Cross-Version Δ-Transfer of QLoRA-Trained Adapters

Yoshiki Kubota, Hirotada Honda, Takashi Sano, Shugo Nakamura
Toyo University, INIAD
Tokyo, Japan
{s3f102500070, honda.hirotada, takashi.sano, shugo.nakamura}@iniad.org

Abstract

To investigate whether QLoRA-trained adapters transfer across model versions like LoRA-trained adapters, we train LoRA and QLoRA adapters on Llama-3-8B-Instruct and transfer them to Llama-3.1-8B-Instruct with a task-specific merge scale λ. Both methods are evaluated on a BF16 target base, isolating the training-time precision difference; NF4 deployment is evaluated separately. Across 20 main tasks, the absolute evaluation-score gap between QLoRA Transfer and LoRA Transfer is at most 0.054 on all tasks and at most 0.03 on 18/20 tasks, so training the adapter on a quantized base does not compromise its cross-version transferability. In a two-task deployment case study, serving QLoRA adapters with an NF4 base uses 5.7–10.0 GB peak allocated inference VRAM, compared with 15.1–19.6 GB for LoRA-merged BF16. Across 44 settings (22 tasks $\times$ 2 methods), the task-optimal λ consistently exceeds the Frobenius-distance optimum (we call this discrepancy the λ gap), suggesting that weight-space proximity alone is insufficient for identifying it. Replicating the pipeline on two further families (Mistral v0.1→v0.3, Qwen2→2.5) reproduces these findings. We also report a negative case: on CommonsenseQA, QLoRA Transfer falls significantly below direct adaptation.

1 Introduction

Fine-tuning large language models (LLMs) for downstream tasks typically demands substantial compute. LoRA (Hu et al., 2022) reduces the number of trainable parameters to about 0.1% via low-rank factorization, and QLoRA (Dettmers et al., 2023) further enables fine-tuning on consumer-grade GPUs by quantizing the base model to 4-bit (NF4). Today, **QLoRA is the de facto standard workflow** for resource-constrained researchers and developers to adapt large models without full-precision fine-tuning.

In parallel, growing attention has been paid to studies that treat the fine-tuning weight difference (ΔW) as a reusable "task vector," including Task Arithmetic (Ilharco et al., 2023), as well as work that directly transfers this delta between different base-model versions, such as ParamΔ (Cao et al., 2025) and Fine-tuning Transfer (Lin et al., 2025). These efforts strongly support the premise that a learned ΔW is reusable beyond its training environment, and they form the conceptual basis for adapter-composition methods such as LoraHub (Huang et al., 2024) and LoRA Soups (Prabhakar et al., 2025).

However, all of these Δ-transfer studies share one implicit assumption: **the adapter to be transferred is trained on a full-precision (BF16) base model**. LoraHub, LoRA Soups, ParamΔ, and Fine-tuning Transfer all consider deltas trained on BF16 bases, leaving systematically unverified the transferability of QLoRA adapters—that is, BF16 LoRA adapters trained while the base model is held in NF4 and dequantized to BF16 only at the forward/backward matmul[1]. This is a practically meaningful, unexplored territory: many researchers choose QLoRA precisely because BF16 training is infeasible under their compute budget, and whether the resulting adapter can be transferred to a different version of the base model is therefore a question of high practical relevance.

The main goal of this paper is to systematically demonstrate that **QLoRA adapters can be the subject of Δ-transfer in the same way that LoRA adapters can** (here verified within the Llama-3 family: Llama-3-8B-Instruct NF4 → Llama-3.1-8B-Instruct BF16). This is an empirical result that broadens the scope of adapter-transfer research

[1]In QLoRA, the base weights themselves are stored in NF4 and frozen, and are expanded to BF16 only immediately before the forward/backward matmul. The LoRA matrices A and B are stored in BF16, and gradient updates apply only to this BF16 adapter. Therefore, the ΔW that we transfer in this work remains BF16 throughout.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 30380–30401
October 24-29, 2026 ©2026 Association for Computational Linguistics

from BF16-trained adapters to QLoRA-trained adapters. Concretely, we compose the QLoRA-trained BF16 adapter ΔW_{src} obtained on Llama-3-8B-Instruct (NF4) into the BF16 Llama-3.1-8B-Instruct base W_{tgt} via a merge-scaling factor λ, and evaluate on 22 tasks $\times$ 2 methods. In a direct comparison with LoRA Transfer (§5.1), the absolute evaluation-score gap is at most 0.054 on all 20 Group A/B tasks and at most 0.03 on 18 of 20; within our setup, training with QLoRA yields transfer behavior nearly indistinguishable from that of LoRA. For fairness, λ sweep evaluation is performed under matched conditions (BF16 target base) for both methods; when QLoRA-trained adapters are deployed under the standard QLoRA inference workflow (NF4 base + runtime adapter scaling), the **post-load steady-state inference VRAM is 5.7 GB for CommonGen (BSZ=8) and 10.0 GB for CSQA (BSZ=64)**, a 30–60% reduction relative to LoRA-merged BF16 ($\sim$15 GB) on the 2 deployment tasks.

As a secondary analysis, we examine how the choice of λ affects transfer performance and find that the task-optimal λ^* consistently exceeds the Frobenius-distance optimum λ_{opt} across all 44 settings (mean ratio $4.35\times$) — we call this discrepancy the λ *gap* — suggesting that simple weight-space proximity to the Direct Adaptation (DA) adapter is insufficient for choosing λ; full details (44-setting consistency, a 34-setting DA λ-scan control, and the weak correlation between aggregate cosine and λ^*) are reported in §5.2.

The main contributions of this paper are: (i) the systematic demonstration that BF16 adapters trained under QLoRA on an NF4 base can be Δ-transferred to a BF16 base of a different version (22 tasks $\times$ 2 methods; absolute gap ≤ 0.054 on all 20 Group A/B tasks, ≤ 0.03 on 18/20); (ii) a 2-task deployment case study showing that the same QLoRA adapters served under the standard NF4 inference workflow reach 5.7–10.0 GB post-load steady-state inference VRAM versus $\sim$15 GB for LoRA-merged BF16; and (iii) empirical λ findings indicating that the task-optimal λ is not determined by weight-space proximity to ΔW_{DA} alone.

2 Related Work

This work is situated at the intersection of (i) Task Arithmetic and model merging, (ii) Cross-model / cross-version transfer, and (iii) Quantization-aware fine-tuning. We review the principal prior work in each of these three areas and clarify our positioning via the comparison in Table 1.

2.1 Task Arithmetic and Model Merging

Task Arithmetic (Ilharco et al., 2023) established the fine-tuning weight difference as a reusable "task vector," and subsequent work explored merging of 16-bit-trained adapters: Model Soups (Wortsman et al., 2022), TIES-Merging (Yadav et al., 2023), DARE (Yu et al., 2024), and LoRA-specific variants LoraHub (Huang et al., 2024), LoRA Soups (Prabhakar et al., 2025), Adaptive LoRA Merge (Miyano and Arase, 2025). These methods all target merging of BF16-trained adapters, distinct from the cross-version transfer of QLoRA-trained adapters that we study.

2.2 Cross-Model and Cross-Version Transfer

Work on transferring a trained delta to a different base-model version has accelerated. Fine-tuning Transfer (Lin et al., 2025) shows that high transferability arises when source and target bases are linearly connected in weight space. ParamΔ (Cao et al., 2025) directly transfers a delta fine-tuned on Llama 3.0 to Llama 3.1 and attains a high retention rate without additional training. More recently, Cross-LoRA (Xia et al., 2025) targets data-free transfer of LoRA adapters across *heterogeneous* LLM families (e.g., LLaMA, Qwen, Gemma) via subspace alignment—an orthogonal axis to ours (cross-family vs cross-precision/version). These studies share the implicit assumption that the delta is trained on a BF16 base; Trans-LoRA (Wang et al., 2024) studies data-free transfer of LoRA modules when the base model changes, aligning a student adapter via synthetic data; our setting differs in that the transfer is across *versions* of the same architecture and the delta being transferred was itself trained on a *quantized* base. Our work is, to our knowledge, the first to systematically verify the transferability of BF16 adapters trained on a 4-bit-quantized base.

2.3 Quantization-Aware Fine-Tuning and Quantized Deltas

QLoRA (Dettmers et al., 2023), building on LLM.int8() (Dettmers et al., 2022), enables fine-tuning by keeping the base in NF4 while training BF16 LoRA adapters. A related line makes low-rank adaptation itself quantization-aware: QA-LoRA (Xu et al., 2024) balances the degrees of freedom of quantization and adaptation so that the

merged model remains quantized, and LoftQ (Li et al., 2024) jointly chooses the quantized backbone and the low-rank initialization to reduce the discrepancy from full-precision fine-tuning. These target the training-time interaction between quantization and adaptation on a *single* base model, whereas we ask whether a delta obtained under such a setup remains transferable to a *different* base. A separate line of work—TVQ (Kim et al., 2025), BitDelta (Liu et al., 2024)—quantizes the *delta* itself post-hoc for storage efficiency. These are orthogonal to our question: whether a delta natively trained on a quantized base can be transferred to a different base.

2.4 Positioning

Table 1 positions our work along three axes (training precision, source-target match, λ-scaling protocol) and highlights the previously unverified cell.

Table 1: Positioning of this work against related work. This work fills the previously unverified cell of cross-version Δ-transfer for QLoRA-trained deltas combined with joint checkpoint (CP) and λ search.

Study	Train	Src-Tgt	λ Search
LoraHub	16-bit	Same	Fixed/learned
LoRA Soups	16-bit	Same	Fixed
ParamΔ	16-bit FT	Cross-ver.	λ=1
FT Transfer	16-bit	Cross-ver.	λ=1
BitDelta	$\rightarrow$1-bit	Same	—
TVQ	$\rightarrow$ q-bit	Same	—
This work	**4-bit QLoRA**	**Cross-prec.**	**Joint (CP, λ)**

3 Method

This section introduces (i) our framework for transferring a QLoRA-trained adapter to a different base-model version, and (ii) an *analytic-vs-empirical* λ *comparison* framework for quantitatively analyzing its properties. The pipeline has two phases: in Phase 1 we perform QLoRA training on the source base (Llama-3.0, NF4); in Phase 2 the resulting BF16 adapter ΔW is composed into the target base of a different version and bit-precision (Llama-3.1, BF16) with a scaling factor λ (for a visual overview, see Appendix A, Figure A.1). Below, we first define the basic transfer equation (§3.1), then describe the evaluation protocol (§3.2), and finally present the analytic-vs-empirical λ comparison framework (§3.3).

3.1 Problem Setup: Cross-Version Δ-Transfer of QLoRA-Trained Adapters

Let the source model M_{src} be the NF4-quantized Llama-3-8B-Instruct (Grattafiori et al., 2024) (i.e., the Llama 3.0 family), with weights $W_{\text{src}} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. Let the target model M_{tgt} be the BF16 Llama-3.1-8B-Instruct, with weights W_{tgt}. The two models share the same architecture (Llama-3 family), and at each projection (q, k, v, o, gate, up, down) of every transformer layer, W_{src} and W_{tgt} have the same shape.

