LESSER: POST-TRAINING DATA SELECTION WITH OUTPUT-LAYER GRADIENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

The choice of post-training data for large language models substantially affects downstream performance. Gradient-based data selection is a popular approach that ranks training data by how well their gradients align with those of a small validation set. However, ranking with full-parameter gradients requires an expensive backward pass on every sample, making computation intractable for large candidate pools. This raises a natural question: can we approximate full-gradient features at a fraction of the cost? Conveniently, we find that output-layer gradients suffice for effective data selection, yet require only the cheaper forward pass. We implement this as LESSER, a drop-in wrapper for selection methods that reduces the feature-extraction FLOP cost by $9.7\times$ for SFT and $3.0\times$ for RL benchmarks, while tracking full-gradient performance on downstream tasks. Empirically, we find that even when output-layer and full gradients rank individual samples differently, they select batches with aligned gradients.

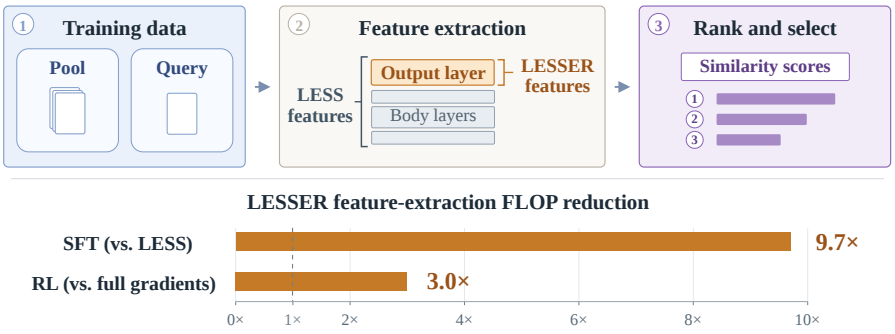

Figure 1: **Efficient data selection with output-layer gradients.** Given a candidate training pool and a small query set representative of downstream tasks, LESSER extracts output-layer gradient features to rank and select training examples. These features approximate the downstream behavior of full-gradient methods like LESS while substantially reducing feature-extraction FLOPs.

1 INTRODUCTION

The choice of post-training data substantially shapes the performance of large language models (LLMs) (Guo et al., 2025; Ouyang et al., 2022; Wang et al., 2023). In particular, training on small, high-quality subsets tailored to a target task can match or even outperform training on a larger, general pool (Muennighoff et al., 2025; Xia et al., 2024). However, identifying which subset to use is non-trivial, as the utility of each sample depends on the model and target task, and selection can become expensive when the candidate pool grows. These challenges have motivated the development of data selection methods for identifying targeted, high-quality subsets (Killamsetty et al., 2021b).

A common approach to targeted data selection maps candidate and query examples into a shared feature space and selects candidates by their similarity to the queries (Zhang et al., 2025). When choosing features, gradients are a natural choice because they describe how each training sample

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

would update the model. This principle underlies modern LLM data selection methods such as LESS (Xia et al., 2024) and GIST (Min et al., 2026) for SFT, GradAlign (Yang et al., 2026) for RL, and GRACE (Panigrahi et al., 2025) for distillation teacher selection.

However, full-parameter gradients are costly to compute at scale: each candidate requires a backward pass through the LLM. For large candidate pools, extraction can cost more than training on the selected data itself (Yin & Rush, 2024). This has motivated more compute-efficient features, such as RDS+ (Ivison et al., 2025), which uses hidden-state features, and proxy-model methods such as Yang et al. (2024b), which use smaller models to guide selection. Despite their cost, however, gradient-based features often yield superior subsets (Nayak et al., 2026). We thus aim to approximate gradients at a fraction of the cost.

In particular, output-layer gradients are attractive because they can be computed directly from last-layer hidden states and output probabilities available during a forward pass, without backpropagating through the model. These features have also proved useful in other settings, including active learning for vision models (Ash et al., 2019), data valuation (Deng et al., 2025), and data attribution (Ran et al., 2026) methods. We find that they are also effective for post-training data selection.

1.1 Our contributions

We propose **LESSER** (**L**ast-layer **E**rror **S**ignals for **S**calable **E**xample **R**anking), a drop-in wrapper for gradient-based data selection methods that only swaps in output-layer gradients, leaving the selection rule unchanged. We implement this for LESS in SFT data selection, for GradAlign in RL problem selection, and for GRACE in distillation teacher selection.

LESSER is cost-efficient. Extracting output-layer gradients is cost-efficient, requiring $9.7\times$ fewer FLOPs on SFT and $3.0\times$ fewer FLOPs on RL. This effect is more pronounced in our wall-clock experiments (Section 4), where LESSER reduces feature-extraction time by $17.8\times$ in SFT, the added scoring time per selection round by $3.5\times$ in RL, and feature-extraction time by 3.9–$5.4\times$ in distillation teacher selection.

LESSER tracks full-gradient performance. Across a variety of setups, output-layer gradients find subsets that reach downstream performance similar to that of subsets found with full gradients.

- **SFT.** Across five SFT tasks and four LLMs, the absolute downstream score gap between LESSER and LESS averages only 1.3 points, growing to 2.2 against RDS+, and to 2.6 against a random selection baseline. Relative to LESS-selected batches, the centered gradient cosine similarity is 0.29 for LESSER, compared with 0.03 for RDS+ and approximately zero for Random. At the base model, the batch gradients of LESSER selections thus align with those of LESS more than other baselines do.
- **RL**. In RL, the problems LESSER and GradAlign select in the first round have a Jaccard similarity of 0.53, compared to the random baseline's 0.075. After training, LESSER and GradAlign reach the same final test accuracy.
- **Distillation.** Across three different distillation settings, LESSER consistently selects the same top teacher from a pool of 10 to 14 teachers as the unmodified GRACE. This suggests that GRACE's teacher choice is robust to replacing full gradients with output-layer gradients.

1.2 Related work

Targeted data selection. Data selection methods differ in the notion of usefulness they aim to capture. Quality filtering and diversity-based curation seek examples that are broadly useful for instruction tuning (Chen et al., 2023; Liu et al., 2023; Zhou et al., 2023). Other methods select data relative to a reference distribution or dataset: DSIR (Xie et al., 2023) matches a target text distribution for pretraining, while CRAIG (Mirzasoleiman et al., 2020) constructs weighted subsets that approximate the full training-set gradient. Selection can also be tied directly to a training objective: GLISTER (Killamsetty et al., 2021b) optimizes held-out validation performance, while GRAD-MATCH (Killamsetty et al., 2021a) matches the gradient of the training or validation set. For targeted post-training, LESS (Xia et al., 2024) uses gradient alignment and RDS+ (Ivison et al.,

2025) uses hidden-state similarity. GIST (Min et al., 2026) projects gradients into a subspace defined by validation gradients, while GradAlign (Yang et al., 2026) uses policy-gradient alignment for online RL selection. Systematic evaluations show that selection gains vary across datasets and budgets (Diddee & Ippolito, 2025; Nayak et al., 2026), and that the cost of selection can outweigh its training gains (Yin & Rush, 2024). We instead focus on the efficiency of data selection pipelines, and show that output-layer gradients are a cost-efficient substitute for full-gradient features.

Efficient gradient representations. Gradient-based attribution connects training examples to their effects on model predictions. Koh & Liang (2017) use influence functions to estimate the effect of upweighting a training example, while TracIn accumulates gradient inner products across training checkpoints (Pruthi et al., 2020). To make attribution scalable, TRAK (Park et al., 2023) combines model linearization with random projection, while LoGra (Choe et al., 2025) exploits the outer-product structure of layer gradients to reduce projection costs. These approaches compress gradient features but still require backpropagation. Restricting the gradient to the last layer is another approach, used in GLISTER (Killamsetty et al., 2021b) and GRAD-MATCH (Killamsetty et al., 2021a) for subset selection and in BADGE (Ash et al., 2019) for active learning.

Closely related, For-Value (Deng et al., 2025) derives a forward-only valuation score from output-gradient inner products, while RISE (Ran et al., 2026) combines sketches of this signal with an additional feature that maps prediction residuals into hidden-state space. Both methods target valuation or attribution, which score individual examples, and treat the output-layer score as the value itself or as an approximation of full-gradient influence, respectively. We instead keep existing SFT, RL, and distillation teacher selectors (Panigrahi et al., 2025) unchanged, replace only their gradient features, and evaluate the selections downstream. In SFT especially, output-layer and full gradients rank individual examples differently, yet their selections perform similarly downstream. This differs from the per-example view taken by prior work, under which weaker agreement on individual examples should yield weaker selections; Section 5 explains it through the aligned gradients of the selected batches.

2 Background

2.1 Feature-based data selection

We consider targeted data selection from a training pool $\mathcal{D} = \{p_1, \ldots, p_N\}$ using a small query set $\mathcal{Q} = \{q_1, \ldots, q_M\}$ drawn from the target test distribution. We refer to examples in $\mathcal{Q}$ as *queries*, since they specify the target task and guide which training examples are selected. Given a budget k, the goal is to choose a subset $S \subseteq \mathcal{D}$ of size k such that training on S improves performance on the target distribution, with $\mathcal{Q}$ serving as the available proxy for this target during selection.

Which subset S should we choose? Many targeted data selection methods answer this question using a feature-based pipeline. Each candidate $p \in \mathcal{D}$ and query $q \in \mathcal{Q}$ is mapped to a feature vector ϕ_p and ϕ_q in a shared representation space. A similarity function $\mathrm{sim}(\phi_p, \phi_q)$ then measures how relevant candidate p is to query q. This separates two design choices: the feature representation and the selection rule, which together are used to extract a size-k training subset from the pool.

The same basic pipeline applies across SFT and RL (Figure 2): (1) compute features for the candidate pool and query set; (2) compare candidate and query features to obtain relevance scores; (3) apply a selection rule to these scores to choose a size-k subset S; and (4) train the model on the selected subset. The precise selection rule can differ across settings. In this work, we hold the selection procedure fixed within each setting and study the choice of feature representation.

2.2 Gradient features

The feature-based view leaves open the central question of what information the representation ϕ_p should preserve. A natural goal is to represent each candidate according to how training on it would affect performance on the target task. We express this effect through a loss function, whose precise form depends on the post-training setting.

