

FraQAT: Quantization Aware Training with Fractional bits

Luca Morreale Alberto Gil C. P. Ramos Malcolm Chadwick Mehid Noroozi

Ruchika Chavhan

Abhinav Mehrotra

Sourav Bhattacharya

Samsung AI Center
Cambridge, UK

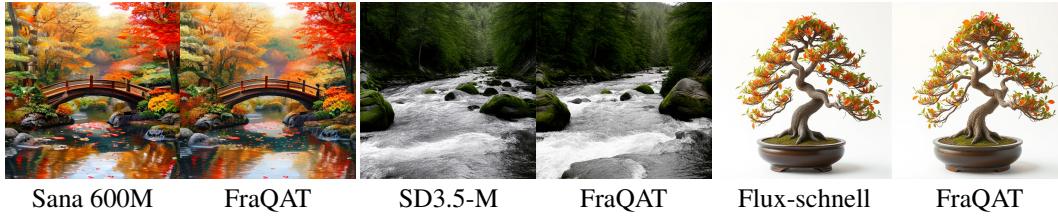


Figure 1: FraQAT is a Quantization aware Training (QAT) technique that grants generative models high fidelity at a fraction of training time required. Large text-to-image (T2I) models quantized with FraQAT (W4A8) achieve 16% lower FiD score than the state-of-the-art.

Abstract

State-of-the-art (SOTA) generative models have demonstrated impressive capabilities in image synthesis or text generation, often with a large capacity model. However, these large models cannot be deployed on smartphones due to the limited availability of on-board memory and computations. Quantization methods lower the precision of the model parameters, allowing for efficient computations, e.g., in INT8. Although aggressive quantization addresses efficiency and memory constraints, preserving the quality of the model remains a challenge. To retain quality in previous aggressive quantization, we propose a new fractional bits quantization (FraQAT) approach. The novelty is a simple yet effective idea: we progressively reduce the model’s precision from 32 to 4 bits per parameter, and exploit the fractional bits during optimization to maintain high generation quality. We show that the FraQAT yields improved quality on a variety of diffusion models, including SD3.5-Medium, Sana, PixArt- Σ , and FLUX.1-schnell, while achieving 4 – 7% lower FiD than standard QAT. Finally, we deploy and run Sana on a Samsung S25U, which runs on the Qualcomm SM8750-AB Snapdragon 8 Elite Hexagon Tensor Processor (HTP).

1 Introduction

Over the past few years, generative models have made impressive progress in synthesizing high-quality images [1, 2, 3] and texts [4, 5]. Such a breakthrough is partly achieved by enlarging the model’s size, e.g., Diffusion Transformer (DiT) models with over 10 billion (10B) parameters are increasingly common. However, larger models require significantly more resources, hence higher inference-time or latency, even for inference. This increase is particularly problematic for deploying these models on resource limited devices, e.g., smartphones, thus limiting their wide-scale usage.

A well-established approach to mitigate these resource constraints is *quantization*: by shifting parameters from 32 bits to a lower precision, e.g., 4 bits, the model’s weight-allocation footprints in its computational graph is significantly reduced. While past quantization research aimed mostly at decreasing model size, low-precision hardware support, such as NPUs on smartphones, drives researchers to further decrease inference latency. For example, latency gains from reduced data movement are boosted by native support for low-precision operations, e.g., using 4-bit weights and 8-bit activations (W4A8). Although, initially few devices offered support for these operations, modern hardware manufacturers readily offer low precision operations across devices: W4A8 in Qualcomm Snapdragon HTP [6], INT8, BF16 and FP16 in Intel CPUs [7], Block FP16 in AMD CPUs [8] and FP8/FP4 in NVIDIA GPU H100/H200 [9] to name a few.

The advantages of deploying cloud-quality generative models on-device are multi-fold: it preserves users’ privacy while offering a low-latency experience. For service providers, it reduces operating costs by pushing compute from expensive servers to users’ personal devices as well as avoiding violating country-specific privacy regulations. In this work, we target mobile deployment, and restrict ourselves to W4A8 given its ubiquitous availability across devices.

Quantization approaches fall under two main categories: Post Training Quantization (PTQ) and Quantization-Aware-Training (QAT). PTQ creates a low-precision model from a high precision pre-trained model using a small calibration dataset. Recent progress in PTQ research has resulted in W4A32¹ and W8A32² high quality quantized models from pre-trained SANA [10], SANA 1.5 [2] and SANA-Sprint [11]. Mixed precision W4A32 and W16A32 approaches like SVDQuant [12] have also yielded high quality quantized models from pre-trained FLUX.1-schnell. In essence, PTQ is ideal for cases where access to a large training dataset or compute cluster is limited.

Despite its success, PTQ requires careful data selection [13]. For example, a poorly selected calibration dataset may manifest in poor prompt adherence or exhibit color shifts during deployment. Instead, Quantization-Aware-Training (QAT) optimizes weights in lower precision to boost the overall model’s performance [14, 15, 16]. In general, QAT approaches yield better results, at lower precision, when a large training dataset or a compute cluster is available. Nonetheless, quantized models suffer from a quality loss compared to the original FP32 model.

We propose fractional bits quantization (FraQAT) to bridge the quality gap between the original and the quantized model. Inspired by Curriculum Learning [17], our training process progressively increases the quantization complexity, i.e., gradually lowers parameter precision, while replicating the original model’s output. We show that FraQAT reduces outliers, stabilizes training and yields improved prompt adherence and image generation quality (Section 2.2). We apply FraQAT to the linear layers of SOTA generative models as they contain the majority of the parameters, and empirically demonstrate the advantages of the proposed techniques on diffusion models (Section 4.1, 4.2). In terms of image quality, FraQAT achieves 16% lower FID than SOTA QAT. To address computational costs, we perform an outlier analysis (Section 4.3), and selectively train a subset of the model’s layers and show its performance. Finally, we quantize and deploy a model on a Samsung S25U, running on Qualcomm SM8750-AB Snapdragon 8 Elite Hexagon Tensor Processor (HTP) (Section 4.4).

2 Method

2.1 Quantization preliminaries

The goal of quantization is to approximate – in dynamic or static finite precision – internal model operations, such as operations within linear layers $\mathbf{x} \times \mathbf{W}$ where $\mathbf{x} \in \mathbb{R}^{B \times m}$ and $\mathbf{W} \in \mathbb{R}^{m \times n}$. Depending on hardware support, the quantization operation \mathbf{W}_b on a matrix \mathbf{W} to b bits can be expressed with narrower range as:

