QSPAN: TRAINING-AWARE LOW-RANK QUANTIZATION ERROR COMPENSATION FOR FULLY QUANTIZED LORA FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-efficient fine-tuning (PEFT) reduces adaptation costs, and quantized PEFT further improves efficiency by primarily quantizing the frozen backbone weights. However, most methods retain activations, gradients, and the LoRA branch in high precision. Fully quantizing both the backbone and LoRA branch reduces these remaining costs but substantially degrades adaptation, as the frozen backbone cannot correct quantization errors and rank-constrained adapters have limited compensation capacity. Existing methods prioritize weight-quantization errors without considering their effects on the low-precision forward computation and training objective. We introduce **QSpan**, a training-aware framework that determines which quantization-error directions each adapter should span and how rank should be distributed across layers. QSpan prioritizes directions of the weight-quantization error according to their exposure through quantized activations and sensitivity to quantized gradients. It then initializes adapters along the most consequential directions through a closed-form solution and allocates a fixed total rank budget toward layers with large fractions of uncompensated training-weighted quantization error. Across language reasoning, personalized language modeling, and text-to-image personalization, QSpan consistently outperforms fully quantized LoRA and representative quantization-aware baselines under matched NVFP4 precision and adaptation budgets. These results demonstrate that aligning limited adapter capacity with training-relevant quantization errors enables effective fully quantized PEFT.

1 INTRODUCTION

Adapting large foundation models remains costly in resource-constrained settings. Parameter-efficient fine-tuning (PEFT), particularly LoRA (Hu et al., 2021), reduces trainable parameters and optimizer states through low-rank adapters, but still incurs substantial memory and computation from the activations and gradients of both the frozen backbone and LoRA branch as model size grows. Quantization provides a complementary solution. Existing quantized PEFT methods, including QLoRA (Dettmers et al., 2023) and its variants (Li et al., 2024b; Liu et al., 2024), mainly quantize the frozen backbone weights, with some methods (Huang et al., 2024) additionally quantizing backbone activations, while keeping the LoRA branch and gradients in high precision. We refer to quantizing the weights, activations, and gradients of the linear projections in both branches as *fully quantized PEFT*, which offers a promising path toward resource-efficient adaptation.

Recent microscaling floating-point formats, such as NVFP4 (NVIDIA et al., 2025), enable efficient low-precision computation on modern GPUs and have made training with 4-bit weights, activations, and gradients increasingly practical. Building on this hardware support, recent fully quantized training methods improve FP4 accuracy and stability through adaptive scaling, improved gradient estimation, and outlier handling (Chen et al., 2025; Panferov et al., 2026; Cook et al., 2025; Kim et al., 2026). However, these methods primarily target full-model training, where all parameters can adapt to low-precision errors. In fully quantized PEFT, the backbone remains frozen, leaving only low-rank adapters to compensate for the remaining quantization errors. Under a fixed LoRA budget, the key challenge is therefore to prioritize the directions of the weight-quantization error that are most relevant to adaptation and allocate the available rank across layers.

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.

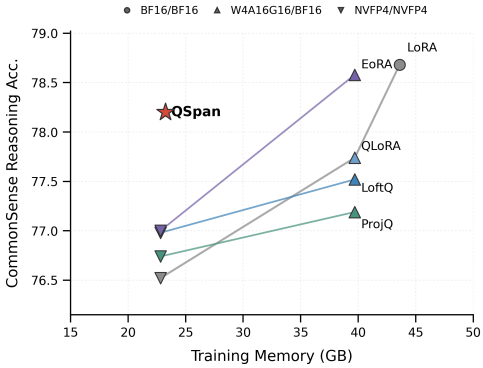


Figure 1: Accuracy–memory trade-off on LLaMA-3.2-3B-Instruct for commonsense reasoning. By quantizing both the backbone and LoRA branch to NVFP4, QSpan substantially reduces training memory while maintaining accuracy close to BF16 LoRA.

Existing quantization-aware LoRA initialization methods construct adapters from weight-quantization errors, as in LoftQ (Li et al., 2024b), or prioritize weight-error directions by their effects on model outputs using input activations, as in EoRA (Liu et al., 2024) and ProjQ (Yu et al., 2026). However, preserving model outputs alone does not indicate how the remaining distortions affect the training loss. Their importance also depends on how they interact with the loss gradients encountered during low-precision training. This distinction becomes particularly important in fully quantized PEFT because gradient quantization changes the backward signals used to optimize the adapters, an effect that forward information alone cannot reflect. Effective LoRA initialization should therefore consider both quantized activations, which characterize how errors affect model outputs, and quantized gradients, which indicate their relevance to the training objective.

We analyze fully quantized PEFT to understand where its performance degradation originates and find that quantized activations and gradients both reduce adaptation quality. Motivated by this analysis, we introduce **QSpan**, a training-aware low-rank framework designed to reduce the loss discrepancy introduced by fully quantized training. Under a fixed rank budget, QSpan determines which directions of the frozen backbone's weight-quantization error each low-rank adapter should span by considering their exposure to quantized input activations and sensitivity to quantized output gradients. This yields a *training-weighted quantization error* that prioritizes directions using both the forward and backward signals encountered during low-precision fine-tuning. QSpan then initializes each adapter to span its dominant subspace through a closed-form low-rank solution. Including both quantized input activations and quantized output gradients in the initialization objective extends the compensation criterion beyond weight-space reconstruction and full-precision forward statistics. This formulation can be applied to the residual errors produced by different quantization schemes without modifying their underlying training recipes.

The importance of residual directions also varies substantially across layers. In some layers, a small number of directions captures most of the training-weighted quantization error, whereas other layers require more directions for comparable compensation. Uniform rank allocation can therefore waste capacity on layers whose error is captured at low rank while under-allocating capacity to layers with substantial remaining error. Existing non-uniform rank methods, such as RA-LoRA (Kim et al., 2024), allocate capacity using optimization dynamics or the weight-error spectra, but they do not measure how much loss-sensitive quantization error remains after the adapter subspace has been selected. QSpan instead uses the same training-aware decomposition to allocate rank according to the loss-sensitive error left uncompensated in each layer. Under a fixed total rank budget, QSpan assigns more dimensions to layers where important quantization errors remain insufficiently captured and fewer dimensions to layers where a low rank is already adequate. The same analysis thus determines both *which* error directions each adapter represents and *how many* dimensions the corresponding layer receives, without requiring separate calibration statistics or additional decompositions.

We evaluate QSpan under fully quantized PEFT with NVFP4 weights, activations, and gradients across language and text-to-image diffusion models. Under matched quantization settings, QSpan consistently outperforms fully quantized LoRA and the state-of-the-art quantized LoRA baselines. These results demonstrate that aligning limited LoRA capacity with training-relevant quantization errors improves fully quantized adaptation without increasing the total adapter budget.

2 Training-Aware Initialization for Fully Quantized PEFT

Fully quantized PEFT extends low-precision computation from the frozen backbone weights to the activations and gradients of both the backbone and LoRA branch. Although this substantially reduces training memory, it creates an adaptation gap that the frozen backbone cannot correct. Under a

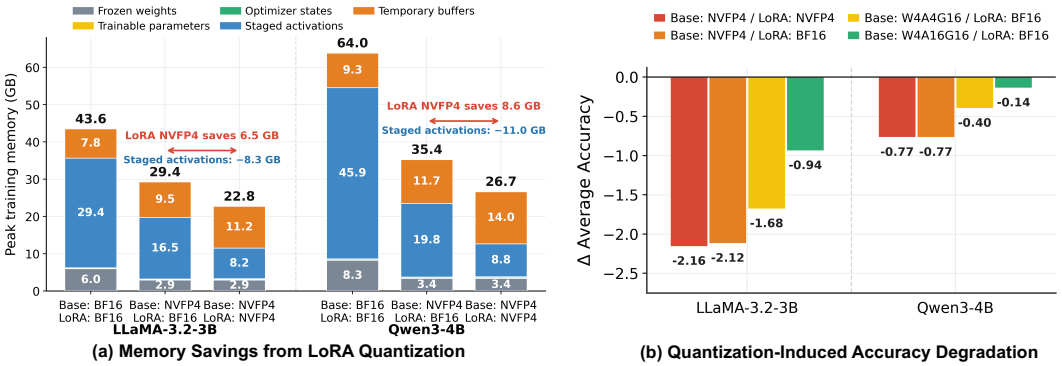


Figure 2: Efficiency and accuracy effects of progressively extending low precision in PEFT. **(a)** Quantizing the frozen backbone weights provides the largest memory reduction, while quantizing the LoRA branch further reduces the staged activations retained for adapter backpropagation. **(b)** Quantizing backbone activations and gradients causes additional degradation beyond weight-only quantization, motivating an initialization criterion that accounts for both quantized forward and backward signals. Quantizing the LoRA branch introduces comparatively limited degradation.

limited LoRA rank, the key question is therefore which directions of the frozen backbone's weight-quantization error each adapter should compensate. QSpan addresses this question by identifying the residual directions most consequential under the quantized forward and backward signals encountered during fine-tuning.

2.1 Fully Quantized PEFT

Low-precision training setup. We use NVFP4 (NVIDIA et al., 2025), a microscaling floating-point format with 4-bit E2M1 values, block-wise FP8 E4M3 scales shared across 16 values, and a tensor-level FP32 scale. Following the NVFP4 training recipe, we apply a Random Hadamard Transform (RHT) and stochastic rounding to gradient. We denote a precision setting by WxAyGz, specifying the precision of weights, activations, and gradients, respectively. We define *fully quantized PEFT* as applying W4A4G4 computation to the linear projections of both the frozen backbone and trainable LoRA branch during fine-tuning and evaluation.

Our implementation executes FPROP, DGRAD, and WGRAD for both the frozen backbone and trainable LoRA branch using NVFP4 GEMMs, including the LoRA-A and LoRA-B computations, without a BF16 LoRA GEMM fallback. BF16/FP32 tensors are materialized only at accumulation, nonlinear, attention, loss, and optimizer boundaries. To reduce redundant quantization and staged-activation memory, the frozen and LoRA branches share packed inputs, and LoRA-A directly produces packed NVFP4 intermediates consumed by LoRA-B. The complete computation pipeline and implementation details are provided in Appendix A.

Why quantize the LoRA branch? Despite its small parameter count, the LoRA branch retains intermediate activations for backpropagation. Keeping the LoRA branch in BF16 therefore leaves these staged activations in high precision even when the backbone is quantized. As shown in Fig. 2 (a), backbone weight quantization provides the largest memory reduction, while extending NVFP4 computation to the LoRA branch further reduces memory by storing its staged activations in low precision. This saving outweighs the modest temporary-buffer overhead of NVFP4 GEMMs, completing the low-precision computation path of fully quantized PEFT. Appendix C provides detailed training-memory breakdowns across models, including an analysis of the memory costs.

