Frontiers of Computer Science CN 10-1014/TP
https://doi.org/10.1007/s11704-026-52197-3 ISSN 2095-2228
REVIEW ARTICLE

A Survey of Edge-side Text-to-image Diffusion Models

Zi-Hao QIU[1], Wenhao YANG[1], Lijun ZHANG[1]

1 State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

Front. Comput. Sci., **Just Accepted Manuscript** •10.1007/s11704-026-52197-3
https://journal.hep.com.cn on May 6, 2026

REVIEW ARTICLE

A Survey of Edge-side Text-to-image Diffusion Models

Zi-Hao Qiu[1], Wenhao Yang[1], Lijun Zhang[1]✉

1. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023, China.

Received December 29, 2025; accepted May 6, 2026

E-mail: zhanglj@lamda.nju.edu.cn.

Abstract

Diffusion models have emerged as a central focus in generative artificial intelligence due to their robust theoretical foundations and exceptional generation capabilities. Edge-side text-to-image diffusion models represent a critical application area, targeting efficient, low-latency image generation on resource-constrained platforms while preserving user privacy and data security. This survey systematically reviews recent advances in edge-side text-to-image diffusion models across theoretical foundations, algorithmic improvements, model architectures, and deployment optimization. We first analyze the core mathematical frameworks-Denoising Diffusion Probabilistic Models (DDPM), Score-based Generative Models (SGM), and Score-based Stochastic Differential Equations-alongside the mathematical basis for conditional generation. We then examine key improvements including latent space modeling, likelihood estimation optimization, efficient sampling algorithms, and consistency-based and flow matching methods. Subsequently, we explore the design, training, and evaluation of large-scale text-to-image models, emphasizing mainstream architectures and their trade-offs. Finally, we summarize essential edge deployment techniques: model quantization, lightweight architecture design, knowledge and step distillation, and computational optimizations, providing practical strategies for efficient inference on constrained hardware. This review aims to serve as a comprehensive reference for researchers, bridging theory and practice, and to promote the implementation and innovation of efficient text-to-image diffusion models in real-world scenarios such as mobile deployment.

Key words

Diffusion Models; Text-to-image Generation; Edge-side Deployment

1 Introduction

Artificial Intelligence Generated Content (AIGC) [1–5] has emerged as an important branch of next-generation artificial intelligence technologies, aiming to leverage deep learning models to learn the distribution of large-scale data and automatically generate high-quality digital content across modalities, including text, images, audio, and video. Traditional generative approaches primarily encompass Variational Autoencoders (VAE) [1,6], Generative Adversarial Networks (GAN) [5], and autoregressive models [2]. Specifically, VAEs learn latent representations of data via variational inference, offering stable training but often limited generation quality. GANs utilize an adversarial training paradigm and have achieved remarkable results in image synthesis, but are challenged by training instability and mode collapse. Autoregressive models generate images pixel by pixel, theoretically modeling complex distributions, yet suffer from slow generation speeds and limited parallelism. Overall, these traditional methods often struggle to balance generation quality, training stability, and inference efficiency.

Diffusion models have recently risen as a novel generative modeling paradigm, achieving high-fidelity content generation by simulating the progressive denoising of data. They have yielded breakthrough advances in image synthesis [7–10], computer vision [11–14], natural language processing [15–17], and time-series modeling [18–20]. The core idea is to sequentially corrupt data with noise through a forward diffusion process until it becomes pure noise, then train a neural network to learn the reverse denoising process, thereby generating realistic data from noise. Compared with traditional generative approaches, diffusion models offer more stable training, higher quality generation, and broader modality coverage. They have demonstrated outstanding visual generation capabilities in representative applications such as Stable Diffusion, DALL·E 2, and Midjourney. With the improvement of computational power in mobile devices and IoT terminals, on-device diffusion models [21–24] have emerged, enabling diffusion models to operate independently on edge devices, free from reliance on cloud servers. This paradigm bolsters user privacy, reduces bandwidth and latency, and enhances personalized experiences, holding significant research and practical value for on-device applications such as mobile photography, real-time interaction, personalized recommendation, and assisted design. However, due to limited computational and memory resources of edge devices, developing on-device diffusion models poses considerable challenges in terms of efficiency, model compactness, and low power consumption. Achieving high-quality and controllable content generation under resource constraints

remains a critical open problem in both academia and industry.

In this literature review, we systematically examine the core technological advances in edge-side text-to-image diffusion models, outlining a comprehensive technical roadmap from theoretical foundations to practical deployment. The paper is organized into four progressive sections, covering mathematical foundations, algorithmic innovations, practical model implementation, and deployment optimization for the edge. The first section focuses on the mathematical theory underpinning diffusion models. We detail three core theoretical frameworks: Denoising Diffusion Probabilistic Models [8, 25], score-based generative models [9, 26], and score-based stochastic differential equation approaches [10, 27, 28]. On this basis, we thoroughly derive the mathematical formulations for conditional generation and elucidate the probabilistic modeling mechanisms of text-to-image generation, providing a solid foundation for subsequent algorithmic and application advances. The second section reviews key algorithmic innovations in diffusion models, highlighting four major areas: (1) diffusion modeling in latent space; (2) likelihood estimation improvements; (3) efficient sampling algorithms; (4) consistency-based and flow matching methods. These improvements have substantially increased generation quality, efficiency, and practical usability, many of which have become standard in current mainstream methods. In the third section, we delve into practical considerations for implementing text-to-image diffusion models, including model architecture design, training data construction, and evaluation benchmarks, providing systematic guidance for efficient model training and assessment to support real-world deployment and widespread adoption. The fourth section focuses on core technologies for deploying text-to-image diffusion models in edge computing scenarios such as mobile devices. This includes (1) parameter quantization and sparsification; (2) lightweight network architecture design; (3) model compression strategies based on knowledge distillation; and (4) low-level computational optimization. The synergistic application of these technologies further drives the deployment and large-scale application of efficient text-to-image models on the edge.

2 Related Work

In recent years, driven by the rapid development of diffusion models, a large number of relevant survey papers have emerged in academia [29–36]. Given the profound mathematical foundations underlying diffusion models, many reviews [29–31, 33–36] choose to start from the basic theory, introducing core theoretical frameworks such as denoising diffusion probabilistic models, score-based generative models, and their associated stochastic differential equations. In the first section of this survey, we also introduce these three principal theoretical frameworks, and further provide more detailed, independent, and self-consistent mathematical definitions. In addition, we supplement a mathematical formulation of conditional generation and review typical guidance methods. Crucially, unlike prior generic surveys, we explicitly bridge these theoretical formulations to realistic edge-side constraints. We systematically analyze how the iterative Markovian nature of these foundational equations directly translates to severe memory bandwidth pressure on mobile SoCs, thereby establishing the mathematical necessity for the edge-oriented optimizations discussed in subsequent sections.

Building on these theoretical frameworks, a series of breakthrough improvements have successively emerged, significantly enhancing both generative efficiency and quality. Among the most representative advances are innovations in efficient latent-space architectures and advanced sampling acceleration algorithms [31, 33–35]. Furthermore, Bie et al. [31] summarize text-image alignment and MoE structures, while Cao et al. [35] provide an overview of physical mechanism-inspired processes. Compared with previous reviews, the second section of this survey provides a more comprehensive summary of four major directions of model improvements. More importantly, we systematically reorganize these algorithms not merely as mathematical novelties, but as targeted solutions to physical hardware bottlenecks. For instance, we evaluate latent-space compression specifically as a mechanism for overcoming mobile RAM limits, and analyze SDE/ODE solvers strictly through the lens of per-step computational cost (e.g., NFE limits) and cache thrashing on edge devices.

At the implementation level—including model architecture, training/testing data, and evaluation criteria—some prior surveys have focused on network architectures [31, 33, 34] and standard vision metrics [31, 33]. In the third section, we present a comprehensive exposition of network evolutions and dataset strategies. To specifically address the realities of on-device deployment, we uniquely highlight the limitations of standard metrics (e.g., FID/CLIP) under small-batch edge evaluation scenarios. Furthermore, we expand the scope of evaluation to include critical system-level benchmarks—such as peak memory footprint, end-to-end latency on Neural Processing Units (NPUs), and thermal throttling—that are indispensable for engineering realization.

With the increasing demand for on-device deployment, a growing body of research focuses on architecture design and inference optimizations in resource-constrained environments. Song et al. [32] classify model light-weighting techniques into knowledge distillation, quantization, pruning, and algorithmic optimizations. In the fourth section, this survey comprehensively summarizes multi-level on-device deployment technologies, spanning from training paradigms to low-level computational optimizations. While works like Song et al. [32] focus broadly on generic lightweighting, our survey uniquely filters theoretical, algorithmic, and engineering advancements strictly through the distinct constraints of edge-side text-to-image synthesis, providing a holistic and highly specialized technical perspective for future mobile deployments.

3 Mathematical Foundations

This section presents the core theoretical foundations of diffusion models, covering the three major paradigms: Denoising Diffusion Probabilistic Models (DDPM) [8, 25], Score-based Generative Models (SGM) [9, 26], and Score-based Stochastic Differential Equation models (Score SDE) [10, 28]. Fundamentally, these methods all follow the "disturb-and-generate" framework: they first design a forward diffusion process in which structured data are gradually perturbed by

stepwise injection of noise, driving the data distribution towards pure noise; subsequently, new samples are generated via a reverse denoising reconstruction process. In this section, we sequentially review the forward and reverse mechanisms of each diffusion model, elucidate their intrinsic relationships, and finally introduce the mathematical formulation of diffusion models under conditional generation scenarios.

We first unify the notation commonly used in diffusion models. Let the discrete time steps be denoted by $t \in \{0, 1, \cdots, T\}$, and the continuous time variable by $t \in [0, T]$. Let the original data be $\mathbf{x}_0 \in \mathbb{R}^d$ following the true data distribution $q(\mathbf{x}_0)$. Diffusion models construct a data trajectory $\{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T\}$ by successively adding Gaussian noise to the original data, such that $\mathbf{x}_T$ eventually approaches a standard Gaussian distribution. The strength of noise injection is controlled by a schedule coefficient β_t (for the discrete case) or a time-dependent function $\beta(t)dt$ (for the continuous case).

3.1 Denoising Diffusion Probabilistic Models (DDPMs)

DDPMs are generative models built upon two coupled Markov chains that map data to noise and back [8, 25]. The forward diffusion process gradually perturbs clean data into a simple prior (typically a standard Gaussian) using a fixed noise schedule; the reverse generative process reconstructs data from noise via a learned, parameterized Markov chain. Sampling proceeds by ancestral sampling [37], transforming an initial noise sample into data through sequential denoising steps.

3.1.1 Forward and Reverse Processes: Core Formulation

Let $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ denote data. The forward (diffusion) process defines a Markov chain $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T$ with Gaussian transitions

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\big(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t\mathbf{I}\big), \quad \beta_t \in (0,1), \tag{1}$$

which admits the closed-form marginal

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}\big(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\big), \tag{2}$$

where $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$. As T grows and $\bar{\alpha}_T \to 0$, the marginal $q(\mathbf{x}_T)$ approaches $\mathcal{N}(\mathbf{0}, \mathbf{I})$ [25].

The reverse (generative) process is parameterized as

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\big(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)\big), \tag{3}$$

with $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and sequential sampling $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ until $\mathbf{x}_0$. A practical and widely used parameterization predicts the noise in $\mathbf{x}_t$:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(\mathbf{x}_t, t), \tag{4}$$

where $\epsilon_\theta(\mathbf{x}_t, t)$ is a neural network estimating the Gaussian noise added at step t [8]. At inference time, one typically uses

$$\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \sigma_q(t)\,\mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{5}$$

with $\sigma_q(t)$ chosen to match the forward-process posterior variance (see Appendix 8.1).

3.1.2 Training Objective: Noise-Prediction View

Training maximizes a variational lower bound (ELBO) on $\log p_\theta(\mathbf{x}_0)$ [25], which simplifies to a weighted denoising (noise-prediction) objective [8]:

$$\mathcal{L}_{\text{DDPM}} = \sum_{t=1}^{T} w_t\, \mathbb{E}_{\mathbf{x}_0, \epsilon}\Big[\,\big\|\epsilon_\theta(\mathbf{x}_t, t) - \epsilon\big\|^2\Big], \tag{6}$$

where $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and the weights w_t arise from the ELBO's KL terms and depend on the noise schedule and posterior variance (see Appendix 8.1 for the exact form). In practice, constant or schedule-aware weights are commonly used without changing the fundamental estimator.

Together, Eqs. (1)–(6) summarize the core mechanics of DDPMs: (i) a tractable Gaussian forward process, (ii) a parameterized Gaussian reverse process that predicts noise, and (iii) a training objective equivalent to supervised denoising across diffusion timesteps. For complete derivations and posterior forms, refer to Appendix 8.1.

3.2 Score-based Generative Models (SGMs)

This section distills the core ideas and principal formulas of score-based generative models (SGMs). We introduce the Stein score function and its use in Langevin sampling, outline learning via score matching (with emphasis on denoising score matching), and summarize the Noise Conditional Score Network (NCSN) as a practical, state-of-the-art approach. Detailed derivations are deferred to Appendix 8.2.

3.2.1 Stein Score Function and Langevin Sampling

For a data distribution with density $p(\mathbf{x})$, the Stein score (hereafter simply the score) is the gradient of the log-density with respect to the data:

$$\mathbf{s}(\mathbf{x}) := \nabla_{\mathbf{x}} \log p(\mathbf{x}). \tag{7}$$

This differs from the Fisher score $\nabla_\theta \log p_\theta(\mathbf{x})$ in that Eq. (7) operates in data space, capturing directions of steepest increase in likelihood [38]. Given an estimate of $\mathbf{s}(\mathbf{x})$, one can sample from $p(\mathbf{x})$ via unadjusted Langevin dynamics:

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\alpha}{2}\,\mathbf{s}(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\alpha}\,\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{8}$$

which converges to $p(\mathbf{x})$ under mild regularity conditions as $\alpha \to 0$ and $T \to \infty$ [39]. In practice, finite-step implementations are commonly used without Metropolis–Hastings correction [40, 41].

3.2.2 Learning Scores via Score Matching

The learning goal is to approximate $\mathbf{s}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$ without requiring the (intractable) normalization of $p(\mathbf{x})$. Several approaches exist:

Explicit Score Matching (ESM). Replace $p(\mathbf{x})$ by a kernel density estimate (KDE) $q_h(\mathbf{x})$ and minimize

$$\mathcal{L}_{\text{ESM}} \approx \frac{1}{2}\,\mathbb{E}_{q_h(\mathbf{x})}\big[\|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log q_h(\mathbf{x})\|^2\big], \tag{9}$$

which can be inaccurate in high dimensions or with limited data [42].

Implicit Score Matching (ISM). Avoid density estimation by minimizing

$$\mathcal{L}_{\text{ISM}} := \mathbb{E}_{p(\mathbf{x})}\big[\text{Tr}(\nabla_{\mathbf{x}}\mathbf{s}_\theta(\mathbf{x})) + \tfrac{1}{2}\|\mathbf{s}_\theta(\mathbf{x})\|^2\big], \tag{10}$$

derived via integration by parts [38]. The trace term can be costly to compute at scale [9].

Denoising Score Matching (DSM). Corrupt $\mathbf{x}_0 \sim p(\mathbf{x})$ by $q(\mathbf{x}'|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}'|\mathbf{x}_0, \sigma^2\mathbf{I})$ and minimize

$$\mathbb{E}_{q(\mathbf{x}'|\mathbf{x}_0)\,p(\mathbf{x}_0)}\left[\frac{1}{2}\left\|\mathbf{s}_\theta(\mathbf{x}') - \nabla_{\mathbf{x}'}\log q(\mathbf{x}'|\mathbf{x}_0)\right\|^2\right], \tag{11}$$

whose Gaussian target simplifies to predicting $-\epsilon/\sigma$ when $\mathbf{x}' = \mathbf{x}_0 + \sigma\epsilon$ [42]. Minimization recovers the score of the noisy marginal $q_\sigma(\mathbf{x}')$.

3.2.3 Noise Conditional Score Network (NCSN)

To address low-manifold support and sparse regions, NCSN trains over a sequence $\{\sigma_i\}_{i=1}^L$ with $\sigma_1 > \cdots > \sigma_L > 0$ [9, 10]:

$$\mathbf{x}' = \mathbf{x}_0 + \sigma_i\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{12}$$

and minimizes the scale-conditioned objective

$$\mathcal{L}_{\text{NCSN}} = \mathbb{E}_{i,\,q(\mathbf{x}'|\mathbf{x}_0),\,p(\mathbf{x}_0)}\left[\frac{1}{2}\lambda(\sigma_i)\left\|\mathbf{s}_\theta(\mathbf{x}', \sigma_i) + \frac{\epsilon}{\sigma_i}\right\|^2\right], \tag{13}$$

where $\lambda(\sigma) = \sigma^2$. Sampling employs annealed Langevin dynamics from coarse to fine noise:

$$\alpha_i \propto \sigma_i^2, \qquad \mathbf{x}' \leftarrow \mathbf{x}' + \frac{\alpha_i}{2}\mathbf{s}_\theta(\mathbf{x}', \sigma_i) + \sqrt{\alpha_i}\,\epsilon,\ \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{14}$$

This improves traversal of low-density regions and sample quality [9]. The learned score in NCSN and the noise-prediction network in diffusion models are closely related up to a scale factor [10].

3.3 Score Stochastic Differential Equations (Score SDEs)

Diffusion models, including Denoising Diffusion Probabilistic Models (DDPMs) [8] and Score-Based Generative Models (SGMs) [9, 10], can be unified and generalized through the lens of stochastic differential equations (SDEs) [10]. This continuous-time perspective provides a powerful mathematical framework for understanding and improving diffusion-based generative modeling.

The forward process in diffusion models can be described by a general SDE of the form:

$$d\mathbf{x} = \mathbf{f}(t, \mathbf{x})dt + g(t)d\mathbf{w}, \tag{15}$$

where $\mathbf{f}(t, \mathbf{x})$ is the drift coefficient, $g(t)$ is the diffusion coefficient, and $\mathbf{w}$ is a standard Wiener process. This SDE progressively transforms the data distribution $p_0(\mathbf{x})$ into a simple prior distribution $p_T(\mathbf{x})$ (typically Gaussian) as time t evolves from 0 to T.

Remarkably, the reverse generative process corresponds to the time-reversal of this forward SDE [43], given by:

$$d\mathbf{x} = \left[\mathbf{f}(t, \mathbf{x}) - g(t)^2\nabla_{\mathbf{x}}\log p_t(\mathbf{x})\right]dt + g(t)d\bar{\mathbf{w}}, \tag{16}$$

where $\nabla_{\mathbf{x}}\log p_t(\mathbf{x})$ is the score function and $\bar{\mathbf{w}}$ is a reverse-time Wiener process.

Additionally, Song et al. [10] introduced the probability flow ordinary differential equation (ODE):

$$d\mathbf{x} = \left[\mathbf{f}(t, \mathbf{x}) - \frac{1}{2}g(t)^2\nabla_{\mathbf{x}}\log p_t(\mathbf{x})\right]dt, \tag{17}$$

which shares the same marginal distributions with the reverse SDE but enables deterministic sampling.

For DDPM [8], the continuous-time limit corresponds to:

$$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}dt + \sqrt{\beta(t)}d\mathbf{w} \quad \text{(forward)} \tag{18}$$

$$d\mathbf{x} = -\beta(t)\left[\frac{\mathbf{x}}{2} + \nabla_{\mathbf{x}}\log p_t(\mathbf{x})\right]dt + \sqrt{\beta(t)}d\bar{\mathbf{w}} \quad \text{(reverse)} \tag{19}$$

For SGM with noise conditional score networks [10], the SDE representation is:

$$d\mathbf{x} = \sqrt{\frac{d[\sigma(t)]^2}{dt}}d\mathbf{w} \quad \text{(forward)} \tag{20}$$

$$d\mathbf{x} = -\frac{d[\sigma(t)]^2}{dt}\nabla_{\mathbf{x}}\log p_t(\mathbf{x})dt + \sqrt{\frac{d[\sigma(t)]^2}{dt}}d\bar{\mathbf{w}} \quad \text{(reverse)} \tag{21}$$

The key challenge in implementing these reverse processes is estimating the score function $\nabla_{\mathbf{x}}\log p_t(\mathbf{x})$, which is typically learned using denoising score matching [42] or similar objectives. Once trained, samples can be generated using various numerical methods including SDE solvers [10, 44], ODE solvers [10, 28, 45, 46], or predictor-corrector schemes [10].

