A. Quantization Recovery Comparison

In quantized linear layers, the forward pass is typically composed of three steps: quantizing the input and weight tensors, performing matrix multiplication in the quantized domain, and then recovering the output via dequantization. Formally, for input $\mathbf{X}$ and weight $\mathbf{W}$, where $\mathbf{X} \in \mathbb{R}^{m\times k}$ and $\mathbf{W} \in \mathbb{R}^{k\times n}$:

$$\mathbf{Y} \approx \mathrm{DeQuant}(\mathrm{Quant}(\mathbf{X}) \cdot \mathrm{Quant}(W)),$$

where $\mathrm{Quant}(\cdot)$ and $\mathrm{DeQuant}(\cdot)$ denote quantization and dequantization, respectively.

This section compares three representative quantization strategies from the perspective of how easily the original scale can be recovered after matrix multiplication:

- **Symmetric quantization**: both inputs and weights are quantized using zero-centered uniform scales without offsets. Specifically, for input $\mathbf{X}$ and weight $\mathbf{W}$, the quantization is defined as:

  $$\hat{\mathbf{X}} = \mathrm{round}\left(\frac{\mathbf{X}}{s_X}\right), \quad \hat{\mathbf{W}} = \mathrm{round}\left(\frac{\mathbf{W}}{s_w}\right),$$

  where s_X and s_W are the quantization scales for input and weight, respectively. The low-bits matrix multiplication yields:

  $$\hat{\mathbf{Y}} = \hat{\mathbf{X}}\hat{\mathbf{W}},$$

  and the recovered output is simply:

  $$\mathbf{Y} \approx s_X s_W \cdot \hat{\mathbf{Y}},$$

  which is efficient and scale-preserving due to the absence of zero-points.
- **Asymmetric quantization**: unlike the symmetric case, asymmetric quantization introduces non-zero offsets (zero-points), which shifts the quantized representation. For input $\mathbf{X}$ and weight $\mathbf{W}$, the quantization process is:

  $$\hat{\mathbf{X}} = \mathrm{round}\left(\frac{\mathbf{X}}{s_X}\right) + z_X, \quad \hat{\mathbf{W}} = \mathrm{round}\left(\frac{\mathbf{W}}{s_W}\right) + z_W,$$

  where s_X, s_W are the quantization scales, and z_X, z_W are the zero-points for input and weight, respectively.
  The low-bits matrix multiplication gives:

  $$\hat{\mathbf{Y}} = \hat{\mathbf{X}}\hat{\mathbf{W}}.$$

  To recover the output in full precision, we must subtract the effects of the zero-points:

  $$\begin{aligned}
  \mathbf{Y} &= \mathbf{X}\mathbf{W} \\
  &\approx s_X\left(\hat{\mathbf{X}} - z_X\right) \cdot s_W\left(\hat{\mathbf{W}} - z_W\right) \\
  &= s_X s_W\left(\hat{\mathbf{X}} \cdot \hat{\mathbf{W}} - z_X \mathbf{J}_{m\times k}\hat{\mathbf{W}} - z_W \hat{\mathbf{X}}\mathbf{J}_{k\times n} + k z_X z_W \mathbf{J}_{m\times n}\right) \\
  &= s_X s_W\left(\hat{\mathbf{X}} \cdot \hat{\mathbf{W}} - z_X \cdot \mathbf{J}_{m\times 1} \cdot \mathrm{rowsum}(\hat{\mathbf{W}}) - z_W\,\mathrm{rowsum}(\hat{\mathbf{X}}) \cdot \mathbf{J}_{1\times n} + k z_X z_W\right),
  \end{aligned}$$

  where $\mathbf{J}$ is a matrix of ones.
  As shown, asymmetric quantization introduces additional terms involving zero-points, which require extra additions and broadcasted summations during the recovery of the GEMM output. Although this scheme can improve accuracy when data distributions are significantly shifted, it increases computational complexity and reduces implementation efficiency.
- **Dual-scale quantization (ours)**: decomposes the input into positive and negative channels before quantization, preserving directional fidelity. This allows for scale-aligned matrix multiplication and avoids zero-point corrections. The final output is recovered by linearly combining the quantized results (see the dual-scale quantization formula).

Our dual-scale quantization method achieves a favorable trade-off between accuracy and computational efficiency. By decomposing the input into positive and negative channels with separate quantization scales, it preserves directional information that symmetric quantization often loses, thereby improving precision. Compared to asymmetric quantization, it avoids costly zero-point corrections and broadcasted summations during recovery. As a result, the output dequantization is simpler and more efficient, making our method well-suited for practical quantized neural network implementations.

B. More Experimental Details and Results

Image Quality Evaluation Metrics We employ several standard metrics to evaluate the quality and similarity of generated images against reference images.