¹https://github.com/NVlabs/Sana/blob/main/asset/docs/4bit_sana.md

²https://github.com/NVlabs/Sana/blob/main/asset/docs/8bit_sana.md

$$\begin{aligned}
\mathcal{Q}(\mathbf{W})_b &:= \left\lfloor \frac{2^{b-1} - 1}{\max_{i,j} |[\mathbf{W}]_{i,j}|} \mathbf{W} \right\rfloor \in \{-2^{b-1}, \dots, 2^{b-1} - 1\} \\
\mathcal{S}(\mathbf{W})_b &:= \frac{\max_{i,j} |[\mathbf{W}]_{i,j}|}{2^{b-1} - 1} \in \mathbb{R}^+ \\
\mathbf{W}_b &:= \mathcal{S}(\mathbf{W})_b \mathcal{Q}(\mathbf{W})_b
\end{aligned} \tag{1}$$

or with wider range as:

$$\begin{aligned}
\mathcal{Q}(\mathbf{W})_b &:= \left\lfloor 2^b \frac{\mathbf{W} - w_{\min}}{w_{\max} - w_{\min}} \right\rfloor \in \{0, \dots, 2^b - 1\} \\
\mathcal{S}(\mathbf{W})_b &:= \frac{w_{\max} - w_{\min}}{2^b} \in \mathbb{R}^+ \\
\mathbf{W}_b &:= \mathcal{S}(\mathbf{W})_b \mathcal{Q}(\mathbf{W})_b + w_{\min}
\end{aligned} \tag{2}$$

where $w_{\min} := \min_{i,j} [\mathbf{W}]_{i,j}$ and $w_{\max} := \max_{i,j} [\mathbf{W}]_{i,j}$. Most simply for (1), matrix multiplications can be rewritten as: $x_{b_x} \mathbf{W}_{b_W} = (\mathcal{S}(x)_{b_x} \mathcal{S}(\mathbf{W})_{b_W})(\mathcal{Q}(x)_{b_x} \mathcal{Q}(\mathbf{W})_{b_W})$ where b_x and b_W may differ. Therefore, matrix multiplication $x_{b_x} \mathbf{W}_{b_W}$ can be reduced to the multiplication of two floats $\mathcal{S}(x)_{b_x} \mathcal{S}(\mathbf{W})_{b_W}$ and matrix multiplication of two integer matrices $\mathcal{Q}(x)_{b_x} \mathcal{Q}(\mathbf{W})_{b_W}$. Furthermore, we refer to dynamic quantization when w_{\min} and w_{\max} are computed at runtime, per sample, based on the input. While, in static quantization, w_{\min} and w_{\max} are pre-computed and shared across all samples. Dynamic quantization, especially when applied to activations, allow to robustly handle outliers as each sample range is computed to maximize representability. On the other hand, static quantization are more restrictive, and generate more outliers, thus making the quantization problem strictly harder. Edge devices, such as smartphones, only support static quantization, while GPUs support both.

In contrast, activations are often quantized through a look-up table mapping from a 2^b sized partition of the input range into a fixed number of quantized output values, e.g., the previous layer output x . In general, weights and activations may be quantized to different precisions, upcasted to the same precision before computation and downcasted after computation.

Hereafter, we make the number of bits in weights and activations explicit with subscripts, e.g. x_{32} refers to a 32 bit approximation of x .

Due to restricted address spaces in most mobile accelerators, it is critical to decrease weights precision aggressively, especially in large vision or language models, e.g., 12 billion parameter models, otherwise these models cannot even be placed on the target devices. However, naively lowering the weight's precision from FP32 to INT4 causes severe degradation in the generated results. This degradation is exacerbated by lowering activations precision, as required by integer-only accelerators, most often to INT8 for reduced generation latency. At a high level, the quality degradation phenomenon is attributed to outliers in both activations and weights due to training. The overall challenge of quantization is to approximate the original network's behavior while lowering the precision:

$$x\mathbf{W} \approx x_8 \mathbf{W}_4. \tag{3}$$

2.2 Fractional Quantization-Aware-Training

Intuitively, Quantization-Aware-Training (QAT) approaches – including the proposed FraQAT – handle outliers, both in weights and activations, by shifting parameters to quantization centroids within or towards adjacent bins. Hence, re-distributing weights in a more compact space. Consequently, the further apart bins are, the harder the optimization problem. We further speculate that it is slower to optimize for lower precisions – INT4 vs FP32 – as the gap between two adjacent representable numbers is much larger. Indeed, it can be observed from Figure 2 that the loss is higher for lower precisions, thus, outliers appear gradually.

To address this issue, we take inspiration from Curriculum Learning [17] literature: we progressively increase the complexity of the task during the optimization by gradually lowering weights' precision while approximating the full precision model's output. This is achieved by two key designs: first FraQAT leverages weights from pre-trained models. Second, FraQAT continuously steps between

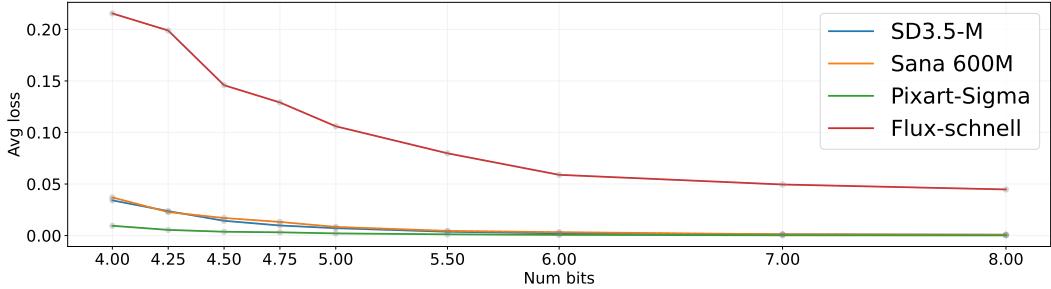


Figure 2: **Bit vs Loss:** as we reduce the precision (number of bits), the average knowledge distillation loss increases – the gap between student and teacher widens. From a quantization perspective, this implies outliers incrementally affect the student model.

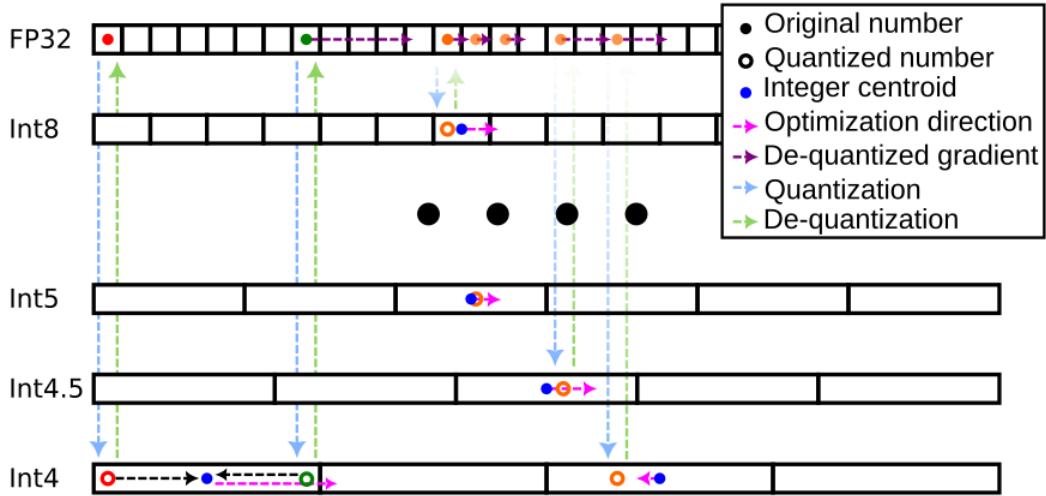


Figure 3: Classic QAT first computes the loss computed at the **lower precision** (\circ), then propagates it back to the original precision and optimize the weights (\rightarrow). This results in coarse and noisy gradients. Fractional Quantization Aware Training rely on intermediate precisions (**from INT8 to INT4 as training progresses**) to incrementally adjust to weights distributions. Consequently, during training Parameters smoothly shift between **bins** (\bullet) thanks to the finer gradients from in intermediate precision.

discrete quantization ranges to exploit the fact that Eq. (1)–(2) are purely a software construct, hence, it is possible to span any continuous — not just discrete — precision $b \in [32; 1] \in \mathbb{R}$.