In SFT, each example $p = (x, y)$ consists of a prompt x and target response y. Let $\pi_\theta(y \mid x)$ denote the model's probability of the response under parameters θ. The supervised response loss is

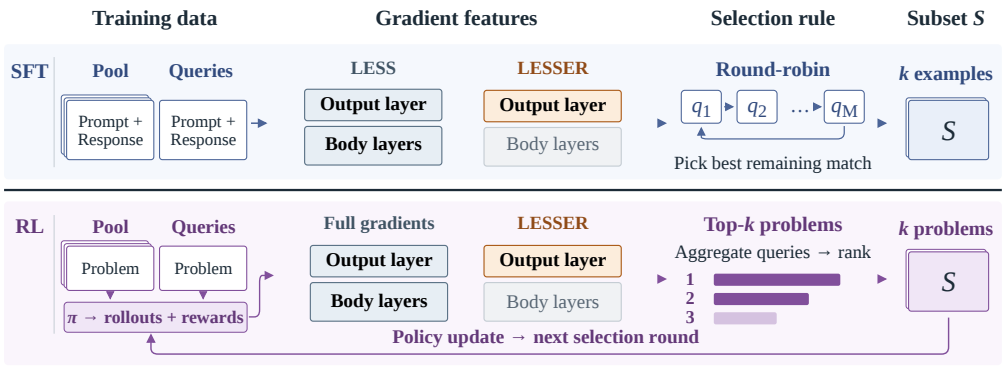

Figure 2: **Data selection with output-layer gradient features in SFT and RL.** Pool and query representations determine similarity scores, while the selection rule produces the training subset. LESSER replaces the gradient representation within this pipeline.

$L_p(\theta) = -\log \pi_\theta(y \mid x)$. The same loss is defined for pool and query examples. In RL, an example p is instead a problem whose sampled responses and rewards determine a policy-optimization loss $L_p(\theta)$. Following the online selection setup of GradAlign (Yang et al., 2026), this loss is constructed from rollouts under the current policy and changes as the policy is updated. In either setting, each candidate and query is therefore associated with a scalar training objective, giving us a common way to describe the effect of training on that example.

Gradients provide a natural feature for these objectives: they describe the local parameter update induced by an example, and their alignment has a direct first-order interpretation in terms of how training on one example affects another. For a candidate $p \in \mathcal{D}$ and query $q \in \mathcal{Q}$, define

$$G_p = \nabla_\theta L_p(\theta), \qquad G_q = \nabla_\theta L_q(\theta).$$

Consider taking one SGD step on candidate p with learning rate $\eta > 0$. Then $\theta' = \theta - \eta G_p$, and a first-order Taylor expansion of the query loss gives

$$L_q(\theta') - L_q(\theta) \approx \nabla_\theta L_q(\theta)^\top (\theta' - \theta) = -\eta G_q^\top G_p.$$

Thus, when $G_q^\top G_p > 0$, training on p is predicted to reduce the loss on query q to first order.

Standard gradient inner products then give a natural notion of similarity. Because gradient magnitude can vary with factors such as sequence length, methods such as LESS (Xia et al., 2024) further perform normalization or use cosine similarity instead. Moreover, since $G_p \in \mathbb{R}^{|\theta|}$ has one coordinate per model parameter, gradient features are often projected to a lower-dimensional space for efficient storage and similarity computation (Park et al., 2023; Xia et al., 2024). However, the main drawback of gradient features is computational cost: obtaining a full-model gradient requires backpropagating through the entire model for every scored example.

3 LESSER: FEATURE EXTRACTION IN ONE FORWARD PASS

Our main idea is to use output-layer gradients in place of full-model gradients, building on their use in settings such as active learning (Ash et al., 2019) and data attribution (Deng et al., 2025; Ran et al., 2026). Figure 2 illustrates this substitution: LESSER replaces the full-model gradients used by methods such as LESS with output-layer gradients while leaving the selection rule unchanged. Crucially, in our SFT and RL settings, these output-layer gradients can be computed from forward-pass quantities alone, avoiding backpropagation through the model.

A simple but key observation makes this possible: the standard SFT and RL scoring objectives we consider share a weighted token-level form:

$$L_p(\theta) = -\sum_t \alpha_t \log \pi_\theta(y_t \mid x, y_{<t}). \tag{1}$$

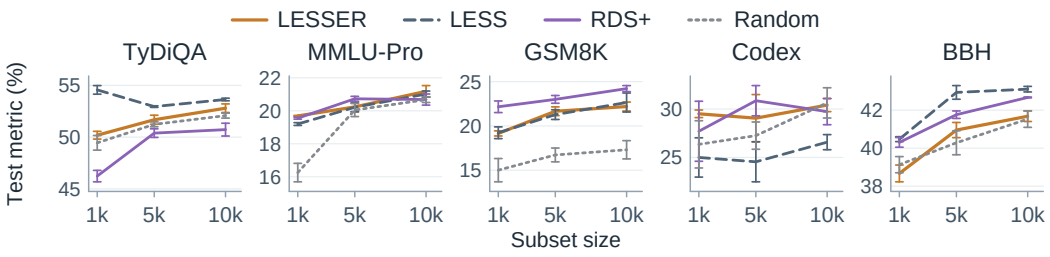

Figure 3: **LESSER closely tracks LESS on downstream performance.** Panels show Llama-2-7B; values are means over three training seeds, and error bars show one sample standard deviation. Across all 60 model–task–budget cells, the absolute score gap of LESSER averages 1.3 points against LESS, 2.2 against RDS+, and 2.6 against Random.

We treat the per-token weights α_t as fixed when differentiating. In SFT, they select the supervised positions; in RL, they additionally incorporate response advantages and token weighting from the GradAlign scoring objective (Yang et al., 2026). For this common form, the output-layer gradient can be written explicitly using forward-pass quantities.

Proposition 1. *Let W be the final vocabulary projection, h_t the hidden state predicting y_t, and e_{y_t} the one-hot vector for y_t. Let W_{out} be a copy of W used only as the readout, so that h_t does not depend on it. Then, the output-layer gradient of Equation* (1) *satisfies*

$$g_p := \nabla_{W_{\text{out}}} L_p(\theta) = \sum_t \alpha_t \big(\text{softmax}(W h_t) - e_{y_t}\big) h_t^\top .$$

Proposition 1 follows directly from the softmax cross-entropy gradient; Appendix B.1 gives a full proof. Since both h_t and $\text{softmax}(Wh_t)$ are available from the forward pass, g_p can be computed without backpropagating through the model. We use $\text{vec}(g_p)$ as the feature for each candidate and query in place of the full-model gradient $G_p = \nabla_\theta L_p(\theta)$.

When needed for efficient storage and similarity computation, we project $\text{vec}(g_p)$ to a lower dimension following standard practice for gradient features (Johnson et al., 1984; Park et al., 2023); Appendix B.2 specifies our two-sided projection.

4 LESSER IS EFFECTIVE AT TARGETED DATA SELECTION

We evaluate LESSER across targeted SFT, RL, and distillation teacher selection. We compare its effectiveness with the corresponding full-gradient methods and measure the resulting computational savings.

SFT. We take 197,196 Tulu V2 (Ivison et al., 2023) instruction–response samples as our candidate pool, and take TyDiQA, MMLU-Pro, GSM8K, Codex, and BBH as our query and test sets, following prior work (Ivison et al., 2025; Nayak et al., 2026; Xia et al., 2024). For data-selection methods, we use **LESSER**, **LESS**, **Random** (wherein samples are randomly chosen), and **RDS+** (hidden-state based), where selection is done with a standard round-robin rule until k samples are chosen. For selection sizes $k \in \{1{,}000, 5{,}000, 10{,}000\}$, we fully fine-tune Llama-2-7B, Llama-3.2-3B, Qwen3-4B-Base, and OLMo3-7B on each selected subset with three seeds. We show Llama-2-7B in the main text and report the full model–task–budget grid in Appendix C.1. Baseline results are from Nayak et al. (2026) except for MMLU-Pro, where we rerun all methods under our evaluation setup. Another caveat is that LESSER uses only the base model, while LESS uses Adam-preconditioned LoRA (Hu et al., 2021) gradients at four warmup checkpoints for computational tractability and empirical improvements. The two pipelines therefore differ beyond the feature, but LESSER is much cheaper and more convenient to extract. Appendix A.1 gives the full training protocol and replication checks.

Finding 1: LESSER *is competitive with* LESS *on downstream SFT performance.* On Llama-2-7B, the two methods are nearly identical on GSM8K and MMLU-Pro, while LESS is stronger on

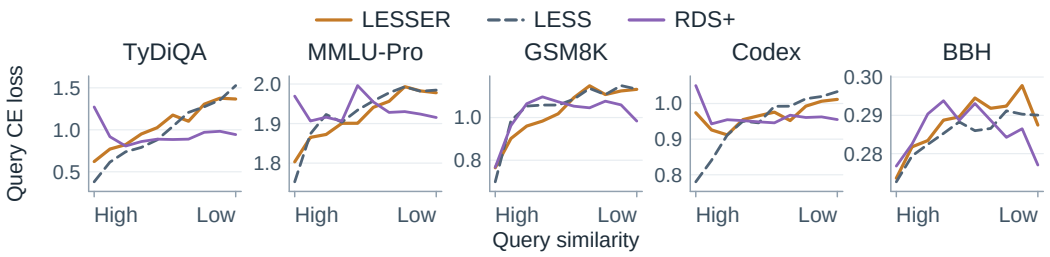

Figure 4: **Higher-similarity training examples lead to lower query loss.** Query loss after SFT on Llama-2-7B, stratified by example score. For both LESS and LESSER, examples with higher query similarity generally yield lower query loss after fine-tuning.

TyDiQA and BBH and LESSER is stronger on Codex (Figure 3). Across the four models, LESSER frequently improves over Random, while RDS+ is also competitive on several benchmarks. We next ask whether LESSER's similarity scores behave consistently across the pool: do more similar examples lead to lower query loss?

A desirable property of data selectors is that they should rank data by their effect on the query loss: high-ranked samples should reduce it the most, while low-ranked samples the least. To test that this holds for LESSER, we use a similarity-bin diagnostic (Nayak et al., 2026) where we partition the candidate pool into ten deciles. We then fine-tune a separate model on the top 500 samples from each bin and measure its loss on the query set. If the similarity score captures target relevance, query loss should increase as we move from higher- to lower-similarity bins, giving a positive Spearman correlation between bin index and query loss.