2.2 Why Forward and Backward Quantization Matter

Activation and gradient quantization introduce additional degradation. Fig. 2 (b) progressively extends low precision from the frozen backbone weights (W4A16G16) to its activations (W4A4G16), gradients (W4A4G4), and the LoRA branch. Quantizing backbone activations and gradients causes additional degradation beyond weight-only quantization, whereas quantizing the LoRA branch introduces comparatively limited degradation while reducing staged-activation mem-

ory. These results show that weight-space error alone does not explain the adaptation gap of fully quantized PEFT and motivate accounting for both quantized forward and backward signals.

Training-relevant error matters. Existing quantization-aware initialization methods (Li et al., 2024b; Liu et al., 2024; Yu et al., 2026) prioritize weight-quantization residuals using weight-space or forward-reconstruction criteria. However, Fig. 2 (b) shows that activation and gradient quantization also contribute to the adaptation gap. A criterion for fully quantized PEFT should therefore prioritize residual directions according to both their exposure through quantized activations and their sensitivity to quantized backward signals.

2.3 Closed-Form Training-Aware Initialization

Let $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ denote a frozen backbone weight matrix and let $Q(W)$ be its quantized counterpart. We define the weight-quantization error and the uncompensated quantization error after applying a rank-r LoRA update as (we use "uncompensated error" as shorthand below):

$$\Delta W = W - Q(W), \qquad E = \Delta W - BA, \tag{1}$$

where $B \in \mathbb{R}^{d_{\text{out}} \times r}$, $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and E denotes the uncompensated error. Because the backbone remains frozen, ΔW cannot be corrected through backbone optimization. The rank-constrained adapter must therefore select a low-dimensional subspace of this error to compensate.

During fully quantized PEFT, we collect quantized layer inputs $Q(X)$ and quantized output gradients $Q(G)$ from N calibration samples and construct

$$R_{Q(X)} = \frac{1}{N} Q(X) Q(X)^\top + \epsilon_X I, \qquad R_{Q(G)} = \frac{1}{N} Q(G) Q(G)^\top + \epsilon_G I, \tag{2}$$

where $\epsilon_X, \epsilon_G > 0$ are damping constants. Here, $R_{Q(X)}$ measures how strongly residual directions are exposed through quantized inputs, while $R_{Q(G)}$ measures the sensitivity of the training objective to the resulting output perturbations.

Theorem 1 (Training-aware QSpan initialization)**.** *Under a local quadratic approximation, a Fisher/Gauss–Newton curvature approximation, and a K–FAC factorization (Martens & Grosse, 2015) based on the quantized activation and gradient statistics, the local loss discrepancy induced by E is approximated by*

$$\Delta \mathcal{L}(E) \approx \frac{1}{2} \operatorname{Tr}\left(E^\top R_{Q(G)} E R_{Q(X)}\right). \tag{3}$$

For a fixed rank r, minimizing this discrepancy is equivalent to

$$\min_{\operatorname{rank}(BA) \le r} \left\| R_{Q(G)}^{1/2} (\Delta W - BA) R_{Q(X)}^{1/2} \right\|_F^2. \tag{4}$$

Let

$$\widetilde{\Delta W} = R_{Q(G)}^{1/2} \Delta W R_{Q(X)}^{1/2} = U \Sigma V^\top. \tag{5}$$

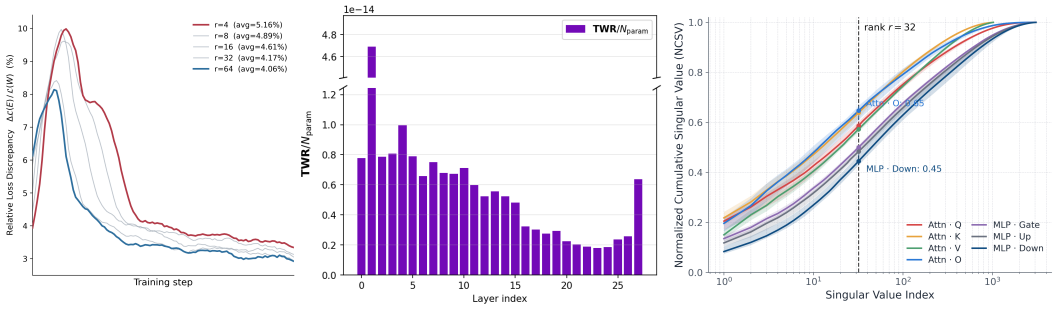
A global rank-r minimizer is

$$B^\star = R_{Q(G)}^{-1/2} U_r \Sigma_r^{1/2}, \qquad A^\star = \Sigma_r^{1/2} V_r^\top R_{Q(X)}^{-1/2}, \tag{6}$$

where U_r, V_r, and Σ_r contain the leading r singular components of $\widetilde{\Delta W}$. The minimum residual objective is $\sum_{i>r} \sigma_i^2$.

The proof is provided in Appendix B. Briefly, the two-sided transformation $\widetilde{B} = R_{Q(G)}^{1/2} B$ and $\widetilde{A} = A R_{Q(X)}^{1/2}$ reduces Eq. 4 to a best rank-r approximation of $\widetilde{\Delta W}$. The result then follows from the Eckart–Young–Mirsky theorem (Eckart & Young, 1936; Mirsky, 1960).

The dominant singular directions of $\widetilde{\Delta W}$ identify the frozen-backbone residual subspace that is most consequential under the quantized forward and backward signals. When $R_{Q(G)} = I$, QSpan reduces to an activation-weighted, forward-only objective. When both $R_{Q(G)} = I$ and $R_{Q(X)} = I$, it further reduces to the truncated SVD of ΔW, corresponding to a single-step LoftQ initialization (Li et al., 2024b). QSpan therefore generalizes weight-space and activation-aware compensation by additionally incorporating the quantized backward signal. Although the LoRA branch is also quantized during fully quantized PEFT, Eq. 4 optimizes the ideal low-rank product BA. This design prioritizes the residual of the frozen backbone, which cannot adapt during fine-tuning, while allowing the trainable adapter to adjust under its low-precision computation.



(a) Relative Loss Discrepancy (b) Layer-wise TWR (c) TWR NCSV

Figure 3: Rank-allocation analysis on LLaMA-3.2-3B-Instruct. (a) Relative loss discrepancy across LoRA ranks. (b) Layer-wise training-weighted residual (TWR) at a uniform rank. (c) Normalized cumulative singular values (NCSV) of the weight-quantization residual under the TWR geometry.

3 Residual-Aware Rank Allocation

QSpan initialization determines which residual directions each adapter spans. We further relax the uniform-rank assumption by redistributing a fixed total rank budget according to the training-relevant residual left uncompensated in each layer.

3.1 Why Uniform Rank Is Suboptimal

Rank provides compensation capacity. Fig. 3 (a) measures the loss discrepancy $\Delta\mathcal{L}(E)$ between BF16 and fully quantized PEFT across different LoRA ranks. A smaller discrepancy indicates that fully quantized PEFT more closely follows the BF16 training loss. Although the discrepancy can be large early in fine-tuning, particularly at rank 64, larger ranks generally produce smaller discrepancies as training proceeds. This trend suggests that increasing the LoRA rank provides additional capacity to compensate for quantization-induced error. Uniformly increasing the rank of every layer, however, may be inefficient because the required compensation capacity can differ substantially across layers.

Compensation demands vary across layers. To quantify the remaining compensation demand after initialization, we introduce the training-weighted residual (TWR), which can be viewed as an empirical approximation of the training-aware objective in Eq. 3. While Eq. 3 evaluates the expected loss discrepancy through quantized activation and gradient statistics, TWR measures the same loss-sensitive residual using calibration samples. For layer l, let $E_l = \Delta W_l - B_l A_l$ denote uncompensated error remaining after LoRA initialization. Given a quantized layer input $x_{q,n}$ and its reference output gradient $g_{\mathrm{ref},n}$, we measure $\mathrm{TWR}(E_l) = \frac{1}{N}\sum_{n=1}^{N}\left(g_{\mathrm{ref},n}^{\top} E_l x_{q,n}\right)^2$. Here, $E_l x_{q,n}$ denotes the output perturbation induced by the uncompensated residual. Weighting this perturbation by $g_{\mathrm{ref},n}$ reflects its impact on the training objective. TWR therefore measures not only how much output error remains, but also how strongly that error can affect the training objective.

As shown in Fig. 3 (b), uniform-rank LoRA leaves substantially different amounts of TWR across layers, indicating that equal rank does not provide comparable compensation throughout the model. Fig. 3 (c) provides further insight into this behavior. Different projection types exhibit markedly different spectral decay rates under the TWR geometry. At a fixed rank budget ($r = 32$), attention projections retain a substantially larger fraction of residual energy than MLP projections such as Down. Thus, some layers are inherently more compressible than others and require less rank to achieve a similar level of residual compensation. These observations motivate allocating a fixed rank budget according to layer-wise compensation demand rather than uniformly across layers.

3.2 Rank Allocation by Uncompensated Error

For L target layers with a uniform baseline rank r, the total rank budget is $R_{\mathrm{total}} = Lr$. Let $\{\sigma_{l,i}\}$ denote the singular values of the training-weighted residual matrix $\widetilde{\Delta W}_l$ defined in 5. Because QSpan initialization captures its leading singular directions, we define the fraction of training-weighted

residual energy left beyond the baseline rank as

$$w_l = 1 - \frac{\sum_{i=1}^{r} \sigma_{l,i}^2}{\sum_i \sigma_{l,i}^2}. \tag{7}$$

We set $w_l = 0$ when $\|\widetilde{\Delta W}_l\|_F^2 = 0$. A small w_l indicates that the baseline rank already captures most of the layer's training-weighted residual energy, whereas a large w_l indicates that important directions remain beyond the captured subspace.

The normalized residual fraction is preferable to the absolute residual magnitude for rank allocation. A large residual may be concentrated in a few dominant directions and therefore require only a small rank. Conversely, a smaller residual with a slowly decaying spectrum may require more dimensions. The score w_l captures this spectral compensation demand relative to the uniform baseline.

QSpan assigns the real-valued quota $q_l = R_{\text{total}} w_l / \sum_m w_m$ and converts the quotas into integer ranks satisfying

$$r_{\min} \le r_l \le \min(r_{\max}, d_{\text{in},l}, d_{\text{out},l}), \qquad \sum_l r_l = R_{\text{total}}.$$

Starting from bounded rounded quotas, we apply unit corrections according to the remaining quota deficits or surpluses until the total budget is satisfied. If all allocation scores are zero, QSpan retains the uniform allocation $r_l = r$.

QSpan performs rank allocation during initialization using the same singular values already computed for training-aware initialization, requiring no additional calibration pass or matrix decomposition. The singular values obtained during QSpan initialization determine how the fixed budget is distributed, while the corresponding singular vectors determine which residual directions each adapter spans through Eq. 6. The same training-aware decomposition therefore determines both the subspace and the capacity of each adapter.

