

FLINT: Influence-Guided Active Learning Framework for LoRA via Curvature-Aware Data Selection and Fine-Tuning

Zixuan Li^{*†1,4} Shenglan Guo^{*‡2,4} Zhigen Li^{*3,4} Yanmeng Wang⁴
Ning Cheng⁴ Shaojun Wang⁴ Deyi Xiong^{†3}

¹The Hong Kong University of Science and Technology

²National University of Defense Technology

³Tianjin University

⁴Ping An Technology

zixuan.li@connect.ust.hk, guoshenglan24@nudt.edu.cn

{zhigenli, dyxiong}@tju.edu.cn

{wangyanmeng219, chengning211, wangshaojun851}@pingan.com.cn

Abstract

In instruction fine-tuning of large language models (LLMs), identifying beneficial samples from large-scale datasets is pivotal to maximizing model performance. Traditional methods rely on static semantic heuristics, ignoring training dynamics and failing to accurately assess sample contributions. The influence function offers a theoretically grounded approach but incurs prohibitive computational cost due to Hessian inversion. Existing influence function-based methods (IF methods) resort to simplifying or omitting the second-order curvature term in the Hessian, leading to distorted contribution estimates on the non-convex parameter manifold. Going beyond such approximations, we propose FLINT: an Active Learning Framework for LoRA via **I**nfluence-Guided Data Selection and Fine-Tuning. By constraining influence computation to the LoRA parameter space, FLINT enables precise second-order curvature computation—the first method to achieve this on consumer-grade GPUs, accurately characterizing optimization trajectories and overcoming the limitations of prior approaches. Furthermore, FLINT utilizes Singular Value Decomposition (SVD) as a cold-start mechanism to address initialization instability, alongside predictive entropy pre-filtering and Kronecker-Factored Approximate Curvature (K-FAC)-based decoupling to maximize selection efficiency. Employing an iterative multi-round active learning mechanism, FLINT dynamically filters the most valuable data as the LoRA manifold evolves. Evaluated across complex reasoning benchmarks (GSM8K, BBH, StrategyQA), FLINT achieves a new state-of-the-art among IF methods, significantly accelerating training while consistently outperforming the full-dataset LoRA baseline with less than 20% of the training data, and establishing a principled approach to the non-convex optimization of underlying LLM parameters.

1 Introduction

Large language models (LLMs) demonstrate remarkable generalization across diverse tasks (Chung et al., 2024; Brown et al., 2020; Longpre et al., 2023; Zhang et al., 2026). Through supervised instruction tuning on large-scale datasets, they acquire enhanced instruction-following, open-domain dialogue, and complex reasoning capabilities (Ouyang et al., 2022; Zellers et al., 2019; Subramaniam et al., 2025; Gunasekar et al., 2023).

^{*}Equal contribution.

[†]Corresponding author: dyxiong@tju.edu.cn.

[‡]Work done while interning at Ping An Technology.

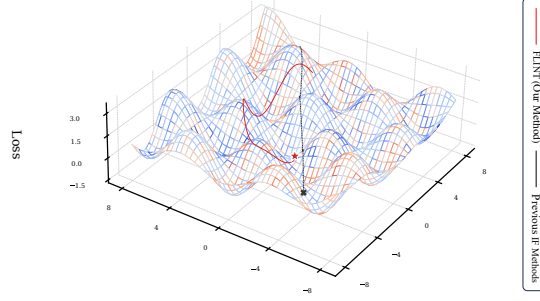


Figure 1: Comparison of optimization trajectories on the loss landscape. Existing methods relying on first-order or curvature-linearized approximations tend to converge to sharp minima, while FLINT preserves second-order curvature information to guide optimization toward flatter, more generalizable minima.

However, as model and data scales grow, full-parameter fine-tuning becomes computationally prohibitive. Furthermore, practical applications often demand improving specific capabilities (e.g., mathematical reasoning) rather than uniform gains across all tasks. This motivates *targeted instruction tuning*: selecting the most valuable data subset to optimize specific capabilities under limited resources (Zhou et al., 2023).

Existing data selection methods can be broadly categorized into three paradigms: quality-based, diversity-based, and importance-based approaches (Qin et al., 2024; Zhang et al., 2025). Quality-based methods assess individual samples based on intrinsic characteristics but rely on static features. Diversity-based methods ensure broad coverage but often overlook sample-wise contribution. Importance-based methods attempt to quantify each sample’s contribution to model performance. However, they often discard curvature information, ignore higher-order interactions among samples, or incur prohibitive computational costs, limiting their effectiveness in the non-convex parameter manifold of LLMs.

To address these limitations, we propose FLINT, an Active Learning Framework for LoRA via **I**nfluence-Guided Data Selection and Fine-Tuning, which retains the influence function formulation by leveraging the LoRA parameter manifold to incorporate curvature. As illustrated in Figure 1, prior first-order or curvature-linearized approximations (collectively denoted as Previous IF Methods) discard second-order curvature information, often driving the model toward sharp minima that generalize poorly (Foret et al., 2021; Keskar et al., 2017; Ye et al., 2025). In contrast, FLINT explicitly captures this geometry, guiding optimization toward flatter, more generalizable minima while remaining computationally tractable.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 details the FLINT framework. Section 4 presents experimental settings and results. Section 5 concludes the paper, followed by limitations, broader impact discussions, and acknowledgments before the references. Detailed derivations and additional results are provided in the appendices.

Our Contributions

1. We propose FLINT: an Active Learning Framework for LoRA via **I**nfluence-Guided Data Selection and Fine-Tuning. On complex reasoning benchmarks including GSM8K, BBH, and StrategyQA, FLINT achieves superior performance over full-dataset fine-tuning with less than 20% training data, significantly surpassing state-of-the-art data selection baselines.
2. We design an iterative curvature-aware influence estimation mechanism for LoRA as FLINT’s core, using the full influence function formulation. Preserving second-order curvature on the low-dimensional LoRA manifold, we achieve accurate sample contribution estimation efficiently on commodity GPUs.

3. We analyze shifts in predictive entropy and influence score distributions across training rounds, revealing geometric evolution patterns within the LoRA parameter space, and establish a principled framework for understanding non-convex optimization trajectories in LLM alignment.

2 Related Work

Data selection for efficient LLM instruction tuning (Ma et al., 2025; Shumailov et al., 2023) broadly falls into quality-, diversity-, and importance-based paradigms (Qin et al., 2024; Zhang et al., 2025). **Quality-based methods** evaluate individual samples via intrinsic characteristics (e.g., correctness, complexity) (Wang et al., 2022; Chen et al., 2023; Li et al., 2024) or surface feature similarity to target representations (Xie et al., 2023; Hanawa et al., 2021). However, relying on such static heuristics overlooks the actual impact of training data on dynamic model parameter updates.

Diversity-based methods focus on ensuring broad coverage of semantic and task spaces to prevent overfitting and improve generalization (Du et al., 2023). Recent work has explored various diversity measurement strategies, including representation-based clustering (Sorscher et al., 2022), n-gram feature matching (Xie et al., 2023), and sample-level novelty scoring (Yang et al., 2025). While effective for preventing model collapse, diversity-based selection often overlooks individual sample contribution to target capabilities and may retain low-quality data that happens to be diverse (Iverson et al., 2025).

Importance-based methods attempt to quantify each sample’s contribution to model performance, often leveraging influence functions or gradient information (Koh & Liang, 2017; Han et al., 2023). Recent work has adapted influence estimation to large language models through various approximations, including low-dimensional gradient projection (Xia et al., 2024; Park et al., 2023; Toneva et al., 2018; Sener & Savarese, 2018; Paul et al., 2021), closed-form Fisher approximations (Kwon et al., 2024; Mindermann et al., 2022), and iterative inverse-Hessian updates (Agarwal et al., 2017; Grosse et al., 2023). Other approaches have further addressed limitations such as inter-sample interactions (Wei et al., 2025), task-level biases (Dai et al., 2025), graph-based dependencies (Zhao et al., 2025b), and iterative in-loop selection (Lin et al., 2025).

Despite these advances, methods explicitly designed for efficient fine-tuning often resort to first-order or curvature-linearized approximations of the influence function to remain tractable, which inevitably distorts contribution estimates on the non-convex parameter manifold. Motivated by the limitations of these approaches, we propose FLINT, which seamlessly integrates the influence formulation adaptively into the LoRA subspace. This design uniquely preserves essential second-order information, ensuring both theoretical fidelity and empirical efficiency without the architectural compromises of prior works.

3 Method: FLINT

As illustrated in Figure 2, FLINT follows an iterative two-stage selection pipeline that combines entropy-guided candidate construction with curvature-aware influence estimation on the LoRA manifold, adaptively selecting the optimal training subset at each epoch according to the evolving manifold of the LoRA parameter space.

3.1 Data Value Categorization

To guide the dynamic selection procedure, we partition the set of training samples $\mathcal{D}_{\text{train}}$ into three categories, based on their contribution to model generalization:

- **Mastered Samples (MS):** Samples the model can already predict correctly. These provide limited benefit for further training.
- **Detrimental Samples (DS):** Samples with high entropy but positive influence scores. These may involve noisy labels or spurious patterns and should be filtered out.