- **PSNR (Peak Signal-to-Noise Ratio)**: Measures fidelity by quantifying the ratio between the maximum possible power of a signal and the power of corrupting noise. It is based on the **Mean Squared Error (MSE)**. A higher value is better. First, given a noise-free $m \times n$ image I and its noisy approximation K, MSE is defined as:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

  The PSNR (in dB) is then defined using MSE and the maximum pixel value MAX_I (e.g., 255 for 8-bit images):

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

- **SSIM (Structural Similarity Index Measure)**: Measures perceptual similarity based on luminance (μ), contrast (σ), and structure (σ_{xy}). A higher value (closer to 1) is better.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

  where μ_x, μ_y are the pixel means, σ_x, σ_y are the standard deviations, σ_{xy} is the covariance of x and y, and C_1, C_2 are stabilization constants.
- **FID (Fréchet Inception Distance)**: Measures the distributional similarity between real (r) and generated (g) images in the Inception-V3 feature space. A lower value is better.

$$FID = ||\mu_r - \mu_g||_2^2 + \text{Tr}(C_r + C_g - 2(C_r C_g)^{\frac{1}{2}})$$

  where μ_r, μ_g are the feature means, C_r, C_g are the covariance matrices, and Tr is the trace of the matrix.
- **LPIPS (Learned Perceptual Image Patch Similarity)**: A learned metric computing the distance between deep features (l) of two images (x, x_0). A lower value is better.

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} ||w_l \odot (\hat{y}_{hw}^l - \hat{y}_{0hw}^l)||_2^2$$

  where l is the layer, $\hat{y}^l, \hat{y}_{0hw}^l$ are the normalized feature activations from that layer, w_l are the learned channel weights, and H_l, W_l are the feature map dimensions.
- **C.SCR (CLIP Score)**: Measures semantic alignment between an image I and text T. A higher value is better.

$$\text{CLIPScore}(I, T) = 100 \cdot \cos(E_I, E_T)$$

  where E_I and E_T are the image and text feature embeddings from the CLIP model, and cos is the cosine similarity.
- **C.IQA & IR (CLIP-IQA & Image Reward)**: No-Reference (NR) metrics that predict human preference scores. They are learned models (Reward Models) and do not have a static formula. A higher value is better.

Extra Hyperparameter Settings and Implementation Details In all experiments involving diffusion models, we enable classifier-free guidance and set the guidance scale to 7. Regarding quantization granularity: all 8-bit quantization uses a per-tensor scheme. In 4-bit settings, weights are quantized per-channel, and activations are quantized dynamically per-token.

For SegQuant, the SmoothQuant algorithm is applied with the α parameter individually selected for each linear layer. We sweep α in the range from 0.0 to 1.0 with a step size of 0.1, choosing the value that minimizes the mean squared error (MSE) between the quantized and full-precision layer outputs. For 4-bit quantization, SegQuant uses SVDQuant as the optimizer instead of SmoothQuant. In the *DualScale* scheme, we focus on polarity-sensitive activation functions, specifically SiLU, GELU, and GEGLU, to ensure asymmetric activations are well preserved.

For SVDQuant, the low-rank setting is fixed at 64. For the FLUX model, singular value decomposition (SVD) is performed using float32 precision due to implementation constraints, while all other models use float64 precision for better numerical stability. For PTQD and TAC-Diffusion, calibration is applied only to the unconditional branch of the model, with 32 images used for sampling. For TAC-Diffusion, we use λ_1=0.8, λ_2=0.1, and a threshold of 4, following the original implementation. Additional implementation details and full configuration files can be found in our released codebase.

To facilitate future research and ensure full reproducibility of the results presented in this paper, we have released our complete implementation at: https://github.com/OptiSys-ZJU/segquant

Dual Scale Sensitivity Analysis To evaluate the sensitivity of our dual-scale quantization to outlier activations, we conducted an analysis comparing the strict Min/Max calibration strategy ($p = 1.0$) against percentile clipping ($p \leq 0.9999$). As detailed in Table 5, using the absolute maximum ($p = 1.0$) consistently yields the lowest relative F-norm error, avoiding the destructive effects of clipping. This finding implies that high-magnitude outlier activations in diffusion models carry critical semantic signals. Furthermore, our dual-scale approach significantly outperforms standard quantization at $p = 1.0$, demonstrating that splitting dimensions according to polarity (X_+, X_-) is structurally much more effective for heavy-tailed distributions than conventional clipping methods. These results remain stable across calibration set sizes of $N = 100$ and $N = 200$, confirming our method does not rely on massive calibration data.

Table 5. Dual Scale Sensitivity Analysis on SD3. We report the relative F-norm error across different calibration percentile settings and set sizes (N).

Method	**Percentile**	**Rel F-Norm ↓**	
		N=100	**N=200**
Standard	1.0 (Max)	0.0150	0.0132
Ours (DualScale)	**1.0 (Max)**	**0.0108**	**0.0102**
Ours (DualScale)	0.9999	0.0115	0.0146
Ours (DualScale)	0.9990	0.0412	0.0583

Pattern Detection Algorithm To ensure robustness across diverse model architectures and compiler transformations, SegQuant employs a topological pattern matching algorithm based on `torch.fx` symbolic tracing. Algorithm 1 details the procedure. The detector first performs symbolic tracing and shape propagation to obtain a computational graph with metadata. It then iterates through the graph to identify two categories of patterns:

- **Weight Segmentation (Forward-Tracing):** We identify `Linear` nodes followed immediately by `chunk` or `split` operations. The segment sizes are explicitly inferred from the operator arguments (e.g., `chunks` count or `split_size`).
- **Input Segmentation (Backward-Tracing):** We identify `Linear` nodes whose inputs originate from `concat`, `stack`, or `reshape` operations. For `Concat`/`Stack`, segment sizes are inferred by retrieving the shapes of constituent input tensors; for `Reshape` (common in multi-head merges), we verify if the flattened input dimension D_{in} decomposes into $N \times S$ to infer the segment size S.

