On the Design Fundamentals of Diffusion Models: A Survey

Ziyi Chang[a], George A. Koulieris[a], Hyung Jin Chang[b], Hubert P. H. Shum[a,*]

[a]*Department of Computer Science, Durham University, Durham, DH1 3LE, UK*
[b]*School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK*

Abstract

Diffusion models are learning pattern-learning systems to model and sample from data distributions with three functional components namely the forward process, the reverse process, and the sampling process. The components of diffusion models have gained significant attention with many design factors being considered in common practice. Existing reviews have primarily focused on higher-level solutions, covering less on the design fundamentals of components. This study seeks to address this gap by providing a comprehensive and coherent review of seminal designable factors within each functional component of diffusion models. This provides a finer-grained perspective of diffusion models, benefiting future studies in the analysis of individual components, the design factors for different purposes, and the implementation of diffusion models.

Keywords: Diffusion Model, Forward Process, Reverse Process, Sampling Process, Deep Learning

1. Introduction

Diffusion models, as a learning system, consist of three functional components, i.e., the forward, reverse, and sampling processes. The generic pipeline of diffusion models [1] involves forward and reverse processes to learn a data distribution, and a sampling process to generate novel data that follow such a distribution. Three com-

*Corresponding author
Email addresses: ziyi.chang@durham.ac.uk (Ziyi Chang), george.koulieris@durham.ac.uk (George A. Koulieris), h.j.chang@bham.ac.uk (Hyung Jin Chang), hubert.shum@durham.ac.uk (Hubert P. H. Shum)

ponents work together to achieve the functionality of diffusion models [2], i.e., the ability to model and sample from data distributions. The forward process is expected to perturb training data by adding noise while the reverse process is expected to remove the aforementioned perturbation via learning a neural network. After optimization, the sampling process is expected to generate novel data that follow the distribution.

Designing the three functional components involves consideration of different major factors and design purposes. To perturb training data, the schedule of noise and the type of noise are two major factors that need to be considered when a forward process is designed. Meanwhile, the terminal point and diffusion space are also required to specify where to stop and where to manipulate data in a forward process. The forward process needs to perturb the training data by adding noise at each timestep. For a reverse process to remove the added noise, architectures, parameterizations, and optimizations of a neural network become major factors to be considered for the reverse process. Architectures specify how to learn denoising while parameterizations concern what to learn and predict. Optimizations allow different emphases on the information that a neural network should focus on more. The sampling process works after optimization, and the controllability and speed become two major factors to be considered. Controllability constrains the generation to obtain data of users' interest while the speed factor accelerates generation without significant quality degradation. While there could be a large number of factors in each functional component, our survey only focuses on major ones that are transferable and usually considered in common practice.

Most existing survey papers on diffusion models focus on a particular application area while ours hierarchically organizes the design fundamentals in a diffusion model. With the recent prosperity in applications of diffusion models, previous survey papers mostly focus on collecting application cases in various domains and data structures. These domains include natural language processing [3], computer vision-related tasks [4], medical analysis [5], natural science [6], time series [7], recommendation [8], personalization [9], memorization [10], and etc. They also cover different types of data, including image [11, 12], text [13], video [14], audio [15], etc. However, they are domain-specific and application-driven, and thus lead to restricted insights into this whole area. Some recent surveys [16, 17, 18] are organized by specific problems,

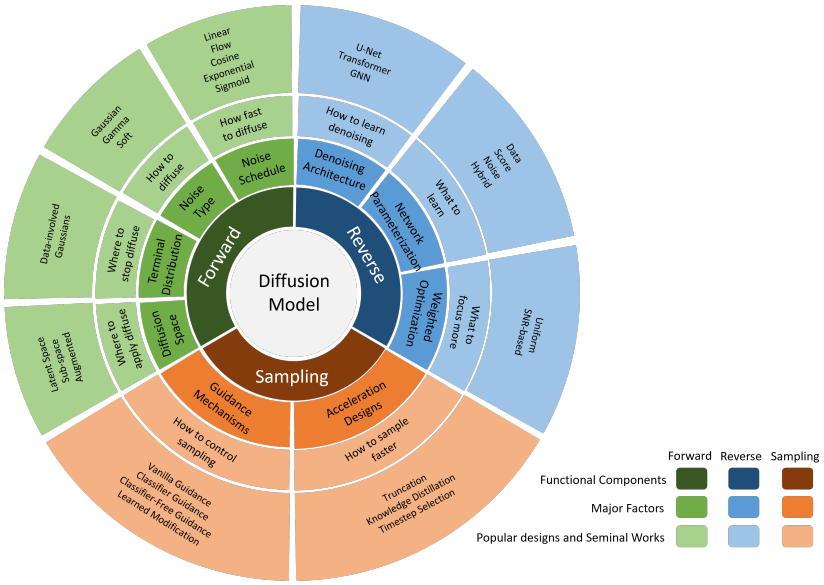


Figure 1: The hierarchical overview of diffusion models. The forward process, the reverse process, and the sampling process are three functional components. Major factors comprise each component. Popular designs and seminal works are presented.

which may hinder a comprehensive understanding. In contrast, our survey adopts a design-centric taxonomy, providing building blocks to facilitate straightforward implementation.

Our paper treats diffusion models as a learning system, discusses the system hierarchically, and mainly focuses on seminal designable factors within each component, as shown in Fig. 1. This breakdown is aligned with the functionality of diffusion models and the intuition of getting to know a system. Therefore, our survey benefits both beginners who want to get fundamental knowledge from seminal works in this area and professionals who want to hierarchically understand critical factors when designing their advanced diffusion models.

The following are questions to answer via our literature review of diffusion models:

1. What are the **functional components** in a diffusion model?
2. What are the **major factors** that comprise each component?
3. What are the **popular designs** and **seminal works** in each factor?

We organize our survey by answering the above questions to build a hierarchical

view of diffusion models. Section 2 introduces the generic pipeline of diffusion models, including the three functional components and two popular formulations. Sections 3, 4, and 5 respectively review the major factors, and popular designs and seminal works of each functional component, i.e., the forward process, the reverse process, and the sampling process. Section 6 provides overall insights on the future trends with respect to this field, and Section 7 gives a brief conclusion on diffusion models.

2. The Generic Pipeline

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

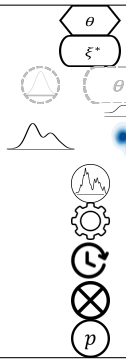
Notation	Legends
	Trainable network with parameters θ Fixed network with parameters ξ^* Component not in use Data distributions The distribution at a timestep Condition c Timestep t Combination, e.g., Addition. With probability p for dropping out

Table 1: Figure legends.

2.1. Three Functional Components

The forward process perturbs a training sample x_0 to $\{x_t\}_{t=1}^T$ as the timestep t increases, as shown in Fig. 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 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. Since only noise is added through the

chain, the forward process does not have any trainable parameters. In particular, the forward process is represented as a chain of forward transitions:

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

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 $p\left(x_t|x_{t-1}\right)$ is a forward distribution transition between two consecutive time steps.

Figure 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 reverse process trains a denoising network to recursively remove the noise, as shown in Fig. 3. 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. In particular, the reverse process is formulated as a chain of reverse transitions:

$$p_\theta\left(x_{0:T}\right) := p\left(x_T\right) \prod_{t=1}^{T} p_\theta\left(x_{t-1}|x_t\right), \tag{2}$$

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:

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

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.

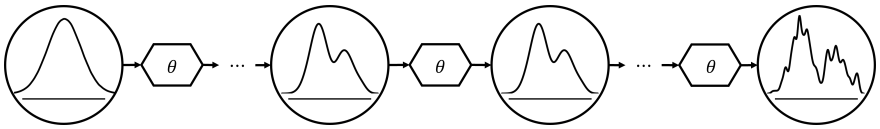


Figure 3: The reverse process trains a neural network θ to recursively remove the noise that has been previously added by the forward process.

The denoising network is trained by the standard variational bound:

$$
\begin{aligned}
L = \mathbb{E}\Bigg[& \underbrace{D_{KL}\Big(p\left(x_T|x_0\right) \| p\left(x_T\right)\Big)}_{\text{prior matching term}} \\
& + \underbrace{\sum_{t\geq 1} D_{KL}\Big(p\left(x_{t-1}|x_t, x_0\right) \| p_\theta\left(x_{t-1}|x_t\right)\Big)}_{\text{denoising matching term}} \\
& - \underbrace{\log p_\theta\left(x_0|x_1\right)}_{\text{reconstruction term}} \Bigg],
\end{aligned}
\tag{4}
$$

where $D_{KL}(\cdot\|\cdot)$ is the Kullback–Leibler (KL) divergence to compute the difference between two distributions. The prior matching term is minimized as the final distribution becomes Gaussian after a sufficiently large T. The reconstruction term can be approximated using a Monte Carlo estimate, and training primarily focuses on the denoising matching term. Overall, minimization of the objective L is to reduce the discrepancy between $p_\theta(x_0)$ and $p(x_0)$.

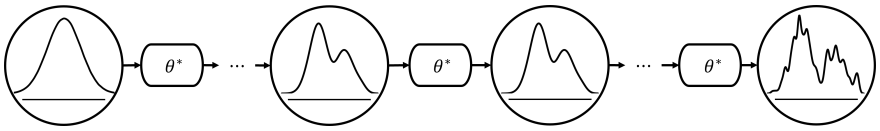


Figure 4: The sampling process uses the trained denoising network θ^* and usually follows the same transitions as the reverse process.

