2025 Design, Automation & Test in Europe Conference (DATE 2025)

BMP-SD: Marrying Binary and Mixed-Precision Quantization for Efficient Stable Diffusion Inference

Cheng Gu[1], Gang Li[2,*], Xiaolong Lin[1], Jiayao Ling[1], Jian Cheng[2,3], Xiaoyao Liang[1]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[2]The Key Laboratory of Cognition and Decision Intelligence for Complex Systems, CASIA, Beijing, China
[3]AiRiA, Nanjing, China
cheng_gu@sjtu.edu.cn, gang.li@ia.ac.cn

***Abstract*—Stable Diffusion (SD) is an emerging deep neural network (DNN) model that has demonstrated impressive capabilities in generative tasks such as text-to-image generation. However, the iterative denoising stage of the SD model is extremely expensive in both computations and memory accesses, making it challenging for fast and energy-efficient edge deployment. To alleviate the overhead of denoising, in this paper we propose BMP-SD, a post-training quantization framework for hardware-efficient SD inference. BMP-SD employs binary weight quantization to significantly reduce the computational complexity and memory footprint of iterative denoising, along with dynamic step-aware mixed-precision activation quantization, based on the observation that not all denoising steps are equally important for a specific input prompt. Experiments on the text-to-image generation task show that BMP-SD achieves mixed-precision (W1.73A4.87) with minimal accuracy loss on MS-COCO 2014 dataset. We also evaluate the BMP-SD quantized model on three state-of-the-art bit-flexible DNN accelerators, results reveal that our method can deliver up to $5.14\times$ performance and $3.85\times$ energy efficiency improvements compared to W8A8 quantization.**

***Index Terms*—Stable Diffusion, Binary Quantization, Mixed-Precision Quantization, Hardware-Efficient Inference**

I. Introduction

Recently, the surge in demand for generative artificial intelligence has been remarkable, fueled by its high quality and vast creative potential. Diffusion models, e.g. Stable Diffusion (SD) [1], have surpassed traditional Generative Adversarial Networks (GANs) [2] in various synthesis applications such as image generation [3], super-resolution [4], image editing [5], and inpainting [6]. The deployment of diffusion models on edge devices is crucial to improve terminal intelligence.

The Stable Diffusion model architecture primarily consists of three components: Text Encoder [7], UNet [8], and Variational Autoencoder (VAE) [9]. A typical text-to-image generation pipeline is shown in Fig. 1, the Text Encoder receives the input prompt to produce prompt embeddings, which are subsequently processed by the UNet in a iterative manner for denoising. The denoised features are then utilized to generate the image through the VAE Decoder. While SD models have remarkable generative abilities, they impose substantial memory and computational overhead. For example, the SD-v1.5 model consists of 1.205 billion parameters, and it costs 211.19 TFLOPs to generate a 512×512 image over 50 denoising steps. In particular, the denoising stage involving UNet accounts for 72% of the model parameters and 96% of the total FLOPs. Therefore, reducing the vast computational complexity and memory footprint of UNet-based denoising is the key for efficient SD inference, especially in edge applications.

*Corresponding author

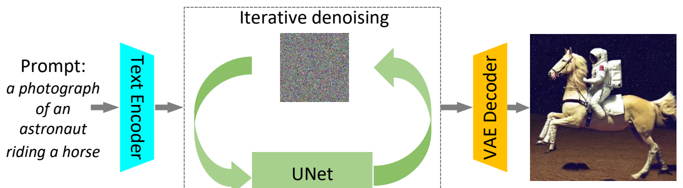


Fig. 1: Illustration of a typical SD text-to-image generation pipline. The denoising stage with the UNet works in an iterative manner (e.g., 50 steps).

To achieve efficient SD inference, quantization that converts floating-point representations to low-precision counterparts is promising. However, existing quantization approaches for SD models, such as Q-Diffusion [10], QuEST [11], and QNCD [12], apply a uniform bit-width across all denoising steps and layers, which fail to effectively identify and leverage the intrinsic characteristics of SD models for further compression.

In this paper, we propose BMP-SD, a post-training quantization framework for hardware-efficient inference of stable diffusion. To mitigate the overhead of vast computational complexity and memory traffic arising from the iterative denoising, BMP-SD proposes binary quantization for weights and mixed-precision quantization for activations, which is motivated by our following two observations: 1) The value range of UNet weights is quite narrow and follows a consistent distribution pattern, making it well-suited for extremely low-bit compression; 2) The importance of each denoising step varies for a given input prompt, suggesting that we can adaptively allocate high precision for activations in the important steps and low precision for the rest, thereby minimizing computations compared to uniform activation quantization. Moreover, BMP-SD also explores layer-wise mixed-precision activation quantization within each denoising step by assessing the importance of individual layers. The contributions of this paper are as follows:

- We propose a post-training binary quantization method for UNet weights, which first segments the weights into blocks, and then binarizes each block based on the weight distribution and significance. To minimize quantization errors, we propose updating the scaling factors at layer, block, and model levels during offline calibration.

