ACCURATE COMPRESSION OF TEXT-TO-IMAGE DIFFUSION MODELS VIA VECTOR QUANTIZATION

Vage Egiazarian[*12] **Denis Kuznedelev**[*13] **Anton Voronov**[*124] **Ruslan Svirschevski**[12]

Michael Goin[5] **Daniil Pavlov**[4] **Dan Alistarh**[56] **Dmitry Baranchuk**[1]

[1]Yandex Research [2]HSE University [3]Skoltech [4]MIPT [5]Neural Magic [6]IST Austria

https://yandex-research.github.io/vqdm

ABSTRACT

Text-to-image diffusion models have emerged as a powerful framework for high-quality image generation given textual prompts. Their success has driven the rapid development of production-grade diffusion models that consistently increase in size and already contain billions of parameters. As a result, state-of-the-art text-to-image models are becoming less accessible in practice, especially in resource-limited environments. *Post-training quantization* (PTQ) tackles this issue by compressing the pretrained model weights into lower-bit representations. Recent diffusion quantization techniques primarily rely on *uniform scalar quantization*, providing decent performance for the models compressed to 4 bits. This work demonstrates that more versatile *vector quantization* (VQ) may achieve higher compression rates for large-scale text-to-image diffusion models. Specifically, we tailor vector-based PTQ methods to recent billion-scale text-to-image models (SDXL and SDXL-Turbo), and show that the diffusion models of 2B+ parameters compressed to around 3 bits using VQ exhibit the similar image quality and textual alignment as previous 4-bit compression techniques.

1 INTRODUCTION

In recent years, diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020; Rombach et al., 2022) have revolutionized text-to-image (T2I) synthesis (Saharia et al., 2022; Podell et al., 2024; Pernias et al., 2023; Rombach et al., 2021; Esser et al., 2024; Betker et al., 2023), transforming user textual prompts into highly realistic images. Motivated by trends in large language models (LLM), practitioners are eager to enhance the performance of text-to-image diffusion models by scaling and curating datasets (Li et al., 2024; Kastryulin et al., 2024; Schuhmann et al., 2022) and increasing model sizes (Esser et al., 2024; Podell et al., 2024; Li et al., 2024). Current state-of-the-art models contain up to 12 billion parameters (Black Forest Labs, 2024), with some studies suggesting that the performance gains from model scaling have yet to reach saturation (Esser et al., 2024). Therefore, we are likely to witness even larger open-source text-to-image models shortly. However, the scaling trend of text-to-image models presages slow inference and large memory consumption in practice. The problem gets more pronounced on edge devices with limited memory and weak processing units. To address this issue, effective post-training diffusion model compression methods need to be developed.

Quantization (Gholami et al., 2021) is a prominent compression technique that has gained significant attention due to its remarkable success in LLM compression (Frantar et al., 2023; Dettmers et al., 2023; Egiazarian et al., 2024; Tseng et al., 2024). Weight quantization typically transforms the original parameter tensor into a condensed representation, where each weight or group of weights is replaced with a low-bit value. Effective quantization algorithms aim to minimize the discrepancy between the model outputs obtained with the original and compressed weights. Quantization methods can be broadly categorized into *scalar quantization*, which projects each individual weight onto a one-dimensional grid, and *vector quantization* (Burton et al., 1983; Gray, 1984), which represents a group of weights as a vector from a *codebook*. Due to its simplicity, scalar quantization is widely adopted across different domains. Several studies (Li et al., 2023b; Shang et al., 2023; He et al., 2023;

[*] Equal contribution

Huang et al., 2024) have employed it for small-scale diffusion models, achieving nearly lossless compression at 4-8 bits. However, for higher compression rates, a more sophisticated representation might be necessary to maintain the original model's performance.

Vector quantization (VQ) is a promising candidate due to its flexibility and ability to account for the non-uniformity of data distribution, thus offering smaller quantization errors for the same compression rate. The encoded vectors are stored as low-bit integer indices, *codes*, which point to vector representations, or *codewords*, in the corresponding codebooks shared within a model layer. Various VQ extensions (Babenko & Lempitsky, 2014; Chen et al., 2010; Jegou et al., 2010; Martinez et al., 2016; 2018) employ multiple codebooks and hence provide even better compression quality. Recently, vector quantization has demonstrated state-of-the-art results in low-bit compression of large language models (Egiazarian et al., 2024; Tseng et al., 2024). Therefore, it is appealing to explore the application of vector quantization for large-scale diffusion models.

Contributions. This work presents a novel post-training quantization method for large-scale text-to-image diffusion models. Our PTQ method relies on *additive quantization* (AQ) (Babenko & Lempitsky, 2014), the generalized multi-codebook vector quantization (MCQ) formulation originated in vector search for extreme vector compression. We explore AQ specifically for billion-parameter diffusion U-Net architectures (Podell et al., 2024; Sauer et al., 2023b) and propose several practical techniques for their fast and accurate compression. Notably, capitalizing on the increased capacity of AQ codebooks compared to uniform quantization, we demonstrate that the initial calibration can largely benefit from additional codebook tuning. Following (He et al., 2024), we propose to fine-tune the quantized model to mimic the final U-Net predictions of the full-precision model. Unlike typical settings in quantization-aware training (QAT) (Nagel et al., 2022; Esser et al., 2020), the fine-tuning approach does not require training on large-scale datasets and takes a tiny fraction of the time needed to train the original model. We demonstrate that fine-tuning significantly eliminates the discrepancy with the full-precision model, unlocking highly accurate 3-bit weight compression for PTQ of billion-scale diffusion models. To sum up, our contributions can be formulated as follows:

- We explore vector-based PTQ strategies for text-to-image diffusion models and demonstrate that the compressed models yield higher quality text-to-image generation than the scalar alternatives under the same bit-widths. Furthermore, we describe an effective fine-tuning technique that further closes the gap between the full-precision and compressed models, leveraging the flexibility of the vector quantized representation.
- To illustrate the power of our technique, we compress the weights of SDXL (Podell et al., 2024), the state-of-the-art text-to-image diffusion model with 2.6B parameters, down to 3 bits per parameter. Extensive human evaluation and automated metrics confirm the superiority of our approach over previous diffusion compression methods under the same bit-widths.
- We illustrate that our approach can be effectively applied to distilled diffusion models, such as SDXL-Turbo (Sauer et al., 2023b), achieving nearly lossless 4-bit compression. This highlights the potential for combining integrating text-to-image model quantization with other diffusion acceleration techniques.