Coupled together, these concepts establish FraQAT as a faster and higher quality QAT scheme dubbed Fractional Quantization Aware Training (FraQAT). Given a model, FraQAT progressively lowers the precision, first coarsely between FP32 and INT8 and then finely from INT8 to INT4, stepping through intermediate fractional bits during training as depicted in Figure 3 and in Algorithm 1. As the training progresses, outliers gradually appear, as shown in Figure 2, and are addressed. Furthermore, by optimizing at fractional bit precision in a curriculum fashion, FraQAT allows the weights to move to stable configurations, yielding higher quality samples, and reducing training time. Finally, throughout the entire training process FraQAT keeps all activation quantization constant (INT8).

As training progresses, this progressive lowering of precision, smoothly moves weights distribution and thus facilitates quantization (c.f., Figure 3). Note that it is possible to even set $b = 5.5$. Although half-bits precision have no meaning, in practice they bridge the gap of range of representable numbers between two precisions: INT6 $\in [-32; 31]$, INT5.5 $\in [-22; 21]$, INT5 $\in [-16; 15]$. In other words, half-bits precisions reduce the distance between adjacent bins, speeding up convergence without ad-hoc hyper-parameters, such as learning rate.

The proposed Fractional Quantization-Aware-Training (FraQAT) approach is generally applicable to any model and quantization level. Given the wide-spread usage of DiT and MM-DiT blocks in SOTA T2I models, we focus the presentation on DiT models. Since model size limits must first be met for any on-device placement, FraQAT quantizes linear layers as they contain the bulk of the parameters of DiT models (99.9%). In particular, FraQAT targets the most aggressive W4A8 quantization, as it allows for a wider range of models to fit edge accelerators with the lowest generation latency. Nonetheless, the proposed technique allows to target any precision.

3 Related works

Large diffusion models are the de-facto framework for image generation [18, 19, 20, 21]. On the other hand, Large Language Models (LLMs) shows human-like abilities with text [4, 22, 23]. However quality and diversity comes at a cost: these models have a huge amount of parameters and cannot be hosted on an on-device NPU without some form of quantization.

Tackling computational complexity Diffusion models’ computational complexity has two major sources: the amount of denoising passes and the conditioning mechanism. The former issue can be addressed by distilling the model to few or a single pass [24, 25], while the latter by modeling the latent-noise space [26] to decrease the number of function evaluations. Despite the success of these approaches, a major bottleneck remains: the memory required for inference. Quantization aim to preserve the original model’s quality when moving to lower precision – thus saving memory and enabling deployment.

Quantization-Aware-Training QAT methods optimize model’s weights at lower precision [15, 27] aiming to recover the original performance. Early approaches [15, 27] study QAT on ResNet for classification: starting from low bit precision – $b = 2$ or 4 – the weights quantization is progressively reduced [27] or selected at random[15]. Although [15, 27] closely relate to FraQAT, they (i) focus on classification networks, (ii) ignore the gap with full precision models and the hierachal nature of different precisions by starting from a low-bit quantization, (iii) aim to get models at different precisions. In parallel, Fracbits [28] introduces bit-width optimization by relying on a non-standard quantization formula for fractional bits. Bit-width are regularized to achieve the desired precision, followed by a binary search and fine tuning process to finalize weights. However, this procedure focuses on average bit length across the layers, obtaining lower bits in some layers at the cost of higher bits in other layers, which may not map to readily available hardware. Finally, as [15, 27], Fracbits focuses on classification rather than generative tasks.

More recently, MatryoshkaQAT [14] exploits the nested structure of a number’s byte representation to encode LLM’s weights at different precision – 8, 4, and 2. The joint training at the three precisions result in a multi-precision model. Parallel to this work, Liu et al. [29] extend [30], and discover that model quantized to lower than 4 bits develop a different representation from the original models. Finally, [31] based on the Teaching Assistant distillation framework [32], quantize LLM models to

Algorithm 1 Fractional Quantization Aware Training

Input: Pre-trained model \mathcal{M}_{W32A32} , dataset \mathcal{D} , loss function \mathcal{L} , quantization schedule \mathcal{B} (e.g., $\{8, 5, 4.5, 4\}$), optimizer \mathcal{O}

Output: Quantized model \mathcal{M}_{W4A8}

- 1: $\mathcal{M}_{WbA8} \leftarrow \mathcal{M}_{W32A32}$
- 2: **for** $b \in \mathcal{B}$ **do**
- 3: $\mathcal{M}_{WbA8} \leftarrow \text{QUANTIZELINEARLAYER}(\mathcal{M}_{WbA8}, \text{WbA8})$
- 4: **for** batch $\in \mathcal{D}$ **do**
- 5: $O_{WbA8} \leftarrow \text{FORWARD}(\mathcal{M}_{WbA8}, \text{batch})$
- 6: $O_{W32A32} \leftarrow \text{FORWARD}(\mathcal{M}_{W32A32}, \text{batch})$
- 7: $l \leftarrow \mathcal{L}(\text{stop_grad}(O_{W32A32}), O_{WbA8})$
- 8: $\text{OPTIMIZE}(\mathcal{O}, l, \mathcal{M}_{WbA8})$
- 9: **end for**
- 10: **end for**
- 11: **return** \mathcal{M}_{WbA8}

W1A1. Similar to our work, the authors use a progressive strategy, however limited to W1A4 (to W1A2) to W1A1, where intermediate models (W1A2) models are used as teachers. Combined with a series of techniques (e.g., gradient clipping, elastic binarization, etc) to stabilize the optimization process the authors achieve a binary quantized model. Since the quantization is binary, the model is not deployable to edge-devices. In this work, we show progressive quantization is enough to quantize a model that can be deployed on edge-devices.

Related to diffusion models, Bitfusion [16] combines different QAT techniques, such as distillation and fine tuning, to convert SD1.5 [19] to 1.99 bits. Following a similar trend, Wang et al. in [33] selectively fine tune SD1.5 to handle activation distribution. BinaryDM [34] takes model quantization one step further by applying a multi-stage QAT approach to quantization. Notwithstanding these impressive results, none of these techniques showcase low-bit quantization of large scale DiT models such as SD3.5-M [1] (2.2B) or FLUX.1-schnell [3] (12B). Ours is the first QAT approach applied to such models.

Post-Training Quantization Among the recent seminal works on LLMs, SmoothQuant [35] proposes a PTQ approach by injecting a smoothing factor in linear layers to reduce the impact of outliers in LLMs. AWQ [36] and MobileQuant [37] extend this approach to lowers the precision to W4A8 thus enabling an LLM to run on-device. These works have been extended to DiT models with specific focus on timesteps. PTQ4DiT [38] builds a calibration dataset by sampling timesteps before quantizing the diffusion model. DiTAS [39] proposes a temporal-aggregated smoothing technique combined with LoRA and a grid-search to reduce quantization errors of small DiT networks with W4A8 quantization. QuEST [33] through layer specific (PTQ) fine tuning achieves W4A4 quantization. Q-DiT [40], inspired by [36, 41, 37], combines a fine-grained group quantization with a novel automatic allocation algorithm to account for weights’ spatial variance. Most recently, SVDQuant [12] and FBQuant [42] have shown impressive preservation of image quality generation when quantizing FLUX.1-schnell [3] to W4A16. The authors rely on a low-rank approximation of the original weights and a residual branch to absorb outliers.

