

GRADFILTERING: Uncertainty-Aware Gradient Signal-to-Noise Data Selection for Instruction Tuning

Zhihang Yuan^{1*}, Chengyu Yue^{1*}, Long Huang¹, Litu Ou², Lei Shi^{1†}

¹Alibaba Cloud Computing, Hangzhou, China ²The University of Edinburgh, Edinburgh, UK
{yuanzhihang.yzh,yuechengyu.ycy, baixuan.hl, juetian.sl}@alibaba-inc.com
litu.ou@ed.ac.uk

Abstract

Modern instruction-tuning mixtures are increasingly large, redundant, and heterogeneous, making full-data supervised fine-tuning costly and often unnecessary. We propose GRADFILTERING, an objective-agnostic, validation-free data selection method that scores examples using uncertainty-aware proxy training dynamics. GRADFILTERING trains a lightweight proxy model with a LoRA ensemble and ranks examples by a Gradient Signal-to-Noise Ratio (G-SNR), which combines early-to-late gradient drop with late-stage ensemble disagreement. Because selection requires neither target validation examples nor target-model gradients, the selected subsets can be reused across downstream targets. Across Alpaca and Tulu-3-52K, dense LLaMA-2-7B, LLaMA-3-8B, and Qwen3-8B targets, and a Qwen3-30B-A3B MoE target, GRADFILTERING-selected 5–15% subsets consistently outperform random selection and are competitive with the target-aware LESS-T reference while remaining validation-free. We further evaluate broad model capabilities on objective benchmarks and analyze proxy choice, gradient windows, ensemble size, human preferences, and selection cost, demonstrating the practical utility of early proxy dynamics for reusable, compact instruction-tuning data selection. Code is available at this [GitHub repository](#).

1 Introduction

Instruction tuning has become a standard recipe for adapting large language models (LLMs) to follow human instructions (Ouyang et al., 2022). Yet modern supervised fine-tuning (SFT) corpora are not only large; they are also redundant, noisy, and increasingly heterogeneous mixtures of sources and task types. This makes full-data fine-tuning expensive and sometimes unnecessary, motivating data

selection methods that can identify compact subsets without sacrificing instruction-following quality.

Existing selection methods occupy different points in a cost–adaptivity trade-off. Static weak-proxy filters such as Superfiltering (Li et al., 2024) are validation-free and cheap, but score examples without observing how they are learned during fine-tuning. Target-aware gradient methods such as LESS-T (Xia et al., 2024) directly align training examples with validation examples, but require target/dev data and are therefore less reusable across downstream models or deployments. Inference-only methods such as DELIFT (Agarwal et al., 2025) avoid backpropagation by constructing pairwise in-context utility scores, but exact all-pairs scoring becomes expensive for large candidate pools. These trade-offs leave open a practical question: can we obtain a reusable, validation-free selection signal that is more adaptive than static proxy scoring?

Training dynamics and epistemic uncertainty provide two complementary signals for this goal. Per-example behavior during training can reveal learnability, ambiguity, redundancy, and noise, while uncertainty helps identify examples for which a model lacks stable evidence or consistent adaptation behavior (Gal and Ghahramani, 2016; Lakshminarayanan et al., 2017). LoRA-Ensemble (Mühlematter et al., 2026) shows that multiple LoRA adapters on a shared backbone can recover much of the accuracy and calibration benefit of deep ensembles at a fraction of the parameter and compute cost, suggesting that lightweight LoRA ensembles can serve as practical uncertainty probes. However, the combination of training dynamics and epistemic uncertainty has not yet been leveraged for uncertainty-aware data valuation in instruction tuning.

We propose GRADFILTERING, an objective-agnostic and validation-free data selection framework based on uncertainty-aware proxy training

*Equal contribution.

†Corresponding author.

dynamics. GRADFILTERING trains a small proxy model (e.g., GPT-2 (Radford et al., 2019)) with a LoRA ensemble and records per-example adapter gradients across early training. It then ranks examples using our new Gradient Signal-to-Noise Ratio (G-SNR) metric, which favors examples whose gradient signal drops consistently from early to later training while penalizing high late-stage ensemble disagreement. Because the score is computed only from gradients of the proxy training objective, it does not require task-specific rewards, manual annotations, or target validation examples.

This design is especially useful when a selected subset should be reused across multiple targets. In our experiments, the same validation-free selection pipeline is evaluated on Alpaca and Tulu-3-52K, dense LLaMA-2-7B, LLaMA-3-8B, and Qwen3-8B targets, and a Qwen3-30B-A3B MoE target. This setup tests reuse across datasets, dense model families, and a larger sparse target.

Our contributions are threefold:

1. **G-SNR.** We propose G-SNR, an uncertainty-aware gradient signal-to-noise criterion that scores examples by combining proxy gradient drop with LoRA-ensemble disagreement.
2. **GRADFILTERING.** We propose an efficient, objective-agnostic selection method that runs a LoRA ensemble with a default GPT-2 proxy and aggregates per-example gradients into our novel G-SNR utility.
3. **Strong empirical gains.** We evaluate GRADFILTERING across *Alpaca* and *Tulu-3-52K*, three dense target models (LLaMA-2-7B, LLaMA-3-8B, and Qwen3-8B), and a larger Qwen3-30B-A3B *MoE* target. With selected 5–15% subsets, GRADFILTERING achieves the best score in 13 of 18 dense-target settings, outperforms or ties Superfiltering in 16 of 18 settings, and remains competitive with the target-aware LESS-T reference while remaining validation-free. We further show that the selected subsets preserve broad model capabilities on objective benchmarks and provide initial evidence of transfer to task-specific summarization on *XSum-20K*.

2 Related Work

2.1 Data Selection for Instruction Tuning

Data valuation has long been studied through influence functions and related gradient criteria. Clas-

sical influence-function methods approximate the effect of upweighting or removing a training point through second-order Taylor expansions (Koh and Liang, 2017) or first-order approximations such as TracIn (Pruthi et al., 2020). While theoretically appealing, these methods are difficult to scale directly to modern LLM fine-tuning pipelines.

