

DILATEQUANT: SUPPLEMENTARY MATERIALS

P.1

$$\begin{aligned}
E(X, W) &= \|XW - Q(X)Q(W)\|_F \\
&= \|XW - XQ(W) + XQ(W) - Q(X)Q(W)\|_F \\
&\leq \|X(W - Q(W))\|_F + \|(X - Q(X))Q(W)\|_F \\
&\leq \|X\|_F \|W - Q(W)\|_F + \|X - Q(X)\|_F \|Q(W)\|_F \\
&\leq \|X\|_F \|W - Q(W)\|_F + \|X - Q(X)\|_F \|W - (W - Q(W))\|_F \\
&\leq \|X\|_F \|W - Q(W)\|_F + \|X - Q(X)\|_F (\|W\|_F + \|W - Q(W)\|_F)
\end{aligned} \tag{18}$$

A SUPPLEMENTARY MATERIAL INTRODUCTION

In this supplementary material, we present the correlative introductions and some experiments mentioned in the paper. The following items are provided:

- Detailed experimental implementations for all experiments in Appendix B.
- Robustness of DilateQuant for time steps and samplers in Appendix C.
- Efficiency comparisons of various quantization frameworks in Appendix D
- Thorough comparison with EfficientDM and QuEST in Appendix E.
- Workflow and effects of Weight Dilation algorithm in Appendix F.
- Different equivalent scaling algorithms for diffusion models in Appendix G.
- Hardware-Friendly quantization in Appendix H.
- Human preference evaluation in Appendix I.

B DETAILED EXPERIMENTAL IMPLEMENTATIONS

In this section, we present detailed experimental implementations, including the pre-training models, quantization settings, and evaluation.

The DDPM¹ models and LDM² models we used for the experiments are obtained from the official websites. For text-guided generation with Stable-Diffusion, we use the CompVis codebase³ and its v1.4 checkpoint. The LDMs consist of a diffusion model and a decoder model. Following the previous works (Liu et al., 2024; He et al., 2023; Wang et al., 2024), DilateQuant focus only on the diffusion models and does not quantize the decoder models. We employ channel-wise asymmetric quantization for weights and layer-wise asymmetric quantization for activations. The input and output layers of models use a fixed 8-bit quantization, as it is a common practice. The weight and activation quantization ranges are initially determined by minimizing values error, and then optimized by our knowledge distillation strategy to align quantized models with full-precision models at block level. Since the two compared methods employ non-standard settings, we modify them to standard settings for a fair comparison. More specifically, we quantize all layers for EfficientDM, including Upsample, Skip_connection, and AttentionBlock’s qkvw, which lack quantization in open-source code⁴. However, when these layers, which are important for quantization, are added, the performance of EfficientDM degrades drastically. To recover performance, we double the number of training iterations. QuEST utilizes channel-wise quantization for activations at 4-bit precision in the code⁵, which is not supported by hardware. Therefore, we adjust the quantization setting to layer-wise quantization for activations. For experimental evaluation, we use open-source tool *pytorch-OpCounter*⁶ to calculate the Size and Bops of models before and after quantization.

¹<https://github.com/ermongroup/ddim>

²<https://github.com/CompVis/latent-diffusion>

³<https://github.com/CompVis/stable-diffusion>

⁴<https://github.com/ThisisBillhe/EfficientDM>

⁵<https://github.com/hatchetProject/QuEST>

⁶<https://github.com/Lyken17/pytorch-OpCounter>

And following the quantization settings, we only calculate the diffusion model part, not the decoder and encoder parts. We use the ADM’s TensorFlow evaluation suite *guided-diffusion*⁷ to evaluate FID, sFID, and IS, and use the open-source code *clip-score*⁸ to evaluate CLIP scores. As the per practice (Liu et al., 2024; Wang et al., 2024), we employ the zero-shot approach to evaluate Stable-Diffusion on COCO-val for the text-guided experiments, resizing the generated 512×512 images and validation images in 300×300 with the center cropping to evaluate FID score and using text prompts from COCO-val to evaluate CLIP score.