4 Experiments

4.1 Experimental Settings

Language Models. We conduct experiments with LLaMA-3.2-3B-Instruct (Grattafiori et al., 2024), Qwen3-4B-Instruct (Yang et al., 2025), Qwen3.5-2B (Team, 2026), and Gemma4-E2B-IT (Team et al., 2026) models. For commonsense reasoning, models are fine-tuned on Commonsense-15K and evaluated on eight benchmarks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SocialIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy, ARC-Challenge (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018). For mathematical reasoning, models are fine-tuned on MetaMathQA-40K (Yu et al., 2023) and evaluated on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). We report the average accuracy across the corresponding evaluation datasets. We further evaluate QSpan on the LaMP (Salemi et al., 2024) benchmark to assess personalized language modeling. LaMP consists of multiple tasks in which a language model generates task-specific outputs conditioned on a user's historical data. We follow the official task-specific evaluation protocols and compare QSpan with the same quantized LoRA baselines used in the other language-model experiments.

Text-to-Image Diffusion Models. For text-to-image personalization, we evaluate QSpan on diffusion backbones including FLUX.1-dev (Labs, 2024) and FLUX.2-Klein-4B (Labs, 2025) using the DreamBooth (Ruiz et al., 2023) dataset. For quantization calibration, we use CustomConcept101 (Kumari et al., 2023) as the calibration set. We measure subject fidelity using DINO (Caron et al., 2021) and CLIP-I (Radford et al., 2021), and text-image alignment using CLIP-T.

Baselines. We compare QSpan with standard LoRA and representative quantization-aware or rank-adaptive methods, including LoftQ (Li et al., 2024b), EoRA (Liu et al., 2024), RoLoRA (Huang et al., 2024), SVDQuant (Li et al., 2024a), ProjQ (Yu et al., 2026), and RA-LoRA (Kim et al., 2024). Unless otherwise specified, the frozen backbone and LoRA branch both use W4A4G4 computation with NVFP4. Following the NVFP4 training recipe, we apply stochastic rounding and a Random Hadamard Transform during gradient quantization. All methods are evaluated using the same model, dataset, precision configuration, and total LoRA rank budget for a fair comparison.

Method	Precision	BoolQ	PIQA	SIQA	HS	WG	ARC-e	ARC-c	OBQA	Avg.
LLaMA-3.2-3B-Instruct										
LoRA	BF16	69.63	81.12	74.36	86.79	69.38	88.47	77.47	82.20	78.68
LoRA	NVFP4	67.83	80.14	73.64	83.35	66.06	86.36	74.74	80.00	76.52
LoftQ	NVFP4	68.20	81.28	74.21	82.77	66.14	**87.46**	74.40	81.40	76.98
RoLoRA	NVFP4	67.68	80.47	73.85	81.06	66.85	85.82	72.78	78.40	75.86
EoRA	NVFP4	67.28	80.79	74.16	84.17	66.93	87.08	75.17	80.40	77.00
SVDQuant	NVFP4	67.16	79.82	72.57	80.53	65.90	85.61	72.18	79.80	75.45
ProjQ	NVFP4	67.83	81.07	73.70	82.81	66.30	86.11	73.89	82.20	76.74
RA-LoRA	NVFP4	**68.99**	80.90	74.05	83.23	66.30	86.36	74.66	83.20	77.21
QSpan	NVFP4	68.29	**81.77**	**74.51**	**85.26**	**68.11**	87.37	**76.88**	**83.40**	**78.20**
Qwen3-4B-Instruct										
LoRA	BF16	69.24	84.77	76.41	90.16	72.30	95.83	88.99	89.00	83.34
LoRA	NVFP4	**69.45**	84.17	75.90	88.41	70.64	95.08	88.14	88.80	82.57
LoftQ	NVFP4	68.81	84.44	75.64	88.18	72.38	**95.50**	88.91	89.00	82.86
RoLoRA	NVFP4	69.14	83.08	75.08	87.12	70.96	93.90	84.81	88.40	81.56
EoRA	NVFP4	69.33	**85.47**	76.15	89.42	72.53	95.41	**89.16**	89.20	83.34
SVDQuant	NVFP4	68.10	83.41	75.95	87.44	72.06	94.40	85.32	87.40	81.76
ProjQ	NVFP4	67.83	85.36	76.66	88.40	70.56	95.03	88.31	89.40	82.69
RA-LoRA	NVFP4	68.53	84.39	75.74	87.75	71.27	**95.50**	88.82	89.40	82.68
QSpan	NVFP4	69.42	85.20	**78.56**	**89.61**	**74.51**	95.16	88.14	**90.60**	**83.90**
Qwen3.5-2B										
LoRA	BF16	64.01	81.23	70.32	82.36	64.40	91.79	82.68	80.00	77.10
LoRA	NVFP4	62.14	77.86	68.78	77.97	61.56	89.65	78.16	76.00	74.01
LoftQ	NVFP4	59.85	79.11	68.42	78.27	63.14	90.07	78.58	78.80	74.53
EoRA	NVFP4	62.39	78.56	67.25	78.62	62.90	90.24	79.01	79.40	74.80
SVDQuant	NVFP4	59.20	75.95	66.89	71.05	60.77	87.04	74.83	75.60	71.42
ProjQ	NVFP4	57.22	76.99	68.58	76.71	60.85	90.28	77.90	77.40	73.24
RA-LoRA	NVFP4	64.31	78.84	67.76	77.75	64.09	90.36	77.82	78.80	74.97
QSpan	NVFP4	**64.65**	**79.54**	**70.27**	**79.56**	**66.38**	**90.70**	**79.35**	**81.60**	**76.51**
Gemma4-E2B-IT										
LoRA	BF16	65.69	78.40	73.08	78.32	65.43	88.89	77.82	78.80	75.80
LoRA	NVFP4	62.78	75.95	67.40	64.83	62.43	84.93	74.49	73.40	70.78
LoftQ	NVFP4	64.22	75.63	68.12	67.08	61.96	83.54	75.26	76.40	71.52
EoRA	NVFP4	63.52	74.92	68.99	69.24	62.19	84.51	**77.82**	74.00	71.90
SVDQuant	NVFP4	62.05	74.97	67.91	63.90	59.59	83.63	70.73	73.60	69.55
ProjQ	NVFP4	64.59	75.14	68.17	67.75	61.09	83.88	74.32	73.60	71.07
RA-LoRA	NVFP4	64.50	75.14	67.35	65.51	62.98	84.26	74.15	74.00	70.99
QSpan	NVFP4	**64.74**	**76.12**	**70.47**	**75.01**	**64.88**	**86.03**	76.54	**78.80**	**74.07**

Table 1: Accuracy (%) on eight commonsense reasoning benchmarks. Precision denotes the precision used for both the frozen backbone and the LoRA branch.

4.2 Evaluation Across Models and Tasks

Commonsense reasoning. Table 1 compares QSpan across eight commonsense reasoning benchmarks. Fully quantizing both the backbone and LoRA branch generally degrades standard LoRA, whereas QSpan consistently outperforms fully quantized LoRA and competing baselines across all four model families. These results demonstrate that prioritizing training-relevant residual directions improves adaptation across architectures under matched precision and rank budgets. Appendix C reports the training-memory breakdowns and QSpan rank-allocation overhead.

Mathematical reasoning. Table 2 reports the results on GSM8K and MATH. Mathematical reasoning exhibits a larger gap between BF16 and fully quantized LoRA, indicating that accurate task-specific adaptation becomes more challenging under low-precision computation. Nevertheless, QSpan achieves the strongest fully quantized performance across all evaluated models, demonstrating its effectiveness on challenging reasoning tasks.

Personalized language modeling. Table 3 reports results on LaMP-2 movie tagging and LaMP-4 news headline generation, which require adaptation to user-specific data. QSpan achieves the strongest fully quantized performance across all metrics, outperforming fully quantized LoRA and representative baselines. These results show that prioritizing training-relevant residual directions helps capture user-specific signals under low-precision adaptation, highlighting the potential of QSpan for resource-constrained on-device personalization with limited memory and computation.

Method	Precision	LLaMA3.2-3B			Qwen3-4B			Qwen3.5-2B			Gemma4-2B		
		GSM8K	MATH	Avg.	GSM8K	MATH	Avg.	GSM8K	MATH	Avg.	GSM8K	MATH	Avg.
LoRA	BF16	73.84	36.66	55.25	84.00	46.14	65.07	77.41	38.92	58.16	80.89	41.02	60.96
LoRA	NVFP4	69.37	31.74	50.56	83.02	43.24	63.13	72.86	33.46	53.16	74.75	36.52	55.64
LoftQ	NVFP4	70.28	31.12	50.70	83.62	42.44	63.03	71.72	31.60	51.66	74.68	34.82	54.75
RoLoRA	NVFP4	68.92	27.68	48.30	80.36	40.50	60.43	–	–	–	–	–	–
EoRA	NVFP4	71.19	31.46	51.33	84.08	43.02	63.55	72.33	33.20	52.76	76.19	35.16	55.68
SVDQuant	NVFP4	68.69	27.54	48.11	80.67	40.28	60.47	68.99	28.30	48.65	67.48	26.64	47.06
ProjQ	NVFP4	71.19	29.96	50.58	81.80	42.96	62.38	71.42	31.40	51.41	72.10	26.94	49.52
RA-LoRA	NVFP4	71.49	30.36	50.93	82.11	43.14	62.62	72.71	32.68	52.69	74.53	35.58	55.05
QSpan	NVFP4	**72.63**	**32.18**	**52.41**	**84.53**	**43.70**	**64.12**	**74.22**	**34.16**	**54.19**	**77.18**	**37.06**	**57.12**

Table 2: Accuracy (%) on mathematical reasoning benchmarks. Precision denotes the numerical precision used for both the frozen backbone and the LoRA branch.

Method	Precision	LaMP-2		LaMP-4	
		Acc.↑	F1↑	R-1↑	R-L↑
LoRA	BF16	0.7050	0.6294	0.1822	0.1646
LoRA	NVFP4	0.6684	0.5748	0.1760	0.1595
LoftQ	NVFP4	0.6766	0.5877	0.1768	0.1592
EoRA	NVFP4	0.6775	0.5823	0.1772	0.1596
SVDQuant	NVFP4	0.6193	0.5158	0.1702	0.1542
ProjQ	NVFP4	0.6799	0.5930	0.1759	0.1589
QSpan	NVFP4	**0.6844**	**0.5943**	**0.1789**	**0.1610**

Table 3: Personalization performance of LLaMA3.2-3B-Instruct on LaMP.

Method	Precision	FLUX.1-dev			FLUX.2-Klein-4B		
		DINO↑	CLIP-I↑	CLIP-T↑	DINO↑	CLIP-I↑	CLIP-T↑
LoRA	BF16	0.7257	0.8099	0.3107	0.7089	0.8043	0.3083
LoRA	NVFP4	0.7140	0.8067	0.3092	0.7026	0.8016	**0.3079**
LoftQ	NVFP4	0.6972	0.7959	**0.3137**	0.7072	0.8080	0.3047
EoRA	NVFP4	0.7099	0.8012	0.3123	0.7266	0.8151	0.3030
RA-LoRA	NVFP4	0.7201	0.8099	0.3092	0.7105	0.8049	0.3077
QSpan	NVFP4	**0.7227**	**0.8103**	0.3099	**0.7323**	**0.8183**	0.3012

Table 4: Average DINO, CLIP-I, and CLIP-T scores on the DreamBooth dataset.

