IntLoRA: Integral Low-rank Adaptation of Quantized Diffusion Models

Hang Guo [1] **Yawei Li** [2 * †] **Tao Dai** [3 †] **Shu-Tao Xia** [1 4] **Luca Benini** [2]

Abstract

Fine-tuning pre-trained diffusion models under limited budgets has gained great success. In particular, the recent advances that directly fine-tune the quantized weights using Low-rank Adaptation (LoRA) further reduces training costs. Despite these progress, we point out that existing adaptation recipes are not inference-efficient. Specifically, additional post-training quantization (PTQ) on tuned weights is needed during deployment, which results in noticeable performance drop when the bit-width is low. Based on this observation, we introduce IntLoRA, which adapts quantized diffusion models with integer-type low-rank parameters, to include inference efficiency during tuning. Specifically, IntLoRA enables pre-trained weights to remain quantized during training, facilitating fine-tuning on consumer-level GPUs. During inference, IntLoRA weights can be seamlessly merged into pre-trained weights to directly obtain quantized downstream weights without PTQ. Extensive experiments show our IntLoRA achieves significant speedup on both training and inference without losing performance. Code is available at https://github.com/csguoh/IntLoRA.

1. Introduction

Recently, large-scale text-to-image diffusion models (Rombach et al., 2022; Saharia et al., 2022; Podell et al., 2023) have shown promising capabilities for image generation. Taking advantage of the strong generative prior of pre-trained parameters, a range of downstream adaptation applications have emerged, such as subject-driven generation (Ruiz et al., 2023), style-customized generation (Sohn et al., 2023), and controllable generation (Zhang et al., 2023). However, fully fine-tuning large pre-trained models for downstream tasks poses challenges on consumer-level GPUs. For instance, only loading the `FP32` FLUX.1-dev (BlackForestLabs, 2024) weights into GPUs can consume over 24GB of memory, let alone subsequent fine-tuning. Therefore, the huge fine-tuning costs hinder personalized diffusion model customization.

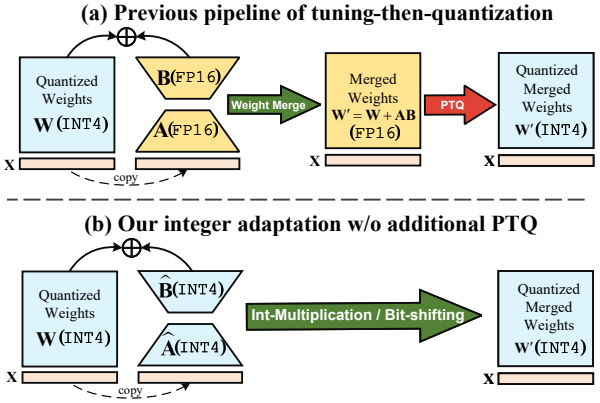

Figure 1: (a) The arithmetic inconsistency between the pre-trained and adaptation weights leads to the merged weights still in `FP16`. Consequently, additional PTQ is needed for low-bit inference. (b) Our IntLoRA allows to work directly on `INT4` arithmetic, ensuring the merged weights seamlessly in `INT4` format and streamlining the whole process.

To facilitate efficient training, recent advances have introduced parameter efficient fine-tuning (PEFT) (Houlsby et al., 2019; Jia et al., 2022) techniques, such as LoRA (Hu et al., 2021), to fine-tune a limited number of parameters. With the reduced gradient and optimizer states, they can achieve comparable or even better adaptation performance than fully fine-tuning. More recently, some works (Dettmers et al., 2024; Qin et al., 2024) have successfully married PEFT and network quantization to allow the low-rank adaptation directly on the quantized weights (as shown in Fig. 1(a)). Through reducing the bit-widths, the GPU costs during fine-tuning are further decreased.

Although the reduced training costs have facilitated user customization, obtaining an inference-aware tuning recipe remains an open challenge. Specifically, existing methods predominantly employ floating-point (*e.g.*, `FP16`) low-rank parameters during training, as a result, it is inevitable to convert the quantized pre-trained weights back to `FP16` for

[1]Tsinghua University [2]ETH Zürich [3]Shenzhen University [4]Peng Cheng Laboratory. [*]Project Lead. [†]Correspondence to: Yawei Li <yawei.li@vision.ee.ethz.ch>, Tao Dai <daitao.edu@gmail.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

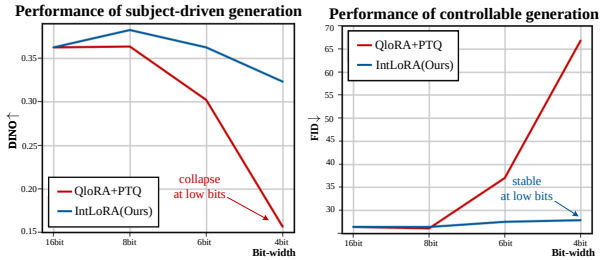

Figure 2: The utilization of PTQ on the downstream merged weights leads to severe performance degradation under low bit-width quantization.

arithmetic consistency to merge low-rank weights into pre-trained weights. During test-time, this pipeline necessitates additional post-training quantization (PTQ) on the `FP16` merged weights for accelerated inference, which is pipeline-complicated and incurs significant performance drop when the bit-width is low (see Fig. 2).

To address these challenges, a potential solution is to also transfer the adaptation weights to integer arithmetic. In this way, all weights during fine-tuning are in integers, thus ensuring the merged weights naturally being quantized. Despite these promising properties, it is non-trivial to accurately quantize the low-rank weights. For example, while zero initializing low-rank weights are advantageous for fine-tuning (Hu et al., 2021), it poses quantization challenges due to substantial quantization errors from small values. Furthermore, the additive form of the original LoRA forces the pre-trained and adaptation weights to share the same quantizer for seamless weight merging, which restricts available parameter space during fine-tuning.

In this work, we propose IntLoRA, which achieves integral low-rank parameters for *both training and inference efficient* diffusion models. In detail, we introduce the Adaptation-Quantization Separation (AQS) technique, which employs a task-agnostic auxiliary matrix to enable quantization-friendly low-rank parameters without disrupting the gradient trajectory of the original LoRA. Additionally, we present the Multiplicative Low-rank Adaptation (MLA), which reformulates the mathematical structure of LoRA from addition to multiplication. This remains mathematically equivalent to the original but allows for independent optimization of adaptation weights. Furthermore, we develop the Variance Matching Control (VMC) to align the pre-trained and auxiliary matrices. For implementation, we provide two versions, *i.e.*, IntLoRA$_{\text{MUL}}$, and IntLoRA$_{\text{SHIFT}}$. The IntLoRA$_{\text{MUL}}$ learns quantized low-rank parameters and can be seamlessly merged through integer multiplication, while IntLoRA$_{\text{SHIFT}}$ introduces log2-quantization and operates by bit-shifting the quantized weights for downstream adaptation. We evaluate our IntLoRA on various diffusion personalization tasks. Extensive experiments show that IntLoRA presents impressive efficiency and performance.

2. Related Work

Parameter-efficient fine-tuning of diffusion models. In order to reduce the fine-tuning cost of large models, parameter-efficient fine-tuning (PEFT) has recently gained great interests (Lian et al., 2022; Chavan et al., 2023; Li & Liang, 2021; He et al., 2021; Jie & Deng, 2023). For example, prompt-based methods (Jia et al., 2022) append learnable prompts to modify the input space. Adapter-based methods (Houlsby et al., 2019; Chen et al., 2022) employ additional bottleneck structures as bypass branches for adaptation. Moreover, LoRA (Hu et al., 2021) adopts low-rank matrices to learn the weight updates for downstream tasks. In this work, we mainly focus on LoRA since it has been widely applied in diffusion model fine-tuning and can be merged into pre-trained weights without increasing inference costs.

Network quantization of diffusion models. Quantization (Nagel et al., 2021) is an effective technique to speed deep-learning models and can be categorized into quantization-aware training (QAT) (Jacob et al., 2018; Li et al., 2024; 2022; Xu et al., 2023a) and post-training quantization (PTQ) (Wang et al., 2023; Nahshan et al., 2021; Li et al., 2021; Wei et al., 2022; Liu et al., 2023; Huang et al., 2024a). In the context of diffusion model quantization, existing works mainly focus on PTQ because of the significant overhead of retraining diffusion models. For example, PTQ4DM (Shang et al., 2023) makes the first attempt to quantize diffusion models to 8 bits. After that, Q-Diffusion (Li et al., 2023) further achieves improved performance and lower bit-width. EfficientDM (He et al., 2023) introduces LoRA to fine-tune the pre-trained model to allow comparable performance with QAT. TFMQ-DM (Huang et al., 2024b) proposes to quantize the time-embedding layer individually for better performance.

Joint PEFT and quantization for efficient fine-tuning. Benefiting from the scaling law, the pre-trained models have become increasingly large, which makes even loading models challenging. To allow fine-tuning on consumer-level GPUs for user customization, some work attempts to apply PEFT techniques directly on the quantized pre-trained weights. Specifically, QLoRA (Dettmers et al., 2024) proposes to quantize the LLMs before fine-tuning the LLMs with LoRA. Despite the reduced GPU usage during training due to the import of only the quantized model, QLoRA does not maintain quantized at inference since the quantized weights need to be converted to `FP16` again so as to be merged with the LoRA weights. QA-LoRA (Xu et al., 2023b) develops a group-wise quantization through sharing parameters across channels but at the cost of impairing the adaptation ability. IR-QLoRA (Qin et al., 2024) analyzes the entropy loss of quantization from an information theory view, but it also needs to convert the quantized weights back to `FP16` during inference.

3. Preliminary

The LoRA (Hu et al., 2021) introduces a low-rank matrix $\Delta\mathbf{W}$ to learn the weight increments for adapting the pre-trained weights $\mathbf{W} \in \mathbb{R}^{C_{out} \times C_{in}}$ to downstream tasks. In implementation, the $\Delta\mathbf{W}$ is formulated as the matrix multiplication of two low-rank matrices $\mathbf{A} \in \mathbb{R}^{C_{out} \times d}$ and $\mathbf{B} \in \mathbb{R}^{d \times C_{in}}$, where the inner dimension d is the pre-defined rank. During fine-tuning, the pre-trained weight $\mathbf{W}$ is frozen and only the low-rank $\mathbf{A}, \mathbf{B}$ are trainable. Since $d \ll \min\{C_{in}, C_{out}\}$, the number of trainable parameters can be very small compared to full fine-tuning, thus reducing the GPU footprint of gradients and optimizer states. The output during downstream fine-tuning is calculated as $\mathbf{y} = \mathbf{W}\mathbf{x} + \lambda \cdot (\mathbf{AB})\mathbf{x}$, where λ is the LoRA scale to adjust the control strength. During inference, the task-specific $\mathbf{AB}$ can naturally be merged into the pre-trained weights, *i.e.*, $\mathbf{W}' = \mathbf{W} + \lambda \cdot \mathbf{AB}$, without increasing additional costs.

Even though the LoRA can alleviate training costs through reduced gradients and optimizer states, it still needs to load huge `FP16` pre-trained weights. Given the increasing pre-trained model size, it becomes impractical to only use LoRA to fine-tune the diffusion models on consumer-level GPUs. To further reduce the training memory, recent advancements (Dettmers et al., 2024; Xu et al., 2023b; Qin et al., 2024) have introduced network quantization to allow direct fine-tuning on the integer weights. Formally, given a tensor $\mathbf{X}$, the target bit-width b, the quantization process can be defined as:

$$\hat{\mathbf{X}} = s \cdot (\text{clip}(\lfloor \frac{\mathbf{X}}{s} \rceil + z, 0, 2^b - 1) - z) \triangleq s \cdot (\mathbf{X}_{\text{round}} - z), \tag{1}$$

where $\lfloor \cdot \rceil$ is the round function, $s = \frac{\max(\mathbf{X}) - \min(\mathbf{X})}{2^b - 1}$ is the scaling factor, and $z = -\lfloor \frac{\min(\mathbf{X})}{s} \rceil$ is the zero-point.

