BRIEF: Bi-level Coreset Selection for Efficient Instruction Tuning in LLMs

Chaoyuan Shen, Chi Zhang, Chengliang Chai, Jiacheng Wang,
Jia Yuan†, Yuping Wang, Ye Yuan, Guoren Wang, Lei Cao†
University of Arizona†, Beijing Institute of Technology

ABSTRACT

Instruction tuning is a key step in adapting large language models (LLMs) to effectively understand and follow human instructions. It enables LLMs to transform general knowledge into task-specific responses that align with user intent. Although many high-quality instruction tuning datasets have been released, efficiently utilizing these data sources during supervised fine-tuning (SFT) is important, as training on the full high-quality corpus can be computationally expensive. To address this inefficiency, we explore whether a compact, high-quality subset of instruction data can achieve comparable performance to full-dataset SFT, thereby reducing training cost without sacrificing effectiveness. To this end, this work proposes to select such a subset (a.k.a., *coreset*) of instruction examples that maintains comparable downstream performance while improving training efficiency. The key idea is inspired by our discovered decomposition that in instruction tuning, the training loss can be decomposed into two components that effectively quantify the contribution of an instruction to the two fundamental capabilities of LLMs, namely *knowledge-related* capability and *instruction following* capability. We then revisit the objective of the classical coreset approaches to balance the two capabilities when selecting instruction examples. Based on a bi-level formulation and a composite gradient distance that makes the objective submodular, we design an effective algorithm to achieve a bounded approximation error. Experiments on 4 datasets across 9 downstream tasks demonstrate that BRIEF reduces computational costs by 3× while improving accuracy by 5% on Llama-3.1-8B, Qwen3-4B and Mistral-Nemo-12B.

PVLDB Reference Format:
Chaoyuan Shen, Chi Zhang, Chengliang Chai, Jiacheng Wang,
Jia Yuan†, Yuping Wang, Ye Yuan, Guoren Wang, Lei Cao†. BRIEF: Bi-level Coreset Selection for Efficient Instruction Tuning in LLMs. PVLDB, 14(1): XXX-XXX, 2026.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/HR10108/BRIEF.

1 INTRODUCTION

Instruction tuning, also known as supervised fine-tuning (SFT), is critical in improving the ability of generic Large Language Models (LLMs) [19, 40, 72] to follow human instructions [50], thus effectively tailoring LLMs to particular domains [33, 63].

Recent research [31, 36, 63] has shown that when fine-tuning a model for a particular domain, it becomes essential to carefully select instruction examples, as many existing instruction-tuning datasets include a large portion of examples that are irrelevant or even harmful to the target domain [29, 55, 63]. However, this *data selection* problem is largely *overlooked* when the goal of SFT is to improve the capacity of LLMs to follow instructions in general [11, 18, 67], while fine-tuning LLMs on large training corpora is computationally expensive and time-consuming [23, 47, 70]. Therefore, many small organizations cannot afford it.

This underscores the necessity of data selection in this scenario. More specifically, if we were able to *select a high-quality, representative subset of data* (a.k.a., the *coreset* [14, 45]), on which fine-tuning an LLM would produce a model with performance competitive to a model fine-tuned on the whole training set, it would significantly improve training efficiency and reduce cost.

In deep learning, coreset selection methods [41, 42, 53] have been successfully applied to construct weighted subsets whose aggregated gradients closely approximate the overall gradient of the full training set. The approximation guarantee is established by bounding the gradient approximation (**GA**) error, defined as the difference between the gradient computed on the full dataset and the weighted sum of gradients from the coreset. By guaranteeing a small GA error, these methods effectively preserve the model's overall capability while significantly speeding up training.

However, unlike traditional deep learning, SFT aims to enhance the capability of an LLM in two different aspects [62], namely knowledge-related capability (KN) and instruction following capability (IF). KN determines whether the model can generate content containing correct real-world knowledge (e.g., Q: "What is the orbital period of the Moon around the Earth?" A: "Approximately 27.3 days"), while IF determines whether the model can understand and accurately execute diverse task instructions (e.g., Q: "Prepare a report following these specified steps." A: "[Detailed step-by-step report]").

By decomposing SFT loss into knowledge-related and instruction following terms, we identify the limitations of existing methods. This insight guides our proposed BRIEF, which optimizes for both capabilities simultaneously.

Decomposition of the SFT Loss. First, in Section4.1, we decompose the SFT training loss into KN and IF terms, aligning them with their measurable formats. This decomposition clarifies how the SFT loss simultaneously optimizes both KN and IF, and it allows us to give precise definitions for each term.

Accordingly, an ideal coreset selection method for SFT should preserve both the KN and IF capabilities. This means that it has

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

to simultaneously constrain the GA errors with respect to the gradients of the KN loss and the IF loss. Unfortunately, traditional coreset methods, which aim to minimize the GA error with respect to the whole loss function – in this case, the SFT loss – tend to fail to achieve this objective.

To illustrate this limitation with an example, consider samples that differ in how their gradients align with IF and KN. Some samples are strongly biased toward one capability: their gradients have a large norm and point mostly toward either IF or KN, while still being non-trivial for the other. Other samples are more balanced: their gradients lie between the two directions but have a much smaller norm, so they are average cases and not very informative for either capability. Current coreset methods, which aim to match the full dataset's average gradient, do not distinguish between combining strong but different capabilities and simply averaging many weak ones. As a result, they tend to select many of these average samples, because their small, balanced gradients naturally lie close to the global mean. Intuitively, repeatedly training on such mediocre examples will not make the model very strong; we would rather use examples that are strong for at least one capability and combine them so that their joint gradient still matches the overall gradient. However, standard objectives do not explicitly favor this second strategy, even though it can approximate the full gradient.

The reason why this happens is as follows. In coreset selection, the GA error measures the magnitude of the *delta vector* between the original gradient and the approximated gradient, which we call the *error vector*. Intuitively, the error vector w.r.t. the whole SFT gradient is the sum of those w.r.t. the KN and IF gradients. However, even if a coreset selection method is able to bound the GA error of the SFT gradient, i.e., an error vector with a small magnitude, in the worst case the GA errors w.r.t. the KN and IF gradients can be infinitely large when the angle of their error vectors is close to 180°.

As illustrated in Figure 1, the red arrow represents the error vector of the SFT gradient, while the green and blue arrows represent the error vectors of the KN gradient and the IF gradient, respectively. When the error vectors of the KN and IF gradients form an obtuse angle, the individual GA errors of KN and IF remain large even if the GA error of the SFT gradient is small (see Figure 1(b)).

Theoretically, this can be quantified by the triangle inequality: the magnitude of the sum of two vectors is smaller than the sum of the magnitudes of these two vectors. Therefore, minimizing the magnitude of the overall SFT error vector, which is the sum of the KN error vector and the IF error vector, does not necessarily lead to a small magnitude of the two individual error vectors, i.e., the GA errors of KN and IF.

New Objective Function. To solve this problem, we propose a new objective function for coreset selection in SFT. Rather than directly bounding the overall GA error of the SFT gradient, BRIEF constrains *the sum* of the GA errors of the KN and IF gradients. In this way, BRIEF simultaneously bounds the GA errors of the KN gradient, the IF gradient, and the overall SFT gradient: the individual GA errors of the KN and IF gradients are guaranteed to be smaller than their sum, while by the triangle inequality, this error sum upper bounds the error of the sum of the KN and IF gradients, i.e., the GA error of the SFT gradient (see Figure 1(c)). This new objective function thus ensures that the selected subset of

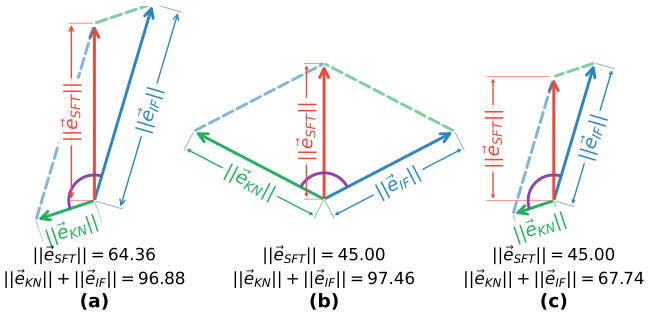


Figure 1: Different GA Error Optimization Objectives: (a) Original vector, (b) Minimizing GA error of the gradient sum, (c) Minimizing Sum of individual GA errors.

BRIEF retains the knowledge-related capability and the instruction following capability.

Optimization Methodology. To solve this reformulated optimization problem, which we prove to be NP-hard, we propose a bi-level optimization framework BRIEF. By introducing an auxiliary variable that partitions the solution space, we transform the original problem into an equivalent bi-level structure that can be solved efficiently. Specifically, at the lower level, given a partitioned solution space, we combine the GA errors of KN and IF with different weights to form a unified coreset selection objective. We prove that through a gradient distance composition technique, this weighted combination can be transformed into a submodular maximization problem, which can be solved by an efficient greedy algorithm with a provable approximation guarantee. This distance composition, which unifies the KN and IF gradient distances into a single metric, is an important design choice in BRIEF: it pinpoints how our method departs from existing coreset objectives while remaining compatible with standard facility-location optimization frameworks. The upper level systematically explores different partitions of the solution space. We show that the upper bound of the overall objective function – the sum of KN and IF gradient approximation errors – exhibits monotonic behavior at both ends of the auxiliary variable spectrum, which enables efficient exploration via ternary search to find a locally optimal partition. This dramatically reduces computational complexity compared to exhaustive search, yielding an effective partitioning between the two capability components.

Contributions. We summarize our main contributions as follows:

- By decomposing the SFT loss, for the first time we quantify the contribution of each instruction–response pair to the model's knowledge-related capability and instruction following capability, offering a new loss-level perspective on instruction tuning.
- Building on this decomposition, we derive a new optimization objective for coreset selection in SFT that minimizes the sum of individual GA errors, rather than only the GA error of the gradient sum, thereby simultaneously retaining both knowledge-related and instruction following capabilities.
- By formalizing and solving a bi-level optimization problem driven by our composite gradient distance, which jointly controls the GA errors of KN and IF, we obtain an efficient coreset selection algorithm that significantly reduces the complexity of coreset selection while providing a theoretical guarantee on the resulting GA error bound.

• Experiments on several advanced LLMs and real world datasets show that BRIEF selects a well-performing coreset, reducing computational costs by 3× while improving accuracy by 5% on Llama-3.1-8B, Qwen3-4B and Mistral-Nemo-12B.

2 RELATED WORK

We first review quality-based and task aware data selection methods for instruction tuning, then discuss gradient-based coreset methods originally developed for deep learning and their extensions to instruction-tuning settings, and finally connect these approaches to the bilevel optimization framework that underlies many recent coreset-based methods.

Quality-based Filtering for Instruction Tuning. Early work focused on removing low-quality data using simple rules [52, 58] or deduplication methods like SemDedup [1]. However, these heuristics often fail to generalize. Other researchers employ powerful LLMs (e.g., GPT-4) to score data quality, though this relies heavily on human-like intuition [17, 61, 71]. Perplexity is another common metric [6, 39, 44, 60], but it tends to favor simple and redundant sentences. To address this, Li et al. [32] introduced the Instruction Following Difficulty (IFD) score, which measures difficulty by comparing response generation with and without context.

Domain and Task aware Data Selection. Many methods select data to match specific downstream domains. Some use n-gram similarity to align corpora with validation sets [16, 65]. Others use influence functions to measure the impact of training data, as seen in LESS [64] and MATES [68]. Another approach involves training surrogate models to predict data relevance. For instance, DeepSeek-Math [55] uses active learning, while RHO-1 [35] employs a high-quality model for token-level filtering. However, training these specialized classifiers or surrogate models often requires significant computational resources and may not adapt well to different domains.