- We propose a mixed-precision activation quantization method for UNet. It exploits a step evaluation module to dynamically assess the significance of each denoising step, which is utilized for step-aware bit-width allocation based on a given input prompt. To further reduce computations, it also quantizes the activations within each denoising step by measuring the layer-wise significance.
- We evaluate our proposed BMP-SD method on the text-to-image generation task. Results show that it can achieve a mixed precision of W1.73A4.87 with minimal accuracy loss on the MS-COCO 2014 [13] validation set. We also evaluate the quantized model obtained from BMP-SD on three state-of-the-art bit-flexible DNN accelerators, AdaS [14], Laconic [15], and Pragmatic [16]. Results show that our BMP-SD can achieve up to 5.14× speedup and 3.85× energy efficiency compared to existing INT4 and INT8 SD quantization.

II. Background and Related Work

A. Stable Diffusion

Diffusion models generate images via a Markov chain [17] process, iteratively applying reverse denoising to the initial data during inference to produce high-quality images [18]. In this process, the original data follows a standard Gaussian distribution $x_t \sim N(0, I)$, and the final reverse adheres to the following equation:

$$p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)=\mathcal{N}\left(\mathbf{x}_{t-1} ; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right) \tag{1}$$

where t is timestep, μ_θ and Σ_θ are calculated from the output of the diffusion model.

Stable Diffusion consists of three principal modules: Variational Autoencoder (VAE) [9], Text Encoder [7], and UNet [8]. For text-to-image generation tasks, Stable Diffusion initially processes the text prompt through a Text Encoder to generate prompt embeddings. Then these embeddings are denoised for multiple steps (e.g., 50 steps) using an UNet model. During each denoising iteration, the UNet network is executed, and the output latent features from the last denoising step are fed into the VAE Decoder to produce the final image.

TABLE I: Breakdown of parameter size and computational complexity of Stable Diffusion-v1.5.

Module	Parameter (M)	Computation (TFLOPS)
VAE Decoder	49.49 (4.11%)	7.55 (3.57%)
Text Encoder	289.71 (24.04%)	0.14 (0.06%)
UNet	865.91 (71.85%)	204 (96.37%)

Fig. 2 shows that the UNet structure within Stable Diffusion is mainly composed of CrossAttnBlock and DownBlcok. The UNet first downsamples and then upsamples the input latent to achieve denoising. For example, the CrossAttnDownBlock includes ResnetBlock and TransformerBlock, representing convolutional and linear operations, respectively, as shown in Fig. 2. In the inference of SD, various nonlinear operations are also employed, including LayerNorm [19], GroupNorm [20], SoftMax [21], GeGLU [22], and SiLU [23]. These operations contribute to the overall effectiveness and performance of the model. Table I shows the breakdown of SD model in terms of parameter size and computational complexity. It is clear that the UNet bears the considerable burden of both aspects, thus warranting it as the focal point of our optimization approaches.

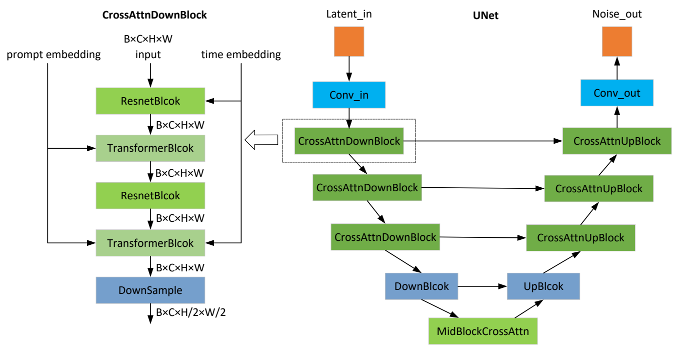


Fig. 2: The structure of UNet and CrossAttnDownBlock in the SD model.

