Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

An Artifact Audit of Budget-Dependent LoRA Rank Comparisons

Haolun Tang
Shanghai Dianji University
Jingyi Zhan
Shanghai Dianji University
Yan Feng
Shanghai Dianji University
Zhipeng Chen
chenzhipeng@sdju.edu.cn

Shanghai Dianji University

Article

Keywords: LoRA, parameter-efficient fine-tuning, few-shot classification, compute normalization, artifact audit

Posted Date: August 20th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10426794/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Scientific Reports on November 5th, 2026. See the published version at https://doi.org/10.1038/s41598-026-78535-w.

An Artifact Audit of Budget-Dependent LoRA Rank Comparisons

Haolun Tang[1], Jingyi Zhan[1], Yan Feng[1*], Zhipeng Chen[1*]

[1]School of Electronic and Information Engineering, Shanghai Dianji University, Shanghai, China.

*Corresponding author(s). E-mail(s): fengyan@sdju.edu.cn; chenzhipeng@sdju.edu.cn;
Contributing authors: tanghl026@163.com; 2274252753@qq.com;

Abstract

This paper reports an audited study of artifacts produced under one repository-specific compute-normalization rule for LoRA rank comparison. The design objective is narrow: ranks are compared under an inverse-rank update schedule, so larger adapters receive sharply fewer optimizer steps by construction. The original comparative evidence comes from 38 completed artifacts, comprising 30 head-only baseline runs and 8 LoRA runs, with the original rank comparisons restricted to completed `sst_2` runs. To address the main audit gaps, we add 58 fresh LoRA reruns: 24 rank-decoupled reruns on the original `sst_2` eight-example-per-class slice, 24 cross-slice diagnostics covering two additional `sst_2` draws and one `ag_news` draw, and a 10-run budget-response curve for ranks 2 and 64. Under the original inverse-rank steps on the first repair slice, rank 2 remains strongest on average (0.7729 test accuracy versus 0.5631, 0.5031, and 0.4969 for ranks 8, 16, and 64). Under the equal-42-step control, rank 64 becomes strongest (0.7836 versus 0.5852, 0.5631, and 0.5814). The cross-slice diagnostics show the same budget dependence, while the response curve shows that rank 64 rises from 0.4908 at one update to 0.6411 at 42 updates and then largely saturates, whereas rank 2 rises to 0.7397 only at 682 updates. The contribution is therefore diagnostic rather than a broad rank recommendation: this normalization rule can make larger ranks look worse through extreme undertraining, and the observed ordering is highly budget-rule dependent.

Keywords: LoRA, parameter-efficient fine-tuning, few-shot classification, compute normalization, artifact audit

1 Introduction

Parameter-efficient adaptation has become a standard way to specialize large pretrained language models without incurring the full cost of end-to-end fine-tuning. Early work on adapters showed that task-specific capacity can be introduced through small bottleneck modules while leaving most pretrained parameters unchanged, thereby reducing storage and transfer costs across tasks (Houlsby et al. 2019). Subsequent approaches expanded this design space through prompt tuning, prefix tuning, P-tuning, and BitFit, each placing a small set of trainable parameters at a different point in an otherwise frozen model (Li and Liang 2021; Lester et al. 2021; Liu et al. 2022b; Ben Zaken et al. 2022). Among these methods, low-rank adaptation (LoRA) has been especially influential because it replaces dense weight updates with trainable low-rank factors inserted into existing matrices, yielding a simple and hardware-friendly adaptation mechanism (Hu et al. 2021).

In LoRA, rank is often treated as a capacity knob: higher rank is expected to provide greater expressive flexibility and therefore to help, or at least not hurt, when other conditions are held fixed (Hu et al. 2021; Zhang et al. 2023). That intuition becomes harder to interpret when training opportunity also changes with rank. Higher-rank adapters contain more trainable parameters, so naive comparisons can mix two effects: any consequence of rank itself and any consequence of giving larger adapters a larger effective update budget. This distinction is especially relevant in low-data classification, where optimization instability and weak supervision can dominate small design choices (Mosbach et al. 2021; Aghajanyan et al. 2021; Schick and Schütze 2021; Gao et al. 2021; Zhang et al. 2021; Liu et al. 2022a). For this reason, the central object of study here is not LoRA rank in the abstract, but one concrete compute-normalization rule for comparing fixed-rank LoRA artifacts under a matched training proxy.

Existing work provides useful background but not a direct answer to that audit question. LoRA established low-rank reparameterization as a strong default for parameter-efficient transfer, while AdaLoRA argued that adaptive allocation of rank budget across layers can improve efficiency relative to a fixed-rank design (Hu et al. 2021; Zhang et al. 2023). Related work on rsLoRA likewise suggests that scaling behavior across ranks requires care rather than naive extrapolation from standard low-rank formulations (Kalajdzievski 2023). More broadly, neighboring parameter-efficient methods indicate that downstream behavior is not a simple function of raw trainable parameter count, because adapters, prompts, prefixes, and selective bias updates impose different inductive biases and optimization geometries (Houlsby et al. 2019; Li and Liang 2021; Lester et al. 2021; Liu et al. 2022b; Ben Zaken et al. 2022). The present study does not attempt to resolve those broader questions. Instead, it asks whether a specific matched-compute proxy can reverse the usual expectation that larger LoRA rank should be safer or better.

This paper therefore presents a deliberately narrow artifact audit of fixed-rank LoRA runs under one compute-normalization rule, centered on recovered SST-2 artifacts rather than a broad few-shot benchmark sweep. The original evidence is limited: it comes from 8 completed LoRA artifacts, with one completed draw per shot count,

embedded within a larger artifact record of 38 runs that also includes 30 baseline artifacts for experimental context. Because that original subset confounds rank with seed and budget, the revised audit adds two targeted local reruns. The first rerun uses the same `sst_2` eight-example-per-class slice but decouples seed from rank across three shared seeds and compares the original inverse-rank step schedule against an equal-42-step control. The second rerun adds two more `sst_2` slice draws and one `ag_news` slice under the same two budget regimes. These additions improve the evidence for budget dependence while still making the study best read as an audit of a budget rule rather than a general evaluation of LoRA rank in few-shot classification.

Within that narrow evidential frame, 12 paired bootstrap rank comparisons were formed on shared SST-2 test records from fixed trained model pairs. Those comparisons reuse the same small set of completed runs and are therefore descriptive contrasts within a single audited artifact pool, not quasi-independent replications. The resulting intervals are conditional on those completed model pairs and do not quantify between-run training uncertainty. Even so, 11 of the 12 pairwise contrasts favored a lower-rank condition over at least one higher-rank condition. The clearest reversals appear for `sst_2` at examples-per-class 8, where rank 2 exceeded rank 8 by 0.167 accuracy points (95% CI [0.136, 0.197]) and rank 16 by 0.250 points (95% CI [0.200, 0.299]).

The operational record still limits scope. One planned sweep stage produced 0 fresh successful runs and 8 error runs, several originally planned conditions remain unavailable as completed artifacts, and no rsLoRA artifacts were present in the checkout. The paper therefore advances a deliberately narrow claim: within this partial artifact study, the apparent lower-rank advantage is real under the original inverse-rank schedule but repeatedly weakens or reverses under an equal-step control. The substantive implication is not that low rank or high rank is uniformly preferable, but that naive rank ordering can reverse under one compute-normalization rule and should not be treated as self-evident without careful budget accounting, seed-decoupled reruns, slice checks, and alternative budget controls.

2 Related Work

The most relevant background for this study comes from three adjacent but distinct literatures: work on LoRA rank and scaling, work on few-shot instability in encoder classification, and work on how adaptation methods should be compared when parameter counts and optimization budgets differ. The present evidence bears only on the intersection of those threads: standard fixed-rank LoRA in a few-shot encoder-classification setting, evaluated under one explicit compute-normalization rule. It does not speak to the full PEFT landscape or to LoRA variants that change the underlying parameterization.