Gradient-based Coreset Methods. Gradient-based methods select a small weighted subset of data to approximate the gradients of the full dataset. CRAIG [41] achieves this by maximizing a submodular objective, while GRAD-MATCH [24] uses a greedy algorithm to match gradients directly. Extensions include Camel [34] for streaming data, and Goodcore [4], which handles incomplete relational data while jointly cleaning and training. Additionally, cluster-based methods [5] perform selection over feature-space clusters rather than individual points. Recently, TAGCOS [69] applied similar gradient clustering ideas to instruction tuning to create diverse datasets. Beyond direct gradient matching, coreset selection can also be formulated as a bi-level optimization problem. GLISTER [25] and RETRIEVE [26] adopt this strategy: the inner loop trains on a subset, while the outer loop greedily selects examples to minimize validation loss using hypergradient approximations.

3 PRELIMINARY

In this section, we first introduce the basic notation and recall supervised fine-tuning (SFT) and classical coreset selection, which our method builds upon. Table 1 summarizes all the symbols used throughout this paper.

Supervised Fine-tuning (SFT). Suppose that θ denotes the model parameters and D denotes the full training dataset. During the SFT stage, the full training dataset D with N data points can be represented as an instruction–response corpus $D = \{(x_i, y_i)\}_{i=1}^{N}$. For each data point, the instruction sequence is x_i and the response sequence is y_i, where $|y_i|$ denotes the length of the response sequence y_i (i.e., how many tokens it contains), y_i^t denotes the t-th token in y_i and $y_i^{<t} = (y_i^1, \ldots, y_i^{t-1})$ denotes the prefix sequence before the t-th token in sequence y_i. Thus, the SFT loss can be written as:

$$\mathcal{L}_{\text{SFT}}\big((x_i, y_i); \theta\big) \;=\; -\frac{1}{|y_i|}\sum_{t=1}^{|y_i|} \log p_\theta\big(y_i^t \,\big|\, (x_i, y_i^{<t})\big). \tag{1}$$

Table 1: Symbols used throughout the paper.

Symbol	Description
θ, Θ	Trainable parameters; Feasible parameter space
D, N	Full training dataset; Number of examples
(x_i, y_i)	i-th instruction–response pair from D
$\lvert y_i\rvert$, y_i^t, $y_i^{<t}$	Token length; t-th token; Prefix before t
$p_\theta(\cdot)$	Model conditional probability under θ
$\mathcal{L}_{\text{SFT}}$, $\mathcal{L}_{\text{KN}}$, $\mathcal{L}_{\text{IF}}$	SFT loss; Knowledge loss; Instruction following loss
$\nabla\mathcal{L}_i(\theta)$	Per-example gradient of the loss at parameters θ
C, W, ω_j	Selected Coreset; Weight set; Weight of element j
$\gamma(j)$	Index map linking coreset item j back to D
K, ε	Size budget; Allowed GA error
$\text{IFD}(y\lvert x;\theta)$, $\text{PPL}(\cdot)$	Instruction following difficulty; Perplexity
d_{ij}^{KN}, d_{ij}^{IF}	Pairwise gradient distances in KN and IF spaces
d_{ij}^{SFT}	Composite distance combining KN and IF terms
$\mathcal{B}_{\text{KN}}(C)$, $\mathcal{B}_{\text{IF}}(C)$	Upper bounds on KN and IF GA errors for C
α, δ	Error budget split auxiliary parameter; Search tolerance
$F(C)$	Facility-location objective
$K_{\text{KN}}(\tau)$, $K_{\text{IF}}(\tau)$	Minimum coreset size achieving error budget τ

Coreset. The state-of-the-art coreset selection framework [14, 45] aims to select a subset of data points $C \subseteq D$ (with non–negative weights $\{\omega_j\}_{j=1}^{|C|}$) such that the coreset gradient well approximates the full gradient , i.e., we keep only a smaller weighted subset of instruction–response pairs while still matching the training effect produced by the original dataset.

Intuitively, each data point (x_i, y_i) contributes a gradient vector $\nabla\mathcal{L}_i(\theta)$, and the goal is to select a smaller weighted subset whose combined gradients approximate those of the entire dataset. This naturally involves a trade-off between subset compactness and approximation fidelity, which we formalize through two equivalent formulations below.

The most direct way is to minimize the gradient approximation error (GA error) subject to a size constraint:

$$\min_{C\subseteq D,\ \omega_j\geq 0}\ \max_{\theta\in\Theta} \underbrace{\Big\|\underbrace{\sum_{i=1}^{|D|}\nabla\mathcal{L}_i(\theta)}_{\text{full gradient}} - \underbrace{\sum_{j=1}^{|C|}\omega_j\nabla\mathcal{L}_{\gamma(j)}(\theta)}_{\text{coreset gradient}}\Big\|}_{\text{gradient approximation error}} \quad \text{s.t.} \quad |C| \leq K \tag{2}$$

Here, K denotes the coreset size budget and Θ represents the feasible parameter space.

Equation (2) aims to minimize the GA error, which measures the gap between the full dataset's gradient and the weighted coreset's gradient. However, directly solving this formulation is computationally intractable. To overcome this difficulty, many studies [5, 41, 42, 53] adopt a dual formulation by interchanging the optimization objective and the constraint, as shown below:

$$\min_{C\subseteq D,\ \omega_j\geq 0} |C| \quad s.t. \quad \max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_i(\theta) - \sum_{j=1}^{|C|}\omega_j\nabla\mathcal{L}_{\gamma(j)}(\theta)\Big\| \leq \varepsilon \tag{3}$$

Equation (3) captures the essential trade-off between approximation accuracy (ε) and subset size ($|C|$). Modern coreset methods [5, 41, 42, 53] typically solve the fixed-budget problem (Eq. (2)) by optimizing this dual formulation—finding the minimal subset for a specific error tolerance—and then adapting the solution to the budget K. We build on the same dual-formulation framework, but explicitly split the SFT GA error into knowledge-related and instruction following terms, letting these two parts guide the data selection process.

Example. Consider a toy dataset of 8 data points forming three groups with identical gradients (group sizes: 3, 3, and 2). To satisfy $K = 3$, we can simply pick one point from each group and weight them by their group size (i.e., weights 3, 3, 2). This achieves near-zero gradient error, demonstrating how the method removes data redundancy without losing the training effect.

4 CORESET SELECTION FOR SFT

In this section, we begin by presenting a simple yet underexplored algebraic decomposition of the SFT loss (Section 4.1), which reveals how individual training data points impact both knowledge-related and instruction following capabilities of an LLM from a new loss-level perspective. Building on this decomposition, we reformulate the coreset selection problem to preserve both capabilities simultaneously (Section 4.2) and provide an overview of our BRIEF approach (Section 4.3). We then detail our efficient data selection algorithm (Section 5).

4.1 SFT Loss Decomposition

By adding $\log p_\theta(y^t \mid y^{<t})$, Eq. (1) can be written as

$$\mathcal{L}_{\text{SFT}}(y \mid x; \theta) = -\tfrac{1}{|y|} \sum_{t=1}^{|y|} \Big[\log p_\theta(y^t \mid y^{<t}) + \log \tfrac{p_\theta(y^t \mid x, y^{<t})}{p_\theta(y^t \mid y^{<t})} \Big]. \quad (4)$$

This leads to the decomposition:

$$\mathcal{L}_{\text{SFT}}(y \mid x; \theta) = \mathcal{L}_{\text{KN}}(y; \theta) + \mathcal{L}_{\text{IF}}(y \mid x; \theta). \quad (5)$$

where

$$\mathcal{L}_{\text{KN}}(y; \theta) = -\tfrac{1}{|y|} \sum_{t=1}^{|y|} \log p_\theta(y^t \mid y^{<t}), \quad (6)$$

and

$$\mathcal{L}_{\text{IF}}(y \mid x; \theta) = -\tfrac{1}{|y|} \sum_{t=1}^{|y|} \log \tfrac{p_\theta(y^t \mid x, y^{<t})}{p_\theta(y^t \mid y^{<t})}. \quad (7)$$

The first component $\mathcal{L}_{\text{KN}}(y; \theta)$ denotes **knowledge-related loss**, which is in the same format as the pretraining loss of LLMs [47]. It represents the negative log-likelihood of predicting the next token y^t given only the previous tokens $y^{<t}$. This component measures how well the model predicts the response tokens without using the instruction x. Therefore, it mainly reflects the knowledge inherently stored in the answers.

Next, we show that the second component represents the **instruction following loss**. It measures the log-probability ratio of generating y with instruction x versus without it. This captures the additional information provided by the instruction x that helps produce the correct response y. Therefore, it effectively quantifies to what extent the instruction improves the model's ability to generate the desired output.

To prove this, we first introduce a metric called *instruction following Difficulty (IFD)* [30–32], which, given an instruction (x, y), identifies discrepancies between the expected responses of a model and its generation capability.

$$\text{IFD}(y \mid x; \theta) = \frac{\text{PPL}(y \mid x; \theta)}{\text{PPL}(y; \theta)}, \quad (8)$$

where

$$\text{PPL}(y \mid x; \theta) = \exp\left(\mathcal{L}_{\text{SFT}}((x, y); \theta)\right), \quad (9)$$

$$\text{PPL}(y; \theta) = \exp\left(\mathcal{L}_{\text{KN}}(y; \theta)\right). \quad (10)$$

Intuitively, IFD measures the potential of this instruction to improve the instruction following capability of a model. The key observation here is that taking the log on $\text{IFD}(y \mid x; \theta)$ will derive the exact form of $\mathcal{L}_{\text{IF}}(y \mid x; \theta)$, i.e., the second component in Eq. (4).

$$\mathcal{L}_{\text{IF}}(y \mid x; \theta) = \log \text{IFD}(y \mid x; \theta). \quad (11)$$

This shows that $\mathcal{L}_{\text{IF}}$ primarily focuses on exploring the training examples to improve the instruction following capacity of a model, thus representing the *instruction following loss*.

Notably, $\mathcal{L}_{\text{IF}}$ is derived directly from the SFT loss, namely the actual objective used during training. This connection provides a principled explanation for the effectiveness of *instruction following difficulty (IFD)* and, in turn, proves the critical role of training examples in improving the instruction following capability of a model.

In summary, this decomposition highlights that the SFT loss reflects two core model capabilities: knowledge-related capability, driven by $\mathcal{L}_{\text{KN}}$, and instruction following capability, driven by $\mathcal{L}_{\text{IF}}$. At the level of individual training examples, each instruction–response pair influences the model through a knowledge-related term and an instruction following term, which are exactly the components we later control when selecting and weighting examples in the coreset.

4.2 Problem Definition

In this section, we reformulate the coreset selection problem for instruction tuning in LLMs. Given a full training dataset $D = \{(x_i, y_i)\}_{i=1}^{|D|}$, our goal is to select a subset $C \subseteq D$ with weights $\{\omega_j\}_{j=1}^{|C|}$ that preserves both knowledge-related and instruction following capabilities of D simultaneously during the instruction tuning period.

SFT Gradient Approximation. Incorporating the SFT loss from Eq. (1) into the traditional coreset objective (Eq. (2)), we obtain:

$$\max_{\theta \in \Theta} \Big\| \sum_{i=1}^{|D|} \nabla \mathcal{L}_{\text{SFT}}\big((x_i, y_i); \theta\big) - \sum_{j=1}^{|C|} \omega_j \nabla \mathcal{L}_{\text{SFT}}\big((x_{\gamma(j)}, y_{\gamma(j)}); \theta\big) \Big\| \quad (12)$$

subject to $|C| \leq K$, where $\gamma(j)$ denotes the index mapping from coreset to the full dataset.