4 Experiments

Models We focus our evaluation on recent text to image (T2I) models since there is an increasing interest in lowering their computational requirements due to their large number of parameters. In particular, we assess the soundness of the proposed approach over 4 different diffusion models: SD3.5-Medium [1], Sana [10], PixArt- Σ [43], and FLUX.1-schnell [3]. These models space a wide range of parameters, 0.6B–12B, and architectural innovations, linear and non-linear attentions, DiT, MM-DiT, etc.

In all our experiments, we start from a pre-trained W32A32 model and through FraQAT reduce it to W4A8. We bootstrap the student at INT8 and optimize it to replicate its FP32 counter-part. This initialization allows FraQAT to start with minimal gap between teacher (FP32) and student (INT8). After T epochs we lower the precision of the model – number of bits –, and continue the optimization. This procedure is repeated until the precision reaches 4 bits. Since we apply a *false quantization* process, we can emulate arbitrary precisions that have no hardware support, e.g., INT4.5. Unless stated otherwise all our experiments follow the same progression: $8 \rightarrow 7 \rightarrow 6 \rightarrow 5.5 \rightarrow 5 \rightarrow 4.75 \rightarrow 4.5 \rightarrow 4.25 \rightarrow 4$ targeting linear layers. Further hyper-parameters are detailed in the appendix. In all cases we distil a W4A8 model through knowledge-distillation loss, using dynamic quantization. Note that the proposed approach is applicable to any quantization precision, e.g., W2A8, and static quantization.

Baselines We compare the proposed approach with state of the art PTQ techniques: DiTAS [39] (W4A8) and SVDQuant [12](W4A16). In both cases, we use the code publicly available and train (calibrate) the model over the train dataset (see below). To further prove the soundness of FraQAT, we implement a vanilla QAT (vQAT) approach, and an SVDQuant-like QAT (SVDQAT). In vQAT, we apply a W4A8 quantization to all linear layers and optimize them with the same loss as FraQAT. Instead in SVDQAT, we inject a LoRA-like layer in all linear layers, as in [12], and optimize both the low-rank and residual branch – almost doubling the number of parameters. Finally, we report the results for the naive quantization (Dynamic Q.) to desired precision through torchao³.

³<https://github.com/pytorch/ao>

Table 1: **Qualitative evaluation:** we evaluate FraQAT using a *fractional quantization schedule* on PixArt Evaluation dataset [43] and MidJourney HQ Evaluation dataset [44] measuring FID, and CLIP-FID wrt the original model, and ImageReward (IR) [45].

PixArt- Σ															
		SD3.5 Medium				Sana 600M				PixArt- Σ			Flux-schnell		
Method	Precision	FID \downarrow	CLIP FID \downarrow	IR \uparrow	FID \downarrow	CLIP FID \downarrow	IR \uparrow	FID \downarrow	CLIP FID \downarrow	IR \uparrow	FID \downarrow	CLIP FID \downarrow	IR \uparrow		
Dynamic Q.	W4A8	9.36	2.08	0.56	2.22	0.24	0.57	13.35	6.19	0.35	8.17	1.13	-0.73		
DiTAS	W4A8	27.93	13.77	0.41	12.87	4.58	0.62	7.30	3.95	0.84	-	-	-		
SVDQuant	W4A16	14.42	3.14	0.66	2.43	0.24	0.60	6.80	2.02	0.79	2.26	0.36	0.84		
SVDQAT	W4A8	2.57	0.28	0.80	1.93	0.13	0.48	5.38	1.48	0.76	-	-	-		
vQAT	W4A8	2.67	0.31	0.78	2.13	0.16	0.45	7.00	2.52	0.79	3.40	0.66	0.87		
FraQAT	W4A8	2.54	0.27	0.82	2.17	0.19	0.48	4.48	1.07	0.79	2.55	0.30	0.86		
MJHQ															
		SD3.5 Medium				Sana 600M				PixArt- Σ			Flux-schnell		
Method	Precision	FID \downarrow	CLIP FID \downarrow	IR \uparrow	FID \downarrow	CLIP FID \downarrow	IR \uparrow	FID \downarrow	CLIP FID \downarrow	IR \uparrow	FID \downarrow	CLIP FID \downarrow	IR \uparrow		
Dynamic Q.	W4A8	10.29	2.11	0.65	2.40	0.28	0.63	15.04	5.55	0.44	8.66	1.24	-0.90		
DiTAS	W4A8	32.04	14.06	0.41	12.91	5.59	0.68	8.63	4.07	1.04	-	-	-		
SVDQuant	W4A16	15.10	3.06	0.78	2.48	0.25	0.62	6.95	1.71	0.99	2.41	0.41	0.96		
SVDQAT	W4A8	2.85	0.32	0.91	2.04	0.16	0.53	5.83	1.44	0.96	-	-	-		
vQAT	W4A8	3.01	0.37	0.89	2.13	0.20	0.47	7.38	2.12	0.99	3.56	0.73	0.99		
FraQAT	W4A8	2.78	0.32	0.96	2.34	0.24	0.50	4.95	1.054	0.97	2.55	0.39	0.99		

Datasets All models in all our experiments are trained – calibrated – on YE-POP⁴ dataset. We split it between training (97.5%) and validation (2.5%). Then, quantized models are evaluated on two different datasets: PixArt- Σ Evaluation dataset [43]⁵, and MidJourney HQ Evaluation dataset [44]. In all cases during training and evaluation, we generate 512×512 images.

Metrics We quantitatively evaluate the proposed technique over a variety of metrics measuring image quality, and features distributions. In particular, we measure the image quality with Image Reward (IR) [45], and measure the features distribution disparity between the generated samples of the quantized model and the original one with FID [46] and CLIP-FID [47]. This choice allows us to quantify the similarity between the original model and its quantized version: similar images have similar features – thus lower FID score.

4.1 Quantitative evaluation

Table 1 shows a quantitative comparison of FraQAT across the aforementioned five models, two SOTA QAT approaches for direct comparison, and three PTQ techniques for overall completeness across all quantization approaches. All combinations are evaluated across two different test datasets. Due to memory requirements, we are unable to apply some techniques to Flux-schnell [3] (12B model).

SVDQuant was developed and optimized for Sana, PixArt- Σ and FLUX.1-schnell, thus achieves lower performance in SD3.5-Medium, as shown in Table 1, yielding particularly worse FID and CLIP FID metrics for both test datasets. Table 1 reveals mixed results for Dynamic Quantization and DiTaS on both test datasets. Specifically, DiTaS outperforms Dynamic Quantization in PixArt- Σ but reveals itself overall worse for SD3.5-Medium and Sana. These models have different architectures, namely DiT, MM-DiT and linear attention, which indicates that DiTaS can be particularly sensitive to the model family.

As for other QAT approaches, our developed SVDQuant-like QAT (SVDQAT) consistently outperforms vanilla QAT (vQAT) across the test datasets. Arguably, the increased number of parameters – LORA and residual branch – better cope with the lower precision. Our motivation to establish this strong QAT baseline was the success of SVDQuant as a PTQ approach alongside its need of a higher precision W4A16 that hinders latency in on-device accelerators. Finally, Table 1 shows FraQAT

⁴<https://huggingface.co/datasets/Ejafa/ye-pop>

⁵<https://huggingface.co/datasets/PixArt-alpha/PixArt-Eval-30K>

outperforms even the strongest QAT baseline we developed namely SVDQAT, with overall higher gains for SD3.5-Medium and PixArt- Σ for both test datasets.



