MODTRAIN: TRAINING MODULATED DIFFUSION MODELS FOR EXTREME LOW-BIT QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models have redefined generative AI, but their multi-step inference remains a critical bottleneck for efficient deployment. Quantization offers a powerful remedy, yet extremely low-bit activation quantization remains challenging even for state-of-the-art quantization-aware training (QAT), as its errors are difficult to absorb during training. Modulated diffusion models offer a promising path by exploiting temporal redundancy across denoising steps, but have so far been limited to post-training quantization. We present ModTrain, a training framework for modulated diffusion models. Training these models is non-trivial because historical states couple consecutive denoising steps, complicating data construction, gradient computation, and activation alignment. ModTrain addresses these issues with sequential forward sampling for efficient data construction, history-aware backpropagation for correct gradient flow, and full-precision alignment to mitigate activation distribution shift. ModTrain is orthogonal to existing QAT methods and integrates seamlessly with them. Experiments on LSUN and ImageNet show that ModTrain outperforms post-training MoDiff, reducing activation precision from 4 to 2 bits, and consistently improves QAT baselines at extremely low precision, with up to a 10.90 FID improvement. ModTrain advances extremely low-bit inference and may further motivate hardware support for such computation.

1 INTRODUCTION

Diffusion models have recently achieved remarkable success in generative tasks spanning images, text, and beyond (Ho et al., 2020; Sahoo et al., 2024). At a high level, they generate samples by progressively denoising Gaussian noise through a learned reverse process. Despite their strong performance, diffusion models often suffer from inefficient generation (Liu et al., 2022). On the one hand, producing a single high-quality sample can require hundreds to thousands of denoising steps, leading to substantial latency and energy consumption. On the other hand, diffusion backbones have grown increasingly large to improve generation quality, which further slows each denoising step (Rombach et al., 2022). Together, these factors hinder practical deployment in real-time settings and on resource-constrained devices such as mobile platforms and edge hardware.

Model quantization is a widely used technique for accelerating inference and reducing memory footprint by compressing model weights and activations into lower-bit representations (Nagel et al., 2021). However, activation quantization remains a key bottleneck, as activations often exhibit dynamic, long-tailed distributions that are difficult to represent accurately at low precision (Xiao et al., 2023). While weights can typically be quantized to 4 bits with little performance degradation, activations often require 8 bits. This gap persists even with training the quantized models: although training can adapt weights to compensate for weight quantization errors, low-bit activation quantization remains substantially harder to absorb (Liang et al., 2025; Chen et al., 2025). These activation quantization errors hinder models from being deployed at extremely low precision during inference, constraining the practical gains of aggressive quantization. Consequently, hardware support for very-low-bit inference remains limited, further restricting the end-to-end efficiency benefits of low-bit inference.

Recently, modulated quantization has been proposed as a promising approach that introduces a framework for diffusion models to alleviate the bottleneck of low-bit activation quantization (Gao et al., 2025). By exploiting the strong redundancy between consecutive generation steps, modulated diffusion models can complement existing quantization methods, enabling activation quantization

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

from 8 bits to 4 bits without sacrificing performance. It offers a novel research path toward extreme-low-bit quantization. However, prior work has only studied modulated quantization in the PTQ setting, and how to train modulated diffusion models remains underexplored. Designing a dedicated training pipeline for modulated quantization could further unlock its potential for lower-bit quantization.

Table 1: Comparison of different diffusion quantization paradigms.

Method	Training Aware	History Aware	Full-Precision Performance
PTQ	✗	✗	✓
MoDiff	✗	✓	✓
QAT	✓	✗	✗
ModTrain	✓	✓	✓

Our preliminary analysis in Section 3.1 reveals that training modulated diffusion models is non-trivial due to their unique forward computation: modulated diffusion models require historical states produced by neighboring denoising steps, while standard quantization-aware training (QAT) (Li et al., 2023b) only optimizes a quantized denoiser at each timestep independently. It triggers three challenges to the existing training paradigm. First, efficiently constructing training data is challenging, since these historical states are generated iteratively during inference, and naively unrolling the full reverse process during training is computationally prohibitive. Second, the introduction of historical states leads to a complex computation graph, and careful backpropagation of caches is required for correct optimization. Finally, our analysis indicates that QAT compromises performance under full-precision activations, causing distribution shift between caches of training and inference. In short, existing QAT methods are insufficient for modulated diffusion models, and a dedicated history-aware training pipeline is needed for the new family of diffusion models.

In this work, we propose Modulated Diffusion Training (ModTrain), a novel training framework for modulated diffusion models. Specifically, we develop a sequential forward sampling strategy to construct training data efficiently without unrolling the generation process. We then introduce a backpropagation scheme tailored to modulated diffusion models that preserves gradient flow through historical states and avoids history misalignment in optimization. Finally, we propose a full-precision alignment loss to avoid activation distribution shift between training and generation. We summarize the advantages of ModTrain in Table 1. Notably, ModTrain is orthogonal to existing QAT techniques and can be seamlessly combined with them. We conduct extensive experiments on LSUN-Churches, LSUN-Bedrooms, and ImageNet using several QAT baselines. The results show that ModTrain substantially outperforms untrained modulated diffusion in extremely low-bit settings, reducing the activation precision from 4 bits to 2 bits. Moreover, ModTrain improves the generation quality of QAT baselines, achieving consistently better performance under low-bit quantization. For example, it reduces FID from 19.93 to 9.03 in the W1A2 setting on LSUN-Churches. In a nutshell, ModTrain takes an important step toward extremely low-bit inference and may further motivate the development of hardware support for such low-precision computation.

2 Background and Related Work

Diffusion Models. Diffusion models have become a cornerstone of generative modeling, achieving remarkable success across diverse domains such as image synthesis, data distillation, and molecular modeling (Ho et al., 2020; Hoogeboom et al., 2022; Su et al., 2024). These models operate on an iterative framework that involves adding noise in the forward process and learning to remove it during the reverse process (Dhariwal & Nichol, 2021). To be specific, diffusion models consist of the forward process and the backward process. Using Denoising Diffusion Probabilistic Models (DDPMs) as an example (Ho et al., 2020), the forward process incrementally adds noise to the image at each step, gradually transforming the data distribution into a standard Gaussian distribution. We can efficiently sample training data $\boldsymbol{x}_t$ from $\boldsymbol{x}_0$ in closed form: $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = N(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$, where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$. Meanwhile, the reverse process progressively denoises the Gaussian distribution, reconstructing the original image distribution step by step with a denoising network $\epsilon_\theta(\mathbf{x}_t, t)$, where the denoising neural network $\epsilon_\theta(\mathbf{x}_t, t)$ predicts the noise ϵ added to $\mathbf{x}_0$ to obtain $\mathbf{x}_t$. The training loss is a variant of the variational lower bound (ELBO) (Kingma & Welling, 2013): $\mathcal{L}_{\text{DM}} = \mathbf{E}_{\boldsymbol{x}_0,\epsilon,t}\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\epsilon, t)\|_2^2$, where ϵ is sampled from a Gaussian distribution.

However, generating samples requires predicting noise across T steps with a heavy backbone ϵ_θ, making the diffusion process computationally expensive for practical applications (Song et al., 2021b; Ho et al., 2020). Some works propose using efficient samplers for generation, which require fewer sampling steps, such as denoising diffusion implicit models (Song et al., 2021a).

Quantization. Quantization aims to reduce inference costs by converting floating-point values into low-bit integers using scaling factors (Nagel et al., 2021; Yang et al., 2019), which is widely applied to accelerate the generation process of diffusion models. Post-training quantization (PTQ) learns only these scaling factors after training Li et al. (2021), and several studies, such as Q-Diffusion and SVDQuant, have applied PTQ to diffusion models (Li et al., 2023a; 2024). Another line of work explores quantization-aware training (QAT), which incorporates the quantization process directly into training, allowing model parameters to adapt to quantization effects (Nagel et al., 2022). Compared to PTQ, QAT enables diffusion models to operate at lower bit widths for both weights and activations, since model weights are explicitly adapted to quantization effects during training. For instance, Quest develops a training procedure with Hardamard rotation (Panferov et al., 2025).

For both PTQ and QAT, uniform quantization is the most commonly used quantization method, where floating-point values are mapped to integers with uniform interval sizes (Nagel et al., 2021). Given a vector $\boldsymbol{x}$, we denote its integer representation as $\boldsymbol{x}_{\text{int}}$ and its dequantized counterpart as $Q(\boldsymbol{x})$. Taking asymmetric quantization as an example, the quantization process can be written as $\boldsymbol{x}_{\text{int}} = \text{clamp}(\lfloor \boldsymbol{x}/s \rceil + \boldsymbol{z}; 0, 2^b - 1)$ and $Q(\boldsymbol{x}) = s(\boldsymbol{x}_{\text{int}} - \boldsymbol{z})$, where b denotes the quantization bitwidth, and $\text{clamp}(\cdot)$ clips values to the specified integer range. The parameters s and $\boldsymbol{z}$ denote the scale factor and zero point, respectively. For symmetric quantization, the integer range is defined as $[-2^{b-1}, 2^{b-1} - 1]$. In deep learning quantization, weights are typically quantized using symmetric quantization, while activations are quantized using asymmetric quantization (Li et al., 2023a).

Despite their effectiveness, quantization methods in diffusion models face a major bottleneck in activation quantization, as activations exhibit dynamic, long-tailed distributions across timesteps. Modulated quantization (Gao et al., 2025) addresses this challenge by exploiting similarities between generation steps and avoiding error accumulation across generation steps by error accumulation. However, MoDiff's efficiency gains are currently limited to PTQ, as it requires historical states during training, which existing quantization training methods cannot readily accommodate. We note that our method is orthogonal to existing QAT and can be combined with them for training MoDiff.