2 Related work

Efficient diffusion models. Text-to-image diffusion models (Saharia et al., 2022; Podell et al., 2024; Esser et al., 2024; Rombach et al., 2021; Pernias et al., 2023; Betker et al., 2023) are a class of generative models that gradually transform noise samples into realistic images corresponding to textual prompts. One of the most prominent challenges in diffusion modeling is a sequential sampling procedure resulting in significantly slower inference compared to feed-forward alternatives (Goodfellow et al., 2014; Kingma & Welling, 2022; Kang et al., 2023; Sauer et al., 2023a).

Several major research directions address the efficiency of text-to-image diffusion models. Diffusion distillation (Sauer et al., 2024; 2023b; Luo et al., 2023; Song et al., 2023; Meng et al., 2023) and advanced sampling algorithms (Zhou et al., 2024; Lu et al., 2022; Zhao et al., 2023) aim to reduce the number of sampling steps in the sequential inference of diffusion models. Intermediate activation caching (Ma et al., 2024; Wimbauer et al., 2024; Li et al., 2023a), efficient architecture designs (Rombach et al., 2021; Podell et al., 2024) and structured pruning (Kim et al., 2023; Fang et al., 2023) focus on faster sampling at the architecture level. Model quantization methods (Li et al., 2023b; Shang et al., 2023; He et al., 2023; Huang et al., 2024; Chang et al., 2023; Wang et al., 2024) aim to reduce the memory footprint and runtime by compressing the weights and/or activations of diffusion models to compact representations.

Model quantization. This work focuses on model quantization, which has shown remarkable accuracy-compression trade-offs in the context of LLM compression (Dettmers et al., 2023; Chee et al., 2023; Egiazarian et al., 2024; Tseng et al., 2024). The quantization methods can generally be categorized into two classes: *post-training quantization* (PTQ) (Li et al., 2021; Nagel et al., 2020) and *quantization-aware training* (QAT) (Nagel et al., 2022; Esser et al., 2020). PTQ performs quantization and weight adjustment on top of the pretrained model and typically requires largely lower costs compared to the training process. QAT methods, in contrast, undergo multiple rounds of training and quantization iterations. While they are likely to achieve higher accuracy than PTQ, they require more sophisticated and resource-intensive quantization procedures.

In practice, quantization is applied to linear or convolution layers, consuming the prevalent portion of the model weights in modern deep learning architectures (Frantar et al., 2022b; Wang et al., 2020; Khan et al., 2020). Uniform scalar quantization first scales full-precision weights to the specified value range and then rounds them into low-bit fixed-point values. Although uniform quantization is efficient and simple to use, it disregards non-uniformity of the underlying distributions and presence of outliers (Dettmers et al., 2023; Lin et al., 2024; Chee et al., 2023), leading to large quantization errors at low-bit compression. In contrast, non-uniform scalar quantization assigns individual weights to the scalar values c_i, *codewords*, from a codebook containing $k{=}2^B$ elements $C{=}\{c_1, ..., c_k\}$. Each weight is associated with a B-bit index i, *code*, of the corresponding codeword c_i. The codebooks are more flexible in capturing the underlying weight distributions than uniform grids, making non-uniform quantization more accurate under the same bit-width. Nevertheless, uniform quantization remains popular due to its highly efficient encoding and decoding processes.

Vector quantization (VQ) (Burton et al., 1983; Gray, 1984) extends non-uniform scalar quantization by increasing the dimensionality of the codewords. Thus, the VQ-based methods approximate a group of consecutive weights by a vector from the codebook. In our work, we exploit multi-codebook generalization of VQ, *additive quantization* (AQ) (Babenko & Lempitsky, 2014; Martinez et al., 2016; 2018). AQ represents the weight groups as a sum of M codewords chosen from multiple learned codebooks $C_1, ...C_M$. Weight group size, the number of codebooks M, and codebook size $k = 2^B$ dictate the trade-off between memory footprint and compression accuracy.

After initialization, the codebooks and other quantization parameters, e.g., scaling factors, are *calibrated* to approximate the activation distribution of the original model. The activation statistics are collected by running the model on small representative *calibration data*. Recent PTQ calibration strategies (Nagel et al., 2020; Frantar et al., 2022b; 2023; Egiazarian et al., 2024) simplify the model quantization problem to layer-wise quantization, enabling scalability to billion-parameter models. These methods minimize the MSE between the outputs of the full-precision weights $W \in \mathbb{R}^{d_{out} \times d_{in}}$ and the corresponding quantized weights W_q given calibration inputs X for each layer individually:

$$\arg\min_{W_q} ||WX - W_qX||_2^2 \tag{1}$$

For AQ quantized W_q, the optimal configuration is learned via alternating optimization of codes and codebooks. Our method adopts the learning procedure in Egiazarian et al. (2024), where AQ is applied to LLM compression.

Quantization of diffusion models. Previous diffusion PTQ approaches (Li et al., 2023b; Shang et al., 2023; He et al., 2023; Huang et al., 2024) employ uniform scalar quantization and calibrate the quantization parameters using adaptive rounding (Nagel et al., 2020) or simple min-max initialization (Nagel et al., 2021b). Some studies introduce diffusion-specific calibration data collection methods (Li et al., 2023b; Shang et al., 2023) and investigate the importance of different diffusion time steps and model layers (Li et al., 2023b; Shang et al., 2023; Yang et al., 2023; Chang et al., 2023; Huang et al., 2024). Other works address the accumulated quantization error caused by sequential diffusion sampling with quantized models (He et al., 2023; Li et al., 2023c). A few propose using variable bit widths for activations at different time steps (He et al., 2023; Tang et al., 2023).

Recent works (He et al., 2024; Wang et al., 2024) have demonstrated that effective quantization-aware fine-tuning (QAFT) may achieve state-of-the-art quantization performance on small-scale benchmarks. EfficientDM (He et al., 2024) introduces quantization-aware parameter-efficient fine-tuning for diffusion models and runs full-precision model distillation after initial calibration. TQD (So et al., 2023) fine-tunes diffusion models in a QAT manner and employs additional MLP modules to estimate quantization scales for each step. QuEST (Wang et al., 2024) proposes to fine-tune only the most vulnerable to quantization parts of the diffusion model: time-embedding, attention-related quantized weights, and activation quantizer parameters.