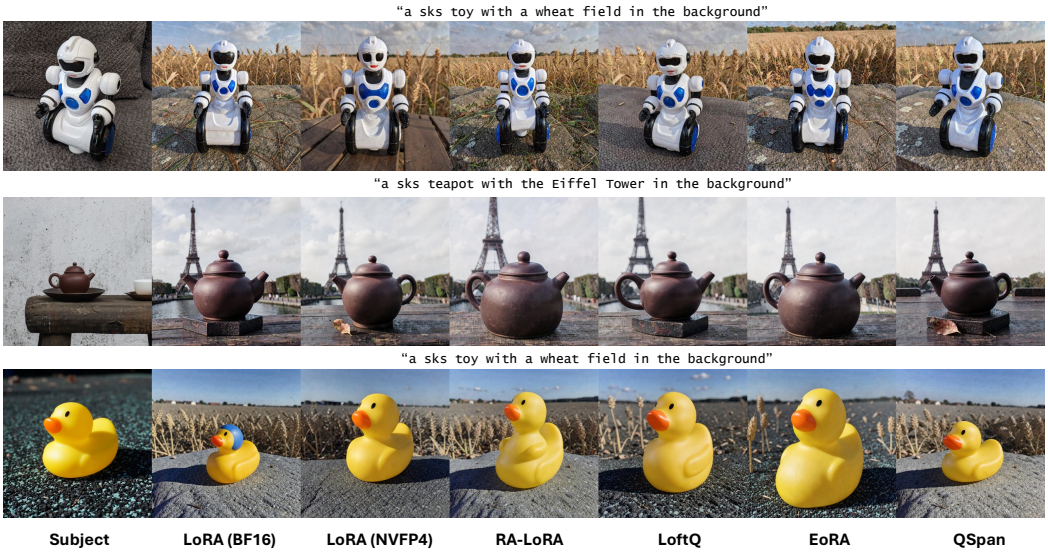


Figure 4: Qualitative comparison of T2I personalization on the DreamBooth using FLUX-2.Klein.

Text-to-image (T2I) personalization. Table 4 reports results on DreamBooth personalization. On FLUX.2-Klein-4B, QSpan achieves the highest DINO and CLIP-I scores among all evaluated methods, including BF16 LoRA, indicating improved subject fidelity under fully quantized fine-tuning. This improvement is accompanied by a modest decrease in CLIP-T, revealing a trade-off between subject preservation and text-image alignment. Consistently, Fig. 4 shows that QSpan more reliably preserves subject-specific visual characteristics across diverse prompts and contexts.

4.3 Analysis on QSpan

Loss discrepancy. Fig. 5 (a) compares the training-loss trajectories of BF16 LoRA and fully quantized adaptation methods. Compared with the competing baselines, QSpan maintains a loss trajectory closer to that of BF16 LoRA throughout fine-tuning. This result directly supports the training-aware objective of QSpan, which selects residual directions according to their influence on the local

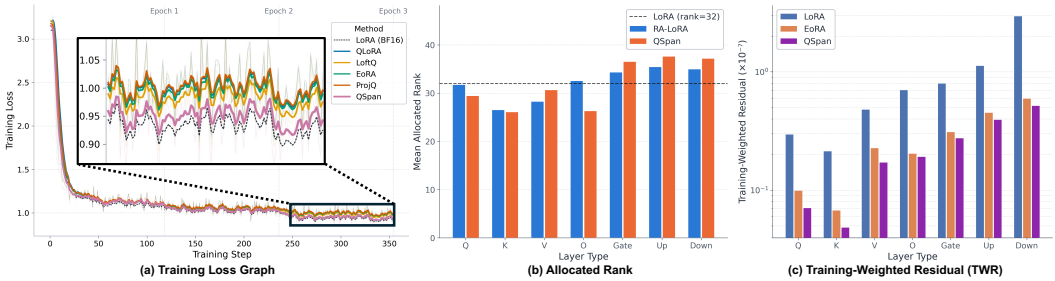


Figure 5: Training dynamics and rank-allocation analysis on LLaMA-3.2-3B-Instruct. (a) Training loss curves on commonsense reasoning tasks. (b) Mean rank allocated to each layer type by QSpan and RA-LoRA. (c) Training-weighted residual (TWR) across layer types.

loss rather than forward reconstruction alone. The benefit therefore extends beyond improving the initial weight approximation and persists during fully quantized optimization.

Rank allocation across layer types. Fig. 5 (b) reports the allocated ranks averaged over layers of each attention and MLP layer type. QSpan produces non-uniform, architecture-dependent allocation patterns rather than imposing a fixed preference for particular layer types. As shown in Fig. 5 (c), uniform LoRA leaves substantial variation in the training-weighted residual (TWR) across layer types, whereas QSpan markedly reduces this imbalance. Consistent with these residual profiles, QSpan assigns fewer ranks to the Q and K projections, which exhibit relatively low TWR, while allocating more ranks to the MLP Up and Down projections, where the remaining TWR is higher. These results show that QSpan adaptively redistributes a fixed total rank budget toward layers with larger training-relevant residuals, yielding a more balanced reduction of quantization-induced error.

Ablation study. Table 5 isolates the contributions of training-aware initialization and layer-wise rank allocation. Training-aware initialization consistently improves fully quantized LoRA across all evaluated models. Adding rank allocation provides further gains without increasing the total rank budget. These complementary improvements support the unified design of QSpan, in which the same training-weighted residual decomposition determines both which directions each adapter spans and how much rank each layer receives.

Method	Qwen3	LLaMA3.2	Qwen3.5	Gemma4
LoRA	82.57	76.52	74.01	70.78
+ Init.	83.38	77.31	75.21	73.22
+ RA	**83.90**	**78.20**	**76.51**	**74.07**

Table 5: Ablation of QSpan initialization and rank allocation (RA).

Residual objective analysis. Table 6 compares residual metrics on LLaMA-3.2-3B-Instruct. For $E_l = W_l - Q(W_l) - B_l A_l$, we define the weight and output reconstruction errors as $\mathrm{WR}(E_l) = \|E_l\|_F^2$ and $\mathrm{QAWR}(E_l) = \frac{1}{N}\sum_{n=1}^{N}\|E_l x_{q,n}\|_2^2$, respectively, and report the previously defined TWR to measure how strongly the remaining output error aligns with loss-sensitive directions. QSpan (Init.) reduces all three metrics over LoRA, with a particularly large reduction in TWR. Under the same fixed total rank budget, QSpan further lowers TWR through layer-wise rank allocation. Its alignment with lower training loss and higher task performance supports TWR as the most relevant compensation criterion for fully quantized adaptation.

Method	WR $\times 10^{-6}$	QAWR $\times 10^{-7}$	TWR $\times 10^{-15}$
LoRA	3.36	2.72	6.54
EoRA	**3.29**	**1.25**	1.95
LoftQ	3.69	3.01	7.47
QSpan (Init.)	3.34	1.30	1.76
QSpan	3.34	1.27	**1.66**

Table 6: Comparison of WR, QAWR, and TWR residual objectives.

5 Conclusion

We introduced QSpan, a training-aware framework that aligns limited LoRA capacity with consequential quantization errors in fully quantized fine-tuning. Using quantized activation and gradient statistics, QSpan derives a closed-form initialization that spans the dominant training-weighted residual directions and allocates a fixed rank budget according to the remaining layer-wise compensation demand. Across language and text-to-image personalization tasks, QSpan consistently improves fully quantized LoRA under matched NVFP4 precision and rank budgets.

AI USE STATEMENT

We used generative AI tools to assist with language editing, proofreading, and improving the clarity and readability of the manuscript. We did not use generative AI tools to generate research ideas, formulate scientific claims, design experiments, analyze results, or draw conclusions. We additionally used generative AI tools to help revise and refine presentation of the text during manuscript preparation. All AI-assisted edits were carefully reviewed and verified by the authors to ensure technical correctness, originality, and consistency with the reported methodology and findings. We take full responsibility for the final content of this work, including all text, claims, figures, tables, and other artifacts.

ETHICS STATEMENT

This work studies efficient adaptation of large language and diffusion models through quantized low-rank fine-tuning. All experiments were conducted using publicly available models and datasets, and we did not collect or use personally identifiable information. QSpan is a model optimization technique that improves the efficiency of adaptation under memory and compute constraints and does not introduce new application-specific capabilities beyond those of the underlying foundation models. Nevertheless, any potential misuse, biases, or harmful outputs associated with the adapted models remain inherited from the underlying pretrained models and datasets. We encourage practitioners to follow existing responsible AI guidelines and deploy adapted models with appropriate safeguards and evaluation procedures.

REPRODUCIBILITY STATEMENT

We make reproducibility a primary consideration throughout this work. The paper provides a complete description of the proposed QSpan methodology, including residual importance estimation, training-aware weighting, and rank allocation procedures. Experimental settings, model architectures, datasets, evaluation metrics, quantization formats, and hyperparameters are described in the main paper and appendix. The appendix additionally details the fully quantized LoRA training pipeline, calibration procedures, implementation details, and hardware-specific execution considerations.

REFERENCES

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Hengjie Cao, Mengyi Chen, Yifeng Yang, Fang Dong, Ruijun Huang, Jixian Zhou, Anrui Chen, Mingzhi Dong, Yujiang Wang, Jinlong Hou, et al. Metis: Training llms with fp4 quantization. In *International Conference on Learning Representations*, volume 2026, pp. 73297–73319, 2026.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Int. Conf. Comput. Vis.*, 2021.

Roberto Castro, Andrei Panferov, Rush Tabesh, Oliver Sieberling, Jiale Chen, Mahdi Nikdan, Saleh Ashkboos, and Dan Alistarh. Quartet: Native fp4 training can be optimal for large language models. *Advances in Neural Information Processing Systems*, 38:43552–43572, 2026.

Yuxiang Chen, Yifan Liu, Xiaoming Xu, Pengle Zhang, Michael Beyer, Martin Rapp, Jun Zhu, and Jianfei Chen. Tetrajet-v2: Accurate nvfp4 training for large language models with oscillation suppression and outlier control. *arXiv preprint arXiv:2510.27527*, 2025.

Brian Chmiel, Ron Banner, Elad Hoffer, Hilla Ben Yaacov, and Daniel Soudry. Accurate neural training with 4-bit matrix multiplications at standard formats. *arXiv preprint arXiv:2112.10769*, 2021.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pp. 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jack Cook, Junxian Guo, Guangxuan Xiao, Yujun Lin, Keith Wyss, Mahdi Nazemi, Asit Mishra, Carlo del Mundo, Tijmen Blankevoort, and Song Han. Four over six: More accurate nvfp4 quantization with adaptive block scaling. *arXiv preprint arXiv:2512.02010*, 2025.

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.

Siyu Ding, Mingchuan Ma, Jiabo Tong, Xingrun Xing, Ziming Wang, and Guoqi Li. Full-stack fp4: Stable llm pretraining with quantized projections, optimizers, and attention. *arXiv preprint arXiv:2607.04422*, 2026.

