COMPRESSED ADAPTATION OF DIFFUSION TRANSFORMERS VIA GLOBALLY-SENSITIVE SKETCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting diffusion transformers under tight memory budgets requires both compression and parameter-efficient fine-tuning. Existing sketch-based compression methods use locally sensitive sketches, where each lookup table is tied to a predetermined spatial region and cannot be shared by statistically similar weight groups at different locations. We propose *Globally-Sensitive Sketch (GSS)*, a unified framework for compression and parameter-efficient fine-tuning. GSS clusters weight groups within each weight matrix by normalized distributional shape, allowing similar groups to share learnable sketches across arbitrary matrix positions. Each weight stores a 4-bit index into its group's assigned sketch, and only the shared sketch entries are updated during fine-tuning. GSS is extensively evaluated on six diffusion transformers spanning 2B to 20B parameters. We consider two settings: adaptation for subject-driven personalization and paired image editing, and standalone text-to-image generation. GSS has the lowest mean weight quantization error among all evaluated 4-bit baselines and preserves text-to-image generation quality across all six backbones. In addition, GSS achieves competitive quality on both subject-driven personalization and paired image editing compared to full fine-tuning while using over 100$\times$ fewer trainable parameters. We provide our code at `https://anonymous.4open.science/r/gss-anonymization/`.

1 INTRODUCTION

Diffusion transformers (Peebles & Xie, 2023; Esser et al., 2024) have become a standard architecture for high-fidelity image generation, yet their billions of parameters make full fine-tuning prohibitively memory-intensive. Parameter-efficient fine-tuning (PEFT) methods such as LoRA (Hu et al., 2021) reduce the number of trainable parameters, but LoRA still requires the full-precision base model to remain in memory during training and inference. The base weights therefore remain a major component of model storage and training memory, regardless of how few parameters are updated.

A growing body of work addresses this limitation by *compressing* the base weights prior to adaptation. For example, QLoRA (Dettmers et al., 2023) uses a universal 4-bit codebook with supplementary low-rank adapters, while SketchTune (Zhang et al., 2024) enables direct fine-tuning of row-specific lookup tables. QLoRA sits at one extreme with one fixed codebook shared by all groups regardless of position or shape, while SketchTune sits at the other with learnable codebooks rigidly tied to rows. Neither lets a group select its codebook by distributional shape. In position-bound methods like SketchTune, a weight group in a given row must use the codebook assigned to that position, even when its values closely resemble groups elsewhere in the matrix. The result is duplicated tables for similar distributions in different rows, and no escape for atypical groups trapped in a mismatched row.

Our proposal. We introduce *Globally-Sensitive Sketch (GSS)*. Whereas prior sketches are locally sensitive in that each codebook is tied to a predetermined position, GSS is globally sensitive in that each group selects its sketch from a matrix-wide pool according to distributional shape rather than location. Each weight matrix has its own pool. GSS normalizes each contiguous group to remove scale differences, represents it by sorted local centroids, and routes it to the best-fitting sketch. Statistically similar groups reuse the same sketch wherever they appear, while dissimilar groups separate even within the same row. Group size controls scale precision and pool size controls codebook capacity, so the trainable footprint per matrix depends only on pool size and no longer

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

grows with row count. Codes and routing are fixed offline, and only the shared sketches are updated during fine-tuning. We summarize our main contributions as follows:

- We propose distribution-routed compression that replaces fixed universal and position-bound codebooks with a shared pool of learnable sketches selected by normalized shape, enabling cross-position reuse and escape from mismatched rows.
- We introduce a training rule with frozen codes and routing where only shared sketches are updated, keeping trainable parameters compact and independent of the number of rows in the weight matrix.
- We evaluate on six diffusion transformers spanning 2B to 20B parameters for standalone post-training quantization and compressed adaptation tasks, showing competitive quality and efficiency compared to the state-of-the-art compressed adaptation and full-precision baselines.

2 METHODS

This section motivates global sketch sharing, describes how GSS constructs a compressed weight representation offline, and explains how it fine-tunes only the shared sketch entries.

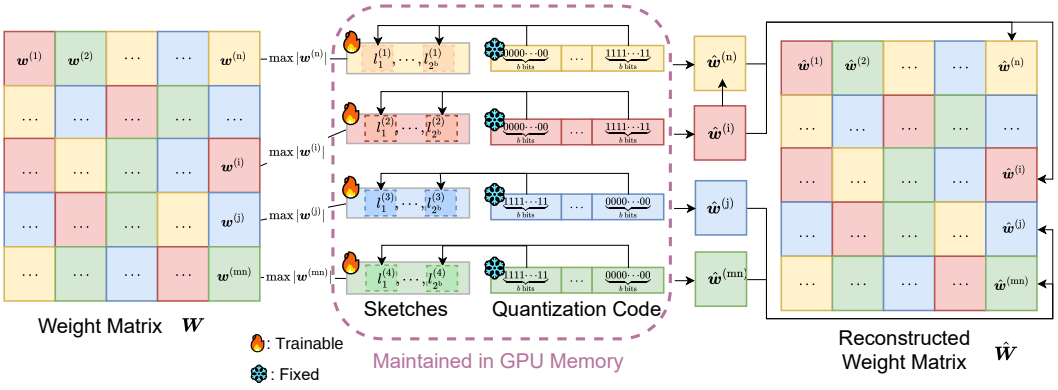

Figure 1: Weight reconstruction with GSS. Weight groups are clustered by normalized distributional shape, allowing groups at arbitrary positions in a matrix to draw from the same matrix-wide sketch pool. Each group stores a scale and a cluster assignment, and each weight stores an index into the assigned sketch. Only the shared sketch entries are updated during fine-tuning.

2.1 MOTIVATION

Prior lookup-based weight-compression methods (Zhang et al., 2024; Kim et al., 2023) tie each codebook to a predetermined spatial region, such as a row or block. This design creates a tradeoff between weight quantization error and storage cost. Smaller regions support more specialized codebooks but require more tables and metadata. Larger regions reduce this overhead but force a fixed-size codebook to represent a more heterogeneous distribution, thereby increasing weight quantization error. GSS breaks this coupling by separating the *group size* from the *pool size*. The group size N determines how many contiguous weights share a scale and a sketch assignment, whereas the pool size K determines how many sketches are available to the groups in a weight matrix. Groups with similar distributions can therefore share a sketch across spatial positions, while atypical groups can be assigned to clusters shared by fewer groups. Throughout this paper, *global* means matrix-wide: each weight matrix has its own sketch pool, and pools are not shared across matrices, since GSS performs layer-wise reconstruction during inference and fine-tuning.