B. Quantization

Given a floating point input x_{fp}, the uniform quantization converts it into a fixed point x_{int} as follow:

$$x_{\text{int}} = \text{Clip}(\lfloor \frac{x_{\text{fp}}}{\text{s}} \rceil + z, 0, 2^b - 1) \tag{2}$$

where s is the scaling factor, z is the zero point, and b is the quantization bit-width. $\lfloor \cdot \rceil$ denotes the nearest rounding operation. As a special case, binary quantization can reduce each high-precision value to only 1 bit using the following method:

$$W_{binary} = \alpha \cdot sign(W_{fp}) \tag{3}$$

$$sign(w) = \begin{cases} 1 & \text{if } w \geq 0 \\ -1 & \text{others} \end{cases} \tag{4}$$

where W_{fp} is the floating-point weight, W_{binary} is the binary weight, α is the scaling factor.

Previous post-training quantization methods for SD models, such as Q-Diffusion [10], focus on timestep-aware calibration and the design of split-shortcut quantization techniques. QuEST [11] adjusts the activation distribution and embeds significant temporal information. QNCD [12] applies embedding-based feature smoothing to reduce intra-quantization noise and uses runtime noise estimation to dynamically filter interquantification noise. While these quantization methods achieve satisfactory accuracy, they uniformly apply the same bit-width across the entire model and fail to identify and leverage the intrinsic characteristics of SD model effectively.

III. BMP-SD Quantization

In this section, we elaborate on the unique properties of the SD model and introduce our BMP-SD quantization framework accordingly. We will discuss the quantization of weights and activations separately.

A. *Binary Weight Quantization*

Fig. 3a shows the weight distribution between input blocks of the UNet. We notice that the weight distribution in the SD model exhibits a narrow range with a consistent pattern. Each weight exhibits a similar distribution, with values ranging within $[-2.5, 2]$, and some outliers are present. Fig. 3b shows the distribution of attention weights of a transformer block. We find that most UNet weights follow a similar normal distribution.

On the basis of the above observation, we apply a binary quantization method to the weights of the UNet. We first assess the importance of each weight. For every weight element w_i in each layer, we calculate its importance score s_i as follow:

$$s_i = \frac{w_i^2}{[\mathbf{H}^{-1}]_{ii}^2} \tag{5}$$

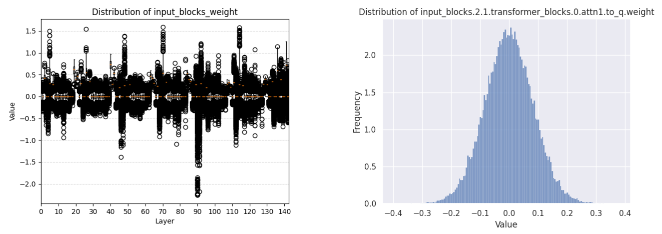
where $\mathbf{H}$ represents the Hessian matrix of each layer. Obviously, s_i comprehensively reflects the magnitude of weights and their impact on the output.

Taking into account the notable disparities in quantization sensitivity between weights, we divide the weights into a high score region and a normal region based on s_i to reduce quantization errors. Despite the relatively small number of elements in the high-score region, their impact on the final outcomes is considerable. Inspired by BiLLM [24], we apply binary quantization specifically to the high score region. For the normal region, we further divide the weights into four groups for binary quantization, $mask_1$ to $mask_4$, corresponding to the ranges $[-\infty, p_1]$, $[p_1, 0]$, $[0, p_2]$, and $[p_2, +\infty]$, respectively. The thresholds p_1 and p_2 are determined using the percentile search method [24].

The entire binary quantization process is performed offline. We construct a calibration dataset to reduce PTQ quantization error. We design a fine-grained multi-level quantization loss function, which supervises and updates the binary quantization parameters at the layer, block, and model levels. The quantization error function is as follows:

$$Loss_{total}(\alpha^*) = L_{out} + L_{blcok} + L_{layer} + L_{weight} \tag{6}$$

where α^* is the learnable scaling factor of binary. The loss functions are all MSE functions.

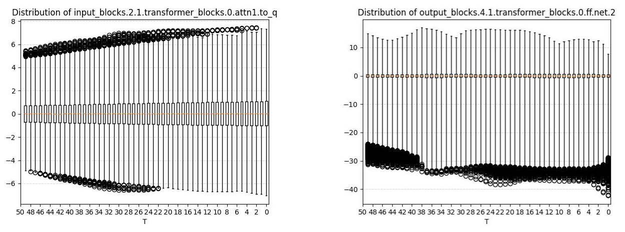


(a) input_blocks weight (b) layer weight

Fig. 3: Distribution of input_blocks' attention weights. (a) Weight distribution exhibits a narrow range with a consistent pattern. (b) Most weights follow a normal distribution.