LoRA / QLoRA adapters. Following QLoRA (Dettmers et al., 2023), the weights of M_{src} are frozen, and at each target projection we introduce a low-rank adapter

$$A \in \mathbb{R}^{r \times d_{\text{in}}}, \quad B \in \mathbb{R}^{d_{\text{out}} \times r},$$

with rank $r = 8$ and scaling factor $\alpha = 16$. The weight update represented by the adapter is defined as

$$\Delta W := \frac{\alpha}{r} \cdot BA \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}. \tag{1}$$

Transfer equation. The weight composition that transfers a trained adapter ΔW into M_{tgt} is given by

$$W_{\text{final}} = W_{\text{tgt}} + \lambda \cdot \Delta W, \tag{2}$$

where $\lambda \geq 0$ is the per-task scaling factor (in Experiment 1, $\lambda = 0$ is also included in the grid to recover the base-only setting), and is the central hyperparameter of this work. Prior work such as ParamΔ (Cao et al., 2025) and Fine-tuning Transfer (Lin et al., 2025) fixes $\lambda = 1.0$; in contrast, we systematically analyze the optimization and properties of λ when transferring QLoRA-trained ΔW.

Terminology.

- **Transferred adapter:** the adapter trained on M_{src} and transferred to M_{tgt}, denoted ΔW_{src}.
- **Direct Adaptation (DA):** the comparison baseline—an adapter trained directly on M_{tgt} under the same protocol (LoRA or QLoRA), denoted ΔW_{DA}. We follow the terminology of ParamΔ (Cao et al., 2025).
- **Task-optimal λ^*:** the empirical optimum that maximizes the TestA score.
- **Analytic optimum λ_{opt}:** the closed-form minimizer of $\|\Delta W_{\text{DA}} - \lambda \Delta W_{\text{src}}\|_F$, defined in §3.3.

3.2 Evaluation Protocol: Joint Grid Search over (CP, λ)

To make transfer work, we must choose (i) a training checkpoint (CP) and (ii) a scaling factor λ. In our pilot experiments we initially tried a two-stage search that first fixes the best CP at $\lambda = 1.0$ and then separately sweeps λ; however, the selected CP turned out to be a "CP locally optimized for $\lambda = 1.0$," making it hard to overturn at other λ values (see Appendix M.1 for the 4-setting verification).

We therefore evaluate TestA (the internal validation split; see §4 for the TestA/TestB split design) jointly over the full grid of CPs $\times$ λ values and select the joint best (CP^*, λ^*):

$$(\text{CP}^*, \lambda^*) = \arg\max_{\text{CP},\lambda} \text{Acc}_{\text{TestA}} \big(W_{\text{tgt}} + \lambda\, \Delta W_{\text{src}}(\text{CP})\big) \quad (3)$$

Here $\Delta W_{\text{src}}(\text{CP})$ is the adapter weight at checkpoint CP. The λ grid differs per experiment: Experiment 1 (§4.3) uses a fine-grained 21-value grid $\lambda \in \{0.0, 0.1, 0.2, \ldots, 2.0\}$, while Experiment 2 (§4.4) uses a 5-value smart grid based on the prior knowledge from Experiment 1 (generation: $\{0.7, 1.0, 1.3, 1.6, 2.0\}$; classification: $\{1.0, 1.3, 1.5, 1.7, 2.0\}$). All reported values are TestB scores at (CP^*, λ^*).

TestA-TestB consistency. To verify that the λ selected on TestA generalizes to TestB without overfitting, we compute the Spearman rank correlation ρ of λ rankings at the joint best CP^*. The mean ρ is 0.96 for Group A (generation, $n = 112$ settings) and 0.80 for Group B/C (classification and the new generation task, $n = 124$ settings); $\rho \geq 0.7$ holds for 97/124 (78%) of Group B/C settings, confirming that the λ chosen on TestA is stable on TestB.

3.3 Analytic vs Empirical λ Comparison

To set λ without a search, a natural candidate is the geometric optimum in weight space, which we compare against the empirical optimum found by search.

Analytic and empirical optima. Subject to $\lambda \geq 0$, the closed-form least-squares optimum that brings the *scaled* update $\lambda \Delta W_{\text{src}}$ closest to ΔW_{DA} in Frobenius distance is

$$\lambda_{\text{opt}} = \max\left(0,\ \frac{\langle \Delta W_{\text{DA}}, \Delta W_{\text{src}} \rangle}{\|\Delta W_{\text{src}}\|_F^2}\right) = \cos(\Delta W_{\text{DA}}, \Delta W_{\text{src}}) \cdot \frac{\|\Delta W_{\text{DA}}\|_F}{\|\Delta W_{\text{src}}\|_F}, \quad (4)$$

where $\langle X, Y \rangle$ denotes the Frobenius inner product over flattened adapter weights, $\cos(X, Y) = \langle X, Y \rangle / (\|X\|_F \|Y\|_F)$, and the clip is inactive in all 44 settings (inner products are positive). Geometrically, λ_{opt} is the projection of ΔW_{DA} onto the ray spanned by ΔW_{src}: among all rescalings of the transferred update, the one landing nearest to what direct adaptation would produce. This purely geometric candidate is the λ we single out and test. The empirical optimum $\lambda^* = \arg\max_\lambda \text{Score}_{\text{TestA}}(W_{\text{tgt}} + \lambda \cdot \Delta W_{\text{src}})$ is obtained from the joint search of §3.2.

Checkpoint selection (ensuring fairness). Comparing λ_{opt} and λ^* is meaningful only when both are evaluated for the same ΔW_{src}. On the source side, ΔW_{src} uses the CP^*_{src} from the Transfer joint best; on the target side, ΔW_{DA} uses the adapter at the joint best CP from DA training on the same task. The two are independent runs, so their step numbers do not coincide, but adopting each protocol's own best CP preserves the intended meaning of λ_{opt}—"the λ that comes closest to the most performant DA adapter in the target environment"—and both optima are computed from the *same* ΔW_{src}.

Central question. The question is then whether these two optima agree, i.e., whether $\lambda^* = \lambda_{\text{opt}}$. If $\lambda^* \approx \lambda_{\text{opt}}$, the task-optimal λ can be predicted from this geometric heuristic alone. Conversely, if $\lambda^* \neq \lambda_{\text{opt}}$, scaling along ΔW_{src} can reach a task-optimal point distinct from the DA adapter's location. We answer this question in §5 via a systematic verification on 22 tasks $\times$ 2 methods (44 settings, including the adapter delta analysis).

Note on numerical computation. Frobenius inner products are accumulated in float64 over all 224 submodules; float32 produces catastrophic cancellation (e.g. $\cos > 1$) on ΔW with $\approx 7 \times 10^9$ elements.

4 Experimental Setup

4.1 Tasks and Datasets

We organize the 22 tasks (11 generation + 11 classification) into three groups according to their roles in our experimental design.

Group A — Experiment 1 generation tasks (10): SAMSum, DialogSum, MeQSum, NarrativeQA, MBPP, E2E NLG, SciTLDR, ViGGO, AESLC, WebNLG.

Group B — Experiment 1 accuracy-scored tasks (10): ARC-Challenge / ARC-Easy, BoolQ,

GSM8K, HellaSwag, OpenBookQA, PIQA, PubMedQA, SST-2, WinoGrande.

Group C — Experiment 2 deployment evaluation (2): CommonGen (generation, 1) and CommonsenseQA (classification, 1; hereafter CSQA). Used for the Transfer vs DA comparison in the practical deployment scenario (§4.4).

Sources for all 22 tasks are listed in Appendix B. Together they span generation (summarization, dialogue, Q&A, code, data-to-text) and classification (commonsense and scientific reasoning, sentiment, medical), covering a wide range of domains.

Evaluation Metrics.

- Generation tasks: corpus-level BLEU as primary, BERTScore-F1 as auxiliary, and Exact Match as a reference indicator. BLEU is computed with `sacrebleu` (Post, 2018) using the default tokenization (13a); BERTScore (Zhang et al., 2020) uses `roberta-large` with `rescale_with_baseline=True`.

- Accuracy-scored tasks (Group B): Accuracy as primary, F1-macro as auxiliary (F1-macro is not applicable to GSM8K, which has no label set).

- PubMedQA: a 3-way classification (yes/no/maybe). The strict EM in lm-evaluation-harness (Biderman et al., 2024) renders the Base model unevaluable due to format non-compliance, so we apply a permissive parser based on case-insensitive label-boundary matching uniformly across Base / DA / Transfer.

Decoding configuration. For generation tasks we use greedy decoding (temperature 0, top-p 1.0); max new tokens are set per task as the 95%-tile of reference length $\times 1.2$ (typically 64–256). For classification tasks, we use the loglikelihood-based scoring of lm-evaluation-harness (Biderman et al., 2024), selecting the predicted label by conditional log-likelihood on the option tokens at the prompt end. GSM8K is an exception within Group B: it has no answer options, so we generate a chain-of-thought completion and score the extracted final numeric answer by exact match, using the strict-match filter of lm-evaluation-harness. Full prompt, decoding, and answer-extraction details are given in Appendix C.

Internal validation / held-out evaluation split. For each task, we partition the official test set into two halves by `doc_id % 2`: an **internal validation split** (which we call TestA) used solely for selecting the joint best (CP, λ), and a **held-out evaluation split** (which we call TestB) that is used only for the final reported scores. TestB is never observed during (CP, λ) selection.

We adopt this internal-split design for two reasons that we believe make it the most defensible choice in our setting. (i) **Lack of usable official validation sets on many of our tasks.** Among the 22 tasks, only roughly half (e.g., BoolQ, ARC, HellaSwag, SST-2, PIQA, WinoGrande, PubMedQA) provide a public validation split of sufficient size to support our 21-value λ grid; the other half (e.g., MeQSum, MBPP, SciTLDR, ViGGO, AESLC, E2E NLG) lack a usable validation split, or have one too small to be reliable. Carving a validation split from the training data is feasible but would reduce training data unevenly across tasks—especially severe for small-data tasks such as MeQSum (800) and MBPP (474), where any training-data reduction directly hurts ΔW_{src} quality. (ii) **Uniformity of the evaluation protocol across all 22 tasks.** Mixing protocols (using official validation on some tasks and a train-carved validation on others) would conflate the difference between LoRA Transfer and QLoRA Transfer with the difference between evaluation pipelines, making our main relative comparison less clean. A uniform `doc_id % 2` split across all 22 tasks therefore keeps the LoRA-vs-QLoRA comparison apples-to-apples.

We emphasize that the TestB scores reported in this paper are designed for **within-paper relative comparison** (LoRA Transfer vs QLoRA Transfer vs DA, all evaluated through the same TestA/TestB pipeline) rather than direct comparison with prior numbers obtained on the full official test set. We also report in §3.2 that the Spearman rank correlation of λ rankings between TestA and TestB is $\rho = 0.96$ on Group A and $\rho = 0.80$ on Group B/C; this serves as quantitative evidence that the λ chosen on TestA generalizes to TestB, mitigating the concern that internal-split selection significantly inflates the reported numbers.

4.2 Models and Training Configuration

Base models.

- Source M_{src}: Llama-3-8B-Instruct, used in

NF4 (for QLoRA) or BF16 (for LoRA).

- Target M_{tgt}: Llama-3.1-8B-Instruct (BF16).

LoRA / QLoRA configuration.