3.3.1 Derivations Summary

The detailed mathematical derivations connecting the discrete-time formulations of DDPM and SGM to their continuous-time SDE representations are provided in Appendix 8.3. These derivations involve taking appropriate continuous-time limits and applying Taylor expansions to transition from the discrete update rules to their differential counterparts.

3.4 Core Methods for Conditional Generation

A central challenge in conditional generation with diffusion models is to effectively inject external information (e.g., text, class labels) into the reverse denoising process. Let y denote the conditioning signal. In principle, conditional generation can be achieved by modeling $p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}, y)$ via architectural conditioning (e.g., cross-attention [47]). However, a direct formulation often underperforms because (i) the standard diffusion objective aligns data distributions but does not explicitly enforce semantic consistency with y; and (ii) fine-grained aspects of y (e.g., layout, attributes) may require step-dependent adjustments that static fusion struggles to capture [48].

3.4.1 Reverse Process of Conditional Diffusion Models (Overview)

Following the DDPM formulation [7], the conditional reverse transition can be written using Bayes' rule as

$$p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}, y) \propto p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1})\,p(y \mid \mathbf{x}_t), \tag{22}$$

where $p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = \mathcal{N}(\mu, \Sigma)$ is the unconditional Gaussian transition. A first-order treatment of $\log p(y \mid \mathbf{x}_t)$ around $\mathbf{x}_t = \mu$ shows that $p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}, y)$ remains Gaussian with the same covariance and a shifted mean:

$$p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}, y) \approx \mathcal{N}\big(\mu + \Sigma\, g,\ \Sigma\big),\ g \triangleq \nabla_{\mathbf{x}_t}\log p(y \mid \mathbf{x}_t)\big|_{\mathbf{x}_t=\mu}. \tag{23}$$

Thus, conditional guidance amounts to specifying $p(y \mid \mathbf{x}_t)$ (or a surrogate) and using its gradient with respect to $\mathbf{x}_t$ to shift the mean.

Table 1 Comparison of algorithmic innovations for diffusion models, categorized by their intervention phase (Training vs. Inference) and their specific pros/cons for edge deployment.

Algorithmic Paradigm	Optimization Phase	Advantages for Edge-side	Disadvantages / Challenges on Edge
Latent Diffusion	Training & Inference	Drastically reduces memory footprint (RAM) and computation by operating in a compressed latent space.	VAE decoding still consumes significant memory spikes; requires full retraining.
First-order Solvers (e.g., Euler)	Purely Inference	NFE=1 per step strictly limits memory bandwidth thrashing; highly stable for distilled models.	May require a larger number of total steps (T) to converge for non-distilled models.
High-order Solvers (e.g., DPM-Solver)	Purely Inference	Achieves high-quality generation in fewer total steps (e.g., 10-20 steps).	NFE > 1 per step causes severe cache misses and thermal throttling on mobile SoCs.
Step Distillation (e.g., LCM)	Training (Distillation)	Extreme latency reduction; enables 1-to-4 step generation, making real-time mobile T2I possible.	Susceptible to mode collapse (reduced diversity); loss of fine-grained high-frequency details.
Flow-Matching	Training (from scratch)	Straighter probability flow enables faster convergence and easier simulation during inference.	Heavy upfront training cost; still requires multi-step sampling without further distillation.

3.4.2 Classifier Guidance

Classifier Guidance [7] instantiates $p(y \mid \mathbf{x}_t)$ with an auxiliary classifier $p_\phi(y \mid \mathbf{x}_t)$ trained on noisy inputs. During sampling, the reverse mean is shifted by the classifier gradient:

$$\hat{\mu}_\theta(\mathbf{x}_t \mid y) = \mu_\theta(\mathbf{x}_t) + s \cdot \Sigma_\theta(\mathbf{x}_t) \cdot \nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t), \tag{24}$$

where s controls guidance strength. This method is effective but requires training and maintaining a noise-aware classifier, and may be constrained by classifier accuracy for complex conditions (e.g., long texts).

3.4.3 CLIP Guidance

CLIP [49] aligns images and text via contrastive learning. Using CLIP's encoders (f, g), one may treat the similarity $f(\mathbf{x}_t) \cdot g(y)$ as a surrogate score for $p(y \mid \mathbf{x}_t)$ [50–52]. The reverse mean update mirrors the classifier case:

$$\hat{\mu}_\theta(\mathbf{x}_t \mid y) = \mu_\theta(\mathbf{x}_t) + s \cdot \Sigma_\theta(\mathbf{x}_t) \cdot \nabla_{\mathbf{x}_t} \big(f(\mathbf{x}_t) \cdot g(y)\big). \tag{25}$$

In practice, CLIP benefits from noise-aware training or denoising-aware features to ensure meaningful gradients at intermediate timesteps, which can limit scalability.

3.4.4 Classifier-Free Guidance

Classifier-Free Guidance (CFG) [53] avoids an explicit classifier by relating conditional and unconditional scores via Bayes' rule:

$$\nabla_{\mathbf{x}_t} \log p(y \mid \mathbf{x}_t) \propto \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid y) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t). \tag{26}$$

In DDPMs, the denoiser $\epsilon_\theta(\mathbf{x}_t \mid y)$ parameterizes the conditional score, while $\epsilon_\theta(\mathbf{x}_t \mid \emptyset)$ (obtained by dropping y during training) parameterizes the unconditional score. The guided prediction is

$$\hat{\epsilon}_\theta(\mathbf{x}_t \mid y) = \epsilon_\theta(\mathbf{x}_t \mid \emptyset) + s \cdot \big(\epsilon_\theta(\mathbf{x}_t \mid y) - \epsilon_\theta(\mathbf{x}_t \mid \emptyset)\big), \tag{27}$$

which effectively amplifies the conditional signal. CFG has become the default guidance mechanism for text-to-image diffusion due to its simplicity and strong performance.

3.5 Theoretical Formulations versus Edge Constraints

While the rigorous formulations of DDPM, SGM, and Score-SDEs provide the mathematical foundation for high-quality generation, they inherently conflict with resource-constrained edge devices. The core bottleneck lies in the iterative Markovian nature of these formulations. For instance, simulating the reverse SDE or evaluating DDPM requires hundreds to thousands of iterative steps (T). On an edge device, each step translates to a full forward pass of a massive U-Net (e.g., 860M parameters). Since mobile SoCs possess limited SRAM, the model weights must be repeatedly fetched from the slower DRAM. This extreme memory bandwidth pressure—rather than pure compute (FLOPs)—is the primary cause of intolerable latency (often minutes per image) and severe battery drain on mobile phones, rendering the raw theoretical formulations impractical for direct on-device deployment without the architectural and algorithmic interventions discussed in subsequent sections.

■ 4 Algorithmic Innovations

In recent years, diffusion models have emerged as a research hotspot in the field of generative models, owing to their solid theoretical foundations and exceptional generative capabilities. However, early diffusion models in practical applications still face three key challenges: (1) the assumption of Euclidean space is often inadequate to accurately capture the complex manifold structures commonly present in real-world data; (2) insufficient accuracy in likelihood estimation constrains both the quality of generated samples and the controllability of the model; and (3) low sampling efficiency limits the speed of generation. To address these bottlenecks, researchers have systematically explored improvements from multiple directions. Firstly, by introducing the diffusion process into latent spaces, models have achieved more effective modeling of structured or complex data distributions. Secondly, more precise methods for likelihood estimation and probability density modeling have been developed to enhance both the quality and control-

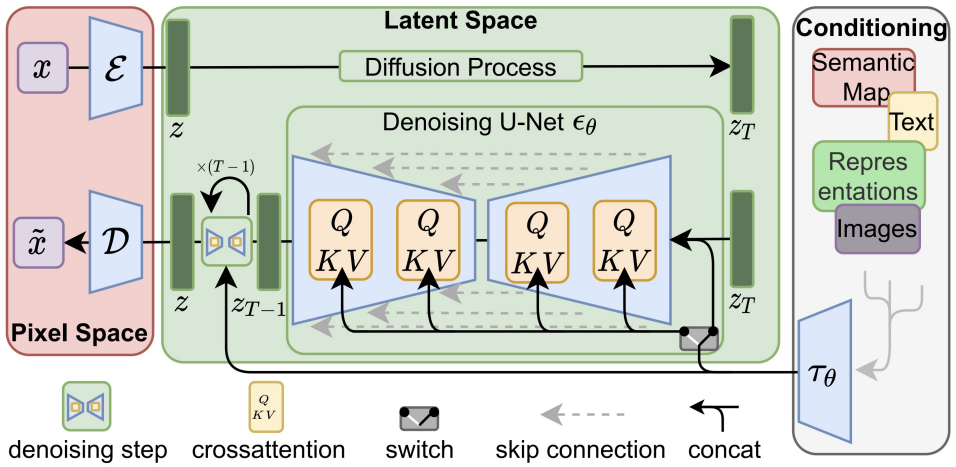

Fig. 1 Overview of the latent diffusion model (LDM) (figure adapted from [3]). The input image x is first encoded into a latent space by the encoder $\mathcal{E}$. LDM then performs the diffusion process in this compressed latent space, where a denoising U-Net learns to reverse the noise addition process over multiple timesteps, incorporating external conditioning information. Finally, the decoder $\mathcal{D}$ transforms the denoised latent representation back into the final output image.

lability of generated samples. Additionally, the design of more efficient sampling algorithms has significantly improved inference speed.

Meanwhile, methods based on the consistency criterion further improve computational efficiency by optimizing the generative process of the probability flow ODE (17). More recently, the emerging flow matching approach provides an alternative pathway, directly learning the distribution transport vector field to enable more efficient generation. In the following sections, we systematically review representative methods and the technical evolution along these directions.

Before delving into specific algorithmic innovations, it is crucial to classify these methods based on their intervention phase: Training-side versus Inference-side optimizations.

- **Training-side techniques** (e.g., Latent Diffusion, Flow-matching, Consistency Models) require fundamentally altering the model architecture or objective function. These methods demand massive computational resources on the cloud to train from scratch or heavily fine-tune, but they yield model weights that are natively more efficient for edge deployment.
- **Inference-side techniques** (e.g., Training-free ODE Solvers, CFG scaling adjustments) do not require retraining. They optimize the sampling trajectory or compute graph directly during deployment, offering plug-and-play latency reductions at the cost of potential quality degradation.

To alleviate text density and provide a clear overview, Table 1 summarizes these primary algorithmic paradigms, explicitly categorizing their intervention phase and comparing their advantages and disadvantages in the context of edge-side deployment.

4.1 Efficient Diffusion Model in Latent Spaces

Conventional diffusion models typically operate directly in the original data space (e.g., pixel space), a paradigm that faces two fundamental challenges. On one hand, the diffusion process in such high-dimensional spaces requires a large number of network evaluation steps to achieve gradual denoising, resulting in low sampling efficiency. On the other hand, redundant high-frequency components in the data space may interfere with the extraction of essential features by the model. To address these issues, researchers have drawn inspiration from the concept of manifold learning to construct lower-dimensional latent spaces, within which the diffusion process is performed. This gives rise to a cascaded "encode-diffuse-decode" architecture, significantly improving computational efficiency.

The Latent Score-Based Generative Model (LSGM) [54] is the first framework to integrate Score-Based Generative Models (SGM) with Variational Autoencoders (VAE). Specifically, the VAE encoder maps high-dimensional data into a lower-dimensional latent space, where the SGM is then trained and sampled. This framework substantially improves computational efficiency while preserving the original SGM's sample quality and probability coverage, thereby achieving a favorable balance between model performance and resource consumption.

In contrast to the joint training paradigm of LSGM, the Latent Diffusion Model (LDM) [3] adopts a staged optimization strategy. The core of this technique lies in first independently training a hierarchical VAE with perceptual equivalence to construct a semantically compressed latent space, followed by training the diffusion probabilistic model in this well-converged latent space. This decoupled training framework not only reduces the difficulty of model optimization but, more importantly, establishes a "perceptually equivalent, semantically compressed" latent space. In this way, the diffusion process can focus more effectively on learning essential features, enabling significant improvements in computational efficiency and visual fidelity for image generation tasks. Figure 1 illustrates the core concept of LDMs.

4.2 Advances in Likelihood Estimation

The training objective of diffusion models is, in essence, the evidence lower bound (ELBO) of the data log-likelihood. However, this lower bound often suffers from looseness, which hinders the model from achieving optimal likelihood performance. Recent research has aimed to systematically improve noise scheduling strategies and the parameterization of variance in the reverse process, thereby tightening the variational bound and enhancing the data modeling capacity of

diffusion models. In this section, we summarize key advances in this area from three perspectives: noise schedule optimization, generalized parameterization frameworks, and reverse variance optimization.

Traditionally, noise scheduling in diffusion models relies on heuristic designs, which may not adapt well to the dynamic denoising requirements of complex data distributions. Improved Denoising Diffusion Probabilistic Models (iDDPM) [55] introduced a cosine-based noise scheduling strategy, which produces a smoother rate of noise change at both ends of the diffusion process ($t \to 0$ and $t \to T$). Experimental results demonstrate that this approach significantly improves the log-likelihood performance of the model. Specifically, iDDPM defines $\bar{\alpha}_t$ as:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \; f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2,$$

where s is a hyper-parameter controlling the noise scale and T is the total number of diffusion steps. The corresponding forward-process variance β_t can be recursively computed as $\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}$. In addition, iDDPM proposes a log-space interpolation method for learning the reverse process variance:

$$\Sigma_\theta(\mathbf{x}_t, t) = \exp(v \log \beta_t + (1-v) \log \sigma_q^2(t)),$$

where v is a learned weight vector output by the diffusion model (with the same dimension as β_t), and $\sigma_q^2(t)$ is defined in (46). This design preserves the stability of the original DDPM variance β_t, while introducing data-dependency via $\sigma_q^2(t)$, enabling the model to adaptively adjust the denoising strength for different time steps.

Beyond the variance learning approach in iDDPM, some models can explicitly predict the variance directly. A representative model in this direction is the Diffusion Transformer (DiT) [56]. DiT leverages a Transformer-based architecture to combine the diffusion process with the global modeling capabilities of the Transformer, making it highly effective for high-resolution image generation tasks. Unlike traditional models that only predict noise or mean, DiT is capable of predicting both the noise and the variance in the diffusion process for the noised input. By jointly predicting mean and variance, DiT improves both the quality and diversity of generated images, while also enhancing sampling stability and the preservation of fine details.

In the area of reverse-process variance optimization, Analytic-DPM [57] derives, via rigorous mathematical analysis based on variational principles, an analytic expression for the optimal reverse variance:

$$\Sigma_\theta(\mathbf{x}_t, t) = \sigma_q^2(t) + \left(\sqrt{\frac{\bar{\beta}_t}{\alpha_t}} - \sqrt{\bar{\beta}_{t-1} - \sigma_q^2(t)}\right)^2 \cdot \left(1 - \bar{\beta}_t \mathbb{E}_{q(\mathbf{x}_t)} \frac{\|\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)\|^2}{d}\right)$$

where $\sigma_q^2(t)$ is defined in (46), $\alpha_t = 1 - \beta_t$, $\bar{\beta}_t = 1 - \bar{\alpha}_t$ (the specific definitions of β_t and $\bar{\alpha}_t$ are provided in Appendix 8.1), $q(\mathbf{x}_t)$ is the marginal distribution of $\mathbf{x}_t$ under the forward process, and d is the data dimension. The major value of this work is that, given a pretrained score model, the optimal reverse variance can be determined via Monte Carlo estimation, leading to a significantly tighter variational bound when used in likelihood evaluation.

Kingma et al. [58] proposed the Variational Diffusion Model (VDM), generalizing the traditional diffusion model framework through a more flexible parameterization. Specifically, VDM expresses the marginal distribution of the forward process as $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \alpha_t \mathbf{x}_0, \sigma_t^2 \mathbf{I})$, where α_t and σ_t^2 independently control signal attenuation and noise injection, respectively. This decoupled design reveals the intrinsic connection between DDPM and NCSN: when $\alpha_t = \sqrt{1-\sigma_t^2}$, VDM reduces to the standard DDPM, characterized by monotonically decreasing signal-to-noise ratio (SNR) α_t^2/σ_t^2; when $\alpha_t = 1$, VDM recovers the continuous-time formulation of NCSN. Furthermore, VDM proposes the following parameterization of σ_t^2 to enable differentiable noise schedules:

$$\sigma_t^2 = \text{sigmoid}(\gamma_\eta(t)),$$

where $\gamma_\eta(t)$ is a monotonic neural network with parameters η, which ensures the monotonic decrease of SNR over time and provides a differentiable parameter space for learning more complex noise schedules.

4.3 Efficient Sampling Techniques

As previously discussed, the reverse process of diffusion models can be formulated either as a reverse-time stochastic differential equation (SDE) or as a probability flow ordinary differential equation (ODE). In order to enhance sampling efficiency, researchers have developed a variety of efficient numerical solvers. These methods can be broadly categorized into three groups: solvers based on reverse-time SDEs, solvers based on probability flow ODEs, and improved solvers for probability flow ODEs utilizing higher-order gradient information. In the following, we will systematically introduce representative works and their innovations in each of these three categories.

4.3.1 Sampling via Reverse-Time SDEs

As described in Section 3.3, the reverse processes of DDPM and SGM can both be regarded as discretized implementations of reverse-time SDEs. Song et al. [9] pioneered the Annealed Langevin Dynamics (ALD) approach, which coordinates noise attenuation and sampling through a double-loop mechanism: the outer loop progressively decreases the noise scale $\sigma_1 > \sigma_2 > \cdots > \sigma_L > 0$, while the inner loop performs Langevin dynamics sampling at each noise level. This hierarchical design allows effective exploration of global structures during early sampling stages, followed by fine adjustment of local details in later stages. Jolicoeur-Martineau et al. [59] pointed out an inherent inconsistency in ALD, whereby the actual noise scale of samples in the i-th outer iteration exceeds the theoretically expected σ_i^2, preventing full convergence of the Langevin dynamics and causing noise to accumulate. To address this, they proposed the Consistent Annealed Sampling (CAS) method, which dynamically matches the variance of the injected noise at each step to the target value σ_i^2 through adaptive scaling of the score function and noise term, thereby enforcing consistency and preventing error propagation.

Song et al. [10] proposed an alternative discretization scheme for the reverse-time SDE that synchronizes with the forward process: for each

discretization step in the forward-time SDE, there exists a corresponding discretization step in the reverse-time SDE. They showed that this discretization effectively serves as a numerical SDE solver for Equation (16). Building on this, Jolicoeur-Martineau et al. [44] developed an adaptive-step SDE solver, which runs high-order and low-order solvers in parallel and uses the similarity between their outputs as a local error estimate to dynamically adjust the step size—adopting larger steps in smooth regions to accelerate computation, and smaller steps in more complex regions to ensure accuracy.

Inspired by physical dynamics, the Critically-Damped Langevin Diffusion (CLD) method [60] innovatively introduces auxiliary velocity variables to construct an augmented state space. The score function in this method is designed to possess smoother perturbation characteristics and a reduced learning burden. Together with a custom-designed SDE integrator, it achieves notable improvements in both sampling speed and quality. In addition, Song et al. [10] proposed the predictor-corrector framework, which combines a numerical SDE solver (predictor) with a Markov Chain Monte Carlo (MCMC) method (corrector). At each time step, the predictor first generates a noisy sample via numerical SDE integration, and the corrector then applies MCMC updates to refine the sample distribution, ensuring consistency between the sample marginals and those of the reverse SDE solution at all time steps.