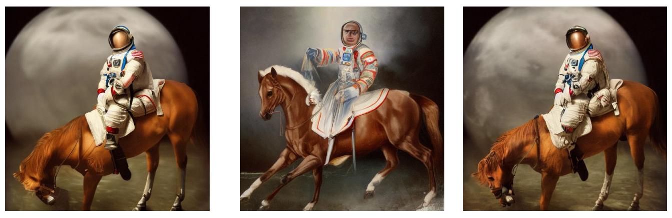
B. *Mixed-precision Activation Quantization*

Unlike traditional neural networks [25], SD model requires iterative denoising to progressively refine the output, eventually generating an image. Existing quantization methods for SD uniformly assign the same bit-width to activations at each step. However, as shown in Fig. 4, the activation distributions of SD layers exhibit dynamic shifts across steps, with notable outliers consistently present in each layer. This suggests that it is preferable to allocate different quantization bit-widths for different steps to achieve the optimal compression ratio. Moreover, we observe that skipping certain steps has minimal impact on the quality of the generated image, while omitting others can lead to significant loss, as shown in Fig. 5. This suggests that not all denoising steps are equally significant during image generation: some are less impactful, while others are crucial for maintaining output quality. Therefore, step-aware mixed-precision quantization is particularly suitable in this scenario. Based on the above observations, we exploit step-aware mixed-precision quantization for UNet activations.



(a) Downsample layer activation (b) Upsample layer activation

Fig. 4: Distribution of layer's activation. The activation distributions exhibit dynamic shifts across steps, with notable outliers consistently present in each layer.

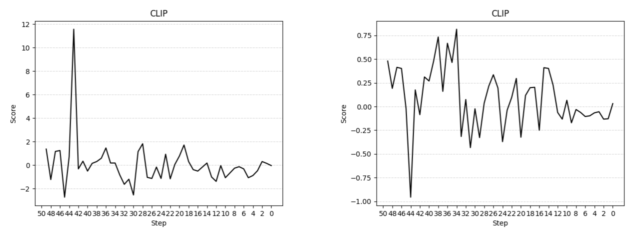


(a) Orign (b) Skip step: 40 (c) Skip step: 14

Fig. 5: Comparison of generated images with and without step skipping. (a) Orign generated image without skipping step. (b) Skipping step 40 severely degrades image semantics and quality. (c) Skipping step 14 makes minimal degeneration.

To perform step-aware mixed-precision quantization, we first analyze the importance of each denoising step offline through the calibration dataset. The CLIP score and the SMD2 score are used to evaluate the semantic and quality scores of the generated images. The CLIP score [7] is calculated based on the ViT-B/16 backbone. Eq.7 represents the calculation formula for SMD2 score, where X is the grayscale image of size $[M, N]$. For a certain step, we directly skip it and proceed to generate the final output. We calculate the differences of CLIP score and the SMD2 score for images generated with and without skipping. A higher difference indicate a greater impact of the step on the semantic or quality of the generated images, highlighting the importance of the step. As shown in Fig. 6 and Fig. 7, the importance of semantics and quality varies with different prompt inputs at each step. Some steps significantly contribute to the final semantic and quality effects, while a few steps have minimal impact on the outcome and can be skipped. Thus, we design a Step Importance Evaluation Module (SIEM) to predict the CLIP score and the SMD2 score for each step, depending on the given input prompt. As shown in Fig. 8, we extract intermediate features from the initial stages of the input block. These intermediate features integrate both the time embedding and the prompt embedding of the current step, which are then fed into the SIEM to predict the semantic and quality levels of the step. The SIEM is composed of a three-layer MLP, which adds negligible overhead to the overall UNet network. We profile each prompt's CLIP and SMD2 scores using the calibration dataset, classing them into five levels, and then train the SIEM offline to perform multi-class classification. When the SD step begins, we feed the intermediate features of input block into a trained SIEM to predict the semantic and quality levels of the step, guiding the quantization strategy for the rest of UNet.

$$\mathrm{SMD2(X)} = \sum_{\mathrm{j}}^{\mathrm{M}} \sum_{\mathrm{i}}^{\mathrm{N}} |\mathrm{X(i,j)} - \mathrm{X(i+1,j)}| \times |\mathrm{X(i,j)} - \mathrm{X(i,j+1)}| \quad (7)$$

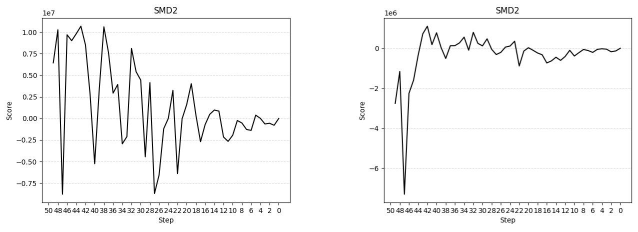


(a) prompt: *a photograph of an astronaut riding a horse.* (b) prompt: *a man falls off his skateboard in a skate park.*

Fig. 6: Distribution of CLIP score. The CLIP score varies with different prompt inputs at each step.