- Rank $r = 8$, scaling factor $\alpha = 16$.
- Applied modules: 7 projections (q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj).
- Inserted at all 32 transformer layers $\times$ 7 projections $= 224$ submodules.
- QLoRA configuration: NF4 quantization with double quantization enabled, compute dtype BF16, adapter (A, B) kept in BF16. Implemented via `bitsandbytes`.

Training hyperparameters.

- Optimizer: AdamW (paged AdamW 8-bit for QLoRA), learning rate 1×10^{-5}, weight decay 0, warmup ratio 0.03, linear schedule.
- Effective batch size: 12 (per-GPU batch size 3 $\times$ gradient accumulation 1 $\times$ 4 GPUs).
- Epochs: 5 epochs uniformly across all tasks. Checkpoints are saved at a fixed `save_steps` interval; depending on the training data size and steps-per-epoch, this yields 5–26 CPs per training run (typically 5–10 CPs for small tasks and 15–26 CPs for medium-to-large tasks). The joint (CP, λ) search uses all available CPs for each task.

Seeds. Seed 50 is the main seed; seeds 55 and 60 are used for multi-seed verification.

4.3 Experiment 1: λ Sweep for Empirical Characterization

Goal. To systematically verify, on the 20 tasks in Groups A and B combined, (i) whether QLoRA-trained adapters can be transferred to the target base and (ii) how adjusting the scaling factor λ contributes to transfer performance.

Transfer pipeline and inference setup (a deviation from the QLoRA standard, with justification). We apply the trained adapter (ΔW_{src}, with LoRA matrices in BF16) to the target W_{tgt} (BF16) according to Eq. (2). In the λ sweep evaluation for our main tables (Tables 2–3), **both LoRA Transfer and QLoRA Transfer load the target base in BF16**, and the BF16 adapter is composed at runtime via scaling λ (by overriding `peft`'s `BaseTunerLayer.scaling`; `merge_and_unload` is not invoked).

This is a deliberate deviation from the standard QLoRA inference workflow (NF4 base + adapter on-the-fly); the primary motivation is to **ensure a fair comparison with LoRA Transfer**. Since LoRA Transfer is by construction trained and evaluated on a BF16 base, evaluating QLoRA Transfer alone on an NF4 base would entangle (a) the training-time base-precision difference (NF4 vs BF16) with (b) the inference-time base-precision difference (NF4 vs BF16). By evaluating both methods in the same BF16 inference environment, the performance gap is attributable **solely to the training-time precision difference**, directly addressing our main question of whether QLoRA-trained adapters transfer like LoRA-trained adapters[2].

Task-performance preservation under the QLoRA-trained adapter's **standard deployment workflow** (NF4 base + adapter on-the-fly) is separately verified for the 2 Group C tasks in §5.3 Table 4 and Appendix D, where the gap between the per-precision bests falls within a practical margin of $|\Delta| \leq 0.03$. In short, we cleanly separate the **performance evaluation phase (BF16 inference, fair comparison)** from the **deployment phase (NF4 inference, practical VRAM reduction)**.

Fine-grained λ sweep grid. For each (task, method), we evaluate TestA on the grid of $\lambda \in \{0.0, 0.1, 0.2, \ldots, 2.0\}$ (21 values) $\times$ all training-time CPs, and select the joint best (CP^*, λ^*). The final reported value is the TestB score.

Additional analyses (derived from Experiment 1). Using the Experiment 1 sweep data, we conduct the following auxiliary analyses:

- **Adapter delta analysis**: compute the cosine, norm, and λ_{opt} between ΔW_{src} and ΔW_{DA} over all 224 submodules.
- **Cross-seed verification**: compare ΔW_{src} trained at seeds 55 and 60 with ΔW_{DA} at seed

[2]Secondary reasons include: (i) BF16 inference has no NF4 dequantize overhead and is faster at inference time; (ii) quantization noise does not enter the evaluation stage, improving reproducibility; (iii) prior work such as ParamΔ (Cao et al., 2025) also evaluates on BF16 bases, which is consistent with community convention.

50.

- **DA λ-scan control**: sweep $\lambda \in \{0.7, 1.0, 1.3, 1.5, 1.7, 2.0\}$ over ΔW_{DA} to determine whether λ tuning is specific to transfer or is a general LoRA phenomenon.

- **Empirical evaluation of analytic λ**: measure task performance near λ_{opt} (typically in the 0.2–0.7 range) using a low-λ grid.

4.4 Experiment 2: Practical Deployment Scenario

Goal and motivation. When a new base-model version is released, a user faces the choice of training an adapter from scratch on the target base (DA) versus transferring an existing source-trained adapter with λ adjustment. In Experiment 2, we compare these two options in a practical setting on Group C (CommonGen, CSQA).

Smart λ grid (5 values). Based on the prior knowledge from Experiment 1 ($\lambda^* > 1.0$ dominates), we narrow the λ grid to 5 values:

- Generation task (CommonGen): $\{0.7, 1.0, 1.3, 1.6, 2.0\}$.
- Classification task (CSQA): $\{1.0, 1.3, 1.5, 1.7, 2.0\}$.

Joint (CP, λ) search. We evaluate TestA on the full CP $\times$ 5-λ grid and select the joint best (CP^*, λ^*). Pilot experiments confirmed that the two-stage search that first fixes the best CP at $\lambda = 1.0$ has a structural bias (see the footnote in §3.2); we therefore adopt the joint search.

Compute-cost measurement. We record the wall time and GPU-seconds for the smart λ grid; detailed measurements are reported in Appendix E.

4.5 Baselines

- **Direct Adaptation (DA):** an adapter trained directly on M_{tgt} under the same protocol as Transfer (rank, α, optimizer, batch, epochs, CP saving interval) and composed at $\lambda = 1.0$. CP selection also follows the Transfer side, picking the CP that maximizes the TestA score, to ensure a fair comparison with Transfer (the control experiment that sweeps λ on the DA side is reported separately in §5.2). DA represents the "train from scratch on the new base" approach and is our primary baseline.

Table 2: Experiment 1, Group A (Generation, corpus BLEU on TestB). Transfer-only main results. Tasks sorted by n_{train}. Numbers in parentheses after Transfer scores are λ^*. Δ = QLoRA − LoRA Transfer. Full DA reference results are in Appendix F; the main claim concerns LoRA-vs-QLoRA transfer behavior.

Task	n_{tr}	Base 3.1	LoRA T.	QLoRA T.	Δ
MBPP	474	0.062	0.131 (1.8)	0.152 (1.8)	+.021
MeQSum	800	0.011	0.235 (1.5)	0.236 (1.4)	+.001
SciTLDR	1,992	0.042	0.106 (0.9)	0.104 (1.1)	−.002
ViGGO	5,103	0.092	0.238 (1.1)	0.259 (1.1)	+.021
NarrativeQA	10,000	0.056	0.216 (0.8)	0.220 (0.8)	+.004
SAMSum	10,000	0.047	0.186 (0.9)	0.195 (0.9)	+.009
DialogSum	10,000	0.030	0.194 (1.1)	0.198 (1.2)	+.004
AESLC	14,438	0.014	0.078 (0.5)	0.093 (0.6)	+.015
E2E NLG	33,525	0.040	0.259 (1.1)	0.267 (1.1)	+.008
WebNLG	35,425	0.186	0.329 (0.7)	0.331 (0.6)	+.002

- **Naive Transfer ($\lambda = 1.0$):** ΔW_{src} composed into W_{tgt} at $\lambda = 1.0$, following the standard treatment in ParamΔ (Cao et al., 2025)-style work.

4.6 Statistical Testing

To assess whether the score difference between Joint Search and DA is statistically meaningful or merely sample-size noise, we apply a paired bootstrap test (Koehn, 2004) with 10,000 resamples to the two CSQA settings (LoRA / QLoRA) in §5.3 Table 4, the classification tasks for which per-sample correctness is preserved by our evaluation framework. CommonGen and all Group A/B tasks are reported only at the corpus level. Computation details (per-sample difference, percentile 95% CI, two-sided p-value) are deferred to Appendix M.2.

5 Results

This section presents the results of Experiment 1 (§5.1, §5.2) and Experiment 2 (§5.3).

5.1 Experiment 1 Main Results: Group A and B

In Experiment 1, we evaluate the transfer performance of QLoRA adapters on Group A (10 generation tasks) and Group B (10 classification tasks). The main tables (Tables 2–3) report Transfer-only comparisons (Base 3.1, LoRA Transfer, QLoRA Transfer, Δ = QLoRA − LoRA); full DA reference results are deferred to Appendix F, since the main claim concerns LoRA-vs-QLoRA Transfer behavior rather than Transfer-vs-DA.

Main observations. (i) **QLoRA Transfer $\approx$ LoRA Transfer.** On Seed 50, the absolute TestB evaluation-score gap is at most 0.054 on all 20

Table 3: Experiment 1, Group B (Classification, accuracy on TestB). Transfer-only main results. Δ = QLoRA − LoRA Transfer. Full DA reference results are in Appendix F.

Task	n_{tr}	Base 3.1	LoRA T.	QLoRA T.	Δ
PubMedQA	800	0.760	0.860 (1.5)	0.860 (1.1)	0.000
ARC-Challenge	1,119	0.537	0.565 (2.0)	0.551 (1.5)	−.014
ARC-Easy	2,251	0.758	0.799 (2.0)	0.794 (2.0)	−.005
OpenBookQA	4,957	0.492	0.508 (1.2)	0.476 (0.8)	−.032
GSM8K	7,473	0.817	0.758 (0.5)	0.772 (0.6)	+.014
BoolQ	9,427	0.837	0.902 (1.0)	0.903 (1.3)	+.001
PIQA	16,113	0.797	0.797 (1.7)	0.795 (2.0)	−.002
HellaSwag	39,905	0.725	0.660 (2.0)	0.662 (2.0)	+.002
WinoGrande	40,398	0.677	0.763 (1.1)	0.709 (0.5)	−.054
SST-2	67,349	0.868	0.963 (1.5)	0.956 (1.3)	−.007

tasks and at most 0.03 on 18/20. Under a practical equivalence margin of $|\Delta_{\text{practical}}| \leq 0.02$, Group A yields tie 8 / QLoRA 2 / LoRA 0, and Group B yields tie 8 / QLoRA 0 / LoRA 2; retention rates are 104.2% (Group A) and 98.7% (Group B). Two auxiliary checks support the same conclusion: a 3-seed paired evaluation on 6 representative tasks ($|\text{mean}(\Delta)| \leq 0.012$; Appendix G) and BERTScore-F1 at the same (CP, λ) on all 10 Group A tasks ($|\Delta(\text{Q}-\text{L})| \leq 0.006$; Appendix F, Table F.2). (ii) **Adapters are effective on most tasks.** Most tasks improve over Base 3.1; exceptions are GSM8K and HellaSwag (where Base 3.1 is already strong and both Transfer and DA fall below it), and PIQA (roughly on par with the base). (iii) **Distribution of λ^*.** Among the MCQ tasks in Group B (ARC, HellaSwag, PIQA), $\lambda^* \geq 1.5$ dominates, while several generation tasks (WebNLG, AESLC, NarrativeQA, SAMSum) prefer $\lambda^* < 1.0$.

