# Diffusion Models Acceleration: A Quick Survey

Student report: Delft University of Technology, Computer Science.
Date of Award: 7-11-2025
Supervisor: Y. Chen

Fulvio Nardi Dei da Filicaia Dotti
EEMCS Data-Intensive Systems
Tu Delft

Basile Lewandowski
Machine Learning Optimization Laboratory
University of Neuchatel

January 2025

**Abstract**

This survey explores state-of-the-art advancements in accelerating diffusion models, focusing on techniques to address their computational and memory inefficiencies. Diffusion models have achieved remarkable success in generative AI, surpassing prior paradigms like GANs in various applications, including image synthesis, text-to-image generation, video generation and more. However, their reliance on a large number of sequential sampling steps significantly hinders their efficiency compared to other generative approaches. This survey categorises and analyses 11 recent works aimed at overcoming these challenges, including quantization techniques, knowledge distillation, and distributed parallel sampling. Through this survey, we aim to provide an understanding, intuition, theory and tradeoffs behind these techniques. Finally, this work offers a valuable reference for researchers and professionals seeking to enhance or utilise fast diffusion model architectures, providing a clear overview of benchmarking parameters used for each of these works.

## 1 Introduction

Generative Diffusion models have made considerable progress in various domains of generative model tasks such as image generation [38, 32], text [35, 40], audio [14, 31] and video [37, 58, 52]. Furthermore, diffusion models have opened the door for tasks such as image inpainting [64, 42, 61], super-enhancing [39, 48], text-to-image generation [29, 44, 46, 49], 3D model generation [57, 27, 43] and more [69].

Their performance and quality surpassed the previous state-of-the art generative adversarial network (GAN) [23]. However, while most GAN models can generate images in under 1 second, diffusion models are not as fast, often needing several seconds to generate samples [56]. Diffusion models require a large number of sequential sampling steps (often hundreds or thousands) to generate high-quality outputs, each step evaluating the model and making the process inherently time-consuming. Futhermore, the sequential nature of the sampling steps means that each step depends on the results of the previous one, limiting opportunities for parallelization, unlike models that can generate content in a single forward pass.

Therefore, diffusion model sampling speed is a major hurdle, and efforts to accelerate the sampling speed focus on 2 major fields, consisting of quantization [15, 56, 60, 63, 65, 68, 55] and knowledge distillation [51, 59, 66]. This literature survey features state-of-the-art research related to diffusion model acceleration in those 2 fields as well as in recent efforts in distributed and parallel diffusion models sampling [51, 59].

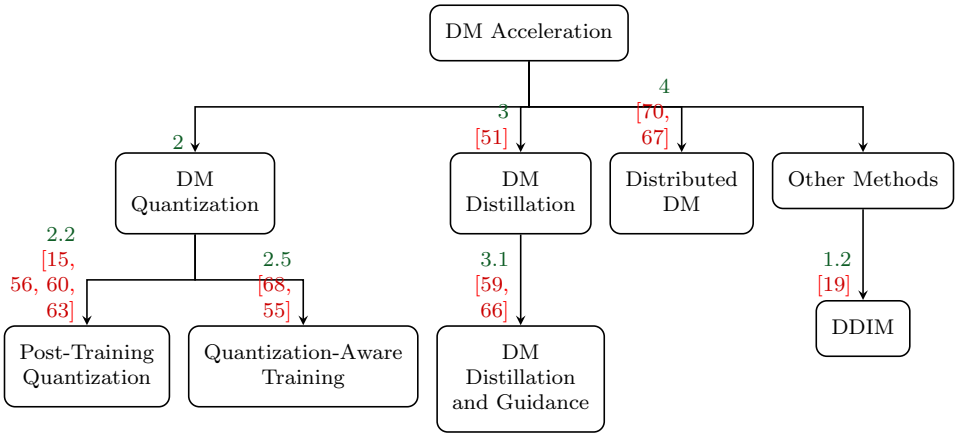


Figure 1: Categorization of fast diffusion models research papers

## 1.1 Survey Structure

This survey explores the state-of-the-art advancements in accelerating diffusion models. The paper is structured as follows. In section 1.2 we begin by introducing the foundational concepts behind denoising diffusion probabilistic models (DDPMs) and their deterministic variant, denoising diffusion implicit models (DDIMs). This section provides a comprehensive overview of the mathematical framework and objective functions underlying these models.

In section 2 we delve into quantization as a means to reduce computational and memory overhead. We discuss various methods, including post-training quantization and quantization-aware training, highlighting their trade-offs and applications in diffusion models.

In section 3 we examine the use of progressive and guided distillation techniques to compress diffusion models, enabling faster sampling without compromising quality. We also discuss recent innovations in student-teacher frameworks and guidance-aware distillation.

In section 4 we explore strategies for distributed and parallel sampling, including ParaDiGMS [70] and DistriFusion[67], which aim to leverage hardware resources for efficient generation of high-resolution samples.

Finally, we discuss the implications of these techniques, their integration into real-world applications, and future directions for accelerating diffusion models.

The structure of fast diffusion models is further categorized in subsections as shown in the tree in figure 1 while a general overview of the acceleration methods is shown in table 1.

## 1.2 Background

### 1.2.1 Denoising Diffusion Probabilistic Models (DDPM) [16]

Denoising Diffusion Probabilistic Models (DDPMs) are a class of probabilistic generative models that utilize a two-step process: a forward diffusion process that incrementally adds noise to data and a reverse generative process that removes noise to recover the original data. These models are governed by a Markovian framework and employ Gaussian noise transitions.

| Research Paper | Method Name | Acceleration Method | Training Required | Demonstrated Tasks |
|---|---|---|---|---|
| 2.3.1 [56] | Q-Diffusion | Quantization 2 | No | Unconditional Image Generation, Conditional Image Generation |
| 2.3.2 [60] | PTQ4DM | Quantization 2 | No | Unconditional Image Generation |
| 2.4 [65] | PTQD | Quantization 2 | No | Unconditional Image Generation, Conditional Image Generation, Text-to-Image Generation |
| 2.4.1 [63] | - | Quantization 2 | No | Unconditional Image Generation, Conditional Image Generation |
| 2.5.1 [68] | Q-DM | Quantization 2 | Yes | Unconditional Image Generation |
| 2.5.2 [55] | EfficientDM | Quantization 2 | Yes | Unconditional Image Generation, Conditional Image Generation, Text-to-Image Generation |
| 3.0.1 [51] | - | Timestep Distillation 3 | Yes | Unconditional Image Generation |
| 3.1.1 [59] | - | Timestep Distillation 3 | Yes | Conditional Image Generation, Image Style Transfer, Image Inpainting, Text-to-Image Generation |
| 3.1.2 [66] | - | Timestep Distillation 3 | Yes | Conditional Image Generation, Image Style Transfer, Text-to-Image Generation |
| 4.0.1 [70] | Distrifusion | Parallelization 4 | No | Text-to-Image Generation |
| 4.0.2 [67] | - | Parallelization 4 | No | Text-to-Image Generation, Robotics |

Table 1: Summary of Acceleration Methods in Diffusion Models: Comparison of techniques, training requirements, and demonstrated generative tasks across recent research studies.

**The forward process** transforms the data distribution $q(\mathbf{x}_0)$ into a noise distribution $q(\mathbf{x}_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ through $T$ time steps using a Markov chain:

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

Where $x_1, x_2, ..., x_T$ are various stages of pertubations of the original data $x$. With $x_T$ representing pure noise and $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ being a transition kernel that incrementally perturbs the original data.

Typically, the most common choice for the transition kernel is

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}).$$

Here, $\beta_t \in (0, 1)$ are predefined noise scheduling parameters where if this value is closer to 1 the noise added to the data at each step is significant and minimal if it is closer to 0. Marginally, this allows noise sampling at any time step $t$:

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

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ accumulates the effect of all previous steps.

**The reverse process** on the other hand, iteratively removes the added noise from the forward process to generate data. In this stage the generative process takes place, gradually removing noise and

generating realistic samples. The reverse process can also be modelled by the Markovian chain:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t),$$

where the DDPM learns the Gaussian transitions parametrized by learnabel parameters $\theta$:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)).$$

Here the model conditioned on timestep $t$ learns to predict the Gaussian mean $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ and covariance matrix $\Sigma_\theta(\mathbf{x}_t, t)$.