Finding 2: Higher LESSER *similarity predicts lower query loss in SFT.* On Llama-2-7B, LESSER similarity strongly predicts lower query loss, with a mean Spearman correlation of 0.86, close to LESS and far above RDS+ (Figure 4). This relationship generalizes across models: LESSER is positively correlated with query loss decrease in 19 of 20 model–task pairs, though somewhat less strongly than LESS overall.

RL. We evaluate LESSER in two GradAlign settings: avoiding problems with corrupted rewards and selecting problems during online RL training. In both settings, we replace only the full-gradient representation with output-layer gradients while keeping the selection procedure fixed. We first test whether LESSER can avoid problems whose rewards carry no useful learning signal. Following Yang et al. (2026), we corrupt the rewards of half the candidate problems with coin flips and select a subset against AMC22 queries.

Finding 3: LESSER *closely matches full policy gradients in avoiding problems with corrupted rewards.* LESSER selects 26.6% corrupted problems, closely matching 27.3% for full gradients and well below **Random** (Figure 5). **AccGreedy**, which selects problems whose empirical accuracy is closest to 50%, instead favors corrupted problems because their coin-flip rewards push empirical accuracy toward 50%. Appendix A.2.2 gives the full protocol.

Another setting is in online RL, where in our case data selection happens after every 10 policy gradient updates. We target Countdown using a mixed-domain training pool in which only 11.8% of the problems are from Countdown, with the remainder from WebInstruct (Yue et al., 2024) and DAPO (Yu et al., 2026). Across five selection rounds, we compare LESSER with full-gradient GradAlign, **dynamic sampling** (Yu et al., 2026), which retains problems with nonzero rollout advantage, and **Random**, while training Qwen2.5-Math-1.5B-Instruct with GRPO (Shao et al., 2024).

Finding 4: LESSER *closely matches full policy gradients in targeted online RL training.* LESSER closely tracks full-gradient GradAlign, and both reach the same final held-out accuracy (Figure 6). LESSER reaches Random's final accuracy after only 15 steps, using 70% **fewer GRPO steps**.

Distillation teacher selection. GRACE performs distillation teacher selection by having the student model backpropagate on each candidate teacher's generated responses; it then uses these gradi-

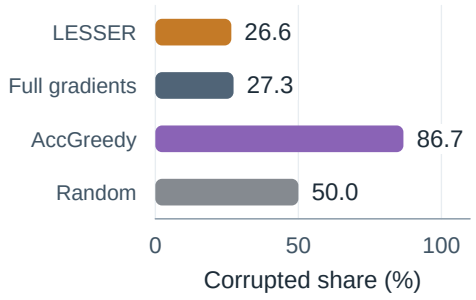

Figure 5: **LESSER closely matches full gradients in detecting problems with corrupted rewards.** Corruption share among the selected problems. **Lower** is better.

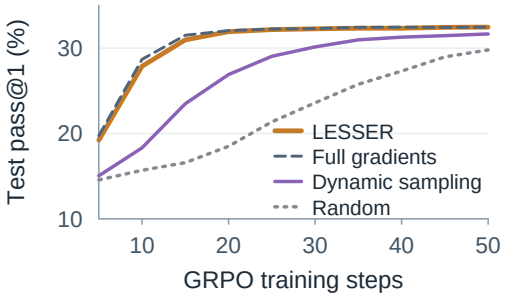

Figure 6: **LESSER closely tracks full gradients in RL.** Held-out Countdown accuracy over GRPO training steps, averaged over two seeds. **Higher** is better.

ents to rank and select teachers. As in SFT and RL, LESSER instead replaces full-gradient features with output-layer gradients, while leaving the rest of the GRACE pipeline unchanged.

Finding 5: LESSER *closely preserves full-gradient GRACE teacher rankings.* Across all settings, the teacher rankings remain closely aligned, with Spearman correlations of 0.94–0.98 (Table 12), and LESSER recovers the same top-ranked teacher as full-gradient GRACE. That is, even though output-layer gradients are used, there is no drop in the quality of teacher selection.

Cost analysis. We give detailed accounting in Appendix A.4, with highlights in Table 1 and below.

SFT. LESSER extracts features with a single forward pass, whereas the LESS setup used in our experiments extracts LoRA-gradient features with full backpropagation at four checkpoints. Counting FLOPs using LESS's configuration, this reduces feature-extraction FLOPs by $9.7\times$ and, extrapolating our A100 measurements to the full candidate pool, wall-clock extraction time by $17.8\times$. Even against a single LESS checkpoint, extraction is $4.4\times$ faster in wall clock. Feature extraction accounts for 97% to 99.7% of LESS's combined extraction and fine-tuning FLOPs across our tested budgets, making it the dominant computational cost.

RL. Full-gradient GradAlign requires a separate forward–backward scoring pass after rollout generation, whereas LESSER scores with projected output-layer features, which need only a forward pass. This reduces scoring FLOPs by $3.0\times$. On the responses of one selection round, the added scoring time falls from 63.3 to 18.1 minutes on one A100, a $3.5\times$ reduction.

Distillation. Full-gradient GRACE backpropagates through the student for every teacher response, whereas LESSER requires only a forward pass. Across the three teacher-selection settings, this reduces feature-extraction time by 3.9 to $5.4\times$ while also lowering peak GPU memory.

5 Why are output-layer gradients effective?

Since LESSER often matches full-gradient methods downstream, we ask why it can do so using just the output-layer gradient. In GRACE the two methods rank teachers almost identically and in RL they largely agree, but in SFT their selections differ. We therefore ask whether LESSER and LESS might still update the model in similar directions. At the base model where selection happens, we find that batches selected by LESSER and LESS produce aligned full-model training gradients, much more so than RDS+ or Random. In addition, both LESSER and LESS achieve most of their query-loss decrease within the first ten updates.

Table 1: **Output-layer gradients are much cheaper to extract.** Cost analysis on one A100 80GB for SFT feature extraction and RL scoring.

SFT Selector	Time (h)	Speedup	RL Selector	Added time (min)	Speedup	Feature size
LESSER	3.16	**17.8**×	LESSER	18.1	**3.5**×	32.8 kB
LESS	56.20	1.0×	Full gradients	63.3	1.0×	6.2 GB

LESSER can select different data from LESS. In SFT, we test whether the similar downstream performance of LESSER and LESS can be explained by choosing similar data. We check for overlap using Jaccard similarity, $|S \cap S'|/|S \cup S'|$, and we compare rankings using Spearman rank correlation.

At $k = 5{,}000$ on Llama-2-7B, the selected sets have mean Jaccard similarity 0.037 (Appendix E). In an audit across the 20 model–task pairs, output-layer and full-gradient rankings have median Spearman correlation 0.52 (Appendix F.1). This level of disagreement is specific to SFT: in RL, the two selections share 69% of their problems in the first round, while in GRACE the statistics used for teacher selection remain highly correlated (Appendix F.4). This leaves the SFT result unexplained, so we turn to how the selected examples affect training and ask whether the different selections induce similar updates.

The selected batches align. We first characterize how a selected batch affects training. For a batch S, let G_S denote its full-model gradient at the base model θ_0. Our goal is to compare the training directions induced by two selections, which requires us to exclude gradient components that are common across the candidate pool. In particular, even independently sampled random batches can have high cosine similarity because their gradients share the population mean. We observe this effect empirically in Appendix F.2. We therefore subtract the expected gradient μ of a random batch and compare the centered gradients $G_S - \mu$ and $G_{S'} - \mu$ using the centered batch-gradient alignment defined in Appendix F.2. This measures whether two selections emphasize similar training directions beyond random sampling.

We evaluate gradient alignment using the actual training batch size of 128 across 20 model–task pairs (Figure 7a). As a reference for the scale of this measurement, we first compare different LESS batches drawn from the same $k = 5{,}000$ selection; their median alignment is 0.53. Against this reference, LESSER batches show substantial alignment with LESS, with a median alignment of 0.29. In contrast, the median alignment with LESS is 0.03 for RDS+ and approximately zero for Random. Thus, under the batch size used in training, LESSER batches align with LESS batches far more than RDS+ or random batches do.

The effect of selection happens early in training. Above, we showed that LESSER and LESS select batches whose full-model gradients are aligned at the base model, despite selecting largely different examples. The remaining question is when selection affects the subsequent training run. We therefore track query loss throughout the full 80-update SFT runs.

The query-loss gains from selection are concentrated early in training. By step 10, LESSER and LESS have achieved a median 84% and 82%, respectively, of their maximum query-loss decrease over the full 80 updates, compared with 35% for Random (Figure 7b). This connects the first-order selection at the base model to the reduction in query loss.

6 Discussion and Conclusion

Discussion. Our results suggest three broader lessons.

First, compared to hidden-state features, output-layer gradients preserve more of the signal used by full-gradient data-selection methods at a similar extraction cost. They are also more directly tied to the optimization objective, making them a natural starting point when extraction cost matters.

Second, agreement is strongest at the level each pipeline acts on. SFT trains on batch-aggregated gradients, RL selects problems after aggregating sampled responses, and GRACE ranks teachers

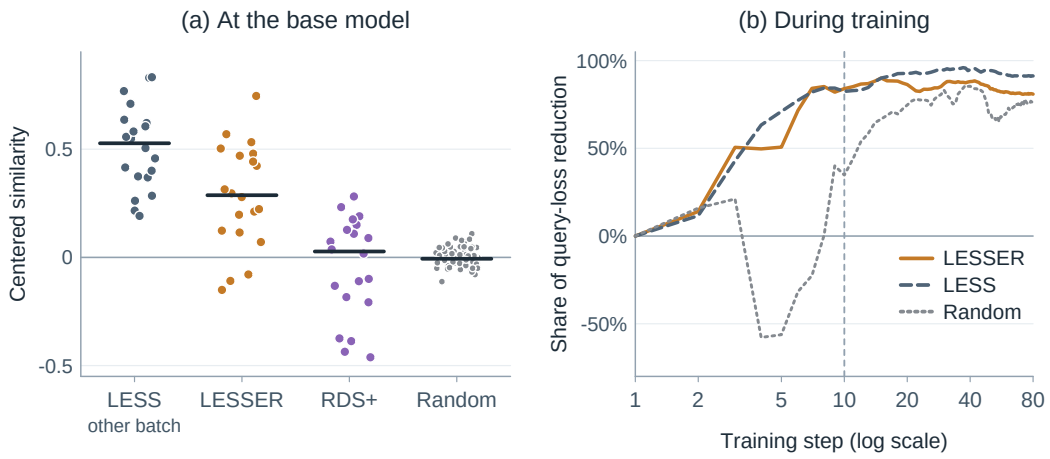

Figure 7: **LESSER-selected batches induce gradients aligned with those from LESS.** (a) Centered batch-gradient alignment between LESS batches and batches selected by each method, at the base model. Dots denote model–task pairs, and bars show medians. (b) Median share across model–task pairs of the query-loss decrease reached by each training step.

using statistics over many responses. This suggests a broader principle: efficient selection features need not recover per-example full gradients, as long as they preserve the aggregate geometry that each pipeline uses.