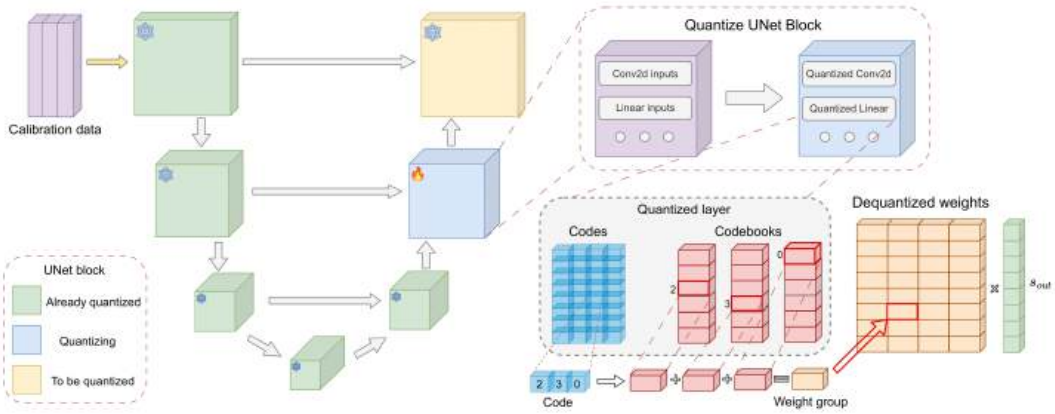

Figure 1: Overview of the proposed layer-wise calibration procedure before fine-tuning.

Contrary to the approaches mentioned above, we investigate more versatile weight quantizers for large-scale diffusion model compression. In this work, we focus solely on weight quantization and leave the investigation of activation quantization for future work.

3 METHOD

In this section, we present a VQ-based method adapted for text-to-image diffusion model compression. Section 3.1 introduces the strategy for applying the existing VQ algorithms to the U-Net architecture, commonly used in diffusion models. Then, in Section 3.2, we describe the proposed calibration method comprising of the layer-wise calibration and global fine-tuning procedures. In Section 3.3, we discuss the inference procedure for the VQ-compressed diffusion models.

3.1 VECTOR QUANTIZATION OF TEXT-TO-IMAGE MODELS

Existing vector quantization algorithms are designed for linear models used in retrieval applications (Jegou et al., 2010; Babenko & Lempitsky, 2014; Martinez et al., 2016; 2018) or large language models with extremely large hidden dimension (Egiazarian et al., 2024; Tseng et al., 2024). Naive transfer of the method introduced in Egiazarian et al. (2024); van Baalen et al. (2024) would not allow one to achieve practically useful compression rates due to significant architectural differences between LLMs and text-to-image diffusion models.

Specifically, unlike LLMs, which are built from homogeneous transformer blocks with the same number of hidden features, modern text-to-image diffusion models contain several heterogeneous components, including the diffusion model itself – typically a U-Net (Ronneberger et al., 2015) or 2D image Transformer (Esser et al., 2024; Peebles & Xie, 2022), one or more text encoders, autoencoders for latent diffusion models (Rombach et al., 2021) or super-resolution models for cascaded diffusion models (Saharia et al., 2022).

As an illustrative example, we consider Stable Diffusion XL (SDXL) (Podell et al., 2024): a latent diffusion model consisting of a large 2D U-Net diffusion model with $\sim$2.6B parameters, two text encoders, a variational autoencoder (Kingma & Welling, 2022), and an optional refiner network. Text encoders are regular transformer models with known high-performance quantization algorithms, including vector quantization (Egiazarian et al., 2024; Tseng et al., 2024; van Baalen et al., 2024). The variational encoder is a relatively small network with less than 0.1B parameters. Since the overall amount of computation and inference time is dominated by the iterative application of the diffusion model, only the compression of the U-Net model is of practical interest.

The U-Net backbone adopted in SDXL comprises stacks of residual and transformer blocks at different resolutions. The standard U-Net architecture includes an encoder, middle block, and decoder. The number of channels increases with the decrease in the feature map size after pooling operations. Specifically, SDXL U-Net has 3 downsampling levels, with 320 channels on the top level and the multiplication rates $[1, 2, 4]$ at different resolutions. A typical number of channels in a layer is in the order of 10^2–10^3, which is smaller by an order of magnitude than modern LLMs. Notably, the 16-bit codebooks with a group size of 8, that yield the highest quality in low-bit LLM compression (Egiazarian et al., 2024), occupy $2^{16} \cdot 8 \cdot 2 = 2^{20}$ bytes and exceed the size of many SDXL layers, making such quantization configuration impractical. Thus, one has to work with small codebooks (of size 2^6, 2^8), and capacity could be increased by increasing the number of codebooks.

Another aspect is the presence of convolutional layers in addition to linear in ResNet-like (He et al., 2016) blocks. Following the common practice (Nagel et al., 2021a) for CNN quantization, we group weights along the input channel dimension rather than the kernel spatial dimensions.

We quantize all convolutional and linear layers within the U-Net blocks, with a few exceptions:

- We do not quantize the first and last convolutional layers: the first layer has an input dimension of 4, and the last layer has a commensurate output dimension. These layers constitute a tiny fraction of an overall number of parameters, and there is no benefit from their compression.
- Time embedding layers are not quantized inspired by recent works (Huang et al., 2024; Wang et al., 2024) demonstrating the importance of the temporal features for quantized diffusion models. In addition, for a given input sample, timestep projections accept only a single vector instead of a whole feature map, as most linear projections and convolutional layers do. Finally, they incur small additional memory overhead. Therefore, we do not quantize these layers based on practical considerations.

3.2 Calibrating Vector-Quantized Diffusion Models

Here, we discuss the calibration procedure used to determine the optimal quantized weight configuration. The procedure involves two stages: i) layer-wise calibration and ii) global fine-tuning. Below, we discuss both stages in more detail.

Layer-wise calibration. First, the calibration set is collected by running the diffusion sampling on a small set of calibration prompts. Text-to-image diffusion models alter intermediate generation steps when using classifier-free guidance (CFG) (Ho & Salimans, 2022). Therefore, we run the diffusion sampling with a default CFG scale 5 and considering both unconditional and conditional inputs as separate calibration samples. Unlike some prior works (Shang et al., 2023; Li et al., 2023b), we collect calibration data for all diffusion timesteps and sample them uniformly.

The calibration procedure is performed sequentially for predefined consequent layer subsets. Specifically, the model, partially quantized up to the $i-$th subset, collects the activations for all layers within the $i+1-$th subset. Then, these layers are quantized in parallel, and the procedure goes to the following subset. The number of layers in subsets controls the calibration speed, memory consumption, and overall model quality. The quantization of all layers is the fastest, but storing the activation statistics requires a large amount of memory. In addition, using the original model activations to quantize deeper layers does not account for the output changes in the previous layers. Another extreme is to recollect the data after each quantized layer sequentially. However, the number of layers in modern diffusion models is several hundred, making such an option intolerably slow. Therefore, we choose the entire stack of blocks (both convolutional and transformer) for a given resolution as a subset of weights quantized at once. Since the SDXL U-Net has 3 downsampling and upsampling levels and a middle block, this constitutes 7 subsets in total. This choice provides a preferable trade-off between accuracy, memory efficiency, and calibration runtime.