Despite current methods allow user to train customized models under a low memory budget, they all require additional PTQ on the fine-tuned weights for fast inference, which leads to noticeable performance degradation when the quantization bit-width is low.

4. Methodology

In this work, we aim to remove the additional PTQ of the merged weights by introducing integer-type low-rank parameters during fine-tuning. In this way, both the $\mathbf{AB}$ and $\mathbf{W}$ are in the same arithmetic type, thus ensuring the merged $\mathbf{W}'$ is naturally already quantized. However, several technical challenges arise when transferring LoRA to integer arithmetic. First, the $\mathbf{AB}$ in the original LoRA is zero-initialized to ensure the behavior of the model is similar to the pre-trained one at the beginning of training. Although helpful for fine-tuning, this initialization complicates the quantization process. For instance, the all-zero distribution requires a separately designed quantizer at the beginning of tuning, since the scaling factor $s = 0$ leads to an infinite $\frac{\mathbf{X}}{s}$ in Eq. (1). Second, the vanilla LoRA merges the `FP16` $\mathbf{AB}$ and $\mathbf{W}$ using addition. When both $\mathbf{AB}$ and $\mathbf{W}$ are quantized, it is essential to ensure that they share identical quantization parameters to enable PTQ-free weight merging. This requirement leads to constrained parameter space, thus limiting the adaptation ability.

4.1. Integral Low-rank Adaptation

To address the above challenges, we propose IntLoRA to operate adaptation on the integer arithmetic. The overall pipeline is shown in Fig. 3.

Adaptation-quantization separation. The vanilla LoRA adopts zero initialization on the adaptation parameter $\mathbf{AB}$. Although this strategy can improve performance, the all-zero distribution is not quantization-friendly. To allow accurate quantization while maintaining the correct gradient, we propose the Adaptation-Quantization-Separation (AQS) mechanism. The key observation is that *the adaptation requires gradients from zero-initialized weights while the quantization does not*. Therefore, we can split the adaptation weights into the gradient-enabled zero part and the gradient-free nonzero part. Formally, let $\mathbf{R}$ be the auxiliary matrix to serve as the nonzero part, $\mathcal{Q}$ be the quant-dequant operator, then our AQS can be formulated as:

$$\mathbf{W}' = \mathcal{Q}[\mathbf{W} - \text{sg}(\mathbf{R})] + \text{sg}(\mathbf{R}) + \mathbf{AB}, \tag{2}$$

where $\text{sg}(\cdot)$ denotes the stop gradient operation. Thanks to the AQS, the $\mathbf{AB}$ can be zero-initialized for the same gradient as the original LoRA, while $\text{sg}(\mathbf{R}) + \mathbf{AB}$ facilitate subsequent quantization by specifically designing the auxiliary matrix $\mathbf{R}$ as discussed in Sec. 5.4. In the following part, we will ignore the $\text{sg}(\cdot)$ notation for clarity.

Multiplicative low-rank adaptation. The vanilla LoRA employs additive form $\mathbf{W} + \mathbf{AB}$ for weight merge. However, it is difficult to seamlessly fuse the quantized $\hat{\mathbf{W}}$ and $\hat{\mathbf{AB}}$ when they are quantized by independent quantizers. To this end, we propose Multiplicative Low-rank Adaptation (MLA) to rewrite the form of the original LoRA into a quantization-friendly multiplication form. Specifically, denote the quant-dequant results as $\mathcal{Q}(\mathbf{W} - \mathbf{R}) = s \cdot (\mathbf{W}_{\text{round}} - z)$, then the MLA can be derived as follows:

$$\begin{aligned} \mathbf{W}' &= \mathcal{Q}(\mathbf{W} - \mathbf{R}) + \mathbf{R} + \mathbf{AB} \\ &= s \cdot (\mathbf{W}_{\text{round}} - z) + \mathbf{R} + \mathbf{AB} \\ &= [s \cdot \mathbf{I} + \frac{1}{\mathbf{W}_{\text{round}} - z} \odot (\mathbf{R} + \mathbf{AB})] \odot (\mathbf{W}_{\text{round}} - z), \end{aligned} \tag{3}$$

where the task-specific adaptation term is trainable and will be quantized, and the pre-trained term is already in integer type and is shared across tasks. $\mathbf{I}$ is an all-one matrix. The

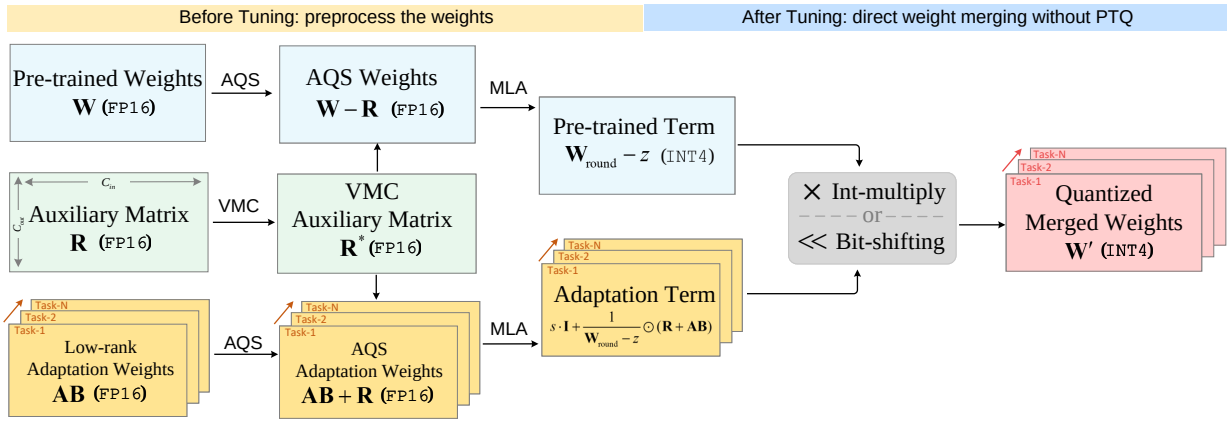

Figure 3: **Before tuning**, we propose the Adaptation Quantization Separation (AQS) to incorporate auxiliary matrix into pre-trained weights and low-rank weights for zero-initialized but quantization-friendly distribution. Then, the Multiplicative Low-rank Adaptation (MLA) is used to reformulate additive LoRA into the product of the "pre-training term" and the "adaptation term". At last, we introduce the Variance Matching Control (VMC) to adjust the distribution of the adaptation term by modulating the auxiliary matrix. **After tuning**, we use hardware-friendly integer multiplication or bit shifting to directly generate quantized merged weights without additional PTQ. The detailed algorithm is given in Appendix A.

operator $\odot$ denotes the Hadamard product of two matrices. The proposed MLA is mathematically equivalent to its additive counterpart, while is more quantization-friendly since it avoids the shared quantizer of pre-trained and adaptation weights. It is noteworthy that the adaptation term is still in FP16 at this step, and we will detail its quantization strategies in Sec. 4.2.

Variance matching control. One opportunity brought from the multiplicative form in Eq. (3) is that we can apply the log2-quantization on the adaptation term, thus allowing more efficient bit-shifting on the pre-trained term. However, log2-quantization is notoriously more difficult than common uniform quantization (Nagel et al., 2021) and requires appropriate distribution properties, *e.g.*, most values concentrated around zero to allow for the utilization of as many quantization bins as possible on the logarithmic scale. Here, we revisit the adaptation term in Eq. (3) aiming to find useful mathematical insights. Given the $\mathbf{AB}$ is orders of magnitude smaller than $\mathbf{R}$ (the justification is shown in Appendix F), we approximate the adaptation term in Eq. (3) by removing $\mathbf{AB}$ from it, namely,

$$\begin{aligned} s\cdot\mathbf{I}+\frac{\mathbf{R}}{\mathbf{W}_{\text{round}}-z} &= s\cdot\mathbf{I}+\frac{s\cdot\mathbf{R}}{s\cdot(\mathbf{W}_{\text{round}}-z)} \\ &\approx s\cdot\mathbf{I}+\frac{s\cdot\mathbf{R}}{\mathbf{W}-\mathbf{R}}=\frac{s\cdot\mathbf{W}}{\mathbf{W}-\mathbf{R}}. \end{aligned} \tag{4}$$

From this derivation, it follows that the auxiliary matrix $\mathbf{R}$ is crucial for controlling the distribution shape of the adaptation term. Unfortunately, we find there exists a dilemma in choosing an appropriate distribution for $\mathbf{R}$. On one hand, it is desirable for the values in $\mathbf{R}$ to be larger. Formally, let $\sigma_{\mathbf{R}}$ be the variance of the element in $\mathbf{R}$ which is a random variable, it can be derived the expectation of the adaptation term converges to zero when $\sigma_{\mathbf{R}}$ approaches infinity, namely,

$$\mathbb{E}\left[\lim_{\sigma_{\mathbf{R}}\to\infty} s\cdot\mathbf{I}+\frac{\mathbf{R}}{\mathbf{W}-\mathbf{R}}\right]=\mathbb{E}\left[\lim_{\sigma_{\mathbf{R}}\to\infty}\frac{s\cdot\mathbf{W}}{\mathbf{W}-\mathbf{R}}\right]=0. \tag{5}$$

On the other hand, setting $\sigma_{\mathbf{R}}$ too large can also lead the $\mathcal{Q}(\mathbf{W}-\mathbf{R})$ uncorrelated to the original $\mathbf{W}$, *i.e*, namely,

$$\lim_{\sigma_{\mathbf{R}}\to\infty}\rho(\mathcal{Q}(\mathbf{W}-\mathbf{R}),\mathbf{W})=\lim_{\sigma_R\to\infty}\frac{\sigma_{\mathbf{W}}}{\sqrt{\sigma_{\mathbf{W}}^2+\sigma_{\mathbf{R}}^2}}=0, \tag{6}$$

where the $\rho(\cdot,\cdot)$ denotes the correlation coefficient. Eq. (6) indicates that a over-large $\sigma_{\mathbf{R}}$ makes it difficult to reconstruct the original signal $\mathbf{W}$ through dequantizing the $\mathcal{Q}(\mathbf{W}-\mathbf{R})$ due to the low correlation. In short, it is important to choose an appropriate $\sigma_{\mathbf{R}}$ to strike a balance between quantization difficulty and information retention. We also give the visualization of this choice dilemma in Fig. 8. To this end, we propose the Variance Matching Control (VMC) mechanism. Specifically, we first multiply $\mathbf{R}$ by the variance ratio $r=\frac{\sigma_{\mathbf{W}}}{\sigma_{\mathbf{R}}}\in\mathbb{R}^{C_{out}}$ for rough alignment from $\mathbf{R}$ to the scale of $\mathbf{W}$. After that, we introduce a scalar α as an exponent of r, *i.e.*, r^α, to fine-grain the search for the optimal $\mathbf{R}^*$. As a result, the variance-matched auxiliary matrix can be denoted as $\mathbf{R}^*=r^\alpha\cdot\mathbf{R}$, and we can use this to obtain the distribution suitable for log2-quantization. Since r^α can be shared across tasks, it is only of negligible cost. In addition to the $\sigma_{\mathbf{R}}$, we observe the distribution shape of $\mathbf{R}$ also has an effect on performance, and we give a detailed discussion in Sec. 5.4. It should be noted that the $\mathbf{R}$ can be online generated during fine-tuning using the distribution statistics and fixed random seed, thus avoiding the need to store its FP16 parameters.