Similarly to MixDQ [26] and BitsFusion [27], we discover that in the SD model, convolutional layers and linear layers serve different purposes: convolutional layers primarily affect image generation quality, while linear layers mainly influence semantic effect. Therefore, after obtaining the semantic and quality levels for the current step, we assign different bit-widths to the convolutional and linear layers of the step. The semantic and image quality levels from 1 to 4 correspond to 2/4/6/8 bits for the linear and convolutional layers. If both the semantic and quality levels are 0, the current step is skipped directly.

After determining the bit-widths for the linear and convolutional layers at each step, we adjust these bit-widths further. The importance of the current layer is calculated using Eq.5. We



(a) prompt: *a photograph of an astronaut riding a horse.* (b) prompt: *a man falls off his skateboard in a skate park.*

Fig. 7: Distribution of SMD2 Score. The SMD2 Score varies with different prompt inputs at each step.

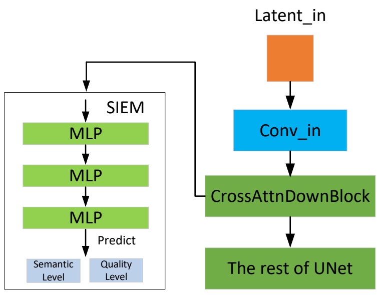


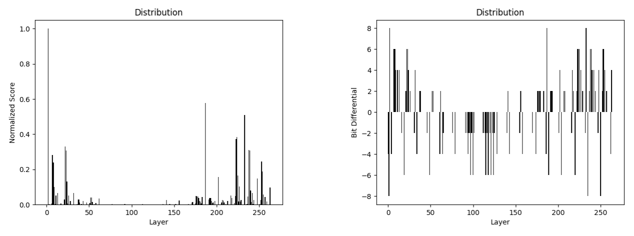
Fig. 8: The structure of SEIM. We extract intermediate features from the initial stages of the input block and feed them into the SIEM to predict the semantic and quality levels of the step.

compute the score by averaging s_i for each element within that layer based on Eq.8, where n denotes the number of elements in the current layer's weights. Fig. 9a shows the distribution of scores for all layers of UNet, and some layers have significantly higher importance, while others are minimally important. Layers with greater importance should have a larger bit-width, while layers with less importance should have a shorter bit-width. Thus, the bit-widths can be adjusted up or down relative to the initially determined bit-widths for the current step. We assign the bit-width of each layer's activation for every step based on Eq.9, where Bit_{step} denotes the bit-width of the current step, and $Norm$ means normalization. Fig. 9b demonstrates the fluctuation of bit-widths for the activation of each layer. It is evident that with our allocation method, layers with higher importance scores are assigned higher bit-widths for their activations, and vice versa. After balancing these fluctuations, the average bit-width equals the base bit-width for the step.

$$Score_{layer} = \frac{1}{n} \sum_{i}^{n} s_i \quad (8)$$

$$Bit_{layer} = Clip(Bit_{step} + 8 \times Norm(Score_{layer}), 2, 8) \quad (9)$$

Before quantization, we first address outliers in the activations. As shown in Fig. 4, there are many outliers in the activations, and in our experiments, we find that these outliers have a significant impact on quantization errors. Since the proportion of outliers is relatively small for the convolutional layers, we quantize these outliers to 8 bits. For linear layers,



(a) Importance score of layers. (b) Fluctuation bit-width of layers.

Fig. 9: Distribution of importance score and allocated bit-width of layers. (a) The importance of each layer varies within the model. (b) Bit-width fluctuations are uniform and aligned with layer significance, with the average bit-width matching the base of each step.

we follow OmniQuant [28] to narrow the outlier distribution range as follow:

$$\mathbf{Y} = \mathbf{X}\mathbf{W} + \mathbf{B} = [(\mathbf{X} - \delta) \oslash \gamma] \cdot [\gamma \odot \mathbf{W}] + [\mathbf{B} + \delta\mathbf{W}] \quad (10)$$

where Y denotes the output, $\gamma \in \mathbb{R}^{1\times C_{in}}$ and $\delta \in \mathbb{R}^{1\times C_{in}}$ are channel-wise scaling and shifting parameters, respectively, $\oslash$ and $\odot$ are element-wise division and multiplication.