2.2 COMPRESSED WEIGHT REPRESENTATION

As shown in Figure 1, we partition a weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ into $G = mn/N$ non-overlapping groups of N contiguous weights. Each group lies within one row, and N divides n. Each weight group $g \in \{0, \ldots, G-1\}$ has a positive scale $\mathbf{S}[g]$ and a cluster assignment $\mathbf{I}[g] \in \{0, \ldots, K-1\}$. Each position $j \in \{0, \ldots, N-1\}$ within the group has a b-bit code $c_{g,j} \in \{0, \ldots, 2^b-1\}$, stored in

Algorithm 1 Offline weight compression

Require: Weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$, number of shared sketches K, bit width b, and group size $N \geq 2^b$
Ensure: Compressed representation $(\mathbf{S}, \mathbf{I}, \mathbf{L}, \mathbf{C})$

Partition $\mathbf{W}$ into $G = mn/N$ groups $\{\mathbf{w}_g\}_{g=0}^{G-1}$, each with N weights
Scale Normalization & Local Signatures
for all groups $g = 0, \ldots, G-1$ **do**
  $\mathbf{S}[g] \leftarrow \max(\max_j |\mathbf{w}_g[j]|, 10^{-8})$; $\tilde{\mathbf{w}}_g \leftarrow \mathbf{w}_g \,/\, \mathbf{S}[g]$
  $\mathbf{s}_g \leftarrow \mathrm{sort}\big(\text{optimal-1D-k-means}(\tilde{\mathbf{w}}_g,\ 2^b)\big)$ ▷ *Local signature*
end for
Global Cluster Assignment
$\{\boldsymbol{\mu}_k\}_{k=0}^{K-1} \leftarrow \text{k-means}(\{\mathbf{s}_g\}_{g=0}^{G-1},\ K)$
for all groups $g = 0, \ldots, G-1$ **do**
  $\mathbf{I}[g] \leftarrow \arg\min_k \|\mathbf{s}_g - \boldsymbol{\mu}_k\|_2$
end for
Density-Weighted Sketch Optimization
for all nonempty clusters $k \in \{0, \ldots, K-1\}$ **do**
  $\boldsymbol{\pi}_k \leftarrow \mathrm{argsort}_{(g,j):\mathbf{I}[g]=k}\, \tilde{w}_{g,j}$ ▷ *Retain the original indices*
  $\mathcal{P}_k[i] \leftarrow \tilde{w}_{\boldsymbol{\pi}_k(i)}$ **for all** sorted positions i
  $\kappa \leftarrow \min\big(\max(\lfloor|\mathcal{P}_k|/100\rfloor, 5),\ \max(\lfloor|\mathcal{P}_k|/2\rfloor, 1)\big)$
  for all sorted positions i in $\mathcal{P}_k$ **do**
    $d_i^- \leftarrow |\mathcal{P}_k[i] - \mathcal{P}_k[i-\kappa]|$ if available, otherwise $+\infty$
    $d_i^+ \leftarrow |\mathcal{P}_k[i+\kappa] - \mathcal{P}_k[i]|$ if available, otherwise $+\infty$
    $d_i \leftarrow \min(d_i^-, d_i^+)$
    $\alpha_i \leftarrow 1\big/\big(\log(1 + 1/(d_i + 10^{-8})) + 10^{-8}\big)$ ▷ *Inverse-density importance score*
  end for
  $\mathbf{L}[k, :],\ \{c_{\boldsymbol{\pi}_k(i)}\}_i \leftarrow \text{weighted-k-means}(\mathcal{P}_k,\ 2^b,\ \{\alpha_i\}_i)$
end for
Code Packing
$\mathbf{C}[g] \leftarrow \mathrm{pack}_b(c_{g,0}, \ldots, c_{g,N-1})$ **for all** g
return $(\mathbf{S}, \mathbf{I}, \mathbf{L}, \mathbf{C})$

the packed code tensor $\mathbf{C}$. We collect the K shared sketches in $\mathbf{L} \in \mathbb{R}^{K\times 2^b}$, with one row per cluster and 2^b entries per sketch. We reconstruct each weight as

$$\hat{w}_{g,j} = \mathbf{S}[g] \cdot \mathbf{L}[\mathbf{I}[g],\, c_{g,j}]. \tag{1}$$

In our $b = 4$ implementation, $\mathbf{S}$ and $\mathbf{L}$ use BF16, $\mathbf{C}$ is a packed uint8 tensor of shape $G \times N/2$, and $\mathbf{I}$ uses 32-bit integers in CUDA kernels. During fine-tuning, $(\mathbf{S}, \mathbf{I}, \mathbf{C})$ remain frozen, and only the shared sketch entries in $\mathbf{L}$ are optimized, which enables parameter-efficient fine-tuning.

2.3 Weight Compression via Globally-Sensitive Sketch

GSS constructs $(\mathbf{S}, \mathbf{I}, \mathbf{L}, \mathbf{C})$ offline through normalization, matrix-wide clustering, density-weighted sketch fitting, and code packing, as shown in Algorithm 1.

To compare groups by distributional shape rather than magnitude, we divide each by its maximum absolute value. For a normalized group, let $\boldsymbol{\ell}_g \in \mathbb{R}^{2^b}$ denote its local centroids and $\mathbf{z}_g \in \{0, \ldots, 2^b-1\}^N$ their assignments. The optimal 1D k-means fit is

$$(\boldsymbol{\ell}_g^\star, \mathbf{z}_g^\star) \in \arg\min_{\boldsymbol{\ell}_g,\, \mathbf{z}_g} \sum_{j=0}^{N-1} \big(\tilde{w}_{g,j} - \ell_{g,z_{g,j}}\big)^2. \tag{2}$$

Its sorted centroids form the signature $\mathbf{s}_g = \mathrm{sort}(\boldsymbol{\ell}_g^\star)$. We cluster these signatures by solving

$$\min_{\{\boldsymbol{\mu}_k\}_{k=0}^{K-1},\, \mathbf{I}} \sum_{g=0}^{G-1} \Big\|\mathbf{s}_g - \boldsymbol{\mu}_{\mathbf{I}[g]}\Big\|_2^2, \tag{3}$$

where $\mathbf{I}[g] \in \{0, \ldots, K-1\}$, allowing similar groups to share a sketch across matrix positions.