Peijie Dong, Ruibo Fan, Yuechen Tao, Di Mou, Wenhu Hu, Zhenheng Tang, Yinghao Yu, Jiamang Wang, Wenbo Su, Guodong Yang, et al. Dissecting outlier dynamics in llm nvfp4 pretraining. *arXiv preprint arXiv:2602.02047*, 2026.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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.

Xijie Huang, Zechun Liu, Shih-Yang Liu, and Kwang-Ting Cheng. Rolora: Fine-tuning rotated outlier-free llms for effective weight-activation quantization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7563–7576, 2024.

Minsoo Kim, Sihwa Lee, Wonyong Sung, and Jungwook Choi. Ra-lora: Rank-adaptive parameter-efficient fine-tuning for accurate 2-bit quantized large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 15773–15786, 2024.

Seonggon Kim, Alireza Khodamoradi, Pranathi Vasireddy, Kristof Denolf, and Eunhyeok Park. Adahop: Fast and accurate low-precision training via outlier-pattern-aware rotation. *arXiv preprint arXiv:2604.02525*, 2026.

Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

Black Forest Labs. FLUX, 2024. URL `https://github.com/black-forest-labs/flux`.

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. `https://bfl.ai/blog/flux-2`, 2025.

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*, 2024a.

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, 2024b.

Baohao Liao, Christian Herold, Shahram Khadivi, and Christof Monz. Apiq: Finetuning of 2-bit quantized large language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20996–21020, 2024.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Vijay Lingam, Atula Tejaswi, Aditya Vavre, Aneesh Shetty, Gautham K Gudur, Joydeep Ghosh, Alex Dimakis, Eunsol Choi, Aleksandar Bojchevski, and Sujay Sanghavi. Svft: Parameter-efficient fine-tuning with singular vectors. *Advances in Neural Information Processing Systems*, 37:41425–41446, 2024.

Shih-Yang Liu, Maksim Khadkevich, Nai Chit Fung, Charbel Sakr, Chao-Han Huck Yang, Chien-Yi Wang, Saurav Muralidharan, Hongxu Yin, Kwang-Ting Cheng, Jan Kautz, et al. Eora: Fine-tuning-free compensation for compressed llm with eigenspace low-rank approximation. *arXiv preprint arXiv:2410.21271*, 2024.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 11(1):50–59, 1960.

Felix Abecassis NVIDIA, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, et al. Pretraining large language models with nvfp4. *arXiv preprint arXiv:2509.25149*, 2025.

Andrei Panferov, Erik Schultheis, Soroush Tabesh, and Dan Alistarh. Quartet ii: Accurate llm pre-training in nvfp4 by improved unbiased gradient estimation. *arXiv preprint arXiv:2601.22813*, 2026.

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. 2021.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

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 *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. Lamp: When large language models meet personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7370–7392, 2024.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. Democratizing large language models via personalized parameter-efficient fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6476–6491, 2024.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Yujin Wang, Shunan Dong, Zongle Huang, Yichen You, Liu He, Huazhong Yang, Yongpan Liu, and Hongyang Jia. Hyc-lora: Memory efficient lora fine-tuning with hybrid activation compression. *Proceedings of Machine Learning and Systems*, 7, 2025.

Haocheng Xi, Changhao Li, Jianfei Chen, and Jun Zhu. Training transformers with 4-bit integers. *Advances in Neural Information Processing Systems*, 36:49146–49168, 2023.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 52401–52418, 2024.

Bowen Xue, Zihan Min, Xingyang Li, Zhekai Zhang, Haocheng Xi, Lvmin Zhang, Maneesh Agrawala, Jun-Yan Zhu, Song Han, Yujun Lin, et al. Fourtune: Towards fully 4-bit efficient post-training for diffusion models. *arXiv preprint arXiv:2607.05711*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Wenya Yu, Chao Zhang, Li Wang, Samson Lasaulce, and Merouane Debbah. Projq: Project-and-quantize for adapter-aware llm compression. *arXiv preprint arXiv:2606.00494*, 2026.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Cheng Zhang, Jianyi Cheng, George A Constantinides, and Yiren Zhao. Lqer: Low-rank quantization error reconstruction for llms. *arXiv preprint arXiv:2402.02446*, 2024.

Cheng Zhang, Jeffrey TH Wong, Can Xiao, George Constantinides, and Yiren Zhao. Qera: an analytical framework for quantization error reconstruction. In *International Conference on Learning Representations*, volume 2025, pp. 12531–12560, 2025.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

A END-TO-END W4A4G4 NVFP4 TRAINING PIPELINE

Fig. 6 illustrates the end-to-end W4A4G4 pipeline used throughout this work. The frozen backbone performs FPROP and DGRAD with NVFP4 GEMM operands, while the trainable LoRA branch additionally performs WGRAD in NVFP4. Following the NVFP4 training recipe, activations and output gradients are quantized to NVFP4, with a Random Hadamard Transform (RHT) and stochastic rounding (SR) applied to gradient quantization. The backbone remains frozen, and only the LoRA parameters are updated. Unlike quantized PEFT settings that retain the LoRA branch in higher precision, all LoRA-A and LoRA-B GEMMs use NVFP4 compute operands without a BF16 GEMM fallback. BF16/FP32 tensors are materialized only at required accumulation and non-GEMM boundaries, including normalization, rotary embeddings, attention, residual addition, SwiGLU, loss computation, and optimizer updates.

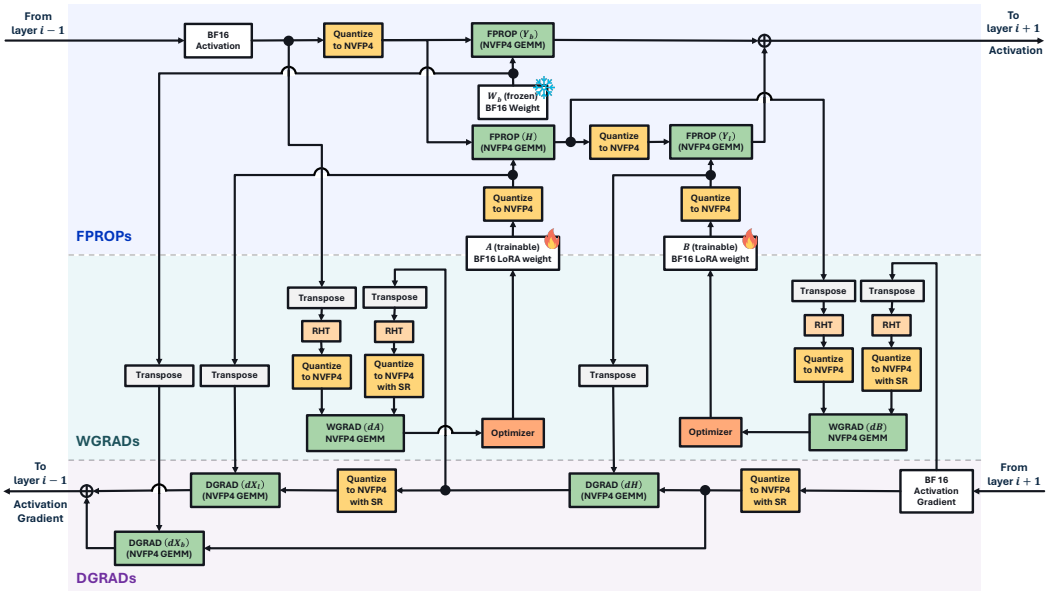


Figure 6: End-to-end W4A4G4 pipeline for fully quantized LoRA fine-tuning. The frozen backbone performs FPROP and DGRAD with NVFP4 GEMM operands, while the trainable LoRA branch additionally performs WGRAD in NVFP4. Only the LoRA parameters are updated, with higher precision retained at the required accumulation, non-GEMM, loss, and optimizer boundaries.

NVFP4 computation backend. We use NVIDIA Transformer Engine for standard NVFP4 GEMMs and custom CUTLASS kernels for grouped and LoRA-specific operations on Blackwell GPUs. The kernels use block-scaled E2M1 operands with FP32 accumulation and produce either BF16 or packed NVFP4 outputs according to the subsequent operation. After conversion, each frozen linear layer retains only a GEMM-ready row-oriented NVFP4 weight, while its BF16 checkpoint weight is released without allocating gradients or optimizer states. Kernel plans, buffers, and workspaces are constructed once per shape and reused.

Packed LoRA forward execution. The frozen projection and LoRA-A share a packed NVFP4 input, with Q/K/V sharing one quantized normalized input and gate/up sharing one quantized MLP input. This avoids redundant activation quantization within grouped projections. LoRA-A directly writes its output to a packed NVFP4 buffer consumed by LoRA-B, avoiding a BF16 intermediate and an additional quantization pass. A heterogeneous grouped NVFP4 kernel jointly evaluates the corresponding LoRA-B projections for Q/K/V or gate/up. The frozen and LoRA outputs are combined at a BF16 accumulation boundary before the subsequent nonlinear or residual operation. When supported by the weight layout, a custom CUDA epilogue performs the scaled LoRA addition with reference BF16 rounding, eliminating separate framework-level multiply and add operations while remaining a distinct kernel launch.

Executing QSpan's non-uniform rank allocation. QSpan assigns non-uniform ranks across layers under a fixed total rank budget. Our grouped LoRA kernels directly execute these allocations, treating each assigned rank r_l as the logical rank used for the LoRA scaling factor α/r_l. For Blackwell kernel alignment, only the storage dimension is padded to a multiple of 32, with the padded rows of LoRA-A and columns of LoRA-B constrained to zero. Each grouped operation accepts an independent reduction or output dimension for every adapter, allowing projections with different QSpan-assigned ranks and output dimensions to execute together. Thus, storage padding does not alter QSpan's logical ranks, scaling factors, or total rank budget.

Native low-precision backward. Each output gradient is quantized to NVFP4 and consumed by the NVFP4 DGRAD GEMMs of both branches. LoRA WGRAD is computed from NVFP4 activation and gradient operands, making the GEMMs themselves A4G4 operations, with the resulting gradients materialized at the optimizer boundary. To avoid retaining a second frozen-weight copy for DGRAD, a custom CUDA kernel reframes the packed E2M1 values and scale factors into the required transposed layout. This reframing overlaps with output-gradient quantization on a separate CUDA stream, and the main stream waits only at the dependent DGRAD GEMM. For grouped QKV and gate/up operations, native grouped DGRAD kernels compute the backbone input gradients, and the LoRA contribution is accumulated in the reference BF16 order. WGRAD is streamed by projection, with local plans and transpose buffers released after their final use.

Packed activation saves and selective recomputation. Eligible BF16 tensors saved by autograd are replaced with row-wise NVFP4 tensors and dequantized only when consumed during backward. Custom projection operators retain their packed inputs directly, avoiding duplicate packing through the generic saved-tensor mechanism. Packed-tensor reuse is keyed by CUDA storage and tensor metadata, including the tensor version, to prevent stale reuse under allocator recycling. For standard decoder attention in Llama and Qwen3, a custom autograd operator stores Q, K, V, and the output-projection input in packed NVFP4 buffers. During backward, Q, K, and V are reconstructed and only the attention core is recomputed, avoiding both BF16 retention and recomputation of the QKV and output-projection GEMMs. For specialized token mixers, including Qwen3.5 linear attention and Gemma-4 attention, we instead checkpoint the token mixer without checkpointing the complete decoder layer. Overall, one-copy frozen weights, packed activation saves, direct packed LoRA intermediates, attention-core recomputation, and last-use buffer release reduce peak memory. Shared packing, grouped execution, packed-output GEMMs, and asynchronous weight reframing further avoid redundant conversion and execution overhead.