First, LoRA and rank-scaling papers already caution against interpreting rank as a simple monotone capacity knob. LoRA parameterizes weight updates through low-rank factors and makes rank a visible design variable, but the original formulation also ties outcomes to scaling and optimization choices rather than to rank alone (Hu et al. 2021). This basic intuition is consistent with broader arguments that downstream

adaptation may occupy a much smaller effective subspace than the ambient parameter space (Aghajanyan et al. 2021). Subsequent LoRA-specific work strengthens the point. AdaLoRA reallocates a fixed budget across matrices during training rather than assuming that a uniformly larger fixed rank is inherently better (Zhang et al. 2023). Rank-stabilized LoRA goes further by arguing that the standard LoRA scaling rule can itself disadvantage higher ranks (Kalajdzievski 2023). That implication is central here. If standard scaling suppresses higher-rank adapters, then a lower-rank win under matched compute may reflect a known parameterization pathology of vanilla LoRA rather than a broader property of rank. For that reason, the present study is framed narrowly: it examines how standard fixed-rank LoRA behaves under the implemented comparison rule, not whether larger rank is intrinsically unhelpful across LoRA variants or tuning regimes.

Second, the few-shot classification literature motivates caution about any single-run ordering of adaptation methods. Encoder benchmarks built around BERT and RoBERTa established strong transfer for sentence and document classification, including SST-2 and the broader GLUE suite (Devlin et al. 2019; Liu et al. 2019; Wang et al. 2019; Socher et al. 2013). At the same time, instability under limited supervision is well documented. Mosbach et al. (2021) showed that fine-tuning performance can vary substantially across seeds and hyperparameters, and Zhang et al. (2021) likewise found that few-sample outcomes depend strongly on regime and calibration. Prompt-based few-shot methods such as PET and LM-BFF reinforce the same broader lesson: changing the adaptation interface can materially change low-resource behavior (Schick and Schütze 2021; Gao et al. 2021). These papers do not directly answer the rank question studied here, but they do justify treating rank comparisons in small-sample encoder settings as potentially sensitive to optimization details rather than as clean measurements of representational capacity.

Third, a separate methodological issue concerns fair comparison when PEFT configurations differ in trainable parameter count. Early classification-oriented PEFT methods such as adapters and BitFit showed that strong downstream performance can be obtained with small trainable subsets (Houlsby et al. 2019; Ben Zaken et al. 2022), but they do not by themselves specify what should be held fixed when comparing alternatives of different sizes. The same is true for adjacent PEFT families such as prefix tuning, prompt tuning, P-tuning, and T-Few, which are often studied in generative or sequence-to-sequence settings with different optimization geometry and deployment constraints (Li and Liang 2021; Lester et al. 2021; Liu et al. 2022b,a). For the present paper, the methodological question is not which PEFT family is best overall, but how to interpret fixed-rank LoRA comparisons once a single training-opportunity proxy is imposed. That is a narrower fair-comparison problem than the broader PEFT literature usually addresses.

Taken together, these literatures motivate a restrained claim. Prior work makes it plausible that nominal rank, optimization opportunity, and parameterization are entangled. The completed SST-2 artifact therefore should not be read as showing that lower LoRA rank generally outperforms higher rank in few-shot classification. It more cautiously shows that, under one matched-compute protocol for standard fixed-rank

LoRA, monotone rank gains did not materialize and can even reverse on the completed slice, with paired bootstrap comparisons on shared test records favoring rank 2 over ranks 8 and 16. The most defensible interpretation is therefore methodological rather than universal: fixed-rank LoRA results in this regime are sensitive to how the comparison is parameterized and budgeted, while explanations based on regularization, undertraining, or standard-scaling pathologies remain hypotheses suggested by prior work rather than mechanisms established here.

3 Method

This study examines LoRA rank under a deliberately narrow but auditable design. The original completed evidence consists of eight standard-LoRA sequence-classification artifacts that vary rank while holding the surrounding training recipe as fixed as the available checkout permits. The broader project record includes planning manifests, baseline runs, and execution attempts spanning multiple few-shot text-classification settings, including `ag_news` (Zhang et al. 2015) and `sst_2` (Socher et al. 2013). However, the original matched-rank LoRA evidence recoverable in the analyzed checkout is SST-2 only. The revised audit then adds two local rerun blocks to address the most important threats in the recovered protocol: seed is no longer tied to rank, the original inverse-rank step schedule is compared with an equal-step control, and the budget-rule stress test is repeated on additional slice and dataset conditions. Rather than asking whether LoRA broadly improves over full fine-tuning or over other parameter-efficient methods such as adapters, prefixes, prompts, or bias-only tuning (Houlsby et al. 2019; Li and Liang 2021; Lester et al. 2021; Ben Zaken et al. 2022; Liu et al. 2022b), the study asks how one recorded fixed-budget few-shot adaptation protocol behaves, and whether the observed rank ordering survives targeted budget-rule stress tests.

3.1 Few-shot adaptation setting

The analyzed LoRA artifacts are all sequence-classification runs based on `roberta-base`. They are evaluated on SST-2 under two few-shot regimes, 8 and 32 examples per class, each for a single completed slice draw (`draw_id`=0). Thus, the inferential sample is not a multi-backbone or multi-dataset benchmark sweep; it is a tightly scoped comparison within one benchmark, one backbone family, two shot counts, and one completed draw per shot count. This narrow scope is important for interpreting all subsequent claims.

Each LoRA run starts from a pretrained encoder with a sequence-classification head and applies PEFT-style low-rank updates to the attention modules named `query` and `value`. For a pretrained weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA parameterizes adaptation as

$$W = W_0 + \Delta W, \qquad \Delta W = \frac{\alpha}{r} BA, \tag{1}$$

where $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$ are trainable matrices, r is rank, and α is the LoRA scaling constant (Hu et al. 2021). In the executed launcher, the LoRA configuration sets `lora_alpha` to $\max(8, 2r)$ and dropout to 0.05. This is another rank-linked design choice: the effective scaling factor α/r is 4 for rank 2 and 2 for ranks 8, 16, and

64, so the recorded protocol does not hold LoRA scaling strictly fixed across ranks. The completed artifacts all record learning rate 2×10^{-4}, weight decay 0.01, and effective batch size 64. The launcher seeds training as `draw_id` $+ r$; for the completed draw this yields seeds 2, 8, 16, and 64 for ranks $r \in \{2, 8, 16, 64\}$. Rank is therefore mechanically entangled with both training seed and the rank-2 scaling exception in the analyzed runs. The paper consequently treats rank contrasts as descriptive features of the recorded protocol rather than as cleanly isolated causal effects of rank alone.

The trainable adaptation package includes the inserted LoRA factors together with the task-specific sequence-classification layers used by the PEFT sequence-classification model. The classifier head is therefore not frozen in the analyzed LoRA runs. Because that head is held fixed across rank conditions, it does not itself define the rank contrast, but it does consume part of every optimizer step. It is also not negligible in this small-head setting: subtracting the rank-linear LoRA factor count implied by the artifact table leaves approximately 592,130 trainable classifier-head parameters, which is about 88.9% of the trainable package at rank 2 and 20.1% at rank 64. The inverse-rank budget proxy does not account for this rank-invariant head cost, so it should not be interpreted as matching total trainable-parameter updates. The budget normalization described below is therefore only a proxy for rank-dependent adapter opportunity under a common training recipe, not an equalization of every trainable degree of freedom in the model.

The completed artifact table records eight original LoRA runs in total: four ranks for SST-2 at 8 examples per class and four ranks for SST-2 at 32 examples per class. No rsLoRA artifacts were present in the analyzed checkout, so the method concerns conventional fixed-rank LoRA only (Kalajdzievski 2023). The repository also contains 30 baseline artifacts, all from a separate head-only fine-tuning stage. Those baseline runs are retained only as contextual records in the merged artifact table and for pipeline checking; they do not enter the paired rank bootstrap reported here. The local repair audit adds 58 additional LoRA runs and is reported separately from the recovered artifact inventory so that the original record, seed-decoupled repair evidence, cross-slice diagnostics, and budget-response curve remain distinguishable.

3.2 Matched-compute rank comparison protocol

The central design choice is an operational matched-compute proxy inherited from the project manifest. Its purpose is not to equalize exact hardware cost, FLOPs, or wall-clock time. Instead, it attempts to hold approximately constant a crude measure of rank-conditioned adaptation opportunity by allocating fewer optimizer updates to larger-rank adapters.