The sampling process leverages the optimized denoising network θ^* to generate novel data x_0^*, as illustrated in Fig. 4. It moves backwards on the chain to recursively apply the optimized network θ^*. 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 $\hat{x}_0 \sim p_{\theta^*}(x_0) \approx p(x_0)$. In particular, the sampling process is defined as a chain of sampling transitions:

$$p_{\theta^*}(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_{\theta^*}(x_{t-1}|x_t), \tag{5}$$

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

To reflect the development of diffusion models, we organize diffusion models by two popular formulations, i.e., discrete and continuous timesteps. To keep our survey simple to understand, especially for beginners, we present and discuss most of the fundamental designs under the discrete-time framework. These choices on the discrete-time framework are generally applicable to the continuous-time framework.

2.2.1. The Discrete Formulation

Initially motivated by unsupervised learning, diffusion models are formulated with discrete timesteps. Regarding the discrete formulation, the denoising diffusion probabilistic model (DDPM) [19] is a popular configuration of such formulated diffusion models. It is straightforward to define, efficient to train, and capable of achieving high quality and high diversity in the generated samples [20].

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 are Gaussian, the forward process 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).

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:

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

The intuition behind the derivation is two-fold. First, all distributions involved in Eq. 4 are Gaussians. Second, with Eq. 6 and Eq. 8, the KL divergence is simplified to be only dependent on the mean that is parameterized by the predicted noise ϵ_t. Finally, the sampling process obtains $x_T \sim p(x_T)$, and applies $p_{\theta^*}(x_{t-1}|x_t)$ to generate $\hat{x}_0$.

2.2.2. The Continuous Formulation

Focusing on the dynamics of diffusion models, continuous formulation is proposed to analyze the complex dynamics and also integrate the domain knowledge of score matching. The continuous formulation manipulates data distributions in continuous time. Noise is added in an infinitesimal interval between timesteps. Therefore, a differential equation (DE) is adopted in such formulated diffusion models to describe changes in continuous timesteps. Furthermore, [21] unifies all diffusion models with differential equations. Flow matching or stochastic interpolants [22, 23, 24, 25] is one of the popular approaches to improve the dynamics of diffusion models, which is often presented using continuous formulation.

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 is also determined by these coefficients. Specifically, the reverse SDE is:

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

where the output of the denoising network $s_\theta(x_t,t) = \nabla_x \log p(x_t)$ is the score. 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:

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

where $\lambda(t)$ is the weighting function. Finally, the sampling process obtains x_T, and applies the trained network θ^* to generate novel data.

The ODE-based formulation, derived from a deterministic process with the same marginal densities $\{p(x_t)\}_{t=0}^T$ as the SDE, is detailed in [21] and expressed as:

$$dx = \left[f(x,t) - \frac{1}{2} g^2(t)\, s_\theta(x_t,t)\right] dt, \tag{13}$$

Its optimization objective matches Eq. 12. Compared to SDE-based formulations, ODE-based approaches enable exact likelihood computation and provide deterministic latent representations, which are advantageous for editing tasks and efficient sampling.

2.3. Evaluation

The evaluation of diffusion models encompasses three dimensions: distribution, individual, and human aspects, the first two of which are often combined with domain knowledge. From a distribution perspective, metrics such as the Frechet Inception Distance [26], assess the distributional similarity to evaluate generation quality. The individual aspect usually involves paired evaluations such as the latent alignment [27] or self-contained evaluations such as physics constraints [28] of generated data. Domain-

specific knowledge is essential to design evaluation metrics for different fields such as medical analysis and meteorology. However, these quantitative measures may not align with human perceptual judgments [29]. Human-centric evaluations, such as user studies, are necessarily employed to provide subjective assessments of performance. Recently, human preferences [30, 31, 32] have been leveraged for better alignment. With these three evaluation dimensions, diffusion models are rigorously assessed and comparable with other models.

3. The Forward Process

The forward process is crucial for the success of diffusion models, as it reduces the complexity of the generation task by positive-incentive noise [33]. It perturbs data by scheduled noise, and results in stable noisy augmentations of data [34] that bridge data and pure noise distributions. These augmentations later facilitate diffusion models to learn the gradual multi-step generation, thereby reducing the task entropy. Moreover, [35] demonstrates that the integration of positive-incentive noise consistently enhances performance from a variational perspective.

3.1. The Noise Schedule

A suitable schedule balances exploration and exploitation [36]. Exploration, defined as a model's ability to generalize to unseen data, requires an adequate level of noise while excessive noise can lead to suboptimal convergence. Conversely, exploitation, where the model effectively fits the training data, is achieved with minimal noise while insufficient noise undermines generalization.

The noise schedule can be either learned by a network or empirically designed using mathematical formulations. Schedules are treated as a parameter to be learned jointly with other parameters [41]. Manually designed noise schedules are formulated with a wide variety of mathematical heuristics. A linear schedule [19] has been initially proposed. For faster perturbation, an exponential schedule [39] is proposed for better exploration. In contrast, schedules such as cosine [38], sigmoid [40], and rectified flow [25] and its variants [42, 43] are proposed for smoother perturbation speed. Table 2 shows several examples of designed noise schedules.

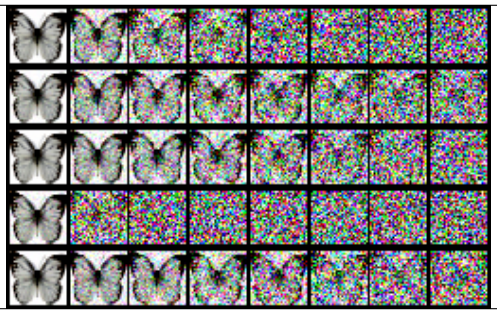
Noise Schedule	Visualization
Linear [19]	
Flow [37]	
Cosine [38]	
Exponential [39]	
Sigmoid [40]	

Table 2: Illustrations of typical manually-designed noise schedules.

Learning schedules enable models to adaptively optimize noise distributions but may lead to overfitting and reduced generalization [41]. In contrast, predefined schedules avoid adding parameters and offer greater interpretability. They also support different perturbation speeds for trading-off exploration and exploitation. However, they often require manual tuning for a new task or a new dataset [37].

3.2. The Noise Type

The selection of the noise type plays a crucial role in diffusion models, influencing distribution approximation and the overall expressiveness of the model. An appropriate noise type enhances the capacity to accurately fit the perturbed distributions at various timesteps [44]. Additionally, different noise types offer varying degrees of freedom [45], providing flexibility in modeling complex data distributions.

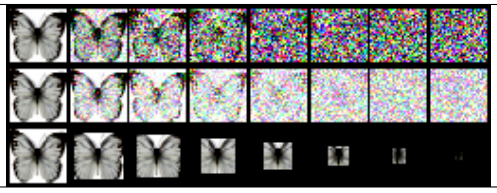
Noise Type	Visualization
Gaussian [39, 19]	
Gamma [44, 45]	
Soft [46, 47]	

Table 3: Comparison of several streams of noise types.

Different noise types have been developed based on empirical experiments. Isotropic Gaussian noise [19] is commonly used for its simplicity and compatibility. It allows for analytical solutions by taking advantage of the additivity of Gaussian distributions

and simplifying the calculation of KL divergence in Eq. 4. Several variants of isotropic Gaussian noise, such as mixture of Gaussian noise [45] and non-isotropic Gaussian noise [48], have also been applied to consider data structures. Correlated noise may be a potential alternative when correlation exists in a data sample, e.g., when frames of a video are considered [49] as correlations, while other video diffusion models usually maintain the default noise type. Additionally, noise from other distributions is also explored. Gamma distribution [44] is another feasible alternative with one more degree of freedom and fits to distributions better.

Soft corruptions can also be treated as a generalized noise for perturbation. Gaussian blur like the heat equation [50] is introduced for disentanglement of overall color and shape and smooth interpolation if it is used for image data. Soft corruption can also be manually defined operators like masking [47] to perturb data. 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 [46]. Table 3 visualizes examples of noise types.

Selecting appropriate types depends on the characteristics of data. Isotropic Gaussian noise is broadly applicable but may overlook the prior knowledge of data structure. Gaussian mixtures are better for data with distinct modes, whereas non-isotropic Gaussian noise considers self-correlation. Soft corruption is effective for known perturbation patterns, as it relies on predefined operators [47] to fulfill its flexibility.

3.3. The Terminal Distribution

Diffusion models are assumed to have zero signal-to-noise ratio (SNR) value for terminal distributions to correctly align diffusion training and inference [51]. Ideally, x_T is heavily perturbed without any original structures, i.e., zero SNR. However, empirically the terminal distribution may not strictly be zero SNR. This misalignment with assumption leads to suboptimal generation quality.

The forward process seeks suitable terminal distributions, either to maintain as many structures of x_0 as possible or to ensure zero SNR at the terminal distribution. Fig. 5 demonstrates the first direction. These approaches usually consider the statistics of the training dataset [52] such as the mean and variance to serve as proxies for data

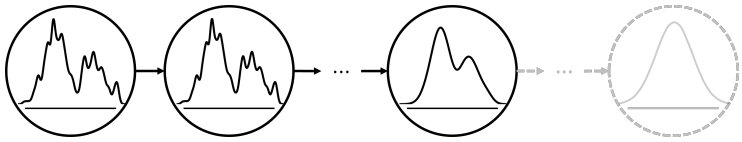


Figure 5: The transition chain no longer seeks an isotropic Gaussian distribution as the terminal distribution. The grey, dashed parts represent that the transition no longer approaches the isotropic Gaussian distribution.

structures. Learning the terminal distribution $p(x_T)$ with additional networks is also feasible [53]. This direction expects that retaining more meaningful structural information from the data distribution mitigates the difficulty of generation. The other direction adjusts the forward process to ensure that the terminal distribution conforms to our assumption. Offset noise is straightforward by altering its mean value but is not devoid of inherent challenges. [51] rescales the noise schedule but requires subsequent fine-tuning across the entire network. [54] relies on training an auxiliary text-conditional network to map pure Gaussian noise to the data-adulterated noise.