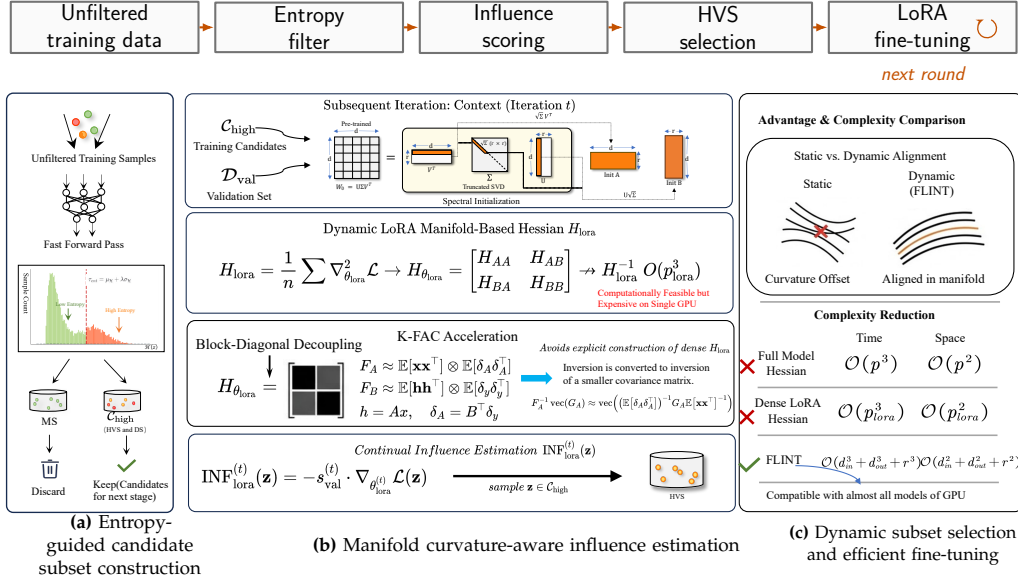


Figure 2: Overview of FLINT. The top row summarizes the iterative selection loop, where unfiltered training data are passed through entropy filtering, influence scoring, HVS selection, and LoRA fine-tuning before the next round. **(a)** Entropy-guided candidate subset construction filters out mastered samples (MS) and retains high-entropy candidates. **(b)** LoRA-manifold curvature-aware influence estimation scores each candidate using validation gradients and an efficient curvature approximation in the LoRA parameter space. **(c)** Dynamic subset selection distinguishes high-value samples (HVS) from detrimental samples (DS), fine-tunes LoRA on the selected HVS subset, and reduces the time and memory cost compared with full-Hessian influence estimation.

- **High-Value Samples (HVS):** Samples with high prediction entropy and negative influence scores. These contain knowledge not yet acquired and are critical for improving target capabilities.

These categories are leveraged in our two-stage selection pipeline. First, entropy-aware sampling (Section 3.2) identifies high-uncertainty samples, effectively discarding most MS. Subsequently, curvature-aware influence estimation (Section 3.3) distinguishes HVS from DS, retaining only those with genuinely positive contributions.

3.2 Entropy-guided candidate subset construction

To identify and filter out MS that the model can already predict correctly, we adopt an entropy-aware dynamic sampling strategy. For the current model, we compute the predictive entropy for each candidate sample:

$$\mathcal{H}(z) = -\frac{1}{T} \sum_{t=1}^T \sum_c p(c|x_{<t}) \log p(c|x_{<t}), \quad (1)$$

where T is the sequence length and $p(c|x_{<t})$ is the predicted token probability. A dynamic threshold $\tau_{\text{ent}} = \mu_{\text{ent}} + \lambda_{\text{ent}} \sigma_{\text{ent}}$ is defined using the mean μ_{ent} and standard deviation σ_{ent} of entropy values across the dataset, with λ_{ent} (Entropy Scaling Factor) a tunable parameter. In our experiments, setting $\lambda_{\text{ent}} = 0.25$ reduces the candidate subset size by approximately 70%–80%. Samples with $\mathcal{H}(z) > \tau_{\text{ent}}$ are retained as the high-uncertainty candidate subset C_{high} , while low-entropy samples are discarded. This filtering step allows computational resources to focus on samples the model has not yet sufficiently learned.

3.3 Manifold curvature-aware influence estimation

After entropy-guided filtering, the high-uncertainty candidate subset $\mathcal{C}_{\text{high}}$ still contains HVS and DS. To distinguish them, we estimate each candidate’s contribution via influence functions, which capture second-order curvature through the inverse-Hessian–gradient product. Directly computing the full Hessian in the billion-parameter space is intractable. However, LoRA fine-tuning freezes the pre-trained backbone and confines optimization to the low-dimensional parameter vector $\theta_{\text{lora}} \in \mathbb{R}^{p_{\text{lora}}}$ ($p_{\text{lora}} \ll p$). This allows us to adopt the influence function formulation while making second-order curvature estimation tractable via Kronecker-factored approximations. The resulting curvature-aware influence score for a candidate z is:

$$\text{INF}_{\text{lora}}(z) = \sum_{z_{\text{val}} \in \mathcal{D}_{\text{val}}} -\nabla_{\theta_{\text{lora}}} L(z_{\text{val}}, \theta_{\text{lora}})^\top \mathbf{H}_{\theta_{\text{lora}}}^{-1} \nabla_{\theta_{\text{lora}}} L(z, \theta_{\text{lora}}), \quad (2)$$

where $\nabla_{\theta_{\text{lora}}} L(z_{\text{val}}, \theta_{\text{lora}})$ is the gradient of the validation loss with respect to model parameters, $\nabla_{\theta_{\text{lora}}} L(z, \theta_{\text{lora}})$ is the gradient of the training loss, and $\mathbf{H}_{\theta_{\text{lora}}} = \frac{1}{n} \sum_{i=1}^n \nabla_{\theta_{\text{lora}}}^2 L(z_i, \theta_{\text{lora}})$ is the Hessian of the empirical risk in the LoRA space. The full derivation and proof are provided in Appendix B. The iterative pipeline proceeds in three stages:

3.3.1 Round 0: Spectral Initialization for Cold Start

In the initial active learning round, the LoRA adapters are randomly initialized, making influence estimation unreliable. To overcome this cold-start limitation, we adopt a spectral initialization strategy inspired by PiSSA (Meng et al., 2024), as detailed in Appendix C.1.

For each linear layer with pre-trained weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we perform SVD: $\mathbf{W} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\top$, retain the top- r singular components, where r is the LoRA rank, and initialize the LoRA adapter matrices as: $\mathbf{A} = \mathbf{\Sigma}_r^{1/2} \mathbf{V}_r^\top / \sqrt{\lambda}$, $\mathbf{B} = \mathbf{U}_r \mathbf{\Sigma}_r^{1/2} / \sqrt{\lambda}$, where λ is the LoRA scaling factor. This constructs a proxy subspace that enables meaningful influence estimation before task-specific fine-tuning.

3.3.2 Subsequent Iterations: Continuous Influence Estimation on the LoRA Manifold

From the first iteration onward, the LoRA adapters have been fine-tuned using previously selected data. Their parameter vector θ_{lora} now encodes current task adaptation. We compute the Hessian matrix $\mathbf{H}_{\theta_{\text{lora}}}$ in this evolving manifold. The LoRA parameters are vectorized and concatenated; the Hessian has the following block structure:

$$\mathbf{H}_{\theta_{\text{lora}}} = \begin{bmatrix} \mathbf{H}_{\text{AA}} & \mathbf{H}_{\text{AB}} \\ \mathbf{H}_{\text{BA}} & \mathbf{H}_{\text{BB}} \end{bmatrix},$$

where d_{in} and d_{out} denote the input and output dimensions of the original linear layer. The blocks $\mathbf{H}_{\text{AA}} \in \mathbb{R}^{(rd_{\text{in}}) \times (rd_{\text{in}})}$, $\mathbf{H}_{\text{BB}} \in \mathbb{R}^{(d_{\text{out}}r) \times (d_{\text{out}}r)}$, and $\mathbf{H}_{\text{AB}} \in \mathbb{R}^{(rd_{\text{in}}) \times (d_{\text{out}}r)}$ capture second-order derivatives with respect to $\mathbf{A}$ and $\mathbf{B}$, with $\mathbf{H}_{\text{BA}} = \mathbf{H}_{\text{AB}}^\top$.

To keep computation tractable, we employ Kronecker-Factored Approximate Curvature (K-FAC) (Martens & Grosse, 2015) (see Appendix C.2 for the complete procedure), reducing computational complexity from $\mathcal{O}(p_{\text{lora}}^3)$ to $\mathcal{O}(d_{\text{in}}^3 + d_{\text{out}}^3 + r^3)$ and memory from $\mathcal{O}(p_{\text{lora}}^2)$ to $\mathcal{O}(d_{\text{in}}^2 + d_{\text{out}}^2 + r^2)$. For each training sample $z \in \mathcal{C}_{\text{high}}$, its curvature-aware influence score is then defined by preconditioning the validation gradient with the approximate inverse Hessian:

$$\text{INF}_{\text{lora}}(z) = -(\tilde{\mathbf{H}}_{\theta_{\text{lora}}}^{-1} \nabla_{\theta_{\text{lora}}} L_{\text{val}}(\hat{\theta}_{\text{lora}}))^\top \nabla_{\theta_{\text{lora}}} L(z), \quad (3)$$

where $\tilde{\mathbf{H}}_{\theta_{\text{lora}}}^{-1}$ denotes the K-FAC approximation. This score directly reflects the expected effect of including z on the validation loss.

3.3.3 Dynamic Subset Selection and Efficient Fine-Tuning

Based on counterfactual influence scores (where more negative indicates larger contribution), samples below an adaptive threshold $\tau_{\text{inf}} = \mu_{\text{inf}} - \lambda_{\text{inf}}\sigma_{\text{inf}}$ form the HVS, where μ_{inf} and σ_{inf} represent the mean and standard deviation of scores, and λ_{inf} is a tunable Influence Scaling Factor. To stabilize training, the raw selection ratio $R_{\text{sel}} = |\text{HVS}|/|\mathcal{D}_{\text{train}}|$ is elastically clipped into an effective ratio:

$$\tilde{R}_{\text{sel}} = \max(R_{\min}, \min(R_{\text{sel}}, R_{\max})).$$

Here, the strictness factor λ_{inf} and the empirical bounds $R_{\min}, R_{\max}$ are externally configurable hyperparameters. The top $\tilde{R}_{\text{sel}}|\mathcal{D}_{\text{train}}|$ lowest-scoring samples from the HVS constitute the final training subset $\mathcal{S}^{(t)}$. This dynamically focuses optimization on the most informative data; the model is then LoRA-finetuned on $\mathcal{S}^{(t)}$, updating its manifold to initialize the next active learning round.

4 Experiments

4.1 Experimental Setup

Training datasets. We evaluated FLINT on three widely used reasoning benchmarks: GSM8K (Cobbe et al., 2021) (mathematical reasoning), BBH (Suzgun et al., 2023) (27 diverse multi-step reasoning tasks), and StrategyQA (Geva et al., 2021) (open-domain yes/no questions requiring implicit reasoning). Dataset splits and preprocessing details are provided in Appendix F.