The construction begins from the standard observation that when LoRA is inserted into a fixed set of matrices, the number of rank-dependent trainable adapter parameters in one transformed layer scales linearly with r,

$$P_{\mathrm{LoRA}}(r) = r(d_{\mathrm{in}} + d_{\mathrm{out}}). \tag{2}$$

If one then adopts the simplifying approximation that the rank-sensitive portion of training cost is proportional to

$$C(r, c) \propto \text{updates}(r, c) \times r \times S(c), \tag{3}$$

where $S(c)$ summarizes condition-level factors held fixed within a slice, then matching $C(r, c)$ across ranks implies

$$\text{updates}(r, c) \propto \frac{1}{r}. \tag{4}$$

The planning script operationalizes this inverse-rank heuristic with a fixed global constant `DESIRED_UPDATE_BUDGET`=65536 and sets

$$\texttt{max_steps}(r) = \max\left(1, \text{round}\left(\frac{65536}{r}\right)\right). \tag{5}$$

For the rank grid used in the completed LoRA runs, this produces manifest step budgets of 32768, 8192, 4096, and 1024 for ranks $r \in \{2, 8, 16, 64\}$, respectively. The same manifest also records an approximate quantity

$$\texttt{approx_compute_units}(r, c) = \texttt{max_steps}(r) \times \texttt{slice_num_examples}(c) \times 768 \times 4, \tag{6}$$

where 768 is the hidden-size constant and 4 is the target-module count used by the planning code. This quantity is not a measured FLOP count. It is a bookkeeping proxy intended to scale with dataset slice size and with a fixed backbone-width surrogate while leaving the inverse-rank dependence in `max_steps`(r) as the dominant normalization rule.

Two features of this construction explain the extreme realized disparities such as 682 versus 1 update. First, the proxy intentionally shrinks the step budget by a factor of roughly $1/r$, so ranks that differ by a factor of 32 also begin with a 32-fold disparity in planned steps. Second, the original `roberta-base` metrics table stores the operative post-launcher fields `resolved_matched_max_steps` and `actual_update_steps`; these recorded fields, not the larger manifest values above, define the budgets analyzed in this paper. The recovered artifacts do not preserve every intermediate setting needed to rederive those post-launcher counts from first principles, so the audit treats the recorded values as the authoritative execution record. For SST-2 at 8 examples per class, the recorded update counts are 682, 42, 10, and 1 for ranks 2, 8, 16, and 64. For SST-2 at 32 examples per class, they are 2730, 170, 42, and 2. All eight runs use the same effective batch size of 64, so the intended normalization acts through the number of optimizer updates rather than through a changing batch schedule. The repair scripts reproduce the original eight-example-per-class schedule by using exactly the recorded 682/42/10/1 counts and then compare it with an equal-42-step control.

The important methodological point is therefore limited but explicit. The matched quantity is not literal compute; it is an inherited proxy that treats rank-dependent update opportunity as the object to be approximately normalized. The proxy is auditable because its ingredients are fixed in code and recorded in artifacts, but it is also contestable. It omits measured runtime, omits token-throughput variation,

omits optimizer-state and activation-memory effects, omits the constant cost of the jointly trained classifier head, and was originally generated with a planning backbone (`distilbert-base-uncased`) that differs from the completed analysis backbone (`roberta-base`). The study can therefore evaluate only what happens under this particular proxy, not whether this proxy is uniquely correct or whether the observed rank ordering would survive alternative equal-compute definitions (Hu et al. 2021; Zhang et al. 2023).

The executed training loop runs until the recorded step budget is exhausted and does not implement early stopping. Validation predictions and validation accuracy are computed after training, not used to terminate training, and not used by the launcher to adapt the step budget online. Within the analyzed checkout, the resolved budget is therefore fixed before a given run starts and is not altered in response to that run's validation or test outcome. At the same time, the study cannot establish from the recovered artifacts alone whether the original budget manifest was designed independently of broader exploratory work elsewhere in the project history. The appropriate claim is thus procedural rather than causal: the checkout enforces a fixed rank-specific budget per completed condition, but it does not by itself validate that this budget is outcome-independent or scientifically privileged.

Within that framing, the comparison is local to a fixed task condition. Let c index a completed condition defined by dataset, shot count, draw, backbone, and training mode. In the analyzed subset there are only two such conditions,

$$c \in \{(\texttt{sst_2}, 8, 0, \texttt{roberta-base}, \texttt{lora}),\ (\texttt{sst_2}, 32, 0, \texttt{roberta-base}, \texttt{lora})\}. \quad (7)$$

Within each condition, all $\binom{4}{2} = 6$ pairwise rank contrasts are evaluated, yielding 12 descriptive rank comparisons in total across the two completed shot regimes. Figure 1 summarizes this restricted pipeline: rank varies within condition, matched step budgets are taken from the recorded artifacts, and all inference is performed on the resulting paired LoRA comparisons.

Table 1 makes the analyzed scope explicit. The table emphasizes the subset that identifies the rank contrast directly and treats the larger project record as context only.

3.3 Seed-decoupled and cross-slice repair audits

The revised audit adds a small rerun designed specifically to test the two main threats identified above. It uses the same normalized SST-2 dataset, the same `examples_per_class`=8 slice, the same four ranks, the same `roberta-base` backbone, and the same LoRA target modules. The training slice is selected by saved `slice_record_ids` rather than by stale integer indices, which avoids the index mismatch that caused the earlier rsLoRA repair attempt to fail after dataset normalization. Three seeds, 101, 202, and 303, are shared across all ranks, so initialization seed is no longer mechanically determined by rank.

Two budget regimes are run. The first reproduces the original inverse-rank step counts for the eight-example-per-class slice: 682, 42, 10, and 1 optimizer updates for ranks 2, 8, 16, and 64. The second uses an equal-step control with 42 optimizer updates

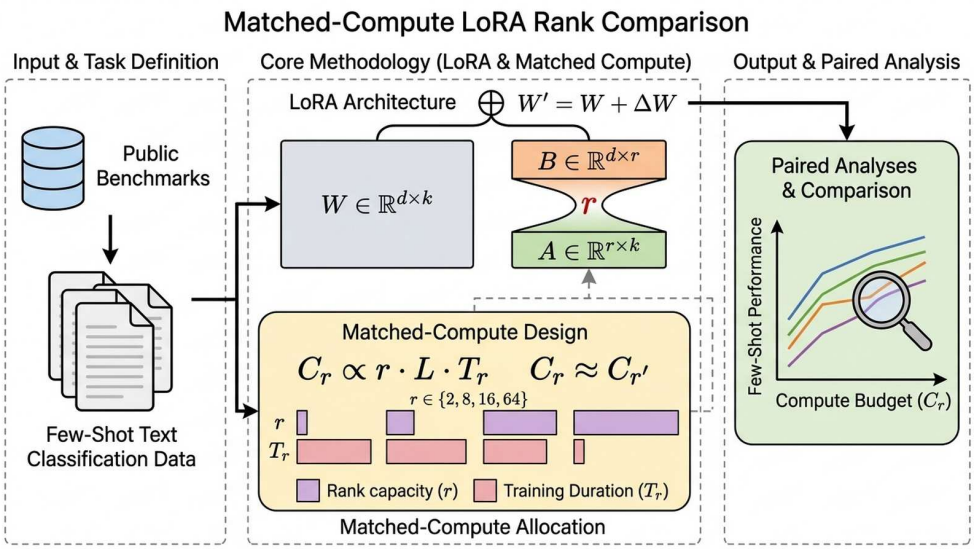


Fig. 1 Overview of the completed matched-compute pipeline. The analyzed subset contains eight `roberta-base` LoRA sequence-classification artifacts on SST-2, spanning ranks $r \in \{2, 8, 16, 64\}$ at 8 and 32 examples per class. For each completed shot condition, rank-specific update budgets are taken from the recorded matched-step counts, after which pairwise rank contrasts within that condition are evaluated by paired bootstrap on shared test records. The evidential target is the behavior of this audited budget rule on the recovered artifacts, not a general causal estimate of LoRA rank.

for every rank. All runs use the same learning rate, weight decay, LoRA scaling rule, dropout, batch size, gradient accumulation, validation subset, and held-out SST-2 test set. This rerun is not broad enough to establish a general rank law, but it is sufficient to distinguish two explanations for the original artifact: a stable low-rank advantage versus a rank ordering induced by the inverse-rank step schedule.

The cross-slice repair audit reuses the same code path and recorded inverse-rank schedule but changes the condition rather than the seed. It adds two further SST-2 eight-example-per-class draws (`draw_id`=1 and 2) and one AG News eight-example-per-class draw (`draw_id`=0), all with seed 404 shared across ranks. The AG News run is included as a diagnostic second dataset, not as a complete benchmark replication: it uses one draw, one seed, the same `roberta-base` backbone, and the same two budget regimes. Its role is to test whether the most direct mechanism claim, that giving rank 64 only one update can suppress performance, is visible outside the original SST-2 slice.