Choosing a suitable terminal distribution remains an open and important problem. The direction of maintaining more original structures may additionally accelerate the sampling process because fewer timesteps are involved, but may require an accurate representation of the terminal distribution [52]. On the contrary, ensuring zero SNR sticks to the assumption of diffusion models and the terminal distribution is accessible, which is more desirable when accurate representations are hard to obtain.

3.4. Representation Space

Latent representation now becomes a common choice for diffusion, as illustrated in Fig. 6. The high dimensionality of data often leads to considerable computational cost and redundancy. One of the representative models is Latent Diffusion Model [55] where images are compressed into lower dimensional vectors. Empirical evidence [56] shows that some transitions in diffusion models are responsible for learning latent representations, which are usually in low-dimensional space and semantically meaningful.

Sub-space methods treat different parts of input separately in corresponding sub-spaces. They usually involve more than one chain of transitions in the generic pipeline. In the conditional case, data and their labels are taken as two sub-spaces and then are

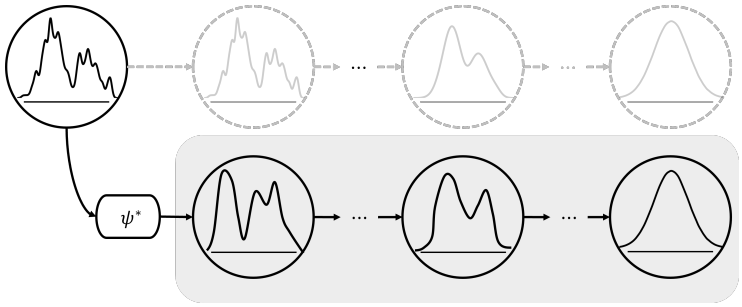


Figure 6: The transition chain in a latent space. ψ^* is a pre-trained encoder. Data are no longer manipulated in the original space (dashed, grey). They are now transformed within the latent space (rounded rectangle).

diffused simultaneously [21]. This design explicitly builds a joint modeling approach for conditional data. Data orthogonal decomposition is widely used where data are decomposed into several complementary parts in their corresponding sub-spaces [57]. This design brings flexibility for modeling data with heterogeneous properties. Fig. 7 shows an example of defining sub-spaces by data decomposition.

Augmented space introduces intermediate variables to extend the original space. Introducing intermediate variables is motivated by the overly simplistic diffusions that cannot represent the full dynamics in the forward process [58], and thus leads to unnecessarily complex denoising processes and limits generative modeling performance. An auxiliary velocity variable is introduced and the forward process is only defined in the augmented space [59]. Introducing stochasticity into the augmented space benefits the smoothness of the evolution of variables.

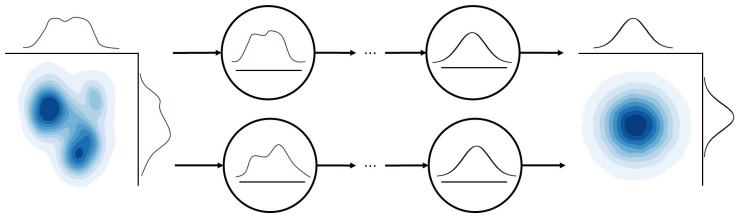


Figure 7: The forward process to separately transform the original data in orthogonal subspace.

The three spaces can be mutually beneficial instead of exclusive. Latent representation is more suitable for compressing high-dimensional data but may risk information loss. Sub-space methods offer flexibility for modeling heterogeneous data but

add pipeline complexity. Augmented space approaches capture indirect dynamics by introducing intermediate variables.

4. The Reverse Process

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

4.1. Network Architectures

4.1.1. Architecture Flexibility

Theoretically, it is feasible to incorporate as a denoising network a wide variety of architectures that keep the dimensionality unchanged [60]. Both U-Net and Transformer have become the mainstream for their high capacity for modeling complex relationships in a wide range of applications and GNN quickly gets more attention. While other architectures may also theoretically be compatible without changing dimensions like GAN [61], they are often adopted for task-specific purposes, e.g., adopting GAN for fast generation [62], and may not be generally applicable to other purposes.

4.1.2. U-Net

Since [19] first introduced diffusion models with U-Net architecture, this architecture has dominated this area and largely remained intact. From a theoretical standpoint, it is a U-shaped encoder-decoder architecture for general purposes. 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 [63] and encourages the network to learn features effectively. Nonetheless, a few modifications have been introduced in some representative U-Net-based diffusion models such as ADM [20] and EDM [2, 64] when compared with the traditional U-Net. [20] ablates several architecture choices such as adaptive group normalization. Based on [2], [64] proposes magnitude-preserving layers to replace the data-dependent group

normalization layers to preserve activation, weight, and update magnitudes. Additionally, the U-shape architecture has been adopted with cross-attention [65] for higher capacity, and cascading for hierarchy modeling [66] has been merged into the U-Net architecture. Latent diffusion models [55] enhance the traditional U-Net architecture by transformer layers to capture long dependency.

4.1.3. Transformer

Transformers are increasingly adapted as an alternative architecture for the denoising network for their superior properties like global dependency, scalability, and multimodality [67] because of self-attention functions [68]. In principle, a transformer can directly substitute U-Nets because it can also maintain the data dimensions [69]. However, transformers also exhibit unique advantages. Scalability [67] of transformers enables better generation quality with less network complexity, which is critical for emergence ability. Besides, as a sequence model, transformers support an arbitrary length of generation. For example, MDM [28] generates an arbitrary length of human motions. Transformers are also friendly for multi-modality in diffusion models. Stable Diffusion 3 [43] enables the alignment of conditions with MMDIT [70] that employs shared full attention weights for visual and textual modalities.

Transformers natively support diverse conditioning via their core mechanisms such as layer normalization and multi-head attention [67]. Adaptive layer norm modulates features via scale-and-shift. Cross-attention allows each token to selectively focus on condition embeddings for precise control. In-context conditional tokens embed guidance into the representation space. Convolutional U-Nets, in contrast, usually need to graft carefully designed layers or attention blocks at each stage, complicating design and scaling. As a unified framework, transformers streamline implementation and scale predictably, boosting both generative capacity and conditioning fidelity.

Since DiT [67] demonstrates the superiority of diffusion transformers, different ways to integrate designs from other architectures into transformers have been explored. U-ViT [71] which utilizes U-Net architecture for diffusion transformers. MaskDiT [72] and MDT [73] use the MAE-like architecture. FiT [74, 75] proposes flexible transformer architecture. SANA [76] with linear transformer architecture.

4.1.4. GNN

Graph Neural Networks (GNNs) have also been an emerging choice for the denoising network especially when graph structures are involved. Their superior performance is attributed to inductive bias from network architecture [6]. Otherwise, the learning may deviate from inherent graph properties [77]. Equivariant property has been widely considered for denoising architecture. [78] keeps the property of invariant permutation in the reverse process. [79] further extends it with equivariant energy guidance to learn the geometric symmetry. Another considered aspect is the formation of graph structure. [80] defines the denoising on the adjacency matrix as well as node features. [81] performs low-rank Gaussian noise insertion with spectral decomposition. Latent space has also been considered. [82] encodes the high-dimensional discrete space to low-dimensional topology-injected latent space.

4.2. Network Parameterizations

The output of the denoising network is applied to parameterize the reverse mean $\mu_\theta(x_t, t)$ in the reverse transition. 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:

$$\mu(x_t, t) := \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t}, \tag{14}$$

where x_0 is the original data but is unavailable during the reverse process. Therefore, x_0 needs to be estimated from the observed perturbed data x_t and timestep t by the 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. (14). 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 [19]. 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 the aforementioned ways are also proposed for special tasks. Table 4 shows a comparison of different parameterization ways. Different out-

puts are equivalent to each other [83, 2] with the following relationships:

$$x_\theta(x_t, t) = \frac{x_t + (1 - \bar{\alpha}_t) s_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}} = \frac{x_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}, \tag{15}$$

$$s_\theta(x_t, t) = -\frac{1}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_\theta(x_t, t). \tag{16}$$

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 modeling. Combining the aforementioned ones provides the flexibility to retain their benefits.

4.2.1. Starting 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:

$$\mu_\theta(x_t, t) := \frac{\sqrt{\bar{\alpha}_{t-1}}(1 - \bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)x_\theta(x_t, t)}{1 - \bar{\alpha}_t}, \tag{17}$$

where α_t indicates the noise level as defined in Section 2.2.1.

Output	Parameterization	Visualization
Data x_θ	$\mu_\theta(x_t, t) := \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_\theta}{1-\bar{\alpha}_t}$	
Score s_θ	$dx := \left[f(x,t) - g^2(t)s_\theta\right]dt + g(t)dw$	
Noise ϵ_θ	$\mu_\theta(x_t, t) := \frac{1}{\sqrt{\alpha_t}}x_t - \frac{1-\alpha_t}{\sqrt{\alpha_t(1-\bar{\alpha}_t)}}\epsilon_\theta$	
Hybrid h_θ	$\mu_\theta(x_t, t) := \mathcal{H}(x_\theta, s_\theta, \epsilon_\theta)$	N/A

Table 4: Visualization of parameterization ways.