C ROBUSTNESS OF DILATEQUANT FOR TIME STEPS AND SAMPLERS

To assess the robustness of DilateQuant for samplers, we conduct experiments over LDM-4 on ImageNet with three distant samplers, including DDIMsampler Song et al. (2020), PLMSsampler Liu et al. (2022), and DPMSolversampler Lu et al. (2022). Given that time step is the most important hyperparameter for diffusion models, we also evaluate DilateQuant for models with different time steps, including 20 steps and 100 steps. As shown in Table 5, our method showcases excellent robustness across different samplers and time steps, leading to significant performance enhancements compared to previous methods. Specifically, our method outperforms the full-precision models in terms of FID and sFID at 6-bit quantization, and the advantages of our method are more pronounced compared to existing methods at the lower 4-bit quantization.

Table 5: The robustness of DilateQuant for time steps and samplers.

Task	Method	Calib.	Prec. (W/A)	FID ↓	sFID ↓	IS ↑
LDM-4 — DDIM time steps = 20	FP	-	32/32	11.69	7.67	364.72
	EDA-DM *	1024	6/6	11.52	8.02	360.77
	EfficientDM †	102.4K	6/6	8.69	8.10	309.52
	DilateQuant	1024	6/6	8.25	7.66	312.30
	EDA-DM *	1024	4/4	20.02	36.66	204.93
	EfficientDM †	102.4K	4/4	12.08	14.75	122.12
	DilateQuant	1024	4/4	8.01	13.92	257.24
	FP	-	32/32	11.71	7.08	379.19
	EDA-DM *	1024	6/6	11.27	6.59	363.00
LDM-4 — PLMS time steps = 20	EfficientDM †	102.4K	6/6	9.85	9.36	325.13
	DilateQuant	1024	6/6	7.68	5.69	315.85
	EDA-DM *	1024	4/4	17.56	32.63	203.15
	EfficientDM †	102.4K	4/4	14.78	9.89	103.34
	DilateQuant	1024	4/4	9.56	8.12	243.72
	FP	-	32/32	11.44	6.85	373.12
	EDA-DM *	1024	6/6	11.14	7.95	357.16
	EfficientDM †	102.4K	6/6	8.54	9.30	336.11
	DilateQuant	1024	6/6	7.32	6.68	330.32
LDM-4 — DPM-Solver time steps = 20	EDA-DM *	1024	4/4	30.86	39.40	138.01
	EfficientDM †	102.4K	4/4	14.36	13.82	109.52
	DilateQuant	1024	4/4	8.98	9.97	247.62
	FP	-	32/32	4.45	6.27	238.39
	EDA-DM *	1024	6/6	12.21	12.13	71.50
	EfficientDM †	102.4K	6/6	5.57	7.50	165.15
	DilateQuant	1024	6/6	5.97	7.44	162.93
	EDA-DM *	1024	4/4	N/A	N/A	N/A
	EfficientDM †	102.4K	4/4	20.70	11.79	72.67
LDM-4 — DDIM time steps = 100	DilateQuant	1024	4/4	9.85	10.79	147.63

⁷<https://github.com/openai/guided-diffusion>

⁸<https://github.com/Taited/clip-score>

D EFFICIENCY COMPARISONS OF VARIOUS QUANTIZATION FRAMEWORKS

We investigate the efficiency of DilateQuant across data resource, time cost, and GPU memory. We compare our method with PTQ-based method (Liu et al., 2024) and variant QAT-based method (He et al., 2023) on the mainstream diffusion models (DDPM, LDM, Stable-Diffusion). As reported in Table 6, our method performs PTQ-like efficiency, while significantly improving the performance of the quantized models. This provides an affordable and efficient quantization process for diffusion models.

Table 6: Efficiency comparisons of various quantization frameworks with 4-bit quantization across data resource, time cost, and GPU memory.

Model	Method	Calib.	Time Cost (hours)	GPU Memory (MB)	FID ↓
DDPM CIFAR-10	PTQ	5120	0.97	3019	120.24
	V-QAT	1.6384M	2.98	9546	81.27
	Ours	5120	1.08	3439	9.13
LDM ImageNet	PTQ	1024	6.43	13831	20.02
	V-QAT	102.4K	5.20	22746	12.08
	Ours	1024	6.56	14680	8.01
Stable-Diffusion MS-COCO	PTQ	512	7.23	30265	236.31
	V-QAT	12.8K	30.25	46082	216.43
	Ours	512	7.41	31942	42.97

E THOROUGH COMPARISON WITH EFFICIENTDM AND QUEST

EfficientDM (He et al., 2023) and QuEST (Wang et al., 2024) are two variance QAT-based methods, which achieve 4-bit quantization of the diffusion models with efficiency. However, both of them are non-standard. Specifically, EfficientDM preserves some layers at full-precision, notably the Upsample, Skip_connection, and the matrix multiplication of AttentionBlock’s qkvw. These layers have been demonstrated to have the most significant impact on the quantization of diffusion models in previous works (Shang et al., 2023; Li et al., 2023a; Liu et al., 2024). QuEST employs standard channel-wise quantization for weights and layer-wise quantization for activations at 6-bit precision. However, at 4-bit precision, it uses channel-wise quantization for the activations of all Conv and Linear layers, which is hardly supported by the hardware because it cannot factor the different scales out of the accumulator summation (please see Appendix H for details), leading to inefficient acceleration.