Figure 4: **Qualitative comparison:** FraQAT(d) generates images similar to the original model (a). Prompts are from MJHQ dataset [44].

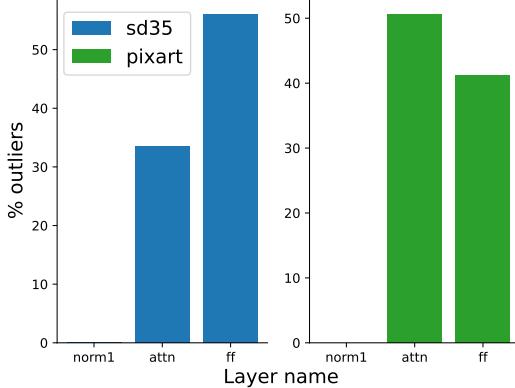


Figure 5: **Outliers:** outliers distribution for activations varies across models. SD3.5-M (left) experience most of its outliers right after Feed Forward layers, while for PixArt- Σ , most outliers are in Attention layers.

Table 2: **Outlier analysis:** we optimize specific layers types while the rest of the model is frozen and quantized (W4A8). FID and CLIP-FID are computed on PixArt- Σ [43] evaluation dataset.

Model	Layer	FID \downarrow	CLIP FID \downarrow
SD3.5-M	FF	2.23	0.23
	Attn	2.32	0.24
	TF	2.49	0.28
	All	2.54	0.22
Sana 600M	FF	2.18	0.17
	Attn	2.10	0.16
	TF	2.13	0.16
	All	2.17	0.19
PixArt- Σ	FF	5.34	1.55
	Attn	6.48	2.23
	TF	4.40	1.13
	All	4.48	1.07

4.2 Qualitative evaluation

To complement the quantitative evaluation in the preceding section, Figure 4 depicts for each of the four models considered: the original model, one PTQ representative (SVDQuant), one QAT alternative (vQAT) and the proposed QAT approach (FraQAT). We have selected SVDQuant among DiTaS and Dynamic Quantization given its popularity. We focus on vQAT rather than our other developed QAT baseline (SVDQAT) given its overall popularity. Note that, the images generated in each row in Figure 4 share the same seed and prompt.

As expected from a PTQ approach, SVDQuant under-performs when generating certain high frequency image details. This is especially visible when multiple faces are present in a generated image as shown in the first row. QAT improves high frequency image details, but generates significantly different images than the original model for the same prompt and seed. Finally, FraQAT preserves both high frequency details and generates images as close to the original model across all baselines.

4.3 Outlier analysis

Activation outliers disrupt the quantization process by introducing artifacts or biases. By analyzing these outliers across different models, we discover that different models produce outliers in different layers. For example, in SD3.5-M outliers emerge after Feed-Forward (FF) layers, while in PixArt- Σ outliers arise from Attention (Attn) layers, see Figure 5. Through selectively training specific layers, we can *reduce FraQAT’s computational demand* while obtaining a deployable model. In this vein, we analyze the impact of selective training, i.e., we optimize only certain layers while the rest of the network is *frozen and quantized* (W4A8). In particular, we focus on attention layers (Attn), feed forward layers (FF), and transformer blocks (TF), and compare it with training the entire network (Full).

Quantitative results in Table 2 show that there is no clear winner – a layer type for all architecture. Different models take advantage from optimizing different layers. Nevertheless we recommend starting from quantizing Transformer Blocks (TF) as it reduces memory requirements, lowers computational demands, and addresses all outliers.

4.4 On device model deployment

To demonstrate the feasibility of deploying models quantized with FraQAT on edge devices, we quantized Sana 600M [10] to W4A8 and deployed it on the Samsung S25U, which runs on the Qualcomm SM8750-AB Snapdragon 8 Elite Hexagon Tensor Processor (HTP). Compared to CPUs and GPUs, integer accelerators support a limited range of precisions and exclusively support static



Figure 6: **On device generation:** we generate images on a mobile phone (c) and compare the results with the samples generated on GPU by the original model (a) and the quantized model (b).

quantization for both weights and activations. Please, refer to Section 2.1, for a discussion on static and dynamic quantization.

Conversely to the baselines used in Section 4.1, FraQAT support both dynamic and static quantization paradigms. To apply FraQAT to Sana 600M [10] with static weight and activation quantization, we pre-compute scale and offset through statistical analysis of the features: we select 100 random samples and pass them through the DiT. Feature values for every layer are recorded and used to compute standard deviation and mean. Following literature [48], we use a 3 std range for inlier. Finally, we use these scale and offset during the QAT, while the overall training procedure does not change from the one discussed in Section 4.1.

All linear layers of the quantized model run with precision W4A8 except for the last layer which runs in W4A16. This is a good compromise to preserve quality, while not impacting latency. Overall, the model has a latency of $66ms$ per forward step, while the same model running in W4A16 (bit-width supported by SVDQuant) has a latency of $95ms$ – a 30.5% latency improvement. Finally, to assess the on-device quality we generate samples, and compare them with those from the original model in Figure 6. The quantized model produces high quality pictures that resemble the original model and its GPU version.

5 Limitations and future work

The proposed approach is a step forward compared to SOTA Quantization-Aware-Training (QAT). Like most – if not all – QAT techniques, FraQAT yields higher quality but is more computational expensive than Post-Training Quantization (PTQ). Compared with multi-precision LLM’s SOTA QAT approaches, such as MatryoshkaQAT, FraQAT’s quantized model is tailored to a single bit precision, and we leave multi-precision support for future work. Furthermore, the intermediate precision levels are hand-picked, in the future, we plan to design an algorithm to select the most impactful precisions.

The proposed training scheme may benefit from regularizers such as weight decay and data augmentation. For example, preliminary regularization tests on Sana 0.6B [10] show that weight decay

boosts the performance by $\sim 10\%$. A proper investigation of regularization and its impact in training is left to future work. Finally, FraQAT’s networks are optimized using knowledge distillation only, but different losses such as feature and task loss may further boost image generation quality.

6 Conclusions

FraQAT is a novel Quantization-Aware-Training technique that exploits fractional bits while progressively reducing the parameter-precision during the quantization process. Thanks to this curriculum learning strategy, we address the outliers, as they arise, at different precisions, achieving a more stable and faster training. The proposed method is evaluated over a variety of state-of-the-art DiT and MM-DiT models. We show that, both qualitatively and quantitatively, the quantized models achieve superior performance compared to the state-of-the-art QAT approaches. Such improved quality, if deployed on-device, may boost mobile users productivity, preserve their privacy, as well as generate personalized contents for users.