Recent methods adapt data selection to instruction tuning. LESS (Xia et al., 2024) builds an optimizer-aware gradient datastore and scores training examples by their alignment with target validation examples. This target-aware design can be strong when representative validation data are available, but the resulting scores are tied to that target setting. Superfiltering (Li et al., 2024) instead uses a weak proxy to compute static instruction-following difficulty scores, making selection cheap and validation-free, but without modeling how examples are learned during fine-tuning.

DELIFT (Agarwal et al., 2025) provides an inference-only alternative to gradient-based selection by computing a pairwise ICL utility between training examples and optimizing a submodular objective. This is attractive because it avoids back-propagation and naturally promotes coverage and diversity. However, exact DELIFT requires constructing an all-pairs utility matrix before selection, which scales quadratically with the candidate-pool size. For our 52K-example Alpaca and Tulu-3 candidate pools, exact scoring would require about 2.7B pairwise utilities per dataset, before considering multiple budgets and target models. We therefore view DELIFT as complementary to GRADFILTERING: DELIFT is preferable when exact or approximate pairwise utility computation is feasible, whereas GRADFILTERING is designed for objective-agnostic, validation-free, linear-in-data proxy scoring over full candidate pools.

2.2 Training Dynamics and Uncertainty

Training dynamics provide a complementary view of data utility. Dataset cartography (Swayamdipta et al., 2020), forgetting events (Toneva et al., 2019), and early-gradient criteria such as GrNd and EL2N (Paul et al., 2021) show that per-example behavior during training can reveal difficulty, ambiguity, redundancy, and noise. GRADFILTERING adapts this perspective to instruction tuning by measuring how per-example LoRA-gradient norms evolve during proxy training.

Epistemic uncertainty is also important in large-scale instruction tuning, where data may be noisy,

out-of-distribution, or underspecified. Approximate Bayesian methods such as Monte Carlo dropout (Gal and Ghahramani, 2016) and deep ensembles (Lakshminarayanan et al., 2017) quantify uncertainty through disagreement, but full LLM ensembles are impractical. LoRA (Hu et al., 2022) enables parameter-efficient adaptation, and LoRA-Ensemble (Mühlematter et al., 2026) uses multiple adapters on a shared backbone to approximate ensemble behavior at much lower cost. We use this mechanism as a lightweight uncertainty probe for data selection, combining early gradient drop with late-stage ensemble disagreement.

3 LoRA-Ensemble Approximation for Gradient-Based Data Valuation

3.1 LoRA-Ensemble Approximation

We employ a parameter-efficient ensemble based on LoRA adapters, where all ensemble members share a frozen backbone θ_0 , but each maintains its own low-rank adapter parameters $\Delta\theta^{(m)}$ for $m = 1, \dots, M$. For each self-attention layer, we attach rank- r adapters to the attention projections as in standard LoRA, and initialize the adapters of different ensemble members independently.

Let $\Delta\theta_e^{(m)}$ denote the LoRA parameters of member m after e epochs of fine-tuning, and

$$\theta_e^{(m)} = \theta_0 + \Delta\theta_e^{(m)} \quad (1)$$

the full parameter vector. All members start from the same pretrained backbone with randomly initialized adapters, i.e., $\Delta\theta_0^{(m)}$ for $m = 1, \dots, M$, and are fine-tuned on $\mathcal{D}$ for T epochs with different random seeds or data orderings. This yields M adapted models that share a common backbone but follow distinct parameter trajectories. We denote by $g_i^{(m,e)}$ the per-example gradient for sample i produced by LoRA member m at epoch e (formally defined in Equation 2). Under first-order optimization, small parameter updates, and hence local changes in loss around a given checkpoint, are approximately linear in these gradients. Our data valuation method therefore relies on the collection of per-example gradients $\{g_i^{(m,e)}\}_{m,e}$ as a compact summary of how (x_i, y_i) drives the ensemble during training.

In practice, we train the ensemble by minimizing the average per-member loss on each batch, $\mathcal{L}_{\text{ens}} = \frac{1}{M} \sum_{m=1}^M \mathcal{L}(f_{\theta_e^{(m)}}(x_i), y_i)$, and backpropagating this objective with respect to each adapter.

This vectorized implementation is equivalent (up to a constant rescaling of the learning rate) to training M independent LoRA models that reuse the same backbone computation. Crucially, we do *not* optimize the loss of an averaged prediction, $\mathcal{L}(\frac{1}{M} \sum_m f_{\theta_e^{(m)}}(x_i), y_i)$, which would explicitly encourage all members to correct the same ensemble error and thus suppress the epistemic disagreement that our gradient-variance statistics are intended to capture.

Gradient-based view of LoRA ensembles. Our use of a LoRA ensemble relies on two structural properties of parameter-efficient fine-tuning. First, empirical studies suggest that LoRA and related adapters typically keep the fine-tuned model close to the pretrained backbone, so that a first-order Taylor expansion around θ_0 provides an accurate local linearization of outputs and losses in the adapter subspace (Li et al., 2025). In this locally linear regime, the per-example LoRA gradient $g_i^{(m,e)}$ is the direction of steepest loss change for member m at epoch e , and its norm reflects how strongly (x_i, y_i) drives parameter updates.

Second, independently initialized LoRA adapters $\{\Delta\theta^{(m)}\}_{m=1}^M$ can be interpreted as approximate samples from a posterior over low-rank updates around θ_0 (Mühlematter et al., 2026; Balabanov and Linander, 2024). Under this view, the collection $\{g_i^{(m,e)}\}_{m=1}^M$ approximates samples from a distribution over update directions induced by (x_i, y_i) at epoch e : the ensemble mean estimates its expected update magnitude, and the ensemble variance quantifies epistemic uncertainty about how the model should adapt on this example. Our data valuation scheme builds on these two moments: we prioritize examples whose expected gradient is large in early epochs and decays over training, and we down-weight examples whose gradient variance remains high at late epochs, corresponding to regions where the ensemble fails to reach a stable consensus. These intuitions are instantiated in the uncertainty-aware gradient-drop score introduced in Section 4.3.