Triangle Inequality Analysis. Building on the SFT loss decomposition established in Eq. (5), we apply the triangle inequality to

the gradient approximation error in Eq. (12):

$$\begin{aligned}
&\max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_{\text{SFT}}\big((x_i,y_i);\theta\big)-\sum_{j=1}^{|C|}\omega_j\nabla\mathcal{L}_{\text{SFT}}\big((x_{\gamma(j)},y_{\gamma(j)});\theta\big)\Big\| \\
&\le \max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_{\text{KN}}(y_i;\theta)-\sum_{j=1}^{|C|}\omega_j\nabla\mathcal{L}_{\text{KN}}(y_{\gamma(j)};\theta)\Big\| \\
&\quad+\max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_{\text{IF}}(y_i|x_i;\theta)-\sum_{j=1}^{|C|}\omega_j\nabla\mathcal{L}_{\text{IF}}(y_{\gamma(j)}|x_{\gamma(j)};\theta)\Big\|.
\end{aligned}\tag{13}$$

which reveals a simple but important insight: minimizing the left side (the GA error of the SFT gradient) provides no guarantee for the individual components on the right. For instance, a small SFT GA error can coexist with large individual component errors when the error vectors form obtuse angles, as illustrated in Figure 1(b). Thus, we reformulate the optimization objective.

$$\begin{aligned}
\min_{C\subseteq D,\,\omega_j\ge 0}\Bigg[&\underbrace{\max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_{\text{KN}}(y_i;\theta)-\sum_{j=1}^{|C|}\omega_j\nabla\mathcal{L}_{\text{KN}}(y_{\gamma(j)};\theta)\Big\|}_{\text{Knowledge-related GA Error}} \\
&+\underbrace{\max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_{\text{IF}}(y_i|x_i;\theta)-\sum_{j=1}^{|C|}\omega_j\nabla\mathcal{L}_{\text{IF}}(y_{\gamma(j)}|x_{\gamma(j)};\theta)\Big\|}_{\text{Instruction Following GA Error}}\Bigg] \\
&\text{s.t.}\quad |C|\le K
\end{aligned}\tag{14}$$

Reformulated Optimization Objective. The reformulation (Eq. (14)) differs from traditional approaches by explicitly constraining both component errors. Specifically, through the triangle inequality, any solution to Eq. (14), where the total error does not exceed a certain upper bound, guarantees that the SFT GA error is also within that bound. This reformulated objective ensures that both knowledge-related (KN) and instruction following (IF) capabilities are preserved in the selected coreset, addressing the unique requirements of SFT that traditional coreset methods overlook. In the following sections, we present our BRIEF framework to efficiently solve this problem.

4.3 BRIEF Overview

The BRIEF framework addresses the coreset selection problem (Eq. (14)) through a bi-level optimization approach that simultaneously retains both knowledge-related and instruction following capabilities.

As illustrated in Figure 2, BRIEF starts with a warm-up phase to compute gradient-based distances for both knowledge-related and instruction following. Next, it employs a bi-level optimization: the upper level adjusts a parameter α to explore trade-offs, while the lower level selects the coreset using a combined metric. This structure transforms the original NP-hard problem into two manageable sub-tasks, drastically reducing computational complexity. Intuitively, instead of searching over both α and the coreset at the same time, we split the problem into two steps: for a given α we select a coreset, and then we adjust α with a simple one-dimensional search. This allows BRIEF to efficiently identify and weight the most representative instruction–response pairs from the SFT corpus.

To make this pipeline more concrete, consider a tiny SFT dataset with only three instruction–response pairs: a math question, a coding task, and a commonsense question. In the warm-up step, the base model is briefly trained on these three pairs and their KN and IF gradients are collected. BRIEF then builds KN and IF distances between all examples. In the upper level, a candidate α is chosen, which decides how much importance to put on KN versus IF. In the lower level, given this α, the algorithm keeps one or two examples that best cover the remaining ones under this capability allocation. The ternary search in the upper level gradually adjusts α so that this allocation between KN and IF becomes balanced.

Warm-up Phase. BRIEF starts by training the base model on a subset $\mathcal{D}_{\text{warm}}$ to capture the data distribution (Line 1). Then, it calculates gradients for both knowledge-related ($\nabla\mathcal{L}_{KN}$) and instruction following ($\nabla\mathcal{L}_{IF}$) capabilities to compute pairwise distances between data points (Lines 2-3). This step organizes training examples in the gradient spaces, enabling precise measurement of data similarity and coverage for the subsequent selection.

Bi-level Optimization Algorithm. The core of BRIEF lies in its bi-level optimization structure (Lines 4-17, see Section 5 for details). Specifically, the upper level employs ternary search over the auxiliary variable $\alpha \in (0, 1)$, which serves as a partition parameter that divides the solution space into different regions based on capability trade-offs. For each partition defined by α, the lower level (Lines 7-8) performs coreset selection by formulating it as a submodular facility location problem for efficient calculation. In each iteration, this means we first fix how much error budget to allocate to KN versus IF and then choose a coreset that best represents the full dataset under this split.

Upper Optimization. The upper optimization employs ternary search (Lines 5-15), a divide-and-conquer approach that iteratively narrows the search interval by evaluating two internal points (m_1 and m_2) and discarding the subinterval with worse performance. The algorithm exploits the monotonicity at both extremes of the α spectrum: the objective decreases for small α (where KN constraints dominate) and increases for large α (where IF constraints dominate). This property ensures that at least one local minimum exists in the interval, and the ternary search converges to a locally optimal partition parameter α^* in $O(\log(1/\delta))$ iterations, where δ is the convergence threshold. Once α^* is determined (Line 16), the final coreset can be selected through the lower-level optimization (Line 17), thereby preserving both model KN and IF capabilities while improving training efficiency.

Lower Optimization. For each iteration of the lower optimization (Algorithm 2), given the partition parameter α from the upper level, we construct a composite distance metric $d^{\text{SFT}} = \frac{1}{\alpha}d^{KN} + \frac{1}{1-\alpha}d^{IF}$ that integrates both capability measures. The algorithm then formulates coreset selection as a submodular facility location problem, where the objective measures the total distance from all data points to their nearest representatives in the coreset. Through greedy selection, it iteratively chooses $K = \lceil\gamma|\mathcal{D}|\rceil$ data points that maximally reduce the sum of distances from unselected points to their nearest selected representatives. This greedy approach provides a $(1 - 1/e)$-approximation guarantee for the submodular maximization problem [48]. Finally, it computes weights for each selected point based on the number of data points assigned to it.

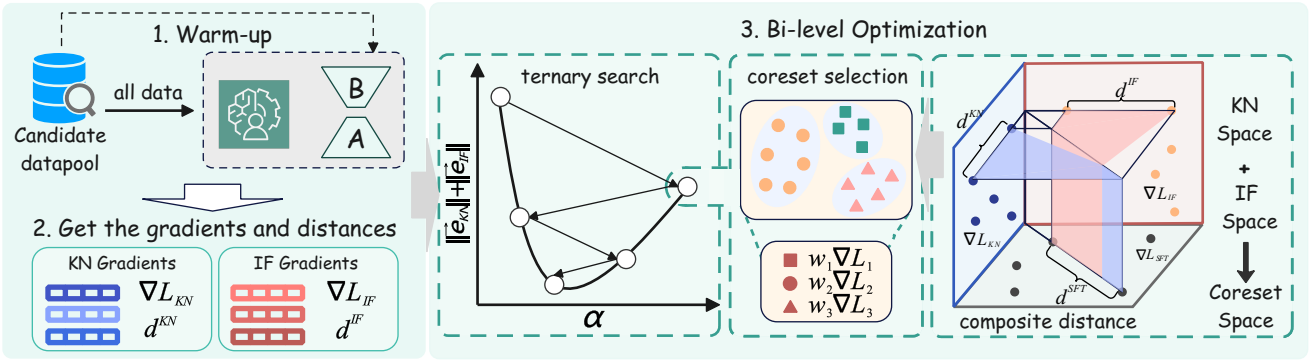


Figure 2: The Overall Framework of BRIEF.

Algorithm 1 BRIEF Framework

Input: Candidate data pool $\mathcal{D}$, selection ratio γ, base model θ
Output: Coreset $C \subseteq \mathcal{D}$, weights W

$\theta \leftarrow \textsc{WarmupTraining}(\mathcal{D}, \theta)$
Compute $\nabla \mathcal{L}_{\text{KN}}(y_i; \theta), \nabla \mathcal{L}_{\text{IF}}(y_i | x_i; \theta)$ for all $(x_i, y_i) \in \mathcal{D}$
Compute pairwise distances $d_{ij}^{\text{KN}}, d_{ij}^{\text{IF}}$ for all $i, j \in [N]$
$l \leftarrow 0, \quad r \leftarrow 1$
while $|r - l| > \delta$ **do** ▷ ternary search over α, $\delta \to 0$
  $m_1 \leftarrow l + \frac{r-l}{3}, \quad m_2 \leftarrow r - \frac{r-l}{3}$
  $(C_1, W_1) \leftarrow \textsc{GreedyCoresetSelection}(\mathcal{D}, \gamma, m_1, d^{\text{KN}}, d^{\text{IF}})$
  $(C_2, W_2) \leftarrow \textsc{GreedyCoresetSelection}(\mathcal{D}, \gamma, m_2, d^{\text{KN}}, d^{\text{IF}})$
  $E_1 \leftarrow \textsc{ComputeGAErrorBound}(C_1, W_1)$
  $E_2 \leftarrow \textsc{ComputeGAErrorBound}(C_2, W_2)$
  if $E_1 \le E_2$ **then**
    $r \leftarrow m_2$
  else
    $l \leftarrow m_1$
  end if
end while
$\alpha^\star \leftarrow \dfrac{l + r}{2}$
return $\textsc{GreedyCoresetSelection}(\mathcal{D}, \gamma, \alpha^\star, d^{\text{KN}}, d^{\text{IF}})$

5 THE BRIEF APPROACH

In this section, we present the technical details that solve the optimization problem formulated in Eq. (14). This problem is proven to be prohibitively expensive (Section 5.1). We first demonstrate that directly optimizing this problem is computationally prohibitive, while transforming it into a bi-level optimization structure makes it possible to design an efficient solution (Section 5.2). We then detail the lower-level optimization, which solves the coreset selection problem w.r.t. a specific auxiliary variable α that partitions the solution space by allocating error budgets between the two capability components (Section 5.3). Finally, we present the upper-level optimization that efficiently searches for a locally optimal α value by exploiting the monotonic properties of the objective function (Section 5.4).

5.1 Problem Complexity

We now establish the computational intractability of the optimization problem in Eq. (14).

Theorem 5.1. *The optimization problem in Eq.* (14) *is NP-hard.*

Proof. We prove NP-hardness by reduction from the classical coreset selection problem for gradient approximation, which is known to be NP-hard [41].

Consider the following special case of Eq. (14). For every training example (x_i, y_i) in D, we set the instruction to be empty, i.e., $x_i = \emptyset$ for all i. In our formulation, an empty instruction does not affect the conditional distribution of the next token, so the instruction following loss for each example becomes

$$\mathcal{L}_{\text{IF}}(y_i \mid \emptyset; \theta) = -\frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \log \frac{p_\theta(y_i^t \mid \emptyset, y_i^{<t})}{p_\theta(y_i^t \mid y_i^{<t})} \tag{15}$$

$$= -\frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \log 1 = 0, \tag{16}$$

and thus its gradient also vanishes:

$$\nabla_\theta \mathcal{L}_{\text{IF}}(y_i \mid \emptyset; \theta) = 0 \quad \text{for all } i, \theta. \tag{17}$$

Plugging this special case into Eq. (14), the second term (the instruction following GA error) is identically zero and the optimization problem reduces to

$$\min_{C \subseteq D,\, \omega_j \ge 0} \max_{\theta \in \Theta} \left\| \sum_{i=1}^{|D|} \nabla \mathcal{L}_{\text{KN}}(y_i; \theta) - \sum_{j=1}^{|C|} \omega_j \nabla \mathcal{L}_{\text{KN}}(y_{\gamma(j)}; \theta) \right\|, \tag{18}$$

which is exactly the standard gradient-based coreset selection objective studied in prior work [41].