Validity of Semantic Graph Alignment By semantic alignment, we mean quantizing according to the model's intrinsic semantic boundaries (e.g., time versus latent features). To isolate the gains brought by this semantic alignment, we ablated the quantization of the `DiT.0.norm1` layer in SD3. As shown in Table 6, our topology-aware segmentation (*SegLinear*) correctly infers $K = 6$ chunks and achieves the lowest F-norm error. In contrast, random chunking with the same number of chunks yields significantly higher error, confirming that aligning the quantization process with the model's topological semantics is vital rather than just performing fine-grained segmentation arbitrarily.

Quantization Runtime and Memory Consumption. To ensure reproducibility and transparency, we analyze the computational resources required by SegQuant. For the SD3 model, the complete quantization process takes approximately 2.5 hours on a single NVIDIA L20 GPU, utilizing 256 calibration images and 50 diffusion steps. Enabling hyperparameter search (e.g., with an α step size of 0.1) extends the runtime to $\sim$25 hours due to iterative calibration passes. The total duration is primarily determined by the calibration set size, diffusion steps, search granularity, and the specific optimizer used (e.g., SVDQuant incurs an additional $\sim$0.5 hours for singular value decomposition).

Algorithm 1 SegQuant Graph Pattern Detection

Require: Pre-trained model $\mathcal{M}$, Example inputs $\mathcal{X}$.
Ensure: Set of detected patterns $\mathcal{P} = \mathcal{P}_{\text{weight}} \cup \mathcal{P}_{\text{input}}$.

Initialization:
$\mathcal{G} \leftarrow$ torch.fx.symbolic_trace($\mathcal{M}$) ▷ Flatten module hierarchy
ShapeProp($\mathcal{G}, \mathcal{X}$) ▷ Propagate tensor metadata
$\mathcal{P} \leftarrow \emptyset$
Main Loop:
for each node $n \in \mathcal{G}$.nodes **do**
  Case 1: Weight Segmentation (Forward-Tracing)
  if n.op is `chunk` or `split` **then**
    $n_{\text{prev}} \leftarrow$ TraceForward(n.args[0]) ▷ Skip transparent ops like Dropout/To
    if n_{prev} is `nn.Linear` **then**
      $D_{\text{out}} \leftarrow n_{\text{prev}}$.out_features
      if n is `chunk` **then**
        $K \leftarrow n$.args[chunks]
        Infer segments: $S_i \approx D_{\text{out}}/K$ for $i \in \{1..K\}$
      else if n is `split` **then**
        Infer segments: $S_i \leftarrow n$.args[split_size]
      end if
      Add $(n_{\text{prev}}, \text{Type:Weight}, \{S_i\})$ to $\mathcal{P}$
    end if
  end if
  Case 2: Input Segmentation (Backward-Tracing)
  if n is `nn.Linear` **then**
    $n_{\text{in}} \leftarrow$ TraceBackward(n.args[0]) ▷ Skip transparent ops like Dropout/To
    $D_{\text{in}} \leftarrow n$.in_features
    if n_{in} is `concat` or `stack` **then**
      Retrieve input tensors $\{T_1, T_2, ...\}$ from n_{in}.args
      Infer segments: $S_j \leftarrow T_j$.meta['shape'][-1]
      Add $(n, \text{Type:Input}, \{S_j\})$ to $\mathcal{P}$
    else if n_{in} is `reshape` **then**
      Retrieve input shape $[..., N, S]$ from n_{in}.args[0].meta
      if $N \times S == D_{\text{in}}$ **then**
        Infer segments: N segments of size S ▷ Detects Reshape Merges
        Add $(n, \text{Type:Input}, \{S\} \times N)$ to $\mathcal{P}$
      end if
    end if
  end if
end for
return $\mathcal{P}$

Table 6. Semantics Ablation on SD3 `DiT.0.norm1`. Comparison of our topology-aware segmentation against random and mismatched chunking strategies.

Strategy	Chunks	F-Norm Error↓
SegLinear (Ours)	**6 (Inferred)**	**0.5415**
Random Chunk	6 (Same K)	0.7080
Mismatched	8 (Misalign)	0.5713

Regarding memory usage, the calibration process typically requires peak memory equivalent to $\sim 2\times$ the model size when

fully loaded (approx. 40 GB for SD3). To accommodate the larger FLUX model within a 48 GB budget (e.g., on NVIDIA L20), we employ a sequential swap-in/swap-out strategy that avoids quality degradation; without this, profiling would necessitate ∼80 GB (e.g., on NVIDIA H800). Crucially, as detailed in Table 7, SegQuant achieves efficiency by selectively targeting structure-sensitive layers—applying DualScale to only 12.0%–28.8% of the network—rather than processing the entire model uniformly. Consequently, the memory overhead for storing separate FP16 quantization scales is negligible; for instance, applying input segmentation to 48 attention layers in SD3 adds only ∼3.4 MB of parameter storage. These results confirm that SegQuant significantly enhances granularity with minimal cost, ensuring compatibility with standard deployment infrastructures.

Table 7. Statistics of Optimized Layers. We report the number of linear layers identified and processed by each component of SegQuant. The "Total Linears" column indicates the number of quantizable linear layers. The "Est. Overhead" column estimates the additional memory required to store the fine-grained quantization scales (calculated in FP16), which is negligible (less than 0.3% of model size).