Third, full-gradient fidelity may therefore be the wrong objective for evaluating a selection proxy. Prior work on training-data attribution (Grosse et al., 2023) shows that recovering full-network influence can require information distributed across layers. This suggests that fine-grained approximation may be important for influence estimation or data attribution, while selection from realistic, already-curated LLM data pools may only require preserving which examples or subsets are useful.

Limitations. Our experiments span multiple models, tasks, and selection settings, but do not cover all post-training regimes or selection objectives. We also focus primarily on near-initialization training effects, leaving a fuller understanding of how output-layer gradient signals evolve over longer training trajectories to future work. Our SFT comparison is between complete pipelines rather than a matched-checkpoint ablation, so LESSER and LESS differ in more than just their features. Nevertheless, LESSER needs neither warmup training nor backpropagation, and it remains competitive with this more expensive pipeline.

Future Work. Future work could develop diagnostics that predict, before running selection, when output-layer gradients will recover the rankings or update directions induced by full gradients. A second direction is to identify which computations inside the model make this agreement possible, for example by tracing how selection-relevant information is transformed across layers. Finally, a stronger theory could characterize when such agreement should emerge under realistic, structured gradient distributions.

Conclusion. We study whether output-layer gradients can replace full-gradient features for data selection in LLM post-training. Across SFT, RL, and distillation teacher selection, they often match full-gradient methods downstream at substantially lower cost. In RL and GRACE, the resulting selection statistics largely agree, while in SFT, different rankings nevertheless induce aligned batch updates. Output-layer gradients therefore offer a simple, efficient alternative to full gradients.

Reproducibility Statement

All models, the candidate pool, and the benchmarks we use are publicly available. Section 4 and Appendix A specify the selection budgets, training recipes, seeds, and evaluation protocols for SFT (Appendix A.1), RL (Appendix A.2), and GRACE teacher ranking (Appendix A.3). Appendix B

gives the proof of Proposition 1 and the construction of the projected feature, and Appendix A.4 gives the FLOP formulas and the timing procedures behind every cost number. Appendices C and F report the complete SFT grid and the protocols of the analyses in Section 5. We will release code for feature extraction and selection, together with the selected subsets.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Sang Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff Schneider, et al. What is your data worth to gpt? llm-scale data valuation with influence functions. *Advances in Neural Information Processing Systems*, 38:145944–145985, 2025.

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. Tydi qa: A benchmark for information-seeking question answering in ty pologically di verse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020.

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

Wenlong Deng, Jiaming Zhang, Qi Zeng, Christos Thrampoulidis, Boying Gong, and Xiaoxiao Li. Efficient forward-only data valuation for pretrained llms and vlms. *arXiv preprint arXiv:2508.10180*, 2025.

Harshita Diddee and Daphne Ippolito. Chasing random: Instruction selection strategies fail to generalize. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1943–1957, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

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

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*, 2023.

Hamish Ivison, Muru Zhang, Faeze Brahman, Pang Wei Koh, and Pradeep Dasigi. Large-scale data selection for instruction tuning. *arXiv preprint arXiv:2503.01807*, 2025.

William B Johnson, Joram Lindenstrauss, et al. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pp. 5464–5474. PMLR, 2021a.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8110–8118, 2021b.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Guanghui Min, Tianhao Huang, Ke Wan, and Chen Chen. Gist: Targeted data selection for instruction tuning via coupled optimization geometry. *arXiv preprint arXiv:2602.18584*, 2026.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pp. 6950–6960. PMLR, 2020.

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. Octopack: Instruction tuning code large language models. In *International Conference on Learning Representations*, volume 2024, pp. 7604–7623, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Nihal V Nayak, Paula Rodriguez-Diaz, Neha Hulkund, Sara Beery, and David Alvarez-Melis. A critical look at targeted instruction selection: Disentangling what matters (and what doesn't). *arXiv preprint arXiv:2602.14696*, 2026.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Abhishek Panigrahi, Bingbin Liu, Sadhika Malladi, Sham Kakade, and Surbhi Goel. In good graces: Principled teacher selection for knowledge distillation. *arXiv preprint arXiv:2511.02833*, 2025.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930, 2020.

Qwen, :, 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, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Yide Ran, Jianwen Xie, Minghui Wang, Wenjin Zheng, Denghui Zhang, Chuan Li, and Zhaozhuo Xu. Sketching the readout of large language models for scalable data attribution and valuation. *arXiv preprint arXiv:2604.16197*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36: 34201–34227, 2023.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024a.

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

Ningyuan Yang, Weihua Du, Weiwei Sun, Sean Welleck, and Yiming Yang. Gradalign: Gradient-aligned data selection for llm reinforcement learning. *arXiv preprint arXiv:2602.21492*, 2026.

Yu Yang, Siddhartha Mishra, Jeffrey Chiang, and Baharan Mirzasoleiman. Smalltolarge (s2l): Scalable data selection for fine-tuning large language models by summarizing training trajectories of small models. *Advances in Neural Information Processing Systems*, 37:83465–83496, 2024b.

Junjie Oscar Yin and Alexander M Rush. Compute-constrained data selection. *arXiv preprint arXiv:2410.16208*, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. Mammoth2: Scaling instructions from the web. *Advances in Neural Information Processing Systems*, 37:90629–90660, 2024.

Bolin Zhang, Jiahao Wang, Qianlong Du, Jiajun Zhang, Zhiying Tu, and Dianhui Chu. A survey on data selection for llm instruction tuning. *Journal of Artificial Intelligence Research*, 83, 2025.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

Table 2: Target tasks. "Val. queries" is the query-set size used to specify each target; metrics and splits follow Nayak et al. (2026), except that MMLU-Pro is evaluated five-shot.

Task	Domain	Val. queries	Metric
TyDiQA (Clark et al., 2020)	multilingual QA	9	F1
MMLU-Pro (Wang et al., 2024)	knowledge / reasoning	70	accuracy
GSM8K (Cobbe et al., 2021)	grade-school math word problems	8	exact match
Codex (Chen et al., 2021)	Python code generation	16	pass@10
BBH (Suzgun et al., 2023)	challenging BIG-Bench reasoning	81	exact match

Appendix guide. Appendix A gives the experimental protocols and cost accounting; Appendix B proves Proposition 1 and gives the projection of the output-layer gradient. The results follow the main experiments: SFT in Appendix C and GRACE teacher selection in Appendix D. Appendix E reports the overlap of the deployed selections, and Appendix F supports Section 5.

A Experimental Setup Details

This appendix specifies SFT on selected examples, RL on selected problems, GRACE teacher ranking, and cost accounting.

A.1 SFT

Baselines and MMLU-Pro. The SFT grid uses the candidate pool, target tasks, training recipe, and evaluation harness of Nayak et al. (2026). On TyDiQA, GSM8K, Codex, and BBH, we add LESSER runs to their released Random, RDS+, and LESS results. Under the zero-shot chat-template evaluation of MMLU-Pro, many fine-tuned models do not produce an answer in the expected format, so accuracy depends mainly on the answer format. We therefore evaluate MMLU-Pro five-shot with the `mmlu_pro` task of the LM Evaluation Harness on all 12,032 test questions, and we retrain all four methods, using the released selections for the three baselines, with three seeds for every model and budget. All MMLU-Pro results in the paper use this evaluation.

Candidate pool. The pool is the subsampled Tulu V2 instruction mixture of 197,196 instruction–response examples (Ivison et al., 2023), in its default order. The same pool is used for every (model, task, method, budget) cell, so methods differ only in which subset they select.

Target tasks. We use the five target tasks of Nayak et al. (2026), listed in Table 2. Each task is specified by a small in-distribution validation/query set, used both to define the target and to rank pool examples for selection, and is scored on a held-out test split.

Models. We evaluate four base models under full fine-tuning: Llama-3.2-3B (Grattafiori et al., 2024), Llama-2-7B (Touvron et al., 2023), Qwen3-4B-Base (Yang et al., 2025), and OLMo3-7B (Olmo et al., 2025).

Training. Selection is followed by full fine-tuning: 2 epochs, learning rate 2×10^{-5} with a linear schedule and 3% warmup, weight decay 0, effective batch size 128 from per-device batch 1 and gradient accumulation 128, bf16, and maximum sequence length 2048.

Evaluation. We run inference with fixed prompts and decoding, and score each task with the metric listed in Table 2. Codex uses HumanEval (Chen et al., 2021) with HumanEvalPack instructions (Muennighoff et al., 2024).

Selection rule and budgets. LESSER, RDS+, and LESS share the round-robin rule. Budgets are $k \in \{1{,}000, 5{,}000, 10{,}000\}$, and every model–task–budget comparison reports three training seeds. Appendix C.1 gives the means and sample standard deviations.

Replication check. We check the training and evaluation pipeline by rerunning released selections in our pipeline. For Llama-3.2-3B at $k = 1{,}000$, we rerun the released LESS selections and Random with three seeds and compare with Table 9; for Qwen3-4B-Base and OLMo3-7B, we rerun the released LESS TyDiQA selection at seed 0 (Table 3). The LESS reruns agree within 0.9 points. Our Random reruns draw new random subsets instead of the released ones and differ by up to 3.0 points. MMLU-Pro is omitted because we rerun it five-shot for all methods.

Table 3: Pipeline replication, reported value / our rerun.