Parameterizing with $\hat{x}_0$ is advantageous at the beginning of the sampling process,

while it leads to inaccuracy when approaching the end of the sampling process. 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 process [84]. This is because $\hat{x}_0$ helps the denoising network with an overall understanding of the global structure [85]. On the contrary, when approaching the end of the sampling process where substantial structures have already been formed and only small noise artifacts need to be removed, finer details are difficult to be recovered [86]. 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. The gradient indicates the most possible changes between two timesteps. Therefore, as shown in Fig. 8, denoising samples by the score forms a trajectory in data space. In particular, given the observed x_t and timestep t, the predicted score is defined as $s_\theta(x_t, t) := \nabla_x \log p(x_t)$ and the corresponding parameterization is the reverse SDE:

$$dx := \left[f(x,t) - g^2(t) s_\theta(x_t, t) \right] dt + g(t) dw, \tag{18}$$

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

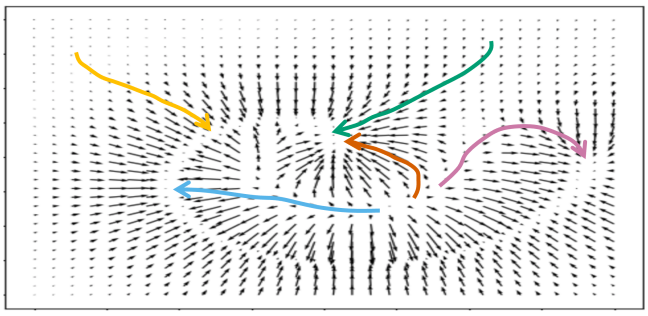


Figure 8: Visualization of the trajectory by predicting score. A score is a direction for the next timesteps. Samples are denoised in the direction at each position. Colors represent the trajectories of different samples.

Score prediction is closely related to flow matching [22] in terms of modeling a vector field. Score-parameterized diffusion models provide unbiased gradients as the vector field under the assumption of Gaussian distribution while flow matching directly

learns the vector field. Score-parameterized diffusion models transport data with Gaussian conditional paths while flow matching also supports conditional optimal transport paths. In particular, the predicted distribution is usually defined as:

$$p_\theta(x) = \frac{\exp^{-f_\theta(x)}}{Z_\theta}, \tag{19}$$

where Z_θ is a normalizing constant to estimate. Predicting score avoids this problem:

$$\nabla_x \log p_t(x) = -\nabla_x f_\theta(x) - \nabla_x \log Z_\theta = -\nabla_x f_\theta(x), \tag{20}$$

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 [19, 77], as shown in Fig. 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:

$$\mu_\theta(x_t, t) := \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1 - \alpha_t}{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}} \epsilon_\theta(x_t, t), \tag{21}$$

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

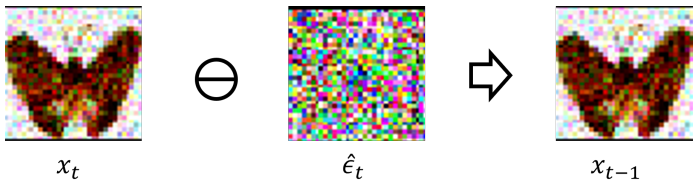


Figure 9: Visualization of the noise-based parameterization. $\ominus$ means $\hat{\epsilon}_t$ has a subtractive relationship with x_t, and $\Rightarrow$ means this results in x_{t-1}.

The consistent magnitude and residual effect of $\epsilon_\theta(x_t, t)$ are advantageous. The fixed statistics, e.g. $\epsilon_\theta(x_t, t) \sim \mathcal{N}(0, I)$, lead to a consistent magnitude. This encourages the learning of the denoising network [83]. Besides, the residual effect to preserve the input x_t in x_{t-1} is available by predicting zero noise. This becomes increasingly beneficial

towards the end of the reverse process where only minor modifications are needed [86].

A large deviation between the ground truth noise ϵ_t and the predicted noise $\epsilon_\theta(x_t, t)$ may occur at the beginning of the sampling process, and is hard to be corrected in the following timesteps. Sampling starts with large noise, with almost no clue for the denoising network to predict noise accurately [19]. This potentially leads to a deviation [86]. The deviation is scaled up by the noise schedule in Eq. (21). The scheduled level of noise is usually large at the beginning of the sampling process. 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 on 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 [85].

4.2.4. Hybrid

Combining two or more predictions is also possible for task-specific benefits. Abstractly, the combination is denoted as $h_\theta(x_t, t) := \mathcal{H}(x_\theta(x_t, t), s_\theta(x_t, t), \epsilon_\theta(x_t, t))$ where $\mathcal{H}$ stands for a combination operator. Therefore, the parameterization is:

$$\mu_\theta(x_t, t) = h_\theta(x_t, t). \tag{22}$$

This has a wide variety of feasible implementations because the output to be combined and the combination operators can be very diverse [16]. Velocity prediction in DDPM is one example that linearly combines $x_\theta(x_t, t)$ and $\epsilon_\theta(x_t, t)$ [87], which is designed as:

$$\mu_\theta(x_t, t) := \alpha_t \epsilon_\theta(x_t, t) - \sigma_t x_\theta(x_t, t), \tag{23}$$

where α_t and σ_t are the scaling factor and noise schedule respectively. It has better stability [88], avoids noise existing in $x_\theta(x_t, t)$ [89] and achieves higher likelihood [90]. Dynamically alternating between $x_\theta(x_t, t)$ and $\epsilon_\theta(x_t, t)$ accelerates the generation [86].

4.2.5. The Reverse Variance

Modeling the reverse variance improves the training efficiency of diffusion models. An appropriate variance minimizes the discrepancy between the predicted reverse tran-

sition $p_\theta(x_{t-1}|x_t)$ and the forward transition $p(x_t|x_{t-1})$, fitting the forward process better [91]. This facilitates fewer 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 [92]. Scaling the schedule by a factor is also researched but does not lead to a large difference [19]. Both choices are considered as upper and lower bounds on reverse process entropy [1], and the interpolation between them is learned for flexibility [38]. Others find the optimal variance can be solved analytically. Its formulation is explicitly derived from the predicted score [91], and improves the efficiency of generation [93].

4.3. Weighted Optimization

Weighted optimization in the reverse process is inspired by the understanding of the learning procedure of diffusion models. A common choice [19] applies uniform weights and may overlook the characteristics of the reverse process. The semantic information of data expressed in the reverse process gradually changes, which requires appropriately set priorities to learn [29]. An alternative choice is a function of intermediate characteristics, e.g., signal-to-noise values. While it takes data into account, the hyperparameters of the designed function may need to be carefully set.

Learning priorities are balanced by weights in the learning objective to enhance the learning quality. The change of learning priorities has been observed in the reverse process. It pays more attention to global structures at the beginning of the reverse process and then changes to local details when approaching its end [94]. A balance is achievable through adjusting weights and beneficial for training.

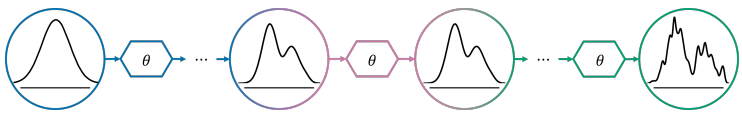


Figure 10: The learning priority changes in the reverse process, which is denoted by different colours.

Directly using the schedule as the weight emphasizes the global structure better by a larger learning weight at the beginning of the reverse process [95]. Despite its

simplicity, 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 [83]. It takes the data into account, and better balances the learning of local details and global structures [94].

5. Sampling Process

Conditional and fast generation are two focused factors of the sampling process in diffusion models. Without modeling conditions, diffusion models usually do not generate data of high quality when data are considered to follow a conditional distribution [96]. Effective mechanisms of guidance are designed to modify transitions in the sampling process to be compatible with conditions. Moreover, the sampling process is several times slower when compared with other generative models [97]. The long generation time is mainly attributed to the large number of timesteps. Thus, designs for acceleration are explored to reduce timesteps without heavily impairing quality.

5.1. Guidance Mechanisms

Vanilla guidance merges conditions via fusion approaches such as the attention layer. However, the weight of conditions is not easy to adjust. Classifier guidance leverages an additional classifier and adjustable weights, but comes with issues of computational cost and stability. Classifier-free guidance additionally trains unconditional diffusion models to achieve better stability. Learned modifications via adapters provides guidance but need to fine-tune extra adapters.

5.1.1. Vanilla Guidance

Vanilla guidance usually merges the given conditions c with timesteps t as the guidance. The intuition of merging is that a timestep t itself is inherently taken as a condition by a denoising network and thus more conditional information can also be conveyed. The approaches of merging can be operations such as addition [19, 98], and attention layer [99, 100]. Fig. 11 shows the condition is added to a timestep in this mechanism.

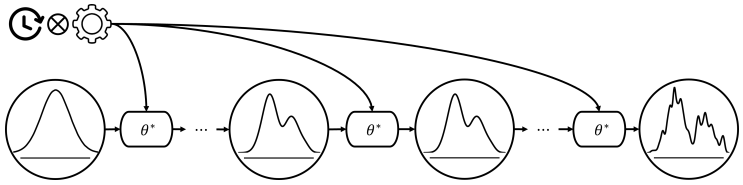


Figure 11: Vanilla guidance adds the conditions c to each timestep t as a new condition.

While vanilla guidance benefits from its simplicity, its effectiveness is undermined by the lack of adjustable conditional strength [101]. Empirical evidence [85] shows that a conditional diffusion model trained with vanilla guidance may not conform to the conditions or underperform in conditional generation.

5.1.2. Classifier Guidance

For effective and adjustable strength of conditions, classifier guidance [20] adopts an extra classifier. The gradient of the classifier is scaled by the weight and then is used to modify the unconditional denoising direction, as shown in Fig. 12. In other words, the weight controls how much to rely on the classifier. To obtain the gradient as accurately as possible, the classifier is usually pre-trained on data with conditions. In particular, classifier guidance is formulated as:

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

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.