Model	**Total Linears**	**DualScale**	**SegLinear (Weight)**	**SegLinear (Input)**	**Est. Overhead**
SD3.5-Medium	340	98 (28.8%)	49 (14.4%)	48 (14.1%)	∼3.4 MB
FLUX.1-dev	504	118 (23.4%)	77 (15.3%)	38 (7.5%)	∼5.4 MB
SDXL-base-1.0	743	89 (12.0%)	78 (10.5%)	156 (21.0%)	∼5.6 MB

Performance Optimization and Implementation Efficiency To further enhance runtime efficiency, we implement several system-level optimizations in SegQuant. Both *SegLinear* and *DualScale* are designed with high parallelism in mind. Their segmentation-aware structure allows the quantization, calibration, and inference processes to be fully **vectorized** using native PyTorch tensor operations, significantly accelerating layer-wise tracing and calibration.

For quantized inference, we develop a series of customized `Quant`/`DeQuant` and batched GEMM kernels with precision-aware execution paths, which effectively reduce latency under low-bit computation. Although *DualScale* requires two consecutive GEMM operations, these computations are fused and parallelized via `BatchedGEMM`/`ArrayGEMM` primitives. We implement these operations using the CUTLASS library to enable efficient dual-scale computation while maintaining compatibility with standard GPU architectures.

In our W8A8 online inference benchmark (see Table 8), we compare the proposed DualScale implementation against the PTQ4ViT[42] method applied to DiT's AdaNorm layer, which exhibits both *output-segmented* and *polarity-asymmetric* properties. The results demonstrate that our kernel achieves higher inference speed while maintaining equivalent numerical accuracy.

Looking ahead, we plan to introduce optimizer- and calibrator-specific kernel customization, along with sparsity-aware and memory-efficient variants of DualScale, to further mitigate memory bottlenecks and enhance deployment scalability.

Table 8. Runtime comparison of AdaNorm layer quantization and inference (W8A8) on an NVIDIA RTX 4090 GPU.

Model	**Hidden Dim**	**PTQ4ViT** (μs)	**DualScale** (μs)	**SegLinear + DualScale** (μs)
FLUX	3072	4201.05	**1074.93**	*1476.85*
SD3.5	1536	1962.20	**908.05**	*1360.18*

Results on Other Datasets and Visual Evidence Beyond the main results, we further evaluate SegQuant on additional datasets and provide corresponding visual analyses. Specifically, we report CLIP-based metrics—CLIP Score [8] and CLIP-IQA [35]—on MJHQ (Table 10), and compare SegQuant with other baselines on COCO (Table 9) and DCI (Table 11), using the `openai/clip-vit-large-patch14` model. Detailed visual comparisons, including both baseline and ablation studies, are shown in Tables 12, 13 and Figures 11, 12, 13, 14, and 15. SegQuant consistently preserves semantic structure and visual fidelity under aggressive quantization settings. In particular, SegQuant-G produces more coherent textures and stronger semantic alignment, demonstrating the advantage of incorporating semantic cues into the quantization process.

Table 9. COCO

Backbone	W/A	Method	Quality				Similarity		
			FID↓	IR↑	C.IQA↑	C.SCR↑	LPIPS↓	PSNR↑	SSIM↑
SD3.5-DiT	FP16	Baseline	34.70	1.027	0.476	16.86	N/A	N/A	N/A
	8/8(int)	Q-Diffusion	184.66	−2.197	0.395	16.02	0.704	8.21	0.388
		PTQD	33.78	0.822	0.434	**16.47**	0.454	10.79	0.523
		PTQ4DiT	31.79	0.912	0.458	16.37	0.403	12.69	0.593
		TAC	32.22	0.863	0.458	16.31	0.417	12.53	0.583
		SegQuant-A	32.45	**1.020**	**0.467**	16.35	**0.362**	**13.27**	**0.618**
		SegQuant-G	**32.02**	0.991	0.457	16.36	0.376	13.14	0.601
	4/8(int)	PTQ4DiT	69.89	0.085	0.411	16.51	0.570	10.83	0.504
		SVDQuant	**39.22**	**0.855**	**0.454**	16.40	**0.432**	12.32	**0.582**
		SegQuant-G	40.70	0.843	0.438	**16.45**	0.434	**12.40**	0.578
SDXL-UNet	FP16	Baseline	28.44	0.841	0.429	16.54	N/A	N/A	N/A
	8/8(int)	Q-Diffusion	68.06	−1.574	0.396	16.48	0.537	13.68	0.517
		PTQ4DiT	28.15	0.639	0.407	16.70	0.213	19.17	0.725
		SegQuant-A	28.37	0.645	0.408	16.71	0.145	21.32	0.785
		SegQuant-G	**28.13**	**0.652**	**0.408**	**16.71**	**0.138**	**21.63**	**0.793**
	8/8(fp)	Q-Diffusion	28.24	0.843	0.428	16.46	0.104	23.32	0.832
		SegQuant-A	28.36	**0.844**	**0.428**	**16.46**	0.104	23.31	0.832
		SegQuant-G	**28.00**	0.839	0.427	16.46	**0.093**	**23.90**	**0.842**
FLUX-DiT	BF16	Baseline	36.25	0.914	0.445	16.89	N/A	N/A	N/A
	8/8(int)	Q-Diffusion	36.04	0.883	0.446	16.52	0.302	15.23	0.624
		PTQ4DiT	38.82	0.716	**0.460**	16.45	0.328	14.73	0.620
		SegQuant-A	36.48	**0.907**	0.445	16.52	0.155	19.40	0.770
		SegQuant-G	**36.02**	0.900	0.444	**16.54**	**0.143**	**19.94**	**0.784**
	4/8(int)	PTQ4DiT	41.16	0.096	0.417	16.65	0.561	11.63	0.537
		SVDQuant	36.98	0.880	0.436	**16.57**	0.242	16.85	0.693
		SegQuant-G	**36.88**	**0.882**	**0.439**	16.56	**0.232**	**17.03**	**0.700**