For each cluster k, define $\mathcal{P}_k[i]$ as the ith smallest normalized weight pooled from all groups in the cluster, and let $\boldsymbol{\pi}_k(i) = (g, j)$ record its originating group and position. To assess local sparsity, we compute the distance d_i from each position to its closest neighbor κ indices away in the sorted list; a larger d_i results in a greater importance score α_i. This density-based weighting ensures that sketch capacity is distributed more evenly: regions with sparser weights receive higher weighting, allowing for improved accuracy on rare or outlier values rather than letting frequently occurring (dense) weight ranges dominate the sketch. The shared sketch for cluster k is then fitted by minimizing a density-weighted reconstruction loss over code assignments $z_{k,i} \in \{0, \ldots, 2^b-1\}$:

$$\min_{\mathbf{L}[k,:],\, \mathbf{z}_k} \sum_{i=0}^{|\mathcal{P}_k|-1} \alpha_i \big(\mathcal{P}_k[i] - \mathbf{L}[k, z_{k,i}]\big)^2. \tag{4}$$

The neighborhood spans 1% of the pool, with a minimum of five neighbors and a half-pool cap for small clusters; the numerical guard is fixed at 10^{-8} for all models. We map the fitted assignments back as $c_{\boldsymbol{\pi}_k(i)} = z_{k,i}$ and pack the codes into $\mathbf{C}$.

2.4 Fine-Tuning via Globally-Sensitive Sketch

Algorithm 2 Fine-tuning a compressed linear layer

Require: Frozen $(\mathbf{S}, \mathbf{I}, \mathbf{C})$, trainable shared sketches $\mathbf{L} \in \mathbb{R}^{K \times 2^b}$, input $\mathbf{X}$, and upstream gradient $\nabla_{\mathbf{Y}}\mathcal{L}$
Ensure: Output $\mathbf{Y}$, gradients $\nabla_{\mathbf{X}}\mathcal{L}$ and $\nabla_{\mathbf{L}}\mathcal{L}$

Forward Pass
for all groups g and positions j **in parallel do**
  $c_{g,j} \leftarrow \mathrm{unpack}_b(\mathbf{C}[g], j)$
  $\hat{w}_{g,j} \leftarrow \mathbf{S}[g] \cdot \mathbf{L}[\mathbf{I}[g],\ c_{g,j}]$ ▷ *Reconstruct weight*
end for
$\hat{\mathbf{W}} \leftarrow \mathrm{reshape}(\{\hat{w}_{g,j}\}_{g,j},\ m,\ n)$
$\mathbf{Y} \leftarrow \mathbf{X}\hat{\mathbf{W}}^\top$
Backward Pass ▷ *Given upstream gradient* $\nabla_{\mathbf{Y}}\mathcal{L}$
$\nabla_{\mathbf{X}}\mathcal{L} \leftarrow \nabla_{\mathbf{Y}}\mathcal{L}\ \hat{\mathbf{W}}$ ▷ *Input gradient*
$\nabla_{\hat{\mathbf{W}}}\mathcal{L} \leftarrow (\nabla_{\mathbf{Y}}\mathcal{L})^\top \mathbf{X}$ ▷ *Weight gradient*
$\{\nabla_{\hat{w}_{g,j}}\mathcal{L}\}_{g,j} \leftarrow \mathrm{reshape}(\nabla_{\hat{\mathbf{W}}}\mathcal{L},\ G,\ N)$
$\nabla_{\mathbf{L}}\mathcal{L} \leftarrow \mathbf{0}$
for all groups g and positions j **in parallel do**
  $k \leftarrow \mathbf{I}[g]; \quad i \leftarrow c_{g,j}$
  $\nabla_{\mathbf{L}}[k, i]$ += $\nabla_{\hat{w}_{g,j}}\mathcal{L} \cdot \mathbf{S}[g]$ ▷ *Reduce over shared references*
end for
return $\mathbf{Y},\ \nabla_{\mathbf{X}}\mathcal{L},\ \nabla_{\mathbf{L}}\mathcal{L}$

In the forward pass, we reconstruct each weight by unpacking its code, retrieving the corresponding entry from its assigned sketch, and multiplying that entry by the group scale. The resulting weight matrix is then multiplied by the input to produce the layer output. The backward pass is standard except for the gradients of the shared sketch entries. Because each reconstructed weight is the product of a sketch entry and its group scale, the gradient of each shared entry is the sum of the scale-weighted gradients from all weights that reference it. Groups with larger scales therefore contribute proportionally more. Algorithm 2 formalizes the weight-path computation for a single compressed linear layer; a frozen bias, if present, is omitted for clarity. We implement both passes as custom CUDA kernels that operate directly on the packed representation. In our $b = 4$ implementation, the forward kernel caches the $K \times 2^b$ BF16 sketch entries in shared memory while parallel warps decode the codes. The backward kernel first accumulates sketch gradients in shared memory, then performs at most one global-memory atomic update per touched sketch entry from each thread block. This design reduces global atomic contention.

3 EXPERIMENTS

We assess the capabilities of GSS under two experimental regimes. For compressed adaptation, we fine-tune only the values within the shared sketch pools, leaving all other parameters fixed. In the standalone post-training quantization setting, GSS is applied without any additional adaptation to specific tasks. Our experiments aim to answer the following three research questions:

- **RQ1:** How does GSS compare with full-precision and 4-bit compressed adaptation baselines in adaptation quality and parameter efficiency?
- **RQ2:** As a standalone 4-bit quantizer, how well does GSS preserve text-to-image generation quality relative to existing post-training quantization methods?
- **RQ3:** How do the shared sketch pool size and density-weighted fitting affect compression quality?

Backbones and datasets. Our evaluation covers six diffusion transformer backbones, ranging from 2B to 20B parameters: FLUX.1-dev, FLUX.1-Kontext, Qwen-Image, SD3.5-large, SD3-medium, and Lumina-Image-2.0. For compressed adaptation, we evaluate subject-driven personalization on all six backbones and paired image editing on FLUX.1-Kontext, since paired image editing is uniquely supported by this model. We omit generic text-to-image generation from adaptation experiments, as it primarily assesses pre-trained capabilities rather than transfer or adaptation. For subject-driven personalization, we use the DreamBooth dataset (Ruiz et al., 2023), fine-tuning pre-trained models on its reference images. Among 25 provided prompts in DreamBooth, we allocate 10 for validation and the remaining 15 for testing. For paired image editing, we employ the InstructPix2Pix (Brooks et al., 2023) dataset, divided into 600 train, 200 validation, and 200 test examples. The modest size of these datasets allows us to systematically compare methods and perform comprehensive hyperparameter sweeps using personal-use hardware, which faithfully represents the resource-constrained environments motivating PEFT (Han et al., 2024). Extending these experiments to significantly larger datasets would demand far greater computational resources and falls outside the scope of both PEFT and this work. In the post-training quantization regime, we evaluate text-to-image generation for all six backbones using 1024 prompts from each of two datasets: MJHQ-30K (Li et al., 2024a) and sDCI (Urbanek et al., 2024). Details regarding the hyperparameter ranges and optimization procedures are provided in Appendix A, and the evaluation metrics used throughout our experiments are described in Appendix C.