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 [101]. Moreover, training the classifier on data with noise tends to be unstable. The data structure is almost destroyed because more and larger noise is added according to the noise schedule. Therefore, the quality of the classifier gradient may not be consistent [102]. Sometimes its direction is arbitrary or even opposite

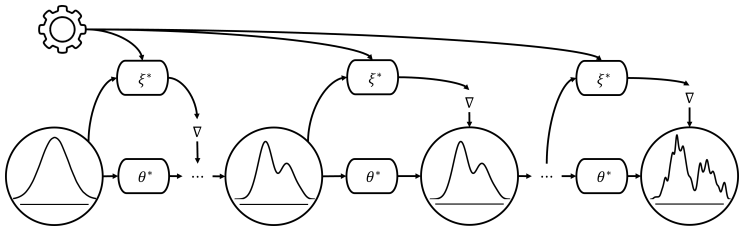


Figure 12: Classifier guidance leverages an extra classifier network ξ^* to compute a gradient ∇ as the modification on the denoising network θ^*. The timestep condition t is omitted here for visualization.

[103] and leads to less effective or wrong guidance. Some imitate the form of classifier guidance but solve the gradient analytically [104] for the inverse problem to bypass the disadvantages of classifiers.

5.1.3. Classifier-Free Guidance

To avoid the extra classifier, classifier-free guidance [101] replaces the classifier with a mixture of unconditional and condition models. It further enhances the sampling process to follow the direction of guidance by discouraging it away from unconditional direction [105]. As shown in Fig. 13, instead of just training a conditional model, an unconditional one is also trained simultaneously by dropping out conditions c with a probability p. In particular, classifier-free guidance is formulated as:

$$\nabla_x \log p(x|c) = w\nabla_x \log p(x|c) + (1 - w)\nabla_x \log p(x), \tag{25}$$

where w is the weight of conditions.

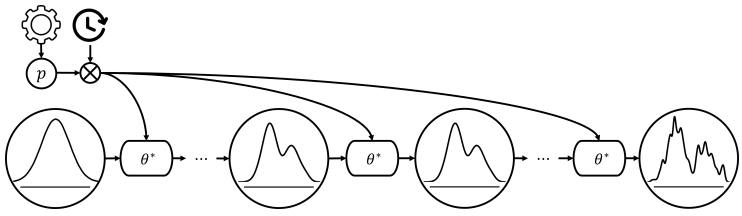


Figure 13: Classifier-free guidance is based on a mixture of vanilla guidance and unconditional model θ^*. A probability p controls whether to drop out the conditions during training.

The weight is slightly different from its counterpart in classifier guidance. When $w = 0$, the classifier-free guidance becomes unconditional models without vanilla guid-

ance. 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 [106]. 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 [107].

5.1.4. Learned Modifications

Learning modifications provides greater flexibility for controllable generation by preserving a generative prior. ControlNet [108] is a pioneering method to fine-tune an extra copied model. This idea for conditional generation has been popular, evidenced by various adapters. SUPIR [109] trains a trimmed ControlNet for image restoration. Uni-ControlNet [110] further proposes all-in-one control. StableSR [111] learns a time-aware encoder to assist super-resolution generation that is conditioned on low-resolution images. T2I-Adapter [112] equips the pre-trained model with several low-complexity adapters. LoRAdapter [113] takes inspiration from LowRank-Adaptations (LoRA) [114] to further reduce the parameters to be learned.

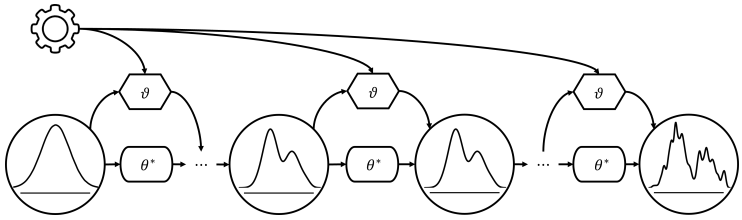


Figure 14: Applying an extra network ϑ to directly learn the required modification for guidance. Timestep condition is omitted here.

5.2. Acceleration Designs

Reducing the number of timesteps for generation is the main goal of acceleration. Generally, the denoising network needs to wait for the results from the timestep $t + 1$ to accomplish the transition at the current timestep t. The inference speed is significantly slowed down especially when a large number of timesteps are required in the sampling process [62]. Truncation directly cuts the sampling process at a certain timestep but

may suffer from distribution deviation. Knowledge distillation is adopted to learn an auxiliary module that enables fewer timesteps in its sampling process but may require careful fine-tuning. Timestep selection skips some timesteps and selects a subset of timesteps for fast generation but may suffer from inherent approximation errors.

5.2.1. Truncation

Truncation involves a partial sampling process 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, as shown in Fig. 15. In other words, the process truncates the whole chain at t', and thereby fewer timesteps remain in the partial chain. An extra network needs to be additionally trained to model $p(t')$ that may not be tractable [19]. Overall, truncation is theoretically effective in acceleration [115], which is proved by the stochastic contraction theory.

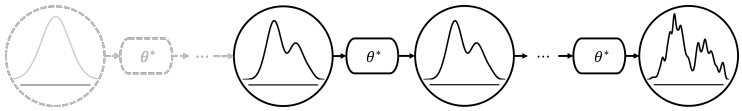


Figure 15: The sampling process is truncated and starts from a selected timestep. The grey, dashed parts represent discarding for generations.

Truncation effects can be two-sided. On the one hand, truncation comes with acceleration for not only inference but also training. Truncation also strikes a balance between acceleration and quality as the selection of such a point depends on the data complexity [53] and the degree of corruption [115]. Besides, truncation takes advantage of the properties of the involved extra network, which is often another generative model [116]. On the other hand, truncation may lead to an increased training expense [116] because the extra network needs to learn $p(t')$ as accurately as possible.

5.2.2. Knowledge Distillation

The technique can also be applied to learn a new sampling process with fewer timesteps. Knowledge distillation is a network compression technique. In terms of diffusion models, it involves the original sampling process as the teacher model and a new one with fewer timesteps as the student model [117]. Fig. 16 shows an example

method of progressive distillation in the sampling process.

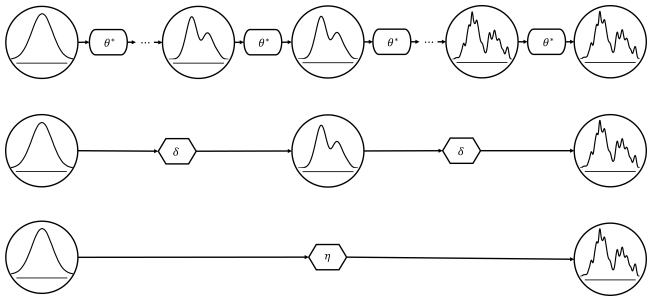


Figure 16: Knowledge distillation learns student denoising networks δ and η with fewer timesteps, based on the teacher denoising network θ^*.

Knowledge distillation is usually applied to merge several timesteps into fewer timesteps for the new sampling process such as progressive distillation, guidance distillation, and consistency models. [118] directly distills all timesteps into a single one with expensive computation since a large dataset of samples from the teacher model is needed [119]. One mitigation is progressively reducing timesteps [88]. Guidance distillation [120] is a plug-and-play acceleration approach with classifier guidance but the memory usage is high. Another family is consistency models where the sampling trajectory of diffusion models are straightened for consistency [121] to accelerate the sampling process. Straightening is achieved via two approaches. Consistency training approach learns a consistency model from scratch, as if it distilled all timesteps into a single timestep to achieve consistency. In contrast, consistency distillation accelerates existing models to become consistency models for single or fewer timesteps.

5.2.3. Timestep Selection

Timestep selection seeks to develop strategies to select only a subset of timesteps without undermining quality. Some timesteps in a sampling process influence quality less and thus they can be skipped safely [122]. Fig. 17 shows a shorter sampling process with selected timesteps. Some may not directly reduce the number of timesteps but select a subset of timesteps for parallel computation [123].

Some acceleration methods essentially solve stochastic or ordinary differential equations. DDIM [92] has been widely applied for its simplicity in using non-Markovian

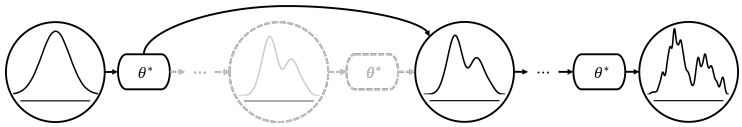


Figure 17: Selection strategies for the sampling process skip the selected time steps for generation.

reverse processes, and PNDM [124] further builds close connections with pseudo-numerical methods for acceleration. Another commonly used method is DPM-Solver [125] and its improved version DPM-Solver++ [126]. They are high-order solvers for ODEs with a convergence order guarantee. Empirically, 20 timesteps are used to generate high-quality results. Euler samplers are also widely used [2] to generate with 20 to 30 timesteps. The selection of timesteps can also be learned by extra networks [127] or rely on dynamic programming algorithms [128].

6. Future Trends

We provide a summary in Table 5 to connect the components in this survey with popular open-source diffusion models.