3 Modulated Diffusion Training

In this section, we propose Modulated Diffusion Training (ModTrain), which is a novel training method for diffusion models. We begin by discussing the challenges of training modulated diffusion models. We then present our training techniques in detail, including training data sampling, backpropagation, and loss computation. Finally, we provide theoretical analyses of ModTrain.

3.1 Challenges of Training Modulated Diffusion Models

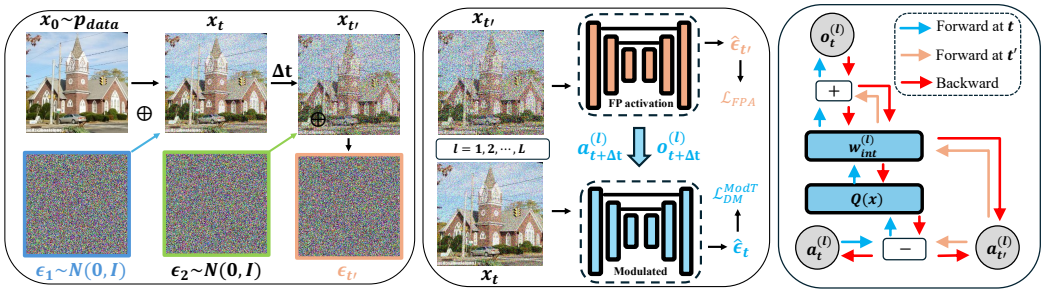
We begin with introducing modulated diffusion models, which provide a general framework that enhances PTQ methods to address these activation quantization challenges (Gao et al., 2025). Let $\mathcal{A}^{(l)}(\cdot)$ denote the l-th linear operator in the denoising network, such as the linear and convolutional layers, where $l \in \{1, 2, \ldots, L\}$. We use $\boldsymbol{w}^{(l)}$ to denote the weight tensor of $\mathcal{A}^{(l)}$, and $\boldsymbol{w}_{\text{int}}^{(l)}$ to denote its quantized counterpart $Q_{\boldsymbol{w}}^{(l)}(\boldsymbol{w}^{(l)})$. We denote the input and output for $\mathcal{A}^{(l)}(\cdot)$ at time step t as $\mathbf{a}_t^{(l)}$ and $\boldsymbol{o}_t^{(l)} = \mathcal{A}^{(l)}(\boldsymbol{a}_t^{(l)})$, respectively. As shown in Eq. 1 and 2, we quantize the differences between nearby timesteps, where $t = T-1, \ldots, 1$. $\hat{\boldsymbol{a}}_t^{(l)}$ is an intermediate variable used to prevent error accumulation during generation by guaranteeing $\hat{\boldsymbol{o}}_t^{(l)} = \mathcal{A}(\hat{\boldsymbol{a}}_t^{(l)})$. The process adopts a warm-up strategy, which is initialized with $\hat{\boldsymbol{a}}_T^{(l)} = \boldsymbol{a}_T^{(l)}$ and $\hat{\boldsymbol{o}}_T^{(l)} = \mathcal{A}^{(l)}(\boldsymbol{a}_T^{(l)})$ for each layer. Please refer to Appendix B for more details. While effective, this design reformulates the forward computation of the linear operator $\mathcal{A}^{(l)}$: instead of depending only on the current input, the computation also requires historical states, namely the previous modulated input $\hat{\boldsymbol{a}}_{t+1}^{(l)}$ and output $\hat{\boldsymbol{o}}_{t+1}^{(l)}$. As a result, training modulated diffusion models becomes history-dependent, introducing challenges in both training data collection and optimization.

$$\hat{\boldsymbol{a}}_t^{(l)} = Q^{(l)}(\boldsymbol{a}_t^{(l)} - \hat{\boldsymbol{a}}_{t+1}^{(l)}) + \hat{\boldsymbol{a}}_{t+1}^{(l)}, \tag{1}$$

$$\hat{\boldsymbol{o}}_t^{(l)} = \mathcal{A}^{(l)}(Q^{(l)}(\boldsymbol{a}_t^{(l)} - \hat{\boldsymbol{a}}_{t+1}^{(l)})) + \hat{\boldsymbol{o}}_{t+1}^{(l)} \tag{2}$$

$$= \mathcal{A}^{(l)}(\hat{\boldsymbol{a}}_t^{(l)}) \approx \boldsymbol{o}_t^{(l)}, \tag{3}$$

First, it is challenging to efficiently collect training data. The historical states $\hat{\boldsymbol{o}}_{t+1}$ and $\hat{\boldsymbol{a}}_{t+1}$ in Eq. 1 and 2 are generated iteratively during inference. However, unrolling the entire generation process from Gaussian noise $\boldsymbol{x}_T$ to obtain a pair of training data is computationally prohibitive. Therefore,



(a) Sequential Forward Sampling (b) Training Loss Computation (c) Backpropagation

Figure 1: **Illustration of ModTrain.** (a) We first follow the standard forward diffusion process to sample $\mathbf{x}_t$ using noise ϵ_1. We then obtain $\mathbf{x}_{t'}$ by adding additional noise to $\mathbf{x}_t$. $\mathbf{x}_{t'}$ is used to compute the historical states for $\mathbf{x}_t$. (b) The pair $(\mathbf{x}_t, \mathbf{x}_{t'})$ obtained from sequential sampling is used to compute the training loss. Both $\mathbf{x}_t$ and $\mathbf{x}_{t'}$ are passed through the network with the same parameters to predict their corresponding noises $\hat{\epsilon}_t$ and $\hat{\epsilon}_{t'}$. (c) The forward computation consists of two branches: the quantized branch, indicated by blue arrows, and the historical branch, indicated by orange arrows. The gradient is backpropagated along the red arrows.

an efficient training data sampling strategy to simulate the historical states $\hat{\boldsymbol{o}}_{t+1}$ and $\hat{\boldsymbol{a}}_{t+1}$ is required. Second, the introduction of historical states leads to a more complex computation graph, and careful design of the backpropagation is required to ensure correct optimization. Naively excluding these historical states from the computation graph results in misalignment between them. Taking a linear layer $\mathcal{A}^{(l)}$ parameterized by $\boldsymbol{w}_{\text{int}}^{(l)}$ as an example, given historical states $\hat{\boldsymbol{a}}_{t+1}^{(l)}$ and $\hat{\boldsymbol{o}}_{t+1}^{(l)}$, any update $\Delta \boldsymbol{w}$ applied to $\boldsymbol{w}_{\text{int}}^{(l)}$ will cause inconsistency, i.e., $\hat{\boldsymbol{o}}_{t+1}^{(l)} \neq (\boldsymbol{w}_{\text{int}}^{(l)} + \Delta \boldsymbol{w})\hat{\boldsymbol{a}}_{t+1}^{(l)}$, contradicting Eq. 3. It is essential to ensure correct optimization while keeping the historical states aligned in training modulated diffusion models between optimization steps.

3.2 Modulated Diffusion Training

We propose Modulated Diffusion Training (ModTrain), a novel training pipeline for modulated quantization with historical states. We first propose sequential forward sampling to efficiently generate pair-wise data for training modulated diffusion models We then describe how to perform backpropagation for training modulated diffusion models. Finally, we introduce a full-precision alignment strategy to avoid the cache distribution shift between training and generation.

Notation. To avoid confusion, we use $\boldsymbol{a}_{\text{hist}}^{(l)}$ and $\boldsymbol{o}_{\text{hist}}^{(l)}$ in place of $\hat{\boldsymbol{a}}_{t+1}^{(l)}$ and $\hat{\boldsymbol{o}}_{t+1}^{(l)}$ in Eq. 2 to denote the training-time historical states, since the training timestep may differ from the inference timestep. We eliminate the " $\hat{}$ " notation for the history because error compensation (Eq. 1) is not applied during training. We use $\epsilon_\theta^{\text{full}}(\boldsymbol{x}_t, t)$ to denote the forward computation of ϵ_θ with $\boldsymbol{o}_t^{(l)} = \mathcal{A}^{(l)}(\boldsymbol{a}_t^{(l)})$, where the weights are quantized while the activations remain in full precision. We use $\epsilon_\theta^{\text{mod}}(\boldsymbol{x}_t, t; \{\boldsymbol{a}_{\text{hist}}^{(l)}, \boldsymbol{o}_{\text{hist}}^{(l)}\})$ to denote the modulated forward computation of ϵ_θ in Eq. 2.

Sequential Forward Sampling. To efficiently simulate the history for training modulated diffusion models, we propose *Sequential Forward Sampling* (SFS), which performs pairwise sampling at each training step. Specifically, SFS samples $\boldsymbol{x}_{t'}$ to compute the historical states at timestep t' and samples $\boldsymbol{x}_t$ as the current training input at timestep t in a sequential manner. Given a clean image $\boldsymbol{x}_0$ and independent standard Gaussian noises ϵ_1 and ϵ_2, Eq. 4 follows the standard forward process, and Eq. 5 adds noise to $\boldsymbol{x}_t$ to match the noise level at timestep t'. We introduce a hyperparameter Δt to control the gap between the historical timestep and the training timestep, where $t' = t + \Delta t$ and $t = 0, 1, \cdots, T - \Delta t - 1$. The procedure of SFS is illustrated in Figure 1a, avoiding expensive unrolling generation for obtaining training data.