Finally, a small budget-response audit varies the number of optimizer updates directly for ranks 2 and 64 on the original SST-2 eight-example-per-class slice. With seed 505 fixed across conditions, both ranks are trained for 1, 10, 42, 170, and 682 updates. This curve is not a full fairness study, but it separates two questions that the equal-step control alone cannot separate: whether rank 64 is damaged by the one-update cap, and whether longer budgets continue to favor the same rank once both ranks receive enough optimization opportunity.

Table 1 Methodological scope of the analyzed matched-compute LoRA subset.

Component	Setting
Backbone entering LoRA analysis	`roberta-base` only
Original dataset entering paired artifact analysis	`sst_2` only
Completed shot counts	8 and 32 examples per class
Completed slice draws	`draw_id`=0 only
LoRA insertion points	Attention modules named `query` and `value`
Trainable adaptation components	LoRA factors plus sequence-classification head
Rank grid	$r \in \{2, 8, 16, 64\}$
Recorded optimizer settings	Learning rate 2×10^{-4}, weight decay 0.01, effective batch size 64
Seed rule in completed LoRA runs	`seed` = `draw_id` + r, hence seed is confounded with rank
Matched-compute proxy	Inverse-rank step budget inherited from `condition_budget_table.csv`; not measured FLOPs or wall-clock time
Early stopping	None in the executed fixed-budget training loop
Validation use during training	Evaluated after training; not used for online stopping or budget adjustment
Recorded matched step budgets	SST-2, epc=8: 682/42/10/1; SST-2, epc=32: 2730/170/42/2 for ranks 2/8/16/64
LoRA artifacts entering inference	8
Seed-decoupled repair reruns	24 fresh LoRA runs on SST-2 epc=8, draw 0, with shared seeds 101/202/303
Cross-slice repair reruns	24 fresh LoRA runs on SST-2 draws 1/2 and AG News draw 0, seed 404
Budget-response reruns	10 fresh LoRA runs on SST-2 epc=8, draw 0, seed 505, for ranks 2 and 64 across 1/10/42/170/682 steps
Repair-audit budget regimes	Original 682/42/10/1 schedule and equal 42-step control
Contextual baseline artifacts	30, excluded from paired inference
Paired rank comparisons	12 total = $2 \times \binom{4}{2}$, treated as structured within-condition contrasts rather than independent experiments

3.4 Evaluation and statistical inference

The primary outcome is held-out test accuracy. For each completed LoRA run, the analysis script reconstructs predictions from the saved `test_predictions.jsonl` artifact and aligns runs by shared `record_id`. The observational unit for the bootstrap is therefore the individual test record, not the run. For a low-rank run a and a higher-rank run b within the same condition c, let $c_i^{(a)}, c_i^{(b)} \in \{0, 1\}$ indicate whether each

model predicts shared test record i correctly. The paired estimand is

$$\hat{\Delta}_{ab} = \frac{1}{n} \sum_{i=1}^{n} \left(c_i^{(a)} - c_i^{(b)} \right), \tag{8}$$

where n is the number of shared test records. In the completed SST-2 comparisons, $n = 872$ for every rank pair because all runs are evaluated on the same SST-2 test set. The analysis then performs 2000 paired bootstrap resamples of these shared records to obtain a confidence interval and the bootstrap probability that the lower-rank condition exceeds the higher-rank condition.

These intervals quantify uncertainty from finite test-set sampling conditional on the two fitted models being treated as fixed. They do not quantify training-run uncertainty, seed sensitivity, or the variability that would arise under repeated reruns of the same rank condition. This distinction is essential here because only one completed training run is available per rank within each shot condition, and the seed itself is deterministically tied to rank. The bootstrap therefore supports statements about paired predictive differences for the recovered model pairs, not a population-level claim that one rank would systematically dominate another across independent retraining.

This construction explains the count of 12 pairwise comparisons exactly. There are four completed ranks within each of two completed shot conditions, so each condition contributes six pairwise contrasts: $(2, 8)$, $(2, 16)$, $(2, 64)$, $(8, 16)$, $(8, 64)$, and $(16, 64)$. These contrasts are not statistically independent, because comparisons within the same shot condition reuse the same test records and the same fitted runs. They are therefore treated as a structured descriptive decomposition of two condition-level findings rather than as 12 quasi-independent experiments. The evidential emphasis falls on whether the within-condition ordering is stable across the available rank pairs, not on the nominal count of comparisons.

Taken together, these procedures define a narrow evidential target. The method does not establish that lower or higher rank is universally preferable and does not claim that inverse-rank step matching is the only defensible compute normalization. It evaluates a specific recorded protocol for few-shot LoRA adaptation on a small completed SST-2 subset and then runs targeted stress tests of that protocol. Under that narrower and more reproducible framing, the methodological contribution is the insistence that rank comparisons should be conditioned on an explicit budget proxy rather than on nominal hyperparameter parity alone, while also making clear that the resulting evidence remains conditional on the chosen proxy, the fixed classifier-head training recipe, the planning-to-execution transfer across backbones, and the limited number of rerun seeds and slices (Hu et al. 2021; Zhang et al. 2023; Mosbach et al. 2021).

4 Experiments

4.1 Experimental Setup

This section reports an artifact study of one repository-implemented compute-normalization rule, not a general benchmark on LoRA rank in few-shot classification.

The project plan and task manifests describe a broader multi-dataset sweep, but the completed original LoRA evidence is much narrower. The direct empirical basis for the recovered rank comparison is eight completed LoRA runs on `sst_2`, spanning two few-shot settings (`examples_per_class` = 8 and 32), one slice identifier (`draw_id` = 0), one backbone (`roberta-base`), and four target ranks ($r \in \{2, 8, 16, 64\}$). There is one completed original run per recovered condition. The revised evidence then adds 58 fresh local reruns: 24 on the original eight-example-per-class slice to test seed coupling and budget sensitivity, 24 on two additional SST-2 slice draws plus one AG News diagnostic draw to test whether the budget effect is visible outside the original slice, and 10 budget-response runs for ranks 2 and 64.

The central protocol is the repository-defined matched-compute proxy. In the budget generator, each condition table row first computes

$$\texttt{epoch_steps} = \left\lceil \frac{\texttt{slice_num_examples}}{\texttt{effective_batch_size}} \right\rceil,$$
$$\texttt{max_steps}(r) = \max\left(1, \operatorname{round}\left(\frac{\texttt{DESIRED_UPDATE_BUDGET}}{r}\right)\right),$$

and then stores

$$\texttt{approx_compute_units}(r) = \texttt{max_steps}(r) \times \texttt{slice_num_examples} \times \texttt{hidden_size} \times \texttt{target_module_count}.$$

In words, the matched quantity is not wall-clock time, token exposure, or an exact FLOP count. It is an implementation-level proxy proportional to the number of update steps, the number of training examples in the slice, the backbone hidden width, and the number of targeted projection modules. For the completed `roberta-base` runs, the target modules are the attention `query` and `value` projections, which the metrics record as 24 targeted modules in total across the encoder stack. Because $\texttt{max_steps}(r) \propto 1/r$ in the manifest construction, the proxy already builds an inverse rank–step schedule into the design before any outcomes are observed. The recovered artifacts do expose the quantities most relevant to this audit table: realized update steps, trainable parameter counts, test accuracy, validation accuracy, and macro F1 for the rerun diagnostics. They do not expose a comparable wall-clock ledger or measured FLOP trace across all completed conditions, so the paper cannot claim a runtime-level fairness comparison and instead stays with the auditable step-budget record.

The completed artifacts then record two related fields: the inherited budget proxy `matched_compute_units` and the realized cap `resolved_matched_max_steps`. The manifest first applies $\texttt{max_steps}(r) = \max(1, \operatorname{round}(65536/r))$, while the final analysis relies on the post-launcher fields recorded in the metrics table. What matters scientifically is the resulting severity of the schedule. In the completed `sst_2` runs, backbone, slice, effective batch size, learning rate, weight decay, and target-module pattern are held fixed while rank varies together with the matched step cap. The effective batch size remains 64 throughout. At `examples_per_class` = 8, ranks 2, 8, 16, and 64 realize 682, 42, 10, and 1 update steps, respectively; at `examples_per_class` = 32, they

realize 2730, 170, 42, and 2. Since the `sst_2` training slices contain 16 and 64 examples in those two settings, and the effective batch size is 64, each update step is also one full pass over the slice. The extreme disparity is therefore literal: in the completed pilot subset, the highest-rank condition sometimes receives only one or two passes over the entire training slice. This makes severe undertraining of higher-rank adapters a directly testable explanation for the observed ordering, not a nuisance detail.