Empirical geometry of LoRA training dynamics.

To empirically support this view, we visualize the trajectories of LoRA-ensemble members in a low-dimensional embedding of their per-example gradient profiles (Figure 1). For each member m and epoch e , we construct a high-dimensional feature vector $\mathbf{g}_{m,e} \in \mathbb{R}^N$ whose i -th coordinate encodes

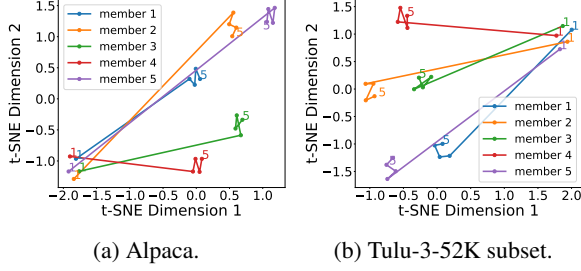


Figure 1: Trajectories of GPT-2 LoRA-ensemble members in a two-dimensional embedding of per-example relative gradient-drop profiles on Alpaca and the Tulu-3-52K subset. Each polyline corresponds to one ensemble member, and markers denote successive proxy-training epochs.

the relative gradient drop on example i between epoch 1 and epoch e . We stack all $\{\mathbf{g}_{m,e}\}_{m,e}$, apply PCA to reduce the dimensionality from N to a small d (e.g., $d = 25$) for denoising and stabilization, and then run t-SNE on the PCA embeddings to obtain two-dimensional points $\mathbf{z}_{m,e} \in \mathbb{R}^2$ in a shared gradient-profile space. For each member m , we connect $\mathbf{z}_{m,1} \rightarrow \mathbf{z}_{m,2} \rightarrow \dots \rightarrow \mathbf{z}_{m,T}$ to form a trajectory; this should be interpreted as mapping each epoch to a global pattern of relative gradient drops across all training examples for that member, rather than as a trajectory of individual examples.

Empirically, each polyline starts from a similar region of the embedding but quickly diverges, with different members moving toward distinct areas and then following stable, non-collapsing trajectories. At the same time, later-epoch points are more tightly clustered than early-epoch points, indicating that ensemble disagreement is gradually reduced as training proceeds. These patterns provide qualitative evidence that (i) LoRA adapters induce a rich, structured distribution over update directions around the frozen backbone, and (ii) the mean and variance of per-example gradients across members capture non-degenerate epistemic uncertainty, thereby justifying our use of these statistics for data valuation. The largest trajectory changes occur early in both Alpaca and Tulu-3-52K, motivating our default epoch $1 \rightarrow 2$ window; Section 5.4 tests longer windows directly.

3.2 Practical Considerations

In practice, we use $M = 5$ ensemble members, which Section 5.4 shows captures most of the gain from larger ensembles at lower cost. Since Figure 1 shows that LoRA-ensemble trajectories change most between epochs 1 and 2 and then stabilize,

we compute gradient statistics with terminal epoch $T = 2$.

4 GRADFILTERING : Data Selection with G-SNR

Figure 2 summarizes our gradient-based data selection pipeline GRADFILTERING. Section 3 establishes that a LoRA ensemble provides a rich, non-degenerate latent space, justifying Phases 1–2 in Figure 2. In this section, we focus on Phase 3 and derive our Gradient Signal-to-Noise Ratio (G-SNR) utility, which turns ensemble training dynamics into data-selection scores.

4.1 Problem Setup

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be a large-scale instruction-response dataset, where x_i is an instruction and y_i its reference response. We are given a pretrained language model and wish to fine-tune it for instruction following. Our goal is to assign each training example (x_i, y_i) a scalar utility score u_i and to use these scores to select a subset $\mathcal{D}' \subset \mathcal{D}$ (or to reweight $\mathcal{D}$) such that fine-tuning on $\mathcal{D}'$ achieves comparable or better performance than using the full dataset.

4.2 Per-Example Gradient Statistics

We characterize each training example by how difficult it is initially, how much the ensemble learns from it, and how stable its effect is across ensemble members, using its per-example LoRA gradients. Let T be the total number of training epochs. For example i in ensemble member m at epoch $e \in \{1, \dots, T\}$ we define

$$g_i^{(m,e)} = \nabla_{\Delta\theta_e^{(m)}} \mathcal{L}(f_{\theta_e^{(m)}}(x_i), y_i), \quad (2)$$

where $\mathcal{L}$ is the token-level training loss and gradients are taken only with respect to the LoRA adapters $\Delta\theta_e^{(m)}$. Thus $g_i^{(m,e)}$ directly measures how strongly (x_i, y_i) pushes the adapted parameters to change at a given epoch.

Gradient magnitude. For each example i and epoch e , we summarize its instantaneous training signal by the ensemble-averaged LoRA-gradient norm

$$G_i^{(e)} = \frac{1}{M} \sum_{m=1}^M \|g_i^{(m,e)}\|_2, \quad (3)$$

where $g_i^{(m,e)}$ is the per-example gradient of member m at epoch e . Larger $G_i^{(e)}$ indicates that (x_i, y_i)