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon_1, \tag{4}$$

$$\boldsymbol{x}_{t'} = \sqrt{\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}\boldsymbol{x}_t + \sqrt{1 - \frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}\epsilon_2, \tag{5}$$

The sampled Gaussian noise ϵ_1 enables us to follow the standard diffusion model training pipeline by predicting added noise from noisy data. We first apply the standard forward computation to $\boldsymbol{x}_{t'}$ and cache $\boldsymbol{o}_{t'}^{(l)}$ and $\boldsymbol{a}_{t'}^{(l)}$ as the historical

$$\hat{\epsilon}_{t'} = \epsilon_\theta^{\text{full}}(\boldsymbol{x}_{t'}, t'), \tag{6}$$

$$\hat{\epsilon}_t = \epsilon_\theta^{\text{mod}}(\boldsymbol{x}_t, t; \{\boldsymbol{a}_{t'}^{(l)}, \boldsymbol{o}_{t'}^{(l)}\}). \tag{7}$$

states required for modulated computation in Eq. 6. We then apply modulated quantization to $\boldsymbol{x}_t$ and the historical states $\boldsymbol{o}_{t'}^{(l)}$ and $\boldsymbol{a}_{t'}^{(l)}$ to obtain the predicted noise $\hat{\epsilon}_t$ in Eq. 7. Note that the two forward computations in Eq. 6 and 7 share the same model parameters θ. We then follow the standard diffusion training objective to train our modulated diffusion model:

$$\mathcal{L}_{\text{DM}}^{\text{ModT}} = \mathbf{E}_{\boldsymbol{x}_0,\epsilon_1,\epsilon_2,t,t'} \|\epsilon_1 - \hat{\epsilon}_t\|_2^2. \tag{8}$$

Backpropagation. We design the backpropagation procedure for training modulated diffusion models by incorporating the historical states $\boldsymbol{o}_{t'}^{(l)}$ and $\boldsymbol{a}_{t'}^{(l)}$ into the computation graph. For a linear layer $\mathcal{A}^{(l)}$ parameterized by $\boldsymbol{w}_{\text{int}}^{(l)}$, we design the backpropagation in Eq. 9, 10, and 11 , where $\partial Q^{(l)}/\partial \boldsymbol{x}\big|_{\boldsymbol{a}}$ denotes the gradient of $Q^{(l)}(\boldsymbol{x})$ evaluated at $\boldsymbol{a}$, and the gradient computation depends on the specific QAT method employed. The key differences appear in Eq. 9 and Eq. 11. Unlike the naive gradient computation, our formulation introduces an additional backward path to $\boldsymbol{a}_{t'}^{(l)}$, allowing gradients to capture the dependence of the historical states on the current model parameters. This gradient design works jointly with SFS, which provides the necessary historical states without unrolling the entire generation trajectory into an excessively large computation graph. By incorporating the historical states into the computation graph, $\boldsymbol{a}_{t'}^{(l)}$ and $\boldsymbol{o}_{t'}^{(l)}$ are no longer treated as static variables, thereby avoiding the misalignment caused by regarding the history as fixed information. The backpropagation path through one modulated layer is shown in Figure 1c, where the gradient is passed to the historical variable $\boldsymbol{a}_{t'}^{(l)}$ through the red arrows. We can further validate the feasibility of our backpropagation through a thought experiment: when $Q^{(l)}(x)$ introduces no quantization error, Eq. 9 reduces to $\boldsymbol{a}_t^{(l)}$, which is consistent with full-precision activation model training, as expected. In practice, our backpropagation can be implemented directly using PyTorch AutoGrad by enabling gradient tracking in the history computation. Finally, we provide the derivation for convolutional layers in Appendix C.

$$\frac{\partial \boldsymbol{o}_t^{(l)}}{\partial \boldsymbol{w}_{\text{int}}^{(l)}} = Q^{(l)}(\boldsymbol{a}_t^{(l)} - \boldsymbol{a}_{t'}^{(l)}) + \boldsymbol{a}_{t'}^{(l)}, \tag{9}$$

$$\frac{\partial \boldsymbol{o}_t^{(l)}}{\partial \boldsymbol{a}_t^{(l)}} = \boldsymbol{w}_{\text{int}}^{(l)} \left.\frac{\partial Q^{(l)}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{a}_t^{(l)} - \boldsymbol{a}_{t'}^{(l)}}, \tag{10}$$

$$\frac{\partial \boldsymbol{o}_t^{(l)}}{\partial \boldsymbol{a}_{t'}^{(l)}} = \boldsymbol{w}_{\text{int}}^{(l)} \left(1 - \left.\frac{\partial Q^{(l)}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{a}_t^{(l)} - \boldsymbol{a}_{t'}^{(l)}}\right). \tag{11}$$

Full-Precision Alignment. The history-dependent forward process introduces another challenge: the distribution shift of the historical states $\boldsymbol{a}_{t'}^{(l)}$ and $\boldsymbol{o}_{t'}^{(l)}$. In SFS, all historical states are generated from the full-precision-activation model according to Eq. 6. During inference, however, the historical states are produced recursively by accumulating quantized outputs step by step. This creates a gap between the training and inference distributions. Moreover, standard QAT often encourages the model to specialize in quantized activations at the expense of degraded performance under full-precision activations, as shown in Appendix F. To address the performance degradation of the full-precision activation model, we propose *Full-Precision Alignment* (FPA) to preserve full-precision performance. Specifically, we first compute the corresponding noise at timestep t' via forward sampling in Eq. 12.

$$\epsilon_{t'} = \frac{\boldsymbol{x}_{t'} - \sqrt{\bar{\alpha}_{t'}}\boldsymbol{x}_0}{\sqrt{1-\bar{\alpha}_{t'}}}. \tag{12}$$

Eq. 13 applies the forward sampling process to compute the corresponding noise of timestep t'. In this way, we recycle the sample $\boldsymbol{x}_{t'}$, and align the full-precision activation performance with our FPA loss in Eq. 13. The final training loss of ModTrain is balanced with a hyperparameter λ as shown in Eq. 14.

$$\mathcal{L}_{\text{FPA}} = \mathbf{E}_{\boldsymbol{x}_0,\epsilon_1,\epsilon_2,t,t'} \|\epsilon_{t'} - \hat{\epsilon}_{t'}\|_2^2. \tag{13}$$

$$\mathcal{L}_{\text{ModT}} = (1-\lambda)\mathcal{L}_{\text{DM}}^{\text{ModT}} + \lambda \mathcal{L}_{\text{FPA}}. \tag{14}$$

By incorporating FPA, we aim to learn a representation space that simultaneously preserves full-precision activation performance and maintains robustness to quantization errors. From the quantization-error perspective, FPA preserves the edge case where activation quantization error vanishes. To be specific, when the activation quantizer is ideally optimized,

$$\hat{\boldsymbol{o}}_t^{(l)} = \mathcal{A}^{(l)}\Big(Q^{(l)}(\boldsymbol{a}_t^{(l)} - \hat{\boldsymbol{a}}_{t+1}^{(l)})\Big) + \mathcal{A}^{(l)}(\hat{\boldsymbol{a}}_{t+1}^{(l)}) = \mathcal{A}^{(l)}(\boldsymbol{a}_t^{(l)} - \hat{\boldsymbol{a}}_{t+1}^{(l)}) + \mathcal{A}^{(l)}(\hat{\boldsymbol{a}}_{t+1}^{(l)}) = \mathcal{A}^{(l)}(\boldsymbol{a}_t^{(l)}),$$

where the second equality is because of no activation quantization error. Thus, the model reduces to a weight-quantized model with full-precision activations, which is exactly preserved by FPA.

Sampling. The model trained by ModTrain is sampler-orthogonal, allowing it to be used with any efficient diffusion sampler, such as DDIM, DPM, and PLMs (Song et al., 2021a; Lu et al., 2022; Liu et al., 2022). We still adopt a warmup strategy following MoDiff, where the first sampling step is

inferred with full-precision activation. For all the following sampling steps, we follow Eq. (1) and (2) to do modulated forward computation. Our ModTrain is orthogonal to the QAT methods, meaning that we can improve those methods in extremely low bits instead of replacing them.

3.3 Theoretical Analysis

To illustrate how the proposed framework works, we present the following theoretical analysis to demonstrate that sequential forward sampling introduces smaller variance between neighboring training timesteps and that our backpropagation correctly optimizes the model with historical states.

Proposition 1. *Suppose we are given $\boldsymbol{x}_0$, independent Gaussian noises $\epsilon_1, \epsilon_2 \sim \mathcal{N}(0, \mathbf{I})$, and a diffusion process defined by $\{\alpha_t\}$. Let $\boldsymbol{x}_t^{\mathrm{FS}}$ and $\boldsymbol{x}_{t'}^{\mathrm{FS}}$ denote two samples produced by the standard forward diffusion process at timesteps t and t' given ϵ_1, ϵ_2, respectively. Let $\boldsymbol{x}_t^{\mathrm{SFS}}$ and $\boldsymbol{x}_{t'}^{\mathrm{SFS}}$ denote the corresponding samples generated by SFS according to Eq. 4 and Eq. 5. Then, for any $t' \in [t, T]$, we have: (1) $\boldsymbol{x}_{t'}^{\mathrm{SFS}} \sim N(\sqrt{\bar{\alpha}_{t'}}\boldsymbol{x}_0, (1-\bar{\alpha}_{t'})\mathbf{I})$ and (2) $Var(\boldsymbol{x}_t^{\mathrm{SFS}} - \boldsymbol{x}_{t'}^{\mathrm{SFS}}) < 2 - \bar{\alpha}_t - \bar{\alpha}_{t'}$*

Proposition 1 shows that SFS preserves the similarity between $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t'}$ while maintaining their marginal distributions, compared with drawing two independent samples from $\boldsymbol{x}_0$. Here, $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_1$ and $\boldsymbol{x}_{t'} = \sqrt{\bar{\alpha}_{t'}}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_{t'}}\epsilon_2$. By preserving the similarity between temporally adjacent samples, SFS better matches the distributions encountered during generation.