Several additional protocol details matter for interpretation. First, the available metrics report `completed_schedule=true` and `stop_reason=matched_max_steps`, which supports reading these budgets as fixed caps rather than outcome-adaptive stopping decisions. Second, the accessible artifacts do not show early stopping, best-checkpoint selection, or rank-specific budget adjustment based on validation outcomes; validation predictions and validation accuracies are recorded, but the available logs support treating them as post-training evaluations rather than as a model-selection device. Third, the completed design couples seed to rank: the training runner sets `run_seed = draw_id + rank`, and the only completed LoRA draw is `draw_id`=0. Rank is therefore not orthogonal to initialization in the available artifacts. Fourth, trainable-parameter counts rise sharply with rank, from 665,858 at r=2 to 2,951,426 at r=64, but the checkout does not separately itemize how classifier-head updates enter the proxy. For that reason, the protocol is described here in the implementation-level terms the repository exposes, rather than as a stronger claim that all relevant notions of training opportunity have been equalized.

Within this narrow evidential frame, LoRA is examined as one member of a broader parameter-efficient tuning literature that includes adapters (Houlsby et al. 2019), prefix tuning (Li and Liang 2021), prompt tuning (Lester et al. 2021), P-tuning (Liu et al. 2022b), and BitFit (Ben Zaken et al. 2022), with later LoRA variants such as AdaLoRA and rsLoRA explicitly motivated by concerns about rank allocation and scaling (Hu et al. 2021; Zhang et al. 2023; Kalajdzievski 2023). The empirical question answered here is therefore limited by construction: within the completed artifacts and targeted reruns produced under this one matched-compute proxy, what accuracy ordering is observed as fixed LoRA rank increases, and does that ordering survive an equal-step budget control? The section does not answer that question for other backbones, other PEFT methods, or other compute-matching rules.

The broader inventory also contains 30 non-LoRA baseline artifacts, all of them head-only fine-tuning runs across `ag_news` and `sst_2`. They are retained as contextual controls for training difficulty, not as direct evidence in the LoRA rank analysis below. On `sst_2`, the five-draw head-only baseline averages 0.540 test accuracy at 8 examples per class and 0.575 at 32 examples per class, with substantial run-to-run variation. The best completed LoRA rank-2 runs exceed those contextual means, but this comparison cannot isolate the effect of rank because the LoRA subset lacks repeated seeds and shares neither the same mode nor the same seed schedule as the head-only baseline.

The revised audit adds targeted reruns rather than expanding the claim to a broad benchmark. The first block completes 24 fresh LoRA runs on the same `sst_2` eight-example-per-class slice: three rank-decoupled seeds, four ranks, and two budget regimes. The second block completes 24 more runs on two additional `sst_2` draws and one `ag_news` draw, all at eight examples per class with one shared seed. In both

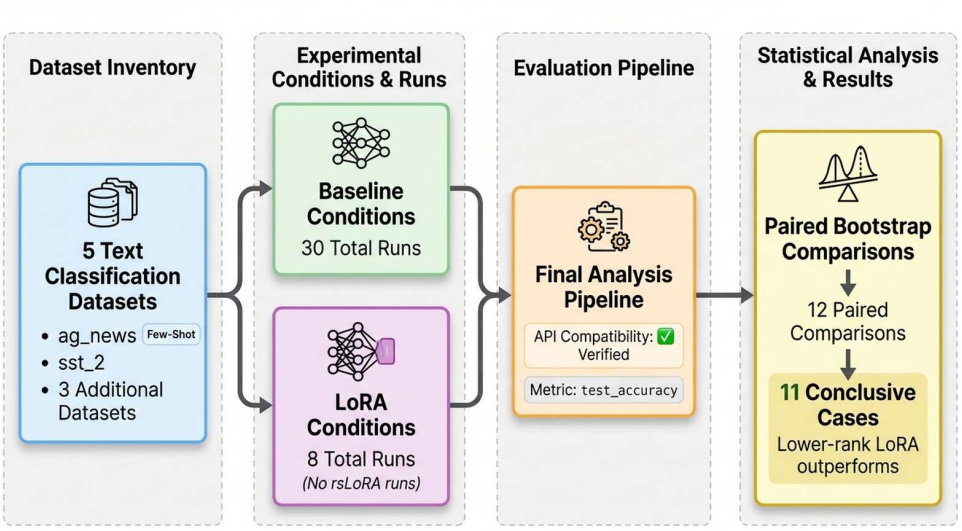


Fig. 2 Pipeline for the completed artifact subset and repair audits analyzed here. A broader planned sweep contracts, in the available checkout, to a small set of completed `sst_2` LoRA runs; targeted local reruns then stress-test the recorded inverse-rank budget rule. Within each slice, the repository's matched-compute procedure defines a proxy budget and then realizes sharply smaller step caps for higher ranks before paired comparisons are computed on shared test records.

blocks, the first regime reproduces the original inverse-rank step counts, while the second assigns 42 optimizer updates to every rank. The third block completes a 10-run budget-response curve for ranks 2 and 64 on the original eight-example-per-class slice. These reruns are used as repair audits for the original evidence, not as a replacement for a full multi-dataset replication.

Figure 2 summarizes this pipeline. The motivation for controlling training opportunity is consistent with the known instability of few-shot fine-tuning on pretrained encoders (Mosbach et al. 2021; Zhang et al. 2021; Liu et al. 2022a). Even so, the evidential status of this section remains deliberately limited. It analyzes completed artifacts and targeted local reruns, does not pool across unexecuted conditions, and treats the gap between the planned study and the completed subset as a substantive limitation rather than a recoverable omission. The question answered below is therefore specific: what happened when LoRA rank increased under the implemented matched-compute normalization, and how did the observed ordering change under an equal-step repair control?

4.2 Main Results

Table 2 reports every completed LoRA condition in the analyzable subset. The limitation is immediate: there is one completed run per LoRA condition, all on `sst_2`, so the section contains no independent within-condition replication and no direct estimate of LoRA training variance. The exact test accuracies nevertheless make the realized ordering transparent. At `examples_per_class` = 8, the completed runs

Table 2 All completed LoRA runs in the analyzed checkout. Each row is one completed condition; no LoRA setting has repeated runs, so no within-condition standard deviation is available.

Dataset	EPC	Draw	Rank	Update steps	Trainable params.	Test acc.	Val. acc.
`sst_2`	8	0	2	682	665,858	0.7408	0.7361
`sst_2`	8	0	8	42	887,042	0.5734	0.5459
`sst_2`	8	0	16	10	1,181,954	0.4908	0.4422
`sst_2`	8	0	64	1	2,951,426	0.5092	0.5578
`sst_2`	32	0	2	2730	665,858	0.9060	0.8840
`sst_2`	32	0	8	170	887,042	0.8922	0.8744
`sst_2`	32	0	16	42	1,181,954	0.7844	0.8016
`sst_2`	32	0	64	2	2,951,426	0.5149	0.5563

Table 3 Budget-response audit on SST-2 epc=8, draw 0, seed 505. Both ranks use the same fixed training recipe while only the update budget changes.

Rank	Steps	Test acc.	Macro F1	Train loss
2	1	0.4908	0.3292	1.3859
2	10	0.5310	0.4203	1.3838
2	42	0.5092	0.3374	1.3379
2	170	0.6755	0.6451	0.3948
2	682	0.7397	0.7305	0.1116
64	1	0.4908	0.3292	1.3859
64	10	0.5573	0.5014	1.3762
64	42	0.6411	0.6092	0.6776
64	170	0.6388	0.6014	0.1752
64	682	0.6628	0.6299	0.0659

are ordered r=2 (0.7408) > r=8 (0.5734) > r=64 (0.5092) > r=16 (0.4908). At `examples_per_class` = 32, the ordering is r=2 (0.9060) > r=8 (0.8922) > r=16 (0.7844) > r=64 (0.5149). On the completed slice, the best-performing condition is the lowest rank in both shot settings, and the highest rank is never competitive. That pattern is descriptive rather than confirmatory: it combines rank, a severe inverse rank–step schedule, and rank-coupled seed assignment in one realized pilot subset.