The training of Denoising Diffusion Probabilistic Models (DDPMs) is based on optimizing a variational lower bound (ELBO) on the negative log-likelihood of the data. The objective can be decomposed into a series of KL divergence terms and a reconstruction term as follows:

$$L = \mathbb{E}_q \Bigg[ \sum_{t=1}^{T} D_{KL}\big(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\big) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \Bigg].$$

Here, $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ represents the true posterior from the forward process, and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is the parameterized reverse process distribution. During training, the model simplifies this objective by reparameterizing the noise prediction using the forward process variance schedule, allowing the model to directly predict the noise added at each timestep. This results in a practical training loss:

$$L_{\text{simple}} = \mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\epsilon}} \big[ \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2 \big],$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the added noise and $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ is the neural network's prediction of the noise. This simplified loss ensures stable training while maintaining alignment with the variational objective, enabling the reverse process to accurately denoise and generate realistic samples.

### 1.2.2 Denoising Diffusion Implicit Models (DDIM) [19]

Denoising Diffusion Implicit Models (DDIMs) build upon the framework of Denoising Diffusion Probabilistic Models (DDPMs) to achieve deterministic sampling, faster inference, and improved sample quality. While DDPMs rely on a stochastic reverse process, DDIMs introduce a non-Markovian deterministic sampling process that retains the same training framework but modifies the sampling dynamics to accelerate generation without compromising the data distribution.

The key innovation in DDIMs lies in defining a deterministic mapping between consecutive timesteps in the reverse process. Instead of sampling from a Gaussian distribution at each step, DDIMs use a reparameterization that directly computes $\mathbf{x}_{t-1}$ from $\mathbf{x}_t$ using a deterministic update rule. This update is derived from the same noise prediction network $\boldsymbol{\epsilon}_\theta$ trained in DDPMs. The update equation is given by:

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1}}\boldsymbol{\epsilon},$$

where $\mathbf{x}_0$ is reconstructed using:

$$\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\big(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\big).$$

DDIMs retain compatibility with the forward process in DDPMs, where the noisy data $\mathbf{x}_t$ is generated via a Markov chain. However, during inference, DDIMs deviate from the stochastic sampling of DDPMs

and instead employ the deterministic reverse process, resulting in a more efficient sampling procedure. Notably, the deterministic process preserves the flexibility to trade off sample quality and generation speed by adjusting the number of sampling steps.

## 2 Diffusion Model Quantization

Quantization is a compression method that attempts to reduce computational and memory resource use by mapping the weight and activation values of a deep learning model to a lower precision. This method has been used to deploy deep network models on mobile devices and mitigate the computational and memory costs of complex deep learning architectures [8]. For example, Wu et al. [2] propose a unified framework to simultaneously accelerate computation and reduce storage of convolutional neural networks on mobile devices with minimal accuracy loss. Ali Zadeh, et al. [22] on the other hand, propose GOBO, an NLP model (BERT) compressed down to just 3 bits reducing DRAM access [33].

Studies [6, 7, 12] show that reducing precision from fp32 (32-bit floating-point) to int8 (8-bit integer) allows deep learning models to maintain performance comparable to full-precision models [17].

Even though there are many adopted variants for quantization one of the more common integer quantization method can be illustrated through Equations (1) and (2) [18], with Equation 1 describing the scaling factor $s$, which depends on precision $b$-bit and range $\alpha$ representing the unquantized range $[\alpha, -\alpha]$.

$$s = \frac{2^{b-1} - 1}{\alpha} \tag{1}$$

In this case, the precision $b$-bit determines the number of discrete values (or mappings) that can be represented within the quantized range (possible mappings $= 2^b$). The bit value also indicates how many bits are used to represent the quantized value, with higher bits representing more accurate values but with higher memory requirements as well (e.g. int8 maps to $2^8 = 256$ distinct integer values each represented by 8 bits).

Multiplying the scaling factor $s$ to a floating point value $x$ results in the quantized **integer** representation $x_q$ that is bounded between the quantized integer range $[-2^{b-1} + 1, 2^{b-1} - 1]$ as shown in the following quantization function:

$$x_q = \text{quantize}(x, b, s) = \text{clip}\left(\text{round}(s \cdot x), -2^{b-1} + 1, 2^{b-1} - 1\right) \tag{2}$$

Equation 3 shows the corresponding dequantize operation for scale quantization which produces the corresponding floating point representation that approximates to the initial floating point value $x$.

$$\hat{x} = \text{dequantize}(x_q, s) = \frac{1}{s} x_q \tag{3}$$

While most research papers included in this survey apply the previously mentioned quantization method (integer quantization), other methods have been successful as well, such as FP16 (mapping to 16-bit floating-point values) used by NVIDIA and ASIC acceleration frameworks [4, 26] to represent values with a lower precision while mantaining minimal accuracy loss. Nonetheless, naively quantizing diffusion model parameters leads to poor performance due to 2 common challenges, consisting on "quantization error accumulation" and "varying activation ranges distributed across multiple timesteps" [56, 56, 68].

In the following sections (2.1 and 2.2) we introduce in detail the challenges of quantizing diffusion models. Then, we introduce works of Post Training Quantization (PTQ) and Quantization Aware Training (QAT) applied to diffusion models that mitigate and circumvent these challenges. Finally, we propose a complete overview of the parameters used in those works as show in table 2.

## 2.1 Quantization Error Accumulation

Quantization error accumulation occurs as the input at each time step is derived from the output of the previous time step, effectively multiplying the number of layers involved in the computation by the number of denoising steps. This iterative process amplifies small errors introduced during quantization, leading to a cascading effect where these errors compound over successive steps. As a result, the quality of the generated outputs degrades, particularly during the later stages of the denoising process, where finer details and high-frequency components of the image are refined. This presents a significant challenge in preserving model performance when reducing the precision of the model to save on computation and memory costs.

## 2.2 Varying Activation Ranges Across all Timesteps

Varying activation ranges across all timesteps presents another significant challenge in quantizing diffusion models, as it directly impacts the effectiveness of fixed-point arithmetic and the fidelity of the quantized representation. In diffusion models, the output activation ranges of the noise estimation network shift gradually across timesteps, reflecting the evolving noise distribution as the denoising process progresses. While neighboring timesteps exhibit similar distributions, the difference becomes more pronounced as the distance between timesteps increases. This non-uniformity introduces a critical problem when applying quantization techniques, as fixed minimum and maximum clipping values fail to adapt to these variations effectively.

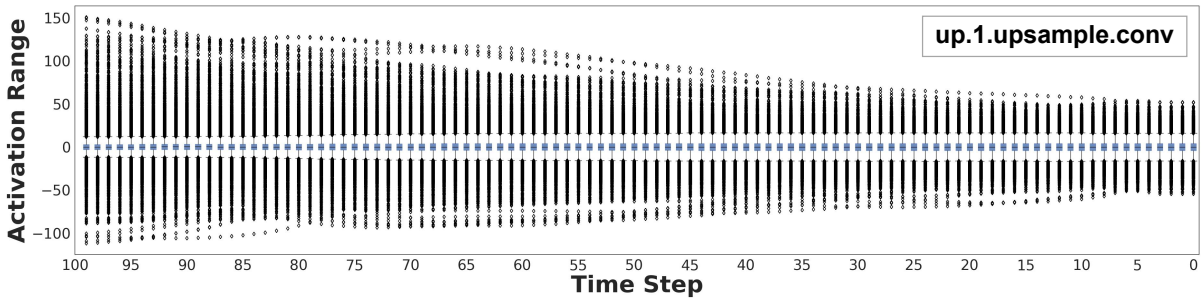


Figure 2: Visualization by [56] of varying activation ranges across timesteps, with neighbouring timesteps having similar activation range.

## 2.3 Post Training Quantization of Diffusion Models (PTQ)

PTQ involves applying quantization methods to model parameters after training to reduce memory use and accelerate inference. Contrary to Training-Aware Quantization (TAQ), PTQ is often seen as