Evaluation datasets and metrics. On GSM8K, we reported exact-match accuracy on the final numerical answer. On BBH and StrategyQA, we reported answer accuracy. All generations used greedy decoding with a maximum of 256 new tokens. Detailed evaluation protocols for each dataset are described in Appendix A.3.

Models for data selection and training. We adopted LLaMA-3.2-3B-Instruct (AI at Meta, 2024) and LLaMA-2-7B-Chat (Touvron et al., 2023). All methods were fine-tuned using LoRA (Hu et al., 2022) with a unified configuration (rank $r = 8$, $\alpha = 32$, dropout 0.05, targeting all attention matrices). Complete hyperparameters are listed in Appendix A.3. Training ran for 3 epochs with effective batch size 16 using the AdamW optimizer at 1×10^{-4} learning rate. All experiments were performed on a single NVIDIA V100 GPU.

Baseline Methods. We compared FLINT with several baselines. The simplest baseline was Full, which fine-tuned on the entire training set (100% data). Random selected 20% of the training data for fine-tuning. We also compared to BM25 (Robertson & Zaragoza, 2009), a classical retrieval-based method that selects samples most similar to the validation set based on TF-IDF similarity. For influence-based approaches, we included DataInf (Kwon et al., 2024), which approximates sample contributions via the empirical Fisher inverse, and LESS (Xia et al., 2024), which constructs low-dimensional gradient representations for efficient influence estimation. All selection methods were allowed to select at most 20% of training data per active learning iteration. Detailed descriptions of these baseline methods are provided in Appendix D.

4.2 Main Results

Table 1 presents the main results. FLINT consistently outperforms all baselines across all settings using only 20% of the data, and notably surpasses Full-LoRA despite using less data. For example, on LLaMA-3.2-3B, FLINT achieves 73.0% on StrategyQA (vs. LESS 72.15% and DataInf 62.0%). On LLaMA-2-7B, FLINT reaches 37.9% on GSM8K (vs. LESS 34.92% and DataInf 26.5%).

Compared to previous influence-based methods (DataInf, LESS), FLINT explicitly models Riemannian curvature on the LoRA manifold, enabling more accurate sample contribution estimation. Semantic similarity-based BM25 performs poorly across tasks, confirming that

Method	LLaMA-3.2-3B			LLaMA-2-7B		
	GSM8K	BBH	StrategyQA	GSM8K	BBH	StrategyQA
FLINT (Ours)	0.7113	0.668	0.73	0.379	0.4906	0.7074
Full	0.6885	0.6414	0.69	0.3538	0.4562	0.6769
Random	0.5997	0.5379	0.6077	0.2564	0.3939	0.6332
DataInf	0.7062	0.4796	0.6200	0.2650	0.3308	0.6400
LESS	0.6878	0.5405	0.7215	0.3492	0.4467	0.6501
BM25	0.6744	0.445	0.63	0.2587	0.2601	0.64

Table 1: Accuracy on reasoning benchmarks under a 20% selection budget after 3 epochs.

Method	Select	Total	Speed	VRAM (GB)	Time Complexity
FLINT (Ours)	0.27h	1.67h	$5.73\times$	16.6	$\mathcal{O}(d_{\text{in}}^3 + d_{\text{out}}^3 + r^3) + \mathcal{O}(\mathcal{C}_{\text{high}} \cdot d_{\text{in}}^2)$
Full	–	4.90h	$1.95\times$	16.6	–
DataInf	3.27h	4.67h	$2.05\times$	~ 25.0	$\mathcal{O}(\mathcal{D} \cdot \mathcal{D}_{\text{val}} \cdot d)$
LESS	8.17h	9.57h	$1.0\times$	~ 30.0	$\mathcal{O}(\mathcal{D} \cdot \mathcal{D}_{\text{val}} \cdot d)$

Table 2: Runtime, memory, and complexity analysis of FLINT vs. baselines.

surface-level similarity is insufficient for identifying generalization-beneficial samples. Tables 4 and 5 further summarize studies on broader instruction-tuning scenarios, selection baselines, and validation-set construction, while Appendix E reports extended analyses, including LoRA rank sensitivity.

4.3 Efficiency Analysis

We compare time cost, GPU memory consumption, and theoretical complexity on BBH with LLaMA-2-7B, including both selection and end-to-end training. Results are shown in Table 2.

FLINT achieves substantial efficiency gains: the selection phase requires only 0.27 hours per iteration, compared to 3.27h for DataInf and 8.17h for LESS, and total end-to-end time (selection + fine-tuning) is 1.67h, achieving a $5.73\times$ speedup over LESS and $2.80\times$ over DataInf. Memory consumption during selection remains comparable to Full-LoRA fine-tuning, whereas DataInf and LESS require significantly more.

Theoretically, FLINT’s dominant cost is $\mathcal{O}(d_{\text{in}}^3 + d_{\text{out}}^3 + r^3)$ from K-FAC covariance inversion (a one-time cost per round), plus a per-sample cost of $\mathcal{O}(d_{\text{in}}^2)$ on the entropy-filtered candidate subset (20%–30% of the data). In contrast, DataInf and LESS require $\mathcal{O}(|\mathcal{D}_{\text{train}}| \cdot |\mathcal{D}_{\text{val}}| \cdot d)$ pairwise gradient similarity computations, where d is the projection dimension. This total complexity is significantly higher than FLINT’s per-sample cost, explaining the observed runtime gap.

4.4 Analysis of Entropy and Influence Score Distributions

We analyze the evolution of predictive entropy and influence scores during FLINT on BBH with LLaMA-2-7B, which is shown in Figure 3. Initially, entropy ranges 0.6–1.6 (mean 1.02) with many low-uncertainty samples. After two iterations, the distribution shifts rightward (mean 1.25) and narrows, indicating progressive filtering of mastered samples.

The influence score distribution initially spans -1.0 to 0.5, with dispersed negative and positive values. After the first round, the range sharply narrows to -0.06–0.04; by the second round, it further concentrates near zero (-0.04–0.04). This occurs because, in each round, samples with lower scores are preferentially selected for training, while those with higher scores are discarded. As a result, the remaining candidate pool becomes increasingly neu-

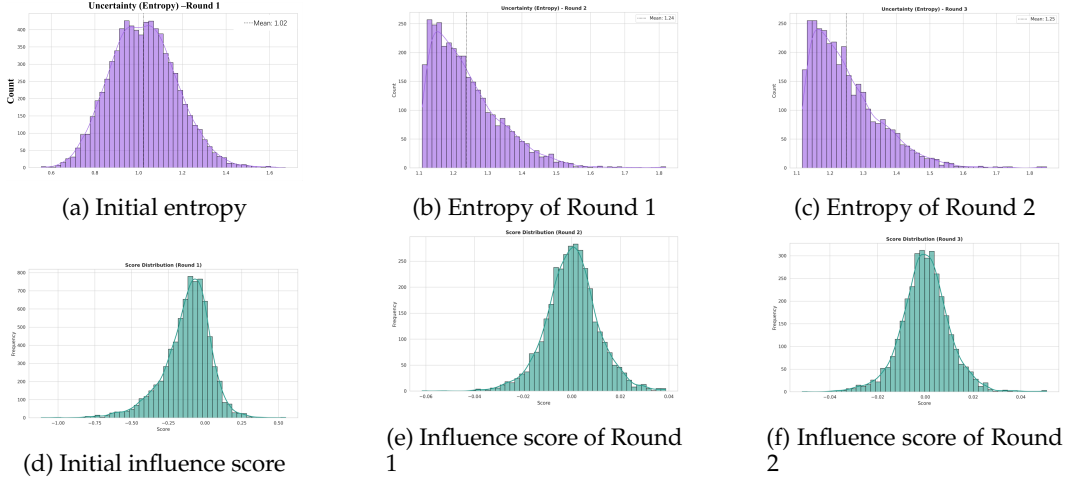


Figure 3: Evolution of entropy and influence scores of selected samples.

Variants	Modification	LLaMA-3.2-3B		Qwen2-7B-Instruct	
		StrategyQA	BBH	StrategyQA	BBH
FLINT (Full)	Complete framework with PiSSA	0.73	0.668	0.7598	0.6925
w/o PiSSA	Random Gaussian initialization	0.7006	0.649	0.7285	0.6710
w/o Entropy Filter	Influence on 100% of data	0.723	0.660	0.7501	0.6835
w/o INF _{lora}	Entropy-only selection	0.621	0.592	0.6515	0.6120
w/o Dynamic Iteration	Static single-shot selection	0.686	0.655	0.7140	0.6785

Table 3: Ablation experiment results.

tral, validating that curvature-aware influence estimation effectively distinguishes sample contributions and explaining why FLINT achieves superior performance with only 20% of the data.

4.5 Ablation Experiment

We conduct ablation studies on StrategyQA and BBH with LLaMA-3.2-3B and Qwen2-7B-Instruct (Yang et al., 2024) to evaluate the contribution of each component. The specific variants and their configurations are detailed in Table 3.

Ablation analysis confirms the necessity of each component. **Spectral initialization (PiSSA) is essential:** replacing it with random initialization degrades performance across both models (e.g., BBH on LLaMA-3.2-3B drops from 0.668 to 0.649; StrategyQA on Qwen2-7B from 0.7598 to 0.7285). **Entropy filtering reduces computational cost without sacrificing performance**—removing it yields comparable accuracy (e.g., 0.723 vs. 0.73 on StrategyQA with LLaMA-3.2-3B) but incurs 60%–80% more influence evaluations. **Curvature-aware influence estimation (LoRA-Inf) is the key driver of performance gains:** using only entropy sampling (w/o LoRA-Inf) causes sharp drops (on LLaMA-3.2-3B, StrategyQA falls from 0.73 to 0.621, BBH from 0.668 to 0.592). Finally, **dynamic iteration enables continuous optimization**, as static single-shot selection (w/o Dynamic Iteration) reduces performance noticeably (e.g., StrategyQA on Qwen2-7B from 0.7598 to 0.7140). Overall, the synergy of all components allows FLINT to surpass full-data fine-tuning.