Also, previous diffusion PTQ methods (Li et al., 2023b; He et al., 2023; Shang et al., 2023) collect raw model activations $X \in \mathbb{R}^{n \times d_{in}}$, where n is the calibration dataset size multiplied by the spatial dimension $h_l \times w_l$ for a layer l. We notice that this limits their scalability to large text-to-image models and calibration sets. Instead, following the practices in LLM quantization (Frantar et al., 2022a; Egiazarian et al., 2024; van Baalen et al., 2024), we collect $XX^T \in \mathbb{R}^{d_{in} \times d_{in}}$ and modify the objective (1) as follows: $||WX - W_qX||_2^2 = ||(W - W_q)X||_2^2 = \left\langle (W - W_q)XX^T, (W - W_q) \right\rangle_F$. In our experiments, this vastly reduces the storage of calibration data.

Finally, we closely adopt the codebook learning approach from Egiazarian et al. (2024). In addition to the codebooks, this method employs learnable scales $s_{out} \in \mathbb{R}^{d_{out}}$ to multiply each output dimension of the dequantized matrix $W_q \in \mathbb{R}^{d_{out} \times d_{in}}$. Thus, the dequantized matrix can be represented as $s_{out}W_q$. Figure 1 illustrates the proposed layer-wise calibration procedure.

Global fine-tuning. Since fine-tuning is known to boost the performance of vector-quantized models (Egiazarian et al., 2024; Tseng et al., 2024), we equip our calibration procedure with an end-to-end fine-tuning stage. Previous diffusion quantization approaches (Li et al., 2023b; Shang et al., 2023; Huang et al., 2024) perform block-wise fine-tuning for each residual or attention block right after their layer-wise calibration. However, the block-wise fine-tuning is unaware of the final model prediction. Therefore, we employ a more accurate solution by minimizing MSE between global U-Net predictions at each denoising timestep. We notice that there is no need for fine-tuning on large-scale data; a few thousand calibration samples generated with the full-precision U-Net are

Model	Method	Avg bits	Pickscore↑	CLIP↑	FID↓
SDXL	Original model	32	0.226	0.357	18.99
	VQDM	4.15	**0.226**	**0.356**	**19.11**
	VQDM	3.15	0.225	0.355	19.18
	VQDM	2.15	0.219	0.341	22.14
	Q-Diffusion	4	0.225	0.355	19.30
	PTQ4DM	4	0.224	0.353	19.78

Table 1: Evaluation of quantized SDXL models for different bit-widths in terms of automatic metrics.

sufficient. In fact, such fine-tuning is an instance of model distillation (Hinton et al., 2015) and helps to compensate inter-layer errors caused by independent layer-wise or block-wise calibration.

The model's forward pass is differentiable with respect to the codebook vectors; therefore, these can be optimized like any other parameter inside the U-Net network. In addition, all non-quantized layers are made trainable to compensate the quantization error.

The resulting calibration procedure is described in Appendix E in Algorithms 1, 2. First, the algorithm iterates over the U-Net down, middle, and up blocks and accumulates input activations within each block by running the diffusion sampling loop with classifier-free guidance (Ho & Salimans, 2022). Then, the layers within each block are quantized and calibrated with the algorithm inherited from AQLM (Egiazarian et al., 2024). Once the model is quantized, we fine-tune the entire model to mimic the teacher output. In the following, we denote the entire calibration method as VQDM.

3.3 Inference procedure

Uniform scalar quantization facilitates faster inference by using more efficient low-precision calculations while maintaining a constant number of arithmetic operations. In contrast, the VQ inference procedure first precomputes *look-up tables* (LUTs) w.r.t. the learned codebooks and then performs matrix multiplication by retrieving values from the LUTs and summing them to obtain the final result. This technique was originally proposed for quantized nearest neighbor search (Jegou et al., 2010; Babenko & Lempitsky, 2014) and can be adapted to quantized weight matrices (Blalock & Guttag, 2021; Egiazarian et al., 2024). Though such a procedure involves fewer arithmetic operations, we observe that it significantly slows down the diffusion inference on high-end GPUs and CPUs due to the absence of specific hardware realizations for efficient look-ups. In other words, software look-ups are slower than hardware-optimized multiplications with tensor cores or AVX-512.

As a result, we found it more effective to simply dequantize the weights into half- or full-precision values in runtime and perform standard matrix multiplication that benefits from hardware optimizations. Still, the proposed quantization method carries additional computational overheads caused by the dequantization procedure during inference. In Section 4.3, we measure the runtime overheads for our CPU and GPU implementations. We believe that this overhead is not a fundamental property of vector quantization but a quirk of modern hardware. Several recent studies (AbouElhamayed et al., 2023; Zhu et al., 2024) explore this further and achieve highly efficient vector quantization inference using FPGA (Field Programmable Gate Arrays) hardware. Both studies use Intel Agilex FPGAs.

4 Experiments

We begin by comparing VQ-compressed SDXL to the full-precision model and previous diffusion quantization methods. Then, we measure the inference overheads and realised memory reduction for our GPU and CPU implementations. Next, we conduct an ablation study on the design choices discussed in our method. Finally, we apply VQDM to the distilled diffusion model, SDXL-Turbo.

4.1 Experimental setup

As a primary quality measure, we consider side-by-side human evaluation (SbS) on 128 prompts specifically selected from PartiPrompts (Yu et al., 2022), following Sauer et al. (2024). We generate 2 images for each prompt and report the SbS score as a portion of the student wins plus half of the tied evaluations. To our knowledge, this is the first work that conducts the human study to assess quantized diffusion models. More details about the human evaluation setup are in Appendix B.

Additionally, we employ the standard metrics such as FID (Heusel et al., 2017) and CLIP Score (Hessel et al., 2021) and also evaluate PickScore (Kirstain et al., 2023), designed to estimate the human preference score. The automated metrics are calculated on 5000 prompts from the COCO2014 validation set (Lin et al., 2014).