4.3.2 Sampling via Probability Flow ODEs

Compared to the highly stochastic nature of SDE-based inference, the deterministic ODE-based inference process offers faster convergence due to the absence of random perturbations, albeit at the potential cost of reduced generative diversity. The Denoising Diffusion Implicit Model (DDIM) [45] is one of the earliest works aimed at accelerating the sampling procedure of diffusion models. As an improvement upon DDPM, the core idea of DDIM is to speed up generation through a non-Markovian diffusion process while retaining the same objective function as DDPM. Specifically, DDIM defines the following non-Markovian forward process:

$$q(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T|\mathbf{x}_0) = q(\mathbf{x}_T|\mathbf{x}_0) \prod_{t=2}^{T} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0),$$

where $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is a Gaussian distribution whose mean depends on both $\mathbf{x}_t$ and $\mathbf{x}_0$, thus characterizing its non-Markovian property. To ensure that the marginal distributions of the forward process match those of DDPM, i.e., $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbf{I})$ and $q(\mathbf{x}_{t-1}|\mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_{t-1}}\mathbf{x}_0, (1-\alpha_{t-1})\mathbf{I})$, DDIM derives the explicit form of $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ as follows:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2\mathbf{I}\right).$$

Both DDIM and DDPM share the same training objective, namely, to train the model $\epsilon_\theta(\mathbf{x}_t, t)$ to predict the noise in $\mathbf{x}_t$ at each time step t. During sampling, DDIM utilizes $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ in the reverse process. First, an estimate of $\mathbf{x}_0$ is obtained from the trained $\epsilon_\theta(\mathbf{x}_t, t)$:

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon \;\Rightarrow\; \hat{\mathbf{x}}_0 = \frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}},$$

and this estimate $\hat{\mathbf{x}}_0$ is substituted into the non-Markovian $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. When setting $\sigma_t = 0$, the update becomes:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\hat{\mathbf{x}}_0 + \sqrt{1-\alpha_{t-1}}\epsilon_\theta(\mathbf{x}_t, t).$$

It is noteworthy that the reverse process is now entirely determined by $\mathbf{x}_t$ and ϵ_θ, with no random noise injection. Essentially, the DDIM reverse process corresponds to a particular discretization of the probability flow ODE, where the absence of noise leads to slower error accumulation, allowing for larger step sizes and thus, reduced sampling steps. Furthermore, owing to its non-Markovian design, Song et al. [45] demonstrated that, under certain conditions, the generative distribution at selected steps of the reverse process well approximates the distribution of the full process, enabling intermediate steps to be skipped during sampling and further decreasing the number of diffusion steps.

Subsequent work, the Generalized Denoising Diffusion Implicit Model (gDDIM) [61], extends DDIM to a broader class of diffusion models, such as the Blurring Diffusion Model (BDM) [62] and Critically-Damped Langevin Diffusion (CLD) [60], by parameterizing the score network accordingly. PNDM [63] interprets the reverse process of diffusion models as solving an ODE on a specific manifold in $\mathbb{R}^N$, and proposes a pseudo-numerical solver that confines sampling to high-density manifolds of the data distribution, thereby mitigating the degradation of generation quality due to deviation from the manifold as observed in conventional numerical solvers. Their theoretical analysis further reveals that DDIM is a special case of PNDM when restricted to Euclidean spaces.

4.3.3 Sampling via High-Order ODE Solvers

As shown in Equation (17), the probability flow ODE possesses a semi-linear structure. Leveraging this property, recent research has developed customized high-order solvers. Karras et al. [28] introduced the second-order Heun method [65], which incorporates an additional correction step to reduce local truncation errors, thereby achieving an effective balance between sample quality and sampling speed. Furthermore, they innovatively incorporated a stochastic Langevin diffusion term—consisting of both the score function and random noise—into the ODE solver, which serves to project samples back onto the data manifold and further balance the trade-off between quality and efficiency.

Both the Diffusion Exponential Integrator Sampler [66] and DPM-solver [46] capitalize on the linear-nonlinear decomposition of the ODE for the design of specialized solvers. In contrast to general-purpose Runge-Kutta methods, these solvers analytically compute the linear component and handle the nonlinear term through exponential integrator techniques, resulting in significantly higher computational efficiency than conventional methods. Experimental results demonstrate that such customized solvers require only 10~20 iterations to generate high-quality samples, whereas traditional diffusion models typically require hundreds of iterations to achieve similar results—thus realizing an order-of-magnitude improvement in sampling speed.

Some recent works further enhance sampling accuracy by leveraging learnable parameters to capture higher-order gradient informa-

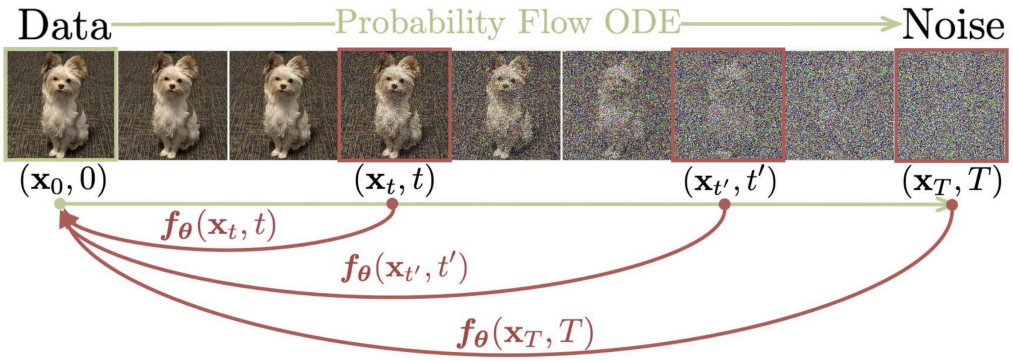

Fig. 2 Core Idea of Consistency Models (figure adapted from [64]). Given a probability flow (PF) ODE that smoothly transforms data into noise, consistency models are designed to learn mappings from any point on the ODE trajectory (e.g., $\mathbf{x}_t$, $\mathbf{x}_{t'}$, $\mathbf{x}_T$) back to its origin (e.g., $\mathbf{x}_0$) for the purpose of generative modeling.

tion during numerical integration, thus mitigating cumulative errors inherent in lower-order methods. GENIE [67] is a novel high-order denoising diffusion solver whose core idea is to approximate the generative trajectory more accurately by incorporating higher-order gradient information (such as second-order score functions), thereby significantly improving the sampling efficiency of diffusion models. GENIE achieves this by augmenting the first-order score network with an additional small network head, which is trained to distill higher-order gradients—computed via automatic differentiation—into this head during training. At inference, this head can efficiently provide high-order gradient information without the computational overhead of explicit high-dimensional derivatives, thereby substantially improving sampling accuracy while retaining computational efficiency.

Shaul et al. [68] proposed Bespoke Solvers, differentiable parameterized ODE solvers tailored for pre-trained diffusion models. Their approach generalizes the coefficients in traditional numerical solvers into model-dependent dynamic parameters, which are optimized through a global error objective based on multi-step consistency constraints. Experiments show that the solver's parameters can dynamically adapt to the evolving ODE dynamics at different sampling stages, paving new pathways for rapid sampling in generative models.

4.3.4 Edge-side applicability of high-order ODE solvers

While cloud environments typically favor higher-order ODE solvers (e.g., DPM-Solver or PNDM) to minimize the total number of sampling steps (T), their applicability on edge devices is fundamentally limited. In edge settings, the dominant constraint is not merely the overall step count, but the severe limitations on per-step memory bandwidth and computational cost. Higher-order solvers mathematically require multiple Neural Function Evaluations (NFE > 1) per step. On mobile hardware, each additional NFE forces the NPU/GPU to repeatedly reload massive weight tensors into a strictly limited SRAM. This redundant fetching exacerbates memory thrashing, cache misses, and rapid thermal throttling. Consequently, for resource-constrained edge deployments—especially when utilizing heavily step-distilled architectures like LCMs—first-order solvers (e.g., Euler) are overwhelmingly preferred. Although first-order solvers might theoretically require more steps to converge, maintaining a strictly bounded per-step computation (NFE=1) drastically alleviates memory bandwidth pressure, ultimately yielding the lowest end-to-end latency and optimal thermal stability on mobile devices.

4.4 Consistency Model Techniques

Traditional diffusion models suffer from high inference latency due to their requirement for multi-step iterative denoising, which limits their practical deployment in real-world applications. To overcome this bottleneck, recent research has proposed various methods that constrain the geometric properties of diffusion trajectories, enabling efficient data generation with fewer steps, or even in a single step. Among these, the Consistency Model (CM) [70] and the Latent Consistency Model (LCM) [71] represent important advances in this area.

The Consistency Model (CM) [70] is a novel generative model capable of achieving one-step or few-step data generation. As shown in Figure 2, its core idea is to learn a consistency function that maps any point along a Probability Flow Ordinary Differential Equation (PF-ODE) trajectory to the starting point of that trajectory (i.e., to the solution of the PF-ODE, which corresponds to the original data). The consistency function is defined as $\mathbf{f} : (\mathbf{x}_t, t) \rightarrow \mathbf{x}_\epsilon$, where ϵ is a small positive constant; for numerical stability, solvers typically stop at $t = \epsilon$, and $\mathbf{x}_\epsilon$ is a datapoint very close to the genuine data sample. The consistency function must satisfy the property of self-consistency:

$$\mathbf{f}(\mathbf{x}_t, t) = \mathbf{f}(\mathbf{x}_{t'}, t'), \ \forall t, t' \in [\epsilon, T].$$

Song et al. [70] showed that an effective parameterized consistency model f_θ can be learned from data by optimizing an objective function that enforces the above property. To ensure that the model satisfies the boundary condition $f_\theta(\mathbf{x}, \epsilon) = \mathbf{x}$, f_θ is parameterized as:

$$\mathbf{f}_\theta(\mathbf{x}, t) = c_{\text{skip}}(t)\mathbf{x} + c_{\text{out}}(t)\mathbf{F}_\theta(\mathbf{x}, t),$$

where $c_{\text{skip}}(t)$ and $c_{\text{out}}(t)$ are differentiable functions satisfying $c_{\text{skip}}(\epsilon) = 1$ and $c_{\text{out}}(\epsilon) = 0$, and $\mathbf{F}_\theta(\mathbf{x}, t)$ is a deep neural network. The consistency model can be trained in two ways: by distillation from a pretrained diffusion model (referred to as consistency distillation), or by training from scratch. To further strengthen the self-consistency property, Song et al. employed an exponential moving average update for the target model θ^-, that is, $\theta^- \leftarrow +\mu\theta^- + (1 - \mu)\theta$, and defined

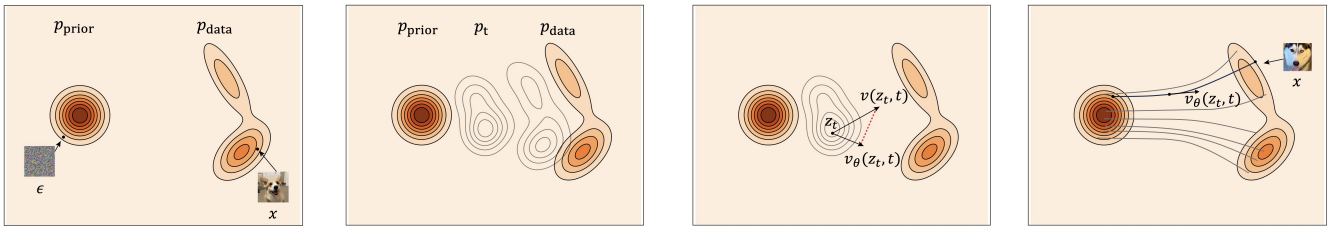

Fig. 3 Overview of flow matching (figure adapted from [69]). (a) The objective is to identify a flow that maps samples ϵ drawn from a prior distribution p_{prior} to samples x from the data distribution p_{data}. (b) To achieve this, a time-continuous probability path $(p_t)_{0\le t\le 1}$ is constructed to interpolate between the source distribution p_{prior} and the target distribution p_{data}. (c) During training, regression techniques are employed to estimate the velocity field v to generate p_t. (d) To generate new samples $x \sim p_{\text{data}}$, the estimated velocity field $v_\theta(z_t, t)$ is integrated from $t = 0$ to $t = 1$.

the consistency loss as follows:

$$\mathcal{L}(\theta, \theta^-; \Phi) = \mathbb{E}_{\mathbf{x},t}\left[d\left(\mathbf{f}_\theta(\mathbf{x}_{t_{n+1}}, t_{n+1}), \mathbf{f}_{\theta^-}(\hat{\mathbf{x}}^\phi_{t_n}, t_n)\right)\right],$$

where $d(\cdot,\cdot)$ denotes a similarity metric between two samples, and $\hat{\mathbf{x}}^\phi_{t_n}$ is an estimate of $\mathbf{x}_{t_n}$ computed from $\mathbf{x}_{t_{n+1}}$:

$$\hat{\mathbf{x}}^\phi_{t_n} \leftarrow \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \phi),$$

with Φ denoting a single-step ODE solver for the PF-ODE, such as the Euler solver [10] or Heun solver [28]. Empirical results demonstrate that CM achieves high-quality sampling with one or very few steps, substantially improving generation efficiency and providing substantial support for the practical deployment of diffusion models.

Building on the foundation of the Consistency Model, the Latent Consistency Model (LCM) [71] introduces a key improvement by transferring the self-consistency constraint into the latent space of a pretrained diffusion model (such as the VAE latent space in Stable Diffusion). Compared to consistency modeling in the pixel space, the significantly reduced dimensionality in the latent space not only greatly improves computational efficiency, but also facilitates the direct use of pretrained text encoders (such as the CLIP encoder), making LCM naturally suited for multi-modal generation tasks such as text-to-image synthesis. To further enhance efficiency in guided generation, the authors propose a guided distillation technique that incorporates classifier-free guidance (CFG) into training by solving the probability flow ODE, while also introducing a skip-step strategy for faster convergence. Experimental results show that this method can compress the number of sampling steps to just 1-4, while preserving image quality and diversity, substantially promoting the practical industrial deployment of conditional diffusion models.

Failure Modes under Edge Constraints: As illustrated in Figure 2, Consistency Models attempt to map any point on the trajectory directly to the origin. However, under extreme edge-side constraints where networks are heavily compressed (e.g., fewer parameters) and inference is restricted to a single step, this mapping exhibits specific failure modes. The highly non-linear nature of the true probability flow is difficult to approximate perfectly with a low-capacity mobile network in one shot. Consequently, edge-deployed one-step models often suffer from noticeable blurriness, loss of high-frequency textures, and diminished fine-grained details compared to their multi-step counterparts.

4.5 Flow Matching Methods

The objective of generative models is to transform a prior distribution into a data distribution. Flow Matching (FM) approaches [69, 72, 74] provide an intuitive and conceptually simple framework for constructing flow paths that transport one distribution to another, with a particular focus on the velocity field used to guide model training. Since their introduction, flow matching methods have been widely adopted in modern generative modeling [75–77]. Figure 3 presents the key elements of the flow matching methods. Specifically, given data samples $x \sim p_{\text{data}}(x)$ and prior noise $\epsilon \sim p_{\text{prior}}(x)$, a flow path can be constructed as follows:

$$z_t = a_t x + b_t \epsilon,$$

where t denotes time, and a_t, b_t are predefined schedulers, typically chosen as $a_t = 1 - t$ and $b_t = t$. From this, the velocity v_t can be defined as:

$$v_t = \frac{d}{dt} z_t = \frac{d}{dt} a_t x + \frac{d}{dt} b_t \epsilon.$$

Since the above velocity depends on both x and ϵ, it is also referred to as the *conditional velocity* [69], denoted as $v_t = v_t(z_t|x)$. Flow matching approaches model the expectation of conditional velocities over all possibilities to obtain the marginal velocity:

$$v(z_t, t) \triangleq \mathbb{E}_{p_t(v_t|z_t)}[v_t]. \tag{28}$$

The training objective of flow matching methods is to optimize a neural network parameterized by θ, denoted v_θ, to learn the marginal velocity field via the loss $\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,p_t(z_t)}\|v_\theta(z_t, t) - v(z_t, t)\|^2$. Although this objective is challenging to compute directly due to the expectation in (28), Lipman et al. [69] show that minimizing the conditional Flow Matching loss $\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,p_t(z_t)}\left\{\|v_\theta(z_t, t) - v(z_t|x)\|^2\right\}$ is equivalent to minimizing $\mathcal{L}_{\text{FM}}$.

Once the marginal velocity field $v(z_t, t)$ is obtained, novel samples can be generated by solving the following ODE:

$$\frac{d}{dt} z_t = v(z_t, t),$$

where the initial point can be set as $z_1 = \epsilon \sim p_{\text{prior}}$. The solution to this ODE can be written as $z_r = z_t - \int_r^t v(z_\tau, \tau)d\tau$, where r denotes another time step. In practice, this integral is typically approximated by numerical integration over discrete time steps.

Table 2 Summary of Subsequent Improvements in Diffusion Models

Improvement Direction	Representative Methods	Core Concepts
Latent Diffusion Models	Latent Score-Based Generative Model [54]	Jointly trains a VAE encoder and a diffusion model
	Latent Diffusion Model [3]	Trains a VAE encoder first and then constructs a diffusion model
Advances in Likelihood Estimation	Noise Schedule Optimization [55]	Designs adaptive noise schedules for smoother denoising
	Reverse Variance Optimization [55–57]	Accurately models reverse variance using analytical formulas or model predictions
	General Parametrization Framework [58]	Independently parameterizes signal attenuation and noise injection
Efficient Sampling Algorithms	Based on Reverse SDE [9, 10, 44, 59, 60]	Uses discrete reverse SDE to generate data from noise
	Based on Probability Flow ODE [45, 61, 63]	Constructs efficient generative paths by solving deterministic ODEs
	Based on High-Order ODE [28, 46, 66–68]	Approximates generative trajectories more accurately by modeling higher-order information
Consistency-Based Methods	Consistency Model [70]	Directly maps any point on the diffusion trajectory to a data point
	Latent Consistency Model [71]	Transfers the consistency model to a low-dimensional latent space
Flow Matching Methods	Rectified Flow [72]	Linearizes the transport path between data and noise
	Mean Flow [73]	Characterizes the transformation of the field by average velocity over a time interval

Rectified Flow (RF) [72] represents an important recent advancement in flow matching methods, aiming to further improve training and inference efficiency. Inspired by ODEs and Optimal Transport (OT) theory, the central idea of RF is to construct straighter transport paths between the prior and data distributions. This design allows for larger step sizes during generation without divergence, thereby effectively reducing the number of sampling iterations required. Specifically, RF defines the forward process as a straight-line interpolation between the data distribution and a standard normal distribution:

$$\mathbf{x}_t = (1 - \sigma_t)\mathbf{x}_0 + \sigma_t \epsilon,$$

where $\mathbf{x}_0$ represents a clean image, t denotes the time step, σ_t is a coefficient depending on t, and ϵ is random noise sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The reverse process in RF is learned by a neural network that models the **velocity field**, $v_\theta(\mathbf{x}_t, t)$:

$$\mathcal{L}_{\text{task}} = \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t} \left[\| (\epsilon - \mathbf{x}_0) - v_\theta(\mathbf{x}_t, t) \|_2^2 \right].$$

As can be seen, the velocity field essentially describes the direction of optimal transport from the noise ϵ to the image $\mathbf{x}_0$. To enhance training stability, Liu et al. [72] propose using a logit-normal sampling strategy for t, which samples middle steps with higher probability. During inference, RF employs the Flow-Euler sampler, which updates the next sample based on the learned velocity field:

$$\mathbf{x}_{t-1} = \mathbf{x}_t + (\sigma_{t-1} - \sigma_t) \cdot v_\theta(\mathbf{x}_t, t),$$

where $\sigma_{t-1} - \sigma_t$ can be interpreted as the step size. By explicitly enforcing linearized trajectories and modeling the velocity field instead of the traditional diffusion process, RF retains the progressive nature of sample generation in diffusion models while eliminating the reliance of conventional diffusion ODEs on the score function, thus enabling efficient sampling. High-quality samples can generally be obtained within only 10–20 steps.

In recent work aimed at further enhancing efficiency and quality in flow matching models, various innovative approaches have been proposed. Flow Map Matching [78] directly models the flow mapping between arbitrary time steps, bypassing the need for numerical integration in the generation process. Here, the flow map is defined as the integral of the velocity field between two time steps (i.e., the net displacement), and researchers have devised multiple matching losses to facilitate effective learning. Building on this concept, Shortcut Models [79] introduce a self-consistency loss within the flow matching framework, explicitly modeling the relationships among flow maps of different step sizes. This enables the model to perform efficient and consistent mappings from noise to data with arbitrary step sizes,

supporting high-quality generation in one or a few steps. Inductive Moment Matching [80] further extends this idea by generalizing self-consistency modeling to multiple different time steps, specifically targeting temporal consistency for stochastic interpolators.