Table 7: Comparison with EfficientDM and QuEST in both standard and non-standard settings.

Task	Mode	Method	Prec. (W/A)	Size (MB)	FID ↓
LSUN-Church (Yu et al., 2015) 256 × 256	-	FP	32/32	1514.5	4.06
	Non-standard Not quantize for all layers	EfficientDM	6/6	315.0	6.29
		DilateQuant	6/6	315.0	4.73
		EfficientDM	4/4	222.7	14.34
		DilateQuant	4/4	222.7	8.68
	Standard Quantize for all layers	EfficientDM	6/6	284.9	6.92
		DilateQuant	6/6	284.9	5.33
		EfficientDM	4/4	190.3	15.08
DilateQuant		4/4	190.3	10.10	
LDM-8 steps = 100 eta = 0.0	Non-standard Channel-wise for A	QuEST	4/4	190.3	11.76
		DilateQuant	4/4	190.3	8.94
	Standard Layer-wise for A	QuEST	4/4	190.3	13.03
		DilateQuant	4/4	190.3	10.10

To thoroughly compare DilateQuant with EfficientDM and QuEST, we conduct experiments on LSUN-church with standard and non-standard quantization settings. When neglecting these layers that are important for quantization, DilateQuant extremely reduces the FID to 8.68 with 4-bit quantization. Compared to the standard setting, the performance improvement is more noticeable. When setting channel-wise quantization for activations, DilateQuant also reduces a 2.84 FID compared with QuEST. Conclusively, DilateQuant significantly outperforms EfficientDM and QuEST at different quantization precisions for both standard and non-standard settings, which demonstrates the stability and standards of DilateQuant.

F WORKFLOW AND EFFECTS OF WEIGHT DILATION ALGORITHM

The comprehensive workflow of Weight Dilation is illustrated in Algorithm 1. We implement WD in three steps: searching unsaturated channels for scaling (Lines 2-3), calculating scaling factor (Lines 5-10), and scaling activations and weights (Line 12). WD alleviates the wide range activations for diffusion models through a novel equivalent scaling algorithm. In addition, all operations of WD can be implemented simply, making it efficient.

Algorithm 1 Overall workflow of WD

Input: full-precision $\mathbf{X} \in \mathbb{R}^{N \times C^i}$ and $\mathbf{W} \in \mathbb{R}^{C^i \times C^o}$

Output: scaled \mathbf{X}' and \mathbf{W}' .

```

1: searching unsaturated channels for scaling:
2:   obtain  $W_{max} \in \mathbb{R}^{C^o}$  and  $W_{min} \in \mathbb{R}^{C^o}$ 
3:   record in-channel indexes of  $W_{max}$  and  $W_{min}$  as set  $A$ 
4: calculating scaling factor:
5:   for  $k = 1$  to  $C^i$  do
6:     if  $k \in A$ :
7:       set  $s_k = 1$ 
8:     else:
9:       calculate scaling factor  $s_k$  with  $W_{max}$  and  $W_{min}$  as constraints
10:   end for
11: scaling  $X$  and  $W$ :
12:   calculate  $\mathbf{X}' = \mathbf{X} / \mathbf{s}$  and  $\mathbf{W}' = \mathbf{W} \cdot \mathbf{s}$ 
13: return  $\mathbf{X}'$  and  $\mathbf{W}'$ 

```

We assess the effects of WD on various quantization tasks. As reported in Table 8, WD stably achieves $s > 1$ while maintaining $\Delta'_w \approx \Delta_w$. It effectively improves performance at different quantized models by losslessly reducing the activation quantization error.

Table 8: Effects of WD on different tasks with 4-bit quantization.