Table 3 reports the budget-response curve on the original SST-2 eight-example-per-class slice. It confirms that the one-update cap is not a benign implementation detail for rank 64: accuracy rises from 0.4908 at one update to 0.6411 at 42 updates. It also prevents overinterpretation of the equal-step result. Rank 64 does not continue improving substantially after 42 updates in this seed, while rank 2 improves most at longer budgets and reaches 0.7397 at 682 updates. Thus the repair evidence supports a specific mechanism claim, severe undertraining under the original rank-64 cap, without establishing that equal update count is the definitive fair-compute target.

To summarize that ordering without multiplying the evidential unit, the analysis constructs all pairwise rank comparisons within each completed shot setting. This produces 12 contrasts in total, but they should be read only as a descriptive decomposition of the two shot-specific orderings in Table 2, not as 12 quasi-independent

Table 4 Paired bootstrap decomposition of the completed LoRA ordering. Δ accuracy is defined as lower-rank minus higher-rank accuracy on shared test records. The intervals are conditional on fixed trained model pairs and should not be interpreted as independent training-run evidence.

EPC	Low rank	High rank	Δ acc.	95% CI	$P(\text{low} > \text{high})$
8	2	8	0.1674	[0.1365, 0.1972]	1.0000
8	2	16	0.2500	[0.1995, 0.2993]	1.0000
8	2	64	0.2317	[0.1961, 0.2661]	1.0000
8	8	16	0.0826	[0.0218, 0.1445]	0.9955
8	8	64	0.0642	[0.0470, 0.0837]	1.0000
8	16	64	-0.0183	[-0.0872, 0.0505]	0.2840
32	2	8	0.0138	[0.0011, 0.0275]	0.9785
32	2	16	0.1216	[0.0940, 0.1480]	1.0000
32	2	64	0.3911	[0.3521, 0.4289]	1.0000
32	8	16	0.1078	[0.0814, 0.1330]	1.0000
32	8	64	0.3773	[0.3406, 0.4140]	1.0000
32	16	64	0.2695	[0.2351, 0.3050]	1.0000

pieces of evidence. The bootstrap is paired on shared test records, so the resampling unit is the individual `sst_2` test example, with $n = 872$ paired records in every comparison. That design is appropriate for estimating conditional performance differences between two fixed trained models on a fixed evaluation set. It is not an estimate of training-process uncertainty. The resulting confidence intervals quantify how the observed test-set gap would vary under record-level resampling of the same two completed models; they do not quantify seed sensitivity, slice-to-slice variation, or the uncertainty that would arise under retraining.

Read in that restricted way, Table 4 sharpens two condition-level observations. First, at both shot settings, the lower-rank end of the grid dominates the higher-rank end on the realized slice: r=2 beats r=16 and r=64 in both settings, and r=8 also beats r=16 and r=64 in both settings. Second, the only near-tie occurs at the low-rank end of the `examples_per_class` = 32 setting, where r=2 exceeds r=8 by only 0.0138 accuracy points. Stated differently, the paired bootstrap does not add a new layer of evidence beyond the eight completed models; it simply decomposes the already observed ordering into shared-test-set gaps. The fact that 11 of the 12 contrasts favor the lower-rank condition is therefore best read as a compact summary of the same two shot-level orderings, not as 11 independent replications.

Table 5 reports the seed-decoupled repair audit. Under the original inverse-rank update counts, the qualitative low-rank pattern persists across the three new seeds: rank 2 averages 0.7729 test accuracy, while ranks 8, 16, and 64 average 0.5631, 0.5031, and 0.4969. The within-seed pair summaries show rank 2 beating ranks 8, 16, and 64 in all three seeds. This result rules out the simplest explanation that the original rank-2 advantage was only a consequence of setting `seed=draw_id+rank`. However, the equal-step control changes the interpretation sharply. When all ranks receive 42 updates, rank 64 averages 0.7836 test accuracy and beats ranks 2, 8, and 16 in all three seeds. The repair audit therefore supports a stronger and more diagnostic claim than the original artifact table alone: the original rank reversal is reproducible under the original budget rule, but it is not stable to an equal-update budget control.

Table 5 Seed-decoupled repair audit on the `sst_2` eight-example-per-class slice. Means and standard deviations are over seeds 101, 202, and 303.

Budget regime	Rank	Steps	Test acc.	Macro F1
Inverse-rank	2	682	0.7729 ± 0.0169	0.7644 ± 0.0188
Inverse-rank	8	42	0.5631 ± 0.0458	0.4576 ± 0.1020
Inverse-rank	16	10	0.5031 ± 0.0106	0.3347 ± 0.0047
Inverse-rank	64	1	0.4969 ± 0.0106	0.3319 ± 0.0047
Equal-step	2	42	0.5852 ± 0.0682	0.5094 ± 0.1529
Equal-step	8	42	0.5631 ± 0.0458	0.4576 ± 0.1020
Equal-step	16	42	0.5814 ± 0.0350	0.4908 ± 0.0674
Equal-step	64	42	0.7836 ± 0.0766	0.7736 ± 0.0907

Table 6 Cross-slice repair audit using seed 404. The inverse schedule gives rank 64 one update; the equal-step control gives every rank 42 updates. The table reports the direct rank-64 gain from changing only that budget regime.

Condition	Inverse r2 acc.	Inverse r64 acc.	Equal r64 acc.	r64 acc. gain	r64 train loss: 1 step → 4
SST-2 epc=8 draw 1	0.6571	0.5092	0.6514	0.1422	1.3641 →
SST-2 epc=8 draw 2	0.7225	0.5092	0.7362	0.2271	1.3946 →
AG News epc=8 draw 0	0.8661	0.2500	0.8661	0.6161	5.6196 →

Table 6 adds the cross-slice repair diagnostic. These runs are not a replacement for a full multi-dataset benchmark because each condition uses one shared seed. They are nevertheless useful as a mechanism check because they ask the same simple question in three new conditions: what happens to rank 64 when it receives 42 updates instead of the one update assigned by the original eight-example inverse-rank schedule? The answer is consistent across all three conditions. Rank 64 improves from 0.5092 to 0.6514 on SST-2 draw 1, from 0.5092 to 0.7362 on SST-2 draw 2, and from chance-level 0.2500 to 0.8661 on AG News draw 0. The associated training losses also fall sharply, giving direct optimization evidence that the one-step rank-64 conditions are undertrained rather than merely low-capacity.

Figure 3 is most informative when read alongside Tables 2, 4, 5, 6, and 3. Its role is to make the original non-monotone ordering visually legible, not to introduce a separate inferential layer. That visual pattern is compatible with prior work showing that few-shot transfer can be highly sensitive to procedural details (Mosbach et al. 2021; Zhang et al. 2021) and that low-data gains can arise from tightly constrained adaptation interfaces rather than from larger trainable subspaces alone (Schick and Schütze 2021; Gao et al. 2021; Liu et al. 2022a). In the present section, however, those connections remain contextual. The direct empirical record supports an artifact audit of this matched-compute rule, strengthened by targeted reruns but still far short of a broad rank benchmark.

4.3 Mechanism Hypotheses and Limits

The completed results and repair audits motivate one stronger mechanism interpretation. The original inverse-rank schedule can make lower rank appear better because

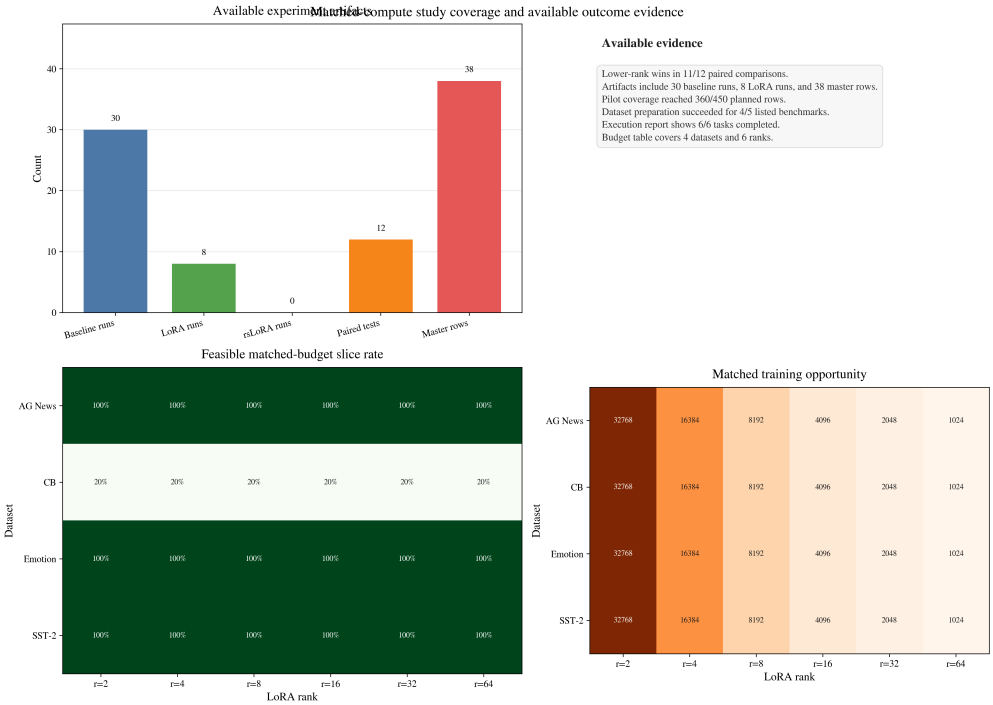


Fig. 3 Visualization of the available artifact table. The decisive LoRA comparisons come only from the completed `sst_2` conditions, while the broader inventory includes contextual head-only baselines for `ag_news` and `sst_2`. The figure should therefore be read as an audit display of what was completed, not as evidence of broad task coverage.