Baselines. For all compression and fine-tuning experiments, GSS is configured with 1024 shared sketches per weight matrix and employs density-weighted fitting, reducing the weights to 4-bit representations. We compare against publicly available state-of-the-art 4-bit baselines that have demonstrated reliable performance in our settings. Our focus on the 4-bit regime is informed by prior work showing that lower precisions cause substantial degradation in generation quality (Li et al., 2023; Sui et al., 2024; Zhao et al., 2025), and by entropy-based analyses indicating a lossless encoding limit of approximately 4.67 bits per weight (Yang et al., 2026). In the compressed adaptation setting, we benchmark GSS against full fine-tuning, LoRA (Hu et al., 2021), QLoRA (Dettmers et al., 2023), and SketchTune (Zhang et al., 2024). Full fine-tuning updates all transformer weights in BF16. LoRA and QLoRA fine-tune rank-4 adapters on top of frozen BF16 and 4-bit NF4 backbones, respectively, while SketchTune learns four density-weighted sketches for each output row. For post-training quantization, we evaluate GSS alongside NF4, SketchTune, and OrbitQuant (Lee et al., 2026), a recent state-of-the-art post-training quantization technique for diffusion transformers. Additional related approaches are discussed in Section 4, though we exclude methods superseded by our chosen baselines or those that would require significant additional engineering to reproduce.

3.1 EFFECTIVENESS OF COMPRESSED ADAPTATION

To address **RQ1**, we evaluate GSS as a 4-bit compressed adaptation technique for subject-driven personalization and paired image editing, and present the results in Table 1.

Compressed adaptation. On subject-driven personalization, GSS ranks first or second among PEFT approaches on every backbone and metric, at times matching or exceeding full fine-tuning. LLM judgments give the clearest signal, with GSS leading PEFT on FLUX.1-dev with 0.754 versus 0.560 for full fine-tuning and placing second on SD3-medium with 0.800 versus 0.518 for full fine-tuning. On paired image editing with FLUX.1-Kontext, GSS is equally strong, tying for best

Table 1: Compressed adaptation performance on **(a)** subject-driven personalization across six backbones and **(b)** paired image editing on FLUX.1-Kontext.

(a) Subject-driven personalization

Backbone	Method	Trainable Params (M)	CLIP-I ↑	CLIP-T ↑	DIV ↑	IR ↑	LLM ↑
FLUX.1-dev	Full FT	11901	0.799	0.267	0.548	1.203	0.560
	LoRA	11.21	0.848	0.244	0.563	0.818	0.720
	QLoRA	11.21	0.874	0.241	0.519	0.508	0.734
	SketchTune	186.78	0.849	0.250	0.583	0.770	0.742
	GSS	7.47	0.869	0.262	0.569	1.083	0.754
FLUX.1-Kontext	Full FT	11901	0.782	0.256	0.552	1.023	0.571
	LoRA	11.21	0.801	0.260	0.582	0.863	0.576
	QLoRA	11.21	0.823	0.260	0.561	0.991	0.632
	SketchTune	186.78	0.824	0.250	0.572	0.734	0.636
	GSS	7.47	0.823	0.261	0.575	0.921	0.642
Qwen-Image	Full FT	20430	0.841	0.262	0.610	1.213	0.813
	LoRA	11.80	0.825	0.257	0.613	1.056	0.760
	QLoRA	11.80	0.790	0.244	0.638	0.587	0.649
	SketchTune	172.62	0.850	0.236	0.650	0.289	0.676
	GSS	13.76	0.845	0.257	0.641	1.092	0.793
SD3.5-large	Full FT	8057	0.828	0.271	0.567	1.309	0.840
	LoRA	5.90	0.831	0.274	0.568	1.302	0.833
	QLoRA	5.90	0.854	0.260	0.509	1.233	0.813
	SketchTune	175.88	0.882	0.247	0.472	0.605	0.764
	GSS	8.67	0.877	0.269	0.538	1.278	0.820
SD3-medium	Full FT	2028	0.788	0.267	0.582	1.140	0.518
	LoRA	2.35	0.821	0.263	0.580	1.263	0.602
	QLoRA	2.35	0.813	0.258	0.577	1.104	0.653
	SketchTune	69.80	0.866	0.246	0.415	0.929	0.820
	GSS	5.46	0.861	0.262	0.579	1.137	0.800
Lumina-Image-2.0	Full FT	2610	0.828	0.267	0.509	1.439	0.718
	LoRA	1.84	0.827	0.267	0.513	1.287	0.611
	QLoRA	1.84	0.781	0.268	0.503	1.386	0.400
	SketchTune	68.12	0.799	0.268	0.481	1.480	0.582
	GSS	3.90	0.823	0.268	0.512	1.450	0.698

(b) Paired image editing

Backbone	Method	Trainable Params (M)	CLIP-I ↑	CLIP-T ↑	SSIM ↑	PSNR ↑	LLM ↑
FLUX.1-Kontext	Full FT	11901	0.817	0.203	0.427	12.47	0.468
	LoRA	11.21	0.812	0.205	0.419	12.11	0.472
	QLoRA	11.21	0.811	0.203	0.427	12.37	0.445
	SketchTune	186.78	0.805	0.203	0.427	12.27	0.436
	GSS	7.47	0.813	0.207	0.429	12.41	0.472

LLM score at 0.472 and leading on CLIP-T and SSIM. Fine-tuning outcomes depend on many factors beyond the method itself. We keep the protocol identical across methods and backbones and sweep only the learning rate with the same number of trials for each method. Refer to Appendix A for a comprehensive description of the candidate learning rate ranges and the selection process for the optimal fine-tuning checkpoint. Under the matched budget, the results in Table 1 support GSS as a valid and competitive compressed adaptation method.