Tasks	CIFAR-10	LSUN-Bedroom	LSUN-Church	ImageNet	MSCOCO
Δ'_w / Δ_w	1.02	1.02	1.01	1.01	1.02
E'_{clip} / E_{clip}	0.83	0.92	0.92	0.93	0.92
proportion of $s > 1$	39.2%	52.4%	32.8%	36.5%	43.8%
Δ'_x / Δ_x	0.91	0.92	0.91	0.92	0.90
FID ↓	9.13 (-0.50)	8.99 (-0.25)	10.10 (-0.20)	8.01 (-0.27)	44.82 (-0.79)

G DIFFERENT EQUIVALENT SCALING ALGORITHMS FOR DIFFUSION MODELS

In this section, we start by analyzing the differences between LLMs and diffusion models in terms of the challenges of activation quantization. As shown in Figure 2(b), the activation outliers of the diffusion models are present in all channels, unlike in LLMs where the activation outliers only exist in fixed channels. Additionally, the range of activations for diffusion models is also larger than that of the LLMs. Therefore, it is essential to scale the number of channels as much as possible for the

diffusion models. Some equivalent scaling algorithms are proposed to smooth out the activation outliers in LLMs, and these methods have achieved success. SmoothQuant (Xiao et al., 2023a) scales all channels using a hand-designed scaling factor. AWQ (Lin et al., 2024) only scales a few of channels based on the salient weight. OmniQuant (Shao et al., 2023) proposes a learnable equivalent transformation to optimize the scaling factors in a differentiable manner. DGQ (Zhang et al., 2023a) devises a percentile scaling scheme to select the scaled channels and calculate the scaling factors. OS+ conducts channel-wise shifting and scaling across all channels.

Unfortunately, when we applied methods similar to these previous equivalent scaling algorithms to diffusion models, we find that none of them work. Specifically, we employ these five methods for diffusion models as follows: (1) For the method similar to SmoothQuant, we scale all channels before quantization using a smoothing factor $\alpha = 0.5$; (2) For the method similar to AWQ, we scale 1% of channels based on the salient weight, setting smoothing factor the same as SmoothQuant; (3) For the method similar to OmniQuant, we modify the scaling factors to be learnable variants and train them block by block with a learning rate of $1e-5$; (4) For the method similar to DGQ, we scale the top 0.5% of quantization-sensitive channels, setting scaling factor based on the clipping threshold. (5) For OS+, we perform shifting and scaling across all channels, consistent with the original work. However, as shown in Table 9, all of these methods result in higher FID and sFID scores compared to no scaling. The reason for this result is that although the range of activations decreases, the range of weights also increases significantly, making it more difficult for the model to converge during the training stage. In contrast, the Weight Dilation algorithm we proposed scales the number of channels as much as possible. It searches for unsaturated in-channel weights and dilates them to a constrained range based on the max-min values of the out-channel weights. The algorithm reduces the range of activations while maintaining the weights range unchanged. This effectively makes activation quantization easier and ensures model convergence, reducing the FID and sFID scores to 9.13 and 6.92 in 4-bit quantization, respectively.

Table 9: The results of various equivalent scaling algorithms for DDIM on CIFAR-10.

Prec.	Metrics	No scaling	SmoothQuant	OmniQuant	AWQ	DGQ	OS+	Ours
W4A4	proportion of $s > 1$	0%	100%	100%	1%	0.5%	100%	39.2%
	FID ↓	9.63	9.99	9.86	10.34	9.72	9.78	9.13
	sFID ↓	7.08	7.29	7.34	7.53	7.78	7.23	6.92
	IS ↑	8.45	8.46	8.50	8.38	8.52	8.36	8.56
W6A6	proportion of $s > 1$	0%	100%	100%	1%	0.5%	100%	39.2%
	FID ↓	5.75	5.44	5.56	5.85	5.09	5.81	4.46
	sFID ↓	4.96	4.87	4.89	5.19	4.84	4.99	4.64
	IS ↑	8.80	8.86	8.81	8.78	8.89	8.76	8.92