Study	Setting	FLINT	Baseline 1	Baseline 2
General	Alpaca-Clean	82.4	Full-LoRA 80.5	Random-LoRA 74.2
General	ShareGPT-Vicuna	78.9	Full-LoRA 77.1	Random-LoRA 71.5
Baselines	GSM8K	71.10	LESS 68.80	SemDeDup 64.59 / K-Center 63.99
Baselines	BBH	66.80	LESS 54.10	SemDeDup 43.09 / K-Center 48.24

Table 4: Generalization and baseline comparisons on LLaMA-3.2-3B. The first two rows report results on general instruction-tuning datasets; the last two rows compare quality and diversity baselines under a 20% selection budget. All scores are reported in percentages.

Validation Strategy	GSM8K	StrategyQA	Stability
Biased / skewed distribution	63.50	65.20	High variance
Random sampling	67.80	69.50	Seed-dependent
Stratified validation (Ours)	71.13	73.00	Robust

Table 5: Validation-set construction robustness with LLaMA-3.2-3B.

4.6 Generalization and Robustness Results

We evaluate FLINT along three axes: transfer to broader instruction-tuning data, comparison with data-selection baselines, and robustness to validation-set construction. As shown in Table 4, FLINT outperforms both full-data LoRA and random 20% selection on Alpaca-Clean and ShareGPT-Vicuna, suggesting that the selected subsets remain effective beyond math and symbolic reasoning. Under the same 20% selection budget, FLINT also outperforms static quality and diversity baselines, including SemDeDup (Abbas et al., 2023) and K-Center Greedy (Sener & Savarese, 2018). These results indicate that training-dynamic influence is more predictive of downstream utility than semantic redundancy removal or geometric coverage alone.

Because FLINT estimates sample utility through validation gradients, its performance can depend on how the validation set is constructed. Table 5 compares three 50-sample validation strategies. A deliberately skewed validation set hurts both accuracy and stability, whereas distribution-aware stratified validation achieves the strongest and most stable results. Appendix E further analyzes LoRA rank and scaling factor sensitivity, showing that FLINT remains consistently ahead across practical adapter capacities.

5 Conclusion

In this paper, we introduced FLINT, an iterative active learning framework that projects the influence function onto the low-dimensional LoRA manifold, preserving second-order curvature information for sample contribution estimation. With less than 20% of the training data, FLINT achieves superior performance over full-data fine-tuning on complex reasoning benchmarks. These results show that training data value is better characterized through its effect on meaningful parameter updates along the optimization manifold than through superficial feature similarity. Future work may extend this manifold-aware perspective to adaptive optimization, where tracked LoRA curvature can inform learning rates, regularization, or other second-order control signals.

Limitations

While FLINT effectively captures manifold geometry, its selection intrinsically depends on the validation set’s alignment with target distributions. Unrepresentative validation data yields suboptimal subsets because influence functions project contributions onto valida-

tion gradients. However, this is practically manageable by curating a single high-quality validation set to consistently filter diverse, large-scale pools.

Broader Impacts and Ethics Statement

As large language models (LLMs) continue to scale, the computational and environmental costs of alignment and fine-tuning have become a pressing ethical concern. FLINT addresses these challenges while introducing specific considerations regarding data bias and model fairness.

Democratization and Environmental Sustainability By identifying highly informative samples, FLINT achieves strong fine-tuning performance using less than 20% of the original training data. This reduction lowers energy consumption and carbon footprint, aligning with Green AI. By making multi-round, curvature-aware active learning tractable on consumer-grade hardware (e.g., a single V100 GPU), FLINT also broadens access to advanced LLM alignment techniques for under-resourced academic institutions and smaller organizations.

Data Minimization and Privacy Compliance In regulated domains such as healthcare, jurisprudence, and fintech, strict data governance policies often restrict large-scale data transmission and centralized processing. FLINT aligns with *data minimization*: by discarding redundant or heavily mastered samples and retaining only a compact high-value subset, it reduces the exposure and movement of potentially sensitive information, thereby mitigating privacy risks and easing compliance with strict data protection regulations (e.g., GDPR, HIPAA).

Risks of Bias Amplification and Validation Dependency Influence-driven selection also introduces fairness risks. Because FLINT projects training gradients onto a predefined validation set, the selected subset depends on that validation distribution. If the validation set contains historical prejudices, demographic imbalances, or systematic biases, FLINT may assign high value to samples that reinforce those patterns, potentially amplifying bias during active learning.

Mitigation and Responsible Deployment Deploying FLINT in high-stakes environments should therefore be accompanied by rigorous fairness auditing. Practitioners must ensure that validation queries are diverse, inclusive, and audited for bias. Because efficient fine-tuning is a dual-use technology, dynamically fine-tuned models should also be subject to robust safety guardrails and comprehensive red-teaming before public-facing deployment.

Acknowledgments

Zhigen Li and Deyi Xiong were supported by the National Key Research and Development Program of China (Grant No. 2023YFE0116400). We thank the anonymous reviewers for their insightful comments.

References

- Amro Abbas, Kushal Tirumala, Daniel Simig, Surya Ganguli, and Ari S. Morcos. Semd-edup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023. URL <https://doi.org/10.48550/arXiv.2303.09540>.
- Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116):1–40, 2017. URL <https://www.jmlr.org/papers/v18/16-491.html>.
- AI at Meta. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL <https://doi.org/10.48550/arXiv.2407.21783>.
- L Douglas Baker and Andrew Kachites McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 96–103, 1998. URL <https://dl.acm.org/doi/pdf/10.1145/290941.290970>.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020. URL <https://doi.org/10.48550/arXiv.2005.14165>.
- Mayee Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! a data-driven skills framework for understanding and training language models. *Advances in Neural Information Processing Systems*, 36:36000–36040, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/70b8505ac79e3e131756f793cd80eb8d-Abstract-Conference.html.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024. URL <http://www.jmlr.org/papers/v25/23-0870.html>.
- 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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL <https://doi.org/10.48550/arXiv.2110.14168>.
- Qirun Dai, Dylan Zhang, Jiaqi W Ma, and Hao Peng. Improving influence-based instruction tuning data selection for balanced learning of diverse capabilities. *arXiv preprint arXiv:2501.12147*, 2025. URL <https://doi.org/10.48550/arXiv.2501.12147>.
- Qianlong Du, Chengqing Zong, and Jiajun Zhang. Mods: Model-oriented data selection for instruction tuning. *arXiv preprint arXiv:2311.15653*, 2023. URL <https://arxiv.org/html/2311.15653>.
- Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, 15(1):3133–3181, 2014. URL https://www.jmlr.org/papers/volume15/delgado14a/delgado14a.pdf?source=post_pag.
- Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. URL <https://doi.org/10.48550/arXiv.2010.01412>.

- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021. URL https://doi.org/10.1162/tac1_a_00370.
- Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiušė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023. URL <https://doi.org/10.48550/arXiv.2308.03296>.
- Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023. URL <https://doi.org/10.48550/arXiv.2306.11644>.
- Xiaochuang Han, Daniel Simig, Todor Mihaylov, Yulia Tsvetkov, Asli Celikyilmaz, and Tianlu Wang. Understanding in-context learning via supportive pretraining data. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12660–12673, 2023. URL <https://aclanthology.org/2023.acl-long.708/>.
- Kazuaki Hanawa, Sho Yokoi, Satoshi Hara, and Kentaro Inui. Evaluation of similarity-based explanations. In *International Conference on Learning Representations*, 2021. URL <https://doi.org/10.48550/arXiv.2006.04528>.
- 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. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nZevKeeFYf9>.
- 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. URL <https://doi.org/10.48550/arXiv.2503.01807>.
- Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017. URL <https://doi.org/10.48550/arXiv.1609.04836>.
- Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017. URL <http://proceedings.mlr.press/v70/koh17a?ref=https://githubhelp.com>.
- Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://arxiv.org/abs/2310.00902>.
- Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7602–7635, 2024. URL <https://aclanthology.org/2024.naacl-long.421/>.
- Xue Li and Paul Groth. How different is different? systematically identifying distribution shifts and their impacts in ner datasets. *Language Resources and Evaluation*, 59(2):1111–1150, 2025. URL <https://link.springer.com/article/10.1007/s10579-024-09754-8>.
- Xiaotian Lin, Yanlin Qi, Yizhang Zhu, Themis Palpanas, Chengliang Chai, Nan Tang, and Yuyu Luo. Lead: iterative data selection for efficient llm instruction tuning. *Proceedings of the VLDB Endowment*, 19(3):426–439, 2025. URL <https://doi.org/10.48550/arXiv.2505.07437>.

- Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and Adam Roberts. The flan collection: Designing data and methods for effective instruction tuning. In *International conference on machine learning*, pp. 22631–22648. PMLR, 2023. URL <https://proceedings.mlr.press/v202/longpre23a.html>.
- Youmi Ma, Sakae Mizuki, Kazuki Fujii, Taishi Nakamura, Masanari Ohi, Hinari Shimada, Taihei Shiotani, Koshiro Saito, Koki Maeda, Kakeru Hattori, Takumi Okamoto, Shigeki Ishida, Rio Yokota, Hiroya Takamura, and Naoaki Okazaki. Building instruction-tuning datasets from human-written instructions with open-weight large language models. *arXiv preprint arXiv:2503.23714*, 2025. URL <https://doi.org/10.48550/arXiv.2503.23714>.
- James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015. URL <https://proceedings.mlr.press/v37/martens15.html>.
- Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: principal singular values and singular vectors adaptation of large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 121038–121072, 2024. URL <https://dl.acm.org/doi/abs/10.5555/3737916.3741762>.
- Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltingen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pp. 15630–15649. PMLR, 2022. URL <https://proceedings.mlr.press/v162/mindermann22a.html>.
- 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. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022. URL <https://proceedings.neurips.cc/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract.html>.
- Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Mądry. Trak: attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 27074–27113, 2023. URL <https://doi.org/10.48550/arXiv.2303.14186>.
- Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34:20596–20607, 2021. URL <https://proceedings.neurips.cc/paper/2021/hash/ac56f8fe9eea3e4a365f29f0f1957c55-Abstract.html>.
- Yulei Qin, Yuncheng Yang, Pengcheng Guo, Gang Li, Hang Shao, Yuchen Shi, Zihan Xu, Yun Gu, Ke Li, and Xing Sun. Unleashing the power of data tsunamis: a comprehensive survey on data assessment and selection for instruction tuning of language models. *Transactions on Machine Learning Research*, 2024. URL <https://arxiv.org/abs/2408.02085>.
- Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 3(4):333–389, 2009. URL <https://dl.acm.org/doi/10.1561/15000000019>.
- Anisa Rula and Jennifer D’Souza. Procedural text mining with large language models. In *Proceedings of the 12th Knowledge Capture Conference 2023*, pp. 9–16, 2023. URL <https://dl.acm.org/doi/abs/10.1145/3587259.3627572>.
- Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *International Conference on Learning Representations*, 2018. URL <https://doi.org/10.48550/arXiv.1708.00489>.

- Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: training on generated data makes models forget. *arXiv e-prints*, pp. arXiv-2305, 2023. URL https://files.antimattercloud.nl/the_curse_of_recursion.pdf.
- Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/7b75da9b61eda40fa35453ee5d077df6-Abstract-Conference.html.
- Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://doi.org/10.48550/arXiv.2501.05707>.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 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. URL <https://aclanthology.org/2023.findings-acl.824>.
- Sahar Tahvili, Leo Hatvani, Enislay Ramentol, Rita Pimentel, Wasif Afzal, and Francisco Herrera. A novel methodology to classify test cases using natural language processing and imbalanced learning. *Engineering applications of artificial intelligence*, 95:103878, 2020. URL <https://www.sciencedirect.com/science/article/pii/S0952197620302244>.
- Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2018. URL <https://doi.org/10.48550/arXiv.1812.05159>.
- 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, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL <https://doi.org/10.48550/arXiv.2307.09288>.
- Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krma Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujana Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 5085–5109, 2022. URL <https://aclanthology.org/2022.emnlp-main.340/>.
- Jingyu Wei, Bo Liu, Tianjiao Wan, Baoyun Peng, Xingkong Ma, and Mengmeng Guo. Ji2s: Joint influence-aware instruction data selection for efficient fine-tuning. In *Proceedings of*

- the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 514–527, 2025. URL <https://aclanthology.org/2025.emnlp-main.26/>.
- Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 54104–54132, 2024. URL <https://doi.org/10.48550/arXiv.2402.04333>.
- 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. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6b9aa8f418bde2840d5f4ab7a02f663b-Abstract-Conference.html.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. URL <https://doi.org/10.48550/arXiv.2407.10671>.
- Yuming Yang, Yang Nan, Junjie Ye, Shihan Dou, Xiao Wang, Shuo Li, Huijie Lv, Tao Gui, Qi Zhang, and Xuan-Jing Huang. Measuring data diversity for instruction tuning: A systematic analysis and a reliable metric. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18530–18549, 2025. URL <https://aclanthology.org/2025.acl-long.908/>.
- Xichen Ye, Yifan Wu, Weizhong Zhang, Cheng Jin, and Yifan Chen. Towards robust influence functions with flat validation minima. In *International Conference on Machine Learning*, pp. 72091–72111. PMLR, 2025. URL <https://arxiv.org/pdf/2505.19097>.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019. URL <https://aclanthology.org/P19-1472/>.
- 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. URL <https://doi.org/10.1613/jair.1.17625>.
- Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Guoyin Wang, and Fei Wu. Instruction tuning for large language models: A survey. *ACM Computing Surveys*, 58(7):1–36, 2026. URL <https://dl.acm.org/doi/full/10.1145/3777411>.
- Chengshuai Zhao, Zhen Tan, Pingchuan Ma, Dawei Li, Bohan Jiang, Yancheng Wang, Yingzhen Yang, and Huan Liu. Is chain-of-thought reasoning of llms a mirage? a data distribution lens. *arXiv preprint arXiv:2508.01191*, 2025a. URL <https://arxiv.org/abs/2508.01191>.
- Yang Zhao, Li Du, Xiao Ding, Yangou Ouyang, Hepeng Wang, Kai Xiong, Jinglong Gao, Zhouhao Sun, Dongliang Xu, Qing Yang, Dongchen Li, Bing Qin, and Ting Liu. Beyond similarity: A gradient-based graph method for instruction tuning data selection. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24391–24404, 2025b. URL <https://aclanthology.org/2025.acl-long.1189/>.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ac662d74829e4407ce1d126477f4a03a-Abstract-Conference.html.

A Implementation Details and Reproducibility

Algorithm 1 FLINT: Iterative Active Instruction Tuning

Require: Pre-trained base weights W , Unfiltered training pool $\mathcal{D}_{\text{train}}$, Validation set $\mathcal{D}_{\text{val}}$
Require: Active rounds K , Empirical bounds $R_{\min}$, $R_{\max}$, Entropy Scaling Factor λ_{ent} , Influence Scaling Factor λ_{inf}

- 1: **Round 0 (Cold Start): Spectral Initialization**
- 2: **for** each pre-trained linear layer weight W **do**
- 3: Compute truncated SVD: $W \approx U_r \Sigma_r V_r^\top$ (retaining top rank- r components).
- 4: Initialize adapters: $A^{(0)} = \Sigma_r^{1/2} V_r^\top / \sqrt{\lambda}$ and $B^{(0)} = U_r \Sigma_r^{1/2} / \sqrt{\lambda}$, where $\lambda = \alpha/r$ is the LoRA scaling factor.
- 5: **end for**
- 6: Aggregate initialized parameters into $\theta_{\text{lora}}^{(0)} = \{A^{(0)}, B^{(0)}\}$.
- 7: **for** $k = 1$ **to** K **do**
- 8: **Stage 1: Entropy-Guided Candidate Construction**
- 9: Compute prediction entropy $\mathcal{H}(z)$ for all $z \in \mathcal{D}_{\text{train}}$ via Eq. (1).
- 10: Compute mean μ_{ent} and standard deviation σ_{ent} of all $\mathcal{H}(z)$.
- 11: Calculate adaptive entropy threshold $\tau_{\text{ent}} = \mu_{\text{ent}} + \lambda_{\text{ent}} \sigma_{\text{ent}}$.
- 12: Form candidate pool $\mathcal{C}_{\text{high}} = \{z \in \mathcal{D}_{\text{train}} \mid \mathcal{H}(z) > \tau_{\text{ent}}\}$.
- 13: **Stage 2: Continuous Influence Estimation on the LoRA Manifold**
- 14: Compute aggregated validation gradient $s_{\text{val}} = \nabla_{\theta_{\text{lora}}} L_{\text{val}}(\hat{\theta}_{\text{lora}}^{(k-1)})$.
- 15: // *K-FAC Block-Diagonal Hessian Approximation & Inversion*
- 16: **for** each target linear layer with LoRA adapters A and B **do**
- 17: Accumulate layer statistics: activation covariance $\Sigma_x = \mathbb{E}[xx^\top]$ and gradient covariance $\Sigma_\delta = \mathbb{E}[\delta\delta^\top]$.
- 18: Approximate diagonal blocks via Kronecker products:
 $H_{AA} \approx \mathbb{E}[(B^\top \delta)(B^\top \delta)^\top] \otimes \Sigma_x$
 $H_{BB} \approx \Sigma_\delta \otimes \mathbb{E}[(Ax)(Ax)^\top]$
- 19: Compute block inverses H_{AA}^{-1}, H_{BB}^{-1} efficiently using $(P \otimes Q)^{-1} = P^{-1} \otimes Q^{-1}$.
- 20: **end for**
- 21: Assemble global block-diagonal inverse $\tilde{H}_{\theta_{\text{lora}}}^{-1}$ from all layer-wise inverses.
- 22: **for** each candidate $z \in \mathcal{C}_{\text{high}}$ **do**
- 23: Compute preconditioned natural gradient $\tilde{g}(z) = \tilde{H}_{\theta_{\text{lora}}}^{-1} \nabla_{\theta_{\text{lora}}^{(k-1)}} L(z)$.
- 24: Compute curvature-aware influence score: $\text{INF}_{\text{lora}}^{(k)}(z) = -s_{\text{val}}^\top \cdot \tilde{g}(z)$.
- 25: **end for**
- 26: **Stage 3: Dynamic Subset Selection and Efficient Fine-Tuning**
- 27: Compute mean μ_{inf} and standard deviation σ_{inf} of $\text{INF}_{\text{lora}}^{(k)}(z)$ across $\mathcal{C}_{\text{high}}$.
- 28: Calculate adaptive influence threshold $\tau_{\text{inf}} = \mu_{\text{inf}} - \lambda_{\text{inf}} \sigma_{\text{inf}}$.
- 29: Form High-Value Subset (HVS): retain samples where $\text{INF}_{\text{lora}}^{(k)}(z) < \tau_{\text{inf}}$.
- 30: Compute raw selection ratio $R_{\text{sel}} = |\text{HVS}|/|\mathcal{D}_{\text{train}}|$.
- 31: Apply elastic clipping: $\tilde{R}_{\text{sel}} = \max(R_{\min}, \min(R_{\text{sel}}, R_{\max}))$.
- 32: Form final training subset $\mathcal{S}^{(k)}$ by selecting the top $\tilde{R}_{\text{sel}}|\mathcal{D}_{\text{train}}|$ samples from HVS (sorted by ascending influence scores).
- 33: Fine-tune model on $\mathcal{S}^{(k)}$ using LoRA to update parameters from $\theta_{\text{lora}}^{(k-1)}$ to $\theta_{\text{lora}}^{(k)}$.
- 34: **end for**
- 35: **return** Fine-tuned LoRA parameters $\theta_{\text{lora}}^{(K)}$.

To ensure the complete reproducibility of our FLINT framework, this section elaborates on the full algorithmic pipeline, critical hyperparameters, and standard evaluation protocols.

A.1 FLINT Algorithm Pipeline

Algorithm 1 formalizes the complete two-stage iterative active learning pipeline of FLINT, detailing the transition from Spectral Initialization (Cold Start) to the continuous curvature-aware influence estimation on the evolving LoRA manifold.