Parameter efficiency. The adaptation quality of GSS comes with a compact trainable footprint, more than two orders of magnitude fewer than full fine-tuning. Each PEFT method scales differently. LoRA and QLoRA use rank-4 adapters whose parameter count grows linearly with matrix size at fixed rank. SketchTune learns row-local sketches whose count grows linearly with output rows. In contrast, GSS uses one shared pool per weight matrix whose per-matrix parameter count stays constant once pool size is configured and independent of matrix dimensions. This also explains why GSS is not the

smallest on every backbone. On backbones with small weight matrices, rank-4 adapters can use fewer parameters, while the GSS advantage grows as matrices get larger, and GSS is the smallest among PEFT methods on FLUX. We fix rank at 4 for LoRA and QLoRA for convenience, since tuning rank per backbone and task would multiply an already large sweep beyond our compute budget. Appendix B complements these counts with memory and latency analysis on FLUX.1-Kontext.

3.2 Post-Training Quantization Quality

To address **RQ2**, we assess whether GSS maintains text-to-image generation quality when applied as a standalone 4-bit post-training quantizer. We also evaluate the associated weight quantization error. The outcomes are summarized in Table 2 and visualized in Figure 2.

Table 2: Post-training quantization quality across six backbones. Arrows indicate the preferred direction. Because BF16 serves as the similarity reference, its LPIPS and PSNR entries are omitted.

Backbone	Method	MJHQ				sDCI			
		Quality		Similarity		Quality		Similarity	
		FID ↓	IR ↑	LPIPS ↓	PSNR ↑	FID ↓	IR ↑	LPIPS ↓	PSNR ↑
FLUX.1-dev	BF16	59.1	0.92	–	–	63.6	1.00	–	–
	NF4	59.1	0.91	0.22	21.3	64.7	0.97	0.24	19.7
	OrbitQuant	59.1	0.87	0.25	20.2	65.1	0.95	0.27	19.0
	SketchTune	60.0	0.89	0.24	20.9	66.2	0.94	0.26	19.5
	GSS	59.8	0.91	0.22	21.2	65.6	0.97	0.24	19.8
FLUX.1-Kontext	BF16	68.7	0.69	–	–	68.1	0.83	–	–
	NF4	68.8	0.65	0.22	19.8	68.9	0.79	0.23	18.9
	OrbitQuant	68.5	0.64	0.24	19.4	68.5	0.79	0.25	18.2
	SketchTune	69.5	0.64	0.23	19.6	70.0	0.79	0.24	18.3
	GSS	69.5	0.64	0.21	20.2	69.1	0.80	0.22	19.0
Qwen-Image	BF16	58.8	0.85	–	–	60.5	0.71	–	–
	NF4	141.0	-0.86	0.60	18.6	156.0	-1.09	0.62	17.9
	OrbitQuant	282.0	-2.14	0.73	14.7	329.0	-2.18	0.72	13.3
	SketchTune	71.8	0.56	0.31	21.0	70.9	0.41	0.33	20.1
	GSS	64.7	0.69	0.27	21.5	66.9	0.51	0.31	20.6
SD3.5-large	BF16	59.4	0.99	–	–	62.4	1.04	–	–
	NF4	59.1	0.98	0.45	13.7	62.4	1.01	0.43	13.7
	OrbitQuant	65.9	0.87	0.61	11.9	67.7	0.92	0.59	12.3
	SketchTune	60.0	1.02	0.45	14.0	62.8	1.05	0.43	14.0
	GSS	60.1	0.99	0.42	14.5	64.1	1.00	0.40	14.4
SD3-medium	BF16	62.5	0.90	–	–	63.1	1.00	–	–
	NF4	69.6	0.70	0.53	12.9	69.3	0.84	0.52	13.5
	OrbitQuant	129.0	-0.18	0.68	10.6	125.0	-0.14	0.69	10.9
	SketchTune	67.0	0.80	0.49	13.4	67.7	0.89	0.48	13.8
	GSS	65.6	0.80	0.47	13.7	67.0	0.90	0.46	14.1
Lumina-Image-2.0	BF16	59.3	1.04	–	–	62.5	1.06	–	–
	NF4	59.1	1.03	0.14	22.8	62.1	1.05	0.17	20.8
	OrbitQuant	59.0	1.01	0.21	21.4	61.6	1.05	0.23	19.7
	SketchTune	58.8	1.02	0.15	23.1	62.8	1.04	0.17	21.2
	GSS	59.7	1.02	0.13	23.8	63.5	1.04	0.15	21.9

Text-to-image generation. Similarity to same-prompt, same-seed BF16 generations provides the clearest signal of quantizer quality, with GSS ranking best or tied in 23 of 24 LPIPS and PSNR comparisons across backbones and datasets. The sole exception is FLUX.1-dev PSNR on the MJHQ dataset, where NF4 has 21.3 versus 21.2 for GSS. The contrast is most striking on Qwen-Image,

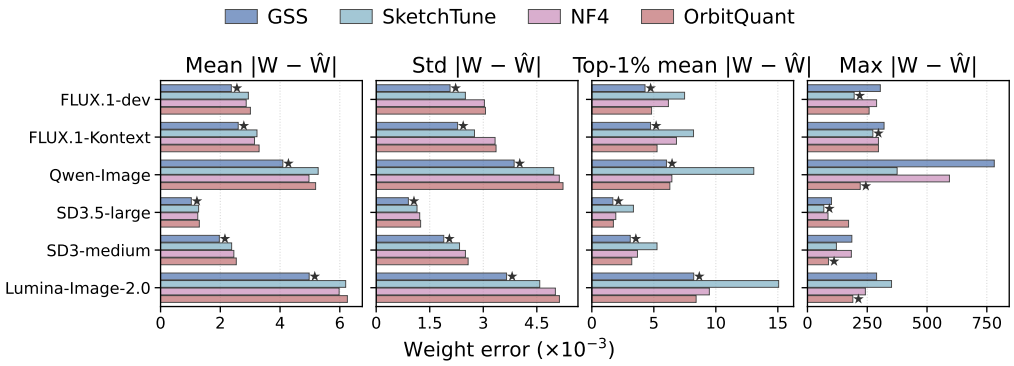

Figure 2: Layer-size-weighted absolute weight quantization error $|W - \hat{W}|$. Panels report the mean, standard deviation, top-1% mean, and maximum across all linear layers; stars mark the lowest error for each backbone. Lower is better, and scales differ across panels.

where on the MJHQ dataset, GSS holds LPIPS down to 0.27 and PSNR up to 21.5, well ahead of NF4 and OrbitQuant and clear of SketchTune. FID and IR instead measure absolute quality rather than fidelity to BF16. Here no method wins on every backbone, but GSS stays closest to BF16 overall, with the largest residual gap on Qwen-Image.