Proposition 2. *Consider a regression problem where the input data $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$, and the ground-truth target corresponding to $\boldsymbol{x}_2$ is $y \in \mathbb{R}$. Given a one-layer model: $f(\boldsymbol{x}_2, \boldsymbol{x}_1, \boldsymbol{w}) = \boldsymbol{w}(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1)) + \boldsymbol{w}\boldsymbol{x}_1$, where $\boldsymbol{w}$ denotes the model parameters initialized as $\boldsymbol{w}_0$, and Q is a dynamic quantizer. We aim to fit $\boldsymbol{x}_2$ to the target y using the MSE loss: $\mathcal{L} = \frac{1}{2}|y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w})|^2$.*

With gradient descent, indicated in Eq. 9 with learning rate η, the update of the optimization is:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \eta\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \boldsymbol{w}_{t-1} + \eta(y - f)(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{x}_1), \tag{15}$$

where $t \geq 1$ is the optimization step. Then we have: $|y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w}_t)|^2 \leq |y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w}_{t-1})|^2$.

Proposition 2 shows that, from an optimization perspective, using our backpropagation method to compute gradients for modulated diffusion models in gradient descent guides the model toward a local minimum. In contrast, treating the historical states as constants during optimization fails to preserve this dependency under certain conditions, as shown in Proposition 3 in Appendix A.

4 Experiments

In this section, we introduce the experimental results of our method. We begin with the experimental settings. We demonstrate that training modulated diffusion models improves generation performance compared with modulated diffusion models in the PTQ setting. Furthermore, when integrated with QAT, our method substantially outperforms QAT baselines. Finally, we conduct comprehensive ablation studies and some visualizations to show how our method works.

4.1 Experimental Settings

Datasets, Models, and Evaluations. We majorly evaluate the effectiveness of our ModTrain on LSUN-Church-Outdoor (256×256), LSUN-Bedrooms (256×256) datasets (Yu et al., 2015). To validate the effectiveness of ModTrain in conditional generation, we also evaluate our method on ImageNet 256×256 (Russakovsky et al., 2015). We use Latent Diffusion Models with downsampling factors of 4 and 8, referred to as LDM-4 (for Bedrooms) and LDM-8 (for Churches and ImageNet), respectively (Rombach et al., 2022). We also evaluate the performance on Diffusion Transformer (DiT XL/2) to show that our method works on the most advanced architecture. We use 200 sampling steps for both LDM-4 and LDM-8 during generation by default, and use 250 steps for DiT XL/2.

We assess generation quality using Fréchet Inception Distance (FID) (Heusel et al., 2017), and Sliced Fréchet Inception Distance (sFID) (Salimans et al., 2016). Without a specific claim, those metrics are computed based on 50,000 generated images. Additionally, we provide precision and recall measurements on LSUN dataset (Sajjadi et al., 2018). Following existing works (Gao et al., 2025;

Table 2: FID, sFID, Precision, and Recall for DDIM on LSUN compared to MoDiff (PTQ). The subscript "w" represents weight-only training.

Method	Bits (W/A)	LSUN-Churches 256x256				LSUN-Bedrooms 256x256			
		FID↓	sFID↓	Prec↑	Rec↑	FID↓	sFID↓	Prec↑	Rec↑
FP	32/32	4.36	11.86	63.26	49.50	3.97	9.45	64.08	45.12
DQ + MoDiff	4/4	4.56	11.78	63.25	49.23	25.44	73.56	46.96	22.85
Q-Diff + MoDiff	4/4	4.58	11.55	64.21	48.97	7.99	29.54	53.90	35.02
LSQ_w+MoDiff	4/4	4.72	11.11	61.25	48.52	18.03	64.56	50.44	32.35
LSQ + ModTrain	4/4	4.69	11.13	60.91	48.49	4.02	9.64	62.47	45.31
DQ + MoDiff	3/3	47.52	24.55	45.63	20.58	78.92	142.54	30.21	18.77
Q-Diff + MoDiff	3/3	43.68	23.79	45.98	21.24	54.68	105.66	36.38	21.83
LSQ_w+MoDiff	3/3	5.61	11.25	58.27	47.00	60.92	120.57	35.22	19.72
LSQ + ModTrain	3/3	5.91	11.24	58.67	46.59	4.61	9.97	60.48	42.83
DQ + MoDiff	2/2	142.21	100.35	25.24	10.59	158.95	200.43	18.94	11.86
Q-Diff + MoDiff	2/2	102.43	87.74	29.97	15.23	140.74	180.59	23.76	15.19
LSQ_w+MoDiff	2/2	62.71	36.14	40.88	8.92	159.63	198.87	19.58	14.05
LSQ + ModTrain	2/2	7.85	12.88	57.13	40.78	5.78	13.78	58.87	39.88
$BinaryDM_w$+MoDiff	1/4	6.02	11.68	57.25	44.80	20.31	69.53	36.43	28.74
BinaryDM + ModTrain	1/4	6.56 [1]	11.51	56.75	44.94	7.80	12.46	58.45	40.01
$BinaryDM_w$+MoDiff	1/3	6.53	12.01	57.59	43.01	64.23	129.87	30.57	24.33
BinaryDM + ModTrain	1/3	7.69	12.32	56.54	42.65	8.87	14.04	55.21	39.76
$BinaryDM_w$+MoDiff	1/2	76.88	42.14	40.87	5.92	174.56	200.13	19.42	13.07
BinaryDM + ModTrain	1/2	9.03	14.21	55.49	39.42	11.78	13.26	56.50	38.22

Wang et al., 2024), we evaluate efficiency by measuring the number of binary operations (Bops) per denoising step for a single image with the help of DeepSpeed (Song et al., 2023) in Appendix G.

Quantization Methods. We build ModTrain on top of two QAT baselines for diffusion models: (1) Learnable Step Size Quantization (LSQ) Esser et al. (2019); He et al. (2023), which serves as the base method in much of the existing literature, and (2) BinaryDM (Zheng et al., 2024), a recent method for diffusion models with binary weights and low-precision activations. We refer to the baselines as **LSQ** and **BinaryDM**, and to our implementations as **LSQ+ModTrain** and **BinaryDM+ModTrain**. More details about the baselines and implementation are provided in Appendix D and E.

To demonstrate that ModTrain outperforms post-training approaches, we compare it with two MoDiff-based PTQ methods. Specifically, following MoDiff, we adopt dynamic quantization and Q-Diffusion (Li et al., 2023a) as baselines, denoted as **DQ+MoDiff** and **Q-Diff+MoDiff**, respectively. To isolate the impact of weight quantization, we also run experiments where QAT methods quantize only the weights during training, and then apply MoDiff with dynamic quantization to post-process activations. We denote these training methods with a subscript "w" to indicate weight-only training, followed by MoDiff. For example, **LSQ_w+MoDiff** denotes applying MoDiff to activations on top of the model with LSQ-trained weights. Finally, we report the full-precision model performance as a reference, denoted as **FP**. As for the quantization hyperparameters, we align weight and activation bitwidths and select them from $\{2, 3, 4\}$ for LSQ. For BinaryDM, we use binary weights and select the activation bitwidth from $\{2, 3, 4\}$. For notational simplicity, we adopt the format **"W/A"**, where "W" denotes the weight precision and "A" denotes the activation precision, respectively.

4.2 Main Results

In the main experiments, we evaluate generation quality under different quantization precision. We first compare ModTrain with MoDiff (PTQ) and the results are reported in Table 2 and Table 4. Since DQ and Q-Diff fail to generate meaningful images with binary weights, we report only the comparison

[1] At higher precision, QAT may not perform as well as PTQ methods as the experiments show in some cases. This is consistent with prior QAT work (Li et al., 2023b; Zheng et al., 2024): training a quantized model can be more challenging than training its FP counterpart, and accurately matching FP performance is difficult. Our goal is to provide a general framework for training modulated diffusion models, rather than to close the remaining gap to full precision. Moreover, these higher-precision settings are not particularly challenging in efficient works, and the minor performance difference does not affect the overall conclusions for efficiency-focused research.

Table 3: FID, sFID, Precision, and Recall for DDIM on LSUN compared to QAT baselines.

Method	Bits (W/A)	LSUN-Churches 256x256				LSUN-Bedrooms 256x256			
		FID↓	sFID↓	Prec↑	Rec↑	FID↓	sFID↓	Prec↑	Rec↑
FP	32/32	4.36	11.86	63.26	49.50	3.97	9.45	64.08	45.12
LSQ	4/32	4.63	11.00	62.15	48.94	4.01	9.55	62.43	45.22
LSQ	4/4	5.31	11.60	59.35	47.64	4.02	10.17	58.97	43.11
LSQ+ModTrain	4/4	4.69	11.13	60.91	48.49	4.02	9.64	62.47	45.31
LSQ	3/32	4.69	11.10	62.42	48.22	4.15	9.87	61.76	45.00
LSQ	3/3	6.74	12.26	56.62	44.35	5.22	11.21	58.90	39.24
LSQ+ModTrain	3/3	5.91	11.24	58.67	46.59	4.61	9.97	60.48	42.83
LSQ	2/32	5.82	12.96	62.63	45.00	4.43	13.75	60.99	42.89
LSQ	2/2	14.09	18.32	49.48	32.55	15.43	20.08	51.27	33.65
LSQ+ModTrain	2/2	7.85	12.88	57.13	40.78	5.78	13.78	58.87	39.88
BinaryDM	1/32	6.62	11.69	57.63	45.59	7.79	12.85	58.99	40.16
BinaryDM	1/4	8.15	13.59	56.50	39.76	9.25	14.70	57.25	36.28
BinaryDM+ModTrain	1/4	6.56	11.51	56.75	44.94	7.80	12.46	58.45	40.01
BinaryDM	1/3	9.45	14.99	54.77	36.57	13.06	16.90	49.98	38.34
BinaryDM+ModTrain	1/3	7.69	12.32	56.54	42.65	8.87	14.04	55.21	39.76
BinaryDM	1/2	19.93	22.24	47.10	25.61	18.22	21.28	45.29	28.74
BinaryDM+ModTrain	1/2	9.03	14.21	55.49	39.42	11.78	13.26	56.50	38.22

between **BinaryDM+MoDiff** and **BinaryDM+ModTrain**. We then evaluate how ModTrain enhances QAT methods by comparing the results of QAT baselines before and after applying ModTrain, as shown in Table 3. According to the results, we draw the following conclusions.