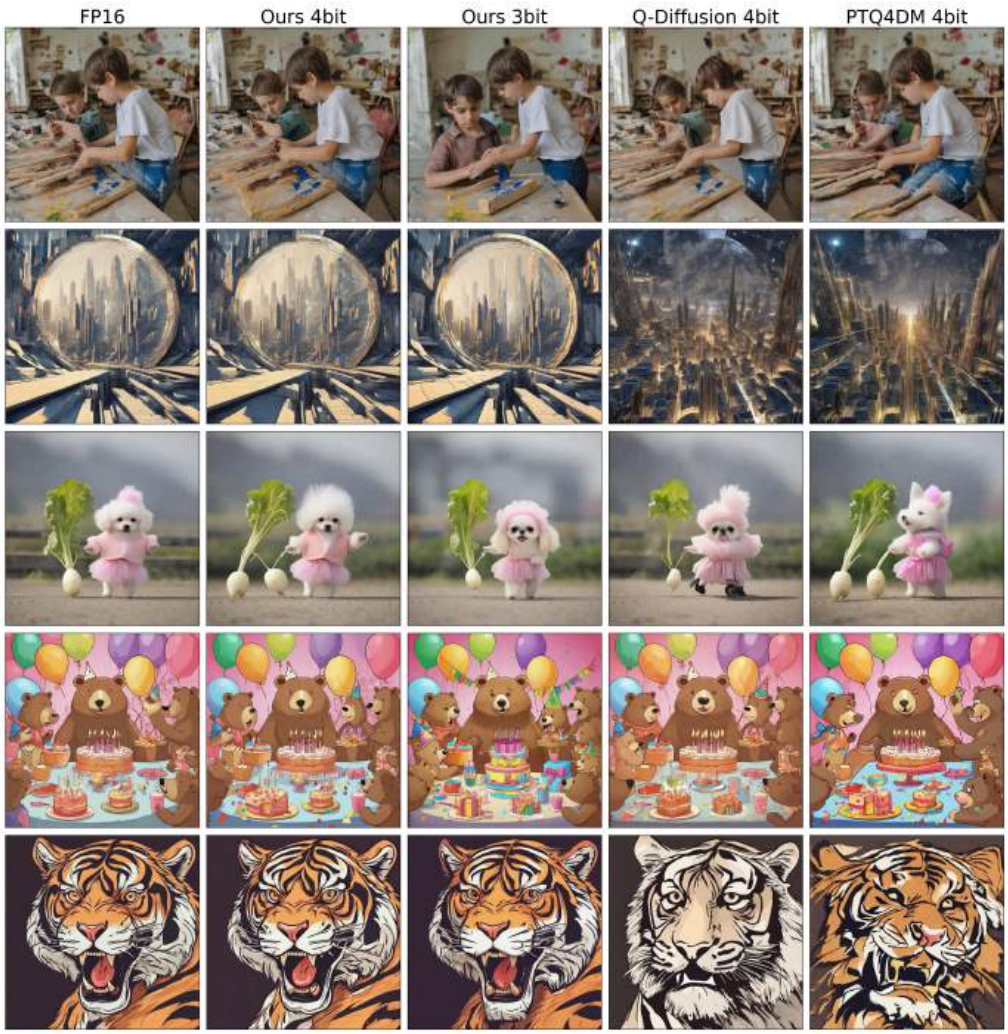

Figure 2: Qualitative comparison of SDXL compressed with VQDM and the baselines.

As a compression measure, we report an average number of bits per model weight in all convolutional and linear layers, including the unquantized ones. The codebook sizes are also included, divided by the overall number of weights.

For VQDM, if not otherwise stated, we set the group size and number of bits per codebook to 8 and vary the number of codebooks from 2 to 4 to achieve $2{-}4$ bit compression, respectively.

4.2 Comparison with baseline methods

Here, we evaluate our approach on one of the most widely adopted open source text-to-image models, SDXL, containing 2.6B parameters and compare VQDM to previous PTQ approaches.

Baselines. We select Q-Diffusion (Li et al., 2023b) as our primary baseline from the family of uniform scalar PTQ techniques. The quantization is applied to all linear and convolutional layers in the U-Net model, followed by a calibration process. To form the calibration set, Q-Diffusion splits sampling time steps into equal intervals and uses one time step from each interval for each of the n prompts. Additionally, Q-Diffusion proposes "shortcut-splitting" in up-sample ResNet blocks of U-Net to mitigate their abnormal activation distribution.

Changing the strategy for collecting timesteps for calibration to sampling from a normal distribution skewed to $t = T_0$ and removing "shortcut-splitting" results in PTQ4DM method (Shang et al., 2023), another baseline we compare with. In contrast to these methods, we use all timesteps for each calibration prompt. Another difference is that we do not quantize time embedding layers.

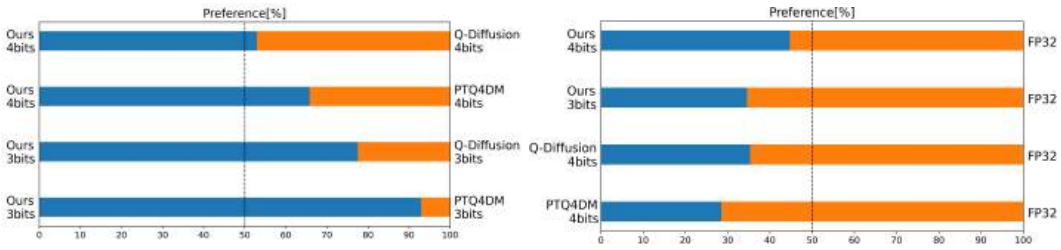

Figure 3: **Human preference study. Left.** Comparison between VQDM and the baselines. **Right.** Comparison between the quantized and full-precision models.

While EfficientDM (He et al., 2024) and QuEST (Wang et al., 2024) are also competitive baselines, adapting their code to text-to-image generation and the SDXL model requires significant effort. Therefore, we left the comparison with these methods for future work.

Experimental results. First, we compare our method with the uncompressed base model. Table 1 reveals that VQDM allows to efficiently compress SDXL to 4.15 bits per parameter with a minor drop in visual quality and prompt alignment. According to the human study, the samples generated with the 4.15 bit VQDM model were almost indistinguishable for human annotators, resulting in an SbS score of 44.5% with a p-value > 0.1 for the null hypothesis claiming that both models perform equally. The 3.15 bit model also demonstrates highly promising performance, being marginally worse than the full precision model in terms of SbS score.

Then, we adapt the Q-Diffusion code for the SDXL U-Net architecture and calibrate the quantized U-Net using 4 bits for weight quantization and keeping activations in full precision to match our setup. We use $n = 64$ calibration prompts and run calibration for 10000 iterations for each layer and block. These hyperparameters were tuned to match the training time with our method without compromising the quality of the baselines.

In terms of automated metrics, the results in Table 1 indicate that 4 bit VQDM outperform both 4-bit baselines while 3-bit matches their performance. Human evaluation in Figure 3 and qualitative comparison in Figure 2 support this claim.

4.3 Inference performance

In Table 2, we report the dequantization runtime overheads and released memory reduction for the SDXL model using our GPU and CPU implementations supporting VQDM.

For GPU measurements, we perform half-precision SDXL inference and use batch size 8 to saturate GPU utilization. The experiments are run on a single NVIDIA A100 80Gb. Compared to the FP16 model, VQDM allows up-to $5\times$ memory reduction at the cost of $\sim 50\%$ runtime overhead.