To bridge the gap between theoretical manifold projection and engineering reality, the algorithm dynamically tracks the parameter manifold without requiring explicit dense matrix constructions. The specific mathematical formulations and engineering techniques that make this efficient pipeline possible on consumer-grade GPUs—namely, the SVD-based cold start and the K-FAC block-diagonal approximation—are fully derived and discussed in Appendix C.2.

A.2 Hyperparameter Configurations

Our experiments are implemented using PyTorch and the Hugging Face transformers library. By constraining calculations to the low-rank parameter space, we efficiently manage memory footprint during dynamic Hessian estimations without relying on heavy tensor partitioning frameworks. The specific hyperparameters for fine-tuning and the FLINT engine are listed in Table 6, strictly adhering to the settings described in Section 4.1.

Configuration	Value
<i>Model & PEFT Settings</i>	
LoRA Rank (r)	8
LoRA Alpha (α)	32
LoRA Dropout	0.05
Target Modules	q, k, v, o, gate, up, down_proj
Adapter Initialization	PiSSA (SVD Spectral Init)
<i>Training Hyperparameters</i>	
Optimizer	AdamW
Learning Rate	1×10^{-4}
Training Epochs per Round	3
Per-device Batch Size	2
Gradient Accumulation Steps	8
Effective Batch Size	16
Max Sequence Length	512
<i>FLINT Specifics</i>	
Active Iteration Rounds (K)	3
Entropy Scaling Factor (λ_{ent})	0.25
Influence Scaling Factor (λ_{inf})	0.25
Max Selection Ratio (R_{max})	0.20 (20%)
Min Selection Ratio (R_{min})	0.00 (0%)

Table 6: Hyperparameter configurations for FLINT.

A.3 Evaluation Protocol

To ensure the performance gains stem exclusively from data quality rather than prompt engineering, we strictly employ a zero-shot Chain-of-Thought (CoT) protocol across all datasets without providing in-context demonstrations. Accuracy is calculated via strict regular expression (regex) matching on the generated tokens. For BBH, which comprises diverse task formats (e.g., multiple-choice, yes/no, and matching tasks), we extract the final answer according to the evaluation protocol of each sub-task and compute accuracy accordingly. For StrategyQA, we determine correctness by checking whether the generated response contains “yes” or “no”. All generations are produced using greedy decoding with a maximum of 256 new tokens. The prompt templates used for each dataset are as follows:

- **GSM8K**: Prompted with “[Question]\nLet’s think step by step.” The final numerical answer is extracted by locating the “####” separator or parsing the last valid digit sequence.
- **BBH**: Prompted with “[Question]\nLet’s think step by step.” Evaluated by matching the generated output against the target multiple-choice option (e.g., “(A)”).
- **StrategyQA**: Prompted with “[Question]\nAnswer the question with Yes or No. Let’s think step by step.” The trailing 100 characters of the response are parsed for explicit boolean targets.

B Influence Function

B.1 Basic Principles of Influence Functions

The influence function is a classical technique from robust statistics for quantifying the effect of individual training samples on model predictions. Let the training dataset be $\mathcal{D}_{train} = \{z_i\}_{i=1}^n$, with model parameter vector θ , loss function L , and optimal parameters $\hat{\theta}$ obtained via empirical risk minimization. Influence functions answer the following counterfactual question: How would the loss at a test point z_{test} change if the weight of a training point z were infinitesimally perturbed? The answer is given by:

$$\mathcal{I}_{up,loss}(z, z_{test}) = -\nabla_{\theta} L(z_{test}, \hat{\theta})^{\top} \mathbf{H}_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta}), \quad (\text{B.1})$$

where $\nabla_{\theta} L(z_{test}, \hat{\theta})$ is the gradient of the test loss with respect to model parameters, $\nabla_{\theta} L(z, \hat{\theta})$ is the gradient of the training loss, and $\mathbf{H}_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^n \nabla_{\theta}^2 L(z_i, \hat{\theta})$ is the Hessian of the empirical risk. A positive value indicates that increasing the weight of z leads to higher test loss (detrimental effect), while a negative value implies a beneficial contribution.

B.2 From a Single Test Point to the Validation Set

To quantify the impact of each training sample on overall generalization, we replace the individual test point with a representative validation set $\mathcal{D}_{val}$ and define the influence score:

$$\text{INF}(z) = \sum_{z_{val} \in \mathcal{D}_{val}} \mathcal{I}_{up,loss}(z, z_{val}). \quad (\text{B.2})$$

This score reflects the overall contribution of z to reducing validation loss. More negative values indicate greater positive impact on generalization.

B.3 Derivation of LoRA Influence Score

This section presents the detailed derivation of the LoRA influence score $\text{INF}_{\text{lor}}(z)$ used in Section 3.3 of the main text. In the LoRA fine-tuning paradigm, the pre-trained weight matrix $\mathbf{W}_{\text{base}}$ is frozen, and only the low-rank adapter matrices $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$ are optimized. Denote all trainable parameters as $\theta_{\text{lor}} = [\text{vec}(\mathbf{A}); \text{vec}(\mathbf{B})] \in \mathbb{R}^{p_{\text{lor}}}$, where $p_{\text{lor}} = rd_{\text{in}} + d_{\text{out}}r$.

Following the notation and assumptions of Appendix B.1, we consider the empirical risk minimizer $\hat{\theta}_{\text{lor}}$ in this low-dimensional parameter space, with the corresponding empirical Hessian $\mathbf{H}_{\text{lor}}$ defined analogously. The classic influence function result then applies directly to θ_{lor} :

$$\left. \frac{d\hat{\theta}_{\text{lor}, \epsilon, z}}{d\epsilon} \right|_{\epsilon=0} = -\mathbf{H}_{\text{lor}}^{-1} \nabla_{\theta_{\text{lor}}} L(z, \hat{\theta}_{\text{lor}}). \quad (\text{B.3})$$

To measure the impact of a sample z on the overall loss of the validation set $\mathcal{D}_{\text{val}}$, we define the total validation loss as

$$L_{\text{val}}(\theta_{\text{loa}}) = \sum_{z_{\text{val}} \in \mathcal{D}_{\text{val}}} L(z_{\text{val}}, \theta_{\text{loa}}).$$

Substituting the perturbed parameter $\hat{\theta}_{\text{loa}, \epsilon, z}$ and taking the derivative, the LoRA influence score can be derived via the chain rule and Equation (B.3):

$$\begin{aligned} \left. \frac{dL_{\text{val}}(\hat{\theta}_{\text{loa}, \epsilon, z})}{d\epsilon} \right|_{\epsilon=0} &= \nabla_{\theta_{\text{loa}}} L_{\text{val}}(\hat{\theta}_{\text{loa}})^\top \left. \frac{d\hat{\theta}_{\text{loa}, \epsilon, z}}{d\epsilon} \right|_{\epsilon=0} \\ &= -(\nabla_{\theta_{\text{loa}}} L_{\text{val}}(\hat{\theta}_{\text{loa}}))^\top \mathbf{H}_{\text{loa}}^{-1} \nabla_{\theta_{\text{loa}}} L(z, \hat{\theta}_{\text{loa}}), \end{aligned} \quad (\text{B.4})$$

where $\nabla_{\theta_{\text{loa}}} L_{\text{val}}(\hat{\theta}_{\text{loa}}) = \sum_{z_{\text{val}}} \nabla_{\theta_{\text{loa}}} L(z_{\text{val}}, \hat{\theta}_{\text{loa}})$. This equation is exactly the LoRA influence score defined in the main text.

C Acceleration Methods

This section provides a detailed exposition of two core technical components underlying FLINT’s efficiency and effectiveness: (1) the SVD-based spectral initialization that addresses the cold-start challenge in active learning, and (2) the Kronecker-Factored Approximate Curvature (K-FAC) method that enables tractable Hessian inversion on the LoRA manifold.

C.1 SVD-Based Spectral Initialization for Cold Start

In the initial active learning round (Round 0), the LoRA adapters are typically initialized with random Gaussian noise. However, this random initialization can lead to unstable influence estimates because the parameter manifold has not yet been aligned with the target task. To overcome this limitation, we adopt a spectral initialization strategy inspired by PiSSA (Meng et al., 2024), which leverages the singular value decomposition (SVD) of the pre-trained weight matrices to construct a principled starting point for LoRA adapters.

C.1.1 Mathematical Formulation

Consider a pre-trained linear layer with weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, where d_{in} and d_{out} denote input and output dimensions. We perform the truncated SVD:

$$\mathbf{W} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\top, \quad (\text{C.1})$$

where $\mathbf{U} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}$, $\mathbf{\Sigma} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is a diagonal matrix of singular values sorted in descending order, and $\mathbf{V} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$. We retain only the top r singular components (where r is the LoRA rank), obtaining $\mathbf{U}_r \in \mathbb{R}^{d_{\text{out}} \times r}$, $\mathbf{\Sigma}_r \in \mathbb{R}^{r \times r}$, and $\mathbf{V}_r \in \mathbb{R}^{d_{\text{in}} \times r}$.

The LoRA adapters are then initialized as:

$$\mathbf{A} = \frac{\mathbf{\Sigma}_r^{1/2} \mathbf{V}_r^\top}{\sqrt{\lambda}}, \quad \mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}, \quad (\text{C.2})$$

$$\mathbf{B} = \frac{\mathbf{U}_r \mathbf{\Sigma}_r^{1/2}}{\sqrt{\lambda}}, \quad \mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}, \quad (\text{C.3})$$

where λ is the LoRA scaling factor (typically set to α/r as in standard LoRA). This initialization ensures that $\mathbf{B} \mathbf{A} \cdot (\alpha/r) \approx \mathbf{W}_r$, where $\mathbf{W}_r$ is the best rank- r approximation of $\mathbf{W}$, thereby preserving the dominant singular directions of the pre-trained weights.

C.1.2 Why Spectral Initialization Matters

1. **Meaningful gradient directions:** Random initialization produces gradients that are largely uncorrelated with the task-specific geometry, making influence estimates noisy. Spectral initialization aligns the LoRA parameters with the principal components of the pre-trained weights, providing a stable and informative starting point for influence-based selection.
2. **Faster convergence:** By initializing adapters near a low-rank approximation of the pre-trained weights, the model begins from a state that already captures useful features, reducing the number of training rounds required to achieve stable influence estimates.
3. **Improved influence estimation accuracy:** In the first iteration, influence scores computed with spectrally initialized adapters exhibit lower variance and correlate more strongly with true sample contributions, as validated by our ablation studies (Section 4.5, Table 3).