| Diffusion Quantization Method | Quantized Model | Dataset | Sampling Steps | Sampling Procedure | W/A Quantization Bits | Metrics | QAT |
|---|---|---|---|---|---|---|---|
| Q-Diffusion 2.3.1 [56] | Song and Ermon [20] | CIFAR-10 32x32 | 100 | DDIM | 4/8, 8/8, 4/32, 8/32, 32/32* | Size (mb), GBops, FID, IS | No |
| | Latent Diffusion Model-4 | LSUN-Bedrooms 256x256 | 200 | DDIM | 4/8, 8/8, 4/32, 8/32, 32/32* | Size (mb), TBops, FID | |
| | Latent Diffusion Model-8 | LSUN-Churches 256x256 | 500 | DDIM | 4/8, 8/8, 4/32, 8/32, 32/32* | Size (mb), TBops, FID | |
| | Stable Diffusion | LAION-5B 512x512 | 50 | DDIM | 4/8 | Qualitative | |
| PTQ4DM 2.3.2 [60] | Song and Ermon [20] | ImageNet 64x64 | 100, 250 | DDIM | 8/8, 32/32* | FID, sFID, IS | No |
| | Song and Ermon [20] | ImageNet 64x64 | 4000 | DDPM | 8/8, 32/32* | FID, sFID, IS | |
| | Song and Ermon [20] | CIFAR-10 32x32 | 100, 250 | DDIM | 8/8, 32/32* | FID, sFID, IS | |
| | Song and Ermon [20] | CIFAR-10 32x32 | 4000 | DDPM | 8/8, 32/32* | FID, sFID, IS | |
| PTQD 2.4 [65] | Latent Diffusion Model-4 | ImageNet 256x256 | 20, 250 | DDIM | 4/8, 8/8, 32/32* | Size (mb), IS BOPs (T), FID, sFID | No |
| | Latent Diffusion Model-4 | LSUN-Bedrooms 256x256 | 200 | DDIM | 4/8, 8/8, 32/32* | FID, sFID | |
| | Latent Diffusion Model-8 | LSUN-Churches 256x256 | 200 | DDIM | 4/8, 8/8, 32/32* | FID, sFID | |
| Chen et Al. 2.4.1 [63] | Song and Ermon [20] | CIFAR-10 | 200 | DDIM | 4/8, 8/8, 32/32* | FID, sFID, Precision, Recall | No |
| | Latent Diffusion Model | LSUN-Bedrooms 256×256 | 200 | DDIM | 4/8, 8/8, 32/32* | FID, sFID, Precision, Recall, Qualitative | |
| | Stable Diffusion | LAION-5B 512x512 | 50 | DDIM | 4/8, 8/8, 32/32* | FID, sFID, Precision, Recall, Qualitative | |
| | Stable Diffusion V1.5 | MS-COCO | 50 | DDIM | 4/8, 8/8, 32/32* | FID, sFID, Precision, Recall | |
| Q-DM 2.5.1 [68] | Song and Ermon [20] | CIFAR-10 32x32 | 50, 100 | DDIM | 2/2, 3/3, 4/4, 32/32* | Size (mb), OPs (G), FID, IS | Yes |
| | Song and Ermon [20] | CIFAR-10 32x32 | 1000 | DDPM | 2/2, 3/3, 4/4, 32/32* | Size (mb), OPs (G), FID, IS | |
| | Song and Ermon [20] | ImageNet 64x64 | 50, 100 | DDIM | 2/2, 3/3, 4/4, 32/32* | Size (mb), OPs (G), FID, IS | |
| | Song and Ermon [20] | ImageNet 64x64 | 1000 | DDPM | 2/2, 3/3, 4/4, 32/32* | Size (mb), OPs (G), FID, IS | |
| EfficientDM 2.5.2 [55] | Latent Diffusion Model-4 | ImageNet 256x256 | 20 | DDIM | 2/8, 4/4, 4/8, 8/8, 32/32* | FID, sFID, IS, Precision (%) | Yes |
| | Latent Diffusion Model-4 | ImageNet 256x256 | 20 | PLMS | 2/8, 4/4, 4/8, 8/8, 32/32* | FID, sFID, IS, Precision (%) | |
| | Latent Diffusion Model-4 | ImageNet 256x256 | 20 | DPM-Solver | 2/8, 4/4, 4/8, 8/8, 32/32* | FID, sFID, IS, Precision (%) | |
| | Song and Ermon [20] | CIFAR-10 32x32 | 100 | DDIM | 4/4, 4/8, 8/8, 32/32* | GPU Time (Hours), Size (mb), FID, IS | |
| | Latent Diffusion Model-4 | ImageNet 256x256 | 20 | DDIM | 2/8, 4/4, 8/8, 32/32* | Size (mb), FID, sFID, IS, Precision (%) | |
| | Latent Diffusion Model-4 | LSUN-Bedrooms 256x256 | 20, 100 | DDIM | 4/4, 6/6, 8/8, 32/32* | Size (mb), FID, IS | |
| | Latent Diffusion Model-8 | LSUN-Churches 256x256 | 100 | DDIM | 4/4, 6/6, 8/8, 32/32* | Size (mb), FID, IS | |

Table 2: Overview of quantized models, datasets, sampling steps, quantization bit-widths (weights/activations), sampling procedures, performance metrics, and use of quantization-aware training (QAT) parameters used in benchmarks across various approaches. *These bitwidths represent the full precision of the model.

a desirable method because it avoids retraining models or handling training data. This characteristic makes PTQ particularly useful when data or computing is limited, however, this comes at a cost of lower accuracy compared to QAT, especially at low bit-widths [34]. To rectify error sources caused by QAT, some works [11, 9, 10, 13] have proposed bias correction methods. For diffusion models, similar solutions have been attempted and proposed, calibrating or correcting quantization to benefit from the low computational overhead of QAT while minimizing its errors.

#### 2.3.1 Q-Diffusion: Quantizing Diffusion Models [56]

This paper proposes a 4-bit quantization tool to mitigate the slow inference, high memory consumption, and computation intensity of diffusion models. The authors attempt to quantize diffusion models through a novel algorithm while tackling the issue of "varying activation ranges across all timesteps".

This paper is one of the first ones to propose the novel "Q-Diffusion Calibration" to tackle that issue. More specifically, the Q-Diffusion algorithm constructs a calibration dataset by randomly sampling 1000 activation value ranges within a set of timesteps drawn from a full-precision diffusion model. The randomly sampled activation value ranges indicate the clipping and scaling values later adopted to quantize the activation layers at particular timesteps during inference, solving the challenge of "varying activation ranges distributed across all time steps".

Furthermore, this paper observes abnormal activation and weight distributions in shortcut layers of the UNet architecture. Specifically, when these features are concatenated, the resulting activation distribution becomes bimodal, and it has two peaks corresponding to the distinct value ranges of the deep and shallow features. If a single quantization scale is applied to this concatenated tensor, it cannot effectively represent both small and large values simultaneously. This leads to significant quantization errors, especially when using low-bit precision (e.g., 4-bit), as the limited range and precision of quantization cannot capture the diverse range of values accurately.

Their proposed "shortcut splitting quantization strategy" addresses this issue by separately quantizing the deep and shallow feature maps before they are concatenated. More specifically the deep feature tensor (X1) and shallow feature tensor (X2) are quantized independently using separate quantization scales (QX1 and QX2). After quantizing X1 and X2, the quantized tensors are concatenated to produce the final activation output for the shortcut layer.

#### 2.3.2 Post-training Quantization on Diffusion Models [60]

This paper proposes Normally Distributed Time-step Calibration (NDTC), a PTQ tool that, similarly to Q-Diffusion [56], uses a calibration dataset to adjust the quantization clipping and scaling values to mitigate the "varying activation ranges distributed across multiple timesteps".

This work provides additional insight regarding properties of the calibration set that further help mitigate quantization error. For example, the authors empirically observe that using generated samples instead of (training) raw images for calibration results in better image synthesis from the quantized model (in terms of FID and IS scores). In other words, using images generated **during the backward process** results in better calibration compared to using **images used for the forward process**.

More specifically, the authors empirically demonstrate that the best calibration set is achieved by using (generated) denoised images $x_t$ at a timestep close to the fully denoised image $x_0$.

These observations are taken into account for the NDTC algorithm, which generates a set of samples for calibration using various time steps, skewing towards those closer to the generated samples at $x_0$.

Thus, the NDTC generates a calibration set by denoising $N$ random Gaussian images until reaching the timestep $t$ randomly drawn from the normal distribution visually depicted in figure 3. These calibration

samples are then used to calibrate the quantized diffusion model, reducing quantization error compared to random selection of calibration data.

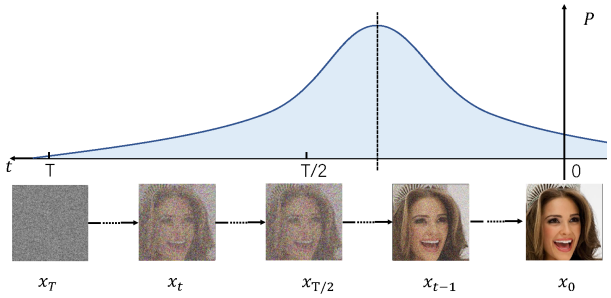


Figure 3: Visualization by [60], depicting high level representation of calibration dataset selection skewed towards $x_0$.