Therefore, the optimization problem in Eq. (14) contains the classical coreset selection problem as a special case. By standard notions of polynomial-time reductions in computational complexity theory [20, 51], any problem that contains an NP-hard problem as a special case is itself NP-hard. Consequently, Eq. (14) is NP-hard. □

5.2 Bi-level Optimization Problem

To efficiently solve the NP-hard problem formulated in Eq. (14), we transform it into a bi-level optimization framework. Following popular coreset selection approaches [41, 42, 53], we first derive the dual formulation of our coreset selection problem and then establish its equivalence to the original objective:

$$\min_{C \subseteq D,\, \omega_j \ge 0} \quad |C|$$

$$\text{s.t.} \quad \max_{\theta \in \Theta} \left\| \sum_{i=1}^{|D|} \nabla \mathcal{L}_{\text{KN}}(y_i; \theta) - \sum_{j=1}^{|C|} \omega_j \nabla \mathcal{L}_{\text{KN}}(y_{\gamma(j)}; \theta) \right\|$$

$$+ \max_{\theta \in \Theta} \left\| \sum_{i=1}^{|D|} \nabla \mathcal{L}_{\text{IF}}(y_i \mid x_i; \theta) - \sum_{j=1}^{|C|} \omega_j \nabla \mathcal{L}_{\text{IF}}(y_{\gamma(j)} \mid x_{\gamma(j)}; \theta) \right\| \le \varepsilon \tag{19}$$

where ε represents the budget of the overall GA error.

Specifically, Eq.(19) serves as the dual formulation of Eq.(14), analogous to the relationship between Eqs.(2) and (3). We adopt this tolerance-constrained view to facilitate our analysis, while the derived algorithm effectively addresses the original size-constrained optimization.. The challenge of directly solving Eq. (19) lies in the coupled constraint that involves the sum of two gradient approximation errors. To decouple this constraint and enable more efficient optimization, we introduce an auxiliary variable $\alpha \in (0,1)$ that partitions the error budget between the KN and IF capabilities. Importantly, α is not a hyperparameter requiring manual tuning but is automatically optimized through our bi-level framework, as detailed below.

Specifically, we can transform our problem into the following equivalent bi-level optimization formulation:

$$\begin{aligned} &\min_{\alpha\in(0,1)} \min_{C\subseteq D,\omega_j\geq 0} \quad |C| \\ \text{s.t.} \quad &\max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_{\text{KN}}(y_i;\theta) - \sum_{j=1}^{|C|}\omega_j\nabla\mathcal{L}_{\text{KN}}(y_{\gamma(j)};\theta)\Big\| \leq \alpha\varepsilon \\ &\max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_{\text{IF}}(y_i \mid x_i;\theta) - \sum_{j=1}^{|C|}\omega_j\nabla\mathcal{L}_{\text{IF}}(y_{\gamma(j)} \mid x_{\gamma(j)};\theta)\Big\| \leq (1-\alpha)\varepsilon \end{aligned} \tag{20}$$

We now show the equivalence of Eq. (19) and Eq. (20).

Theorem 5.2. *The bi-level optimization problem in Eq.* (20) *is equivalent to the formulation in Eq.* (19)*. In particular, any optimal solution* (C^*, W^*) *to Eq.* (19) *corresponds to an optimal solution of Eq.* (20) *for some* $\alpha^* \in (0,1)$*, and vice versa.*

Proof. Let (C^*, W^*) denote an optimal solution of Eq. (19). Define the individual gradient approximation errors as

$$\begin{aligned} e^*_{\text{KN}} &= \max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_{\text{KN}}(y_i;\theta) - \sum_{j=1}^{|C^*|}\omega^*_j\nabla\mathcal{L}_{\text{KN}}(y_{\gamma^*(j)};\theta)\Big\|, \\ e^*_{\text{IF}} &= \max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_{\text{IF}}(y_i \mid x_i;\theta) - \sum_{j=1}^{|C^*|}\omega^*_j\nabla\mathcal{L}_{\text{IF}}(y_{\gamma^*(j)} \mid x_{\gamma^*(j)};\theta)\Big\|. \end{aligned} \tag{21}$$

From the feasibility of (C^*, W^*) in Eq. (19), we have $e^*_{\text{KN}} + e^*_{\text{IF}} \leq \varepsilon$. In consideration $e^*_{\text{KN}}, e^*_{\text{IF}} \in (0,\varepsilon)$, we have $\alpha^* = \frac{e^*_{\text{KN}}}{\varepsilon} \in (0,1)$. With this choice of α^*, we have $e^*_{\text{KN}} = \alpha^*\varepsilon$, $\quad e^*_{\text{IF}} \leq \varepsilon - e^*_{\text{KN}} = (1-\alpha^*)\varepsilon$, which shows that (C^*, W^*) is feasible for the lower-level problem in Eq. (20) under α^*. Since $|C^*|$ is minimal by assumption, (C^*, W^*) is also optimal for that lower-level problem.

Conversely, let $(C^\dagger, W^\dagger)$ be an optimal solution to Eq. (20) for some $\alpha \in (0,1)$. By the constraints, it satisfies $e^\dagger_{\text{KN}} \leq \alpha\varepsilon, e^\dagger_{\text{IF}} \leq (1-\alpha)\varepsilon$. Summing these inequalities yields $e^\dagger_{\text{KN}} + e^\dagger_{\text{IF}} \leq \varepsilon$, which shows that $(C^\dagger, W^\dagger)$ is feasible for Eq. (19). Since both problems minimize the same objective $|C|$, the optimality is preserved. □

This bi-level structure offers significant computational advantages. At the upper level, we only tune the continuous parameter α, which splits the GA error budget between KN and IF. Because the objective is unimodal in α, we can search for a good value using ternary search. For each fixed α, the lower level reduces to a standard submodular facility-location coreset selection problem with approximation guarantees.

5.3 Lower-level Optimization Problem

Given a specific parameter α learned by the upper-level optimization, the lower-level problem focuses on finding the minimal coreset C and corresponding weights W that satisfy the decoupled gradient approximation constraints in Eq. (20).

This lower-level problem represents a constrained coreset selection task, where we must simultaneously bound the gradient approximation errors for both knowledge-related capability (with error budget $\alpha\varepsilon$) and instruction following capability (with error budget $(1-\alpha)\varepsilon$). While the parameter α partitions the total error budget between the two capabilities, solving this optimization problem remains computationally challenging due to its combinatorial nature and the need to evaluate gradient approximations over the parameter space Θ. However, we show that the aforementioned dual-constraint optimization problem can be reformulated as a submodular set cover problem, which admits efficient approximation algorithms.

Formulation as a Submodular Set Cover Problem. To precisely formulate our coreset selection task as a submodular set cover problem, we define deterministic bounds $\mathcal{B}_{\text{KN}}(C)$ and $\mathcal{B}_{\text{IF}}(C)$ for any subset $C \subseteq D$ and parameter $\theta \in \Theta$.

$$\mathcal{B}_{\text{KN}}(C) \overset{\text{def}}{=} \sum_{i=1}^{|D|}\min_{c_j\in C} d^{\text{KN}}_{ij}, \quad \mathcal{B}_{\text{IF}}(C) \overset{\text{def}}{=} \sum_{i=1}^{|D|}\min_{c_j\in C} d^{\text{IF}}_{ij}. \tag{22}$$

Here, pairwise distances d^{KN}_{ij} and d^{IF}_{ij} are used to measure the normed difference between the gradient of data point $c_i \in D$ and data point $c_j \in C$, which are defined separately as:

$$d^{\text{KN}}_{ij} \overset{\text{def}}{=} \max_{\theta\in\Theta} \|\nabla\mathcal{L}_{\text{KN}}(y_i;\theta) - \nabla\mathcal{L}_{\text{KN}}(y_j;\theta)\| \tag{23}$$

$$d^{\text{IF}}_{ij} \overset{\text{def}}{=} \max_{\theta\in\Theta} \|\nabla\mathcal{L}_{\text{IF}}(y_i|x_i;\theta) - \nabla\mathcal{L}_{\text{IF}}(y_j|x_j;\theta)\| \tag{24}$$

Following the theoretical framework established in Mirzasoleiman et al. [41], we can derive upper bounds for the gradient approximation errors. Specifically, the maximum gradient approximation error for any subset C can be bounded by the sum of minimum pairwise distances between the gradient of each point in D and its closest representative in C. Intuitively, if every point in D has a close neighbour in C in the gradient space, then the sum of gradients over D can be well-approximated by reweighting the gradients of points in C; the quantities $\mathcal{B}_{\text{KN}}(C)$ and $\mathcal{B}_{\text{IF}}(C)$ measure how well C covers D under the KN and IF gradient distances.

Formally, this gives us:

$$\max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_{\text{KN}}(y_i;\theta) - \sum_{j=1}^{|C^*|}\omega^*_j\nabla\mathcal{L}_{\text{KN}}(y_{\gamma^*(j)};\theta)\Big\| \leq \mathcal{B}_{\text{KN}}(C), \tag{25}$$

$$\max_{\theta\in\Theta}\Big\|\sum_{i=1}^{|D|}\nabla\mathcal{L}_{\text{IF}}(y_i \mid x_i;\theta) - \sum_{j=1}^{|C^*|}\omega^*_j\nabla\mathcal{L}_{\text{IF}}(y_{\gamma^*(j)} \mid x_{\gamma^*(j)};\theta)\Big\| \leq \mathcal{B}_{\text{IF}}(C). \tag{26}$$

This allows us to transform the intractable optimization problem in Eq. (20) into a more manageable form with scalar constraints.

Substituting $\mathcal{B}_{\text{KN}}(C)$ and $\mathcal{B}_{\text{IF}}(C)$ into Eq. (20), we obtain a simplified optimization problem with scalar constraints:

$$\min_{C\subseteq D} |C| \quad \text{s.t.} \quad \mathcal{B}_{\text{KN}}(C) \leq \alpha\varepsilon, \quad \mathcal{B}_{\text{IF}}(C) \leq (1-\alpha)\varepsilon. \tag{27}$$

Unifying Dual Constraints into a Single Constraint. While the dual constraints in Eq. (27) provide explicit control over both capabilities, solving this bi-constraint optimization problem remains computationally challenging. To address this, we propose to transform the dual constraints into a single unified constraint that can be efficiently solved using existing submodular optimization techniques.

Our key insight is to construct a composite distance metric that effectively combines the KN and IF distance components, which is defined as follows:

$$d_{ij}^{\text{SFT}} \stackrel{\text{def}}{=} \frac{1}{\alpha} d_{ij}^{\text{KN}} + \frac{1}{1-\alpha} d_{ij}^{\text{IF}}. \tag{28}$$

By weighting d_{ij}^{KN} with $\frac{1}{\alpha}$ and d_{ij}^{IF} with $\frac{1}{1-\alpha}$, we make each component's contribution inversely proportional to its allocated error budget. We now show that any coreset satisfying a constraint on d_{ij}^{SFT} automatically satisfies both original constraints.