Table 1: Quantitative comparison on subject-driven generation tasks. The notion "WxAy" represents the bit-widths of weights "W" and activations "A". The best results are **bolded**.

methods	nbits	DINO↑	CLIP-I↑	CLIP-T↑	LPIPS↓
LoRA (Hu et al., 2021)	W16A16	0.4828	0.6968	0.2954	0.8076
QLoRA (Dettmers et al., 2024)	W8A8	0.4153	0.6661	0.2824	0.8088
QA-LoRA (Xu et al., 2023b)	W8A8	0.4156	0.6664	0.2834	0.8086
IR-QLoRA (Qin et al., 2024)	W8A8	0.4070	0.6630	0.2841	0.8110
IntLoRA$_{\text{SHIFT}}$ (Ours)	W8A8	0.4353	0.6842	0.2841	0.8257
IntLoRA$_{\text{MUL}}$ (Ours)	W8A8	**0.4498**	**0.6882**	**0.2858**	**0.8062**
QLoRA (Dettmers et al., 2024)	W4A8	0.2136	0.6134	0.2510	0.8201
QA-LoRA (Xu et al., 2023b)	W4A8	0.4127	0.6897	0.2700	0.8281
IR-QLoRA (Qin et al., 2024)	W4A8	0.3722	0.6719	0.2707	0.8186
IntLoRA$_{\text{SHIFT}}$ (Ours)	W4A8	0.4039	0.6716	0.2709	0.8187
IntLoRA$_{\text{MUL}}$ (Ours)	W4A8	**0.4242**	**0.6913**	**0.2710**	**0.8181**

4.2. Implementation of IntLoRA

Benefiting from the quantization-friendly weight distribution, we can implement our IntLoRA with two variants according to different quantizers on the adaptation term. The first variant employs the uniform affine quantizer on the adaptation term, thus enabling weight merge through integer-type multiplication. The second variant introduces the more hardware-friendly log2 quantizer to achieve downstream adaptation by bit-shifting the quantized pre-trained weights. More details are given below.

Integer multiplication form. We employ uniform affine quantization on the adaptation term, with the scaling factor and zero-point denoted as $\bar{s}$ and $\bar{z}$, and the quantized results as $\mathbf{U}_{\text{round}}$, then our IntLoRA$_{\text{MUL}}$ can be formalized as:

$$\mathbf{W}' = \bar{s} \cdot (\mathbf{U}_{\text{round}} - \bar{z}) \odot (\mathbf{W}_{\text{round}} - z). \tag{7}$$

Bit-shifting form. Denote the adaptation term in Eq. (3) as $\mathbf{V}$ for clarity, we first compute the bit shift value as follows:

$$\text{shift} = \text{clip}(\lfloor -\log_2 |\mathbf{V}| \rceil, 0, 2^b - 1). \tag{8}$$

Then the weight adaptation with IntLoRA$_{\text{SHIFT}}$ can be represented as:

$$\begin{aligned} \mathbf{W}' &= \text{sign}(\mathbf{V}) \odot 2^{-\text{shift}} \odot (\mathbf{W}_{\text{round}} - z) \\ &= \text{sign}(\mathbf{V}) \odot [(\mathbf{W}_{\text{round}} - z) \gg \text{shift}] \\ &= \frac{1}{2^N} \odot \text{sign}(\mathbf{V}) \odot [(\mathbf{W}_{\text{round}} - z) \ll (N - \text{shift})], \end{aligned} \tag{9}$$

where $\text{sign}(\mathbf{V}) \in \{-1, 1\}$ and $N = 2^b - 1$. Since the direct right-shifting on $\mathbf{W}_{\text{round}} - z$ may lead to truncation error, we thus use $N - \text{shift}$ with a scaling factor $\frac{1}{2^N}$ to equivalently convert to the left-shifting for error reduction.

5. Experiments

5.1. Experimental Setup

Datasets. We evaluate on multiple adaptation tasks, including subject-driven generation (Ruiz et al., 2023), controllable generation (Zhang et al., 2023), and style-customized image generation (Sohn et al., 2023). For the subject-driven generation, we use a subset which contains 15 text-subject pairs from the Dreambooth (Ruiz et al., 2023) dataset, for fast training and evaluation. For controllable generation, we consider three sub-tasks, *i.e*, Segmentation map to Image (S2I) on ADE20K dataset (Zhou et al., 2017), Landmark to Face (L2F) on CelebA-HQ dataset (Karras, 2017), and the Canny edge to Image (C2I) on the COCO dataset (Lin et al., 2014). For the style-customized generation, we employ the StyleDrop (Sohn et al., 2023) dataset, which includes 18 style images, and we use 6 text prompts for each style to generate images with style similar to the style image and content aligned with the text prompt.

Implementation details. We employ the StableDiffusionV1.5 (Rombach et al., 2022) as the pre-trained backbone for subject-driven generation and controllable generation. We further employ larger SDXL (Podell et al., 2023) as the pre-trained model in the style-customized generation. We use uniform quantization (Nagel et al., 2021) to quantize the weights per-channel and activations per-tensor. Since previous methods mainly focus on efficient training, with the tuned weights still in `FP16`, to make a fair comparison, we apply additional PTQ on the merged weights for efficient inference. As for the training of the quantized adaptation term, we use the Straight Through Estimator (STE) to allow back-propagation. For the proposed variance matching control, we employ the ratio of the maxima of the sampled distributions as a fast estimator for the variance. Due to the page limit, we provide more details in Appendix C.

Figure 4: Qualitative comparison on subject-driven generation tasks. More results are provided in Appendix H.

5.2. Main Results

Subject driven generation. Tab. 1 gives the results of weight-activation quantization on subject-driven generation task. It can be seen that the proposed method consistently outperforms other competitors under different bit-widths. For instance, the IntLoRA$_{\text{MUL}}$ suppresses the IR-QLoRA by even 0.0428 DINO score on the W8A8 setup. Notably, the QLoRA and IR-QLoRA baselines, which use additional PTQ on the merged weights, suffer a significant performance drop under the W4A8 setup. In contrast, even the challenging log2-quantization of our IntLoRA$_{\text{SHIFT}}$ works well under W4A8. We also give qualitative visualization in Fig. 4, where one can see that our IntLoRA can facilitate subject-faithful and photo-realistic image generation.

Controllable image generation. The results of controllable image generation are shown in Tab. 2. One can see that our IntLoRA continues to outperform existing strong baselines, *e.g.*, our IntLoRA$_{\text{MUL}}$ outperforms the IR-QLoRA by 4.96 FID on the 4-bit S2I setting. And it can be seen that QLoRA struggles to produce meaningful results at low 4 bit-width. We also give a qualitative comparison in Fig. 5, and it can be seen that the images generated by the IntLoRA-tuned model are well-matched with the control signals.

Style customized generation. The results of the style customized generation task are shown in Fig. 6. It can be seen that our IntLoRA achieves a favorable balance between style images and text prompts, whereas some existing approaches fail. For instance, in the third row, both the QALoRA and IR-QLoRA methods directly copy the original style image under the text prompt "The letter 'G' in [V] style".

Table 2: Quantitative comparison of FID↓ score on controllable image generation.

methods	8-bitwidth			4-bitwidth		
	S2I	L2F	C2I	S2I	L2F	C2I
LoRA(FP16)	31.39	37.50	16.05	31.39	37.50	16.05
QLoRA	31.09	38.88	15.34	71.75	117.37	62.49
QALoRA	31.32	38.88	15.34	31.51	43.09	16.73
IR-QLoRA	31.81	36.30	15.70	35.83	39.63	18.30
IntLoRA$_{\text{SHIFT}}$	31.38	**34.46**	15.76	32.85	35.06	17.65
IntLoRA$_{\text{MUL}}$	**31.08**	37.52	**15.26**	**30.87**	**33.62**	**16.32**

5.3. Efficiency Comparison

We compare the training and inference efficiency of our IntLoRA against other baselines in Tab. 3. As for training, both IntLoRA and QLoRA only need to load quantized pre-trained weights, which is more memory efficient compared to the vanilla LoRA fine-tuning. As a result, our IntLoRA and QLoRA have similar training speeds and memory costs. However, for the inference phase, our IntLoRA can naturally obtain the quantized merged weights without additional PTQ, thus streamlining the adaptation pipeline and avoiding potential performance degradation under low bit-width. In short, compared with existing methods which only focus on training efficiency, our IntLoRA presents a both training and inference efficient paradigm.

5.4. Ablation Studies

Ablation on the smoothing factor. As discussed in Sec. 4.1, there is a dilemma in choosing an appropriate $\sigma_{\mathbf{R}}$. For example, setting it too large can lead to information loss

Table 3: Comparison of training and inference efficiency with other methods. We fine-tuning the StableDiffusionV1.5 model on the Dreambooth task. The training speed is tested on one NVIDIA RTX 3090 GPU.

method	nbits	Training Stage		Inference Stage		
		training speed	model size	PTQ	CLIP-I↑	CLIP-T↑
LoRA (Hu et al., 2021)	W32A32	0.68s/img	7700MB	✔	0.6968	0.2954
QLoRA (Dettmers et al., 2024)	W8A8	0.85s/img	1925MB	✔	0.6661	0.2824
IntLoRA$_{\text{SHIFT}}$ (Ours)	W8A8	0.84s/img	1925MB	✘	0.6842	0.2841
IntLoRA$_{\text{MUL}}$ (Ours)	W8A8	0.87s/img	1925MB	✘	0.6882	0.2858
QLoRA (Dettmers et al., 2024)	W4A8	0.85s/img	963.1MB	✔	0.6134	0.2510
IntLoRA$_{\text{SHIFT}}$ (Ours)	W4A8	0.84s/img	963.1MB	✘	0.6716	0.2709
IntLoRA$_{\text{MUL}}$ (Ours)	W4A8	0.87s/img	963.1MB	✘	0.6913	0.2710

Figure 5: Qualitative comparison on controllable generation tasks. More results are provided in Appendix H.

of the original weights, while a too-small one results in a large quantization error. To this end, we introduce r^{α} in the proposed VMC as the hyperparameter to search for the task-oriented variance. We give the downstream task performance with varying α in Fig. 7. It can be seen that setting $\sigma_{\mathbf{R}}$ slightly smaller than $\sigma_{\mathbf{W}}$ can obtain better performance, indicating that the information loss has a greater impact than the quantization error. In the implementation, we chose a moderate smoothing factor $\alpha = 1.5$ for the trade-off.

Distribution selection for auxiliary matrix. In this work, the auxiliary matrix $\mathbf{R}$ plays a crucial role in both AQS and VMC. Therefore, the distribution shape of $\mathbf{R}$ can potentially influence the performance. To this end, we investigate different distribution shapes of $\mathbf{R}$ through ablation experiments and give the results in Tab. 4. It can be seen that the Laplace distribution performs better than other options on most metrics. This is because a light-tailed distribution, such as Laplace, clusters most samples around zero, which facilitates smaller errors for log2-quantization. Therefore, the light-tailed distribution performs empirically better than its heavy-tailed counterparts.