Distinct from previous flow matching techniques that focus on modeling pointwise instantaneous velocities along the path, the Mean Flow method [73] captures the overall transformation of the field by considering the average velocity over an interval. This work not only derives explicit relationships between average and instantaneous velocities, but also leverages this insight for neural network training. A significant advantage of the mean flow approach is that, if the average velocity can be accurately modeled, the entire flow path can be effectively approximated via a single forward inference, greatly improving generative efficiency. Collectively, these methods open new avenues toward efficient and controllable generation with diffusion models.

4.6 Taxonomy of Optimizations: Training vs. Inference

To provide a comprehensive perspective on the rapid development of diffusion models, it is essential to establish a clear distinction between techniques that optimize the training phase and those that focus on the inference stage. While the ultimate goal of both directions is to enhance generative quality, stability, and speed, they intervene at fundamentally different stages of the model's lifecycle. Training-side optimizations generally involve modifying the data space, model architecture, loss functions, or learning paradigms, fundamentally altering the "knowledge" and generative pathways the model acquires. In contrast, inference-side optimizations typically treat the pre-trained model as a fixed entity, focusing purely on algorithmic and numerical improvements to navigate the established generative pathways more efficiently without requiring any network retraining.

Among the training-side optimizations, a primary direction is the reconstruction of the training space and the parameterization of the network. For instance, Latent Diffusion Models (such as LDM [3] and LSGM [54]) fundamentally shift the diffusion process from the high-dimensional pixel space to a low-dimensional semantically compressed latent space, drastically reducing the computational burden required during training. Similarly, works focusing on advances in likelihood estimation—such as iDDPM [55], DiT [56], and VDM [58]—optimize the noise schedules and explicitly learn or parameterize the variance during the training phase. These modifications tighten the variational lower bound and enhance the inherent data modeling capacity of the network, ensuring that the model is intrinsically better conditioned before any sampling occurs.

Another vital class of training-side optimizations focuses on linearizing or constraining the generative trajectories to explicitly facilitate rapid sampling later on. Flow Matching methods [69, 74], particularly Rectified Flow [72], rethink the training objective by replacing the standard score-matching loss with velocity field regression, forcing the model to learn straightened transport paths that naturally permit larger step sizes. Similarly, Consistency Models (CM) [70] and Latent Consistency Models (LCM) [71] introduce profound training-side innovations, often realized via distillation or custom loss functions. By enforcing self-consistency across the ODE trajectory directly within the training objective, these models restructure their weights to map any noisy state directly to the data manifold, explicitly optimizing for one- or few-step generation from the ground up.

Conversely, inference-side optimizations seek to accelerate the sampling process or improve generation quality by exclusively refining the numerical solvers, leaving the pre-trained model parameters untouched. The transition from standard Markovian processes to non-Markovian samplers like DDIM [45], as well as the development of probability flow ODE solvers like PNDM [63], DPM-Solver [46], and Exponential Integrator Samplers [66], are classic examples of this paradigm. By analytically computing linear components or skipping redundant integration steps, these deterministic methods dynamically accelerate generation. Furthermore, SDE-based refinements, such as the predictor-corrector framework [10] and Consistent Annealed Sampling (CAS) [59], optimize the stochastic inference process by carefully coordinating noise injection and denoising steps at test time, enhancing sample diversity and manifold fidelity without incurring any additional training costs.

Finally, the boundary between training and inference optimizations is increasingly blurring, giving rise to hybrid approaches that codesign both stages. For example, high-order ODE solvers like GENIE [67] and Bespoke Solvers [68] introduce lightweight, learnable components or dynamic parameters tailored specifically to the inference solver. While their primary goal is to optimize the inference process by minimizing truncation errors, they require a dedicated, albeit brief, training or distillation phase to adapt the solver to the pre-trained model's specific dynamics. Ultimately, the most profound recent leaps in diffusion models demonstrate a clear trend: modern training-side techniques (like Flow Matching and Consistency Models) are increasingly and explicitly designed to serve the ultimate goal of ultra-fast inference, reflecting a holistic optimization philosophy that unifies both stages.

4.7 Bridging Theoretical Foundations with Edge Constraints

Despite the robust generative capabilities of DDPM, SGM, and Score SDE, their direct deployment on edge devices is impeded by a fundamental conflict between the rigorous mathematical formulation and the stringent hardware constraints (e.g., latency, energy, and memory). The standard diffusion process is computationally expensive, and understanding its theoretical underpinnings is prerequisite for developing efficient **Edge AIGC**. Specifically, the mathematical bottlenecks of diffusion models can be decomposed into three dimensions, which define the current research roadmap for edge optimization:

First, regarding the **temporal dimension**, the reverse diffusion process is mathematically defined as a sequential integration over a long trajectory $t \in [T, 0]$, typically requiring hundreds of iterative steps. This sequential nature leads to prohibitive inference latency on edge processors. To address this, mathematical approximations that treat the reverse process as an Ordinary Differential Equation (ODE) allow for larger step sizes. Techniques such as DDIM [45] and DPM-Solver [46] significantly accelerate sampling by utilizing high-order solvers, while Consistency Models [64] and Progressive Distillation [81] aim to dis-

till the entire trajectory into a single function evaluation, enabling real-time generation on mobile devices.

Second, regarding the **model complexity**, the score function estimator $\epsilon_\theta(\mathbf{x}_t, t)$ is parameterized by deep neural networks with high arithmetic intensity, which challenges the limited memory bandwidth and power budget of edge hardware. Recent works leverage the mathematical property that diffusion models are inherently robust to noise to perform aggressive compression. Methods like Q-Diffusion [82] and PTQ4DM [83] demonstrate that the weights and activations can be quantized to low-bit formats (e.g., INT4) without collapsing the generative distribution. Furthermore, architecture-level optimizations, such as MobileDiffusion [29] and SnapFusion [21], restructure the network topology θ to align with the cache hierarchy of mobile NPUs, reducing the computational cost of each step.

Third, regarding the **spatial dimension**, the standard diffusion formulation operates in the high-dimensional pixel space $\mathbf{x} \in \mathbb{R}^d$, where computational cost scales quadratically with resolution. This is often infeasible for high-resolution generation on battery-powered devices. To mitigate this, Latent Diffusion Models (LDM) [3] shift the mathematical diffusion process into a compressed low-dimensional latent space. By decoupling the generative modeling from pixel-level rendering, LDM and its lightweight variants (e.g., Tiny-Fusion [84]) dramatically reduce the dimensionality d of the stochastic differential equations, making high-quality synthesis accessible on edge platforms.

In summary, the specific variables—time steps T, network parameters θ, and data dimensionality d—introduced in the following mathematical formulations serve as the primary "knobs" for optimizing edge performance. The subsequent review of the forward and reverse mechanisms provides the theoretical blueprint necessary to understand how these variables interact and how they can be manipulated to achieve efficient, low-resource generative intelligence.

■ 5 Text-to-image Diffusion Models

Previously, we discussed the mathematical foundations of diffusion models as well as methods for optimizing training and inference. In this section, we focus on three additional key components of text-to-image generation systems: model architecture, training data, and evaluation metrics. Using representative models such as Stable Diffusion as examples, we analyze how architectural innovations enable efficient multimodal integration. We also examine how strategies for constructing training datasets affect the quality of generated outputs, and review the strengths and limitations of current evaluation methods. These practical insights are crucial for the development of robust and usable generative systems.

5.1 Model Architecture

Architectural design has played a crucial role in the advancement of text-to-image diffusion models. Early works [8, 9] predominantly relied on the UNet architecture [85]. Centered around convolutional operations, UNet implements a hierarchical workflow comprising successive downsampling and upsampling phases, thus enabling comprehensive feature extraction and image reconstruction. Specifically, the downsampling path progressively reduces spatial resolution, extracting high-level semantic information, while the upsampling path fuses low-level spatial details with high-level features via skip connections, effectively restoring local structures and image textures. The intermediate, or bottleneck, stage acts as a bridge between downsampling and upsampling, often consisting of several processing units responsible for deep feature transformation and global information integration. Although UNet excels in multi-scale denoising tasks, its convolution-based structure inherently limits its receptive field and capacity to model long-range dependencies. Furthermore, UNet's ability to incorporate external conditional information (e.g., text prompts) is somewhat restricted, which can undermine its effectiveness in text-to-image generation tasks.

To better integrate external conditioning signals, researchers have proposed various innovative approaches. For instance, GLIDE [86] enhances the UNet framework through a dual mechanism for text incorporation: first, a Transformer encoder processes text prompts, feeding the resulting global embedding vector into the model as a conditioning input; second, the intermediate text token features are projected into each attention layer of the UNet, facilitating cross-modal interaction between image and text features at different semantic levels. Imagen [48] employs a pretrained text encoder (e.g., T5 [87]) to generate semantic embeddings and introduces cross-attention layers into the UNet for dynamic alignment between text and image features. Additionally, global text vectors, obtained via pooling, are injected into every network layer to ensure semantic consistency. In terms of generative strategy, both GLIDE and Imagen utilize a hierarchical paradigm: a text-conditional diffusion model first generates a low-resolution image, which is subsequently refined via super-resolution modules. DALL·E-2 [88] innovatively leverages a CLIP text encoder to first generate semantic representations, followed by hierarchical diffusion models to synthesize high-resolution images. While these approaches have greatly improved text-image alignment, their cascaded design or direct high-resolution generation typically incurs substantial computational overhead.

Latent Diffusion Models (LDMs) [3] address these challenges by offering a novel solution that boosts computational efficiency and enhances the modeling of long-range dependencies and external conditions. Building upon the core UNet design with hierarchical sampling and skip connections, LDMs introduce both self-attention and cross-attention mechanisms: the former captures global dependencies within the image, while the latter enables effective conditioning on external textual information. Moreover, by incorporating timestep embeddings into residual blocks, LDMs better model the dynamics of the diffusion process. This conditional information injection paradigm has been widely adopted in subsequent works and forms the basis for the Stable Diffusion model family—a structure referred to henceforth as the Conditional UNet. With increasing model capacity and continual refinement of training strategies (e.g., SDXL [89]), the performance of this architecture has been significantly advanced, making it a mainstream solution in both academia and industry.

In recent years, the rise of Vision Transformers (ViTs) [90] has stim-

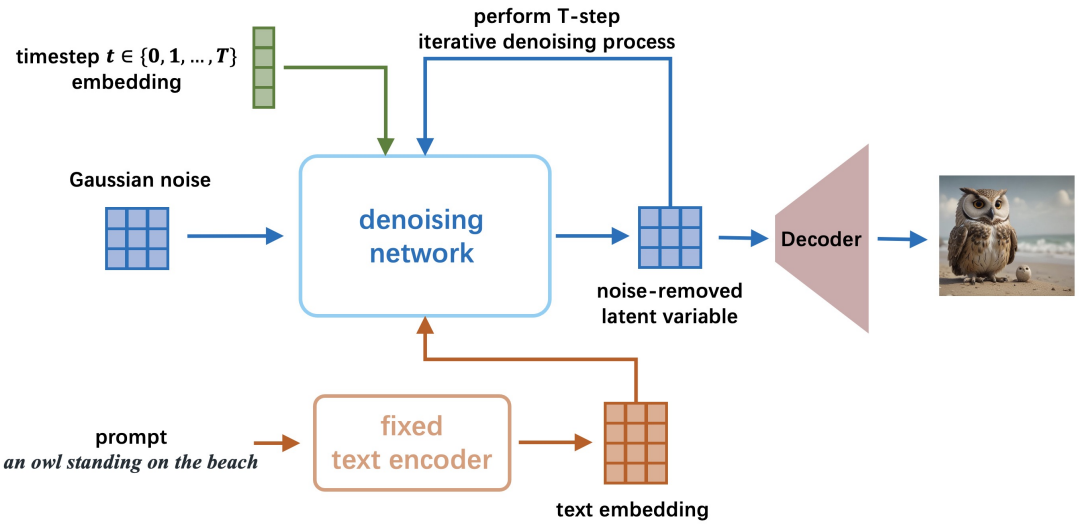

Fig. 4 The inference process of the Stable Diffusion Model (SDM).

ulated explorations of more efficient and streamlined architectures. For example, UViT [91] integrates Transformer modules into the bottleneck of the Conditional UNet, replacing conventional convolutional layers and substantially enhancing the model's global feature modeling capabilities. Although self-attention theoretically incurs high computational complexity, in practice, Transformer and MLP modules can fully leverage the parallelism of modern GPUs/TPUs, thereby maintaining practical efficiency. Owing to its architectural similarity to the Conditional UNet, UViT is treated as a variant thereof and will be discussed jointly in the remainder.

Pushing diffusion model architectures further, the Diffusion Transformer (DiT) [56] abandons convolution entirely, instead processing its input as sequences of non-overlapping patches. This patch-based representation preserves the powerful sequence modeling capacity of Transformers while alleviating the computational burden of high-resolution inputs. The DiT block is fundamentally a standard Transformer block (multi-head self-attention plus feed-forward network), but innovatively incorporates Adaptive Layer Normalization with zero-initialization (AdaLN-Zero), allowing time-step and text prompt features to be used as conditioning signals for fine-grained control over the diffusion process. After encoding, the final token representations are decoded to produce noise and variance predictions, completing the generative process. Such architectural innovations have significantly improved visual fidelity and sampling efficiency, opening new avenues for future diffusion model development.

A notable application of the LDM framework is found in Stable Diffusion Models (SDMs) [3], among the most widely used open-source models for text-to-image generation. Owing to their superior generative abilities, SDMs are now adopted as core architectures for a variety of text-guided vision tasks [92–95]. In essence, SDMs are text-to-image optimized LDMs leveraging diffusion in a semantically compressed latent space [45, 53, 63] for improved computational efficiency. The model utilizes either a Conditional UNet or DiT architecture for iterative latent sampling/denoising, synergized with a text encoder [49] and image decoder [96, 97] to produce text-aligned images. The architecture and inference workflow of SDMs are shown in Figure 4.

We begin by detailing the text-to-image pipeline of SDMs and the core denoising architecture. As shown in Figure 4, SDM adopts a multi-stage process: a user prompt is encoded into a text embedding via a pretrained, frozen CLIP encoder; initial noise is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$; and a discrete timestep $t \in \{0, \ldots, T\}$ is embedded into a continuous vector. These inputs—text embedding, noise latent, and timestep embedding—are fed into a conditional denoiser (Conditional U-Net or DiT), which iteratively predicts noise and updates the latent over T steps, steering it toward the target semantics. The refined latent is finally decoded by a VAE to produce a high-resolution image aligned with the prompt.

Before presenting acceleration strategies, we analyze the computation, parameter distribution, and latency of each core module and step in SDMs. Building on Li et al. [21] and focusing on Stable Diffusion v1.5, we quantify layer-wise parameters and compute cost. These insights expose inefficiencies in text-to-image diffusion and inform subsequent optimization under resource constraints, particularly for mobile deployment.

As shown in Figure 4, SDM consists of three main modules: the text encoder, denoising network, and VAE decoder. Li et al. [21] quantitatively investigated the parameter count and latency (measured on an iPhone 14 Pro) of each module in the SD-v1.5 model. Specifically, SD-v1.5 employs a ViT-H model [49] as its text encoder. Since the diffusion pipeline operates with classifier-free guidance, according to Eq. (27), the text encoder is executed twice per image—once with the text prompt and once with an empty string, producing both conditional and unconditional embeddings. Despite this, the text encoder contributes only a minor portion of total inference latency (approximately 8 milliseconds) and has negligible impact on overall performance. In

contrast, the VAE decoder is responsible for reconstructing the final image from the compressed latent representation, incurring a latency of about 369 milliseconds—mainly due to the complexity of the decoding process. By far the most computationally expensive module, however, is the denoising network: its per-step latency reaches up to 1.7 seconds, and the generative process requires repeated forward passes. As a result, it is the dominant performance bottleneck in the entire inference pipeline.

To dissect the internal structure of the Conditional UNet, it is crucial to focus on the attention modules and residual blocks. The cross-attention module fuses textual information into image representations at each stage; this operation is given by:

$$\text{Cross-Attention}(Q_{\mathbf{z}_t}, K_{\mathbf{c}}, V_{\mathbf{c}}) = \text{Softmax}\left(\frac{Q_{\mathbf{z}_t} \cdot K_{\mathbf{c}}^{\top}}{\sqrt{d}}\right) \cdot V_{\mathbf{c}},$$

where $Q_{\mathbf{z}_t}$ is projected from the noise input $\mathbf{z}_t$, and $K_{\mathbf{c}}$ and $V_{\mathbf{c}}$ are projected from the textual condition $\mathbf{c}$, with d denoting feature dimensionality. Through cross-attention, the model achieves multi-scale text-image alignment. The residual blocks, in turn, are responsible for capturing and preserving local spatial features. The interplay between attention and residual modules thus enables the diffusion model to balance global semantic comprehension with high-fidelity detail reconstruction, supporting the realism and richness of generated images.

Li et al. [21] further compares layerwise latency and parameter distribution within the Conditional UNet. Results indicate that parameters are highly concentrated in the middle layers of the network, largely due to increased channel widths, with the residual blocks dominating the total parameter count. In terms of latency, the input and output stages are the most time-consuming, which stems from the high spatial resolution of features at these stages; the computational complexity of spatial cross-attention grows quadratically with feature size (i.e., the token count in ViT), leading to exponential computational overhead. As such, this inefficiency constitutes a critical target for further optimization of SDM inference performance.

To ground the latency breakdown illustrated in Figure 4, we refer to the concrete profiling methodology established by Li et al. [21]. The latency assertions presented are based on generating a standard 512×512 resolution image using 20 diffusion steps on an iPhone 14 Pro equipped with the A16 Bionic chip and its Neural Engine (NPU). Under these reproducible edge constraints, the structural bottleneck becomes evident. As shown in Figure 4, while the VAE decoding is a one-time operation, the UNet is executed repeatedly. The cross-attention mechanisms and ResNet blocks within the UNet demand massive memory bandwidth. On mobile devices with limited SRAM, this repetitive execution forces frequent read/write operations to the slower DRAM, accumulating to the severe 1.7 seconds-per-step latency bottleneck. This empirical evidence directly motivates the necessity for the architectural and step-reduction optimizations discussed in Section 6.

5.2 Training Data

The quality of data plays a crucial role in determining the ultimate performance of machine learning models. For generative model training, datasets with labels—such as CIFAR10 [98], ImageNet [99], and LSUN [100]—are of vital importance, particularly for unconditional or class-conditional generation tasks. These datasets, with their precise class annotations, provide structured learning objectives for diffusion models, enabling the generation of category-specific images. For example, ImageNet, with its 1,000 fine-grained classes, facilitates the learning of rich visual representations, while CIFAR10, owing to its smaller scale and clear class divisions, is often used for rapid model prototyping. Moreover, such datasets also serve as indispensable resources for training image encoders or super-resolution modules [3], functioning as essential components for pre- and post-processing in diffusion models. However, due to the lack of textual descriptions, their application in text-to-image tasks is limited; they are mainly suitable for class-conditional or foundational generative ability evaluation.

With the emergence of large-scale image-text paired datasets, such as LAION [101] and Conceptual Captions [102, 103], diffusion models have made notable advances in text-conditional image generation. The LAION dataset contains billions of web-scraped image-text pairs, with high-quality alignment ensured by CLIP-based filtering. Similarly, Conceptual Captions employs automated pipelines to refine caption quality, rendering the dataset both informative and learnable. These datasets encompass a wide range of visual concepts and linguistic expressions, enabling diffusion models to acquire complex cross-modal correspondences and achieve strong zero-shot generation performance. Although the MSCOCO dataset [104] is much smaller in scale, its diverse scene coverage and five high-quality captions per image make it a valuable resource for initial training and benchmarking of text-to-image models, especially in evaluating zero-shot generation capabilities.