## 2.4 PTQD: Accurate Post-Training Quantization for Diffusion Models [65]

This paper proposes a post-training quantization tool that corrects errors introduced during the quantization procedure. The authors do so by proposing and empirically showing that quantization noise is linearly correlated with the full-precision model output. More specifically, given the predicted noise $\epsilon_\theta(\mathbf{x}_t, t)$ of a full-precision model, the quantization error follows a linear trend with correlation coefficient $k$ and intercept uncorrelated residual component of the quantization noise $\Delta'_{\epsilon_\theta}(\mathbf{x}_t, t)$ as shown in equation 4.

$$\Delta_{\epsilon_\theta}(\mathrm{x}_t, t) = k\epsilon_\theta(\mathrm{x}_t, t) + \Delta'_{\epsilon_\theta}(\mathrm{x}_t, t)\,. \tag{4}$$

The authors derive a quantization noise correction method caused by the correlation coefficient $k$ and the uncorrelated residual component of the quantization noise $\Delta'_{\epsilon_\theta}(\mathbf{x}_t, t)$ (produced from converting a full-precision model to a quantization model). The derivation starts with the standard reverse diffusion method for probabilistic models (DDIM) which finds the full precision denoised image $x$ at timestep $t-1$, as shown in equation 5.

$$\mathrm{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathrm{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\epsilon}_\theta(\mathrm{x}_t, t)\right) + \sigma_t \mathrm{z} \tag{5}$$

In equation 6 the authors demonstrate the equivalent denoising process for producing latent images $x^{(q)}$ of quantized models at timestep $t-1$. With $\Delta_{\epsilon_\theta}(x_t, t)$ denoting the quantization noise introduced due to quantization.

$$\mathrm{x}^{(\boldsymbol{q})}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathrm{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\left(\boldsymbol{\epsilon_\theta(\mathbf{x}_t, t) + \Delta_{\epsilon_\theta}(\mathbf{x}_t, t)}\right)\right) + \sigma_t \mathrm{z} \tag{6}$$

Therefore, the authors show that the **quantization noise from latent image $\boldsymbol{x^{(q)}_{t-1}}$ can be rectified by dividing the predicted noise $\boldsymbol{\epsilon_\theta(\mathbf{x}_t, t)}$ by $\boldsymbol{1+k}$** as shown in equation 7 after expanding the quantization noise with the correlation equation 4.

$$\mathbf{x}_{t-1}^{(q)} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\left((\boldsymbol{1+k})\boldsymbol{\epsilon_\theta}(\mathbf{x_t}, \boldsymbol{t}) + \boldsymbol{\Delta'_{\epsilon_\theta}}(\mathbf{x_t}, \boldsymbol{t})\right)\right) + \sigma_t \mathrm{z}. \tag{7}$$

Furthermore, the authors derive that the uncorrelated quantization error $\Delta'_{\epsilon_\theta}(\mathbf{x}_t, t)$ can be corrected as well. The derivation of the correction of the uncorrelated quantization error starts by dividing the predicted noise by $1+k$ and marginalizing $\Delta'_{\epsilon_\theta}(\mathbf{x}_t, t)$ from equation 7 to get equation 8.

$$\mathbf{x}_{t-1}^{(q)} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \boldsymbol{\sigma_t}\mathrm{z} - \frac{\beta_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}(1+k)}\Delta'_{\epsilon_\theta}(\mathbf{x}_t, t). \tag{8}$$

Therefore, the uncorrelated error $\Delta'_{\epsilon_\theta}(\mathbf{x}_t, t)$ is corrected by the authors by substituting the calibrated variance schedule $\sigma'^2$ into equation 8 while keeping the variance of each step unaltered. The calibrated variance be

$$\sigma_t^2, = \sigma_t^{'2} + \frac{\beta_t^2}{\alpha_t(1-\bar{\alpha}_t)(1+k)^2}\sigma_q^2,$$

$$\sigma_t^{'2} = \begin{cases} \sigma_t^2 - \frac{\beta_t^2}{\alpha_t(1-\bar{\alpha}_t)(1+k)^2}\sigma_q^2, & \text{if } \sigma_t^2 \geq \frac{\beta_t^2}{\alpha_t(1-\bar{\alpha}_t)(1+k)^2}\sigma_q^2, \\ 0, \text{otherwise.} \end{cases}$$

In addition, the authors mention that this solution is possible if $\Delta'_{\epsilon_\theta}(\mathbf{x}_t, t)$ follows a normal distribution such that $\Delta'_{\epsilon_\theta}(\mathbf{x}_t, t) \sim \mathcal{N}(\mu_q, \sigma_q)$.

Finally, the step-aware mixed precision scheme is a method introduced in the research paper to maintain a high signal-to-noise ratio (SNR) across the denoising steps of a quantized diffusion model. This approach adapts the bitwidth of activations dynamically during the reverse diffusion process, ensuring computational efficiency while preserving the quality of generated outputs. More specifically, the authors ensure that the minimum amount of bitwidths are used throughout quantization making sure that $\mathrm{SNR}^{\mathrm{Q}}_{b_{\min}}(t) > \mathrm{SNR}^{\mathrm{F}}(t)$ where $\mathrm{SNR}^{Q}(t)$ and $\mathrm{SNR}^{F}(t)$ are the followings:

$$\mathrm{SNR}^{F}(t) = \frac{\alpha_t^2}{\sigma_t^2}$$

$$\mathrm{SNR}^{Q}(t) = \frac{\|\epsilon_\theta(\mathbf{x}_t, t)\|^2}{\|\Delta\epsilon_\theta(\mathbf{x}_t, t)\|^2}$$

This method is particularly useful due to the varying SNR of both full precision and quantized models. The $\mathrm{SNR}_F$ being higher at earlier timesteps but decaying and getting overtaken by $\mathrm{SNR}^Q$ at later timesteps.

#### 2.4.1 Low-Bitwidth Floating Point Quantization for Efficient High-Quality Diffusion Models [63]

The paper "Low-Bitwidth Floating Point Quantization for Efficient High-Quality Diffusion Models" hypothesises and experiments if converting (full precision) fp32 diffusion model brings better results when mapped to a floating point or integer value of the same bitwidth. More specifically, the full precision

values are mapped to fp8 and fp4 (8 and 4 bit floating point values) and their integer counterpart int8 and int4 (8 and 4 bit integer values) to benchmark the 2 methods in terms of FID and IS metrics.

A standard floating-point number is represented as follows:

$$f = \underbrace{(-1)^s}_{\text{sign}} \cdot \underbrace{2^{e-\text{bias}}}_{\text{exponent}} \cdot \underbrace{(1 + \frac{d_1}{2} + \frac{d_2}{2^2} + \cdots + \frac{d_m}{2^m})}_{\text{mantissa}} \tag{9}$$

Where 1 bit is dedicated to the sign component $s \in \{0, 1\}$. The exponent $e$ is an integer that determines the magnitude or scale of the floating point number by controlling the power of 2. The bias is a fixed value used to shift the exponent range and it is typically calculated as bias $= 2^{\text{number of exponent bits}} - 1$. Finally, the mantissa bits $d_i \in \{0, 1\}$ represent the fractional part of the number in binary after normalization.

The more bits are dedicated to the exponent the larger the range of representable values, however, it does not make the representation more precise. On the other hand, the more bits are dedicated to the mantissa the more precise the representation can be.

Due to the limited number of bits used to represent values (especially after quantization), there is a tradeoff between representable range and precision. The more exponent bits leave fewer bits for the mantissa, reducing precision but increasing range and vice-versa. For example, in fp8 there are 4 possible configurations consisting of E2M5 (2-bit exponent and 5-bit mantissa), E3M4 (3-bit exponent and 4-bit mantissa), E4M3 (4-bit exponent and 3-bit mantissa), and E5M2 (5-bit exponent and 2-bit mantissa). For fp4, there are 2 encodings: E1M2 (1-bit exponent and 2-bit mantissa) and E2M1 (2-bit exponent and 1-bit mantissa) (Note that 1 bit is always dedicated to the sign bit).

The author finds the optimal bit encoding and bias by employing a greedy search. More specifically the algorithm starts with the tensors in the first layer evaluating all combinations of encoding and biases selecting the combination with the lowest MSE (compared to the full precision model). After the lowest MSE combination is found, the algorithm fixes the selected encoding and bias before proceeding to the next layer and repeating this procress for all layers in a breadth-first order.

In addition to this algorithm, the authors also incorporate a rounding learning technique (first proposed by [18]) using a gradient descend technique to learn how to map more effectively from full precision fp32 value to lower bit fp values (fp8 and fp4). This method is particularly useful where only a few mantissa bits are available, and naive rounding introduces significant errors. This method treats rounding as a learnable decision for each weight instead of statically rounding values to the nearest quantized level. More specifically a sigmoid function is used to represent the probability of rounding up or down:

$$w_{\text{quantized}} = \lfloor w \rfloor + \sigma(\alpha) \tag{10}$$

Where $w$ is the rounded weight, $\sigma$ is the sigmoid function and $\alpha$ is the learnable parameter. According to the authors' benchmarks, the fp quantization proposed by the authors appears to perform better than the uniform int quantization for the same bitwidth. Notably, rounding learning is a major contribution to the improved fp quantization results in terms of FID and IS metrics.