Table 4: Ablation on different distribution shape choices of the auxiliary matrix.

settings	DINO↑	CLIP-I↑	CLIP-T↑	LPIPS↓
Guassian	0.4135	0.6756	0.2492	0.8179
Cauchy	0.1367	0.5617	0.1870	0.8067
StudentT	0.2935	0.6420	0.2487	0.8048
Laplace	0.4492	0.6980	0.2588	0.8110

6. Further Discussion

Results on NLP tasks. In addition to fine-tuning diffusion models for image generation, we further validate the effectiveness of the NLP tasks. Specifically, we fine-tune the Llama3-8B model (Dubey et al., 2024) and use the MetaMathQA dataset (Yu et al., 2023) for training and GSM8K dataset (Cobbe et al., 2021) for testing. The comparison

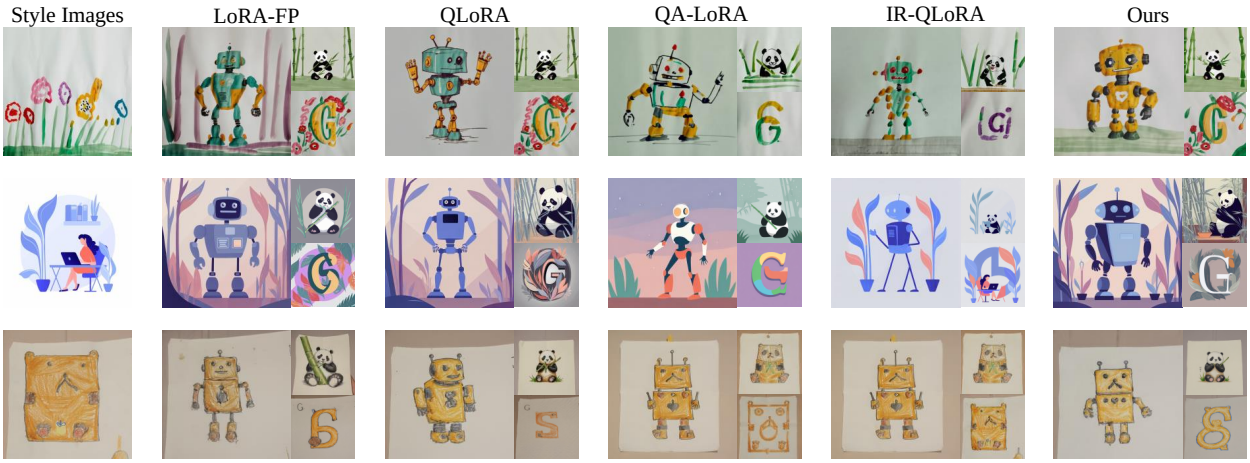

Figure 6: Qualitative comparison on style-customized generation. The text prompt is "A friendly robot in [V] style", "A panda eating bamboo in [V] style", and "The letter 'G' in [V] style", respectively. "Ours" denotes the IntLoRA$_{\text{SHIFT}}$. More results are provided in Appendix H.

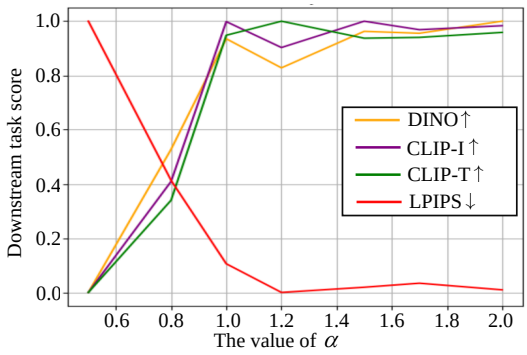

Figure 7: The normalized performance under different α.

results are shown in Tab. 5. It can be seen that our IntLoRA maintains stable performance when transferring to natural language. For example, IntLoRA$_{\text{MUL}}$ outperforms QLoRA by 0.17% QA accuracy under 8-bit quantization, demonstrating the generalization of our IntLoRA.

Difference from EfficientDM. EfficientDM (He et al., 2023) employs QAT-like LoRA fine-tuning on the `FP16` diffusion weights for network quantization. Despite it is inference-efficient as it can directly produce quantized merged weights, we would like to point out that it is not training-efficient. Specifically, EfficientDM requires load `FP16` pre-trained weights at the training stage, which is unacceptable for fine-tuning large-size models on consumer-level GPUs. By contrast, our IntLoRA is both training and inference efficient. Moreover, we also explore our IntLoRA on the diffusion quantization task. The results are shown in Tab. 6. It can be seen that our IntLoRA$_{\text{MUL}}$ achieves even better performance than the EfficientDM. It should be noted that our IntLoRA only needs to load the quantized weights during calibration instead of the floating-point weights in EfficientDM, thus reducing the training memory cost.

Table 5: Comparison on the natural language task of mathematical answering. More qualitative results in Appendix H.

Methods	LoRA	QLoRA	QA-LoRA	Ours$_{\text{SHIFT}}$	Ours$_{\text{MUL}}$
nbits	W16A16	W8A8	W8A8	W8A8	W8A8
accuracy	64.24%	64.06%	63.53%	64.10%	**64.23%**

Table 6: Comparison with EfficientDM on W4A4 diffusion model quantization. We evaluate on the ImageNet 256×256 image generation, and train with ddim_step=20 on LDM-4 model with 500 training epochs.

methods	IS↑	FID↓	sFID↓	precision↑
EfficientDM	178.20	13.42	26.67	0.70
Ours$_{\text{SHIFT}}$	116.50	20.20	26.79	0.63
Ours$_{\text{MUL}}$	**199.20**	**10.43**	**24.02**	**0.79**

7. Conclusion

We propose IntLoRA, which employs integer-type low-rank parameters, to remove the additional PTQ on the merged weights. Specifically, we introduce the quantization-adaptation separation to allow the coexistence of zero-initialized gradient and quantization-friendly distribution. We further develop the multiplicative low-rank adaptation to achieve a decoupled quantizer of pre-trained and adaptation weights, accompanied by the variance matching control to adjust the variance for accurate adaptation control. Benefiting from these elegant designs, we provide two variants of IntLoRA, which either use int-multiplication or bit-shifting to adapt the quantized pre-trained models. Through transferring the adaptation weights to the integer arithmetic, our IntLoRA demonstrates its effectiveness across different pre-trained models and various downstream tasks, while exhibiting impressive both training and inference efficiency.

Acknowledges

This work is supported in part by the National Natural Science Foundation of China, under Grant (62302309, 62171248), Shenzhen Science and Technology Program (JCYJ20220818101014030, JCYJ20220818101012025).

Impact Statement

This work aims to achieve efficient fine-tuning of quantized diffusion models, reducing both training and inference costs without sacrificing performance. It has no ethical concerns and can lower the resource barrier for model customization, thus enabling broader access to generative AI.

References

BlackForestLabs. FLUX. https://github.com/black-forest-labs/flux/, 2024.

Chavan, A., Liu, Z., Gupta, D., Xing, E., and Shen, Z. One-for-All: Generalized LoRA for parameter-efficient fine-tuning. *arXiv preprint arXiv:2306.07967*, 2023.

Chen, S., Ge, C., Tong, Z., Wang, J., Song, Y., Wang, J., and Luo, P. AdaptFormer: Adapting vision transformers for scalable visual recognition. In *NeurIPS*, volume 35, pp. 16664–16678, 2022.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. QLoRA: Efficient finetuning of quantized llms. In *NeurIPS*, volume 36, 2024.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., and Neubig, G. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.

He, Y., Liu, J., Wu, W., Zhou, H., and Zhuang, B. EfficientDM: Efficient quantization-aware fine-tuning of low-bit diffusion models. *arXiv preprint arXiv:2310.03270*, 2023.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In *ICML*, pp. 2790–2799. PMLR, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Huang, W., Qin, H., Liu, Y., Li, Y., Liu, X., Benini, L., Magno, M., and Qi, X. SliM-LLM: Salience-driven mixed-precision quantization for large language models. *arXiv preprint arXiv:2405.14917*, 2024a.

Huang, Y., Gong, R., Liu, J., Chen, T., and Liu, X. TFMQ-DM: Temporal feature maintenance quantization for diffusion models. In *CVPR*, pp. 7362–7371, 2024b.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, pp. 2704–2713, 2018.

Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S., Hariharan, B., and Lim, S.-N. Visual prompt tuning. In *ECCV*, pp. 709–727. Springer, 2022.

Jie, S. and Deng, Z.-H. FacT: Factor-tuning for lightweight adaptation on vision transformer. In *AAAI*, pp. 1060–1068, 2023.

Karras, T. Progressive growing of GANs for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Li, X., Liu, Y., Lian, L., Yang, H., Dong, Z., Kang, D., Zhang, S., and Keutzer, K. Q-Diffusion: Quantizing diffusion models. In *ICCV*, pp. 17535–17545, 2023.

Li, X. L. and Liang, P. Prefix-Tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Li, Y., Gong, R., Tan, X., Yang, Y., Hu, P., Zhang, Q., Yu, F., Wang, W., and Gu, S. BRECQ: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021.

Li, Y., Xu, S., Zhang, B., Cao, X., Gao, P., and Guo, G. Q-ViT: Accurate and fully quantized low-bit vision transformer. In *NeurIPS*, volume 35, pp. 34451–34463, 2022.

Li, Y., Xu, S., Cao, X., Sun, X., and Zhang, B. Q-DM: An efficient low-bit quantized diffusion model. In *NeurIPS*, volume 36, 2024.

Lian, D., Zhou, D., Feng, J., and Wang, X. Scaling & shifting your features: A new baseline for efficient model tuning. In *NeurIPS*, volume 35, pp. 109–123, 2022.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: Common objects in context. In *ECCV*, pp. 740–755. Springer, 2014.

Liu, J., Niu, L., Yuan, Z., Yang, D., Wang, X., and Liu, W. PD-Quant: Post-training quantization based on prediction difference metric. In *CVPR*, pp. 24427–24437, 2023.

Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., Van Baalen, M., and Blankevoort, T. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

Nahshan, Y., Chmiel, B., Baskin, C., Zheltonozhskii, E., Banner, R., Bronstein, A. M., and Mendelson, A. Loss aware post-training quantization. *Machine Learning*, 110 (11):3245–3262, 2021.

Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., and Rombach, R. SDXL: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

Qin, H., Ma, X., Zheng, X., Li, X., Zhang, Y., Liu, S., Luo, J., Liu, X., and Magno, M. Accurate LoRA-finetuning quantization of LLMs via information retention. *arXiv preprint arXiv:2402.05445*, 2024.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *CVPR*, pp. 10684–10695, 2022.

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, pp. 22500–22510, 2023.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, volume 35, pp. 36479–36494, 2022.

Shang, Y., Yuan, Z., Xie, B., Wu, B., and Yan, Y. Post-training quantization on diffusion models. In *CVPR*, pp. 1972–1981, 2023.

Sohn, K., Ruiz, N., Lee, K., Chin, D. C., Blok, I., Chang, H., Barber, J., Jiang, L., Entis, G., Li, Y., et al. Styledrop: Text-to-image generation in any style. *arXiv preprint arXiv:2306.00983*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Wang, C., Wang, Z., Xu, X., Tang, Y., Zhou, J., and Lu, J. Towards accurate data-free quantization for diffusion models. *arXiv preprint arXiv:2305.18723*, 2(5), 2023.

Wei, X., Gong, R., Li, Y., Liu, X., and Yu, F. QDrop: Randomly dropping quantization for extremely low-bit post-training quantization. *arXiv preprint arXiv:2203.05740*, 2022.

Xu, S., Li, Y., Lin, M., Gao, P., Guo, G., Lü, J., and Zhang, B. Q-DETR: An efficient low-bit quantized detection transformer. In *CVPR*, pp. 3842–3851, 2023a.