it gives high-rank adapters too little optimization opportunity. The equal-step reruns do not prove that rank 64 is generally best, but they do show that the original low-rank ordering is budget-rule dependent: once rank 64 receives 42 updates instead of one, it becomes the strongest condition in the three-seed original-slice repair audit and in each of the three cross-slice diagnostics. This finding is still compatible with implicit-regularization accounts in which smaller adapters can help under severe label scarcity, and with optimization-efficiency accounts in which the relevant correction may lie in a small effective subspace (Aghajanyan et al. 2021). The new evidence, however, makes the undertraining explanation more concrete than before: the observed reversal cannot be interpreted without the step schedule that produced it.

These interpretations should remain explicitly tentative. The repair audits add repeated seeds, two additional SST-2 slice draws, and one AG News diagnostic draw, but they still contain no second backbone, no rsLoRA or AdaLoRA implementation, no broad dataset sweep, and no direct representation-level mechanism measurements. The paired bootstrap described above adds precision only over resampled test records for fixed realized models; the reruns add training-process information, but only for small few-shot slices and two budget regimes. No regression fit is used in the main evidentiary chain here, since the relevant question is the budget-rule contrast rather than a parametric trend fit. Likewise, because no completed rsLoRA or AdaLoRA artifacts

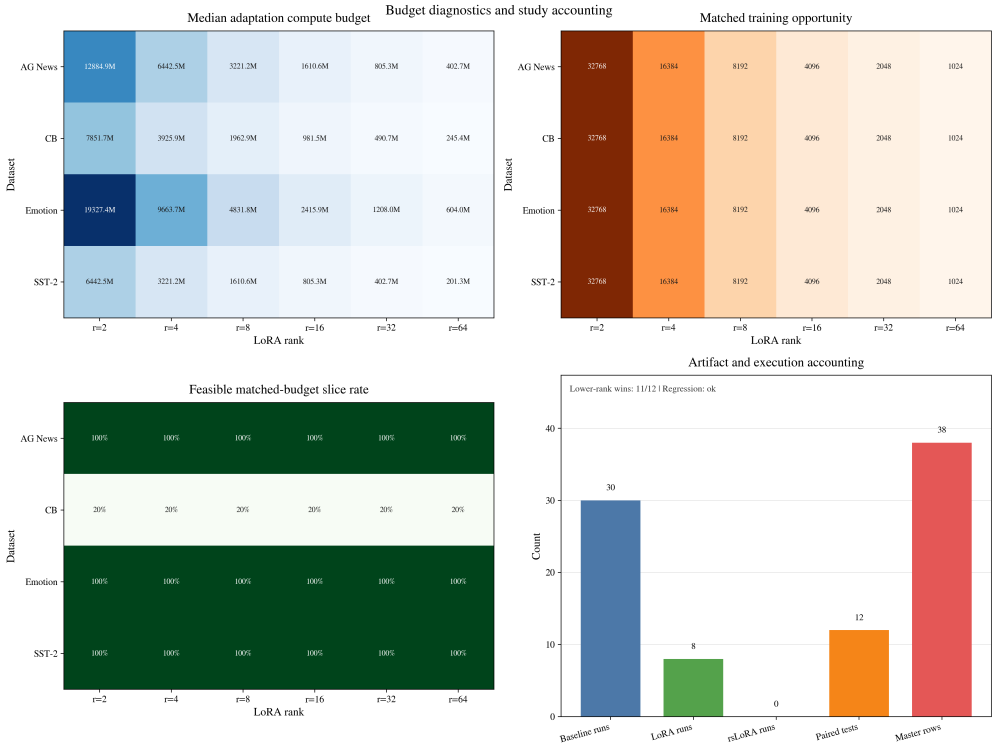


Fig. 4 Interpretive schematic for the completed subset. The figure summarizes hypotheses suggested by the observed ordering under matched compute, but it should not be read as direct causal evidence or as a substitute for independent training replications.

are available in the checkout, the present evidence cannot distinguish between a limitation of standard fixed-rank LoRA specifically and a broader interaction between rank and constrained optimization (Zhang et al. 2023; Kalajdzievski 2023). The most defensible reading is therefore narrow: lower rank often coincides with better accuracy under the implemented inverse-rank schedule, but the targeted reruns show that this ordering is not stable to an equal-step control.

Figure 4 should be read in that narrow sense. It organizes why rank reversal is plausible in low-data adaptation, while the repair audits clarify that the most immediate driver in this checkout is the inverse-rank step schedule. The contribution of this section is therefore deliberately bounded: it documents a concrete reversal pattern in a small completed artifact subset, makes the operational compute-normalization rule auditable at the implementation level, and shows through equal-step controls that the resulting rank ordering should not be generalized beyond the recorded budget regime.

5 Discussion

The most defensible practical implication of the current evidence is narrow. In the completed `sst_2` pilot under the repository's matched-compute protocol, increasing

LoRA rank did not yield a monotonic improvement pattern once the implemented inverse-rank budget proxy was enforced. The repair audits strengthen that diagnostic claim while also narrowing it: under the original inverse-rank step counts, rank 2 remains strongest across three new seeds on the original slice, averaging 0.7729 test accuracy, but under an equal-42-step control rank 64 becomes strongest, averaging 0.7836. Two further SST-2 draws and one AG News draw show the same qualitative budget dependence for rank 64. The central point is therefore methodological rather than generic. This specific proxy-budget rule can generate non-monotonic observed rank orderings, and the ordering is not stable when the budget rule is changed. That evidence is sufficient to show that the original protocol can make higher rank look artificially weak; it is not sufficient to support a broad recommendation about LoRA rank choice across few-shot text classification.

The reason for that restraint is that the inferential base is still small and operationally uneven. The original LoRA evidence that enters the paired artifact comparison consists only of eight `sst_2` artifacts: four ranks at `examples_per_class`=8 and four ranks at `examples_per_class`=32, all from a single completed draw. The repair evidence adds 58 fresh LoRA reruns, but these are targeted diagnostics rather than a balanced benchmark sweep: the original-slice repair has three seeds, the cross-slice repair has one shared seed per new condition, and the budget-response curve studies only ranks 2 and 64 on one slice. The operational record is additionally incomplete: one sweep stage produced zero fresh successful runs and eight error runs, task status files elsewhere are only partially complete, and no rsLoRA artifacts were available in the checkout, so the study cannot determine whether rank-aware scaling rules such as those of Kalajdzievski (2023) would preserve, weaken, or remove the same pattern. The standard-LoRA configuration itself leaves open additional rank-linked confounds: rank 2 receives a larger effective α/r scaling factor than the other ranks, the jointly trained classifier head contributes a large rank-invariant cost ignored by the inverse-rank proxy, and the available artifacts do not include measured FLOPs or runtime. The dataset construction process introduces a further threat to validity. At least one planned few-shot configuration was infeasible before training because the requested class-balanced support set could not be realized, indicating that the nominal grid over datasets and examples per class was not uniformly attainable. That limitation matters because it raises the possibility that the completed comparisons are conditioned on a favorable subset of planned tasks and slice draws rather than on the full intended design.