Table 10. MJHQ-30K

Backbone	W/A	Method	Quality		W/A	Method	Quality	
			C.IQA↑	C.SCR↑			C.IQA↑	C.SCR↑
SD3.5-DiT	FP16	Baseline	0.481	15.91	4/8(int)	PTQ4DiT	0.428	15.86
	8/8(int)	PTQD	0.441	15.91		SVDQuant	**0.452**	15.85
		PTQ4DiT	0.461	**15.91**				
		TAC	0.460	15.90				
		Smooth+	0.458	15.85				
		SegQuant-A	**0.468**	15.86		SegQuant-G	0.452	**15.89**
		SegQuant-G	0.466	15.88				
SDXL-UNet	FP16	Baseline	0.434	15.77	8/8(fp)	Q-Diffusion	0.417	15.71
	8/8(int)	PTQ4DiT	0.430	15.76		SegQuant-A	0.417	**15.71**
		Smooth+	0.430	15.76		SegQuant-G	**0.418**	15.71
		SegQuant-A	**0.433**	15.77				
		SegQuant-G	0.433	**15.78**				
FLUX-DiT	BF16	Baseline	0.440	16.00	4/8(int)	PTQ4DiT	0.416	15.98
	8/8(int)	Q-Diffusion	0.444	**15.98**		SVDQuant	0.435	15.99
		PTQ4DiT	**0.461**	15.88				
		Smooth+	0.431	15.86				
		SegQuant-A	0.440	15.94		SegQuant-G	**0.437**	**16.01**
		SegQuant-G	0.440	15.93				

Table 11. DCI

Backbone	W/A	Method	Quality				Similarity		
			FID↓	IR↑	C.IQA↑	C.SCR↑	LPIPS↓	PSNR↑	SSIM↑
SD3.5-DiT	FP16	Baseline	22.17	0.685	0.451	18.05	N/A	N/A	N/A
	8/8(int)	Q-Diffusion	143.36	–1.973	0.460	17.92	0.691	7.80	0.254
		PTQD	65.42	–0.454	0.416	18.08	0.582	10.00	0.296
		PTQ4DiT	27.01	0.485	0.430	18.08	0.445	12.33	0.492
		TAC	28.33	0.430	0.432	18.10	0.461	12.11	0.478
		SegQuant-A	**24.13**	0.639	0.440	**18.07**	**0.407**	**12.74**	**0.521**
		SegQuant-G	24.58	**0.639**	**0.447**	18.06	0.412	12.40	0.516
	4/8(int)	PTQ4DiT	61.68	–0.314	0.403	**18.21**	**0.566**	10.94	0.386
		SVDQuant	25.22	0.518	0.430	18.14	0.446	**12.34**	0.495
		SegQuant-G	**23.50**	**0.576**	**0.435**	18.14	0.438	12.25	**0.497**
SDXL-UNet	FP16	Baseline	24.27	0.495	0.417	17.94	N/A	N/A	N/A
	8/8(int)	Q-Diffusion	73.39	–1.466	0.402	17.65	0.552	13.64	0.448
		PTQ4DiT	24.34	0.472	0.415	17.94	0.197	19.63	0.686
		SegQuant-A	24.44	0.487	0.416	17.94	0.132	21.69	0.752
		SegQuant-G	**24.12**	**0.504**	**0.416**	**17.94**	**0.123**	**22.11**	**0.763**
	8/8(fp)	Q-Diffusion	26.98	0.474	**0.406**	17.89	0.102	23.28	0.780
		SegQuant-A	26.13	**0.475**	0.406	**17.89**	0.101	23.34	0.782
		SegQuant-G	**26.10**	0.474	0.405	17.89	**0.092**	**23.82**	**0.793**
FLUX-DiT	BF16	Baseline	22.10	0.618	0.451	18.03	N/A	N/A	N/A
	8/8(int)	Q-Diffusion	23.67	0.513	0.449	18.01	0.298	14.40	0.563
		PTQ4DiT	26.39	0.438	**0.472**	17.97	0.552	9.22	0.321
		SegQuant-A	22.28	**0.592**	0.452	18.01	0.149	18.29	0.715
		SegQuant-G	**22.19**	0.590	0.451	**18.02**	**0.131**	**18.97**	**0.737**
	4/8(int)	PTQ4DiT	72.86	–0.393	0.418	17.88	0.592	10.98	0.465
		SVDQuant	**22.57**	**0.575**	0.442	18.03	0.224	16.19	0.635
		SegQuant-G	23.31	0.566	**0.446**	**18.03**	**0.219**	**16.31**	**0.639**

Table 12. Additional results on quantization errors of single layers.