5.2 Adapter Delta Analysis: Empirical Properties of λ Tuning

Why compare against λ_{opt}? Our starting point was practical: we wanted a λ that could be set without a search. λ_{opt} (Eq. (4)) is the first candidate one would reach for — closed-form, and geometrically as close as one can get to what direct adaptation produces, in the same weight space that task-arithmetic and model-merging methods operate in. We took it as a working assumption and tested it; it does not hold, and the disagreement below is systematic rather than incidental.

Figure 1 shows the central λ result: across all 44 settings (22 tasks × 2 methods, Seed 50), the empirical task-optimal λ^* is strictly larger than the Frobenius-distance optimum λ_{opt} (Eq. (4)), with a mean ratio of 4.35× (range $[1.22, 12.11]\times$)[3]. The

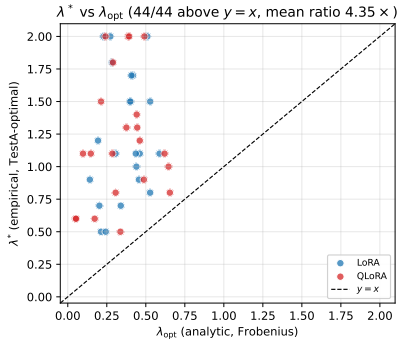


Figure 1: λ^* (TestA-maximizing) vs λ_{opt} (Frobenius-minimizing to ΔW_{DA}) across 44 settings (Seed 50). All points lie above $y = x$ (mean ratio 4.35×).

λ^* grid value also yields higher TestB scores than the grid value closest to λ_{opt} in most settings, with a median gap of +5.4% (max +22.2% on MeQSum LoRA); a comparable coarse λ scan on DA (34 settings) shows a smaller median gain (+1.1%). The gap is task-type dependent: near zero on the accuracy-scored tasks (median 0.008) but large on the generation tasks (median 0.060, up to 0.260), consistent with BLEU being more λ-sensitive than discrete accuracy. Aggregate cosine predicts neither λ^* nor task performance, whereas the effective norm ratio $\|\lambda^* B\|_F / \|A\|_F$ correlates with λ^* categories. Supporting analyses and the full 44-setting numbers are in Appendices H and I.

5.3 Experiment 2: Practical Deployment Scenario

In Experiment 2, we use Group C (CommonGen, CSQA) to compare DA against Joint Search (5-value λ grid) in a practical setting that mimics releasing a new base-model version. Results are shown in Table 4.

Performance. Results are reported with the caveat that DA uses a fixed $\lambda = 1.0$ while Joint Search uses a 5-value λ grid, so the comparison is not on identical scaling-search budgets. With that caveat, Joint Search reaches the same range as DA in 3 of 4 settings. On CSQA QLoRA, DA is significantly superior ($p = 0.014$); we record it as an honest failure case, and it is not isolated. We therefore also ran DA over the same 5-value grid on Group C (Appendix K). Under this matched budget DA improves over its base $\lambda = 1.0$ by a median of +0.013, and the CSQA LoRA difference is no longer significant ($p = 0.28$); the picture is unchanged, with Joint Search still reaching the same

[3] λ^* saturates at the grid upper bound 2.0 in 7/44 settings (all MCQ), so the reported ratio is a lower bound there.

Table 4: Experiment 2 results on Group C with paired bootstrap test and deployment VRAM. **Bold** $p < 0.05$. "–" = bootstrap not applicable (per-sample BLEU not preserved by our eval framework). **Deploy VRAM**: post-load steady-state inference VRAM during TestB evaluation (transient peak during model loading is reported separately in Appendix D); LoRA loads target as BF16 with merged adapter (model-level, task-independent), QLoRA loads target as NF4 with runtime adapter scaling (BSZ depends on task: CommonGen 8 (gen), CSQA 64 (MCQ)).

Task	Method	Joint	DA	Δ	p	Deploy VRAM
CommonGen	LoRA	0.319	0.305	+0.014	–	15.1 GB
CommonGen	QLoRA	0.337	0.319	+0.018	–	**5.7 GB**
CSQA	LoRA	**0.821**	0.795	**+0.026**	**0.038**	19.6 GB
CSQA	QLoRA	0.764	**0.797**	−0.033	**0.014**	10.0 GB

range as DA in 3 of 4 settings and CSQA QLoRA the single failure. Across the 20 Llama Group A/B tasks (Appendix F) Transfer is within the margin of DA on 14, and *all five* tasks where DA is clearly better are multiple-choice, against 9/10 generation tasks within the margin; on Mistral the counts over the 18 comparably scored tasks are 10/7/1 (Appendix J). Over these 38 settings the median gap is -0.005 and 34 fall within $|\Delta| \leq 0.05$: DA stays an upper bound that transfer approaches without any training.

Deployment VRAM. Deploying QLoRA-trained adapters via NF4 base + runtime adapter scaling reaches post-load steady-state inference VRAM of 5.7 GB (CommonGen, BSZ=8) and 10.0 GB (CSQA, BSZ=64), a 49–63% reduction relative to LoRA-merged BF16 measured on the same tasks (15.1 and 19.6 GB). A per-precision joint (CP, λ) search on both targets confirms that the gap between per-precision bests stays within $|\Delta| \leq 0.03$ (Appendix D); QLoRA training itself reduces peak VRAM from 29–30 GB to 21–25 GB at the cost of 17–22% slower steps (Appendix E). Extension of the deployment evaluation to all 20 Group A/B tasks is ongoing work.

5.4 Replication across Three Model Families

To test whether the findings are specific to the Llama-3 family, we replicated the pipeline on two independently developed families—**Mistral-7B-Instruct v0.1→v0.3** (Jiang et al., 2023) and **Qwen2→Qwen2.5** (Yang et al., 2024a,b)—using the same 20-task suite, protocol, and split.

All three findings replicate (Table 5): the two training methods stay within the practical margin on **54/60** settings, the λ gap holds on **111/124** (89.5%, with every family in the 85–93% range), and NF4 deployment stays within the margin on **53/60**. The failure of weight-space proximity is therefore not a Llama-specific artifact.

Table 5: Replication of the three findings across three families on the same 20-task suite; counts with mean in parentheses. (i) QLoRA vs LoRA Transfer at each method's joint best; (ii) λ^* vs the closed-form Frobenius optimum at exact λ_{opt}, score gap $=$ TestB(λ_{opt}) − TestB(λ^*); (iii) NF4 vs BF16 deployment. Margin as in §5.1. Per-task: Appendix L.

	Llama 3.0→3.1	Mistral v0.1→v0.3	Qwen 2→2.5	**All**
(i) $\lvert\Delta\rvert \leq 0.02$	16/20	20/20	18/20	**54/60**
(mean $\lvert\Delta\rvert$)	.011	.005	.009	.008
(ii) $\lambda^* \succ \lambda_{\text{opt}}$	41/44	34/40	36/40	**111/124**
(mean score gap)	−.080	−.059	−.060	−.067
(iii) $\lvert\Delta\rvert \leq 0.02$	17/20	18/20	18/20	**53/60**
(mean $\lvert\Delta\rvert$)	.011	.010	.013	.011

6 Conclusion

We have shown that QLoRA-trained adapters cross-version transfer much like LoRA-trained adapters (score gap $\leq$ 0.054 on all 20 Llama tasks; NF4 deployment at 5.7–10.0 GB vs 15.1–19.6 GB for LoRA-merged BF16), and that the λ gap holds across all 44 settings: weight-space proximity alone is insufficient for choosing λ. All three findings reproduce on two further families (§5.4).

These conclusions are scoped to **7–8B instruction-tuned models, NF4 quantization, and same-architecture version pairs**; we make no claim about other scales or quantization schemes. We also record a negative case: on CommonsenseQA, QLoRA Transfer falls significantly below direct adaptation ($p = 0.014$); transfer is not uniformly safe.

Limitations

We organize the main limitations into five themes.

(1) Scope of models and tasks. Our cross-version Δ-transfer is verified on three same-architecture, same-size version pairs (Llama-3-8B-Instruct → Llama-3.1-8B-Instruct, Mistral-7B-Instruct v0.1 → v0.3, and Qwen2-7B-Instruct → Qwen2.5-7B-Instruct), all at the 7–8B scale and all using NF4 quantization. **Whether the same behaviour holds at other model scales, under other quantization schemes (e.g., GPTQ, AWQ), or with quantization-aware adaptation methods such as QA-LoRA (Xu et al., 2024) and LoftQ**

(Li et al., 2024) is not established by our experiments. Cross-size and cross-family transfer, other version pairs, and other quantization combinations remain future work, as is the full disentanglement of precision change vs version change (e.g., QLoRA adapter on BF16 Llama-3.0 as an intermediate control). Head-to-head comparisons with SOTA quantization-aware LoRA merging methods are out of scope, as they target different training/quantization setups.

(2) Internal validation / held-out evaluation split. (CP, λ) is selected on TestA (internal validation split carved from the official test set); TestB is the held-out evaluation split. This design is adopted to maintain a uniform protocol across all 22 tasks (many lack a usable official validation split) and to keep the LoRA-vs-QLoRA comparison apples-to-apples (§4); the high TestA–TestB Spearman correlation (mean $\rho = 0.96$ for Group A and 0.80 for Group B/C; §3.2) suggests the design does not significantly inflate the reported numbers. Our TestB scores are intended for within-paper relative comparison rather than direct comparison with prior benchmark numbers on the full official test set; an external evaluation using official validation splits is future work.

(3) Statistical support. Paired bootstrap is reported only for the two CSQA settings in Experiment 2 (per-sample BLEU is not preserved for the generation tasks, and Group A/B results are reported at the corpus level). Multi-seed equivalence (Table G.1) is verified on 6 representative tasks with $n = 3$ seeds; a formal non-inferiority test on the full 20 tasks with larger n is future work.

(4) DA comparison and λ-grid asymmetry. The DA reference columns in Appendix F use a fixed $\lambda = 1.0$; a matched 5-value grid is applied to DA only on Group C (Appendix K). A fully matched comparison on all 20 tasks is future work, so we treat Transfer-vs-DA comparisons as supplementary reference rather than the main claim. The λ search range is also limited to $\lambda \leq 2.0$; 7 of 44 settings (all MCQ classification) saturate at $\lambda^* = 2.0$, so reported $\lambda^*/\lambda_{\text{opt}}$ ratios are lower bounds for those settings.

(5) Metrics and NF4-deployment coverage. For generation, the main tables use corpus BLEU as the primary metric, and Appendix F (Table F.2) reports BERTScore-F1 for all 10 Group A tasks as an auxiliary semantic-similarity check; task-specific best-practice metrics such as MBPP pass@k and ROUGE-L for summarization remain future work. Because our main comparison is relative and uses the same metric for LoRA and QLoRA, this limitation does not directly affect the LoRA-vs-QLoRA comparison. NF4-deployment performance is evaluated on 2 Group C tasks (Appendix D); extension to the 20 Group A/B tasks is ongoing work, and the low-VRAM claim is supported only as a 2-task case study at this stage.

Ethics Statement

All datasets used in this work are established public benchmarks (obtainable via the Hugging Face Hub) and are used as distributed by their original creators; we did not curate, re-annotate, or independently audit them. We note that AESLC is derived from the publicly released Enron email corpus and therefore contains real personal names and workplace correspondence. The Llama-3-family base models (Grattafiori et al., 2024) are used for research purposes under the license released by Meta. We do not perform any human evaluation; no IRB review is therefore required.