## 2.5 Quantization-Aware Training

QAT involves training a neural network directly in its target data type, meaning that both forward and backward passes are performed using quantized parameters. This approach often achieves more optimal solutions compared to PTQ [28]. However, the improved accuracy of QAT comes at the cost of additional computational overhead during training. Despite these challenges, QAT can sometimes be considered

more advantageous due to its ability to produce higher-performing quantized models especially when deploying models for an extended period [34].

### 2.5.1 Q-DM: AnEfficient Low-bit Quantized Diffusion Model [68]

This paper proposes a QAT for Diffusion Models motivated to address and mitigate the "Varying activation ranges across all timesteps" and "Quantization error accumulation" commonly occurring when quantizing Diffusion Models.

To address the first challenge of "Activation distribution oscillation," the authors propose Timestep-Aware Quantization (TAQ). TAQ adjust the activation value, given mini-batch collected during training, with statistical values of mean $\gamma_t$ and variance $\sigma_t^2$ for a given timestep $t$ as calculated in Equations (11) and (12).

$$\gamma_t = \sum_{i=1}^{B} \frac{1}{b_i} \sum_{j=1}^{b_i} \mathbf{a}(\mathbf{x}_{t_j}, t_j), \tag{11}$$

$$\sigma_t^2 = \sum_{i=1}^{B} \frac{1}{b_i} \sum_{j=1}^{b_i} \left[\mathbf{a}(\mathbf{x}_{t_j}, t_j) - \gamma_t\right]^2, \tag{12}$$

Where $b_i$ is the batch size of the $i$-th batch, with $i \in \{1, ..., B\}$. With the calculated mean $\gamma_t$ and variance $\sigma_t^2$ the authors propose Equation 13 to estimate the adjusted activation value $\tilde{a}(x, t)$ for a specific timestep $t$.

$$\tilde{a}(x, t) = \frac{a(x, t) - \gamma_t}{\sqrt{\sigma_t^2 + \psi}} \tag{13}$$

The adjusted activation value can be used within the quantization equation for a more accurate activation estimate w.r.t as defined by the TaQ Equations (14) and (15).

$$\text{TaQ}(a) = s \cdot Q(\tilde{a}) = s \cdot \left\lfloor \text{clip}\left(\frac{\tilde{a}}{s}, -2^{b-1}, 2^{b-1} - 1\right)\right\rfloor \tag{14}$$

$$\text{TaQ}(a) = s \cdot \left\lfloor \text{clip}\left(\frac{a - \gamma_t}{s_a \cdot \sqrt{\sigma_t^2 + \psi}}, -2^{b-1}, 2^{b-1} - 1\right)\right\rfloor \tag{15}$$

To address the second challenge of "Quantization error accumulation" the author proposes the Noise-estimating Mimicking (NeM). NeM compares the noise estimation of the quantized model with that of the full-precision model, this way, the quantized model learns to better approximate the noise estimation capability of its full-precision counterpart. NeM is a loss function that minimizes the predicted noise at a given timestep between the full precision diffusion model and the quantized diffusion model. The derived NeM loss function is the following:

$$\mathcal{L}_{\text{NeM}}(\theta_Q, \theta_{FP}) = \mathbb{E}_{t, x_0, \epsilon}\left[\left\|\epsilon_{\theta_{FP}}\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t\right) - \epsilon_{\theta_Q}\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t\right)\right\|^2\right] \tag{16}$$

With $\epsilon_{\theta_{FP}}$ and $\epsilon_{\theta_Q}$ being the predicted noise of the respective full precision and quantized models, and $\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$ being the The noisy input at timestep $t$, formed by mixing the original data sample $x_0$ with the Gaussian noise $\epsilon$, controlled by the noise schedule parameters.

#### 2.5.2 EfficientDM: Efficient Quantization-Aware Fine-Tuning of Low-Bit Diffusion Models [55]

This paper introduces EfficientDM, a framework for efficient, data-free quantization-aware fine-tuning (QAT) to low-bit diffusion models. This paper achieves low-bit quantization through QAT by leveraging their proposed Quantization-aware Low-rank Adapter (QALoRA). QALoRA is based on Low-rank adapter (LoRA) [25], a technique designed to adapt large pre-trained models efficiently for specific tasks. It avoids the computational and storage costs of traditional fine-tuning by freezing the pre-trained weights and introducing trainable low-rank matrices into the model. For example, Zhang et al. [62] use LoRA to fine-tune and adapt a pre-trained text-to-image sampler to help improve the fidelity of the generated 3D samples. Similarly, Guo et al. [54] use LoRA to extend a pre-trained text-to-image diffusion model with a motion module (e.g., zooming or panning) using a small number of reference videos.

In practice, LoRA can be illustrated by updating a linear model $\mathbf{Y} = \mathbf{X}\mathbf{W}_0$ with learnable low-rank matrices $\mathbf{B}$ and $\mathbf{A}$ as shown in the following example.

$$\mathbf{Y} = \mathbf{X}\mathbf{W}_0 + \mathbf{X}\mathbf{B}\mathbf{A} \tag{17}$$

Where $\mathbf{X} \in \mathbb{R}^{b \times c_{\text{in}}}$, $\mathbf{W}_0 \in \mathbb{R}^{c_{\text{in}} \times c_{\text{out}}}$, $\mathbf{B} \in \mathbb{R}^{c_{\text{in}} \times r}$, $\mathbf{A} \in \mathbb{R}^{c_{\text{in}} \times r}$, with $b$ representing the batch size, $c_{\text{in}}$ and $c_{\text{out}}$ representing the number of input and output channels, and $r \ll min(c_{\text{in}}, c_{\text{out}})$. The high-level example from equation 17 demonstrates how the initial weight $W_0$ can be adjusted thanks to LoRA to update the model $Y$ for new tasks.

The proposed QALoRA, similarly to LoRA, fine-tunes (pre-trained) model parameters, but it additionally integrates these updates directly into quantized weights to align with the precision of the quantized model.

$$\mathbf{Y} = Q_U(\mathbf{X}, s_x)Q_U(\mathbf{W}_0 + \mathbf{B}\mathbf{A}, s_w) = \hat{\mathbf{X}}\hat{\mathbf{W}} \tag{18}$$

Where $s_w$ denotes the channel-wise quantization scale for weights and $s_x$ is the layer-wise quantization scale for activations. The denoise step-specific learnable parameters $A$ and $B$ from Equation 18 are found through knowledge distillation between a full-precision model into its quantized counterpart. More specifically, the loss function 19 minimizes the mean squared error (MSE) between the full precision model $\mu_\theta(\mathbf{x}_t, t)$ and the quantized model $\hat{\mu}_\theta(\mathbf{x}_t, t)$ at timestep $t$, to obtain the optimal $A$ and $B$ from 18.

$$\mathcal{L}_t = \|\mu_\theta(\mathbf{x}_t, t) - \hat{\mu}_\theta(\mathbf{x}_t, t)\|^2 \tag{19}$$

Finally, the authors propose to allocate temporal-aware quantization scales (TALSQ) for activations and optimize them individually for each step. TALSQ is proposed to account for the high variability between activation across different denoising steps in diffusion models.

By leveraging the proposed Quantization-aware Low-rank Adapter (QALoRA), EfficientDM achieves both parameter efficiency and low-bit quantization without requiring the original training data.

## 3 Diffusion Model Distillation

Knowledge Distillation is the process of replicating or transferring knowledge from a large model (teacher) to a smaller and more efficient model for deployment (student) [5, 1]. Chen et al. [3] for example, propose a knowledge distillation framework to compress highly complicated object detection models (teacher) into lighter and more efficient models (student). Chae et al. [53] on the other hand apply knowledge distillation on blackbox LLM such as chatGPT to train smaller LLM with dialogue chain-of-thought capabilities.

Progressive distillation is a technique first proposed by Salimans et al. [51], which uses knowledge distillation iteratively to teach a student model to replicate the diffusion model sampling of a teacher model in half of the required timestep as shown in figure 3.