Model	Task	LESS	Random
Llama-3.2-3B	TyDiQA	66.4 / 66.1	53.7 / 50.7
Llama-3.2-3B	GSM8K	34.2 / 34.0	29.2 / 27.6
Llama-3.2-3B	Codex	56.3 / 55.4	59.2 / 59.9
Llama-3.2-3B	BBH	47.6 / 47.2	47.1 / 47.2
Qwen3-4B-Base	TyDiQA	71.6 / 71.4	–
OLMo3-7B	TyDiQA	62.5 / 62.5	–

Feature extraction. LESSER features are extracted from the pretrained base model, unlike LESS. We sum residual–hidden-state outer products over supervised response positions, apply the 64×128 two-sided projection in Appendix B.2, then flatten and normalize the resulting $8{,}192$ entries. Pool and query examples share projection matrices. Prompt tokens are masked from the response loss.

Similarity decile training. We rank the pool by each method's similarity to the queries, split it into deciles with bin 0 closest to the queries, and take the top 500 examples from each bin. For each bin we fine-tune a separate model with the downstream recipe above and one seed, and measure the mean loss on the query set; Appendix C.2 reports the results.

A.2 Targeted reinforcement learning

We follow the setup and released implementation of GradAlign (Yang et al., 2026), and change only the selection feature. GradAlign scores a problem p by the gradient G_p of its advantage-weighted response loss $L_p(\theta) = \sum_j A_j \ell_j(\theta)$, where the K sampled responses y_j have rewards R_j, centered advantages $A_j = R_j - \bar{R}$, and token-weighted response losses ℓ_j. We replace G_p with the output-layer gradient of the same loss,

$$g_p = \sum_{j,t} w_{jt} A_j \, r_{jt} h_{jt}^{\top}, \tag{2}$$

where w_{jt} are the fixed token weights of the released scorer, h_{jt} is the final hidden state that predicts token y_{jt}, and r_{jt} is its prediction residual. Query problems are aggregated with the same weights, and candidates are ranked by cosine similarity with the aggregated query feature. A problem whose sampled rewards are all equal has zero advantages and therefore scores zero under both features.

A.2.1 Data, training, and evaluation

The policy is Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024a). Following the data proportions of GradAlign Scenario 2, we construct a pool of $34{,}000$ problems: $4{,}000$ Countdown problems, $20{,}000$ WebInstruct problems (Yue et al., 2024), and $10{,}000$ DAPO problems (Yu et al., 2026). The Countdown instances are drawn separately from those used in GradAlign. We additionally use 200 Countdown problems as queries and 500 held-out Countdown problems for evaluation; both sets are disjoint from the training pool. Table 4 summarizes the experimental settings.

Training proceeds for multiple selection rounds. In each round, a method scores a chunk of $5{,}120$ candidate problems from the pool, selects 256 of them, trains on them for 10 GRPO steps, and then generates fresh responses for both the candidate and query problems using its updated policy. These new responses are used for selection in the next round. All methods start from the same initial policy and use the same released reward functions and GRPO hyperparameters.

For evaluation, we sample 16 responses for each of the 500 held-out Countdown problems and report their mean correctness. Results are averaged over seeds 42 and 43 at evaluation steps completed by all runs.

Table 4: RL selection, training, and evaluation settings.

Setting	Value
Candidates and selection	5,120 candidate problems; 256 selected per round
Scoring responses	16 per candidate and per query problem
Training between selections	10 GRPO steps
Training batch and responses	128 problems per step; 128 responses per problem
Optimizer	AdamW (Loshchilov & Hutter, 2017), learning rate 10^{-6}
Sampling	Temperature 0.7; maximum response length 3,072 tokens
KL penalty	Disabled
Evaluation	500 test problems; 16 responses per problem
Training repetitions	Two seeds per method, averaged, for all four methods

A.2.2 Selection under corrupted rewards

Following the corrupted-reward setting of GradAlign (Yang et al., 2026), we use Qwen3-8B-Base (Yang et al., 2025), with 512 DAPO candidate problems (Yu et al., 2026) and 43 AMC22 query problems. We sample 64 responses per problem. For half of the candidates, we replace the correctness rewards of their sampled responses with independent fair-coin flips; the remaining candidates retain their true rewards. At the base model, each method selects 128 candidates. AccGreedy ranks candidates by how close their empirical reward rate is to 0.5. LESSER, full gradients, and AccGreedy select 34, 35, and 111 corrupted candidates, respectively, compared with 64 expected under uniform random selection.

A.3 GRACE teacher ranking

We use the released GRACE scorer (Panigrahi et al., 2025) with its default settings, replacing only the student's gradient features with LESSER. The teachers, prompts, responses, 512-dimensional random projection, and scoring rule are otherwise unchanged. Each teacher generates four responses to each of 512 shared prompts using vLLM (Kwon et al., 2023).

We evaluate three settings: (1) GSM8K with a Llama-3.2-1B student and 14 teachers, consisting of the original ten teachers plus Qwen2.5-14B-Instruct (Qwen et al., 2025), Phi-4 (Abdin et al., 2024), OLMo-2-13B-Instruct (OLMo et al., 2024), and Gemma-2-27B-it (Team et al., 2024); (2) GSM8K with an OLMo-2-1B student (OLMo et al., 2024) and 10 teachers; and (3) MATH (Hendrycks et al., 2021) with a Llama-3.2-3B student and 10 teachers. Appendix D reports the resulting teacher rankings.

A.4 Efficiency accounting and timing

SFT extraction FLOPs. We compare feature-extraction FLOPs for LESSER and the released four-checkpoint LESS configuration on Llama-2-7B. The quantities used in the calculation are summarized in Table 5.

A forward pass costs approximately $2PT$ FLOPs (Hoffmann et al., 2022; Kaplan et al., 2020). LESSER requires one forward pass, while its two-sided projection of the output-layer gradient adds $2T_r\big(|V|m + dn + mn\big)$, giving

$$F_{\text{LESSER}} = 2PT + 2T_r\big(|V|m + dn + mn\big).$$

At each checkpoint, LESS performs a forward pass, propagates activation gradients through the frozen model, computes the LoRA-gradient terms, and projects each example's gradient to $D = 8{,}192$ dimensions. Its total cost is therefore

$$F_{\text{LESS}} = 4\left(4PT + 6P_{\text{LoRA}}T + 2DP_{\text{LoRA}}N\right).$$

Table 5: Quantities used for the Llama-2-7B SFT feature-extraction FLOP calculation.

Quantity	Symbol	Value
Model parameters	P	6,738,415,616
LoRA parameters	P_{LoRA}	134,217,728
Pool examples	N	197,196
Processed tokens	T	86,343,509
Response tokens	T_r	25,430,302
Vocabulary size	$\lvert V\rvert$	32,000
Hidden dimension	d	4,096
LESS projection dimension	D	8,192
LESSER projection dimensions	(m, n)	$(64, 128)$
LESS checkpoints	C	4

Using the values in Table 5, the resulting ratio is

$$\frac{F_{\mathrm{LESS}}}{F_{\mathrm{LESSER}}} = 9.7\times .$$

Under the standard approximation of $2PT$ for a forward pass and $6PT$ for a forward–backward pass, abstracting away LoRA and projection overhead, four-checkpoint LESS costs approximately $24PT$, giving the commonly used estimate of approximately $12\times$.

SFT extraction wall clock. We measure SFT feature-extraction time for LESSER and LESS on the same A100 80GB and the same 10,000-example prefix of the candidate pool, timing the full extraction pipeline from data loading through feature writing. LESSER takes 576 seconds, while one checkpoint of released LESS takes 2,563 seconds. Extrapolating both measurements to the full 197,196-example pool and accounting for the four checkpoints used by LESS gives 3.16 hours for LESSER and 56.2 hours for LESS, a $17.8\times$ difference.

SFT extraction FLOPs relative to downstream training. We next ask whether the extraction savings remain substantial once downstream fine-tuning is included. For one target and training run, we model the combined extraction and fine-tuning cost as $2PT_{\mathrm{pool}} + 6PT_{\mathrm{train}}$ for LESSER and $F_{\mathrm{LESS}} + 6PT_{\mathrm{train}}$ for LESS, where F_{LESS} is the detailed four-checkpoint extraction cost above. As shown in Table 6, feature extraction accounts for at least 97.0% of LESS's modeled extraction-plus-training FLOPs across the tested budgets. Consequently, the combined extraction and fine-tuning cost is 7.7–$9.5\times$ larger for LESS than for LESSER. We assume two training epochs and equal mean token length for pool and selected examples. These estimates exclude warmup training, query processing, selection, and evaluation.

Table 6: SFT computation costs for one target and training run.

	Extraction share / %		
Budget k	LESSER	Four-checkpoint LESS	Extraction + training FLOP ratio
1,000	97.0	99.7	$9.47\times$
5,000	86.8	98.5	$8.58\times$
10,000	76.7	97.0	$7.69\times$

RL scoring FLOPs and storage. In its scoring pass, LESSER runs one forward pass over each scored token and projects the output residual and hidden state before accumulating their outer product. For vocabulary size $|V| = 151{,}936$, hidden dimension $d = 1{,}536$, and projection dimensions $m = 64$ and $n = 128$, the projection costs

$$2(|V|m + dn) + 2mn = 19{,}857{,}408$$

FLOPs per scored token, and the forward pass $2P = 3.09\times10^9$ for $P = 1{,}543{,}714{,}304$ parameters, 3.11×10^9 in total. Full-gradient scoring instead requires a separate forward–backward pass, costing approximately $6P = 9.26\times10^9$ FLOPs per scored token, $3.0\times$ the cost of LESSER.

Table 7: Computation and per-problem storage for RL scoring.

Feature	FLOPs / scored token	Stored feature
LESSER	3.11×10^9	32.8 kB
Full gradients	9.26×10^9	6.2 GB

RL scoring wall clock. LESSER scores each selection round with projected output-layer features, computed in a separate forward pass over the sampled responses after rollout generation. On the $81{,}920$ candidate and $3{,}200$ query responses of one round, on one A100 80GB PCIe and in one job, this pass takes 18.1 minutes, against 63.3 minutes for the full-gradient forward–backward pass, a $3.5\times$ reduction.

GRACE extraction wall clock. We estimate the cost of extracting GRACE features for all teacher responses by timing a subset and extrapolating to the full dataset. For each teacher, we randomly sample 32 responses and extract their features one at a time on one A100 80GB, after three untimed warm-up responses. We use the fp32 student and the same 512-dimensional projector configuration for both methods. For each teacher, we multiply the measured mean time per response by its $2{,}048$ responses, then sum across teachers to obtain the total extraction times in Table 8. Peak memory is measured directly during these runs and is not extrapolated. Under this estimate, LESSER is 3.9–$5.4\times$ faster than full-gradient extraction while using less peak memory. As a correctness check, the features produced by the timed implementation have cosine similarity of at least 0.99 with the stored features used for ranking on two sampled responses from each setting.