Improvement over PTQ methods. As shown in Table 2, our method preserves high generation quality and significantly outperforms PTQ, especially at lower activation precision. Specifically, DQ+MoDiff and Q-Diff+MoDiff produce meaningful samples only in the W4A4 setting. The performance degrades sharply at W3A3, where the FID on Churches worsens to 43.68. Even when isolating weight effects, LSQ+MoDiff still performs poorly, yielding an FID of 62.71 at W2A2. In contrast, ModTrain maintains strong generation quality at W2A2, achieving an FID of 7.85. Finally, we show that ModTrain is also effective for class-conditional generation, as illustrated in Table 4.

Improvement over QAT baselines. As shown in Table 3, our method also improves generation quality over the QAT baselines across different precisions. For instance, on the Churches dataset with the LSQ baseline, ModTrain more closely matches the full-precision activation performance at W4A4 than naïve LSQ. In the extreme W2A2 setting, ModTrain reduces the FID from 14.09 to 7.85, where the full-precision has a FID of 4.36, demonstrating a significant improvement in generation quality. It suggests that ModTrain effectively mitigates the generation loss observed at extremely low bit widths, bringing performance close to the lossless setting.

Table 4: FID and sFID for DDIM on ImageNet compared to MoDiff (PTQ). The subscript "w" represents weight-only training.

Method	Bits (W/A)	FID↓	sFID↓
FP	32/32	10.91	7.67
LSQ_w+MoDiff	4/4	10.99	7.71
LSQ + ModTrain	4/4	10.90	7.70
LSQ_w+MoDiff	3/3	12.31	8.90
LSQ + ModTrain	3/3	11.02	7.92
LSQ_w+MoDiff	2/2	73.61	41.24
LSQ + ModTrain	2/2	13.23	9.05

Table 5: FID using 5,000 images for DiT XL/2 on ImageNet compared to LSQ.

Method	Bits (W/A)	IS ↑	FID↓
FP	32/32	278.58	8.81
LSQ	4/4	160.77	13.19
LSQ + ModTrain	4/4	249.96	9.31

Broad Compatibility Across Baselines and Architectures. The comparisons with both PTQ and QAT baselines show that ModTrain consistently improves generation quality across multiple quantization methods and datasets at extremely low bits. Moreover, the results in Table 5 show that our method also improves the performance of DiT-XL/2 on ImageNet 256×256 under W4A4 quantization based on LSQ with 50,000 training iterations. This indicates that ModTrain is orthogonal to existing quantization techniques and sufficiently general to be effectively combined with them.

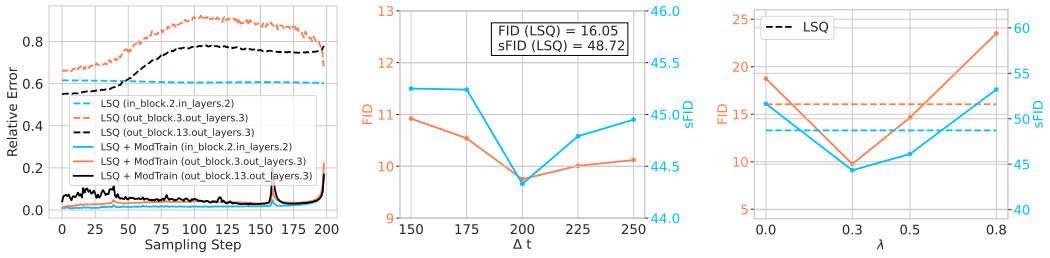
4.3 Ablation Study

We conduct comprehensive ablation studies in this section. We begin with a quantization error analysis to explain how our method achieves better generation quality. We then evaluate the effectiveness of sequential forward sampling, followed by a study of full-precision alignment. We provide visualizations of generated images to illustrate how ModTrain outperforms the baselines in Appendix H and the kernel efficiency analysis in Appendix G. Due to computational constraints, all measurements in ablation results are reported using $5{,}000$ generated images.

Quantization Error Analysis. To illustrate how ModTrain improves quantization, we visualize the relative activation quantization error of selected layers: $\|Q(\boldsymbol{x}) - \boldsymbol{x}\|_2^2 / \|\boldsymbol{x}\|_2^2$. We compare W2A2 models trained with LSQ and LSQ+ModTrain on LSUN-Churches. As shown in Figure 2a, ModTrain yields substantially smaller activation quantization error than the baseline, indicating that ModTrain pushes the quantized model toward the full-precision activation regime across all timesteps. For instance, the relative quantization error of layer *in_block.2.in_layers.2* (blue curves) drops from about 0.6 to below 0.2 across timesteps with ModTrain.

Effectiveness of Sequential Forward Sampling. To demonstrate the effectiveness of SFS, we use LSQ + ModTrain to train a W2A2 model on Churches and study how Δt affects generation quality. Since there are quantization errors during generation, we find that adopting a relatively large Δt will simulate the quantization error accumulated across time steps during training, which avoids distribution shift between training and generation. As shown in Figure 2b, both FID and sFID improve steadily compared to LSQ, and performance is not sensitive to Δt within a reasonable range. For instance, the FID ranges from 9.75 to 10.92 across a wide range of Δt selections (from 150 to 250), significantly outperforming LSQ, which achieves an FID of 16.05.

Effectiveness of Full-Precision Alignment. To evaluate how FPA affects performance, we use ModTrain (built on LSQ) to train a W2A2 model on Churches with different values of λ. As shown in Figure 2c, incorporating FPA is essential for improving generation quality. In particular, we achieve better quality than LSQ when $\lambda \in [0.3, 0.5]$, whereas performance degrades without FPA ($\lambda = 0$). Moreover, we provide visualizations of how our FPA improves generation quality in Appendix F.



(a) (b) (c)

Figure 2: (a) Comparison of quantization errors $\|Q(\boldsymbol{x}) - \boldsymbol{x}\|_2^2 / \|\boldsymbol{x}\|_2^2$ in selected layers between LSQ and LSQ+ModTrain. The dashed lines denote LSQ, while the solid lines denote LSQ+ModTrain. (b) FID and sFID on LSUN-Churches with different Δt. The orange curve denotes FID, the blue one denotes sFID, while the results of the baseline LSQ is shown in the legends. (c) FID and sFID on LSUN-Churches with different λ. The orange curve denotes FID, and the blue one denotes sFID.

5 Conclusion

In this paper, we propose ModTrain, a novel training framework orthogonal to QAT methods for modulated diffusion models. Our preliminary studies show that training modulated diffusion models faces both efficiency and optimization challenges. To address these issues, we introduce sequential forward sampling to efficiently construct training data, a tailored backpropagation scheme, and full-precision alignment to mitigate the distribution shift between training and generation. Experimental results show that ModTrain consistently outperforms MoDiff under PTQ settings at low bits and further improves the performance of several QAT baselines. Despite these gains, ModTrain has two main limitations. First, its training hyperparameters still require task-specific tuning. Second, we have not yet evaluated ModTrain in hardware-accelerated deployments. We leave principled approaches for hyperparameter selection and experiments on hardware acceleration as future work.

6 AI USE STATEMENT.

In this work, we used generative AI tools to improve the clarity and readability of the manuscript and to identify potential logical inconsistencies in our writing and arguments. The tools were not used to generate research hypotheses, theoretical claims, proofs, methodology, experimental results, or interpretations of results. All AI-assisted suggestions were critically reviewed by the authors, and any revisions were independently assessed for correctness and consistency with the research. We take full responsibility for the final content of this work.

7 ETHICS STATEMENT.

Enhancing the efficiency of diffusion models can facilitate the broader adoption of generative AI, benefiting both AI research and hardware development. We do not foresee any significant ethical concerns associated with this approach.

REFERENCES

Mengzhao Chen, Chaoyi Zhang, Jing Liu, Yutao Zeng, Zeyue Xue, Zhiheng Liu, Yunshui Li, Jin Ma, Jie Huang, Xun Zhou, et al. Scaling law for quantization-aware training. *arXiv preprint arXiv:2505.14302*, 2025.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

Weizhi Gao, Zhichao Hou, Junqi Yin, Feiyi Wang, Linyu Peng, and Xiaorui Liu. Modulated diffusion: Accelerating generative modeling with modulated quantization. In *International Conference on Machine Learning*, pp. 18337–18362. PMLR, 2025.

Yefei He, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. EfficientDM: Efficient quantization-aware fine-tuning of low-bit diffusion models. *International Conference on Learning Representations*, 2023.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pp. 8867–8887. PMLR, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024.

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-Diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17535–17545, 2023a.

Yanjing Li, Sheng Xu, Xianbin Cao, Xiao Sun, and Baochang Zhang. Q-dm: An efficient low-bit quantized diffusion model. *Advances in neural information processing systems*, 36:76680–76691, 2023b.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. BRECQ: Pushing the limit of post-training quantization by block reconstruction. *International Conference on Learning Representations*, 2021.

Guang Liang, Xinyao Liu, and Jianxin Wu. Gplq: A general, practical, and lightning qat method for vision transformers. *arXiv preprint arXiv:2506.11784*, 2025.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *International Conference on Learning Representations*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In *International Conference on Machine Learning*, pp. 16318–16330. PMLR, 2022.