Xu, Y., Xie, L., Gu, X., Chen, X., Chang, H., Zhang, H., Chen, Z., Zhang, X., and Tian, Q. QA-LoRA: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717*, 2023b.

Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. MetaMath: bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Zhang, L., Rao, A., and Agrawala, M. Adding conditional control to text-to-image diffusion models. In *ICCV*, pp. 3836–3847, 2023.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. Scene parsing through ADE20K dataset. In *CVPR*, pp. 633–641, 2017.

A. Summery of IntLoRA Algorithm

Before tuning, we pre-process the pre-trained weights in Algo. 1, followed by the forward process of IntLoRA$_{\text{MUL}}$ and IntLoRA$_{\text{SHIFT}}$ in Algo. 2 and Algo. 3, respectively.

Algorithm 1 The weight pre-process of the linear layer in IntLoRA

Input: Pre-trained wight $\mathbf{W} \in \mathbb{R}^{C_{out} \times C_{in}}$, auxiliary matrix $\mathbf{R} \in \mathbb{R}^{C_{out} \times C_{in}}$, smooth factor $\alpha \in \mathbb{R}$
Output: Quantitzed pre-trained weights $\mathbf{W}_{\text{round}}$, scaling factor s_{round}, zero point z_{round}
sigma_R $\leftarrow$ variance estimation of $\mathbf{R}$
sigma_W $\leftarrow$ variance estimation of $\mathbf{W}$
r = (sigma_W / sigma_R)$^{\alpha}$
$\mathbf{R}_{\text{star}}$ = r * $\mathbf{R}$
$\mathbf{W}_{\text{process}} = \mathbf{W} - \mathbf{R}_{\text{star}}$
$\mathbf{W}_{\text{round}}, s_{\text{round}}, z_{\text{round}} \leftarrow$ uniform_quantizer($\mathbf{W}_{\text{process}}$)

Algorithm 2 The forward process of the linear layer in IntLoRA$_{\text{MUL}}$

Input: Pre-processed quantized weights $\mathbf{W}_{\text{round}}$, scaling factor s_{round}, zero point z_{round}, auxiliary matrix $\mathbf{R}_{\text{star}} \in \mathbb{R}^{C_{out} \times C_{in}}$, LoRA parameters $\mathbf{A} \in \mathbb{R}^{C_{out} \times d}$, $\mathbf{B} \in \mathbb{R}^{C_{out} \times d}$, input tensor $\mathbf{x} \in \mathbb{R}^{C_{in} \times L}$
Output: Output tensor $\mathbf{y} \in \mathbb{R}^{C_{out} \times L}$
$\mathbf{W}_{\text{adapt}} = s_{\text{round}} \cdot \mathbf{I} + \frac{1}{\mathbf{W}_{\text{round}} - z_{\text{round}}} \odot (\mathbf{R}_{\text{star}} + \mathbf{AB})$
$\mathbf{W}^{\text{INT}}_{\text{adapt}}, s_{\text{adapt}}, z_{\text{adapt}} \leftarrow$ uniform_quantizer($\mathbf{W}_{\text{adapt}}$)
$\mathbf{x}^{\text{INT}}, s_{\text{x}} \leftarrow$ act_quantizer($\mathbf{x}$)
$\mathbf{W}_{\text{merge}} = (\mathbf{W}^{\text{INT}}_{\text{adapt}} - z_{\text{adapt}}) \odot (\mathbf{W}_{\text{round}} - z_{\text{round}})$
$\mathbf{y} = s_{\text{x}} s_{\text{round}} \mathbf{W}_{\text{merge}} \mathbf{x}^{\text{INT}}$

Algorithm 3 The forward process of the linear layer in IntLoRA$_{\text{SHIFT}}$

Input: Pre-processed quantized weights $\mathbf{W}_{\text{round}}$, scaling factor s_{round}, zero point z_{round}, auxiliary matrix $\mathbf{R}_{\text{star}} \in \mathbb{R}^{C_{out} \times C_{in}}$, LoRA parameters $\mathbf{A} \in \mathbb{R}^{C_{out} \times d}$, $\mathbf{B} \in \mathbb{R}^{C_{out} \times d}$, input tensor $\mathbf{x} \in \mathbb{R}^{C_{in} \times L}$, desired bit-width b, pre-defined max bit-width number $N = 32$
Output: Output tensor $\mathbf{y} \in \mathbb{R}^{C_{out} \times L}$
$\mathbf{W}_{\text{adapt}} = s_{\text{round}} \cdot \mathbf{I} + \frac{1}{\mathbf{W}_{\text{round}} - z_{\text{round}}} \odot (\mathbf{R}_{\text{star}} + \mathbf{AB})$
shift = clip($\lfloor -\log_2 |\mathbf{W}_{\text{adapt}}| \rceil, 0, 2^b - 1$)
$\mathbf{W}_{\text{merge}} = \frac{1}{2^N} \odot \text{sign}(\mathbf{W}_{\text{adapt}}) \odot [(\mathbf{W}_{\text{round}} - z) \ll (N - \text{shift})]$
$\mathbf{x}^{\text{INT}}, s_{\text{x}} \leftarrow$ act_quantizer($\mathbf{x}$)
$\mathbf{y} = s_{\text{x}} \mathbf{W}_{\text{merge}} \mathbf{x}^{\text{INT}}$

B. Distribution Visualization of $\sigma_{\mathbf{R}}$

In Sec. 4.2, we have theoretically pointed out that there is a choice dilemma for $\sigma_{\mathbf{R}}$. Here we elaborate on its effect through distribution visualization. Specifically, we remove the VMC and use a scaling scalar to generate a too-large or too-small auxiliary variance, followed by the log2 quantization on the adaptation term. The results are shown in Fig. 8. On the one hand, setting $\sigma_{\mathbf{R}}$ too large can lead to a low correlation $\rho(\mathbf{W}, \mathbf{W} - \mathbf{R})$, which makes it hard to reconstruct $\mathbf{W}$ from $\mathbf{W} - \mathbf{R}$ using estimator $\mathbf{W} \approx \mathcal{Q}(\mathbf{W} - \mathbf{R}) + \mathbf{R}$. On the other hand, a too small $\sigma_{\mathbf{R}}$ prevents the expectation of adaptation term converging to zero, causing few log bins to be used. In experiments, we find that the training of both settings fails to converge. By contrast, the proposed VMC can precisely control $\sigma_{\mathbf{R}}$ to allow most values of the adaptation term to be zero-neighbored, facilitating more challenging log2 quantization. Moreover, it should be noted that the too-small $\sigma_{\mathbf{R}}$ can also be regarded as an approximation of direct quantization on the zero-initialized $\mathbf{AB}$, and thus the experimental results also justify the AQS for zero-initialized $\mathbf{AB}$.

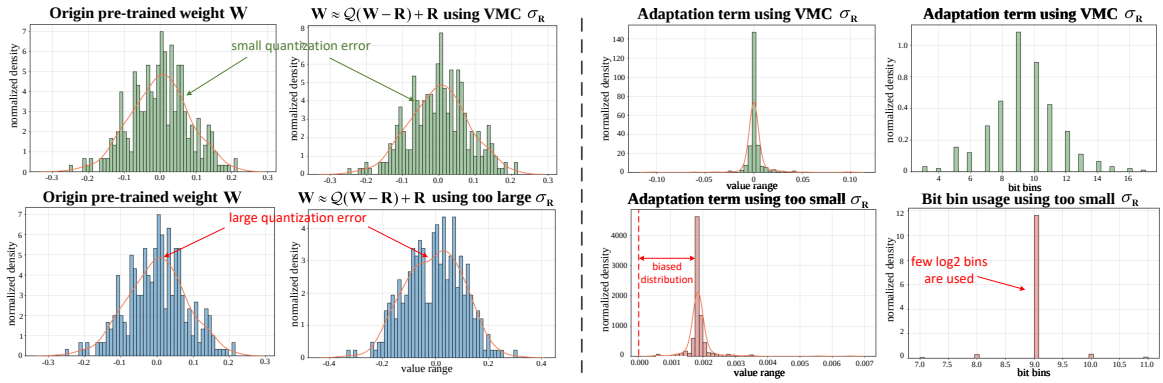

Figure 8: The distribution visualization using Kernel Density Estimate (KDE) on different weight tensors. **Left**: the KDE plot of pre-trained weights and estimated weights under different $\sigma_{\mathbf{R}}$. **Right**: the KDE plot of the adaptation term and the log2 bins usage with different $\sigma_{\mathbf{R}}$.

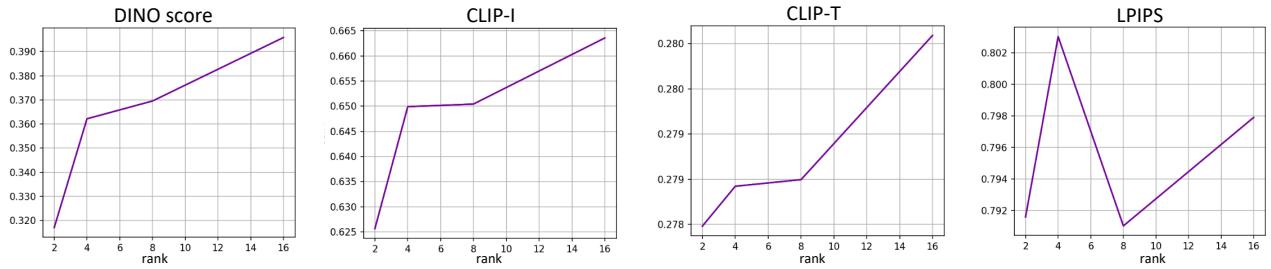

Figure 9: Ablation experiments of different LoRA ranks.

C. More Implementation Details

For the subject-driven generation, we use the AdamW optimizer with a weight decay of 1e-2 and fine-tune the query, key, value, and output projection layer. The learning rate is set to 6e-5. The batch size is set to 1, and the number of training steps is 400. The rank of the LoRA is set to 4. We adopt the prior preservation strategy as Dreambooth (Ruiz et al., 2023) to generate 200 class images. For the controllable generation, we fine-tune the model for 11 epochs for Canny-to-Image tasks and 20 epochs for Landmark-to-Face and Segmentation-to-Image tasks. The learning rate is set to 1e-5 using the AdamW optimizer. The LoRA rank is set to 4. The batch size is set to 8 and the image resolution is 512×512 for all three tasks. For the style-customized generation, we fine-tune the pre-trained model using the AdamW optimizer with a learning rate of 5e-5. Since it involves a larger SDXL, we chose a relatively large LoRA rank of 64 for all compared methods, since there is only one style image as well as the larger pre-trained parameters. We fine-tune for 500 steps with batch size 1. Similar to StyleDrop (Sohn et al., 2023), we only use one image as the style image and find it works well. The style images and text prompts for evaluation are given in Appendix H. The variance ratio in the variance matching control is surrogated as the value range ratio, *i.e.*, $r = \frac{\max\{|\max(\mathbf{W})|, |\min(\mathbf{W})|\}}{\min\{|\max(\mathbf{R})|, |\min(\mathbf{R})|\}}$. We append the trainable low-rank parameters on the Query, Key, Value, and Out projection, in the attention layers (Vaswani et al., 2017) and keep all other layers frozen and quantized. The rank of LoRA is set to 4 for subject-driven generation and controllable generation, and 64 for style-customized generation.

