippe, V. De Bortoli, E. Mathieu, A. Doucet, R. Barzilay, and T. Jaakkola, "Se (3) diffusion model with application to protein backbone generation," *arXiv preprint arXiv:2302.02277*, 2023.

[251] C. I. Bercea, M. Neumayr, D. Rueckert, and J. A. Schnabel, "Mask, stitch, and re-sample: Enhancing robustness and generalizability in anomaly detection through automatic diffusion models," 2023.

[252] S. Zhai, Y. Dong, Q. Shen, S. Pu, Y. Fang, and H. Su, "Text-to-image diffusion models can be easily backdoored through multimodal data poisoning," *arXiv preprint arXiv:2305.04175*, 2023.

[253] C. Xiao, Z. Chen, K. Jin, J. Wang, W. Nie, M. Liu, A. Anandkumar, B. Li, and D. Song, "Densepure: Understanding diffusion models towards adversarial robustness," *arXiv preprint arXiv:2211.00322*, 2022.

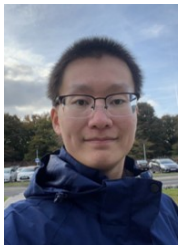
[254] R. Imam, M. Huzaifa, and M. E.-A. Azz, "On enhancing the robustness of vision transformers: Defensive diffusion," *arXiv preprint arXiv:2305.08031*, 2023.

[255] M. Lee and D. Kim, "Robust evaluation of diffusion-based adversarial purification," *arXiv preprint arXiv:2303.09051*, 2023.

[256] T. Altstidl, D. Dobre, B. Eskofier, G. Gidel, and L. Schwinn, "Raising the bar for certified adversarial robustness with diffusion models," *arXiv preprint arXiv:2305.10388*, 2023.

[257] Z. Zhang, D. Chen, H. Zhou, F. Meng, J. Zhou, and X. Sun, "Diffusion theory as a scalpel: Detecting and purifying poisonous dimensions in pre-trained language models caused by backdoor or bias," *arXiv preprint arXiv:2305.04547*, 2023.

[258] J. Chen, H. Chen, K. Chen, Y. Zhang, Z. Zou, and Z. Shi, "Diffusion models for imperceptible and transferable adversarial attack," *arXiv preprint arXiv:2305.08192*, 2023.
[259] H. Xue, A. Araujo, B. Hu, and Y. Chen, "Diffusion-based adversarial sample generation for improved stealthiness and controllability," 2023.
[260] M. A. Franks and A. E. Waldman, "Sex, lies, and videotape: Deep fakes and free speech delusions," *Md. L. Rev.*, vol. 78, p. 892, 2018.
[261] L. Guarnera, O. Giudice, and S. Battiato, "Level up the deepfake detection: a method to effectively discriminate images generated by gan architectures and diffusion models," *arXiv preprint arXiv:2303.00608*, 2023.
[262] Y. Qu, X. Shen, X. He, M. Backes, S. Zannettou, and Y. Zhang, "Unsafe diffusion: On the generation of unsafe images and hateful memes from text-to-image models," 2023.
[263] D. A. Coccomini, A. Esuli, F. Falchi, C. Gennaro, and G. Amato, "Detecting images generated by diffusers," *arXiv preprint arXiv:2303.05275*, 2023.
[264] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Sehwag, F. Tramer, B. Balle, D. Ippolito, and E. Wallace, "Extracting training data from diffusion models," *arXiv preprint arXiv:2301.13188*, 2023.
[265] R. Webster, "A reproducible extraction of training images from diffusion models," *arXiv preprint arXiv:2305.08694*, 2023.
[266] J. Liu, C. P. Lau, and R. Chellappa, "Diffprotect: Generate adversarial examples with diffusion models for facial privacy protection," 2023.
[267] S. Lyu, M. Vinaroz, M. F. Liu, and M. Park, "Differentially private latent diffusion models," 2023.
[268] M. V. Perera and V. M. Patel, "Analyzing bias in diffusion-based face generation models," *arXiv preprint arXiv:2305.06402*, 2023.
[269] S. Um and J. C. Ye, "Don't play favorites: Minority guidance for diffusion models," *arXiv preprint arXiv:2301.12334*, 2023.
[270] V. Sehwag, C. Hazirbas, A. Gordo, F. Ozgenel, and C. Canton, "Generating high fidelity data from low-density regions using diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 492–11 501.
[271] G. Somepalli, V. Singla, M. Goldblum, J. Geiping, and T. Goldstein, "Understanding and mitigating copying in diffusion models," 2023.
[272] L. Dunlap, A. Umino, H. Zhang, J. Yang, J. E. Gonzalez, and T. Darrell, "Diversify your vision datasets with automatic diffusion-based augmentation," 2023.
[273] A. S. Luccioni, C. Akiki, M. Mitchell, and Y. Jernite, "Stable bias: Analyzing societal representations in diffusion models," *arXiv preprint arXiv:2303.11408*, 2023.
[274] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.



Ziyi Chang is a PhD student in the Department of Computer Science at Durham University. His research focuses on human-related data in 3D computer vision and graphics, including 3D reconstruction, 3D style analysis and synthesis, human motion analysis and synthesis, and etc. He graduated from the University of Edinburgh in 2020 and Renmin University of China in 2019.

George A. Koulieris (BSc with Hons, MSc, PhD in Electronic and Computer Engineering) is an Associate Professor and member of the ViVID Group at the Department of Computer Science at Durham University. His primary research interests lie in the field of computer graphics, in particular applied visual perception to rendering and display hardware, with a strong focus on near-eye displays and virtual reality. Before joining Durham in 2018, he was a post-doctoral researcher at Inria, France, team GraphDeco, and visiting scholar at UC Berkeley Vision Science, USA, working on near-eye, stereo displays.

Hubert P. H. Shum (Senior Member, IEEE) is an Associate Professor and the Deputy Director of Research in Computer Science at Durham University, researching on human-centric computer vision and graphics. Before this, he was an Associate Professor at Northumbria University and a Postdoctoral Researcher at RIKEN Japan. He received his PhD degree from the University of Edinburgh. He chaired conferences such as Pacific Graphics, BMVC and SCA, and has authored over 150 research publications.