For CPU measurements, we run full-precision SDXL inference since CPUs generally do not support effective operations in half-precision. The experiments are run on a single core of an Intel Sapphire Rapids CPU. We observe that memory reductions of up to $9.7\times$ can be realized in practice with relatively low runtime overheads ($23-26\%$).

	GPU implementation FP16		CPU implementation FP32	
Avg. bits	**Runtime overhead**	**Memory reduction**	**Runtime overhead**	**Memory reduction**
4.15	49%	3.76×	26%	7.42×
3.15	48%	5.06×	23%	9.70×

Table 2: Inference results for our GPU and CPU implementations, showing decompression overheads and released memory savings for 3-4-bit VQDM configurations.

4.4 Ablation

Next, we investigate the influence of the fine-tuning and architecture choices for VQDM on final model performance. The results in Table 4 fine-tuning consistently increases the compression quality in terms of both automated metrics and the side-by-side evaluation (SbS score), especially for higher compression rates. According to Table 3, the effect of non-quantizing the timestep embedding layers is less significant. Note that we omit the SbS scores for non-finetuned versions of 2- and 3-bit VQDM

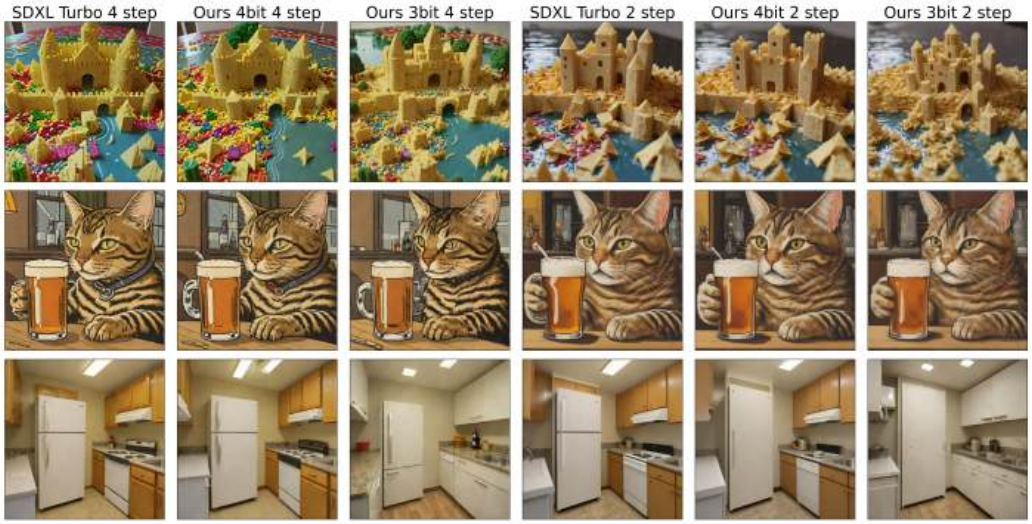

Figure 4: Qualitative comparison of SDXL-Turbo quantized with VQDM and the full-precision model for different sampling steps.

Method	Avg bits	Pickscore↑	CLIP↑	FID↓	SbS,% ↑
VQDM-all	4.06	0.226	0.357	18.88	43.2
	3.05	0.225	0.355	19.30	31.1
VQDM	4.15	0.226	0.356	19.11	44.7
	3.15	0.225	0.355	19.18	34.6

Table 3: Comparison of VQDM with and without quantization of the timestep embedding layers. VQDM-all denotes quantizing the timestep embedding layers.

Method	Avg bits	Pickscore↑	CLIP↑	FID↓	SbS,% ↑
Original model	32	0.226	0.357	18.99	50.0
VQDM w/ FT	4.15	0.226	0.356	19.11	44.7
	3.15	0.225	0.355	19.18	34.6
	2.15	0.219	0.341	22.14	9.2
VQDM w/o FT	4.15	0.225	0.355	20.24	34.2
	3.15	0.221	0.347	24.88	–
	2.15	0.206	0.287	86.88	–

Table 4: Comparison of VQDM quantization settings before and after fine-tuning (FT).

since their metrics and our preliminary analysis of the generated images shown that they were not competitive with the teacher model.

4.5 Distilled diffusion models

Finally, we apply our approach to SDXL-Turbo (Sauer et al., 2023b) operating in different sampling steps. We present the results in Table 5 and mark green the SbS scores providing p-value > 0.1 for the null hypothesis that both models perform equally. The results show that the distilled diffusion models can be compressed using VQDM to 4 bits without noticeable loss in performance.

Method	Steps	Avg bits	Pickscore↑	CLIP↑	FID↓	SbS,%↑
Original model	4	32	0.229	0.355	24.49	50.0
VQDM	4	4.15	0.228	0.355	24.89	43.9
VQDM	4	3.15	0.226	0.352	26.55	29.9
Original model	2	32	0.230	0.357	26.43	50.0
VQDM	2	4.15	0.230	0.357	26.43	46.5
VQDM	2	3.15	0.230	0.357	26.59	30.1
Original model	1	32	0.228	0.359	26.07	50.0
VQDM	1	4.15	0.226	0.357	26.30	45.5
VQDM	1	3.15	0.221	0.346	30.64	26.6

Table 5: Evaluation of the quantized SDXL-Turbo model for different bit-widths and sampling steps. Green indicates p-value > 0.1 for the null hypothesis that the models perform equally.

5 CONCLUSION

We introduce VQDM, a vector quantization method combined with a fine-tuning procedure aimed at compressing modern text-to-image diffusion models to low bit-widths. We take into account the specific architecture and inference process of diffusion models and adapt vector quantization to better suit these types of models. VQDM outperforms baseline methods in $3-4$ bit compression of SDXL. Notably, our 3-bit compressed models perform on par with previous 4-bit quantization methods. Additionally, we demonstrate that vector quantization can be effectively applied to distilled diffusion models.

REFERENCES

Ahmed F AbouElhamayed, Angela Cui, Javier Fernandez-Marques, Nicholas D Lane, and Mohamed S Abdelfattah. Pqa: Exploring the potential of product quantization in dnn hardware acceleration. *arXiv preprint arXiv:2305.18334*, 2023.

Artem Babenko and Victor Lempitsky. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 931–938, 2014.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

Black Forest Labs. Flux.1. `https://huggingface.co/black-forest-labs/FLUX.1-dev`, 2024.

Davis Blalock and John Guttag. Multiplying matrices without multiplying. In *International Conference on Machine Learning*, pp. 992–1004. PMLR, 2021.

D. Burton, J. Shore, and J. Buck. A generalization of isolated word recognition using vector quantization. In *ICASSP '83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pp. 1021–1024, 1983. doi: 10.1109/ICASSP.1983.1171915.