Table 8: **GRACE feature-extraction cost on one A100 80GB.** Extraction time is estimated from 32 timed responses per teacher and extrapolated to all $2{,}048$ responses per teacher; peak memory is measured directly.

Setting	Full gradients	LESSER	Speedup	Peak memory, GiB full / LESSER
GSM8K, Llama-3.2-1B, 14 teachers	6.49 h	1.65 h	3.9×	19.6 / 12.3
GSM8K, OLMo-2-1B, 10 teachers	5.34 h	0.99 h	5.4×	17.0 / 7.0
MATH, Llama-3.2-3B, 10 teachers	14.68 h	3.09 h	4.8×	39.8 / 24.2

B Output-layer gradient feature and projection

B.1 Proof of Proposition 1

Proposition 1. *Let W be the final vocabulary projection, h_t the hidden state predicting y_t, and e_{y_t} the one-hot vector for y_t. Let W_{out} be a copy of W used only as the readout, so that h_t does not depend on it. Then, the output-layer gradient of Equation* (1) *satisfies*

$$g_p \coloneqq \nabla_{W_{\text{out}}} L_p(\theta) = \sum_t \alpha_t \big(\operatorname{softmax}(W h_t) - e_{y_t}\big) h_t^\top .$$

Proof. Fix a token t in the loss, and write $a_t = W h_t \in \mathbb{R}^{|V|}$ for its logits and $\pi_t = \operatorname{softmax}(a_t)$ for the predicted distribution $\pi_\theta(\cdot \mid x, y_{<t})$. The loss at position t is

$$\ell_t = -\log \pi_t[y_t] = -a_t[y_t] + \log \sum_{v=1}^{|V|} \exp\big(a_t[v]\big).$$

Step 1: gradient with respect to the logits. For each vocabulary index i,

$$\frac{\partial \ell_t}{\partial a_t[i]} = -e_{y_t}[i] + \frac{\exp\big(a_t[i]\big)}{\sum_{v=1}^{|V|} \exp\big(a_t[v]\big)} = \pi_t[i] - e_{y_t}[i], \qquad \text{so} \qquad \nabla_{a_t} \ell_t = \pi_t - e_{y_t} = r_t.$$

Step 2: gradient with respect to W_{out}. The entry $(W_{\text{out}})_{ij}$ enters only the logit $a_t[i] = \sum_{j'=1}^{d} (W_{\text{out}})_{ij'} h_t[j']$, with $\partial a_t[i]/\partial (W_{\text{out}})_{ij} = h_t[j]$. By the chain rule,

$$\frac{\partial \ell_t}{\partial (W_{\text{out}})_{ij}} = \frac{\partial \ell_t}{\partial a_t[i]}\, h_t[j] = r_t[i]\, h_t[j], \qquad \text{so} \qquad \nabla_{W_{\text{out}}} \ell_t = r_t h_t^\top .$$

Step 3: sum over positions. By Equation (1), $L_p = \sum_t \alpha_t \ell_t$ with fixed weights α_t, and the gradient is linear, so $g_p = \nabla_{W_{\text{out}}} L_p = \sum_t \alpha_t\, r_t h_t^\top$. □

B.2 Projection

Storing the output-layer gradient g_p of Proposition 1 for every pool example is impractical, so LESSER stores a random projection of it, as in TRAK (Park et al., 2023). The matrices $\Pi_1 \in \{\pm 1/\sqrt{m}\}^{m \times |V|}$ and $\Pi_2 \in \{\pm 1/\sqrt{n}\}^{d \times n}$ have independent Rademacher entries, are drawn once per model, and are shared across pool and query examples. The feature is

$$\phi_p = \frac{\text{vec}(\Pi_1 g_p \Pi_2)}{\|\Pi_1 g_p \Pi_2\|_F}, \qquad \Pi_1 g_p \Pi_2 = \sum_t \alpha_t\, (\Pi_1 r_t)(\Pi_2^\top h_t)^\top, \tag{3}$$

so each token's residual and hidden state are projected before their outer product, without forming the $|V| \times d$ gradient. We use $m = 64$ and $n = 128$, giving 8,192 dimensions, for SFT and the sketched RL comparison.

C Additional Results: Supervised Fine-Tuning

C.1 SFT results across four models

We test whether the Llama-2 results extend across model families, using four models, five tasks, three budgets, and three training seeds. The full grid is reported in Tables 9–11.

LESSER has a higher mean than Random on TyDiQA for every model and budget, by 0.5 to 9.6 points. On TyDiQA, LESSER is within 0.9 points of LESS or above it on Qwen3 and OLMo3 at every budget. On the two Llama models, LESS is higher on TyDiQA by 3.1 to 4.4 points at $k = 1{,}000$, and the difference shrinks to 0.8 to 1.2 points at $k = 10{,}000$. On MMLU-Pro, the four methods lie within 1.3 points of each other on every model at $k = 5{,}000$ and $k = 10{,}000$.

Table 9: Held-out performance at $k = 1{,}000$, mean $\pm$ sample standard deviation over three seeds; each task uses its target metric on a 0–100 scale, higher is better.

Model	Task	LESSER	LESS	RDS+	Random
Llama-3.2-3B	TyDiQA	63.3±0.5	**66.4±0.2**	45.7±0.4	53.7±1.6
Llama-3.2-3B	MMLU-Pro	22.1±2.3	22.8±0.5	**25.3±0.8**	14.9±8.9
Llama-3.2-3B	GSM8K	32.3±0.9	**34.2±0.5**	32.3±0.7	29.2±0.7
Llama-3.2-3B	Codex	**61.3±2.4**	56.3±3.0	60.6±1.7	59.2±7.8
Llama-3.2-3B	BBH	48.0±0.2	47.6±0.2	**48.7±0.5**	47.1±0.5
Llama-2-7B	TyDiQA	50.2±0.4	**54.6±0.4**	46.2±0.6	49.4±0.7
Llama-2-7B	MMLU-Pro	**19.7±0.0**	19.2±0.1	19.6±0.1	16.3±0.6
Llama-2-7B	GSM8K	19.2±0.3	19.2±0.7	**22.2±0.7**	15.0±1.3
Llama-2-7B	Codex	**29.5±0.4**	25.0±2.0	27.7±3.1	26.4±2.4
Llama-2-7B	BBH	38.7±0.4	**40.5±0.1**	40.3±0.3	39.1±0.4
Qwen3-4B-Base	TyDiQA	70.7±0.4	**71.6±0.0**	66.4±0.6	68.6±1.5
Qwen3-4B-Base	MMLU-Pro	48.5±0.9	46.7±0.5	**49.3±1.8**	45.6±2.8
Qwen3-4B-Base	GSM8K	79.6±0.2	81.8±0.2	79.8±0.5	**82.3±0.2**
Qwen3-4B-Base	Codex	90.3±1.0	**92.1±1.0**	91.2±1.4	91.7±1.4
Qwen3-4B-Base	BBH	70.2±0.2	70.7±0.2	70.6±0.2	**70.8±0.3**
OLMo3-7B	TyDiQA	61.1±0.8	**62.0±0.5**	55.3±0.4	57.8±1.1
OLMo3-7B	MMLU-Pro	**38.6±0.1**	37.4±0.0	29.7±8.3	37.6±1.6
OLMo3-7B	GSM8K	73.3±0.5	74.2±0.0	**75.0±0.4**	61.5±14.2
OLMo3-7B	Codex	**84.0±2.4**	80.0±0.4	**84.0±3.1**	80.6±0.8
OLMo3-7B	BBH	**64.5±0.1**	64.4±0.1	63.0±0.0	64.4±0.2

Table 10: Held-out performance at $k = 5{,}000$.

Model	Task	LESSER	LESS	RDS+	Random
Llama-3.2-3B	TyDiQA	64.3±1.4	**66.3±0.2**	58.2±0.9	60.5±0.2
Llama-3.2-3B	MMLU-Pro	**26.9±0.4**	26.1±0.2	**26.9±0.1**	26.6±0.1
Llama-3.2-3B	GSM8K	34.6±0.1	34.5±0.6	**40.0±0.6**	33.3±0.9
Llama-3.2-3B	Codex	57.4±2.0	58.1±3.1	**61.0±0.4**	59.0±1.0
Llama-3.2-3B	BBH	48.5±0.3	49.6±0.4	**49.7±0.3**	48.5±0.5
Llama-2-7B	TyDiQA	51.7±0.4	**52.9±0.1**	50.4±0.4	51.2±0.2
Llama-2-7B	MMLU-Pro	20.2±0.3	20.2±0.3	**20.7±0.1**	20.0±0.4
Llama-2-7B	GSM8K	21.7±0.5	21.3±0.5	**23.0±0.4**	16.7±0.8
Llama-2-7B	Codex	29.1±2.4	24.5±2.1	**30.9±1.6**	27.3±1.4
Llama-2-7B	BBH	41.0±0.4	**42.9±0.4**	41.8±0.2	40.3±0.6
Qwen3-4B-Base	TyDiQA	**73.4±0.2**	71.7±0.3	71.2±0.5	70.9±0.7
Qwen3-4B-Base	MMLU-Pro	48.7±0.4	49.2±0.0	**50.0±0.4**	49.0±0.2
Qwen3-4B-Base	GSM8K	79.7±0.8	80.3±0.7	78.7±0.3	**80.4±0.5**
Qwen3-4B-Base	Codex	90.3±0.4	92.1±1.4	**92.3±2.2**	**92.3±2.2**
Qwen3-4B-Base	BBH	69.5±0.1	**71.5±0.0**	70.5±0.6	70.3±0.6
OLMo3-7B	TyDiQA	63.5±0.4	**63.6±0.5**	59.1±0.2	60.6±0.5
OLMo3-7B	MMLU-Pro	37.8±0.1	37.8±0.4	**38.2±0.2**	37.6±0.4
OLMo3-7B	GSM8K	73.6±0.4	73.0±1.8	**76.4±0.8**	28.7±12.1
OLMo3-7B	Codex	81.5±1.7	81.5±0.8	**82.9±4.6**	82.0±1.0
OLMo3-7B	BBH	64.3±0.2	64.7±0.2	63.2±0.2	**65.0±0.2**

Table 11: Held-out performance at $k = 10{,}000$.