Weight quantization error. We complement the text-to-image generation evaluation with direct weight quantization error. As shown in Figure 2, GSS achieves the lowest mean, standard deviation, and top-1% mean on every backbone. Maximum error is the sole exception. Each matrix-wide sketch in GSS is shared across positions, which may lead to some isolated outlier weights not being represented well. We leave improved outlier handling to future work.

3.3 Ablation Studies



Figure 3: Effects of shared sketch pool size and density-weighted fitting on weight quantization error. Open and filled markers denote uniform and density-weighted fitting, respectively, while darker shades indicate larger pools. Values are relative to the 256-sketch uniform baseline for each backbone.

To address **RQ3**, we ablate two design choices in GSS: shared sketch pool size and density-weighted fitting. Given our compute budget, evaluating fine-tuning and text-to-image generation quality for every configuration on all six backbones is too expensive, so we analyze weight quantization error instead and present the results in Figure 3.

Sketch pool size and density-weighted fitting. Larger pools reduce all four error statistics on every backbone, most clearly in the tail and maximum errors. Density-weighted fitting trades slightly higher

mean error for tighter spread and lower maximum error. We use it by default for more balanced outlier coverage; uniform fitting is preferable if only mean error matters.

4 Related Work

Parameter-efficient fine-tuning. PEFT methods adapt a pre-trained model by updating only a small fraction of its parameters. LoRA (Hu et al., 2021) injects trainable low-rank decomposition matrices while freezing the base model, but relies on maintaining full-precision weights. To reduce the memory footprint, QLoRA (Dettmers et al., 2023) and LoftQ (Li et al., 2024c) combine low-rank adapters with quantized base weights. For diffusion models, recent approaches utilize PEFT to recover the performance degradation caused by low-bit quantization, such as selective layer updating in QuEST (Wang et al., 2025) and quantization-aware low-rank adapters in EfficientDM (He et al., 2024). In contrast, GSS directly optimizes shared sketches within the compressed representation, avoiding auxiliary low-rank branches while eliminating the need to store a full-precision backbone during either training or inference.

Weight quantization for diffusion models. General post-training quantization techniques such as GPTQ (Frantar et al., 2022) and AWQ (Lin et al., 2024), originally designed for large language models, rely on second-order error compensation or activation-aware scaling to retain model quality. When extended to diffusion models, quantizers must additionally handle dynamic, timestep-dependent activation trajectories. Existing diffusion-specific approaches address this using calibration data, timestep-aware reconstruction objectives, or targeted fine-tuning (Shang et al., 2023; Li et al., 2023; Zhao et al., 2025). Recent work explores more aggressive schemes, such as sub-2-bit weight representation in BitsFusion (Sui et al., 2024), outlier-channel absorption via low-rank residuals in SVDQuant (Li et al., 2024b), and data-agnostic orthogonal rotations in OrbitQuant (Lee et al., 2026). Our evaluation aligns closely with data-free and sketch-based baselines, including NF4 (Dettmers et al., 2023), SketchTune (Zhang et al., 2024), and OrbitQuant. Notably, both GSS and OrbitQuant operate as purely data-free weight quantizers without requiring calibration activations.

Learnable non-uniform codebooks. Beyond uniform grids, SqueezeLLM (Kim et al., 2023) constructs sensitivity-weighted non-uniform codebooks, while SketchTune (Zhang et al., 2024) optimizes row-local sketches during fine-tuning. However, both tie each codebook strictly to fixed structural partitions, such as individual rows. In contrast, GSS decouples codebook assignment from matrix coordinates by routing weight groups to sketches based on their normalized distributional profiles from a matrix-wide shared pool. This dynamic grouping enables cross-position codebook sharing and prevents atypical weight blocks from being degraded by rigid row-level constraints.

5 Conclusion and Future Work

We introduced Globally-Sensitive Sketch (GSS), a unified framework for 4-bit compression and parameter-efficient fine-tuning of diffusion transformers. By clustering normalized weight groups by distributional shape, GSS enables statistically similar groups to share learnable sketches across arbitrary matrix positions, decoupling group size from pool size so that the trainable parameter footprint depends solely on pool size rather than matrix row count. During adaptation, group scales, cluster assignments, and 4-bit codes remain fixed while only the shared sketch entries are updated. Across six diffusion transformers spanning 2B–20B parameters, GSS achieves the lowest mean, standard deviation, and top-1% mean absolute weight quantization error among evaluated 4-bit methods, and closely preserves BF16 generation fidelity. Furthermore, it matches or exceeds PEFT baselines on subject-driven personalization and paired image editing with only few million trainable parameters across all six backbones.

Looking forward, isolated outlier weights remain challenging for a matrix-wide shared pool, leading to higher maximum error. A promising direction is incorporating a sparse residual matrix to capture extreme values. In addition, because cluster assignments and codes are fixed offline, jointly optimizing routing during adaptation presents an avenue to expand capacity beyond the parameter-efficient regime toward full fine-tuning. Finally, investigating whether matrix-wide sketch sharing generalizes to other modalities and architectures, including autoregressive language models and video generators, represents an exciting direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine research hypotheses, design or provide feedback on research methodology or experiments, support qualitative and thematic data analysis, or interpret results. Other tasks, such as generating synthetic data sets, assisting with translation, and cleaning or reformatting datasets, are not applicable to this work. Additionally, we used generative AI tools to create or edit software code and edit the manuscript to improve readability. We have reviewed all AI-assisted work: all generated code was verified, tested for correctness, and integrated by the authors, and all suggested textual edits were manually vetted for clarity and factual precision. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, all algorithms introduced in this work are formalized with detailed step-by-step procedures in Algorithms 1 and 2. Comprehensive hyperparameter search grids, optimization configurations, and candidate learning rate ranges are provided in Appendix A and Table 3. Evaluation metrics and their sources are described in Appendix C. Complete source code is available in an anonymous repository at `https://anonymous.4open.science/r/gss-anonymization/`.