Andrei Panferov, Jiale Chen, Soroush Tabesh, Roberto L Castro, Mahdi Nikdan, and Dan Alistarh. Quest: Stable training of llms with 1-bit weights and activations. *arXiv preprint arXiv:2502.05003*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. *Advances in Neural Information Processing Systems*, 31, 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in Neural Information Processing Systems*, 29, 2016.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *International Conference on Learning Representations*, 2021a.

Shuaiwen Leon Song, Bonnie Kruft, Minjia Zhang, Conglong Li, Shiyang Chen, Chengming Zhang, Masahiro Tanaka, Xiaoxia Wu, Jeff Rasley, Ammar Ahmad Awan, et al. DeepSpeed4Science Initiative: Enabling large-scale scientific discovery through sophisticated ai system technologies. *NeurIPS 2023 AI for Science Workshop*, 2023.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*, 2021b.

Duo Su, Junjie Hou, Weizhi Gao, Yingjie Tian, and Bowen Tang. D^4M: Dataset distillation via disentangled diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5809–5818, 2024.

Changyuan Wang, Ziwei Wang, Xiuwei Xu, Yansong Tang, Jie Zhou, and Jiwen Lu. Towards accurate post-training quantization for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16026–16035, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xiansheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7308–7316, 2019.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Xingyu Zheng, Xianglong Liu, Haotong Qin, Xudong Ma, Mingyuan Zhang, Haojie Hao, Jiakai Wang, Zixiang Zhao, Jinyang Guo, and Michele Magno. Binarydm: Accurate weight binarization for efficient diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2024.

A PROOF

In this section, we first prove the propositions stated in the main paper. We also present Proposition 3 to show that treating historical states as static constants does not guarantee that the model can reach a local minimum of the loss.

Proposition 1 (Restated, 1)**.** *Suppose we are given $\boldsymbol{x}_0$, independent Gaussian noises $\epsilon_1, \epsilon_2 \sim \mathcal{N}(0, \mathbf{I})$, and a diffusion process defined by $\{\alpha_t\}$. Let $\boldsymbol{x}_t^{\mathrm{FS}}$ and $\boldsymbol{x}_{t'}^{\mathrm{FS}}$ denote two samples produced by the standard forward diffusion process at timesteps t and t' given ϵ_1, ϵ_2, respectively. Let $\boldsymbol{x}_t^{\mathrm{SFS}}$ and $\boldsymbol{x}_{t'}^{\mathrm{SFS}}$ denote the corresponding samples generated by SFS according to Eq. equation 4 and Eq. equation 5. Then, for any $t' \in [t, T]$, we have:*

1. $\boldsymbol{x}_{t'}^{\mathrm{SFS}} \sim N(\sqrt{\bar{\alpha}_{t'}}\boldsymbol{x}_0, (1-\bar{\alpha}_{t'})\mathbf{I})$
2. $Var(\boldsymbol{x}_t^{\mathrm{SFS}} - \boldsymbol{x}_{t'}^{\mathrm{SFS}}) < Var(\boldsymbol{x}_t^{\mathrm{FS}} - \boldsymbol{x}_{t'}^{\mathrm{FS}})$

Proof: We directly substitute $\boldsymbol{x}_t$ from Eq. equation 4 into Eq. equation 5:

$$\boldsymbol{x}_{t'}^{\mathrm{SFS}} = \sqrt{\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_1) + \sqrt{1-\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}\epsilon_2 \tag{16}$$

$$= \sqrt{\bar{\alpha}_{t'}}\boldsymbol{x}_0 + \sqrt{\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}(1-\bar{\alpha}_t)}\epsilon_1 + \sqrt{1-\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}\epsilon_2, \tag{17}$$

Since ϵ_1 and ϵ_2 are two independent standard Gaussian noises, $\boldsymbol{x}_{t'}$ follows a Gaussian distribution with the following mean and variance:

$$\mathbf{E}[\boldsymbol{x}_{t'}^{\mathrm{SFS}}] = \sqrt{\bar{\alpha}_{t'}}\boldsymbol{x}_0 \tag{18}$$

$$\mathrm{Var}(\boldsymbol{x}_{t'}^{\mathrm{SFS}}) = \left(\sqrt{\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}(1-\bar{\alpha}_t)}\right)^2 + \left(\sqrt{1-\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}\right)^2 \tag{19}$$

$$= 1 - \bar{\alpha}_{t'}. \tag{20}$$

The computation of $\boldsymbol{x}_t^{\mathrm{FS}} - \boldsymbol{x}_{t'}^{\mathrm{FS}}$ is as follows:

$$\boldsymbol{x}_t^{\mathrm{FS}} - \boldsymbol{x}_{t'}^{\mathrm{FS}} = (\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_1) - (\sqrt{\bar{\alpha}_{t'}}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_{t'}}\epsilon_2) \tag{21}$$

$$= (\sqrt{\bar{\alpha}_t} - \sqrt{\bar{\alpha}_{t'}})\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_1 - \sqrt{1-\bar{\alpha}_{t'}}\epsilon_2. \tag{22}$$

Then the variance of $\boldsymbol{x}_t^{\mathrm{FS}} - \boldsymbol{x}_{t'}^{\mathrm{FS}}$ is:

$$\mathrm{Var}(\boldsymbol{x}_t^{\mathrm{FS}} - \boldsymbol{x}_{t'}^{\mathrm{FS}}) = (1-\bar{\alpha}_t) + (1-\bar{\alpha}_{t'}) \tag{23}$$

$$= 2 - \bar{\alpha}_t - \bar{\alpha}_{t'} \tag{24}$$

The computation of $\boldsymbol{x}_t^{\mathrm{SFS}} - \boldsymbol{x}_{t'}^{\mathrm{SFS}}$ is as follows:

$$\boldsymbol{x}_t^{\mathrm{SFS}} - \boldsymbol{x}_{t'}^{\mathrm{SFS}} = (\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_1) - (\sqrt{\bar{\alpha}_{t'}}\boldsymbol{x}_0 + \sqrt{\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}(1-\bar{\alpha}_t)}\epsilon_1 + \sqrt{1-\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}\epsilon_2) \tag{25}$$

$$= (\sqrt{\bar{\alpha}_t} - \sqrt{\bar{\alpha}_{t'}})\boldsymbol{x}_0 + (1 - \sqrt{\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}})\sqrt{1-\bar{\alpha}_t}\epsilon_1 - \sqrt{1-\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}\epsilon_2. \tag{26}$$

Then the variance of $\boldsymbol{x}_t^{\mathrm{SFS}} - \boldsymbol{x}_{t'}^{\mathrm{SFS}}$ is:

$$\mathrm{Var}(\boldsymbol{x}_t^{\mathrm{SFS}} - \boldsymbol{x}_{t'}^{\mathrm{SFS}}) = (1-\bar{\alpha}_t)(\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t} - 2\sqrt{\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}} + 1) + 1 - \frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t} \tag{27}$$

$$= 2 - \bar{\alpha}_t - \bar{\alpha}_{t'} + 2(\sqrt{\bar{\alpha}_t\bar{\alpha}_{t'}} - \sqrt{\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}) \tag{28}$$

$$= \mathrm{Var}(\boldsymbol{x}_t^{\mathrm{FS}} - \boldsymbol{x}_{t'}^{\mathrm{FS}}) + 2\sqrt{\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}(\bar{\alpha}_t - 1). \tag{29}$$

Given that $\bar{\alpha}_t < 1$, the proof is completed. □

Proposition 2 (Restated, 2)**.** *Consider a regression problem where the input data $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$, and the ground-truth target corresponding to $\boldsymbol{x}_2$ is $y \in \mathbb{R}$. Given a one-layer model:*

$$f(\boldsymbol{x}_2, \boldsymbol{x}_1, \boldsymbol{w}) = \boldsymbol{w}(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1)) + \boldsymbol{w}\boldsymbol{x}_1, \tag{30}$$

where $\boldsymbol{w}$ denotes the model parameter initialized as $\boldsymbol{w}_0$, and Q is a dynamic quantizer without learnable parameters. We aim to fit $\boldsymbol{x}_2$ to the target y using the MSE loss:

$$\mathcal{L} = \frac{1}{2}\|y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w})\|^2. \tag{31}$$

Using gradient descent, indicated in Eq. equation 9 with learning rate η, the update rule of the optimization is:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}} \tag{32}$$

$$= \boldsymbol{w}_{t-1} + \eta(y - f)(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{x}_1), \tag{33}$$

where $t \geq 1$ is the optimization step. Then, we have:

$$|y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w}_t)|^2 \leq |y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w}_{t-1})|^2. \tag{34}$$

Proof: According to the chain rule, the gradient of $\mathcal{L}$ with respect to $\boldsymbol{w}$ is given by:

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}} = \frac{\partial\mathcal{L}}{\partial f}\frac{\partial f}{\partial\boldsymbol{w}} \tag{35}$$

$$= -(y - f)(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{x}_1). \tag{36}$$

Then the loss will decrease by updating $\boldsymbol{w}_t$:

$$|y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w}_t)|^2 = |y - (\boldsymbol{w}_t(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{w}_t\boldsymbol{x}_1))| \tag{37}$$

$$= \|y - (\boldsymbol{w}_{t-1} + \eta(y - f)(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{x}_1))(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{x}_1)\|^2 \tag{38}$$

$$= (1 - \eta\|Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{x}_1\|_2^2)|y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w}_{t-1})|^2 \tag{39}$$