Hanwen Chang, Haihao Shen, Yiyang Cai, Xinyu Ye, Zhenzhong Xu, Wenhua Cheng, Kaokao Lv, Weiwei Zhang, Yintong Lu, and Heng Guo. Effective quantization for diffusion models on cpus. *ArXiv*, abs/2311.16133, 2023. URL `https://api.semanticscholar.org/CorpusID:265466086`.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. Quip: 2-bit quantization of large language models with guarantees, 2023.

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2023.

Junsong Chen, Yue Wu, Simian Luo, Enze Xie, Sayak Paul, Ping Luo, Hang Zhao, and Zhenguo Li. Pixart-δ: Fast and controllable image generation with latent consistency models, 2024.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016.

Yongjian Chen, Tao Guan, and Cheng Wang. Approximate nearest neighbor search by residual vector quantization. *Sensors*, 10(12):11259–11273, 2010.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization, 2024.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. *CoRR*, abs/2403.03206, 2024.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgO66VKDS.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=d4f40zJJIS.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training compression for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*, 2022a.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal Brain Compression: a framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 36, 2022b.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate Quantization for Generative Pre-trained Transformers. *International Conference on Learning Representations (ICLR)*, 2023.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference, 2021.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

R. Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984. doi: 10.1109/MASSP.1984.1162229.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pp. 770–778. IEEE, June 2016. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459.

Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. PTQD: Accurate post-training quantization for diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Y3g1PV5R9l.

Yefei He, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. EfficientDM: Efficient quantization-aware fine-tuning of low-bit diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=UmMa3UNDAz.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *CoRR*, abs/2104.08718, 2021. URL https://arxiv.org/abs/2104.08718.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. URL http://arxiv.org/abs/1706.08500.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Yushi Huang, Ruihao Gong, Jing Liu, Tianlong Chen, and Xianglong Liu. Tfmq-dm: Temporal feature maintenance quantization for diffusion models. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Sergey Kastryulin, Artem Konev, Alexander Shishenya, Eugene Lyapustin, Artem Khurshudov, Alexander Tselousov, Nikita Vinokurov, Denis Kuznedelev, Alexander Markovich, Grigoriy Livshits, Alexey Kirillov, Anastasiia Tabisheva, Liubov Chubarova, Marina Kaminskaia, Alexander Ustyuzhanin, Artemii Shvetsov, Daniil Shlenskii, Valerii Startsev, Dmitrii Kornilov, Mikhail Romanov, Artem Babenko, Sergei Ovcharenko, and Valentin Khrulkov. Yaart: Yet another art rendering technology, 2024.

Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review*, 53:5455–5516, 2020.

Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. Bk-sdm: Architecturally compressed stable diffusion for efficient text-to-image generation. *ICML Workshop on Efficient Systems for Foundation Models (ES-FoMo)*, 2023. URL https://openreview.net/forum?id=bOVydU0XKC.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=G5RwHpBUv0.

Hao Li, Yang Zou, Ying Wang, Orchid Majumder, Yusheng Xie, R. Manmatha, Ashwin Swaminathan, Zhuowen Tu, Stefano Ermon, and Stefano Soatto. On the scalability of diffusion-based text-to-image generation, 2024.

Senmao Li, Taihang Hu, Fahad Shahbaz Khan, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. Faster diffusion: Rethinking the role of unet encoder in diffusion models, 2023a.

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 (ICCV)*, pp. 17535–17545, October 2023b.

Yanjing Li, Sheng Xu, Xianbin Cao, Xiao Sun, and Baochang Zhang. Q-DM: An efficient low-bit quantized diffusion model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023c. URL https://openreview.net/forum?id=sFGkL5BsPi.

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. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=POWv6hDd9XH.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

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. *arXiv preprint arXiv:2206.00927*, 2022.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Julieta Martinez, Joris Clement, Holger H Hoos, and James J Little. Revisiting additive quantization. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 137–153. Springer, 2016.

Julieta Martinez, Shobhit Zakhmi, Holger H Hoos, and James J Little. Lsq++: Lower running time and higher recall in multi-codebook quantization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 491–506, 2018.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14297–14306, 2023.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training, 2018.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7197–7206. PMLR, 13–18 Jul 2020.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *CoRR*, abs/2106.08295, 2021a. URL `https://arxiv.org/abs/2106.08295`.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization, 2021b.

Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In *International Conference on Machine Learning*, 2022. URL `https://api.semanticscholar.org/CorpusID:247595112`.

William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

Pablo Pernias, Dominic Rampas, Mats L. Richter, Christopher J. Pal, and Marc Aubreville. Wuerstchen: An efficient architecture for large-scale text-to-image diffusion models, 2023.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=di52zR8xgf`.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

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.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=08Yk-n5l2Al.

Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. StyleGAN-T: Unlocking the power of GANs for fast large-scale text-to-image synthesis. volume abs/2301.09515, 2023a. URL https://arxiv.org/abs/2301.09515.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation, 2023b.

Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation, 2024.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022.

Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *CVPR*, 2023.

Junhyuk So, Jungwon Lee, Daehyun Ahn, Hyungjun Kim, and Eunhyeok Park. Temporal dynamic quantization for diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=D1sECc9fiG.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Siao Tang, Xin Wang, Hong Chen, Chaoyu Guan, Zewen Wu, Yansong Tang, and Wenwu Zhu. Post-training quantization with progressive calibration and activation relaxing for text-to-image diffusion models, 2023.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks, 2024.

Mart van Baalen, Andrey Kuzmin, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. Gptvq: The blessing of dimensionality for llm quantization, 2024.

Haoxuan Wang, Yuzhang Shang, Zhihang Yuan, Junyi Wu, and Yan Yan. Quest: Low-bit diffusion model quantization via efficient selective finetuning, 2024.

Peisong Wang, Qiang Chen, Xiangyu He, and Jian Cheng. Towards accurate post-training network quantization via bit-split and stitching. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, Christian Rupprecht, Daniel Cremers, Peter Vajda, and Jialiang Wang. Cache me if you can: Accelerating diffusion models through block caching, 2024.

Yuewei Yang, Xiaoliang Dai, Jialiang Wang, Peizhao Zhang, and Hongbo Zhang. Efficient quantization strategies for latent diffusion models, 2023.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5, 2022.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *NeurIPS*, 2023.

Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps, 2024.

Rui-Jie Zhu, Yu Zhang, Ethan Sifferman, Tyler Sheaves, Yiqiao Wang, Dustin Richmond, Peng Zhou, and Jason K Eshraghian. Scalable matmul-free language modeling. *arXiv preprint arXiv:2406.02528*, 2024.