References

- [1] S. AI, “Stable diffusion 3.5.” <https://stability.ai/news/introducing-stable-diffusion-3-5>, 2024.
- [2] E. Xie, J. Chen, Y. Zhao, J. Yu, L. Zhu, C. Wu, Y. Lin, Z. Zhang, M. Li, J. Chen, H. Cai, B. Liu, D. Zhou, and S. Han, “SANA 1.5: Efficient Scaling of Training-Time and Inference-Time Compute in Linear Diffusion Transformer,” [arXiv:2501.18427](https://arxiv.org/abs/2501.18427), 2025.
- [3] B. F. Labs, “Flux.” <https://github.com/black-forest-labs/flux>, 2024.
- [4] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al., “The llama 3 herd of models,” [arXiv preprint arXiv:2407.21783](https://arxiv.org/abs/2407.21783), 2024.
- [5] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, et al., “Gemma 3 technical report,” [arXiv preprint arXiv:2503.19786](https://arxiv.org/abs/2503.19786), 2025.
- [6] Q. Snapdragon®, “Snapdragon 8 Elite Mobile Platform.” https://docs.qualcomm.com/bundle/publicresource/87-83196-1_REV_D_Snapdragon_8_Elite_Mobile_Platform_Product_Brief.pdf.
- [7] X. Intel®, “Processors with Performance-Cores (P-Cores).” <https://www.intel.com/content/www/us/en/products/details/processors/xeon/xeon6-p-cores.html>.
- [8] A. Ryzen™, “AI 300 Series Processors.” <https://www.amd.com/content/dam/amd/en/documents/partner-hub/ryzen/amd-ryzen-ai-300-series-vs-qualcomm-snapdragon-x-elite-deck.pdf>.
- [9] N. HGX™, “NVIDIA HGX Platform.” <https://www.nvidia.com/en-gb/data-center/hgx>.
- [10] E. Xie, J. Chen, J. Chen, H. Cai, H. Tang, Y. Lin, Z. Zhang, M. Li, L. Zhu, Y. Lu, and S. Han, “SANA: Efficient High-Resolution Image Synthesis with Linear Diffusion Transformers,” [arXiv:2410.10629](https://arxiv.org/abs/2410.10629), 2024.
- [11] J. Chen, S. Xue, Y. Zhao, J. Yu, S. Paul, J. Chen, H. Cai, E. Xie, and S. Han, “SANA-Sprint: One-Step Diffusion with Continuous-Time Consistency Distillation,” [arXiv:2503.09641](https://arxiv.org/abs/2503.09641), 2025.
- [12] M. Li, Y. Lin, Z. Zhang, T. Cai, X. Li, J. Guo, E. Xie, C. Meng, J.-Y. Zhu, and S. Han, “SVDQuant: Absorbing Outliers by Low-Rank Components for 4-Bit Diffusion Models,” [arXiv:2411.05007](https://arxiv.org/abs/2411.05007), 2024.
- [13] Z. Zhang, Y. Gao, J. Fan, Z. Zhao, Y. Yang, and S. Yan, “Selectq: Calibration data selection for post-training quantization,” [Machine Intelligence Research](https://www.mir.org.cn/2025/01/2025010100001.pdf), pp. 1–12, 2025.
- [14] P. Nair, P. Datta, J. Dean, P. Jain, and A. Kusupati, “Matryoshka Quantization,” [arXiv:2502.06786](https://arxiv.org/abs/2502.06786), 2025.
- [15] A. Bulat and G. Tzimiropoulos, “Bit-Mixer: Mixed-precision networks with runtime bit-width selection,” in [ICCV](https://openaccess.thecvf.com/content/ICCV2021/html/Bulat_Bit-Mixer_Mixed-Precision_Networks_With_Runtime_Bit-Width_Selection_ICCV_2021_paper.html), pp. 5188–5197, 2021.

- [16] Y. Sui, Y. Li, A. Kag, Y. Idelbayev, J. Cao, J. Hu, D. Sagar, B. Yuan, S. Tulyakov, and J. Ren, “Bitsfusion: 1.99 bits weight quantization of diffusion model,” [arXiv:2406.04333](#), 2024.
- [17] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in [Proceedings of the 26th annual international conference on machine learning](#), pp. 41–48, 2009.
- [18] P. Dhariwal and A. Nichol, “Diffusion models beat gans on image synthesis,” [Advances in neural information processing systems](#), vol. 34, pp. 8780–8794, 2021.
- [19] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-Resolution Image Synthesis with Latent Diffusion Models,” in [CVPR](#), pp. 10684–10695, 2022.
- [20] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, “Flow Matching for Generative Modeling,” [arXiv:2210.02747](#), 2023.
- [21] I. Gat, T. Remez, N. Shaul, F. Kreuk, R. T. Chen, G. Synnaeve, Y. Adi, and Y. Lipman, “Discrete flow matching,” [Advances in Neural Information Processing Systems](#), vol. 37, pp. 133345–133385, 2024.
- [22] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al., “Gpt-4 technical report,” [arXiv preprint arXiv:2303.08774](#), 2023.
- [23] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al., “Gemini: a family of highly capable multimodal models,” [arXiv preprint arXiv:2312.11805](#), 2023.
- [24] T. Salimans and J. Ho, “Progressive distillation for fast sampling of diffusion models,” [arXiv preprint arXiv:2202.00512](#), 2022.
- [25] M. Noroozi, I. Hadji, B. Martinez, A. Bulat, and G. Tzimiropoulos, “You Only Need One Step: Fast Super-Resolution with Stable Diffusion via Scale Distillation,” in [ECCV](#), pp. 145–161, Springer, 2025.
- [26] M. Noroozi, A. G. Ramos, L. Morreale, R. Chavhan, M. Chadwick, A. Mehrotra, and S. Bhattacharya, “Guidance free image editing via explicit conditioning,” [arXiv preprint arXiv:2503.17593](#), 2025.
- [27] Q. Jin, L. Yang, and Z. Liao, “AdaBits: Neural Network Quantization with Adaptive Bit-Widths,” in [CVPR](#), pp. 2146–2156, 2020.
- [28] L. Yang and Q. Jin, “Fracbits: Mixed precision quantization via fractional bit-widths,” in [Proceedings of the AAAI Conference on Artificial Intelligence](#), vol. 35, pp. 10612–10620, 2021.
- [29] Z. Liu, C. Zhao, H. Huang, S. Chen, J. Zhang, J. Zhao, S. Roy, L. Jin, Y. Xiong, Y. Shi, et al., “ParetoQ: Scaling Laws in Extremely Low-bit LLM Quantization,” [arXiv:2502.02631](#), 2025.
- [30] M. Nagel, M. Fournarakis, Y. Bondarenko, and T. Blankevoort, “Overcoming Oscillations in Quantization-Aware Training,” in [ICML](#), pp. 16318–16330, PMLR, 2022.
- [31] Z. Liu, B. Oguz, A. Pappu, L. Xiao, S. Yih, M. Li, R. Krishnamoorthi, and Y. Mehdad, “Bit: Robustly binarized multi-distilled transformer,” [Advances in neural information processing systems](#), vol. 35, pp. 14303–14316, 2022.
- [32] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, “Improved knowledge distillation via teacher assistant,” in [Proceedings of the AAAI conference on artificial intelligence](#), vol. 34, pp. 5191–5198, 2020.
- [33] H. Wang, Y. Shang, Z. Yuan, J. Wu, J. Yan, and Y. Yan, “QuEST: Low-bit Diffusion Model Quantization via Efficient Selective Finetuning,” [arXiv:2402.03666](#), 2024.
- [34] X. Zheng, X. Liu, H. Qin, X. Ma, M. Zhang, H. Hao, J. Wang, Z. Zhao, J. Guo, and M. Magno, “BinaryDM: Accurate Weight Binarization for Efficient Diffusion Models,” [arXiv:2404.05662](#), 2024.
- [35] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, “SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models,” [arXiv:2211.10438](#), 2024.
- [36] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, “AWQ: Activation-aware Weight Quantization for On-device LLM Compression and Acceleration,” [Proceedings of Machine Learning and Systems](#), vol. 6, pp. 87–100, 2024.