H HARDWARE-FRIENDLY QUANTIZATION

In this section, we investigate the correlation between quantization settings and hardware acceleration. We start with the principle of quantization to achieve hardware acceleration. A matrix-vector multiplication, $y = Wx + b$, is calculated by a neural network accelerator, which comprises two fundamental components: the processing elements $C_{n,m}$ and the accumulators A_n . The calculation operation of accelerator is as follows: firstly, the bias values b_n are loaded into accumulators; secondly, the weight values $W_{n,m}$ and the input values x_m are loaded into $C_{n,m}$ and computed in a single cycle; finally, their results are added in the accumulators. The overall operation is also referred to as Multiply-Accumulate (MAC):

$$A_n = \sum_m W_{n,m} x_m + b_n \quad (19)$$

where n and m represent the out-channel and in-channel of the weights, respectively. The pre-trained models are commonly trained using FP32 weights and activations. In addition to MAC calculations, data needs to be transferred from memory to the processing units. Both of them severely impact the speed of inference. Quantization transforms floating-point parameters into fixed-point parameters, which not only reduces the amount of data transfer but also the size and energy consumption

of the MAC operation. This is because the cost of digital arithmetic typically scales linearly to quadratically with the number of bits, and fixed-point addition is more efficient than its floating-point counterpart. Quantization approximates a floating-point tensor x as:

$$\hat{x} = \Delta \cdot x_{int} \approx x \quad (20)$$

where x_{int} and \hat{x} are integer tensors and quantized tensors, respectively, and Δ is scale factor.

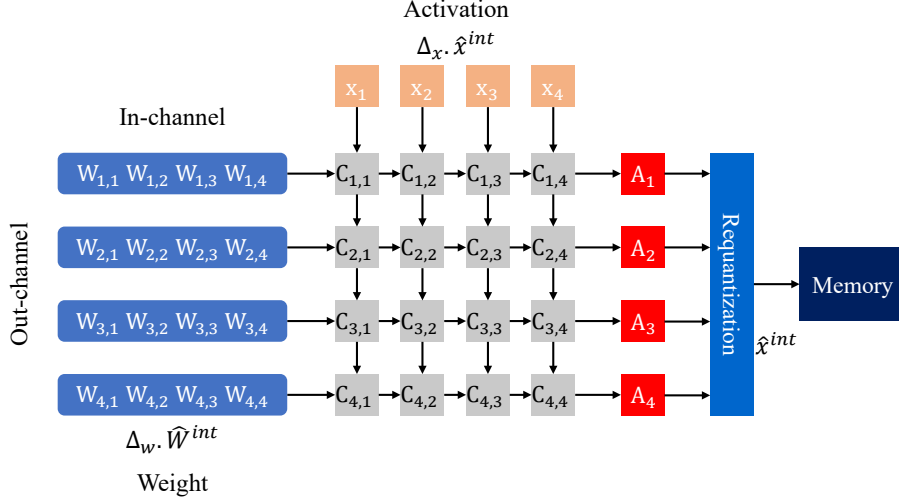


Figure 5: A schematic of matrix-multiply logic in accelerator for quantized inference.

Quantization settings have different granularity levels. Figure 5 shows the accelerator operation after the introduction of quantization. If we set both activations and weights to be layer-wise quantization, the new MAC operation can be represented as:

$$\begin{aligned} \hat{A}_n &= \sum_m \hat{W}_{n,m} \hat{x}_m + b_n \\ &= \sum_m (\Delta_w \hat{W}_{n,m}^{int}) (\Delta_x \hat{x}_m^{int}) + b_n \\ &= \Delta_w \Delta_x \sum_m \hat{W}_{n,m}^{int} \hat{x}_m^{int} + b_n \end{aligned} \quad (21)$$

where Δ_w and Δ_x are scale factors for weights and activations, respectively, $\hat{W}_{n,m}^{int}$ and \hat{x}_m^{int} are integer values. The bias is typically stored in higher bit-width (32-bits), so we ignore bias quantization for now. As can be seen, this scheme factors out the scale factors from the summation and performs MAC operations in fixed-point format, which accelerates the calculation process. The activations are quantized back to integer values \hat{x}_m^{int} through a requantization step, which reduces data transfer and simplifies the operations of the next layer.

To approximate the operations of quantization to full-precision, channel-wise quantization for weights is widely used, which sets quantization parameters to each out-channel. With this setting, the MAC operation in Eq. 21 can be represented as:

$$\begin{aligned} \hat{A}_n &= \sum_m (\Delta_{w_n} \hat{W}_{n,m}^{int}) (\Delta_x \hat{x}_m^{int}) + b_n \\ &= \Delta_{w_n} \Delta_x \sum_m \hat{W}_{n,m}^{int} \hat{x}_m^{int} + b_n \end{aligned} \quad (22)$$

where Δ_{w_n} is scale factor for the n_{th} out-channel of weights. However, the channel-wise quantization for activations sets quantization parameters to each in-channel. This setting is hardly supported

by hardware, as the MAC operation is performed as follows:

$$\begin{aligned}\hat{A}_n &= \sum_m (\Delta_w \hat{W}_{n,m}^{int}) (\Delta_{x_m} \hat{x}_m^{int}) + b_n \\ &= \Delta_w \sum_m \Delta_{x_m} \hat{W}_{n,m}^{int} \hat{x}_m^{int} + b_n\end{aligned}\quad (23)$$

where Δ_{x_m} is scale factor for the m_{th} in-channel of activations. Due to its inability to factor out the different scales from the accumulator summation, it is not hardware-friendly, leading to invalid acceleration.