Despite automated cleansing of web-scale image-text pair data, inherent noise in real-world datasets—such as semantic drift, annotation errors, and context fragmentation—remains a significant challenge, impeding further model improvement. In this context, the use of synthetic data for model training has attracted increasing research interest. EdgeFusion [23], for example, employs GPT-4 for prompt generation and SDXL [89] for image synthesis, significantly enhancing the controllability and diversity of training data. By systematically varying attributes such as gender, ethnicity, and age, this method effectively mitigates data bias and achieves broader, more inclusive coverage. Experimental results demonstrate that synthetic samples excel in both image quality and text-image semantic alignment, leading to substantial improvements in overall model performance.

5.3 Evaluation Metrics

With the rapid advancement of text-to-image generation, diffusion models have emerged as a key research focus due to their high-quality outputs and remarkable training stability. Against this backdrop, establishing objective and comprehensive performance evaluation metrics has become crucial for assessing technological effectiveness. This section focuses on evaluation methods for diffusion models, systematically introducing three core metrics—inception score (IS) [105], Fréchet inception distance (FID) [106], and CLIP score—while discussing their applicability and inherent limitations in different application scenarios.

Inception Score (IS) [105] is a classic metric for evaluating generative models, relying on a pretrained Inception-v3 network [107] to quantify both the quality and diversity of generated images. Its calculation proceeds as follows:

The computation process of Inception Score (IS):

1. For each generated image $\mathbf{x}$, the Inception-v3 model predicts its class distribution $p(y|\mathbf{x})$;
2. The marginal class distribution over all generated images is computed as $p(y) = \mathbb{E}_{\mathbf{x}}[p(y|\mathbf{x})]$;
3. The average KL divergence between $p(y|\mathbf{x})$ and $p(y)$ is calculated and exponentiated:

$$\text{IS} = \exp(\mathbb{E}_{\mathbf{x}}[\text{KL}(p(y|\mathbf{x})\|p(y))]).$$

Intuitively, if $p(y|\mathbf{x})$ is sharply peaked (i.e., the model assigns high confidence to a single class), it suggests that the image is clear and easily recognizable; if $p(y)$ is close to uniform, it indicates that generated images are diverse and cover the range of possible classes. Thus, IS measures the divergence between per-image class distributions and the overall class distribution: higher IS values indicate both high individual image quality and substantial diversity across the set.

The advantages of IS lie in its computational efficiency—it requires only generated images and a pretrained classifier, with no need for manual annotation—and its ability to capture both clarity and diversity. The scores are also directly comparable across models. Nevertheless, IS exhibits limitations: its reliability depends on the classifier's accuracy and the alignment between the generated and training label distributions; it is insensitive to mode collapse and class imbalance; and when generated data significantly differ from the classifier's original training classes (e.g., ImageNet), IS becomes unreliable and may fail to reflect true quality and diversity.

Fréchet Inception Distance (FID) [106] mitigates some of IS's drawbacks, providing a more robust and holistic assessment by comparing the distributions of real and generated images in feature space. Its calculation is as follows:

The computation process of Fréchet Inception Distance (FID):

1. Pass both generated and real images through Inception-v3 to extract 2048-dimensional features from the final pooling layer;
2. Model the features as multivariate Gaussian distributions for both sets, estimating their mean vectors (m, m_w) and covariance matrices (C, C_w);
3. Compute the Fréchet (Wasserstein-2) distance:

$$\text{FID} = \|m - m_w\|^2 + \text{Tr}\left(C + C_w - 2\sqrt{CC_w}\right),$$

where $\text{Tr}(\cdot)$ denotes the trace operation.

FID measures the "distance" between the distributions of real and generated images in the neural feature space. Lower FID values indicate that the generated distribution is closer to the real one, reflecting higher fidelity and diversity. FID effectively detects mode collapse and is now a mainstream evaluation standard. Unlike IS, FID does not depend on categorical labels, making it more broadly applicable to various datasets and tasks. However, FID assumes Gaussianity in feature distributions, which may introduce approximation errors, and its reliability is sensitive to the sample size and feature extractor used—requiring large sample numbers and consistent network settings for robust evaluation.

Chong et al. [108] note that IS and FID can both exhibit systematic biases under finite-sample conditions, yielding estimates that deviate from true values and potentially unreliable model comparisons. To address this, they propose extrapolation-based unbiased estimators (IS_∞ and FID_∞) for the infinite-sample limit and employ quasi-Monte Carlo techniques to improve finite-sample accuracy, thereby enabling reliable and unbiased comparisons.

CLIP score is a recently popular metric for text-to-image generation, rooted in the CLIP (Contrastive Language-Image Pre-Training) [49] model. It measures the semantic alignment between generated images and their descriptive texts:

The computation process of CLIP score:

1. Suppose a diffusion model generates a set of images $\{\mathbf{x}_i\}$ conditioned on textual prompts $\{\mathbf{t}_i\}$;
2. Encode each text $\mathbf{t}_i$ as $f_T(\mathbf{t}_i)$ using CLIP's text encoder, and each image $\mathbf{x}_i$ as $f_I(\mathbf{x}_i)$ using CLIP's image encoder;
3. Compute the cosine similarity between each image-text feature pair, averaging across all pairs to yield the CLIP score:

$$\text{CLIP score} = \mathbb{E}_{(\mathbf{x}_i,\mathbf{t}_i)}\left[\frac{f_I(\mathbf{x}_i)\cdot f_T(\mathbf{t}_i)}{\|f_I(\mathbf{x}_i)\|\cdot\|f_T(\mathbf{t}_i)\|}\right].$$

In essence, the CLIP score quantifies the semantic consistency between generated images and their corresponding text prompts. Higher scores indicate better image-text alignment, reflecting the model's ability to "understand" and correctly visualize the input text. The CLIP score is annotation-free, efficient to compute, and captures cross-modal semantic matching, making it widely applicable to multimodal generative tasks. However, its reliability is closely tied to the generalization ability of the CLIP model; for out-of-distribution image-text pairs, scores may be distorted. Additionally, it is less sensitive to image quality and diversity, and thus should generally be used in conjunction with FID or other metrics for comprehensive evaluation.

Similar to the CLIP score, R-precision [109] also assesses the semantic consistency between generated images and input text descriptions. As a retrieval-based metric, it computes the proportion of top-R candidate captions, ranked by feature similarity (often via cosine similarity in a joint embedding space computed by a pretrained multimodal model such as CLIP), that contain the ground truth caption for each

generated image—thereby reflecting the semantic match between generated outputs and textual inputs.

Limitations of Standard Metrics for Edge Scenarios: While IS, FID, and CLIP scores are standard benchmarks, their reliability degrades significantly when evaluating edge-specific optimizations. Firstly, **small-batch bias** is a prominent issue. Standard FID calculation requires 30K to 50K generated images to estimate true distributions. Given the limited computational power of mobile devices, edge models are often evaluated on much smaller subsets (e.g., 1K to 5K images), which statistically inflates the FID score and renders cross-paper comparisons unfair. Secondly, **calibration effects** induced by Post-Training Quantization (PTQ) or structural pruning can cause subtle feature shifts in the intermediate representations. This causes standard Inception-v3 based FID to report massive quality degradation, even when human perceptual quality remains largely unchanged. Lastly, there is a distinct **prompt coverage gap**. Cloud-based models are benchmarked on lengthy, highly descriptive prompts (e.g., MS-COCO or LAION). In contrast, real-world edge-user interactions typically involve short, colloquial, or fragmented prompts. Current CLIP score evaluations fail to capture this mobile-specific linguistic distribution, highlighting the urgent need for edge-tailored evaluation benchmarks.

5.4 System-Level Metrics and Edge Hardware Platforms

Evaluating edge-side diffusion models requires moving beyond standard vision metrics (e.g., FID) to system-level profiling. Realistic edge benchmarks must account for the physical limitations of the hardware:

Hardware Platforms: Edge deployments primarily target mobile Systems-on-Chip (SoCs), such as Qualcomm Snapdragon (Android) and Apple A-series/M-series chips (iOS/iPadOS). Unlike discrete cloud GPUs, these SoCs utilize Unified Memory Architectures (UMA) and rely heavily on specialized Neural Processing Units (NPUs) or Apple's Neural Engine (ANE) to accelerate matrix multiplications under strict power envelopes.

System-Level Metrics:

- **Peak Memory Footprint (RAM):** Edge devices typically have shared memory limits (e.g., iOS restricts single apps to 2-3 GB of RAM). Models exceeding this trigger out-of-memory (OOM) crashes.
- **End-to-End Latency:** Measured in seconds, capturing the total time from prompt input to image rendering. It includes text encoder processing, iterative U-Net denoising, and VAE decoding.
- **Energy Consumption and Thermal Throttling:** Running large models intensively drains battery and generates heat. Edge benchmarks must monitor thermal design power (TDP), as mobile OSs will aggressively throttle NPU/GPU frequencies (slowing down inference) if the device overheats.

6 Edge-side Deployment

This section analyzes mainstream acceleration methods for stable diffusion models (SDMs). We begin with network quantization, which reduces compute, memory, and storage costs with minimal architectural change. We then survey streamlined architectural designs that simplify network structure for efficiency, followed by knowledge and step distillation: the former transfers capabilities from larger to smaller models, while the latter substantially reduces inference steps. Finally, we cover low-level computational optimizations that leverage hardware- and system-level techniques to further boost performance.

6.1 Network Parameter Quantization

In the field of accelerating diffusion models, network parameter quantization techniques have demonstrated unique advantages due to their ability to reduce model size and computation without significant modification of the network architecture. Compared to methods such as efficient architectural design and network pruning, quantization reduces computational memory and storage requirements by converting floating-point weights into low-bit fixed-point representations [110, 111], all while preserving the integrity of the pre-trained model structure. Existing quantization approaches can be broadly categorized into Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). Below, we introduce representative works from both categories.

Post-Training Quantization (PTQ) techniques have attracted considerable attention due to their deployment efficiency. These methods require only a small calibration dataset to adjust quantization parameters, thus avoiding the lengthy and resource-intensive full retraining process. However, the unique property of diffusion models—the time-varying data distribution—poses additional challenges for conventional PTQ approaches. To address this, researchers have proposed several innovative solutions: PTQ4DM [83] introduces a timestep calibration approach based on skew-normal distribution (NDTC), dynamically generating calibration samples across multiple timesteps to achieve 8-bit quantization performance on par with full precision models. Q-Diffusion [82] develops timestep-aware calibration and block-wise quantization strategies for residual connections, achieving stable performance even at 4-bit quantization for the first time. PTQD [112] decouples quantization noise and applies correlation coefficient correction and variance scheduling for compensation, introducing a step-aware mixed-precision policy that significantly reduces computational demand while maintaining generation quality. TDQ [113] proposes a dynamic quantization method leveraging timestep information, employing a learnable temporal-aware module to automatically generate optimal quantization configurations. TFMQ [114] addresses temporal feature distortion by introducing timestep-independent temporal information blocks, combined with temporal-aware reconstruction and limited-set calibration, achieving near full-precision generation quality under 4-bit quantization.

Although PTQ methods are efficient and convenient, they tend to incur considerable errors at extremely low bit-widths (e.g., 4 bits or lower), sometimes causing models to completely lose denoising capability [112]. **Quantization-Aware Training (QAT)** [115, 116], on the other hand, enables the recovery of low-bit performance through fine-tuning, but usually comes with significantly higher computational overhead compared to PTQ. To balance efficiency and performance, a series of intermediate solutions have been explored: EfficientDM [117] employs the LoRA framework [118] to propose Quantization-Aware

Low-Rank Adapters (QALoRA), enabling joint low-bit quantization of both weights and activations. This approach achieves QAT-level generation quality with efficiency comparable to PTQ. Q-DM [119] targets activation distribution oscillation and quantization error accumulation by introducing Timestep-aware Quantization (TaQ) and Noise Estimation Mimicking (NeM). BitsFusion [120] analyzes layer-wise quantization sensitivity and designs a hierarchical bit-width allocation strategy. By combining timestep embedding precomputed caches, balanced integer initialization, and alternating scaling factor optimization, it successfully compresses the UNet in Stable Diffusion v1.5 to an average of 1.99 bits.

6.2 Streamlined Network Architecture

In Section 5.1, we reviewed major architectures adopted in current large-scale text-to-image diffusion models, including the conditional UNet [3], its variant U-ViT [91], and the fully Transformer-based DiT [56]. Next, we summarize network simplification methods specifically designed for these three model classes. It is important to note that structurally simplified models typically require additional fine-tuning; detailed fine-tuning strategies will be discussed further in Section 6.3.

Model Simplification for Conditional UNet: BK-SDM [121] proposed a staged pruning strategy based on sensitivity analysis. This method selectively prunes redundant pairs of residual-attention modules in the downsampling stages, retaining only the foremost modules at each resolution to handle critical spatial transformations. During upsampling, the final modules at each resolution are preserved to ensure effective skip connections. Experimental verification further revealed that middle-stage modules have high redundancy and can be completely removed with minimal impact on generation quality. Module importance was assessed by quantifying the performance degradation caused by removing individual modules, validating the rationality of the pruning strategy. Results show that this approach can reduce parameters by 30–50% while maintaining generation quality, effectively balancing computational efficiency and model performance.

To improve UNet efficiency while avoiding the performance degradation often caused by architectural modifications, SnapFusion [21] introduced a novel architectural evolution method. First, a robust training framework is employed, where modules such as cross-attention or ResNet blocks are randomly skipped during training, improving the model's robustness to architectural changes and adaptation to various module combinations. Based on this, an architectural optimization policy is devised, leveraging an action set that includes module addition/removal, and introducing an evaluation metric ($\frac{\Delta\text{CLIP}}{\Delta\text{Latency}}$) to precisely measure each module's contribution to performance and efficiency. Modules with a higher score are retained, while less-contributive and redundant modules are dynamically removed, resulting in substantial efficiency gains without requiring additional fine-tuning. For image decoders, the method adopts structured channel pruning to achieve a lightweight architecture, transferring generative ability from the full SD-v1.5 model via knowledge distillation.

Model Simplification for U-ViT: Chen et al. [24] proposed SnapGen, an efficient diffusion model for on-device image generation, leveraging several architectural innovations: (1) Streamlined attention, retaining self-attention only at the lowest resolution; (2) Use of expanded separable convolutions [122] for better efficiency-performance tradeoff; (3) Adopting multi-query attention (MQA) [123] in place of multi-head self-attention (MHSA) to dramatically reduce parameters; (4) Optimized conditioning by introducing cross-attention at the earliest UNet stage; (5) A simplified FFN to lower computation; (6) The use of QK RMSNorm [124,125] to prevent softmax saturation and accelerate convergence—combined with 2D RoPE encoding [126] to address object duplication at high resolutions, both with negligible overhead. With just 0.38B parameters, SnapGen achieves fast 1024×1024 image synthesis on mobile devices. To address decoder efficiency for high-resolution generation, attention layers were removed, normalization streamlined, and depthwise separable convolutions employed to significantly reduce memory consumption on mobile hardware.

MobileDiffusion [22], also based on U-ViT, systematically accelerates both Transformer and convolution modules. Key Transformer optimizations include: (1) increasing block depth while narrowing channel width in the UViT backbone to improve throughput without inflating parameter count; (2) replacing self-attention with cross-attention at high resolutions for a balance of speed and quality; (3) sharing projection matrices between keys and values to minimize parameters; (4) substituting GELU with Swish for improved computational stability; (5) replacing softmax attention with ReLU attention, requiring only minor fine-tuning for adaptation; (6) simplifying FFN structure to further reduce parameterization without sacrificing performance. Convolution layers—except the outermost—are replaced with separable convolutions [122] and redundant residual blocks are pruned. For the image decoder, the architecture is made lightweight via width and depth reduction, followed by removal of KL regularization and lowering adversarial loss weight during fine-tuning, yielding a 3x decoding speedup.

Liu et al. [127] addressed the quadratic complexity of self-attention layers in diffusion models by proposing the LinFusion module, which employs a non-causal normalized linear attention mechanism. This design achieves three main breakthroughs: (1) removing temporal order constraints from sequence modeling, enabling direct interactions across all positions for global context parallelism; (2) introducing state-space low-rank projections to further reduce complexity; and (3) leveraging dynamic diagonal normalization (mimicking softmax's probabilistic expressiveness) to achieve linear complexity without loss of attention capacity. This innovation enables self-attention with linear complexity in diffusion models, offering new prospects for efficient high-resolution image synthesis.

Model Simplification for DiT: SANA [128] builds systematically on DiT and the LDM approach with several innovations: first, a highly compressed autoencoder delivers superior training efficiency and supports ultra-high-resolution generation. The linear DiT architecture replaces the standard attention mechanism with linear attention for computational gains, and integrates a Mix-FFN with 3×3 depthwise convolutions for improved local modeling, enabling a position-encoding-free (NoPE) design. For textual understanding, Gemma LLM [129]

is adopted as the encoder, together with a complex human instruction (CHI) mechanism to enhance prompt interpretation. Training data is further optimized by multi-model visual language annotation, and training employs flow-based methods [28, 72] and the Flow-DPM-Solver, enabling high-fidelity generation in as few as 14–20 sampling steps. Collectively, these modifications improve training efficiency, inference speed, and output quality.

6.3 Knowledge Distillation and Step Distillation

Knowledge Distillation is a prominent model compression technique whose core idea is to train a compact "student model" to mimic the outputs of a larger "teacher model," thereby achieving comparable or even competitive performance with significant reduction in model size [130]. In recent years, knowledge distillation has become a central approach for improving the performance of lightweight models, accelerating inference, and facilitating efficient real-world deployment. This section first introduces basic distillation approaches, including alignment of model outputs and intermediate features. We then systematically review four major classes of techniques for reducing diffusion model inference steps: progressive distillation, step-distillation by constraining diffusion trajectories, adversarially guided step-distillation, and distribution-matching step-distillation.

Output-level and Feature-level Distillation: BK-SDM [121] utilizes a novel multi-objective joint training framework to distill pruned models: standard denoising loss is first employed to ensure basic generative capability; then, an output-level distillation loss aligns student and teacher outputs; finally, feature-level distillation aligns corresponding stage-end features in both UNets. This multi-granular distillation ensures precise imitation of teacher behaviors by lightweight student models. In LinFusion [127], only the parameters of the LinFusion module are trained during fine-tuning, with an objective that innovatively combines a noise prediction loss and dual knowledge distillation losses for both final outputs and hierarchical features.

Progressive Distillation: Salimans et al. [81] introduced the Progressive Distillation framework for efficient sampling in diffusion models through iterative knowledge transfer. Their approach follows a multi-stage distillation pipeline: a teacher model is first trained for N-step sampling; then, an $N/2$-step student model is trained by minimizing the error between its single-step prediction and the teacher's two-step prediction. In subsequent rounds, the student becomes the next generation's teacher, enabling further model compaction. The method employs deterministic DDIM [45] sampling and specific parameterizations for stability, ultimately achieving image quality with only 4–8 sampling steps that matches that of the original models requiring hundreds or thousands of steps.

For classifier-free guidance (CFG) diffusion models, Meng et al. [131] proposed a two-stage distillation approach: in the first stage, the student learns to combine weighted conditional and unconditional outputs of the teacher, compressing its multi-step computation into a single-step process and leveraging random guidance strengths for flexible adaptation. The second stage uses progressive distillation [81] to further compress inference to 1–4 steps, yielding a 10–256× speedup with little quality degradation.

SnapFusion [21] introduced an efficient optimization framework with two key innovations: (1) UNet is fine-tuned under the $\mathbf{v}$-prediction regime (modeling a linear combination of noise ϵ and noise-free latents $\mathbf{x}$ as $\mathbf{v} = \alpha_t \epsilon - \sigma_t \mathbf{x}$), resulting in smoother gradients and more stable multi-step denoising; (2) a CFG-aware step-distillation scheme dynamically adjusts the guidance coefficient to control sample diversity while preserving quality. Leveraging progressive distillation, SnapFusion compresses a 50-step model to an 8-step student UNet, offering substantial speedups with little loss in fidelity.

Step Distillation via Constrained Diffusion Trajectories: Previously, we systematically introduced three new paradigms that leverage trajectory-level optimization—Rectified Flow (RF) [72], Consistency Models (CM) [70], and Latent Consistency Models (LCM) [71]. By geometrically constraining diffusion trajectories, these frameworks significantly simplify the mapping from noise to data space, enabling few-step or even single-step high-quality generation. Importantly, such approaches not only train generative models from scratch, but also provide general distillation frameworks to endow pretrained diffusion models with efficient inference capabilities.