Theorem 5.3. *Let $C \subseteq D$ be a coreset and d_{ij}^{SFT} be defined as in Eq. (28). If $\sum_{i=1}^{|D|} \min_{c_j \in C} d_{ij}^{SFT} \leq \varepsilon$, then C satisfies both constraints in Eq. (27):*

$$\mathcal{B}_{KN}(C) \leq \alpha\varepsilon, \quad \mathcal{B}_{IF}(C) \leq (1-\alpha)\varepsilon. \tag{29}$$

Proof. By construction, d_{ij}^{SFT} is a single budget that is split between d_{ij}^{KN} and d_{ij}^{IF} according to the ratios α and $1-\alpha$. The scaling by $1/\alpha$ and $1/(1-\alpha)$ ensures that each component can never exceed its own share of the composite budget, i.e., $d_{ij}^{\text{KN}} \leq \alpha d_{ij}^{\text{SFT}}$ and $d_{ij}^{\text{IF}} \leq (1-\alpha) d_{ij}^{\text{SFT}}$. Summing over all data points immediately yields $\mathcal{B}_{\text{KN}}(C) \leq \alpha\varepsilon$ and $\mathcal{B}_{\text{IF}}(C) \leq (1-\alpha)\varepsilon$ whenever $\sum_{i=1}^{|D|} \min_{c_j \in C} d_{ij}^{\text{SFT}} \leq \varepsilon$. A detailed proof is provided in our technical report [56]. □

Therefore, the scaling $1/\alpha$ and $1/(1-\alpha)$ is not arbitrary. Instead, it reflects the error budgets for each capability. This ensures that components with tighter budgets are assigned larger distances, prompting the algorithm to prioritize them because satisfying their constraints yields greater benefit. This theoretical result allows us to simplify the original dual constraint optimization problem (Eq. (27)) into a single scalar constraint:

$$\min_{C \subseteq D} |C| \quad \text{s.t.} \quad \sum_{i=1}^{|D|} \min_{c_j \in C} d_{ij}^{\text{SFT}} \leq \varepsilon, \tag{30}$$

Since d_{ij}^{SFT} is a non-negative linear combination of the distances d_{ij}^{KN} and d_{ij}^{IF}, and submodular functions are closed under non-negative linear combinations [15, 27], the function $\sum_{i=1}^{|D|} \min_{c_j \in C} d_{ij}^{\text{SFT}}$ inherits the submodularity property. Previous work [41, 42, 53] has demonstrated that the optimization problems of the form in Eq. (30), where the objective involves minimizing sums of minimum distances, can be equivalently formulated as submodular set cover problems, allowing the use of efficient approximation algorithms.

Formulation with Fixed Coreset Size. While Eq. (30) provides a theoretically elegant formulation with an error bound ε, in practice, directly specifying such a bound is challenging without prior knowledge of the dataset's gradient structure. Following standard practice in coreset selection literature [41, 42] and consistent with the dual formulation in Eq. (2), we adopt a more practical approach: we minimize the gradient approximation error with a coreset size constraint

Algorithm 2 `GreedyCoresetSelection`

```
1:  function GreedyCoresetSelection(D, γ, α, d^KN, d^IF)
2:      k ← ⌈γ|D|⌉, C ← ∅
3:      d_ij^SFT ← (1/α) d_ij^KN + (1/(1−α)) d_ij^IF
4:      Let B(S) = Σ_{i∈D} min_{c∈S∪{s_0}} d_ic^SFT        ▷ with auxiliary s_0
5:      while |C| < k do
6:          u* ← arg max_{u∈D\C} (B(C) − B(C ∪ {u}))
7:          C ← C ∪ {u*}
8:      end while
9:      w_j ← |{i ∈ D : j = arg min_{c∈C} d_ic^SFT}| for all j ∈ C
10:     return C, W = {w_j}_{j∈C}
11: end function
```

K. Specifically, our practical optimization objective becomes:

$$\min_{C \subseteq D} \sum_{i=1}^{|D|} \min_{c_j \in C} d_{ij}^{\text{SFT}} \quad \text{s.t.} \quad |C| \leq K \tag{31}$$

This formulation transforms the problem into a submodular maximization problem through the facility location framework. It is equivalent to solving the constraint problem in Eq. (30) for an appropriately chosen ε value, but eliminates the need to specify ε a priori. The greedy algorithm naturally adapts to minimize the total gradient approximation error given the budget of K data points, making it more practical for real-world applications where the appropriate error tolerance is unknown beforehand.

Solving the Unified Constraint via a Greedy Framework. We solve the unified optimization problem in Eq. (31) by adapting the standard facility location framework [41]. While the greedy optimization strategy is well-established, our approach uniquely incorporates the composite distance metric d_{ij}^{SFT} to balance KN and IF budgets. This formulation allows us to leverage the submodular property of the objective, theoretically guaranteeing a $(1 - \frac{1}{e})$-approximation of the optimal solution (Algorithm 2).

The algorithm begins by computing the composite distance metric $d_{ij}^{\text{SFT}} = \frac{1}{\alpha} d_{ij}^{\text{KN}} + \frac{1}{1-\alpha} d_{ij}^{\text{IF}}$ for all pairs of data points (Line 3). Following Eq. (31), the optimization objective $\mathcal{B}(S)$ is defined (Line 4), incorporating an auxiliary element s_0 to implicitly transform the minimization problem into a maximization one. This objective represents the total gradient approximation error, measuring how well the coreset covers the full dataset $\mathcal{D}$.

The core of Algorithm 2 is the iterative greedy selection process (Lines 5-8). At each iteration, the algorithm directly identifies the candidate data point $u^\star$ that maximizes the marginal gain in the objective function (Line 6). This step effectively selects the point that maximally reduces the gradient approximation error. The selected point is then added to C (Line 7). This greedy strategy ensures a compact yet representative coreset.

After selecting K data points, the algorithm calculates their weights (Line 9) by assigning each data point in $\mathcal{D}$ to its nearest representative in C under the composite distance metric. The weight w_j for each coreset instance is equal to the number of mapped data points. In practice, each greedy step picks the instruction–response pair that most reduces the remaining composite distance for other examples, so the algorithm naturally focuses on regions of the SFT corpus that are still poorly covered.

5.4 Upper-level Optimization Problem

The upper-level optimization in Eq. (20) seeks the optimal partition parameter $\alpha \in (0, 1)$ that minimizes the coreset size. Different values of α decide which aspect of the selected training examples is emphasized, that is, whether their contribution to KN or to IF matters more. We establish the existence of at least one local minimum through upper and lower bounds, which enables efficient optimization via ternary search.

Notation and Setup. For any subset $C \subseteq D$, recall the error bounds defined in Eq. (22). For a given error budget $\tau \geq 0$, we define the single-constraint minimum size functions:

$$K_{KN}(\tau) := \min\{|C| : C \subseteq D, \mathcal{B}_{KN}(C) \leq \tau\}, \tag{32}$$

$$K_{IF}(\tau) := \min\{|C| : C \subseteq D, \mathcal{B}_{IF}(C) \leq \tau\}. \tag{33}$$

These functions are monotonically non-increasing in τ since relaxing the error constraint cannot increase the minimum coreset size.

The dual-constraint minimum size function, corresponding to the lower-level optimal value in Eq. (20), is:

$$K^*(\alpha) := \min\{|C| : C \subseteq D, \mathcal{B}_{KN}(C) \leq \alpha\varepsilon, \mathcal{B}_{IF}(C) \leq (1-\alpha)\varepsilon\}. \tag{34}$$

Lower Bound: Unimodal Structure. We first establish a lower bound for $K^*(\alpha)$ by observing that any feasible coreset must satisfy both constraints independently:

Theorem 5.4 (Lower Bound). *For any $\alpha \in (0, 1)$:*

$$K^*(\alpha) \geq \max\{K_{KN}(\alpha\varepsilon), K_{IF}((1-\alpha)\varepsilon)\}. \tag{35}$$

Proof. Let C^* be an optimal coreset for $K^*(\alpha)$. Since C^* must satisfy $\mathcal{B}_{KN}(C^*) \leq \alpha\varepsilon$, we have $|C^*| \geq K_{KN}(\alpha\varepsilon)$ by the definition of K_{KN}. Similarly, from $\mathcal{B}_{IF}(C^*) \leq (1-\alpha)\varepsilon$, we get $|C^*| \geq K_{IF}((1-\alpha)\varepsilon)$. Thus, $K^*(\alpha) = |C^*| \geq \max\{K_{KN}(\alpha\varepsilon), K_{IF}((1-\alpha)\varepsilon)\}$. □

This lower bound exhibits a unimodal structure: $K_{KN}(\alpha\varepsilon)$ is non-increasing in α (as $\alpha\varepsilon$ increases), while $K_{IF}((1-\alpha)\varepsilon)$ is non-decreasing in α (as $(1-\alpha)\varepsilon$ decreases). The maximum of these two monotonic functions with opposite trends creates a unimodal lower bound with at least one minimum point.

Upper Bound: U-shaped Structure. We construct an upper bound through a union-based approach:

Theorem 5.5 (Upper Bound). *For any $\alpha \in (0, 1)$:*

$$K^*(\alpha) \leq K_{KN}(\alpha\varepsilon) + K_{IF}((1-\alpha)\varepsilon). \tag{36}$$

Proof. Let C_{KN} be an optimal coreset of size $K_{KN}(\alpha\varepsilon)$ satisfying $\mathcal{B}_{KN}(C_{KN}) \leq \alpha\varepsilon$, and C_{IF} be an optimal coreset of size $K_{IF}((1-\alpha)\varepsilon)$ satisfying $\mathcal{B}_{IF}(C_{IF}) \leq (1-\alpha)\varepsilon$. The union $C = C_{KN} \cup C_{IF}$ satisfies both constraints:

$$\mathcal{B}_{KN}(C) \leq \mathcal{B}_{KN}(C_{KN}) \leq \alpha\varepsilon, \tag{37}$$

$$\mathcal{B}_{IF}(C) \leq \mathcal{B}_{IF}(C_{IF}) \leq (1-\alpha)\varepsilon, \tag{38}$$

where the first inequalities follow from the monotonicity of error bounds with respect to set inclusion. Thus, C is feasible for $K^*(\alpha)$, yielding $K^*(\alpha) \leq |C| \leq |C_{KN}| + |C_{IF}| = K_{KN}(\alpha\varepsilon) + K_{IF}((1-\alpha)\varepsilon)$. □

This upper bound exhibits U-shaped behavior: as $\alpha \to 0$, we have $K_{KN}(\alpha\varepsilon) \to |D|$ (requiring nearly all data to satisfy a stringent KN constraint), while $K_{IF}((1-\alpha)\varepsilon)$ remains bounded. Similarly, as $\alpha \to 1$, $K_{IF}((1-\alpha)\varepsilon) \to |D|$ while $K_{KN}(\alpha\varepsilon)$ remains bounded. Thus, the upper bound is large at both extremes.

Efficient Optimization via Ternary Search. The proven bounds and their monotonic properties enable efficient optimization of the upper-level problem via ternary search. The algorithm iteratively narrows the search interval by evaluating two internal points and discarding the subinterval with worse performance. This divide-and-conquer approach converges to a local minimum in $O(\log(1/\delta))$ iterations, where δ is the convergence threshold.

Equivalence Between Subset Size and GA Error Minimization. The dual formulation in Eq. (19) (minimizing coreset size subject to error constraint) and the original objective in Eq. (14) (minimizing total GA error subject to size constraint) are equivalent through duality in constrained optimization [41]. This duality relationship ensures that solving either formulation yields the same optimal trade-off between coreset size and gradient approximation error, allowing us to choose the more convenient formulation for our bi-level optimization framework.

Time Complexity. Our bi-level optimization framework achieves efficient computation through its hierarchical structure. The inner loop (greedy coreset selection) has the same time complexity as the traditional coreset selection algorithm [41], requiring $O(|D|^2 \cdot K)$ operations to compute pairwise distances and perform greedy selection. The outer loop employs ternary search over α, which converges to a locally optimal α^* in $O(\log(1/\delta))$ iterations where δ is the convergence threshold. Therefore, the overall time complexity of our method is $O(\log(1/\delta) \cdot |D|^2 \cdot K)$.