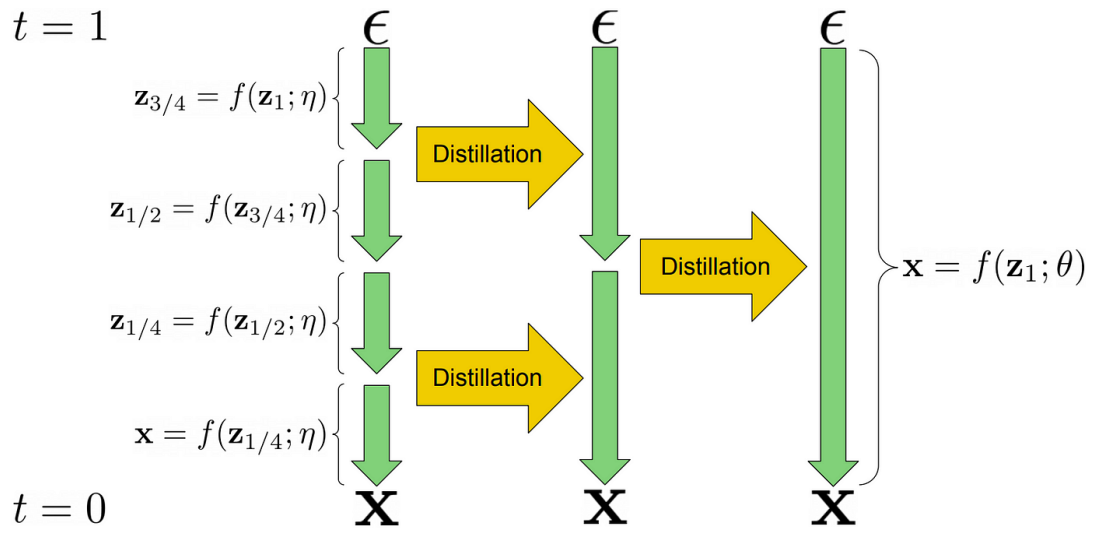


Figure 4: Visualization of the progressive distillation process proposed by Salimans et al. [51]. The latent variable $\mathbf{z}$ is initially denoised in 4 steps by sampler $f(\mathbf{z}; \eta)$. Through the distillation process, the algorithm halves the denoising steps until the sampler can generate samples through a single denoising step.

3.0.1 [51] is the first research paper to propose this method on diffusion models as well as offering some empirical findings about the optimal loss function and weighting function. 3.1.1 [59] extends this process by adding a free guidance distillation process where the student model learns to sample based on the guidance weight of the teacher model. Finally, 3.1.2 [66] applies both timestep reduction and free guidance distillation on a guide model to train a fast and lightweight model with a fraction of the learning parameters compared to the teacher model.

These 3 research paper experiments are further categorized based on the models, datasets, distilled timesteps and guidance weight used as shown in table 3, with the guidance weight further explained in section 3.1.

### 3.0.1 Progressive Distillation for Fast Sampling of Diffusion Models [51]

This paper proposes a novel approach to accelerating diffusion model sampling using knowledge distillation techniques. Their method iteratively reduces the number of sampling steps required for generating samples using diffusion models (e.g., reducing from thousands of steps to as few as 4).

The approach begins with a pre-trained diffusion model (teacher) that uses many sampling steps. The student model copies the teacher model learnable parameters and tunes them to replicate the teacher model latent image prediction in 1 step instead of 2. Therefore, the student model is tasked to predict latent images $z_t$ and $z_{t-1/N}$ of the teacher model in a single step where $N$ is the total student timesteps.

| Diffusion Distillation Method | Teacher Model for Step Reduction | Teacher Timesteps | Student Timesteps | Dataset | Teacher Model for Guidance | Guidance Weights |
|---|---|---|---|---|---|---|
| **3.0.1** [51] | Dhariwal and Nichol* [24] | 8192 | 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 | CIFAR-10 | - | - |
| | Dhariwal and Nichol† [24] | 1024 | 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 | ImageNet 64x64 | - | - |
| | Dhariwal and Nichol† [24] | 1024 | 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 | LSUN Bedrooms 128x128 | - | - |
| | Dhariwal and Nichol† [24] | 1024 | 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 | LSUN Church Outdoor 128x128 | - | - |
| **3.1.1** [59] | Stable Diffusion* [47] | 1024 | 1, 2, 4, 8, 16 | Laion 512x512 | Ho and Salimans* [36] | 0.0, 0.3, 1.0, 2.0, 4.0 |
| | Ho and Salimans* [36] | 1024 | 1, 2, 4, 8, 16 | ImageNet 64x64 | Ho and Salimans* [36] | 0.0, 0.3, 1.0, 2.0, 4.0 |
| | Ho and Salimans* [36] | 1024 | 1, 2, 4, 8, 16 | CIFAR-10 | Ho and Salimans* [36] | 0.0, 0.3, 1.0, 2.0, 4.0 |
| **3.1.2** [66] | Stable Diffusion* 1.5 | 1000 | 8, 16, 50 | Laion 512x512 | Stable Diffusion* 1.5 | 2.0, 4.0, 6.0, 8.0 |

Table 3: Key parameters used during experiment and evaluation of distilled diffusion models 3.0.1, 3.1.1, and 3.1.2. *These models' architecture are unaltered. †These models' architecture are altered.

The halving of the timestep is encouraged by the loss function, where the target $\widetilde{x}$ is the teacher 2 step latent progression, $\hat{x}_\theta(z_t)$ is the student 1 step latent progression given the current latent image $z_t$, $\lambda_t$ is the signal to noise ratio $\log[\alpha_t^2/\sigma_t^2]$, and $\omega(\lambda_t)$ is a pre-specified weighting function.

$$L_\theta = w(\lambda_t) \left\| \widetilde{x} - \hat{x}_\theta(z_t) \right\|_2^2 \tag{20}$$

Furthermore, the authors of this paper empirically found and argue that minimizing image prediction x loss compared to noise prediciton $\epsilon$ loss ensures greater stability and accuracy during progressive distillation. This is particularly relevant at low signal-to-noise ratios (SNR), where predicting the clean image x (opposed to added noise $\epsilon$) is more informative for the student to best mimic the teacher samples.

Finally, the paper benchmarks the results of the model distillation taking into consideration 3 different weighting functions $w(\lambda_t)$: SNR, truncated SNR and SNR+1. SNR+1 and Truncated SNR (contrary to SNR) are better suited for progressive distillation because they stabilize the loss during low signal-to-noise conditions, which are frequent when sampling steps are reduced.

The distilled models achieve results close to the original teacher models. For example, on CIFAR-10, they achieve an FID (Fréchet Inception Distance) of 3.0 with just 4 sampling steps.

## 3.1 Guided Diffusion Models

Classifier-free guided models, first proposed by Ho et al. [36], balance the trade-off between diversity and quality in image generation. These models use a guidance weight $w$ parameter to control the diversity and quality of the generated samples and have been adopted by frameworks such as GLIDE [30], Stable Diffusion [47], DALL-E 2 [45], and Imagen [50]. A higher guidance weight term $w$ leads to outputs with higher quality but lower diversity, conversely lower guidance results in higher diversity but lower quality.

In practice, the relationship between the guidance weight and the generation process can be expressed mathematically as:

$$\hat{x}_\theta^w(z_t) = (1+w)\hat{x}_{c,\theta}(z_t) - w\hat{x}_\theta(z_t), \quad z_t \sim q(z_t|x) \tag{21}$$

The formula combines the conditional model's prediction ($\hat{x}_{c,\theta}(z_t)$) with the unconditional model's prediction ($\hat{x}(z_t)$) to balance between quality and diversity. Where $z_t$ is a latent image at timestep $t$ and the guidance weight $w$ determines how strongly the output is biased toward the conditional prediction. In the case of $w = 0$ the model functions as a conditional generation $\hat{x}_{c,\theta}(z_t)$.

### 3.1.1 On Distillation of Guided Diffusion Models [59]

This paper proposes a student-teacher model distillation that produces a classifier-free guidance model with as low as 4 denoising steps without loss of performance compared to the teacher model. The paper highlights 2 distillation stages. The first stage describes the process of training a student model classifier-free guidance properties, the second stage describes the process of reducing the timestep for image generation inference.

In the first stage of distillation, the student model is trained to replicate the outputs of the teacher model across various guidance weight parameters $w \in [w_{min}, w_{max}]$. This stage allows the student model to approximate the teacher's behaviour for any $w$, reducing the need to evaluate conditional and unconditional models separately during inference. The student-teacher model uses the following loss function 22.

$$\mathbb{E}_{w\sim p_w,\, t\sim U[0,1],\, \mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\omega(\lambda_t)\left\|\hat{x}_{\eta_1}(\mathbf{z}_t, w) - \hat{x}_\theta^w(\mathbf{z}_t)\right\|_2^2\right] \tag{22}$$

Where $p(w) = [w_{min}, w_{max}]$, $\lambda_t$ is the signal to noise ratio $\log[\alpha_t^2/\sigma_t^2]$, and $\omega(\lambda_t)$ is a pre-specified weighting function.