- [37] F. Tan, R. Lee, Ł. Dudziak, S. X. Hu, S. Bhattacharya, T. Hospedales, G. Tzimiropoulos, and B. Martinez, “MobileQuant: Mobile-friendly Quantization for On-device Language Models,” [arXiv:2408.13933](#), 2024.
- [38] J. Wu, H. Wang, Y. Shang, M. Shah, and Y. Yan, “PTQ4DiT: Post-training Quantization for Diffusion Transformers,” [arXiv:2405.16005](#), 2024.
- [39] Z. Dong and S. Q. Zhang, “DiTAS: Quantizing Diffusion Transformers via Enhanced Activation Smoothing,” [arXiv:2409.07756](#), 2024.
- [40] L. Chen, Y. Meng, C. Tang, X. Ma, J. Jiang, X. Wang, Z. Wang, and W. Zhu, “Q-DiT: Accurate Post-Training Quantization for Diffusion Transformers,” [arXiv:2406.17343](#), 2024.
- [41] Y. Zhao, C.-Y. Lin, K. Zhu, Z. Ye, L. Chen, S. Zheng, L. Ceze, A. Krishnamurthy, T. Chen, and B. Kasikci, “Atom: Low-bit quantization for efficient and accurate llm serving,” [Proceedings of Machine Learning and Systems](#), vol. 6, pp. 196–209, 2024.
- [42] Y. Liu, H. Fang, L. He, R. Zhang, Y. Bai, Y. Du, and L. Du, “FBQuant: FeedBack Quantization for Large Language Models,” [arXiv:2501.16385](#), 2025.
- [43] J. Chen, C. Ge, E. Xie, Y. Wu, L. Yao, X. Ren, Z. Wang, P. Luo, H. Lu, and Z. Li, “Pixart- σ : Weak-to-strong training of diffusion transformer for 4k text-to-image generation,” in [European Conference on Computer Vision](#), pp. 74–91, Springer, 2024.
- [44] D. Li, A. Kamko, E. Akhgari, A. Sabet, L. Xu, and S. Doshi, “Playground v2. 5: Three insights towards enhancing aesthetic quality in text-to-image generation,” [arXiv preprint arXiv:2402.17245](#), 2024.
- [45] J. Xu, X. Liu, Y. Wu, Y. Tong, Q. Li, M. Ding, J. Tang, and Y. Dong, “Imagereward: Learning and evaluating human preferences for text-to-image generation,” [Advances in Neural Information Processing Systems](#), vol. 36, pp. 15903–15935, 2023.
- [46] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in [Proceedings of the IEEE conference on computer vision and pattern recognition](#), pp. 2818–2826, 2016.
- [47] T. Kynkänniemi, T. Karras, M. Aittala, T. Aila, and J. Lehtinen, “The role of imagenet classes in fr\’echet inception distance,” [arXiv preprint arXiv:2203.06026](#), 2022.
- [48] R. Wang, Y. Gong, X. Liu, G. Zhao, Z. Yang, B. Guo, Z. Zha, and P. Cheng, “Optimizing large language model training using fp4 quantization,” [arXiv preprint arXiv:2501.17116](#), 2025.
- [49] Z. Lin, D. Pathak, B. Li, J. Li, X. Xia, G. Neubig, P. Zhang, and D. Ramanan, “Evaluating text-to-visual generation with image-to-text generation,” [arXiv preprint arXiv:2404.01291](#), 2024.
- [50] J. Wang, K. C. Chan, and C. C. Loy, “Exploring clip for assessing the look and feel of images,” in [Proceedings of the AAAI conference on artificial intelligence](#), vol. 37, pp. 2555–2563, 2023.
- [51] G. Team, “Gemma,” 2024.
- [52] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” [Journal of machine learning research](#), vol. 21, no. 140, pp. 1–67, 2020.
- [53] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, “Boolq: Exploring the surprising difficulty of natural yes/no questions,” [arXiv preprint arXiv:1905.10044](#), 2019.
- [54] A. Talmor, J. Herzig, N. Lourie, and J. Berant, “Commonsenseqa: A question answering challenge targeting commonsense knowledge,” [arXiv preprint arXiv:1811.00937](#), 2018.

A Experimental evaluation

A.1 Baselines

For state of the art baselines we rely on code released by authors⁶⁷ and use the default parameters. For all approaches we use pre-trained models with default resolution 512×512 . Where needed we change the baselines configurations to use the same model.

A.2 Hyper-parameters for QAT

We detail the various hyper parameters for all QAT experiments in Table 3. In all cases we rely on FuseAdam as optimizer and optimize for 25 epochs. All experiments run on AMD MI300X and are implemented using PyTorch⁸, Lightning⁹, torchao¹⁰, with seed 1234.

Table 3: **Hyper-parameters:** Detailed hyper-parameters required to replicate all experiments.

	SD3.5-M			Sana 600M			PixArt- Σ			
	lr	batch size	low rank	lr	batch size	low rank	lr	batch size	low rank	
SVDQAT	W4A8	10^{-5}	128	32	10^{-6}	128	16	10^{-6}	256	16
vQAT	W4A8	10^{-5}	256	-	10^{-6}	128	-	10^{-6}	128	-
FraQAT	W4A8	10^{-6}	256	-	10^{-7}	128	-	10^{-6}	128	-

For all FraQAT experiments, we follow the schedule highlighted in Table 4.

Table 4: **Precision schedule:** During training we progressively reduce the precision following the prescribed schedule.

Precision	8	7	6	5.5	5	4.75	4.5	4.25	4
# epochs	1	1	1	1	1	2	2	2	14

Experiments with the configuration highlighted above take on average 192 GPUh for Sana, 576 GPUh for PixArt- Σ , 1008 GPUh for SD3.5-Medium.

A.3 Qualitative evaluation

For additional qualitative evaluation on MJHQ dataset[44], please see the html pages in the zip file.

A.4 Quantitative evaluation

Here we report additional evaluation of the proposed approach with a wider set of metrics. In particular, we rely on VQA [49] to measure the adherence of the generated samples to the input prompts. We measure the image quality with CLIP-IQA [50].

Table 5 shows FraQAT outperforms even the strongest QAT baseline we developed namely SVDQAT, with overall higher gains for SD3.5-Medium and PixArt- Σ for both test datasets.

A.5 Quantization schedule

To validate the benefits of a *Fractional* quantization schedule (Table 4) we compare it with its *Integer* counterpart ($8 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 4$), and a simpler progressive schedule ($16 \rightarrow 8 \rightarrow 4$). For a fair comparison, all experiments have the same computational budget. We measure the validation

⁶SVDQuant <https://github.com/mit-han-lab/deepcompressor>

⁷DiTAS <https://github.com/DZY122/DiTAS>

⁸<https://pytorch.org/>

⁹<https://lightning.ai/docs/pytorch/stable/>

¹⁰<https://github.com/pytorch/ao>

Table 5: Qualitative evaluation: we evaluate FraQAT using a *fractional quantization schedule* on PixArt Evaluation dataset [43] and MidJourney HQ Evaluation dataset [44] measuring FID, and CLIP-FID wrt the original model, CLIP-IQA [50], ImageReward (IR) [45], and VQA [49].