D. Additional Ablation Experiments

Ablation on the LoRA rank. The low-rank d in LoRA is a trade-off between performance and efficiency. A larger rank improves the adaptation ability by training more parameters but comes with larger training and storage costs, and vice versa. Here, we give the impact of different rank setups on performance in Fig. 9. One can see that the performance generally improves as we increase the rank, but the rate of growth varies. For instance, the increased speed from rank=4 to rank=8 increases inferior to the one from rank=2 to rank=4. Moreover, increasing the rank to 16 can generally obtain better results than its lower counterpart. In practice, considering the trade-off between performance and efficiency, we select a moderate rate rank=4.

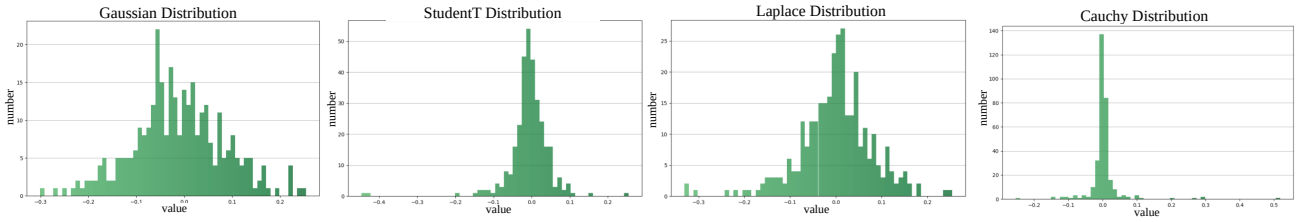

Figure 11: The shape of different distributions for initialing the auxiliary matrix.

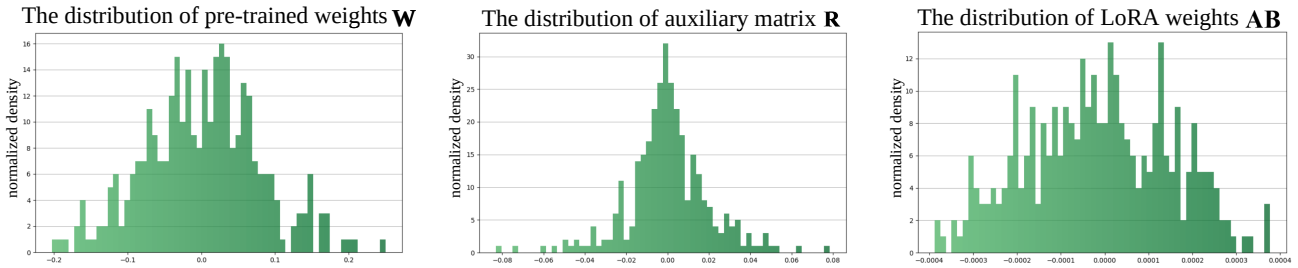

Figure 12: The distribution visualization of the original weights $\mathbf{W}$, the auxiliary matrix $\mathbf{R}$, and the learned low-rank weights $\mathbf{AB}$.

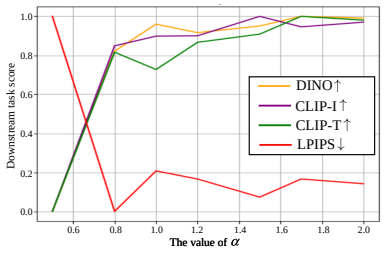

Figure 10: The effects of VMC for IntLoRA$_{\text{MUL}}$.

The effects of variance matching control for IntLoRA$_{\text{MUL}}$. In this work, we propose the variance matching control to adjust the variance of $\mathbf{R}$, so that allows the log2-quantization of the adaptation term to obtain the IntLoRA$_{\text{SHIFT}}$. In other words, the VMC is primarily introduced for IntLoRA$_{\text{SHIFT}}$. Despite we also apply the VMC to IntLoRA$_{\text{MUL}}$, given the IntLoRA$_{\text{MUL}}$ does not require such strict constraints on the distribution shape of the adaptation term, it is interesting to investigate the influence of variance matching control on the performance of IntLoRA$_{\text{MUL}}$. To this end, we adjust the smoothing factor α to adjust the strength of the VMC, *e.g.*, setting α to zero can lead to the removal of the VMC. The results of the IntLoRA$_{\text{MUL}}$ under different VMC scales are shown in Fig. 10. As one can see, despite the VMC being initially proposed for the log2-quantization, the well-structured distribution also facilitates uniform quantization. For example, when we set the α approaching zero, *i.e.*, the VMC is close to being removed, and the performance of IntLoRA$_{\text{MUL}}$ appears similar pattern as the IntLoRA$_{\text{SHIFT}}$, which suffers a significant performance drop. Moreover, the performance gains gradually converge when the $\alpha > 1.5$. In short, the VMC can not only allow the log2-quantization to work but also improve the performance of the uniform quantization.

Distribution shape for auxiliary matrix. In Sec. 5.4, we provided different symmetric distributions including Gaussian, StudentT, Laplace, and Cauchy. Fig. 11 gives the results of sampling from different distributions. The Laplace distribution possesses light tails, and the shape of the distribution is convex, *i.e.*, $f''(x) > 0, x \neq 0$. This unique property makes it easy to control the value of the adaptation term to produce distributions that are friendly to log2 quantization, *i.e.*, most samples are clustered around the zero to use as many bins as possible. This analysis is also verified by the experiments in Tab. 4, which shows that the Laplace distribution achieves the best performance.

E. Impacts from the Auxiliary Matrix

In Eq. (2) of the proposed AQS, we introduce an additional auxiliary matrix $\mathbf{R}$ to the original pre-trained weight $\mathbf{W}_0$ to achieve adaptation-quantization separation. However, this extra $\mathbf{R}$ potentially introduces outliers and thus causes quantization error for $\mathbf{W}$. Here, we point out that since the proposed VMC can control the range of $\mathbf{R}$ through the variance scaling factor $r = \sigma_{\mathbf{W}}/\sigma_{\mathbf{R}}$, the introduction of $\mathbf{R}$ in the AQS is ensured not result in additional outliers. For validation, we also give the distribution visualization of the original $\mathbf{W}$ and the VMC re-scaled $\mathbf{R}$ in Fig. 12. It can be seen that the range of $\mathbf{R}$ is effectively controlled within the range of $\mathbf{W}$, thus effectively avoiding the detrimental effect of additional outliers.

F. Justification for the Value Orders

A key assumption in the derivation for VMC is that the learned values of low-rank parameters $\mathbf{AB}$ are orders of magnitude smaller than the auxiliary matrix $\mathbf{R}$. Based on this assumption we ignore $\mathbf{AB}$ as an approximation. Here, we give the specific evidence for this approximation. Specifically, we visualize the weights of the trained $\mathbf{AB}$ and the distribution of $\mathbf{R}$, as shown in Fig. 12. It can be seen the range of $\mathbf{AB}$ is constrained to $[-0.0004, 0.0004]$, while the range of $\mathbf{R}$ is $[-0.08, 008]$. Therefore, the experimental visualization above confirms the soundness of our approximation. Since the $\mathbf{AB}$ in LoRA is zero-initialized, it tends to be distributed around zero with learned small values aiming to not disturb the pre-training weights too much.

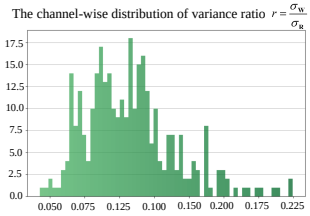

Figure 13: The value distribution of the channel-wise variance ratio r

G. Limitation and Future Works

Although the proposed IntLoRA can effectively improve the efficiency of diffusion model fine-tuning by allowing the adaptation parameters also on the integer arithmetic, the proposed framework can be further improved in the following aspects. First, although the trainable low-rank weights are quantized with STE, these quantized weights are still in `FP16` type during tuning to enable accurate gradient updates. Therefore, it is promising to specifically design integer-type propagation. Despite this seems challenging, it can further reduce the training cost and accelerate the adaptation process. Second, although we introduce a feasible way that uses hyperparameter search of the smoothing factor α to find a compatible $\sigma_{\mathbf{R}}$ as well as the appropriate distribution shape of $\mathbf{R}$, it can be more elegant if we can use advanced mathematical analysis techniques, such as functional analysis, to find the statistical properties a suitable $\mathbf{R}$ should satisfy. Third, this work mainly focuses on the efficient acceleration of LoRA due to its prevalence among the PEFT techniques. Applications to other PEFT methods for hardware-efficient adaptation could be interesting future work.

H. Additional Visualization Results

- Fig. 14 gives more samples on the subject-driven generation tasks.
- In Fig. 15, we give more samples of the segmentation-to-image tasks.
- In Fig. 16, we give more samples of the face landmark-to-face image tasks.
- In Fig. 17, we give more samples of the canny edge-to-image tasks.
- Fig. 18 provides more samples of the results of the style-accustomed generation.
- In Fig. 19, we give the style images and the text prompts used for evaluation on the style customized generation tasks.
- In Appendix H.1, we give some case studies of the mathematical question-answering task using the fine-tuned Llama3-8B model.

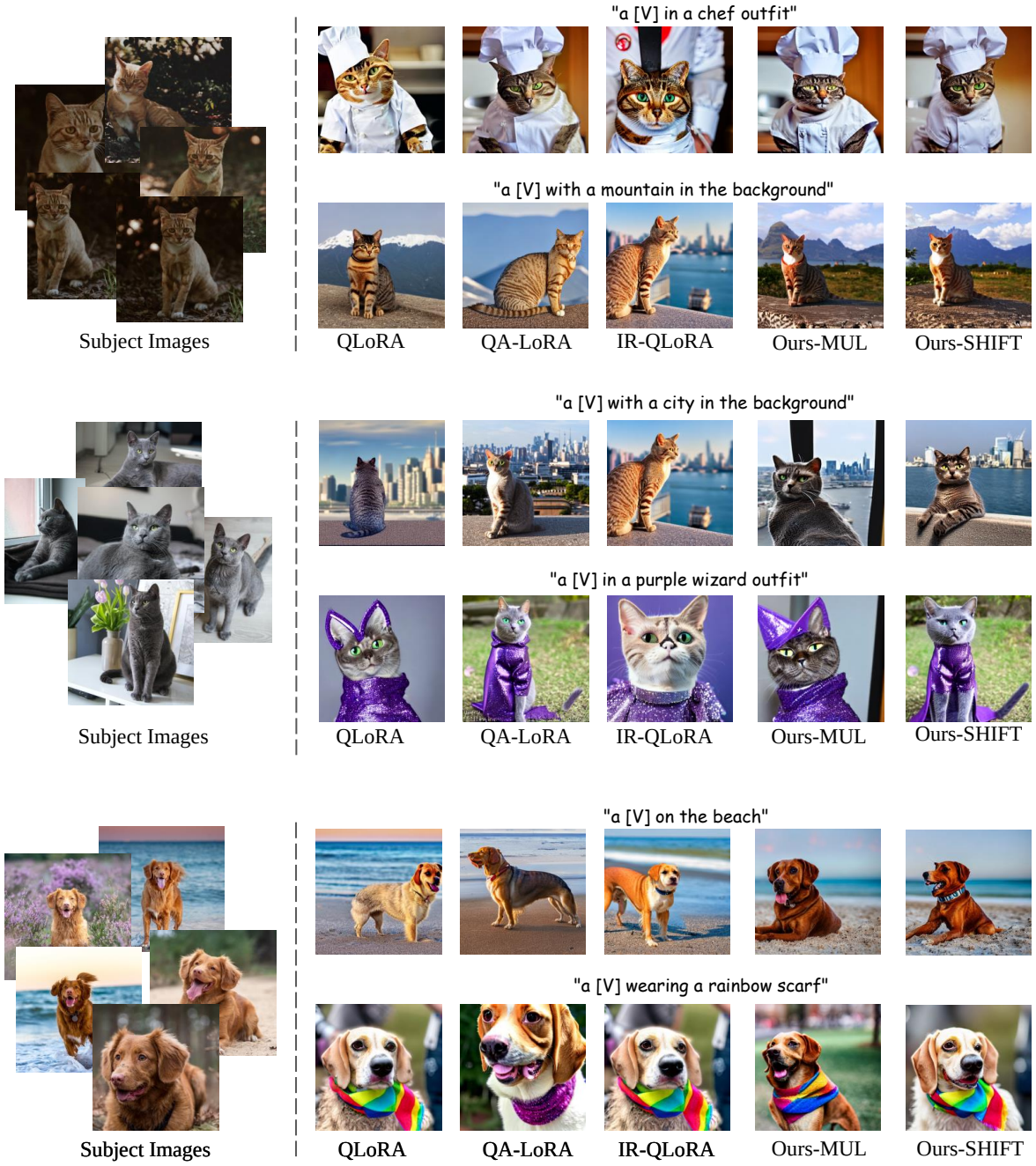

Figure 14: More qualitative comparison results on subject-driven generation.