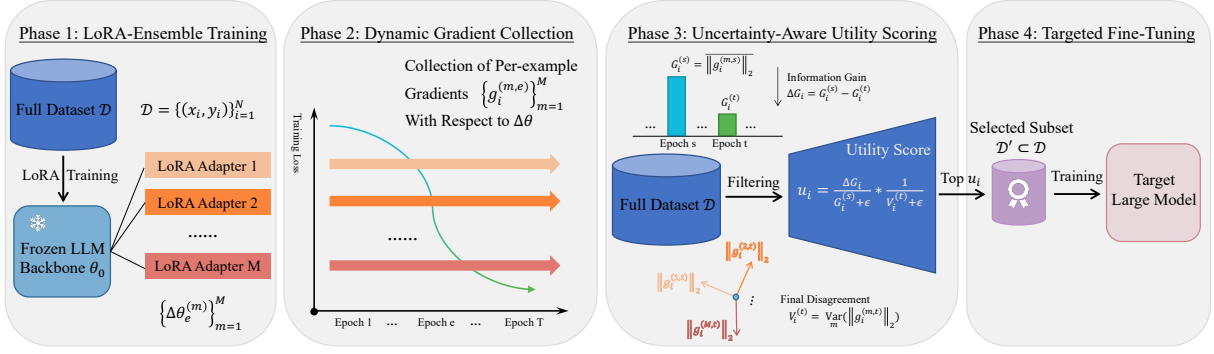


Figure 2: Overview of our gradient-based data selection pipeline. **Phase 1** trains a set of LoRA-ensemble members that share the same frozen backbone θ_0 on the full dataset $\mathcal{D}$, producing multiple adapters $\{\Delta\theta_e^{(m)}\}_{m=1}^M$ after e epochs. **Phase 2** collects *per-example* gradients with respect to $\Delta\theta$ at different epochs, yielding gradient profiles $\{g_i^{(m,e)}\}_{m=1}^M$. **Phase 3** calculates an uncertainty-aware utility score based on these profiles by combining (i) an *information-gain* term $\Delta G_i = G_i^{(s)} - G_i^{(t)}$ that measures gradient drop from early to later training, and (ii) a *disagreement* term $V_i^{(t)} = \text{Var}_m(\|g_i^{(m,t)}\|_2)$ that captures ensemble variability, via an SNR-style form $u_i \propto \frac{\Delta G_i}{G_i^{(s)} + \epsilon} \cdot \frac{1}{V_i^{(t)} + \epsilon}$. **Phase 4** selects the top-ranked subset $\mathcal{D}' \subset \mathcal{D}$ based on the utility score u_i for training the target model.

induces a stronger parameter update at that stage of training, reflecting its current difficulty or influence.

Ensemble disagreement via gradient variance.

To quantify how consistently the ensemble reacts to each example, we measure the variance of gradient norms across ensemble members at each epoch,

$$V_i^{(e)} = \frac{1}{M} \sum_{m=1}^M \|g_i^{(m,e)}\|_2^2 - \left(\frac{1}{M} \sum_{m=1}^M \|g_i^{(m,e)}\|_2 \right)^2 \quad (4)$$

$V_i^{(e)}$ serves as a stability indicator: low values indicate that ensemble members agree on the update magnitude for (x_i, y_i) , while high values signal persistent disagreement and potential ambiguity or noise.

4.3 Gradient Signal-to-Noise Ratio (G-SNR) for Data Selection

We propose a **Gradient Signal-to-Noise Ratio (G-SNR)** utility to select high-quality training examples from a large candidate pool. The key motivation is a global optimization view: because all samples share parameters, the utility of a data point should reflect not only its local training signal but also how *consistently* this signal aligns with the overall descent direction across different optimization trajectories.

Gradient-drop signal. We measure an example’s learning progress using a *gradient-drop* signal computed between an early training stage and

a later stage. Concretely, we train an ensemble of M lightweight adapters (e.g., LoRA) and record per-example gradient-norm statistics at two time points: an early epoch s and a later epoch t ($s < t$). Let $G_i^{(m,s)}$ and $G_i^{(m,t)}$ denote the gradient norm of sample i under ensemble member m at epochs s and t , and we define the ensemble means $G_i^{(s)} = \frac{1}{M} \sum_{m=1}^M G_i^{(m,s)}$ and $G_i^{(t)} = \frac{1}{M} \sum_{m=1}^M G_i^{(m,t)}$, as well as the later-stage variance $V_i^{(t)} = \frac{1}{M} \sum_{m=1}^M (G_i^{(m,t)})^2 - (G_i^{(t)})^2$. The raw drop $\Delta_i = G_i^{(s)} - G_i^{(t)}$ captures how much the sample’s gradient magnitude decreases as training proceeds: intuitively, examples that induce stable and effective descent tend to exhibit a clearer reduction.

Uncertainty normalization (G-SNR). To account for uncertainty in the gradient signal, we down-weight examples whose gradient statistics are highly variable across ensemble members. Our G-SNR utility is

$$u_i^{\text{G-SNR}} = \frac{G_i^{(s)} - G_i^{(t)}}{G_i^{(s)} + \epsilon} \cdot \frac{1}{V_i^{(t)} + \epsilon}, \quad (5)$$

where ϵ is a small constant for numerical stability. The first factor is a *relative* gradient drop that normalizes out scale effects (so that large-gradient examples are not trivially favored), while the second factor penalizes high-variance, unreliable signals by down-weighting examples whose late-stage gradients disagree across ensemble members. In this

sense, G-SNR behaves like a signal-to-noise ratio: it prefers large, consistent gradient drops (signal) while suppressing examples with high ensemble disagreement (noise).

Selection protocol. Given utilities $\{u_i^{\text{G-SNR}}\}_{i=1}^N$, we select the top- α fraction of examples as the training subset. We use the same computation and selection procedure across datasets and target models, making G-SNR objective-agnostic: it only assumes access to per-example gradients under a parametric loss, without relying on target validation examples, task-specific rewards, or handcrafted difficulty scores.

5 Experiments

Overview. In this section, we evaluate whether GRADFILTERING selects compact, reusable instruction-tuning subsets across dataset and target shifts. Unless otherwise specified, we select 5/10/15% subsets from Alpaca and Tulu-3-52K, LoRA-tune LLaMA-2-7B, LLaMA-3-8B, and Qwen3-8B, and report PWS against the corresponding full-data model. We compare with Random, Perplexity, Superfiltering, and target-aware LESS-T, and further evaluate objective benchmark performance, gradient-window and ensemble-size sensitivity, proxy choice, selection cost, MoE transfer, human preferences, and task-specific generalization on XSum-20K.