Model	Task	LESSER	LESS	RDS+	Random
Llama-3.2-3B	TyDiQA	66.0±0.2	**67.2±0.1**	60.3±0.1	63.6±0.4
Llama-3.2-3B	MMLU-Pro	26.5±0.2	27.1±0.4	**27.3±0.1**	26.9±0.3
Llama-3.2-3B	GSM8K	35.2±0.4	36.7±0.9	**39.6±0.3**	35.2±0.9
Llama-3.2-3B	Codex	57.4±0.7	58.3±2.1	**60.1±2.0**	58.3±4.3
Llama-3.2-3B	BBH	49.3±0.5	49.1±0.6	**49.4±0.4**	48.6±0.2
Llama-2-7B	TyDiQA	52.8±0.4	**53.6±0.1**	50.7±0.6	52.1±0.2
Llama-2-7B	MMLU-Pro	**21.2±0.3**	21.0±0.2	20.7±0.3	20.7±0.2
Llama-2-7B	GSM8K	22.2±0.5	22.6±1.1	**24.2±0.3**	17.3±1.0
Llama-2-7B	Codex	30.4±0.7	26.6±0.8	29.7±1.4	**30.6±1.6**
Llama-2-7B	BBH	41.7±0.3	**43.1±0.1**	42.7±0.0	41.5±0.4
Qwen3-4B-Base	TyDiQA	**72.7±0.1**	72.2±0.0	70.8±0.1	70.7±0.3
Qwen3-4B-Base	MMLU-Pro	48.3±0.4	48.8±0.3	**49.5±0.3**	48.5±0.3
Qwen3-4B-Base	GSM8K	78.7±0.6	**80.7±0.6**	79.2±0.4	79.7±0.6
Qwen3-4B-Base	Codex	91.9±1.2	**92.8±1.7**	91.9±1.2	92.3±1.7
Qwen3-4B-Base	BBH	69.7±0.5	**71.7±0.1**	70.2±0.2	69.8±0.3
OLMo3-7B	TyDiQA	63.6±0.2	**64.4±0.1**	60.1±0.5	61.0±0.5
OLMo3-7B	MMLU-Pro	37.4±0.4	**38.7±0.2**	38.1±1.0	38.1±0.3
OLMo3-7B	GSM8K	74.7±0.2	74.6±0.1	**76.1±0.6**	70.0±6.7
OLMo3-7B	Codex	**82.9±2.6**	79.1±0.7	81.8±0.7	82.7±2.1
OLMo3-7B	BBH	65.0±0.5	65.0±0.2	63.4±0.1	**65.1±0.1**

C.2 Query-loss curves across models and tasks

This subsection gives the detailed results of the similarity-bin diagnostic in Section 4. Figure 4 shows Llama-2; Figures 8–10 show the other models.

Higher similarity generally predicts lower query loss: mean bin-index Spearman correlation is 0.714 for LESSER, 0.887 for LESS, and 0.058 for RDS+. LESSER has positive correlation in 19 of 20 pairs and its closest bin has the lowest loss in 18. OLMo3/MMLU-Pro is the correlation exception, with $\rho = -0.030$.

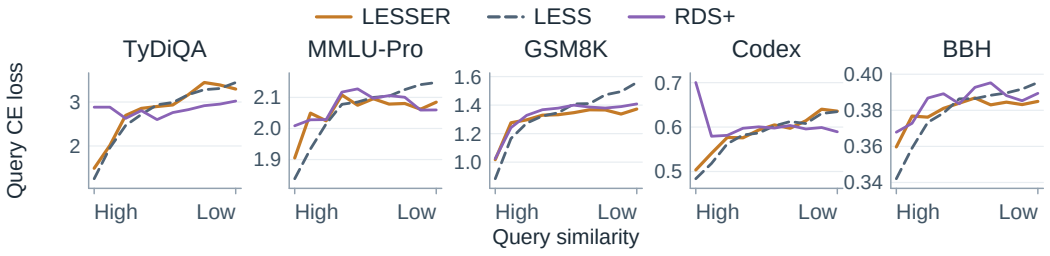

Figure 8: **Llama-3.2-3B: query loss across similarity bins.** Each bin uses one training run; bin 0 is closest to the queries.

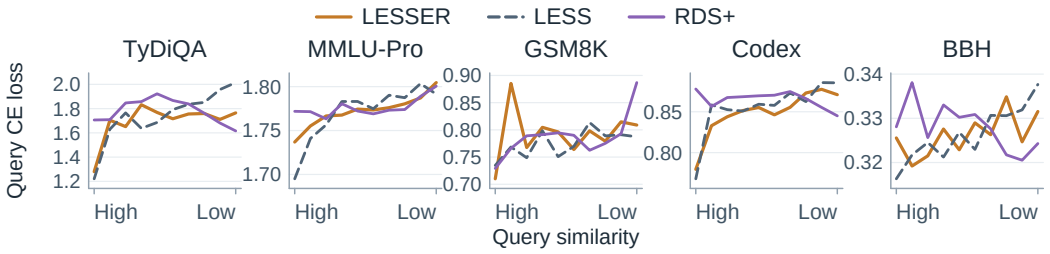

Figure 9: **Qwen3-4B-Base: query loss across similarity bins.** Each bin uses one training run; bin 0 is closest to the queries.

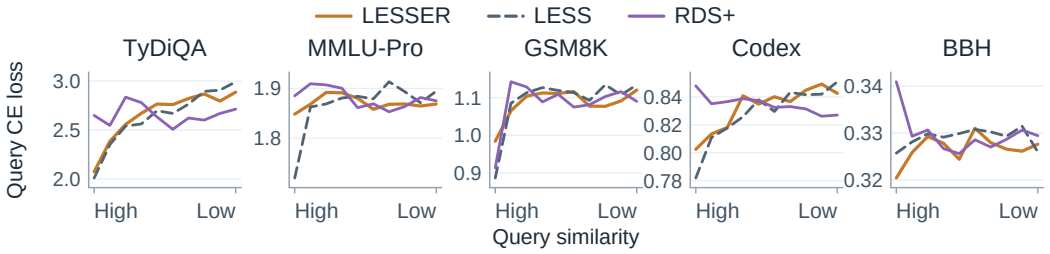

Figure 10: **OLMo3-7B: the closest LESSER bin has the lowest query loss on all five tasks.** Loss is not monotone across bins on MMLU-Pro. Each bin uses one training run.

D GRACE TEACHER RANKINGS

Table 12 gives the detailed results of the GRACE experiment in Section 4. With LESSER features, GRACE ranks the teachers nearly as it does with full gradients and picks the same top-ranked teacher in all three settings.

Table 12: Teacher-ranking agreement between LESSER and full-gradient GRACE. Spearman correlation uses the complete ranking.

Task	Student	Teachers	Spearman	Top-ranked teacher
GSM8K	Llama-3.2-1B	14	0.938	Qwen2.5-1.5B-Instruct
GSM8K	OLMo-2-1B	10	0.964	Qwen2.5-3B-Instruct
MATH	Llama-3.2-3B	10	0.976	Qwen2.5-3B-Instruct

E OVERLAP OF DIFFERENT SELECTION ALGORITHMS IN SFT

Table 13 compares the saved LESSER and released LESS selections on all four models. Each set contains 5,000 unique indices in the same 197,196-example Tulu pool. Jaccard similarity is $|S_{\text{LESSER}} \cap S_{\text{LESS}}|/|S_{\text{LESSER}} \cup S_{\text{LESS}}|$; it measures item overlap.

Table 13: LESSER and LESS select substantially different examples. Each cell gives the number of shared examples and the Jaccard similarity.

Model	TyDiQA	MMLU-Pro	GSM8K	Codex	BBH	Mean Jaccard
Llama-2-7B	491 / 0.052	505 / 0.053	324 / 0.033	214 / 0.022	228 / 0.023	0.037
Llama-3.2-3B	407 / 0.042	433 / 0.045	441 / 0.046	266 / 0.027	186 / 0.019	0.036
Qwen3-4B-Base	594 / 0.063	173 / 0.018	337 / 0.035	269 / 0.028	143 / 0.015	0.032
OLMo3-7B	1,004 / 0.112	190 / 0.019	250 / 0.026	476 / 0.050	182 / 0.019	0.045

F Details of the analysis in Section 5

F.1 Per-example scores

Motivation. We first ask how much of the full-gradient selection signal is already present in the output layer at the level of individual examples. Let L_Q be the aggregate loss on the query set $\mathcal{Q}$, with gradient $G_Q = \nabla_\theta L_Q$. For a candidate p with gradient G_p, we let local utility be the reduction in query loss from a small SGD step on p, normalized by the size of that step:

$$U(p) = \frac{L_Q(\theta) - L_Q(\theta - \eta G_p)}{\eta \|G_p\|} \approx \frac{G_Q^\top G_p}{\|G_p\|}. \tag{4}$$

Thus, the cosine between candidate and query gradients provides a first-order measure of how useful an individual candidate is for the query set.

Setup. We evaluate these per-example scores on 20 model–task pairs. For each pair, we sample up to 3,000 candidates from the pool and, for the two 7B models, keep only examples of at most 1,024 tokens, leaving 1,903–3,000 candidates per setting. For each example, we compute fp32 gradients of all native parameters at the base model. To measure how the ranking changes as more layers of the model are included, Figure 11 starts with the output layer and progressively adds decoder blocks from last to first, including each block's attention, MLP, and normalization parameters.

Findings. Output-layer and full-gradient cosine scores have a median Spearman correlation of 0.52 across the 20 model–task pairs, with per-model medians ranging from 0.36 to 0.72. Adding more layers steadily closes this gap: the median correlation with the full-gradient ranking rises from 0.52 for the output layer alone to 0.65 with half of the decoder blocks and to 1.00 when all decoder blocks are included. Thus, output-layer gradients capture a substantial but incomplete part of the per-example ordering; recovering the full-gradient ranking generally requires information from much of the network.