A IMPLEMENTATION DETAILS

When applying this algorithm to large text-to-image models, we employ several tricks to make calibration more scalable. We use half-precision to quickly generate inference trajectories for the calibration set but run the rest of the calibration in full precision to avoid numeric issues. For fine-tuning, we opt for mixed precision Micikevicius et al. (2018) for forward and backward passes. On top of that, we use gradient checkpointing Chen et al. (2016) and gradient accumulation to reduce the memory footprint during fine-tuning.

We use Adam optimizer for layer-wise calibration and fine-tuning with standard parameters β_1 and β_2. During fine-tuning, we found that using a sufficiently large batch size is crucial for the training to be effective. Namely, we fine-tune with at least 32 samples per batch and use 2048 generated samples for training. The calibration dataset uses 256 prompts from the Pic-a-Pic dataset Kirstain et al. (2023).

For each quantization experiment, we used either the 2 NVIDIA H100 or 4 A100 GPUs. Quantizing the SDXL U-Net model took approximately 52h on 4 NVIDIA A100 and 36h on 2 NVIDIA H100. Fine-tuning the quantized model was conducted on 4 NVIDIA A100 and took from 8 to 28 hours, depending on the size of the fine-tuning dataset.

B HUMAN EVALUATION SETUP

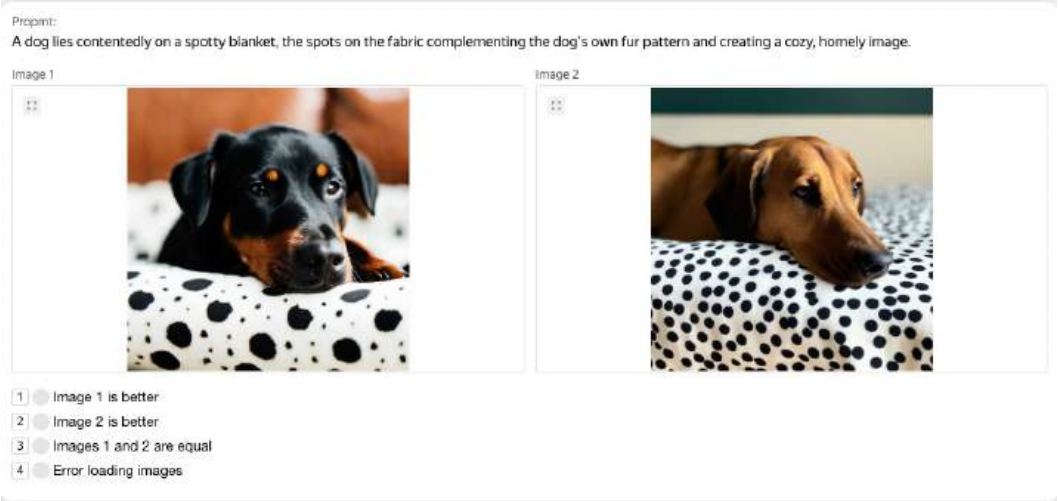

Figure 5: Side-by-side comparison interface for text-to-image human evaluation.

The evaluation is conducted by professional assessors where they are asked to make a decision between two images given a textual prompt. The decision is made according to image quality and textual alignment. For each evaluated pair, three responses are collected and the final prediction is determined by majority voting. We present the evaluation interface in Figure 5.

C LIMITATIONS

Due to the limited computational resources and costly calibration and fine-tuning procedures for large diffusion models, we only experiment with two popular diffusion models: SDXL and SDXL Turbo that represent latent diffusion models based on the efficient U-Net architecture Podell et al. (2024). In general, VQDM is applicable to arbitrary diffusion architectures, including the transformer-based ones Peebles & Xie (2022) that have also become widespread in practice Esser et al. (2024); Chen et al. (2023; 2024). We leave the application of our method to the transformer-based Peebles & Xie (2022) diffusion models for future work. We also exclude the quantization of activations from the scope of our work, as our primary goal is to reduce memory requirements for modern diffusion models.

D BROADER IMPACT

In this study, we propose a method for nearly lossless compression of diffusion models to $3-4$ bits, making the deployment of modern diffusion models feasible on general-purpose devices (e.g., gaming consoles or smartphones). This paves the way for numerous new means to incorporate generative technology into everyday life, such as text-controlled photo editing or augmented reality applications based on diffusion models without sending data from the user device to the third-party server for computation.

On the other hand, as our method is applicable to all diffusion-based generative models, it also shares all the possible negative societal impacts of this technology, such as the generation of fake images and videos or the spread of social and cultural biases inherited by diffusion models. However, we believe that our method does not introduce any new possibly harmful use cases for distributing text-to-image generation technology and would have a positive societal impact.

E ALGORITHMS

In this section, we describe both the quantization process (Algorithm 1) and the fine-tuning process (Algorithm 2). First, the layers in the U-Net are quantized using Algorithm 1. Then, we fine-tune the trainable weights of the quantized model to mimic the teacher's output (see Algorithm 2).

Algorithm 1 VQDM: Quantization stage.

Require: unet, prompt_embed
for $block \in$ `unet.blocks` **do**
  $\mathbf{X}_{block}$ = *collect_block_inputs*(`unet, prompt_embed`) `// sampling with current unet`
  for `layer in get_linear_and_conv(block)` **do**
    $\mathbf{W}$:= `layer.weight`
    $\mathbf{X}$:= `get_layer_inputs(layer,` $\mathbf{X}_{block}$`)`
    C, b, s := `initialize_codebooks(`$\mathbf{W}$`)` `// k-means`
    while $\mathcal{L} > \tau$ **do**
      C, s := `train_Cs_adam(`$X, \mathbf{W}, C, b, s$`)`
      b := `beam_search(`$X, \mathbf{W}, C, b, s$`)` (Egiazarian et al., 2024)
    end while
    `layer.weight` := `QuantizedLayer(`C, b, s`)`
  end for
end for

Algorithm 2 VQDM: Quantization-aware fine-tuning.

Require: unet_teacher, unet_student, prompt_embed
dataset := *get_sampling_trajectory*(*unet_teacher*, *prompt_embed*)
for $i = 1, \ldots,$ `num_epochs` **do**
  for *latents_batch* `in dataset` **do**
    `calculate_loss(`*unet_teacher*, *unet_student*, *latents_batch*, *prompt_embed*`)`
    optimize(*unet_student*)
  end for
end for

`func` **calculate_loss**
Require: unet_teacher, unet_student, latents, prompt_embeds
$\mathbf{Y}_{teacher}$ = `unet_teacher(`*latents*, *prompt_embeds*`)`
$\mathbf{Y}_{student}$ = `unet_student(`*latents_batch*, *prompt_embeds*`)`
$loss = ||\mathbf{Y}_{student} - \mathbf{Y}_{teacher}||_2^2$