6 EXPERIMENT

In this section, we use 4 massive datasets to fine-tune 3 popular base models to demonstrate the efficiency and effectiveness of BRIEF. We organize the results around five research questions:

RQ1 (Overall performance). How does BRIEF compare with other baselines in accuracy under various selection ratios?

RQ2 (Generalization). Does BRIEF generalize across base models?

RQ3 (Efficiency). How efficient is BRIEF compared with baselines?

RQ4 (Mechanism). Why does BRIEF work—what do the validation of the automatically selected auxiliary variable α and the loss decomposition analysis reveal?

RQ5 (Hyperparameter). What is the impact of the hyperparameter on BRIEF?

6.1 Experiment Setup

Training Settings. We evaluate our method on three widely used foundation models (i.e., Llama-3.1-8B [11], Qwen3-4B [66], and Mistral-Nemo-12B [43]). Mistral-Nemo-12B is a 12-billion parameter model jointly developed by Mistral AI and NVIDIA. Specifically, we train each model for 4 epochs with a maximum token length of 2048 on 8 A800 GPUs. Following the setting of Tulu-3-SFT-Mixture [28], we employ AdamW optimizer and a maximum learning rate of 5e-6. A linear learning rate scheduler with 0.03 warm-up ratio is used and the batch size is set to 128.

Data Selection Settings. Following LESS [64], we first conduct a warm-up phase by training the model for one epoch on a randomly sampled 5% subset from the candidate data pool D, allowing the model to capture the data distribution of the candidate dataset. Then,

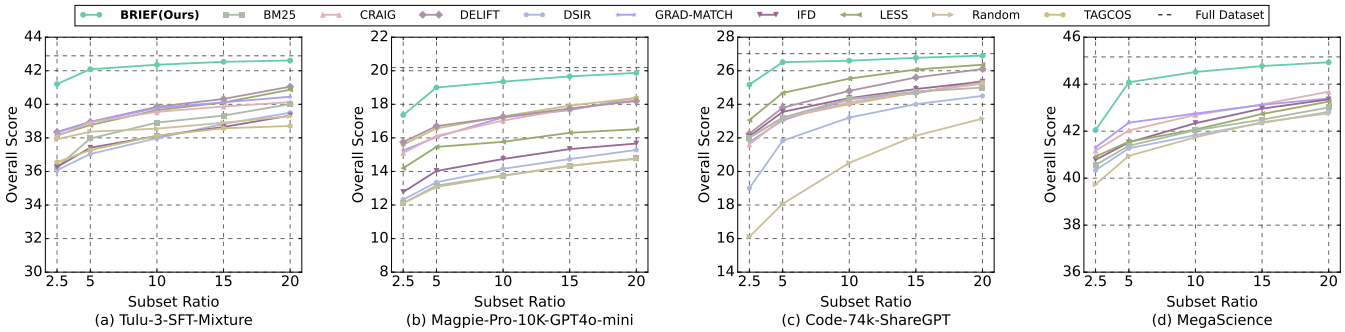


Figure 3: Overall Performance Comparison with Baselines.

this warmed-up model is used as the basis for gradient computation during the data selection period. For methods requiring a reference set, following previous work [63], we sample 5 examples from the training set of MMLU dataset [21] to serve as the reference set.

Datasets. We use four popular datasets as candidate data pools D for our experiments.

(1) Tulu-3-SFT-Mixture [28] comprises a diverse mixture of approximately 900k instruction following examples drawn from various public sources (e.g., CoCoNot [12], FLAN v2 [37], etc.), which emphasizes high-quality responses in reasoning, general knowledge, and multilingual tasks.

(2) Magpie-Pro-10K-GPT4o-mini [46] contains 10k professionally curated instruction–response pairs generated with GPT-4o-mini. This dataset emphasizes high-quality, diverse data covering reasoning, commonsense and practical application scenarios.

(3) Code-74k-ShareGPT [3] is a code-centric dataset containing approximately 74k examples distilled from ShareGPT interactions, which focuses on programming-related queries.

(4) MegaScience [13] is a comprehensive scientific SFT candidate pool covering diverse fields such as math, medicine, biology, physics, and chemistry.

Baselines. We compare BRIEF with several baselines that are classified into two categories according to whether they rely on a reference set.

Reference-based methods: These methods select data points with a distribution similar to a reference set (e.g., MMLU).

(1) DSIR [65] assigns weights to the candidate training data points based on the similarity of n-gram features between the training data points and the reference set, and then selects top k% data points according to these estimated weights.

(2) BM25 [54] selects top k% data points by TF-IDF score, which measures the word similarity between the training data points and the reference set.

(3) LESS [64] selects k% training data points that have the highest influence scores between the reference set and the data points in the candidate pool.

Reference-free methods: These methods aim to enhance the general capabilities of LLMs (e.g., selecting a set of data points without relying on a reference set and evaluating it on multiple different downstream tasks).

(1) Random randomly samples k% of the data points from D, which are then used for fine-tuning.

(2) Total fine-tunes the target model using all training data points in the candidate data pool D, i.e., full-dataset training.

(3) IFD [31] selects k% of the data points that pose greater instruction following difficulty for the target model based on the IFD metric.

(4) TAGCOS [69] selects k% data points from a candidate data pool D by clustering gradients to identify the coreset.

(5) DELIFT [2] is a recent data-efficient instruction fine-tuning method that computes a pairwise utility score between training examples and then applies submodular optimization to select a diverse and informative k% subset. We follow its Instruction Tuning setting, which requires no reference set, and compute the utilities using the training model.

(6) CRAIG [41] reformulates the problem of minimizing gradient approximation (GA) error as a submodular optimization problem and uses a greedy algorithm to select the k% data points as the coreset.

(7) GRAD-MATCH [24] selects a coreset of k% data points whose weighted gradients match those of the full training set, and optimizes this gradient-matching objective with an orthogonal matching pursuit (OMP) algorithm.

(8) BRIEF is our solution.

Evaluation Datasets. To comprehensively evaluate the capabilities of fine-tuned models, we evaluate them on various downstream tasks covering the following categories:

(1) *General Tasks*: MMLU-PRO [59] evaluates broad general knowledge across 14 domains; MMLU [21] evaluates multidisciplinary knowledge with multiple-choice questions across 57 subjects; DROP [10] tests reading comprehension with discrete reasoning; AGIEval [73] benchmarks standardized exams and math-cloze tasks; KorBench [38] is about diverse reasoning challenges across symbolic and logical problem solving.

(2) *Mathematical Tasks*: GSM8K [8] focuses on grade school math problems, while MATH [22] covers competition-level mathematics.

(3) *Coding Tasks*: HumanEval [7] evaluates code synthesis, whereas LCB [49] spans code generation and test output prediction.

Evaluation Metrics. Evaluations are conducted using the OpenCompass [9] framework and the overall score is reported for comparison. For coding tasks (i.e., HumanEval and LCB), we use pass@1, which represents the fraction of problems solved correctly on the first attempt. For other tasks, we report accuracy as the primary metric. We also report ExaFLOPs [1] (EFLOPs) to quantify the total

[1]FLOPs (Floating Point Operations) measure the total number of floating-point computations, serving as a standard metric for computational cost in LLMs. 1 EF corresponds to 10^{18} floating-point operations.

GPU cost across the following three stages: model warm-up, data selection and model training.

6.2 RQ1: Overall Performance

Figure 3 compares BRIEF with other baselines using Llama-3.1-8B on average accuracy (Y-axes), with varying data selection ratios: 2.5%, 5%, 10%, 15% and 20% (X-axes). In summary, BRIEF outperforms other baselines on all candidate data pools at different data selection ratios.

For reference-based methods (i.e., BM25, DSIR and LESS), they primarily select data associated with a reference set (MMLU in our experiment setting). This causes the trained model to excel only on specific downstream tasks. However, such data selection undermines the model's generalization ability to other downstream tasks, leading to performance degradation under diverse evaluation scenarios. For example, under the same 10% data selection ratio, BRIEF outperforms LESS by 3.59% on the Magpie-Pro-10K-GPT4o-mini dataset, highlighting its superior effectiveness in data selection.

For reference-free methods, BRIEF achieves a 3.722% accuracy improvement over random selection on the Tulu-3-SFT-Mixture dataset at a 5% selection ratio. Additionally, unlike IFD which overlooks knowledge capabilities, BRIEF optimizes them jointly. This leads to a 3.08% improvement over IFD on Code-74k-ShareGPT at a 2.5% selection ratio. Similarly, BRIEF outperforms DELIFT by approximately 3.14% on Tulu-3-SFT-Mixture at a 5% selection ratio, as DELIFT only considers pairwise relationships based on loss, rather than the relationship between the subset and the whole dataset. Finally, while coreset methods (i.e., CRAIG, GRAD-MATCH, and TAGCOS) are strong baselines, they rely on raw SFT gradients that cause interference between capabilities. BRIEF addresses this by performing fine-grained minimization of KN and IF error vectors instead of just the overall SFT error. Consequently, BRIEF outperforms CRAIG by approximately 3.165% and GRAD-MATCH by approximately 2.99% on Tulu-3-SFT-Mixture at a 5% selection ratio. The same trend is observed on MegaScience, demonstrating that our method is robust to different domains and maintains good performance.

6.3 RQ2: Generalization Across Models

In this section, we report the performance of BRIEF across 9 downstream tasks under different base models. As shown in Table 2, we fine-tune Llama-3.1-8B, Qwen3-4B and Mistral-Nemo-12B using 5% data points selected from Tulu-3-SFT-Mixture. Specifically, BRIEF exhibits superior accuracy compared to all baselines like GRAD-MATCH, IFD, LESS, CRAIG, DELIFT and TAGCOS. Notably, BRIEF shows a 2.18% accuracy improvement over IFD, a leading baseline at Qwen3-4B. We observe that reference-based methods like LESS or BM25 sometimes perform better on MMLU and MMLU-Pro (e.g., LESS outperforms our method by 3.03% on MMLU with Llama-3.1-8B), which is attributed to their use of reference sets drawn from these benchmarks. Nevertheless, our method achieves the best Overall Score across different model families and sizes, demonstrating that it is robust to model architecture and scale.

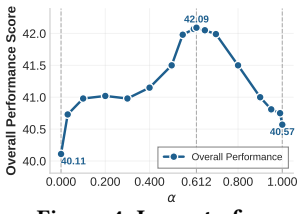


Figure 4: Impact of α.

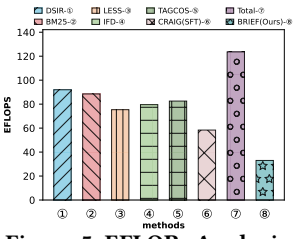


Figure 5: EFLOPs Analysis.

6.4 RQ3: Efficiency

To assess the practical feasibility of our approach, we compare the total computational costs required to achieve the same accuracy across three phases: the warm-up phase, the data selection phase, and the training phase. As shown in Figure 5, BRIEF achieves the lowest EFLOPs among all baseline methods while attaining performance comparable to that of using the full candidate data pool, reducing the total computational cost by 3×. Meanwhile, Table 3 presents an analysis of the time complexity of BRIEF compared with other coreset methods (e.g., CRAIG and TAGCOS), along with a comparison of the actual time consumption. We note that BRIEF and other coreset methods exhibit similar time complexity. However, when selecting coresets for comparable performance, BRIEF significantly improves the model's efficiency by 2–3×.