In the second stage, the authors describe the process of reducing (halving) the required timesteps to generate a sample of comparable quality. This process is done through knowledge distillation, similarly to 3.0.1 this paper minimizes the loss function between the teacher's 2 steps denoised latent variable with the corresponding student's 1 step denoised latent variable

The paper demonstrates that their two-stage distillation approach reduces sampling steps for classifier-free guided diffusion models by up to 256× for pixel-space models and 16× for latent-space models, while achieving comparable FID and IS scores to the teacher models, enabling efficient high-quality image generation, editing, and inpainting with as few as 1-4 denoising steps.

### 3.1.2 Plug-and-Play Diffusion Distillation [66]

This paper proposes training a lightweight, predefined guide model with a fixed architecture using knowledge distillation from the teacher model. Unlike traditional approaches that use the same complex

architecture as the teacher and student, their method leverages the predefined student guide model to replicate the teacher's output with significantly fewer parameters and reduced computational cost.

Their method is divided into 2 stages. The first stage involves using knowledge distillation to train the predefined guide lightweight model to replicate the teacher model samples. The second stage involves using knowledge distillation to reduce (halve) the timestep required to generate a sample of the newly trained guide model.

During the first stage, similarly to paper 3.1.1 the authors train the guide model under a teacher model across a uniform sample of guidance weights. Thus, allowing the student model to be capable of producing samples of various diversity and quality based on user preference without evaluating conditional and unconditional models separately during inference.

The key difference between this approach and paper 3.1.1 is the application of knowledge distillation of guidance weight on a lightweight predefined ControlNet architecture instead of encoding guidance on the same model. Specifically, the tiny guide model adopted in this paper consists of an efficient ControlNet architecture with no encoding layers. The guidance vector, text embedding and time embedding are passed to a zero-convolution through the decoding layer.

During the second stage, similarly to the process explored previously, the sampling steps are distilled to halve the steps of the guide mode iteratively. This process is applied to the lightweight guide model reaching as low as 8–16 steps while maintaining competitive image fidelity compared to the teacher model.

The tiny guide model reduces inference FLOPs (floating-point operations) by nearly half compared to classifier-free guidance (CFG) in traditional diffusion models. The tiny guide model has ∼1% of the trainable parameters compared to Stable Diffusion v1.5.

# 4 Distributed Diffusion Model

So far most of the acceleration techniques for diffusion models trade-off sample quality for sample generation speed, namely using quantization and denoising step reduction. However, distribution and parallelization of sample generation task trade-off computing for sample generation speed.

While naively generating samples in parallel increases throughput, 4.0.1 and 4.0.2 each propose a method to accelerate the generation time for a single sample.

### 4.0.1 Parallel Sampling of Diffusion Models [70]

The authors present ParaDiGMS, a novel method to accelerate the sampling of pre trained diffusion models by denoising multiple steps in parallel. Their method leverages Picard Iterations to predict the future denoised sample $x_f$ based on the current latent variable $x_c$. Each device applies the Picard Iteration to predict denoising timesteps between starting and ending points $c$ and $f$. The Picard Iteration approximates the denoising process from timestep $c$ to $f$, improving its estimation the more $k$ repetitions are applied as shown in Equation 23.

$$x_f^{k+1} = x_c^k + \int_c^f s(x_u^k, u)\, du, \tag{23}$$

Here, $s(x_u^k, u)$ is the drift term, which is derived from the Stochastic Differential Equation (SDE) Equation 24 formulation of the denoising process. The SDE is expressed as:

$$\mathrm{d}x_t = \underbrace{\left(f(t)x_t - g^2(t)\nabla_x \log q_t(x)\right)}_{\text{drift } s}\, \mathrm{d}t + \underbrace{g(t)}_{\sigma_t}\, \mathrm{d}\bar{w}_t, \quad x_T \sim q(x_T), \tag{24}$$

This SDE-based denoising process is adopted in many works [41, 21] as an alternative to the more traditional reverse-time Ordinary Differential Equation (ODE).

In practice, ParaDiGMS divides the denoising time series into windows of size $p$ (a batch of timesteps). Each device computes updates for all timesteps within its batch window in parallel during a single iteration. Then, for each timestep $t$ in the window $[t, t+p)$ (or $[c, f)$), the drift term $s(x_u^x, u)$ is computed in parallel for all timesteps within the window. Finally, after computing the drift terms, the updates for all timesteps in the window are applied using a cumulative (prefix) sum of the drift terms. This step refines the values across the window simultaneously. Once the updates for the current window have converged, the window slides forward to the next portion of timesteps.

By combining Picard Iterations with parallel computation, ParaDiGMS achieves a significant reduction in sampling time without compromising sample quality. The method is orthogonal to existing fast sampling techniques like DDIM and DPM-Solver, meaning it can be combined with these approaches to achieve even further acceleration. For example, ParaDiGMS can compute fewer timesteps (as done in DDIM) while performing multiple steps in parallel, providing an optimal tradeoff between speed and accuracy.

#### 4.0.2 DistriFusion: Distributed Parallel Inference for High-Resolution Diffusion Models [67]

This paper proposes "Distrifusion", a distributed hardware solution as a method to improve image sampling generation speed.

Their method is motivated by the lack of a solution for distributing the workload to devices for the generation of a diffusion model sample which can be attributed to the sequential nature of denoising images. Naive approaches may attempt to patch multiple separate generated images, each in a distinct device (Naive Patch) however, at the cost of clear visual artifacts separating the image into multiple sections.

Distrifusion, on the other hand, proposes a method that accelerates image sampling comparable to the speed of the Naive Patch, while maintaining a similar quality of image generation to that of a single device.

Their method involves using the previous denoising timestep to provide cohesion context (to avoid separation artifacts) for generating a patch for the current timestep for each device. This method allows each device to focus its hardware resources on generating a single patch of image while maintaining cohesion between these patches.

More specifically, cohesion context is given through the input activation of the previous denoise timestep (stale activation) using the Scatter operation. The Scatter operation distributes the stale activation $A_{t+1}^l$ to each device $i \in \{1, 2, ..., N\}$ where $l$ is a specific layer. Here, $A_{t+1}^l$ is the full spatial context, and each device can individually produce its respective next timestep activation patch $A_t^{l,(i)}$ which is only $1/N$ of the original image. This way, for each device only a patch of the stale activation is updated, cohering with the rest of the unmodified stale activation image.

Next, the AllGather operation gathers all the freshly updated activation patches $A_t^{l,(i)}$, combining them to $A_t^l$ in order to repeat the previously mentioned Scatter operation.

Distrifusion overlaps the AllGather communication with computation. By asynchronously communicating the activation patches while continuing to perform computations, it effectively hides the communication overhead. This optimization is critical for achieving high performance without compromising latency.

To improve initial performance, Distrifusion uses synchronous patch operations for a few steps at the

start before switching to displaced patch parallelism. This ensures accurate patch cohesion at the start of the denoising process.

# 5 Conclusion and Future Work

In this survey, we explored state-of-the-art techniques aimed at accelerating diffusion models, focusing on quantization, knowledge distillation, and distributed parallel sampling. While these approaches have achieved significant progress in reducing the inference time of diffusion models, there remain opportunities for further enhancement.

As a future direction, the survey could be expanded to include a comprehensive benchmarking of the discussed techniques under common experimental settings. By standardizing key parameters such as the model architecture, dataset, and evaluation metrics, we can better understand the relative strengths and trade-offs of each approach. Furthermore, assessing the acceleration techniques not only quantitatively, in terms of inference speed and resource utilization, but also qualitatively by comparing the fidelity and diversity of generated samples, would provide a more holistic evaluation of these methods. These steps would serve to deepen the understanding of diffusion model acceleration and guide the development of more robust and efficient solutions.

# References


[1] Geoffrey Hinton. "Distilling the Knowledge in a Neural Network". In: *arXiv preprint arXiv:1503.02531* (2015).

[2] Jiaxiang Wu et al. "Quantized convolutional neural networks for mobile devices". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4820–4828.

[3] Guobin Chen et al. "Learning efficient object detection models with knowledge distillation". In: *Advances in neural information processing systems* 30 (2017).

[4] Dipankar Das et al. "Mixed precision training of convolutional neural networks using integer operations". In: *arXiv preprint arXiv:1802.00930* (2018).

[5] Saihui Hou et al. "Lifelong learning via progressive distillation and retrospection". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 437–452.

[6] Raghuraman Krishnamoorthi. "Quantizing deep convolutional networks for efficient inference: A whitepaper". In: *arXiv preprint arXiv:1806.08342* (2018).

[7] Jun Haeng Lee et al. "Quantization for rapid deployment of deep neural networks". In: *arXiv preprint arXiv:1810.05488* (2018).