The contribution of this work is a technical workflow for resource-constrained researchers and developers to transfer QLoRA-trained adapters to a different base model, and is not intended for direct application to misinformation spread, bias amplification, or automated decision making. Nonetheless, the ethical concerns common to adapter transfer in general (e.g., the possibility that bias from the source task is transferred to the target base) apply, in line with the LoRA-composer and Task Arithmetic literature, and responsible deployment is left to the implementer.

In terms of compute, training and evaluation together consumed approximately 600 GPU-h on a mix of NVIDIA A100 80GB and NVIDIA RTX PRO 6000 (Blackwell, 96GB).

Acknowledgments

This work was supported by TOYO University Top Priority Research Program Grant Number 2025T2.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Asma Ben Abacha and Dina Demner-Fushman. 2019. On the summarization of consumer health questions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2228–2234, Florence, Italy. Association for Computational Linguistics.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, and 11 others. 2024. Lessons from the trenches on reproducible evaluation of language models. *Preprint*, arXiv:2405.14782.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439.

Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel Weld. 2020. TLDR: Extreme summarization of scientific documents. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4766–4777, Online. Association for Computational Linguistics.

Sheng Cao, Mingrui Wu, Karthik Prasad, Yuandong Tian, and Zechun Liu. 2025. ParamΔ for direct mixing: Post-train large language model at zero cost. In *International Conference on Learning Representations*.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. DialogSum: A real-life scenario dialogue summarization dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5062–5074, Online. Association for Computational Linguistics.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. 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)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing Systems*, 36.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 538 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. LoraHub: Efficient cross-task generalization via dynamic LoRA composition. In *First Conference on Language Modeling (COLM)*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *Preprint*, arXiv:2310.06825.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. 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), pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Juraj Juraska, Kevin Bowden, and Marilyn Walker. 2019. ViGGO: A video game corpus for data-to-text generation in open-domain conversation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 164–172, Tokyo, Japan. Association for Computational Linguistics.

Youngeun Kim, Seunghwan Lee, Aecheon Jung, Bogon Ryu, and Sungeun Hong. 2025. Task vector quantization for memory-efficient model merging. *Preprint*, arXiv:2503.06921.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. 2024. LoftQ: LoRA-fine-tuning-aware quantization for large language models. In *International Conference on Learning Representations*.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

Pin-Jie Lin, Rishab Balasubramanian, Fengyuan Liu, Nikhil Kandpal, and Tu Vu. 2025. Efficient model development through fine-tuning transfer. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2617–2636, Suzhou, China. Association for Computational Linguistics.

James Liu, Guangxuan Xiao, Kai Li, Jason D Lee, Song Han, Tri Dao, and Tianle Cai. 2024. BitDelta: Your fine-tune may only be worth one bit. In *Advances in Neural Information Processing Systems*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. 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*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Ryota Miyano and Yuki Arase. 2025. Adaptive LoRA merge with parameter pruning for low-resource generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19353–19366. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Akshara Prabhakar, Yuanzhi Li, Karthik Narasimhan, Sham Kakade, Eran Malach, and Samy Jelassi. 2025. LoRA soups: Merging LoRAs for practical skill composition tasks. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 644–655. Association for Computational Linguistics.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Runqian Wang, Soumya Ghosh, David Cox, Diego Antognini, Aude Oliva, Rogerio Feris, and Leonid Karlinsky. 2024. Trans-LoRA: Towards data-free transferable parameter efficient finetuning. In *Advances in Neural Information Processing Systems*, volume 37.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, pages 23965–23998.

Feifan Xia, Mingyang Liao, Yuyang Fang, Defang Li, Yantong Xie, Weikang Li, Yang Li, Deguo Xia, and Jizhou Huang. 2025. Cross-LoRA: A data-free LoRA transfer framework across heterogeneous LLMs. *Preprint*, arXiv:2508.05232.

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

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024a. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024b. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Rui Zhang and Joel Tetreault. 2019. This email could save your life: Introducing the task of email subject line generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 446–456, Florence, Italy. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*.

A Pipeline Overview

Figure A.1 summarizes the two-phase pipeline used throughout this work.

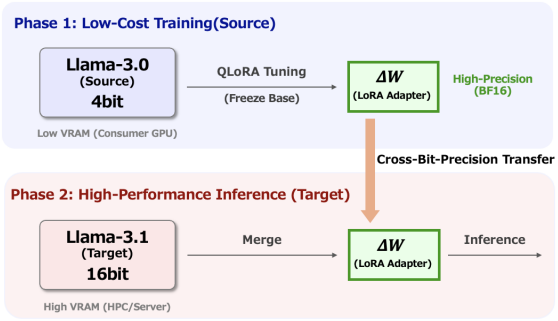


Figure A.1: Overview of Cross-Version Δ-Transfer for QLoRA-trained adapters. **Phase 1 (Low-Cost Training):** a LoRA adapter ΔW is trained on a 4-bit (NF4) quantized source base (Llama-3.0) under the QLoRA protocol, where the base is frozen and only the BF16 adapter receives gradient updates. **Phase 2 (Cross-Version Transfer):** the resulting BF16 adapter is merged into a 16-bit target base of a different version (Llama-3.1) via $W_{\text{final}} = W_{\text{tgt}} + \lambda \cdot \Delta W$, where λ is the per-task merge-scaling factor analyzed in this work.

B Evaluation Datasets

Table B.1 lists the source of every task used in this work. All are publicly available benchmarks obtained through the Hugging Face Hub. Training-set sizes are given in Tables 2 and 3. ARC-Challenge and ARC-Easy come from the same release, so the 22 tasks draw on 21 datasets.

C GSM8K Evaluation Protocol

GSM8K is grouped with the accuracy-scored tasks (Group B) because its primary metric is accuracy, but—unlike the multiple-choice tasks in that group—it has no answer options, so the option-token log-likelihood scoring described in §4 does not apply. We therefore evaluate it as follows.

- **Prompt and decoding.** The question is presented in the same `messages` chat format as the other tasks, and the model generates a chain-of-thought completion under greedy decoding (temperature 0, top-p 1.0).
- **Answer extraction.** The final numeric answer is extracted from the generated text with the `strict-match` filter of lm-evaluation-harness (Biderman et al., 2024); the extracted string is compared against the reference answer.
- **Metric.** Exact match on the extracted final answer, reported as accuracy. F1-macro, which is reported as an auxiliary metric for the multiple-choice tasks, is not defined here because there is no fixed label set.

Table B.1: Source and primary metric for the 22 evaluation tasks.

Task	Source	Metric
Group A — generation (10)		
AESLC	Zhang and Tetreault (2019)	BLEU
DialogSum	Chen et al. (2021)	BLEU
E2E NLG	Novikova et al. (2017)	BLEU
MBPP	Austin et al. (2021)	BLEU
MeQSum	Ben Abacha and Demner-Fushman (2019)	BLEU
NarrativeQA	Kočiský et al. (2018)	BLEU
SAMSum	Gliwa et al. (2019)	BLEU
SciTLDR	Cachola et al. (2020)	BLEU
ViGGO	Juraska et al. (2019)	BLEU
WebNLG	Gardent et al. (2017)	BLEU
Group B — accuracy-scored (10)		
ARC-Challenge	Clark et al. (2018)	acc
ARC-Easy	Clark et al. (2018)	acc
BoolQ	Clark et al. (2019)	acc
GSM8K	Cobbe et al. (2021)	EM
HellaSwag	Zellers et al. (2019)	acc
OpenBookQA	Mihaylov et al. (2018)	acc
PIQA	Bisk et al. (2020)	acc
PubMedQA	Jin et al. (2019)	acc
SST-2	Socher et al. (2013)	acc
WinoGrande	Sakaguchi et al. (2021)	acc
Group C — deployment evaluation (2)		
CommonGen	Lin et al. (2020)	BLEU
CommonsenseQA	Talmor et al. (2019)	acc

- **Evaluation size.** All 1,318 official test items are used, split into TestA/TestB by `doc_id % 2` as for every other task.

No conversion of GSM8K into a multiple-choice format is performed at any point.

D NF4 Deployment Performance under Per-Precision Joint Search

Table 4 in §5.3 showed that the QLoRA Transfer adapter can be deployed via the NF4 base + runtime adapter scaling at a post-load steady-state inference VRAM of 5.7–10.0 GB on the 2 Group C tasks. In this appendix, we verify that this NF4 deployment setting also preserves task performance comparable to BF16-target inference.

Design: per-precision fair joint search. Reusing $(\mathrm{CP}^*, \lambda^*)$ chosen on the BF16 target as-is on the NF4 target would ignore the shift in optima due to the precision difference. In this experiment, we therefore independently perform a joint (CP, λ) grid search on the NF4 target as well. For both tasks, the grid is **26 CPs** $\times$ **5** λ **values** $= 130$ evaluations per task: CPs are all 26 checkpoints available per training run (the denser CP set saved for CSQA / CommonGen at the `save_steps` interval described in §4), and λ uses the same 5-value smart grid as in Experiment 2. The comparison is then between the **BF16 best and the NF4 best**—i.e., the best for each precision.

Target tasks. We evaluate the 2 new Group C tasks (CSQA, classification; CommonGen, generation) as Phase 1, covering both the generation and classification axes for a representative appendix table. Extension to all Group A/B tasks and to a finer λ grid on HellaSwag is left for future work.

Results. Table D.1 shows the TestB scores at the BF16 best and the NF4 best. CSQA degrades by -0.82 pp in accuracy, and CommonGen by -2.05 pp in BLEU; in both cases the gap stays within the **practical equivalence margin of** $|\Delta| \leq 0.03$.

λ^* is precision-dependent. For CommonGen, λ^* on the NF4 target shifts down to 0.7 from 1.0 on the BF16 target. For generation tasks, quantization noise accumulates over long-context generation, so reducing λ can be interpreted as moving the optimum in a direction that compensates for noise. CSQA (a 5-way MCQ) involves only short decision steps and is less affected by noise; both precisions agree on λ^*=2.0. This observation is consistent with the interpretation in §5.2 that λ tuning may capture factors not explainable by weight-space

Table D.1: NF4 deployment vs BF16 deployment under per-precision joint (CP, λ) search. NF4-target inference uses NF4-quantized Llama-3.1-8B-Instruct base with double quantization and BF16 adapter applied via runtime scaling (no merge). Both precisions select $(\mathrm{CP}^*, \lambda^*)$ independently on their own TestA split.

Task	Metric	BF16 best	NF4 best	Δ
CSQA	acc	0.7639 (λ^*=2.0)	0.7557 (λ^*=2.0)	-0.0082
CommonGen	BLEU	0.3367 (λ^*=1.0)	0.3162 (λ^*=0.7)	-0.0205

proximity alone, such as quantization-error compensation.

VRAM. The **post-load steady-state inference VRAM during evaluation** for NF4 inference is 5.7 GB on CommonGen (BSZ=8) and 10.0 GB on CSQA (BSZ=64), confirming a 49–63% VRAM reduction relative to LoRA-merged BF16 on the same tasks (15.1 and 19.6 GB)[4]. These numbers agree with the Deploy VRAM column of Table 4 in §5.3.