5.1 Experiments Setup

Datasets. We evaluate GRADFILTERING in two settings. For instruction tuning, we use two 52K candidate pools: **Alpaca** (Taori et al., 2023), a relatively homogeneous instruction-following corpus, and **Tulu-3-52K**, constructed by source-proportional sampling from the Tulu-3 SFT mixture (Lambert et al., 2025). This pairing contrasts homogeneous and heterogeneous candidate pools at matched scale. We evaluate instruction-following quality on WizardLM (Xu et al., 2024) (218 instructions) and Vicuna (Zheng et al., 2023) (80 instructions), and additionally use ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), TruthfulQA (Lin et al., 2022), and GSM8K (Cobbe et al., 2021) for objective evaluation. For task-specific generalization, we construct **XSum-20K** by sampling 20K examples from the official XSum training split (Narayan et al., 2018) and evaluate on

the complete test split; further dataset and sampling details are provided in the Appendix A.1.

Baselines. We compare GRADFILTERING with Random selection, Perplexity-based selection, Superfiltering (Li et al., 2024), and LESS-T (Xia et al., 2024). Superfiltering is the closest baseline, as it also uses a weak proxy and requires no target validation data, but relies on static difficulty scoring. LESS-T is target-aware and uses validation examples to construct gradient-alignment scores; we therefore treat it as an oracle-style reference rather than a strictly comparable validation-free baseline.

Target models and adaptation. We use LoRA fine-tuning (Hu et al., 2022) as the adaptation protocol, following its common use in efficient instruction tuning and enabling controlled comparison across target models under a fixed compute budget. We evaluate three dense targets that cover both generation-level and family-level transfer: LLaMA-2-7B, LLaMA-3-8B (Touvron et al., 2023; Grattafiori et al., 2024), and Qwen3-8B (Yang et al., 2025). LLaMA-2-7B and LLaMA-3-8B test transfer across different generations within the LLaMA family, while Qwen3-8B tests transfer to a distinct modern model family. We further evaluate Qwen3-30B-A3B (Yang et al., 2025) in Section 5.7 to probe transfer to a larger sparse MoE architecture.

Implementation details. Unless otherwise specified, GRADFILTERING uses GPT-2 as the proxy model with a five-member LoRA ensemble, where each member uses rank $r = 8$ and $\alpha = 16$. We train the proxy ensemble with learning rate 5×10^{-5} and compute G-SNR from the epoch $1 \rightarrow 2$ gradient window, selecting the top $\alpha \in \{5\%, 10\%, 15\%\}$ examples. For downstream evaluation, all target models are LoRA fine-tuned for three epochs with learning rate 2×10^{-5} under the same configuration across selection methods; only the selected subset differs. For proxy-choice analysis, we replace GPT-2 with Qwen3-0.6B while keeping the target fine-tuning and evaluation protocol unchanged.

Evaluation. We use pairwise LLM-as-a-judge evaluation, comparing each selected-subset model against the corresponding full-data model with the Vicuna prompt template (Zheng et al., 2023), as detailed in Appendix A.3. We report Pairwise Winning Scores (PWS), computed as $1 + (\text{Num}(\text{Win}) - \text{Num}(\text{Lose})) / \text{Num}(\text{All})$, averaged over GPT-5.5 and Qwen3.5-397B (Qwen

Method	LLaMA2-7B						LLaMA3-8B						Qwen3-8B					
	Alpaca			Tulu-3 52K			Alpaca			Tulu-3 52K			Alpaca			Tulu-3 52K		
	5%	10%	15%	5%	10%	15%	5%	10%	15%	5%	10%	15%	5%	10%	15%	5%	10%	15%
Random	0.97	0.93	0.92	0.88	0.85	0.96	0.91	0.95	0.94	0.93	1.01	0.92	0.97	0.99	1.00	0.94	0.98	0.95
Perplexity	0.78	0.74	0.83	0.88	0.92	0.89	0.93	0.65	0.72	0.93	0.78	0.79	1.06	0.92	0.79	0.97	0.84	0.80
Superfiltering	0.99	0.98	0.99	0.88	1.05	1.10	0.98	1.03	1.01	1.23	1.23	1.23	1.03	1.11	1.11	1.35	1.33	1.20
LESS-T	1.02	1.03	1.06	0.81	1.02	0.94	0.99	1.03	1.04	1.15	1.32	1.39	1.10	1.09	1.14	1.34	1.50	1.28
GRADFILTERING _{GPT2}	1.06	1.05	1.09	0.95	1.07	1.02	1.03	1.07	1.06	1.25	1.23	1.21	1.05	1.13	1.17	1.37	1.40	1.34

Table 1: Main results under LoRA fine-tuning. Columns group results for Alpaca and Tulu-3-52K across LLaMA-2-7B, LLaMA-3-8B, and Qwen3-8B using 5%, 10%, or 15% selected subsets. The Pairwise Winning Score (PWS) is computed as $1 + (\text{Num}(\text{Win}) - \text{Num}(\text{Lose})) / \text{Num}(\text{All})$; values above 1 indicate a preference over the corresponding full-data LoRA baseline. Perplexity, Superfiltering, and GRADFILTERING are validation-free, whereas LESS-T is target-aware and validation-dependent, and is therefore treated as an oracle reference. Bold marks the best score for each dataset, target, and budget.

Target Model	Full Data	Random	Perplexity	Superfiltering	LESS-T	GRADFILTERING
Qwen3-8B	0.6837	0.6836	0.6714	<u>0.6918</u>	0.6912	0.6935
LLaMA3-8B	0.6099	0.6065	0.5977	<u>0.6149</u>	<u>0.6195</u>	0.6196
LLaMA2-7B	0.4494	0.4404	0.4355	0.4476	0.4394	<u>0.4476</u>

Table 2: Objective benchmark performance after fine-tuning on 10% subsets of Tulu-3-52K. Values are averaged over ARC-Challenge, HellaSwag, MMLU, TruthfulQA, and GSM8K. Full Data uses the complete 52K training pool; all other methods use 10% subsets. Bold and underline mark the best and second-best averages for each target model.