Method	Precision	SD3.5 Medium						Sana 600M						PixArt- Σ						Flux-schnell						
		FID ↓	CLIP FID ↓	CLIP IQA ↑	IR ↑	VQA score ↑	FID ↓	CLIP FID ↓	CLIP IQA ↑	IR ↑	VQA score ↑	FID ↓	CLIP FID ↓	CLIP IQA ↑	IR ↑	VQA score ↑	FID ↓	CLIP FID ↓	CLIP IQA ↑	IR ↑	VQA score ↑	FID ↓	CLIP FID ↓	CLIP IQA ↑	IR ↑	VQA score ↑
Dynamic Q.	W4A8	9.36	2.08	0.44	0.56	0.84	2.22	0.24	0.46	0.57	0.82	13.35	6.19	0.44	0.35	0.82	8.17	1.13	0.43	-0.73	0.77					
DiTAS	W4A8	27.93	13.77	0.47	0.41	0.82	12.87	4.58	0.45	0.62	0.82	7.30	3.95	0.46	0.84	0.86	-	-	-	-	-					
SVDQuant	W4A16	14.42	3.14	0.42	0.66	0.85	2.43	0.24	0.43	0.60	0.82	6.80	2.02	0.43	0.79	0.86	2.26	0.36	0.42	0.84	0.85					
SVDQAT	W4A8	2.57	0.28	0.45	0.80	0.85	1.93	0.13	0.43	0.48	0.82	5.38	1.48	0.43	0.76	0.86	-	-	-	-	-					
vQAT	W4A8	2.67	0.31	0.44	0.78	0.85	2.13	0.16	0.43	0.45	0.81	7.00	2.52	0.45	0.79	0.85	3.40	0.66	0.41	0.87	0.86					
FraQAT	W4A8	2.54	0.27	0.45	0.82	0.86	2.17	0.19	0.42	0.48	0.82	4.48	1.07	0.45	0.79	0.86	2.55	0.30	0.41	0.86	0.85					
MJHQ																										
Method	Precision	SD3.5 Medium						Sana 600M						PixArt- Σ						Flux-schnell						
		FID ↓	CLIP FID ↓	CLIP IQA ↑	IR ↑	VQA score ↑	FID ↓	CLIP FID ↓	CLIP IQA ↑	IR ↑	VQA score ↑	FID ↓	CLIP FID ↓	CLIP IQA ↑	IR ↑	VQA score ↑	FID ↓	CLIP FID ↓	CLIP IQA ↑	IR ↑	VQA score ↑	FID ↓	CLIP FID ↓	CLIP IQA ↑	IR ↑	VQA score ↑
Dynamic Q.	W4A8	10.29	2.11	0.44	0.65	0.79	2.40	0.28	0.45	0.63	0.74	15.04	5.55	0.43	0.44	0.74	8.66	1.24	0.42	-0.90	0.65					
DiTAS	W4A8	32.04	14.06	0.47	0.41	0.73	12.91	5.59	0.45	0.68	0.75	8.63	4.07	0.46	1.04	0.80	-	-	-	-	-					
SVDQuant	W4A16	15.10	3.06	0.42	0.78	0.78	2.48	0.25	0.42	0.62	0.75	6.95	1.71	0.43	0.99	0.80	2.41	0.41	0.42	0.96	0.79					
SVDQAT	W4A8	2.85	0.32	0.45	0.91	0.80	2.04	0.16	0.43	0.53	0.74	5.83	1.44	0.43	0.96	0.81	-	-	-	-	-					
vQAT	W4A8	3.01	0.37	0.44	0.89	0.80	2.13	0.20	0.43	0.47	0.74	7.38	2.12	0.44	0.99	0.80	3.56	0.73	0.41	0.99	0.80					
FraQAT	W4A8	2.78	0.32	0.45	0.96	0.81	2.34	0.24	0.42	0.50	0.74	4.95	1.05	0.44	0.97	0.80	2.55	0.39	0.41	0.99	0.80					

loss across training and plot it in Figure 7. The integer and simple schedules perform comparably to each other. On the other hand, the Fractional schedule consistently outperforms the two competitors during training, resulting in a sensibly lower validation loss.

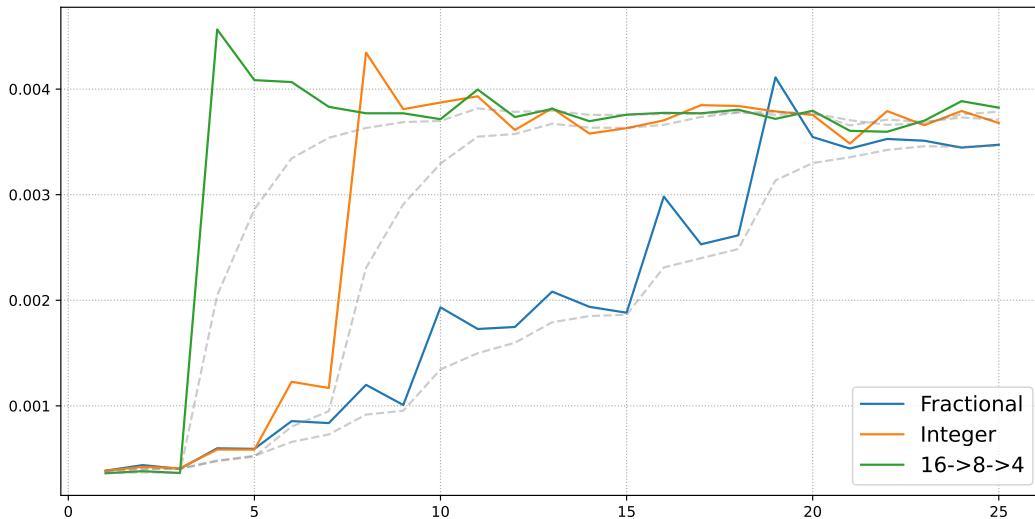


Figure 7: Fractional schedule: we train SD3.5-M using a simple progressive schedule (green), an integer schedule (orange), and a fractional schedule (blue). As seen in the graph, the Fractional schedule achieves a lower validation loss.

B Additional evaluation

B.1 Language Model

The proposed method is agnostic to the architecture and the application. We apply FraQAT to Gemma2 2B IT [51]¹¹. Start from the FP16 model – original –, then we quantize it to 4 bits in a similar fashion as we did with T2I models in the main paper. We follow the same schedule as in Section A.2. The quantized model (W4A8) is then compared with the original model.

As training set we rely on a subset of C4 dataset [52]: we pick randomly 384K samples for training and 38.4K samples for validation. The model is evaluated on two datasets in zero-shot fashion: BoolQ [53]¹², and Commonsense QA [54]¹³. Table 6 shows minimal drop when FraQAT is applied

¹¹<https://huggingface.co/google/gemma-2-2b-it>

¹²<https://huggingface.co/datasets/google/boolq>

¹³https://huggingface.co/datasets/allenai/social_i_qa

Table 6: **Evaluation on Language Models:** We apply FraQAT to Gemma2 2B IT [51] exactly as we did to T2I models. The quantized model is evaluated on Common Sense QA and BoolQ datasets. The resulting model has minimal drop from original language model.

Model	Precision	Common Sense QA \uparrow	BoolQ \uparrow	COQA \uparrow
Original	(W16A32)	0.70 \pm 0.01	0.76 \pm 0.01	0.66 \pm 0.01
FraQAT	(W4A8)	0.69 \pm 0.01	0.72 \pm 0.01	0.70 \pm 0.01

to Gemma2 2B IT model. Therefore, proving FraQAT can be applied to Language Models as well as Vision Models.