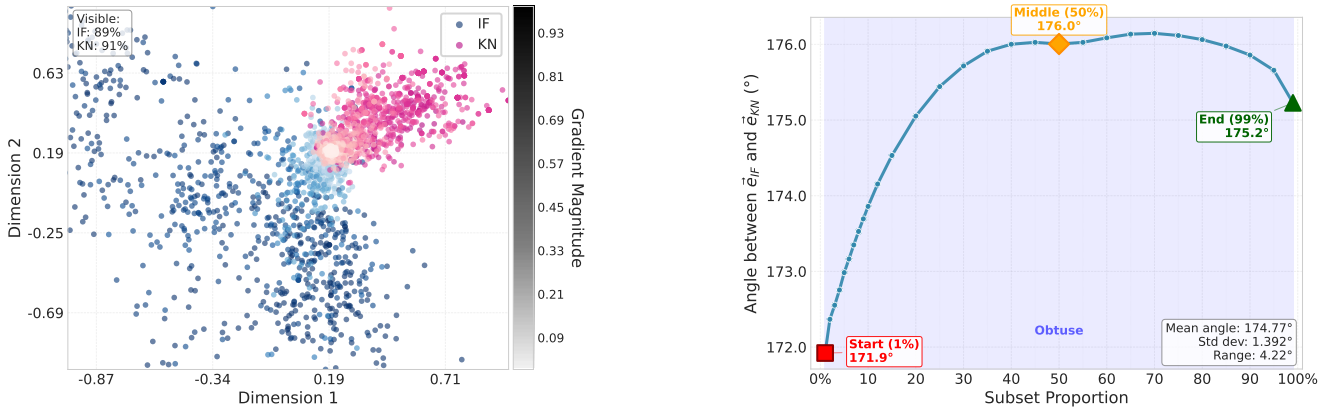
6.5 RQ4: Mechanism Analyses

Effectiveness of α Selection To further validate the effectiveness of our method for choosing α, we enumerate various values of α and evaluate the performance of the coresets obtained from the inner optimization corresponding to each α (shown in Figure 4). Notably, $\alpha = 0.612$ is the value automatically determined by the BRIEF algorithm, which achieves the highest performance across all tested α values, thus validating the effectiveness of our method. To further illustrate the role of α, we plot the data points in the two-dimensional space of KN gradients and IF gradients, as shown in Figure 6(a). Specifically, data points exhibit a more scattered distribution in the IF gradient space, while they appear more concentrated in the KN gradient space, reflecting that the model has obtained extensive knowledge-related abilities during pretraining. This is consistent with BRIEF's choice of α, which favors the IF space. Since the GA error in the IF gradient space is relatively larger, assigning it a larger error bound during coreset selection can better balance the model's capabilities in both the KN and IF spaces.

Effectiveness of Loss Decomposition. To verify our optimization objective, we present the GA errors in both the KN and IF gradient spaces under different coreset optimization objectives, along with their corresponding accuracy, as shown in Figure 7. Specifically, the x-axis represents the GA error in the IF gradient space, and the y-axis represents the GA error in the KN gradient space; points closer to the origin indicate smaller GA errors. We compare BRIEF with the following methods: CRAIG uses gradients from the standard SFT loss, selecting a coreset without distinguishing KN and IF; in BRIEF we extend α from $(0, 1)$ to $[0, 1]$, where $\alpha = 0$ (denoted BRIEF (KN)) only uses the KN gradient and $\alpha = 1$ (denoted BRIEF (IF)) only uses the IF gradient, in which case our method reduces to a Craig-style single-gradient variant. This result highlights a critical limitation in traditional coreset selection approaches: when optimizing with

Table 2: Comprehensive evaluation of Llama-3.1-8B, Qwen3-4B and Mistral-Nemo-12B on 9 downstream tasks. Results are reported for various data selection methods at a 5% sampling ratio. The best scores are highlighted in bold and the second-best scores are underlined. We run each experiment three times and present the averaged results.

Method (5%)	General					Mathematical		Coding		Overall
	DROP	AGIEval	MMLU-PRO	MMLU	KorBench	GSM8K	MATH	HumanEval	LCB	
Llama-3.1-8B										
DSIR	48.00	39.20	33.79	61.67	22.80	54.66	23.36	35.37	14.52	37.041
BM25	46.81	36.13	**35.29**	61.19	21.76	64.44	**27.22**	32.93	15.90	37.964
LESS	47.13	35.56	34.05	**62.06**	19.68	61.90	23.26	49.39	15.84	38.763
IFD	49.02	35.52	32.44	51.90	21.36	61.25	25.88	44.15	15.20	37.413
DELIFT	50.89	38.66	32.31	51.51	23.12	62.42	26.38	49.49	15.77	38.950
TAGCOS	47.41	37.64	30.39	42.12	22.60	63.84	25.18	50.00	16.26	37.272
GRAD-MATCH	52.01	38.53	31.57	53.11	23.03	62.56	25.33	49.25	15.18	38.952
CRAIG	57.66	37.95	31.34	51.96	21.12	61.64	24.72	49.39	14.55	38.925
Random	51.03	38.84	33.71	51.09	23.36	60.27	26.54	44.51	15.97	38.368
BRIEF (Ours)	**60.09**	**39.83**	33.92	59.03	**24.40**	**66.41**	27.05	**51.61**	**16.48**	**42.090**
Qwen3-4B										
DSIR	62.88	47.39	39.66	60.05	46.56	79.87	57.16	72.93	36.58	55.898
BM25	68.92	48.06	56.27	**76.37**	51.52	79.49	57.26	73.05	37.97	60.990
LESS	75.46	48.17	53.72	72.33	49.28	79.27	56.48	69.39	41.18	60.586
IFD	76.13	47.97	55.49	67.98	51.97	80.87	**57.51**	71.33	41.51	61.201
DELIFT	76.13	48.22	56.02	68.17	50.21	79.98	55.81	71.29	41.23	60.784
TAGCOS	76.68	47.79	55.71	67.75	50.60	77.16	55.06	70.05	41.03	60.203
GRAD-MATCH	77.05	47.64	56.14	67.52	50.94	78.16	55.76	71.33	42.01	60.728
CRAIG	76.38	47.52	55.19	68.50	51.00	78.37	56.27	72.51	40.33	60.674
Random	75.21	47.09	51.99	67.46	51.20	75.88	56.62	73.22	41.85	60.058
BRIEF (Ours)	**78.59**	**51.30**	**59.11**	69.33	**55.51**	**83.02**	57.10	**74.01**	**42.46**	**63.381**
Mistral-Nemo-12B										
DSIR	52.69	32.30	33.58	56.44	20.09	58.82	26.47	45.45	15.95	37.977
BM25	52.12	33.65	35.19	62.11	20.78	60.81	24.53	45.84	16.37	39.044
LESS	53.57	32.91	33.77	**63.86**	20.54	59.14	26.95	46.03	16.19	39.218
IFD	54.28	34.49	33.89	57.34	21.62	59.54	26.19	46.62	16.81	38.976
DELIFT	54.67	34.10	33.94	57.45	21.90	60.18	27.01	46.94	17.10	39.254
TAGCOS	54.42	33.40	33.90	56.90	21.29	59.22	26.67	47.20	17.46	38.940
GRAD-MATCH	55.08	34.18	34.63	57.65	21.49	59.91	27.38	46.54	16.72	39.287
CRAIG	55.47	33.90	34.10	57.06	21.98	59.46	26.90	46.29	16.54	39.078
Random	53.99	32.79	33.79	57.00	21.90	59.63	26.28	45.92	16.31	38.623
BRIEF (Ours)	**57.29**	**36.14**	**35.37**	58.46	**23.78**	**62.98**	**28.09**	**48.58**	**18.36**	**41.006**



(a) IF vs KN Gradient Comparison **(b) Angle changes between $\vec{e}_{IF}$ and $\vec{e}_{KN}$**

Figure 6: Comparison of IF and KN: (a) gradient similarity, (b) angle changes with CRAIG.

the full SFT gradient, the knowledge-related (KN) and instruction-following (IF) components can interfere destructively with each other.

Specifically, BRIEF (KN) prioritizes knowledge-related gradients (characterized by a low KN error), excelling in math and coding tasks that demand factual knowledge and systematic reasoning. Similarly, BRIEF (IF), with a low IF error, demonstrates superior performance on general-purpose tasks that typically require strong instruction interpretation and response generation capabilities. In contrast, BRIEF is located in the bottom-left corner, where the GA

Table 3: The time complexity and the actual time of each coreset method when achieving comparable accuracy. Here, $|D|$ denotes the candidate data pool size, K the subset size, δ the convergence threshold, and c the number of clusters.

Method	Data Selection		Total	Accuracy
	Complexity	Actual (h)	Actual (h)	
TAGCOS	$O(\|D\| \cdot K + \|D\| \cdot c)$	0.6	28.9	42.083
CRAIG	$O(\|D\|^2 \cdot K)$	0.7	23.8	42.071
BRIEF (Ours)	$O(\log(1/\delta) \cdot \|D\|^2 \cdot K)$	2.4	13.6	42.090

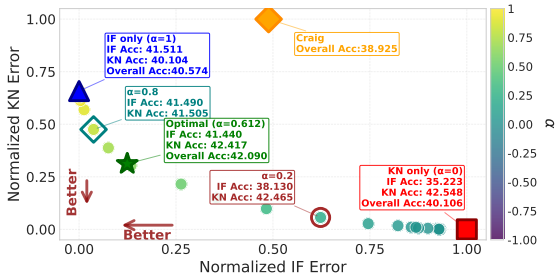


Figure 7: Error of KN and IF under varying α

errors in both the KN and IF gradient spaces are the smallest, achieving the best overall performance: it outperforms BRIEF (KN) and BRIEF (IF) because those variants approximate only one capability at a time, yielding limited gains, whereas BRIEF balances both dimensions simultaneously for superior accuracy across tasks. We further analyze the impact of α by categorizing benchmarks into instruction-following tasks (e.g., DROP, AGIEval, and KorBench) and knowledge-related tasks (e.g., MMLU, MMLU-PRO, GSM8K, MATH, HumanEval, and LCB). Figure 7 reveals a clear trade-off: increasing α improves instruction following but compromises knowledge accuracy, while decreasing α does the reverse. Crucially, BRIEF automatically selects an α in the balanced region, ensuring strong performance on both task types.

Evaluation of Error Vector Angles. To further verify the conflict between the KN and IF error vectors, we plot the angle between them under different coreset selection ratios, as shown in Figure 6(b). Specifically, we vary the selection ratio in increments of 1%, and for each ratio, we compute the angle between the error vectors of the selected coreset data points and those of the entire candidate data pool in the KN and IF gradient spaces. Across all selection ratios, the angle between the KN and IF error vectors remains close to 180 degrees, indicating that these two error vectors are in conflict during the coreset selection process. This further demonstrates that the traditional coreset selection methods that merely bound the sum of the two error vectors in the instruction tuning setting can lead to substantial errors, as we theoretically analyzed in Section 4.2. In contrast, BRIEF, by bounding the sum of these two error vectors, better preserves both the KN and IF capabilities of the candidate data pool.

6.6 RQ5: Impact of Hyperparameters

We now study two hyperparameters in BRIEF: the warm-up data ratio and the training budget (number of fine-tuning epochs).

Effect of warm-up ratio. We first study how the warm-up ratio used to compute gradients affects BRIEF. Using Llama-3.1-8B on different candidate data pools, we compare BRIEF when gradients

Table 4: Effect of training budget on overall score and total EFLOPs.

Method	1 epoch		2 epochs		4 epochs		6 epochs	
	Score	EFLOPs	Score	EFLOPs	Score	EFLOPs	Score	EFLOPs
Full Data (100%)	40.90	28.95	41.96	60.88	42.88	123.60	42.02	190.74
Random (5%)	36.50	1.55	37.22	3.10	38.37	6.08	37.72	10.01
CRAIG (5%)	36.91	15.54	37.89	17.22	38.93	20.27	38.21	23.28
BRIEF (5%)	39.99	27.67	41.10	29.03	42.09	32.18	41.28	35.33

Table 5: Ablation Study on Warm-up Ratio.