Layer Name	Method	F-norm	
		w/o SEG.	w/ SEG.
`DiT.6.norm1_context`	SMOOTHQUANT	1.7670	**1.0541**
`DiT.22.attn.add_out`		313.96	**283.04**
`DiT.6.norm1_context`	SVDQUANT	0.2454	**0.2265**
`DiT.22.attn.add_out`		326.17	**285.67**

Table 13. Example result of the computation-graph-based automatic semantic search for layer segmentation, using the torch FX graph of SD3.5 model. The listed layer names correspond to those defined in the Diffusers implementation.

Layer Name	Detection Result

`DiT.*.norm1`
`DiT.*.norm1_context`
`DiT.*.norm_out`
	OUTPUT-SEGMENTED

`DiT.*.attn.out` `DiT.*.add_out`	INPUT-SEGMENTED

`TimeEmbedder.linear_2`
`TextProjection.linear_2`
`DiT.*.norm1`
`DiT.*.norm1_context`
`DiT.*.attn.ff.2`
`DiT.*.attn.ff_context.2`
`DiT.*.norm_out`
	POLARITY-ASYMMETRIC

Figure 11. More visualization of main results on the MJHQ dataset with SD3.5 and W8A8 DiT quantization.

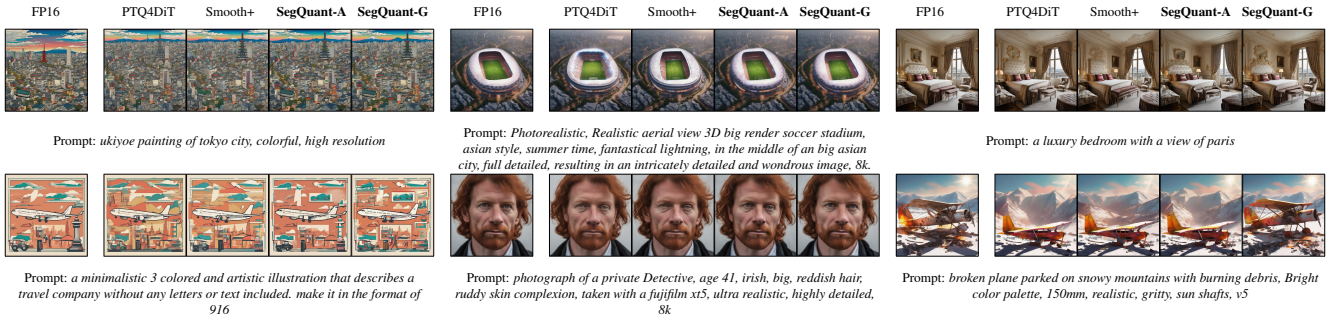
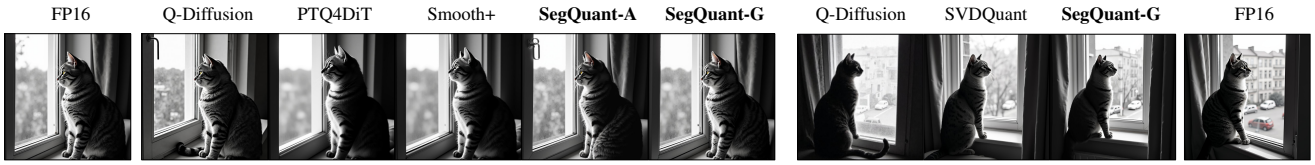


Figure 12. More visualization of main results on the MJHQ dataset with SDXL and W8A8 DiT quantization.



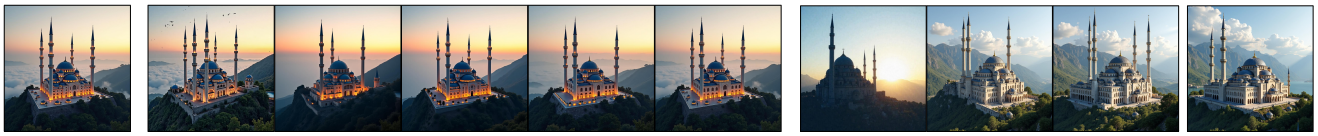
Prompt: *a cat, its face looks like Winston Churchill, standing up like a human, in a black and white photo, looking out of a window.*

Prompt: *an owl sits on a rock and looks outward, in the style of nikon d850, light beige and amber, exaggerated poses, explosive wildlife, dansaekhwa, multiple points of view.*

Prompt: *medium sized kitchen, bombay furniture, horizontal ombre yellow to green, French provincial style, braai in central hearth, refrigerator, large butcher block, golden hour.*

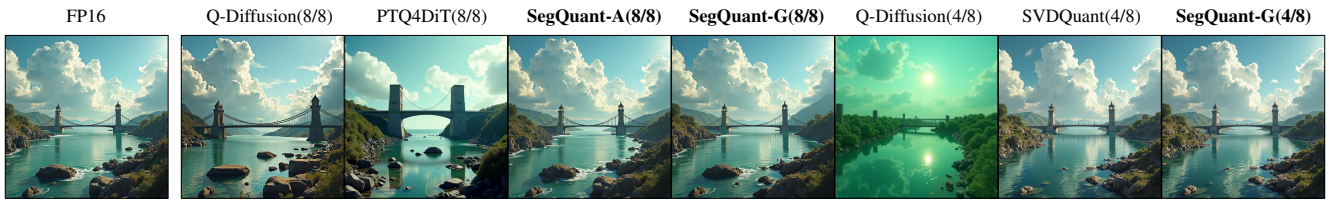
Prompt: *a lasagne outside an italian restaurant in the city bologna at midday with bright light. Ultra High Definition. High detailed. HD. Photorealistic.*