E Detailed Training and Sweep Compute Cost

We detail the representative training and λ-sweep costs summarized in §5.3, on three representative tasks (small / medium / large by n_{train}): Table E.1 reports both training (DA) cost and λ-sweep cost. Wall-clock advantages depend on task size and grid size; the 5-value smart grid (§4.4) can be slower than DA on small tasks, but the VRAM advantage at NF4 deployment (§5.3) is stable across task sizes.

F Full Group A/B Results with DA Reference Columns

Tables F.1 and F.3 report the Experiment 1 results with DA columns included. The DA columns use a fixed $\lambda = 1.0$ while the Transfer columns use a 21-value λ scan; the two are therefore not on identical scaling-search budgets, and we treat Transfer-vs-DA comparisons as supplementary reference rather than the main claim. Under this caveat, Transfer and DA are broadly in the same range across tasks (small-data tasks such as MBPP, MeQSum, BoolQ, PubMedQA tend to favor Transfer; large-data tasks such as HellaSwag, AESLC, PIQA tend to favor DA); the gap is expected to narrow further when DA is also given a λ scan (§5.2).

BERTScore companion for Group A. Table F.2 reports BERTScore-F1 at the *same* checkpoint and λ as Table F.1 (Seed 50). The LoRA-vs-QLoRA Transfer gap in BERT-F1 stays within $|\Delta| \leq 0.006$ on all 10 tasks, consistent with the BLEU-based observation that QLoRA Transfer is close to LoRA Transfer.

G Multi-Seed Paired Evaluation (Full Table)

Table G.1 reports the multi-seed paired evaluation referenced in §5.1: 6 representative tasks (Group A: MeQSum, SAMSum, WebNLG; Group B: BoolQ, HellaSwag, SST-2) at seeds $\{50, 55, 60\}$ ($n = 3$). All six tasks satisfy $|\text{mean}(\Delta)| \leq 0.012$ with SD ≤ 0.012, supporting practical non-inferiority within a ± 0.02 margin; a formal non-inferiority test on the full 20 tasks with larger n is future work.

H Supplementary Lambda Analyses

Figure H.1 presents three analyses that complement the main result of §5.2 (Figure 1).

I Full 44-Setting Lambda Table

Table I.1 reports the full 44-setting numbers from the three-stage analysis in §5.2 (baseline similarity $\rightarrow$ analytic $\lambda_{\text{opt}} \rightarrow$ empirical λ^*).

J Transfer vs Direct Adaptation on Mistral

Table J.1 gives the per-task QLoRA Transfer vs DA comparison for the Mistral pair, the counterpart of the DA reference columns in Appendix F. DA is evaluated at $\lambda = 1.0$, as in Table F.1. Two tasks (†) were scored with corpus BLEU in this run rather than with the metric used for the Llama suite (exact match for GSM8K, accuracy for PubMedQA); the aggregate counts quoted in §5.3 exclude them, which moves the mean Δ only from -0.0185 to -0.0187.

[4] All VRAM figures are peak *allocated* memory (`torch.cuda.max_memory_allocated`), i.e. peak tensor memory. They exclude the CUDA context and the caching allocator's reservations; on a 96 GB device the latter grow opportunistically and do not track what a deployment requires (we measured NF4 *reserved* peaks exceeding those of BF16 on both tasks). The reported VRAM reflects the post-load inference steady-state; model loading may require a transient peak of ∼14.5 GB due to NF4 quantization setup with temporary BF16 buffers. Practical deployment must provision for this loading transient, but the steady-state value is what bounds sustained serving VRAM.

Table E.1: Compute cost on three representative tasks: **small (MeQSum, 800) / medium (SAMSum, 10K) / large (HellaSwag, 40K)**. VRAM in GB, wall in seconds, throughput (tput) in samples/s. Training tput is $n_{\text{train}} \times 5\,\text{epochs}/\text{wall}$; sweep tput is $n_{\text{eval}}/\text{wall_real}$ per job. Both methods load the target base in BF16 during sweep evaluation, so sweep VRAM is similar across methods.

		Training (DA)			λ-sweep	
Task	Method	VRAM	wall	tput	VRAM	tput
MeQSum	LoRA	29.4	133	30.1	∼17	6.8
	QLoRA	24.6	171	23.4	∼17	17.4
SAMSum	LoRA	29.4	2,029	24.6	∼18	2.2
	QLoRA	24.6	2,544	19.7	∼18	1.8
HellaSwag	LoRA	30.2	6,923	28.9	∼19	6.8
	QLoRA	21.5	8,596	23.3	∼19	7.2

Table F.1: Experiment 1 results on Group A (Generation, corpus BLEU on TestB) with DA reference columns. **Bold** indicates the best score per row. Numbers in parentheses after Transfer scores are λ^*.

			LoRA		QLoRA	
Task	n_{tr}	Base 3.1	Transfer	DA	Transfer	DA
MBPP	474	0.062	0.131 (1.8)	0.144	**0.152** (1.8)	0.138
MeQSum	800	0.011	0.235 (1.5)	**0.257**	0.236 (1.4)	0.210
SciTLDR	1,992	0.042	0.106 (0.9)	**0.111**	0.104 (1.1)	0.110
ViGGO	5,103	0.092	0.238 (1.1)	0.233	**0.259** (1.1)	0.255
NarrativeQA	10,000	0.056	0.216 (0.8)	0.220	0.220 (0.8)	**0.222**
SAMSum	10,000	0.047	0.186 (0.9)	0.199	0.195 (0.9)	**0.202**
DialogSum	10,000	0.030	0.194 (1.1)	**0.209**	0.198 (1.2)	0.206
AESLC	14,438	0.014	0.078 (0.5)	**0.092**	0.093 (0.6)	0.090
E2E NLG	33,525	0.040	0.259 (1.1)	**0.266**	0.267 (1.1)	0.262
WebNLG	35,425	0.186	0.329 (0.7)	0.325	**0.331** (0.6)	0.325

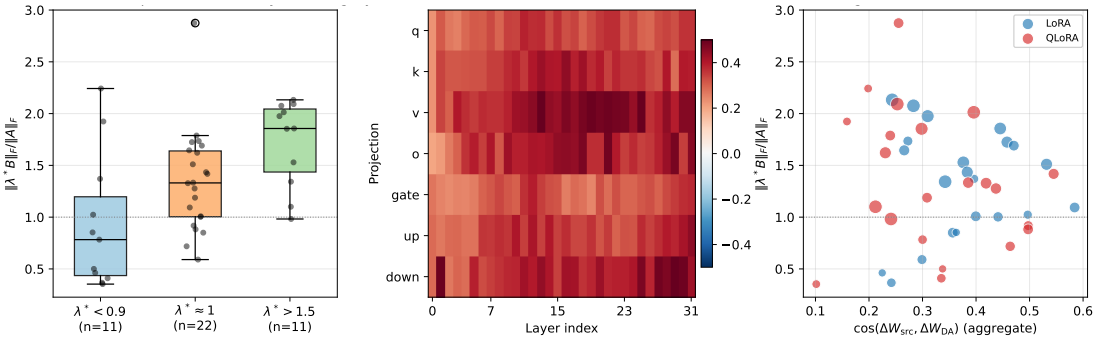


Figure H.1: Supplementary λ analyses across 44 settings; panels are ordered left to right. **(a)** Effective adapter norm ratio $\|\lambda^* B\|_F / \|A\|_F$ grouped by λ^* category. **(b)** Per-layer cosine heatmap $\cos(\Delta W_{\text{src}}, \Delta W_{\text{DA}})$ across 32 layers $\times$ 7 projections, averaged over settings. **(c)** Aggregate cosine vs effective norm ratio, marker size proportional to λ^*. The three views together indicate that (i) the effective scaling magnitude tracks λ^*, (ii) per-layer cosine structure is non-uniform across layers and projections, and (iii) aggregate cosine is not a strong predictor of λ^* — supporting the main observation that the Frobenius-distance heuristic does not capture the task-optimal λ.

K Matched-Budget Control for Direct Adaptation

Table K.1 removes the scaling-search asymmetry discussed in §5.3: DA is re-evaluated over the *same* 5-value λ grid used by Joint Search, with λ selected on TestA and scores reported on TestB. A paired bootstrap (10,000 resamples, seed 42) is applied to the two CSQA settings, where per-sample scores are preserved. On CommonGen QLoRA the selected $\lambda^{\text{DA}} = 0.7$ sits at the lower end of the grid, so the corresponding gain is a lower bound.

L Per-Task Results for the Three-Family Replication

Tables L.1 and L.2 give the per-task numbers behind Table 5. All values are TestB scores under

Table F.2: Experiment 1, Group A (Generation, BERTScore-F1 on TestB). Transfer-only auxiliary check, evaluated at the same (CP, λ) as Table 2 (Seed 50). Δ = QLoRA − LoRA Transfer; $|\Delta| \leq 0.006$ on all 10 tasks.

Task	n_{tr}	Base 3.1	LoRA T.	QLoRA T.	Δ
MBPP	474	0.852	0.891	0.895	+.004
MeQSum	800	0.869	0.932	0.936	+.004
SciTLDR	1,992	0.876	0.898	0.899	+.001
ViGGO	5,103	0.886	0.935	0.939	+.003
NarrativeQA	10,000	0.869	0.939	0.940	+.001
SAMSum	10,000	0.884	0.923	0.923	0.000
DialogSum	10,000	0.868	0.921	0.921	0.000
AESLC	14,438	0.832	0.866	0.872	+.006
E2E NLG	33,525	0.880	0.941	0.942	+.001
WebNLG	35,425	0.921	0.946	0.947	+.001

the protocol of §3.2; the metric column indicates BLEU for generation tasks and accuracy for the accuracy-scored tasks.

Family-specific observations. The largest method gaps occur on GSM8K and WinoGrande, both small-sample or reasoning-heavy tasks. The Mistral pair spans a larger version gap than the other two—v0.3 extends the vocabulary from 32,000 to 32,768 tokens—yet shows the *smallest* method gap (20/20 within margin), indicating that the QLoRA-vs-LoRA equivalence is not sensitive to the size of the version change. Qwen, in contrast, shows by far the strongest tendency for λ^* to saturate at the upper end of the original grid; we therefore extended the grid beyond $\lambda = 2.0$ for the affected settings, and Table 5 reports the post-extension values. Applying the same extension to all three families leaves the win counts for Mistral and Qwen unchanged and only deepens their mean gaps, confirming that the saturation was specific to how far each target base had drifted, not to the comparison itself.

M Protocol and Statistical Details

M.1 Pilot Verification of the Two-Stage Bias

For the protocol choice in §3.2, we verified the two-stage bias on 4 settings (CommonGen / CSQA × LoRA / QLoRA): the two-stage search (fix the best CP at $\lambda = 1.0$, then sweep λ) selected $\lambda^* = 1.0$ in 4/4 cases, but re-evaluating on the full CP × λ grid selected $\lambda^* \neq 1.0$ in 3/4 cases, yielding a $+0.067$ accuracy improvement on CSQA LoRA. This can be interpreted as follows: the CP that maximizes TestA at $\lambda = 1.0$ is implicitly optimized to that scale during training, suppressing latent performance at other λ values. We therefore adopt the joint (CP, λ) search throughout.