C.2 Kronecker-Factored Approximate Curvature (K-FAC)

A central computational bottleneck in influence-based methods is the inversion of the Hessian matrix $\mathbf{H}_\theta$. For the LoRA parameter space, direct inversion has complexity $\mathcal{O}(p_{\text{loRa}}^3)$, which is prohibitive even for moderate LoRA ranks. K-FAC (Martens & Grosse, 2015) provides a scalable approximation by exploiting the structure of neural network layers.

C.2.1 K-FAC for a Single Linear Layer with LoRA

Consider a single linear layer with LoRA adapter matrices $\mathbf{A}$ and $\mathbf{B}$. The forward pass computes:

$$\mathbf{h} = (\mathbf{W}_{\text{base}} + \mathbf{B}\mathbf{A})\mathbf{x}, \quad (\text{C.4})$$

where $\mathbf{x}$ is the input activation vector. The gradient of the loss L with respect to the LoRA parameters depends on both the input activations and the backpropagated error signals.

Let $\delta = \nabla_{\mathbf{h}} L$ be the gradient vector with respect to the layer output. For the LoRA matrices, the gradients are:

$$\nabla_{\mathbf{A}} L = \mathbf{B}^\top \delta \mathbf{x}^\top, \quad (\text{C.5})$$

$$\nabla_{\mathbf{B}} L = \delta(\mathbf{A}\mathbf{x})^\top. \quad (\text{C.6})$$

The Hessian for this layer, when considered jointly over $\mathbf{A}$ and $\mathbf{B}$, has a block structure as shown in Section 3.3.2 of the main text. K-FAC approximates each block using Kronecker products.

C.2.2 Kronecker Product Approximation

K-FAC assumes that the Fisher information matrix (or Hessian) for a layer can be approximated as:

$$\mathbf{H} \approx \hat{\mathbf{H}} = \mathbb{E}[\delta\delta^\top] \otimes \mathbb{E}[\phi\phi^\top], \quad (\text{C.7})$$

where ϕ denotes the input activation vector to the layer (or a transformed version depending on the parameter structure), and $\otimes$ denotes the Kronecker product. For the LoRA case, we need to handle two sets of parameters $\mathbf{A}$ and $\mathbf{B}$ simultaneously.

Block-wise K-FAC for LoRA. We approximate the full Hessian $\mathbf{H}_{\theta_{\text{LoRA}}}$ as block-diagonal with respect to layers, and for each layer, we approximate the blocks $\mathbf{H}_{\text{AA}}$, $\mathbf{H}_{\text{BB}}$, and $\mathbf{H}_{\text{AB}}$ using Kronecker-structured approximations:

- For $\mathbf{H}_{\text{AA}}$ (second derivatives with respect to $\mathbf{A}$):

$$\mathbf{H}_{\text{AA}} \approx \mathbb{E}[(\mathbf{B}^\top \boldsymbol{\delta})(\mathbf{B}^\top \boldsymbol{\delta})^\top] \otimes \mathbb{E}[\mathbf{x}\mathbf{x}^\top]. \quad (\text{C.8})$$

- For $\mathbf{H}_{\text{BB}}$ (second derivatives with respect to $\mathbf{B}$):

$$\mathbf{H}_{\text{BB}} \approx \mathbb{E}[\boldsymbol{\delta}\boldsymbol{\delta}^\top] \otimes \mathbb{E}[(\mathbf{A}\mathbf{x})(\mathbf{A}\mathbf{x})^\top]. \quad (\text{C.9})$$

- For the cross-term $\mathbf{H}_{\text{AB}}$, a more intricate approximation is needed, but in practice, the block-diagonal approximation (ignoring $\mathbf{H}_{\text{AB}}$) or a structured low-rank approximation suffices for influence computation, as validated by prior work (Grosse et al., 2023).

C.2.3 Inverting the Approximate Hessian

Once the Kronecker-factored approximations are computed, the inverse H^{-1} can be obtained efficiently using the identity:

$$(\mathbf{P} \otimes \mathbf{Q})^{-1} = \mathbf{P}^{-1} \otimes \mathbf{Q}^{-1}. \quad (\text{C.10})$$

Thus, instead of inverting a $p_{\text{LoRA}} \times p_{\text{LoRA}}$ matrix, we invert smaller matrices of sizes $d_{\text{in}} \times d_{\text{in}}$ and $d_{\text{out}} \times d_{\text{out}}$, reducing complexity from $\mathcal{O}(p_{\text{LoRA}}^3)$ to $\mathcal{O}(d_{\text{in}}^3 + d_{\text{out}}^3 + r^3)$. The preconditioned natural gradient $\tilde{\mathbf{g}}(z) = \mathbf{H}_{\text{LoRA}}^{-1} \nabla_{\theta_{\text{LoRA}}} L(z)$ can then be computed via a sequence of Kronecker product operations without explicitly forming the full Hessian.

C.2.4 Implementation Details

In our implementation, we accumulate the following statistics over a subset of training data (or the candidate pool) after each active learning round:

- Activation covariance $\mathbb{E}[\mathbf{x}\mathbf{x}^\top]$ for each linear layer.
- Gradient covariance $\mathbb{E}[\boldsymbol{\delta}\boldsymbol{\delta}^\top]$ for the output-side signals.
- For the $\mathbf{A}$ and $\mathbf{B}$ adapters, we additionally compute transformed covariances as described above.

C.3 Integration with FLINT

The synergy between SVD-based spectral initialization and K-FAC-based influence estimation forms the backbone of FLINT’s efficiency:

1. **Spectral initialization** ensures that the LoRA manifold is well-conditioned from the start, providing stable gradients and curvatures for K-FAC estimation.
2. **K-FAC** enables the curvature-aware influence scores to be computed in time linear in the model dimension, rather than cubic, making multi-round active learning feasible on a single GPU.

Together, these components allow FLINT to achieve both theoretical rigor (preserving second-order curvature) and practical efficiency (runtime and memory comparable to first-order methods), as demonstrated in Table 2 of the main text.

D Baseline Implementation Details

To ensure a fair and rigorous evaluation, we benchmark FLINT against several representative data selection paradigms, spanning random sampling, heuristic retrieval, and state-of-the-art influence-based methods. All baseline methods (except Full fine-tuning) are constrained to select exactly the same budget of training data (20% of the full training set, $R_{\max} = 0.20$) per experiment. The detailed implementations and adaptations for each baseline are described below.

D.1 Full Fine-Tuning (Full)

The Full baseline represents the standard supervised fine-tuning over the entire available training dataset $\mathcal{D}_{\text{train}}$ without any data selection or filtering. We use the identical LoRA configuration (rank $r = 8$, $\alpha = 32$, dropout 0.05, target modules: $q, k, v, o, gate, up, down_proj$) and training hyperparameters (AdamW optimizer, learning rate 1×10^{-4} , 3 epochs, effective batch size 16, max sequence length 512) as our proposed method. This serves as the performance upper bound for standard practice and provides a reference to demonstrate whether data selection methods can achieve “less is more” generalization.

D.2 Random Selection (Random)

The **Random** baseline constructs the training subset by uniformly sampling 20% of the data from the unfiltered training pool $\mathcal{D}_{\text{train}}$ without replacement. The random seed is fixed at 42 for reproducibility. This serves as the most basic sanity check to verify whether sophisticated selection mechanisms provide statistically significant advantages over merely reducing the dataset size.

D.3 BM25

BM25 (Robertson & Zaragoza, 2009) is a classical lexical-matching retrieval function widely used to evaluate surface-level semantic similarity. In our targeted instruction tuning context, we treat the validation set $\mathcal{D}_{\text{val}}$ as the query set and the training samples as the document corpus.

Implementation details:

- **Text preprocessing:** We tokenize both training samples and validation queries using word-level tokenization. Following standard practice, we use unigram ($n = 1$) tokenization for both query and document representations.
- **Score aggregation:** For each training sample, we compute its BM25 score against every validation query and then take the **maximum** score across all validation queries. This aggregation strategy selects training samples that are highly similar to at least one validation example, which is more effective for capturing task-relevant content than averaging (which tends to favor broadly similar but less distinctive samples).
- **Selection:** We sort the training pool by the aggregated BM25 scores in descending order and select the top 20% samples with the highest scores.

Rationale: This baseline evaluates whether simple lexical and n-gram overlap with the target task is a sufficient indicator of data value for LLM fine-tuning.

D.4 DataInf

DataInf (Kwon et al., 2024) is a recently proposed efficient influence estimation method specifically designed for LoRA-tuned LLMs. It mitigates the intractable Hessian inversion by approximating the Hessian with the empirical Fisher Information Matrix (FIM) and using a closed-form inverse.

Implementation details:

- **LoRA configuration:** Identical to FLINT (rank $r = 8$, $\alpha = 32$, dropout 0.05, target modules as above).
- **Initialization:** Standard LoRA random initialization (Gaussian) is used, as in the original DataInf implementation.
- **Fisher matrix approximation:** We adopt the **block-diagonal approximation** as recommended in the original work, where the Fisher matrix is approximated per layer and aggregated.
- **Statistics accumulation:** The Fisher matrix statistics are accumulated following the original DataInf implementation, using the same data batches and accumulation strategy described in the official codebase.
- **Validation set:** The same validation set $\mathcal{D}_{\text{val}}$ as used in FLINT is employed to compute the validation gradient for influence scoring.
- **Selection budget:** Select exactly 20% of the training data.