B Derivation of Theorem 1

This section provides the derivation of Theorem 1, details of the rank-allocation procedure, and definitions of the residual metrics. We use $R_X = R_{Q(X)}$ and $R_G = R_{Q(G)}$ for brevity.

Local quadratic approximation. For a perturbation $-E$ of the frozen weight, Taylor expansion gives

$$\mathcal{L}(W - E) \approx \mathcal{L}(W) - \langle \nabla_W \mathcal{L}(W), E \rangle + \frac{1}{2} \operatorname{vec}(E)^\top H_W \operatorname{vec}(E). \tag{8}$$

Because the initialization is constructed as a local perturbation around the frozen pre-trained weight, we focus on the second-order loss variation and approximate the local loss discrepancy using the curvature term. We therefore approximate the local loss discrepancy using the quadratic sensitivity term.

$$\Delta\mathcal{L}(E) \approx \frac{1}{2} \operatorname{vec}(E)^\top F_W \operatorname{vec}(E). \tag{9}$$

Quantized gradient structure and K-FAC. For a linear layer $y = Wx$, the per-sample weight gradient is $\nabla_W \mathcal{L} = gx^\top$. Under fully quantized computation, the corresponding calibration signal is $Q(g)Q(x)^\top$, and hence

$$\begin{aligned} F_W &= \mathbb{E}\left[\operatorname{vec}(Q(g)Q(x)^\top) \operatorname{vec}(Q(g)Q(x)^\top)^\top\right] \\ &= \mathbb{E}\left[Q(x)Q(x)^\top \otimes Q(g)Q(g)^\top\right]. \end{aligned} \tag{10}$$

The K-FAC approximation (Martens & Grosse, 2015) separates the activation and gradient factors:

$$F_W \approx \mathbb{E}[Q(x)Q(x)^\top] \otimes \mathbb{E}[Q(g)Q(g)^\top]. \tag{11}$$

We use the damped empirical estimates R_X and R_G defined in equation 2, yielding the practical approximation $F_W \approx R_X \otimes R_G$. Substituting this approximation into equation 9 and using $\mathrm{vec}(AZC) = (C^\top \otimes A)\,\mathrm{vec}(Z)$ yields

$$\begin{aligned} \Delta\mathcal{L}(E) &\approx \frac{1}{2}\,\mathrm{vec}(E)^\top (R_X \otimes R_G)\,\mathrm{vec}(E) \\ &= \frac{1}{2}\,\mathrm{vec}(E)^\top\,\mathrm{vec}(R_G E R_X) \\ &= \frac{1}{2}\,\mathrm{Tr}(E^\top R_G E R_X). \end{aligned} \tag{12}$$

Thus substituting $E = \Delta W - BA$ and dropping the common factor $1/2$ for optimization gives equation 4.

Two-sided transformation. Because $R_G, R_X \succ 0$,

$$\begin{aligned} \mathrm{Tr}(E^\top R_G E R_X) &= \mathrm{Tr}\Big[(R_G^{1/2} E R_X^{1/2})^\top (R_G^{1/2} E R_X^{1/2})\Big] \\ &= \|R_G^{1/2} E R_X^{1/2}\|_F^2. \end{aligned} \tag{13}$$

Define $\widetilde{B} = R_G^{1/2} B$, $\widetilde{A} = A R_X^{1/2}$, and $\widetilde{\Delta W} = R_G^{1/2} \Delta W R_X^{1/2}$. Since the transformations are invertible, they preserve rank and give

$$\min_{\mathrm{rank}(\widetilde{B}\widetilde{A}) \le r} \|\widetilde{\Delta W} - \widetilde{B}\widetilde{A}\|_F^2. \tag{14}$$

Truncated SVD and factor recovery. For $\widetilde{\Delta W} = U \Sigma V^\top$, the Eckart–Young–Mirsky theorem (Eckart & Young, 1936; Mirsky, 1960) states that the best rank-r Frobenius approximation is

$$\widetilde{M}^\star = U_r \Sigma_r V_r^\top. \tag{15}$$

Mapping back gives

$$(BA)^\star = R_G^{-1/2} U_r \Sigma_r V_r^\top R_X^{-1/2}. \tag{16}$$

Using $\Sigma_r = \Sigma_r^{1/2} \Sigma_r^{1/2}$, one valid balanced factorization is

$$B^\star = R_G^{-1/2} U_r \Sigma_r^{1/2}, \qquad A^\star = \Sigma_r^{1/2} V_r^\top R_X^{-1/2}, \tag{17}$$

which proves Theorem 1.

Model	Precision	Frozen Weights	Trainable Params	Optimizer States	Staged Activations	Temporary Buffers	Peak Memory
LLaMA-3.2-3B	BF16	5.985	0.182	0.181	29.407	7.845	43.600
	NVFP4	2.946	0.261	0.181	8.222	11.206	**22.816**
Qwen3-4B	BF16	8.270	0.246	0.246	45.904	9.291	63.957
	NVFP4	3.375	0.354	0.246	8.807	13.964	**26.747**
Qwen3.5-2B	BF16	3.505	0.126	0.125	48.073	15.174	67.003
	NVFP4	1.668	0.181	0.125	12.962	18.849	**33.785**
Gemma4-E2B	BF16	9.523	0.180	0.180	48.774	16.017	74.674
	NVFP4	7.037	0.259	0.180	21.136	19.642	**48.254**

Table 7: Peak training-memory breakdown across language models. BF16 uses BF16 computation for both the frozen backbone and LoRA branch, whereas NVFP4 applies W4A4G4 computation to both branches. All experiments use a batch size of 16, sequence length of 512, four gradient-accumulation steps, and a uniform LoRA rank of 32. All memory values are reported in GB.

Model	Precision	Frozen Weights	Trainable Params	Optimizer States	Staged Activations	Temporary Buffers	Peak Memory
FLUX.1-dev	BF16	22.701	1.027	1.026	6.949	0.135	31.838
	NVFP4	6.395	1.030	1.028	1.901	0.412	**10.765**
FLUX.2-Klein-4B	BF16	7.394	0.518	0.511	4.648	0.091	13.160
	NVFP4	2.087	0.513	0.511	1.325	0.394	**4.830**

Table 8: Peak training-memory breakdown across diffusion models. BF16 uses BF16 computation for both the frozen backbone and LoRA branch, whereas NVFP4 applies W4A4G4 computation to both branches. All experiments use a batch size of 1, training resolution is 512×512, four gradient accumulation steps, and a uniform LoRA rank of 128. All memory values are reported in GB.

Frozen Component	BF16	NVFP4	Reduction
Token embedding / tied LM head	0.9470	0.9470	0.0%
MLP projections	1.6880	0.4750	71.9%
Linear-attention projections	0.7050	0.1990	71.8%
Full-attention projections	0.1640	0.0460	72.0%
Linear-attention conv. / state	0.0008	0.0008	0.0%
Norms and small buffers	0.0002	0.0002	0.0%
Allocator padding	∼0.0000	∼0.0001	–
Actual frozen allocation	**3.505**	**1.668**	**52.4%**

Table 9: Frozen-weight memory breakdown for Qwen3.5-2B. NVFP4 reduces the memory of quantized projection weights, while the token embedding, tied LM head, linear-attention state, and normalization parameters remain in higher precision. All memory values are reported in GB.

Frozen Component	FLUX.1-dev			FLUX.2-Klein-4B		
	BF16	NVFP4	Reduction	BF16	NVFP4	Reduction
Conditioning embeddings	0.062	0.017	72.5%	0.290	0.081	72.1%
Input projections	0.024	0.007	70.8%	0.046	0.013	71.7%
Double-stream modulation/norms	4.105	1.156	71.8%	–	–	–
Double-stream attention	2.737	0.770	71.9%	0.720	0.203	71.8%
Double-stream image FFN	2.737	0.770	71.9%	0.810	0.228	71.9%
Double-stream text FFN	2.737	0.770	71.9%	0.810	0.228	71.9%
Single-stream blocks	10.262	2.888	71.9%	4.680	1.316	71.9%
Output norm/projection	0.036	0.010	72.2%	0.037	0.010	73.0%
Allocator rounding/unmapped	0.001	0.007	–	0.002	0.008	–
Actual frozen allocation	**22.701**	**6.395**	**71.8%**	**7.394**	**2.087**	**71.8%**

Table 10: Frozen-weight memory breakdown for FLUX.1-dev and FLUX.2-Klein-4B. NVFP4 reduces the memory of quantized projection weights, while conditioning embeddings, normalization parameters, and other numerically sensitive components remain in higher precision. All memory values are reported in GB.

Model	LoRA BF16	LoRA NVFP4	QSpan NVFP4	Memory Overhead
LLaMA-3.2-3B	43.600	22.816	**23.247**	+1.89%
Qwen3-4B	63.957	26.747	**27.375**	+2.35%
Qwen3.5-2B	67.003	33.785	**34.146**	+1.07%
Gemma4-E2B	74.674	48.254	**49.011**	+1.57%

Model	LoRA BF16	LoRA NVFP4	QSpan NVFP4	Memory Overhead
FLUX.1-dev	31.838	10.773	**11.341**	+5.27%
FLUX.2-Klein-4B	13.160	4.834	**5.039**	+4.24%

Table 11: Peak training memory with QSpan rank allocation. The left table reports language models and the right table reports diffusion models. LoRA BF16 uses BF16 computation for both the frozen backbone and LoRA branch. LoRA NVFP4 uses W4A4G4 computation with a uniform rank of 32 for language models and 128 for diffusion models, whereas QSpan NVFP4 redistributes the same logical rank budget across layers. Memory overhead is measured relative to uniform-rank NVFP4 LoRA. All memory values are reported in GB.

C Training Memory Efficiency and Analysis

Tables 7 and 8 reports the peak training-memory breakdown across the language models and text-to-image diffusion models, respectively. Across both model families, fully quantizing both the frozen backbone and LoRA branch with NVFP4 substantially reduces peak memory compared with BF16 training. For language models, peak memory decreases from 43.600–74.674 GB to 22.816–48.254 GB across the evaluated architectures. The largest reduction comes from staged activations, as the NVFP4 pipeline stores eligible activations and LoRA intermediates in packed low precision and selectively recomputes attention states during backward. NVFP4 computation requires somewhat larger temporary buffers for quantization, packed GEMMs, and kernel workspaces, but this overhead is considerably smaller than the resulting savings in frozen weights and staged activations. The same execution format extends these memory savings to diffusion models, reducing peak memory from 31.838 GB to 10.773 GB for FLUX.1-dev and from 13.160 GB to 4.834 GB for FLUX.2-Klein-4B.