M.2 Paired Bootstrap Computation Details

For each (task, method), we collect TestB per-sample correctness (1 if correct, 0 otherwise; for classification) under Joint Search and DA, and compute the per-sample difference $d_i = \text{Joint}_i - \text{DA}_i$ ($i = 1, \ldots, N$). We resample N values with replacement 10,000 times and compute the mean difference $\bar{d}$ for each resample. From this empirical distribution we derive (i) a percentile bootstrap 95% confidence interval bounded by the 2.5% and 97.5% percentiles of the empirical distribution, and (ii) a two-sided p-value equal to twice the smaller of $P(\bar{d}^* \leq 0)$ and $P(\bar{d}^* \geq 0)$. If the 95% CI does not contain 0, we deem the difference statistically significant.

Table F.3: Experiment 1 results on Group B (Classification, accuracy on TestB) with DA reference columns. **Bold** indicates the best score per row.

			LoRA		**QLoRA**	
Task	n_{tr}	Base 3.1	Transfer	DA	Transfer	DA
PubMedQA	800	0.760	0.860 (1.5)	0.860	0.860 (1.1)	0.860
ARC-Challenge	1,119	0.537	0.565 (2.0)	0.572	0.551 (1.5)	**0.582**
ARC-Easy	2,251	0.758	0.799 (2.0)	0.805	0.794 (2.0)	**0.812**
OpenBookQA	4,957	0.492	0.508 (1.2)	0.496	0.476 (0.8)	**0.512**
GSM8K	7,473	**0.817**	0.758 (0.5)	0.749	0.772 (0.6)	0.753
BoolQ	9,427	0.837	0.902 (1.0)	0.899	**0.903** (1.3)	0.892
PIQA	16,113	0.797	0.797 (1.7)	**0.826**	0.795 (2.0)	0.820
HellaSwag	39,905	0.725	0.660 (2.0)	**0.742**	0.662 (2.0)	0.735
WinoGrande	40,398	0.677	**0.763** (1.1)	0.749	0.709 (0.5)	0.736
SST-2	67,349	0.868	**0.963** (1.5)	0.949	0.956 (1.3)	0.949

Table G.1: Multi-seed paired evaluation ($n = 3$ seeds, Seed $\in \{50, 55, 60\}$). Δ = QLoRA $-$ LoRA Transfer. Each cell shows mean$\pm$std.

Task	Metric	LoRA	QLoRA	Δ(Q$-$L)
MeQSum	BLEU	.242±.013	.248±.011	+.006±.008
MeQSum	BERT-F1	.933±.002	.938±.002	+.005±.001
SAMSum	BLEU	.185±.006	.196±.005	+.012±.003
SAMSum	BERT-F1	.923±.001	.923±.001	0.000±.001
WebNLG	BLEU	.333±.004	.332±.001	−.002±.004
WebNLG	BERT-F1	.947±.001	.947±.000	0.000±.001
BoolQ	acc	.899±.005	.901±.003	+.002±.003
HellaSwag	acc	.656±.008	.653±.009	−.003±.012
SST-2	acc	.960±.005	.955±.007	−.005±.012

		Stage 1 (baseline)		Stage 2 & 3 (optima)		Comparison	
Task	Method	cos	$\|\Delta W_{\text{DA}}\|/\|\Delta W_{\text{src}}\|$	λ_{opt}	λ^*	$\lambda^*/\lambda_{\text{opt}}$	$\|\lambda^* B\|/\|A\|$
AESLC	LoRA	0.363	0.586	0.213	0.50	2.35	0.853
AESLC	QLoRA	0.159	0.312	0.050	0.60	12.11	1.924
ARC-Challenge	LoRA	0.243	0.938	0.228	2.00	8.77	2.133
ARC-Challenge	QLoRA	0.231	0.925	0.213	1.50	7.04	1.622
ARC-Easy	LoRA	0.283	0.964	0.273	2.00	7.33	2.076
ARC-Easy	QLoRA	0.253	0.956	0.242	2.00	8.27	2.092
BoolQ	LoRA	0.441	0.997	0.440	1.00	2.27	1.003
BoolQ	QLoRA	0.385	0.975	0.376	1.30	3.46	1.333
CommonGen	LoRA	0.497	0.684	0.340	0.70	2.06	1.024
CommonGen	QLoRA	0.464	1.392	0.646	1.00	1.55	0.718
CSQA	LoRA	0.445	0.916	0.408	1.70	4.17	1.856
CSQA	QLoRA	0.396	0.994	0.393	2.00	5.09	2.013
DialogSum	LoRA	0.356	1.294	0.461	1.10	2.38	0.850
DialogSum	QLoRA	0.545	0.846	0.461	1.20	2.60	1.418
E2E NLG	LoRA	0.471	0.651	0.306	1.10	3.59	1.691
E2E NLG	QLoRA	0.255	0.383	0.098	1.10	11.26	2.874
GSM8K	LoRA	0.225	1.083	0.243	0.50	2.06	0.462
GSM8K	QLoRA	0.101	1.698	0.172	0.60	3.48	0.353
HellaSwag	LoRA	0.342	1.490	0.510	2.00	3.92	1.342
HellaSwag	QLoRA	0.241	2.036	0.491	2.00	4.08	0.982
MBPP	LoRA	0.310	0.911	0.282	1.80	6.38	1.975
MBPP	QLoRA	0.298	0.971	0.290	1.80	6.21	1.853
MeQSum	LoRA	0.458	0.870	0.398	1.50	3.77	1.725
MeQSum	QLoRA	0.419	1.054	0.442	1.40	3.17	1.328
NarrativeQA	LoRA	0.242	2.182	0.528	0.80	1.52	0.367
NarrativeQA	QLoRA	0.335	1.952	0.654	0.80	1.22	0.410
OpenBookQA	LoRA	0.266	0.729	0.194	1.20	6.19	1.645
OpenBookQA	QLoRA	0.300	1.022	0.307	0.80	2.61	0.783
PIQA	LoRA	0.376	1.112	0.418	1.70	4.06	1.529
PIQA	QLoRA	0.212	1.817	0.385	2.00	5.19	1.100
PubMedQA	LoRA	0.384	1.046	0.401	1.50	3.74	1.434
PubMedQA	QLoRA	0.308	0.926	0.286	1.10	3.85	1.188
SAMSum	LoRA	0.299	1.525	0.456	0.90	1.97	0.590
SAMSum	QLoRA	0.498	0.980	0.488	0.90	1.84	0.918
SST-2	LoRA	0.532	0.993	0.528	1.50	2.84	1.510
SST-2	QLoRA	0.437	1.018	0.445	1.30	2.92	1.277
SciTLDR	LoRA	0.273	0.519	0.141	0.90	6.36	1.735
SciTLDR	QLoRA	0.240	0.615	0.147	1.10	7.46	1.788
ViGGO	LoRA	0.584	1.006	0.588	1.10	1.87	1.093
ViGGO	QLoRA	0.497	1.247	0.620	1.10	1.77	0.882
WebNLG	LoRA	0.396	0.511	0.203	0.70	3.46	1.370
WebNLG	QLoRA	0.198	0.268	0.053	0.60	11.30	2.242
WinoGrande	LoRA	0.400	1.091	0.436	1.10	2.52	1.008
WinoGrande	QLoRA	0.337	0.999	0.337	0.50	1.48	0.500
Mean (44 settings)	—	**0.348**	**1.034**	**0.354**	**1.23**	**4.35**	**1.338**

Table I.1: Master λ analysis across 22 tasks $\times$ 2 methods ($N = 44$ settings, Seed 50). **Stage 1 (baseline geometry):** aggregate cosine $\cos(\Delta W_{\text{src}}, \Delta W_{\text{DA}})$ and norm ratio $\|\Delta W_{\text{DA}}\|_F / \|\Delta W_{\text{src}}\|_F$ between transferred adapter and target DA adapter, at $\lambda = 1$ baseline. **Stage 2 (analytic):** λ_{opt} from Eq. (4). **Stage 3 (empirical):** λ^* from joint (CP, λ) search on TestA. **Comparison:** ratio $\lambda^*/\lambda_{\text{opt}}$ (all 44 settings > 1) and effective adapter norm $\|\lambda^* B\|_F / \|A\|_F$ at the empirical optimum.

Table J.1: QLoRA Transfer vs DA on Mistral-7B-Instruct v0.1→v0.3, TestB. Δ = Transfer − DA (positive favours Transfer). † marks the two tasks whose metric differs from the Llama suite.

Task	Metric	Transfer	DA	Δ
AESLC	BLEU	0.086	0.126	−0.040
DialogSum	BLEU	0.223	0.209	+0.014
E2E_NLG	BLEU	0.261	0.308	−0.047
MBPP	BLEU	0.063	0.151	−0.088
MeQSum	BLEU	0.131	0.189	−0.058
NarrativeQA	BLEU	0.230	0.249	−0.019
SAMSum	BLEU	0.215	0.205	+0.009
SciTLDR	BLEU	0.107	0.106	+0.001
ViGGO	BLEU	0.221	0.278	−0.056
WebNLG	BLEU	0.309	0.350	−0.041
ARC-Challenge	acc	0.560	0.560	+0.000
ARC-Easy	acc	0.801	0.795	+0.006
BoolQ	acc	0.884	0.888	−0.004
HellaSwag	acc	0.622	0.628	−0.006
OpenBookQA	acc	0.404	0.388	+0.016
PIQA	acc	0.785	0.801	−0.016
SST-2	acc	0.959	0.929	+0.030
WinoGrande	acc	0.665	0.701	−0.036
GSM8K	BLEU†	0.501	0.444	+0.057
PubMedQA	BLEU†	0.700	0.790	−0.090

Table K.1: DA under a matched 5-value λ grid on Group C. Values in parentheses are the selected λ^{DA}; DA at $\lambda = 1$ is in Table 4. Δ compares Joint Search against the λ-tuned DA; "–" = bootstrap not applicable (per-sample BLEU not preserved). Under the matched budget, Transfer no longer wins significantly on any setting.

Task	Method	Joint	DA_{λ^*}	Δ (p)
CommonGen	LoRA	0.319	0.305 (1.0)	+.014 (–)
CommonGen	QLoRA	0.337	0.346 (0.7)	−.009 (–)
CSQA	LoRA	0.821	0.808 (1.5)	+.013 (.28)
CSQA	QLoRA	0.764	0.810 (1.5)	−.046 (<.001)

Table L.1: Per-task results for finding (i): QLoRA Transfer vs LoRA Transfer across three families, on TestB at each method's joint best (CP^*, λ^*). Δ = QLoRA − LoRA; † marks $|\Delta| > 0.02$. Generated from `master_3family_claim1_method_compare.csv`.