EdgeFusion [23], built upon the lightweight BK-SDM architecture, employs a synthetic high-quality image-text alignment dataset for enhanced performance. Combined with LCM-based step-distillation, EdgeFusion can produce high-quality images in only a few inference steps.

Adversarially Guided Step-Distillation: Sauer et al. [132] proposed Adversarial Diffusion Distillation (ADD), which fuses adversarial training and distillation to address the slow inference of standard diffusion models. The core concept of ADD is to utilize adversarial losses to ensure single-step, high-fidelity generations while distilling the data modeling capability of pretrained diffusion teachers. Remarkably, ADD achieves SDXL [89]-level quality with only 1–4 sampling steps.

Building on this, Latent Adversarial Diffusion Distillation (LADD) [133] migrates the teacher–student operation to the latent space, substantially reducing computational overhead and enhancing support for high-resolution, multi-scale images and large model architectures. LADD leverages multi-scale latent features from the teacher for adversarial loss, effectively mitigating overfitting risks from pixel-level alignment and improving cross-modal generality. SnapGen [**?**], via a three-stage training regime, achieves high-quality lightweight diffusion: first, it uses RF [72] to establish a linear generative path and Flow-Euler inference with logit-normal timestep sampling for improved high-res image quality; second, it proposes a hierarchical distillation framework (output-level velocity field matching, cross-architecture feature alignment, and timestep-aware dynamic weighting) for efficient knowledge transfer from SD3.5-Large to a compact student; finally, LADD-based step-distillation synergizes diffusion and adversarial training, enabling Turbo-grade quality with very few denoising steps.

UFOGen [134] realizes single-step, high-quality, text-conditioned image synthesis by coupling diffusion and GAN objectives. Its ad-

versarial loss emphasizes visual realism in one-step outputs, ensuring handling of high noise levels and fine details, while the diffusion loss stabilizes adversarial training and preserves the semantic and structural integrity of generated content. MobileDiffusion [22] adopts adversarial fine-tuning with parameter-efficient adaptation: inheriting the generator–discriminator framework from UFOGen, it boosts single-step visual fidelity and, via LoRA [118], adapts generator weights efficiently through low-rank updates while keeping most pretrained parameters frozen.

Distribution-Matching Step-Distillation: Distribution Matching Distillation (DMD) [135] aims to distill multi-step diffusion models into single-step generators, achieving high-quality synthesis with vastly reduced inference latency. DMD combines distribution matching with regression regularization: the generator is guided to move towards the true data distribution and away from a negative (fake) distribution, while a precomputed set of teacher multi-step outputs and LPIPS loss [136] ensure the student output matches the teacher's distribution. This dual optimization framework allows one-step, high-quality generation.

Although DMD delivers strong step-distillation via distribution matching, its reliance on regression losses for stability increases computation and restricts sample quality to the teacher's limitations. To address this, DMD2 [137] introduces several enhancements: First, a discriminator is added for adversarial training, following a two-timescale update rule that prioritizes discriminator convergence so that the generator can be trained solely by discriminator gradients, eliminating the regression loss entirely. Second, adversarial loss terms directly supervised by real data distributions improve sample quality. Third, DMD2 innovatively employs reverse simulation training to align training input distributions with inference, resolving distribution shift issues.

Pitfalls of Classifier-Free Guidance (CFG) on the Edge: While CFG is crucial for improving text-image alignment, it introduces significant deployment pitfalls in edge scenarios. For heavily distilled models operating under extreme edge constraints (e.g., 1 to 4 steps), tuning the guidance scale becomes highly sensitive. A high CFG scale often leads to severe visual artifacts, such as oversaturation and color degradation, while aggressively penalizing generation diversity. Furthermore, calculating the unconditional prediction effectively doubles the Neural Function Evaluations (NFE) per step, directly doubling the on-device latency. Consequently, recent edge-optimized models (e.g., SnapFusion) advocate for CFG-aware distillation techniques, baking the CFG effect directly into the model weights to eliminate the need for dual forward passes during mobile inference.

6.4 Low-level Computational Optimization

Low-level computational optimization methods refer to techniques that improve computational efficiency, memory access patterns, and hardware resource scheduling at the algorithmic implementation level, thereby enhancing performance without altering the mathematical architecture of the model. Such methods typically involve fine-grained operations such as operator fusion, memory layout optimization, and parallel computation strategy adjustment, with a focus on fully leveraging hardware characteristics—such as the SIMT architecture and multi-level cache hierarchy of GPUs—to minimize redundant computation and communication overhead. In deep learning, these optimizations are often manifested as custom kernel design (e.g., CUDA/OpenCL), computation graph rewriting (e.g., TVM [138], TensorRT), and compiler-level optimization (e.g., MLIR [139]), and are particularly critical for compute-intensive tasks such as diffusion model inference.

Chen et al. [140] addressed the challenges of deploying large diffusion models on mobile devices by proposing a GPU-aware low-level optimization framework. The core techniques include: (1) designing fused operators for GroupNorm and GELU, enabling kernel-level combination on GPUs to significantly reduce memory access and overall latency; (2) selectively fusing certain Softmax operations with FlashAttention [141] to enhance execution efficiency while maintaining the precision of attention computations; and (3) optimizing standard convolution operations using the Winograd algorithm to accelerate network inference. This approach provides a reusable low-level optimization paradigm for the efficient execution of billion-parameter-scale diffusion models on edge devices.

The Model-Level Tiling (MLT) strategy introduced in EdgeFusion [23] partitions the U-Net model into smaller computation blocks, adapting to the memory constraints of edge NPUs. This method dynamically adjusts the tiling configuration to address the DRAM access bottlenecks caused by the large feature maps in residual and attention blocks, while maximizing SRAM utilization. Notably, it leverages NPU hardware features such as tensor/vector engines to automatically configure tiling parameters based on SRAM requirements, thereby significantly improving inference efficiency for U-Net on edge devices.

SANA [128] enables efficient edge inference through several techniques: employing a mixed-precision quantization policy with per-token symmetric INT8 quantization for activations, per-channel symmetric INT8 quantization for weights, and retaining FP16 precision for normalization layers, linear attention, and KV projection in cross-attention blocks to preserve semantic fidelity. Furthermore, a low-level optimization scheme based on Triton [142] is introduced. Triton is a high-performance custom GPU kernel programming framework for deep learning, providing a high level of abstraction for efficient tensor operations. SANA utilizes Triton to implement fused kernel functions for linear attention's forward and backward passes, also integrating operations such as activation functions and precision conversion within matrix multiplication, thus greatly reducing the overhead of memory data transfer and access.

6.5 Performance Trade-offs in Edge Deployment

As summarized in Table 3, achieving real-time text-to-image synthesis on edge devices is fundamentally a multi-objective optimization problem, requiring a delicate balance between memory footprint, inference latency, and generation quality.

Different optimization paradigms exhibit distinct Pareto trade-offs. **Quantization (e.g., Q-Diffusion)** excels at reducing memory consumption and memory bandwidth requirements, which is crucial for

Table 3 Quantitative comparison of representative edge-side text-to-image diffusion models. Performance metrics are reported based on their primary target mobile hardware.

Method	Optimization Category	Target Hardware	Resolution / Steps	Latency	Memory/Params
Standard SD v1.5	Baseline	iPhone 14 Pro	512×512 / 50	~ 45s	> 2.0 GB
Q-Diffusion [82]	PTQ Quantization	Snapdragon 8 Gen 2	512×512 / 20	——	——
SnapFusion [21]	Arch + Distillation	iPhone 14 Pro (NPU)	512×512 / 8	1.84s	< 1.0 GB
MobileDiffusion [22]	Architectural Design	iOS / Android	512×512 / 1	0.2s	520M Params
LCM [71]	Step Distillation	Generic Mobile GPU	512×512 / 4	——	——

mobile devices with constrained unified memory architectures. However, aggressively quantizing weights and activations below 4-bits without extensive fine-tuning typically leads to severe degradation in image quality. **Step Distillation (e.g., LCM, InstaFlow)** drastically cuts latency by reducing the required sampling steps from 50 to 1-4 steps. Nevertheless, this aggressive compression of the generation trajectory increases the risk of mode collapse, where the model loses its ability to generate diverse outputs. **Architectural Redesign (e.g., SnapFusion, MobileDiffusion)** provides the most robust Pareto improvements by fundamentally optimizing the bottleneck (e.g., removing redundant cross-attention blocks). Yet, these methods demand massive computational resources for retraining from scratch. Therefore, the current state-of-the-art for edge deployment often relies on a hybrid pipeline: employing architectural distillation followed by post-training quantization to achieve sub-second generation on flagship mobile NPUs.

6.6 Safety and Robustness Considerations on the Edge

Unlike cloud-based deployments where prompts and generated images are heavily moderated by proprietary safety APIs and NSFW (Not Safe For Work) filters, edge-side diffusion models operate completely offline. This localized execution bypasses standard safety guardrails, making mobile deployments highly susceptible to malicious prompt injections and jailbreak attacks aimed at generating harmful, explicit, or copyrighted content. Furthermore, the processes of extreme quantization and architectural pruning may inadvertently alter the model's decision boundaries, leading to unpredictable shifts in robustness. Developing lightweight, on-device safety checkers that do not bottleneck the mobile inference latency remains a critical, yet under-explored, challenge in edge AI.

■ 7 Conclusion

This paper presents a comprehensive and systematic review of text-to-image diffusion models, traversing the spectrum from fundamental mathematical theories to practical on-device deployment. We began by elucidating the foundational principles, algorithmic advances in efficient sampling, and the mainstream architectures (e.g., conditional UNet, U-ViT, and DiT) that drive large-scale generative systems. We further examined data preparation strategies and critically analyzed evaluation metrics. Crucially, addressing the imperative need for mobile edge deployment, we provided an in-depth analysis of efficient inference techniques, including network parameter quantization, streamlined structural designs, knowledge and step distillation, and low-level computational optimizations. Overall, this work serves as a structured technological roadmap, bridging the gap between theoretical research and real-world engineering implementation.

Despite the remarkable breakthroughs witnessed in recent years, achieving ubiquitous, real-time, and high-fidelity generation on resource-constrained devices remains a formidable challenge, thereby opening several promising research directions. First, the future of extreme compression lies in algorithm-hardware co-design. Rather than treating mathematical optimizations (e.g., Rectified Flow, Consistency Models) and system-level compressions (e.g., Quantization-Aware Training, operator fusion) in isolation, deep integration is required. For instance, the inherently stable trajectories of Flow Matching could be explicitly leveraged to enable ultra-low-bit quantization with minimal accuracy loss. Furthermore, the development of unified single-step generation paradigms—integrating deterministic trajectory constraints with adversarial distribution matching—and dynamic, adaptive inference architectures that elastically allocate compute based on real-time thermal constraints and image complexity, will be crucial for pushing mobile deployment to its physical limits.

Beyond computational efficiency, the evolution of foundational models necessitates advancements in data, modality, and trustworthiness. As architectures transition toward multi-modal and temporal generation (e.g., video diffusion models), addressing the quadratic complexity of attention mechanisms via spatio-temporal low-rank projections and linear attention will become indispensable for edge devices. Concurrently, as models scale, automated data annotation and the utilization of high-quality synthetic data will be critical for enhancing cross-modal generalization. Moreover, deploying such powerful generative capabilities on personal edge devices elevates the urgency of addressing security, personalization, and privacy protection. Ensuring robust watermarking, copyright compliance, and secure local fine-tuning are paramount for fostering trustworthy on-device AI.

Ultimately, the next quantum leap in edge-side diffusion models will not emerge solely from isolated mathematical proofs or aggressive hardware pruning, but from a holistic, synergistic approach. By intertwining probability flow optimization, hardware-aware intelligent scheduling, and robust privacy safeguards, the field is steadily marching toward the ultimate vision of instantaneous, photorealistic, and pervasive generative AI across all edge platforms.

■ 8 Appendixes

8.1 Derivations for DDPM Training and Inference

• ELBO Decomposition

Starting from the forward Markov structure and Jensen's inequality (cf. [25]), the ELBO for $\log p_\theta(\mathbf{x}_0)$ can be written as

$$\log p_\theta(\mathbf{x}_0) \geq \mathbb{E}_q\big[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\big] - \sum_{t=2}^{T} \mathbb{E}_q\big[\mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))\big] + \text{const}, \tag{29}$$

where expectations are w.r.t. the forward process q.

• Closed-Form Forward Marginals and Posteriors

The forward marginal in Eq. (2) follows from the Gaussian composition in Eq. (1). The corresponding posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ is also Gaussian with

$$\mu_q(\mathbf{x}_t,\mathbf{x}_0) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\,\mathbf{x}_t + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\,\mathbf{x}_0}{1-\bar{\alpha}_t}, \tag{30}$$

$$\sigma_q^2(t) = \frac{(1-\bar{\alpha}_{t-1})(1-\alpha_t)}{1-\bar{\alpha}_t}, \tag{31}$$

and the reparameterization $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$. Eliminating $\mathbf{x}_0$ yields the equivalent noise-form expression

$$\mu_q(\mathbf{x}_t,\mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}}\,\boldsymbol{\epsilon}. \tag{32}$$

• Reverse Parameterization and KL Terms

With a Gaussian reverse kernel $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mu_\theta(\mathbf{x}_t,t),\Sigma_\theta(\mathbf{x}_t,t))$, a common choice is to set the variance to the forward-posterior variance $\Sigma_\theta(\mathbf{x}_t,t) = \sigma_q^2(t)\mathbf{I}$ and to parameterize the mean via noise prediction (Eq. (4)). Under equal variances, each KL term simplifies to a mean-squared error:

$$\mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \tag{33}$$

$$= \frac{1}{2\sigma_q^2(t)}\big\|\mu_q(\mathbf{x}_t,\mathbf{x}_0) - \mu_\theta(\mathbf{x}_t,t)\big\|^2. \tag{34}$$

Substituting Eqs. (32) and (4) into (34) yields a weighted MSE between the true noise $\boldsymbol{\epsilon}$ and the predicted noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)$. Aggregating all terms leads to the denoising loss in Eq. (6) with explicit weights

$$w_t = \frac{1}{2\sigma_q^2(t)} \cdot \frac{(1-\alpha_t)^2}{\alpha_t(1-\bar{\alpha}_t)}. \tag{35}$$

• Sampling

Given $\boldsymbol{\epsilon}_\theta$, the sampler follows

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right) + \sigma_q(t)\,\mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I}), \tag{36}$$

which is equivalent to Eqs. (4) and (5). When $t = 1$, one may set the stochastic term to zero to obtain a deterministic final step [8].

• Derivation of the Evidence Lower Bound (ELBO) for DDPM

From the perspective of probabilistic modeling, the core of the DDPM training process lies in optimizing the model by maximizing the evidence lower bound (ELBO) of the data log-likelihood. Specifically, given a data sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, the log-likelihood of generating $\mathbf{x}_0$ through the reverse process is $\log p_\theta(\mathbf{x}_0)$, and its variational lower bound can be derived as follows:

$$\begin{aligned}
\log p_\theta(\mathbf{x}_0) &= \log \int p_\theta(\mathbf{x}_0,\cdots,\mathbf{x}_T)\,d\mathbf{x}_{1:T} \\
&= \log \int p_\theta(\mathbf{x}_0,\cdots,\mathbf{x}_T)\,\frac{q(\mathbf{x}_1,\cdots,\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1,\cdots,\mathbf{x}_T|\mathbf{x}_0)}\,d\mathbf{x}_{1:T} \\
&= \log \int q(\mathbf{x}_1,\cdots,\mathbf{x}_T|\mathbf{x}_0)\,\frac{p_\theta(\mathbf{x}_0,\cdots,\mathbf{x}_T)}{q(\mathbf{x}_1,\cdots,\mathbf{x}_T|\mathbf{x}_0)}\,d\mathbf{x}_{1:T} \\
&= \log \mathbb{E}_q\left[\frac{p_\theta(\mathbf{x}_0,\cdots,\mathbf{x}_T)}{q(\mathbf{x}_1,\cdots,\mathbf{x}_T|\mathbf{x}_0)}\right] \geq \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_0,\cdots,\mathbf{x}_T)}{q(\mathbf{x}_1,\cdots,\mathbf{x}_T|\mathbf{x}_0)}\right].
\end{aligned} \tag{37}$$

The last inequality above is based on Jensen's inequality. Moreover, due to the Markov property of both the forward and reverse processes, i.e.,

$$p_\theta(\mathbf{x}_0,\cdots,\mathbf{x}_T) = p_\theta(\mathbf{x}_T)\,p_\theta(\mathbf{x}_0|\mathbf{x}_1)\prod_{t=2}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t),$$

$$q(\mathbf{x}_1,\cdots,\mathbf{x}_T|\mathbf{x}_0) = q(\mathbf{x}_T|\mathbf{x}_{T-1})\prod_{t=1}^{T-1} q(\mathbf{x}_t|\mathbf{x}_{t-1}),$$

we can further expand (Eq. 37) as follows:

$$\begin{aligned}
\log p_\theta(\mathbf{x}_0) &\geq \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_0,\cdots,\mathbf{x}_T)}{q(\mathbf{x}_1,\cdots,\mathbf{x}_T|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_T)\,p_\theta(\mathbf{x}_0|\mathbf{x}_1)\prod_{t=2}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_T|\mathbf{x}_{T-1})\prod_{t=1}^{T-1} q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \\
&= \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_T)\,p_\theta(\mathbf{x}_0|\mathbf{x}_1)\prod_{t=1}^{T-1} p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})}{q(\mathbf{x}_T|\mathbf{x}_{T-1})\prod_{t=1}^{T-1} q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \\
&= \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_T)\,p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_T|\mathbf{x}_{T-1})}\right] + \mathbb{E}_q\left[\log \prod_{t=1}^{T-1} \frac{p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \\
&= \mathbb{E}_q[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)] - \mathbb{E}_q[\mathrm{KL}(q(\mathbf{x}_T|\mathbf{x}_{T-1})\|p_\theta(\mathbf{x}_T))] \\
&\quad - \sum_{t=1}^{T-1} \mathbb{E}_q[\mathrm{KL}(q(\mathbf{x}_t|\mathbf{x}_{t-1})\|p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}))],
\end{aligned} \tag{38}$$

where the last equality above follows from the definition of KL divergence. In addition, since the computation of $p_\theta(\mathbf{x}_T)$ does not involve the diffusion model and actually contains no learnable parameters, it can be treated as a constant. Finally, we obtain the following simplified form of the variational lower bound:

$$\log p_\theta(\mathbf{x}_0) \geq \mathbb{E}_q[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)] - \sum_{t=1}^{T-1} \mathbb{E}_q[\mathrm{KL}(q(\mathbf{x}_t|\mathbf{x}_{t-1})\|p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}))] + \text{const}. \tag{39}$$

When we closely examine the term $\mathbb{E}_q[\mathrm{KL}(q(\mathbf{x}_t|\mathbf{x}_{t-1})\|p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}))]$ above, we find that computing this expectation is very challenging. Specifically, we need to sample $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t+1}$ from the distribution $q(\mathbf{x}_{t-1},\mathbf{x}_{t+1}|\mathbf{x}_0)$ to evaluate the expectation, but the exact form of this distribution is unknown. To address this issue, we can *simulate* a *reverse transition function* defined by several forward transition functions q to replace $q(\mathbf{x}_t|\mathbf{x}_{t-1})$. Since this new reverse transition function shares the same direction as $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$, the expectation calculation can be greatly simplified. Specifically, according to Bayes' theorem, we have:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t)\,q(\mathbf{x}_t)}{q(\mathbf{x}_{t-1})}.$$

Moreover, since $\mathbf{x}_{t-1}$ is actually generated from $\mathbf{x}_0$ in the forward process, the distribution of $\mathbf{x}_{t-1}$ depends on $\mathbf{x}_0$. Therefore, we can rewrite the above equation in the following form based on $\mathbf{x}_0$:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}. \tag{40}$$

It can be seen that $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ in the above equation is exactly the *reverse transition function* we are looking for. At this point, we can derive a new variational lower bound for $\log p_\theta(\mathbf{x}_0)$ based on (Eq. 37) as follows:

$$\begin{aligned}\log p_\theta(\mathbf{x}_0) &\geq \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_0,\cdots,\mathbf{x}_T)}{q(\mathbf{x}_1,\cdots,\mathbf{x}_T|\mathbf{x}_0)}\right]\\ &= \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)\prod_{t=2}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_1|\mathbf{x}_0)\prod_{t=2}^T q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)}\right]\\ &= \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] + \mathbb{E}_q\left[\log\prod_{t=2}^T \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)}\right].\end{aligned} \tag{41}$$