After handling the outliers, we perform quantization based on the assigned bit-width. Since the activation range varies for different prompts, we use different quantizers for each bit-width to match tensors with similar distribution trends. We calculate the distance between the maximum and minimum values of the tensor and the quantizer group corresponding to the given bit-width as follow:

$$Dist(X, Q_i) = (min(X) - u_i)^2 + (max(X) - l_i)^2 \quad (11)$$

where X denotes the input tensor, u_i and l_i denotes the maximum and minimum values of i-th quantizer, respectively. We then select the quantizer with the smallest distance to the tensor for quantization.

Similarly to weight quantization, we supervise quantization parameters at the granularity of layers, blocks, steps, and outputs during calibration. We incorporate the quantizer parameters s, z, and outlier parameters γ, δ into the loss function simultaneously, thereby achieving optimal quantization performance, in the following equation:

$$Loss_{total}(s^*, z^*, \gamma^*, \delta^*) = L_{layer} + L_{blcok} + L_{step} + L_{out} \quad (12)$$

where $s^*, z^*, \gamma^*, \delta^*$ is learnable quantizer parameters and outlier parameters, respectively. The loss functions are all MSE functions. The detailed implementation of BMP-SD is shown in Algorithm 1.

IV. Evaluation

A. Methodology

In this section, we evaluate the effectiveness of our BMP-SD algorithm and its hardware efficiency. We perform BMP-SD on Stable Diffusion-v1.5, and utilize the DDIM sampler to sample 50 time steps. We generate a total of 50,000 images of size 512×512 on the MS-COCO 2014 [13] validation set. We employ CLIP score and FID to assess the impact on precision before and after quantization. We compare our method with state-of-the-art SD PTQ algorithms Q-Diffusion [10], QuEST [11], and QNCD [12].

For hardware efficiency, we deploy the BMP-SD quantized model on state-of-the-art bit-flexible DNN accelerators, including AdaS [14], Pragmatic [16], and Laconic [15], and compare performance and energy efficiency against existing INT4 and INT8 quantization. We develop cycle-accurate simulators for each accelerator to model compute cycles and memory accesses. We implement all three accelerators in Verilog and synthesize them using the Synopsis Design Compiler with a TSMC 28nm HPC+ standard cell library at 500MHz to obtain the power numbers.

Algorithm 1 BMP-SD Pipeline

```
Input: $\mathbf{W}$ - weight matrix
       $\mathbf{X}$ - calibration activations
       $\mathbf{P}$ - calibration prompt
       $M$ - the number of layers of the weights
       $N$ - the number of blocks of the layer
Output: $\hat{\mathbf{W}}$ - binarized weights
        $\hat{\mathbf{X}}$ - quantized activations
  Step 1: Weight Quantization
 1: Calculate the weight score $score_w$ based on Eq. 5
 2: for $layer = 1, 2, 3, ..., M$ do
 3:     for $b = 0, \beta, 2\beta, ..., N$ do
 4:         $\mathbf{W}^b := \mathbf{W}_{:,b:b+\beta}$
 5:         $mask_0 = TopK(score_w)$
 6:         $p_1, p_2 = seg\_seach(\mathbf{W}^b_{\notin mask_0})$
 7:         Calculate $mask_{1-4}$ based on Section III-A
 8:         $\hat{\mathbf{W}}^b = binary(\mathbf{W}^b[mask_{0-4}])$
 9:         $\hat{\mathbf{W}} = cat(\hat{\mathbf{W}}^b)$
10:     end for
11:     block_layer_backward()
12: end for
13: out_backward()
  Step 2: Activation Quantization
14: Training SEIM offline using calibration data
15: for $step = 50, 49, 48, ..., 1$ do
16:     $level = SEIM(step, \mathbf{P})$
17:     for $layer = 1, 2, 3, ..., M$ do
18:         Calculate the $Bit_{layer}$ based on Eq. 9
19:         Remove outliers based on Eq. 10
20:         Allocate quantizer to activation based on Eq. 11
21:         $\hat{\mathbf{X}} = quantize(\mathbf{X})$
22:         block_layer_backward()
23:     end for
24:     step_backward()
25: end for
26: out_backward()
27: return $\hat{\mathbf{W}}$, $\hat{\mathbf{X}}$
```