Base Model	Llama-3.1-8B				Qwen3-4B			
Warm-up Ratio	0%	5% (default)	25%	100%	0%	5 % (default)	25%	100%
Tulu-3-SFT-Mixture	40.15%	42.09%	42.31%	42.55%	60.11%	63.38%	63.51%	63.73%
Magpie-Pro	17.55%	19.11%	19.23%	19.37%	28.06%	31.56%	31.61%	31.72%
Code-74k-ShareGPT	23.38%	26.51%	26.87%	26.96%	41.15%	43.73%	43.90%	44.01%
Average Score	27.03%	29.24%	29.47%	29.63%	43.11%	46.22%	46.34%	46.49%

are computed on the base model (no warm-up) and on models trained with warm-up ratios of 5%, 25%, and 100% of the candidate data. Table 5 shows that using the base model without warm-up greatly hurts BRIEF's performance, likely due to the shift between the pre-training and fine-tuning input distributions [64]. As the warm-up ratio increases, the model performance improves, but the total training cost (FLOPs) also grows. Notably, a 5% warm-up already achieves performance comparable to using 25% data, at a much lower cost. Therefore, we use a 5% warm-up ratio as the default, which is consistent with prior work [64, 69].

Effect of training budget. We also add an explicit ablation on training budget to study how longer training interacts with data selection. The result is reported in Table 4, where we report overall score and total EFLOPs as we increase the number of fine-tuning epochs. For this experiment, we fix Llama-3.1-8B and Tulu-3-SFT-Mixture as the candidate data pool, keep a 5% subset size for all methods, and vary the number of fine-tuning epochs from 1 to 2, 4, and 6. For each epoch setting, we fine-tune the model with BRIEF, CRAIG, random selection, and a full-dataset baseline.

We make three observations. First, training longer does not always help: all methods peak at the 4th epoch and then drop at the 6th epoch due to overfitting, a phenomenon also observed in prior work on LLM [57]. Second, full-data training obtains the best accuracy at each epoch, while BRIEF closely matches this performance and consistently outperforms all other subset baselines. Third, although BRIEF has an initial cost for data selection, it becomes efficient quickly: by the 4th epoch, this cost is offset, and BRIEF saves about 70% EFLOPs compared to full-data training.

7 CONCLUSION

This work takes a simple but informative algebraic view of the SFT loss, showing that it naturally decomposes into knowledge-related and instruction-following components and clarifying why conventional coreset methods can fail to preserve both capabilities in instruction tuning. Building on this perspective, we propose an objective that explicitly minimizes the aggregate GA error over these two components and use a bi-level optimization framework. Experiments on state-of-the-art LLMs and real-world datasets demonstrate that our approach, BRIEF, reduces fine-tuning costs by up to 3× while improving accuracy by 5%.

REFERENCES

[1] Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. 2023. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540* (2023).

[2] Ishika Agarwal, Krishnateja Killamsetty, Lucian Popa, and Marina Danilevsky. [n.d.]. DELIFT: Data Efficient Language model Instruction Fine-Tuning. In *The Thirteenth International Conference on Learning Representations*.

[3] Ajibawa. 2025. Code-74k-ShareGPT. https://huggingface.co/datasets/ajibawa-2023/Code-74k-ShareGPT. https://huggingface.co/datasets/ajibawa-2023/Code-74k-ShareGPT HuggingFace dataset.

[4] Chengliang Chai, Jiabin Liu, Nan Tang, Ju Fan, Dongjing Miao, Jiayi Wang, Yuyu Luo, and Guoliang Li. 2023. Goodcore: Data-effective and data-efficient machine learning through coreset selection over incomplete data. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–27.

[5] Chengliang Chai, Jiayi Wang, Nan Tang, Ye Yuan, Jiabin Liu, Yuhao Deng, and Guoren Wang. 2023. Efficient coreset selection with cluster-based methods. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 167–178.

[6] Jie Chen, Zhipeng Chen, Jiapeng Wang, Kun Zhou, Yutao Zhu, Jinhao Jiang, Yingqian Min, Wayne Xin Zhao, Zhicheng Dou, Jiaxin Mao, et al. 2024. Towards Effective and Efficient Continual Pre-training of Large Language Models. *arXiv preprint arXiv:2407.18743* (2024).

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

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

[9] OpenCompass Contributors. 2023. OpenCompass: A Universal Evaluation Platform for Foundation Models. https://github.com/open-compass/opencompass.

[10] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2368–2378.

[11] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[12] Vidhisha Balachandran Pradeep Dasigi Valentina Pyatkin Abhilasha Ravichander Sarah Wiegreffe Nouha Dziri Khyathi Chandu Jack Hessel Yulia Tsvetkov Noah A. Smith Yejin Choi Hannaneh Hajishirzi Faeze Brahman, Sachin Kumar. 2024. The Art of Saying No: Contextual Noncompliance in Language Models. (2024).

[13] Run-Ze Fan, Zengzhi Wang, and Pengfei Liu. 2025. MegaScience: Pushing the Frontiers of Post-Training Datasets for Science Reasoning. *arXiv preprint arXiv:2507.16812* (2025). https://arxiv.org/abs/2507.16812

[14] Dan Feldman. 2020. Core-sets: Updated survey. *Sampling techniques for supervised or unsupervised tasks* (2020), 23–44.

[15] Satoru Fujishige. 2005. *Submodular functions and optimization*. Vol. 58. Elsevier.

[16] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027* (2020).

[17] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644* (2023).

[18] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[19] Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints* (2023).

[20] Juris Hartmanis. 1982. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review* 24, 1 (1982), 90.

[21] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).

[22] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. [n.d.]. Measuring Mathematical Problem Solving With the MATH Dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[23] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

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

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

[26] Krishnateja Killamsetty, Xujiang Zhao, Feng Chen, and Rishabh Iyer. 2021. Retrieve: Coreset selection for efficient and robust semi-supervised learning. *Advances in neural information processing systems* 34 (2021), 14488–14501.

[27] Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. *Tractability* 3, 71-104 (2014), 3.

[28] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124* (2024).

[29] Changho Lee, Janghoon Han, Seonghyeon Ye, Stanley Jungkyu Choi, Honglak Lee, and Kyunghoon Bae. 2024. Instruction Matters: A Simple yet Effective Task Selection for Optimized Instruction Tuning of Specific Tasks. *arXiv preprint arXiv:2404.16418* (2024).

[30] Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Jiuxiang Gu, and Tianyi Zhou. 2024. Selective Reflection-Tuning: Student-Selected Data Recycling for LLM Instruction-Tuning. In *Findings of the Association for Computational Linguistics ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand and virtual meeting, 16189–16211. https://aclanthology.org/2024.findings-acl.958

[31] 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. *arXiv preprint arXiv:2402.00530* (2024).

[32] Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024. 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)*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 7595–7628. https://aclanthology.org/2024.naacl-long.421

[33] Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Ling-Hao Chen, Junhao Liu, Tongliang Liu, et al. 2023. One-shot learning as instruction data prospector for large language models. *arXiv preprint arXiv:2312.10302* (2023).

[34] Yiming Li, Yanyan Shen, and Lei Chen. 2022. Camel: Managing data for efficient stream learning. In *Proceedings of the 2022 International Conference on Management of Data*. 1271–1285.

[35] Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, et al. 2024. Rho-1: Not all tokens are what you need. *arXiv preprint arXiv:2404.07965* (2024).

[36] Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2023. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685* (2023).

[37] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The Flan Collection: Designing Data and Methods for Effective Instruction Tuning. *arXiv preprint arXiv:2301.13688* (2023).

[38] Kaijing Ma, Xeron Du, Yunran Wang, Haoran Zhang, ZhoufutuWen, Xingwei Qu, Jian Yang, Jiaheng Liu, minghao liu, Xiang Yue, Wenhao Huang, and Ge Zhang. 2025. KOR-Bench: Benchmarking Language Models on Knowledge-Orthogonal Reasoning Tasks. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=SVRRQ8goQo

[39] Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564* (2023).

[40] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196* (2024).

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

[42] Baharan Mirzasoleiman, Kaidi Cao, and Jure Leskovec. 2020. Coresets for robust training of deep neural networks against noisy labels. *Advances in Neural Information Processing Systems* 33 (2020), 11465–11477.

[43] Mistral AI and NVIDIA. 2024. Mistral-Nemo-Base-2407. https://huggingface.co/mistralai/Mistral-Nemo-Base-2407.

[44] Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. 2024. Scaling data-constrained language models. *Advances in Neural Information Processing Systems* 36 (2024).

[45] Alexander Munteanu and Chris Schwiegelshohn. 2018. Coresets-methods and history: A theoreticians design pattern for approximation and streaming algorithms. *KI-Künstliche Intelligenz* 32 (2018), 37–53.

[46] Mxode. 2025. Magpie-Pro-10K-GPT4o-mini. https://huggingface.co/datasets/Mxode/Magpie-Pro-10K-GPT4o-mini. https://huggingface.co/datasets/Mxode/Magpie-Pro-10K-GPT4o-mini HuggingFace dataset.

[47] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435* (2023).

[48] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* 14 (1978), 265–294.

[49] OpenCompass. [n.d.]. *CodeBench: LCBench - LeetCode Contest Benchmark Dataset.* https://github.com/open-compass/CodeBench/

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

[51] Christos H Papadimitriou. 1994. Computational complexity. (1994).

[52] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116* (2023).

[53] Omead Pooladzandi, David Davini, and Baharan Mirzasoleiman. 2022. Adaptive second order coresets for data-efficient machine learning. In *International Conference on Machine Learning*. PMLR, 17848–17869.

[54] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

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

[56] Chaoyuan Shen, Chi Zhang, Chengliang Chai, Jiacheng Wang, Jia Yuan, Yuping Wang, Ye Yuan, Guoren Wang, and Lei Cao. 2025. *BRIEF: Bi-level Coreset Selection for Efficient Instruction Tuning in LLMs.* Technical Report. https://github.com/HR10108/BRIEF/blob/main/BRIEF_Technical_Report.pdf Technical Report.

[57] Zhengyan Shi, Adam X Yang, Bin Wu, Laurence Aitchison, Emine Yilmaz, and Aldo Lipani. 2024. Instruction tuning with loss over instructions. *Advances in Neural Information Processing Systems* 37 (2024), 69176–69205.

[58] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. 2024. Dolma: An open corpus of three trillion tokens for language model pretraining research. *arXiv preprint arXiv:2402.00159* (2024).

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

[60] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2019. CCNet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359* (2019).

[61] Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. 2024. QuRating: Selecting High-Quality Data for Training Language Models. In *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=GLGYYqPwjy

[62] Xuansheng Wu, Wenlin Yao, Jianshu Chen, Xiaoman Pan, Xiaoyang Wang, Ninghao Liu, and Dong Yu. 2024. From Language Modeling to Instruction Following: Understanding the Behavior Shift in LLMs after 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)*. 2341–2369.

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

[64] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. LESS: Selecting Influential Data for Targeted Instruction Tuning. In *Forty-first International Conference on Machine Learning*.

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

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

[67] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115* (2024).

[68] Zichun Yu, Spandan Das, and Chenyan Xiong. 2024. MATES: Model-Aware Data Selection for Efficient Pretraining with Data Influence Models. *arXiv preprint arXiv:2406.06046* (2024).

[69] Jipeng Zhang, Yaxuan Qin, Renjie Pi, Weizhong Zhang, Rui Pan, and Tong Zhang. 2024. TAGCOS: Task-agnostic Gradient Clustered Coreset Selection for Instruction Tuning Data. *arXiv preprint arXiv:2407.15235* (2024).

[70] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792* (2023).

[71] Yifan Zhang, Yifan Luo, Yang Yuan, and Andrew C Yao. 2024. Autonomous data selection with language models for mathematical texts. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*.

[72] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[73] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364* (2023).