Substituting (40) into the second term of (41), we obtain:

$$\begin{aligned}\prod_{t=2}^T \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)} &= \prod_{t=2}^T \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}}\\ &= \prod_{t=2}^T \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)} \times \prod_{t=2}^T \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}\\ &= \prod_{t=2}^T \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)} \times \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{q(\mathbf{x}_T|\mathbf{x}_0)},\end{aligned} \tag{42}$$

where the last equality holds because, for any sequence $a_1, a_2, \cdots, a_T$, we have $\prod_{t=2}^T \frac{a_{t-1}}{a_t} = \frac{a_1}{a_2}\times\frac{a_2}{a_3}\times\cdots\times\frac{a_{T-1}}{a_T} = \frac{a_1}{a_T}$. Then, substituting (42) into (41), we obtain:

$$\begin{aligned}&\mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] + \mathbb{E}_q\left[\log\prod_{t=2}^T \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)}\right]\\ &= \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right]\\ &\quad + \mathbb{E}_q\left[\log\left(\prod_{t=2}^T \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)} \times \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{q(\mathbf{x}_T|\mathbf{x}_0)}\right)\right]\\ &= \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log\frac{q(\mathbf{x}_1|\mathbf{x}_0)}{q(\mathbf{x}_T|\mathbf{x}_0)}\right]\\ &\quad + \mathbb{E}_q\left[\log\prod_{t=2}^T \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\right]\\ &= \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_T|\mathbf{x}_0)}\right] + \mathbb{E}_q\left[\log\prod_{t=2}^T \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\right]\\ &= \mathbb{E}_q[p_\theta(\mathbf{x}_0|\mathbf{x}_1)] + \mathbb{E}_q\left[\frac{p_\theta(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)}\right] + \sum_{t=2}^T \mathbb{E}_q \log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\\ &= \mathbb{E}_q[p_\theta(\mathbf{x}_0|\mathbf{x}_1)] - \mathrm{KL}(q(\mathbf{x}_T|\mathbf{x}_0)\|p_\theta(\mathbf{x}_T))\\ &\quad - \sum_{t=2}^T \mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)).\end{aligned} \tag{43}$$

In the last equality above, we used the definition of KL divergence. Since $p_\theta(\mathbf{x}_T)$ does not need to be learned, $\mathrm{KL}(q(\mathbf{x}_T|\mathbf{x}_0)\|p_\theta(\mathbf{x}_T))$ can be treated as a constant. Therefore, we obtain the following variational lower bound for the data log-likelihood $\log p_\theta(\mathbf{x}_0)$:

$$\begin{aligned}\log p_\theta(\mathbf{x}_0) \geq& \mathbb{E}_q[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)]\\ &- \sum_{t=2}^T \mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) + \text{const.}\end{aligned} \tag{44}$$

It can be seen that $\mathbf{x}_t$ and $\mathbf{x}_0$ required to compute the KL terms in (44) (which are essentially expectations) can actually be sampled from the distribution $q(\mathbf{x}_t|\mathbf{x}_0)$, which has a simple form. Therefore, compared to the previously mentioned (39), the computation of (44) is much simpler.

- Derivation of $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ in DDPM

Next, we need to determine the explicit form of $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$. According to (40), $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ is determined by $q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)$, $q(\mathbf{x}_{t-1}|\mathbf{x}_0)$ and $q(\mathbf{x}_t|\mathbf{x}_0)$. Based on the definition of the forward process in the diffusion model, the expressions for these three terms are as follows:

$$\begin{aligned}q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_t|\sqrt{\alpha_t}\mathbf{x}_{t-1},(1-\alpha_t)\mathbf{I}),\\ q(\mathbf{x}_{t-1}|\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_{t-1}|\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0,(1-\bar{\alpha}_{t-1})\mathbf{I}),\\ q(\mathbf{x}_t|\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_t|\sqrt{\bar{\alpha}_t}\mathbf{x}_0,(1-\bar{\alpha}_t)\mathbf{I}).\end{aligned}$$

Therefore, according to (40), we have:

$$\begin{aligned}&q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\\ &= \frac{\mathcal{N}(\mathbf{x}_t|\sqrt{\alpha_t}\mathbf{x}_{t-1},(1-\alpha_t)\mathbf{I})\mathcal{N}(\mathbf{x}_{t-1}|\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0,(1-\bar{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(\mathbf{x}_t|\sqrt{\bar{\alpha}_t}\mathbf{x}_0,(1-\bar{\alpha}_t)\mathbf{I})}.\end{aligned} \tag{45}$$

Subsequently, we can actually show that $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ also follows a Gaussian distribution and takes the form $\mathcal{N}(\mathbf{x}_{t-1}|\mathcal{A}\mathbf{x}_t+\mathcal{B}\mathbf{x}_0,C\mathbf{I})$. To determine the specific forms of $\mathcal{A}$, $\mathcal{B}$, and C, we can first rewrite the product of the three Gaussians on the right-hand side of (45) as a sum in the exponent, and then obtain $\mathcal{A}$, $\mathcal{B}$, and C by computing the first and second derivatives. Here, we omit the detailed derivation, and present the final form of $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ as follows:

$$\begin{aligned}q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_{t-1}|\mu_q(\mathbf{x}_t,\mathbf{x}_0),\Sigma_q(t)),\\ \mu_q(\mathbf{x}_t,\mathbf{x}_0) &= \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\mathbf{x}_0\\ \Sigma_q(t) &= \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I} \stackrel{\text{def}}{=} \sigma_q^2(t)\mathbf{I},\end{aligned} \tag{46}$$

where $\bar{\alpha}_t = \Pi_{i=1}^t \alpha_i$.

8.2 Derivations for Score-based Generative Models

- Langevin Dynamics and Convergence

Starting from the overdamped Langevin SDE

$$d\mathbf{x}_t = \tfrac{1}{2}\nabla_{\mathbf{x}}\log p(\mathbf{x}_t)\,dt + d\mathbf{w}_t, \tag{47}$$

Euler–Maruyama discretization yields the unadjusted Langevin algorithm (ULA):

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\alpha}{2}\nabla_{\mathbf{x}}\log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\alpha}\,\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(\mathbf{0},\mathbf{I}), \tag{48}$$

which converges to $p(\mathbf{x})$ as $\alpha \to 0, T \to \infty$ under regularity assumptions [39]. Finite-step bias can be corrected by Metropolis–Hastings, often omitted in practice [40,41].

- Explicit and Implicit Score Matching

Given a KDE $q_h(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{h}K((\mathbf{x}-\mathbf{x}^{(m)})/h)$, ESM targets

$$\begin{aligned}\mathcal{L}_{\text{ESM}} =& \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}\left[\|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}}\log p(\mathbf{x})\|^2\right] \\ \approx& \frac{1}{2}\mathbb{E}_{q_h(\mathbf{x})}\left[\|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}}\log q_h(\mathbf{x})\|^2\right],\end{aligned} \tag{49}$$

but degrades in high dimensions [42]. Implicit Score Matching [38] minimizes

$$\mathcal{L}_{\text{ISM}} = \mathbb{E}_{p(\mathbf{x})}\left[\text{Tr}(\nabla_{\mathbf{x}}\mathbf{s}_\theta(\mathbf{x})) + \frac{1}{2}\|\mathbf{s}_\theta(\mathbf{x})\|^2\right], \tag{50}$$

derived via integration by parts (under boundary conditions), removing dependence on the log-normalizer. The trace can be estimated with Hutchinson's trick but remains a computational bottleneck [9].

- DSM Target under Gaussian Corruption

For $q(\mathbf{x}'|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}'|\mathbf{x}_0, \sigma^2\mathbf{I})$,

$$\nabla_{\mathbf{x}'}\log q(\mathbf{x}'|\mathbf{x}_0) = \nabla_{\mathbf{x}'}\left[-\frac{1}{2\sigma^2}\|\mathbf{x}'-\mathbf{x}_0\|^2 - \frac{d}{2}\log(2\pi\sigma^2)\right] = -\frac{\mathbf{x}'-\mathbf{x}_0}{\sigma^2}. \tag{51}$$

With $\mathbf{x}' = \mathbf{x}_0 + \sigma\epsilon$, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$,

$$\nabla_{\mathbf{x}'}\log q(\mathbf{x}'|\mathbf{x}_0) = -\frac{\epsilon}{\sigma}, \tag{52}$$

and the DSM objective reduces to

$$\mathbb{E}_{q(\mathbf{x}'|\mathbf{x}_0)p(\mathbf{x}_0)}\left[\frac{1}{2}\left\|\mathbf{s}_\theta(\mathbf{x}') + \frac{\epsilon}{\sigma}\right\|^2\right], \tag{53}$$

whose minimizer equals $\nabla_{\mathbf{x}'}\log q_\sigma(\mathbf{x}')$ almost surely [42].

- Noise Conditioning and Annealed Langevin in NCSN

With a descending schedule $\{\sigma_i\}_{i=1}^{L}$, NCSN parameterizes $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ and minimizes

$$\mathcal{L}_{\text{NCSN}} = \mathbb{E}_{i,\, q(\mathbf{x}'|\mathbf{x}_0),\, p(\mathbf{x}_0)}\left[\frac{1}{2}\lambda(\sigma_i)\left\|\mathbf{s}_\theta(\mathbf{x}', \sigma_i) + \frac{\epsilon}{\sigma_i}\right\|^2\right], \quad \lambda(\sigma) = \sigma^2, \tag{54}$$

which balances loss magnitudes across noise levels [9]. Sampling uses an annealed ULA with $\alpha_i \propto \sigma_i^2$:

$$\mathbf{x}' \leftarrow \mathbf{x}' + \frac{\alpha_i}{2}\mathbf{s}_\theta(\mathbf{x}', \sigma_i) + \sqrt{\alpha_i}\,\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{55}$$

progressing from large to small σ_i to improve mode traversal and fidelity [9, 10].

- Relation to Diffusion Models

Comparing NCSN's objective with diffusion model denoising losses shows that $\epsilon_\theta(\mathbf{x}, t)$ and $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ are proportional under appropriate parameterizations, revealing a close correspondence between diffusion and score-based models [10].

8.3 Detailed SDE Derivations

This appendix provides the detailed mathematical derivations for the SDE representations of DDPM and SGM discussed in Section 3.3.

- DDPM SDE Derivation

Starting from the DDPM forward process definition [8]:

$$\mathbf{x}_i = \sqrt{1-\beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_{i-1}, \quad \mathbf{z}_{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

We introduce continuous-time parameterization:

$$\begin{aligned}&\mathbf{x}_i = \mathbf{x}(t+\Delta t), \quad \mathbf{x}_{i-1} = \mathbf{x}(t), \\ &\mathbf{z}_{i-1} = \mathbf{z}(t), \quad \beta_i = \beta(t+\Delta t)\Delta t.\end{aligned}$$

Substituting and applying Taylor expansion:

$$\begin{aligned}\mathbf{x}(t+\Delta t) &= \sqrt{1-\beta(t+\Delta t)\Delta t}\,\mathbf{x}(t) + \sqrt{\beta(t+\Delta t)\Delta t}\,\mathbf{z}(t) \\ &\approx \left(1 - \frac{1}{2}\beta(t+\Delta t)\Delta t\right)\mathbf{x}(t) + \sqrt{\beta(t+\Delta t)\Delta t}\,\mathbf{z}(t) \\ &\approx \mathbf{x}(t) - \frac{1}{2}\beta(t)\Delta t\,\mathbf{x}(t) + \sqrt{\beta(t)\Delta t}\,\mathbf{z}(t).\end{aligned}$$

Taking the limit $\Delta t \to 0$ yields the forward SDE:

$$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}dt + \sqrt{\beta(t)}d\mathbf{w}.$$

The reverse SDE follows directly from applying the time-reversal formula [43].

- SGM SDE Derivation

For Score-Based Generative Models [10], consider the forward process:

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\mathbf{z}_{i-1}, \quad \mathbf{z}_{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

With continuous-time parameterization:

$$\begin{aligned}&\mathbf{x}_i = \mathbf{x}(t+\Delta t), \quad \mathbf{x}_{i-1} = \mathbf{x}(t), \\ &\mathbf{z}_{i-1} = \mathbf{z}(t), \quad \sigma_i = \sigma(t+\Delta t), \quad \sigma_{i-1} = \sigma(t).\end{aligned}$$

We derive:

$$\begin{aligned}\mathbf{x}(t+\Delta t) &= \mathbf{x}(t) + \sqrt{\sigma(t+\Delta t)^2 - \sigma(t)^2}\,\mathbf{z}(t) \\ &\approx \mathbf{x}(t) + \sqrt{\frac{d[\sigma(t)]^2}{dt}\Delta t}\,\mathbf{z}(t).\end{aligned}$$

Taking $\Delta t \to 0$ gives the forward SDE:

$$d\mathbf{x} = \sqrt{\frac{d[\sigma(t)]^2}{dt}}d\mathbf{w}.$$

The corresponding reverse SDE is obtained through time-reversal [43].

8.4 Derivations for Conditional Reverse Process

- From Bayes' Rule to a Gaussian Mean Shift

Let $p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = \mathcal{N}(\mu, \Sigma)$ be the unconditional reverse transition [7]. By Bayes' rule,

$$p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}, y) = \frac{1}{Z}\,p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1})\,p(y \mid \mathbf{x}_t), \tag{56}$$

with Z independent of $\mathbf{x}_t$. Write

$$\log p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = -\tfrac{1}{2}(\mathbf{x}_t - \mu)^\top\Sigma^{-1}(\mathbf{x}_t - \mu) + C_0, \tag{57}$$

and take a first-order Taylor expansion of $\log p(y \mid \mathbf{x}_t)$ at $\mathbf{x}_t = \mu$:

$$\log p(y \mid \mathbf{x}_t) \approx \log p(y \mid \mu) + (\mathbf{x}_t - \mu)^\top g,\ g \triangleq \nabla_{\mathbf{x}_t} \log p(y \mid \mathbf{x}_t)\big|_{\mathbf{x}_t=\mu}. \tag{58}$$

Summing the two logs and completing the square yields

$$\log p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}, y) = -\tfrac{1}{2}(\mathbf{x}_t - \mu)^\top \Sigma^{-1}(\mathbf{x}_t - \mu) + (\mathbf{x}_t - \mu)^\top g + C \tag{59}$$

$$= -\tfrac{1}{2}(\mathbf{x}_t - \mu - \Sigma g)^\top \Sigma^{-1}(\mathbf{x}_t - \mu - \Sigma g) + C', \tag{60}$$

which implies

$$p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}, y) = \mathcal{N}(\mu + \Sigma g,\ \Sigma), \tag{61}$$

establishing the mean-shift form in Eq. (23).

- Classifier Guidance Update

Using the auxiliary classifier $p_\phi(y \mid \mathbf{x}_t)$ [7] as a proxy for $p(y \mid \mathbf{x}_t)$, the gradient term g in the Gaussian mean shift becomes $g = \nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t)$. Incorporating a tunable scale s yields

$$\hat{\mu}_\theta(\mathbf{x}_t \mid y) = \mu_\theta(\mathbf{x}_t) + s \cdot \Sigma_\theta(\mathbf{x}_t) \cdot \nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t), \tag{62}$$

matching Eq. (24).

- CLIP Guidance Update

Let $f(\cdot)$ and $g(\cdot)$ be CLIP image and text encoders [49]. With the similarity $f(\mathbf{x}_t) \cdot g(y)$ used as a surrogate score for $p(y \mid \mathbf{x}_t)$ [50–52], we take $g = \nabla_{\mathbf{x}_t}\big(f(\mathbf{x}_t) \cdot g(y)\big)$, leading to

$$\hat{\mu}_\theta(\mathbf{x}_t \mid y) = \mu_\theta(\mathbf{x}_t) + s \cdot \Sigma_\theta(\mathbf{x}_t) \cdot \nabla_{\mathbf{x}_t}\big(f(\mathbf{x}_t) \cdot g(y)\big), \tag{63}$$

as in Eq. (25).

- Classifier-Free Guidance Derivation

By Bayes' rule,

$$\log p(y \mid \mathbf{x}_t) = \log p(\mathbf{x}_t \mid y) - \log p(\mathbf{x}_t) + \text{const}, \tag{64}$$

hence

$$\nabla_{\mathbf{x}_t} \log p(y \mid \mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid y) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t). \tag{65}$$

In DDPMs, the denoiser $\epsilon_\theta(\mathbf{x}_t \mid y)$ parameterizes the conditional score, while replacing y with a null token $\emptyset$ during training yields $\epsilon_\theta(\mathbf{x}_t \mid \emptyset)$ for the unconditional score [53]. Scaling the difference gives

$$\hat{\epsilon}_\theta(\mathbf{x}_t \mid y) = \epsilon_\theta(\mathbf{x}_t \mid \emptyset) + s \cdot \big(\epsilon_\theta(\mathbf{x}_t \mid y) - \epsilon_\theta(\mathbf{x}_t \mid \emptyset)\big), \tag{66}$$

which is Eq. (27).

- Derivation of $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}, y)$ in the Conditional Reverse Process

Suppose we already have the unconditional distribution $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$. When incorporating external information y, the conditional generative distribution $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}, y)$ can be derived using Bayes' theorem as follows:

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{x}_{t+1}, y) &= \frac{p(\mathbf{x}_t, \mathbf{x}_{t+1}, y)}{p(\mathbf{x}_{t+1}, y)} = \frac{p(\mathbf{x}_t, \mathbf{x}_{t+1}, y)}{p(y|\mathbf{x}_{t+1})p(\mathbf{x}_{t+1})} \\ &= \frac{p(\mathbf{x}_t, y|\mathbf{x}_{t+1})p(\mathbf{x}_{t+1})}{p(y|\mathbf{x}_{t+1})p(\mathbf{x}_{t+1})} \\ &= \frac{p(\mathbf{x}_t|\mathbf{x}_{t+1})p(y|\mathbf{x}_t, \mathbf{x}_{t+1})p(\mathbf{x}_{t+1})}{p(y|\mathbf{x}_{t+1})p(\mathbf{x}_{t+1})} \\ &= \frac{p(\mathbf{x}_t|\mathbf{x}_{t+1})p(y|\mathbf{x}_t, \mathbf{x}_{t+1})}{p(y|\mathbf{x}_{t+1})}. \end{aligned} \tag{67}$$

Besides, we have

$$\begin{aligned} p(y|\mathbf{x}_t, \mathbf{x}_{t+1}) &= p(\mathbf{x}_{t+1}|\mathbf{x}_t, y)\frac{p(y|\mathbf{x}_t)}{p(\mathbf{x}_{t+1}|\mathbf{x}_t)} \\ &= p(\mathbf{x}_{t+1}|\mathbf{x}_t)\frac{p(y|\mathbf{x}_t)}{p(\mathbf{x}_{t+1}|\mathbf{x}_t)} = p(y|\mathbf{x}_t). \end{aligned}$$

That is, predicting the external information y from $\mathbf{x}_t$ is actually independent of $\mathbf{x}_{t+1}$. Substituting the above equation into (67), we obtain:

$$p(\mathbf{x}_t|\mathbf{x}_{t+1}, y) = \frac{p(\mathbf{x}_t|\mathbf{x}_{t+1})p(y|\mathbf{x}_t)}{p(y|\mathbf{x}_{t+1})}.$$

Moreover, since $p(y|\mathbf{x}_{t+1})$ is independent of the generation of $\mathbf{x}_t$, we can treat this term as a constant Z. In addition, $p(\mathbf{x}_t|\mathbf{x}_{t+1})$ in the above equation can be approximated by the previously introduced neural network $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$. Thus, we have:

$$p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}, y) = \frac{1}{Z}p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})p(y|\mathbf{x}_t).$$

References

[1] Kingma D P, Welling M. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013

[2] Oord v. d A, Kalchbrenner N, Kavukcuoglu K. Pixel recurrent neural networks. In: Balcan M, Weinberger K Q, eds, International Conference on Machine Learning. 2016, 1747–1756

[3] Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B. High-resolution image synthesis with latent diffusion models. In: IEEE Conference on Computer Vision and Pattern Recognition. 2022, 10684–10695