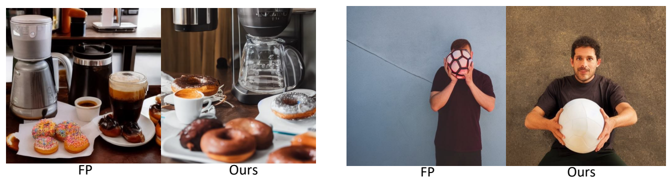
B. Algorithm

We use binary quantization for 80.61% weights in the SD UNet, while 19.39% weights with larger quantization errors are quantized to INT8 or INT4, resulting an avereage of 1.73 bits. Table II illustrates the impact of different strategies of BMP-SD on the accuracy. Compared to the FP model, our binary quantization method reduces the weight bit-width to 1.73 bits, with a CLIP score decrease of 2.21. We further apply mixed-precision quantization to compress the activation bit-width to 4.89 while retaining outliers as FP values, resulting in only a 0.31 decrease in CLIP score. This demonstrates that our mixed-precision approach effectively allocates appropriate bit-widths to both steps and layers, achieving minimal impact on accuracy. We further address outliers as described in Section III-B, leading to a CLIP score decrease to 28.32, which remains within a reasonable range.

TABLE II: The ablation experiments of BMP-SD.

Method (W/A)	FP	1.73/FP	1.73/4.89-outlier	1.73/4.89
CLIP Score	31.48	29.27	28.96	28.32

Fig. 10 shows that the keywords from the prompts are accurately reflected in the generated images, and the image quality remains sharp after quantization. Compared to the FP model, our quantization framework maintains semantic information and image quality without significant degradation.



(a) prompt: *Donuts with frosting and glazed toppings sit on table next to coffee maker.* (b) prompt: *Man holding a ball he is about to throw.*

Fig. 10: Visualization of BMP-SD generated images.

Table III illustrates the comparison of our quantization method with the existing SD PTQ algorithms (Q-Diffusion [10], QuEST [11], and QNCD [12]) in terms of model size and accuracy. Compared to INT8 and INT4 quantization, which compress model size by 3.46× and 6.08× respectively, BMP-SD can achieve a compression ratio of 12.38×. This is because we utilize binary quantization to achieve extreme compression of the majority of the model weights. Table III also shows that accuracy loss from W8A8 to W4A8 is minimal, while a significant drop occurs when quantizing to W4A4. This suggests activations are highly sensitive to quantization. Compared to the three quantization algorithms, BMP-SD achieves a CLIP score of 28.32 and FID of 38.90, which is close to the accuracy of W4A4 but with halved model size.

TABLE III: Comparison of BMP-SD with state-of-the-art PTQ methods for SD.

Bitwidth (W/A)	Size (MB)	Ratio	Method	CLIP Score↑	FID↓
32/32	3279.12	1	FP	31.48	23.82
8/8	949.08	3.46×	Q-Diffusion [10]	31.41	27.84
			QuEST [11]	31.47	27.26
			QNCD [12]	31.36	27.33
4/8	539.14	6.08×	Q-Diffusion [10]	31.37	31.01
			QuEST [11]	31.5	30.47
			QNCD [12]	29.89	26.05
4/4	539.14	6.08×	Q-Diffusion [10]	N/A	N/A
			QuEST [11]	28.85	38.49
			QNCD [12]	28.35	38.93
1.73/4.89	264.85	12.38×	BMP-SD (Ours)	28.32	38.90

C. *Evaluations on DNN Accelerators*

We deploy the BMP-SD quantized model on three bit-flexible DNN accelerator AdaS [14], Pragmatic [16], and Laconic [15] to demonstrate the effectiveness on hardware acceleration. Table IV shows that compared to INT4 and INT8 quantization, our approach can achieve a maximum of 2.78× and 5.14× speedup, respectively. This is because our quantization framework reduces the bit-width and increases the sparsity of weights and activations. Consequently, bit-flexible DNN accelerators save a considerable number of computational cycles, thereby enhancing overall performance.

Table V also shows the energy efficiency comparison of BMP-SD with INT4 and INT8 quantization on three accelerators. Compared to INT4, BMP-SD achieves up to a maximum 1.48× improvement in energy efficiency. In comparison to INT8, our algorithm achieves significant energy savings in hardware deployment, with improvements of up to 3.85×, 3.02×, and 2.99×, respectively. Our quantization method reduces memory access and computational resource requirements, resulting in lower overall energy consumption compared to both INT8 and INT4.

TABLE IV: Performance comparison of BMP-SD with INT4 and INT8 quantization on state-of-the-art DNN accelerators AdaS [14], Laconic [15], and Pragmatic [16].

Bit-width (W/A)	Adas [14]	Laconic [15]	Pragmatic [16]
8/8	1×	1×	1×
4/4	1.85×	2.43×	1.63×
1.73/4.89	5.14×	4.95×	1.74×

TABLE V: Energy efficiency comparison of BMP-SD with INT4 and INT8 quantization on state-of-the-art DNN accelerators AdaS [14], Laconic [15], and Pragmatic [16].