Prompt: *birdseye view of Mediterranean castle with Romanesque revival influences, photorealistic, 8k highest resolution, horizon line visible in the back, rocky cliff tropical and clear sky.*



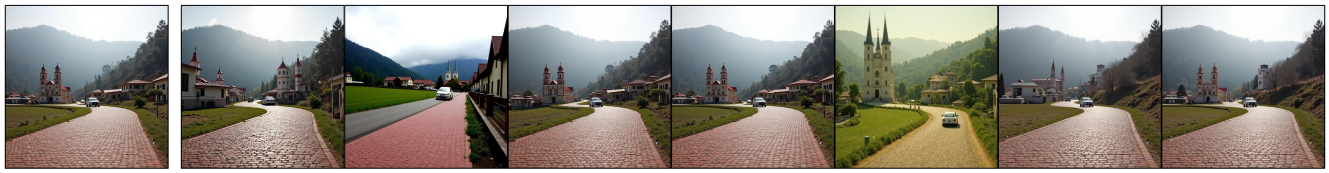
Prompt: *majestic islamic mosque in the mountain landscape beautiful.*

Figure 13. Additional visualizations of the main results on the MJHQ dataset, comparing FLUX and W8A8 DiT (left) with W4A8 (right) quantization.



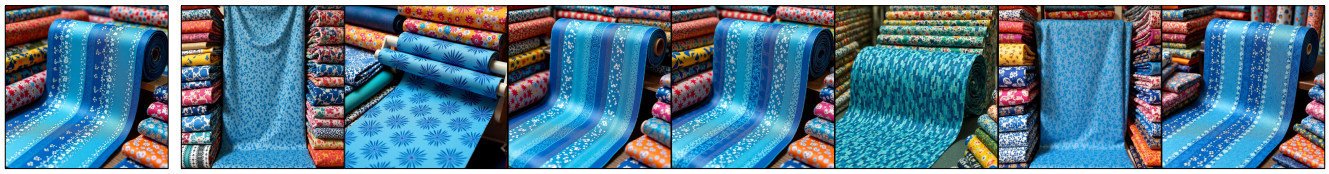
Prompt: *A body of water it is brown and green the sky is bright and there are many clouds in it A bright day the sky is very blue also there are clouds here that are very thick and very full also in the center of the sky is the biggest clouds. In the center of the image there is a large body of water that is green and blue. Going across the body of water is a large gray bridge with four towers on it two on each side and connecting each set of towers is a large beam that is a walkway. The walkway has windows that allow looking out over the bay. There is a large bed of rocks that is close to the camera of different colors dark gray brown white and light gray also there is also a yellow water bottle that has ben deposited amongst the rocks.*

Prompt: *This is a small room with blue walls inside of a church with a white alter with elaborate gold trim in the center. Two gold stands are in the room near the front of the alter. There is a bay window in the back. There is a plant in front of each window. The top of the alter is round with gold trim and has a gold cross on the top. A light is hanging on the top right side of the room.*



Prompt: *A brick road that starts in the foreground goes down a slope and into the background. There are buildings on the left and right sides of the road. A brick road that starts in the foreground extends back and to the left into the background. There is a white vehicle traveling along the road. Buildings can be seen lining the area to the right of the road. There is a field to the left of the road, and buildings to the left of it in the background. There is a religious building in the background that has two towers rising up from it. A wooded area can be seen on the right hand side of the background. A mountainside can be seen on the left hand side of the background. There are trees growing on the mountainside, and mist in the air to the right of the mountainside.*

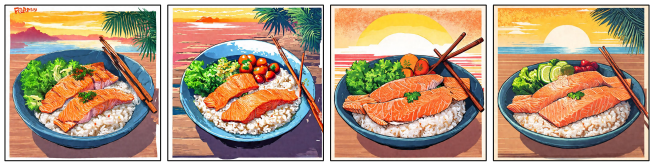
Prompt: *Tourist merchandise is hanging on a wall on display for people to buy. There are shirts on the left, magnets in the middle, and various objects on the right. Five rows of various shirts that are folded into squares and inside of clear plastic are on the left side of the image. A wall in the middle section is covered in rows of bright colored magnets for sale. There is a round yellow and black price tag in the magnet section. The right side of the image has shelves at the top with miniature figures of the clock tower and red phone booth. Four rows of keychains are hanging below the miniature figures. A shelf with stacks of toy taxis and busses are below the key chains. A book is on its side on a shelf below the toys along with round silver plates. A shelf filled with various sizes of snow globes is below the book. The bottom shelf on the right has decorative plates on stands. A line of mugs is between the magnets and the items on the right.*



Prompt: *This is a roll of a fabric that is multiple shades of blue and has a leaf pattern woven into it. It is surrounded by other bright and intricate patterned rolls of fabric. This is a roll of a light blue with dark blue pattern, woven fabric surrounded by rolls and stacks of folded brightly patterned fabrics.*

Prompt: *A rectangular sign that contains the logo for Aldi grocery stores can be seen on the side of a building. There are several trees rising up behind the building. A blue-colored wall of a building that has several lines going down its surface can be seen going across the bottom half of the image. There is a rectangular, white-colored sign hanging off of the wall. The sign has a red border going around its edges. There is a large letter "A" in the middle of the sign that is made out of curving blue and light blue stripes. There is a blue rectangle going across the bottom of the sign that has "ALDI" going across it in white lettering. There is a white-colored object on the right side of the wall that has a rounded red top. Several trees can be seen rising up from behind the building on the right hand side of the image. It is daytime, and the sky above has gray clouds floating across its surface.*

Figure 14. Additional visualizations of the main results on the DCI dataset, comparing FLUX and W8A8 DiT with W4A8 quantization.