Dataset	Budget	Avg. PWS	W/T/L	Overlap
Alpaca	5%	1.001	2/0/2	5.4%
	10%	0.992	1/2/1	11.3%
	15%	1.046	2/1/1	16.5%
Tulu-3-52K	5%	1.052	2/0/2	6.6%
	10%	0.990	1/1/2	11.7%
	15%	1.039	2/0/2	17.1%

Table 3: Effect of proxy choice under fixed GRADFILTERING hyperparameters. Avg. PWS compares GPT-2-selected and Qwen3-0.6B-selected subsets after target fine-tuning; values above 1 favor GPT-2. W/T/L is counted across the three dense targets and Qwen3-30B-A3B. Overlap is the shared selected-example fraction; the random-overlap expectation is approximately the selection budget.

Team, 2026); the latter serves as an open-source judge for reproducibility. We additionally evaluate ARC-Challenge, HellaSwag, MMLU, TruthfulQA, and GSM8K for objective capability assessment, and report ROUGE-1, ROUGE-2, and ROUGE-L for XSum-20K summarization. We further validate the judge preferences with five human annotators.

5.2 Main Results

Table 1 reports selected-vs-full PWS under LoRA fine-tuning. Overall, GRADFILTERING achieves

the best score in 13 of 18 settings and improves over Random selection in all settings, confirming that its selected subsets retain substantially more training utility than naive subsampling.

We additionally compare with the validation-free Perplexity baseline that selects examples with the highest per-token GPT-2 loss. Across the 18 settings, Perplexity obtains a mean PWS of 0.846, compared with 1.142 for GRADFILTERING, indicating that static proxy loss alone does not explain the gains from G-SNR.

The gains are most pronounced on Tulu-3-52K, the more heterogeneous and realistic candidate pool. On Qwen3-8B, GRADFILTERING obtains PWS scores of 1.37, 1.40, and 1.34 at 5%, 10%, and 15% budgets, respectively, outperforming Superfiltering at all three budgets. This suggests that uncertainty-aware gradient dynamics are particularly useful when the candidate pool contains diverse sources, uneven difficulty, and redundant examples.

Compared with Superfiltering, the closest validation-free baseline, GRADFILTERING outperforms or ties it in 16 of 18 settings. More notably, GRADFILTERING remains competitive with the target-aware LESS-T reference, matching or exceeding its numerical score in 14 of 18 settings

Method	Selection signal	Target/ dev	Proxy train	Grad.	GPU hrs	Wall time	VRAM
Superfiltering	static weak-proxy score	No	No	No	0.7	0.4 h	2 GB
LESS-T	target-aware gradient alignment	Yes	Yes/ warmup	Yes	7.7	1.1 h	6 GB
GRADFILTERING	gradient dynamics + ensemble uncertainty	No	Yes	Yes	12.4	2.5 h	2 GB

Table 4: Pre-fine-tuning selection cost and requirements. GPU hours denote aggregate NVIDIA H100 accelerator-hours, while wall time denotes elapsed runtime under each method’s parallel execution setting.

despite not using target validation examples. This result highlights the practical advantage of GRADFILTERING: it provides reusable, target-agnostic subsets while remaining competitive with a target-aware oracle-style reference in most settings.

5.3 Objective Benchmark Evaluation

To complement the PWS evaluation, we additionally assess whether the selected subsets preserve broader knowledge and reasoning capabilities rather than merely improving conversational preference. We evaluate the 10% Tulu-3-52K models on ARC-Challenge, HellaSwag, MMLU, TruthfulQA, and GSM8K, and report the average performance across the five benchmarks in Table 2.

Across all three target models, GRADFILTERING consistently outperforms Random on the five-benchmark average. More importantly, it achieves the best overall performance on Qwen3-8B and LLaMA3-8B, and second-best on LLaMA2-7B. These results indicate that the compact subsets selected by GRADFILTERING preserve strong performance on standard knowledge and reasoning benchmarks, rather than improving conversational preference at the expense of broader model capabilities.

5.4 Gradient Window and Ensemble Size Analysis

Figure 3 studies gradient-window length and ensemble size. The epoch $1 \rightarrow 2$ window performs best, consistent with Figure 1 showing that the largest proxy-trajectory changes occur early. Increasing the LoRA ensemble size improves mean PWS from 2 to 5 members, with only marginal gain at 7 members. We therefore use the $1 \rightarrow 2$ window and $M = 5$ ensemble as the default cost-performance setting.

5.5 Effect of Proxy Choice

We replace the default GPT-2 proxy with Qwen3-0.6B while keeping the GRADFILTERING pipeline

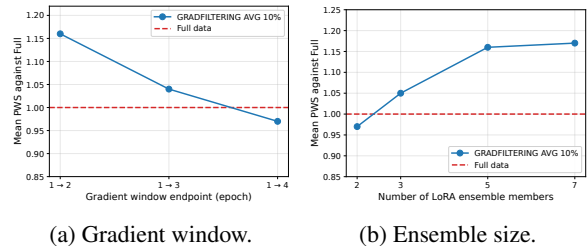


Figure 3: Practical sensitivity of GRADFILTERING. Mean PWS is averaged across LLaMA2-7B, LLaMA3-8B, Qwen3-8B and the Alpaca/Tulu-3-52K pools using 10% subsets. The default epoch $1 \rightarrow 2$ window performs best, while increasing the LoRA ensemble from 2 to 5 members improves selection quality and largely saturates at 7 members.

and selection hyperparameters fixed.

Table 3 shows that the two proxies select substantially different subsets. Yet downstream utility remains comparable: GPT-2 obtains a balanced 10/4/10 win/tie/loss record against Qwen3-0.6B, and Avg. PWS ranges from 0.990 to 1.052. This suggests that modern instruction-tuning pools contain many high-utility examples, so effective selection need not recover a unique subset. Thus, G-SNR acts as a proxy-conditioned learnability signal that can select different useful regions while filtering lower-utility or unstable examples. Given its lower cost and comparable performance, we use GPT-2 as the default proxy.