Models	Forward	Reverse	Sampling
DiT [67]	Schedule: Linear Type: Gaussian Space: Latent	Architecture: Transformer Parametrization: Noise Weight: Pre-defined	Guidance: Vanilla/CFG Acceleration: Solvers
SD1 [55]	Schedule: Scaled Linear Type: Gaussian Space: Latent	Architecture: U-Net Parametrization: Noise Weight: Pre-Defined	Guidance: Vanilla/CFG Acceleration: Solvers
SD2 [129]	Schedule: Scaled Linear Type: Gaussian Space: Latent	Architecture: U-Net Parametrization: Hybrid (v) Weight: Pre-Defined	Guidance: Vanilla/CFG Acceleration: Solvers
SD3 [43]	Schedule: Rectified Flow Type: Gaussian Space: Latent	Architecture: Transformer Parametrization: Score (flow) Weight: Pre-Defined	Guidance: Vanilla/CFG Acceleration: Solvers
SVD [130]	Schedule: Linear Type: Gaussian Space: Latent	Architecture: U-Net Parametrization: Hybrid (v) Weight: Pre-Defined	Guidance: Vanilla/CFG Acceleration: Solvers
OpenSora1.2 [131]	Schedule: Rectified Flow Type: Gaussian Space: Latent	Architecture: Transformer Parametrization: Score (flow) Weight: Pre-Defined	Guidance: CFG Acceleration: Solvers
FLUX [132]	Schedule: Rectified Flow Type: Gaussian Spcae: Latent	Architecture: Transformer Parametrization: Score (flow) Weight: Pre-Defined	Guidance: Vanilla/LoRA/CFG Acceleration: Distillation

Table 5: Popular diffusion models with their designs. Guidance can vary across tasks because of the versatility and adaptability of diffusion models.

6.1. Generalization Capability

Understanding the reasons why diffusion models generalize well remains an open problem. Recent advances have attributed the generalization to a few designs such as the Gaussian structure [133] and geometry-adaptive harmonic representations [134], while the comprehensive underlying mechanism remains unclear. Further investigation can explore other designs such as parameterizations. This may need to further consider the interplay effects of other designs with extensive analysis and experiments since designs are often deeply correlated.

6.2. Denoising-Oriented Architecture

The network architectures of diffusion models present significant research opportunities. Section 4.1 showcases success through integration with other areas. Nonetheless, a lack of specialized denoising architectures may result in suboptimal denoising performance. Future trends may borrow network architectures from related fields like image restoration [135], which also aim to recover unperturbed data from noisy inputs. Adaptation to the wide variety of degradations may require additional modifications to the forward and reverse processes. Additionally, timestep-adaptive architectures may also be considered to better leverage the inherent denoising priority [60] in diffusion processes. However, adopting timestep-adaptive architectures increases the computational and architectural complexity, requiring additional efforts to balance the tradeoffs.

6.3. Responsible Applications

Diffusion models have become popular in sciences, such as physics [136] and medicine [137], etc. Nonetheless, concerns remain on whether they are reliable to capture the underlying causal relations of a domain as diffusion models tend to rely on statistical associations on the training dataset [138]. One possible direction may integrate causality-aware guidance mechanisms to condition diffusion models on directed causal graphs. While promising, handling unobserved confounding relationships usually requires extensive assumption tests to achieve higher reliability [139].

Diffusion models such as Stable Diffusion [43] have been significantly employed in the creative industry. Nonetheless, concerns about originality in creative works have

been raised [140] and generated content may infringe copyrights. One potential direction adopts concept forgetting [141] or adjusts the sample process with likelihood-aware guidance [142] to reduce the similarity. The quantification of originality remains an open problem for creative industries [143], and establishing industry standards to generate content ethically requires broad industry cooperation.

6.4. Societal Impacts

Diffusion models showcase biases, and critically, the definition of biases evolves over time and varies across cultures. Current solutions do not consider this evolution and variation. Combining fairness-aware algorithms [144] with incremental learning [145] may facilitate "incremental fairness". The main difficulty lies in the interdependent and multifaceted nature of biases where additional data may give rise to new biases in existing data and addressing one type of bias without considering others can result in incomplete or skewed mitigation efforts.

Diffusion models democratize personalization and boost productivity by enabling non-professional users to automatically generate highly customizable content. This accessibility may reshape the job market and further impact educational systems. Consequently, governments and individuals may require additional expenses for employee retraining and educational reforms [9]. Addressing this requires interdisciplinary collaboration among computer science, politics, and economics to promote democratization and productivity while reducing potential social risks [146].

7. Conclusion

Diffusion models involve three main components: a forward process and a reverse process for optimization, and a sampling process for generation. The forward process focuses on perturbing data with different noise schedules, noise types, terminal distributions and representations. The reverse process focuses on training a denoising network to remove noise with different architectures, parameterizations, and weights. The sampling process works for generation and mainly focuses on guidance and acceleration. These designs have all contributed to the current powerful diffusion models. Several future trends have been introduced to boost this field.

References

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

[2] T. Karras, M. Aittala, T. Aila, S. Laine, Elucidating the design space of diffusion-based generative models, Advances in Neural Information Processing Systems 35 (2022) 26565–26577.

[3] Y. Zhu, Y. Zhao, Diffusion models in nlp: A survey, arXiv preprint arXiv:2303.07576 (2023).

[4] J. Wang, H. Zhang, Y. Yuan, Adv-cpg: A customized portrait generation framework with facial adversarial attacks, arXiv preprint arXiv:2503.08269 (2025).

[5] A. Kazerouni, E. K. Aghdam, M. Heidari, R. Azad, M. Fayyaz, I. Hacihaliloglu, D. Merhof, Diffusion models in medical imaging: A comprehensive survey, Medical Image Analysis (2023) 102846.

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

[7] L. Lin, Z. Li, R. Li, X. Li, J. Gao, Diffusion models for time-series applications: a survey, Frontiers of Information Technology & Electronic Engineering (2023) 1–23.

[8] D. Jiangzhou, W. Songli, Y. Jianmei, J. Lianghao, W. Yong, Dgrm: Diffusion-gan recommendation model to alleviate the mode collapse problem in sparse environments, Pattern Recognition (2024) 110692.

[9] X. Zhang, X.-Y. Wei, W. Zhang, J. Wu, Z. Zhang, Z. Lei, Q. Li, A survey on personalized content synthesis with diffusion models, arXiv preprint arXiv:2405.05538 (2024).

[10] W. Wang, Y. Sun, Z. Yang, Z. Hu, Z. Tan, Y. Yang, Replication in visual diffusion models: A survey and outlook, arXiv preprint arXiv:2408.00001 (2024).

[11] G. Kim, W. Jang, G. Lee, S. Hong, J. Seo, S. Kim, Depth-aware guidance with self-estimated depth representations of diffusion models, Pattern Recognition 153 (2024) 110474.

[12] T. Wang, K. Zhang, Y. Zhang, W. Luo, B. Stenger, T. Lu, T.-K. Kim, W. Liu, Lldiffusion: Learning degradation representations in diffusion models for low-light image enhancement, Pattern Recognition (2025) 111628.

[13] Y. Li, K. Zhou, W. X. Zhao, J.-R. Wen, Diffusion models for non-autoregressive text generation: a survey, in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, 2023, pp. 6692–6701.

[14] Z. Xing, Q. Feng, H. Chen, Q. Dai, H. Hu, H. Xu, Z. Wu, Y.-G. Jiang, A survey on video diffusion models, arXiv preprint arXiv:2310.10647 (2023).

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

[16] H. Cao, C. Tan, Z. Gao, Y. Xu, G. Chen, P.-A. Heng, S. Z. Li, A survey on generative diffusion models, IEEE Transactions on Knowledge and Data Engineering (2024).

[17] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, M.-H. Yang, Diffusion models: A comprehensive survey of methods and applications, ACM Computing Surveys 56 (4) (2023) 1–39.

[18] M. Chen, S. Mei, J. Fan, M. Wang, An overview of diffusion models: Applications, guided generation, statistical rates and optimization, arXiv preprint arXiv:2404.07771 (2024).

[19] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, Advances in Neural Information Processing Systems 33 (2020) 6840–6851.

[20] P. Dhariwal, A. Nichol, Diffusion models beat gans on image synthesis, Advances in Neural Information Processing Systems 34 (2021) 8780–8794.

[21] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, B. Poole, Score-based generative modeling through stochastic differential equations, in: International Conference on Learning Representations, 2021.

[22] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, M. Le, Flow matching for generative modeling, in: The Eleventh International Conference on Learning Representations, 2023.

[23] M. S. Albergo, E. Vanden-Eijnden, Building normalizing flows with stochastic interpolants, in: The Eleventh International Conference on Learning Representations, 2023.

[24] M. S. Albergo, N. M. Boffi, E. Vanden-Eijnden, Stochastic interpolants: A unifying framework for flows and diffusions, arXiv preprint arXiv:2303.08797 (2023).

[25] X. Liu, C. Gong, qiang liu, Flow straight and fast: Learning to generate and transfer data with rectified flow, in: The Eleventh International Conference on Learning Representations, 2023.

[26] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, Gans trained by a two time-scale update rule converge to a local nash equilibrium, Advances in neural information processing systems 30 (2017).

[27] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al., Photorealistic text-to-image diffusion models with deep language understanding, Advances in neural information processing systems 35 (2022) 36479–36494.

[28] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-or, A. H. Bermano, Human motion diffusion model, in: The Eleventh International Conference on Learning Representations, 2023.

[29] G. Stein, J. Cresswell, R. Hosseinzadeh, Y. Sui, B. Ross, V. Villecroze, Z. Liu, A. L. Caterini, E. Taylor, G. Loaiza-Ganem, Exposing flaws of generative model evaluation metrics and their unfair treatment of diffusion models, Advances in Neural Information Processing Systems 36 (2024).

[30] X. Wu, K. Sun, F. Zhu, R. Zhao, H. Li, Human preference score: Better aligning text-to-image models with human preference, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 2096–2105.