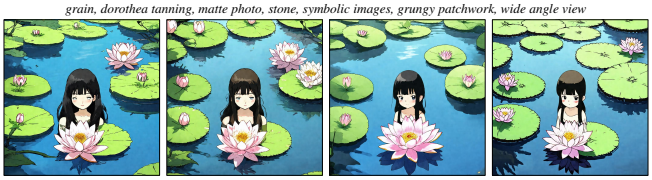
Prompt: *cartoon, vintage poster look, poke bowl, salmon, with fresh vegetables, with a bowl of rice, hand drawn, tropical sunset, chopsticks, 4k*

Prompt: *Star Wars poster design by Obey, retro 1980 street color propaganda style, arrangement with golden ratio, high quality render texture*

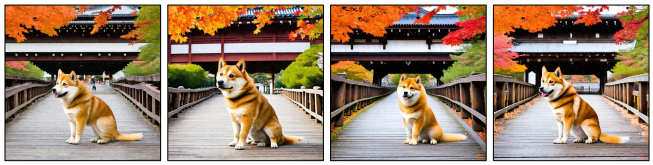
Prompt: *Castle made of smoke, castle, Evil, scary, dark, ghostly, green aurora borealis in the sky, spooky, intricate detail, high resolution, atmospheric, 4K HDR, Film Still, Hyperrealistic, Disney movie*

Prompt: *an old black mans face painted on a wall with some cracks, in the style of topographical realism, grit and grain, dorothea tanning, matte photo, stone, symbolic images, grungy patchwork, wide angle view*

Prompt: *kitchen design thinking studio, cinematic lighting, phorealistic, 8k*

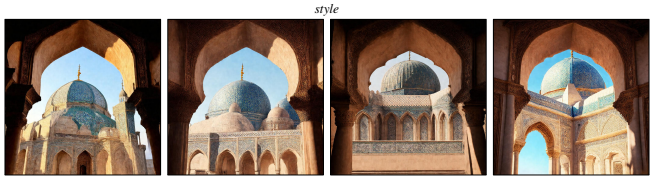


Prompt: *lotus blossoms ghibli style animation*



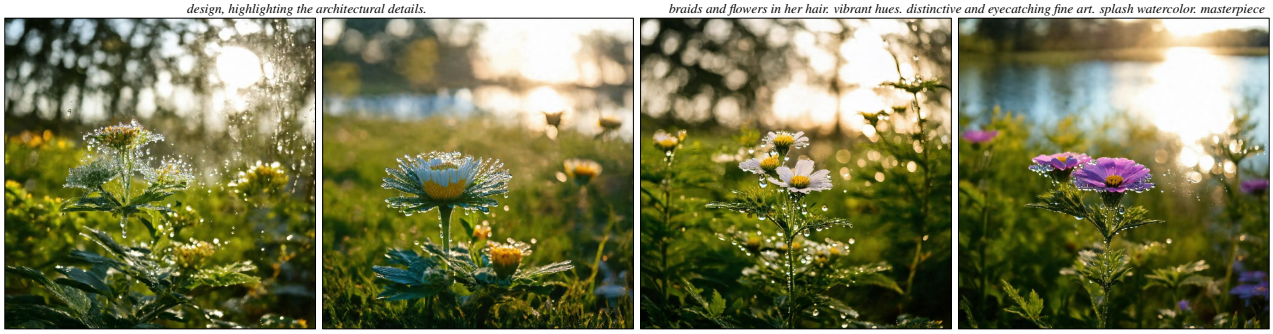
Prompt: *shibainu, dog, japan, dog on the bridge, behind of bridge exists japanese temple, leaves changing color, ziburi style*

Prompt: *temples of various religions stand as separate buildings in a beautiful forest on which sunlight falls and birds fly*



Prompt: *Create a hyperrealistic portrayal of arabesque architecture, showcasing the elaborate archways and domes adorned with intricate carvings and mosaics. Ensure that the camera lens captures the depth and shadows of the design, highlighting the architectural details.*

Prompt: *pretty anime girl. Kawaii Face Style. flowing black hair. sparkling blue eyes. dancing in a dreamy field of flowers. rays of shimmering light. polar stratospheric clouds in the background. wearing bohemianchic clothing. braids and flowers in her hair. vibrant hues. distinctive and eyecatching fine art. splash watercolor. masterpiece*



Prompt: *beautiful nature scene with dew dripping from flowers The photo was skillfully taken with a Nikon camera. The D850 DSLR paired with the versatile Nikkor 2470mm f2.8 lens, renowned for its sharpness and exceptional color reproduction. The f8 aperture is chosen to provide a deep depth of field and sharp detail capture of the entire scene. The ISO sensitivity is set to 200 and the shutter speed is 1500 second. photography uses bright, natural sunlight reflecting off a lake, illuminating the entire scene with harsh, cool light and highlighting the contrasting shadows that define the contours of the landscape.*

Figure 15. More visualization of ablation results on the MJHQ dataset with SD3.5 and W8A8 DiT quantization. From left to right: baseline, SEG., DUAL. and SEG.+DUAL.