5.6 Computational Cost and Scalability

Table 4 compares selection-stage cost before downstream LoRA fine-tuning. Superfiltering is cheapest because it only requires forward scoring with a weak proxy. GRADFILTERING costs 12.4 aggregate GPU-hours and 2.5 hours of wall time, but avoids target validation data and uses less VRAM than LESS-T. For scalability, with a fixed ensemble size $M = 5$, GRADFILTERING scales linearly with candidate-pool size N , and the resulting ranking is computed once per candidate pool and can be reused across downstream targets, includ-

ing LLaMA-2-7B, LLaMA-3-8B, Qwen3-8B, and Qwen3-30B-A3B.

5.7 Generalization to MoE and Larger Architectures

Dataset	Budget	Superfiltering	Ours _{Qwen}	Ours _{GPT2}
Alpaca	5%	1.03	1.05	1.02
	10%	1.07	1.06	1.08
	15%	1.06	1.03	1.04
Tulu-3-52K	5%	1.29	1.33	1.34
	10%	1.30	1.32	1.36
	15%	1.21	1.21	1.26

Table 5: Generalization to the Qwen3-30B-A3B MoE target under LoRA fine-tuning. PWS is measured against the corresponding full-data baseline; values above 1 favor the selected-subset model. Ours_{Qwen} and Ours_{GPT2} use Qwen3-0.6B and GPT-2 proxies; bold marks the best score.

To probe architecture-level transfer, we evaluate Qwen3-30B-A3B, a larger sparse MoE target, using the same selected subsets. As shown in Table 5, GRADFILTERING remains competitive on Alpaca and is consistently strongest on Tulu-3-52K. With the default GPT-2 proxy, GRADFILTERING achieves PWS scores of 1.34, 1.36, and 1.26 on Tulu-3-52K at 5%, 10%, and 15% budgets, outperforming Superfiltering in all three cases. These results suggest that G-SNR-selected subsets transfer beyond dense 7B–8B targets and remain effective for larger MoE architectures.

5.8 Human Evaluation

We compare Qwen3-8B LoRA fine-tuning on the 10% GRADFILTERING-selected subset against the 100% baseline for Alpaca and Tulu-3-52K, using 100 prompts sampled from WizardLM (Xu et al., 2024) and Vicuna (Zheng et al., 2023). Annotators use the same pairwise criteria as the automatic evaluation: helpfulness, relevance, accuracy, and level of detail. The win/tie/loss counts are 43/22/35 on Alpaca and 52/7/41 on Tulu-3-52K, consistent with the LLM-judge trend and supporting that Table 1 reflects perceived quality.

5.9 Task-Specific Generalization

We further examine whether GRADFILTERING extends beyond general instruction tuning through task-specific summarization on XSum-20K.

As shown in Table 6, GRADFILTERING achieves the strongest ROUGE-1, ROUGE-2, and ROUGE-L scores among all 2K-example selection meth-

Method	ROUGE-1	ROUGE-2	ROUGE-L
Full XSum-20K (ref.)	36.51	14.83	29.06
Random	34.65	13.82	27.49
Perplexity	34.67	13.92	27.59
Superfiltering	34.17	13.42	27.34
LESS-T	34.72	13.69	27.69
GRADFILTERING	35.92	14.23	28.32

Table 6: Task-specific summarization on XSum-20K with Qwen3-8B. The full XSum-20K pool serves as the reference, while all selection methods use a 10% budget (2K examples). ROUGE scores are reported on the 0–100 scale. Bold marks the best result among the 2K subset-selection methods.

ods, retaining 97.26% of the full-pool performance on average. Unlike the heterogeneous instruction-tuning mixtures studied above, XSum is a curated single-task dataset with a relatively homogeneous distribution and fewer cross-source conflicts; the full candidate pool therefore remains highly informative, making substantial gains over the full-pool reference less likely. Nevertheless, GRADFILTERING consistently outperforms all subset-selection baselines across the three ROUGE metrics.

Importantly, GRADFILTERING applies the same validation-free selection criterion without task-specific modification, while LESS-T requires held-out target validation examples for target-aware selection. These results provide initial evidence that G-SNR can transfer beyond general instruction tuning to task-specific supervised fine-tuning.

6 Conclusion

We presented GRADFILTERING, an objective-agnostic and validation-free data selection method for instruction tuning. By training a lightweight proxy model with a LoRA ensemble, GRADFILTERING converts early proxy training dynamics into a G-SNR utility that combines gradient drop with late-stage ensemble disagreement. This yields reusable subsets without target validation examples or target-model gradient stores. Across three dense targets on Alpaca and Tulu-3-52K, GRADFILTERING-selected 5–15% subsets consistently improve over random selection, outperform or tie Superfiltering in 16 of 18 settings, and remain competitive with target-aware LESS-T. Objective benchmarks further support broad capability preservation, while Qwen3-30B-A3B and XSum-20K extend the evaluation to larger MoE architectures and task-specific summarization.

Limitations

First, although we show that small proxies can select subsets that transfer to larger instruction-tuned target models, our proxy study is limited to small proxy backbones. Due to computational constraints, we evaluate GPT-2 and Qwen3-0.6B under fixed selection hyperparameters, rather than performing proxy-specific tuning or exploring larger proxies. Larger or better-aligned proxies may provide stronger G-SNR estimates, but they would also increase selection cost; understanding this accuracy–cost trade-off is left to future work.

Additionally, GRADFILTERING currently uses gradient norms and ensemble variance, but does not explicitly model gradient direction. Prior work such as LESS (Xia et al., 2024) incorporates gradient direction, and direction-aware criteria remain an important direction for future work. However, incorporating direction typically requires choosing an anchor objective, validation set, or target gradient, which would move the method closer to target-aware gradient alignment and weaken its objective-agnostic, validation-free design. A key open question is how to exploit directional information while preserving the reusable proxy-only setting; we leave this exploration to future work. Finally, our evidence beyond instruction tuning is currently limited to XSum summarization; broader task-specific and domain-specific transfer remains to be established.