I HUMAN PREFERENCE EVALUATION

In this section, we use an open-source *aesthetic predictor*⁹ to evaluate Aesthetic Score \uparrow , mimicking human preference assessment of the generated images. As reported in Table 10, DilateQuant has a better aesthetic representation compared to EfficientDM, which demonstrates that the quantized models with our method are more aesthetically pleasing to humans. For the large text-to-image model, we use the convincing DrawBench benchmark to evaluate human performance, as shown in Figure 6. Additionally, we visualize the random samples of quantization results in Figure 7 (LSUN-church), 8 (LSUN-Bedroom), and 9 (ImageNet). As can be seen, DilateQuant outperforms previous methods in terms of image quality, fidelity, and diversity.

Table 10: Aesthetic assessment of the different quantized models with 4-bit quantization.

Method	LSUN-Bedroom	ImageNet	DrawBench
FP	5.91	5.32	5.80
EfficientDM	5.47	3.51	2.84
DilateQuant	5.72	4.85	5.23

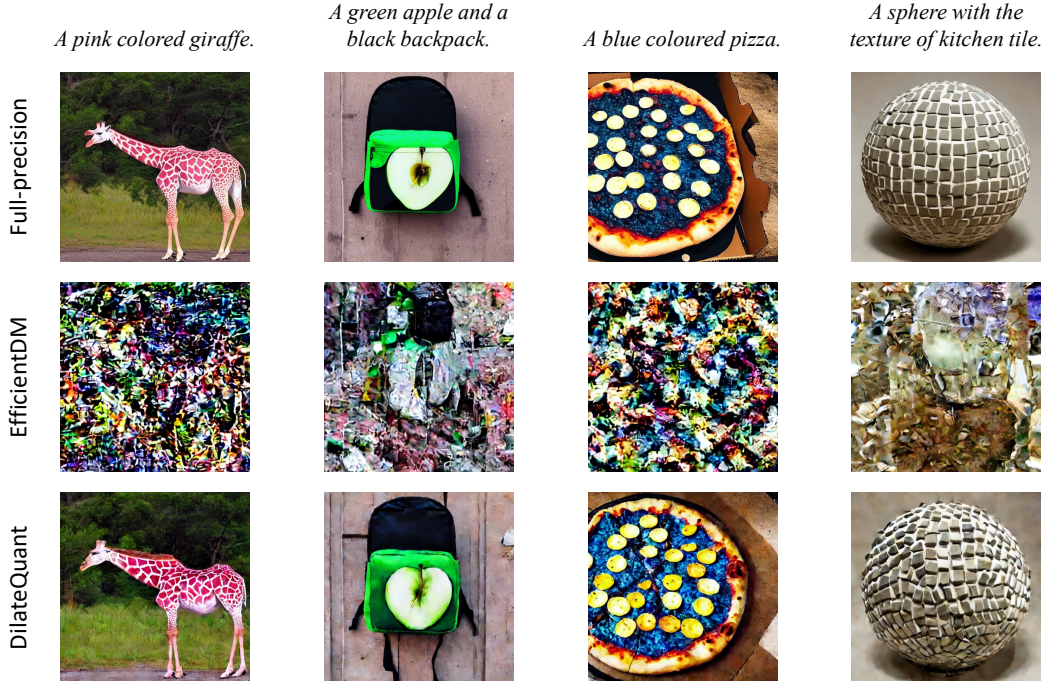


Figure 6: Random samples of different quantized models on DrawBench with 6-bit quantization.

⁹<https://github.com/shunk031/simple-aesthetics-predictor>



Figure 7: Random samples of quantized models with DilateQuant on LSUN-Church.



Figure 8: Random samples of different quantized models on LSUN-Bedroom with 4-bit quantization.



Figure 9: Random samples of different quantized models on ImageNet with 4-bit quantization.