[4] Zhao J, Mathieu M, LeCun Y. Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126, 2016

[5] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Advances in Neural Information Processing Systems. 2014, 139–144

[6] Rezende D J, Mohamed S, Wierstra D. Stochastic backpropagation and approximate inference in deep generative models. In: International Conference on Machine Learning. 2014, 1278–1286

[7] Dhariwal P, Nichol A. Diffusion models beat gans on image synthesis. In: Advances in Neural Information Processing Systems. 2021, 8780–8794

[8] Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems. 2020, 6840–6851

[9] Song Y, Ermon S. Generative modeling by estimating gradients of the data distribution. In: Advances in Neural Information Processing Systems. 2019

[10] Song Y, Sohl-Dickstein J, Kingma D P, Kumar A, Ermon S, Poole B. Score-based generative modeling through stochastic differential equations. In: International Conference on Learning Representations. 2020

[11] Amit T, Nachmani E, Shaharbany T, Wolf L. Segdiff: Image segmentation with diffusion probabilistic models. arXiv preprint arXiv:2112.00390, 2021

[12] Ho J, Salimans T, Gritsenko A, Chan W, Norouzi M, Fleet D J. Video diffusion models. arXiv preprint arXiv:2204.03458, 2022

[13] Kawar B, Vaksman G, Elad M. Stochastic image denoising by sampling from the posterior distribution. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021, 1866–1875

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

[15] Austin J, Johnson D D, Ho J, Tarlow D, Berg v. d R. Structured denoising diffusion models in discrete state-spaces. In: Advances in Neural Information Processing Systems. 2021

[16] Li X L, Thickstun J, Gulrajani I, Liang P, Hashimoto T B. Diffusion-lm improves controllable text generation. arXiv preprint arXiv:2205.14217, 2022

[17] Yu P, Xie S, Ma X, Jia B, Pang B, Gao R, Zhu Y, Zhu S C, Wu Y N. Latent diffusion energy-based model for interpretable text modelling. In: International Conference on Machine Learning. 2022, 25702–25720

[18] Alcaraz J M L, Strodthoff N. Diffusion-based time series imputation and forecasting with structured state space models. arXiv preprint arXiv:2208.09399, 2022

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

[20] Tashiro Y, Song J, Song Y, Ermon S. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. In: Advances in Neural Information Processing Systems. 2021, 24804–24816

[21] Li Y, Wang H, Jin Q, Hu J, Chemerys P, Fu Y, Wang Y, Tulyakov S, Ren J. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. In: Advances in Neural Information Processing Systems. 2023, 20662–20678

[22] Zhao Y, Xu Y, Xiao Z, Jia H, Hou T. Mobilediffusion: Instant text-to-image generation on mobile devices. In: European Conference on Computer Vision. 2024, 225–242

[23] Castells T, Song H K, Piao T, Choi S, Kim B K, Yim H, Lee C, Kim J G, Kim T H. Edgefusion: on-device text-to-image generation. arXiv preprint arXiv:2404.11925, 2024

[24] Chen J, Hu D, Huang X, Coskun H, Sahni A, Gupta A, Goyal A, Lahiri D, Singh R, Idelbayev Y, others . Snapgen: Taming high-resolution text-to-image models for mobile devices with efficient architectures and training. In: Proceedings of the Computer Vision and Pattern Recognition Conference. 2025, 7997–8008

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

[26] Song Y, Ermon S. Improved techniques for training score-based generative models. In: Advances in Neural Information Processing Systems. 2020, 12438–12448

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

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

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

[30] Croitoru F A, Hondru V, Ionescu R T, Shah M. Diffusion models in vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023, 45(9): 10850–10869

[31] Bie F, Yang Y, Zhou Z, Ghanem A, Zhang M, Yao Z, Wu X, Holmes C, Golnari P, Clifton D A, others . Renaissance: A survey into ai text-to-image generation in the era of large model. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024

[32] Song W, Ma W, Zhang M, Zhang Y, Zhao X. Lightweight diffusion models: a survey. Artificial Intelligence Review, 57

[33] Jiang R, Zheng G C, Li T, Yang T R, Wang J D, Li X. A survey of multimodal controllable diffusion models. Journal of Computer Science and Technology, 2024, 39(3): 509–541

[34] Fuest M, Ma P, Gui M, Schusterbauer J, Hu V T, Ommer B. Diffusion models and representation learning: A survey. arXiv preprint arXiv:2407.00783, 2024

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

[36] He C, Shen Y, Fang C, Xiao F, Tang L, Zhang Y, Zuo W, Guo Z, Li X. Diffusion models in low-level vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2025

[37] Koller D, Friedman N. Probabilistic graphical models: principles and techniques. MIT press, 2009

[38] Hyvärinen A. Estimation of non-normalized statistical models by score matching. J. Mach. Learn. Res., 2005, 6: 695–709

[39] Welling M, Teh Y W. Bayesian learning via stochastic gradient langevin dynamics. In: International Conference on Machine Learning. 2011, 681–688

[40] Du Y, Mordatch I. Implicit generation and generalization in energy-based models. arXiv preprint arXiv:1903.08689, 2019

[41] Nijkamp E, Hill M, Han T, Zhu S C, Wu Y N. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. arXiv preprint arXiv:1903.12370, 2019

[42] Vincent P. A connection between score matching and denoising autoencoders. Neural computation, 2011, 23(7): 1661–1674

[43] Anderson B D. Reverse-time diffusion equation models. Stochastic Processes and their Applications, 1982, 12(3): 313–326

[44] Jolicoeur-Martineau A, Li K, Piché-Taillefer R, Kachman T, Mitliagkas I. Gotta go fast when generating data with score-based models. 2021

[45] Song J, Meng C, Ermon S. Denoising diffusion implicit models. In: International Conference on Learning Representations. 2020

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

[47] Chen C F R, Fan Q, Panda R. Crossvit: Cross-attention multi-scale vision transformer for image classification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2021

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

[49] Radford A, Kim J W, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, others . Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning. 2021, 8748–8763

[50] Cimino M G C A, Galatolo F A, Vaglini G. Generating images from caption and vice versa via clip-guided generative latent space search. In: Proceedings of the International Conference on Image Processing and Vision Engineering. 2021, 166–174

[51] Patashnik O, Wu Z, Shechtman E, Cohen-Or D, Lischinski D. Styleclip: Text-driven manipulation of stylegan imagery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2021, 2085–2094

[52] Gal R, Patashnik O, Maron H, Bermano A H, Chechik G, Cohen-Or D. Stylegan-nada: Clip-guided domain adaptation of image generators.

ACM Trans. Graph., 2022, 41(4)

[53] Ho J, Salimans T. Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598, 2022

[54] Vahdat A, Kreis K, Kautz J. Score-based generative modeling in latent space. In: Advances in Neural Information Processing Systems. 2021, 11287–11302

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

[56] Peebles W, Xie S. Scalable diffusion models with transformers. In: 2023 IEEE/CVF International Conference on Computer Vision. 2023, 4172–4182

[57] Bao F, Li C, Zhu J, Zhang B. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In: International Conference on Learning Representations. 2021

[58] Kingma D, Salimans T, Poole B, Ho J. Variational diffusion models. In: Advances in Neural Information Processing Systems. 2021, 21696–21707

[59] Jolicoeur-Martineau A, Piche-Taillefer R, Combes d R T, Mitliagkas I. Adversarial score matching and improved sampling for image generation. ArXiv, 2021, abs/2009.05475

[60] Dockhorn T, Vahdat A, Kreis K. Score-based generative modeling with critically-damped langevin diffusion. In: International Conference on Learning Representations. 2021

[61] Zhang Q, Tao M, Chen Y. gddim: Generalized denoising diffusion implicit models. arXiv preprint arXiv:2206.05564, 2022

[62] Hoogeboom E, Salimans T. Blurring diffusion models. In: International Conference on Learning Representations. 2023

[63] Liu L, Ren Y, Lin Z, Zhao Z. Pseudo numerical methods for diffusion models on manifolds. In: International Conference on Learning Representations. 2021

[64] Song Y, Dhariwal P, Chen M, Sutskever I. Consistency models. 2023

[65] Ascher U M, Petzold L R. Computer methods for ordinary differential equations and differential-algebraic equations. volume 61. Siam, 1998

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

[67] Dockhorn T, Vahdat A, Kreis K. GENIE: Higher-Order Denoising Diffusion Solvers. Advances in Neural Information Processing Systems, 2022

[68] Shaul N, Perez J, Chen R T, Thabet A, Pumarola A, Lipman Y. Bespoke solvers for generative flow models. arXiv preprint arXiv:2310.19075, 2023

[69] Lipman Y, Chen R T, Ben-Hamu H, Nickel M, Le M. Flow matching for generative modeling. In: The Eleventh International Conference on Learning Representations. 2022

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

[71] Luo S, Tan Y, Huang L, Li J, Zhao H. Latent consistency models: Synthesizing high-resolution images with few-step inference. arXiv preprint arXiv:2310.04378, 2023

[72] Liu X, Gong C, others . Flow straight and fast: Learning to generate and transfer data with rectified flow. In: International Conference on Learning Representations. 2023

[73] Geng Z, Deng M, Bai X, Kolter J Z, He K. Mean flows for one-step generative modeling. arXiv preprint arXiv:2505.13447, 2025

[74] Albergo M S, Vanden-Eijnden E. Building normalizing flows with stochastic interpolants. In: International Conference on Learning Representations. 2022

[75] Esser P, Kulal S, Blattmann A, Entezari R, Müller J, Saini H, Levi Y, Lorenz D, Sauer A, Boesel F, others . Scaling rectified flow transformers for high-resolution image synthesis. In: the 41st International Conference on Machine Learning. 2024

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

[77] Polyak A, Zohar A, Brown A, Tjandra A, Sinha A, Lee A, Vyas A, Shi B, Ma C, Chuang C, others . Movie gen: A cast of media foundation models, 2025. URL https://arxiv.org/abs/2410.13720, 2024, 51

[78] Boffi N M, Albergo M S, Vanden-Eijnden E. Flow map matching. arXiv preprint arXiv:2406.07507, 2024

[79] Frans K, Hafner D, Levine S, Abbeel P. One step diffusion via shortcut models. In: International Conference on Learning Representations. 2025

[80] Zhou L, Ermon S, Song J. Inductive moment matching. In: Proceedings of the 42nd International Conference on Machine Learning. 2025

[81] Salimans T, Ho J. Progressive distillation for fast sampling of diffusion models. In: International Conference on Learning Representations. 2021

[82] Li X, Liu Y, Lian L, Yang H, Dong Z, Kang D, Zhang S, Keutzer K. Q-diffusion: Quantizing diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023, 17535–17545

[83] Shang Y, Yuan Z, Xie B, Wu B, Yan Y. Post-training quantization on diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023, 1972–1981

[84] Fang G, Li K, Ma X, Wang X. Tinyfusion: Diffusion transformers learned shallow. arXiv preprint arXiv:2412.01199, 2024

[85] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI). 2015, 234–241

[86] Nichol A Q, Dhariwal P, Ramesh A, Shyam P, Mishkin P, Mcgrew B, Sutskever I, Chen M. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. In: International Conference on Machine Learning. 2022, 16784–16804

[87] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu P J. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 2020, 21(140): 1–67

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

[89] Podell D, English Z, Lacey K, Blattmann A, Dockhorn T, Müller J, Penna J, Rombach R. Sdxl: Improving latent diffusion models for high-resolution image synthesis. arXiv preprint arXiv:2307.01952, 2023

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

[91] Hoogeboom E, Heek J, Salimans T. Simple diffusion: end-to-end diffusion for high resolution images. In: Proceedings of the 40th

International Conference on Machine Learning. 2023, 13213–13232

[92] Blattmann A, Rombach R, Ling H, Dockhorn T, Kim S W, Fidler S, Kreis K. Align your latents: High-resolution video synthesis with latent diffusion models. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2023

[93] Brooks T, Holynski A, Efros A A. Instructpix2pix: Learning to follow image editing instructions. In: IEEE Conference on Computer Vision and Pattern Recognition. 2023, 18392–18402

[94] Wang H, Du X, Li J, Yeh R A, Shakhnarovich G. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023, 12619–12629

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

[96] Esser P, Rombach R, Ommer B. Taming transformers for high-resolution image synthesis. In: IEEE Conference on Computer Vision and Pattern Recognition. 2021, 12873–12883

[97] Van Den Oord A, Vinyals O, others . Neural discrete representation learning. In: Advances in Neural Information Processing Systems. 2017

[98] Krizhevsky A. Learning multiple layers of features from tiny images. 2009

[99] Deng J, Dong W, Socher R, Li L J, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition. 2009, 248–255

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

[101] Schuhmann C, Vencu R, Beaumont R, Kaczmarczyk R, Mullis C, Katta A, Coombes T, Jitsev J, Komatsuzaki A. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. arXiv preprint arXiv:2111.02114, 2021

[102] Sharma P, Ding N, Goodman S, Soricut R. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. 2018, 2556–2565

[103] Changpinyo S, Sharma P, Ding N, Soricut R. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, 3558–3568

[104] Lin T Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick C L. Microsoft coco: Common objects in context. In: ECCV. 2014, 740–755

[105] Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X. Improved techniques for training gans. In: Advances in Neural Information Processing Systems. 2016

[106] Dowson D, Landau B. The fréchet distance between multivariate normal distributions. Journal of multivariate analysis, 1982, 12(3): 450–455

[107] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, 2818–2826

[108] Chong M J, Forsyth D. Effectively unbiased fid and inception score and where to find them. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, 6070–6079

[109] Xu T, Zhang P, Huang Q, Zhang H, Gan Z, Huang X, He X. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, 1316–1324

[110] Gholami A, Kim S, Dong Z, Yao Z, Mahoney M W, Keutzer K. A survey of quantization methods for efficient neural network inference. In: Low-Power Computer Vision. 2022, 291–326

[111] Jin Q, Ren J, Zhuang R, Hanumante S, Li Z, Chen Z, Wang Y, Yang K, Tulyakov S. F8net: Fixed-point 8-bit only multiplication for network quantization. In: International Conference on Learning Representations. 2022

[112] He Y, Liu L, Liu J, Wu W, Zhou H, Zhuang B. Ptqd: accurate post-training quantization for diffusion models. In: Adcances in Neural Information Processing Systems. 2023

[113] So J, Lee J, Ahn D, Kim H, Park E. Temporal dynamic quantization for diffusion models. In: Adcances in Neural Information Processing Systems. 2023

[114] Huang Y, Gong R, Liu J, Chen T, Liu X. Tfmq-dm: Temporal feature maintenance quantization for diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024, 7362–7371

[115] Krishnamoorthi R. Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342, 2018

[116] Esser S K, McKinstry J L, Bablani D, Appuswamy R, Modha D S. Learned step size quantization. arXiv preprint arXiv:1902.08153, 2019

[117] He Y, Liu J, Wu W, Zhou H, Zhuang B. Efficientdm: Efficient quantization-aware fine-tuning of low-bit diffusion models. In: International Conference on Learning Representations. 2024

[118] Hu E J, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, Wang L, Chen W. Lora: Low-rank adaptation of large language models. In: International Conference on Learning Representations. 2022

[119] Li Y, Xu S, Cao X, Sun X, Zhang B. Q-dm: an efficient low-bit quantized diffusion model. In: Adcances in Neural Information Processing Systems. 2023

[120] Sui Y, Li Y, Kag A, Idelbayev Y, Cao J, Hu J, Sagar D, Yuan B, Tulyakov S, Ren J. Bitsfusion: 1.99 bits weight quantization of diffusion model. arXiv preprint arXiv:2406.04333, 2024

[121] Kim B K, Song H K, Castells T, Choi S. Bk-sdm: Architecturally compressed stable diffusion for efficient text-to-image generation. In: Workshop on Efficient Systems for Foundation Models@ ICML2023. 2023

[122] Howard A G. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017

[123] Shazeer N. Fast transformer decoding: One write-head is all you need. arXiv preprint arXiv:1911.02150, 2019

[124] Henry A, Dachapally P R, Pawar S, Chen Y. Query-key normalization for transformers. arXiv preprint arXiv:2010.04245, 2020

[125] Zhang B, Sennrich R. Root mean square layer normalization. In: Advances in Neural Information Processing Systems. 2019

[126] Su J, Ahmed M, Lu Y, Pan S, Bo W, Liu Y. Roformer: Enhanced transformer with rotary position embedding. Neurocomputing, 2024, 568: 127063

[127] Liu S, Yu W, Tan Z, Wang X. Linfusion: 1 gpu, 1 minute, 16k image. arXiv preprint arXiv:2409.02097, 2024

[128] Xie E, Chen J, Chen J, Cai H, Tang H, Lin Y, Zhang Z, Li M, Zhu L, Lu Y, others . Sana: Efficient high-resolution image synthesis with linear diffusion transformers. In: International Conference on Learning Representations. 2025

[129] Team G, Riviere M, Pathak S, Sessa P G, Hardin C, Bhupatiraju S, Hussenot L, Mesnard T, Shahriari B, Ramé A, others . Gemma 2: Improving open language models at a practical size. arXiv preprint arXiv:2408.00118, 2024
[130] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015
[131] Meng C, Gao R, Kingma D P, Ermon S, Ho J, Salimans T. On distillation of guided diffusion models. In: NeurIPS 2022 Workshop on Score-Based Methods. 2022
[132] Sauer A, Lorenz D, Blattmann A, Rombach R. Adversarial diffusion distillation. In: European Conference on Computer Vision. 2024, 87–103
[133] Sauer A, Boesel F, Dockhorn T, Blattmann A, Esser P, Rombach R. Fast high-resolution image synthesis with latent adversarial diffusion distillation. In: SIGGRAPH Asia 2024 Conference Papers. 2024, 1–11
[134] Xu Y, Zhao Y, Xiao Z, Hou T. Ufogen: You forward once large scale text-to-image generation via diffusion gans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024, 8196–8206
[135] Yin T, Gharbi M, Zhang R, Shechtman E, Durand F, Freeman W T, Park T. One-step diffusion with distribution matching distillation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2024, 6613–6623
[136] Zhang R, Isola P, Efros A A, Shechtman E, Wang O. The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, 586–595
[137] Yin T, Gharbi M, Park T, Zhang R, Shechtman E, Durand F, Freeman B. Improved distribution matching distillation for fast image synthesis. In: Advances in Neural Information Processing Systems. 2024, 47455–47487
[138] Chen T, Moreau T, Jiang Z, Zheng L, Yan E, Cowan M, Shen H, Wang L, Hu Y, Ceze L, Guestrin C, Krishnamurthy A. Tvm: an automated end-to-end optimizing compiler for deep learning. In: Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation. 2018, 579–594
[139] Lattner C, Amini M, Bondhugula U, Cohen A, Davis A, Pienaar J, Riddle R, Shpeisman T, Vasilache N, Zinenko O. MLIR: Scaling compiler infrastructure for domain specific computation. In: 2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO). 2021, 2–14
[140] Chen Y H, Sarokin R, Lee J, Tang J, Chang C L, Kulik A, Grundmann M. Speed is all you need: On-device acceleration of large diffusion models via gpu-aware optimizations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023, 4651–4655
[141] Dao T, Fu D, Ermon S, Rudra A, Ré C. Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in neural information processing systems, 2022, 35: 16344–16359
[142] Tillet P, Kung H T, Cox D. Triton: an intermediate language and compiler for tiled neural network computations. In: Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages. 2019, 10–19

Zi-Hao Qiu received the B.E. and Ph.D. degrees in the School of Computer Science from Nanjing University, China, in 2019 and 2025, respectively. His research interests include machine learning and optimization.

Wenhao Yang received the B.E. degree in the School of Electronic Engineering from Xidian University, China, in 2022. He is currently working toward the Ph.D. degree with the School of Artificial Intelligence, Nanjing University, China. His research interests include machine learning and optimization.

Lijun Zhang received the B.E. and Ph.D. degrees in Software Engineering and Computer Science from Zhejiang University, China, in 2007 and 2012, respectively. He is currently a Professor of the School of Artificial Intelligence, Nanjing University, China. Prior to joining Nanjing University, he was a postdoctoral researcher at the Department of Computer Science and Engineering, Michigan State University, USA. His research interests include machine learning and optimization.