Bit-width (W/A)	Adas [14]	Laconic [15]	Pragmatic [16]
8/8	1×	1×	1×
4/4	2.63×	2.04×	2.03×
1.73/4.89	3.85×	3.02×	2.99×

V. Conclusion

In this work, we propose BMP-SD, a post-training quantization framework designed for hardware-efficient stable diffusion inference. To mitigate the memory and computational overhead of the iterative denoising process, BMP-SD jointly employs binary and mixed-precision quantization for UNet weights and activations, respectively, achieving significant compression without compromising accuracy. Evaluations on three bit-flexible DNN accelerators demonstrate substantial performance and energy efficiency improvements of BMP-SD compared to state-of-the-art INT4/INT8 quantization methods.

Acknowledgment

This work was supported in part by the National Key R&D Program of China (No. 2022ZD0160304), the Beijing Municipal Science and Technology Project (No. Z231100010323002), and the Key Research and Development Program of Jiangsu Province (No. BE2023016).

References

[1] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[3] L. K. Saul, Y. Weiss, and L. Bottou, *Advances in neural information processing systems 17: proceedings of the 2004 conference*. MIT Press, 2005, vol. 17.

[4] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, "Image super-resolution via iterative refinement," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 4, pp. 4713–4726, 2022.

[5] B. Kawar, S. Zada, O. Lang, O. Tov, H. Chang, T. Dekel, I. Mosseri, and M. Irani, "Imagic: Text-based real image editing with diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6007–6017.

[6] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool, "Repaint: Inpainting using denoising diffusion probabilistic models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11 461–11 471.

[7] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[8] O. Ronneberger, P. Fischer, and T. 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*. Springer, 2015, pp. 234–241.

[9] D. P. Kingma, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[10] X. Li, Y. Liu, L. Lian, H. Yang, Z. Dong, D. Kang, S. Zhang, and K. Keutzer, "Q-diffusion: Quantizing diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 535–17 545.

[11] H. Wang, Y. Shang, Z. Yuan, J. Wu, and Y. Yan, "Quest: Low-bit diffusion model quantization via efficient selective finetuning," *arXiv preprint arXiv:2402.03666*, 2024.

[12] H. Chu, W. Wu, C. Zang, and K. Yuan, "Qncd: Quantization noise correction for diffusion models," *arXiv preprint arXiv:2403.19140*, 2024.

[13] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. 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*. Springer, 2014, pp. 740–755.

[14] X. Lin, G. Li, Z. Liu, Y. Liu, F. Zhang, Z. Song, N. Jing, and X. Liang, "Adas: A fast and energy-efficient cnn accelerator exploiting bit-sparsity," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[15] S. Sharify, A. D. Lascorz, M. Mahmoud, M. Nikolic, K. Siu, D. M. Stuart, Z. Poulos, and A. Moshovos, "Laconic deep learning inference acceleration," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 304–317.

[16] J. Albericio, A. Delmás, P. Judd, S. Sharify, G. O'Leary, R. Genov, and A. Moshovos, "Bit-pragmatic deep neural network computing," in *Proceedings of the 50th annual IEEE/ACM international symposium on microarchitecture*, 2017, pp. 382–394.

[17] J. R. Norris, *Markov chains*. Cambridge university press, 1998, no. 2.

[18] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10 850–10 869, 2023.

[19] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *ArXiv e-prints*, pp. arXiv–1607, 2016.

[20] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[21] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[22] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[23] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural networks*, vol. 107, pp. 3–11, 2018.

[24] W. Huang, Y. Liu, H. Qin, Y. Li, S. Zhang, X. Liu, M. Magno, and X. Qi, "Billm: Pushing the limit of post-training quantization for llms," *arXiv preprint arXiv:2402.04291*, 2024.

[25] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.

[26] T. Zhao, X. Ning, T. Fang, E. Liu, G. Huang, Z. Lin, S. Yan, G. Dai, and Y. Wang, "Mixdq: Memory-efficient few-step text-to-image diffusion models with metric-decoupled mixed precision quantization," *arXiv preprint arXiv:2405.17873*, 2024.

[27] 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 preprint arXiv:2406.04333*, 2024.

[28] W. Shao, M. Chen, Z. Zhang, P. Xu, L. Zhao, Z. Li, K. Zhang, P. Gao, Y. Qiao, and P. Luo, "Omniquant: Omnidirectionally calibrated quantization for large language models," *arXiv preprint arXiv:2308.13137*, 2023.