The bootstrap analysis also requires explicit restraint in interpretation. The paired comparisons resample shared test records from fixed trained model pairs, which is appropriate for estimating uncertainty in conditional performance differences on a fixed evaluation slice, but the resulting confidence intervals quantify only record-level resampling uncertainty. They do not quantify uncertainty from retraining, reseeding, rerunning the slice construction, or changing the budget schedule. Put differently, the intervals are conditional on the eight completed fitted models already in hand. This is why the 12 pairwise contrasts should not be read as 12 quasi-independent confirmations of the same effect: they are mechanically dependent because they reuse the same small set of trained artifacts and the same evaluation records. The repair

audits directly address part of this concern by rerunning shared seeds and additional slices, but they still do not create a large factorial experiment. Because few-shot adaptation is known to be highly sensitive to initialization and sampling variance (Mosbach et al. 2021; Schick and Schütze 2021; Gao et al. 2021; Houlsby et al. 2019), the safer reading remains condition-level and descriptive. The available artifacts show that the recorded protocol can reverse the expected rank ordering; the reruns show that the same ordering does not survive equalized update count.

That last qualification is especially important because the repair audits now point directly to severe undertraining of higher-rank adapters under the chosen step schedule. In the completed `sst_2` artifacts, the inverse-rank budgeting rule yields extremely unequal update counts, including 682 versus 1 steps at `examples_per_class`=8 and 2730 versus 2 steps at `examples_per_class`=32 for ranks 2 and 64, respectively. Across the three original-slice seeds and the three cross-slice diagnostics, moving rank 64 from one update to 42 updates improves test accuracy in every case, with gains ranging from 0.1422 to 0.6161. The budget-response curve makes the point more precise: rank 64 rises quickly from one to 42 updates and then mostly saturates in that seed, whereas rank 2 improves most when trained for 170 or 682 updates. That pattern does not prove that equal update count is the correct fairness target; rank 64 still has more adapter parameters, while rank 2 receives a larger LoRA scaling factor and a much larger share of rank-invariant classifier-head training per adapter parameter. It does, however, show that the original inverse-rank result is not an intrinsic rank law. The contextual head-only baselines reinforce this interpretation rather than resolving it: on `sst_2`, five-draw head-only tuning averages only 0.540 accuracy at 8 examples per class and 0.575 at 32 examples per class, so the successful LoRA runs look meaningfully better than a weak head-only context, but the comparison is not action-matched to the LoRA rank grid. The evidentiary claim should therefore remain conditional: under this particular proxy-budget rule, the expected monotonic ordering with rank was empirically fragile.

Within those limits, the results are best viewed as hypothesis-generating rather than mechanism-establishing. One possibility is that, in low-data settings with sharply budgeted optimization, smaller adapters benefit from stronger implicit regularization or from a search space that is easier to optimize stably (Mosbach et al. 2021; Zhang et al. 2021; Liu et al. 2022a). The repair audits make another possibility more likely for this checkout: the apparent advantage of lower rank is largely a consequence of how the matched-compute proxy allocates training opportunity across ranks. Both readings are compatible with a broader PEFT literature in which constrained update mechanisms can remain competitive under limited supervision (Houlsby et al. 2019; Ben Zaken et al. 2022; Li and Liang 2021; Lester et al. 2021; Liu et al. 2022b), but the present study does not directly test representation-level mechanisms or compare alternative rank-scaling LoRA variants. What it shows is narrower and still useful: this audited matched-compute rule does not justify assuming that higher LoRA rank will help or hurt by default without a declared and stress-tested budget definition.

6 Conclusion

This paper reports a deliberately narrow audit of matched-compute LoRA artifacts in few-shot text classification. Its contribution is not a broad empirical claim about LoRA rank, but a benchmark-conditional observation about one implemented compute-normalization rule and its failure modes. The original evidential base is limited to eight completed LoRA runs from `sst_2`; other project artifacts provide context about execution, baseline difficulty, and design constraints rather than additional direct evidence on rank. The revised audit adds 58 local reruns: 24 on the original eight-example-per-class SST-2 slice to decouple seed from rank, 24 cross-slice diagnostics on two additional SST-2 draws and one AG News draw, and a 10-run budget-response curve for ranks 2 and 64.

Within those completed `sst_2` artifacts, the paired bootstrap summaries should be read only as conditional comparisons of fixed trained model pairs evaluated on shared test records, not as estimates of training-run variability. Under that restricted interpretation, the original descriptive result is that the matched-compute rule examined here can produce apparent rank reversals rather than a monotone advantage for larger adapters. The repair audits change the final interpretation: under the original inverse-rank schedule, rank 2 still averages 0.7729 accuracy across three rank-decoupled seeds on the original slice, but under the equal-42-step control rank 64 averages 0.7836 and becomes the strongest local condition. The cross-slice diagnostics show the same direct budget effect for rank 64, with accuracy gains from one update to 42 updates of 0.1422, 0.2271, and 0.6161. The budget-response curve adds that the effect is not simply monotone preference for rank 64: in one seed, rank 64 saturates after the early recovery, while rank 2 needs substantially longer training to reach its best result. These comparisons show that lower rank can outperform higher rank under one budget rule; they also show that the same ordering is not robust to an equal-update or multi-budget stress test.

The most defensible implication is therefore methodological. Because the observed ordering depends on artifacts produced under a specific compute-normalization protocol, future work should validate the normalization target itself, alongside replication structure and comparison baselines, before drawing broader conclusions about LoRA rank trends. The immediate lesson is especially about undertraining risk: an inverse-rank step schedule that leaves high-rank adapters with one or two updates can make rank comparisons appear decisive while mostly testing optimization opportunity. The audit also shows what this repository does not settle: the comparison does not hold effective LoRA scaling constant for rank 2, does not freeze or separately budget the classifier head, and does not measure actual FLOPs or wall-clock time. At most, the present audit offers a practical caution: in small pilot studies, rank should be interpreted only together with a declared budget rule, and any apparent advantage or disadvantage should be stress-tested against at least one alternative budget definition, multiple seeds, and at least a small slice or dataset check (Wang et al. 2019; Socher et al. 2013; Zhang et al. 2015; Kalajdzievski 2023; Zhang et al. 2023).

Data availability

The derived data and analysis code supporting the findings of this study are available in Zenodo at https://doi.org/10.5281/zenodo.21617113. The archived materials include the consolidated experiment summaries, execution metadata, targeted repair-rerun plans, mixed-effects results, and analysis scripts used in the artifact audit.

Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 62501380), awarded to Zhipeng Chen.

References

Aghajanyan A, Gupta S, Zettlemoyer L (2021) Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp 7319–7328

Ben Zaken E, Goldberg Y, Ravfogel S (2022) Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp 1–9

Devlin J, Chang MW, Lee K, et al (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186

Gao T, Fisch A, Chen D (2021) Making pre-trained language models better few-shot learners. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp 3816–3830

Houlsby N, Giurgiu A, Jastrzebski S, et al (2019) Parameter-efficient transfer learning for NLP. In: Proceedings of the 36th International Conference on Machine Learning, pp 2790–2799

Hu EJ, Shen Y, Wallis P, et al (2021) Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:210609685

Kalajdzievski D (2023) A rank stabilization scaling factor for fine-tuning with LoRA. arXiv preprint arXiv:231203732

Lester B, Al-Rfou R, Constant N (2021) The power of scale for parameter-efficient prompt tuning. In: Proceedings of the 2021 Conference on Empirical Methods in

Natural Language Processing, pp 3045–3059

Li XL, Liang P (2021) Prefix-tuning: Optimizing continuous prompts for generation. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp 4582–4597

Liu H, Tam D, Muqeeth M, et al (2022a) Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. arXiv preprint arXiv:220505638

Liu X, Ji K, Fu Y, et al (2022b) P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp 61–68

Liu Y, Ott M, Goyal N, et al (2019) RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:190711692

Mosbach M, Andriushchenko M, Klakow D (2021) On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. In: International Conference on Learning Representations

Schick T, Schütze H (2021) Exploiting cloze-questions for few-shot text classification and natural language inference. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp 255–269

Socher R, Perelygin A, Wu J, et al (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp 1631–1642

Wang A, Singh A, Michael J, et al (2019) GLUE: A multi-task benchmark and analysis platform for natural language understanding. In: International Conference on Learning Representations

Zhang Q, Chen M, Bukharin A, et al (2023) Adaptive budget allocation for parameter-efficient fine-tuning. In: International Conference on Learning Representations

Zhang T, Wu F, Katiyar A, et al (2021) Revisiting few-sample BERT fine-tuning. In: International Conference on Learning Representations

Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. Advances in Neural Information Processing Systems 28:649–657