Family	Task	Metric	QLoRA (λ^*)	LoRA (λ^*)	Δ
Llama-3.0→3.1	AESLC	BLEU	0.093 (0.6)	0.078 (0.5)	+0.015
	ARC-Challenge	accuracy	0.551 (1.5)	0.565 (2.0)	-0.014
	ARC-Easy	accuracy	0.794 (2.0)	0.799 (2.0)	-0.005
	BoolQ	accuracy	0.903 (1.3)	0.901 (1.0)	+0.002
	DialogSum	BLEU	0.198 (1.2)	0.194 (1.1)	+0.004
	E2E_NLG	BLEU	0.268 (1.1)	0.259 (1.1)	+0.008
	GSM8K	accuracy	0.772 (0.6)	0.758 (0.5)	+0.014
	HellaSwag	accuracy	0.662 (2.0)	0.660 (2.0)	+0.003
	MBPP	BLEU	0.152 (1.8)	0.131 (1.8)	+0.021†
	MeQSum	BLEU	0.236 (1.4)	0.235 (1.5)	+0.002
	NarrativeQA	BLEU	0.220 (0.8)	0.216 (0.8)	+0.004
	OpenBookQA	accuracy	0.476 (0.8)	0.508 (1.2)	-0.032†
	PIQA	accuracy	0.795 (2.0)	0.796 (1.7)	-0.001
	PubMedQA	accuracy	0.860 (1.1)	0.860 (1.5)	+0.000
	SAMSum	BLEU	0.195 (0.9)	0.186 (0.9)	+0.009
	SST-2	accuracy	0.956 (1.3)	0.963 (1.5)	-0.007
	SciTLDR	BLEU	0.104 (1.1)	0.106 (0.9)	-0.001
	ViGGO	BLEU	0.259 (1.1)	0.238 (1.1)	+0.021†
	WebNLG	BLEU	0.331 (0.6)	0.329 (0.7)	+0.002
	WinoGrande	accuracy	0.709 (0.5)	0.763 (1.1)	-0.054†
Mistral v0.1→v0.3	AESLC	bleu_corpus	0.086 (1.0)	0.091 (0.7)	-0.005
	ARC-Challenge	accuracy	0.560 (1.3)	0.561 (1.0)	-0.002
	ARC-Easy	accuracy	0.801 (2.0)	0.797 (2.0)	+0.004
	BoolQ	accuracy	0.884 (1.0)	0.885 (1.0)	-0.001
	DialogSum	bleu_corpus	0.223 (1.3)	0.226 (1.3)	-0.002
	E2E_NLG	bleu_corpus	0.261 (1.0)	0.269 (0.7)	-0.008
	GSM8K	bleu_corpus	0.501 (1.0)	0.487 (1.0)	+0.014
	HellaSwag	accuracy	0.622 (1.0)	0.624 (1.0)	-0.002
	MBPP	bleu_corpus	0.063 (2.0)	0.064 (2.0)	-0.001
	MeQSum	bleu_corpus	0.131 (1.3)	0.121 (1.3)	+0.009
	NarrativeQA	bleu_corpus	0.230 (0.7)	0.230 (0.7)	+0.000
	OpenBookQA	accuracy	0.404 (2.0)	0.396 (2.0)	+0.008
	PIQA	accuracy	0.784 (1.3)	0.795 (1.5)	-0.011
	PubMedQA	bleu_corpus	0.700 (1.5)	0.710 (1.5)	-0.010
	SAMSum	bleu_corpus	0.214 (1.0)	0.215 (1.0)	-0.000
	SST-2	accuracy	0.959 (2.0)	0.963 (2.0)	-0.005
	SciTLDR	bleu_corpus	0.107 (1.0)	0.099 (1.0)	+0.008
	ViGGO	bleu_corpus	0.221 (1.0)	0.224 (1.0)	-0.002
	WebNLG	bleu_corpus	0.309 (1.0)	0.315 (0.7)	-0.006
	WinoGrande	accuracy	0.665 (1.0)	0.668 (1.0)	-0.003
Qwen2→2.5	AESLC	bleu_corpus	0.043 (2.0)	0.041 (2.0)	+0.002
	ARC-Challenge	accuracy	0.468 (1.5)	0.473 (1.5)	-0.005
	ARC-Easy	accuracy	0.709 (2.0)	0.703 (2.0)	+0.006
	BoolQ	accuracy	0.876 (2.0)	0.876 (2.0)	+0.001
	DialogSum	bleu_corpus	0.121 (2.0)	0.121 (2.0)	+0.000
	E2E_NLG	bleu_corpus	0.257 (2.0)	0.276 (2.0)	-0.018
	GSM8K	bleu_corpus	0.720 (2.0)	0.668 (2.0)	+0.052†
	HellaSwag	accuracy	0.579 (2.0)	0.578 (2.0)	+0.001
	MBPP	bleu_corpus	0.084 (0.7)	0.086 (1.3)	-0.003
	MeQSum	bleu_corpus	0.032 (2.0)	0.036 (2.0)	-0.004
	NarrativeQA	bleu_corpus	0.074 (2.0)	0.076 (2.0)	-0.002
	OpenBookQA	accuracy	0.340 (1.5)	0.336 (1.7)	+0.004
	PIQA	accuracy	0.739 (1.0)	0.733 (2.0)	+0.005
	PubMedQA	bleu_corpus	0.520 (2.0)	0.560 (2.0)	-0.040†
	SAMSum	bleu_corpus	0.146 (2.0)	0.140 (2.0)	+0.006
	SST-2	accuracy	0.940 (2.0)	0.933 (2.0)	+0.007
	SciTLDR	bleu_corpus	0.050 (2.0)	0.051 (2.0)	-0.001
	ViGGO	bleu	0.183 (2.0)	0.180 (1.3)	+0.002
	WebNLG	bleu_corpus	0.339 (2.0)	0.329 (2.0)	+0.009
	WinoGrande	accuracy	0.615 (2.0)	0.611 (1.7)	+0.003

Table L.2: Per-task results for finding (iii): NF4 vs BF16 deployment of the same QLoRA-trained adapter, across three families. Δ = NF4 − BF16; † marks $|\Delta| > 0.02$. Generated from `master_3family_claim3_nf4_deploy.csv`.

Family	Task	Metric	BF16 (λ)	NF4 (λ)	Δ
Llama-3.0→3.1	MeQSum	bleu	0.251 (1.4)	0.224 (1.4)	-0.027†
	SAMSum	bleu	0.190 (0.9)	0.180 (0.9)	-0.009
	WebNLG	bleu	0.324 (0.6)	0.322 (0.6)	-0.002
	BoolQ	accuracy	0.860 (1.3)	0.853 (1.3)	-0.007
	HellaSwag	accuracy	0.583 (2.0)	0.573 (2.0)	-0.010
	SST-2	accuracy	0.938 (1.3)	0.936 (1.3)	-0.002
	ViGGO	bleu	0.220 (1.1)	0.204 (1.1)	-0.015
	AESLC	bleu	0.085 (0.6)	0.085 (0.6)	+0.000
	DialogSum	bleu	0.192 (1.2)	0.189 (1.2)	-0.004
	E2E_NLG	bleu	0.260 (1.1)	0.256 (1.1)	-0.004
	MBPP	bleu	0.140 (1.8)	0.141 (1.8)	+0.002
	NarrativeQA	bleu	0.209 (0.8)	0.202 (0.8)	-0.008
	SciTLDR	bleu	0.095 (1.1)	0.101 (1.1)	+0.006
	ARC-Challenge	accuracy	0.536 (1.5)	0.536 (1.5)	+0.000
	ARC-Easy	accuracy	0.833 (2.0)	0.831 (2.0)	-0.002
	GSM8K	gsm8k_permissive	0.730 (0.6)	0.664 (0.6)	-0.066†
	OpenBookQA	accuracy	0.336 (0.8)	0.344 (0.8)	+0.008
	PIQA	accuracy	0.800 (2.0)	0.794 (2.0)	-0.005
	PubMedQA	yes_no_maybe_permissive	0.810 (0.9)	0.840 (0.9)	+0.030†
	WinoGrande	accuracy	0.754 (0.5)	0.768 (0.5)	+0.014
Mistral v0.1→v0.3	BoolQ	accuracy	0.884 (1.0)	0.876 (1.0)	-0.008
	PIQA	accuracy	0.784 (1.3)	0.791 (2.0)	+0.006
	HellaSwag	accuracy	0.622 (1.0)	0.622 (1.0)	-0.000
	MeQSum	bleu_corpus	0.131 (1.3)	0.120 (1.3)	-0.011
	SAMSum	bleu_corpus	0.214 (1.0)	0.220 (1.0)	+0.005
	WebNLG	bleu_corpus	0.309 (1.0)	0.306 (1.0)	-0.003
	ARC-Challenge	accuracy	0.560 (1.3)	0.567 (2.0)	+0.007
	ARC-Easy	accuracy	0.801 (2.0)	0.801 (2.0)	+0.000
	OpenBookQA	accuracy	0.404 (2.0)	0.420 (2.0)	+0.016
	GSM8K	bleu_corpus	0.501 (1.0)	0.460 (1.0)	-0.041†
	WinoGrande	accuracy	0.665 (1.0)	0.675 (1.5)	+0.009
	SST-2	accuracy	0.959 (2.0)	0.963 (1.7)	+0.005
	PubMedQA	bleu_corpus	0.700 (1.5)	0.740 (1.5)	+0.040†
	MBPP	bleu_corpus	0.063 (2.0)	0.064 (2.0)	+0.001
	SciTLDR	bleu_corpus	0.107 (1.0)	0.092 (1.3)	-0.015
	DialogSum	bleu_corpus	0.223 (1.3)	0.210 (1.6)	-0.013
	AESLC	bleu_corpus	0.086 (1.0)	0.081 (0.7)	-0.005
	NarrativeQA	bleu_corpus	0.230 (0.7)	0.225 (0.7)	-0.005
	E2E_NLG	bleu_corpus	0.261 (1.0)	0.259 (1.0)	-0.003
	ViGGO	bleu	0.221 (1.0)	0.228 (1.0)	+0.007
Qwen2→2.5	BoolQ	accuracy	0.876 (2.0)	0.872 (2.0)	-0.005
	PIQA	accuracy	0.739 (1.0)	0.737 (2.0)	-0.002
	HellaSwag	accuracy	0.579 (2.0)	0.575 (1.7)	-0.004
	ARC-Challenge	accuracy	0.468 (1.5)	0.459 (2.0)	-0.009
	ARC-Easy	accuracy	0.709 (2.0)	0.726 (2.0)	+0.018
	OpenBookQA	accuracy	0.340 (1.5)	0.344 (2.0)	+0.004
	GSM8K	bleu_corpus	0.720 (2.0)	0.687 (2.0)	-0.033†
	WinoGrande	accuracy	0.615 (2.0)	0.615 (1.7)	+0.000
	SST-2	accuracy	0.940 (2.0)	0.947 (2.0)	+0.007
	PubMedQA	bleu_corpus	0.520 (2.0)	0.430 (2.0)	-0.090†
	MeQSum	bleu_corpus	0.032 (2.0)	0.029 (2.0)	-0.003
	SAMSum	bleu_corpus	0.146 (2.0)	0.145 (2.0)	-0.000
	WebNLG	bleu_corpus	0.339 (2.0)	0.335 (2.0)	-0.004
	MBPP	bleu_corpus	0.084 (0.7)	0.091 (2.0)	+0.007
	SciTLDR	bleu_corpus	0.050 (2.0)	0.054 (2.0)	+0.004
	DialogSum	bleu_corpus	0.121 (2.0)	0.114 (2.0)	-0.007
	AESLC	bleu_corpus	0.043 (2.0)	0.057 (2.0)	+0.014
	NarrativeQA	bleu_corpus	0.074 (2.0)	0.059 (2.0)	-0.015
	E2E_NLG	bleu_corpus	0.257 (2.0)	0.274 (2.0)	+0.017
	ViGGO	bleu	0.183 (2.0)	0.171 (2.0)	-0.012