ETHICS STATEMENT

This work presents a parameter-efficient framework for adapting diffusion transformers, evaluated strictly on established, publicly available open-source benchmarks and architectures. By substantially reducing computational and memory requirements, our method democratizes access to fine-tuning generative models on consumer-grade hardware. However, lowering barriers to generative adaptation carries potential dual-use risks, including the unauthorized creation of misleading media, non-consensual imagery, or biased representations. While our framework does not inherently alter the base models' foundational safety alignments, it facilitates domain-specific tuning that could circumvent coarse filtering. We strongly advocate for responsible deployment, the integration of downstream watermarking and provenance standards, and proactive alignment monitoring within democratized fine-tuning workflows.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18392–18402. IEEE, 2023.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

Yefei He, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Efficientdm: Efficient quantization-aware fine-tuning of low-bit diffusion models. In *International Conference on Learning Representations*, volume 2024, pp. 15731–15750, 2024.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

Donghyun Lee, Jitesh Chavan, Duy Nguyen, Sam Huang, Liming Jiang, Priyadarshini Panda, Timo Mertens, and Saurabh Shukla. Orbitquant: Data-agnostic quantization for image and video diffusion transformers. *arXiv preprint arXiv:2607.02461*, 2026.

Daiqing Li, Aleks Kamko, Ehsan Akhgari, Ali Sabet, Linmiao Xu, and Suhail Doshi. Playground v2. 5: Three insights towards enhancing aesthetic quality in text-to-image generation. *arXiv preprint arXiv:2402.17245*, 2024a.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024b.

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17489–17499. IEEE, 2023.

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 13409–13424, 2024c.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6: 87–100, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22500–22510. IEEE, 2023.

Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1972–1981. IEEE, 2023.

Yang Sui, Yanyu Li, Anil Kag, Yerlan Idelbayev, Junli Cao, Ju Hu, Dhritiman Sagar, Bo Yuan, Sergey Tulyakov, and Jian Ren. Bitsfusion: 1.99 bits weight quantization of diffusion model. *Advances in Neural Information Processing Systems*, 37:76775–76818, 2024.

Jack Urbanek, Florian Bordes, Pietro Astolfi, Mary Williamson, Vasu Sharma, and Adriana Romero-Soriano. A picture is worth more than 77 text tokens: Evaluating clip-style models on dense captions. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26690–26699. IEEE, 2024.

Haoxuan Wang, Yuzhang Shang, Zhihang Yuan, Junyi Wu, Junchi Yan, and Yan Yan. Quest: Low-bit diffusion model quantization via efficient selective finetuning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15542–15551. IEEE, 2025.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

Zeyu Yang, Tianyi Zhang, Jianwen Xie, Chuan Li, Zhaozhuo Xu, and Anshumali Shrivastava. To compress or not? pushing the frontier of lossless genai model weights compression with exponent concentration. In *International Conference on Learning Representations*, volume 2026, pp. 53467–53487, 2026.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Tianyi Zhang, Junda Su, Aditya Desai, Oscar Wu, Zhaozhuo Xu, and Anshumali Shrivastava. Sketch to adapt: fine-tunable sketches for efficient llm adaptation. *arXiv preprint arXiv:2410.06364*, 2024.

Maosen Zhao, Pengtao Chen, Chong Yu, Yan Wen, Xudong Tan, and Tao Chen. Pioneering 4-bit fp quantization for diffusion models: Mixup-sign quantization and timestep-aware fine-tuning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18134–18143. IEEE, 2025.

A HYPERPARAMETER SEARCH

Algorithm 3 Learning-rate and checkpoint selection

Require: Pretrained backbone M; training, validation, and test sets $(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{validation}}, \mathcal{D}_{\text{test}})$; candidate learning rates $\mathcal{H}$
Ensure: Selected learning rate η^* and checkpoint C^*

```
Select a fixed screening set S containing examples from D_validation
Evaluate all trials
for all learning rates η ∈ H do
    Initialize an independent trial from the pretrained backbone M
    s*_η ← −∞                                              ▷ Best screening score for η
    for t ∈ {50, 100, ..., 750} do
        Continue training on D_train with η through step t and save C_{η,t}
        s ← mean LLM score of C_{η,t} on S
        if s > s*_η then
            C*_η ← C_{η,t};   s*_η ← s
        end if
        if 5 consecutive checkpoints fail to improve s*_η then
            break                                          ▷ Early stopping
        end if
    end for
    Evaluate C*_η on D_validation to obtain mean LLM score v_η
end for
Select the best trial
η* ← arg max_{η∈H} v_η;   C* ← C*_{η*}
Evaluate C* on D_test
return η*, C*
```

Table 3: Learning-rate search ranges. Pers. is short for subject-driven personalization. Edit. is short for paired image editing. Full FT is short for full fine-tuning.

Method	Task	Backbones	Search Range
Full FT	Pers.	All except Lumina-Image-2.0	$[5.0\times10^{-7},\ 1.0\times10^{-6}]$
	Pers.	Lumina-Image-2.0	$[2.0\times10^{-6},\ 5.0\times10^{-5}]$
	Edit.	FLUX.1-Kontext	$[5.0\times10^{-7},\ 1.0\times10^{-6}]$
SketchTune	Pers.	All	$[2.0\times10^{-5},\ 8.0\times10^{-5}]$
	Edit.	FLUX.1-Kontext	$[2.0\times10^{-5},\ 8.0\times10^{-5}]$
GSS	Pers.	FLUX.1-dev, FLUX.1-Kontext, Lumina-Image-2.0	$[3.0\times10^{-4},\ 1.0\times10^{-3}]$
	Pers.	Qwen-Image, SD3.5-large, SD3-medium	$[1.0\times10^{-4},\ 6.0\times10^{-4}]$
	Edit.	FLUX.1-Kontext	$[3.0\times10^{-4},\ 1.0\times10^{-3}]$
LoRA	Pers.	FLUX.1-dev	$[7.0\times10^{-5},\ 2.8\times10^{-4}]$
	Pers.	FLUX.1-Kontext	$[7.0\times10^{-5},\ 4.0\times10^{-4}]$
	Edit.	FLUX.1-Kontext	$[2.7\times10^{-5},\ 1.1\times10^{-4}]$
	Pers.	Qwen-Image	$[2.7\times10^{-5},\ 1.1\times10^{-4}]$
	Pers.	SD3.5-large, SD3-medium	$[3.5\times10^{-5},\ 1.4\times10^{-4}]$
	Pers.	Lumina-Image-2.0	$[1.9\times10^{-4},\ 7.8\times10^{-4}]$
QLoRA	Pers.	FLUX.1-dev, Qwen-Image	$[7.0\times10^{-5},\ 2.8\times10^{-4}]$
	Pers.	FLUX.1-Kontext	$[7.0\times10^{-5},\ 4.0\times10^{-4}]$
	Edit.	FLUX.1-Kontext	$[2.7\times10^{-5},\ 1.1\times10^{-4}]$
	Pers.	SD3.5-large, SD3-medium	$[3.5\times10^{-5},\ 1.4\times10^{-4}]$
	Pers.	Lumina-Image-2.0	$[1.9\times10^{-4},\ 7.8\times10^{-4}]$

Each adaptation trial is trained using AdamW for up to 750 steps with a batch size of 1 and a constant learning rate. Across each method and backbone, we use Optuna's grid sampler (Akiba et al., 2019) to test 16 logarithmically spaced learning rates from the ranges in Table 3. As outlined in Algorithm 3,

a fixed screening set guides early stopping and intra-trial checkpoint selection. The top candidate from each trial is then evaluated on the full validation set, and the checkpoint with the highest LLM score is selected for final evaluation on the test set.