Given that the learning rate η is positive, we complete the proof. □

Proposition 3. *Consider a regression problem where the input data $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$, and the ground-truth target corresponding to $\boldsymbol{x}_2$ is $y \in \mathbb{R}$. Given a one-layer model:*

$$f(\boldsymbol{x}_2, \boldsymbol{x}_1, \boldsymbol{w}) = \boldsymbol{w}(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1)) + \boldsymbol{w}\boldsymbol{x}_1, \tag{40}$$

where $\boldsymbol{w}$ denotes the model parameter initialized as $\boldsymbol{w}_0$, and Q is a dynamic quantizer without learnable parameters. We aim to fit $\boldsymbol{x}_2$ to the target y using the MSE loss:

$$\mathcal{L} = \frac{1}{2}\|y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w})\|^2. \tag{41}$$

In this setting, using gradient descent with static-cache optimization (i.e., treating $\boldsymbol{w}\boldsymbol{x}_1$ as constant) and learning rate η, the update rule of the optimization is:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} \tag{42}$$

$$= \boldsymbol{w}_{t-1} + \eta(y - f)Q(\boldsymbol{x}_2 - \boldsymbol{x}_1), \tag{43}$$

where $t \geq 1$ is the optimization step. In the condition that $Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) \cdot (Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{x}_1) < 0$, we have:

$$|y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w}_t)|^2 > |y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w}_{t-1})|^2. \tag{44}$$

Proof: According to the chain rule, the gradient of $\mathcal{L}$ with respect to $\boldsymbol{w}$ is given by:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \frac{\partial \mathcal{L}}{\partial f} \frac{\partial f}{\partial \boldsymbol{w}} \tag{45}$$

$$= -(y - f)Q(\boldsymbol{x}_2 - \boldsymbol{x}_1). \tag{46}$$

Then the loss will increase by updating $\boldsymbol{w}_t$:

$$|y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w}_t)|^2 = |y - (\boldsymbol{w}_t(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1)) + \boldsymbol{w}_t\boldsymbol{x}_1)| \tag{47}$$

$$= |y - (\boldsymbol{w}_{t-1} + \eta(y - f)Q(\boldsymbol{x}_2 - \boldsymbol{x}_1))(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{x}_1)|^2 \tag{48}$$

$$= (1 - \eta(\|Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{x}_1\|_2^2 - \boldsymbol{x}_1(Q(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{x}_1)))|(y - f(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{w}_{t-1}))|^2 \tag{49}$$

Given that the learning rate η is positive and taking the condition in, we complete the proof. □

B MODULATED DIFFUSION MODELS

Modulated diffusion models (MoDiff) have recently been proposed to alleviate the activation quantization challenges (Gao et al., 2025). It allows the quantization with almost no performance sacrifice in 4-bit activation in the PTQ cases. Given a linear operator $\mathcal{A}^{(l)}$ and its input $\boldsymbol{a}_t^{(l)}$ and output $\boldsymbol{o}_t^{(l)}$, it starts with the reformulation of linear operators:

$$\begin{cases} \boldsymbol{o}_T^{(l)} & = \mathcal{A}^{(l)}(\boldsymbol{a}_T^{(l)}) \\ \boldsymbol{o}_{T-1}^{(l)} & = \mathcal{A}^{(l)}(\boldsymbol{a}_{T-1}^{(l)}) = \mathcal{A}^{(l)}(\boldsymbol{a}_{T-1}^{(l)} - \boldsymbol{a}_T^{(l)}) + \boldsymbol{o}_T^{(l)} \\ & \cdots \\ \boldsymbol{o}_t^{(l)} & = \mathcal{A}^{(l)}(\boldsymbol{a}_t^{(l)}) \quad = \mathcal{A}^{(l)}(\boldsymbol{a}_t^{(l)} - \boldsymbol{a}_{t+1}^{(l)}) + \boldsymbol{o}_{t+1}^{(l)} \\ & \cdots \\ \boldsymbol{o}_1^{(l)} & = \mathcal{A}^{(l)}(\boldsymbol{a}_1^{(l)}) \quad = \mathcal{A}^{(l)}(\boldsymbol{a}_1^{(l)} - \boldsymbol{a}_2^{(l)}) + \boldsymbol{o}_2^{(l)} \end{cases} \tag{50}$$

Then it quantizes the temporal differences, which has a much smaller magnitude and concentrated distribution:

$$\begin{cases} \hat{\boldsymbol{o}}_T & = \mathcal{A}\Big(Q(\boldsymbol{a}_T)\Big) & \approx \mathcal{A}(\boldsymbol{a}_T) \\ \hat{\boldsymbol{o}}_{T-1} & = \mathcal{A}\Big(Q(\boldsymbol{a}_{T-1} - \boldsymbol{a}_T)\Big) + \hat{\boldsymbol{o}}_T & \approx \mathcal{A}(\boldsymbol{a}_{T-1}) \\ & \cdots \\ \hat{\boldsymbol{o}}_t & = \mathcal{A}\Big(Q(\boldsymbol{a}_t - \boldsymbol{a}_{t+1})\Big) + \hat{\boldsymbol{o}}_{t+1} & \approx \mathcal{A}(\boldsymbol{a}_t) \\ & \cdots \\ \hat{\boldsymbol{o}}_1 & = \mathcal{A}\Big(Q(\boldsymbol{a}_1 - \boldsymbol{a}_2)\Big) + \hat{\boldsymbol{o}}_2 & \approx \mathcal{A}(\boldsymbol{a}_1) \end{cases} \tag{51}$$

However, the linearity is broken by the quantization, where the error will accumulate across the iterative generation process. The error compensation is proposed to solve this challenge by introducing intermediate variables $\hat{\boldsymbol{a}}_t^{(l)}$:

$$\begin{aligned}
\hat{\boldsymbol{a}}_T &= Q(\boldsymbol{a}_T) && (52)\\
\hat{\boldsymbol{o}}_T &= \mathcal{A}(\hat{\boldsymbol{a}}_T) && (53)\\
\hat{\boldsymbol{a}}_{T-1} &= Q(\boldsymbol{a}_{T-1} - \hat{\boldsymbol{a}}_T) + \hat{\boldsymbol{a}}_T && (54)\\
\hat{\boldsymbol{o}}_{T-1} &= \mathcal{A}(\hat{\boldsymbol{a}}_{T-1}) = \mathcal{A}\Big(Q(\boldsymbol{a}_{T-1} - \hat{\boldsymbol{a}}_T)\Big) + \hat{\boldsymbol{o}}_T && (55)\\
&\dots && (56)\\
\hat{\boldsymbol{a}}_t &= Q(\boldsymbol{a}_t - \hat{\boldsymbol{a}}_{t+1}) + \hat{\boldsymbol{a}}_{t+1} && (57)\\
\hat{\boldsymbol{o}}_t &= \mathcal{A}(\hat{\boldsymbol{a}}_t) = \mathcal{A}\Big(Q(\boldsymbol{a}_t - \hat{\boldsymbol{a}}_{t+1})\Big) + \hat{\boldsymbol{o}}_{t+1} && (58)\\
&\dots && (59)\\
\hat{\boldsymbol{a}}_1 &= Q(\boldsymbol{a}_1 - \hat{\boldsymbol{a}}_2) + \hat{\boldsymbol{a}}_2 && (60)\\
\hat{\boldsymbol{o}}_1 &= \mathcal{A}(\hat{\boldsymbol{a}}_1) = \mathcal{A}\Big(Q(\boldsymbol{a}_1 - \hat{\boldsymbol{a}}_2)\Big) + \hat{\boldsymbol{o}}_2 && (61)
\end{aligned}$$

where $\hat{\boldsymbol{a}}_t^{(l)}$ stores the activation that is actually computed through quantization, which keeps track of the quantization errors:

$$\begin{aligned}
\boldsymbol{e}_t &= (\boldsymbol{a}_t - \hat{\boldsymbol{a}}_{t+1}) - Q(\boldsymbol{a}_t - \hat{\boldsymbol{a}}_{t+1}) && (62)\\
&= (\boldsymbol{a}_t - \hat{\boldsymbol{a}}_{t+1}) - (\hat{\boldsymbol{a}}_t - \hat{\boldsymbol{a}}_{t+1}) = \boldsymbol{a}_t - \hat{\boldsymbol{a}}_t,
\end{aligned}$$

C Backpropagation for Convolutional Layers

In this section, we present the backpropagation formulas for convolutional layers. Suppose the convolutional layer $\mathcal{A}^{(l)}$ is parameterized by the convolution kernel $\boldsymbol{w}^{(l)}$, which is quantized as $\boldsymbol{w}_{\text{int}}^{(l)}$. The backpropagation is then given as follows:

$$\frac{\partial \boldsymbol{o}_t^{(l)}}{\partial \boldsymbol{w}_{\text{int}}^{(l)}}[\Delta \boldsymbol{w}_{\text{int}}^{(l)}] = \Delta \boldsymbol{w}_{\text{int}}^{(l)} * \left(Q^{(l)}(\boldsymbol{a}_t^{(l)} - \boldsymbol{a}_{t'}^{(l)}) + \boldsymbol{a}_{t'}^{(l)}\right), \tag{63}$$

$$\frac{\partial \boldsymbol{o}_t^{(l)}}{\partial \boldsymbol{a}_t^{(l)}}[\Delta \boldsymbol{a}_t^{(l)}] = \boldsymbol{w}_{\text{int}}^{(l)} * \left(\left.\frac{\partial Q^{(l)}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{a}_t^{(l)} - \boldsymbol{a}_{t'}^{(l)}} \odot \Delta \boldsymbol{a}_t^{(l)}\right), \tag{64}$$

$$\frac{\partial \boldsymbol{o}_t^{(l)}}{\partial \boldsymbol{a}_{t'}^{(l)}}[\Delta \boldsymbol{a}_{t'}^{(l)}] = \boldsymbol{w}_{\text{int}}^{(l)} * \left(\left(1 - \left.\frac{\partial Q^{(l)}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{a}_t^{(l)} - \boldsymbol{a}_{t'}^{(l)}}\right) \odot \Delta \boldsymbol{a}_{t'}^{(l)}\right), \tag{65}$$

where $*$ denotes the cross-correlation operation and $\odot$ denotes elementwise multiplication. The notation $\frac{\partial \boldsymbol{o}_t^{(l)}}{\partial \boldsymbol{a}_t^{(l)}}[\Delta \boldsymbol{a}_t^{(l)}]$ denotes the Jacobian of $\boldsymbol{o}_t^{(l)}$ with respect to $\boldsymbol{a}_t^{(l)}$ applied to the perturbation $\Delta \boldsymbol{a}_t^{(l)}$. The difference between the linear layer and the convolutional layer is that the matrix multiplications in Eq. equation 10 and Eq. equation 11 are replaced by the corresponding cross-correlation operators.