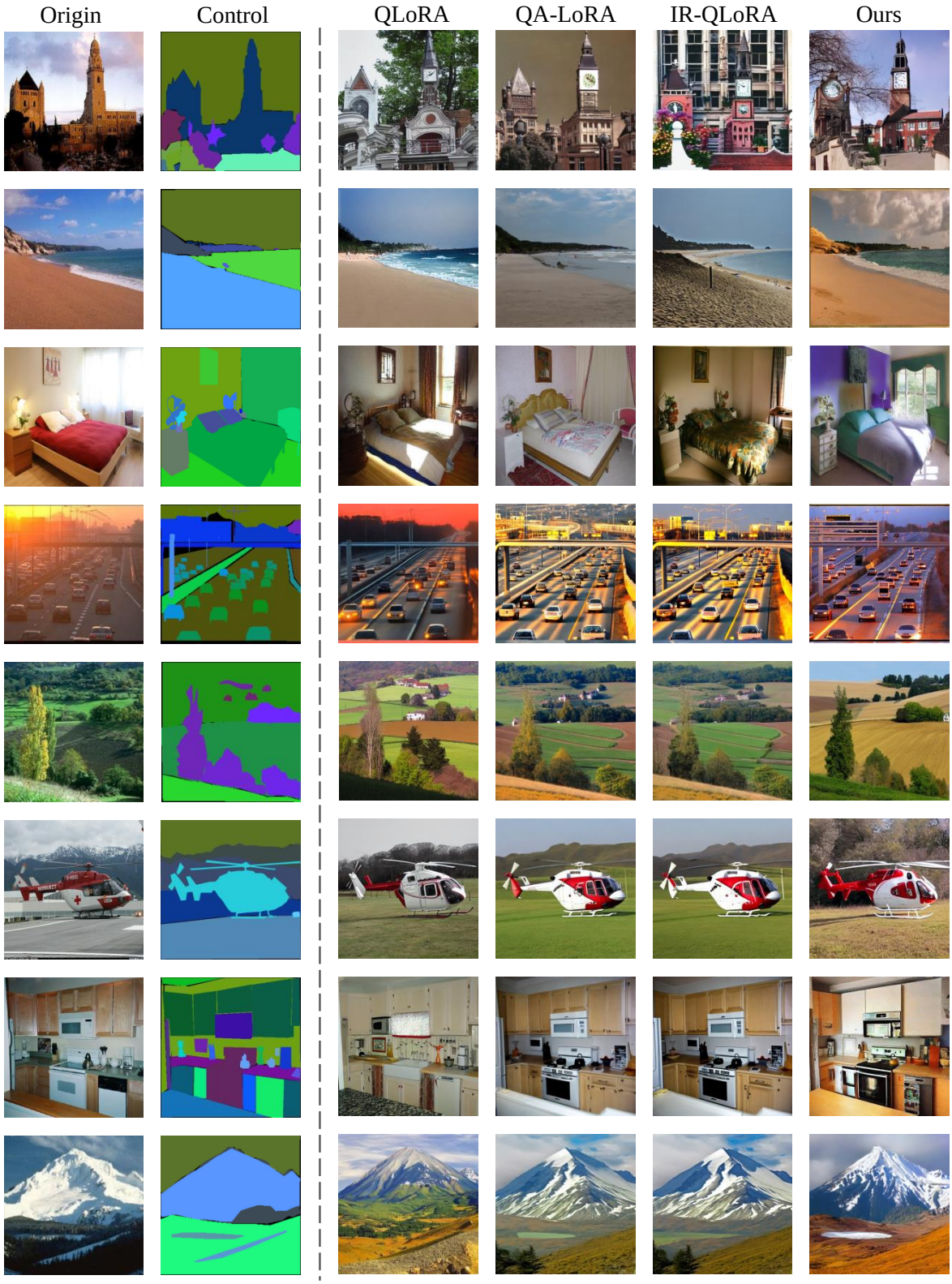

Figure 15: More qualitative comparison results on segmentation to image task. The 'Ours' denotes the IntLoRA$_{\text{SHIFT}}$. Zoom in for better effects.

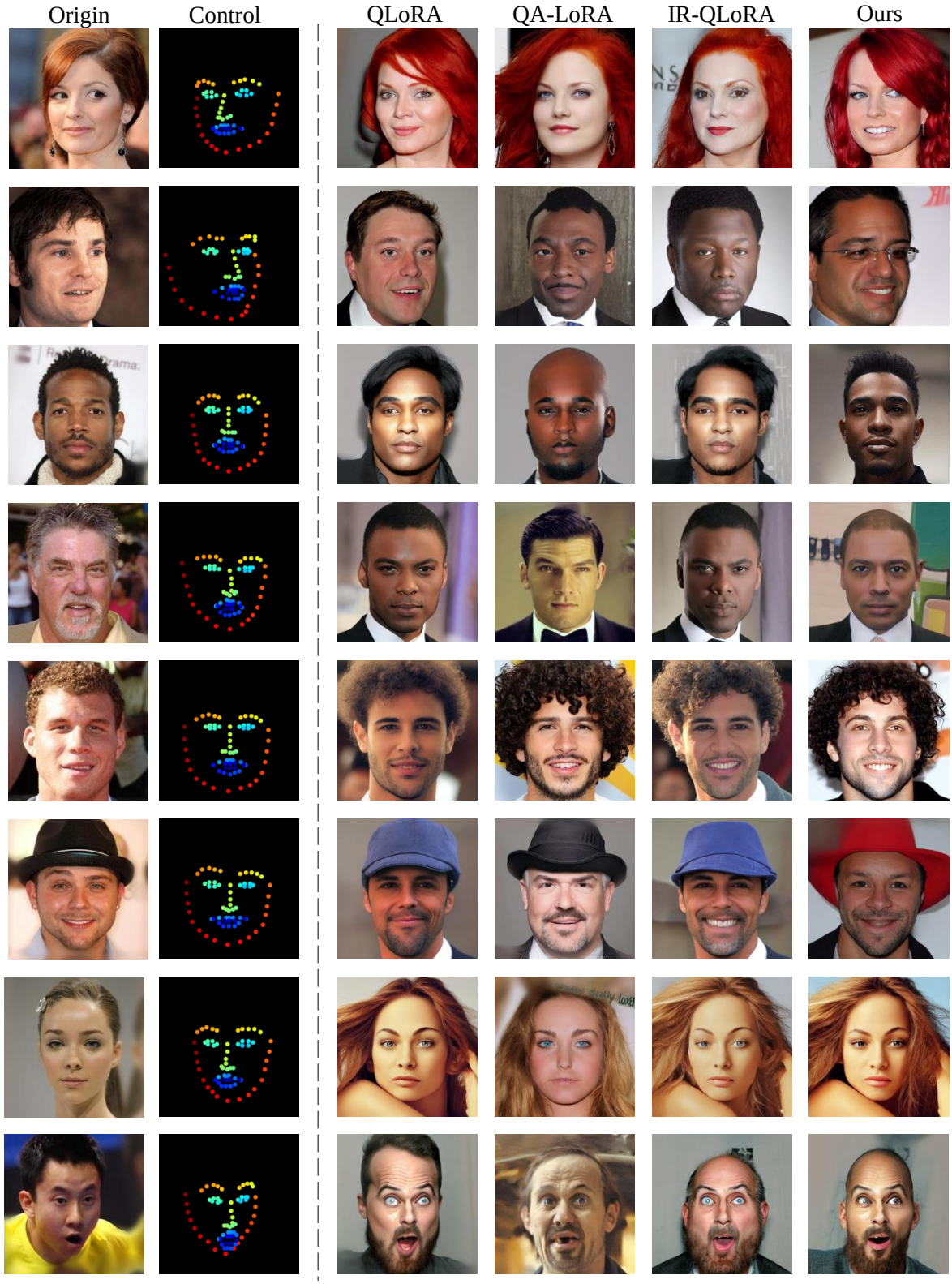

Figure 16: More qualitative comparison results on landmark to face task. The 'Ours' denotes the IntLoRA$_{\text{SHIFT}}$. Zoom in for better effects.

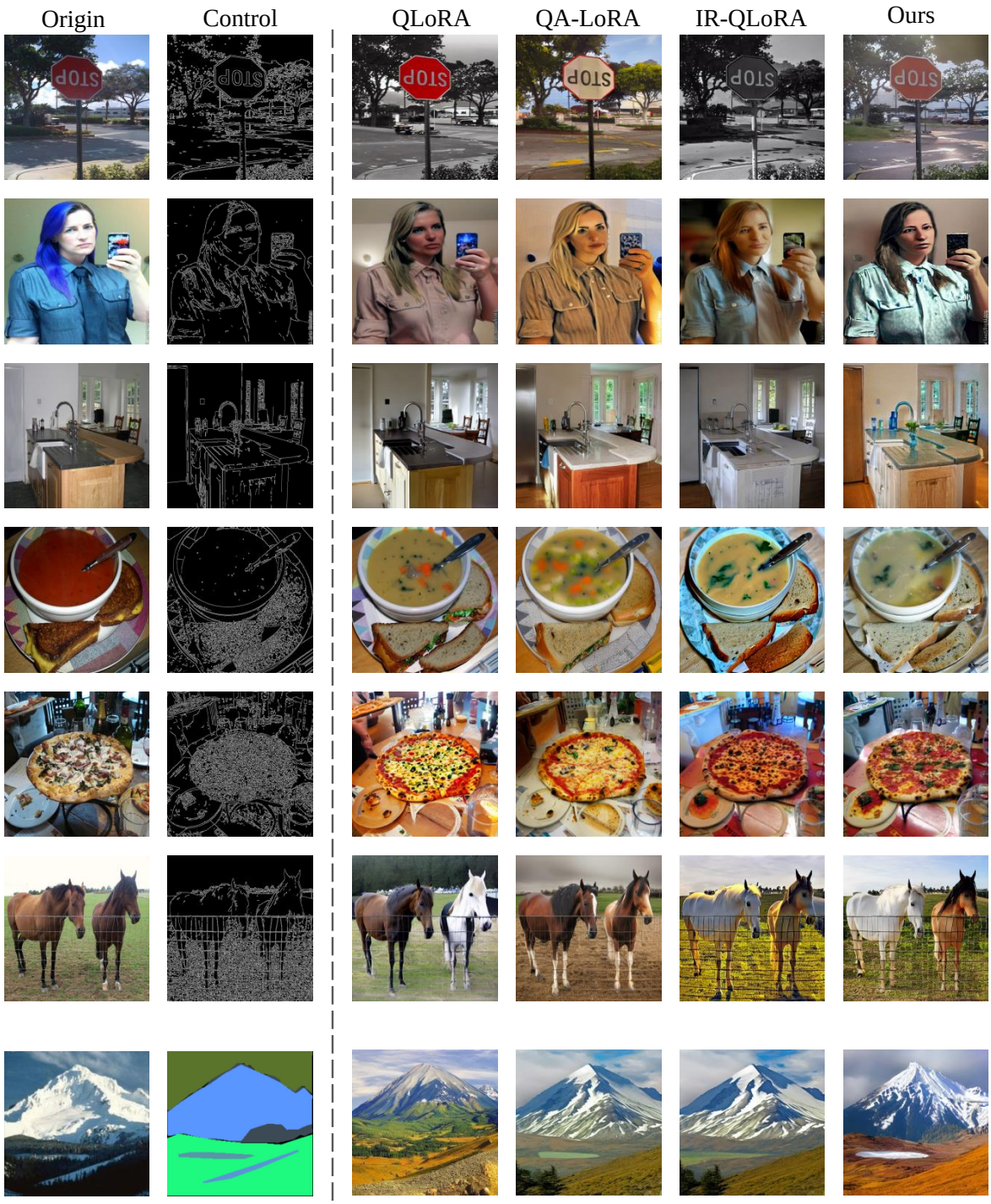

Figure 17: More qualitative comparison results on canny to image task. The 'Ours' denotes the IntLoRA$_{\text{SHIFT}}$. Zoom in for better effects.