Acknowledgments

We used generative AI assistants for writing polishing and coding assistance. All AI-assisted outputs were independently reviewed by the authors.

References

- Ishika Agarwal, Krishnateja Killamsetty, Lucian Popa, and Marina Danilevsky. 2025. [Delift: Data efficient language model instruction fine-tuning](#). In *International Conference on Learning Representations*, volume 2025, pages 100488–100509.
- Oleksandr Balabanov and Hampus Linander. 2024. Uncertainty quantification in fine-tuned llms using lora ensembles. *arXiv preprint arXiv:2402.12264*.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.
- Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.
- Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxin Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, and 4 others. 2025. [Tulu 3: Pushing frontiers in open language model post-training](#). In *Second Conference on Language Modeling*.
- Dongyue Li, Ziniu Zhang, Lu Wang, and Hongyang R. Zhang. 2025. [Efficient ensemble for fine-tuning language models on multiple datasets](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25347–25364, Vienna, Austria. Association for Computational Linguistics.
- Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024. [Superfiltering: Weak-to-strong data filtering for fast instruction-tuning](#). In *Proceedings of the*

- 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 14255–14273, Bangkok, Thailand. Association for Computational Linguistics.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. [TruthfulQA: Measuring how models mimic human falsehoods](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.
- Dominik J. Mühlematter, Michelle Halbheer, Alexander Becker, Dominik Narnhofer, Helge Aasen, Konrad Schindler, and Mehmet Ozgur Turkoglu. 2026. [LoRA-ensemble: Efficient uncertainty modelling for self-attention networks](#). *Transactions on Machine Learning Research*.
- Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. [Don’t give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.
- Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34:20596–20607.
- Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930.
- Qwen Team. 2026. [Qwen3.5: Towards native multi-modal agents](#). Official Qwen blog post. Accessed 30 August 2026.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.
- Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. [Dataset cartography: Mapping and diagnosing datasets with training dynamics](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.
- Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. 2019. [An empirical study of example forgetting during deep neural network learning](#). In *International Conference on Learning Representations*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *Preprint*, arXiv:2307.09288.
- Mengzhou Xia, Sathika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org.
- Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. [Wizardlm: Empowering large pre-trained language models to follow complex instructions](#). In *International Conference on Learning Representations*, volume 2024, pages 30745–30766.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can a machine really finish your sentence?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-judge with MT-bench and chatbot arena](#). In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

A Appendix

A.1 Dataset Construction

Tulu-3-52K. We construct Tulu-3-52K from the training split of the Tulu-3 SFT mixture (Lambert et al., 2025). For each example, non-empty user and assistant turns are concatenated into the instruction and response, respectively; examples lacking either are discarded. We then group valid examples by source and sample 52,000 examples proportionally to the source distribution, with rounding adjusted to obtain the exact target size. Sampling is performed without replacement using a NumPy random generator with seed 42, followed by a global shuffle. The resulting pool contains examples from 19 sources and approximately preserves the original mixture composition while matching the scale of Alpaca. No additional length filtering, deduplication, or source exclusion is applied.

XSum-20K. For task-specific summarization, we construct XSum-20K by uniformly sampling 20,000 examples without replacement from the 204,045-example XSum training split (Narayan et al., 2018), using a NumPy random generator with seed 42. Each example is formatted with the fixed instruction “*Summarize the following news article in one sentence.*”, using the document as input and the reference summary as output. No additional filtering or truncation is applied when constructing the pool. All subset-selection methods use a 10% budget of XSum-20K (2,000 examples), and evaluation is performed on the complete official test split of 11,334 examples.

A.2 Training and Selection Details

Proxy scoring. Unless otherwise specified, GRADFILTERING uses GPT-2 with a five-member LoRA ensemble ($r = 8$, $\alpha = 16$, dropout 0.05). Each member is trained with AdamW using learning rate 5×10^{-5} , weight decay 0.01, a linear learning-rate schedule with warmup ratio 0.03. The training objective is next-token cross-entropy over response tokens, with prompt and padding tokens masked. Per-example LoRA-gradient norms are collected at each epoch, and G-SNR is computed from the epoch 1 $\rightarrow$ 2 window. Examples are ranked globally and the top 5%, 10%, or 15% are selected.

Proxy sensitivity. For the proxy-choice analysis, GPT-2 is replaced by Qwen3-0.6B while all shared

training and selection hyperparameters are kept fixed.

Baselines. All methods select from the same candidate pools under matched selection budgets. For the Perplexity baseline, we score each example once using pretrained GPT-2 without fine-tuning and select examples with the highest token-averaged cross-entropy.

A.3 Prompt for Evaluation

We follow the LLM-as-a-judge protocol described in the main text, using a fixed, symmetric prompt to compare responses from two candidate models on the same instruction.

For each evaluation example, we use GPT-5.5 and Qwen3.5-397B as judge models, consistent with Section 5.1. We adopt the prompt template from Vicuna (Zheng et al., 2023), with only minimal renaming of model identifiers.

Prompt for Performance Evaluation

System Prompt

You are a helpful and precise assistant for checking the quality of the answer.

User Prompt

[Question]

Question

[The Start of Assistant 1’s Answer]

Answer 1

[The End of Assistant 1’s Answer]

[The Start of Assistant 2’s Answer]

Answer 2

[The End of Assistant 2’s Answer]

We would like to request your feedback on the performance of two AI assistants in response to the user question displayed above. Please rate the helpfulness, relevance, accuracy, and level of detail of their responses. Each assistant receives an overall score on a scale of 1 to 10, where a higher score indicates better overall performance. Please first output a single line containing only two values indicating the scores for Assistant 1 and Assistant 2, respectively. The two scores are separated by a space. In the subsequent line, please provide a comprehensive explanation of your evaluation, avoiding any potential bias and ensuring that the order in which the responses were presented does not affect your judgment.

Table 7: Prompt template for pairwise LLM-as-a-judge evaluation. The judge assigns 1–10 scores to two anonymized responses and explains the judgment; response order is shuffled to reduce ordering bias.