Tables 9 and 10 further show how the frozen-weight savings arise in the two model families. The frozen-weight reduction is smaller than the nominal difference between 16-bit and 4-bit storage because numerically sensitive components, particularly the token embedding and tied LM head, remain in BF16. As illustrated by the Qwen3.5-2B breakdown in Table 9, NVFP4 substantially compresses the linear projection weights, whereas the unquantized embedding and output head account for a larger fraction of the remaining allocation. The resulting reduction therefore varies across architectures according to the relative sizes of these components. For FLUX.1-dev and FLUX.2-Klein-4B, the same representation consistently reduces the memory of the major projection groups, including their attention, feed-forward, and single-stream blocks. This reduces the total frozen allocation from 22.701 GB to 6.395 GB for FLUX.1-dev and from 7.394 GB to 2.087 GB for FLUX.2-Klein-4B, corresponding to a 71.8% reduction for both models. The consistent reductions across these structurally different components show that the NVFP4 pipeline captures the dominant projection-weight memory in both transformer language models and diffusion transformers.

Table 11 examines whether QSpan's non-uniform rank allocation preserves these memory benefits. QSpan redistributes the same logical rank budget used by uniform NVFP4 LoRA, with a baseline rank of 32 for language models and 128 for diffusion models. Across the four language models, its peak-memory overhead remains between 1.07% and 2.35%. The corresponding overhead is 5.27% for FLUX.1-dev and 4.24% for FLUX.2-Klein-4B. Thus, QSpan retains the substantial reduction achieved by fully quantized LoRA across both model families while providing training-aware allocation of the available adapter capacity. Together, Tables 7–11 show that the same end-to-end W4A4G4 execution format supports language and diffusion models, and that QSpan preserves its memory-efficient operation under model-dependent non-uniform rank assignments.

D Implementation Details of QSpan

Algorithm 1 summarizes the complete QSpan procedure, including calibration-based covariance estimation, training-aware closed-form initialization, and bounded rank allocation under a fixed total rank budget. The following paragraphs provide the detailed training and evaluation settings for each task.

Commonsense reasoning. Following SVFT (Lingam et al., 2024) and prior PEFT studies, we fine-tune each model on Commonsense-15K for three epochs and evaluate it on BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SocialIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy, ARC-Challenge (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018). We apply LoRA to all linear layers with rank $r = 32$, scaling factor $\alpha = 64$, learning rate 3×10^{-4}, dropout 0.05, maximum sequence length 512, batch size 16, and four gradient accumulation steps. For QSpan, EoRA (Liu et al., 2024), and other methods requiring calibration data for LoRA initialization, we use the same 128 samples drawn from Commonsense-15K as calibration data. QSpan non-uniformly distributes ranks across layers while preserving the same total rank budget as the uniform $r = 32$ configuration. We report the average accuracy across the eight evaluation benchmarks.

Mathematical reasoning. We follow the training and evaluation protocol of SVFT (Lingam et al., 2024), fine-tuning models on MetaMathQA-40K (Yu et al., 2023) and evaluating them on

GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). We use LoRA rank $r = 32$, scaling factor $\alpha = 64$, maximum sequence length 512, batch size 16, and four gradient accumulation steps. For QSpan, EoRA (Liu et al., 2024), and other methods requiring calibration data for LoRA initialization, we use the same 128 samples drawn from MetaMathQA-40K as calibration data. QSpan uses the same total rank budget as the uniform-rank baseline but allocates ranks non-uniformly across layers. All remaining training and evaluation settings follow SVFT, and we report the average accuracy across GSM8K and MATH.

Personalized language modeling. We evaluate QSpan on LaMP-2 movie tagging and LaMP-4 news headline generation, for which sufficient training and evaluation data are available. We adopt the official OPPU implementation (Tan et al., 2024) and follow its two-stage personalization pipeline, first training a task LoRA for each LaMP task and then performing user-specific fine-tuning. For both tasks, we use LoRA rank $r = 32$, scaling factor $\alpha = 64$, and learning rate 1×10^{-4}. QSpan preserves the same total rank budget while allocating ranks non-uniformly across layers. All remaining hyperparameters follow OPPU. To avoid impractical per-user calibration, we initialize QSpan only once per task using 128 samples drawn from the multi-user training data used to train the corresponding task LoRA. We then reuse the same initialized QSpan for user-specific fine-tuning and evaluation across all 100 users, without additional calibration for individual users. This task-level initialization provides a practical personalization setting by avoiding repeated initialization and access to separate user-specific calibration data. We report Accuracy and F1 for LaMP-2, and ROUGE-1 and ROUGE-L (Lin, 2004) for LaMP-4.

Text-to-image personalization. We evaluate QSpan on FLUX.1-dev and FLUX.2-Klein-4B using DreamBooth (Ruiz et al., 2023), with CustomConcept101 (Kumari et al., 2023) as the calibration set. For each subject, we train with the class-specific instance prompt, "a photo of sks *class*," for 500 steps at 512×512 resolution using AdamW, a learning rate of 1×10^{-4}, a batch size of 1, and four gradient accumulation steps. The LoRA rank and alpha are both 128. All calibration-based initialization methods use the same CustomConcept101 samples for a fair comparison. During fully quantized PEFT, both the frozen backbone and trainable LoRA branch use NVFP4 W4A4G4 computation, while only the LoRA parameters are optimized. We evaluate on the 25 object-oriented DreamBooth prompts covering diverse scenes, backgrounds, attributes, and compositions. We generate four 1024×1024 images per prompt, yielding 100 images per subject and 3,000 images per method and model across 30 subjects. We report DINO and CLIP image similarity (CLIP-I) against the subject instance images, along with CLIP text similarity (CLIP-T) to the corresponding prompts.

E Related Work

Fully quantized low-precision training. Fully quantized training reduces the cost of both forward and backward propagation by quantizing weights, activations, and gradients. Early studies demonstrated accurate 4-bit training through careful scaling and outlier handling (Xi et al., 2023; Chmiel et al., 2021). Recent work builds on microscaling floating-point formats (Rouhani et al., 2023), including NVFP4 (NVIDIA et al., 2025), and improves low-precision training through fine-grained scaling, randomized transformations, gradient estimation, spectral partitioning, outlier control, and selective high-precision computation (Cao et al., 2026; Castro et al., 2026; Chen et al., 2025; Panferov et al., 2026; Cook et al., 2025; Dong et al., 2026; Kim et al., 2026). This scope has been further expanded to optimizer states, optimizer computation, and selected attention paths while retaining a compact projection subspace in BF16 (Ding et al., 2026). These approaches primarily target pre-training or full fine-tuning, where all model parameters can adapt to quantization errors, and focus on improving how low-precision tensors are represented. In contrast, QSpan addresses fully quantized PEFT, where the frozen backbone leaves the remaining quantization errors to be compensated by quantized, rank-constrained adapters.

Low-rank adaptation of quantized models. Prior work combines quantized backbones with low-rank components to improve memory efficiency or compensate for weight-quantization errors (Dettmers et al., 2023; Li et al., 2024b; Xu et al., 2024; Liao et al., 2024; Huang et al., 2024). These methods jointly design quantization and adaptation, initialize trainable adapters from the weight-quantization error, or modify the model representation to reduce difficult-to-quantize components. Another line of work constructs low-rank components from calibration statistics to

Algorithm 1 QSpan with bounded rank allocation

Require: Frozen weights $\{W_l\}$, quantizer Q, calibration tensors, uniform rank r, floor $r_{\min}$, caps $\{\mathrm{cap}_l\}$
Ensure: Integer ranks $\{r_l\}$ and factors $\{B_l^\star, A_l^\star\}$

$R_{\mathrm{total}} \leftarrow Lr$
for $l = 1, \ldots, L$ **do**
  Compute $R_{X,l}$ and $R_{G,l}$ from $Q(X_l)$ and $Q(G_l)$
  Compute $U_l \Sigma_l V_l^\top \leftarrow \mathrm{SVD}(R_{G,l}^{1/2}(W_l - Q(W_l))R_{X,l}^{1/2})$
  if $\sum_i \sigma_{l,i}^2 = 0$ **then**
    $w_l \leftarrow 0$
  else
    $w_l \leftarrow 1 - (\sum_{i=1}^{r} \sigma_{l,i}^2)/(\sum_i \sigma_{l,i}^2)$
  end if
end for
if $\sum_l w_l = 0$ **then**
  $r_l \leftarrow r$ for all l
else
  $q_l \leftarrow R_{\mathrm{total}} w_l / (\sum_m w_m)$ and bounded-round each q_l
  Apply unit increments/decrements within $[r_{\min}, \mathrm{cap}_l]$ until $\sum_l r_l = R_{\mathrm{total}}$
end if
for $l = 1, \ldots, L$ **do**
  $B_l^\star \leftarrow R_{G,l}^{-1/2} U_{l,r_l} \Sigma_{l,r_l}^{1/2}$
  $A_l^\star \leftarrow \Sigma_{l,r_l}^{1/2} V_{l,r_l}^\top R_{X,l}^{-1/2}$
end for
return $\{r_l, B_l^\star, A_l^\star\}_{l=1}^{L}$

reconstruct layer outputs or isolate residual components before quantization (Liu et al., 2024; Li et al., 2024a; Zhang et al., 2024; 2025). Despite their different formulations, these approaches select low-rank directions using the weight-quantization error alone or its interaction with input activations. Related approaches reduce buffered-activation memory through hybrid compression and recomputation or support W4A4G4 diffusion post-training using a quantized backbone together with a full-precision stabilizer and trainable LoRA branch (Wang et al., 2025; Xue et al., 2026). QSpan additionally incorporates quantized output gradients, selecting residual directions according to their contribution to the training-weighted quantization error under fully quantized PEFT.

Adaptive rank allocation. Adaptive rank methods distribute LoRA capacity non-uniformly across layers based on importance estimates obtained during or before fine-tuning (Zhang et al., 2023; Kim et al., 2024). Existing approaches determine layer-wise ranks from optimization dynamics or the singular value spectrum of the weight-quantization error. QSpan derives the layer-wise ranks from the singular value spectrum of the training-weighted quantization error, which incorporates both quantized input activations and quantized output gradients. The same residual decomposition determines both the subspace represented by each adapter and the distribution of the fixed rank budget across layers.

F Additional Experiments

Residual Objective Metrics For each target module l, we define the weight-quantization residual and the uncompensated residual after initialization as

$$\Delta W_l = W_l - Q(W_l), \qquad E_l = \Delta W_l - B_l A_l. \tag{18}$$

Given N calibration samples, let $x_{q,l,n}$ denote the quantized input activation to module l and let $g_{\mathrm{ref},l,n}$ denote its reference output gradient for sample n. We define three empirical residual metrics

Method	FLUX.1-dev			FLUX.2-Klein-4B		
	WR ($\times 10^{-6}$)	QAWR ($\times 10^{-6}$)	TWR ($\times 10^{-19}$)	WR ($\times 10^{-6}$)	QAWR ($\times 10^{-6}$)	TWR ($\times 10^{-19}$)
LoftQ	**7.096**	11.455	7.098	**15.938**	2.769	13.449
EoRA	7.367	**1.928**	1.419	16.879	1.067	1.473
QSpan	7.345	1.977	**1.402**	16.780	**1.030**	**1.412**
QSpan-RandG	7.345	1.977	1.416	16.780	**1.030**	1.487

Table 12: Residual objective analysis on FLUX.1-dev and FLUX.2-Klein-4B.