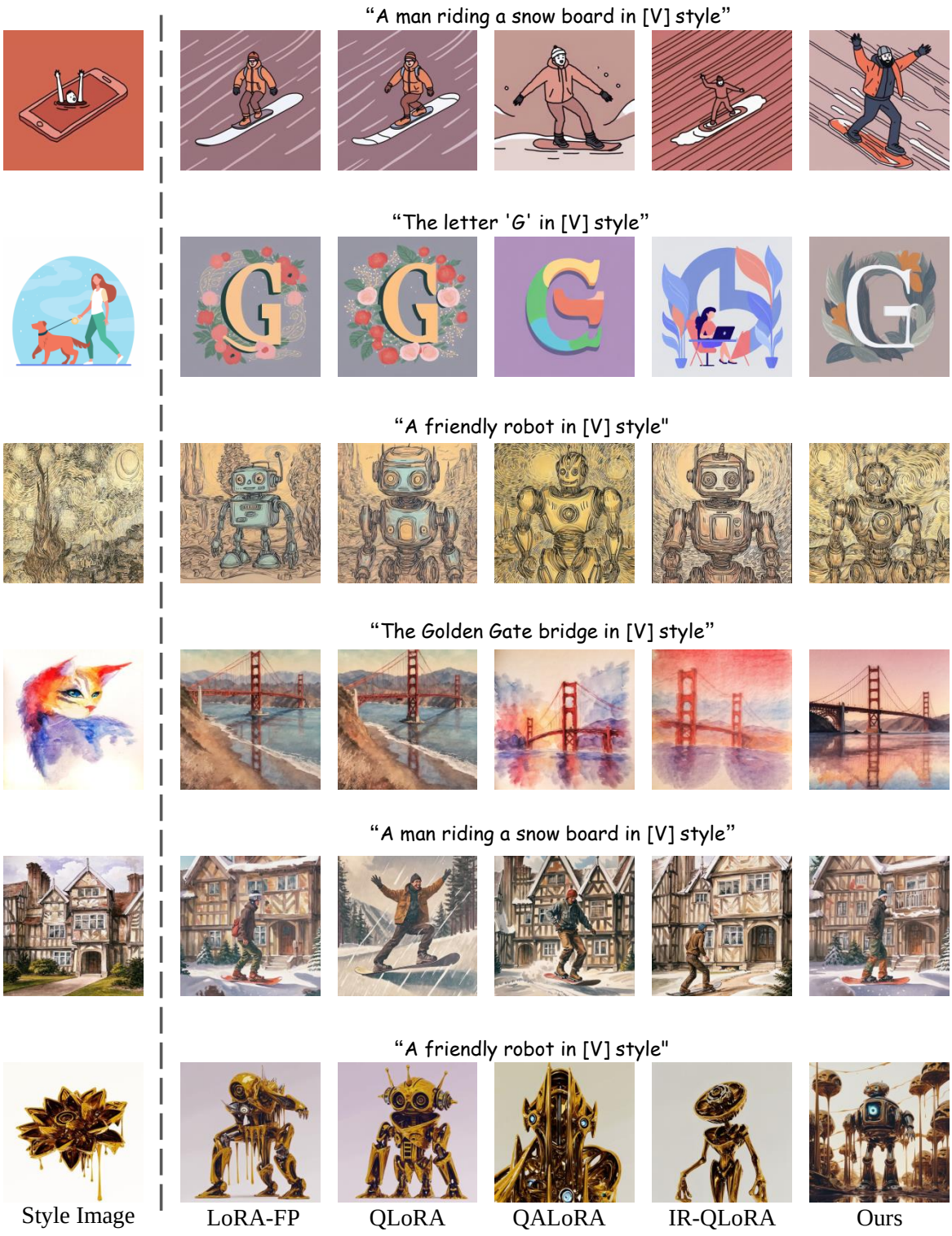

Figure 18: More qualitative comparison results on style-accustomed generation. The ‘Ours’ denotes the IntLoRA$_{\text{SHIFT}}$. Zoom in for better effects.

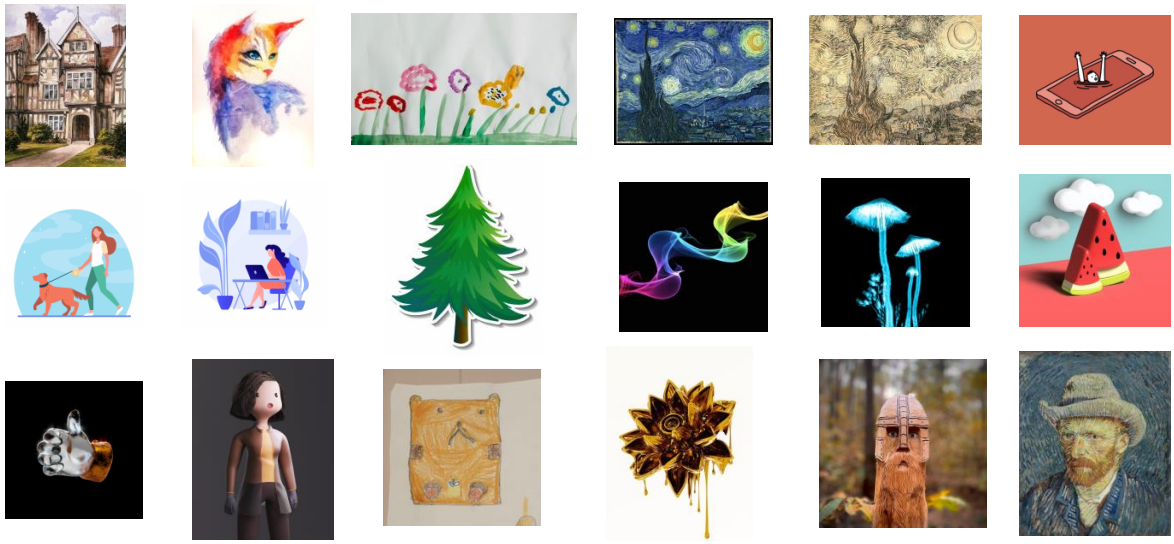

Figure 19: The style images used for training on the style-customized generation task. The text prompts are “A friendly robot in [V] style”, “A fluffy baby sloth with a knitted hat trying to figure out a laptop in [V] style”, “A panda eating bamboo in [V] style”, “The Golden Gate bridge in [V] style”, “A man riding a snowboard in [V] style”, “The letter ‘G’ in [V] style”.

H.1. Cases of mathematical question answering.

Case Study 1

Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market? **(Ground-truth answer is 18)**

LoRA-FP Answer: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning, so she has 16 - 3 = 13 eggs left. She bakes muffins for her friends every day with four eggs, so she has 13 - 4 = 9 eggs left. She sells the remainder at the farmers' market daily for $2 per fresh duck egg, so she makes 9 * $2 = $18. Therefore, Janet makes $18 every day at the farmers' market. The answer is 18.

QLoRA Answer: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning, so she has 16 - 3 = 13 eggs left. She bakes muffins for her friends every day with four eggs, so she has 13 - 4 = 9 eggs left. She sells the remainder at the farmers' market daily for $2 per fresh duck egg, so she makes 9 * $2 = $18. Therefore, Janet makes $18 every day at the farmers' market. The answer is: 18

QA-LoRA Answer: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning, so she has 16 - 3 = 13 eggs left. She bakes muffins for her friends every day with four, so she has 13 - 4 = 9 eggs left. She sells the remainder at the farmers' market daily for $2 per fresh duck egg, so she makes 9 * $2 = $18. Therefore, Janet makes $18 every day at the farmers' market. The answer is: 18

QA-LoRA Answer: Janet lays 16 eggs per day. She eats three for breakfast, so she has 16 - 3 = 13 eggs left. She bakes muffins with four eggs, so she has 13 - 4 = 9 eggs left. She sells the remainder at the farmers' market for $2 per egg, so she makes 9 * $2 = $18. Therefore, Janet makes $18 every day at the farmers' market. The answer is: 18

Our Answer: Janet lays 16 eggs per day. She eats three for breakfast, so she has 16 - 3 = 13 eggs left. She bakes muffins with four eggs, so she has 13 - 4 = 9 eggs left. She sells the remainder at the farmers' market for $2 per fresh duck egg, so she makes 9 * $2 = $18. Therefore, Janet makes $18 every day at the farmers' market. The answer is: 18.

Case Study 2

Question: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take? **(Ground-truth answer is 3)**

LoRA-FP Answer: We know that the robe takes 2 bolts of blue fiber. To find the amount of white fiber, we can use the ratio of blue fiber to white fiber, which is 2:1. So, the amount of white fiber is half of the blue fiber, which is 2/2 = 1 bolt. Therefore, the total number of bolts needed is 2 + 1 = 3 bolts. The answer is 3.

QLoRA Answer: The robe takes 2 bolts of blue fiber. To find the amount of white fiber, we can use the ratio of 1:2. Since the ratio is 1:2, we can set up the equation 1/2 * 2 = x, where x represents the amount of white fiber. Solving for x, we get x = 1. Therefore, the robe takes 2 bolts of blue fiber and 1 bolt of white fiber. In total, the robe takes 2 + 1 = 3 bolts of fiber. The answer is: 3

QA-LoRA Answer:The robe takes 2 bolts of blue fiber. The robe takes half that much white fiber, which is 2/2 = 1 bolt of white fiber. So, in total, the robe takes 2 bolts of blue fiber + 1 bolt of white fiber = 3 bolts of fiber. Therefore, the robe takes 3 bolts of fiber in total. The answer is: 3

IR-QLoRA Answer: The robe takes 2 bolts of blue fiber. The robe takes half that much white fiber, which is 2/2 = 1 bolt of white fiber. So, the robe takes 2 bolts of blue fiber + 1 bolt of white fiber = 3 bolts in total. The answer is: 3

Our Answer: If a robe takes 2 bolts of blue fiber, then it takes 2/2 = 1 bolt of white fiber. So, the total number of bolts needed is 2 bolts of blue fiber + 1 bolt of white fiber = 3 bolts in total. The answer is: 3.