D.5 LESS

LESS (Xia et al., 2024) (Selecting Influential Data for Targeted Instruction Tuning) represents the state-of-the-art in gradient-based data selection. To reduce the massive memory footprint of storing gradients for billions of parameters, LESS projects the gradients into a much smaller, low-dimensional space before computing dot-product similarities. **Implementation details:**

- **LoRA configuration:** We follow the original LESS settings, using rank $r = 128$, $\alpha = 512$, with target modules the same as above, to ensure fair comparison and reproducibility.
- **Initialization:** Standard LoRA random initialization (Gaussian) is used, as in the original LESS implementation.
- **Projection dimension:** We set the projection dimension to **4096**, following the recommended configuration in the original LESS paper for 3B–7B scale models.
- **Warm-up phase:** Prior to data selection, LESS performs a warm-up fine-tuning on 5% of the training data (randomly sampled) to obtain stable gradient representations. This warm-up uses the same LoRA configuration and learning rate as the main training.
- **Gradient storage:** We employ **gradient checkpointing** to reduce memory consumption during gradient computation, consistent with the original LESS implementation.
- **Validation set:** The same validation set $\mathcal{D}_{\text{val}}$ as used in FLINT is employed to compute the validation gradient projection for similarity scoring.
- **Selection budget:** Select exactly 20% of the training data.

E Extended Experimental Analyses

This section provides extended analyses that complement the main experimental results. Unless otherwise noted, all experiments use the same LoRA fine-tuning recipe and 20% data-selection budget as the main experiments.

E.1 General Instruction-Tuning Scenarios

The main paper focuses on reasoning benchmarks with reliable exact-match evaluation. To test whether FLINT also transfers to broader instruction-following settings, we evaluate LLaMA-3.2-3B on Alpaca-Clean and ShareGPT-Vicuna. Table 7 shows that FLINT consistently outperforms both full-data LoRA and random 20% selection, indicating that its gains are not restricted to math or symbolic reasoning tasks.

Dataset	FLINT (Ours)	Full-LoRA	Random-LoRA
Alpaca-Clean	82.4	80.5	74.2
ShareGPT-Vicuna	78.9	77.1	71.5

Table 7: Accuracy (%) on general instruction-tuning datasets with LLaMA-3.2-3B.

E.2 Expanded Quality and Diversity Baselines

We further compare FLINT with representative static quality/diversity selection strategies, including SemDeDup (Abbas et al., 2023) and K-Center Greedy (Sener & Savarese, 2018). Unlike these methods, FLINT explicitly models training-time contribution through entropy filtering and LoRA-space curvature-aware influence estimation. Table 8 shows that this training-dynamic criterion is more effective than static semantic redundancy removal or geometric coverage.

Paradigm	Method	GSM8K	BBH
Influence-based	FLINT (Ours)	71.10	66.80
Influence-based	LESS	68.80	54.10
Quality-based	SemDeDup	64.59	43.09
Diversity-based	K-Center Greedy	63.99	48.24

Table 8: Expanded baseline comparisons on GSM8K and BBH using LLaMA-3.2-3B under a 20% selection budget. All scores are reported in percentages.

E.3 Validation-Set Construction Robustness

Since FLINT computes influence by projecting training gradients onto validation gradients, validation-set representativeness is important. We compare three 50-sample validation strategies: a deliberately skewed validation set, random sampling without stratification, and FLINT’s distribution-aware stratified validation set. As shown in Table 9, stratified validation substantially improves both GSM8K and StrategyQA, supporting our choice of distribution-aware validation construction.

Validation Strategy	GSM8K	StrategyQA	Stability
Biased / skewed distribution	63.50	65.20	High variance
Random sampling	67.80	69.50	Seed-dependent
Stratified validation (Ours)	71.13	73.00	Robust

Table 9: Ablation of validation-set construction strategies with LLaMA-3.2-3B.

E.4 LoRA Configuration Sensitivity

Finally, we vary the LoRA rank and scaling factor on StrategyQA. Table 10 shows that FLINT remains consistently stronger than both full-data and random-selection LoRA across practical adapter capacities, while keeping total clock time stable.

Config	FLINT Acc.	Full-LoRA Acc.	Random Acc.	FLINT Time
$r = 8, \alpha = 16$	74.40	67.26	62.80	1.647h
$r = 16, \alpha = 32$	73.21	67.56	60.12	1.507h
$r = 32, \alpha = 64$	75.89	72.02	61.61	1.465h

Table 10: LoRA parameter sensitivity on StrategyQA with LLaMA-3.2-3B.

F Preparation Details of Training and Validation Datasets

Following standard experimental protocols in natural language processing, we partition each dataset into disjoint training, validation, and test splits (Fernández-Delgado et al., 2014). For researchers adapting this framework to other reasoning tasks, we recommend first conducting task-specific data mining and distributional analysis tailored to their specific scenarios and research goals (Li & Groth, 2025; Rula & D’Souza, 2023). Specifically, researchers should first characterize the distribution paradigm of the target reasoning task (e.g., reasoning step granularity, question type taxonomy) and quantify its core properties (Zhao et al., 2025a; Baker & McCallum, 1998); this enables principled validation set construction (e.g., sampling strategy, sample size, stratification criteria) to ensure the validation set is representative of the target task and effectively guides model training (Tahvili et al., 2020).

For GSM8K and StrategyQA, we adopt distribution-aware stratified sampling to select 50 samples from their official training partitions for constructing dedicated validation sets. Specifically, we first perform quantitative distributional analysis on the test sets to clarify the distribution characteristics of reasoning steps and question types (e.g., arithmetic calculation and logical reasoning for GSM8K; commonsense question answering and fact verification for StrategyQA). We then stratify the official training sets based on these two features, ensuring the proportion of each stratum aligns with that of the corresponding test set. Finally, we sample proportionally from each stratum to obtain 50 validation samples, whose representativeness is verified via quantitative analysis (e.g., KL divergence < 0.05) and visualization in Figure 4.

For BBH, which comprises 27 distinct sub-tasks, we follow the experimental setup of LESS (Xia et al., 2024) and construct the validation set by sampling 3 samples from each sub-task.

The remaining official training samples are randomly shuffled to form the final training sets, yielding training set sizes of 6,474 for GSM8K, 5,511 for BBH, and 2,601 for StrategyQA.

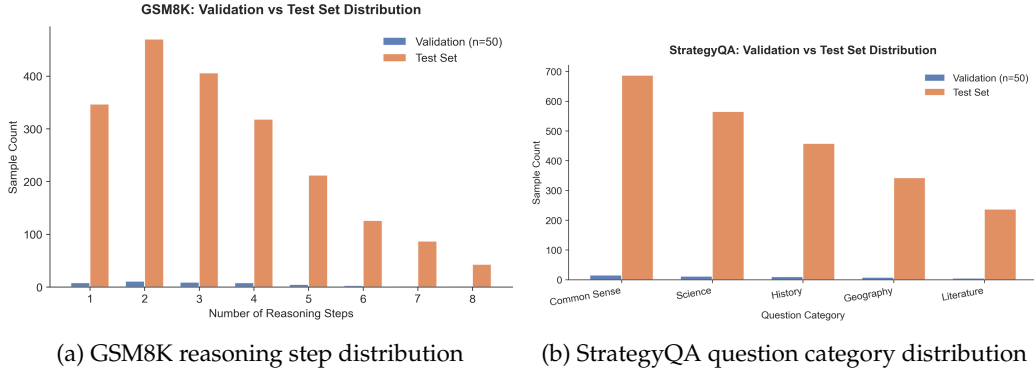


Figure 4: Distribution comparison between validation and test sets.

For clear exposition, we categorize all 27 BBH sub-tasks into three core reasoning categories in Table 11, which cover three essential requirements of real-world reasoning tasks.

For full reproducibility, we showcase representative training samples from GSM8K, StrategyQA, and BBH in the standard JSON format in Figures 5, 6, and 7, respectively. All datasets and their corresponding validation splits adhere strictly to this JSON structure for consistency.

Reasoning Category	BBH Sub-task Names
Logical & Mathematical Reasoning	algorithm reasoning, dyck languages, formal fallacies, logical deduction, multistep arithmetic, object counting, tracking shuffled objects, true false qa, web of lies, zeno godel
Linguistic & Common-sense Reasoning	causal reasoning, disambiguation qa, hyperbaton, movie recommendation, novel concepts, penguins in a table, semantic parsing, snarks, sujata riddle, word sorting
Perceptual & Situational Reasoning	date understanding, geometric shapes, navigate, reasoning about color, ruin names, sports understanding, temporal sequences

Table 11: Categorization of 27 BBH sub-tasks

```
{  "question": "It takes Roque two hours to walk to work and one hour to ride his bike to work. Roque walks to and from work three times a week and rides his bike to and from work twice a week. How many hours in total does he take to get to and from work a week with walking and biking?",

  "answer": "Roque takes 2*3 = <<2*3=6>>6 hours a week to walk to work.\nRoque takes 6*2 = <<6*2=12>>12 hours a week to walk to and from work.\nRoque takes 1*2 = <<1*2=2>>2 hours a week to bike to work.\nRoque takes 2*2 = <<2*2=4>>4 hours a week to bike to and from work.\nIn total, Roque takes 12+4 = <<12+4=16>>16 hour a week to go to and from work.\n#### 16" }
```

Figure 5: A representative sample from GSM8K.

```
{ "question": "Can the Great Depression be treated with Prozac?",

  "answer": "Thinking Process: Prozac is a pharmaceutical antidepressant for treatment of psychological disorders The Great Depression was an economic phenomenon occurring in the early 20th century\nSo the answer is No." }
```

Figure 6: A representative sample from StrategyQA.

```
{
  "input": "Sort the following words alphabetically: List: allyn carbonaceous majorca investigatory
supersede cetacean pathogenic johann tung paradigmatic pray",
  "target": "allyn carbonaceous cetacean investigatory johann majorca paradigmatic pathogenic pray
supersede tung",
  "subset": "word_sorting"
},
{
  "input": "If you follow these instructions, do you return to the starting point? Always face forward.
Take 1 step forward. Take 4 steps left. Take 6 steps left. Take 10 steps right. Take 7 steps backward. Take
5 steps left.\nOptions:\n- Yes\n- No",
  "target": "No",
  "subset": "navigate"
},
{
  "input": "In the following sentences, explain the antecedent of the pronoun (which thing the pronoun
refers to), or state that it is ambiguous.\nSentence: The developer met with the secretary because she
made a mistake.\nOptions:\n(A) The developer made a mistake\n(B) The secretary made a mistake\n(C)
Ambiguous",
  "target": "(C)",
  "subset": "disambiguation_qa"
}
```

Figure 7: Three representative samples from BBH.