[31] X. Wu, Y. Hao, K. Sun, Y. Chen, F. Zhu, R. Zhao, H. Li, Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis, arXiv preprint arXiv:2306.09341 (2023).

[32] J. Xu, X. Liu, Y. Wu, Y. Tong, Q. Li, M. Ding, J. Tang, Y. Dong, Imagereward: Learning and evaluating human preferences for text-to-image generation, Advances in Neural Information Processing Systems 36 (2023) 15903–15935.

[33] X. Li, Positive-incentive noise, IEEE Transactions on Neural Networks and Learning Systems (2022).

[34] H. Zhang, Y. Xu, S. Huang, X. Li, Data augmentation of contrastive learning is estimating positive-incentive noise, arXiv preprint arXiv:2408.09929 (2024).

[35] H. Zhang, S. Huang, X. Li, Variational positive-incentive noise: How noise benefits models, arXiv preprint arXiv:2306.07651 (2023).

[36] T. Hang, S. Gu, Improved noise schedule for diffusion training, arXiv preprint arXiv:2407.03297 (2024).

[37] T. Chen, On the importance of noise scheduling for diffusion models, arXiv preprint arXiv:2301.10972 (2023).

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

[39] Y. Song, S. Ermon, Generative modeling by estimating gradients of the data distribution, Advances in Neural Information Processing Systems 32 (2019).

[40] A. Jabri, D. J. Fleet, T. Chen, Scalable adaptive computation for iterative generation, in: International Conference on Machine Learning, PMLR, 2023, pp. 14569–14589.

[41] S. S. Sahoo, A. Gokaslan, C. D. Sa, V. Kuleshov, Diffusion models with learned adaptive noise, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.

[42] N. Ma, M. Goldstein, M. S. Albergo, N. M. Boffi, E. Vanden-Eijnden, S. Xie, Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers, in: European Conference on Computer Vision, Springer, 2024, pp. 23–40.

[43] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al., Scaling rectified flow transformers for high-resolution image synthesis, in: Forty-first International Conference on Machine Learning, 2024.

[44] E. Nachmani, R. S. Roman, L. Wolf, Denoising diffusion gamma models, arXiv preprint arXiv:2110.05948 (2021).

[45] E. Nachmani, R. S. Roman, L. Wolf, Non gaussian denoising diffusion models, arXiv preprint arXiv:2106.07582 (2021).

[46] A. Bansal, E. Borgnia, H.-M. Chu, J. Li, H. Kazemi, F. Huang, M. Goldblum, J. Geiping, T. Goldstein, Cold diffusion: Inverting arbitrary image transforms without noise, Advances in Neural Information Processing Systems 36 (2024).

[47] G. Daras, M. Delbracio, H. Talebi, A. Dimakis, P. Milanfar, Soft diffusion: Score matching with general corruptions, Transactions on Machine Learning Research (2023).

[48] V. Voleti, C. Pal, A. M. Oberman, Score-based denoising diffusion with non-isotropic gaussian noise models, in: NeurIPS 2022 Workshop on Score-Based Methods, 2022.

[49] S. Ge, S. Nah, G. Liu, T. Poon, A. Tao, B. Catanzaro, D. Jacobs, J.-B. Huang, M.-Y. Liu, Y. Balaji, Preserve your own correlation: A noise prior for video diffusion models, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 22930–22941.

[50] S. Rissanen, M. Heinonen, A. Solin, Generative modelling with inverse heat dissipation, in: International Conference on Learning Representations, 2023.

[51] S. Lin, B. Liu, J. Li, X. Yang, Common diffusion noise schedules and sample steps are flawed, in: Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2024, pp. 5404–5411.

[52] S. gil Lee, H. Kim, C. Shin, X. Tan, C. Liu, Q. Meng, T. Qin, W. Chen, S. Yoon, T.-Y. Liu, Priorgrad: Improving conditional denoising diffusion models with data-dependent adaptive prior, in: International Conference on Learning Representations, 2022.

[53] H. Zheng, P. He, W. Chen, M. Zhou, Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders, in: The Eleventh International Conference on Learning Representations, 2023.

[54] M. Hu, J. Zheng, C. Zheng, C. Wang, D. Tao, T.-J. Cham, One more step: A versatile plug-and-play module for rectifying diffusion schedule flaws and enhancing low-frequency controls, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 7331–7340.

[55] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, 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. 10684–10695.

[56] K. Pandey, A. Mukherjee, P. Rai, A. Kumar, Diffusevae: Efficient, controllable and high-fidelity generation from low-dimensional latents, Transactions on Machine Learning Research (2022).

[57] H.-Y. Choi, S.-H. Lee, S.-W. Lee, Dddm-vc: Decoupled denoising diffusion models with disentangled representation and prior mixup for verified robust voice conversion, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 2024, pp. 17862–17870.

[58] T. Dockhorn, A. Vahdat, K. Kreis, Score-based generative modeling with critically-damped langevin diffusion, in: International Conference on Learning Representations, 2022.

[59] T. Chen, J. Gu, L. Dinh, E. Theodorou, J. M. Susskind, S. Zhai, Generative modeling with phase stochastic bridge, in: The Twelfth International Conference on Learning Representations, 2024.

[60] Y. Lee, J. Kim, H. Go, M. Jeong, S. Oh, S. Choi, Multi-architecture multi-expert diffusion models, in: Thirty-Eighth Association of Advancement Artificial Intelligence, Vol. 38, 2024, pp. 13427–13436.

[61] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Communications of the ACM 63 (11) (2020) 139–144.

[62] Z. Xiao, K. Kreis, A. Vahdat, Tackling the generative learning trilemma with denoising diffusion GANs, in: International Conference on Learning Representations, 2022.

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

[64] T. Karras, M. Aittala, J. Lehtinen, J. Hellsten, T. Aila, S. Laine, Analyzing and improving the training dynamics of diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 24174–24184.

[65] C.-F. R. Chen, Q. Fan, 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.

[66] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, T. Salimans, Cascaded diffusion models for high fidelity image generation, J. Mach. Learn. Res. 23 (2022) 47–1.

[67] W. Peebles, S. Xie, Scalable diffusion models with transformers, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 4195–4205.

[68] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[69] H. Lu, G. Yang, N. Fei, Y. Huo, Z. Lu, P. Luo, M. Ding, Vdt: General-purpose video diffusion transformers via mask modeling, arXiv preprint arXiv:2305.13311 (2023).

[70] M. Reuss, Ö. E. Yağmurlu, F. Wenzel, R. Lioutikov, Multimodal diffusion transformer: Learning versatile behavior from multimodal goals, in: First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024, 2024.

[71] F. Bao, S. Nie, K. Xue, Y. Cao, C. Li, H. Su, J. Zhu, All are worth words: A vit backbone for diffusion models, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2023, pp. 22669–22679.

[72] H. Zheng, W. Nie, A. Vahdat, A. Anandkumar, Fast training of diffusion models with masked transformers, Transactions on Machine Learning Research (2024).

[73] S. Gao, P. Zhou, M.-M. Cheng, S. Yan, Masked diffusion transformer is a strong image synthesizer, in: Proceedings of the IEEE/CVF international conference on computer vision, 2023, pp. 23164–23173.

[74] Z. Lu, Z. Wang, D. Huang, C. Wu, X. Liu, W. Ouyang, L. BAI, Fit: Flexible vision transformer for diffusion model, in: Forty-first International Conference on Machine Learning, 2024.

[75] Z. Wang, Z. Lu, D. Huang, C. Zhou, W. Ouyang, et al., Fitv2: Scalable and improved flexible vision transformer for diffusion model, arXiv preprint arXiv:2410.13925 (2024).

[76] E. Xie, J. Chen, J. Chen, H. Cai, H. Tang, Y. Lin, Z. Zhang, M. Li, L. Zhu, Y. Lu, et al., Sana: Efficient high-resolution image synthesis with linear diffusion transformers, arXiv preprint arXiv:2410.10629 (2024).

[77] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, J. Tang, Geodiff: A geometric diffusion model for molecular conformation generation, in: International Conference on Learning Representations, 2022.

[78] E. Hoogeboom, V. G. Satorras, C. Vignac, M. Welling, Equivariant diffusion for molecule generation in 3d, in: International Conference on Machine Learning, PMLR, 2022, pp. 8867–8887.

[79] F. Bao, M. Zhao, Z. Hao, P. Li, C. Li, J. Zhu, Equivariant energy-guided sde for inverse molecular design, in: The eleventh international conference on learning representations, 2022.

[80] J. Jo, S. Lee, S. J. Hwang, Score-based generative modeling of graphs via the system of stochastic differential equations, in: International conference on machine learning, PMLR, 2022, pp. 10362–10383.

[81] T. Luo, Z. Mo, S. J. Pan, Fast graph generation via spectral diffusion, IEEE Transactions on Pattern Analysis and Machine Intelligence (2023).

[82] L. Yang, Z. Zhang, W. Zhang, S. Hong, Score-based graph generative modeling with self-guided latent diffusion (2023).

[83] D. Kingma, T. Salimans, B. Poole, J. Ho, Variational diffusion models, Advances in neural information processing systems 34 (2021) 21696–21707.

[84] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, M. Chen, Hierarchical text-conditional image generation with clip latents, arXiv preprint arXiv:2204.06125 (2022).

[85] C. Luo, Understanding diffusion models: A unified perspective, arXiv preprint arXiv:2208.11970 (2022).

[86] Y. Benny, L. Wolf, Dynamic dual-output diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 11482–11491.

[87] 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).

[88] T. Salimans, J. Ho, Progressive distillation for fast sampling of diffusion models, in: International Conference on Learning Representations, 2022.

[89] S. Lin, B. Liu, J. Li, X. Yang, Common diffusion noise schedules and sample steps are flawed, in: Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2024, pp. 5404–5411.