[8] Yiren Zhou et al. "Adaptive quantization for deep neural network". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[9] Ron Banner, Yury Nahshan, and Daniel Soudry. "Post training 4-bit quantization of convolutional networks for rapid-deployment". In: *Advances in Neural Information Processing Systems* 32 (2019).

[10] Ron Banner, Yury Nahshan, and Daniel Soudry. "Post training 4-bit quantization of convolutional networks for rapid-deployment". In: *Advances in Neural Information Processing Systems* 32 (2019).

[11] Alexander Finkelstein, Uri Almog, and Mark Grobman. "Fighting quantization bias with bias". In: *arXiv preprint arXiv:1906.03193* (2019).

[12] Markus Nagel et al. “Data-free quantization through weight equalization and bias correction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1325–1334.

[13] Markus Nagel et al. “Data-free quantization through weight equalization and bias correction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1325–1334.

[14] Nanxin Chen et al. “Wavegrad: Estimating gradients for waveform generation”. In: *arXiv preprint arXiv:2009.00713* (2020).

[15] Jun Fang et al. “Post-training piecewise linear quantization for deep neural networks”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer. 2020, pp. 69–86.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[17] Animesh Jain et al. “Efficient execution of quantized deep learning models: A compiler approach”. In: *arXiv preprint arXiv:2006.10226* (2020).

[18] Markus Nagel et al. “Up or down? adaptive rounding for post-training quantization”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7197–7206.

[19] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising diffusion implicit models”. In: *arXiv preprint arXiv:2010.02502* (2020).

[20] Yang Song and Stefano Ermon. “Improved techniques for training score-based generative models”. In: *Advances in neural information processing systems* 33 (2020), pp. 12438–12448.

[21] Yang Song et al. “Score-based generative modeling through stochastic differential equations”. In: *arXiv preprint arXiv:2011.13456* (2020).

[22] Ali Hadi Zadeh et al. “Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference”. In: *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE. 2020, pp. 811–824.

[23] Prafulla Dhariwal and Alexander Nichol. “Diffusion models beat gans on image synthesis”. In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.

[24] Prafulla Dhariwal and Alexander Nichol. “Diffusion models beat gans on image synthesis”. In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.

[25] Edward J Hu et al. “Lora: Low-rank adaptation of large language models”. In: *arXiv preprint arXiv:2106.09685* (2021).

[26] Tailin Liang et al. “Pruning and quantization for deep neural network acceleration: A survey”. In: *Neurocomputing* 461 (2021), pp. 370–403.

[27] Shitong Luo and Wei Hu. “Diffusion probabilistic models for 3d point cloud generation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 2837–2845.

[28] Markus Nagel et al. “A white paper on neural network quantization”. In: *arXiv preprint arXiv:2106.08295* (2021).

[29] Alex Nichol et al. “Glide: Towards photorealistic image generation and editing with text-guided diffusion models”. In: *arXiv preprint arXiv:2112.10741* (2021).

[30] Alex Nichol et al. “Glide: Towards photorealistic image generation and editing with text-guided diffusion models”. In: *arXiv preprint arXiv:2112.10741* (2021).

[31] Vadim Popov et al. “Grad-tts: A diffusion probabilistic model for text-to-speech”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8599–8608.

[32] Arash Vahdat, Karsten Kreis, and Jan Kautz. “Score-based generative modeling in latent space”. In: *Advances in neural information processing systems* 34 (2021), pp. 11287–11302.

[33] Han Cai et al. “Enable deep learning on mobile devices: Methods, systems, and applications”. In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 27.3 (2022), pp. 1–50.

[34] Amir Gholami et al. “A survey of quantization methods for efficient neural network inference”. In: *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[35] Shansan Gong et al. “Diffuseq: Sequence to sequence text generation with diffusion models”. In: *arXiv preprint arXiv:2210.08933* (2022).

[36] Jonathan Ho and Tim Salimans. “Classifier-free diffusion guidance”. In: *arXiv preprint arXiv:2207.12598* (2022).

[37] Jonathan Ho et al. “Imagen video: High definition video generation with diffusion models”. In: *arXiv preprint arXiv:2210.02303* (2022).

[38] Bahjat Kawar et al. “Denoising diffusion restoration models”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 23593–23606.

[39] Haoying Li et al. “Srdiff: Single image super-resolution with diffusion probabilistic models”. In: *Neurocomputing* 479 (2022), pp. 47–59.

[40] Xiang Li et al. “Diffusion-lm improves controllable text generation”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 4328–4343.

[41] C Lu et al. “A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps”. In: *Proc. Adv. Neural Inf. Process. Syst., New Orleans, United States* (2022), pp. 1–31.

[42] Andreas Lugmayr et al. “Repaint: Inpainting using denoising diffusion probabilistic models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11461–11471.

[43] Ben Poole et al. “Dreamfusion: Text-to-3d using 2d diffusion”. In: *arXiv preprint arXiv:2209.14988* (2022).

[44] Aditya Ramesh et al. “Hierarchical text-conditional image generation with clip latents”. In: *arXiv preprint arXiv:2204.06125* 1.2 (2022), p. 3.

[45] Aditya Ramesh et al. “Hierarchical text-conditional image generation with clip latents”. In: *arXiv preprint arXiv:2204.06125* 1.2 (2022), p. 3.

[46] Robin Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[47] Robin Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[48] Chitwan Saharia et al. “Image super-resolution via iterative refinement”. In: *IEEE transactions on pattern analysis and machine intelligence* 45.4 (2022), pp. 4713–4726.

[49] Chitwan Saharia et al. “Photorealistic text-to-image diffusion models with deep language understanding”. In: *Advances in neural information processing systems* 35 (2022), pp. 36479–36494.

[50] Chitwan Saharia et al. “Photorealistic text-to-image diffusion models with deep language understanding”. In: *Advances in neural information processing systems* 35 (2022), pp. 36479–36494.

[51] Tim Salimans and Jonathan Ho. “Progressive distillation for fast sampling of diffusion models”. In: *arXiv preprint arXiv:2202.00512* (2022).

[52] Uriel Singer et al. “Make-a-video: Text-to-video generation without text-video data”. In: *arXiv preprint arXiv:2209.14792* (2022).

[53] Hyungjoo Chae et al. “Dialogue chain-of-thought distillation for commonsense-aware conversational agents”. In: *arXiv preprint arXiv:2310.09343* (2023).

[54] Yuwei Guo et al. “Animatediff: Animate your personalized text-to-image diffusion models without specific tuning”. In: *arXiv preprint arXiv:2307.04725* (2023).

[55] Yefei He et al. “Efficientdm: Efficient quantization-aware fine-tuning of low-bit diffusion models”. In: *arXiv preprint arXiv:2310.03270* (2023).

[56] Xiuyu Li et al. “Q-diffusion: Quantizing diffusion models”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 17535–17545.

[57] Chen-Hsuan Lin et al. “Magic3d: High-resolution text-to-3d content creation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 300–309.

[58] Zhengxiong Luo et al. “Videofusion: Decomposed diffusion models for high-quality video generation”. In: *arXiv preprint arXiv:2303.08320* (2023).

[59] Chenlin Meng et al. “On distillation of guided diffusion models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 14297–14306.

[60] Yuzhang Shang et al. “Post-training quantization on diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 1972–1981.

[61] Shaoan Xie et al. “Smartbrush: Text and shape guided object inpainting with diffusion model”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 22428–22437.

[62] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. “Adding conditional control to text-to-image diffusion models”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 3836–3847.

[63] Cheng Chen, Christina Giannoula, and Andreas Moshovos. “Low-Bitwidth Floating Point Quantization for Efficient High-Quality Diffusion Models”. In: *arXiv preprint arXiv:2408.06995* (2024).

[64] Ciprian Corneanu, Raghudeep Gadde, and Aleix M Martinez. “Latentpaint: Image inpainting in latent space with diffusion models”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024, pp. 4334–4343.

[65] Yefei He et al. “Ptqd: Accurate post-training quantization for diffusion models”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[66] Yi-Ting Hsiao et al. “Plug-and-Play Diffusion Distillation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 13743–13752.

[67] Muyang Li et al. “Distrifusion: Distributed parallel inference for high-resolution diffusion models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 7183–7193.

[68] Yanjing Li et al. “Q-dm: An efficient low-bit quantized diffusion model”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[69] Xinyin Ma, Gongfan Fang, and Xinchao Wang. “Deepcache: Accelerating diffusion models for free”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 15762–15772.

[70] Andy Shih et al. “Parallel sampling of diffusion models”. In: *Advances in Neural Information Processing Systems* 36 (2024).