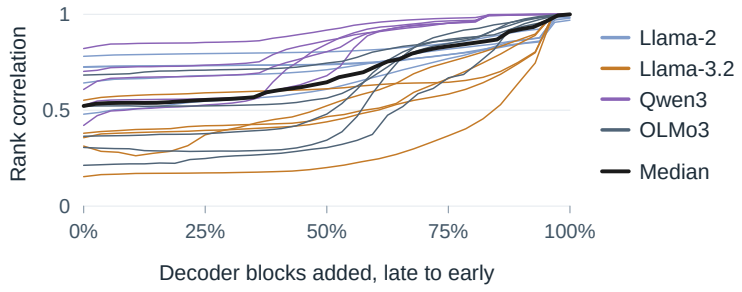

Figure 11: **Per-example rankings depend on gradients throughout the network.** Spearman correlation with the full-gradient cosine ranking as decoder blocks are added to the output layer from last to first. Candidates are ranked by their cosine with the query gradient over the included parameters. Thin lines denote model–task pairs, colored by model, and the black line shows the median.

To characterize what is lost by using only the output layer, we decompose the full-gradient score into contributions from the output layer and the remaining parameters. Expand

$$G_p = \begin{bmatrix} G_p^{\text{out}} \\ G_p^{\text{body}} \end{bmatrix}, \qquad G_Q = \begin{bmatrix} G_Q^{\text{out}} \\ G_Q^{\text{body}} \end{bmatrix},$$

$$G_Q^\top G_p = \left(G_Q^{\text{out}}\right)^\top G_p^{\text{out}} + \left(G_Q^{\text{body}}\right)^\top G_p^{\text{body}}.$$

Since $\|G_Q\|$ is constant across candidates, have

$$\frac{G_Q^\top G_p}{\|G_p\|} = \underbrace{\frac{\left(G_Q^{\text{out}}\right)^\top G_p^{\text{out}}}{\|G_p\|}}_{H(p)} + \underbrace{\frac{\left(G_Q^{\text{body}}\right)^\top G_p^{\text{body}}}{\|G_p\|}}_{B(p)}.$$

So $H(p)$ and $B(p)$ are the relative contributions of output-layer and body layers.

Across model–task pairs, $H(p)$ and $B(p)$ have a median correlation of 0.36, indicating that the remaining layers do not simply reproduce the output-layer contribution. However, candidates favored by the output layer also tend to receive favorable contributions from the rest of the network. Let $\mathcal{T}$ denote the top 10% of candidates under the output-layer cosine score, and define

$$\Delta_H = \mathbb{E}_{p\in\mathcal{T}}[H(p)] - \mathbb{E}_{p\in\mathcal{D}}[H(p)], \qquad \Delta_B = \mathbb{E}_{p\in\mathcal{T}}[B(p)] - \mathbb{E}_{p\in\mathcal{D}}[B(p)].$$

We find $\Delta_B > 0$ in 19 of 20 model–task pairs, with median $\frac{\Delta_B}{\Delta_H} = 0.82$. So candidates favored by the output layer are usually also favored by the rest of the network.

F.2 Batch gradient similarity between selections

Setup. We compare the full-model gradients induced by batches from different selections at the base model θ_0. Let G_S be the gradient of the recipe's token-averaged loss on batch S, and let μ be the expected gradient of a uniformly sampled batch of the same size. To measure alignment beyond this shared random-batch direction, we use the centered batch-gradient alignment between the batches S of one method and the batches S' of another

$$\frac{\operatorname{avg}(G_S-\mu)^\top(G_{S'}-\mu)}{\sqrt{\operatorname{avg}\|G_S-\mu\|^2\ \operatorname{avg}\|G_{S'}-\mu\|^2}}, \tag{5}$$

where the numerator averages over all pairs of a batch S and a batch S', and each squared norm averages over the batches of one sample. Because the population mean random-batch gradient $\mu = \mathbb{E}[G_S]$ is unknown, we estimate it from the random samples that are not part of the comparison, described below. When neither sample is random, two of the four random samples give $\hat{\mu}_1$ and the other two give $\hat{\mu}_2$, averaged over the three ways to split them; when one sample is random, any two of the other three give $\hat{\mu}_1$ and $\hat{\mu}_2$, averaged over the choices. We then replace $(G_S-\mu)^\top(G_{S'}-\mu)$ with $(G_S-\hat{\mu}_1)^\top(G_{S'}-\hat{\mu}_2)$, so that no random example contributes to both mean estimates, and estimate each squared norm as $(G_S-\hat{\mu}_1)^\top(G_S-\hat{\mu}_2)$.

For each of the 20 model–task pairs, we sample $1{,}024$ examples from the $k = 5{,}000$ selections of LESSER, LESS, RDS+, and Random. We also draw three additional uniform samples of $1{,}024$ examples from the candidate pool, giving four random samples in total.

We use these samples to construct two baselines in Figure 7a. The Random baseline compares each random sample with LESS, while the LESS baseline compares two disjoint batches drawn from the same LESS selection.

Findings. Raw batch gradients can be strongly aligned even under random sampling: at batch size 128, two random batches have median cosine 0.81 across the 20 model–task pairs. Thus, raw cosine largely reflects a direction shared by typical training batches, rather than the effect of selection itself. We therefore center each batch gradient by the expected random-batch gradient.

Under this centered measure, LESSER batches remain substantially aligned with LESS batches. At batch size 128, their similarity is positive in 17 of 20 pairs and typically exceeds the RDS+ and random baselines (Figure 7a). Across batch sizes 64–512, the median over model–task pairs of the ratio between the LESSER–LESS similarity and the similarity between two disjoint LESS batches is 0.58–0.66.

F.3 Training trajectories

Setup. For each of the 20 model–task pairs, we train for 80 AdamW steps with batch size 128 on the $k = 5{,}000$ selected examples, following Appendix A.1. We use two training seeds each for LESSER and LESS and one run for each of two random selections. At every step, we record the query loss.

Figure 12 shows the query-loss curves of every model–task pair; Figure 7b summarizes these curves by the median share of the largest reduction reached at each step.

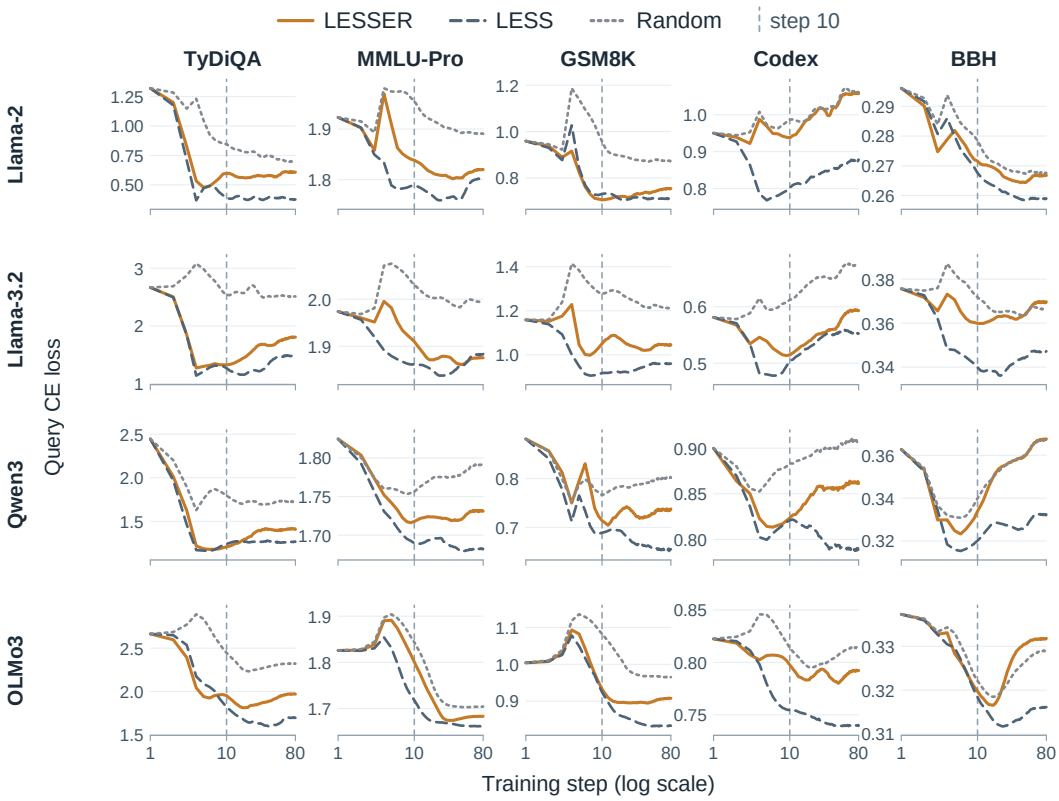

Figure 12: **Query loss across training steps.** Each panel is one model–task pair.

F.4 RL AND GRACE STATISTICS

RL. We analyze the first selection round of the Countdown experiment, restricting to the 1,835 of 5,120 candidate problems with nonzero advantage. Each problem aggregates 16 sampled responses, and we compare its LESSER and full-gradient scores at the base policy. The two rankings have Spearman correlation $\rho = 0.69$. At the selection budget, they select 69% of the same problems, a Jaccard similarity of 0.53 against 0.075 for two random selections.

GRACE. We analyze the 14 teachers in the GSM8K experiment with the Llama-3.2-1B student. For each teacher, GRACE summarizes the student's gradients on 2,048 teacher responses using statistics of their spectrum and pairwise similarities. Replacing full gradients with output-layer gradients largely preserves these teacher-level quantities: across teachers, spectral entropy and effective rank have Spearman correlations 0.94 and 0.93, while the remaining aggregate statistics have correlations 0.75–0.98. This agreement does not hold at the response level: the 64 most central responses under the two representations have Jaccard similarity only 0.12.

AI USE STATEMENT

In this work, we used generative AI tools to help develop the conceptual framework of the analysis in Section 5, formulate mathematical claims, provide ingredients for their proofs, assist in writing proofs, including the proof of Proposition 1 and the derivations in Appendices B and A.4, propose and refine hypotheses, design and provide feedback on the research methodology and experiments, implement methods, including the LESSER feature extraction in the SFT, RL, and GRACE pipelines and the analysis code, clean and reformat the public Countdown, WebInstruct, and DAPO data into the RL candidate pool and its validation and test splits, and interpret results. We have not used generative AI tools to generate synthetic datasets, and assistance with translation and support for qualitative and thematic data analysis are not applicable to this work.

We additionally used generative AI tools to suggest experimental parameters, create and edit software code, create and modify scientific figures, draft and edit parts of the paper, suggest its structure, identify relevant literature, and search for information. All AI-assisted text, code, figures, analyses, and references were reviewed and verified by the first author. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.