[90] K. Zheng, C. Lu, J. Chen, J. Zhu, Improved techniques for maximum likelihood estimation for diffusion odes, in: International Conference on Machine Learning, PMLR, 2023, pp. 42363–42389.

[91] F. Bao, C. Li, J. Zhu, B. Zhang, Analytic-DPM: an analytic estimate of the optimal reverse variance in diffusion probabilistic models, in: International Conference on Learning Representations, 2022.

[92] J. Song, C. Meng, S. Ermon, Denoising diffusion implicit models, in: International Conference on Learning Representations, 2021.

[93] F. Bao, C. Li, J. Sun, J. Zhu, B. Zhang, Estimating the optimal covariance with imperfect mean in diffusion probabilistic models, in: International Conference on Machine Learning, PMLR, 2022, pp. 1555–1584.

[94] J. Choi, J. Lee, C. Shin, S. Kim, H. Kim, S. Yoon, Perception prioritized training of diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 11472–11481.

[95] Y. Song, C. Durkan, I. Murray, S. Ermon, Maximum likelihood training of score-based diffusion models, Advances in Neural Information Processing Systems 34 (2021) 1415–1428.

[96] S. Chen, S. Chewi, J. Li, Y. Li, A. Salim, A. Zhang, Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions, in: The Eleventh International Conference on Learning Representations, 2023.

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

[98] Z. Chang, E. J. C. Findlay, H. Zhang, H. P. H. Shum, Unifying human motion synthesis and style transfer with denoising diffusion probabilistic models, in: Proceedings of the 2023 International Conference on Computer Graphics Theory and Applications, GRAPP '23, SciTePress, 2023, pp. 64–74.

[99] H. Chefer, Y. Alaluf, Y. Vinker, L. Wolf, D. Cohen-Or, Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models, ACM Transactions on Graphics (TOG) 42 (4) (2023) 1–10.

[100] S. Hong, G. Lee, W. Jang, S. Kim, Improving sample quality of diffusion models using self-attention guidance, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 7462–7471.

[101] J. Ho, T. Salimans, Classifier-free diffusion guidance, in: NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications, 2021.

[102] B. Wallace, A. Gokul, S. Ermon, N. Naik, End-to-end diffusion latent optimization improves classifier guidance, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 7280–7290.

[103] S. Srinivas, F. Fleuret, Rethinking the role of gradient-based attribution methods for model interpretability, in: International Conference on Learning Representations, 2021.

[104] L. Yang, S. Ding, Y. Cai, J. Yu, J. Wang, Y. Shi, Guidance with spherical gaussian constraint for conditional diffusion, in: Forty-first International Conference on Machine Learning, 2024.

[105] J. Zhao, H. Zheng, C. Wang, L. Lan, W. Huang, W. Yang, Null-text guidance in diffusion models is secretly a cartoon-style creator, in: Proceedings of the 31st ACM International Conference on Multimedia, 2023, pp. 5143–5152.

[106] T. Pang, C. Lu, C. Du, M. Lin, S. Yan, Z. Deng, On calibrating diffusion probabilistic models, Advances in Neural Information Processing Systems 36 (2024).

[107] 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, in: Deep Reinforcement Learning Workshop NeurIPS 2022.

[108] L. Zhang, A. Rao, M. Agrawala, Adding conditional control to text-to-image diffusion models, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 3836–3847.

[109] F. Yu, J. Gu, Z. Li, J. Hu, X. Kong, X. Wang, J. He, Y. Qiao, C. Dong, Scaling up to excellence: Practicing model scaling for photo-realistic image restoration in the wild, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 25669–25680.

[110] S. Zhao, D. Chen, Y.-C. Chen, J. Bao, S. Hao, L. Yuan, K.-Y. K. Wong, Uni-controlnet: All-in-one control to text-to-image diffusion models, Advances in Neural Information Processing Systems 36 (2024).

[111] J. Wang, Z. Yue, S. Zhou, K. C. Chan, C. C. Loy, Exploiting diffusion prior for real-world image super-resolution, International Journal of Computer Vision (2024) 1–21.

[112] C. Mou, X. Wang, L. Xie, Y. Wu, J. Zhang, Z. Qi, Y. Shan, T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 2024, pp. 4296–4304.

[113] N. Stracke, S. A. Baumann, J. Susskind, M. A. Bautista, B. Ommer, Ctrloralter: Conditional loradapter for efficient 0-shot control & altering of t2i models (2024).

[114] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-rank adaptation of large language models, in: International Conference on Learning Representations, 2022.

[115] H. Chung, B. Sim, 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. 12413–12422.

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

[117] T. Huang, Y. Zhang, M. Zheng, S. You, F. Wang, C. Qian, C. Xu, Knowledge diffusion for distillation, Advances in Neural Information Processing Systems 36 (2024).

[118] E. Luhman, T. Luhman, Knowledge distillation in iterative generative models for improved sampling speed, arXiv preprint arXiv:2101.02388 (2021).

[119] C. Meng, R. Rombach, R. Gao, D. Kingma, S. Ermon, J. Ho, T. Salimans, On distillation of guided diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 14297–14306.

[120] Y.-T. Hsiao, S. Khodadadeh, K. Duarte, W.-A. Lin, H. Qu, M. Kwon, R. Kalarot, Plug-and-play diffusion distillation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 13743–13752.

[121] Y. Song, P. Dhariwal, M. Chen, I. Sutskever, Consistency models, in: Proceedings of the 40th International Conference on Machine Learning, 2023, pp. 32211–32252.

[122] G. Fang, X. Ma, X. Wang, Structural pruning for diffusion models, in: Thirty-seventh Conference on Neural Information Processing Systems, 2023.

[123] A. Shih, S. Belkhale, S. Ermon, D. Sadigh, N. Anari, Parallel sampling of diffusion models, Advances in Neural Information Processing Systems 36 (2024).

[124] L. Liu, Y. Ren, Z. Lin, Z. Zhao, Pseudo numerical methods for diffusion models on manifolds, in: International Conference on Learning Representations, 2022.

[125] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, J. Zhu, Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, Advances in Neural Information Processing Systems 35 (2022) 5775–5787.

[126] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, J. Zhu, Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models, arXiv preprint arXiv:2211.01095 (2022).

[127] Z. Duan, C. Wang, C. Chen, J. Huang, W. Qian, Optimal linear subspace search: Learning to construct fast and high-quality schedulers for diffusion models, in: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 2023, pp. 463–472.

[128] D. Watson, J. Ho, M. Norouzi, W. Chan, Learning to efficiently sample from diffusion probabilistic models, arXiv preprint arXiv:2106.03802 (2021).

[129] S. AI, Stable diffusion 2 (November 2022).

[130] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, et al., Stable video diffusion: Scaling latent video diffusion models to large datasets, arXiv preprint arXiv:2311.15127 (2023).

[131] Z. Zheng, X. Peng, T. Yang, C. Shen, S. Li, H. Liu, Y. Zhou, T. Li, Y. You, Open-sora: Democratizing efficient video production for all (March 2024).

[132] B. F. Labs, Flux (August 2024).

[133] X. Li, Y. Dai, Q. Qu, Understanding generalizability of diffusion models requires rethinking the hidden gaussian structure, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.

[134] Z. Kadkhodaie, F. Guth, E. P. Simoncelli, S. Mallat, Generalization in diffusion models arises from geometry-adaptive harmonic representations, in: The Twelfth International Conference on Learning Representations, 2024.

[135] J. Jiang, Z. Zuo, G. Wu, K. Jiang, X. Liu, A survey on all-in-one image restoration: Taxonomy, evaluation and future trends, arXiv preprint arXiv:2410.15067 (2024).

[136] A. Shmakov, K. Greif, M. Fenton, A. Ghosh, P. Baldi, D. Whiteson, End-to-end latent variational diffusion models for inverse problems in high energy physics, Advances in Neural Information Processing Systems 36 (2024).

[137] C. Gou, Y. Yu, Z. Guo, C. Xiong, M. Cai, Cascaded learning with transformer for simultaneous eye landmark, eye state and gaze estimation, Pattern Recognition (2024) 110760.

[138] L. Lorch, A. Krause, B. Schölkopf, Causal modeling with stationary diffusions, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2024, pp. 1927–1935.

[139] P. Chao, P. Blöbaum, S. P. Kasiviswanathan, Interventional and counterfactual inference with diffusion models, arXiv preprint arXiv:2302.00860 4 (2023) 16.

[140] X. Gu, C. Du, T. Pang, C. Li, M. Lin, Y. Wang, On memorization in diffusion models, arXiv preprint arXiv:2310.02664 (2023).

[141] G. Zhang, K. Wang, X. Xu, Z. Wang, H. Shi, Forget-me-not: Learning to forget in text-to-image diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 1755–1764.

[142] C. Chen, D. Liu, C. Xu, Towards memorization-free diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 8425–8434.

[143] K. Erickson, Intellectual property and creative industries policy in the uk, in: Research Handbook on Intellectual Property and Creative Industries, Edward Elgar Publishing, 2018, pp. 79–84.

[144] Y. Choi, J. Park, H. Kim, J. Lee, S. Park, Fair sampling in diffusion models through switching mechanism, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 2024, pp. 21995–22003.

[145] R. Gao, W. Liu, Ddgr: Continual learning with deep diffusion-based generative replay, in: International Conference on Machine Learning, PMLR, 2023, pp. 10744–10763.

[146] Y. Wei, G. Tyson, Understanding the impact of ai-generated content on social media: The pixiv case, in: Proceedings of the 32nd ACM International Conference on Multimedia, 2024, pp. 6813–6822.