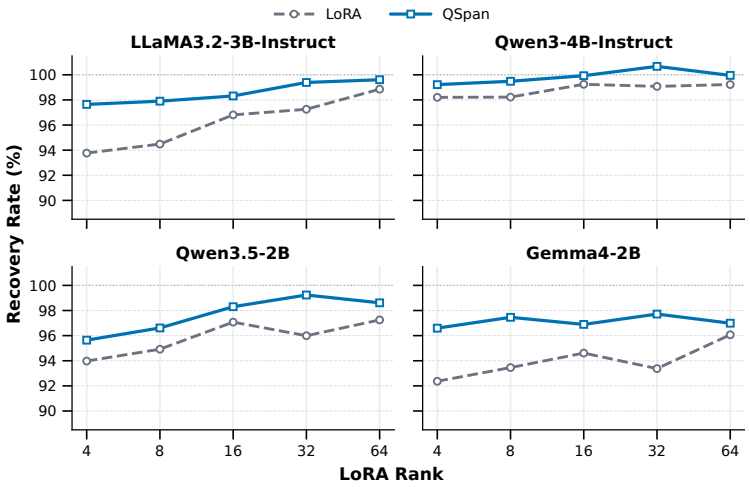


Figure 7: Rank-budget analysis of LoRA and QSpan across different models and rank budgets. QSpan consistently preserves a larger fraction of the effective low-rank subspace, with particularly clear advantages under constrained rank budgets.

with progressively stronger notions of training relevance:

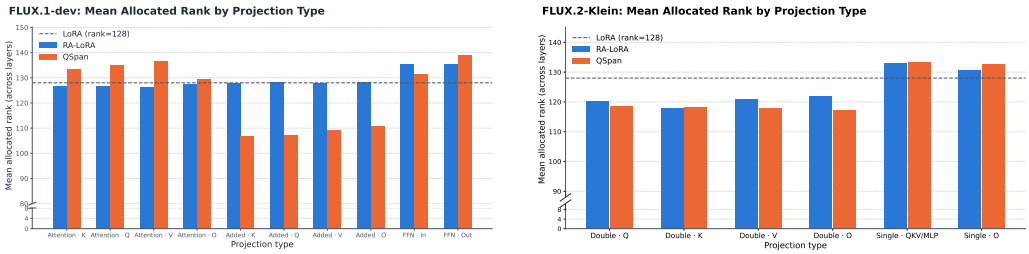
$$\mathrm{WR}(E_l) = \|E_l\|_F^2, \tag{19}$$

$$\mathrm{QAWR}(E_l) = \frac{1}{N}\sum_{n=1}^{N} \|E_l x_{q,l,n}\|_2^2, \tag{20}$$

$$\mathrm{TWR}(E_l) = \frac{1}{N}\sum_{n=1}^{N} \left(g_{\mathrm{ref},l,n}^{\top} E_l x_{q,l,n}\right)^2. \tag{21}$$

WR measures the uncompensated residual purely in weight space. QAWR additionally measures the output perturbation induced when the residual is exposed through quantized input activations. TWR further weights this perturbation according to its alignment with loss-sensitive output directions represented by the reference gradients. TWR is therefore the sample-level empirical counterpart of the training-aware loss discrepancy in equation 3 and provides a practical measure of the remaining compensation demand.

For each metric, we report the global sum across target modules normalized by the total number of target-module parameters N_{param}, rather than averaging layer-wise normalized metrics. Table 12 reports the residual metrics on FLUX.1-dev and FLUX.2-Klein-4B. Consistent with its design objective, EoRA achieves the lowest activation-weighted residual (QAWR) across both diffusion backbones, reflecting its emphasis on activation-aware reconstruction. By contrast, QSpan achieves the lowest training-weighted residual (TWR) on both models, indicating that incorporating output-gradient information more effectively prioritizes loss-relevant quantization errors. Although QSpan does not explicitly optimize WR or QAWR, it produces the smallest residual under the proposed training-aware criterion. This observation is consistent with the improved personalization performance reported in Section 4.



(a) FLUX.1-dev (b) FLUX.2-Klein-4B

Figure 8: Mean rank allocated to each projection type by QSpan and RA-LoRA on text-to-image diffusion models. QSpan produces different allocation patterns across model architectures by distributing a fixed total rank budget according to the layer-wise training-weighted quantization-error spectrum.

Rank-budget analysis. Fig. 7 compares QSpan and uniform LoRA across different rank budgets. QSpan preserves a larger fraction of the effective low-rank subspace across the evaluated models and ranks. The improvement remains consistent as the available adaptation capacity changes, demonstrating that QSpan is not tied to a particular default rank. These results further show that allocating limited capacity to training-relevant residual directions is especially beneficial under constrained rank budgets.

Rank Allocation Analysis Fig. 8 compares the ranks allocated by QSpan and RA-LoRA across projection types in the evaluated diffusion models. QSpan produces model-dependent allocation patterns according to the training-weighted quantization-error spectrum of each layer, rather than applying a fixed preference for specific projection types. All methods use the same total rank budget.

Qualitative results of text-to-image personalization. Fig. 9 and Fig. 10 provide additional qualitative comparisons on the DreamBooth dataset using FLUX.1-dev and FLUX.2-Klein-4B, respectively. Compared with other methods, QSpan generally achieves better subject fidelity and text alignment under fully quantized PEFT. Notably, QSpan produces qualitatively better results than LoRA (BF16) in some cases, suggesting that its training-aware initialization and rank allocation can improve adaptation quality beyond merely compensating for quantization errors.

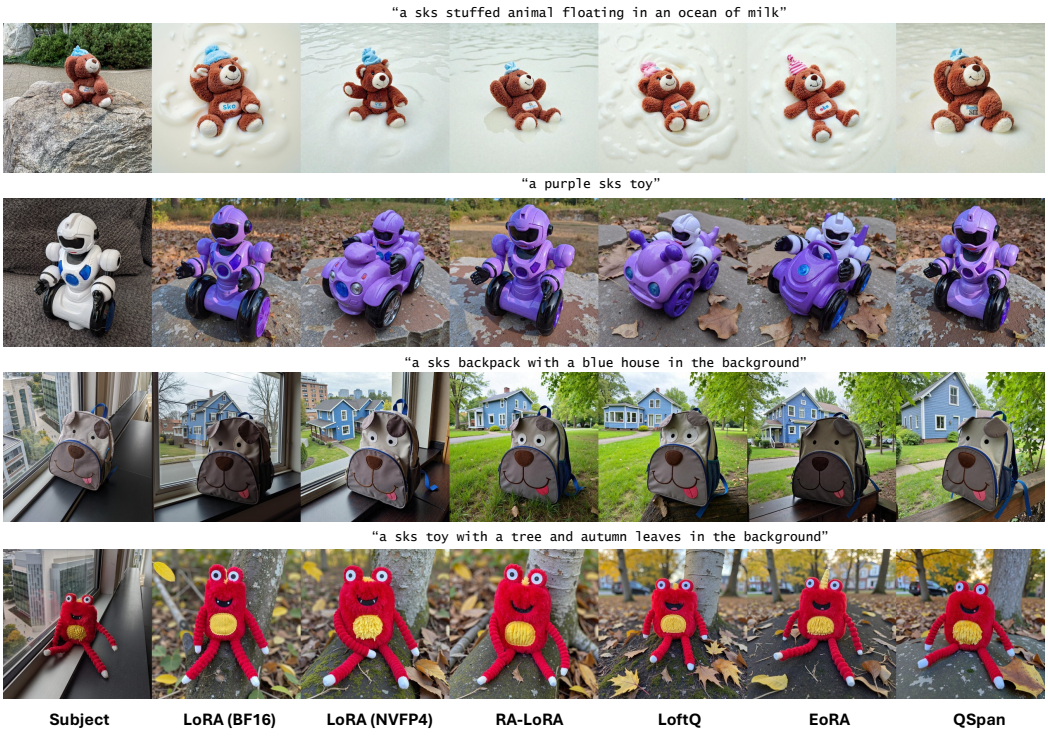


Figure 9: Additional qualitative comparison of FLUX.1-dev on the DreamBooth dataset.

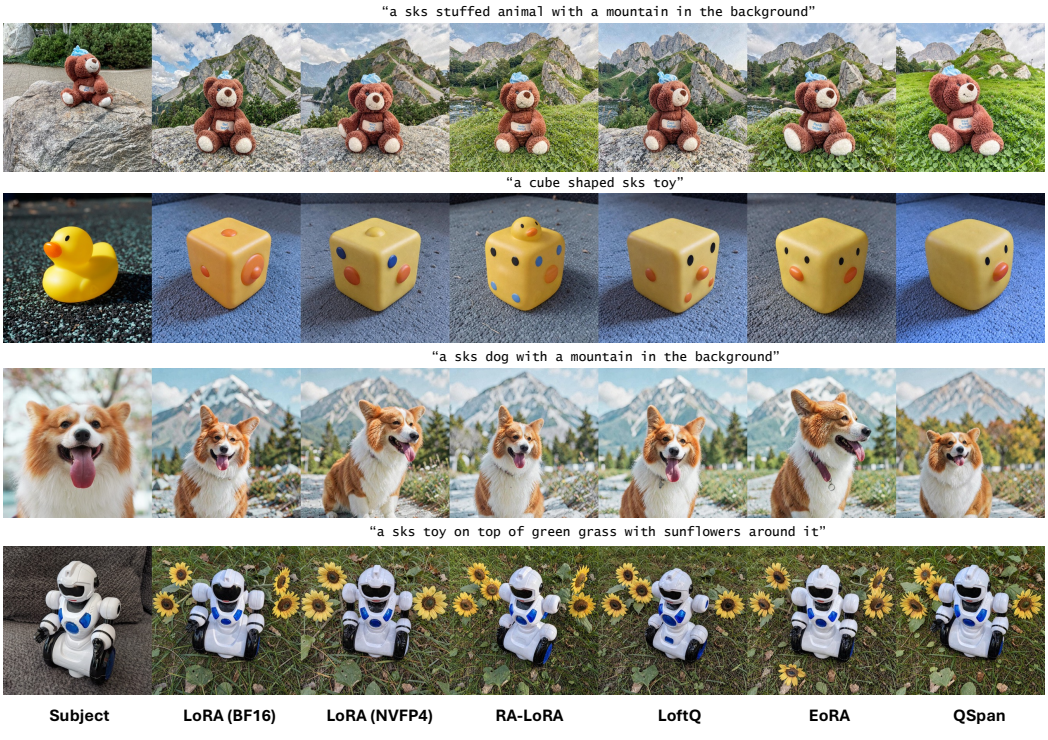


Figure 10: Additional qualitative comparison of FLUX.2-Klein-4B on the DreamBooth dataset.