B MEMORY AND LATENCY

Trainable parameter counts reflect storage cost but do not capture runtime efficiency. Rather than profiling every backbone separately, we evaluate FLUX.1-Kontext as a representative testbed because it is a large backbone that supports both downstream adaptation tasks. We measure peak GPU memory and wall-clock time during training and inference for subject-driven personalization on DreamBooth and paired image editing on InstructPix2Pix, each operating at its native benchmark resolution (1024 px and 512 px, respectively). Memory profiling is restricted to the transformer backbone and excludes auxiliary pipeline components, while time profiling is end-to-end. Because standalone text-to-image generation shares the identical inference pipeline and resolution as subject-driven personalization, its runtime cost is equivalent and not reported separately.

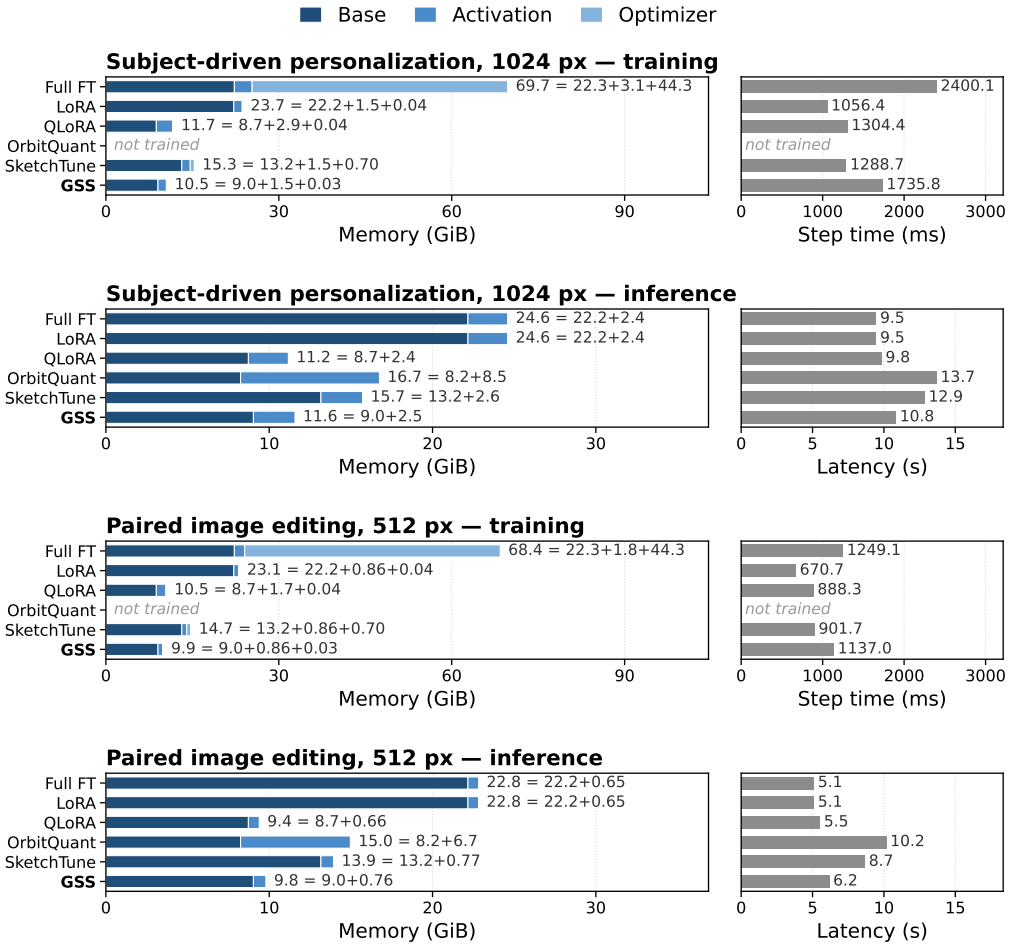

Figure 4: Training and inference costs on FLUX.1-Kontext for subject-driven personalization at 1024 px and paired image editing at 512 px. Stacked bars show the memory usage of the base model, activations, and optimizer state. Adjacent bars show the training-step time or generation latency. OrbitQuant supports inference only and therefore has no training measurement.

C Evaluation Metrics

Table 4 defines each evaluation metric and identifies the tasks to which it applies.

Table 4: Evaluation metrics and their applicable tasks.

Metric	Applicable tasks	Interpretation
FID ↓	✓ Text-to-image generation	Fréchet distance between the feature distributions of generated and reference images (Heusel et al., 2017)
IR ↑	✓ Subject-driven personalization ✓ Text-to-image generation	Prompt-conditioned image preference estimated by a pre-trained model (Xu et al., 2023)
LPIPS ↓	✓ Text-to-image generation	Perceptual distance to the BF16 reference generated from the same prompt and seed (Zhang et al., 2018)
PSNR ↑	✓ Paired image editing ✓ Text-to-image generation	Pixel-level similarity to the paired image editing target or the same-prompt, same-seed BF16 reference
CLIP-I ↑	✓ Subject-driven personalization ✓ Paired image editing	CLIP image similarity between the generated image and the reference subject images or paired editing target (Radford et al., 2021)
CLIP-T ↑	✓ Subject-driven personalization ✓ Paired image editing	CLIP text–image similarity between the generated image and the prompt or editing instruction (Radford et al., 2021)
DIV ↑	✓ Subject-driven personalization	Pairwise LPIPS between images generated from the same prompt with different seeds (Zhang et al., 2018)
SSIM ↑	✓ Paired image editing	Structural similarity between the edited image and its paired target (Wang et al., 2004)
LLM ↑	✓ Subject-driven personalization ✓ Paired image editing	Score from 0 to 1 assigned by an LLM judge based on task fidelity, prompt adherence, and visual quality