D Quantization Methods

In this section, we introduce the PTQ and QAT baselines we use in our paper. For weight quantization, both DQ + MoDiff and Q-Diff + MoDiff adopt the same method as Q-Diffusion across various weight bitwidths (Li et al., 2023a). In contrast, LSQ + MoDiff and BinaryDM + MoDiff use weights obtained from QAT methods, detailed in Sec. D.2.

D.1 PTQ Baselines

For the activation quantization of PTQ, we adopt Dynamic Quantization and Q-Diffusion.

Dynamic Quantization. Dynamic quantization computes the scaling factors based on the dynamic range of the input, i.e., its maximum and minimum values. It introduces only rounding error, without overflow error, and does not need a calibration dataset to learn the scaling factors. Specifically,

$$s = \frac{\boldsymbol{x}_{\text{max}} - \boldsymbol{x}_{\text{min}}}{2^b - 1}, \tag{66}$$

where b denotes the bitwidth of the quantizer. Unlike MoDiff, which uses channel-wise quantization, we adopt a tensor-wise quantizer for better hardware efficiency.

For the implementation of MoDiff with dynamic quantization, we strictly follow the original methodology described in (Gao et al., 2025).

Q-Diffusion. Q-Diffusion learns static scaling factors with a calibration dataset by optimizing the mean square error (MSE) loss for each layer.

For the implementation of MoDiff with Q-Diffusion, we use the calibration dataset provided in their official code repository (`https://github.com/WeizhiGao/MoDiff`), modifying only the bitwidth settings as needed.

D.2 QAT Baselines

We use two QAT baselines in our main paper.

Learned Step Size Quantization (LSQ) (Esser et al., 2019). LSQ learns the scaling factors of the quantizers when we train the model. To be specific, the quantizer is defined as follows:

$$\boldsymbol{x}_{\text{int}} = \text{clamp}(\left\lfloor \frac{\boldsymbol{x}}{s} \right\rceil; 0, 2^b - 1), \tag{67}$$

$$Q(\boldsymbol{x}) = s\boldsymbol{x}_{\text{int}} \tag{68}$$

For the weight quantizer, we use the symmetric version. To enable the backward process through the rounding operation, the gradient with respect to s is simulated as follows:

$$\frac{\partial Q(\boldsymbol{x})}{\partial s} = \begin{cases} -\frac{\boldsymbol{x}}{s} + \left\lfloor \frac{\boldsymbol{x}}{s} \right\rceil, & 0 \le Q(x) \le 2^b - 1 \\ 0, & Q(x) \le 0 \\ 2^b - 1, & Q(x) \ge 2^b - 1 \end{cases} \tag{69}$$

In our implementation, we quantize only the weights in LSQ + MoDiff, keeping activations in full precision during training. Then we apply dynamic quantization for the activation with MoDiff during inference. In all other settings (LSQ and LSQ + ModTrain), both weights and activations are quantized using the aforementioned quantizers.

BinaryDM (Zheng et al., 2024). BinaryDM is a quantization method that binarizes the weights while allowing higher-bit activations. The binary weight quantizer transforms the input weights as follows:

$$Q(\boldsymbol{x}) = \sigma \text{sign}(\boldsymbol{x}), \tag{70}$$

where σ is the scaling factor, and sign$\cdot$ maps the input to $+1$ or -1. During backpropagation, we use the straight-through estimator (STE) to approximate the gradient:

$$\frac{\partial}{\partial \boldsymbol{x}} \text{sign}(\boldsymbol{x}) = \begin{cases} 1, & -1 \le \boldsymbol{x} \le 1 \\ 0, & \boldsymbol{x} < -1 \text{ or } \boldsymbol{x} > 1 \end{cases} \tag{71}$$

In the implementation of BinaryDM + MoDiff, activations are kept in full precision during training. During inference, we apply dynamic quantization to the activations in combination with MoDiff. In all other settings (BinaryDM and BinaryDM + ModTrain), activations are quantized using LSQ quantizers.

E Implementation Details

In this section, we provide the implementation details of ModTrain, including the training hyperparameters and sampling settings. Our code is developed based on `https://github.com/Xingyu-Zheng/BinaryDM`.

For LSUN-Churches, we use a learning rate of 5.0×10^{-5} with a batch size of 16. We set $\Delta t = 200$ and $\lambda = 0.3$ in ModTrain. For LSUN-Bedrooms, we use a learning rate of 2.0×10^{-5} with a batch size of 4. We set $\Delta t = 300$ and $\lambda = 0.5$ in ModTrain. Both models are trained for 200k iterations.

For sampling, we adopt the DDIM sampler (Song et al., 2021a) for both LSUN-Churches and LSUN-Bedrooms. We use 200 sampling steps, with $\eta = 0.0$ for Churches and $\eta = 1.0$ for Bedrooms.

Our experiments are run on A6000 GPUs and H100 GPUs.

F Full-Precision Activation Generation

In this section, we present the performance degradation of the full-precision activation model caused by QAT. As shown in Figure 3, the generation quality of the full-precision activation counterpart trained with LSQ (W2A2) is poor. This indicates that QAT encourages the model to specialize in quantized activations, often at the expense of degraded performance under full-precision activations.

We also show how our FPA technique preserves full-precision activation performance in Figure 4. We draw two conclusions. First, ModTrain minimizes activation quantization errors, which is consistent with modulated diffusion models. Second, FPA improves the generation quality of the full-precision activation model.

Table 6: The number of operations and FID scores for different quantization methods at various precision levels.

Method	Bits (W/A)	Size (MB)	GBops	FID
FP	32/32	1219.2	20060	4.36
Q-Diff + MoDiff	4/4	144.2	315	4.58
Q-Diff + MoDiff	3/3	104.3	178	5.61
LSQ	2/2	76.2	151	14.90
LSQ + MoQAT	2/2	76.2	153	7.85

Table 7: Latency Comparison (μs) Under Different Precisions

Kernel Shape	**Input Shape**	**FP16**	**W8A8**	**W8A8 Modulate**	**W4A4**	**W4A4 Modulate**
$512 \times 512 \times 3 \times 3$	8×8	430 (1.0×)	305 (1.41×)	326 (1.32×)	232 (1.85×)	247 (1.74×)
$512 \times 512 \times 3 \times 3$	32×32	3000 (1.0×)	1880 (1.60×)	1940 (1.55×)	1330 (2.26×)	1380 (2.17×)
$256 \times 256 \times 3 \times 3$	32×32	1050 (1.0×)	660 (1.59×)	680 (1.54×)	470 (2.23×)	490 (2.14×)

Table 8: HBM Traffic (MB) Under Different Precisions

Kernel Shape	**Input Shape**	**FP16**	**W8A8**	**W8A8 Modulate**	**W4A4**	**W4A4 Modulate**
$512 \times 512 \times 3 \times 3$	8×8	6.85	3.68	5.34	2.09	3.92
$512 \times 512 \times 3 \times 3$	32×32	48.85	28.43	53.34	18.21	46.67
$256 \times 256 \times 3 \times 3$	32×32	16.21	10.11	24.33	7.05	22.17

G Efficiency Analysis

In this section, we demonstrate the efficiency advantages of our method from the perspective of operation counts. All the models are trained on LSUN-Churches. As shown in Table 6, ModTrain preserves the generation quality with almost no additional computational cost compared to LSQ.

We additionally evaluate kernel-level efficiency on the A40 GPU through layer-wise experiments across different linear-layer dimensions, measuring both latency and HBM traffic. As shown in Tables 7 and 8, modulated diffusion models still achieve substantial computational speedups, with only a modest performance degradation compared with standard quantization kernels. Although the additional HBM traffic introduced by modulation can become a potential bottleneck, with carefully



(a) W2A2



(b) W2A32

Figure 3: Visualization of generated images on LSUN-Churches with the standard diffusion model trained by LSQ (W2A2)

optimized kernel implementations, much of the associated memory-access latency can be effectively hidden by overlapping memory operations with computation.

H VISUALIZATION

In this section, we present generated images produced by LSQ and LSQ+ModTrain at different precision levels in Figures 5, 6, and 7. The images are generated on LSUN-Churches.



(a) W2A32 w/o FPA



(b) W2A2 w/o FPA



(c) W2A32 w/ FPA



(d) W2A2 w/ FPA

Figure 4: Visualization of generated images on LSUN-Churches with the modulated diffusion trained by LSQ (W2A2)



(a) LSQ (W4A4)



(b) LSQ + ModTrain (W4A4)

Figure 5: Visualization of generated images on LSUN-Churches of LSQ (W4A4)



(a) LSQ (W3A3)



(b) LSQ + ModTrain (W3A3)

Figure 6: Visualization of generated images on LSUN-Churches of LSQ (W3A3)



(a) LSQ (W2A2)



(b) LSQ + ModTrain (W2A2)

Figure 7: Visualization of generated images on LSUN-Churches of LSQ (W2A2)